MPC555 / MPC556

USER'S MANUAL

Revised 15 October 2000

© Copyright 2000 MOTOROLA; All Rights Reserved

MPC555 / MPC556

USER'S MANUAL

Revised 15 October 2000

© Copyright 2000 MOTOROLA; All Rights Reserved

Paragraph Number

TABLE OF CONTENTS

Page Number

PREFACE

Section 1
OVERVIEW

1.1 Block Diagram 1-1
1.2 MPC555 / MPC556 Features 1-2
1.2.1 RISC MCU Central Processing Unit (RCPU) 1-2
1.2.2 Four-Bank Memory Controller 1-3
1.2.3 U-Bus System Interface Unit (USIU) 1-3
1.2.4 Flexible Memory Protection Unit 1-3
1.2.5 448 Kbytes of CDR MoneT Flash EEPROM Memory (CMF) 1-3
1.2.6 26 Kbytes of Static RAM 1-3
1.2.7 General-Purpose I/O Support 1-4
1.2.8 Two Time Processor Units (TPU3) 1-4
1.2.9 18-Channel Modular I/O System (MIOS1) 1-4
1.2.10 Two Queued Analog-to-Digital Converter Modules (QADC) 1-4
1.2.11 Two CAN 2.0B Controller Modules (TouCANs) 1-5
1.2.12 Queued Serial Multi-Channel Module (QSMCM) 1-5
1.3 MPC555 / MPC556 Address Map 1-5

Section 2
SIGNAL DESCRIPTIONS

2.1 Packaging and Pinout Descriptions 2-1
2.2 Pin Functionality 2-6
2.3 Signal Descriptions 2-12
2.3.1 USIU Pads 2-12
2.3.1.1 ADDR[8:31]/SGPIOA[8:31] 2-12
2.3.1.2 DATA[0:31]/SGPIOD[0:31] 2-13
2.3.1.3 $\overline{\text{IRQ}}$[0]/SGPIOC[0] 2-13
2.3.1.4 $\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1] 2-13
2.3.1.5 $\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$ 2-13
2.3.1.6 $\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC[3] 2-13
2.3.1.7 $\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4] 2-14
2.3.1.8 $\overline{\text{IRQ}}$[5]/SGPIOC[5]/MODCK[1] 2-14
2.3.1.9 $\overline{\text{IRQ}}$[6:7]/MODCK[2:3] 2-14
2.3.1.10 TSIZ[0:1] 2-14
2.3.1.11 RD/$\overline{\text{WR}}$ 2-15
2.3.1.12 $\overline{\text{BURST}}$ 2-15
2.3.1.13 $\overline{\text{BDIP}}$ 2-15
2.3.1.14 $\overline{\text{TS}}$ 2-15
2.3.1.15 $\overline{\text{TA}}$ 2-15

Paragraph Number		Page Number

2.3.1.16 TEA . . . 2-15
2.3.1.17 RSTCONF/TEXP . . . 2-16
2.3.1.18 OE . . . 2-16
2.3.1.19 BI/STS . . . 2-16
2.3.1.20 CS[0:3] . . . 2-16
2.3.1.21 WE[0:3]/BE[0:3]/AT[0:3] . . . 2-16
2.3.1.22 PORESET . . . 2-17
2.3.1.23 HRESET . . . 2-17
2.3.1.24 SRESET . . . 2-17
2.3.1.25 SGPIOC[6]/FRZ/PTR . . . 2-17
2.3.1.26 SGPIOC[7]/IRQOUT/LWP[0] . . . 2-18
2.3.1.27 BG/VF[0]/LWP[1] . . . 2-18
2.3.1.28 BR/VF[1]/IWP[2] . . . 2-18
2.3.1.29 BB/VF[2]/IWP[3] . . . 2-18
2.3.1.30 IWP[0:1]/VFLS[0:1] . . . 2-19
2.3.1.31 TMS . . . 2-19
2.3.1.32 TDI/DSDI . . . 2-19
2.3.1.33 TCK/DSCK . . . 2-19
2.3.1.34 TDO/DSDO . . . 2-19
2.3.1.35 TRST . . . 2-20
2.3.1.36 XTAL . . . 2-20
2.3.1.37 EXTAL . . . 2-20
2.3.1.38 XFC . . . 2-20
2.3.1.39 CLKOUT . . . 2-20
2.3.1.40 EXTCLK . . . 2-20
2.3.1.41 VDDSYN . . . 2-20
2.3.1.42 VSSSYN . . . 2-20
2.3.1.43 ENGCLK/BUCLK . . . 2-21
2.3.2 QSMCM PADS . . . 2-21
2.3.2.1 PCS0/SS/QGPIO[0] . . . 2-21
2.3.2.2 PCS(1:3)/QGPIO[1:3] . . . 2-21
2.3.2.3 MISO/QGPIO[4] . . . 2-21
2.3.2.4 MOSI/QGPIO[5] . . . 2-21
2.3.2.5 SCK/QGPIO[6] . . . 2-22
2.3.2.6 TXD[1:2]/QGPO[1:2] . . . 2-22
2.3.2.7 RXD[1:2]/QGPI[1:2] . . . 2-22
2.3.2.8 ECK . . . 2-22
2.3.3 MIOS PADS . . . 2-22
2.3.3.1 MDA[11], [13] . . . 2-22
2.3.3.2 MDA[12], [14] . . . 2-23
2.3.3.3 MDA[15], [27:31] . . . 2-23
2.3.3.4 MPWM[0:3], [16:19] . . . 2-23
2.3.3.5 VF[0:2]/MPIO32B[0:2] . . . 2-23

Paragraph Number		Page Number

2.3.3.6 VFLS[0:1]/MPIO32B[3:4] 2-23
2.3.3.7 MPIO32B[5:15] 2-24
2.3.4 TPU_A/TPU_B PADS 2-24
2.3.4.1 TPUCH[0:15]_[A:B] 2-24
2.3.4.2 T2CLK 2-24
2.3.5 QADC_A/QADC_B PADS 2-24
2.3.5.1 ETRIG[1:2] 2-24
2.3.5.2 AN[0]/ANW/PQB[0]_[A:B] 2-24
2.3.5.3 AN[1]/ANX/PQB[1]_[A:B] 2-24
2.3.5.4 AN[2]/ANY/PQB[2]_[A:B] 2-25
2.3.5.5 AN[3]/ANZ/PQB[3]_[A:B] 2-25
2.3.5.6 AN[48:51]/PQB[4:7]_[A:B] 2-25
2.3.5.7 AN[52:54]/MA[0:2]/PQA[0:2]_[A:B] 2-25
2.3.5.8 AN[55:59]/PQA[3:7]_[A:B] 2-26
2.3.5.9 VRH 2-26
2.3.5.10 VRL 2-26
2.3.5.11 VDDA 2-26
2.3.5.12 VSSA 2-26
2.3.6 TOUCAN_A/TOUCAN_B PADS 2-26
2.3.6.1 CNTX0_[A:B] 2-26
2.3.6.2 CNRX0_[A:B] 2-26
2.3.7 CMF PADS 2-27
2.3.7.1 EPEE 2-27
2.3.7.2 VPP 2-27
2.3.7.3 VDDF 2-27
2.3.7.4 VSSF 2-27
2.3.8 GLOBAL POWER SUPPLIES 2-27
2.3.8.1 VDDL 2-27
2.3.8.2 VDDH 2-27
2.3.8.3 VDDI 2-27
2.3.8.4 VSSI 2-27
2.3.8.5 KAPWR 2-28
2.3.8.6 VDDSRAM 2-28
2.3.8.7 VSS 2-28
2.4 Reset State 2-28
2.4.1 Pin Functionality Out of Reset 2-28
2.4.2 Pad Module Configuration Register (PDMCR) 2-28
2.4.3 Pin State During Reset 2-29
2.4.4 Power-On Reset and Hard Reset 2-30
2.4.5 Pull-Up and Pull-Down Enable and Disable for 5-V Only Pins 2-30
2.4.6 Pull-Up and Pull-Down Enable and Disable for 3-V / 5-V Multiplexed Pins 2-30
2.4.6.1 PRDS Signal 2-30
2.4.6.2 Encoded 3-V / 5-V Select 2-31

Paragraph Number		Page Number

2.4.6.3 Examples 2-31
2.4.7 Special Pull Resistor Disable Control (SPRDS) 2-31
2.4.8 Pin Reset States 2-32
2.5 Pad Types 2-37
2.5.1 Pad Interface Signals 2-37
2.5.2 Three-Volt Output Pad 2-38
2.5.2.1 Type A Interface 2-38
2.5.2.2 Type B Interface (Clock Pad) 2-39
2.5.3 Three-Volt Input Pad 2-39
2.5.3.1 Type C Interface 2-40
2.5.3.2 Type CH Interface 2-40
2.5.3.3 Type CNH Interface 2-41
2.5.3.4 Type D Interface 2-41
2.5.4 Three-Volt Input/Output Pad 2-41
2.5.4.1 Type E Interface 2-42
2.5.4.2 Type EOH Interface 2-42
2.5.4.3 Type F Interface 2-43
2.5.4.4 Type G Interface 2-44
2.5.5 Five-Volt Input/Output Pad 2-45
2.5.5.1 Type H Interface 2-45
2.5.5.2 Type I Interface 2-46
2.5.5.3 Type IH Interface 2-48
2.5.5.4 Type J Interface 2-49
2.5.5.5 Type JD Interface 2-49
2.5.6 Type K Interface (EPEE Pad) 2-50
2.5.7 Analog Pads 2-51
2.5.7.1 Type L Interface (QADC Port A) 2-51
2.5.7.2 Type M Interface (QADC Port B) 2-52
2.5.7.3 Type N Interface (ETRIG) 2-53
2.5.8 Pads with Fast Mode 2-53
2.5.8.1 Type O Interface (QSMCM Pads) 2-53
2.5.8.2 Type P Interface (TPU and MIOS Pads) 2-54
2.5.9 5V Input, 5V Output Pads 2-55
2.5.9.1 5V Output (Type Q) 2-55
2.5.9.2 Type R Interface 2-56
2.5.9.3 5V Output for Clock Pad 2-57
2.6 Pad Groups 2-57
2.7 Pin Names and Abbreviations 2-58

Section 3
CENTRAL PROCESSING UNIT

3.1 RCPU Features 3-1
3.2 RCPU Block Diagram 3-2

Paragraph Number		Page Number

3.3 Instruction Sequencer 3-3
3.4 Independent Execution Units 3-4
3.4.1 Branch Processing Unit (BPU) 3-5
3.4.2 Integer Unit (IU) 3-5
3.4.3 Load/Store Unit (LSU) 3-6
3.4.4 Floating-Point Unit (FPU) 3-6
3.5 Levels of the PowerPC Architecture 3-7
3.6 RCPU Programming Model 3-7
3.7 PowerPC UISA Register Set 3-11
3.7.1 General-Purpose Registers (GPRs) 3-12
3.7.2 Floating-Point Registers (FPRs) 3-12
3.7.3 Floating-Point Status and Control Register (FPSCR) 3-12
3.7.4 Condition Register (CR) 3-15
3.7.4.1 Condition Register CR0 Field Definition 3-16
3.7.4.2 Condition Register CR1 Field Definition 3-16
3.7.4.3 Condition Register CR*n* Field — Compare Instruction 3-17
3.7.5 Integer Exception Register (XER) 3-17
3.7.6 Link Register (LR) 3-18
3.7.7 Count Register (CTR) 3-19
3.8 PowerPC VEA Register Set — Time Base 3-19
3.9 PowerPC OEA Register Set 3-20
3.9.1 Machine State Register (MSR) 3-20
3.9.2 DAE/Source Instruction Service Register (DSISR) 3-22
3.9.3 Data Address Register (DAR) 3-22
3.9.4 Time Base Facility (TB) — OEA 3-23
3.9.5 Decrementer Register (DEC) 3-23
3.9.6 Machine Status Save/Restore Register 0 (SRR0) 3-24
3.9.7 Machine Status Save/Restore Register 1 (SRR1) 3-24
3.9.8 General SPRs (SPRG0–SPRG3) 3-25
3.9.9 Processor Version Register (PVR) 3-25
3.9.10 Implementation-Specific SPRs 3-26
3.9.10.1 EIE, EID, and NRI Special-Purpose Registers 3-26
3.9.10.2 Floating-Point Exception Cause Register (FPECR) 3-26
3.9.10.3 Additional Implementation-Specific Registers 3-27
3.10 Instruction Set 3-28
3.10.1 Instruction Set Summary 3-29
3.10.2 Recommended Simplified Mnemonics 3-33
3.10.3 Calculating Effective Addresses 3-33
3.11 Exception Model 3-34
3.11.1 Exception Classes 3-34
3.11.2 Ordered Exceptions 3-34
3.11.3 Unordered Exceptions 3-34
3.11.4 Precise Exceptions 3-35

Paragraph Number		Page Number

3.11.5 Exception Vector Table 3-35
3.12 Instruction Timing 3-36
3.13 PowerPC User Instruction Set Architecture (UISA) 3-38
3.13.1 Computation Modes 3-38
3.13.2 Reserved Fields 3-38
3.13.3 Classes of Instructions 3-38
3.13.4 Exceptions 3-39
3.13.5 The Branch Processor 3-39
3.13.6 Instruction Fetching 3-39
3.13.7 Branch Instructions 3-39
3.13.7.1 Invalid Branch Instruction Forms 3-39
3.13.7.2 Branch Prediction 3-39
3.13.8 The Fixed-Point Processor 3-39
3.13.8.1 Fixed-Point Instructions 3-39
3.13.9 Floating-Point Processor 3-40
3.13.9.1 General 3-40
3.13.9.2 Optional instructions 3-40
3.13.10 Load/Store Processor 3-40
3.13.10.1 Fixed-Point Load With Update and Store With Update Instructions 3-41
3.13.10.2 Fixed-Point Load and Store Multiple Instructions 3-41
3.13.10.3 Fixed-Point Load String Instructions 3-41
3.13.10.4 Storage Synchronization Instructions 3-41
3.13.10.5 Floating-Point Load and Store With Update Instructions 3-41
3.13.10.6 Floating-Point Load Single Instructions 3-41
3.13.10.7 Floating-Point Store Single Instructions 3-41
3.13.10.8 Optional Instructions 3-42
3.13.10.9 Little-Endian Byte Ordering 3-42
3.14 PowerPC Virtual Environment Architecture (VEA) 3-42
3.14.1 Atomic Update Primitives 3-42
3.14.2 Effect of Operand Placement on Performance 3-42
3.14.3 Storage Control Instructions 3-42
3.14.4 Instruction Synchronize (isync) Instruction 3-42
3.14.4.1 Enforce In-Order Execution of I/O (eieio) Instruction 3-43
3.14.5 Timebase 3-43
3.15 POWERPC Operating Environment Architecture (OEA) 3-43
3.15.1 Branch Processor Registers 3-43
3.15.1.1 Machine State Register (MSR) 3-43
3.15.1.2 Branch Processors Instructions 3-43
3.15.2 Fixed-Point Processor 3-43
3.15.2.1 Special Purpose Registers 3-43
3.15.3 Storage Control Instructions 3-44
3.15.4 Interrupts 3-44
3.15.4.1 System Reset Interrupt 3-44

3.15.4.2 Machine Check Interrupt 3-45
3.15.4.3 Data Storage Interrupt 3-45
3.15.4.4 Instruction Storage Interrupt 3-46
3.15.4.5 Alignment Interrupt 3-46
3.15.4.6 Floating-Point Enabled Exception Type Program Interrupt 3-46
3.15.4.7 Illegal Instruction Type Program Interrupt 3-46
3.15.4.8 Privileged Instruction Type Program interrupt 3-46
3.15.4.9 Floating-Point Unavailable Interrupt 3-47
3.15.4.10 Trace Interrupt 3-47
3.15.4.11 Floating-Point Assist Interrupt 3-47
3.15.4.12 Implementation-Dependent Software Emulation Interrupt 3-48
3.15.4.13 Implementation-Specific Instruction Storage Protection Error Interrupt 3-49
3.15.4.14 Implementation-Specific Data Storage Protection Error Interrupt 3-50
3.15.4.15 Implementation-Specific Debug Interrupts 3-51
3.15.4.16 Partially Executed Instructions 3-52
3.15.5 Timer Facilities 3-53
3.15.6 Optional Facilities and Instructions 3-53

Section 4
BURST BUFFER

4.1 Burst Buffer Block Diagram 4-1
4.2 Burst Buffer Features 4-2
4.3 Instruction VocabularyBased Compression Model Main Principles 4-3
4.3.1 Compression Model Features 4-3
4.3.2 Model Limitations 4-4
4.3.3 Vocabulary Based Instruction Compression Algorithm 4-4
4.3.4 Memory Organization 4-6
4.3.5 Compressed Code Address Format 4-8
4.3.6 Compressed Address Format – Direct Branches 4-9
4.3.7 Compressed Address Format – Indirect Branches 4-11
4.3.8 Compression Process 4-11
4.3.9 Decompression 4-13
4.3.10 Compression Environment Initialization 4-14
4.4 Modes Of Operation 4-14
4.4.1 Normal Operation 4-15
4.4.2 Slave Operation 4-15
4.4.3 Reset Operation 4-15
4.4.4 Debug Mode Operation 4-15
4.4.5 Standby Mode Operation 4-15
4.4.6 Burst Operation 4-15
4.4.7 Error Detection 4-16
4.5 Exception Table Relocation 4-16
4.5.1 Exception Table Relocation Operation 4-17

4.6 Burst Buffer Programming Model 4-20
4.6.1 Region Base Address Registers 4-21
4.6.2 Region Attribute Registers MI_RA[0:3] Description 4-21
4.6.3 Global Region Attribute Register Description (MI_GRA) 4-23
4.6.4 BBC Module Configuration Register (BBCMCR) 4-24

Section 5
UNIFIED SYSTEM INTERFACE UNIT

5.1 Module Overview 5-1
5.2 SIU Architecture 5-2
5.3 USIU Address Map 5-2
5.4 USIU PowerPC Memory Map 5-5

Section 6
SYSTEM CONFIGURATION AND PROTECTION

6.1 System Configuration 6-3
6.1.1 USIU Pins Multiplexing 6-3
6.1.2 Memory Mapping 6-3
6.1.3 Arbitration Support 6-4
6.2 External Master Modes 6-5
6.2.1 Operation of External Master Modes 6-5
6.2.2 Address Decoding for External Accesses 6-6
6.3 USIU General-Purpose I/O 6-6
6.4 Interrupt Controller 6-8
6.4.1 SIU Interrupt Sources Priority 6-11
6.5 Hardware Bus Monitor 6-12
6.6 MPC555 / MPC556 Decrementer 6-12
6.7 MPC555 / MPC556 Time Base (TB) 6-13
6.8 Real-Time Clock (RTC) 6-14
6.9 Periodic Interrupt Timer (PIT) 6-15
6.10 Software Watchdog Timer (SWT) 6-16
6.11 Freeze Operation 6-17
6.12 Low Power Stop Operation 6-17
6.13 System Configuration and Protection Registers 6-18
6.13.1 System Configuration Registers 6-18
6.13.1.1 SIU Module Configuration Register 6-18
6.13.1.2 Internal Memory Map Register 6-21
6.13.1.3 External Master Control Register (EMCR) 6-22
6.13.2 SIU Interrupt Registers 6-23
6.13.2.1 SIPEND Register 6-23
6.13.2.2 SIU Interrupt Mask Register (SIMASK) 6-24
6.13.2.3 SIU Interrupt Edge Level Register (SIEL) 6-25
6.13.2.4 SIU Interrupt Vector Register 6-25
6.13.3 System Protection Registers 6-26

6.13.3.1 System Protection Control Register (SYPCR) 6-26
6.13.3.2 Software Service Register (SWSR) 6-26
6.13.3.3 Transfer Error Status Register (TESR). 6-27
6.13.4 System Timer Registers 6-28
6.13.4.1 Decrementer Register 6-28
6.13.4.2 Time Base SPRs 6-28
6.13.4.3 Time Base Reference Registers. 6-29
6.13.4.4 Time Base Control and Status Register 6-29
6.13.4.5 Real-Time Clock Status and Control Register 6-30
6.13.4.6 Real-Time Clock Register (RTC) 6-31
6.13.4.7 Real-Time Clock Alarm Register (RTCAL) 6-31
6.13.4.8 Periodic Interrupt Status and Control Register (PISCR). 6-32
6.13.4.9 Periodic Interrupt Timer Count Register (PITC) 6-32
6.13.4.10 Periodic Interrupt Timer Register (PITR) 6-33
6.13.5 General-Purpose I/O Registers 6-34
6.13.5.1 SGPIO Data Register 1 (SGPIODT1). 6-34
6.13.5.2 SGPIO Data Register 2 (SGPIODT2). 6-34
6.13.5.3 SGPIO Control Register (SGPIOCR) 6-35

Section 7
RESET

7.1 Reset Operation 7-1
7.1.1 Power On Reset 7-1
7.1.2 Hard Reset 7-2
7.1.3 Soft Reset. 7-2
7.1.4 Loss of Lock 7-3
7.1.5 On-Chip Clock Switch. 7-3
7.1.6 Software Watchdog Reset 7-3
7.1.7 Checkstop Reset 7-3
7.1.8 Debug Port Hard Reset 7-3
7.1.9 Debug Port Soft Reset 7-3
7.1.10 JTAG Reset 7-3
7.2 Reset Actions Summary. 7-3
7.3 Data Coherency During Reset 7-4
7.4 Reset Status Register 7-5
7.5 Reset Configuration 7-6
7.5.1 Hard Reset Configuration 7-6
7.5.2 Hard Reset Configuration Word 7-11
7.5.3 Soft Reset Configuration 7-12

Section 8
CLOCKS AND POWER CONTROL

8.1 Overview 8-1
8.2 System Clock Sources 8-3

Paragraph Number		Page Number

8.3 System PLL 8-3
8.3.1 Frequency Multiplication 8-4
8.3.2 Skew Elimination 8-4
8.3.3 Pre-Divider 8-4
8.3.4 PLL Block Diagram 8-4
8.3.5 PLL Pins 8-5
8.4 System Clock During PLL Loss of Lock 8-6
8.5 Low-Power Divider 8-6
8.6 MPC555 / MPC556 Internal Clock Signals 8-7
8.6.1 General System Clocks 8-9
8.6.2 CLKOUT 8-12
8.6.3 Engineering Clock 8-12
8.7 Clock Source Switching 8-13
8.8 Low-Power Modes 8-15
8.8.1 Entering a Low-Power Mode 8-15
8.8.2 Power Mode Descriptions 8-16
8.8.3 Exiting from Low-Power Modes 8-16
8.8.3.1 Exiting from Normal-Low Mode 8-17
8.8.3.2 Exiting from Doze Mode 8-17
8.8.3.3 Exiting from Deep-Sleep Mode 8-17
8.8.3.4 Exiting from Power-Down Mode 8-18
8.8.3.5 Low-Power Modes Flow 8-18
8.9 Basic Power Structure 8-20
8.9.1 Clock Unit Power Supply 8-20
8.9.2 Chip Power Structure 8-20
8.9.2.1 VDDL 8-20
8.9.2.2 VDDI 8-20
8.9.2.3 VDDSYN, VSSSYN 8-21
8.9.2.4 KAPWR 8-21
8.9.2.5 VDDA, VSSA 8-21
8.9.2.6 VPP 8-21
8.9.2.7 VDDF, VSSF 8-21
8.9.2.8 VDDH 8-21
8.9.2.9 VDDSRAM 8-21
8.9.2.10 VSS 8-21
8.9.3 Keep Alive Power 8-22
8.9.3.1 Keep Alive Power Configuration 8-22
8.9.3.2 Keep Alive Power Registers Lock Mechanism 8-23
8.10 VDDSRAM Supply Failure Detection 8-25
8.11 Power Up/Down Sequencing 8-25
8.12 Clocks Unit Programming Model 8-29
8.12.1 System Clock Control Register (SCCR) 8-30
8.12.2 PLL, Low-Power, and Reset-Control Register (PLPRCR) 8-33

8.12.3 Change of Lock Interrupt Register (COLIR) 8-35
8.12.4 VDDSRAM Control Register (VSRMCR) 8-36

Section 9
EXTERNAL BUS INTERFACE

9.1 Features 9-1
9.2 Bus Transfer Signals 9-1
9.3 Bus Control Signals 9-2
9.4 Bus Interface Signal Descriptions 9-3
9.5 Bus Operations 9-7
9.5.1 Basic Transfer Protocol 9-8
9.5.2 Single Beat Transfer 9-8
9.5.2.1 Single Beat Read Flow 9-8
9.5.2.2 Single Beat Write Flow 9-11
9.5.2.3 Single Beat Flow with Small Port Size 9-14
9.5.3 Burst Transfer 9-15
9.5.4 Burst Mechanism 9-16
9.5.5 Alignment and Packaging of Transfers 9-28
9.5.6 Arbitration Phase 9-30
9.5.6.1 Bus Request 9-31
9.5.6.2 Bus Grant 9-32
9.5.6.3 Bus Busy 9-32
9.5.6.4 Internal Bus Arbiter 9-33
9.5.7 Address Transfer Phase Signals 9-35
9.5.7.1 Transfer Start 9-36
9.5.7.2 Address Bus 9-36
9.5.7.3 Read/Write 9-36
9.5.7.4 Burst Indicator 9-36
9.5.7.5 Transfer Size 9-37
9.5.7.6 Address Types 9-37
9.5.7.7 Burst Data in Progress 9-38
9.5.8 Termination Signals 9-38
9.5.8.1 Transfer Acknowledge 9-38
9.5.8.2 Burst Inhibit 9-39
9.5.8.3 Transfer Error Acknowledge 9-39
9.5.8.4 Termination Signals Protocol 9-39
9.5.9 Storage Reservation 9-40
9.5.10 Bus Exception Control Cycles 9-43
9.5.10.1 Retrying a Bus Cycle 9-43
9.5.10.2 Termination Signals Protocol Summary 9-47
9.5.11 Bus Operation in External Master Modes 9-47
9.5.12 Contention Resolution on External Bus 9-52
9.5.13 Show Cycle Transactions 9-54

Paragraph Number | Page Number

Section 10
MEMORY CONTROLLER

10.1 Overview 10-1
10.2 Memory Controller Architecture 10-3
10.2.1 Associated Registers 10-4
10.2.2 Port Size Configuration. 10-5
10.2.3 Write-Protect Configuration 10-5
10.2.4 Address and Address Space Checking 10-5
10.2.5 Burst Support 10-5
10.3 Chip-Select Timing. 10-6
10.3.1 Memory Devices Interface Example. 10-7
10.3.2 Peripheral Devices Interface Example 10-8
10.3.3 Relaxed Timing Examples 10-10
10.3.4 Extended Hold Time on Read Accesses 10-14
10.3.5 Summary of GPCM Timing Options 10-18
10.4 Global (Boot) Chip-Select Operation 10-20
10.5 Write and Byte Enable Signals. 10-21
10.6 Dual Mapping of the Internal Flash EEPROM Array 10-21
10.7 Memory Controller External Master Support 10-24
10.8 Programming Model. 10-27
10.8.1 General Memory Controller Programming Notes 10-27
10.8.2 Memory Controller Status Registers (MSTAT) 10-28
10.8.3 Memory Controller Base Registers (BR0 – BR3) 10-28
10.8.4 Memory Controller Option Registers (OR0 – OR3) 10-30
10.8.5 Dual Mapping Base Register (DMBR) 10-31
10.8.6 Dual-Mapping Option Register 10-32

Section 11
L-BUS TO U-BUS INTERFACE (L2U)

11.1 General Features 11-1
11.2 DMPU Features 11-1
11.3 L2U Block Diagram 11-2
11.4 Modes Of Operation. 11-2
11.4.1 Normal Mode 11-3
11.4.2 Reset Operation 11-3
11.4.3 Factory Test Mode 11-3
11.4.4 Peripheral Mode 11-3
11.5 Data Memory Protection 11-4
11.5.1 Functional Description 11-4
11.5.2 Associated Registers 11-5
11.5.3 L-bus Memory Access Violations 11-7
11.6 Reservation Support 11-7
11.6.1 The Reservation Protocol 11-7

11.6.2 L2U Reservation Support 11-7
11.6.3 Reserved Location (Bus) and Possible Actions 11-8
11.7 L-Bus Show Cycle Support 11-9
11.7.1 Programming Show Cycles 11-9
11.7.2 Performance Impact 11-9
11.7.3 Show Cycle Protocol 11-10
11.7.4 L-Bus Write Show Cycle Flow 11-10
11.7.5 L-Bus Read Show Cycle Flow 11-11
11.7.6 Show Cycle Support Guidelines 11-11
11.8 L2U Programming Model 11-12
11.8.1 U-bus Access 11-13
11.8.2 Transaction Size 11-13
11.8.3 L2U Module Configuration Register (L2U_MCR) 11-13
11.8.4 Region Base Address Registers (L2U_RBAx) 11-14
11.8.5 Region Attribute Registers (L2U_RAx) 11-15
11.8.6 Global Region Attribute Register 11-15

Section 12
U-BUS TO IMB3 BUS INTERFACE (UIMB)

12.1 Features 12-1
12.2 UIMB Block Diagram 12-2
12.3 Clock Module 12-2
12.4 Interrupt Operation 12-3
12.4.1 Interrupt Sources and Levels on IMB 12-4
12.4.2 IMB Interrupt Multiplexing 12-4
12.4.3 ILBS Sequencing 12-4
12.4.4 Interrupt Synchronizer 12-6
12.5 Programming Model 12-7
12.5.1 UIMB Module Configuration Register (UMCR) 12-7
12.5.2 Test control register (UTSTCREG) 12-8
12.5.3 Pending Interrupt Request Register (UIPEND) 12-8

Section 13
QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64

13.1 Overview 13-1
13.2 Features 13-2
13.3 QADC64 Pin Functions 13-2
13.3.1 Port A Pin Functions 13-3
13.3.1.1 Port A Analog Input Pins 13-3
13.3.1.2 Port A Digital Input/Output Pins 13-3
13.3.2 Port B Pin Functions 13-4
13.3.2.1 Port B Analog Input Pins 13-4
13.3.2.2 Port B Digital Input Pins 13-4
13.3.3 External Trigger Input Pins 13-4

Paragraph Number		Page Number

13.3.4 Multiplexed Address Output Pins . . . 13-4
13.3.5 Multiplexed Analog Input Pins . . . 13-5
13.3.6 Voltage Reference Pins . . . 13-5
13.3.7 Dedicated Analog Supply Pins . . . 13-5
13.3.8 External Digital Supply Pin . . . 13-5
13.3.9 Digital Supply Pins . . . 13-5
13.4 QADC64 Bus Interface. . . . 13-6
13.5 Module Configuration . . . 13-6
13.5.1 Low-Power Stop Mode . . . 13-6
13.5.2 Freeze Mode . . . 13-6
13.5.3 Supervisor/Unrestricted Address Space. . . . 13-7
13.6 General-Purpose I/O Port Operation . . . 13-7
13.6.1 Port Data Register . . . 13-8
13.6.2 Port Data Direction Register. . . . 13-8
13.7 External Multiplexing Operation . . . 13-9
13.8 Analog Input Channels. . . . 13-10
13.9 Analog Subsystem . . . 13-11
13.9.1 Conversion Cycle Times. . . . 13-12
13.9.1.1 Amplifier Bypass Mode Conversion Timing . . . 13-13
13.9.2 Front-End Analog Multiplexer . . . 13-14
13.9.3 Digital-to-Analog Converter Array. . . . 13-14
13.9.4 Comparator. . . . 13-14
13.9.5 Successive Approximation Register . . . 13-14
13.10 Digital Control Subsystem . . . 13-14
13.10.1 Queue Priority. . . . 13-15
13.10.2 Queue Boundary Conditions . . . 13-17
13.10.3 Scan Modes . . . 13-18
13.10.3.1 Disabled Mode . . . 13-18
13.10.3.2 Reserved Mode . . . 13-18
13.10.3.3 Single-Scan Modes . . . 13-18
13.10.3.4 Continuous-Scan Modes . . . 13-21
13.10.4 QADC64 Clock (QCLK) Generation . . . 13-24
13.10.5 Periodic/Interval Timer . . . 13-29
13.11 Interrupts . . . 13-29
13.11.1 Interrupt Sources . . . 13-30
13.11.2 Interrupt Register . . . 13-31
13.11.3 Interrupt Levels and Time Multiplexing. . . . 13-31
13.12 Programming Model. . . . 13-31
13.12.1 QADC64 Module Configuration Register . . . 13-33
13.12.2 QADC64 Test Register. . . . 13-33
13.12.3 QADC64 Interrupt Register . . . 13-33
13.12.4 Port A/B Data Register . . . 13-34
13.12.5 Port Data Direction Register. . . . 13-35

13.12.6 QADC64 Control Register 0 (QACR0) 13-35
13.12.7 QADC64 Control Register 1 (QACR1) 13-36
13.12.8 QADC64 Control Register 2 (QACR2) 13-38
13.12.9 QADC64 Status Register 0 (QASR0) 13-40
13.12.10 QADC64 Status Register 1 (QASR1) 13-42
13.12.11 Conversion Command Word Table 13-43
13.12.12 Result Word Table 13-49

Section 14
QUEUED SERIAL MULTI-CHANNEL MODULE

14.1 Overview 14-1
14.2 Block Diagram 14-1
14.3 Signal Descriptions 14-2
14.4 Memory Map 14-2
14.5 QSMCM Global Registers 14-4
14.5.1 Low-Power Stop Operation 14-5
14.5.2 Freeze Operation 14-5
14.5.3 Access Protection 14-5
14.5.4 QSMCM Interrupts 14-6
14.5.5 QSMCM Configuration Register (QSMCMMCR) 14-7
14.5.6 QSMCM Test Register (QTEST) 14-8
14.5.7 QSMCM Interrupt Level Registers (QDSCI_IL, QSPI_IL) 14-8
14.6 QSMCM Pin Control Registers 14-9
14.6.1 Port QS Data Register (PORTQS) 14-10
14.6.2 PORTQS Pin Assignment Register (PQSPAR) 14-11
14.6.3 PORTQS Data Direction Register (DDRQS) 14-12
14.7 Queued Serial Peripheral Interface 14-13
14.7.1 QSPI Registers 14-15
14.7.1.1 QSPI Control Register 0 14-16
14.7.1.2 QSPI Control Register 1 14-18
14.7.1.3 QSPI Control Register 2 14-18
14.7.1.4 QSPI Control Register 3 14-19
14.7.1.5 QSPI Status Register 14-20
14.7.2 QSPI RAM 14-21
14.7.2.1 Receive RAM 14-22
14.7.2.2 Transmit RAM. 14-22
14.7.2.3 Command RAM 14-22
14.7.3 QSPI Pins 14-23
14.7.4 QSPI Operation 14-24
14.7.4.1 Enabling, Disabling, and Halting the SPI 14-25
14.7.4.2 QSPI Interrupts. 14-26
14.7.4.3 QSPI Flow 14-26
14.7.5 Master Mode Operation 14-33

Paragraph Number		Page Number

14.7.5.1 Clock Phase and Polarity 14-34
14.7.5.2 Baud Rate Selection 14-34
14.7.5.3 Delay Before Transfer 14-35
14.7.5.4 Delay After Transfer 14-35
14.7.5.5 Transfer Length 14-36
14.7.5.6 Peripheral Chip Selects 14-36
14.7.5.7 Master Wraparound Mode 14-37
14.7.6 Slave Mode 14-37
14.7.6.1 Description of Slave Operation 14-39
14.7.7 Slave Wraparound Mode 14-40
14.7.8 Mode Fault 14-41
14.8 Serial Communication Interface 14-41
14.8.1 SCI Registers 14-44
14.8.2 SCI Control Register 0 14-45
14.8.3 SCI Control Register 1 14-45
14.8.4 SCI Status Register (SCxSR) 14-47
14.8.5 SCI Data Register (SCxDR) 14-49
14.8.6 SCI Pins 14-50
14.8.7 SCI Operation 14-50
14.8.7.1 Definition of Terms 14-50
14.8.7.2 Serial Formats 14-51
14.8.7.3 Baud Clock 14-51
14.8.7.4 Parity Checking 14-52
14.8.7.5 Transmitter Operation 14-52
14.8.7.6 Receiver Operation 14-54
14.8.7.7 Receiver Functional Operation 14-56
14.8.7.8 Idle-Line Detection 14-57
14.8.7.9 Receiver Wake-Up 14-58
14.8.7.10 Internal Loop Mode 14-58
14.9 SCI Queue Operation 14-58
14.9.1 Queue Operation of SCI1 for Transmit and Receive 14-58
14.9.2 Queued SCI1 Status and Control Registers 14-59
14.9.2.1 QSCI1 Control Register 14-59
14.9.2.2 QSCI1 Status Register 14-61
14.9.3 QSCI1 Transmitter Block Diagram 14-61
14.9.4 QSCI1 Additional Transmit Operation Features 14-62
14.9.5 QSCI1 Transmit Flow Chart Implementing the Queue 14-64
14.9.6 Example QSCI1 Transmit for 17 Data Bytes 14-66
14.9.7 Example SCI Transmit for 25 Data Bytes 14-67
14.9.8 QSCI1 Receiver Block Diagram 14-68
14.9.9 QSCI1 Additional Receive Operation Features 14-68
14.9.10 QSCI1 Receive Flow Chart Implementing The Queue 14-71
14.9.11 QSCI1 Receive Queue Software Flow Chart 14-72

14.9.12 Example QSCI1 Receive Operation of 17 Data Frames 14-73

Section 15
MODULAR INPUT/OUTPUT SUBSYSTEM (MIOS1)

15.1 MIOS1 Features 15-1
15.2 Submodule Numbering, Naming and Addressing 15-3
15.3 MIOS1 Signals 15-3
15.4 Block Diagram 15-4
15.5 MIOS1 Bus System 15-6
15.5.1 Read/Write and Control Bus 15-6
15.5.2 Request Bus 15-6
15.5.3 Counter Bus Set 15-6
15.6 MIOS1 Programmer's Model 15-6
15.7 MIOS1 I/O Ports 15-8
15.8 MIOS Bus Interface Submodule (MBISM) 15-8
15.8.1 MIOS Bus Interface (MBISM) Registers 15-8
15.8.1.1 MIOS1 Test and Pin Control Register 15-8
15.8.1.2 MIOS1 Vector Register 15-9
15.8.1.3 MIOS1 Module and Version Number Register 15-9
15.8.1.4 MIOS1 Module Configuration Register 15-9
15.8.2 MBISM Interrupt Registers 15-10
15.8.2.1 MIOS1 Interrupt Level Register 0 (MIOS1LVL0) 15-10
15.8.2.2 MIOS1 Interrupt Level Register 1 (MIOS1LVL1) 15-11
15.8.3 Interrupt Control Section (ICS) 15-11
15.9 MIOS Counter Prescaler Submodule (MCPSM) 15-12
15.9.1 MIOS Counter Prescaler Submodule (MCPSM) Registers 15-12
15.9.1.1 MCPSM Status/Control Register (MCPSMCSCR) 15-13
15.10 MIOS Modulus Counter Submodule (MMCSM) 15-13
15.10.1 MIOS Modulus Counter Submodule (MMCSM) Registers 15-15
15.10.1.1 MMCSM Up-Counter Register (MMCSMCNT) 15-16
15.10.1.2 MMCSM Modulus Latch Register (MMCSMML) 15-16
15.10.1.3 MMCSM Status/Control Register (Duplicated) 15-16
15.10.1.4 MMCSM Status/Control Register (MMCSMSCR) 15-17
15.11 MIOS Double Action Submodule (MDASM) 15-18
15.11.1 MIOS Double Action Submodule (MDASM) Registers 15-19
15.11.1.1 MDASM Data A Register 15-21
15.11.1.2 MDASM Data B Register (MDASMBR) 15-21
15.11.1.3 MDASM Status/Control Register (Duplicated) 15-22
15.11.1.4 MDASM Status/Control Register 15-23
15.12 MIOS Pulse Width Modulation Submodule (MPWMSM) 15-25
15.12.1 MIOS Pulse Width Modulation Submodule (MPWMSM) Registers 15-26
15.12.1.1 MPWMSM Period Register (MPWMSMPERR) 15-27
15.12.1.2 MPWMSM Pulse Width Register (MPWMSMPULR) 15-27

15.12.1.3 MPWMSM Counter Register (MPWMSMCNTR) 15-28
15.12.1.4 MPWMSM Status/Control Register(MPWMSMCR) 15-28
15.13 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM) 15-30
15.13.1 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM) Registers. 15-30
15.13.1.1 MPIOSM Data Register (MPIOSMDR). 15-30
15.13.1.2 MPIOSM Data Direction Register (MPIOSMDDR) 15-31
15.14 MIOS1 Interrupts 15-31
15.14.1 MIOS Interrupt Request Submodule (MIRSM) 15-32
15.14.2 MIOS Interrupt Request Submodule 0 (MIRSM0) Registers 15-33
15.14.2.1 MIRSM0 Interrupt Status Register (MIOS1SR0) 15-34
15.14.2.2 MIRSM0 Interrupt Enable Register (MIOS1ER0). 15-35
15.14.2.3 MIRSM0 Request Pending Register (MIOS1RPR0) 15-35
15.14.3 MIOS Interrupt Request Submodule 1 (MIRSM1) Registers 15-36
15.14.3.1 MIRSM1 Interrupt Status Register (MIOS1SR1) 15-36
15.14.3.2 MIRSM1 Interrupt Enable Register (MIOS1ER1). 15-37
15.14.3.3 MIRSM1 Request Pending Register (MIOS1RPR1) 15-37
15.15 MIOS1 Function Examples 15-38
15.15.1 MIOS1 Input Double Edge Pulse Width Measurement 15-38
15.15.2 MIOS1 Input Double Edge Period Measurement 15-40
15.15.3 MIOS1 Double Edge Single Output Pulse Generation 15-41
15.15.4 MIOS1 Output Pulse Width Modulation With MDASM 15-42
15.15.5 MIOS1 Input Pulse Accumulation. 15-43
15.16 MIOS1 Configuration 15-43

Section 16
CAN 2.0B CONTROLLER MODULE

16.1 Features. 16-1
16.2 External Pins 16-2
16.3 TouCAN Architecture 16-3
16.3.1 TX/RX Message Buffer Structure 16-3
16.3.1.1 Common Fields for Extended and Standard Format Frames. 16-4
16.3.1.2 Fields for Extended Format Frames. 16-6
16.3.1.3 Fields for Standard Format Frames 16-6
16.3.1.4 Serial Message Buffers 16-6
16.3.1.5 Message Buffer Activation/Deactivation Mechanism 16-7
16.3.1.6 Message Buffer Lock/Release/Busy Mechanism 16-7
16.3.2 Receive Mask Registers. 16-7
16.3.3 Bit Timing 16-8
16.3.3.1 Configuring the TouCAN Bit Timing 16-9
16.3.4 Error Counters 16-9
16.3.5 Time Stamp 16-10
16.4 TouCAN Operation. 16-11
16.4.1 TouCAN Reset 16-11

Paragraph Number | Page Number

16.4.2 TouCAN Initialization 16-11
16.4.3 Transmit Process 16-12
16.4.3.1 Transmit Message Buffer Deactivation. 16-13
16.4.3.2 Reception of Transmitted Frames 16-13
16.4.4 Receive Process. 16-13
16.4.4.1 Receive Message Buffer Deactivation 16-14
16.4.4.2 Locking and Releasing Message Buffers 16-15
16.4.5 Remote Frames 16-15
16.4.6 Overload Frames 16-16
16.5 Special Operating Modes. 16-16
16.5.1 Debug Mode 16-16
16.5.2 Low-Power Stop Mode 16-17
16.5.3 Auto Power Save Mode 16-18
16.6 Interrupts 16-19
16.7 Programmer's Model 16-20
16.7.1 TouCAN Module Configuration Register 16-22
16.7.2 TouCAN Test Configuration Register 16-24
16.7.3 TouCAN Interrupt Configuration Register. 16-24
16.7.4 Control Register 0. 16-25
16.7.5 Control Register 1. 16-26
16.7.6 Prescaler Divide Register 16-27
16.7.7 Control Register 2. 16-28
16.7.8 Free Running Timer 16-29
16.7.9 Receive Global Mask Registers 16-29
16.7.10 Receive Buffer 14 Mask Registers 16-30
16.7.11 Receive Buffer 15 Mask Registers 16-30
16.7.12 Error and Status Register. 16-30
16.7.13 Interrupt Mask Register 16-32
16.7.14 Interrupt Flag Register 16-33
16.7.15 Error Counters 16-33

Section 17
TIME PROCESSOR UNIT 3

17.1 Overview 17-1
17.2 TPU3 Components. 17-2
17.2.1 Time Bases. 17-2
17.2.2 Timer Channels 17-2
17.2.3 Scheduler 17-2
17.2.4 Microengine 17-2
17.2.5 Host Interface 17-3
17.2.6 Parameter RAM 17-3
17.3 TPU Operation 17-3
17.3.1 Event Timing. 17-3

Paragraph Number		Page Number

17.3.2 Channel Orthogonality 17-3
17.3.3 Interchannel Communication 17-4
17.3.4 Programmable Channel Service Priority 17-4
17.3.5 Coherency 17-4
17.3.6 Emulation Support 17-4
17.3.7 TPU3 Interrupts 17-5
17.3.8 Prescaler Control for TCR1 17-5
17.3.9 Prescaler Control for TCR2 17-7
17.4 Programming Model. 17-8
17.4.1 TPU Module Configuration Register. 17-10
17.4.2 TPU3 Test Configuration Register 17-12
17.4.3 Development Support Control Register 17-12
17.4.4 Development Support Status Register 17-14
17.4.5 TPU3 Interrupt Configuration Register 17-14
17.4.6 Channel Interrupt Enable Register 17-15
17.4.7 Channel Function Select Registers 17-15
17.4.8 Host Sequence Registers. 17-16
17.4.9 Host Service Request Registers. 17-17
17.4.10 Channel Priority Registers 17-18
17.4.11 Channel Interrupt Status Register 17-19
17.4.12 Link Register. 17-19
17.4.13 Service Grant Latch Register 17-19
17.4.14 Decoded Channel Number Register. 17-19
17.4.15 TPU3 Module Configuration Register 2 17-20
17.4.16 TPU Module Configuration Register 3 17-21
17.4.17 TPU3 Test Registers 17-22
17.4.18 TPU3 Parameter RAM 17-22
17.5 Time Functions. 17-23

Section 18
DUAL-PORT TPU RAM (DPTRAM)

18.1 Features. 18-1
18.2 DPTRAM Configuration and Block Diagram 18-2
18.3 Programming Model. 18-2
18.3.1 DPTRAM Module Configuration Register (DPTMCR) 18-3
18.3.2 DPTRAM Test Register 18-4
18.3.3 Ram Base Address Register (RAMBAR) 18-5
18.3.4 MISR High (MISRH) and MISR Low (MISRL) 18-5
18.3.5 MISC Counter (MISCNT) 18-6
18.4 Operation 18-6
18.4.1 Normal Operation 18-6
18.4.2 Standby Operation 18-6
18.4.3 Reset Operation 18-7

18.4.4 Stop Operation 18-7
18.4.5 Freeze Operation 18-8
18.4.6 TPU3 Emulation Mode Operation. 18-8
18.5 Multiple Input Signature Calculator (MISC) 18-8

Section 19
CDR MoneT FLASH EEPROM

19.1 Introduction 19-1
19.1.1 MPC555 / MPC556 CMF Features. 19-2
19.1.2 Glossary of Terms for the CMF EEPROM 19-2
19.2 Programming Model. 19-4
19.2.1 CMF EEPROM Control Registers 19-4
19.2.1.1 CMF EEPROM Configuration Register (CMFMCR) 19-5
19.2.1.2 CMF EEPROM Test Register (CMFTST). 19-7
19.2.1.3 CMF EEPROM High Voltage Control Register (CMFCTL). 19-9
19.2.2 CMF EEPROM Array Addressing. 19-11
19.2.2.1 Read Page Buffers 19-12
19.2.2.2 Program Page Buffers 19-13
19.2.2.3 Array Configuration for CMF Module A 19-14
19.2.2.4 Array Configuration for CMF Module B 19-15
19.3 Shadow Information 19-15
19.3.1 Address Range of Shadow Information 19-16
19.3.2 Reset Configuration Word (CMFCFIG) 19-16
19.4 Array Read Operation 19-17
19.5 Programming the CMF Array 19-18
19.5.1 Program Sequence. 19-18
19.5.2 Program Margin Reads 19-22
19.5.3 Over-Programming 19-23
19.6 Erasing CMF Array Blocks 19-23
19.6.1 Erase Sequence 19-23
19.6.2 Erase Margin Reads. 19-26
19.6.3 Erasing Shadow Information Words 19-26
19.7 Voltage Control for Programming and Erasing 19-27
19.7.1 Pulse Status 19-27
19.7.2 Pulse Width Timing Equation 19-27
19.7.3 System Clock Scaling. 19-28
19.7.4 Exponential Clock Multiplier 19-29
19.7.5 Linear Clock Multiplier 19-29
19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM 19-29
19.7.7 Starting and Ending a Program or Erase Sequence 19-30
19.7.8 Controlling the Program/Erase Voltage 19-31
19.8 Censored and Non-Censored Accesses 19-31
19.8.1 Uncensored Mode 19-31

19.8.2 Censored Mode 19-31
19.8.3 Device Modes and Censorship Status 19-32
19.8.4 Setting and Clearing Censor 19-33
19.8.5 Switching the CMF EEPROM Censorship 19-35
19.9 Pin Descriptions 19-36
19.9.1 E_{PEE} Signal 19-36
19.9.2 FLASH Program/Erase Voltage Conditioning 19-37
19.10 Reset Operation 19-39
19.10.1 Master Reset 19-39
19.10.2 Soft Reset 19-39
19.10.3 Emulation Operation 19-40
19.11 Disabling the CMF Module 19-40

Section 20
STATIC RANDOM ACCESS MEMORY (SRAM)

20.1 Features 20-1
20.2 Block Diagram 20-1
20.3 Programming Model 20-2
20.3.1 SRAM Module Configuration Register (SRAMMCR) 20-2
20.3.2 **SRAM Test Register (SRAMTST)** 20-3

Section 21
DEVELOPMENT SUPPORT

21.1 Overview 21-1
21.2 Program Flow Tracking 21-1
21.2.1 Program Trace Cycle 21-2
21.2.1.1 Instruction Queue Status Pins — VF [0:2] 21-3
21.2.1.2 History Buffer Flushes Status Pins— VFLS [0..1] 21-4
21.2.1.3 Queue Flush Information Special Case 21-4
21.2.2 Program Trace when in Debug Mode 21-4
21.2.3 Sequential Instructions Marked as Indirect Branch 21-5
21.2.4 The External Hardware 21-5
21.2.4.1 Synchronizing the Trace Window to the CPU Internal Events 21-5
21.2.4.2 Detecting the Trace Window Start Address 21-6
21.2.4.3 Detecting the Assertion/Negation of VSYNC 21-7
21.2.4.4 Detecting the Trace Window End Address 21-7
21.2.4.5 Compress 21-7
21.2.5 Instruction Fetch Show Cycle Control 21-8
21.3 Watchpoints and Breakpoints Support 21-8
21.3.1 Internal Watchpoints and Breakpoints 21-11
21.3.1.1 Restrictions 21-13
21.3.1.2 Byte and Half-Word Working Modes 21-13
21.3.1.3 Examples 21-14
21.3.1.4 Context Dependent Filter 21-15

Paragraph Number		Page Number

21.3.1.5 Ignore First Match . . . 21-15
21.3.1.6 Generating Six Compare Types . . . 21-16
21.3.2 Instruction Support . . . 21-16
21.3.2.1 Load/Store Support . . . 21-17
21.3.3 Watchpoint Counters . . . 21-21
21.3.3.1 Trap Enable Programming . . . 21-21
21.4 Development System Interface . . . 21-21
21.4.1 Debug Mode Support . . . 21-24
21.4.1.1 Debug Mode Enable vs. Debug Mode Disable . . . 21-26
21.4.1.2 Entering Debug Mode . . . 21-26
21.4.1.3 The Check Stop State and Debug Mode . . . 21-29
21.4.1.4 Saving Machine State upon Entering Debug Mode . . . 21-29
21.4.1.5 Running in Debug Mode . . . 21-30
21.4.1.6 Exiting Debug Mode . . . 21-30
21.5 Development Port . . . 21-31
21.5.1 Development Port Pins . . . 21-31
21.5.2 Development Serial Clock . . . 21-31
21.5.3 Development Serial Data In . . . 21-31
21.5.4 Development Serial Data Out . . . 21-32
21.5.5 Freeze Signal . . . 21-32
21.5.5.1 SGPIO6/FRZ/$\overline{\text{PTR}}$ Pin . . . 21-32
21.5.5.2 IWP[0:1]/VFLS[0:1] Pins . . . 21-32
21.5.5.3 VFLS[0:1]_MPIO32B[3:4] Pins . . . 21-32
21.5.6 Development Port Registers . . . 21-32
21.5.6.1 Development Port Shift Register . . . 21-33
21.5.6.2 Trap Enable Control Register . . . 21-33
21.5.6.3 Development Port Registers Decode . . . 21-33
21.5.6.4 Development Port Serial Communications — Clock Mode Selection . . . 21-34
21.5.6.5 Development Port Serial Communications — Trap Enable Mode . . . 21-38
21.5.6.6 Serial Data into Development Port — Trap Enable Mode . . . 21-38
21.5.6.7 Serial Data Out of Development Port — Trap Enable Mode . . . 21-39
21.5.6.8 Development Port Serial Communications — Debug Mode . . . 21-39
21.5.6.9 Serial Data Into Development Port . . . 21-40
21.5.6.10 Serial Data Out of Development Port . . . 21-41
21.5.6.11 Fast Download Procedure . . . 21-41
21.6 Software Monitor Debugger Support . . . 21-43
21.6.1 Freeze Indication . . . 21-43
21.7 Development Support Registers . . . 21-43
21.7.1 Register Protection . . . 21-44
21.7.2 Comparator A–D Value Registers (CMPA–CMPD) . . . 21-45
21.7.3 Comparator E–F Value Registers . . . 21-46
21.7.4 Breakpoint Address Register (BAR) . . . 21-46
21.7.5 Comparator G–H Value Registers (CMPG–CMPH) . . . 21-46

21.7.6 I-Bus Support Control Register 21-47
21.7.7 L-Bus Support Control Register 1 21-49
21.7.8 L-Bus Support Control Register 2 21-50
21.7.9 Breakpoint Counter A Value and Control Register 21-52
21.7.10 Breakpoint Counter B Value and Control Register 21-53
21.7.11 Exception Cause Register (ECR) 21-53
21.7.12 Debug Enable Register (DER) 21-55
21.7.13 Development Port Data Register (DPDR) 21-57

Section 22
IEEE 1149.1-COMPLIANT INTERFACE (JTAG)

22.1 JTAG Interface Block Diagram 22-1
22.2 JTAG Signal Descriptions 22-2
22.3 Operating Frequency 22-3
22.4 TAP Controller 22-3
22.5 Instruction Register 22-4
22.5.1 EXTEST 22-5
22.5.2 SAMPLE/PRELOAD 22-5
22.5.3 BYPASS 22-5
22.5.4 CLAMP 22-6
22.5.5 HI-Z 22-6
22.6 Restrictions 22-6
22.7 Low-Power Stop Mode 22-6
22.8 Non-IEEE 1149.1-1990 Operation 22-7
22.9 Boundary Scan Register 22-7

Appendix A
MPC555 / MPC556 INTERNAL MEMORY MAP

Appendix B
REGISTER GENERAL INDEX

Appendix C
REGISTER DIAGRAM INDEX

Appendix D
TPU ROM FUNCTIONS

D.1 Overview D-1
D.2 Programmable Time Accumulator (PTA) D-4
D.3 Queued Output Match TPU Function (QOM) D-6
D.4 Table Stepper Motor (TSM) D-8
D.5 Frequency Measurement (FQM) D-11
D.6 Universal Asynchronous Receiver/Transmitter (UART) D-13
D.7 New Input Capture/Transition Counter (NITC) D-16
D.8 Multiphase Motor Commutation (COMM) D-18
D.9 Hall Effect Decode (HALLD) D-20

Paragraph Number | Page Number

D.10 Multichannel Pulse-Width Modulation (MCPWM) D-22
D.11 Fast Quadrature Decode TPU Function (FQD) D-29
D.12 Period/Pulse-Width Accumulator (PPWA) D-32
D.13 Output Compare (OC) D-34
D.14 Pulse-Width Modulation (PWM) D-36
D.15 Discrete Input/Output (DIO) D-38
D.16 Synchronized Pulse-Width Modulation (SPWM) D-40
D.17 Read / Write Timers and Pin TPU Function (RWTPIN) D-43
D.18 ID TPU Function (ID) D-45
D.19 Serial Input/Output Port (SIOP) D-47
D.19.1 Parameters D-48
D.19.1.1 CHAN_CONTROL D-50
D.19.1.2 BIT_D D-50
D.19.1.3 HALF_PERIOD D-50
D.19.1.4 BIT_COUNT D-50
D.19.1.5 XFER_SIZE D-50
D.19.1.6 SIOP_DATA D-50
D.19.2 Host CPU Initialization of the SIOP Function D-51
D.19.3 SIOP Function Performance D-51
D.19.3.1 XFER_SIZE Greater Than 16 . . . . . D-52
D.19.3.2 Data Positioning D-52
D.19.3.3 Data Timing D-52

Appendix E
CLOCK AND BOARD GUIDELINES

E.1 INTRODUCTION E-1
E.2 MPC555 / MPC556 Family Power Distribution E-2
E.3 PLL and Crystal Oscillator External Components E-4
E.3.1 Crystal Oscillator External Components E-4
E.3.2 KAPWR Filtering E-5
E.3.3 PLL External Components E-6
E.3.4 PLL Off-Chip Capacitor C_{XFC} E-7
E.4 Clock Oscillator and PLL External Components Layout Requirements E-7
E.4.1 Traces and Placement E-7
E.4.2 Grounding/Guarding E-8

Appendix F
MEMORY ACCESS TIMING

F.1 Introduction F-1

Appendix G
ELECTRICAL CHARACTERISTICS

G.1 Absolute Maximum Ratings (V_{SS} = 0 V) G-1
G.2 Target Failure Rate G-2
G.3 Package G-2
G.4 EMI Characteristics G-2
G.4.1 Reference Documents G-2
G.4.2 Definitions and Acronyms G-2
G.4.3 Testing Characteristics G-3
G.5 Thermal Characteristics G-3
G.5.1 Thermal References: G-5
G.6 ESD Protection G-6

Paragraph Number	Page Number

G.7 DC Electrical Characteristics G-7
G.8 Oscillator and PLL Electrical Characteristics G-12
G.9 Power Up/Down Sequencing G-12
G.10 FLASH Electrical Characteristics G-13
G.10.1 Flash Module Life G-14
G.10.2 Programming and Erase Algorithm G-15
G.11 Generic Timing G-16
G.12 Interrupt Timing G-39
G.13 Debug Port Timing G-40
G.14 Reset Timing G-43
G.15 IEEE 1149.1 Electrical Characteristics G-47
G.16 QADC64 Electrical Characteristics G-52
G.17 QSMCM Electrical Characteristics G-53
G.18 GPIO Electrical Characteristics G-57
G.19 TPU3 Electrical Characteristics G-57
G.20 TouCAN Electrical Characteristics G-58
G.21 MIOS Timing Characteristics G-59
G.21.1 MPWMSM Timing Characteristics G-60
G.21.2 MMCSM Timing Characteristics G-62
G.21.3 MDASM Timing Characteristics G-65
G.21.4 MPIOSM Timing Characteristics G-68

Appendix H
FLASH ELECTRICAL CHARACTERISTICS FOR ALL J76N MASK SETS AND 0K02A AND 1K02A ONLY

H.1 Electrical Characteristics H-1
H.1.1 Flash Module Life H-3
H.2 Programming and Erase Algorithm H-3

INDEX

Online publishing by JABIS™, **http://www.jabis.com**

LIST OF FIGURES

Figure Number		Page Number
1-1	MPC555 / MPC556 Block Diagram	1-2
1-2	MPC555 / MPC556 Memory Map	1-6
1-3	MPC555 / MPC556 Internal Memory Map	1-7
2-1	MPC555 / MPC556 Case Dimensions and Packaging	2-2
2-2	MPC555 / MPC556 Pinout Data	2-3
2-3	Type A Interface	2-39
2-4	Type B Interface	2-39
2-5	Type C Interface	2-40
2-6	Type CH Interface	2-40
2-7	Type CNH Interface	2-41
2-8	Type D Interface	2-41
2-9	Type E Interface	2-42
2-10	3-V Type EOH Interface	2-43
2-11	Type F Interface	2-44
2-12	Type G Interface	2-45
2-13	Type H Interface	2-46
2-14	Type I Interface	2-47
2-15	Type IH Interface	2-48
2-16	Type J Interface	2-49
2-17	Type JD Interface	2-50
2-18	EPEE Pad (Type K)	2-51
2-19	Type L Interface	2-52
2-20	Type M Interface	2-52
2-21	Type N Interface	2-53
2-22	Type O Interface	2-54
2-23	Type P Interface	2-55
2-24	Type Q Interface	2-56
2-25	Type R Interface	2-56
2-26	Type S Interface	2-57
3-1	RCPU Block Diagram	3-2
3-2	Sequencer Data Path	3-4
3-3	RCPU Programming Model	3-8
3-4	Basic Instruction Pipeline	3-37
4-1	Burst Buffer Block Diagram	4-2
4-2	Example of Compressed Code	4-5
4-3	Instruction Coding	4-5
4-4	Two Streams Memory Organization — Before Compression	4-6
4-5	Two Streams Memory Organization — After Compression	4-6
4-6	Examples of Compressed Symbols Layout	4-7
4-7	Compressed Address Format	4-8

Figure Number		Page Number

4-8 Examples of Instruction Layout in Memory 4-9
4-9 Generating Compressed Code Address for PowerPC Direct Branches 4-10
4-10 Extracting Direct Branch Target Address in the Decompressor 4-11
4-11 Code Compression Process (Phase A) 4-12
4-12 Bounded Huffman Code Tree 4-13
4-13 Code Decompression Process 4-14
4-14 Exception Table Entries Mapping 4-19

5-1 MPC555 / MPC556 USIU Block Diagram 5-2

6-1 System Configuration and Protection Logic 6-2
6-2 MPC555 / MPC556 Memory Map 6-4
6-3 SGPIO Cell 6-8
6-4 MPC555 / MPC556 Interrupt Structure 6-9
6-5 MPC555 / MPC556 Interrupt Configuration 6-11
6-6 RTC Block Diagram 6-14
6-7 PIT Block Diagram 6-15
6-8 SWT Interrupts and Exceptions 6-16
6-9 SWT Block Diagram 6-17

7-1 Reset Configuration Basic Scheme 7-7
7-2 Reset Configuration Sampling Scheme For “Short” PORESET Assertion, Limp Mode Disabled 7-8
7-3 Reset Configuration Timing for “Short” $\overline{\text{PORESET}}$ Assertion, Limp Mode Enabled 7-9
7-4 Reset Configuration Timing for “Long” $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled 7-9
7-5 Reset Configuration Sampling Timing Requirements 7-10

8-1 Clock Unit Block Diagram 8-2
8-2 Main System Oscillator (OSCM) 8-3
8-3 System PLL Block Diagram 8-5
8-4 MPC555 / MPC556 Clocks 8-7
8-5 General System Clocks Select 8-10
8-6 Divided System Clocks Timing Diagram 8-11
8-7 Clocks Timing For DFNH = 1 (or DFNL = 0) 8-12
8-8 Clock Source Flow Chart 8-14
8-9 MPC555 / MPC556 Low-Power Modes Flow Diagram 8-19
8-10 Basic Power Supply Configuration 8-22
8-11 External Power Supply Scheme 8-23
8-12 Keep Alive Register Key State Diagram 8-25
8-13 No Standby, No KAPWR, All System Power On/Off 8-27
8-14 Standby and KAPWR, Other Power On/Off 8-28

Figure Number		Page Number

9-1 Input Sample Window 9-2
9-2 MPC555 / MPC556 Bus Signals 9-3
9-3 Basic Transfer Protocol 9-8
9-4 Basic Flow Diagram of a Single Beat Read Cycle 9-9
9-5 Single Beat Read Cycle–Basic Timing–Zero Wait States 9-10
9-6 Single Beat Read Cycle–Basic Timing–One Wait State 9-11
9-7 Basic Flow Diagram of a Single Beat Write Cycle 9-12
9-8 Single Beat Basic Write Cycle Timing, Zero Wait States 9-13
9-9 Single Beat Basic Write Cycle Timing, One Wait State 9-14
9-10 Single Beat 32-Bit Data Write Cycle Timing, 16 Bit-Port Size 9-15
9-11 Basic Flow Diagram Of A Burst Read Cycle 9-18
9-12 Burst-Read Cycle–32-Bit Port Size–Zero Wait State 9-19
9-13 Burst-Read Cycle–32-Bit Port Size–One Wait State 9-20
9-14 Burst-Read Cycle–32-Bit Port Size–Wait States Between Beats 9-21
9-15 Burst-Read Cycle, 16-Bit Port Size 9-22
9-16 Basic Flow Diagram of a Burst Write Cycle 9-23
9-17 Burst-Write Cycle, 32-Bit Port Size, Zero Wait States 9-24
9-18 Burst-Inhibit Cycle, 32-Bit Port Size (Emulated Burst) 9-25
9-19 Non-Wrap Burst with Three Beats 9-26
9-20 Non-Wrap Burst with One Data Beat 9-27
9-21 Internal Operand Representation 9-28
9-22 Interface To Different Port Size Devices 9-29
9-23 Bus Arbitration Flowchart 9-31
9-24 Masters Signals Basic Connection 9-32
9-25 Bus Arbitration Timing Diagram 9-33
9-26 Internal Bus Arbitration State Machine 9-35
9-27 Termination Signals Protocol Basic Connection 9-39
9-28 Termination Signals Protocol Timing Diagram 9-40
9-29 Reservation On Local Bus 9-41
9-30 Reservation On Multilevel Bus Hierarchy 9-42
9-31 Retry Transfer Timing–Internal Arbiter 9-44
9-32 Retry Transfer Timing–External Arbiter 9-45
9-33 Retry On Burst Cycle 9-46
9-34 Basic Flow of an External Master Read Access 9-48
9-35 Basic Flow of an External Master Write Access 9-49
9-36 Peripheral Mode: External Master Reads from MPC555 / MPC556 — Two Wait States 9-50
9-37 Peripheral Mode: External Master Writes to MPC555 / MPC556; Two Wait States 9-51
9-38 Flow of Retry of External Master Read Access 9-53
9-39 Retry of External Master Access (Internal Arbiter) 9-54
9-40 Instruction Show Cycle Transaction 9-55
9-41 Data Show Cycle Transaction 9-56

Figure Number		Page Number

10-1 Memory Controller Function Within the USIU 10-1
10-2 Memory Controller Block Diagram 10-2
10-3 MPC555 / MPC556 Simple System Configuration 10-3
10-4 Bank Base Address and Match Structure 10-4
10-5 MPC555 / MPC556 GPCM–Memory Devices Interface 10-7
10-6 Memory Devices Interface Basic Timing (ACS = 00,TRLX = 0) 10-8
10-7 Peripheral Devices Interface 10-9
10-8 Peripheral Devices Basic Timing (ACS = 11,TRLX = 0) 10-9
10-9 Relaxed Timing–Read Access (ACS = 11, SCY = 1, TRLX = 1) 10-11
10-10 Relaxed Timing–Write Access (ACS = 10, SCY = 0, CSNT = 0, TRLX = 1) 10-12
10-11 Relaxed Timing–Write Access (ACS = 11, SCY = 0, CSNT = 1, TRLX = 1) 10-13
10-12 Relaxed Timing–Write Access (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1 10-14
10-13 Consecutive Accesses (Write After Read, EHTR = 0) 10-15
10-14 Consecutive Accesses (Write After Read, EHTR = 1) 10-16
10-15 Consecutive Accesses (Read After Read From Different Banks, EHTR = 1) 10-17
10-16 Consecutive Accesses (Read After Read From Same Bank, EHTR = 1) 10-18
10-17 Aliasing Phenomena Illustration 10-23
10-18 Synchronous External Master Configuration For GPCM–Handled Memory Devices 10-25
10-19 Synchronous External Master Basic Access (GPCM Controlled) 10-26

11-1 L2U Bus Interface Block Diagram 11-2
11-2 DMP Basic Functional Diagram 11-4
11-3 Region Base Address Example 11-6

12-1 UIMB Interface Module Block Diagram 12-2
12-2 IMB Clock – Full-Speed IMB Bus 12-3
12-3 IMB Clock – Half-Speed IMB Bus 12-3
12-4 Interrupt Synchronizer Signal Flow 12-4
12-5 Time-Multiplexing Protocol for IRQ pins 12-5
12-6 Interrupt Synchronizer Block diagram 12-6

13-1 QADC64 Block Diagram 13-1
13-2 QADC64 Input and Output Signals 13-3
13-3 Example of External Multiplexing 13-10
13-4 QADC64 Module Block Diagram 13-12
13-5 Conversion Timing 13-13
13-6 Bypass Mode Conversion Timing 13-13

Figure Number		Page Number

13-7 QADC64 Queue Operation with Pause 13-16
13-8 QADC64 Clock Subsystem Functions 13-26
13-9 QADC64 Clock Programmability Examples 13-28
13-10 QADC64 Interrupt Flow Diagram 13-30
13-11 Interrupt Levels on IRQ with ILBS 13-31
13-12 QADC64 Conversion Queue Operation 13-44

14-1 QSMCM Block Diagram 14-2
14-2 QSMCM Interrupt Levels 14-6
14-3 QSPI Interrupt Generation 14-7
14-4 QSPI Block Diagram 14-14
14-5 QSPI RAM 14-22
14-6 Flowchart of QSPI Initialization Operation 14-27
14-7 Flowchart of QSPI Master Operation (Part 1) 14-28
14-8 Flowchart of QSPI Master Operation (Part 2) 14-29
14-9 Flowchart of QSPI Master Operation (Part 3) 14-30
14-10 Flowchart of QSPI Slave Operation (Part 1) 14-31
14-11 Flowchart of QSPI Slave Operation (Part 2) 14-32
14-12 SCI Transmitter Block Diagram 14-42
14-13 SCI Receiver Block Diagram 14-43
14-14 Start Search Example 14-56
14-15 Queue Transmitter Block Enhancements 14-62
14-16 Queue Transmit Flow 14-64
14-17 Queue Transmit Software Flow 14-65
14-18 Queue Transmit Example for 17 Data Bytes 14-66
14-19 Queue Transmit Example for 25 Data Frames 14-67
14-20 Queue Receiver Block Enhancements 14-68
14-21 Queue Receive Flow 14-71
14-22 Queue Receive Software Flow 14-72
14-23 Queue Receive Example for 17 Data Bytes 14-73

15-1 MIOS1 Block Diagram 15-5
15-2 MIOS1 Memory Map 15-7
15-3 MCPSM Block Diagram 15-12
15-4 MMCSM Block Diagram 15-15
15-5 MDASM Block Diagram 15-19
15-6 MPWMSM Block Diagram 15-25
15-7 MPIOSM One-Bit Block Diagram 15-30
15-8 MIOS Interrupt Structure 15-32
15-9 MIOS1 Example: Double Capture Pulse Width Measurement 15-39
15-10 MIOS1 Example: Double Capture Period Measurement 15-40
15-11 MIOS1 Example: Double Edge Output Compare 15-41
15-12 MIOS1 Example: Pulse Width Modulation Output 15-43

16-1 TouCAN Block Diagram 16-1
16-2 Typical CAN Network 16-3

Figure Number		Page Number

16-3 Extended ID Message Buffer Structure 16-4
16-4 Standard ID Message Buffer Structure 16-4
16-5 Interrupt levels on IRQ with ILBS 16-20
16-6 TouCAN Message Buffer Memory Map 16-22

17-1 TPU3 Block Diagram 17-1
17-2 TPU3 Interrupt Levels 17-5
17-3 TCR1 Prescaler Control 17-7
17-4 TCR2 Prescaler Control 17-8

18-1 DPTRAM Configuration 18-2
18-2 DPTRAM Memory Map 18-3

19-1 CMF Array and Control Register Addressing 19-4
19-2 Shadow Information 19-16
19-3 Program State Diagram 19-20
19-4 Erase State Diagram 19-25
19-5 Pulse Status Timing 19-27
19-6 Censorship States and Transitions 19-35
19-7 EPEE Digital Filter and Latch 19-36
19-8 CMF_EPEE Timing Diagram 19-37
19-9 VPP and VDDL Power Switching 19-38
19-10 VPP Conditioning Circuit 19-39

20-1 SRAM Block Diagram 20-1
20-2 SRAM Memory Map 20-2

21-1 Watchpoints and Breakpoint Support in the CPU 21-10
21-2 Partially Supported Watchpoint/Breakpoint Example 21-15
21-3 Instruction Support General Structure 21-17
21-4 Load/Store Support General Structure 21-20
21-5 Functional Diagram of MPC555 / MPC556 Debug Mode Support 21-23
21-6 Debug Mode Logic 21-25
21-7 Debug Mode Reset Configuration 21-27
21-8 Asynchronous Clock Serial Communications 21-35
21-9 Synchronous Self Clock Serial Communication 21-36
21-10 Enabling Clock Mode Following Reset 21-37
21-11 Download Procedure Code Example 21-42
21-12 Slow Download Procedure Loop 21-42
21-13 Fast Download Procedure Loop 21-42

22-1 JTAG Pins 22-1
22-2 Test Logic Block Diagram 22-2
22-3 TAP Controller State Machine 22-4
22-4 Bypass Register 22-6
22-5 Output Pin Cell (O.pin) 22-8

Figure Number		Page Number

22-6 Observe-Only Input Pin Cell (I.Obs) 22-8
22-7 Output Control Cell (IO.CTL) 22-9
22-8 General Arrangement of Bidirectional Pin Cells 22-9

D-1 TPU3 Memory Map D-1
D-2 PTA Parameters D-5
D-3 QOM Parameters D-7
D-4 TSM Parameters — Master Mode D-9
D-5 TSM Parameters — Slave Mode D-10
D-6 FQM Parameters D-12
D-7 UART Transmitter Parameters D-14
D-8 UART Receiver Parameters D-15
D-9 NITC Parameters D-17
D-10 COMM Parameters (Part 1 of 2) D-19
D-11 COMM Parameters (Part 2 of 2) D-20
D-12 HALLD Parameters D-21
D-13 MCPWM Parameters — Master Mode D-23
D-14 MCPWM Parameters — Slave Edge-Aligned Mode D-24
D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode D-25
D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode D-26
D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode D-27
D-18 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode D-28
D-19 FQD Parameters — Primary Channel D-30
D-20 FQD Parameters — Secondary Channel D-31
D-21 PPWA Parameters D-33
D-22 OC Parameters D-35
D-23 PWM Parameters D-37
D-24 DIO Parameters D-39
D-25 SPWM Parameters, Part 1 of 2 ..... D-41
D-26 SPWM Parameters, Part 2 of 2 ..... D-42
D-27 RWTPIN Parameters D-44
D-28 ID Parameters D-46
D-29 Two Possible SIOP Configurations D-47
D-30 SIOP Parameters D-49
D-31 SIOP Function Data Transition Example D-53

E-1 MPC555 / MPC556 Family Power Distribution Diagram — 3 V E-2
E-2 MPC555 / MPC556 Family Power Distribution Diagram — 5 V and Analog .. E-3
E-3 Crystal Oscillator Circuit E-4
E-4 RC Filter Example E-5
E-5 Bypass Capacitors Example (Alternative) E-6
E-6 RC Filter Example E-6

Figure Number		Page Number

E-7 LC Filter Example (Alternative) E-7
E-8 PLL Off-Chip Capacitor Example E-7

G-1 CLKOUT Timing G-16
G-2 External Clock Timing G-23
G-3 Synchronous Output Signals Timing G-24
G-4 Synchronous Active Pull-Up and Open Drain Outputs Signals Timing G-25
G-5 Synchronous Input Signals Timing G-26
G-6 Input Data Timing in Normal Case G-27
G-7 External Bus Read Timing (GPCM Controlled — ACS = ‘00’) G-28
G-8 External Bus Read Timing (GPCM Controlled — TRLX = ‘0’ ACS = ‘10’) .. G-29
G-9 External Bus Read Timing (GPCM Controlled — TRLX = ‘0’ ACS = ‘11’) .. G-30
G-10 External Bus Read Timing (GPCM Controlled — TRLX = ‘1’, ACS = ‘10’, ACS = ‘11’) G-31
G-11 Address Show Cycle Bus Timing G-32
G-12 Address and Data Show Cycle Bus Timing G-33
G-13 External Bus Write Timing (GPCM Controlled — TRLX = ‘0’, CSNT = ‘0’) . G-34
G-14 External Bus Write Timing (GPCM Controlled — TRLX = ‘0’, CSNT = ‘1’) . G-35
G-15 External Bus Write Timing (GPCM Controlled — TRLX = ‘1’, CSNT = ‘1’) . G-36
G-16 External Master Read from Internal Registers Timing G-37
G-17 External Master Write to Internal Registers Timing G-38
G-18 Interrupt Detection Timing for External Level Sensitive Lines G-39
G-19 Interrupt Detection Timing for External Edge Sensitive Lines G-40
G-20 Debug Port Clock Input Timing G-41
G-21 Debug Port Timings G-42
G-22 Reset Timing — Configuration from Data Bus G-44
G-23 Reset Timing — Data Bus Weak Drive During Configuration G-45
G-24 Reset Timing — Debug Port Configuration G-46
G-25 JTAG Test Clock Input Timing G-48
G-26 JTAG — Test Access Port Timing Diagram G-49
G-27 JTAG — $\overline{\text{TRST}}$ Timing Diagram G-50
G-28 Boundary Scan (JTAG) Timing Diagram G-51
G-29 QSPI Timing — Master, CPHA = 0 ..... G-55
G-30 QSPI Timing — Master, CPHA = 1 ..... G-55
G-31 QSPI Timing — Slave, CPHA = 0 ..... G-56
G-32 QSPI Timing — Slave, CPHA = 1 ..... G-56
G-33 TPU3 Timing G-58
G-34 MCPSM Enable to vs_pclk Pulse Timing Diagram G-59
G-35 MPWMSM Minimum Output Pulse Example Timing Diagram G-60
G-36 MCPSM Enable to MPWMO Output Pin Rising Edge Timing Diagram G-61
G-37 MPWMSM Enable to MPWMO Output Pin Rising Edge Timing Diagram G-61
G-38 MPWMSM Interrupt Flag to MPWMO Output Pin Falling Edge Timing Diagram G-62
G-39 MMCSM Minimum Input Pin (Either Load or Clock)

Figure Number		Page Number

Timing Diagram G-63
G-40 MMCSM Clock Pin to Counter Bus Increment Timing Diagram G-63
G-41 MMCSM Load Pin to Counter Bus Reload Timing Diagram G-63
G-42 MMCSM Counter Bus Reload to Interrupt Flag Setting Timing Diagram G-64
G-43 MMCSM Prescaler Clock Select to Counter Bus Increment Timing Diagram G-64
G-44 MDASM Minimum Input Pin Timing Diagram G-65
G-45 MDASM Input Pin to Counter Bus Capture Timing Diagram G-66
G-46 MDASM Input Pin to MDASM Interrupt Flag Timing Diagram G-66
G-47 MDASM Minimum Output Pulse Width Timing Diagram G-66
G-48 Counter Bus to MDASM Output Pin Change Timing Diagram G-67
G-49 Counter Bus to MDASM Interrupt Flag Setting Timing Diagram G-67
G-50 MPIOSM Input Pin to MPIOSM_DR (Data Register) Timing Diagram G-68

H-1 Typical Program Time vs. V_{PP} and Temperature (for CDR1 “Target” Process) H-2

Figure Number	Page Number

LIST OF TABLES

Table Number | Page Number

2-1 MPC555 / MPC556 Pin Functions for 272-Pin PBGA 2-4
2-2 Pin Functionality Table 2-7
2-3 PDMCR Bit Descriptions..... 2-29
2-4 Pin Reset State..... 2-32
2-5 Pad Groups Based on 3-V / 5-V Select 2-57
2-6 Pin Names and Abbreviations 2-58

3-1 RCPU Execution Units..... 3-5
3-2 Supervisor-Level SPRs..... 3-9
3-3 Development Support SPRs..... 3-11
3-4 FPSCR Bit Categories 3-13
3-5 FPSCR Bit Descriptions..... 3-14
3-6 Floating-Point Result Flags in FPSCR..... 3-15
3-7 Bit Descriptions for CR0 Field of CR 3-16
3-8 Bit Descriptions for CR1 Field of CR 3-17
3-9 CR*n* Field Bit Descriptions for Compare Instructions 3-17
3-10 Integer Exception Register Bit Definitions 3-18
3-11 Time Base Field Definitions (Read Only)..... 3-19
3-12 Machine State Register Bit Descriptions 3-21
3-13 Floating-Point Exception Mode Bits..... 3-22
3-14 Time Base Field Definitions (Write Only)..... 3-23
3-15 Uses of SPRG0–SPRG3 3-25
3-16 Processor Version Register Bit Descriptions 3-26
3-17 EIE, EID, AND NRI Registers..... 3-26
3-18 FPECR Bit Descriptions..... 3-27
3-19 Instruction Set Summary 3-29
3-20 MPC555 / MPC556 Exception Classes 3-34
3-21 Exception Vector Offset Table 3-36
3-22 Instruction Latency and Blockage..... 3-38
3-23 Floating-Point Exception Mode Encoding..... 3-43

4-1 Exception Addresses Mapping by BBC 4-18
4-2 Region Base Address Registers RBA[0:1] 4-20
4-3 Region Attributes Registers 4-20
4-4 BBC Module Configuration Register..... 4-20
4-5 MI_RBA[0:3] Bit Descriptions 4-21
4-6 MI_RA[0:3] Registers Bits Description..... 4-22
4-7 MI_GRA Bit Descriptions..... 4-23
4-8 BBCMCR Bit Descriptions 4-24

5-1 USIU Address Map 5-3
5-2 USIU Special-Purpose Registers..... 5-6
5-3 PowerPC Address Range..... 5-6

Table Number		Page Number

6-1 USIU Pins Multiplexing Control 6-3
6-2 SGPIO Configuration 6-7
6-3 Priority of Interrupt Sources 6-12
6-4 Decrementer Time-Out Periods 6-13
6-5 SIUMCR Bit Descriptions 6-19
6-6 Debug Pins Configuration 6-20
6-7 Debug Port Pins Configuration 6-20
6-8 General Pins Configuration 6-20
6-9 Single-Chip Select Field Pin Configuration 6-20
6-10 Multi-Level Reservation Control Pin Configuration 6-21
6-11 IMMR Bit Descriptions 6-22
6-12 EMCR Bit Descriptions 6-23
6-13 SYPCR Bit Descriptions 6-26
6-14 SWSR Bit Descriptions 6-27
6-15 TESR Bit Descriptions 6-28
6-16 TBSCR Bit Descriptions 6-30
6-17 RTCSC Bit Descriptions 6-31
6-18 PISCR Bit Descriptions 6-32
6-19 PITC Bit Descriptions 6-33
6-20 PIT Bit Descriptions 6-33
6-21 SGPIODT1 Bit Descriptions 6-34
6-22 SGPIODT2 Bit Descriptions 6-35
6-23 SGPIOCR Bit Descriptions 6-35
6-24 Data Direction Control 6-36

7-1 Reset Action Taken For Each Reset Cause 7-4
7-2 Reset Configuration Word and Data Corruption/Coherency 7-4
7-3 Reset Status Register Bit Descriptions 7-5
7-4 Reset Configuration Options 7-7
7-5 Hard Reset Configuration Word Bit Descriptions 7-11

8-1 Reset Clocks Source Configuration 8-9
8-2 TMBCLK Divisions 8-9
8-3 Status of Clock Source 8-15
8-4 Power Mode Control Bit Descriptions 8-16
8-5 Power Mode Descriptions 8-16
8-6 Power Mode Wake-Up Operation 8-17
8-7 Clock Unit Power Supply 8-20
8-8 KAPWR Registers and Key Registers 8-24
8-9 SCCR Bit Descriptions 8-30
8-10 PLPRCR Bit Descriptions 8-34
8-11 COLIR Bit Descriptions 8-36
8-12 VSRMCR Bit Descriptions 8-36

9-1 MPC555 / MPC556 SIU Signals 9-4
9-2 Data Bus Requirements For Read Cycles 9-30

Table Number		Page Number

9-3 Data Bus Contents for Write Cycles 9-30
9-4 Priority Between Internal and External Masters over External Bus 9-34
9-5 Burst Length and Order 9-36
9-6 BURST/TSIZE Encoding 9-37
9-7 Address Type Pins 9-37
9-8 Address Types Definition 9-38
9-9 Termination Signals Protocol 9-47

10-1 Timing Attributes Summary 10-6
10-2 Programming Rules for Strobes Timing 10-19
10-3 Boot Bank Fields Values After Hard Reset 10-20
10-4 Write Enable/Byte Enable Signals Function 10-21
10-5 Memory Controller Address Map 10-27
10-6 MSTAT Bit Descriptions 10-28
10-7 BR0 – BR3 Bit Descriptions 10-29
10-8 OR0 – OR3 Bit Descriptions 10-30
10-9 DMBR Bit Descriptions 10-32
10-10 DMOR Bit Descriptions 10-33

11-1 DMPU Registers 11-6
11-2 Reservation Snoop Support 11-9
11-3 L2U_MCR LSHOW Modes 11-9
11-4 L2U Show Cycle Support Chart 11-12
11-5 L2U (PPC) Register Decode 11-12
11-6 Hex Address For SPR Cycles 11-13
11-7 L2U_MCR Bit Descriptions 11-14
11-8 L2U_RBAx Bit Descriptions 11-14
11-9 L2U_RAx Bit Descriptions 11-15
11-10 L2U_GRA Bit Descriptions 11-16

12-1 STOP and HSPEED Bit Functionality 12-2
12-2 Bus Cycles and System Clock Cycles 12-3
12-3 ILBS Signal functionality 12-5
12-4 IRQMUX Functionality 12-5
12-5 UIMB Interface Register Map 12-7
12-6 UMCR Bit Descriptions 12-8
12-7 UIPEND Bit Descriptions 12-9

13-1 Multiplexed Analog Input Channels 13-5
13-2 Analog Input Channels 13-11
13-3 Queue 1 Priority Assertion 13-15
13-4 QADC64 Clock Programmability 13-28
13-5 QADC64 Status Flags and Interrupt Sources 13-30
13-6 QADC64 Address Map 13-32
13-7 QADC64MCR Bit Descriptions 13-33
13-8 QADC64INT Bit Descriptions 13-34

Table Number		Page Number

13-9 PORTQA, PORTQB Bit Descriptions 13-34
13-10 DDRQA Bit Descriptions 13-35
13-11 QACR0 Bit Descriptions 13-36
13-12 QACR1 Bit Descriptions 13-37
13-13 Queue 1 Operating Modes 13-38
13-14 QACR2 Bit Descriptions 13-39
13-15 Queue 2 Operating Modes 13-40
13-16 QASR0 Bit Descriptions 13-41
13-17 Queue Status 13-42
13-18 QASR0 Bit Descriptions 13-43
13-19 CCW Bit Descriptions 13-47
13-20 Non-Multiplexed Channel Assignments and Pin Designations 13-48
13-21 Multiplexed Channel Assignments and Pin Designations 13-48

14-1 QSMCM Register Map 14-3
14-2 QSMCM Global Registers 14-5
14-3 Interrupt Levels 14-6
14-4 QSMCMMCR Bit Descriptions 14-8
14-5 QDSCI_IL Bit Descriptions 14-8
14-6 QSPI_IL Bit Descriptions 14-9
14-7 QSMCM Pin Control Registers 14-9
14-8 Effect of DDRQS on QSPI Pin Function 14-10
14-9 QSMCM Pin Functions 14-11
14-10 PQSPAR Bit Descriptions 14-12
14-11 DDRQS Bit Descriptions 14-13
14-12 QSPI Register Map 14-16
14-13 SPCR0 Bit Descriptions 14-17
14-14 Bits Per Transfer 14-17
14-15 SPCR1 Bit Descriptions 14-18
14-16 SPCR2 Bit Descriptions 14-19
14-17 SPCR3 Bit Descriptions 14-20
14-18 SPSR Bit Descriptions 14-21
14-19 Command RAM Bit Descriptions 14-23
14-20 QSPI Pin Functions 14-24
14-21 Example SCK Frequencies with a 40-MHz IMB Clock 14-35
14-22 SCI Registers 14-44
14-23 SCCxR0 Bit Descriptions 14-45
14-24 SCCxR1 Bit Descriptions 14-46
14-25 SCxSR Bit Descriptions 14-48
14-26 SCxSR Bit Descriptions 14-50
14-27 SCI Pin Functions 14-50
14-28 Serial Frame Formats 14-51
14-29 Examples of SCIx Baud Rates 14-52
14-30 QSCI1CR Bit Descriptions 14-60
14-31 QSCI1SR Bit Descriptions 14-61
15-1 MIOS1 I/O Ports 15-8

Table Number		Page Number

15-2 MBISM Address Map 15-8
15-3 MIOS1TPCR Bit Descriptions 15-9
15-4 MIOS1VNR Bit Descriptions 15-9
15-5 MIOS1MCR Bit Descriptions 15-10
15-6 MBISM Interrupt Registers Address Map 15-10
15-7 MIOS1LVL0 Bit Descriptions 15-11
15-8 MIOS1LVL1 Bit Descriptions 15-11
15-9 MCPSM Address Map 15-13
15-10 MCPSMSCR Bit Descriptions 15-13
15-11 MMCSM Address Map 15-15
15-12 MMCSMCNT Bit Descriptions 15-16
15-13 MMCSMML Bit Descriptions 15-16
15-14 MMCSMSCR Bit Descriptions 15-17
15-15 MMCSMCR CP and MPWMSMSCR CP Values 15-18
15-16 MDASM Address Map 15-20
15-17 MDASMSCR Bit Descriptions 15-23
15-18 MDASM Mode Selects 15-24
15-19 MPWMSM Address Map 15-26
15-20 MPWMSMPERR Bit Descriptions 15-27
15-21 MPWMSMPULR Bit Descriptions 15-27
15-22 MPWMSMCNTR Bit Descriptions 15-28
15-23 MPWMSMSCR Bit Descriptions 15-29
15-24 PWMSM Output Pin Polarity Selection 15-29
15-25 MPIOSM Address Map 15-30
15-26 MPIOSMDR Bit Descriptions 15-31
15-27 MPIOSMDDR Bit Descriptions 15-31
15-28 MIRSM0 Address Map 15-34
15-29 MIOS1SR0 Bit Descriptions 15-34
15-30 MIOS1ER0 Bit Descriptions 15-35
15-31 MIOS1RPR0 Bit Descriptions 15-36
15-32 MIRSM1 Address Map 15-36
15-33 MIOS1SR1 Bit Descriptions 15-37
15-34 MIOS1ER1 Bit Descriptions 15-37
15-35 MIOS1RPR1 Bit Descriptions 15-38
15-36 MIOS1 Configuration 15-44

16-1 Common Extended/Standard Format Frames 16-5
16-2 Message Buffer Codes for Receive Buffers 16-5
16-3 Message Buffer Codes for Transmit Buffers 16-5
16-4 Extended Format Frames 16-6
16-5 Standard Format Frames 16-6
16-6 Receive Mask Register Bit Values 16-8
16-7 Mask Examples for Normal/Extended Messages 16-8
16-8 Example IMB Clock, CAN Bit Rate and S-Clock Frequencies 16-9
16-9 Interrupt Levels 16-19

Table Number		Page Number

16-10 TouCAN Register Map 16-21
16-11 TCNMCR Bit Descriptions 16-23
16-12 CANICR Bit Descriptions 16-25
16-13 CANCTRL0 Bit Descriptions 16-25
16-14 RX MODE[1:0] Configuration 16-26
16-15 Transmit Pin Configuration 16-26
16-16 CANCTRL1 Bit Descriptions 16-27
16-17 PRESDIV Bit Descriptions 16-28
16-18 CANCTRL2 Bit Descriptions 16-28
16-19 TIMER Bit Descriptions 16-29
16-20 RXGMSKHI, RXGMSKLO Bit Descriptions 16-30
16-21 ESTAT Bit Descriptions 16-31
16-22 Transmit Bit Error Status 16-32
16-23 Fault Confinement State Encoding 16-32
16-24 IMASK Bit Descriptions 16-32
16-25 IFLAG Bit Descriptions 16-33
16-26 RXECTR, TXECTR Bit Descriptions 16-33

17-1 Enhanced TCR1 Prescaler Divide Values 17-6
17-2 TCR1 Prescaler Values 17-6
17-3 TCR2 Counter Clock Source 17-7
17-4 TCR2 Prescaler Control 17-8
17-5 TPU3 Register Map 17-9
17-6 TPUMCR Bit Descriptions 17-11
17-7 DSCR Bit Descriptions 17-13
17-8 DSSR Bit Descriptions 17-14
17-9 TICR Bit Descriptions 17-15
17-10 CIER Bit Descriptions 17-15
17-11 CFSRx Bit Descriptions 17-16
17-12 HSQRx Bit Descriptions 17-17
17-13 HSSRx Bit Descriptions 17-18
17-14 CPRx Bit Descriptions 17-18
17-15 Channel Priorities 17-19
17-16 CISR Bit Descriptions 17-19
17-17 TPUMCR2 Bit Descriptions 17-20
17-18 Entry Table Bank Location 17-21
17-19 IMB Clock Frequency/Minimum Guaranteed Detected Pulse 17-21
17-20 TPUMCR3 Bit Descriptions 17-21
17-21 Parameter RAM Address Offset Map 17-22

18-1 DPTRAM Register Map 18-3
18-2 DPTMCR Bit Descriptions 18-4
18-3 RAMBAR Bit Descriptions 18-5

19-1 CMF Register Programmer's Model 19-5
19-2 CMFMCR Bit Descriptions 19-6

Table Number		Page Number

19-3 CMFTST Bit Descriptions 19-8
19-4 CMF Programming Algorithm (v6 and Later) 19-8
19-5 CMF Erase Algorithm (v6) 19-9
19-6 CMFCTL Bit Descriptions 19-10
19-7 EEPROM Array Addressing 19-12
19-8 CMF EEPROM Array Address Fields 19-12
19-9 Program Interlock State Descriptions 19-21
19-10 Results of Programming Margin Read 19-22
19-11 Erase Interlock State Descriptions 19-26
19-12 System Clock Range 19-28
19-13 Clock Period Exponent and Pulse Width Range 19-29
19-14 Censorship Control Bits 19-31
19-15 Levels of Censorship 19-32
19-16 CMF EEPROM Devices Modes and Censorship Status 19-33
19-17 NVM Fuse States 19-34

20-1 SRAMMCR Bit Descriptions 20-3

21-1 VF Pins Instruction Encodings 21-3
21-2 VF Pins Queue Flush Encodings 21-4
21-3 VFLS Pin Encodings 21-4
21-4 Detecting the Trace Buffer Start Point 21-7
21-5 Fetch Show Cycles Control 21-8
21-6 Instruction Watchpoints Programming Options 21-17
21-7 Load/Store Data Events 21-18
21-8 Load/Store Watchpoints Programming Options 21-19
21-9 The Check Stop State and Debug Mode 21-29
21-10 Trap Enable Data Shifted into Development Port Shift Register 21-38
21-11 Debug Port Command Shifted Into Development Port Shift Register 21-38
21-12 Status / Data Shifted Out of Development Port Shift Register 21-39
21-13 Debug Instructions / Data Shifted Into Development Port Shift Register 21-40
21-14 Development Support Programming Model 21-44
21-15 Development Support Registers Read Access Protection 21-45
21-16 Development Support Registers Write Access Protection 21-45
21-17 CMPA-CMPD Bit Descriptions 21-45
21-18 CMPE-CMPF Bit Descriptions 21-46
21-19 BAR Bit Descriptions 21-46
21-20 CMPG-CMPH Bit Descriptions 21-46
21-21 ICTRL Bit Descriptions 21-48
21-22 ISCT_SER Bit Descriptions 21-49
21-23 LCTRL1 Bit Descriptions 21-50
21-24 LCTRL2 Bit Descriptions 21-51
21-25 Breakpoint Counter A Value and Control Register (COUNTA) 21-52
21-26 Breakpoint Counter B Value and Control Register (COUNTB) 21-53
21-27 ECR Bit Descriptions 21-54
21-28 DER Bit Descriptions 21-55

Table Number		Page Number

22-1 JTAG Interface Pin Descriptions 22-3
22-2 Instruction Decoding 22-5
22-3 Boundary Scan Bit Definition 22-10

A-1 SPR (Special Purpose Registers) A-2
A-2 CMF (CDR MoneT Flash EEPROM) Flash Array A-4
A-3 USIU (Unified System Interface Unit) A-5
A-4 CMF (CDR MoneT Flash EEPROM) A-8
A-5 DPTRAM (Dual-Port TPU RAM) A-9
A-6 DPTRAM Array A-9
A-7 TPU3 (Time Processor Unit) A-10
A-8 QADC64 (Queued Analog-to-Digital Converter) A-13
A-9 QSMCM (Queued Serial Multi-Channel Module) A-15
A-10 MIOS1 (Modular Input/Output Subsystem) A-16
A-11 TouCAN (CAN 2.0B Controller) A-22
A-12 UIMB (U-Bus to IMB3 Bus Interface) A-25
A-13 SRAM (Static RAM Access Memory) A-25
A-14 SRAM (Static RAM Access Memory) Array A-25

D-1 Bank 0 Functions D-2
D-2 Bank 1 Functions D-3
D-3 QOM Bit Encoding D-6
D-4 SIOP Function Valid CHAN_Control Options D-50
D-5 SIOP State Timing D-52

E-1 External Components Value For Different Crystals (Q1) E-4

F-1 Memory Access Times Using Different Buses F-1
F-2 Timing Examples F-2

G-1 Absolute Maximum Ratings G-1
G-2 Thermal Characteristics G-3
G-3 ESD Protection G-6
G-4 DC Electrical Characteristics G-7
G-5 Oscillator and PLL G-12
G-6 Program and Erase Characteristics G-13
G-7 CMF AC and DC Power Supply Characteristics G-14
G-8 Flash Module Life G-14
G-9 CMF Programming Algorithm (v6 and Later) G-15
G-10 CMF Erase Algorithm (v6) G-15
G-11 Bus Operation Timing G-17
G-12 Interrupt Timing G-39
G-13 Debug Port timing G-40
G-14 RESET Timing G-43
G-15 JTAG Timing G-47
G-16 QADC64 Conversion Characteristics (Operating) G-52

Table Number		Page Number

G-17 QSPI Timing G-53
G-18 GPIO Timing G-57
G-19 TPU3 Timing G-57
G-20 TouCAN Timing G-58
G-21 MCPSM Timing Characteristics G-59
G-22 MPWMSM Timing Characteristics G-60
G-23 MMCSM Timing Characteristics G-62
G-24 MDASM Timing Characteristics G-65
G-25 MPIOSM Timing Characteristics G-68

H-1 Program and Erase Characteristics H-1
H-2 CMF AC and DC Power Supply Characteristics H-2
H-3 Flash Module Life H-3
H-4 CMF Programming Algorithm (v5) H-3
H-5 CMF Erase Algorithm (v5) H-3

Table Number	Page Number

PREFACE

This manual defines the functionality of the MPC555 / MPC556 for use by software and hardware developers. The MPC555 / MPC556 is based on the PowerPC processor used in the Motorola MPC500 family of microcontrollers. For further information refer to the ***MPC500 Family RCPU Reference Manual*, RCPURM/AD** (Motorola order number).

Boxed sections appear throughout this manual. These boxes designate optional features that are only available on the MPC556.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products for the MPC555 / MPC556. It is assumed that the reader understands operating systems, microprocessor and microcontroller system design, and the basic principles of RISC processing.

Additional Reading

For additional reading that provides background to or supplements the information in this manual see:

- John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc., San Mateo, CA
- ***PowerPC Microprocessor Family: the Programming Environments***, MPCFPE/AD (Motorola order number)
- ***MPC500 Family RCPU Reference Manual***, RCPURM/AD (Motorola order number)

Conventions

This document uses the following notational conventions:

ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are high and negated when they are low.
ACTIVE_LOW	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted (active) when they are low and negated when they are high.
0x0F	Hexadecimal numbers
0b0011	Binary numbers
REG[FIELD]	Abbreviations or acronyms for registers are shown in uppercase text. Specific bit fields or ranges are shown in brackets.

x In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary encoded 0bx001, the state of the first bit is a don't care.

NOTE:

Throughout this manual references to 3 V refer to the nominal supply voltage of 3.3 volts.

Nomenclature

Logic level one is the voltage that corresponds to Boolean true (1) state.

Logic level zero is the voltage that corresponds to Boolean false (0) state.

To **set** a bit or bits means to establish logic level one on the bit or bits.

To **clear** a bit or bits means to establish logic level zero on the bit or bits.

A signal that is **asserted** is in its active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

A signal that is **negated** is in its inactive logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

SECTION 1
OVERVIEW

The MPC555 / MPC556 is a member of Motorola's MPC500 PowerPC™ RISC Microcontroller family. The MPC555 / MPC556 offers the following features:

- PowerPC core with floating-point unit
- 26 Kbytes fast RAM and 6 Kbytes TPU microcode RAM
- 448 Kbytes flash EEPROM with 5-V programming
- 5-V I/O system
- Serial system: queued serial multi-channel module (QSMCM), dual CAN 2.0B controller modules (TouCAN™)
- 50-channel timer system: dual time processor units (TPU3), modular I/O system (MIOS1)
- 32 analog inputs: dual queued analog-to-digital converters (QADC64)
- Submicron HCMOS (CDR1) technology
- 272-pin plastic ball grid array (PBGA) packaging
- 40-MHz operation, -40° C to 125° C with dual supply (3.3 V, 5 V)
- MPC556 supports code compression to increase code density.

NOTE

Throughout this manual references to 3 V refer to the nominal supply voltage of 3.3 V.)

1.1 Block Diagram

Figure 1-1 is a block diagram of the MPC555 / MPC556.

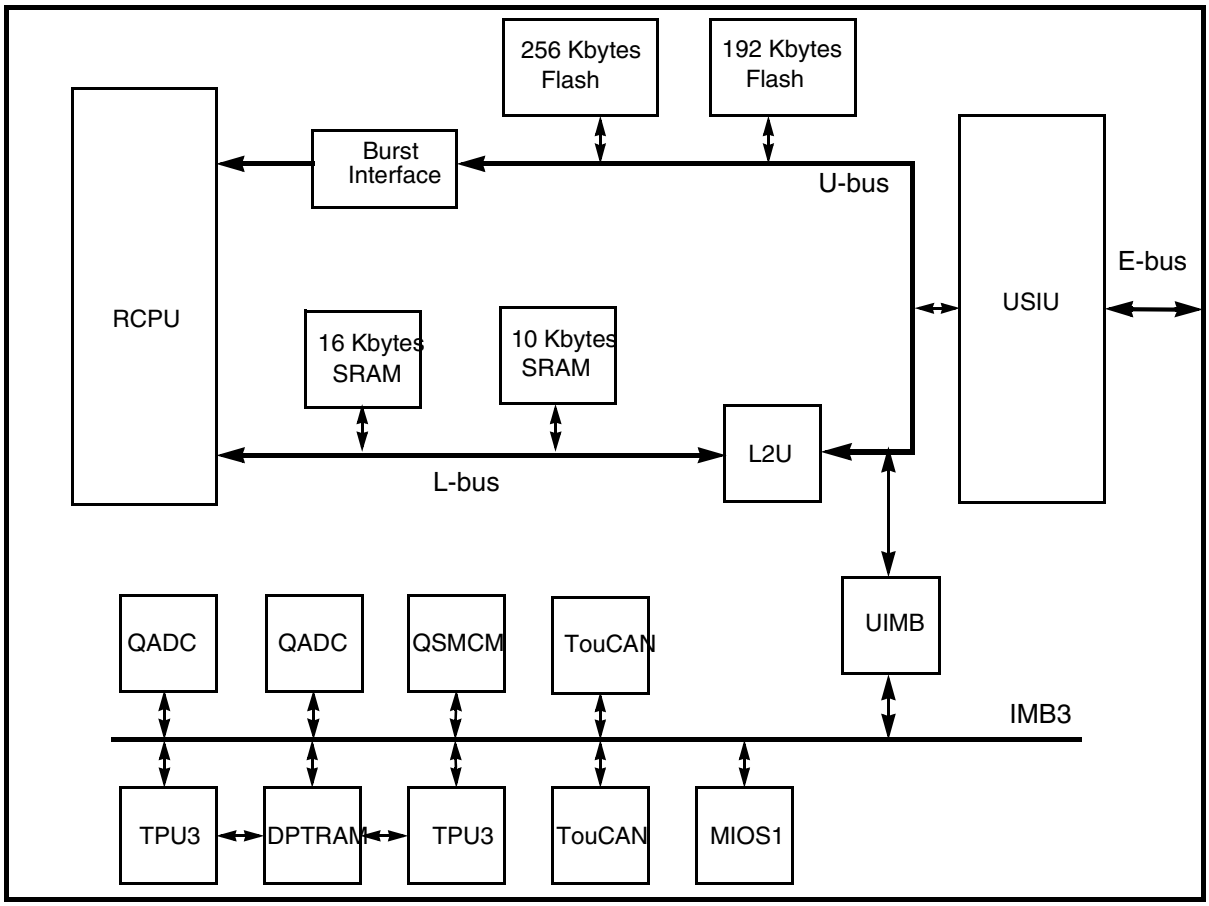


Figure 1-1 MPC555 / MPC556 Block Diagram

1.2 MPC555 / MPC556 Features

Features of each module on the MPC555 / MPC556 are listed below.

1.2.1 RISC MCU Central Processing Unit (RCPU)

- 32-bit PowerPC architecture (compliant with PowerPC Architecture Book 1)
- Core performance measured at 52.7 Kmips (Dhrystone 2.1) @ 40 MHz.

NOTE

This assumes the RCPU core is running in “normal” mode and show cycles is turned off (ISCT_SER of the ICTRL register is set to 111). See **Table 21-21**.

- Fully static, low power operation
- Integrated floating-point unit
- Precise exception model
- Extensive system development support
  — On-chip watchpoints and breakpoints
  — Program flow tracking

— On-chip emulation (OnCE™) development interface

1.2.2 Four-Bank Memory Controller

- Works with SRAM, EPROM, flash EEPROM, and other peripherals
- Byte write enables
- 32-bit address decodes with bit masks
- Memory transfer start (MTS): This pin is the transfer start signal to access a slave's external memory by an external bus master

1.2.3 U-Bus System Interface Unit (USIU)

- Clock synthesizer
- Power management
- Reset controller
- PowerPC decrementer and time base
- Glueless interface to SRAMs and burstable FLASHs
- Real-time clock register
- Periodic interrupt timer
- Hardware bus monitor and software watchdog timer
- Interrupt controller that supports up to eight external and eight internal interrupts
- IEEE 1149.1 JTAG test access port
- External bus interface
  — 24 address pins, 32 data pins
  — Supports multiple master designs
  — Four-beat transfer bursts, two-clock minimum bus transactions
  — Tolerates 5-V inputs, provides 3.3-V outputs

1.2.4 Flexible Memory Protection Unit

- Four instruction regions and four data regions
- 4-Kbyte to 16-Mbyte region size support
- Default attributes available in one global entry
- Attribute support for speculative accesses

1.2.5 448 Kbytes of CDR MoneT Flash EEPROM Memory (CMF)

- One 256-Kbyte and one 192-Kbyte module
- Page read mode
- Block (32-Kbyte) erasable
- External 4.75-V to 5.25-V program and erase power supply

1.2.6 26 Kbytes of Static RAM

- One 16-Kbyte and one 10-Kbyte module
- Fast (one-clock) access
- Keep-alive power
- Soft defect detection (SDD)

1.2.7 General-Purpose I/O Support

- Address (24) and data (32) pins can be used for general-purpose I/O in single-chip mode
- 9 general-purpose I/O pins in MIOS1 unit
- Many peripheral pins can be used for general-purpose I/O when not used for primary function
- 5-V outputs

1.2.8 Two Time Processor Units (TPU3)

- Each TPU3 module provides these features:
  - — A dedicated micro-engine operates independently of the RCPU
  - — 16 independent programmable channels and pins
  - — Each channel has an event register consisting of a 16-bit capture register, a 16-bit compare register and a 16-bit comparator
  - — Nine pre-programmed timer functions are available
  - — Any channel can perform any time function
  - — Each timer function can be assigned to more than one channel
  - — Two timer count registers with programmable prescalers
  - — Each channel can be synchronized to one or both counters
  - — Selectable channel priority levels
  - — 5-V outputs
- 6-Kbyte dual port TPU RAM (DPTRAM) is shared by the two TPU3 modules for TPU microcode

1.2.9 18-Channel Modular I/O System (MIOS1)

- Ten double action submodules (DASMs)
- Eight dedicated PWM sub-modules (PWMSMs)
- Two 16-bit modulus counter submodules (MCSMs)
- Two parallel port I/O submodules (PIOSM)
- 5-V outputs

1.2.10 Two Queued Analog-to-Digital Converter Modules (QADC)

Each QADC provides:

- Up to 16 analog input channels, using internal multiplexing
- Up to 41 total input channels, using internal and external multiplexing
- 10-bit A/D converter with internal sample/hold
- Typical conversion time of 10 µsec (100,000 samples per second)
- Two conversion command queues of variable length
- Automated queue modes initiated by:
  - — External edge trigger/level gate
  - — Software command
- 64 result registers
- Output data that is right- or left-justified, signed or unsigned

1.2.11 Two CAN 2.0B Controller Modules (TouCANs)

Each TouCAN provides these features:

- Full implementation of CAN protocol specification, version 2.0 A and B
- Each module has 16 receive/transmit message buffers of 0 to 8 bytes data length
- Global mask register for message buffers 0 to 13
- Independent mask registers for message buffers 14 and 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- 16-bit free-running timer for message time-stamping
- Low power sleep mode with programmable wake-up on bus activity
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity

1.2.12 Queued Serial Multi-Channel Module (QSMCM)

- Queued serial peripheral interface (QSPI)
  - — Provides full-duplex communication port for peripheral expansion or interprocessor communication
  - — Up to 32 preprogrammed transfers, reducing overhead
  - — Has 160-byte queue
  - — Programmable transfer length: from eight to 16 bits, inclusive
  - — Synchronous interface with baud rate of up to system clock / 4
  - — Four programmable peripheral-select pins support up to 16 devices
    - Wrap-around mode allows continuous sampling of a serial peripheral for efficient interfacing to serial A/D converters
- Two serial communications interfaces (SCI). Each SCI offers these features:
  - — UART mode provides NRZ format and half- or full-duplex interface
  - — 16 register receive buffer and 16 register transmit buffer (SCI1)
  - — Advanced error detection and optional parity generation and detection
  - — Word length programmable as eight or nine bits
  - — Separate transmitter and receiver enable bits and double buffering of data
  - — Wakeup functions allow the CPU to run uninterrupted until either a true idle line is detected or a new address byte is received
  - — External source clock for baud generation
  - — Multiplexing of transmit data pins with discrete outputs and receive data pins with discrete inputs

1.3 MPC555 / MPC556 Address Map

The internal memory map is organized as a single 4-Mbyte block. The user can assign this block to one of eight locations by programming a register in the USIU. The eight possible locations are the first eight 4-Mbyte memory blocks starting with address

0x0000 0000. (Refer to **Figure 1-2**). The programmability of the internal memory map location allows the user to implement a multiple-chip system.

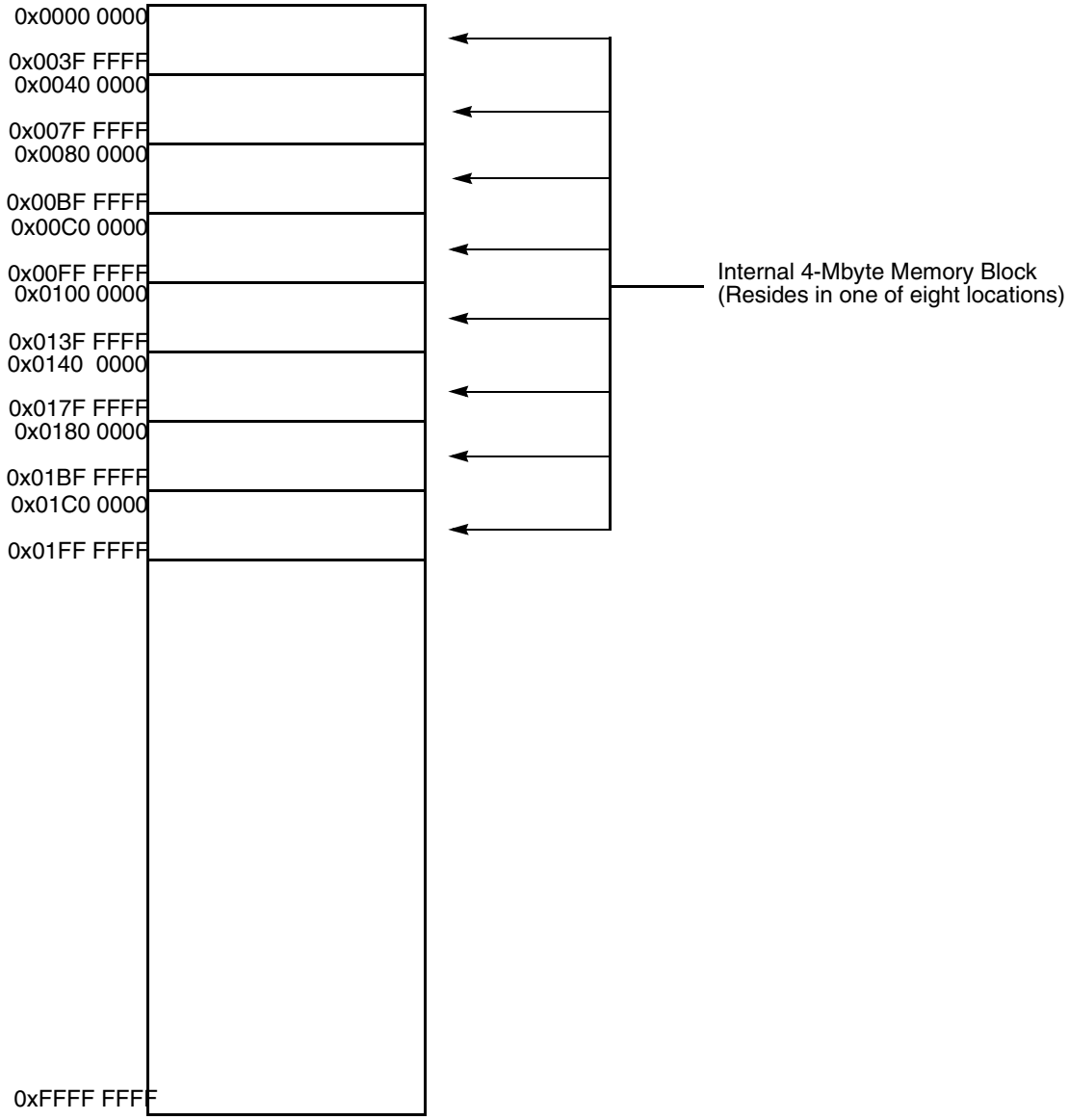


Figure 1-2 MPC555 / MPC556 Memory Map

The internal memory space is divided into the following sections:

- Flash memory (448 Kbytes)
- Static RAM memory (26 Kbytes)
- Control registers and IMB2 modules (64 Kbytes):
  — USIU and flash control registers
  — UIMB interface and IMB2 modules
  — SRAM control registers

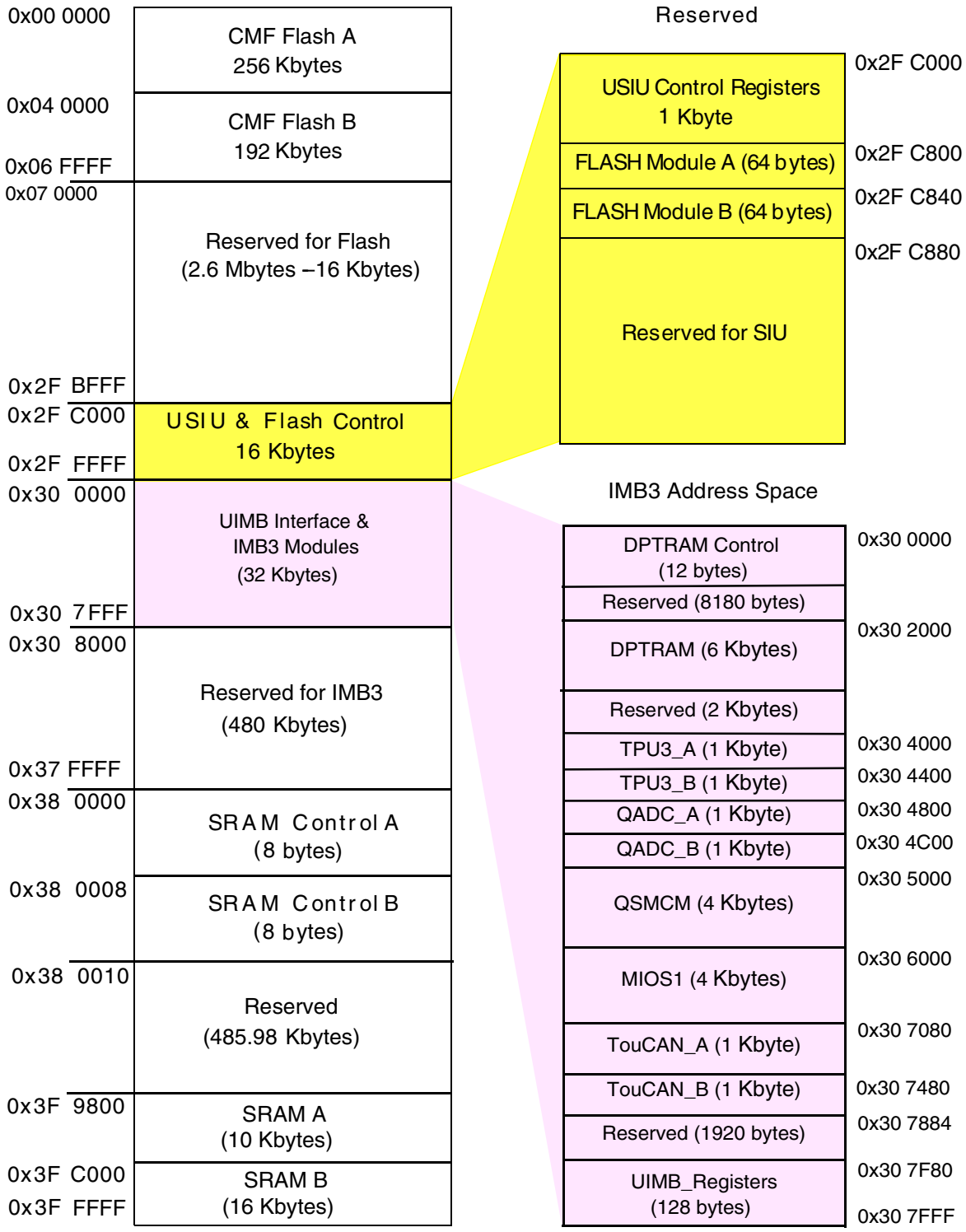
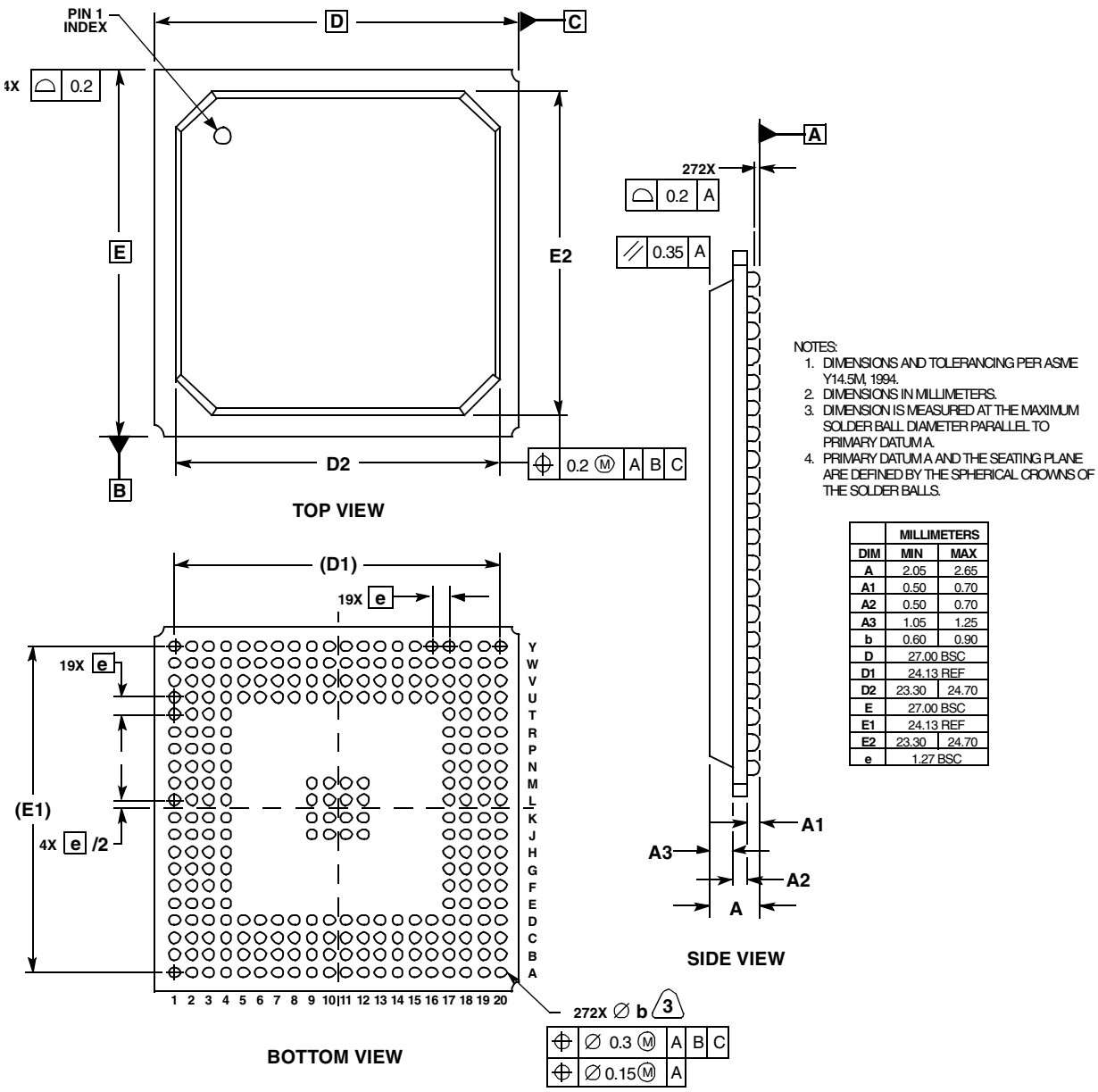


Figure 1-3 MPC555 / MPC556 Internal Memory Map

SECTION 2
SIGNAL DESCRIPTIONS

2.1 Packaging and Pinout Descriptions

Figure 2-1 gives the case configuration and packaging information for the MPC555 / MPC556. **Figure 2-2** gives the MPC555 / MPC556 pinout data. **Table 2-1** gives an overview of the pins on the MPC555 / MPC556.



	MILLIMETERS	
DIM	MIN	MAX
A	2.05	2.65
A1	0.50	0.70
A2	0.50	0.70
A3	1.05	1.25
b	0.60	0.90
D	27.00 BSC	
D1	24.13 REF	
D2	23.30	24.70
E	27.00 BSC	
E1	24.13 REF	
E2	23.30	24.70
e	1.27 BSC	

Figure 2-1 MPC555 / MPC556 Case Dimensions and Packaging

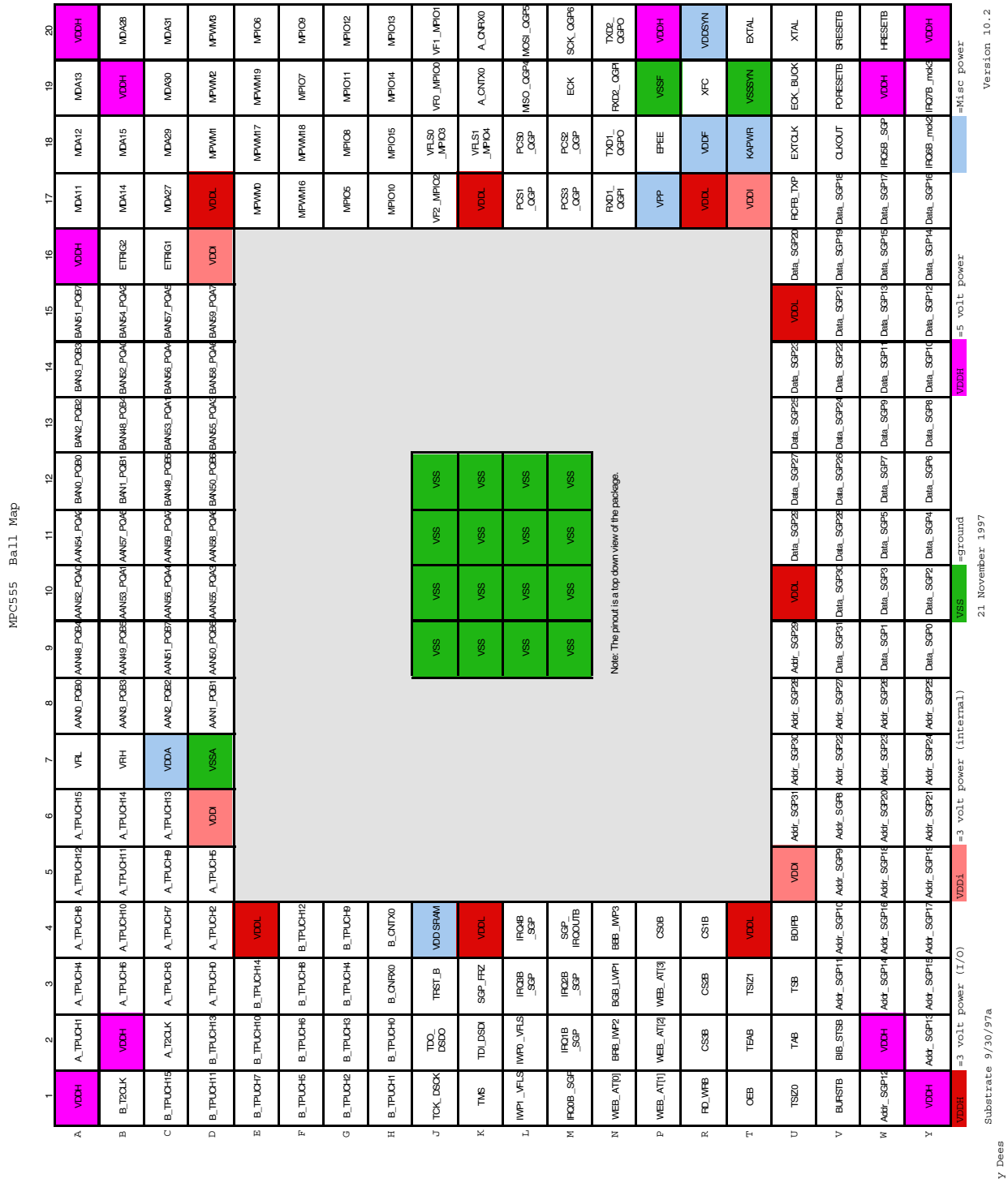


Figure 2-2 MPC555 / MPC556 Pinout Data

Table 2-1 MPC555 / MPC556 Pin Functions for 272-Pin PBGA

Functional Group	Signals[1]	Pins	3 V / 5 V[2]
24 Address lines (16-Mbyte address space)	ADDR[8:31]/SGPIOA[8:31]	24	3-V / 5-V GPIO
32-bit data bus	DATA[0:31]/SGPIOD[0:31]	32	
External interrupts	$\overline{\text{IRQ}}$[0]/SGPIOC[0]	8	3-V / 5-V GPIO
	$\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1]		
	$\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$		
	$\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC[3]		
	$\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4]		
	$\overline{\text{IRQ}}$[5]/SGPIOC[5]/MODCK[1][3]		
	$\overline{\text{IRQ}}$[6:7]/MODCK[2:3][3]		
Bus control	TSIZ[0:1]	11	3 V
	RD/$\overline{\text{WR}}$		
	$\overline{\text{BURST}}$		
	$\overline{\text{BDIP}}$		
	$\overline{\text{TS}}$		
	$\overline{\text{TA}}$		
	$\overline{\text{TEA}}$		
	$\overline{\text{RSTCONF}}$/TEXP[3]		
	$\overline{\text{OE}}$		
	$\overline{\text{BI}}$/$\overline{\text{STS}}$		
General purpose chip select machine (multiplexed with development and debug support)	$\overline{\text{CS}}$[0:3]	8	3 V
	$\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3]/AT[0:3]		
Power-on reset and reset configuration	$\overline{\text{PORESET}}$[3]	3	3 V
	$\overline{\text{HRESET}}$[3]		
	$\overline{\text{SRESET}}$[3]		
Development and debug support	SGPIOC[6]/FRZ/$\overline{\text{PTR}}$	5	3-V / 5-V GPIO
	SGPIOC[7]/$\overline{\text{IRQOUT}}$/LWP[0]		
	$\overline{\text{BG}}$/VF[0]/LWP[1]		
	$\overline{\text{BR}}$/VF[1]/IWP[2]		
	$\overline{\text{BB}}$/VF[2]/IWP[3]		
JTAG and debug port	TMS	7	3 V
	TDI/DSDI		
	TCK/DSCK		
	TDO/DSDO		
	$\overline{\text{TRST}}$		
	IWP[0:1]/VFLS[0:1]		

Table 2-1 MPC555 / MPC556 Pin Functions for 272-Pin PBGA (Continued)

Functional Group	Signals[1]	Pins	3 V / 5 V[2]
Clocks and PLL	XTAL[3]	5	3 V
	EXTAL[3]		
	CLKOUT		
	EXTCLK[3]		
	XFC		
	ENGCLK/BUCLK	1	5 V
QSMCM	PCS0/ $\overline{SS}$/QGPIO[0]	12	5 V
	PCS[1:3]/QGPIO[1:3]		
	MISO/QGPIO[4]		
	MOSI/QGPIO[5]		
	SCK/QGPIO[6]		
	TXD[1:2]/QGPO[1:2]		
	RXD[1:2]/QGPI[1:2]		
	ECK		
MIOS	MDA[11], [13]	18	5 V
	MDA[12], [14]		
	MDA[15], [27:31]		
	MPWM[0:3], [16:19]		
General-Purpose I/O from MIOS	VF[0:2]/MPIO32B[0:2]	5	3-V / 5-V GPIO
	VFLS[0:1]/MPIO32B[3:4]		
	MPIO32B[5:15]	11	5 V
TPU	A_TPUCH[0:15], B_TPUCH[0:15]	34	5 V
	A_T2CLK, B_T2CLK		
QADC	ETRIG[1:2]	34	5 V
	A_AN0/ANW/PQB0, B_AN0/ANW/PQB0		
	A_AN1/ANX/PQB1, B_AN1/ANX/PQB1		
	A_AN2/ANY/PQB2, B_AN2/ANY/PQB2		
	A_AN3/ANZ/PQB3, B_AN3/ANZ/PQB3		
	A_AN[48:51]/PQB[4:7], B_AN[48:51]/PQB[4:7]		
	A_AN[52:54]/MA[0:2]/PQA[0:2], B_AN[52:54]/MA[0:2]/PQA[0:2]		
	A_AN[55:56]/PQA[3:4], B_AN[55:56]/PQA[3:4]		
	A_AN[57:59]/PQA[5:7], B_AN[57:59]/PQA[5:7]		
TouCAN	A_CNTX0, B_CNTX0, A_CNRX0, B_CNRX0	4	5 V
Flash EEPROM	E_{PEE}	1	3 V
Supplies			
Ground	VSS, VSSF, VSSSYN	18	
Analog Ground	VSSA, VRL	2	
Low Voltage Supply	VDDI, VDDL, VDDSRAM, VDDSYN, KAPWR3, VDDF	16	3 V

Table 2-1 MPC555 / MPC556 Pin Functions for 272-Pin PBGA (Continued)

Functional Group	Signals[1]	Pins	3 V / 5 V[2]
High voltage Supply	VDDH, VDDA, VRH	12	5 V
Programming Voltage	VPP	1	3-V / 5-V

NOTES:
1. "/" implies that the corresponding functions are multiplexed on the pin
2. All inputs are 5 V friendly. All 5 V outputs are slow slew rate except for SCI transmit pins.
3. These pins are powered by KAPWR (Keep Alive Power Supply).

2.2 Pin Functionality

The pad ring supports 234 functional pins (284 including all power and ground). Some pins serve multiple functions. The pad characteristics for each pin are described in **Table 2-2**. This table contains the following columns:

- Pin – List of functional (signal) names for each pin. (For actual pin names, see **2.7 Pin Names and Abbreviations**.
- Function – Name of function (signal). Each pin supports one or more functions, and each function (signal) name is a separate entry in the table.
- Driver Type – Type of driver that is used to drive the pin (for output functionality). Types of output drivers are:
  - — Totem pole (TP). This driver type uses a push pull scheme to drive the pin. These pins can be driven high or low or can be three-stated. Care must be taken to ensure that there is no contention on this pin (for example, an external driver driving the pin high while an internal driver is driving it low).
  - — Open drain (OD). This driver type uses an open drain approach to drive the pins. Pins with an OD driver can be either driven low or three-stated. This driver scheme is typically used for pins that could potentially be asserted by multiple modules.
  - — Active negated (ANG). This driver type fully drives a low level. A high level is driven and then released. A pull-up resistor may be needed on this type of output.
- Receiver Type – Type of receiver used for the pin. Some inputs need to have a synchronizer to prevent latching a metastable signal at the pins. Such requirements are indicated in this column with the abbreviation "synch." Another possible entry is "glitch filter." It is added to reset signals.
- Direction – Direction of the pin for each function it supports. The possible directions are input (I), output (O) and bi-directional (I/O).
- Voltage – Voltage requirement for each function of a pin. There are two supply voltages: 5 V and 3 V.
- Slew rate – Timing needed from the 5-V drivers. The options are with slew rate (typically 200/50 ns with 50 pF load) or fast 5-V driver.
- Drive strength – Drive strength for 3-V drivers of the output load. For all 3-V outputs, the drive strength is 25/50 pF. For two pads (clkout and engclk) the drive strength is 45/90 pF.
- Pad Type – Functional pad structure used for a pin. For pad type descriptions, see **2.5 Pad Types**.

Table 2-2 Pin Functionality Table

Pin	Function	Type		Direction[1]	Voltage	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
USIU								
ADDR[8:31]/ SGPIOA[8:31]	ADDR[8:31]	TP	—	I/O	3 V	—	25 / 50	J
	SGPIOA[8:31]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
DATA[0:31] SGPIOD[0:31]	DATA[0:31]	TP	—	I/O	3 V	—	25 / 50	JD
	SGPIOD[0:31]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
$\overline{\text{IRQ}}$[0]/ SGPIOC[0]	$\overline{\text{IRQ}}$[0]	—	Hysteresis, Synch	I	3 V	—	—	IH
	SGPIOC[0]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
$\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/ SGPIOC[1]	$\overline{\text{IRQ}}$[1]	—	Hysteresis, Synch	I	3 V	—	—	IH
	$\overline{\text{RSV}}$	TP	—	O	3 V	—	25 / 50	
	SGPIOC[1]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
$\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/ SGPIOC[2]/ $\overline{\text{MTS}}$	$\overline{\text{IRQ}}$[2]	—	Hysteresis, Synch	I	3 V	—	—	IH
	$\overline{\text{CR}}$	—	—	I	3 V	—	—	
	SGPIOC[2]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
	MTS	TP	—	O	3 V	25 / 50	—	
$\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$, $\overline{\text{RETRY}}$/ SGPIOC[3]	$\overline{\text{IRQ}}$[3]	—	Hysteresis, Synch	I	3 V	—	—	IH
	$\overline{\text{KR}}$, $\overline{\text{RETRY}}$	TP	—	I/O	3 V	—	25 / 50	
	SGPIOC[3]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
$\overline{\text{IRQ}}$[4]/AT[2]/ SGPIOC[4]	$\overline{\text{IRQ}}$[4]	—	Hysteresis, Synch	I	3 V	—	—	IH
	AT[2]	TP	—	O	3 V	—	25 / 50	
	SGPIOC[4]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
$\overline{\text{IRQ}}$[5]/ SGPIOC[5]/ MODCK[1][2]	$\overline{\text{IRQ}}$[5]	—	Hysteresis, Synch	I	3 V	—	—	IH
	SGPIOC[5]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
	MODCK[1]	—	—	I	3 V	—	—	
$\overline{\text{IRQ}}$[6:7]/ MODCK[2:3][2]	$\overline{\text{IRQ}}$[6:7]	—	Hysteresis, Synch	I	3 V	—	—	CH
	MODCK[2:3]	—	—	I	3 V	—	—	
TSIZ[0:1]	TSIZ[0:1]	TP	—	I/O	3 V	—	25 / 50	F
RD/$\overline{\text{WR}}$	RD/$\overline{\text{WR}}$	TP	—	I/O	3 V	—	25 / 50	F
$\overline{\text{BURST}}$	$\overline{\text{BURST}}$	TP	—	I/O	3 V	—	25 / 50	F

Table 2-2 Pin Functionality Table (Continued)

Pin	Function	Type		Direction[1]	Voltage	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
$\overline{BDIP}$	$\overline{BDIP}$	TP	—	I/O	3 V	—	25 / 50	F
$\overline{TS}$[3]	$\overline{TS}$	ANG	—	I/O	3 V	—	25 / 50	E
$\overline{TA}$[3]	$\overline{TA}$	ANG	—	I/O	3 V	—	25 / 50	E
$\overline{TEA}$	$\overline{TEA}$	OD	—	I/O	3 V	—	25 / 50	E
$\overline{RSTCONF}$/ TEXP[2]	$\overline{RSTCONF}$	—	—	I	3 V	—	—	E
	TEXP	TP	—	O	3 V	—	25 / 50	
$\overline{OE}$	$\overline{OE}$	TP	—	O	3 V	—	25 / 50	A
$\overline{BI}$/$\overline{STS}$	$\overline{BI}$[3]	ANG	—	I/O	3 V	—	25 / 50	E
	$\overline{STS}$	TP	—	O	3 V	—	25 / 50	
$\overline{CS}$[0:3]	$\overline{CS}$[0:3]	TP	—	O	3 V	—	25 / 50	A
$\overline{WE}$[0:3]/ $\overline{BE}$[0:3]/AT[0:3]	$\overline{WE}$[0:3]/$\overline{BE}$[0:3]	TP	—	O	3 V	—	25 / 50	F
	AT[0:3]	TP	—	O	3 V	—	25 / 50	
$\overline{PORESET}$[2]	$\overline{PORESET}$	—	Hysteresis Glitch filter	I	3 V	—	—	CNH
$\overline{HRESET}$[2]	$\overline{HRESET}$	OD	Hysteresis Glitch filter	I/O	3 V	—	25 / 50	EOH
$\overline{SRESET}$[,2]	$\overline{SRESET}$	OD	Hysteresis Glitch filter	I/O	3 V	—	25 / 50	EOH
SGPIOC[6]/ FRZ/$\overline{PTR}$	SGPIOC[6]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	I
	FRZ	TP	—	O	3 V	—	25 / 50	
	$\overline{PTR}$	TP	—	O	3 V	—	25 / 50	
SGPIOC[7]/ $\overline{IRQOUT}$/ LWP[0]	SGPIOC[7]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	I
	$\overline{IRQOUT}$	TP	—	O	3 V	—	25 / 50	
	LWP[0]	TP	—	O	3 V	—	25 / 50	
$\overline{BG}$/VF[0]/ LWP[1]	$\overline{BG}$	TP	—	I/O	3 V	—	25 / 50	G
	VF[0]	TP	—	O	3 V	—	25 / 50	
	LWP[1]	TP	—	O	3 V	—	25 / 50	
$\overline{BR}$/VF[1]/ IWP[2]	$\overline{BR}$	TP	—	I/O	3 V	—	25 / 50	G
	VF[1]	TP	—	O	3 V	—	25 / 50	
	IWP[2]	TP	—	O	3 V	—	25 / 50	
$\overline{BB}$/VF[2]/ IWP[3]	$\overline{BB3}$	ANG	—	I/O	3 V	—	25 / 50	G
	VF[2]	TP	—	O	3 V	—	25 / 50	
	IWP[3]	TP	—	O	3 V	—	25 / 50	
IWP[0:1]/ VFLS[0:1]	IWP[0:1]	TP	—	O	3 V	—	25 / 50	A
	VFLS[0:1]	TP	—	O	3 V	—	25 / 50	
TMS	TMS	—	—	I	3 V	—	—	C
TDI/DSDI	TDI	—	—	I	3 V	—	—	C
	DSDI	—	—	I	3 V	—	—	

Table 2-2 Pin Functionality Table (Continued)

Pin	Function	Type		Direction[1]	Voltage	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
TCK/DSCK	TCK	—	—	I	3 V	—	—	D
	DSCK	—	—	I	3 V	—	—	
TDO/DSDO	TDO	TP	—	O	3 V	—	25 / 50	A
	DSDO	TP	—	O	3 V	—	25 / 50	
$\overline{TRST}$	$\overline{TRST}$	—	—	I	3 V	—	—	C
XTAL[2]	XTAL	TP	—	O	3 V	—	—	—
EXTAL[2]	EXTAL	—	—	I	3 V	—	—	—
XFC	XFC	—	—	I/O	3 V	—	—	—
CLKOUT	CLKOUT	TP	—	O	3 V	—	30 / 90[4]	B
EXTCLK[2]	EXTCLK	—	—	I	3 V	—	—	—
ENGCLK/ BUCLK	ENGCLK	TP	—	O	5 V	—	25 / 50[5]	S
	BUCLK	TP	—	O	5 V	—	25 / 50	
QSMCM								
PCS0/ $\overline{SS}$/QGPIO[0]	PCS0	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	O
	$\overline{SS}$	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	
	QGPIO[0]	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	
PCS[1:3]/ QGPIO[1:3]	PCS[1:3]	TP/OD	Synch	I/O	5 V	50 / fast	—	O
	QGPIO[1:3]	TP/OD	Synch	I/O	5 V	50 / fast	—	
MISO/QGPIO[4]	MISO	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	O
	QGPIO[4]	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	
MOSI/QGPIO[5]	MOSI	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	O
	QGPIO[5]	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	
SCK/QGPIO[6]	SCK	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	O
	QGPIO[6]	TP/OD	Synch/ No Synch	I/O	5 V	50 / fast	—	
TXD[1:2]/ QGPO[1:2]	TXD[1:2]	TP/OD	—	O	5 V	200 / fast	—	Q
	QGPO[1:2]	TP/OD	—	O	5 V	200 / fast	—	
RXD[1:2]/ QGPI[1:2]	RXD[1:2]	—	—	I	5 V	—	—	R
	QGPI[1:2]	—	—	I	5 V	—	—	
ECK	ECK	—	—	I	5 V	—	—	R
MIOS								
MDA[11:15]	MDA[11:15]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P
MDA[27:31]	MDA[27:31]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P

Table 2-2 Pin Functionality Table (Continued)

Pin	Function	Type		Direction[1]	Voltage	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
MPWM[0:3], [16:19]	MPWM[0:3], [16:19]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P
VF[0:2]/ MPIO32B[0:2]	VF[0:2]	TP	—	O	3 V	—	25 / 50	H
	MPIO32B[0:2]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
VFLS[0:1]/ MPIO32B[3:4]	VFLS[0:1]	TP	—	O	3 V	—	25 / 50	H
	MPIO32B[3:4]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	
MPIO32B[5:15]	MPIO32B[5:15]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	O
TPU_A/TPU_B								
A_TPUCH[0:15]	TPUCH[0:15]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P
A_T2CLK	T2CLK	TP	Hysteresis Synch	I/O	5 V	200 / fast	—	P
B_TPUCH[0:15]	TPUCH[0:15]	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P
B_T2CLK	T2CLK	TP	Hysteresis, Synch	I/O	5 V	200 / fast	—	P
QADC_A/QADC_B								
ETRIG[1:2]	ETRIG[1:2]	—	Synch	I	5 V	—	—	N
AN0/ ANW/ PQB0	AN0	—	Analog	I	5 V	—	—	M
	ANW	—	Analog	I	5 V	—	—	
	PQB0	—	Hysteresis, Synch	I	5 V	—	—	
AN1/ANX/PQB1	AN1	—	Analog	I	5 V	—	—	M
	ANX	—	Analog	I	5 V	—	—	
	PQB1	—	Hysteresis, Synch	I	5 V	—	—	
AN2/ANY/PQB2	AN2	—	Analog	I	5 V	—	—	M
	ANY	—	Analog	I	5 V	—	—	
	PQB2	—	Hysteresis, Synch	I	5 V	—	—	
AN3/ANZ/PQB3	AN3	—	Analog	I	5 V	—	—	M
	ANZ	—	Analog	I	5 V	—	—	
	PQB3	—	Hysteresis, Synch	I	5 V	—	—	
AN[48:51]/ PQB[4:7]	AN[48:51]	—	Analog	I	5 V	—	—	M
	PQB[4:7]	—	Hysteresis, Synch	I	5 V	—	—	
AN[52:54]/ MA[0:2]/ PQA[0:2]	AN[52:54]	—	Analog	I	5 V	—	—	L
	MA[0:2]	OD	—	O	5 V	—	—	
	PQA[0:2]	OD	Hysteresis, Synch	I/O	5 V	—	—	

Table 2-2 Pin Functionality Table (Continued)

Pin	Function	Type		Direc-tion[1]	Volt-age	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
AN[55:56]/ PQA[3:4]	AN[55:56]	—	Analog	I	5 V	—	—	L
	PQA[3:4]	OD	Hysteresis, Synch	I/O	5 V	—	—	
AN[57:59]/ PQA[5:7]	AN[57:59]	—	Analog	I	5 V	—	—	L
	PQA[5:7]	OD	Hysteresis, Synch	I/O	5 V	—	—	
AN0/ANW/ PQB0	AN0	—	Analog	I	5 V	—	—	M
	ANW	—	Analog	I	5 V	—	—	
	PQB0	—	Hysteresis, Synch	I	5 V	—	—	
AN1/ANX/PQB1	AN1	—	Analog	I	5 V	—	—	M
	ANX	—	Analog	I	5 V	—	—	
	PQB1	—	Hysteresis, Synch	I	5 V	—	—	
AN2/ANY/PQB2	AN2	—	Analog	I	5 V	—	—	M
	ANY	—	Analog	I	5 V	—	—	
	PQB2	—	Hysteresis, Synch	I	5 V	—	—	
AN3/ANZ/PQB3	AN3	—	Analog	I	5 V	—	—	M
	ANZ	—	Analog	I	5 V	—	—	
	PQB3	—	Hysteresis, Synch	I	5 V	—	—	
AN[48:51]/ PQB[4:7]	AN[48:51]	—	Analog	I	5 V	—	—	M
	PQB[4:7]	—	Hysteresis, Synch	I	5 V	—	—	
AN[52:54]/ MA[0:2]/ PQA[0:2]	AN[52:54]	—	Analog	I	5 V	—	—	L
	MA[0:2]	OD	—	O	5 V	—	—	
	PQA[0:2]	OD	Hysteresis, Synch	I/O	5 V	—	—	
AN[55:56]/ PQA[3:4]	AN[55:56]	—	Analog	I	5 V	—	—	L
	PQA[3:4]	OD	Hysteresis, Synch	I/O	5 V	—	—	
AN[57:59]/ PQA[5:7]	AN[57:59]	—	Analog	I	5 V	—	—	L
	PQA[5:7]	OD	Hysteresis, Synch	I/O	5 V	—	—	
TOUCAN_A/TOUCAN_B								
A_CNTX0	CNTX0_A	TP/OD	—	O	5 V	50 / fast	—	Q
B_CNTX0	CNTX0_B	TP/OD	—	O	5 V	50 / fast	—	Q
A_CNRX0	CNRX0_A	—	Synch / No Synch	I	5 V	—	—	R
B_CNRX0	CNRX0_B	—	Synch / No Synch	I	5 V	—	—	R

Table 2-2 Pin Functionality Table (Continued)

Pin	Function	Type		Direc-tion[1]	Volt-age	Slew Rate ns / 50 pF	Drive Strength (pF)	Pad Type
		Driver	Receiver					
CMF								
EPEE	EPEE	—	Sequencer	I	3 V	—	—	K
VPP	VPP	—	—	I	5 V	—	—	—
Global Power Supplies								
VDDA	VDDA	—	—	I	5 V	—	—	—
VDDF	VDDF	—	—	I	3 V	—	—	—
VDDL	VDDL	—	—	I	3 V	—	—	—
VDDH	VDDH	—	—	I	5 V	—	—	—
VDDI	VDDI	—	—	I	3 V	—	—	—
VDDSYN	VDDSYN	—	—	I	3 V	—	—	—
VRH	VRH	—	—	I	5 V	—	—	—
VRL	VRL	—	—	I	—	—	—	—
VSSA	VSSA	—	—	I	—	—	—	—
VSSF	VSSF	—	—	I	—	—	—	—
VSSSYN	VSSSYN	—	—	I	—	—	—	—
KAPWR[2]	KAPWR	—	—	I	3 V	—	—	—
VDDSRAM	VDDSRAM	—	—	I	3 V	—	—	—
VSS	VSS	—	—	I	—	—	—	—

NOTES:
1. All inputs are 5-V friendly. All 5-V outputs are slow slew rate. The QSMCM and TouCAN pins have some slew rate control, but are faster than the general/purpose I/O and timer pins.
2. These pins are powered by KAPWR (Keep Alive Power Supply).
3. This pin is an active negate signal and may need an external pull-up resister.
4. Drive strength was 45/90 in make sets prior to K62N.
5. Drive strength was 45/90 in make sets prior to K62N.

2.3 Signal Descriptions

The pad ring supports 234 functional pins (284 including all power and ground). Each pin and the functionality it supports are described in this section. All references to timing in this document are numbers that are expected for a typical case process with a 50-pF load at 25°C. The supply voltages are assumed to be typical, as well: 5 V or 3.3 V. The 5-V supply is generally referred to as the 5-V supply, and the 3.3-V supply is referred to as the 3-V supply in this section.

2.3.1 USIU Pads

2.3.1.1 ADDR[8:31]/SGPIOA[8:31]

Pin Name: addr_sgpioa[8:31] (24 pins)

Address Bus – Specifies the physical address of the bus transaction. The address is driven onto the bus and kept valid until a transfer acknowledge is received from the slave. ADDR8 is the most significant signal for this bus.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.2 DATA[0:31]/SGPIOD[0:31]

Pin Name: data_sgpiod[0:31] (32 pins)

Data Bus – Provides the general purpose data path between the chip and all other devices. Although the data path is a maximum of 32 bits wide, it can be sized to support 8-, 16-, or 32-bit transfers. DATA[0] is the MSB of the data bus.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.3 $\overline{\text{IRQ}}$[0]/SGPIOC[0]

Pin Name: irq0_b_sgpioc0

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU. IRQ0 is a nonmaskable interrupt (NMI).

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.4 $\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1]

Pin Name: irq1_b_rsv_b_sgpioc1

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Reservation – This line used together with the address bus to indicate that the internal core initiated a transfer as a result of a STWCX or a LWARX instruction.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.5 $\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$

Pin Name: irq2_b_cr_b_sgpioc2_mts

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Cancel Reservation – Instructs the chip to clear its reservation, some other master has touched its reserved space. An external bus snooper would assert this signal.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

Memory Transfer Start – This pin is the transfer start signal from the MPC555 / MPC556 memory controller to allow external memory access by an external bus master.

2.3.1.6 $\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC[3]

Pin Name: irq3_b_kr_b_retry_b_sgpioc3

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Kill Reservation – In case of a bus cycle initiated by a STWCX instruction issued by the CPU core to a non-local bus on which the storage reservation has been lost, this signal is used by the non-local bus interface to back-off the cycle.

Retry – Indicates to a master that the cycle is terminated but should be repeated. As an input, it is driven by the external slave to retry a cycle.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.7 $\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4]

Pin Name: irq4_b_at2_sgpioc4

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Address Type – A bit from the address type bus which indicates one of the 16 “address types” to which the address applies. The address type signals are valid at the rising edge of the clock in which the special transfer start ($\overline{\text{STS}}$) is asserted. AT[2] identifies an access as either data or instrucion.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.1.8 $\overline{\text{IRQ}}$[5]/SGPIOC[5]/MODCK[1]

Pin Name: irq5_b_sgpioc5_modck1

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

Mode Clock [1] – Sampled at the negation of $\overline{\text{PORESET}}$ in order to configure the phase-locked loop (PLL)/clock mode of operation.

2.3.1.9 $\overline{\text{IRQ}}$[6:7]/MODCK[2:3]

Pin Name: irq6_b_modck2 - irq7_b_modck3 (2 pins)

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Mode Clock [2:3] – Sampled at the negation of $\overline{\text{PORESET}}$ in order to configure the PLL/clock mode of operation.

2.3.1.10 TSIZ[0:1]

Pin Name: tsiz0 - tsiz1 (2 pins)

Transfer size – Indicates the size of the requested data transfer in the current bus cycle.

2.3.1.11 RD/$\overline{\text{WR}}$

Pin Name: rd_wr_b

Read/Write – Indicates the direction of the data transfer for a transaction. A logic one indicates a read from a slave device; a logic zero indicates a write to a slave device.

2.3.1.12 $\overline{\text{BURST}}$

Pin Name: burst_b

Burst Indicator – Indicates whether the current transaction is a burst transaction or not.

2.3.1.13 $\overline{\text{BDIP}}$

Pin Name: bdip_b

Burst data in progress – Indicates to the slave that there is a data beat following the current data beat.

2.3.1.14 $\overline{\text{TS}}$

Pin Name: ts_b

Transfer Start – Indicates the start of a bus cycle that transfers data to/from a slave device. This signal is driven by the master only when it gained the ownership of the bus. Every master should negate this signal before the bus relinquish. Every master should negate this signal before the bus is relinquished. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

2.3.1.15 $\overline{\text{TA}}$

Pin Name: ta_b

Transfer Acknowledge – This line indicates that the slave device addressed in the current transaction has accepted the data transferred by the master (write) or has driven the data bus with valid data (read). The slave device negates the TA_B signal after the end of the transaction and immediately three-state it to avoid contentions on the line if a new transfer is initiated addressing other slave devices. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

2.3.1.16 $\overline{\text{TEA}}$

Pin Name: tea_b

Transfer Error Acknowledge – This signal indicates that a bus error occurred in the current transaction. The MCU asserts this signal when the bus monitor does not detect a bus cycle termination within a reasonable amount of time. The assertion of $\overline{TEA}$ causes the termination of the current bus cycle, regardless of the state of $\overline{TEA}$. An external pull-up device is required to negate $\overline{TEA}$ quickly, before a second error is detected. That is, the pin must be pulled up within one clock cycle of the time it was three-stated by the MPC555 / MPC556.

2.3.1.17 $\overline{RSTCONF}$/TEXP

Pin Name: rstconf_b_texp

Reset Configuration – Input. This input line is sampled by the chip during the assertion of the $\overline{HRESET}$ signal in order to sample the reset configuration. If the line is asserted, the configuration mode will be sampled from the external data bus. When this line is negated, the configuration mode adopted by the chip will be the default one.

Timer Expired – This output line reflects the status of the TEXPS bit in the PLPRCR register in the USIU. This indicates an expired timer value.

2.3.1.18 $\overline{OE}$

Pin Name: oe_b

Output Enable – This output line is asserted when a read access to an external slave controlled by the GPCM in the memory controller is initiated by the chip.

2.3.1.19 $\overline{BI}$/$\overline{STS}$

Pin Name: bi_b_sts_b

Burst Inhibit – This bi-directional, active low, three-state line indicates that the slave device addressed in the current burst transaction is not able to support burst transfers. When the chip drives out the signal for a specific transaction, it asserts or negates $\overline{BI}$ during the transaction according to the value specified by the user in the appropriate control registers. Negation of the signal occurs after the end of the transaction followed by the immediate three-state. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

Special Transfer Start – This output signal is driven by the chip to indicate the start of a transaction on the external bus or signals the beginning of an internal transaction in showcycle mode.

2.3.1.20 $\overline{CS}$[0:3]

Pin Name: cs0_b - cs3_b (4 pins)

Chip Select – These output signals enable peripheral or memory devices at programmed addresses if defined appropriately in the memory controller. $\overline{CS0}$ can be configured to be the global chip select for the boot device.

2.3.1.21 $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3]

Pin Name: we_b_at[0:3](4 pins)

Write Enable[0:3]/Byte Enable[0:3] – This output line is asserted when a write access to an external slave controlled by the GPCM in the memory controller is initiated by the chip. It can be optionally be asserted on all read and write accesses. See WEBS bit definition in **Table 10-7**. $\overline{WE}$[0]/$\overline{BE}$[0] is asserted if the data lane DATA[0:7] contains valid data to be stored by the slave device. $\overline{WE}$[1]/$\overline{BE}$[1] is asserted if the data lane DATA[8:15] contains valid data to be stored by the slave device. $\overline{WE}$[2]/$\overline{BE}$[2] is asserted if the data line DATA[16:23] contains valid data to be stored by the slave device. $\overline{WE}$[3]/$\overline{BE}$[3] is asserted if the data lane DATA[24:31] contains valid data to be stored by the slave device.

Address Type – Indicates one of the 16 address types to which the address applies. The address type signals are valid at the rising edge of the clock in which the Special Transfer Start ($\overline{STS}$) is asserted.

2.3.1.22 $\overline{PORESET}$

Pin Name: poreset_b

Power on Reset – This pin should be activated as a result of a voltage failure on the keep-alive power pins. The pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{PORESET}$ signal is asserted only if $\overline{PORESET}$ is asserted for more than 100 ns. See **SECTION 7 RESET** for more details on timing.

2.3.1.23 $\overline{HRESET}$

Pin Name: hreset_b

Hard Reset – The chip can detect an external assertion of $\overline{HRESET}$ only if it occurs while the chip is not asserting reset. After negation of $\overline{HRESET}$ or $\overline{SRESET}$ is detected, a 16 cycles period is taken before testing the presence of an external reset. The internal $\overline{HRESET}$ signal is asserted only if $\overline{HRESET}$ is asserted for more than 100 ns. To meet external timing requirements, an external pull-up device is required to negate $\overline{HRESET}$. See **SECTION 7 RESET** for more details on timing.

2.3.1.24 $\overline{SRESET}$

Pin Name: sreset_b

Soft Reset – The chip can detect an external assertion of $\overline{SRESET}$ only if it occurs while the chip is not asserting reset. After negation of $\overline{HRESET}$ or $\overline{SRESET}$ is detected, a 16-cycle period is taken before testing the presence of an external soft reset. To meet external timing requirements, an external pull-up device is required to negate $\overline{SRESET}$. See **SECTION 7 RESET** for more details on timing.

2.3.1.25 SGPIOC[6]/FRZ/$\overline{PTR}$

Pin Name: sgpioc6_frz_ptr_b

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

Freeze – Indicates that the RCPU is in debug mode.

Program Trace – Indicates an instruction fetch is taking place in order to allow program flow tracking.

2.3.1.26 SGPIOC[7]/$\overline{\text{IRQOUT}}$/LWP[0]

Pin Name: sgpioc7_irqout_b_lwp0

SGPIO – This function allows the pins to be used as general purpose inputs/outputs.

Interrupt Out – Indicates that an interrupt has been requested to all external devices.

Load/Store Watchpoint 0 – This output line reports the detection of a data watchpoint in the program flow executed by the RCPU. See **SECTION 21 DEVELOPMENT SUPPORT** for more details.

2.3.1.27 $\overline{\text{BG}}$/VF[0]/LWP[1]

Pin Name: bg_b_vf0_lwp1

Bus Grant – Indicates external data bus status. Is asserted low when the arbiter of the external bus grants to the specific master the ownership of the bus.

Visible Instruction Queue Flush Status – This output line together with VF1 and VF2 is output by the chip when a program instructions flow tracking is required by the user. VF report the number of instructions flushed from the instruction queue in the internal core. See **SECTION 21 DEVELOPMENT SUPPORT** for more details.

Load/Store Watchpoint – This output line reports the detection of a data watchpoint in the program flow executed by the RCPU.

2.3.1.28 $\overline{\text{BR}}$/VF[1]/IWP[2]

Pin Name: br_b_vf1_iwp2

Bus Request – Indicates that the data bus has been requested for external cycle.

Visible Instruction Queue Flush Status – This output line together with VF1 and VF2 is output by the chip when a program instructions flow tracking is required by the user. VF report the number of instructions flushed from the instruction queue in the internal core. See **SECTION 21 DEVELOPMENT SUPPORT** for more details.

Instruction Watchpoint 2 – This output line reports the detection of an instruction watchpoint in the program flow executed by the RCPU.

2.3.1.29 $\overline{\text{BB}}$/VF[2]/IWP[3]

Pin Name: bb_b_vf2_iwp3

Bus Busy – Indicates that the master is using the bus. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

Visible Instruction Queue Flush Status – This output line together with VF0 and VF1 is output by the chip when a program instructions flow tracking is required by the user. VF report the number of instructions flushed from the instruction queue in the internal core.

Instruction Watchpoint 3 – This output line reports the detection of an instruction watchpoint in the program flow executed by the internal core.

2.3.1.30 IWP[0:1]/VFLS[0:1]

Pin Name: iwp0_vfls0 - iwp1_vfls1 (2 pins)

Instruction Watchpoint – These output lines report the detection of an instruction watchpoint in the program flow executed by the RCPU.

Visible History Buffer Flush Status – These signals are output by the chip to enable program instruction flow tracking. They report the number of instructions flushed from the history buffer in the RCPU. See **SECTION 21 DEVELOPMENT SUPPORT** for details.

2.3.1.31 TMS

Pin Name: tms

Test Mode Select – This input controls test mode operations for on-board test logic (JTAG).

2.3.1.32 TDI/DSDI

Pin Name: tdi_dsdi

Test Data In – This input is used for serial test instructions and test data for on-board test logic (JTAG).

Development Serial Data Input – This input line is the data in for the debug port interface. See **SECTION 21 DEVELOPMENT SUPPORT** for details.

2.3.1.33 TCK/DSCK

Pin Name: tck_dsck

Test Clock – This input provides a clock for on-board test logic (JTAG).

Development Serial Clock – This input line is the clock for the debug port interface. See **SECTION 21 DEVELOPMENT SUPPORT** for details.

2.3.1.34 TDO/DSDO

Pin Name: tdo_dsdo

Test Data Out – This output is used for serial test instructions and test data for on-board test logic (JTAG).

Development Serial Data Output – This output line is the data-out line of the debug port interface. See **SECTION 21 DEVELOPMENT SUPPORT** for details.

2.3.1.35 $\overline{\text{TRST}}$

Pin Name: trst_b

Test Reset – This input provides asynchronous reset to the test logic (JTAG).

For non-JTAG test applications, $\overline{\text{TRST}}$ should be connected to ground or $\overline{\text{PORESET}}$ via an external resistor.

2.3.1.36 XTAL

Pin Name: xtal

XTAL – This output line is one of the connections to an external crystal for the internal oscillator circuitry.

2.3.1.37 EXTAL

Pin Name: extal

EXTAL – This line is one of the connections to an external crystal for the internal oscillator circuitry. If this pin is unused, it must be grounded.

2.3.1.38 XFC

Pin Name: xfc

External Filter Capacitance – This input line is the connection pin for an external capacitor filter for the PLL circuitry.

2.3.1.39 CLKOUT

Pin Name: clkout

Clock Out – This output line is the clock system frequency. The CLKOUT drive strength can be configured to full strength, half strength, or disabled. The drive strength is configured using the COM[0:1] bits in the SCCR register in the USIU.

2.3.1.40 EXTCLK

Pin Name: extclk

EXTCLK – Input. This is the external frequency source for the chip. If this is unused, the pin must be grounded.

2.3.1.41 VDDSYN

Pin Name: vddsyn

VDDSYN – This is the power supply of the PLL circuitry.

2.3.1.42 VSSSYN

Pin Name: vsssyn

VSSSYN – This is the power supply of the PLL circuitry.

2.3.1.43 ENGCLK/BUCLK

Pin Name: engclk_buclk

ENGCLK – This is the engineering clock output. Drive strength can be configured to full strength, half strength or disabled. The drive strength is configured using the EECLK[0:1] bits in the SCCR register in the SIU.

BUCLK – When the chip is in limp mode, it is operating from a less precise on-chip ring oscillator to allow the system to continue minimum functionality until the system clock is fixed. This backup clock can be seen externally based on the values of the EECLK[0:1] bits in the SCCR register in the USIU.

2.3.2 QSMCM PADS

2.3.2.1 PCS[0]/$\overline{\text{SS}}$/QGPIO[0]

Pin Name: pcs0_ss_b_qgpio0

PCS[0] – This signal provides QSPI peripheral chip select 0.

$\overline{\text{SS}}$ – Assertion of this bi-directional signal places the QSPI in slave mode.

QSPI GPIO[0] – When this pin is not needed for a QSPI application it can be configured as a general purpose input/output.

2.3.2.2 PCS[1:3]/QGPIO[1:3]

Pin Name: pcs1_qgpio1 - pcs3_qgpio3 (3 pins)

PCS[1:3] – These signals provide three QSPI peripheral chip selects.

QGPIO[1:3] – When these pins are not needed for QSPI applications they can be configured as a general purpose input/output.

2.3.2.3 MISO/QGPIO[4]

Pin Name: miso_qgpio4

Master-In Slave-Out (MISO) – This bi-directional signal furnishes serial data input to the QSPI in master mode, and serial data output from the QSPI in slave mode.

QGPIO[4] – When this pin is not needed for a QSPI application it can be configured as a general purpose input/output.

2.3.2.4 MOSI/QGPIO[5]

Pin Name: mosi_qgpio5

Master-Out Slave-In (MOSI) – This bi-directional signal furnishes serial data output from the QSPI in master mode and serial data input to the QSPI in slave mode.

QGPIO[5] – When this pin is not needed for a QSPI application it can be configured as a general purpose input/output.

2.3.2.5 SCK/QGPIO[6]

Pin Name: sck_qgpio6

SCK – This bi-directional signal furnishes the clock from the QSPI in master mode or furnishes the clock to the QSPI in slave mode.

QGPIO[6] – When this pin is not needed for a QSPI application, it can be configured as a general purpose input/output. When the QSPI is enabled for serial transmitting, the pin can *not* function as a GPIO.

2.3.2.6 TxD[1:2]/QGPO[1:2]

Pin Name: txd1_qgpo1 - txd2_qgpo2 (2 pins)

Transmit Data – These output signals are the serial data outputs from the SCI1 and SCI2.

QSCI GPO[1:2] – When these pins are not needed for a SCI applications, they can be configured as general-purpose outputs. When the transmit enable bit in the SCI control register is set to a logic 1, these pins can *not* function as general purpose outputs

2.3.2.7 RxD[1:2]/QGPI[1:2]

Pin Name: rxd1_qgpi1 - rxd2_qgpi2 (2 pins)

Receive Data – These input signals furnish serial data inputs to the SCI1 and SCI2.

QSCI GPI[1:2] – When these pins are not needed for SCI applications they can be configured as general purpose inputs. When the receive enable bit in the SCI control register is set to a logic 1, these pins can *not* function as general purpose inputs.

2.3.2.8 ECK

Pin Name: eck

External Baud Clock – This signal provides an external baud clock used by SCI1 and SCI2.

2.3.3 MIOS PADS

2.3.3.1 MDA[11], [13]

Pin Name: mda11, mda13 (2 pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

Clock Input – Each of these pins provide a clock input to the modulus counter submodule. MDA11 can be used as the clock input to the MMCSM6 modulus counter. MDA13 can be used as the clock input to the MMCSM22 modulus counter.

2.3.3.2 MDA[12], [14]

Pin Name: mda12, mda14, (2 pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

Load Input – Each of these pins provide a load input to the modulus counter submodule. MDA12 can be used as the load input to the MMCSM6 modulus counter. MDA14 can be used as the load input to the MMCSM22 modulus counter.

2.3.3.3 MDA[15], [27:31]

Pin Name: mda15, mda27 - mda31 (6 pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

2.3.3.4 MPWM[0:3], [16:19]

Pin Name: mpwm0 - mpwm3, mpwm16 - mpwm19 (8 pins)

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

2.3.3.5 VF[0:2]/MPIO32B[0:2]

Pin Name: vf0_mpio32b0 - vf2_mpio32b2 (3 pins)

Visible Instruction Queue Flush Status – These lines output by the chip when Program instruction flow tracking is required by the user. VF reports the number of instructions flushed from the instruction queue in the internal core.

MIOS GPIO – This function allows the pins to be used as general-purpose inputs/outputs.

2.3.3.6 VFLS[0:1]/MPIO32B[3:4]

Pin Name: vfls0_mpio32b3 - vfls1_mpio32b4 (2 pins)

Visible History Buffer Flush Status – These signals are output by the chip to allow program instruction flow tracking. They report the number of instructions flushed from the history buffer in the RCPU. See **SECTION 21 DEVELOPMENT SUPPORT** for details.

MIOS GPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.3.7 MPIO32B[5:15]

Pin Name: mpio32b5 - mpio32b15 (11 pins)

MIOS GPIO – This function allows the pins to be used as general purpose inputs/outputs.

2.3.4 TPU_A/TPU_B PADS

2.3.4.1 TPUCH[0:15]_[A:B]

Pin Name: a_tpuch0 - a_tpuch15 (16 pins for first TPU), b_tpuch0 - b_tpuch15 (16 pins for second TPU)

TPU Channels – These signals provide each TPU with 16 input/output programmable timed events.

2.3.4.2 T2CLK

Pin Name: a_t2clk (1 pin for first TPU), b_t2clk (1 pin for second TPU)

T2CLK – This signal is used to clock or gate the timer count register 2 (TCR2) within the TPU. This pin is an output-only in special test mode.

2.3.5 QADC_A/QADC_B PADS

2.3.5.1 ETRIG[1:2]

Pin Name: etrig1 - etrig2

ETRIG – These are the external trigger inputs to the QADC_A and QADC_B modules. ETRIG[1] can be configured to be used by both QADC_A and QADC_B. Likewise, ETRIG[2] can be used for both QADC_B and QADC_A. The trigger input pins are associated with the scan queues.

2.3.5.2 AN[0]/ANW/PQB[0]_[A:B]

Pin Name: a_an0_anw_pqb0 (1 pin for first QADC), b_an0_anw_pqb0 (1 pin for second QADC)

Analog Channel (AN0) – Internally multiplexed input-only analog channels. Passed on as a separate signal to the QADC.

Multiplexed Analog Input (ANW) – Externally multiplexed analog input.

Port (PQB0) – Input-only port. This is a 5-V input. This path is synchronized in the pad. The input is level-shifted before it is sent internally to the QADC.

2.3.5.3 AN[1]/ANX/PQB[1]_[A:B]

Pin Name: a_an1_anx_pqb1 (1 pin for first QADC), b_an1_anx_pqb1 (1 pin for second QADC)

Analog Channel (AN1) – Internally multiplexed input-only analog channels. Passed on as a separate signal to the QADC.

Multiplexed Analog Input (ANX) – Externally multiplexed analog input.

Port (PQB1) – Input-only port. This is a 5-V input. This path is synchronized in the pad. The input is level-shifted before being sent internally to the QADC.

2.3.5.4 AN[2]/ANY/PQB[2]_[A:B]

Pin Name: a_an2_any_pqb2 (1 pin for first QADC), b_an2_any_pqb2 (1 pin for second QADC)

Analog Channel (AN2) – Internally multiplexed input-only analog channel. The input is passed on as a separate signal to the QADC.

Multiplexed Analog Input (ANY) – Externally multiplexed analog input.

Port (PQB2) – Input-only port. This is a 5-V input. This path is synchronized in the pad. The input is level-shifted before it is sent internally to the QADC.

2.3.5.5 AN[3]/ANZ/PQB[3]_[A:B]

Pin Name: a_an3_anz_pqb3 (1 pin for first QADC), b_an3_anz_pqb3 (1 pin for second QADC)

Analog Input (AN3) – Internally multiplexed input-only analog channel. The input is passed on as a separate signal to the QADC.

Multiplexed Analog Input (ANZ) – Externally multiplexed analog input.

Port (PQB3) – Input-only port. This is a 5-V input. This path is synchronized in the pad. The input is level-shifted before it is sent internally to the QADC.

2.3.5.6 AN[48:51]/PQB[4:7]_[A:B]

Pin Name: a_an48_pqb4 – a_an51_pqb7 (4 pins for first QADC), b_an48_pqb4 – b_an51_pqb7 (4 pins for second QADC).

Analog Input (AN[48:51]) – Analog input channel. The input is passed on as a separate signal to the QADC.

Port (PQB[4:7]) – Input-only port. Has a synchronizer with an input enable and clock. The input is level-shifted before it is sent internally to the QADC.

2.3.5.7 AN[52:54]/MA[0:2]/PQA[0:2]_[A:B]

Pin Name: a_an52_ma0_pqa0 – a_an54_ma2_pqa2 (3 pins for first QADC), b_an52_ma0_pqa0 – b_an54_ma2_pqa2 (3 pins for second QADC).

Analog Input (AN[52:54]) – Input-only. The input is passed on as a separate signal to the QADC.

Multiplexed Address (MA[0:2]) – Output. Provides a three-bit multiplexed address output to the external multiplexer chip to allow selection of one of the eight inputs.

Port (PQA[0:2]) – Bi-directional.

2.3.5.8 AN[55:59]/PQA[3:7]_[A:B]

Pin Name: a_an55_pqa3 - a_an59_pqa7 (5 pins for first QADC), b_an55_pqa3 – b_an59_pqa7 (5 pins for second QADC).

Analog Input (AN[55:59]) – Input-only. The input is passed on as a separate signal to the QADC.

Port (PQA[3:7]) – Bi-directional.

2.3.5.9 VRH

Pin Name: vrh

VRH – Input pin for high reference voltage for the QADC_A and QADC_B modules.

2.3.5.10 VRL

Pin Name: vrl

VRL – Input pin for low reference voltage for the QADC_A and QADC_B modules.

2.3.5.11 VDDA

Pin Name: vdda

VDDA – Power supply input to analog subsystems of the QADC_A and QADC_B modules.

2.3.5.12 VSSA

Pin Name: vssa

VSSA – Input. Ground level for analog subsystems of the QADC_A and QADC_B modules.

2.3.6 TOUCAN_A/TOUCAN_B PADS

2.3.6.1 CNTX0_[A:B]

Pin Name: a_cntx0 (1 pin for first CAN), b_cntx0 (1 pin for second CAN)

TouCAN Transmit Data 0 – This signal is the serial data output.

2.3.6.2 CNRX0_[A:B]

Pin Name: a_cnrx0 (1 pin for first CAN), b_cnrx0 (1 pin for second CAN)

TouCAN Receive Data – This signal furnishes serial input data.

2.3.7 CMF PADS

2.3.7.1 EPEE

Pin Name: epee

EPEE – Input. This control signal will externally control the program or erase operations.

2.3.7.2 VPP

Pin Name: vpp

VPP – Input. Flash supply voltage (5-V supply) used during program and erase operations of the CMF.

2.3.7.3 VDDF

Pin Name: vddf

VDDF – Flash core voltage input (3-V supply). This separate supply voltage is needed in order to reduce noise in the read path of CMF.

2.3.7.4 VSSF

Pin Name: vssf

VSSF – Flash core zero supply input. This separate supply is needed in order to reduce noise in the read path of CMF.

2.3.8 GLOBAL POWER SUPPLIES

2.3.8.1 VDDL

Pin Name: vddl

VDDL – 3-V voltage supply input.

2.3.8.2 VDDH

Pin Name: vddh

VDDH – 5-V voltage supply input.

2.3.8.3 VDDI

Pin Name: vddi

VDDI – 3-V voltage supply input for internal logic.

2.3.8.4 VSSI

Pin Name: vssi

VSSI – Zero supply input for internal logic. In packaged devices, VSSI is not a separate input from VSS.

2.3.8.5 KAPWR

Pin Name: kapwr

Keep-Alive Power – 3-V voltage supply input for the oscillator and keep-alive registers.

2.3.8.6 VDDSRAM

Pin Name: vddsram

SRAM Keep-Alive Power – 3-V voltage supply input for the SRAM.

2.3.8.7 VSS

Pin Name: vss

VSS – Ground level reference input.

2.4 Reset State

All input pins, with the exception of the power supply and clock related pins, are “weakly pulled” to a value during reset by a 130-microampere resistor based on certain conditions. In reset state all I/O pins become inputs, and all outputs except clkout, hreset_b, sreset_b, will be pulled only by the pull-up/pull-down.

2.4.1 Pin Functionality Out of Reset

The functionality out of reset of some pins that support multiple functionality is defined in the SIUMCR through the reset configuration word. For details on which multiplexed pins are configured by the reset configuration word and how they are configured, refer to **7.5.2 Hard Reset Configuration Word**.

The 3-V related pins have selectable output buffer drive strengths which are controlled by the COM[0] bit in the USIU’s system clock and reset control register (SCCR). The control is as follows:

0 = 3-V bus pins full drive (50-pF load)*

1 = 3-V bus pins reduced drive (25-pF load)

* The bus pin drive selectability definition is inverted from the selectability of the pin control in the PDMCR register (for the TPU, QADC, USIU (SGPIO), QSPI, TouCAN, QSCI, and MIOS pins).

2.4.2 Pad Module Configuration Register (PDMCR)

The slew rate and weak pull-up/pull-down characteristics of some pins are controlled by bits in the PDMCR. This register resides in the SIU memory map. The contents of the PDMCR are illustrated below. The $\overline{\text{PORESET}}$ signal resets all the PDMCR bits asynchronously.

PDMCR – Pad Module Configuration Register **0x2F C03C**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLRC0	SLRC 1	SLRC 2	SLRC 3	Reserved		PRDS	SPRDS	FTPU_PU[1]	Reserved						
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE:
1. FTPU_PU is only available on mask set K62N and later.

Table 2-3 PDMCR Bit Descriptions

Bit(s)	Name	Description
0	SLRC0	SLRC0 controls the slew rate of the following modules: TPU, QADC, USIU (SGPIO). 0 = Slow slew rate for pins. Controls slew rate pins of 200 ns. 1 = Normal slew rate for pins
1	SLRC1	SLRC1 controls the slew rate of the QSPI and TouCAN modules. 0 = Slow slew rate for pins. Controls slew rate pins of 50 ns. 1 = Normal slew rate for pins
2	SLRC2	SLRC2 controls the slew rate of the QSCI module. 0 = Slow slew rate for pins. Controls slew rate pins of 200 ns. 1 = Normal slew rate for pins
3	SLRC3	SLRC3 controls the slew rate of the MIOS module. 0 = Slow slew rate for pins. Controls slew rate pins of 200 ns. 1 = Normal slew rate for pins
4:5	—	Reserved
6	PRDS	The PRDS bit is used to enable or disable the weak pull-up/pull-down devices in the pads related to SGPIO and all pads related to IMB modules. **Table 2-4** illustrates which pins are affected by PRDS. 0 = Enable pull-up/pull-down devices 1 = Disable pull-up/pull-down devices

Table 2-3 PDMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
7	SPRDS	The SPRDS bit is used to enable or disable the weak pull-up/pull-down devices in special 3-V only bus pads. **Table 2-4** illustrates which pins are affected by SPRDS. For more details on how this bit affects the pins see **2.4.7 Special Pull Resistor Disable Control (SPRDS)**. 0 = Enable pull-up/pull-down devices 1 = Disable pull-up/pull-down devices
8	FTPU_PU	Follow TPU Pull-Up — Controls the pull-up devices for all T2CLK pins. FTPU_PU is only available on mask set K62N and later. 0 = Pull-ups are active when the pins are defined as inputs 1 = Pull-ups for the TPU T2CLK pins are enabled or disabled based on the state of PRDS
9:31	—	Reserved

2.4.3 Pin State During Reset

During reset, the functionality of some pins is undetermined. Their functionality is based on the bits in the SIUMCR. Since the SIUMCR bits are undetermined during reset, there is no way of predicting how the pins will function. However, the pins must not cause any spurious conditions or consume an excessive amount of power during reset. To prevent these conditions, the pins need to have a defined reset state. **Table 2-4** describes the reset state of the pins based on pin functionality.

All pins are initialized to a “reset state” during reset. This state remains active until reset is negated or until software disables the pull-up or pull-down device based on the pin functionality. Upon assertion of the corresponding bits in the pin control registers and negation of reset, the pin acquires the functionality that was programmed.

2.4.4 Power-On Reset and Hard Reset

Power-on reset and hard reset affect the functionality of the pins out of reset. (During soft reset, the functionality of the pins is unaltered.) Upon assertion of the power-on reset signal ($\overline{\text{PORESET}}$) the functionality of the pin is not yet known. The pull-up or pull-down resistors are enabled. The reset configuration word configures the system, and towards the end of reset the pin functionality is known. Based upon pin functionality, the pull-up or pull-down devices are either disabled immediately at the negation of reset or remain enabled.

Hard reset can occur at any time, and there may be a bus cycle pending. For this reason, the bits in PDMCR that control the enabling and disabling of the pull-up or pull-down resistors in the pads are set or reset synchronously. ($\overline{\text{PORESET}}$ affects these bits asynchronously.) This causes the pull-up or pull-down resistors to be enabled at a time when they do not cause contention on the pins and are disabled before they can cause any contention on the pins.

2.4.5 Pull-Up and Pull-Down Enable and Disable for 5-V Only Pins

For 5-V only pins, the enabling and disabling of the pull-up and pull-down devices is controlled by the PRDS bit in PDMCR. If the bit is negated, the devices are active. If the bit is asserted, the devices are inactive.

2.4.6 Pull-Up and Pull-Down Enable and Disable for 3-V / 5-V Multiplexed Pins

Two signals are needed to enable or disable the pull-up/pull-down devices in the 3-V / 5-V multiplexed pads:

- The PRDS signal
- An encoded 3-V / 5-V select

2.4.6.1 $\overline{\text{PRDS}}$ Signal

The $\overline{\text{PRDS}}$ signal is derived from the PRDS bit in the PDMCR. A single signal controls all affected pads (all pads related to SGPIO and all pads related to the UIMB modules). The bit is reset by default (pull-ups enabled) and must be explicitly set by software after reset. The bit is reset immediately following power-on reset and by hard reset after data coherency. This bit is not affected by soft reset.

2.4.6.2 Encoded 3-V / 5-V Select

This signal selects between the 3-V functionality and the 5-V functionality of the pin. 5-V operation is selected until the function of the pin is determined (based on the reset configuration word) and $\overline{\text{PORESET}}$ is negated. At this point the 3-V / 5-V select signal assumes the intended state (high for 5 V and low for 3 V).

Upon hard reset assertion, if the 3-V / 5-V select line is in 3-V select mode, it remains in this mode until any external bus access completes. After this the 3-V / 5-V select signal switches to 5-V mode to enable the pull-ups. This ensures that there is no contention on the bus due to the pull-up being enabled. This signal is not affected by soft reset.

Each pad group has a 3-V / 5-V select signal. Internal to the pad, logic combines these signals to control the pull-up.

2.4.6.3 Examples

The combination of this 3-V / 5-V select signal and the resistor disable signal enables or disables the pull-up.The logic to enable the pull-up is:

pull_enable = $\overline{\text{PRDS}}$ & 3-V / 5-V select

For example, if a pin is configured as a GPIO pin (5 V), the 3-V / 5-V select is high throughout reset. This causes the pull-up to be enabled. At the end of reset, the 3-V / 5-V select line remains high. The PRDS is high by default until cleared by software. This causes the pull-up to be enabled until software clears the PRDS bit in the PDMCR.

If a pin is configured as a bus pin (3 V), the 3 V / 5 V remains high throughout reset. This causes the pull-up to be enabled. At the end of reset, the 3-V / 5-V select line goes low. This causes the pull-up to be disabled, preventing any power loss if the MCU starts fetching from external memory immediately out of reset.

2.4.7 Special Pull Resistor Disable Control (SPRDS)

For the pins that support debug and opcode-tracking functionality, the pull-up and pull-down resistors are controlled by the SPRDS signal, which is somewhat like the encoded 3-V / 5-V select. During reset this signal is used synchronously to enable the pull-up resistors in the pads. On negation of reset, based on which functionality is selected for the pins, this signal is set to disable the pull-up resistors or remains held in its reset state to indicate that the pull-ups are disabled only when the output driver is enabled.

For example, if a pin is configured as a bus arbitration pin, The SPRDS signal remains low throughout reset. This causes the pull-up to be enabled. When reset is released, SPRDS remains low. The output enable for the driver is negated by default. When the output driver is enabled, the pull-up is disabled.

When a pin is configured as an opcode-tracking or debug pin, SPRDS remains low throughout reset. This causes the pull-up to be enabled. When reset is released, SPRDS is asserted. This disables the pull-up resistor immediately. The output driver drives the pin to the required state after reset.

2.4.8 Pin Reset States

Table 2-4 summarizes the reset states of all the pins on the MPC555 / MPC556.

Table 2-4 Pin Reset State

Pin	Function	Port	Voltage	Reset State
USIU				
ADDR[8:31]/ SGPIOA[8:31]	ADDR[8:31]	I/O	3 V	PU5 until reset negates[1]
	SGPIOA[8:31]	IO	5 V	PU5 until PRDS is set
DATA[0:31]/ SGPIOD[0:31]	DATA[0:31]	I/O	3 V	PD until reset negates
	SGPIOD[0:31]	I/O	5 V	PD until PRDS is set
$\overline{\text{IRQ}}$[0]/SGPIOC[0]	$\overline{\text{IRQ}}$[0]	I	3 V	PU5 until reset negates[1]
	SGPIOC[0]	I/O	5 V	PU5 until PRDS is set
$\overline{\text{IRQ}}$[1]/ $\overline{\text{RSV}}$/SGPIOC[1]	$\overline{\text{IRQ}}$[1]	I	3 V	PU5 until reset negates[1]
	$\overline{\text{RSV}}$	O	3 V	PU5 until reset negates[1]
	SGPIOC[1]	I/O	5 V	PU5 until PRDS is set
$\overline{\text{IRQ}}$[2]/ $\overline{\text{CR}}$/SGPIOC[2]/ MTS	$\overline{\text{IRQ}}$[2]	I	3 V	PU5 until reset negates[1]
	$\overline{\text{CR}}$	I	3 V	PU5 until reset negates[1]
	SGPIOC[2]	I/O	5 V	PU5 until PRDS is set
	MTS	O	3 V	PU5 until PRDS negates
$\overline{\text{IRQ}}$[3]/ $\overline{\text{KR}}$, $\overline{\text{RETRY}}$/ SGPIOC[3]	$\overline{\text{IRQ}}$[3]	I	3 V	PU5 until reset negates[1]
	$\overline{\text{KR}}$, $\overline{\text{RETRY}}$	I/O	3 V	PU5 when driver not enabled[2]
	SGPIOC[3]	I/O	5 V	PU5 until PRDS is set
$\overline{\text{IRQ}}$[4]/ AT[2]/ SGPIOC[4]	$\overline{\text{IRQ}}$[4]	I	3 V	PU5 until reset negates[1]
	AT[2]	O	3 V	PU5 until reset negates[1]
	SGPIOC[4]	I/O	5 V	PU5 until PRDS is set

Table 2-4 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State
$\overline{IRQ}$[5]/SGPIOC[5]/ MODCK[1][3]	$\overline{IRQ}$[5]	I	3 V	PU5 until reset negates[1]
	SGPIOC[5]	I/O	5 V	PU5 until PRDS is set
	MODCK[1]	I	3 V	PU5 until reset negates[1]
$\overline{IRQ}$[6:7]/ MODCK[2:3][3]	$\overline{IRQ}$[6:7]	I	3 V	PU3 until SPRDS is set
	MODCK[2:3]	I	3 V	PU3 until reset negates
TSIZ[0:1]	TSIZ[0:1]	I/O	3 V	PD when driver not enabled or until SPRDS is set
RD/$\overline{WR}$	RD/$\overline{WR}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
$\overline{BURST}$	$\overline{BURST}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
$\overline{BDIP}$	$\overline{BDIP}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
$\overline{TS}$[4]	$\overline{TS}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
$\overline{TA}$[4]	$\overline{TA}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
$\overline{TEA}$	$\overline{TEA}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set An external pull-up is required in order to negate the pin in appropriate time
$\overline{RSTCONF}$/TEXP[3]	$\overline{RSTCONF}$	I	3 V	PU3 when driver not enabled or until SPRDS is set
	TEXP	O	3 V	
$\overline{OE}$	$\overline{OE}$	O	3 V	PU3 until reset negates
$\overline{BI}$/$\overline{STS}$	$\overline{BI}$4	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
	$\overline{STS}$	O	3 V	
$\overline{CS}$[0:3]	$\overline{CS}$[0:3]	O	3 V	PU3 until reset negates
$\overline{WE}$[0:3]/$\overline{BE}$[0:3]/ AT[0:3]	$\overline{WE}$[0:3]/$\overline{BE}$[0:3]	O	3 V	PU3 when driver not enabled or until SPRDS is set
	AT[0:3]	O	3 V	
$\overline{PORESET}$[3]	$\overline{PORESET}$	I	3 V	—
$\overline{HRESET}$[3]	$\overline{HRESET}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set An external pull-up is required in order to negate the pin in appropriate time
$\overline{SRESET}$[3]	$\overline{SRESET}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set An external pull-up is required in order to negate the pin in appropriate time
SGPIOC[6]/ FRZ/ $\overline{PTR}$	SGPIOC[6]	I/O	5 V	PU5 until PRDS is set
	FRZ	O	3 V	PU5 until reset negates[1]
	$\overline{PTR}$	O	3 V	PU5 until reset negates[1]

Table 2-4 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State
SGPIOC[7]/ $\overline{\text{IRQOUT}}$/LWP[0]	SGPIOC[7]	I/O	5 V	PU5 until PRDS is set
	$\overline{\text{IRQOUT}}$	O	3 V	PU5 until reset negates[1]
	LWP[0]	O	3 V	PU5 until reset negates[1]
$\overline{\text{BG}}$/ VF[0]/ LWP[1]	$\overline{\text{BG}}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
	VF[0]	O	3 V	
	LWP[1]	O	3 V	
$\overline{\text{BR}}$/ VF[1]/ IWP[2]	$\overline{\text{BR}}$	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
	VF[1]	O	3 V	
	IWP[2]	O	3 V	
$\overline{\text{BB}}$/ VF[2]/ IWP[3]	$\overline{\text{BB}}$[4]	I/O	3 V	PU3 when driver not enabled or until SPRDS is set
	VF[2]	O	3 V	
	IWP[3]	O	3 V	
IWP[0:1]/ VFLS[0:1]	IWP[0:1]	O	3 V	PU3 until reset negates
	VFLS[0:1]	O	3 V	
TMS	TMS	I	3 V	PU3 until SPRDS is set
TDI/ DSDI	TDI	I	3 V	PU3 until SPRDS is set
	DSDI	I	3 V	
TCK/ DSCK	TCK	I	3 V	PD until SPRDS is set
	DSCK	I	3 V	
TDO/ DSDO	TDO	O	3 V	PU3 until reset negates
	DSDO	O	3 V	
$\overline{\text{TRST}}$	$\overline{\text{TRST}}$	I	3 V	PU3 until SPRDS is set
XTAL[3]	XTAL	I	3 V	—
EXTAL[3]	EXTAL	I	3 V	—
XFC	XFC	I	3 V	—
CLKOUT	CLKOUT	O	3 V	—
EXTCLK[3]	EXTCLK	I	3 V	—
ENGCLK/ BUCLK	ENGCLK	O	5 V	—
	BUCLK	O	5 V	—

Table 2-4 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State
QSMCM				
PCS0/ $\overline{SS}$/ QGPIO[0]	PCS0	I/O	5 V	PU5 until PRDS is set
	$\overline{SS}$	I/O	5 V	
	QGPIO[0]	I/O	5 V	
PCS[1:3]/ QGPIO[1:3]	PCS[1:3]	I/O	5 V	PU5 until PRDS is set
	QGPIO[1:3]	I/O	5 V	
MISO/ QGPIO[4]	MISO	I/O	5 V	PU5 until PRDS is set
	QGPIO[4]	I/O	5 V	
MOSI/ QGPIO[5]	MOSI	I/O	5 V	PU5 until PRDS is set
	QGPIO[5]	I/O	5 V	
SCK/ QGPIO[6]	SCK	I/O	5 V	PU5 until PRDS is set
	QGPIO[6]	I/O	5 V	
TXD[1:2]/ QGPO[1:2]	TXD[1:2]	O	5 V	PU5 until PRDS is set
	QGPO[1:2]	O	5 V	
RXD[1:2]/QGPI[1:2]	RXD[1:2]	I	5 V	PU5 until PRDS is set
	QGPI[1:2]	I	5 V	
ECK	ECK	I	5 V	PU5 until PRDS is set
MIOS				
MDA[4:13]	MDA[4:13]	I/O	5 V	PU5 until PRDS is set
MPWM[0:3], [16:19]	MPWM[0:3], [16:19]	I/O	5 V	PU5 until PRDS is set
VF[0:2]/ MPIO32B[0:2]	VF[0:2]	O	3 V	PU5 until PRDS is set
	MPIO32B[0:2]	I/O	5 V	
VFLS[0:1]/ MPIO32B[3:4]	VFLS[0:1]	O	3 V	PU5 until PRDS is set
	MPIO32B[3:4]	I/O	5 V	
MPIO32B[5:15]	MPIO32B[5:15]	I/O	5 V	PU5 until PRDS is set
TPU_A/TPU_B				
A: TPUCH[0:15]	TPUCH[0:15]	I/O	5 V	PU5 until PRDS is set
A: T2CLK	T2CLK	I/O	5 V	PU5 when driver not enabled[2]
B: TPUCH[0:15]	TPUCH[0:15]	I/O	5 V	PU5 until PRDS is set
B: T2CLK	T2CLK	I/O	5 V	PU5 when driver not enabled[2]
QADC_A/QADC_B				
ETRIG[1:2]	ETRIG[1:2]	I	5 V	PD
A: AN0/ANW/PQB0	AN0	I	5 V	PU5 until PRDS is set
	ANW	I	5 V	PU5 until PRDS is set
	PQB0	I	5 V	PU5 until PRDS is set
A: AN1/ANX/PQB1	AN1	I	5 V	PU5 until PRDS is set
	ANX	I	5 V	PU5 until PRDS is set
	PQB1	I	5 V	PU5 until PRDS is set
A: AN2/ANY/PQB2	AN2	I	5 V	PU5 until PRDS is set
	ANY	I	5 V	PU5 until PRDS is set
	PQB2	I	5 V	PU5 until PRDS is set

Table 2-4 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State
A: AN3/ANZ/PQB3	AN3	I	5 V	PU5 until PRDS is set
	ANZ	I	5 V	PU5 until PRDS is set
	PQB3	I	5 V	PU5 until PRDS is set
A: AN[48:51]/ PQB[4:7]	AN[48:51]	I	5 V	PU5 until PRDS is set
	PQB[4:7]	I	5 V	PU5 until PRDS is set
A: AN[52:54]/ MA[0:2]/PQA[0:2]	AN[52:54]	I	5 V	PU5 until PRDS is set
	MA[0:2]	I	5 V	PU5 until PRDS is set
	PQA[0:2]	I/O	5 V	PU5 until PRDS is set
A: AN[55:56]]/ PQA[3:4]	AN[55:56]	I	5 V	PU5 until PRDS is set
	PQA[3:4]	I/O	5 V	PU5 until PRDS is set
A: AN[57:59]/ PQA[5:7]	AN[57:59]	I	5 V	PU5 until PRDS is set
	PQA[5:7]	I/O	5 V	PU5 until PRDS is set
B: AN0/ANW/PQB0	AN0	I	5 V	PU5 until PRDS is set
	ANW	I	5 V	PU5 until PRDS is set
	PQB0	I	5 V	PU5 until PRDS is set
B: AN1/ANX/PQB1	AN1	I	5 V	PU5 until PRDS is set
	ANX	I	5 V	PU5 until PRDS is set
	PQB1	I	5 V	PU5 until PRDS is set
B: AN2/ANY/PQB2	AN2	I	5 V	PU5 until PRDS is set
	ANY	I	5 V	PU5 until PRDS is set
	PQB2	I	5 V	PU5 until PRDS is set
B: AN3/ANZ/PQB3	AN3	I	5 V	PU5 until PRDS is set
	ANZ	I	5 V	PU5 until PRDS is set
	PQB3	I	5 V	PU5 until PRDS is set
B: AN[48:51]/ PQB[4:7]	AN[48:51]	I	5 V	PU5 until PRDS is set
	PQB[4:7]	I	5 V	PU5 until PRDS is set
B: AN[52:54]/ MA[0:2]/PQA[0:2]	AN[52:54]	I	5 V	PU5 until PRDS is set
	MA[0:2]	I	5 V	PU5 until PRDS is set
	PQA[0:2]	I/O	5 V	PU5 until PRDS is set
B: AN[55:56]/ PQA[3:4]	AN[55:56]	I	5 V	PU5 until PRDS is set
	PQA[3:4]	I/O	5 V	PU5 until PRDS is set
B: AN[57:59]/ PQA[5:7]	AN[57:59]	I	5 V	PU5 until PRDS is set
	PQA[5:7]	I/O	5 V	PU5 until PRDS is set
VRH	VRH	I	5 V	—
VRL	VRL	I	—	—
VDDA	VDDA	I	5 V	—
VSSA	VSSA	I	—	—
TouCAN_A/TouCAN_B				
A: CNTX0	A_CNTX0	O	5 V	PU5 until PRDS is set
B: CNTX0	B_CNTX0	O	5 V	PU5 until PRDS is set
A: CNRX0	A_CNRX0	I	5 V	PU5 until PRDS is set
B: CNRX0	B_CNRX0	I	5 V	PU5 until PRDS is set

Table 2-4 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State
CMF				
EPEE	EPEE	I	3 V	PD
VPP	VPP	I	5 V	—
VDDF	VDDF	I	3 V	—
VSSF	VSSF	I	3 V	—
Global Power Supplies				
VDDL	VDDL	I	3 V	—
VDDH	VDDH	I	5 V	—
VDDI	VDDSI	I	3 V	—
VSSI	VSSI	I	3 V	—
KAPWR[3]	KAPWR	I	3 V	—
VDDSRAM	VDDSRAM	I	3 V	—
VDDSYN	VDDSYN	I	3 V	—
VSS	VSS	I	—	—
VSSSYN	VSSSYN	I	3 V	—

NOTES:

1. During reset, the output enable to the pad driver is negated and the PU3/PU5 is active. After reset is negated, the output enable is continuously enabled and the PU3 is disabled. The driver is responsible for driving a valid state on the pin.
2. Pull-up/pull-down is active when pin is defined as an input and/or during reset; therefore, output enable is negated. This also means that external pull-up/pull-down is *not* required unless specified.
3. These pins are powered by KAPWR (Keep-Alive Power Supply).
4. This pin is an active negate signal and may need an external pull-up resister.

2.5 Pad Types

There are different pad types based on functional characteristics. Even pads with the same functionality may be different due to different electrical characteristics. All 5-V inputs have hysteresis. There is no synchronization in the pads; it is all in the modules.

2.5.1 Pad Interface Signals

The pad interface consists of an internal interface and an external interface. The external interface is to the pin. The internal interface is the set of signals that interface the pad to the chip's internal logic. The following internal interface signals are used:

- Data – The line driven from an internal module of the chip to the pad. For bi-directional pins, the internal interface may be a single line for both input and output or two separate paths for input and output. The descriptions of individual pad types specify which.
- 3-V / 5-V select – Selects a 3-V or 5-V driver, for pads that support both. This signal is driven from the USIU.
- Output enable (OE) – Enables the output driver. For 3-V / 5-V pads, the appropriate driver is enabled based on the pin functionality selected.
- Input enable – Enables the receiver. For 3-V / 5-V pads, the appropriate receiver is enabled based on the pin functionality selected.

- Drive select – Selects the drive strength of the pad. For example, data pin drivers can be configured to drive a 25-pF load or a 50-pF load.
- Synchronizer clock – Some pins have synchronizer logic to handle metastable signals at the input of a pin. For pads that have synchronizers and support synchronized or normal data input, the corresponding interface signals to the internal logic are “Normal Data In” and “Sync Data In.”
- Slew rate control – GPIO pins have slow slew rates, with edge rates in the range of 90 ns to 600 ns. The slew rate and weak pull-up/pull-down characteristics of these pins are controlled by bits in the PDMCR, see **2.4.2 Pad Module Configuration Register (PDMCR)**. For a description of PDMCR bits SLRC[0:3] that have controllable slew rates, see **Table 2-3**.
- Hysteresis input – Slow pads contains hysteresis input buffers to reduce the sensitivity to noises. The input hyst_sel is used to configure the pad to provide hysteresis according to the pad configuration.
- Open drain enable – For selected 3-V / 5-V pads, this signal determines the type of drive (open drain or totem pole) seen at the pin.
- Pull resistor disable select (PRDS) – Reflects the state of the PRDS bit in the pad module configuration register (PDMCR). This signal controls the pull-up/pull-down resistor for the SGPIO pins and the pins for the modules on the UIMB.
- Special pull resistor disable select (SPRDS) – Reflects the state of the SPRDS bit in the PDMCR. For pins that support bus arbitration functionality multiplexed with opcode-tracking and debug functionality, this signal controls the pull-up resistors.
- Analog – Analog input signals to the QADC. The corresponding digital interface signals are referred to as “Dig. In” and Dig. Out”.
- JTAG – Joint Test Access Group related signals that are used for connectivity tests at the board level. These signals are not shown in the pad block diagrams in this section. In addition, the effect of the pull-up/pull-down resistors is not illustrated in the pad block diagrams.

These interface signals are referred to in the following pad descriptions and shown in the pad diagrams.

2.5.2 Three-Volt Output Pad

The output driver of a 3-V output-only pad can be configured to drive a 25-pF or 50-pF load. There are two subtypes: one with a pull-up device and the other with a pull-down device. The SPRDS and OE signals enable the pull-up and pull-down resistors.

2.5.2.1 Type A Interface

This pad has a pull-up device to 3 V which can be conditionally turned off based on the value placed on OE. For a totem pole (push pull) pin with no three-state drive time, the OE can be connected to VDD, indicating a continuous drive. For a continuous drive, the pull-up can be disabled.

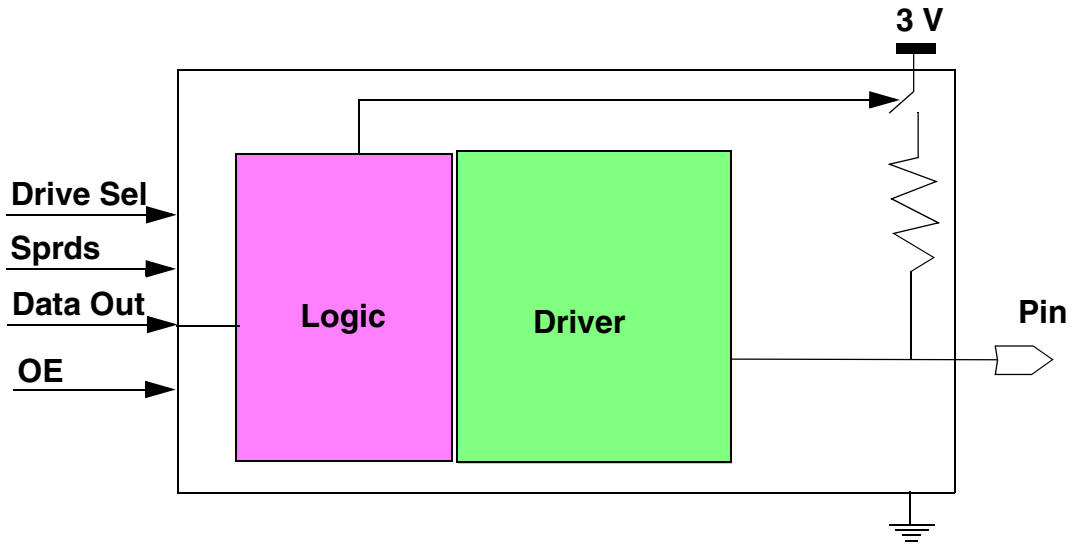


Figure 2-3 Type A Interface

2.5.2.2 Type B Interface (Clock Pad)

The pad has a capability to select the buffer for the appropriate load (45 or 90 pF). The OE input drives the totem pole output or three-states the output.

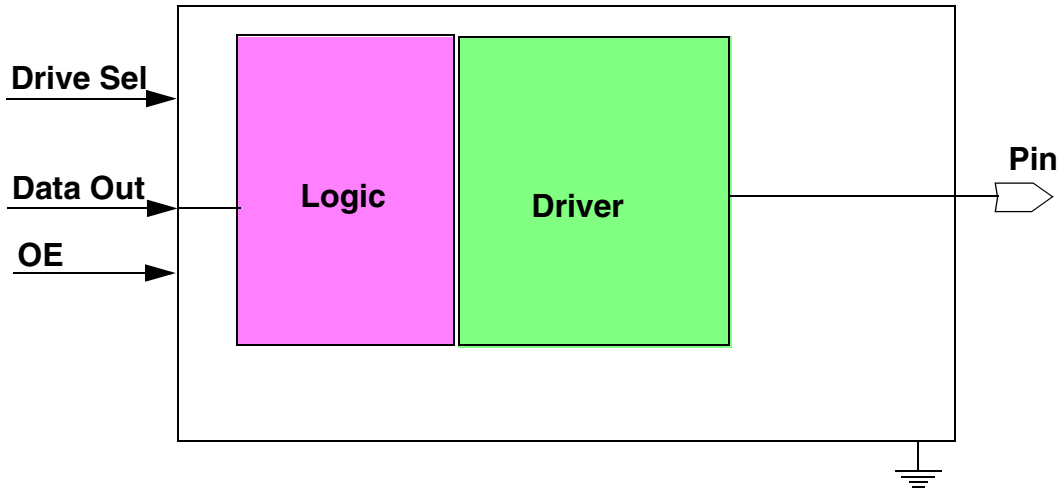


Figure 2-4 Type B Interface

2.5.3 Three-Volt Input Pad

Four subtypes are defined for the 3-V input-only pad: one with a pull-up resistor, one with a pull-up resistor and with or without hysteresis in the receiver, one with hysteresis

(no resistor), and one with a pull-down resistor. The SPRDS signal may disable the pull-up or pull-down resistor.

2.5.3.1 Type C Interface

The type C interface has a 3-V input with a pull-up resistor.

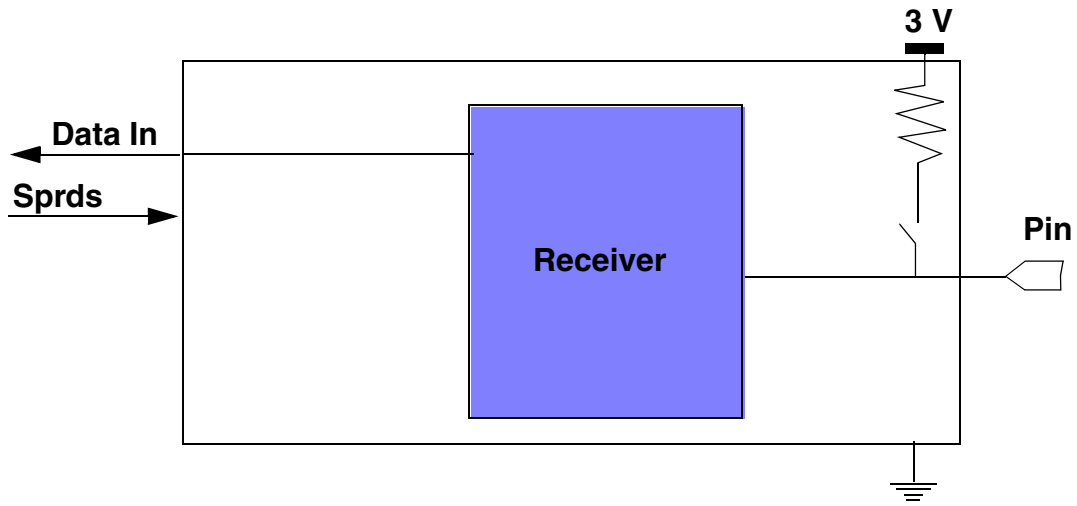


Figure 2-5 Type C Interface

2.5.3.2 Type CH Interface

Pad type CH has a 3-V input with hysteresis and a pull-up resistor. The hyst_sel signal selects the receiver with or without hysteresis.

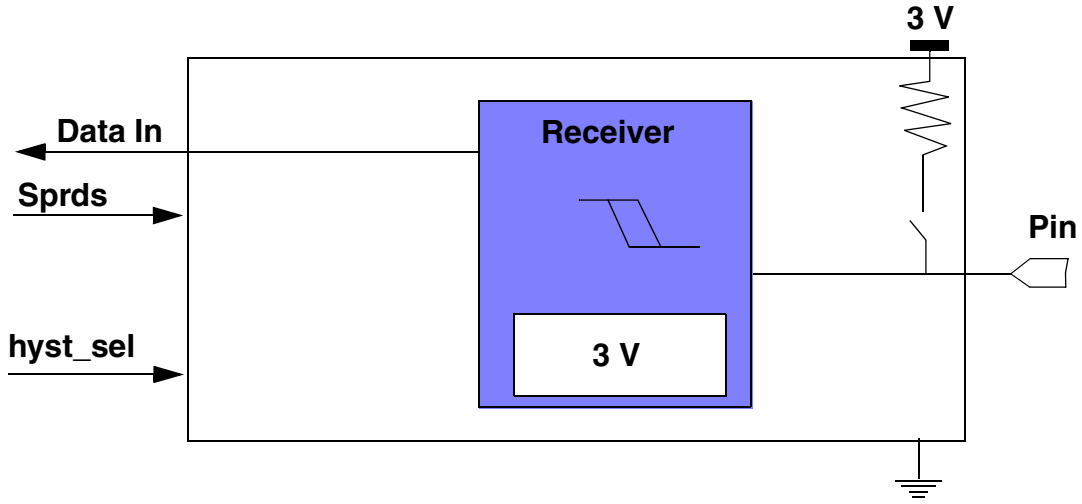


Figure 2-6 Type CH Interface

2.5.3.3 Type CNH Interface

The CNH pad type has a 3-V input with hysteresis but no pull-up or pull-down device.

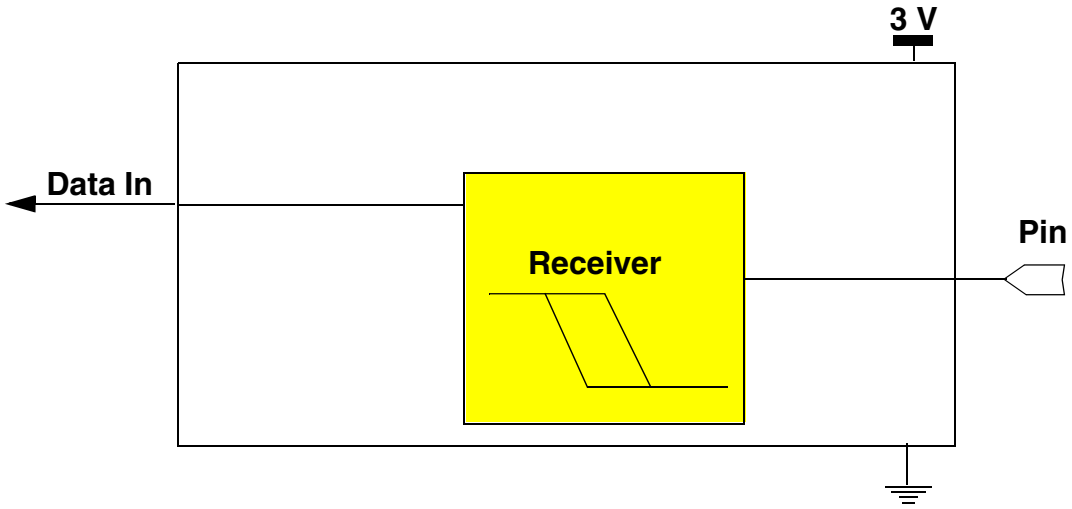


Figure 2-7 Type CNH Interface

2.5.3.4 Type D Interface

This type of pad has a 3-V input and an internal pull-down resistor.

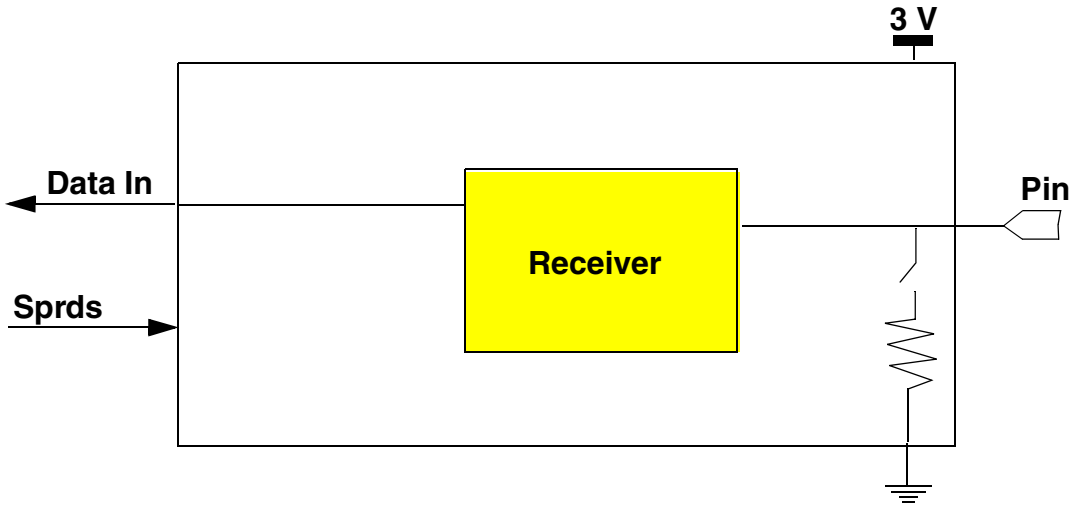


Figure 2-8 Type D Interface

2.5.4 Three-Volt Input/Output Pad

This is a 3-V bi-directional pad with a pull-up device. The drive strength for the output driver can be configured for either a 25-pF or a 50-pF load. The SPRDS and OE signals control the pull-up devices.

2.5.4.1 Type E Interface

In this pad type the data interface to the internal logic has separate paths for input and output. This pad also has a open drain enable input. For totem pole driven outputs, the signal is connected to VSS to disable the open-drain drive.

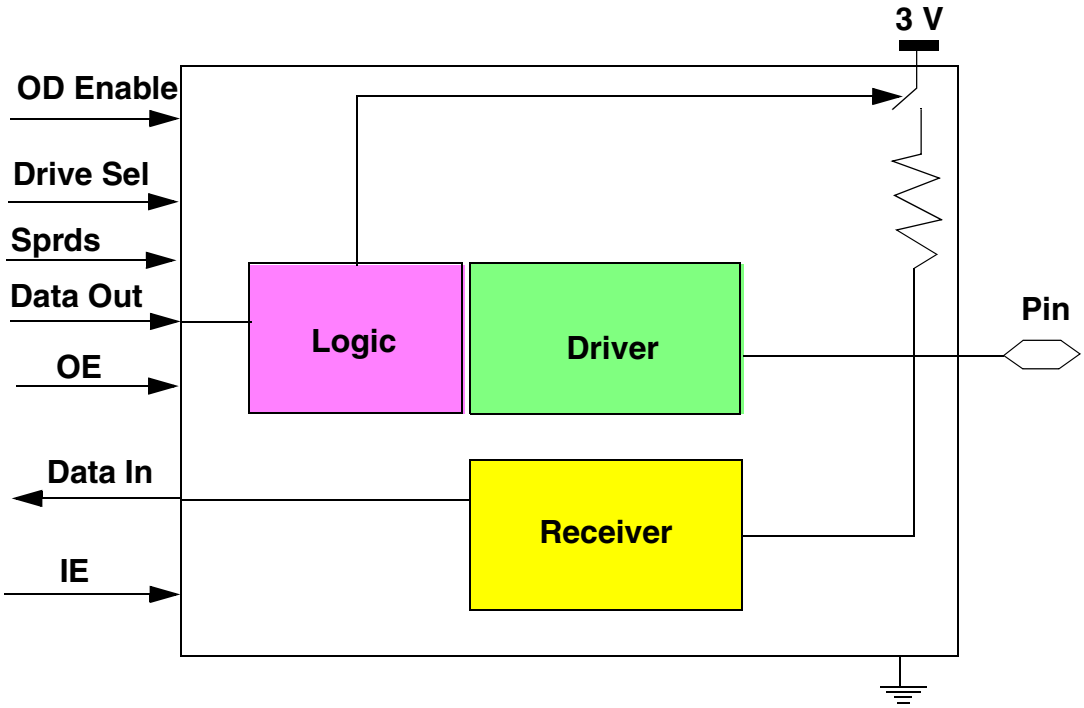


Figure 2-9 Type E Interface

2.5.4.2 Type EOH Interface

In this pad type the data interface to the internal logic has separate paths for input and output. The receiver has hysteresis. The pull-up is active when the driver is not enabled.

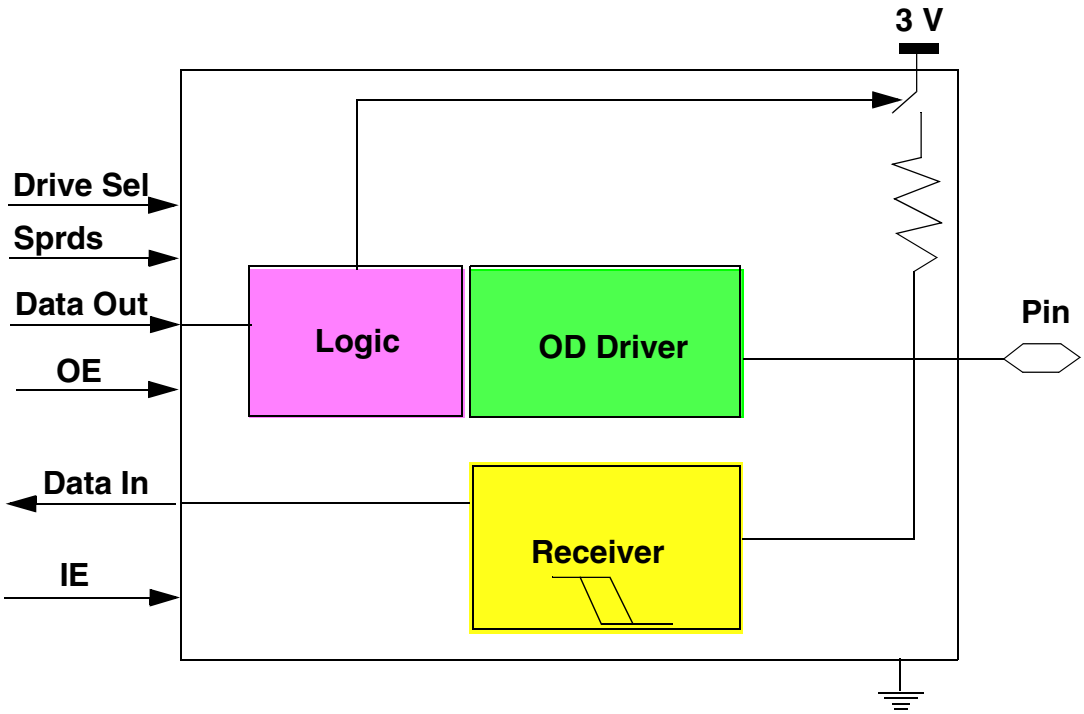


Figure 2-10 3-V Type EOH Interface

2.5.4.3 Type F Interface

In this pad type the data interface to the internal logic has the same path for both input and output. The pull-up is inactive when the driver is enabled.

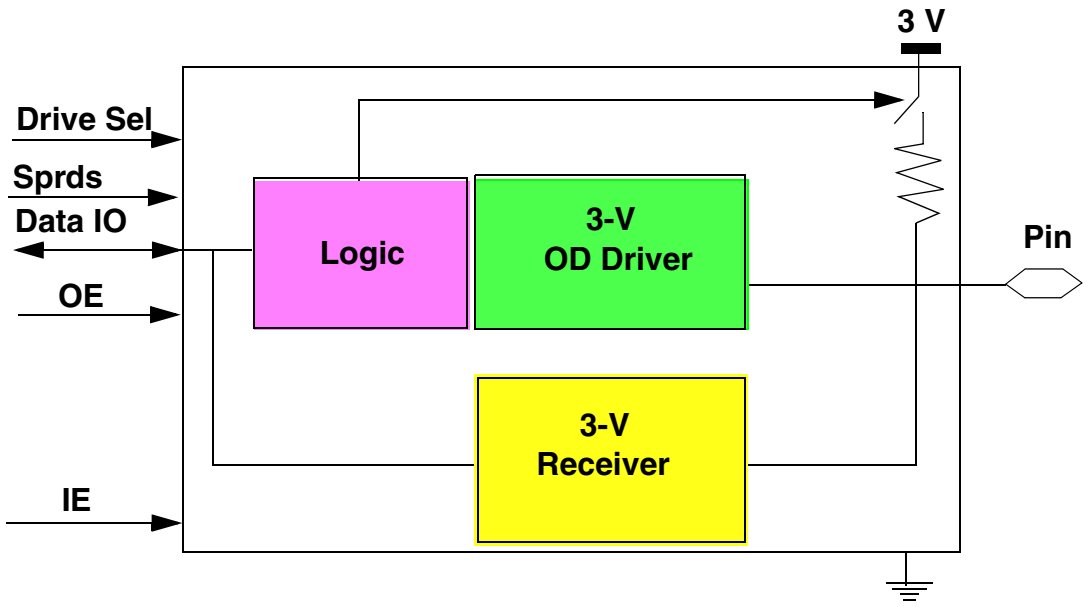


Figure 2-11 Type F Interface

2.5.4.4 Type G Interface

In this pad type the data interface to the internal logic has the same path for both input and output. This pad type also has the SPRDS signal as an input to disable the resistor when the pad is a non-bus function.

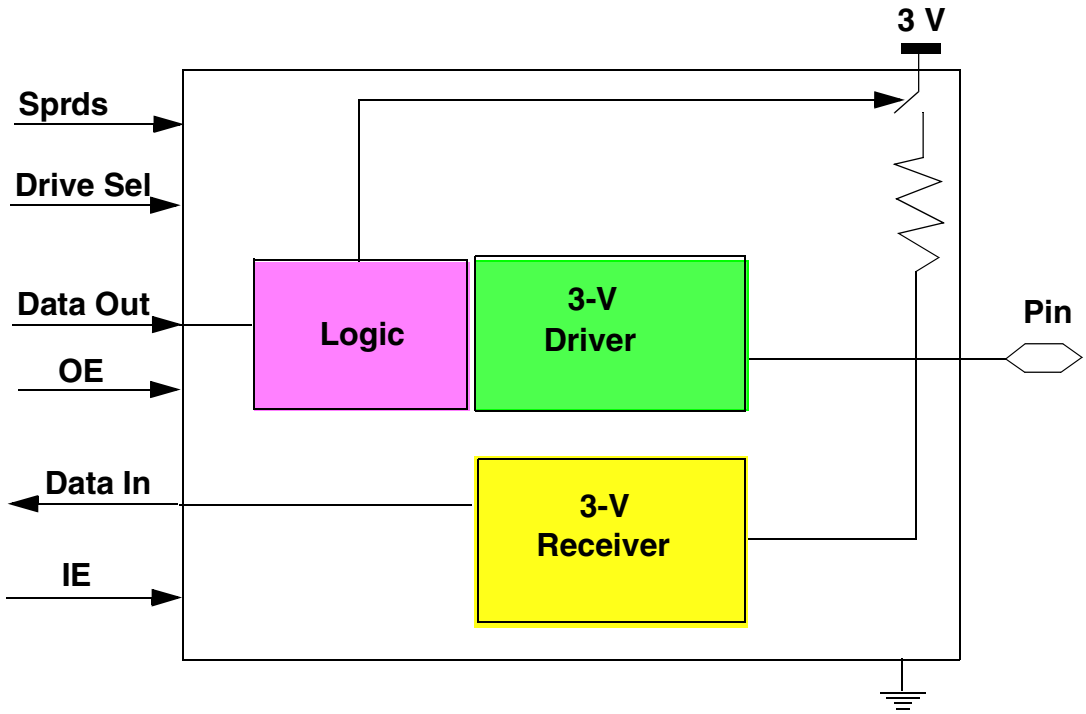


Figure 2-12 Type G Interface

2.5.5 Five-Volt Input/Output Pad

This pad type is for 5-V bi-directional pins. There is provision to pull the pin up to 5 V and logic to control when the pull-up is enabled. For a 5-V driver, the internal “Fast Mode” signal selects the slow or fast driver. All 5-V inputs have hyteresis.

2.5.5.1 Type H Interface

This pad has logic for a 3-V output function as well as a 5-V input-output function. A “3-V / 5-V sel” interface signal determines which driver gets selected.

This pad type has two separate data output paths. These paths are multiplexed onto the output pin based on the 3-V / 5-V select signal. This pad also has a dedicated synchronous input path.

If only one of the output paths is used on a device, the other can be connected to ground. In this case, the 3-V / 5-V select signal must be tied to the appropriate value to disable the other path.

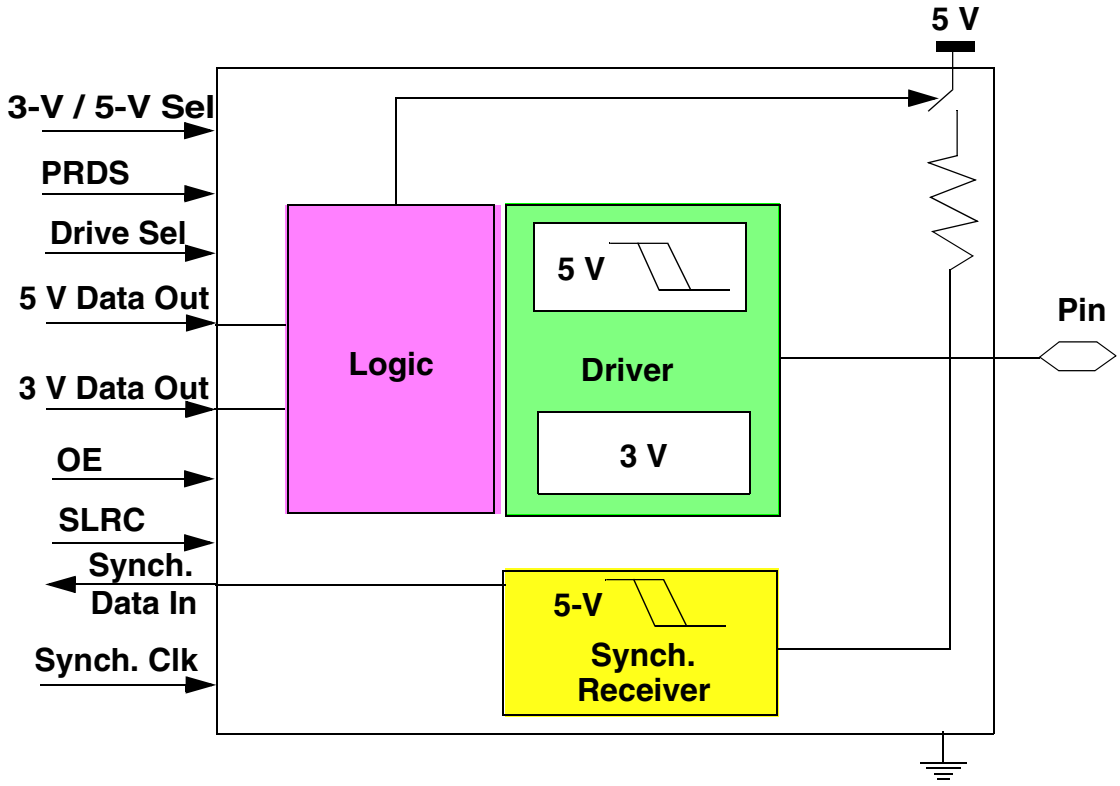


Figure 2-13 Type H Interface

2.5.5.2 Type I Interface

This pad has logic for a 3-V input/output function as well as a 5-V input/output function. A “3-V / 5-V sel” interface signal indicates which driver gets selected.The data interface to the internal logic has separate paths for input and output.

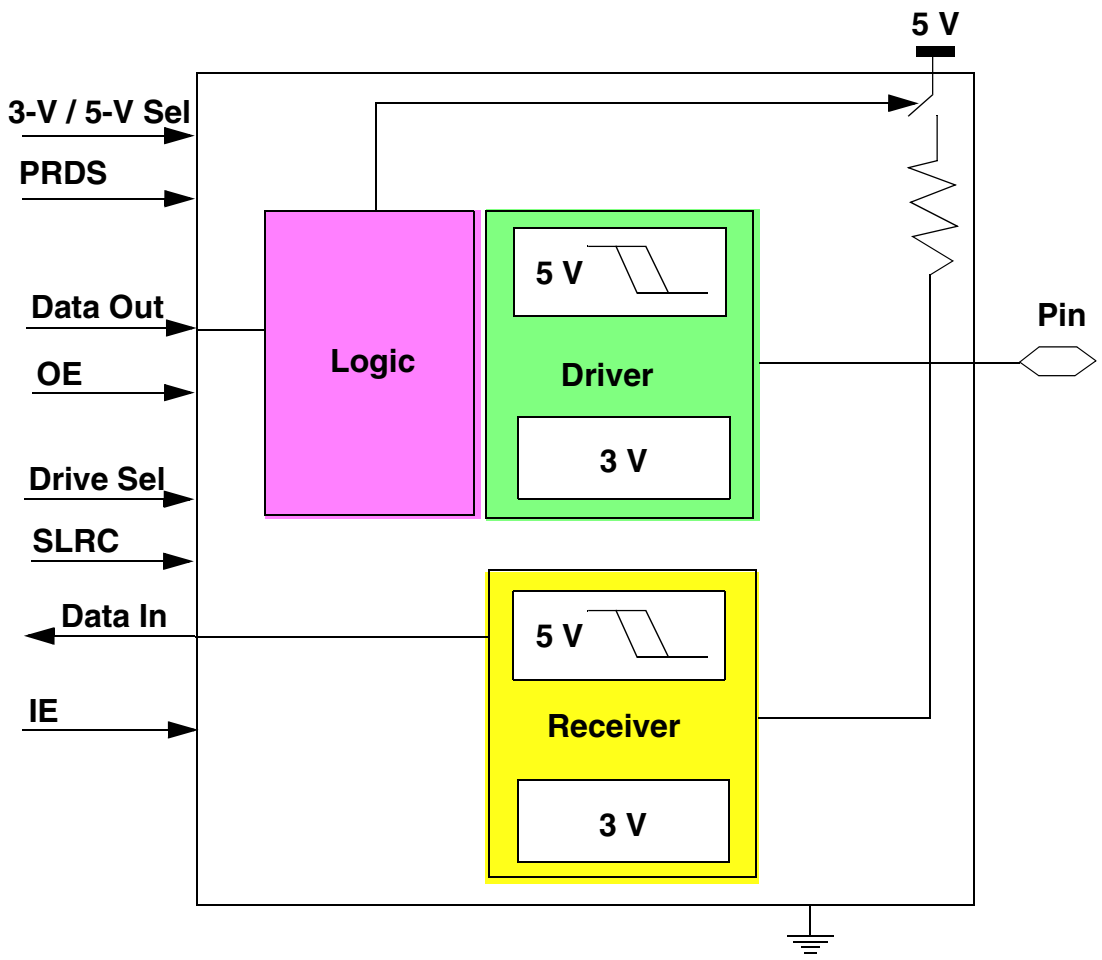


Figure 2-14 Type I Interface

2.5.5.3 Type IH Interface

This pad has logic for a 3-V input/output function as well as a 5-V input/output function. A “3-V / 5-V sel” interface signal determines which driver gets selected.

In this pad type the data interface to the internal logic has separate paths for input and output. The 3-V receiver has 2 possible paths: with or without hysteresis. The hyst_sel signal selects the appropriate path.

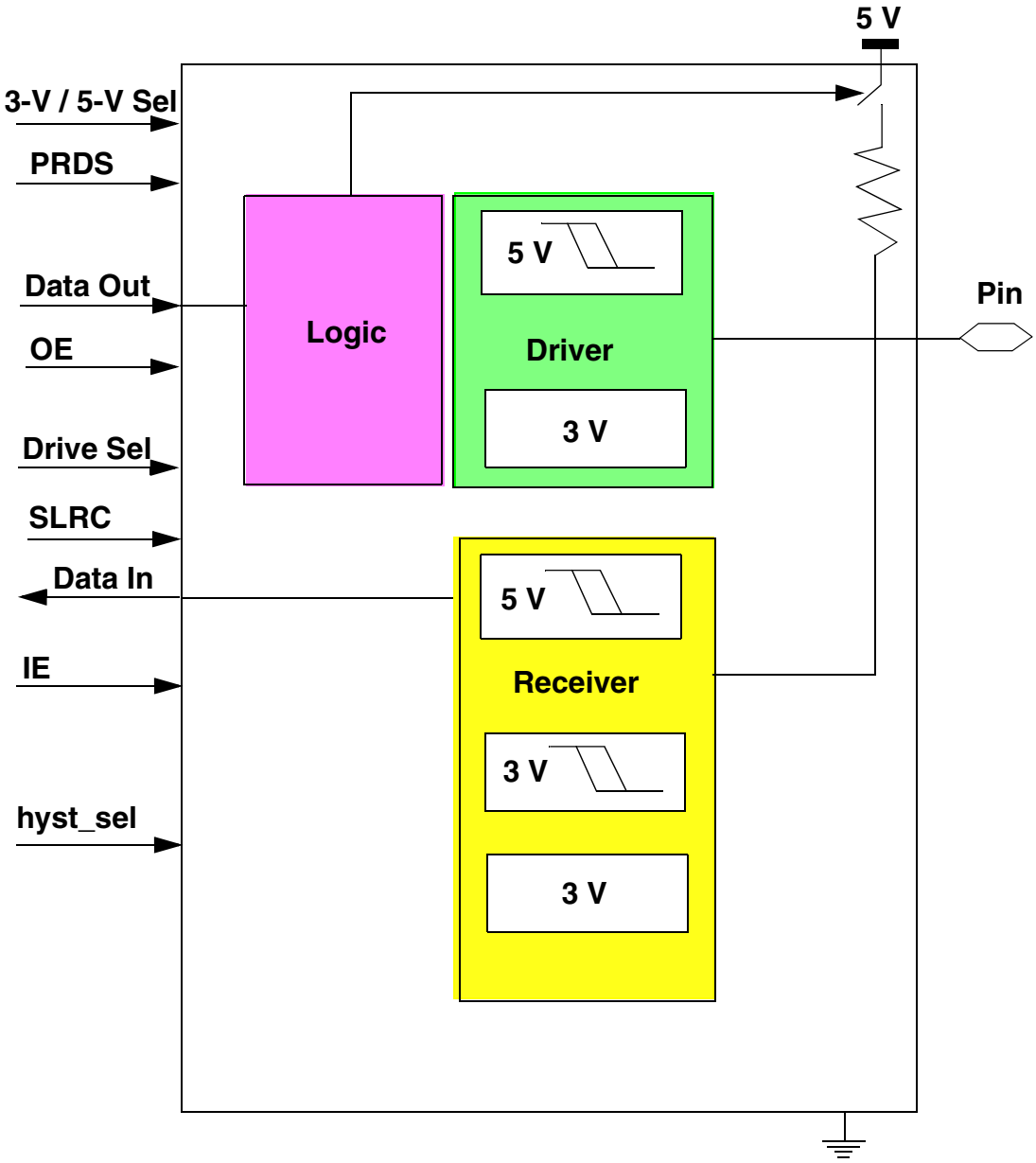


Figure 2-15 Type IH Interface

2.5.5.4 Type J Interface

This pad has logic for a 3-V input/output function as well as a 5-V input/output function. A “3-V / 5-V sel” interface signal indicates which driver gets selected. The data interface to the internal logic has the same path for both input and output.

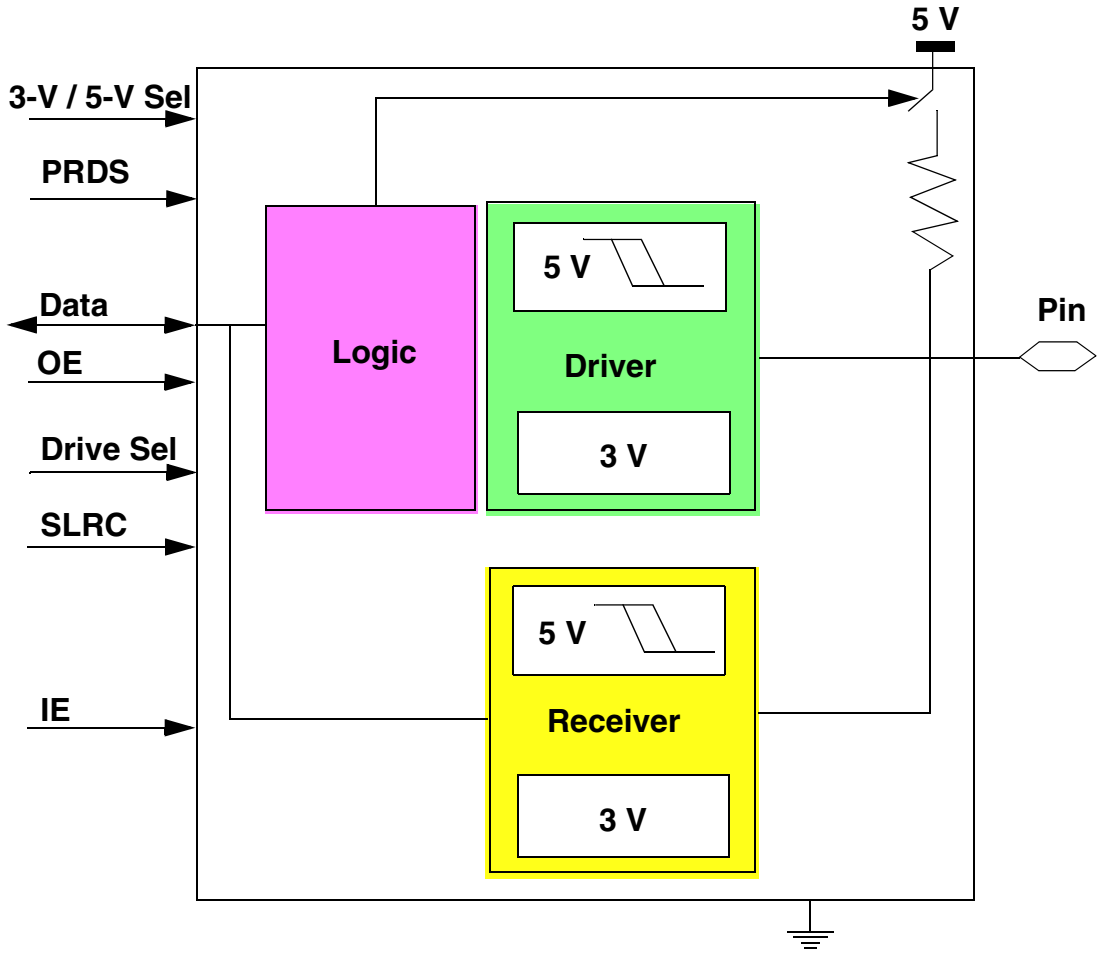


Figure 2-16 Type J Interface

2.5.5.5 Type JD Interface

This pad has logic for a 3-V input/output function as well as a 5-V input/output function. A “3-V / 5-V sel” interface signal indicates which driver gets selected.

The data interface to the internal logic has the same path for both input and output. The pad has a pull-down resistor which is activated by reset and/or PRDS.

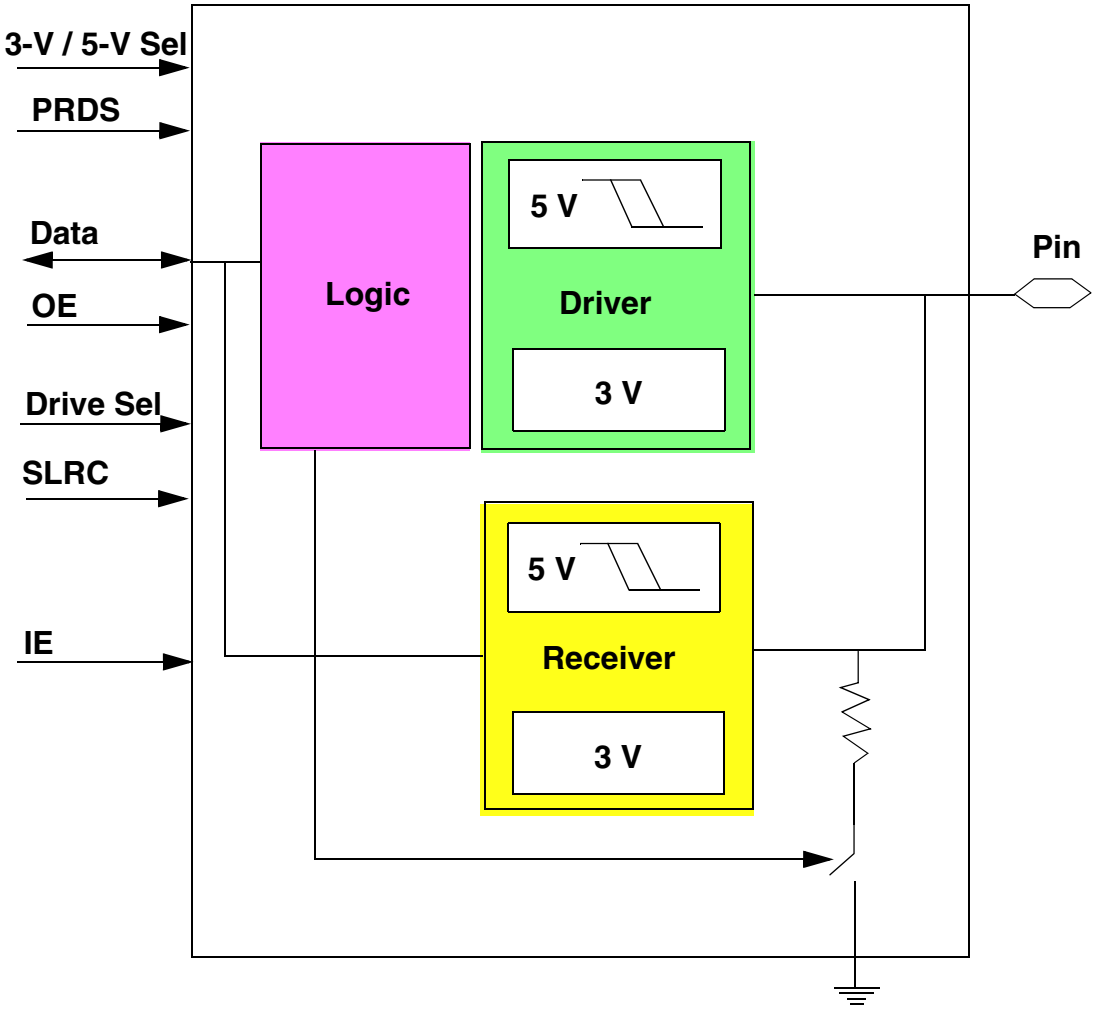


Figure 2-17 Type JD Interface

2.5.6 Type K Interface (EPEE Pad)

This pad has a pull-down device that is enabled at all times. The module checks to see that a transition to a new state on the pin is maintained for at least two clocks before the information is passed on internally to the sequencer implemented in the flash. The synchronizer clock to this pad is GCLK2.

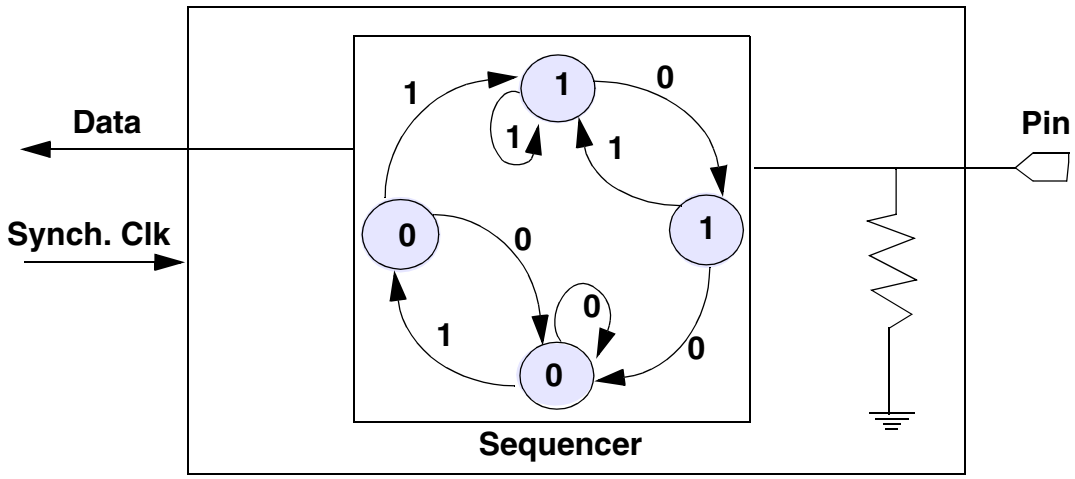


Figure 2-18 EPEE Pad (Type K)

2.5.7 Analog Pads

The 5-V analog pads interface to the QADC modules internally. They have separate analog and digital paths in the pad in order to implement the functionality that is multiplexed on the pin.

2.5.7.1 Type L Interface (QADC Port A)

This pad is used for interfacing to the port A of the QADC. The digital portion of the pad supports bi-directional operation. The receiver has a synchronizer. The digital input is level-shifted from 5 V to 3 V before it is sent internally to the QADC.

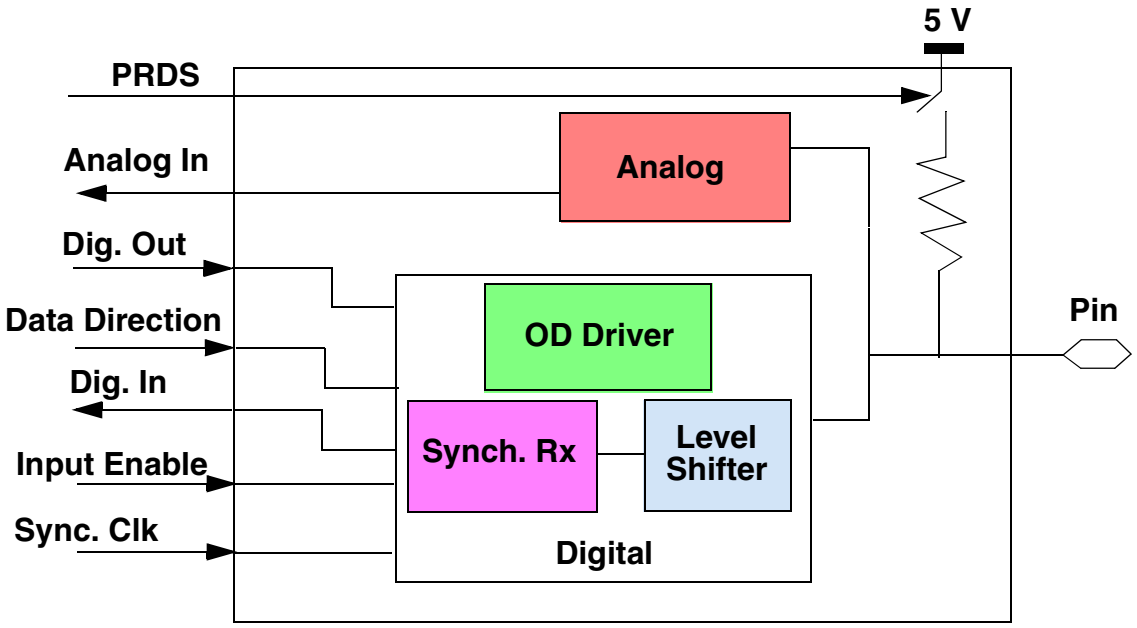


Figure 2-19 Type L Interface

2.5.7.2 Type M Interface (QADC Port B)

This pad is used for interfacing to port B of the QADC. This is an input-only pad. The receiver has a synchronizer. The digital input is level-shifted from 5 V to 3 V before it is sent internally to the QADC.

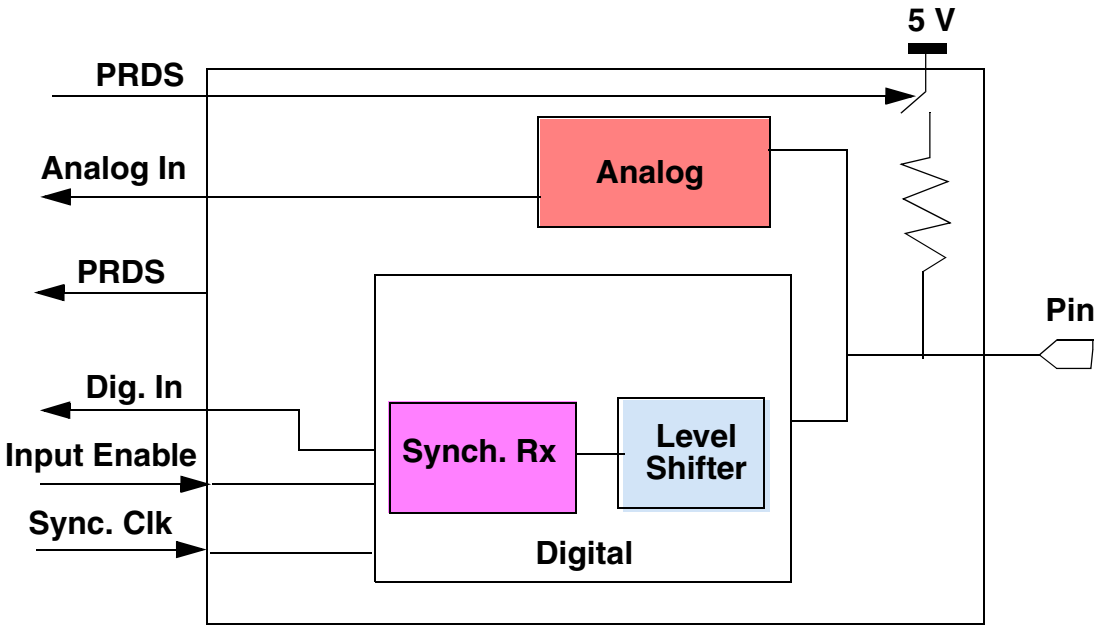


Figure 2-20 Type M Interface

2.5.7.3 Type N Interface (ETRIG)

This is the pad for the ETRIG function of the QADC. The input signal is level-shifted before being sent to the QADC module. The pad also serves as an output pad in test mode.

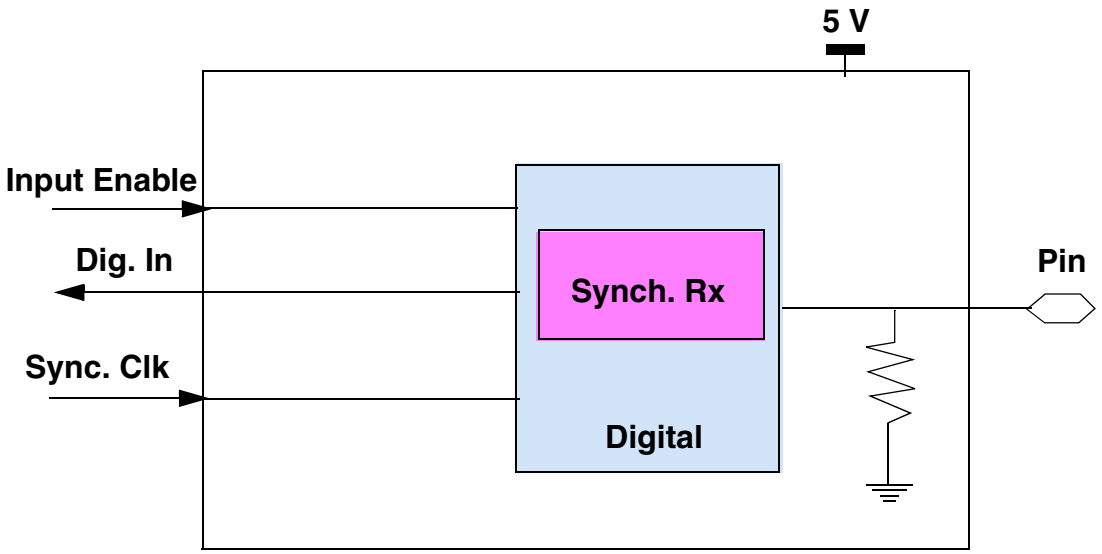


Figure 2-21 Type N Interface

2.5.8 Pads with Fast Mode

The type O pads (for interfacing to the QSMCM) and type P pads (for interfacing to the TPU and MIOS) have a fast mode provision.

2.5.8.1 Type O Interface (QSMCM Pads)

This pad is used for interfacing to the QSMCM. It is a 5-V, bi-directional pad and has provision for a fast mode in which the slow slew rate driver is bypassed and data is driven by a fast slew rate driver. When the pin is an input, the data can be driven either synchronously or asynchronously. A pull-up device is available which can be disabled using the PRDS signal.

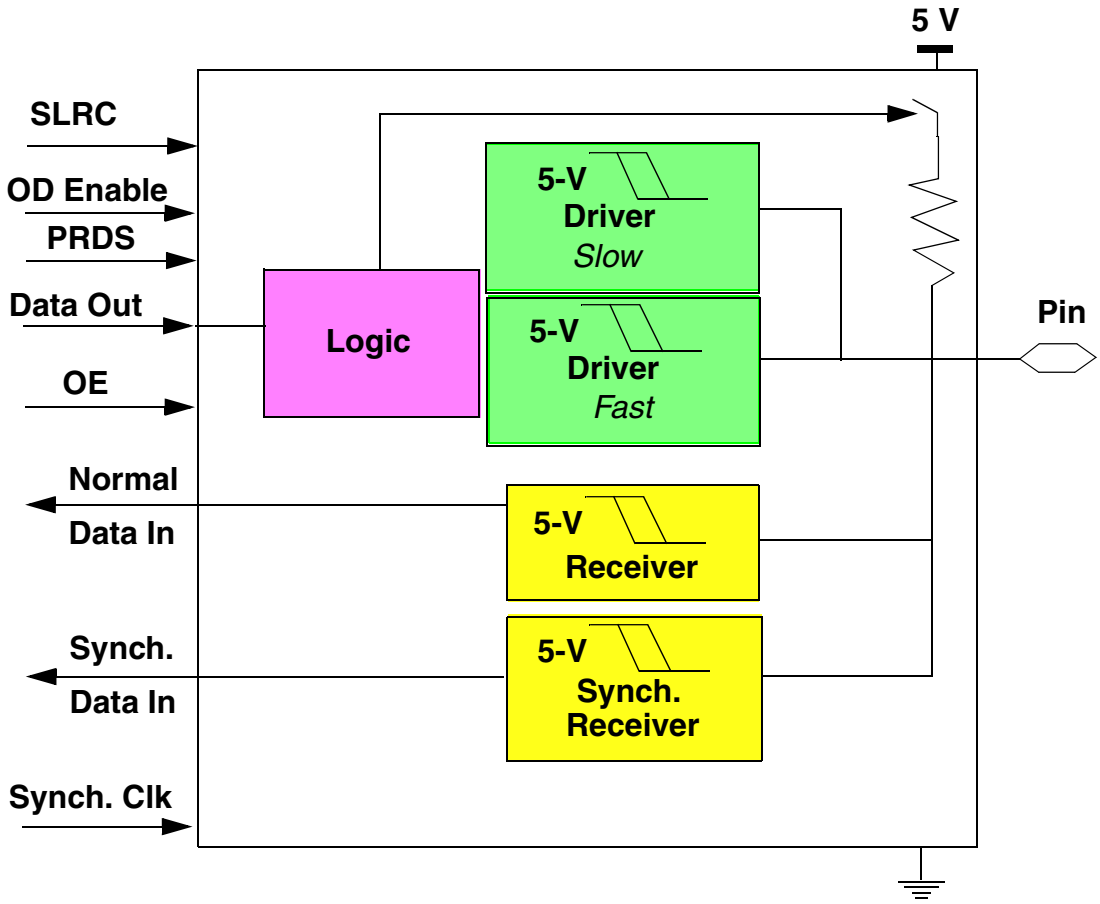


Figure 2-22 Type O Interface

2.5.8.2 Type P Interface (TPU and MIOS Pads)

This is a 5-V, bi-directional pad that has a fast mode provision like the QSMCM pads. The input path is always synchronous. The receiver has hysteresis in order to minimize the effect of noise on the pins. In addition, the receiver has a digital filter (somewhat like the sequencer for the EPEE pad) to check for a state on the pin for a particular number of clocks. The pad also has a pull-up device. Depending on the reset state (see **Table 2-4**) the pull-up may be controlled by the PRDS signal.

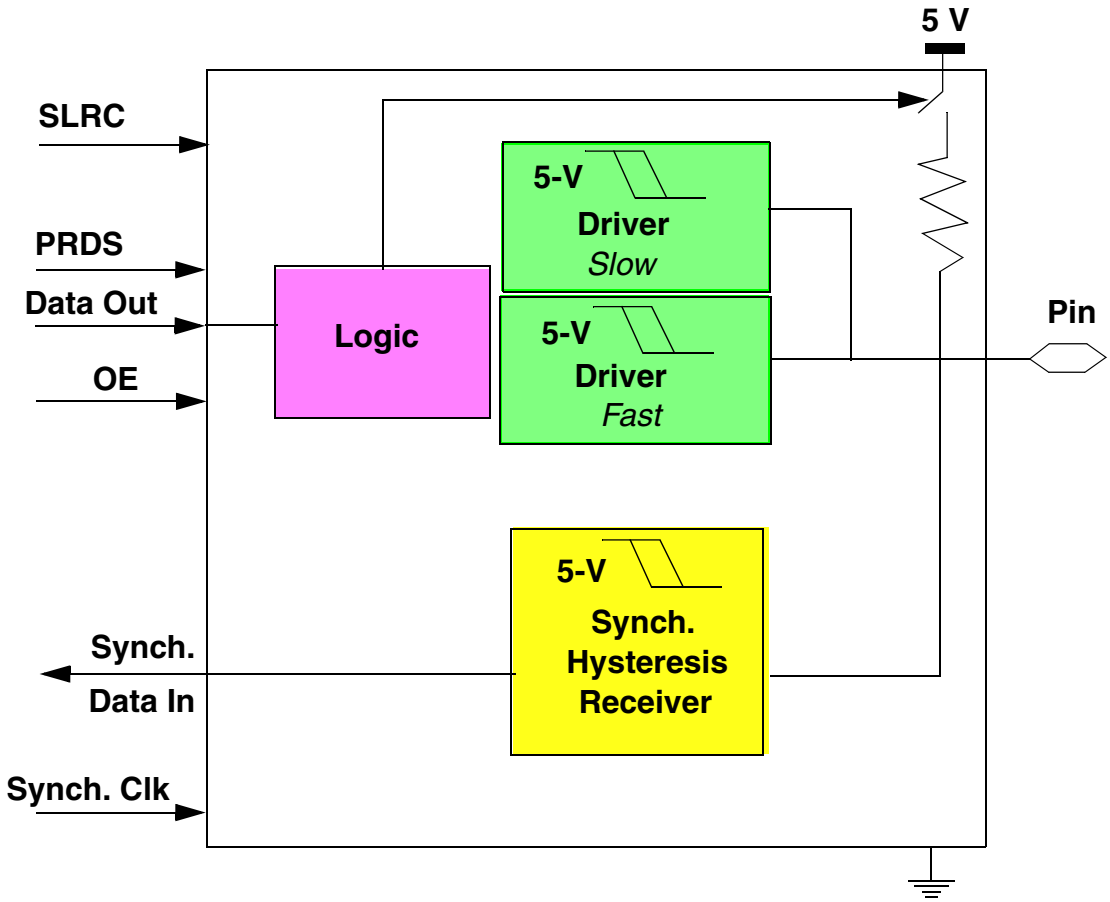


Figure 2-23 Type P Interface

2.5.9 5V Input, 5V Output Pads

These pads are 5-V only pads.

2.5.9.1 5V Output (Type Q)

This pad is a 5-V output-only pad with slow and fast drive capability. The driver is configureable to be either push pull or open drain using the OD enable signal. This pad type has a pull-up device that can be controlled using the PRDS signal.

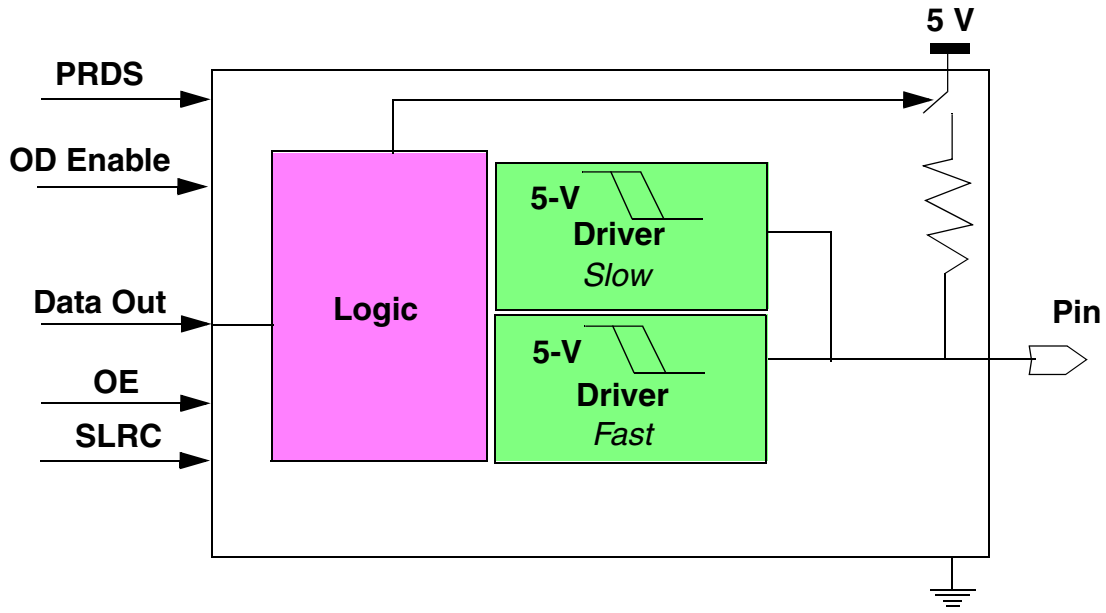


Figure 2-24 Type Q Interface

2.5.9.2 Type R Interface

This is a 5-V input-only pad with a synchronous and asynchronous receiver. Both synchronous and asynchronous data are driven in from the internal module that interfaces to this pad. A pull-up device can be controlled using the PRDS signal.

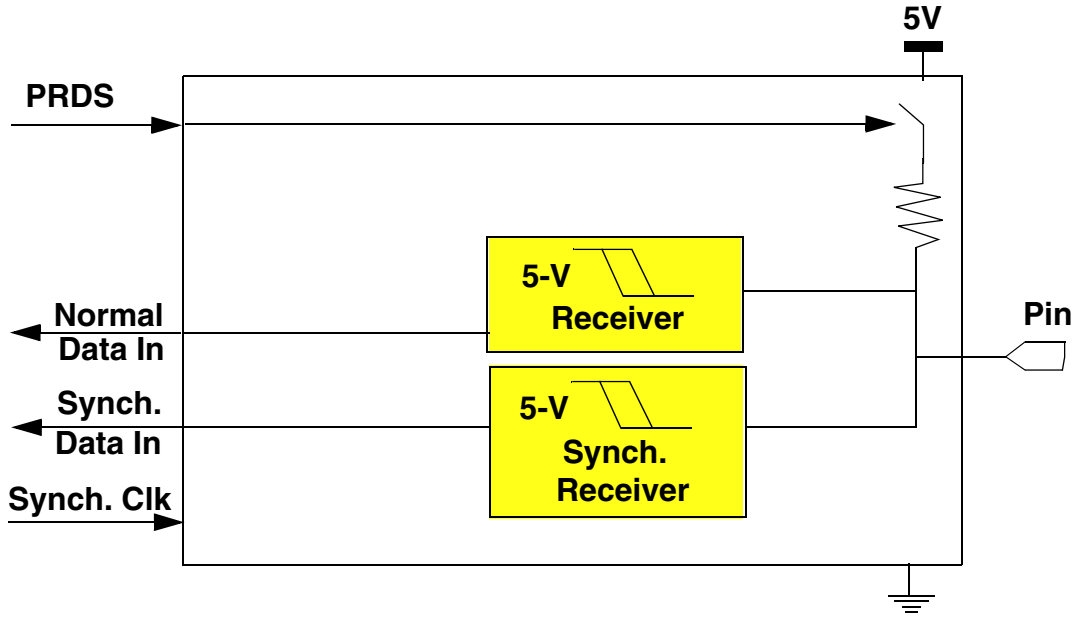


Figure 2-25 Type R Interface

2.5.9.3 5V Output for Clock Pad

This interface is used for a 5-V clock pad output. The drive select signal selects the buffer for a 45- or 90-pF load.

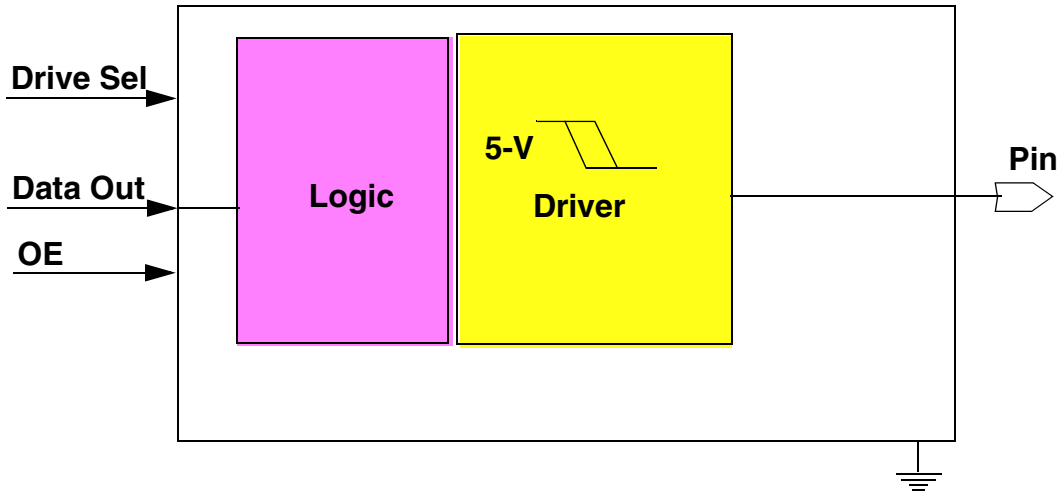


Figure 2-26 Type S Interface

2.6 Pad Groups

A pad group is a set of pins that exhibits similar functional characteristics. Within a group the individual pads may be of different types. The functionality of some pins is defined based on the control bits that are set in the SIUMCR from the reset configuration word. Refer to the section on pin functionality out of reset in the reset section of the document.

The following is a list of pad groups which were obtained based on the 3-V / 5-V selection from the information in the “pin configuration out of reset” tables. In other words, each group receives a different encoded 3-V / 5-V select signal.

Table 2-5 Pad Groups Based on 3-V / 5-V Select

Group	Pins
1	FRZ/PTR/SGPIOC[6], SGPIO[7]/$\overline{\text{IRQOUT}}$/LWP[0]
2	DATA[0:31]/SGPIOD[0:31]
3	ADDR[8:31]/SGPIOA[8:31]
4	$\overline{\text{IRQ}}$[0]/SGPIOC[0], $\overline{\text{IRQ}}$[1]/SGPIOC[1], $\overline{\text{IRQ}}$[4]/SGPIOC[4]
5	$\overline{\text{IRQ}}$[2]/SGPIOC[2], $\overline{\text{IRQ}}$[3]/SGPIOC[3], $\overline{\text{IRQ}}$[5]/SGPIOC[5]

All pins that drive 3 V have the provision to choose between drive strengths for a 25-pF load or a 50-pF load.

2.7 Pin Names and Abbreviations

The following table lists the recommended abbreviations for all the pins on the MPC555 / MPC556. The abbreviations can be used in applications for which the actual name is too long. For example, they can be used to on circuit boards to map the pin location on the boards.

Table 2-6 Pin Names and Abbreviations

Pin List	Pin Name	Abbreviation	Ball
ADDR[8:31]/SGPIOA[8:31]	addr_sgpioa[8]	addr_sgp[8]	V6
	addr_sgpioa[9]	addr_sgp[9]	V5
	addr_sgpioa[10]	addr_sgp[10]	V4
	addr_sgpioa[11]	addr_sgp[11]	V3
	addr_sgpioa[12]	addr_sgp[12]	W1
	addr_sgpioa[13]	addr_sgp[13]	Y2
	addr_sgpioa[14]	addr_sgp[14]	W3
	addr_sgpioa[15]	addr_sgp[15]	Y3
	addr_sgpioa[16]	addr_sgp[16]	W4
	addr_sgpioa[17]	addr_sgp[17]	Y4
	addr_sgpioa[18]	addr_sgp[18]	W5
	addr_sgpioa[19]	addr_sgp[19]	Y5
	addr_sgpioa[20]	addr_sgp[20]	W6
	addr_sgpioa[21]	addr_sgp[21]	Y6
	addr_sgpioa[22]	addr_sgp[22]	V7
	addr_sgpioa[23]	addr_sgp[23]	W7
	addr_sgpioa[24]	addr_sgp[24]	Y7
	addr_sgpioa[25]	addr_sgp[25]	Y8
	addr_sgpioa[26]	addr_sgp[26]	W8
	addr_sgpioa[27]	addr_sgp[27]	V8
	addr_sgpioa[28]	addr_sgp[28]	U8
	addr_sgpioa[29]	addr_sgp[29]	U9
	addr_sgpioa[30]	addr_sgp[30]	U7
	addr_sgpioa[31]	addr_sgp[31]	U6

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
DATA[0:31]/SGPIOD[0:31]	data_sgpiod[0]	data_sgp[0]	Y9
	data_sgpiod[1]	data_sgp[1]	W9
	data_sgpiod[2]	data_sgp[2]	Y10
	data_sgpiod[3]	data_sgp[3]	W10
	data_sgpiod[4]	data_sgp[4]	Y11
	data_sgpiod[5]	data_sgp[5]	W11
	data_sgpiod[6]	data_sgp[6]	Y12
	data_sgpiod[7]	data_sgp[7]	W12
	data_sgpiod[8]	data_sgp[8]	Y13
	data_sgpiod[9]	data_sgp[9]	W13
	data_sgpiod[10]	data_sgp[10]	Y14
	data_sgpiod[11]	data_sgp[11]	W14
	data_sgpiod[12]	data_sgp[12]	Y15
	data_sgpiod[13]	data_sgp[13]	W15
	data_sgpiod[14]	data_sgp[14]	Y16
	data_sgpiod[15]	data_sgp[15]	W16
	data_sgpiod[16]	data_sgp[16]	Y17
	data_sgpiod[17]	data_sgp[17]	W17
	data_sgpiod[18]	data_sgp[18]	V17
	data_sgpiod[19]	data_sgp[19]	V16
	data_sgpiod[20]	data_sgp[20]	U16
	data_sgpiod[21]	data_sgp[21]	V15
	data_sgpiod[22]	data_sgp[22]	V14
	data_sgpiod[23]	data_sgp[23]	U14
	data_sgpiod[24]	data_sgp[24]	V13
	data_sgpiod[25]	data_sgp[25]	U13
	data_sgpiod[26]	data_sgp[26]	V12
	data_sgpiod[27]	data_sgp[27]	U12
	data_sgpiod[28]	data_sgp[28]	V11
	data_sgpiod[29]	data_sgp[29]	U11
	data_sgpiod[30]	data_sgp[30]	V10
	data_sgpiod[31]	data_sgp[31]	V9
$\overline{\text{IRQ}}$[0]/SGPIOC[0]	irq0_b_sgpioc0	irq0b_sgp	M1
$\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1]	irq1_b_rsv_b_sgpioc1	irq1b_sgp	M2
$\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$	irq2_b_cr_b_sgpioc2_mts	irq2b_sgp	M3
$\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$, $\overline{\text{RETRY}}$/SGPIOC[3]	irq3_b_kr_b_retry_b_sgpioc3	irq3b_sgp	L3
$\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4]	irq4_b_at2_sgpioc4	irq4b_sgp	L4
$\overline{\text{IRQ}}$[5]/SGPIOC[5]/MODCK[1]	irq5_b_sgpioc5_modck1	irq5b_sgp	W18
$\overline{\text{IRQ}}$[6:7]/MODCK[2:3]	irq6_b_modck2	irq6b_mck2	Y18
	irq7_b_modck3	irq7b_mck3	Y19
TSIZ[0:1]	tsiz0	tsiz0	U1
	tsiz1	tsiz1	T3

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
RD/ $\overline{\text{WR}}$	rd_wr_b	rd_wrb	R1
$\overline{\text{BURST}}$	burst_b	burstb	V1
$\overline{\text{BDIP}}$	bdip_b	bdipb	U4
$\overline{\text{TS}}$	ts_b	tsb	U3
$\overline{\text{TA}}$	ta_b	tab	U2
$\overline{\text{TEA}}$	tea_b	teab	T2
$\overline{\text{RSTCONF}}$/TEXP	rstconf_b_texp	rcfb_txp	U17
$\overline{\text{OE}}$	oe_b	oeb	T1
$\overline{\text{BI}}$/$\overline{\text{STS}}$	bi_b_sts_b	bib_stsb	V2
$\overline{\text{CS}}$[0:3]	cs0_b	cs0b	P4
	cs1_b	cs1b	R4
	cs2_b	cs2b	R3
	cs3_b	cs3b	R2
$\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3]/AT[0:3]	we0_b_be0_b_at0	web_at[0]	N1
	we1_b_be1_b_at1	web_at[1]	P1
	we2_b_be2_b_at2	web_at[2]	P2
	we3_b_be3_b_at3	web_at[3]	P3
$\overline{\text{PORESET}}$	poreset_b	poresetb	V19
$\overline{\text{HRESET}}$	hreset_b	hresetb	W20
$\overline{\text{SRESET}}$	sreset_b	sresetb	V20
SGPIOC[6]/FRZ/$\overline{\text{PTR}}$/	sgpioc6_frz_ptr_b	sgp_frz	K3
SGPIOC[7]/$\overline{\text{IRQOUT}}$/LWP[0]	sgpioc7_irqout_b_lwp0	sgp_irqoutb	M4
$\overline{\text{BG}}$/VF[0]/LWP[1]	bg_b_vf0_lwp1	bgb_lwp1	N3
$\overline{\text{BR}}$/VF[1]/IWP[2]	br_b_vf1_iwp2	brb_iwp2	N2
$\overline{\text{BB}}$/VF[2]/IWP[3]	bb_b_vf2_iwp3	bbb_iwp3	N4
IWP[0:1]/VFLS[0:1]	iwp0_vfls0	iwp0_vfls	L2
	iwp1_vfls1	iwp1_vfls	L1
TMS	tms	tms	K1
TDI/DSDI	tdi_dsdi	tdi_dsdi	K2
TCK/DSCK	tck_dsck	tck_dsck	J1
TDO/DSDO	tdo_dsdo	tdo_dsdo	J2
$\overline{\text{TRST}}$	trst_b	trst_b	J3
XTAL	xtal	xtal	U20
EXTAL	extal	extal	T20
XFC	xfc	xfc	R19
CLKOUT	clkout	clkout	V18
EXTCLK	extclk	extclk	U18
VDDSYN	vddsyn	vddsyn	R20
VSSSYN	vsssyn	vsssyn	T19
ENGCLK/BUCLK	engclk_buclk	eck_buck	U19

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
QSMCM			
PCS0/$\overline{SS}$/QGPIO[0]	pcs0_ss_b_qgpio0	pcs0_qgp	L18
PCS[1:3]/QGPIO[1:3]	pcs1_qgpio1	pcs1_qgp	L17
	pcs2_qgpio2	pcs2_qgp	M18
	pcs3_qgpio3	pcs3_qgp	M17
MISO/QGPIO[4]	miso_qgpio4	miso_qgp4	L19
MOSI/QGPIO[5]	mosi_qgpio5	mosi_qgp5	L20
SCK/QGPIO[6]	sck_qgpio6	sck_qgp6	M20
TXD[1:2]/QGPO[1:2]	txd1_qgpo1	txd1_qgpo	N18
	txd2_qgpo2	txd2_qgpo	N20
RXD[1:2]/QGPI[1:2]	rxd1_qgpi1	rxd1_qgpi	N17
	rxd2_qgpi2	rxd2_qgpi	N19
ECK	eck	eck	M19
MIOS			
MDA[11:15]	mda11	mda11	A17
	mda12	mda12	A18
	mda13	mda13	A19
	mda14	mda14	B17
	mda15	mda15	B18
MDA[27:31]	mda27	mda27	C17
	mda28	mda28	B20
	mda29	mda29	C18
	mda30	mda30	C19
	mda31	mda31	C20
MPWM[0:3], [16:19]	mpwm0	mpwm0	E17
	mpwm1	mpwm1	D18
	mpwm2	mpwm2	D19
	mpwm3	mpwm3	D20
	mpwm16	mpwm16	F17
	mpwm17	mpwm17	E18
	mpwm18	mpwm18	F18
	mpwm19	mpwm19	E19
VF[0:2]/MPIO32B[0:2]	vf0_mpio32b0	vf0_mpio0	J19
	vf1_mpio32b1	vf1_mpio1	J20
	vf2_mpio32b2	vf2_mpio2	J17
VFLS[0:1]/MPIO32B[3:4]	vfls0_mpio32b3	vfls0_mpio3	J18
	vfls1_mpio32b4	vfls1_mpio4	K18

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
MPIO32B[5:15]	mpio32b5	mpio5	G17
	mpio32b6	mpio6	E20
	mpio32b7	mpio7	F19
	mpio32b8	mpio8	G18
	mpio32b9	mpio9	F20
	mpio32b10	mpio10	H17
	mpio32b11	mpio11	G19
	mpio32b12	mpio12	G20
	mpio32b13	mpio13	H20
	mpio32b14	mpio14	H19
	mpio32b15	mpio15	H18
TPU_A/TPU_B			
A: TPUCH[0:15]	a_tpuch0	a_tpuch0	D3
	a_tpuch1	a_tpuch1	A2
	a_tpuch2	a_tpuch2	D4
	a_tpuch3	a_tpuch3	C3
	a_tpuch4	a_tpuch4	A3
	a_tpuch5	a_tpuch5	D5
	a_tpuch6	a_tpuch6	B3
	a_tpuch7	a_tpuch7	C4
	a_tpuch8	a_tpuch8	A4
	a_tpuch9	a_tpuch9	C5
	a_tpuch10	a_tpuch10	B4
	a_tpuch11	a_tpuch11	B5
	a_tpuch12	a_tpuch12	A5
	a_tpuch13	a_tpuch13	C6
	a_tpuch14	a_tpuch14	B6
	a_tpuch15	a_tpuch15	A6
A: T2CLK	a_t2clk	a_t2clk	C2

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
B: TPUCH[0:15]	b_tpuch0	b_tpuch0	H2
	b_tpuch1	b_tpuch1	H1
	b_tpuch2	b_tpuch2	G1
	b_tpuch3	b_tpuch3	G2
	b_tpuch4	b_tpuch4	G3
	b_tpuch5	b_tpuch5	F1
	b_tpuch6	b_tpuch6	F2
	b_tpuch7	b_tpuch7	E1
	b_tpuch8	b_tpuch8	F3
	b_tpuch9	b_tpuch9	G4
	b_tpuch10	b_tpuch10	E2
	b_tpuch11	b_tpuch11	D1
	b_tpuch12	b_tpuch12	F4
	b_tpuch13	b_tpuch13	D2
	b_tpuch14	b_tpuch14	E3
	b_tpuch15	b_tpuch15	C1
B: T2CLK	b_t2clk	b_t2clk	B1
QADC_A/QADC_B			
ETRIG[1:2]	etrig1	etrig1	C16
	etrig2	etrig2	B16
A: AN0/ANW/PQB0	a_an0_anw_pqb0	aan0_pqb0	A8
A: AN1/ANX/PQB1	a_an1_anx_pqb1	aan1_pqb1	D8
A: AN2/ANY/PQB2	a_an2_any_pqb2	aan2_pqb2	C8
A: AN3/ANZ/PQB3	a_an3_anz_pqb3	aan3_pqb3	B8
A: AN[48:51]/PQB[4:7]	a_an48_pqb4	aan48_pqb4	A9
	a_an49_pqb5	aan49_pqb5	B9
	a_an50_pqb6	aan50_pqb6	D9
	a_an51_pqb7	aan51_pqb7	C9
A: AN[52:54]/MA[0:2]/PQA[0:2]	a_an52_ma0_pqa0	aan52_pqa0	A10
	a_an53_ma1_pqa1	aan53_pqa1	B10
	a_an54_ma2_pqa2	aan54_pqa2	A11
A: AN[55:56]]/PQA[3:4]	a_an55_pqa3	aan55_pqa3	D10
	a_an56_pqa4	aan56_pqa4	C10
A: AN[57:59]/PQA[5:7]	a_an57_pqa5	aan57_pqa5	B11
	a_an58_pqa6	aan58_pqa6	D11
	a_an59_pqa7	aan59_pqa7	C11
B: AN0/ANW/PQB0	b_an0_anw_pqb0	ban0_pqb0	A12
B: AN1/ANX/PQB1	b_an1_anx_pqb1	ban1_pqb1	B12
B: AN2/ANY/PQB2	b_an2_any_pqb2	ban2_pqb2	A13
B: AN3/ANZ/PQB3	b_an3_anz_pqb3	ban3_pqb3	A14

Table 2-6 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Abbreviation	Ball
B: AN[48:51]/PQB[4:7]	b_an48_pqb4	ban48_pqb4	B13
	b_an49_pqb5	ban49_pqb5	C12
	b_an50_pqb6	ban50_pqb6	D12
	b_an51_pqb7	ban51_pqb7	A15
B: AN[52:54]/MA[0:2]/PQA[0:2]	b_an52_ma0_pqa0	ban52_pqa0	B14
	b_an53_ma1_pqa1	ban53_pqa1	C13
	b_an54_ma2_pqa2	ban54_pqa2	B15
B: AN[55:56]/PQA[3:4]	b_an55_pqa3	ban55_pqa3	D13
	b_an56_pqa4	ban56_pqa4	C14
B: AN[57:59]/PQA[5:7]	b_an57_pqa5	ban57_pqa5	C15
	b_an58_pqa6	ban58_pqa6	D14
	b_an59_pqa7	ban59_pqa7	D15
VRH	vrh	vrh	B7
VRL	vrl	vrl	A7
VDDA	vdda	vdda	C7
VSSA	vssa	vssa	D7
TOUCAN_A/TOUCAN_B			
A: CNTX0	a_cntx0	a_cntx0	K19
B: CNTX0	b_cntx0	b_cntx0	H4
A: CNRX0	a_cnrx0	a_cnrx0	K20
B: CNRX0	b_cnrx0	b_cnrx0	H3
CMF			
EPEE	epee	epee	P18
VPP	vpp	vpp	P17
VDDF	vddf	vddf	R18
VSSF	vssf	vssf	P19
Global Power Supplies			
VDDL	vddl	vddl	D17, E4, K4, K17, R17. T4, U10, U15
VDDH	vddh	vddh	A1, A16, A20, B2, B19, P20, Y1, Y20, W2, W19
VDDI	vddi	vddi	T17, U5, D6, D16
KAPWR	kapwr	kapwr	T18
VDDSRAM	vddsram	vddsram	J4
VSS	vss	vss	J9, J10, J11, J12, K9, K10, K11, K12, L9, L10, L11, L12, M9, M10, M11, M12

SECTION 3
CENTRAL PROCESSING UNIT

The PowerPC-based RISC processor (RCPU) used in the MPC500 family of microcontrollers integrates five independent execution units: an integer unit (IU), a load/store unit (LSU), and a branch processing unit (BPU), floating-point unit (FPU) and integer multiplier divider (IMD). The use of simple instructions with rapid execution times yields high efficiency and throughput for MPC555 / MPC556-based systems.

Most integer instructions execute in one clock cycle. Instructions can complete out of order for increased performance; however, the processor makes execution appear sequential.

This section provides an overview of the RCPU. For a detailed description of this processor, refer to the ***RCPU Reference Manual* (RCPURM/AD)**.

3.1 RCPU Features

Major features of the RCPU include the following:

- High-performance microprocessor
  - — Single clock-cycle execution for many instructions
- Five independent execution units and two register files
  - — Independent LSU for load and store operations
  - — BPU featuring static branch prediction
  - — A 32-bit IU
  - — Fully IEEE 754-compliant FPU for both single- and double-precision operations
  - — Thirty-two general-purpose registers (GPRs) for integer operands
  - — Thirty-two floating-point registers (FPRs) for single- or double-precision operands
- Facilities for enhanced system performance
  - — Programmable big- and little-endian byte ordering
  - — Atomic memory references
- In-system testability and debugging features
- High instruction and data throughput
  - — Condition register (CR) look-ahead operations performed by BPU
  - — Branch-folding capability during execution (zero-cycle branch execution time)
  - — Programmable static branch prediction on unresolved conditional branches
  - — A pre-fetch queue that can hold up to four instructions, providing look-ahead capability
  - — Interlocked pipelines with feed-forwarding that control data dependencies in hardware

3.2 RCPU Block Diagram

Figure 3-1 provides a block diagram of the RCPU.

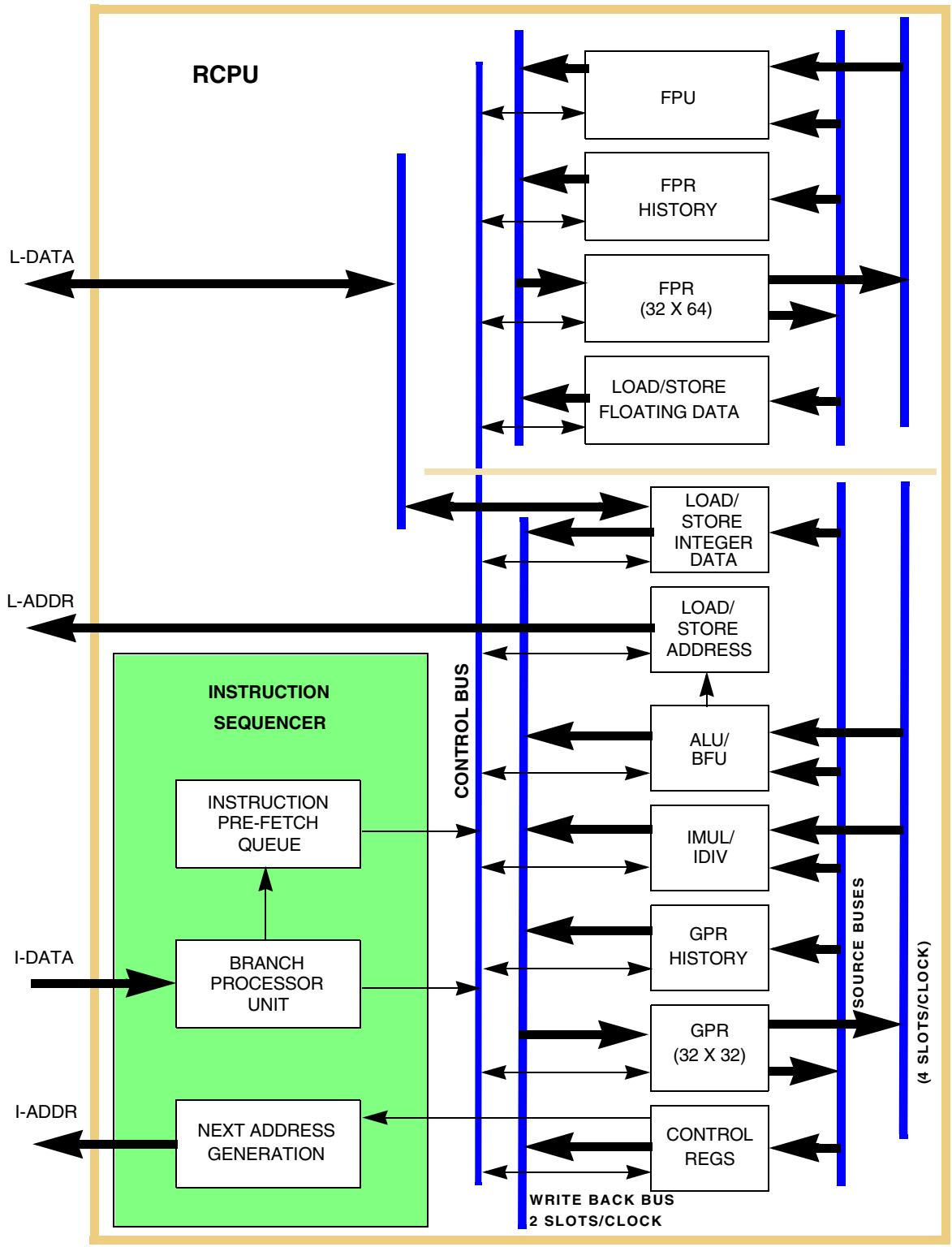


Figure 3-1 RCPU Block Diagram

3.3 Instruction Sequencer

The instruction sequencer provides centralized control over data flow between execution units and register files. The sequencer implements the basic instruction pipeline, fetches instructions from the memory system, issues them to available execution units, and maintains a state history so it can back the machine up in the event of an exception.

The instruction sequencer fetches instructions from the burst buffer controller into the instruction pre-fetch queue. The BPU extracts branch instructions from the pre-fetch queue and uses static branch prediction on unresolved conditional branches to allow the instruction unit to fetch instructions from a predicted target instruction stream while a conditional branch is evaluated. The BPU folds out branch instructions for unconditional branches or conditional branches unaffected by instructions in the execution stage.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

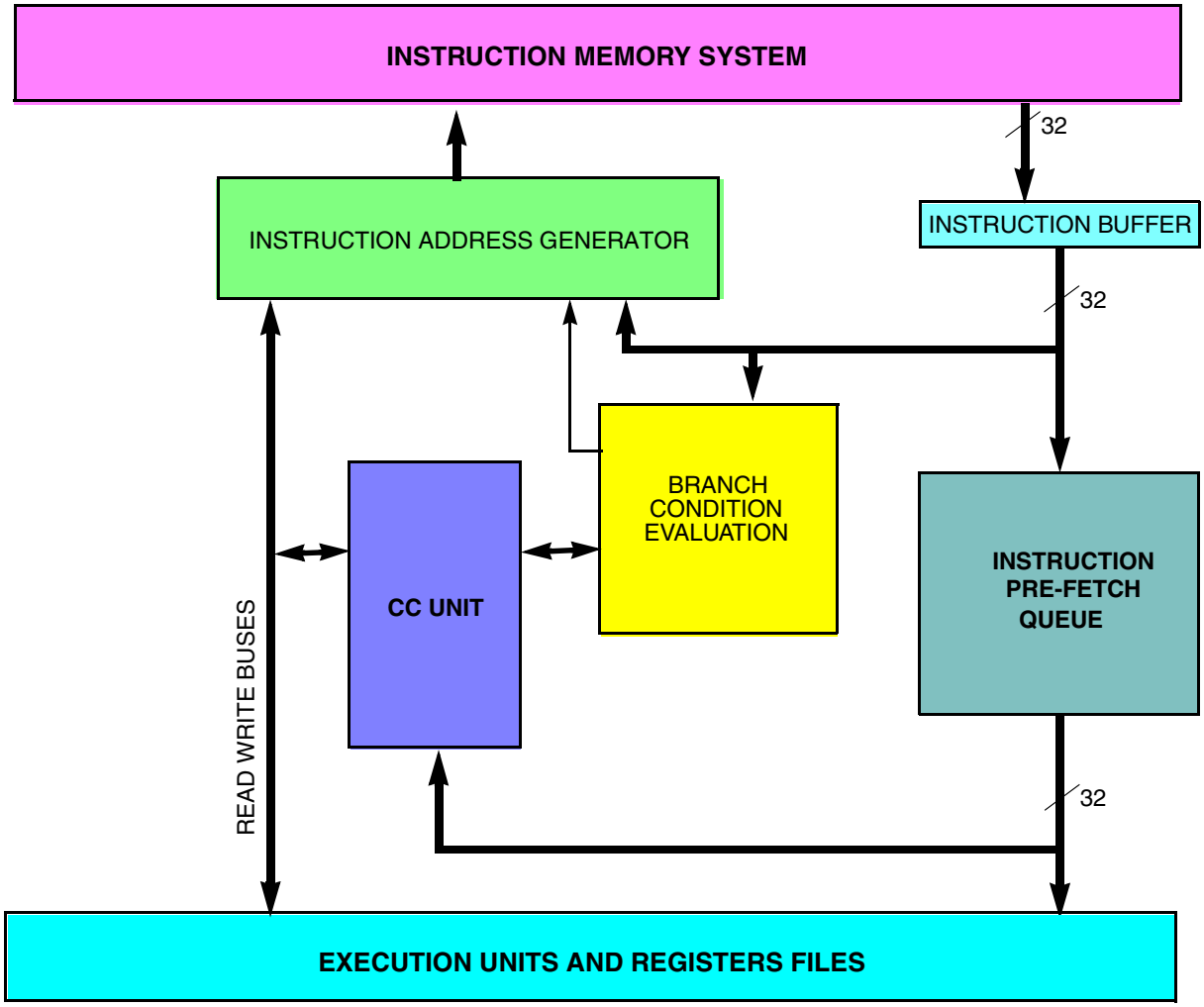


Figure 3-2 Sequencer Data Path

3.4 Independent Execution Units

The PowerPC architecture supports independent floating-point, integer, load/store, and branch processing execution units, making it possible to implement advanced features such as look-ahead operations. For example, since branch instructions do not depend on GPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

Table 3-1 summarizes the RCPU execution units.

Table 3-1 RCPU Execution Units

Unit	Description
Branch processing unit (BPU)	Includes the implementation of all branch instructions
Load/store unit (LSU)	Includes implementation of all load and store instructions, whether defined as part of the integer processor or the floating-point processor
Integer unit (IU)	Includes implementation of all integer instructions except load/store instructions. This module includes the GPRs (including GPR history and scoreboard) and the following subunits: The IMUL-IDIV includes the implementation of the integer multiply and divide instructions. The ALU-BFU includes implementation of all integer logic, add and subtract instructions, and bit field instructions.
Floating-point unit (FPU)	Includes the FPRs (including FPR history and scoreboard) and the implementation of all floating-point instructions except load and store floating-point instructions

The following sections describe the execution units in greater detail.

3.4.1 Branch Processing Unit (BPU)

The BPU, located within the instruction sequencer, performs condition register look-ahead operations on conditional branches. The BPU looks through the instruction queue for a conditional branch instruction and attempts to resolve it early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the processor pre-fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an calculation feature to compute branch target addresses and three special-purpose, user-accessible registers: the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it into the LR. The LR also contains the branch target address for the branch conditional to link register (**bclr*x***) instruction. The CTR contains the branch target address for the branch conditional to count register (**bcctr*x***) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than general-purpose or floating-point registers, execution of branch instructions is independent from execution of integer instructions.

3.4.2 Integer Unit (IU)

The IU executes all integer processor instructions, except the integer storage access instructions, which are implemented by the load/store unit. The IU contains the following subunits:

- The IMUL–IDIV unit includes the implementation of the integer multiply and divide instructions.
- The ALU–BFU unit includes the implementation of all integer logic, add and subtract, and bit field instructions.

The IU also includes the integer exception register (XER) and the general-purpose register file.

IMUL–IDIV and ALU–BFU are implemented as separate execution units. The ALU–BFU unit can execute one instruction per clock cycle. IMUL–IDIV instructions require multiple clock cycles to execute. IMUL–IDIV is pipelined for multiply instructions, so that consecutive multiply instructions can be issued on consecutive clock cycles. Divide instructions are not pipelined; an integer divide instruction preceded or followed by an integer divide or multiply instruction results in a stall in the processor pipeline. Note that since IMUL–IDIV and ALU–BFU are implemented as separate execution units, an integer divide instruction preceded or followed by an ALU–BFU instruction does not cause a delay in the pipeline.

3.4.3 Load/Store Unit (LSU)

The load/store unit handles all data transfer between the general-purpose register file and the internal load/store bus (L-bus). The load/store unit is implemented as an independent execution unit so that stalls in the memory pipeline do not cause the master instruction pipeline to stall (unless there is a data dependency). The unit is fully pipelined so that memory instructions of any size may be issued on back-to-back cycles.

There is a 32-bit wide data path between the load/store unit and the general-purpose register file. Single-word accesses can be achieved with an internal on-chip data RAM, resulting in two clocks latency. Double-word accesses require two clocks, resulting in three clocks latency. Since the L-bus is 32 bits wide, double-word transfers require two bus accesses. The load/store unit performs zero-fill for byte and half-word transfers and sign extension for half-word transfers.

Addresses are formed by adding the source one register operand specified by the instruction (or zero) to either a source two register operand or to a 16-bit, immediate value embedded in the instruction.

3.4.4 Floating-Point Unit (FPU)

The FPU contains a double-precision multiply array, the floating-point status and control register (FPSCR), and the FPRs. The multiply-add array allows the MPC555 / MPC556 to efficiently implement floating-point operations such as multiply, multiply-add, and divide.

The MPC555 / MPC556 depends on a software envelope to fully implement the IEEE floating-point specification. Overflows, underflows, NaNs, and denormalized numbers cause floating-point assist exceptions that invoke a software routine to deliver (with hardware assistance) the correct IEEE result.

To accelerate time-critical operations and make them more deterministic, the MPC555 / MPC556 provides a mode of operation that avoids invoking the software envelope and attempts to deliver results in hardware that are adequate for most applications, if not in strict conformance with IEEE standards. In this mode, denormalized numbers, NaNs, and IEEE invalid operations are treated as legitimate, returning default results rather than causing floating-point assist exceptions.

3.5 Levels of the PowerPC Architecture

The PowerPC architecture consists of three layers. Adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture are implemented:

- PowerPC user instruction set architecture (UISA) — Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- PowerPC virtual environment architecture (VEA) — Describes the memory model for a multiprocessor environment, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- PowerPC operating environment architecture (OEA) — Defines the memory management model, supervisor-level registers, synchronization requirements, and the exception model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

3.6 RCPU Programming Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between memory and on-chip registers.

PowerPC processors have two levels of privilege: supervisor mode of operation (typically used by the operating environment) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, special-purpose registers (SPRs), and several miscellaneous registers.

Supervisor-level access is provided through the processor's exception mechanism. That is, when an exception is taken (either due to an error or problem that needs to be serviced, or deliberately through the use of a trap instruction), the processor begins operating in supervisor mode. The level of access is indicated by the privilege-level (PR) bit in the machine state register (MSR).

Figure 3-3 shows the user-level and supervisor-level RCPU programming models and also illustrates the three levels of the PowerPC architecture. The numbers to the left of the SPRs indicate the decimal number that is used in the syntax of the instruction operands to access the register.

Note that registers such as the general-purpose registers (GPRs) are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as move to special-purpose register (**mtspr**) and move from special-purpose register (**mfspr**) instructions) or implicitly as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

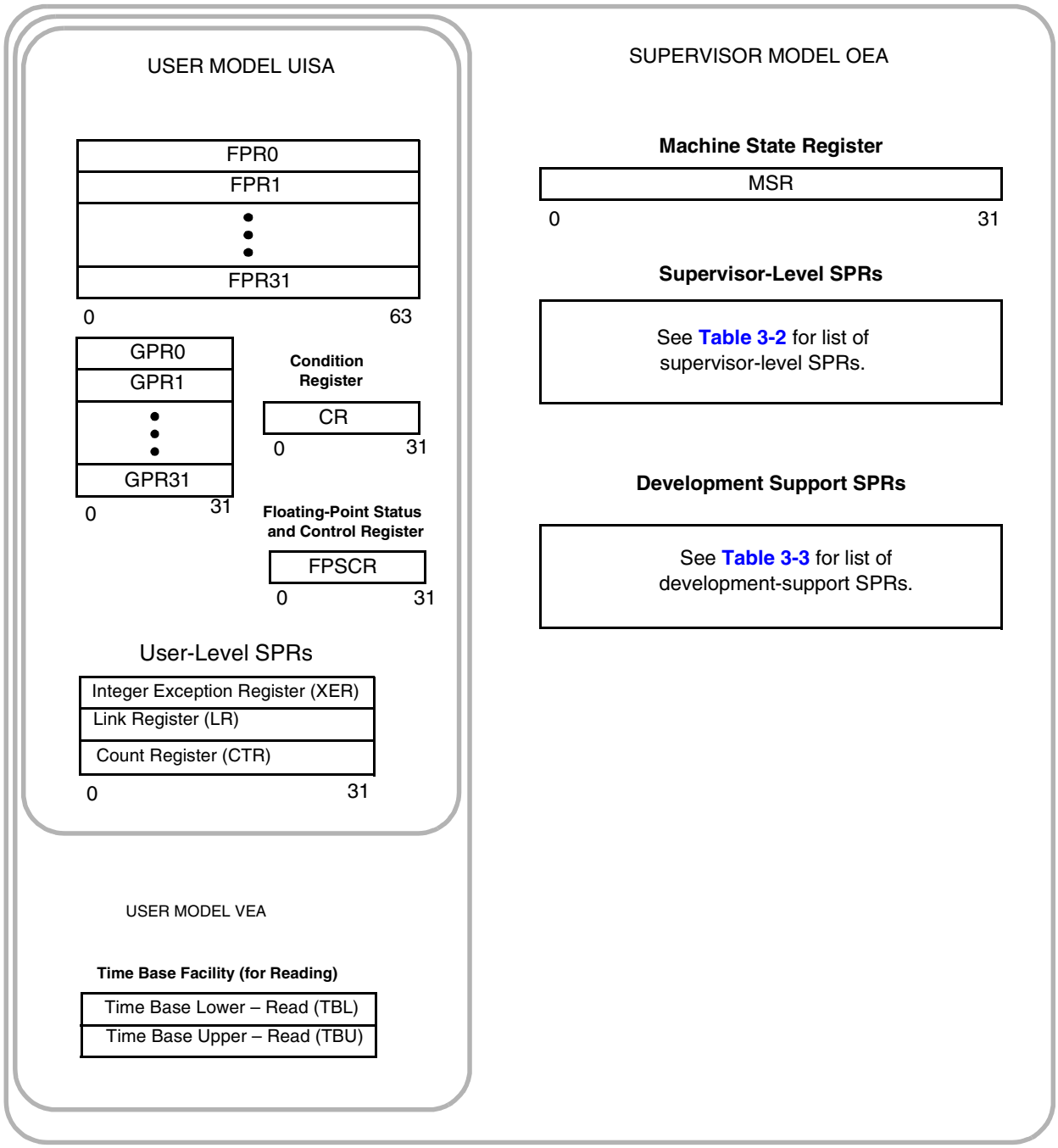


Figure 3-3 RCPU Programming Model

Table 3-2 lists the MPC555 / MPC556 supervisor-level registers.

Table 3-2 Supervisor-Level SPRs

SPR Number (Decimal)	Special-Purpose Register
18	DAE/Source Instruction Service Register (DSISR) See **3.9.2 DAE/Source Instruction Service Register (DSISR)** for bit descriptions.
19	Data Address Register (DAR) See **3.9.3 Data Address Register (DAR)** for bit descriptions.
22	Decrementer Register (DEC) See **3.9.5 Decrementer Register (DEC)** for bit descriptions.
26	Save and Restore Register 0 (SRR0) See **3.9.6 Machine Status Save/Restore Register 0 (SRR0)** for bit descriptions.
27	Save and Restore Register 1 (SRR1) See **3.9.7 Machine Status Save/Restore Register 1 (SRR1)** for bit descriptions.
80	External Interrupt Enable (EIE)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
81	External Interrupt Disable (EID)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
82	Non-Recoverable Interrupt (NRI)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
272	SPR General 0 (SPRG0) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
273	SPRGeneral 1 (SPRG1) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
274	SPR General 2 (SPRG2) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
275	SPR General 3 (SPRG3) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
284	Time Base Lower – Write (TBL) See **Table 3-14** for bit descriptions.
285	Time Base Upper – Write (TBU) See **Table 3-14** for bit descriptions.
287	Processor Version Register (PVR) See **Table 3-16** for bit descriptions.
528	IMPU Global Region Attribute (MI_GRA)[1] See **Table 4-7** for bit descriptions.
536	L2U Global Region Attribute (L2U_GRA)1 See **Table 11-10** for bit descriptions.
560	BBC Module Configuration Register (BBCMCR)1 See **Table 4-8** for bit descriptions.

Table 3-2 Supervisor-Level SPRs (Continued)

SPR Number (Decimal)	Special-Purpose Register
568	L2U Module Configuration Register (L2U_MCR)[1] See **Table 11-7** for bit descriptions.
784	IMPU Region Base Address 0 (MI_RBA0)1 See **Table 4-5** for bit descriptions.
785	IMPU Region Base Address 1 (MI_RBA1)1 See **Table 4-5** for bit descriptions.
786	IMPU Region Base Address 2 (MI_RBA2)1 See **Table 4-5** for bit descriptions.
787	IMPU Region Base Address 3 (MI_RBA3)[1] See **Table 4-5** for bit descriptions.
792	L2U Region Base Address Register 0 (L2U_RBA0)1 See **Table 11-8** for bit descriptions.
793	L2U Region Base Address Register 1 (L2U_RBA1)1 See **Table 11-8** for bit descriptions.
794	L2U Region Base Address Register 2 (L2U_RBA2)1 See **Table 11-8** for bit descriptions.
795	L2U Region Base Address Register 3 (L2U_RBA3)1 See **Table 11-8** for bit descriptions.
816	IMPU Region Attribute Register 0 (MI_RA0)1 See **Table 4-6** for bit descriptions.
817	IMPU Region Attribute Register 1 (MI_RA1)1 See **Table 4-6** for bit descriptions.
818	IMPU Region Attribute Register 2 (MI_RA2)1 See **Table 4-6** for bit descriptions.
819	IMPU Region Attribute Register 3 (MI_RA3)1 See **Table 4-6** for bit descriptions.
824	L2U Region Attribute Register 0 (L2U_RA0)1 See **Table 11-9** for bit descriptions.
825	L2U Region Attribute Register 1 (L2U_RA1)1 See **Table 11-9** for bit descriptions.
826	L2U Region Attribute Register 2 (L2U_RA2)1 See **Table 11-9** for bit descriptions.
827	L2U Region Attribute Register 3 (L2U_RA3)1 See **Table 11-9** for bit descriptions.
1022	Floating-Point Exception Cause Register (FPECR)1 See **3.9.10.2 Floating-Point Exception Cause Register (FPECR)** for bit descriptions.

NOTES:
1. Implementation-specific SPR.

Table 3-3 lists the MPC555 / MPC556 SPRs used for development support.

Table 3-3 Development Support SPRs[1]

SPR Number (Decimal)	Special-Purpose Register
144	Comparator A Value Register (CMPA) See **Table 21-17** for bit descriptions.
145	Comparator B Value Register (CMPB) See **Table 21-17** for bit descriptions.
146	Comparator C Value Register (CMPC) See **Table 21-17** for bit descriptions.
147	Comparator D Value Register (CMPD) See **Table 21-17** for bit descriptions.
148	Exception Cause Register (ECR) See **Table 21-27** for bit descriptions.
149	Debug Enable Register (DER) See **Table 21-28** for bit descriptions.
150	Breakpoint Counter A Value and Control (COUNTA) See **Table 21-25** for bit descriptions.
151	Breakpoint Counter B Value and Control (COUNTB) See **Table 21-26** for bit descriptions.
152	Comparator E Value Register (CMPE) See **Table 21-18** for bit descriptions.
153	Comparator F Value Register (CMPF) See **Table 21-18** for bit descriptions.
154	Comparator G Value Register (CMPG) See **Table 21-20** for bit descriptions.
155	Comparator H Value Register (CMPH) See **Table 21-20** for bit descriptions.
156	L-bus Support Comparators Control 1 (LCTRL1) See **Table 21-23** for bit descriptions.
157	L-bus Support Comparators Control 2 (LCTRL2) See **Table 21-24** for bit descriptions.
158	I-bus Support Control Register (ICTRL) See **Table 21-21** for bit descriptions.
159	Breakpoint Address Register (BAR) See **Table 21-19** for bit descriptions.
630	Development Port Data Register (DPDR) See **21.7.13** for bit descriptions.

NOTES:
1. All development-support SPRs are implementation-specific.

Where not otherwise noted, reserved fields in registers are ignored when written and return zero when read. An exception to this rule is XER[16:23]. These bits are set to the value written to them and return that value when read.

3.7 PowerPC UISA Register Set

The PowerPC UISA registers can be accessed by either user- or supervisor-level instructions. The general-purpose registers are accessed through instruction operands.

3.7.1 General-Purpose Registers (GPRs)

Integer data is manipulated in the integer unit's thirty-two 32-bit GPRs, shown below. These registers are accessed as source and destination registers through operands in the instruction syntax.

GPRs — General-Purpose Registers

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
GPR0																															
GPR1																															
. . .																															
. . .																															
GPR31																															

RESET: UNCHANGED

3.7.2 Floating-Point Registers (FPRs)

The PowerPC architecture provides thirty-two 64-bit FPRs. These registers are accessed as source and destination registers through operands in floating-point instructions. Each FPR supports the double-precision, floating-point format. Every instruction that interprets the contents of an FPR as a floating-point value uses the double-precision floating-point format for this interpretation. That is, all floating-point numbers are stored in double-precision format.

All floating-point arithmetic instructions operate on data located in FPRs and, with the exception of the compare instructions (which update the CR), place the result into an FPR. Information about the status of floating-point operations is placed into the floating-point status and control register (FPSCR) and in some cases, into the CR, after the completion of the operation's writeback stage. For information on how the CR is affected by floating-point operations, see **3.7.4 Condition Register (CR)**.

FPRs— Floating-Point Registers

MSB 0	LSB 63
FPR0	
FPR1	
. . .	
. . .	
FPR31	

RESET: UNCHANGED

3.7.3 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. FPSCR[0:23] are status bits. FPSCR[24:31] are control bits.

FPSCR[0:12] and FPSCR[21:23] are floating-point exception condition bits. These bits are sticky, except for the floating-point enabled exception summary (FEX) and floating-point invalid operation exception summary (VX). Once set, sticky bits remain set until they are cleared by an **mcrfs**, **mtfsfi**, **mtfsf**, or **mtfsb0** instruction.

Table 3-4 summarizes which bits in the FPSCR are sticky status bits, which are normal status bits, and which are control bits.

Table 3-4 FPSCR Bit Categories

Bits	Type
[0], [3:12], [21:23]	Status, sticky
[1:2], [13:20]	Status, not sticky
[24:31]	Control

FEX and VX are the logical ORs of other FPSCR bits. Therefore these two bits are not listed among the FPSCR bits directly affected by the various instructions.

FPSCR — Floating-Point Status and Control Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FX	FEX	VX	OX	UX	ZX	XX	VXS-NAN	VXISI	VXIDI	VXZDZ	VXIMZ	VXVC	FR	FI	FPRF 0

RESET: UNCHANGED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
FPRF[1:4]				0	VX-SOFT	VX-SQRT	VXCVI	VE	OE	UE	ZE	XE	NI	RN	

RESET: UNCHANGED

A listing of FPSCR bit descriptions is shown in **Table 3-5**.

Table 3-5 FPSCR Bit Descriptions

Bit(s)	Name	Description
0	FX	Floating-point exception summary. Every floating-point instruction implicitly sets FPSCR[FX] if that instruction causes any of the floating-point exception bits in the FPSCR to change from 0 to 1. The **mcrfs** instruction implicitly clears FPSCR[FX] if the FPSCR field containing FPSCR[FX] is copied. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions can set or clear FPSCR[FX] explicitly. This is a sticky bit.
1	FEX	Floating-point enabled exception summary. This bit signals the occurrence of any of the enabled exception conditions. It is the logical OR of all the floating-point exception bits masked with their respective enable bits. The **mcrfs** instruction implicitly clears FPSCR[FEX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[FEX] explicitly. This is not a sticky bit.
2	VX	Floating-point invalid operation exception summary. This bit signals the occurrence of any invalid operation exception. It is the logical OR of all of the invalid operation exceptions. The **mcrfs** instruction implicitly clears FPSCR[VX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[VX] explicitly. This is not a sticky bit.
3	OX	Floating-point overflow exception. This is a sticky bit.
4	UX	Floating-point underflow exception. This is a sticky bit.
5	ZX	Floating-point zero divide exception. This is a sticky bit.
6	XX	Floating-point inexact exception. This is a sticky bit.
7	VXSNAN	Floating-point invalid operation exception for SNaN. This is a sticky bit.
8	VXISI	Floating-point invalid operation exception for ×-×. This is a sticky bit.
9	VXIDI	Floating-point invalid operation exception for ×/×. This is a sticky bit.
10	VXZDZ	Floating-point invalid operation exception for 0/0. This is a sticky bit.
11	VXIMZ	Floating-point invalid operation exception for ×*0. This is a sticky bit.
12	VXVC	Floating-point invalid operation exception for invalid compare. This is a sticky bit.
13	FR	Floating-point fraction rounded. The last floating-point instruction that potentially rounded the intermediate result incremented the fraction. This bit is not sticky.
14	FI	Floating-point fraction inexact. The last floating-point instruction that potentially rounded the intermediate result produced an inexact fraction or a disabled exponent overflow. This bit is not sticky.
[15:19]	FPRF	Floating-point result flags. This field is based on the value placed into the target register even if that value is undefined. Refer to **Table 3-6** for specific bit descriptions. 15 Floating-point result class descriptor (C). Floating-point instructions other than the compare instructions may set this bit with the FPCC bits, to indicate the class of the result. 16–19 Floating-point condition code (FPCC). Floating-point compare instructions always set one of the FPCC bits to one and the other three FPCC bits to zero. Other floating-point instructions may set the FPCC bits with the C bit, to indicate the class of the result. Note that in this case the high-order three bits of the FPCC retain their relational significance indicating that the value is less than, greater than, or equal to zero. 16 Floating-point less than or negative (FL or <) 17 Floating-point greater than or positive (FG or >) 18 Floating-point equal or zero (FE or =) 19 Floating-point unordered or NaN (FU or ?)
20	—	Reserved

Table 3-5 FPSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
21	VXSOFT	Floating-point invalid operation exception for software request. This bit can be altered only by the **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, or **mtfsb1** instructions. The purpose of VXSOFT is to allow software to cause an invalid operation condition for a condition that is not necessarily associated with the execution of a floating-point instruction. For example, it might be set by a program that computes a square root if the source operand is negative. This is a sticky bit.
22	VXSQRT	Floating-point invalid operation exception for invalid square root. This is a sticky bit. This guarantees that software can simulate **fsqrt** and **frsqrte**, and to provide a consistent interface to handle exceptions caused by square-root operations.
23	VXCVI	Floating-point invalid operation exception for invalid integer convert. This is a sticky bit.
24	VE	Floating-point invalid operation exception enable.
25	OE	Floating-point overflow exception enable.
26	UE	Floating-point underflow exception enable. This bit should not be used to determine whether denormalization should be performed on floating-point stores.
27	ZE	Floating-point zero divide exception enable.
28	XE	Floating-point inexact exception enable.
29	NI	Non-IEEE mode bit.
30–31	RN	Floating-point rounding control. 00Round to nearest 01Round toward zero 10Round toward +infinity 11Round toward -infinity

Table 3-6 illustrates the floating-point result flags that correspond to FPSCR[15:19].

Table 3-6 Floating-Point Result Flags in FPSCR

Result Flags (Bits 15–19) C<>=?	Result value class
10001	Quiet NaN
01001	– Infinity
01000	– Normalized number
11000	– Denormalized number
10010	– Zero
00010	+ Zero
10100	+ Denormalized number
00100	+ Normalized number
00101	+ Infinity

3.7.4 Condition Register (CR)

The condition register (CR) is a 32-bit register that reflects the result of certain operations and provides a mechanism for testing and branching. The bits in the CR are grouped into eight 4-bit fields, CR0 to CR7.

CR — Condition Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
CR0				CR1				CR2				CR3				CR4				CR5				CR6				CR7			

RESET: UNCHANGED

The CR fields can be set in the following ways:

- Specified fields of the CR can be set by a move instruction (**mtcrf**) to the CR from a GPR.
- Specified fields of the CR can be moved from one CR*x* field to another with the **mcrf** instruction**.**
- A specified field of the CR can be set by a move instruction (**mcrxr**) to the CR from the XER.
- Condition register logical instructions can be used to perform logical operations on specified bits in the condition register.
- CR0 can be the implicit result of an integer operation.
- A specified CR field can be the explicit result of an integer compare instruction.

Instructions are provided to test individual CR bits.

3.7.4.1 Condition Register CR0 Field Definition

In most integer instructions, when the CR is set to reflect the result of the operation (that is, when Rc = 1), and for **addic.**, **andi.**, and **andis.**, the first three bits of CR0 are set by an algebraic comparison of the result to zero; the fourth bit of CR0 is copied from XER[SO]. For integer instructions, CR[0:3] are set to reflect the result as a signed quantity. The result as an unsigned quantity or a bit string can be deduced from the EQ bit.

The CR0 bits are interpreted as shown in **Table 3-7**. If any portion of the result (the 32-bit value placed into the destination register) is undefined, the value placed in the first three bits of CR0 is undefined.

Table 3-7 Bit Descriptions for CR0 Field of CR

CR0 Bit	Description
0	Negative (LT) — This bit is set when the result is negative.
1	Positive (GT) — This bit is set when the result is positive (and not zero).
2	Zero (EQ) — This bit is set when the result is zero.
3	Summary overflow (SO) — This is a copy of the final state of XER[SO] at the completion of the instruction.

3.7.4.2 Condition Register CR1 Field Definition

In all floating-point instructions when the CR is set to reflect the result of the operation (that is, when Rc = 1), the CR1 field (bits 4 to 7 of the CR) is copied from FPSCR[0:3] to indicate the floating-point exception status. For more information about the FPSCR, see **3.7.3 Floating-Point Status and Control Register (FPSCR)**. The bit descriptions for the CR1 field are shown in **Table 3-8**.

Table 3-8 Bit Descriptions for CR1 Field of CR

CR1 Bit	Description
0	Floating-point exception (FX) — This is a copy of the final state of FPSCR[FX] at the completion of the instruction.
1	Floating-point enabled exception (FEX) — This is a copy of the final state of FPSCR[FEX] at the completion of the instruction.
2	Floating-point invalid exception (VX) — This is a copy of the final state of FPSCR[VX] at the completion of the instruction.
3	Floating-point overflow exception (OX) — This is a copy of the final state of FPSCR[OX] at the completion of the instruction.

3.7.4.3 Condition Register CR*n* Field — Compare Instruction

When a specified CR field is set by a compare instruction, the bits of the specified field are interpreted as shown in **Table 3-9**. A condition register field can also be accessed by the **mfcr**, **mcrf**, and **mtcrf** instructions.

Table 3-9 CR*n* Field Bit Descriptions for Compare Instructions

CR*n* Bit[1]	Description
0	Less than, floating-point less than (LT, FL). For integer compare instructions, (**r**A) < SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) < (**fr**B).
1	Greater than, floating-point greater than (GT, FG). For integer compare instructions, (**r**A) > SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) > (**fr**B).
2	Equal, floating-point equal (EQ, FE). For integer compare instructions, (**r**A) = SIMM, UIMM, or (**r**B). For floating-point compare instructions, (**fr**A) = (**fr**B).
3	Summary overflow, floating-point unordered (SO, FU). For integer compare instructions, this is a copy of the final state of XER[SO] at the completion of the instruction. For floating-point compare instructions, one or both of (**fr**A) and (**fr**B) is not a number (NaN).

NOTES:
1. Here, the bit indicates the bit number in any one of the four-bit subfields, CR0–CR7

3.7.5 Integer Exception Register (XER)

The integer exception register (XER) is a user-level, 32-bit register.

XER — Integer Exception Register **SPR 1**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SO		OV		CA		Reserved																			BYTES						
RESET:																															
U	U	U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U

The bit definitions for XER, shown in **Table 3-10**, are based on the operation of an instruction considered as a whole, not on intermediate results. For example, the result of the Subtract from Carrying (**subfc***x*) instruction is specified as the sum of three values. This instruction sets bits in the XER based on the entire operation, not on an intermediate sum.

In most cases, reserved fields in registers are ignored when written to and return zero when read. However, XER[16:23] are set to the value written to them and return that value when read.

Table 3-10 Integer Exception Register Bit Definitions

Bit(s)	Name	Description
0	SO	Summary Overflow (SO) — The summary overflow bit is set whenever an instruction sets the overflow bit (OV) to indicate overflow and remains set until software clears it. It is not altered by compare instructions or other instructions that cannot overflow.
1	OV	Overflow (OV) — The overflow bit is set to indicate that an overflow has occurred during execution of an instruction. Integer and subtract instructions having OE=1 set OV if the carry out of bit 0 is not equal to the carry out of bit 1, and clear it otherwise. The OV bit is not altered by compare instructions or other instructions that cannot overflow.
2	CA	Carry (CA) — In general, the carry bit is set to indicate that a carry out of bit 0 occurred during execution of an instruction. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA to one if there is a carry out of bit 0, and clear it otherwise. The CA bit is not altered by compare instructions or other instructions that cannot carry, except that shift right algebraic instructions set the CA bit to indicate whether any ‘1’ bits have been shifted out of a negative quantity.
3:24	—	Reserved
25:31	BYTES	This field specifies the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

3.7.6 Link Register (LR)

The 32-bit link register supplies the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction, and can be used to hold the logical address of the instruction that follows a branch and link instruction.

Note that although the two least-significant bits can accept any values written to them, they are ignored when the LR is used as an address.

Both conditional and unconditional branch instructions include the option of placing the effective address of the instruction following the branch instruction in the LR. This is done regardless of whether the branch is taken.

LR — Link Register **SPR 8**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Branch Address																															

RESET: UNCHANGED

3.7.7 Count Register (CTR)

The count register (CTR) is a 32-bit register for holding a loop count that can be decremented during execution of branch instructions that contain an appropriately coded BO field. If the value in CTR is 0 before being decremented, it is –1 afterward. The count register provides the branch target address for the Branch Conditional to Count Register (**bcctr*x***) instruction.

CTR — Count Register **SPR 9**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Loop Count																															

RESET: UNCHANGED

3.8 PowerPC VEA Register Set — Time Base

The PowerPC virtual environment architecture (VEA) defines registers in addition to those in the UISA register set. The PowerPC VEA register set can be accessed by all software with either user- or supervisor-level privileges.

The PowerPC VEA includes the time base facility (TB), a 64-bit structure that contains a 64-bit unsigned integer that is incremented periodically. The frequency at which the counter is updated is implementation-dependent. For details on the time base clock in the MPC555 / MPC556, refer to **6.7 MPC555 / MPC556 Time Base (TB)**, **8.6 MPC555 / MPC556 Internal Clock Signals**, and **8.12.1 System Clock Control Register (SCCR)**.

The TB consists of two 32-bit registers: time base upper (TBU) and time base lower (TBL). In the context of the VEA, user-level applications are permitted read-only access to the TB. The OEA defines supervisor-level access to the TB for writing values to the TB. Different SPR encodings are provided for reading and writing the time base.

TB — Time Base (Read Only) **SPR 268, 269**

0	31	32	63
TBU		TBL	

RESET: UNCHANGED

Table 3-11 Time Base Field Definitions (Read Only)

Bits	Name	Description
0-31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32-63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

In 32-bit PowerPC implementations such as the RCPU, it is not possible to read the entire 64-bit time base in a single instruction. The **mftb** simplified mnemonic copies the lower half of the time base register (TBL) to a GPR, and the **mftbu** simplified mnemonic copies the upper half of the time base (TBU) to a GPR.

3.9 PowerPC OEA Register Set

The PowerPC operating environment architecture (OEA) includes a number of SPRs and other registers that are accessible only by supervisor-level instructions. Some SPRs are RCPU-specific; some RCPU SPRs may not be implemented in other PowerPC processors, or may not be implemented in the same way.

3.9.1 Machine State Register (MSR)

The machine state register is a 32-bit register that defines the state of the processor. When an exception occurs, the current contents of the MSR are loaded into SRR1, and the MSR is updated to reflect the exception-processing machine state. The MSR can also be modified by the **mtmsr**, **sc**, and **rfi** instructions. It can be read by the **mfmsr** instruction.

MSR — Machine State Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED													POW	0	ILE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
EE	PR	FP	ME	FE0	SE	BE	FE1	0	IP	IR	DR	RE-SERVED	DC-MPEN [2]	RI	LE
RESET:															
0	0	0	U	0	0	0	0	0	ID1 [1]	0	0	0	0	0	0

NOTES:
1. Reset value of this bit depends on the value of the internal data bus line during reset.
2. This bit is only available on the MPC556.

Table 3-12 shows the bit definitions for the MSR.

Table 3-12 Machine State Register Bit Descriptions

Bit(s)	Name	Description
0:12	—	Reserved
13	POW	Power management enable 0 = Power management disabled (normal operation mode) 1 = Power management enabled (reduced power mode)
14	—	Reserved
15	ILE	Exception little-endian mode. When an exception occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the exception. 0 = Processor runs in big-endian mode during exception processing. 1 = Processor runs in little-endian mode during exception processing.
16	EE	External interrupt enable. Interrupts should only be negated while the EE bit is disabled (0). Software should disable interrupts in the CPU core prior to masking or disabling any interrupt which might be currently pending at the CPU core. For external interrupts, it is recommended that the edge triggered interrupt scheme be used. 0 = The processor delays recognition of external interrupts and decrementer exception conditions. 1 = The processor is enabled to take an external interrupt or the decrementer exception.
17	PR	Privilege level 0 = The processor can execute both user- and supervisor-level instructions. 1 = The processor can only execute user-level instructions.
18	FP	Floating-point available 0 = The processor prevents dispatch of floating-point instructions, including floating-point loads, stores and moves. Floating-point enabled program exceptions can still occur and the FPRs can still be accessed. 1 = The processor can execute floating-point instructions, and can take floating-point enabled exception type program exceptions.
19	ME	Machine check enable 0 = Machine check exceptions are disabled. 1 = Machine check exceptions are enabled.
20	FE0	Floating-point exception mode 0 (See **Table 3-13**.)
21	SE	Single-step trace enable 0 = The processor executes instructions normally. 1 = The processor generates a single-step trace exception upon the successful execution of the next instruction. When this bit is set, the processor dispatches instructions in strict program order. Successful execution means the instruction caused no other exception. Single-step tracing may not be present on all implementations.
22	BE	Branch trace enable 0 = No trace exception occurs when a branch instruction is completed 1 = Trace exception occurs when a branch instruction is completed
23	FE1	Floating-point exception mode 1 (See **Table 3-13**.)
24	—	Reserved
25	IP	Exception prefix. The setting of this bit specifies the location of the exception vector table. 0 = Exception vector table starts at the physical address 0x0000 0000. 1 = Exception vector table starts at the physical address 0xFFF0 0000.
26	IR	Instruction relocation. 0 = Instruction address translation is off, the BBC IMPU does not check for address permission attributes. 1 = Instruction address translation is on, the BBC IMPU checks for address permission attributes.
27	DR	Data relocation 0 = Data address translation is off, the L2U DMPU does not check for address permission attributes. 1 = Data address translation is on, the L2U DMPU checks for address permission attributes.

Table 3-12 Machine State Register Bit Descriptions (Continued)

Bit(s)	Name	Description
28	—	Reserved
29	DC-MPEN[1]	Decompression On/Off 0 = RCPU Normal Operation 1 = RCPU is running in Compressed mode
30	RI	Recoverable exception (for machine check and non-maskable breakpoint exceptions) 0 = Machine state is not recoverable. 1 = Machine state is recoverable.
31	LE	Little-endian mode 0 = Processor operates in big-endian mode during normal processing. 1 = Processor operates in little-endian mode during normal processing.

NOTES:
1. This bit is only available on the MPC556.

The floating-point exception mode bits are interpreted as shown in **Table 3-13**.

Table 3-13 Floating-Point Exception Mode Bits

FE[0:1]	Mode
00	Ignore exceptions mode — Floating-point exceptions do not cause the floating-point assist error handler to be invoked.
01, 10, 11	Floating-point precise mode — The system floating-point assist error handler is invoked precisely at the instruction that caused the enabled exception.

3.9.2 DAE/Source Instruction Service Register (DSISR)

The 32-bit DSISR identifies the cause of data access and alignment exceptions.

DSISR — DAE/Source Instruction Service Register **SPR 18**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
DSISR																															

RESET: UNCHANGED

3.9.3 Data Address Register (DAR)

After an alignment exception, the DAR is set to the effective address of a load or store element.

DAR — Data Address Register **SPR 19**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Data Address																															

RESET: UNCHANGED

3.9.4 Time Base Facility (TB) — OEA

As described in **3.8 PowerPC VEA Register Set — Time Base**, the time base (TB) provides a 64-bit incrementing counter. The VEA defines user-level, read-only access to the TB. Writing to the TB is reserved for supervisor-level applications such as operating systems and bootstrap routines. The OEA defines supervisor-level, write access to the TB.

TB — Time Base (Write Only) **SPR 284, 285**

0 – 31	32 – 63
TBU	TBL

RESET: UNCHANGED

Table 3-14 Time Base Field Definitions (Write Only)

Bits	Name	Description
0:31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32:63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

The TB can be written to at the supervisor privilege level only. The **mttbl** and **mttbu** simplified mnemonics write the lower and upper halves of the TB, respectively. The **mtspr**, **mttbl**, and **mttbu** instructions treat TBL and TBU as separate 32-bit registers; setting one leaves the other unchanged. It is not possible to write the entire 64-bit time base in a single instruction.

For information about reading the time base, refer to **3.8 PowerPC VEA Register Set — Time Base**.

3.9.5 Decrementer Register (DEC)

The decrementer (DEC, SPR 22) is a 32-bit decrementing counter defined by the MPC555 / MPC556 to provide a decrementer exception after a programmable delay. The DEC satisfies the following requirements:

- Loading a GPR from the DEC has no effect on the DEC.
- Storing a GPR to the DEC replaces the value in the DEC with the value in the GPR.
- Whenever bit 0 of the DEC changes from zero to one, a decrementer exception request (unless masked) is signaled. Multiple DEC exception requests may be received before the first exception occurs; however, any additional requests are canceled when the exception occurs for the first request.
- If the DEC is altered by software and the content of bit 0 is changed from zero to one, an exception request is signaled.
- PORESET resets and stops the decrementer, HRESET/SRESET do not.

The decrementer frequency is based on a subdivision of the processor clock. A bit in the system clock control register (SCCR) in the SIU determines the clock source of both the decrementer and the time base. For details on the decrementer and time base

clock in the MPC555 / MPC556, refer to **6.6 MPC555 / MPC556 Decrementer**, **8.6 MPC555 / MPC556 Internal Clock Signals**, and **8.12.1 System Clock Control Register (SCCR)**.

The DEC does not run after power-up and must be enabled by setting the TBE bit in the TBSCR register, see **Table 6-16**. Power-on reset stops its counting and clears the register. A decrementer exception may be signaled to software prior to initialization.

DEC — Decrementer Register **SPR 22**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Decrementing Counter																															

RESET: UNCHANGED

3.9.6 Machine Status Save/Restore Register 0 (SRR0)

The machine status save/restore register 0 (SRR0) is a 32-bit register that identifies where instruction execution should resume when an **rfi** instruction is executed following an exception. It also holds the effective address of the instruction that follows the System Call (**sc**) instruction.

When an exception occurs, SRR0 is set to point to an instruction such that all prior instructions have completed execution and no subsequent instruction has begun execution. The instruction addressed by SRR0 may not have completed execution, depending on the exception type. SRR0 addresses either the instruction causing the exception or the immediately following instruction. The instruction addressed can be determined from the exception type and status bits.

SRR0 — Machine Status Save/Restore Register 0 **SPR 26**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SRR0																															

RESET: UNDEFINED

When an exception occurs, SRR0 is set to point to an instruction such that all prior instructions have completed execution and no subsequent instruction has begun execution. The instruction addressed by SRR0 may not have completed execution, depending on the exception type. SRR0 addresses either the instruction causing the exception or the immediately following instruction. The instruction addressed can be determined from the exception type and status bits.

3.9.7 Machine Status Save/Restore Register 1 (SRR1)

SRR1 is a 32-bit register used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.

SRR1 — Machine Status Save/Restore Register 1 **SPR 27**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SRR1																															

RESET: UNDEFINED

In general, when an exception occurs, SRR1[0:15] are loaded with exception-specific information, and MSR[16:31] are placed into SRR1[16:31].

3.9.8 General SPRs (SPRG0–SPRG3)

SPRG0–SPRG3 are 32-bit registers provided for general operating system use, such as performing a fast-state save and for supporting multiprocessor implementations. SPRG0–SPRG3 are shown below.

SPRG0–SPRG3 — General Special-Purpose Registers 0–3 **SPR 272 – SPR 275**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SPRG0																															
SPRG1																															
SPRG2																															
SPRG3																															

RESET: UNCHANGED

Uses for SPRG0–SPRG3 are shown in **Table 3-15**.

Table 3-15 Uses of SPRG0–SPRG3

Register	Description
SPRG0	Software may load a unique physical address in this register to identify an area of memory reserved for use by the exception handler. This area must be unique for each processor in the system.
SPRG1	This register may be used as a scratch register by the exception handler to save the content of a GPR. That GPR then can be loaded from SPRG0 and used as a base register to save other GPRs to memory.
SPRG2	This register may be used by the operating system as needed.
SPRG3	This register may be used by the operating system as needed.

3.9.9 Processor Version Register (PVR)

The PVR is a 32-bit, read-only register that identifies the version and revision level of the PowerPC processor. The contents of the PVR can be copied to a GPR by the **mfspr** instruction. Read access to the PVR is available in supervisor mode only; write access is not provided.

PVR — Processor Version Register **SPR 287**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
VERSION																REVISION															

RESET: UNCHANGED

Table 3-16 Processor Version Register Bit Descriptions

Bit(s)	Name	Description
0:15	VERSION	A 16-bit number that identifies the version of the processor and of the PowerPC architecture. MPC555 / MPC556 value is 0x0002.
16:31	REVISION	A 16-bit number that distinguishes between various releases of a particular version. The MPC555 / MPC556 value is 0x0020.

3.9.10 Implementation-Specific SPRs

The MPC555 / MPC556 includes several implementation-specific SPRs that are not defined by the PowerPC architecture. These registers can be accessed by supervisor-level instructions only. These registers are listed in **Table 3-2** and **Table 3-3**.

3.9.10.1 EIE, EID, and NRI Special-Purpose Registers

The RCPU includes three implementation-specific SPRs to facilitate the software manipulation of the MSR[RI] and MSR[EE] bits. Issuing the **mtspr** instruction with one of these registers as an operand causes the RI and EE bits to be set or cleared as shown in **Table 3-17**.

A read (**mfspr**) of any of these locations is treated as an unimplemented instruction, resulting in a software emulation exception.

Table 3-17 EIE, EID, AND NRI Registers

SPR Number (Decimal)	Mnemonic	MSR[EE]	MSR[RI]
80	EIE	1	1
81	EID	0	1
82	NRI	0	0

3.9.10.2 Floating-Point Exception Cause Register (FPECR)

The FPECR is a 32-bit supervisor-level internal status and control register used by the floating-point assist firmware envelope. It contains four status bits indicating whether the result of the operation is tiny and whether any of three source operands are denormalized. In addition, it contains one control bit to enable or disable SIE mode. This register must not be accessed by user code.

FPECR — Floating-Point Exception Cause Register **SPR 1022**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SIE	RESERVED														
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED												DNC	DNB	DNA	TR
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A listing of FPECR bit descriptions is shown in **Table 3-18**.

Table 3-18 FPECR Bit Descriptions

Bit(s)	Name	Description
0	SIE	SIE mode control bit 0 = Disable SIE mode 1 = Enable SIE mode
[1:27]	—	Reserved
28	DNC	Source operand C denormalized status bit 0 = Source operand C is not denormalized 1 = Source operand C is denormalized
29	DNB	Source operand B denormalized status bit 0 = Source operand B is not denormalized 1 = Source operand B is denormalized
30	DNA	Source operand A denormalized status bit 0 = Source operand A is not denormalized 1 = Source operand A is denormalized
31	TR	Floating-point tiny result 0 = Floating-point result is not tiny 1 = Floating-point result is tiny

NOTE

Software must insert a **sync** instruction before reading the FPECR.

3.9.10.3 Additional Implementation-Specific Registers

Refer to the following sections for details on additional implementation-specific registers in the MPC555 / MPC556:

- **4.6 Burst Buffer Programming Model**
- **6.13.1.2 Internal Memory Map Register**
- **11.8 L2U Programming Model**
- **SECTION 21 DEVELOPMENT SUPPORT**

3.10 Instruction Set

All PowerPC instructions are encoded as single words (32 bits). Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions include computational and logical instructions.
  - — Integer arithmetic instructions
  - — Integer compare instructions
  - — Integer logical instructions
  - — Integer rotate and shift instructions
- Floating-point instructions include floating-point computational instructions, as well as instructions that affect the floating-point status and control register (FPSCR).
  - — Floating-point arithmetic instructions
  - — Floating-point multiply/add instructions
  - — Floating-point rounding and conversion instructions
  - — Floating-point compare instructions
  - — Floating-point status and control instructions
- Load/store instructions include integer and floating-point load and store instructions.
  - — Integer load and store instructions
  - — Integer load and store multiple instructions
  - — Floating-point load and store
  - — Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - — Branch and trap instructions
  - — Condition register logical instructions
- Processor control instructions are used for synchronizing memory accesses.
  - — Move to/from SPR instructions
  - — Move to/from MSR
  - — Synchronize
  - — Instruction synchronize

Note that this grouping of the instructions does not indicate which execution unit executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs.

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

PowerPC processors follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

3.10.1 Instruction Set Summary

Table 3-19 provides a summary of RCPU instructions. Refer to the ***RCPU Reference Manual*** **(RCPURM/AD)** for a detailed description of the instruction set.

Table 3-19 Instruction Set Summary

Mnemonic	Operand Syntax	Name
add (add. addo addo.)	**r**D,**r**A,**r**B	Add
addc (addc. addco addco.)	**r**D,**r**A,**r**B	Add Carrying
adde (adde. addeo addeo.)	**r**D,**r**A,**r**B	Add Extended
addi	**r**D,**r**A,SIMM	Add Immediate
addic	**r**D,**r**A,SIMM	Add Immediate Carrying
addic.	**r**D,**r**A,SIMM	Add Immediate Carrying and Record
addis	**r**D,**r**A,SIMM	Add Immediate Shifted
addme (addme. addmeo addmeo.)	**r**D,**r**A	Add to Minus One Extended
addze (addze. addzeo addzeo.)	**r**D,**r**A	Add to Zero Extended
and (and.)	**r**A,**r**S,**r**B	AND
andc (andc.)	**r**A,**r**S,**r**B	AND with Complement
andi.	**r**A,**r**S,UIMM	AND Immediate
andis.	**r**A,**r**S,UIMM	AND Immediate Shifted
b (ba bl bla)	target_addr	Branch
bc (bca bcl bcla)	BO,BI,target_addr	Branch Conditional
bcctr (bcctrl)	BO,BI	Branch Conditional to Count Register
bclr (bclrl)	BO,BI	Branch Conditional to Link Register
cmp	**crf**D,L,**r**A,**r**B	Compare
cmpi	**crf**D,L,**r**A,SIMM	Compare Immediate
cmpl	**crf**D,L,**r**A,**r**B	Compare Logical
cmpli	**crf**D,L,**r**A,UIMM	Compare Logical Immediate
cntlzw (cntlzw.)	**r**A,**r**S	Count Leading Zeros Word
crand	**crb**D,**crb**A,**crb**B	Condition Register AND
crandc	**crb**D,**crb**A, **crb**B	Condition Register AND with Complement
creqv	**crb**D,**crb**A, **crb**B	Condition Register Equivalent
crnand	**crb**D,**crb**A,**crb**B	Condition Register NAND
crnor	**crb**D,**crb**A,**crb**B	Condition Register NOR
cror	**crb**D,**crb**A,**crb**B	Condition Register OR

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
crorc	**crb**D,**crb**A, **crb**B	Condition Register OR with Complement
crxor	**crb**D,**crb**A,**crb**B	Condition Register XOR
divw (divw. divwo divwo.)	**r**D,**r**A,**r**B	Divide Word
divwu divwu. divwuo divwuo.	**r**D,**r**A,**r**B	Divide Word Unsigned
eieio	—	Enforce In-Order Execution of I/O
eqv (eqv.)	**r**A,**r**S,**r**B	Equivalent
extsb (extsb.)	**r**A,**r**S	Extend Sign Byte
extsh (extsh.)	**r**A,**r**S	Extend Sign Half Word
fabs (fabs.)	**fr**D,**fr**B	Floating Absolute Value
fadd (fadd.)	**fr**D,**fr**A,**fr**B	Floating Add (Double-Precision)
fadds (fadds.)	**fr**D,**fr**A,**fr**B	Floating Add Single
fcmpo	**crf**D,**fr**A,**fr**B	Floating Compare Ordered
fcmpu	**crf**D,**fr**A,**fr**B	Floating Compare Unordered
fctiw (fctiw.)	**fr**D,**fr**B	Floating Convert to Integer Word
fctiwz (fctiwz.)	**fr**D,**fr**B	Floating Convert to Integer Word with Round toward Zero
fdiv (fdiv.)	**fr**D,**fr**A,**fr**B	Floating Divide (Double-Precision)
fdivs (fdivs.)	**fr**D,**fr**A,**fr**B	Floating Divide Single
fmadd (fmadd.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Add (Double-Precision)
fmadds (fmadds.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Add Single
fmr (fmr.)	**fr**D,**fr**B	Floating Move Register
fmsub (fmsub.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Subtract (Double-Precision)
fmsubs (fmsubs.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Subtract Single
fmul (fmul.)	**fr**D,**fr**A,**fr**C	Floating Multiply (Double-Precision)
fmuls (fmuls.)	**fr**D,**fr**A,**fr**C	Floating Multiply Single
fnabs (fnabs.)	**fr**D,**fr**B	Floating Negative Absolute Value
fneg (fneg.)	**fr**D,**fr**B	Floating Negate
fnmadd (fnmadd.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Add (Double-Precision)
fnmadds (fnmadds.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Add Single
fnmsub (fnmsub.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Subtract (Double-Precision)
fnmsubs (fnmsubs.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Subtract Single
frsp (frsp.)	**fr**D,**fr**B	Floating Round to Single
fsub (fsub.)	**fr**D,**fr**A,**fr**B	Floating Subtract (Double-Precision)
fsubs (fsubs.)	**fr**D,**fr**A,**fr**B	Floating Subtract Single
isync	—	Instruction Synchronize
lbz	**r**D,d(**r**A)	Load Byte and Zero
lbzu	**r**D,d(**r**A)	Load Byte and Zero with Update
lbzux	**r**D,**r**A,**r**B	Load Byte and Zero with Update Indexed
lbzx	**r**D,**r**A,**r**B	Load Byte and Zero Indexed
lfd	**fr**D,d(**r**A)	Load Floating-Point Double
lfdu	**fr**D,d(**r**A)	Load Floating-Point Double with Update

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
lfdux	**fr**D,**r**A,**r**B	Load Floating-Point Double with Update Indexed
lfdx	**fr**D,**r**A,**r**B	Load Floating-Point Double Indexed
lfs	**fr**D,d**(r**A**)**	Load Floating-Point Single
lfsu	**fr**D,d**(r**A**)**	Load Floating-Point Single with Update
lfsux	**fr**D,**r**A,**r**B	Load Floating-Point Single with Update Indexed
lfsx	**fr**D,**r**A,**r**B	Load Floating-Point Single Indexed
lha	**r**D,d(**r**A)	Load Half-Word Algebraic
lhau	**r**D,d(**r**A)	Load Half-Word Algebraic with Update
lhaux	**r**D,**r**A,**r**B	Load Half-Word Algebraic with Update Indexed
lhax	**r**D,**r**A,**r**B	Load Half-Word Algebraic Indexed
lhbrx	**r**D,**r**A,**r**B	Load Half-Word Byte-Reverse Indexed
lhz	**r**D,d(**r**A)	Load Half-Word and Zero
lhzu	**r**D,d(**r**A)	Load Half-Word and Zero with Update
lhzux	**r**D,**r**A,**r**B	Load Hal-Word and Zero with Update Indexed
lhzx	**r**D,**r**A,**r**B	Load Half-Word and Zero Indexed
lmw	**r**D,d(**r**A)	Load Multiple Word
lswi	**r**D,**r**A,NB	Load String Word Immediate
lswx	**r**D,**r**A,**r**B	Load String Word Indexed
lwarx	**r**D,**r**A,**r**B	Load Word and Reserve Indexed
lwbrx	**r**D,**r**A,**r**B	Load Word Byte-Reverse Indexed
lwz	**r**D,d(**r**A)	Load Word and Zero
lwzu	**r**D,d(**r**A)	Load Word and Zero with Update
lwzux	**r**D,**r**A,**r**B	Load Word and Zero with Update Indexed
lwzx	**r**D,**r**A,**r**B	Load Word and Zero Indexed
mcrf	**crf**D,**crf**S	Move Condition Register Field
mcrfs	**crf**D,**crf**S	Move to Condition Register from FPSCR
mcrxr	**crf**D	Move to Condition Register from XER
mfcr	**r**D	Move from Condition Register
mffs (mffs.)	**fr**D	Move from FPSCR
mfmsr	**r**D	Move from Machine State Register
mfspr	**r**D,SPR	Move from Special Purpose Register
mftb	**r**D, TBR	Move from Time Base
mtcrf	CRM,**r**S	Move to Condition Register Fields
mtfsb0 (mtfsb0.)	**crb**D	Move to FPSCR Bit 0
mtfsb1 (mtfsb1.)	**crb**D	Move to FPSCR Bit 1
mtfsf (mtfsf.)	FM,**fr**B	Move to FPSCR Fields
mtfsfi (mtfsfi.)	**crf**D,IMM	Move to FPSCR Field Immediate
mtmsr	**r**S	Move to Machine State Register
mtspr	SPR,**r**S	Move to Special Purpose Register
mulhw (mulhw.)	**r**D,**r**A,**r**B	Multiply High Word
mulhwu (mulhwu.)	**r**D,**r**A,**r**B	Multiply High Word Unsigned

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
mulli	**r**D,**r**A,SIMM	Multiply Low Immediate
mullw (mullw. mullwo mullwo.)	**r**D,**r**A,**r**B	Multiply Low
nand (nand.)	**r**A,**r**S,**r**B	NAND
neg (neg. nego nego.)	**r**D,**r**A	Negate
nor (nor.)	**r**A,**r**S,**r**B	NOR
or (or.)	**r**A,**r**S,**r**B	OR
orc (orc.)	**r**A,**r**S,**r**B	OR with Complement
ori	**r**A,**r**S,UIMM	OR Immediate
oris	**r**A,**r**S,UIMM	OR Immediate Shifted
rfi	—	Return from Interrupt
rlwimi (rlwimi.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then Mask Insert
rlwinm (rlwinm.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then AND with Mask
rlwnm (rlwnm.)	**r**A,**r**S,**r**B,MB,ME	Rotate Left Word then AND with Mask
sc	—	System Call
slw (slw.)	**r**A,**r**S,**r**B	Shift Left Word
sraw (sraw.)	**r**A,**r**S,**r**B	Shift Right Algebraic Word
srawi (srawi.)	**r**A,**r**S,SH	Shift Right Algebraic Word Immediate
srw (srw.)	**r**A,**r**S,**r**B	Shift Right Word
stb	**r**S,d(**r**A)	Store Byte
stbu	**r**S,d(**r**A)	Store Byte with Update
stbux	**r**S,**r**A,**r**B	Store Byte with Update Indexed
stbx	**r**S,**r**A,**r**B	Store Byte Indexed
stfd	**fr**S,d(**r**A)	Store Floating-Point Double
stfdu	**fr**S,d(**r**A)	Store Floating-Point Double with Update
stfdux	**fr**S,**r**B	Store Floating-Point Double with Update Indexed
stfdx	**fr**S,**r**B	Store Floating-Point Double Indexed
stfiwx	**fr**S,**r**B	Store Floating-Point as Integer Word Indexed
stfs	**fr**S,d(**r**A)	Store Floating-Point Single
stfsu	**fr**S,d(**r**A)	Store Floating-Point Single with Update
stfsux	**fr**S,**r**B	Store Floating-Point Single with Update Indexed
stfsx	**fr**S,**r** B	Store Floating-Point Single Indexed
sth	**r**S,d(**r**A)	Store Half Word
sthbrx	**r**S,**r**A,**r**B	Store Half Word Byte-Reverse Indexed
sthu	**r**S,d(**r**A)	Store Half Word with Update
sthux	**r**S,**r**A,**r**B	Store Half Word with Update Indexed
sthx	**r**S,**r**A,**r**B	Store Half Word Indexed
stmw	**r**S,d(**r**A)	Store Multiple Word
stswi	**r**S,**r**A,NB	Store String Word Immediate
stswx	**r**S,**r**A,**r**B	Store String Word Indexed
stw	**r**S,d(**r**A)	Store Word

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
stwbrx	**r**S,**r**A,**r**B	Store Word Byte-Reverse Indexed
stwcx.	**r**S,**r**A,**r**B	Store Word Conditional Indexed
stwu	**r**S,d(**r**A)	Store Word with Update
stwux	**r**S,**r**A,**r**B	Store Word with Update Indexed
stwx	**r**S,**r**A,**r**B	Store Word Indexed
subf (subf. subfo subfo.)	**r**D,**r**A,**r**B	Subtract From
subfc (subfc. subfco subfco.)	**r**D,**r**A,**r**B	Subtract from Carrying
subfe (subfe. subfeo subfeo.)	**r**D,**r**A,**r**B	Subtract from Extended
subfic	**r**D,**r**A,SIMM	Subtract from Immediate Carrying
subfme (subfme. subfmeo subfmeo.)	**r**D,**r**A	Subtract from Minus One Extended
subfze (subfze. subfzeo subfzeo.)	**r**D,**r**A	Subtract from Zero Extended
sync	—	Synchronize
tw	TO,**r**A,**r**B	Trap Word
twi	TO,**r**A,SIMM	Trap Word Immediate
xor (xor.)	**r**A,**r**S,**r**B	XOR
xori	**r**A,**r**S,UIMM	XOR Immediate
xoris	**r**A,**r**S,UIMM	XOR Immediate Shifted

3.10.2 Recommended Simplified Mnemonics

To simplify assembly language coding, a set of alternative mnemonics is provided for some frequently used operations (such as no-op, load immediate, load address, move register, and complement register).

For a complete list of simplified mnemonics, see the ***RCPU Reference Manual* (RCPURM/AD)**. Programs written to be portable across the various assemblers for the PowerPC architecture should not assume the existence of mnemonics not described in that manual.

3.10.3 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + 16-bit offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the storage operand is considered to wrap around from the maximum effective address to effective address 0.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

3.11 Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. When exceptions occur, information about the state of the processor is saved to certain registers, and the processor begins execution at an address (exception vector) predetermined for each exception. Processing of exceptions occurs in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception — for example, the DAE/source instruction service register (DSISR). Additionally, some exception conditions can be explicitly enabled or disabled by software.

3.11.1 Exception Classes

The MPC555 / MPC556 exception classes are shown in **Table 3-20**.

Table 3-20 MPC555 / MPC556 Exception Classes

Class	Exception Type
Asynchronous, unordered	Machine check System reset
Asynchronous, ordered	External interrupt Decrementer
Synchronous (ordered, precise)	Instruction-caused exceptions

3.11.2 Ordered Exceptions

In the MPC555 / MPC556, all exceptions except for reset, debug port non-maskable interrupts, and machine check exceptions are ordered. Ordered exceptions satisfy the following criteria:

- Only one exception is reported at a time. If, for example, a single instruction encounters multiple exception conditions, those conditions are encountered sequentially. After the exception handler handles an exception, instruction execution continues until the next exception condition is encountered.
- When the exception is taken, no program state is lost.

3.11.3 Unordered Exceptions

Unordered exceptions may be reported at any time and are not guaranteed to preserve program state information. The processor can never recover from a reset exception. It can recover from other unordered exceptions in most cases. However, if a

debug port non-maskable interrupt or machine check exception occurs during the servicing of a previous exception, the machine state information in SRR0 and SRR1 (and, in some cases, the DAR and DSISR) may not be recoverable; the processor may be in the process of saving or restoring these registers.

To determine whether the machine state is recoverable, the user can read the RI (recoverable exception) bit in SRR1. During exception processing, the RI bit in the MSR is copied to SRR1 and then cleared. The operating system should set the RI bit in the MSR at the end of each exception handler's prologue (after saving the program state) and clear the bit at the start of each exception handler's epilogue (before restoring the program state). Then, if an unordered exception occurs during the servicing of an exception handler, the RI bit in SRR1 will contain the correct value.

3.11.4 Precise Exceptions

In the MPC555 / MPC556, all synchronous (instruction-caused) exceptions are precise. When a precise exception occurs, the processor backs the machine up to the instruction causing the exception. This ensures that the machine is in its correct architecturally-defined state. The following conditions exist at the point a precise exception occurs:

1. Architecturally, no instruction following the faulting instruction in the code stream has begun execution.
2. All instructions preceding the faulting instruction appear to have completed with respect to the executing processor.
3. SRR0 addresses either the instruction causing the exception or the immediately following instruction. Which instruction is addressed can be determined from the exception type and the status bits.
4. Depending on the type of exception, the instruction causing the exception may not have begun execution, may have partially completed, or may have completed execution.

3.11.5 Exception Vector Table

The setting of the exception prefix (IP) bit in the MSR determines how exceptions are vectored. If the bit is cleared, the exception vector table begins at the physical address 0x0000 0000; if IP is set, the exception vector table begins at the physical address 0xFFF0 0000. **Table 3-21** shows the exception vector offset of the first instruction of the exception handler routine for each exception type.

NOTE

In the MPC555 / MPC556, the exception table can additionally be relocated by the BBC module to internal memory and reduce the total size required by the exception table (see **4.5 Exception Table Relocation**).

Table 3-21 Exception Vector Offset Table

Vector Offset (Hexadecimal)	Exception Type
00000	Reserved
00100	System reset, NMI interrupt
00200	Machine check
00300	Reserved
00400	Reserved
00500	External interrupt
00600	Alignment
00700	Program
00800	Floating-point unavailable
00900	Decrementer
00A00	Reserved
00B00	Reserved
00C00	System call
00D00	Trace
00E00	Floating-point assist
01000	Implementation-dependent software emulation
01100	Reserved
01200	Reserved
01300	Implementation-dependent instruction protection error
01400	Implementation-dependent data protection error
01500–01BFF	Reserved
01C00	Implementation-dependent data breakpoint
01D00	implementation-dependent instruction breakpoint
01E00	Implementation-dependent maskable external breakpoint
01F00	Implementation-dependent non-maskable external breakpoint

3.12 Instruction Timing

The MPC555 / MPC556 processor is pipelined. Because the processing of an instruction is broken into a series of stages, an instruction does not require the entire resources of the processor.

The instruction pipeline in the MPC555 / MPC556 has four stages:

1. The dispatch stage is implemented using a distributed mechanism. The central dispatch unit broadcasts the instruction to all units. In addition, scoreboard information (regarding data dependencies) is broadcast to each execution unit. Each execution unit decodes the instruction. If the instruction is not implemented, a program exception is taken. If the instruction is legal and no data dependency is found, the instruction is accepted by the appropriate execution unit, and the data found in the destination register is copied to the history buffer. If a data dependency exists, the machine is stalled until the dependency is resolved.

2. In the execute stage, each execution unit that has an executable instruction executes the instruction. (For some instructions, this occurs over multiple cycles.)
3. In the writeback stage, the execution unit writes the result to the destination register and reports to the history buffer that the instruction is completed.
4. In the retirement stage, the history buffer retires instructions in architectural order. An instruction retires from the machine if it completes execution with no exceptions and if all instructions preceding it in the instruction stream have finished execution with no exceptions. As many as six instructions can be retired in one clock.

The history buffer maintains the correct architectural machine state. An exception is taken only when the instruction is ready to be retired from the machine (i.e., after all previously-issued instructions have already been retired from the machine). When an exception is taken, all instructions following the excepting instruction are canceled, i.e., the values of the affected destination registers are restored using the values saved in the history buffer during the dispatch stage.

Figure 3-4 shows basic instruction pipeline timing.

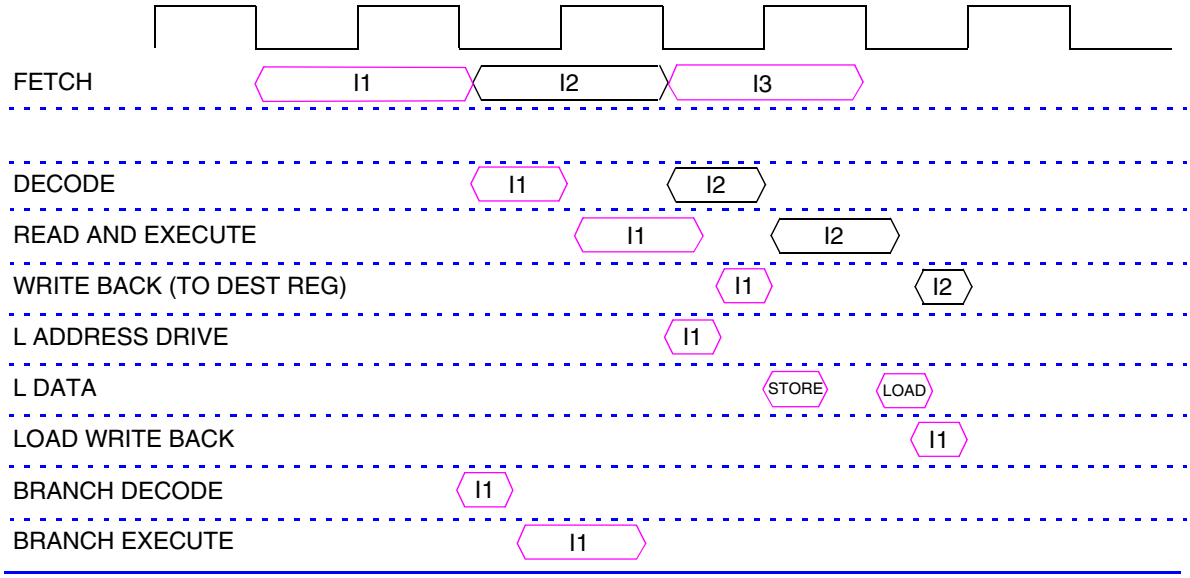


Figure 3-4 Basic Instruction Pipeline

Table 3-22 indicates the latency and blockage for each type of instruction. Latency refers to the interval from the time an instruction begins execution until it produces a result that is available for use by a subsequent instruction. Blockage refers to the interval from the time an instruction begins execution until its execution unit is available for a subsequent instruction. Note that when the blockage equals the latency, it is not possible to issue another instruction to the same unit in the same cycle in which the first instruction is being written back.

Table 3-22 Instruction Latency and Blockage

Instruction Type	Precision	Latency	Blockage
Floating-point multiply-add	Double Single	7 6	7 6
Floating-point add or subtract	Double Single	4 4	4 4
Floating-point multiply	Double Single	5 4	5 4
Floating-point divide	Double Single	17 10	17 10
Integer multiply	—	2	1 or 2[1]
Integer divide	—	2 to 11[1]	2 to 11[1]
Integer load/store	—	See note[1]	See note[1]

NOTES:
1. Refer to **Section 7 Instruction Timing,** in the ***RCPU Reference Manual*** **(RCPURM/AD)** for details.

3.13 PowerPC User Instruction Set Architecture (UISA)

3.13.1 Computation Modes

The core of the MPC555 / MPC556 is a 32-bit implementation of the PowerPC architecture. Any reference in the PowerPC Architecture Books (UISA, VEA, OEA) regarding 64-bit implementations are not supported by the core. All registers except the floating-point registers are 32 bits wide.

3.13.2 Reserved Fields

Reserved fields in instructions are described under the specific instruction definition sections. Unless otherwise stated in the specific instruction description, fields marked ***"I***, "***II"*** and "***III"*** in the instruction are discarded by the core decoding. Thus, this type of invalid form instructions yield results of the defined instructions with the appropriate field zero.

In most cases, the reserved fields in registers are ignored on write and return zeros for them on read on any control register implemented by the MPC555 / MPC556. Exception to this rule are bits 16:23 of the fixed-point exception cause register (XER) and the reserved bits of the machine state register (MSR), which are set by the source value on write and return the value last set for it on read.

3.13.3 Classes of Instructions

Non-optional instructions are implemented by the hardware. Optional instructions are executed by implementation-dependent code and any attempt to execute one of these commands causes the MPC555 / MPC556 to take the implementation-dependent software emulation interrupt (offset 0x01000 of the vector table).

Illegal and reserved instruction class instructions are supported by implementation-dependent code and, thus, the MPC555 / MPC556 hardware generates the implemen-

tation-dependent software emulation interrupt. Invalid and preferred instruction forms treatment by the MPC555 / MPC556 is described under the specific processor compliance sections.

3.13.4 Exceptions

Invocation of the system software for any instruction-caused exception in the MPC555 / MPC556 is precise, regardless of the type and setting.

3.13.5 The Branch Processor

3.13.6 Instruction Fetching

The core fetches a number of instructions into its internal buffer (the instruction pre-fetch queue) prior to execution. If a program modifies the instructions it intends to execute, it should call a system library program to ensure that the modifications have been made visible to the instruction fetching mechanism prior to execution of the modified instructions.

3.13.7 Branch Instructions

The core implements all the instructions defined for the branch processor by the **UISA** in the hardware. For performance of various instructions, refer to **Table 3-22** of this manual.

3.13.7.1 Invalid Branch Instruction Forms

Bits marked with ***z*** in the BO encoding definition are discarded by the MPC555 / MPC556 decoding. Thus, these types of invalid form instructions yield result of the defined instructions with the ***z*** bit zero. If the decrement and test CTR option is specified for the **bcctr** or **bcctrl** instructions, the target address of the branch is the new value of the CTR. Condition is evaluated correctly, including the value of the counter after decrement.

3.13.7.2 Branch Prediction

The core uses the ***y*** bit to predict path for pre-fetch. Prediction is only done for not-ready branch conditions. No prediction is done for branches to link or count register if the target address is not ready. Refer to RCPU Reference Manual *(*Conditional Branch Control) for more information.

3.13.8 The Fixed-Point Processor

3.13.8.1 Fixed-Point Instructions

The core implements the following instructions:

- Fixed-point arithmetic instructions
- Fixed-point compare instructions
- Fixed-point trap instructions
- Fixed-point logical instructions

- Fixed-point rotate and shift instructions
- Move to/from system register instructions

All instructions are defined for the fixed-point processor in the **UISA** in the hardware. For performance of the various instructions, refer to **Table 3-22**.

— **Move To/From System Register Instructions.** Move to/from invalid special registers in which spr0 = 1 yields invocation of the privilege instruction error interrupt handler if the processor is in problem state. For a list of all implemented special registers, refer to **Table 3-2 Supervisor-Level SPRs**, and **Table 3-3 Development Support SPRs.**

— **Fixed-Point Arithmetic Instructions.** If an attempt is made to perform any of the divisions in the divw[o][.] instruction:
0x80000000 ÷ -1
<anything> ÷ 0
Then, the contents of RT are 0x80000000 and if Rc =1, the contents of bits in CR field 0 are LT = 1, GT = 0, EQ = 0, and SO is set to the correct value. If an attempt is made to perform any of the divisions in the divw[o][.] instruction, <anything> ÷ 0. Then, the contents of RT are 0x80000000 and if Rc = 1, the contents of bits in CR field 0 are LT = 1, GT = 0, EQ = 0, and SO is set to the correct value. In cmpi, cmp, cmpli, and cmpl instructions, the L-bit is applicable for 64-bit implementations. In 32-bit implementations, if L = 1 the instruction form is invalid. The core ignores this bit and therefore, the behavior when L = 1 is identical to the valid form instruction with L = 0

3.13.9 Floating-Point Processor

3.13.9.1 General

The MPC555 / MPC556 implements all floating-point features as defined in the **UISA**, including the non-IEEE working mode. Some features require software assistance. For more information refer to RCPU Reference Manual (Floating-point Load Instructions) for more information.

3.13.9.2 Optional instructions

The only optional instruction implemented by MPC555 / MPC556 hardware is store floating point as integer word indexed (**stfiwx**). An attempt to execute any other optional instruction causes the implementation dependent software emulation interrupt to be taken.

3.13.10 Load/Store Processor

The load/store processor supports all of the 32-bit implementation fixed-point PowerPC load/store instructions in the hardware.

3.13.10.1 Fixed-Point Load With Update and Store With Update Instructions

For load with update and store with update instructions, where RA = 0, the EA is written into R0. For load with update instructions, where RA = RT, RA is boundedly undefined.

3.13.10.2 Fixed-Point Load and Store Multiple Instructions

For these types of instructions, EA must be a multiple of four. If it is not, the system alignment error handler is invoked. For a **lmw** instruction (if RA is in the range of registers to be loaded), the instruction completes normally. RA is then loaded from the memory location as follows:

$$RA \leftarrow MEM(EA+(RA-RT)*4, 4)$$

3.13.10.3 Fixed-Point Load String Instructions

Load string instructions behave the same as load multiple instructions, with respect to invalid format in which RA is in the range of registers to be loaded. In case RA is in the range, it is updated from memory.

3.13.10.4 Storage Synchronization Instructions

For these type of instructions, EA must be a multiple of four. If it is not, the system alignment error handler is invoked.

3.13.10.5 Floating-Point Load and Store With Update Instructions

For Load and Store with update instructions, if RT = 0 then the EA is written into R0.

3.13.10.6 Floating-Point Load Single Instructions

In case the operand falls in the range of a single denormalized number the floating-point assist interrupt handler is invoked.

Refer to RCPU Reference Manual (Floating-Point Assist for Denormalized Operands) for complete description of handling denormalized floating-point numbers.

3.13.10.7 Floating-Point Store Single Instructions

In case the operand falls in the range of a single denormalized number, the floating-point assist interrupt handler is invoked.

In case the operand is ZERO it is converted to the correct signed ZERO in single-precision format.

In case the operand is between the range of single denormalized and double denormalized it is considered a programming error. The hardware will handle this case as if the operand was single denormalized.

In case the operand falls in the range of double denormalized numbers it is considered a programming error. The hardware will handle this case as if the operand was ZERO.

The following check is done on the stored operand in order to determine whether it is a denormalized single-precision operand and invoke the floating-point assist interrupt handler handler:

$$(FRS_{1:11} \neq 0) \text{ AND } (FRS_{1:11} \leq 896)$$

Refer to RCPU Reference Manual (Floating-Point Assist for Denormalized Operands) for complete description of handling denormalized floating-point numbers.

3.13.10.8 Optional Instructions

No optional instructions are supported.

3.13.10.9 Little-Endian Byte Ordering

The load/store unit supports little-endian byte ordering as specified in the **UISA**. In little-endian mode, if an attempt is made to execute an individual scalar unaligned transfer, as well as a multiple or string instruction, an alignment interrupt is taken.

3.14 PowerPC Virtual Environment Architecture (VEA)

3.14.1 Atomic Update Primitives

Both the **lwarx** and **stwcx** instructions are implemented according to the PowerPC architecture requirements. The MPC555 / MPC556 does not provide support for snooping an external bus activity outside the chip. The provision is made to cancel the reservation inside the MPC555 / MPC556 by using the CR_B and KR_B input pins.

3.14.2 Effect of Operand Placement on Performance

The load/store unit hardware supports all of the PowerPC load/store instructions. An optimal performance is obtained for naturally aligned operands. These accesses result in optimal performance (one bus cycle) for up to 4 bytes size and good performance (two bus cycles) for double precision floating-point operands. Unaligned operands are supported in hardware and are broken into a series of aligned transfers. The effect of operand placement on performance is as stated in the **VEA***,* except for the case of 8-byte operands. In that case, since the MPC555 / MPC556 uses a 32-bit wide data bus, the performance is good rather than optimal***.***

3.14.3 Storage Control Instructions

The MPC555 / MPC556 does not implement cache control instructions (**icbi**, **isync**, **dcbt**, **dcbi**, **dcbf**, **dcbz**, **dcbst**, and **dcbtst**) .

3.14.4 Instruction Synchronize (isync) Instruction

The **isync** instruction causes a reflect which waits for all prior instructions to complete and then executes the next sequential instruction. Any instruction after an **isync** will see all effects of prior instructions.

3.14.4.1 Enforce In-Order Execution of I/O (eieio) Instruction

When executing an **eieio** instruction, the load/store unit will wait until all previous accesses have terminated before issuing cycles associated with load/store instructions following the **eieio** instruction.

3.14.5 Timebase

A description of the timebase register may be found in **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** and in **SECTION 8 CLOCKS AND POWER CONTROL**.

3.15 POWERPC Operating Environment Architecture (OEA)

The MPC555 / MPC556 has an internal memory space that includes memory-mapped control registers and internal memory used by various modules on the chip. This memory is part of the main memory as seen by the MPC555 / MPC556 but cannot be accessed by any external system master.

3.15.1 Branch Processor Registers

3.15.1.1 Machine State Register (MSR)

The floating-point exception mode encoding in the MPC555 / MPC556 core is as follows:

Table 3-23 Floating-Point Exception Mode Encoding

Mode	FE0	FE1
Ignore exceptions	0	0
Precise	0	1
Precise	1	0
Precise	1	1

The SF bit is reserved set to zero

The IP bit initial state after reset is set as programmed by the reset configuration as specified by the USIU specification.

3.15.1.2 Branch Processors Instructions

The MPC555 / MPC556 implements all the instructions defined for the branch processor in the **UISA** in the hardware.

3.15.2 Fixed-Point Processor

3.15.2.1 Special Purpose Registers

- **Unsupported Registers —** The following registers are not supported by the MPC555 / MPC556: SDR, EAR, IBAT0U, IBAT0L, IBAT1U, IBAT1L, IBAT2U, IBAT2L, IBAT3U, IBAT3L, DBAT0U, DBAT0L, DBAT1U, DBAT1L, DBAT2L,

DBAT3U, DBAT3L

- **Added Registers —** For a list of added special purpose registers, refer to **Table 3-2**, and **Table 3-3**.

3.15.3 Storage Control Instructions

Storage Control Instructions **mtsr, mtsrin, mfsr, mfsrin, dcbi, tlbie, tlbia,** and **tlbsync** are not implemented by the MPC555 / MPC556.

3.15.4 Interrupts

The core implements all storage-associated interrupts as precise interrupts. This means that a load/store instruction is not complete until all possible error indications have been sampled from the load/store bus. This also implies that a store, or a nonspeculative load instruction is not issued to the load/store bus until all previous instructions have completed. In case of a late error, a store cycle (or a nonspeculative load cycle) can be issued and then aborted.

In each interrupt handler, when registers SRR0 and SRR1 are saved, MSR_{RI} can be set to 1.

The following paragraphs define the types of OEA interrupts The exception table vector defines the offset value by interrupt type. Refer to **Table 3-21**.

3.15.4.1 System Reset Interrupt

A system reset interrupt occurs when the IRQ0 pin is asserted and the following registers are set.

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that the processor attempts to execute next if no interrupt conditions are present
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

3.15.4.2 Machine Check Interrupt

A machine check interrupt indication is received from the U-bus as a possible response either to the address or data phase. It is usually caused by one of the following conditions:

- The accessed address does not exist
- A data error is detected

As defined in the **OEA**, machine check interrupts are enabled when MSR_{ME} = 1. If MSR_{ME} = 0 and a machine check interrupt indication is received, the processor enters the checkstop state. The behavior of the MPC555 / MPC556 in checkstop state is dependent on the working mode as defined in **21.4.1.1 Debug Mode Enable vs. Debug Mode Disable**. When the processor is in debug mode enable, it enters the debug mode instead of the checkstop state. When in debug mode disable, instruction processing is suspended and cannot be restarted without resetting the core.

An indication is sent to the SIU which may generate an automatic reset in this condition. Refer to **SECTION 7 RESET** for more details. If the machine check interrupt is enabled, MSR_{ME} = 1, it is taken. If SRR1 Bit 30 = 1, the interrupt is recoverable and the following registers are set.

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1	Set to 1 for instruction fetch-related errors and 0 for load/ store-related errors
	2:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

For load/store bus cases, these registers are also set:

Execution resumes at offset 0x00200 from the base address indicated by MSR_{IP}.

3.15.4.3 Data Storage Interrupt

A data storage interrupt is never generated by the hardware. The software may branch to this location as a result of implementation-specific data storage protection error interrupt.

Register Name	Bits	Description
Data/Storage Interrupt Status Register (DSISR)	0:14	Set to 0
	15:16	Set to bits 29:30 of the instruction if X-form and to 0b00 if D-form
	17	Set to Bit 25 of the instruction if X-form and to Bit 5 if D-form
	18:21	Set to bits 21:24 of the instruction if X-form and to bits 1:4 if D-form
	22:31	Set to bits 6:15 of the instruction
Data Address Register (DAR)		Set to the effective address of the data access that caused the interrupt

3.15.4.4 Instruction Storage Interrupt

An instruction storage interrupt is never generated by the hardware. The software may branch to this location as a result of an implementation-specific instruction storage protection error interrupt.

3.15.4.5 Alignment Interrupt

An alignment exception occurs as a result of one of the following conditions:

- The operand of a floating-point load or store is not word aligned.
- The operand of load/store multiple is not word aligned.
- The operand of **lwarx** or **stwcx** is not word aligned.
- The operand of load/store individual scalar instruction is not naturally aligned when $MSR_{LE} = 1$.
- An attempt to execute multiple/string instruction is made when $MSR_{LE} = 1$.

3.15.4.6 Floating-Point Enabled Exception Type Program Interrupt

A floating-point enabled exception type program interrupt is generated if $((MSR_{FE0} | MSR_{FE1}) \& FPSCR_{FEX})$ is set as a result of move to FPSCR instruction, move to MSR instruction or the execution of the **rfi** instruction. A floating-point enabled exception type program interrupt is not generated by floating-point arithmetic instructions. Instead if $((MSR_{FE0} | MSR_{FE1}) \& FPSCR_{FEX})$ is set, the floating-point assist interrupt is generated.

3.15.4.7 Illegal Instruction Type Program Interrupt

An illegal instruction type program interrupt is not generated by the MPC555 / MPC556. An implementation dependent software emulation interrupt is generated instead.

3.15.4.8 Privileged Instruction Type Program interrupt

A privileged instruction type program interrupt is generated for an on-core valid SPR field or any SPR encoded as an external to the core special register if $SPR_0 = 1$ and $MSR_{PR} = 1$, as well as an attempt to execute privileged instruction when $MSR_{PR} = 1$.

3.15.4.9 Floating-Point Unavailable Interrupt

The floating-point unavailable interrupt is generated by the MPC555 / MPC556 core as defined in the **OEA**.

3.15.4.10 Trace Interrupt

A trace interrupt occurs if MSR_{SE} = 1 and any instruction except **rfi** is successfully completed or MSR_{BE} = 1 and a branch is completed. Notice that the trace interrupt does not occur after an instruction that caused an interrupt (for instance, **sc**). A monitor/debugger software must change the vectors of other possible interrupt addresses to single-step such instructions. If this is unacceptable, other debug features can be used. Refer to **SECTION 21 DEVELOPMENT SUPPORT** for more information. The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction following the executed instruction
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x00D00 from the base address indicated by MSR_{IP}.

3.15.4.11 Floating-Point Assist Interrupt

A floating-point assist interrupt occurs in the following cases:

- When a floating-point exception condition is detected, the corresponding floating-point enable bit in the FPSCR (floating-point status and control register) is set (exception enabled) and ((MSR_{FE0} | MSR_{FE1}) = 1). Note that when ((MSR_{FE0} | MSR_{FE1}) and $FPSCR_{FEX}$) is set as a result of move to FPSCR, move to MSR or **rfi**, the floating-point assist interrupt handler is not invoked.
- When an intermediate result is detected and the floating-point underflow exception is disabled ($FPSCR_{UE}$ = 0)
- In some cases when at least one of the source operands is denormalized.

The following registers are set:

Execution resumes at offset 0x00E00 from the base address indicated by MSR_{IP}.

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR[1]
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

NOTES:
1. In the current implementation bit 30 of the SRR1 is never cleared other then by loading zero value from MSR RI.

3.15.4.12 Implementation-Dependent Software Emulation Interrupt

An implementation-dependent software emulation interrupt occurs in the following instances:

- When executing any non-implemented instruction. This includes all illegal and unimplemented optional instructions and all floating-point instructions.
- When executing a **mtspr** or **mfspr** that specifies on-core non-implemented register, regardless of SPR_0.
- When executing a **mtspr** or **mfspr** that specifies off-core non-implemented register and $SPR_0 = 0$ or $MSR_{PR} = 0$ (no program interrupt condition).
- Program interrupt is generated if $((MSR_{FE0} \mid MSR_{FE1})$ and $FPSCR_{FEX})$ is set as a result of move to FPSCR instruction, move to MSR instruction, or the execution of the **rfi** instruction.
- Floating-point enabled exception type program interrupt is not generated by floating-point arithmetic instructions, instead if $((MSR_{FE0} \mid MSR_{FE1})$ $\&FPSCR_{FEX})$ is set, the floating-point assist interrupt is generated.

In addition, the following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x01000 from the base address indicated by MSR_{IP}.

3.15.4.13 Implementation-Specific Instruction Storage Protection Error Interrupt

The implementation-specific instruction storage protection error interrupt occurs in the following cases:

- The fetch access violates storage protection.
- The fetch access is to guarded storage and $MSR_{IR} = 1$.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1	Set to 0
	2	Set to 0
	3	Set to 1 if the fetch access was to a guarded storage when MSR_{IR} = 1, otherwise set to 0
	4	Set to 1 if the storage access is not permitted by the protection mechanism; otherwise set to 0
	10	Set to 0
	11:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x01300 from the base address indicated by MSR_{IP}.

3.15.4.14 Implementation-Specific Data Storage Protection Error Interrupt

The implementation-specific data storage protection error interrupt occurs in the following case:

- The access violates the storage protection.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0
Data/Storage Interrupt Status Register (DSISR)	0	Set to 0
	1	Set to 0
	2:3	Set to 0
	4	Set to 1 if the storage access is not permitted by the protection mechanism. Otherwise set to 0
	5	Set to 0
	6	Set to 1 for a store operation and to 0 for a load operation
	7:31	Set to 0
Data Address Register (DAR)		Set to the effective address of the data access that caused the interrupt

Execution resumes at offset 0x01400 from the base address indicated by MSR_{IP}.

3.15.4.15 Implementation-Specific Debug Interrupts

Implementation-specific debug interrupts occur in the following cases:

- When there is an internal breakpoint match (for more details, refer to **SECTION 21 DEVELOPMENT SUPPORT**.
- When a peripheral breakpoint request is asserted to the MPC555 / MPC556 core.
- When the development port request is asserted to the MPC555 / MPC556 core. Refer to **SECTION 21 DEVELOPMENT SUPPORT** for details on how to generate the development port-interrupt request.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		For I-breakpoints, set to the effective address of the instruction that caused the interrupt. For L-breakpoint, set to the effective address of the instruction following the instruction that caused the interrupt. For development port maskable request or a peripheral breakpoint, set to the effective address of the instruction that the processor would have executed next if no interrupt conditions were present. If the development port request is asserted at reset, the value of SRR0 is undefined.
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}. If the development port request is asserted at reset, the value of SRR1 is undefined.
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

For L-bus breakpoint instances, these registers are set to:

Register Name	Bits	Description
BAR		Set to the effective address of the data access as computed by the instruction that caused the interrupt
DAR and DSISR		Do not change

Execution resumes at offset from the base address indicated by MSR_{IP} as follows:

- 0x01D00 – For instruction breakpoint match
- 0x01C00 – For data breakpoint match
- 0x01E00 – For development port maskable request or a peripheral breakpoint
- 0x01F00 – For development port non-maskable request

3.15.4.16 Partially Executed Instructions

In general, the architecture permits instructions to be partially executed when an alignment or data storage interrupt occurs. In the core, instructions are not executed at all if an alignment interrupt condition is detected and data storage interrupt is never generated by the hardware. In the MPC555 / MPC556, the instruction can be partially executed only in the case of the load/store instructions that cause multiple access to the memory subsystem. These instructions are:

- Multiple/string instructions
- Unaligned load/store instructions

In the last case, the store instruction can be partially completed if one of the accesses (except the first one) causes the data storage protection error. The implementation-specific data storage protection interrupt is taken in this case. For the update forms, the update register (RA) is not altered.

3.15.5 Timer Facilities

Descriptions of the timebase and decrementer registers can be found in **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** and in **SECTION 8 CLOCKS AND POWER CONTROL**.

3.15.6 Optional Facilities and Instructions

Any other **OEA** optional facilities and instructions (except those that are discussed here) are not implemented by the MPC555 / MPC556 hardware. Attempting to execute any of these instructions causes an implementation dependent software emulation interrupt to be taken.

SECTION 4
BURST BUFFER

The burst buffer module consists of the burst buffer controller (BBC) and the instruction memory protection unit (IMPU).

The BBC delivers the RCPU instruction fetch accesses from the instruction bus onto the U-bus. It utilizes the full U-bus pipeline and a special page access attribute in order to take full advantage of the U-bus bandwidth. It can handle both burstable and non-burstable external memories as well as non-burstable internal memories (flash EEPROM, SRAM).

Code compression features are only available on the MPC556. The MPC556 utilizes a version of code compression / decompression which is called “Phase A”. Phase A code compression / decompression is described in this manual. Future parts may have a different type of code compression. The BBC also contains the functional module which is called the instruction code decompressor unit (ICDU). The ICDU is responsible for on-line (previously compressed) instruction code decompression in the “Decompression-ON” mode. In the “Decompression-OFF” mode, the ICDU is bypassed and the BBC is in normal function.

The IMPU allows the memory to be divided into four regions with different attributes, as well as a default global region (for memory space that is not included in either of the two regions). Each of the two regions can be of size four Kbytes to four Gbytes. Overlap between regions is allowed.

The IMPU includes registers that contain the following information: region base address, region size and the region’s access permissions. For each access (from the processor to the memory), the IMPU finds which region matches the address. If more than one region matches, the region with the lowest index is chosen. If no region is matched, the global region is chosen.

The IMPU compares the attributes of the access from the processor to the attributes of the appropriate region. If the access is allowed, the proper signals are sent to the BBC. If the access is not permitted, an interrupt is sent to the processor.

The IMPU does not support address translation. The effective fetch address issued by the processor is the one that is transferred to the U-bus.

4.1 Burst Buffer Block Diagram

Figure 4-1 is a block diagram of the burst buffer.

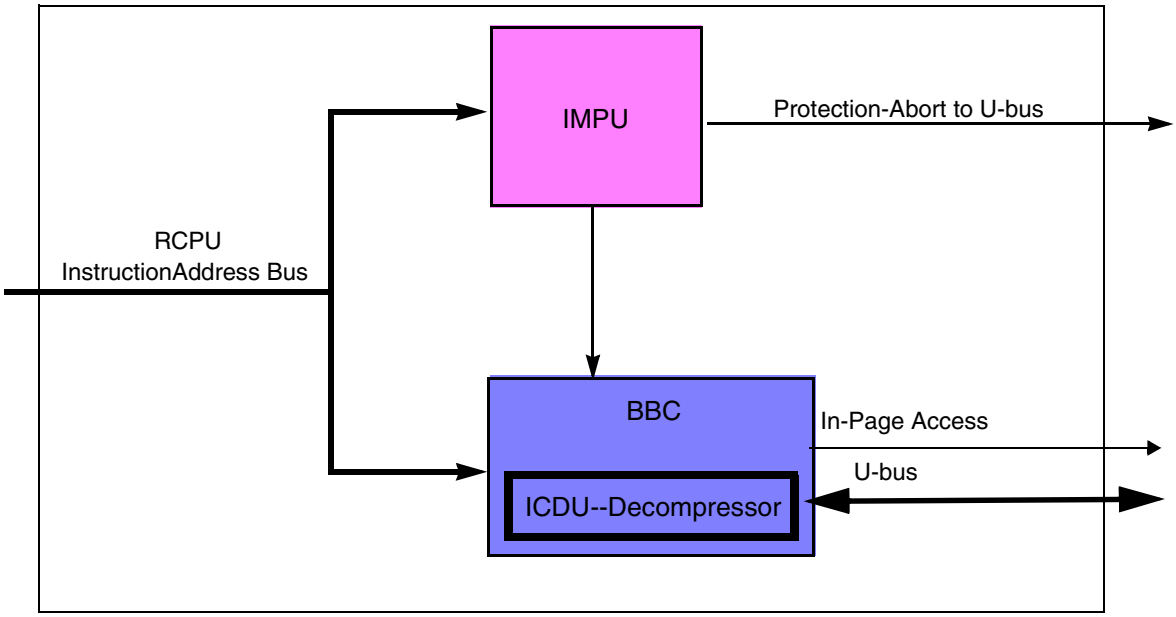


Figure 4-1 Burst Buffer Block Diagram

4.2 Burst Buffer Features

The BBC offers the following features:

- Supports pipelined access to internal memory and burstable access to the external memory.
- Supports the de-coupled interface with the RCPU instruction unit.
- Serves as parked master on the U-bus, resulting in zero clocks delay for RCPU fetch access to cross to the U-bus.
- Full utilization of the U-bus pipeline for fetch accesses.
- Tightly interfaced with L2U interface module, taking advantage of full U-bus bandwidth and back-to-back accesses.
- Supports program trace and show cycle attributes.
- Supports special attribute for debug port fetch accesses.
- Is programmed using the MPC555 / MPC556 **mtspr/mfspr** instructions to/from implementation specific special-purpose registers.
- Designed for minimum power consumption.

The ICDU offers the following features:

- Instruction code on-line decompression based on a fixed vocabulary (bounded Huffman) algorithm.
- No need for address translation between compressed and non-compressed address spaces — ICDU provides “next instruction address” to the RCPU
- Instruction decompression takes one clock cycle.
- Code decompression is pipelined.
- No performance penalty during sequential program flow execution

— Minimal performance penalty due to change of program flow execution
- Two operation modes are available: “Decompression ON” and “Decompression OFF”. Switch between compressed and non-compressed user application software parts is possible.

The IMPU has the following features:

- Four regions in which the base address and size can be programmed.
- Region sizes of four Kbytes up to four Gbytes (in powers of two) can be programmed. (A region must start on the specified region size boundary.)
- Overlap between regions is allowed.
- Each of the four regions supports the following attributes:
  — Access protection (use r/ supervisor fetch or no access).
  — Guarded attribute (causes an interrupt in case of fetch try).
  — On / off option
  — Compressed / non-compressed.
- Global region entry declares the default access protection and guarded attributes for all memory areas not covered by the four regions:
- Interrupt generated upon access violation or fetch from guarded region.
- MPC555 / MPC556 MSR[IR] bit controls MPU protection.
- Programming is done using MPC555 / MPC556 **mtspr/mfspr** instructions to/from implementation specific special purpose registers.
- Designed for minimum power consumption.
- Compressed/non-compressed region with enable/disable option.
- Special reset exception vector for “Decompression ON” mode.

4.3 Instruction VocabularyBased Compression Model Main Principles

4.3.1 Compression Model Features

- Implemented for PowerPC architecture
- Up to 30% code size reduction
- No need for address translation tables
- No changes in the CPU architecture
- Compression is done off line by a special “compressor” tool, using a fixed vocabulary instruction based algorithm, optimized for the PowerPC instruction set.
- Decompression is done at run-time with special hardware.
- Optimized for cache-less systems:
  — Highly effective in system solutions for low cache-hit ratio environment and for systems with fast embedded program memory
  — Deterministic program execution time
  — No performance penalty during sequential program flow execution
  — Minimal performance penalty due to change of program flow execution
- Switch between compressed and non-compressed user application sections is possible. (Compressed subroutine can call non-compressed one and be called from non-compressed portion of user application)

- Slight changes in the core and existing RISC development tools — compilers, simulators, manually coded libraries.
- Compressed address space is up to four Megabytes (4 Mbytes).
- Branch displacement from its target:

  — Conditional branch displacement is up to two Kbytes (2 Kbytes).
  — Unconditional branch displacement is up to two Mbytes (2 Mbytes).

NOTE

Branch displacement is hardware limited. The compiler can enlarge the branch scope by creating branch chains.

4.3.2 Model Limitations

No address arithmetic is allowed, because the address map changes during compression and no software tool can identify address arithmetic structures in the code.

4.3.3 Vocabulary Based Instruction Compression Algorithm

The code compression algorithm is based on creating vocabularies of frequently appearing PowerPC RISC instructions or instruction halves and replacing these instructions with pointers to the vocabularies.

Compressed and bypass field lengths may vary. An example of compressed code is shown in **Figure 4-2**.

Compression of the instructions in a vocabulary may be in one of the following modes.

1. Compression of the whole instruction into four vocabulary byte pointers. The four compacted bytes may start on any bit location. Four of the decoded bits and another bit for starting from the left or right side of the address location determine the bit location for the byte start
2. Compression of a combination of the instruction's bytes into vocabulary pointers and bypass of the other byte(s). Bypass is the placing of the field's uncompressed instruction information into the compressed code.
3. Bypass of the whole instruction. No compaction permitted.

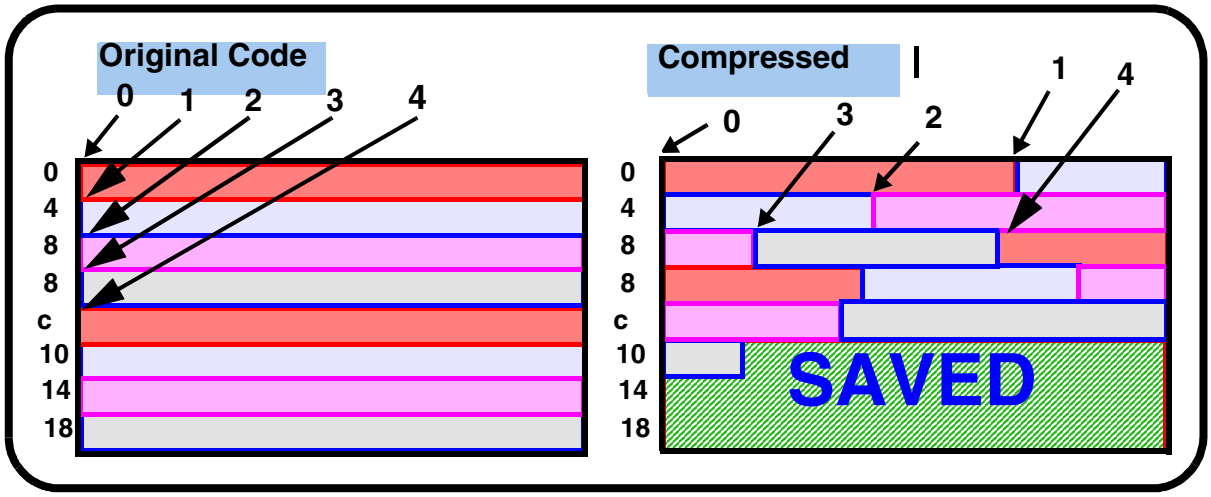


Figure 4-2 Example of Compressed Code

Each instruction is divided to four bytes, marked X1, X2, X3 and X4. For each such byte a separate (Huffman coding) vocabulary is generated, marked Tx1, Tx2, Tx3 and Tx4. Once compressed, each instruction yields four symbols (corresponding to the X1, X2, X3, and X4 input bytes). Therefore, in order to compress a given code, four vocabularies are required. This partitionong produced a better comression ratio.

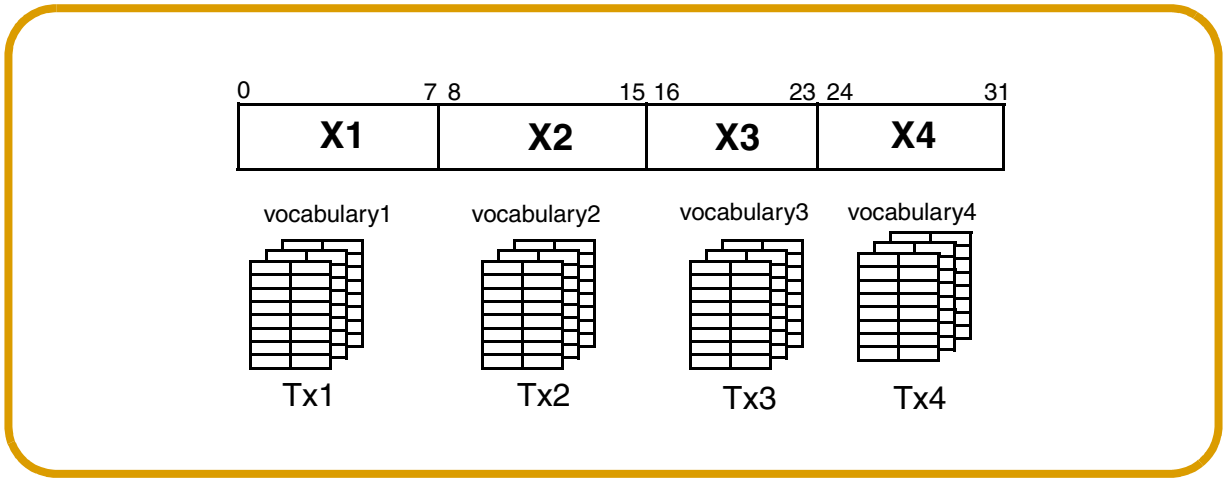


Figure 4-3 Instruction Coding

4.3.4 Memory Organization

In order to enhance performance, the logic is built to decode two halves of an instruction in parallel. The memory is arranged to support this as two streams of compressed symbols: the left stream for the compressed symbols of X1 and X2 bytes, and the right stream for the compressed symbols of X3 and X4 bytes.

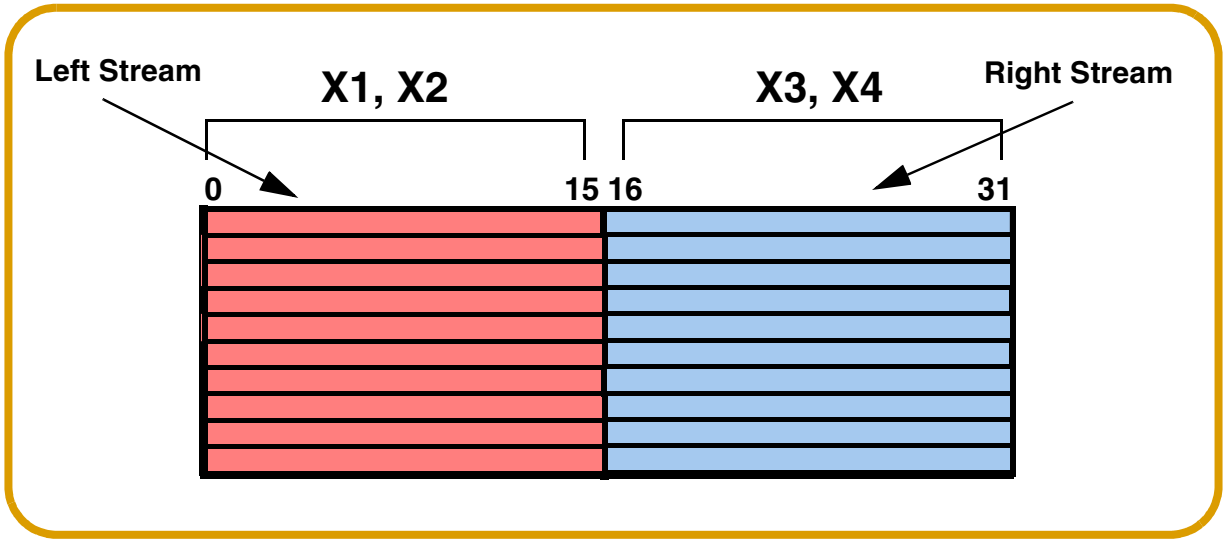


Figure 4-4 Two Streams Memory Organization — Before Compression

In **Figure 4-4**, each left and right stream line includes two original bytes of the instruction. **Figure 4-5**, shows the memory after compressed streams have been put into it.

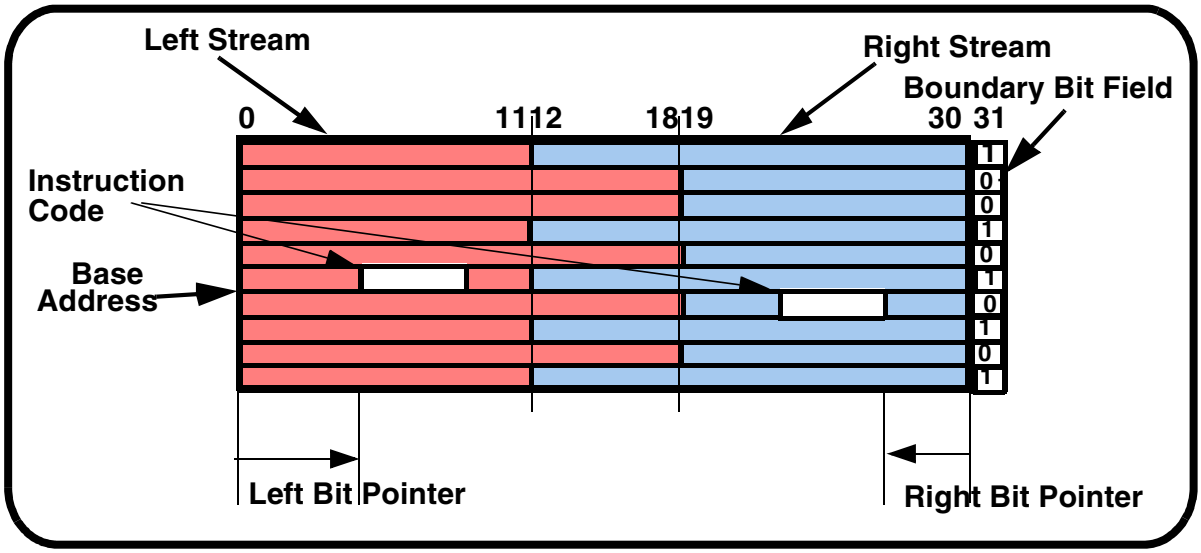


Figure 4-5 Two Streams Memory Organization — After Compression

The compiler will set the left and right stream boundary at either bit 12 or bit 19. This will be determined by the most efficient placement of compressed instruction code. The boundary will be placed between bits 11 and 12 if bit 31 is equal to one. The boundary will be placed between bits 18 and 19 if bit 31 is equal to zero. The original right and left streams may span an adjacent base addresses before or after each other. This fact will also determine the placement of the boundary bit field.

Each stream line may include a variable number of compressed symbols, depending on how well the bytes in the original stream were compressed.

The decompressor has to maintain two bit pointers (left and right) in order to have access to the start location of any instruction's half.

The decompressor maintains tracking of the base address, to start fetching from the next address in the memory.

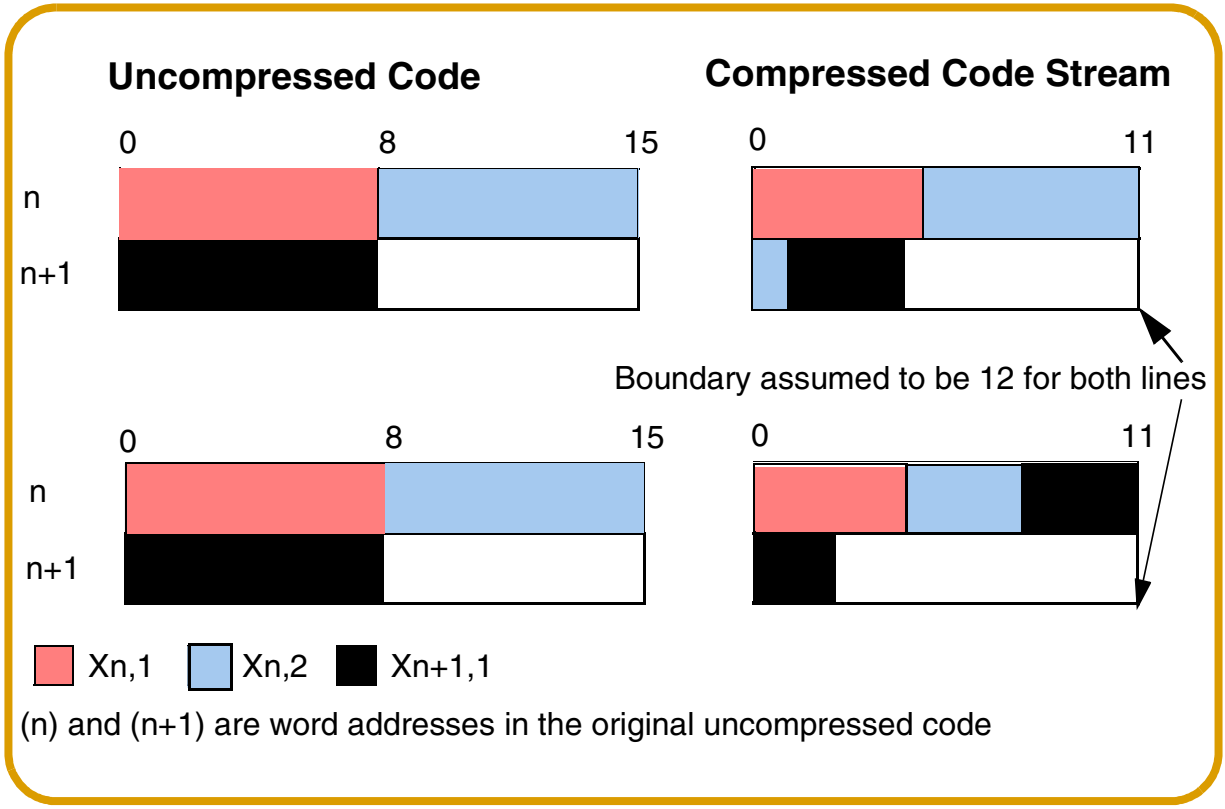


Figure 4-6 Examples of Compressed Symbols Layout

4.3.5 Compressed Code Address Format

The format of the compressed code in memory requires special addressing. The Decompressor module is responsible for generating compressed code addresses. The compressed instruction stream may start on any of the 32 bits. Thus, five bits are needed to locate such instruction inside a memory word. The instruction address in "Decompression ON" mode consists of a 20-bit word pointer for the base address, bits 20 and 21 to show the relation between the left and right streams, and two 5-bit instruction pointers. This is known as the two-pointer address form. See **Figure 4-7**.

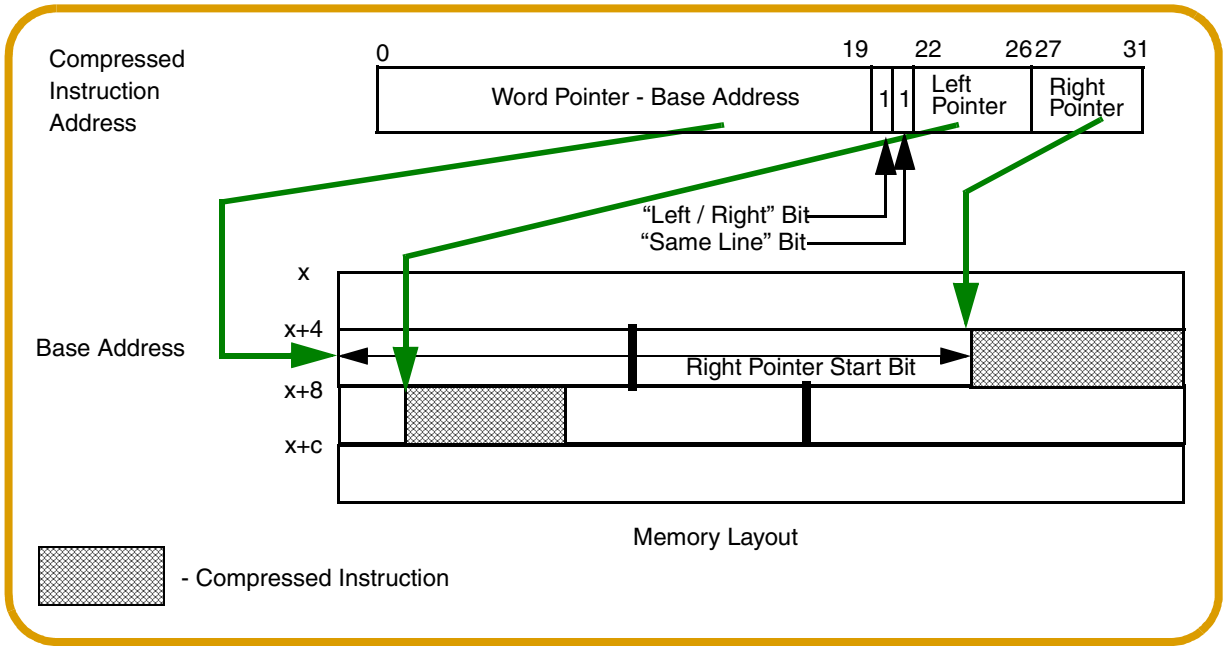


Figure 4-7 Compressed Address Format

The base address contains the lowest word address of physical memory where the instruction resides.

The "left / right" bit, bit number 20, indicates which instruction stream side (left or right) resides in the memory word location being pointed to by the base address. A zero "0" for bit 20 will indicate that the left side is resident in the base address location. A one "1" for bit 20 will indicate that the right side is resident in the base address location. The instruction stream side not pointed to will reside in the following address location.

The "same line" bit, bit 21, reflects the relative location of the two side streams for the instruction. If bit 21 is zero "0", both left and right streams are located at the base address location. In this case, bit 20 has no meaning and is a "don't care" value of X. If bit 21 is one "1", then the two parts of the instruction are located in different address word locations (one at "x" base address, the other at "x+4").

Figure 4-8 illustrates the three possible cases for bits 20 and 21.

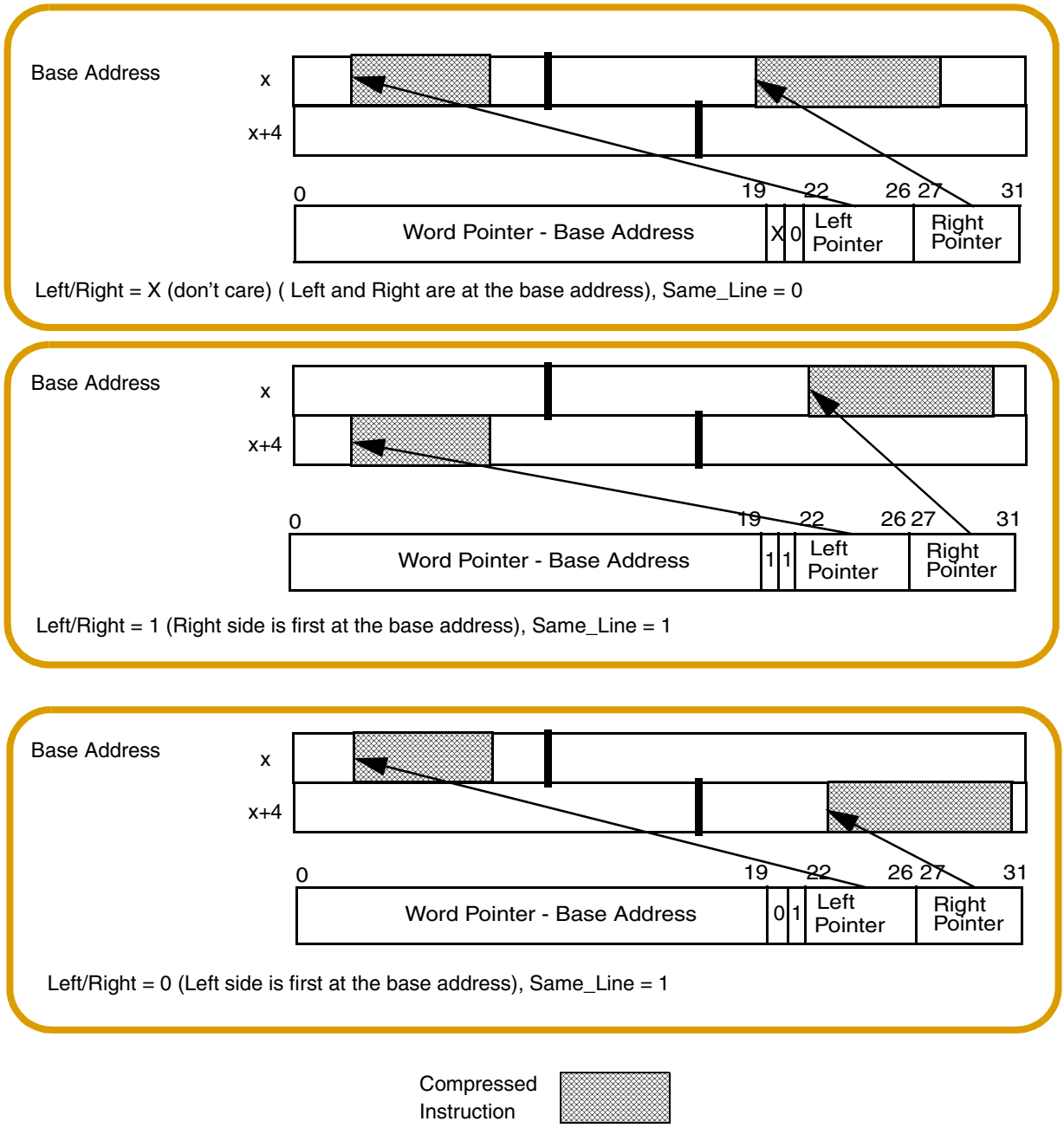


Figure 4-8 Examples of Instruction Layout in Memory

4.3.6 Compressed Address Format – Direct Branches

The one pointer format is used for the conditional and unconditional direct branches. **Figure 4-9** illustrates the one pointer format. The word pointer for the unconditional branch has nineteen bits (the lower two-byte bits are ignored). This will yield an unconditional branch displacement limit of two Mbytes. The word pointer for the conditional

branch has nine bits. This will yield a conditional branch displacement limit of two Kbytes.

When a change of flow occurs, the sequencer of the PPC core will issue the new address in compression mapped format. The address extractor unit of the BBC generates the direct branch address format to internal memory.

Indirect branches use the regular two pointer format described in the previous section.

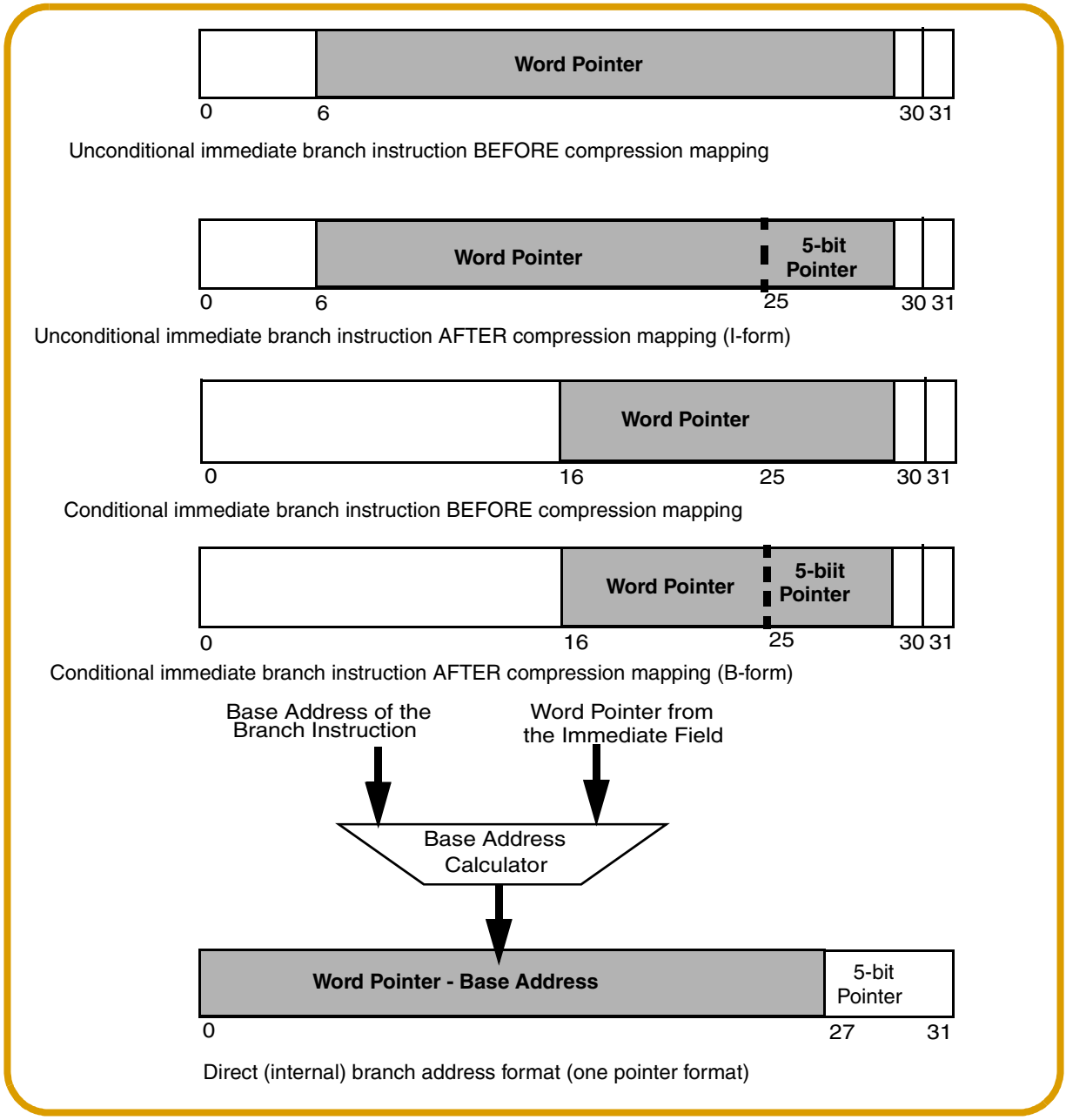


Figure 4-9 Generating Compressed Code Address for PowerPC Direct Branches

An instruction in memory which will serve as the target of a branch will have a label attached. The label provides the needed pointer to the other half of the branch target instruction. The label token will be skipped in normal sequential operation. The label has three parts. First, the label prefix character (which is skipped by the decompressor). Second, a 5-bit pointer to the second half of the instruction. Third, a bit which indicates the location of the second instruction half on the same line or the next line.

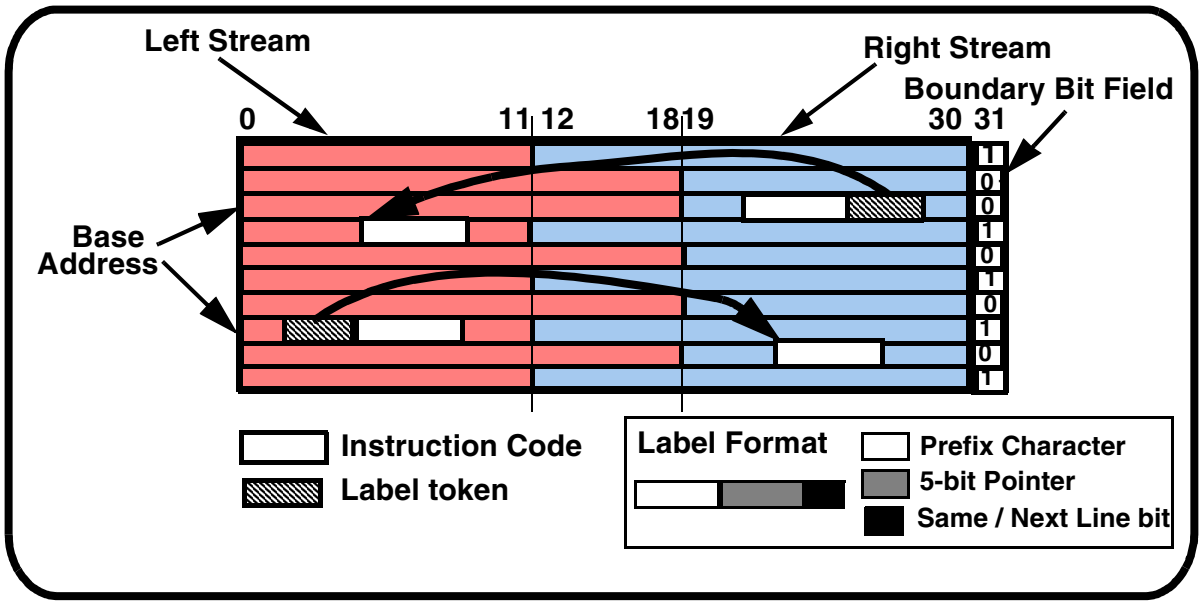


Figure 4-10 Extracting Direct Branch Target Address in the Decompressor

4.3.7 Compressed Address Format – Indirect Branches

Indirect branches use the regular two pointer format described above. The indirect branch destination address is copied without any change from one of the following registers:

- LR
- CTR
- SRR0

See the PowerPC™ ***RCPU User's Manual*, RCPURM/AD**, for more details.

4.3.8 Compression Process

The compression process is implemented by the following steps (See **Figure 4-11**):

- User code compilation/linking
- User application code compression by software compression tool.

The compiler will add a few simple "hooks" to the compiled code which will make compression possible. Compiled code will be generated in the "elf" format for code

compression purposes. The resulting uncompressed elf code (with compression hooks) will load and run like any other elf code.

The software compression tool compresses the elf code (x.elf) and produces a compressed elf code (x.elf.sqz). The system sees the compressed elf code as regular elf formatted code for purposes of loading into hardware (when programming the flash). However, the decompression module must now be used to run the compressed elf code.

Figure 4-11 illustrates the several steps for generating compressed executable code. Once generated, this code can be loaded into flash or SRAM (internal or external) .

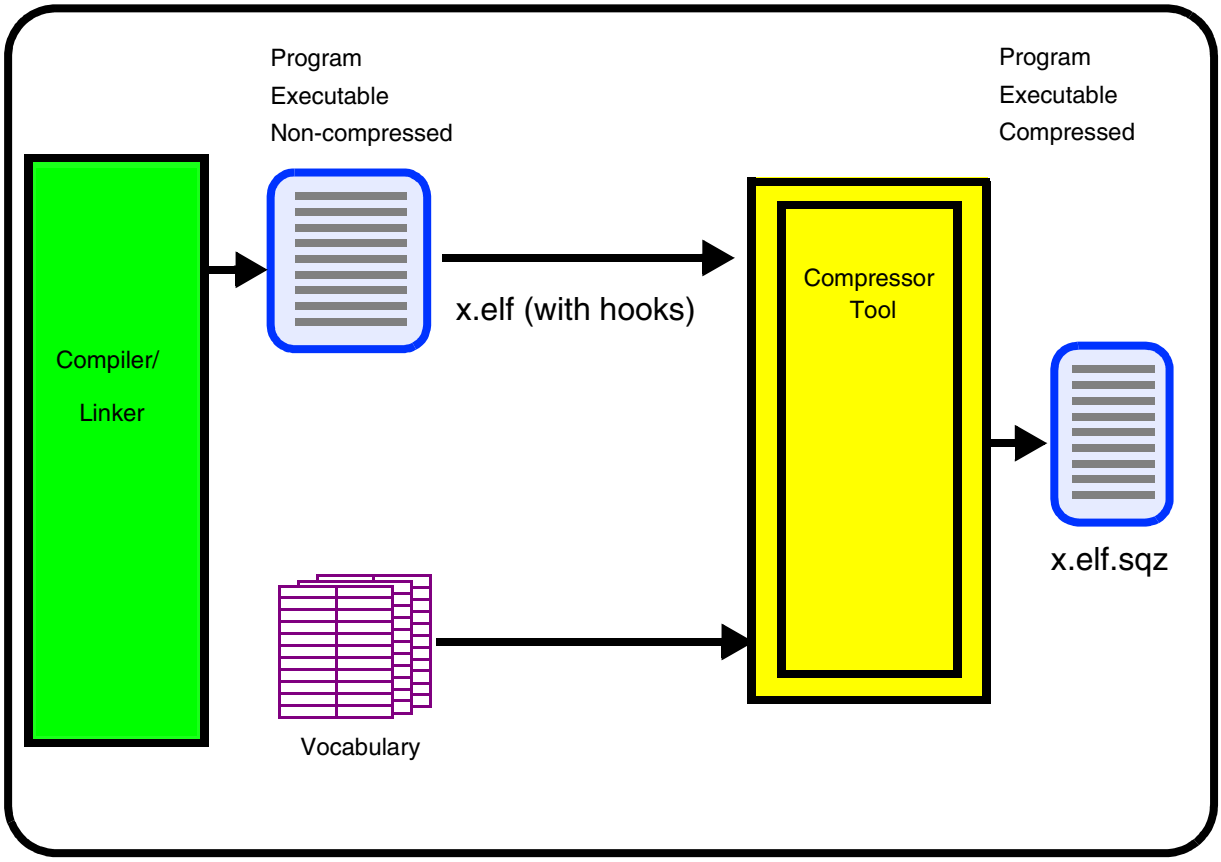


Figure 4-11 Code Compression Process (Phase A)

The compression tool replaces regular PowerPC instructions by their “compressed” representation which contain fewer data bits. The compressed data bit representation is contained in the vocabulary. The vocabulary is structured into a binary bounded Huffman code tree. This method has the result of the first instructions being represented by fewer bits. Further instructions require more bits for unique decoding. Therefore, the instructions that occur most in code should be represented earlier in the vocabulary structure. This would produce the most condensed code. A statistical study was made of typical application code. The existing vocabulary is fixed for Phase A

code compression, and is a result of the statistical study. **Figure 4-12** illustrates the binary decode tree for specific instructions.

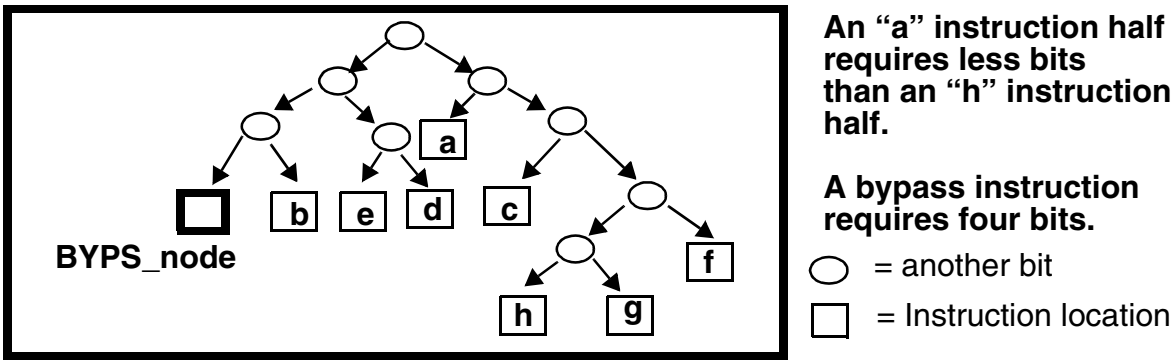


Figure 4-12 Bounded Huffman Code Tree

In **Figure 4-12**, instruction "a" would require two bits. The bypass node would require four bits. The bounded form of the Huffman code tree is limited in size for implementation into hardware. The largest compressed instruction is 36 bits — four bits for the bypass mode plus the normal uncompressed 32-bit instruction.

4.3.9 Decompression

- The instruction code is stored in the memory in the compressed form
- The decode vocabulary is stored in the burst buffer controller (BBC).
- The decompression is done on-line by the dedicated decompressor unit in the BBC.
- Decompression flow: (See **Figure 4-11**)
  - — RCPU provides a **"bit aligned COF[1] address"** to the BBC.
  - — ICDU:
    - Converts COF address to **"word aligned physical address"** to access the memory
    - Fetches the compressed instruction code data from the memory, decompresses it and delivers **"non-compressed instruction code"** together with the bit aligned **"next instruction address"** to the RCPU, that uses it for subroutine and exceptions handling.
    - When instructions are running without a COF, the next instruction is prefetched and decoded in the current cycle. This eliminates any delays from code compression during regular sequential (non-COF) operation.

[1] COF = Change of Flow

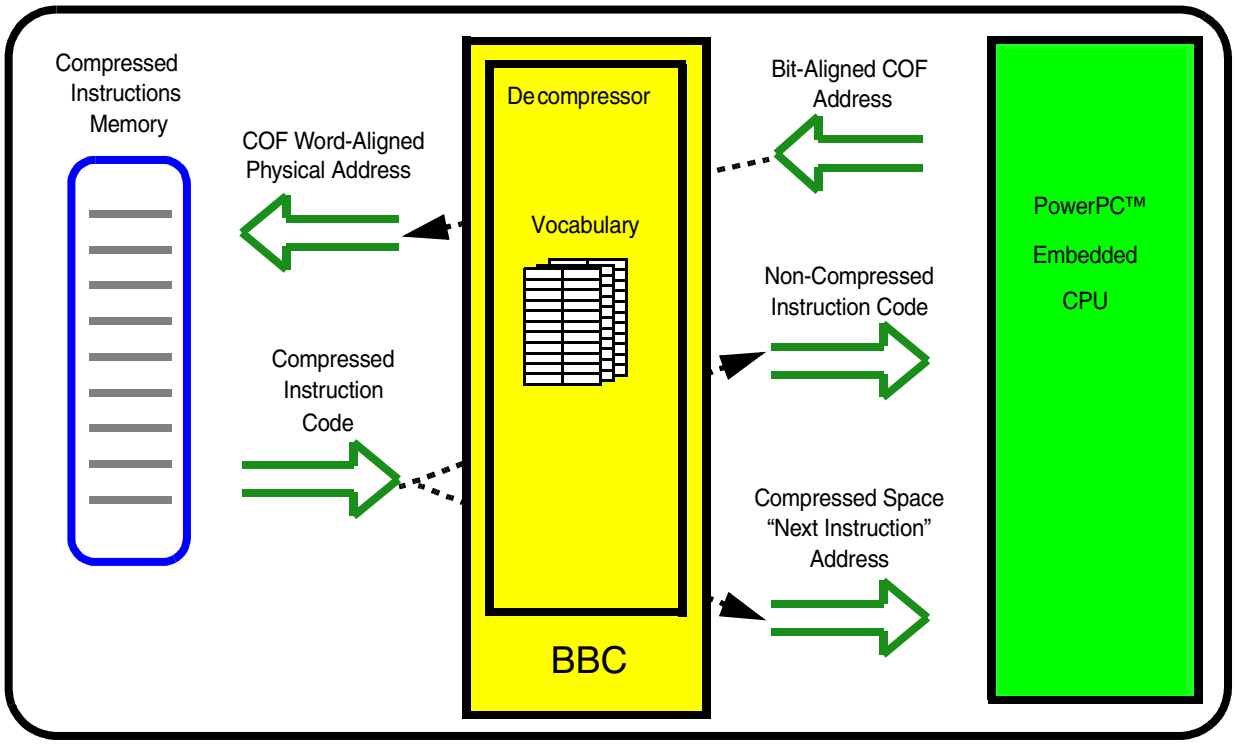


Figure 4-13 Code Decompression Process

4.3.10 Compression Environment Initialization

At power on reset (POR) or with a hard reset, the default settings will be activated unless the configuration word inputs override these defaults. The compression mode configuration data to be programmed is supplied by the user software in the flash.

The hard reset configuration word (described in **SECTION 7 RESET**) has two bits which control the code compression mode. Bit 21 enables code compression when equal to "1" and disables code compression when equal to "0". Bit 22 defines the exception table code as either compressed with a value of "1" or non-compressed with a value of "0".

4.4 Modes Of Operation

The burst buffer module can operate in the following modes:

- Normal
- Slave
- Reset
- Debug
- Standby
- Burst

The modes of operation are described in the following paragraphs.

4.4.1 Normal Operation

During normal operation, the burst buffer module transfers fetch accesses from the CPU to the U-bus. When a new access is issued by the CPU, it is transferred in parallel to both the IMPU and the BBC. The IMPU compares the address of the access to its region programming. The BBC determines whether the access can be immediately transferred to the U-bus. If not, it requests the U-bus for the next clock.

Each new BBC U-bus access is accompanying by the burst request attribute. If burstable access is enabled, the BBC performs a burst access; otherwise, it performs a single access.

If the IMPU detects an access violation, it does the following:

- Cancels the request that was forwarded to the BBC
- Informs the RCPU core that the requested address generated an exception

If the required address contains show cycle or program trace attributes, the BBC delivers the access to the U-bus even if the request is cancelled (due to the exception it caused).

The BBC forwards show cycle, program trace and debug port access attributes accompanying the CPU access along with the U-bus access.

4.4.2 Slave Operation.

The burst buffer module is operating as a U-bus slave module when the instruction memory protection unit (IMPU) registers are accessed by the user in order to be programmed. This programming is done using the **mtspr /mfspr** instructions.

4.4.3 Reset Operation

On reset the BBC goes to an idle state, and all pending U-bus accesses are ignored. The IMPU goes to a disabled state in which all memory space is accessible to both user and supervisor.

4.4.4 Debug Mode Operation

When the CPU is in debug mode, fetch accesses are attached with a special attribute. If this attribute is asserted, the BBC must initiate not-burstable accesses to the debug port.

4.4.5 Standby Mode Operation

In this low-power mode the CPU stops issuing further accesses. The BBC clocks are turned off, and the BBC enters a power-save state. When the low-power mode is exited, clocks are activated and a new access from the CPU will activate the BBC.

4.4.6 Burst Operation

The BBC can run burst accesses on the U-bus. Such burst cycles, if forwarded to external memory, are then exported to the EBI as burst cycles (if bursts are enabled by the USIU).

The BE bit defined in **4.6.4 BBC Module Configuration Register (BBCMCR)** determines whether the BBC operates burst cycles or not. Burst requests are enabled only when the BE bit is set.

NOTE

The negated state of the BE bit is useful mainly when the RCPU core runs in serialized mode. (Refer to **21.7.6 I-Bus Support Control Register** for the ICTRL register.)

4.4.7 Error Detection

If the IMPU detects access violation, the following actions must be taken:

1. Cancel the request that was forwarded to the burst buffer controller
2. Inform the RCPU core that the requested address generated an exception

If the required address contains show cycle or program trace attributes, than the BBC delivers the access onto the U-bus even if the request is cancelled (due to the exception it caused).

The way the IMPU notifies the RCPU core for an interrupt is by feeding error information into four bits (1, 3, 4 and 10) in the SRR1 register in the core. Only one bit is set at a time. The exception vector (address) that the core issues for this event is 0xnnn0-1300. The encoding of the status bits is as follows:

- SRR1 = 0
- SRR3 = Guarded storage.
- SRR4 = Protected storage.
- SRR10 = 0

4.5 Exception Table Relocation

The BBC has the ability to relocate the exception table. Exception table relocation is a feature to save memory space in the exception table. See **3.11.5 Exception Vector Table** for normal operation of the exception vector table. This is done by mapping exceptions to be separated by eight bytes instead of 256 bytes (see **Table 4-1**). The relocation feature maps the exception table into the internal memory space of the MPC555 / MPC556 and requries MSR[IP] = 1. This feature is important in multi-MPC555 / MPC556 systems, where more than one MCU can have internal exception tables with the same exception addresses issued by the RCPU.

The relocation feature also saves the wasted space between exception table entries when each exception entry contains only a branch instruction to the exception routine, which is located elsewhere.

If exception relocation is enabled (ETRE bit is set in the BBCMCR), all exception routines (except the reset exception routine) can be controlled to either remain in the lower addresses of the memory (base address + exception offset) BBCMCR[OERC] = 0 or to be relocated to memory (base address + 32 Kbytes) by setting BBCMCR[OERC] = 1. The reset exception routine location is fixed in memory (base address + the reset exception offset) and can not be relocated.

See **4.6.4 BBC Module Configuration Register (BBCMCR)** for programming details.

4.5.1 Exception Table Relocation Operation

When an exception is requested, the CPU initiates a fetch cycle that branches to the exception routine associated with the exception that caused the fetch. The exception addresses are fixed within the RCPU architecture and are 0x100 bytes apart from each other, starting at address 0x0000_0100 or 0xFFF0_0100, depending on the value of the MSR[IP] bit.

If the relocation feature is disabled, the BBC transfers the exception fetch address to the internal bus of the MPC555 / MPC556 with no interference.

In order to activate exception table relocation, the following steps are required:

1. Set the MSR[IP] bit. To set this bit out of reset, set the appropriate bit in the reset configuration word.
2. Set the ETRE bit in BBCMCR register. See **4.6.4 BBC Module Configuration Register (BBCMCR)** for programming details.
3. Program absolute branch instructions at the locations indicated in **Table 4-1** pointing to the desired exception handler routines.

If the relocation feature is enabled, the BBC translates the starting address of the exception routine into the address located at the lowest portion of the internal memory. At that location, the user must insert a series (table) of consecutive branch instructions that point to the appropriate exception routines.

NOTE

These branch instructions must utilize absolute addressing modes of the RCPU (relative branches can not be used).

Thus, the CPU branches twice to reach the appropriate exception routine.

NOTE 1

The eight Kbytes allocated for the exception table can be almost fully utilized. This is possible if the MPC555 / MPC556's address space is *not* mapped to the exception address space — that is, if addresses 0xFFF0_0000 to 0xFFF0_1FFF are not part of the MPC555 / MPC556 address space. In this case, these eight Kbytes can be fully utilized by the compiler, except for the lower 64 words (256 bytes), which are reserved for the exception pointers.

NOTE 2

If the CPU issues any address that falls between two successive exception entries (e.g., 0xFFF0_0104), then an exception is generated to the CPU if exception relocation is enabled. See **4.6.4 BBC Module Configuration Register (BBCMCR)**.

Table 4-1 Exception Addresses Mapping by BBC[1]

Name of Exception	Address Issued by CPU (MSR[IP] = 1)	Mapped Address by Exception Table Relocation Logic (BBCMCR[ETRE] = 1)	
		BBCMCR[OERC] = 0[2]	BBCMCR[OERC] = 1
Reserved	0xFFF0_0000	0x0000	0x8000
System Reset	0xFFF0_0100	0x0008	0x0008[3]
Machine Check	0xFFF0_0200	0x0010	0x8010
Data Storage	0xFFF0_0300	0x0018	0x8018
Instruction Storage	0xFFF0_0400	0x0020	0x8020
External Interrupt	0xFFF0_0500	0x0028	0x8028
Alignment	0xFFF0_0600	0x0030	0x8030
Program	0xFFF0_0700	0x0038	0x8038
Floating Point unavailable	0xFFF0_0800	0x0040	0x8040
Decrementer	0xFFF0_0900	0x0048	0x8048
Reserved	0xFFF0_0A00	0x0050	0x8050
Reserved	0xFFF0_0B00	0x0058	0x8058
System Call	0xFFF0_0C00	0x0060	0x8060
Trace	0xFFF0_0D00	0x0068	0x8068
Floating Point Assist	0xFFF0_0E00	0x0070	0x8070
Implementation Dependant Software Emulation	0xFFF0_1000	0x0080	0x8080
Implementation Dependant Storage Error	0xFFF0_1300	0x0098	0x8098
Implementation Dependant Data Breakpoint	0xFFF0_1C00	0x00E0	0x80E0
Implementation Dependant Instruction Breakpoint	0xFFF0_1D00	0x00E8	0x80E8
Implementation Dependant Maskable External Breakpoint	0xFFF0_1E00	0x00F0	0x80F0
Non-Maskable External Breakpoint	0xFFF0_1F00	0x00F8	0x80F8

NOTES:
1. See **Table 3-21** and **3.11.5 Exception Vector Table** for Exception Relocation Table with ETRE = 0.
2. See **4.6.4 BBC Module Configuration Register (BBCMCR)**
3. The reset exception is NOT affected by OERC.

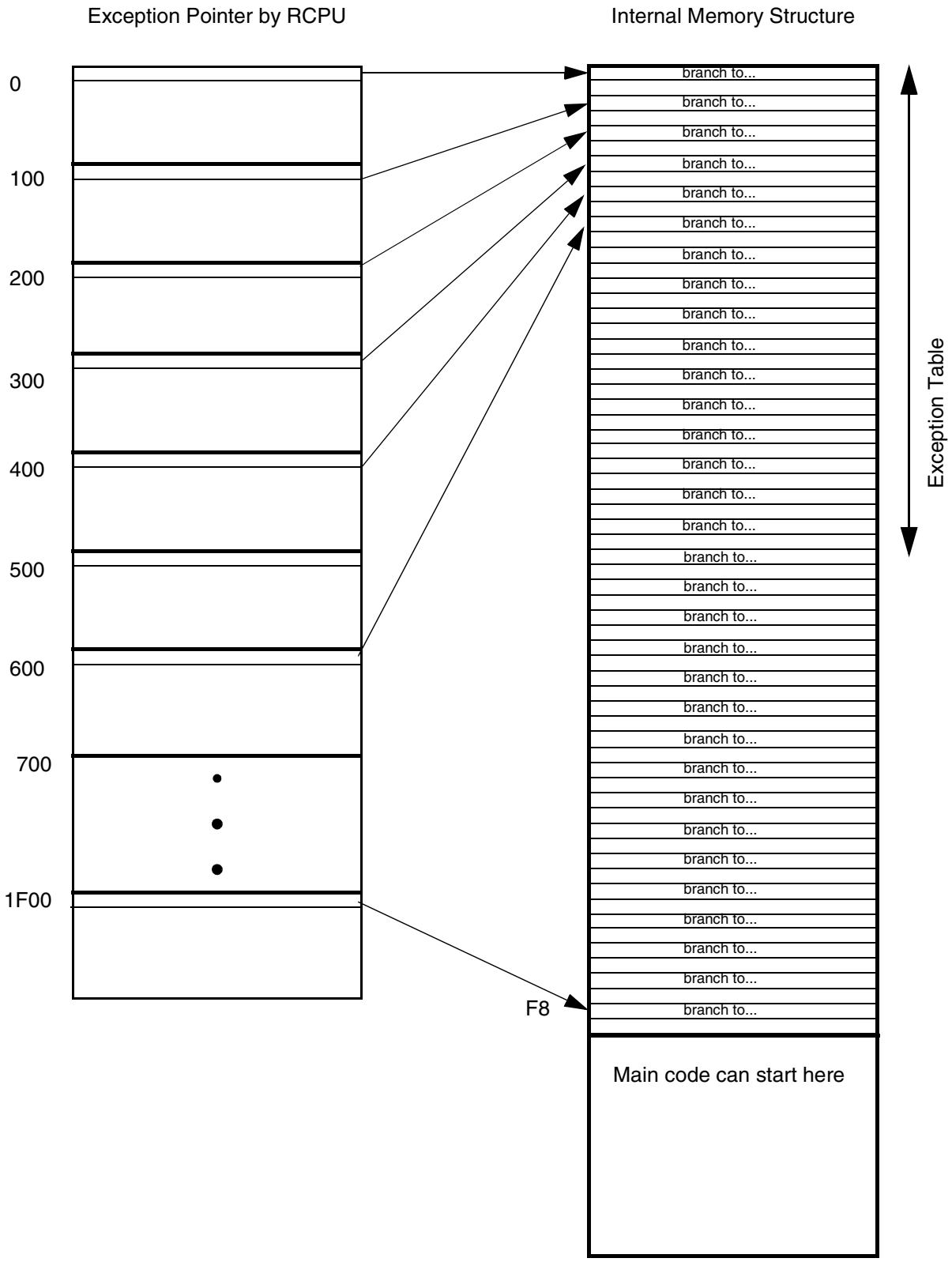


Figure 4-14 Exception Table Entries Mapping

4.6 Burst Buffer Programming Model

The BBC and IMPU module configuration registers are MPC555 / MPC556 special-purpose registers (SPRs). They are programmed with the MPC555 / MPC556 **mtspr/ mfspr** instructions.

All the registers can be accessed in supervisor mode only. The processor generates an exception internally if an attempt is made to access the registers from user mode.

The following 32-bit registers contain the starting address and the size of the region. There is one register for each region.

Table 4-2 Region Base Address Registers RBA[0:1]

Register Name	Address (Decimal)	ub_addr[18:27] (hex)
MI_RBA[0]	784	0x2180
MI_RBA[1]	785	0x2380
MI_RBA[2]	786	0x2580
MI_RBA[3]	787	0x2780

The following registers hold the attributes of the corresponding regions and of the default region. Each of the four MI_RAx registers contains access permission attributes. The MI_GRA (global region attribute) register contains two additional bits to enable each of the MI_RBAx registers.

Table 4-3 Region Attributes Registers

Register Name	Address (Decimal)	ub_addr [18:27] (Hex)
MI_RA[0]	816	0x2190
MI_RA[1]	817	0x2390
MI_RA[2]	818	0x2590
MI_RA[3]	818	0x2790
MI_GRA	528	0x2100

The BBC holds only one register, the BBC module configuration register (BBCMCR).

Table 4-4 BBC Module Configuration Register

Register name	Addr (Decimal)	ub_addr [0:31] (Hex)
BBCMCR	560	0x2110

4.6.1 Region Base Address Registers

MI_RBA[0:3] — Region Base Address Register **SPR 784 – 787**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RA															
RESET:															
Unaffected by Reset															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RA				Reserved											
RESET:															
Unaffected by Reset															

Table 4-5 MI_RBA[0:3] Bit Descriptions

Bit(s)	Name	Description
0:19	RA	Region address. This field defines the base address (most significant 20 bits) for the region.
20:31	—	Reserved

4.6.2 Region Attribute Registers MI_RA[0:3] Description

MI_RA[0:3] — Region Attribute Register **SPR 816 – 819**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RS															
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RS				PP		RESERVED			G	CMPR[1]		RESERVED			
HRESET															
U	U	U	U	U	U	0	0	0	U	U	U	U	0	0	0

NOTES:
1. Available only on the MPC556.

Table 4-6 MI_RA[0:3] Registers Bits Description

Bit(s)	Name	Description
0:19	RS	Region size. The region size is a power of two, determined as follows: 0000_0000_0000_0000_0000 — 4 Kbytes 0000_0000_0000_0000_0001 — 8 Kbytes 0000_0000_0000_0000_0011 — 16 Kbytes 0000_0000_0000_0000_0111 — 32 Kbytes 0000_0000_0000_0000_1111 — 64 Kbytes 0000_0000_0000_0001_1111 — 128 Kbytes 0000_0000_0000_0011_1111 — 256 Kbytes 0000_0000_0000_0111_1111 — 512 Kbytes 0000_0000_0000_1111_1111 — 1 Mbyte 0000_0000_0001_1111_1111 — 2 Mbytes 0000_0000_0011_1111_1111 — 4 Mbytes 0000_0000_0111_1111_1111 — 8 Mbytes 0000_0000_1111_1111_1111 — 16 Mbytes 0000_0001_1111_1111_1111 — 32 Mbytes 0000_0011_1111_1111_1111 — 64 Mbytes 0000_0111_1111_1111_1111 — 128 Mbytes 0000_1111_1111_1111_1111 — 256 Mbytes 0001_1111_1111_1111_1111 — 512 Mbytes 0011_1111_1111_1111_1111 — 1 Gbyte 0111_1111_1111_1111_1111 — 2 Gbytes 1111_1111_1111_1111_1111 — 4 Gbytes
20:31	PP[1]	Protection bits: 00: Supervisor — No Access, User — No Access. 01: Supervisor — Fetch, User — No Access. 1x: Supervisor — Fetch, User — Fetch.
22:24	—	Reserved
25	G	Guard attribute for region 0 = Speculative fetch is not prohibited from region. Region is not guarded. 1 = Speculative fetch is prohibited from guarded region. An exception will occur under such attempt.
26:27	CMPR[2]	Compressed Region. x0 = The region in not restricted. 01 = Region is considered a non-compressed code region. Access to the region is allowed only in “Decompression Off” mode. 11 = Region is considered a compressed code region. Access to the region is allowed only in “Decompression On” mode.
28:30	—	Reserved

NOTES:
1. G and PP attributes perform similar protection activities on a region. The more protective attribute will be implied on the region if the attributes programming oppose each other.
2. This bit is available only on the MPC556.

4.6.3 Global Region Attribute Register Description (MI_GRA)

MI_GRA — Global Region Attribute Register **SPR 528**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ENR0	ENR1	ENR2	ENR3	RESERVED											
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				PP		RESERVED			G	CMPR[1]		RESERVED			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. Available only on the MPC556.

Table 4-7 MI_GRA Bit Descriptions

Bit(s)	Name	Description
0	ENR0	Enable region 0 of IMPU 0 = Region 0 is off 1 = Region 0 is on
1	ENR1	Enable region 1 of IMPU 0 = Region 1 is off 1 = Region 1 is on
2	ENR2	Enable region 2 of IMPU 0 = Region 2 is off 1 = Region 2 is on
3	ENR3	Enable region 3 of IMPU 0 = Region 3 is off 1 = Region 3 is on
4:19	—	Reserved
20:21	PP	Protection bits 00 = No supervisor access, no user access 01 = Supervisor fetch access, no user access 10 = Supervisor fetch access, user fetch access 11 = Supervisor fetch access, user fetch access
22:24	—	Reserved
25	G	Guarded attribute for region 0 = Fetch is allowed from guarded region. 1 = Fetch is prohibited from guarded region. An attempted fetch will generate an exception.
26:27	CMPR[1]	Compressed region x0 = The region in not restricted 01 = Region is considered a non-compressed code region. Access to the region is allowed only in "Decompression Off" mode 11 = Region is considered a compressed code region. Access to the region is allowed only in "De-compression On" mode
28:31	—	Reserved

NOTES:
1. Available only on the MPC556.

4.6.4 BBC Module Configuration Register (BBCMCR)

BBCMCR — BBC Module Configuration Register **SPR 560**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED		BE	ETRE	OERC	EN_ COMP[2]	EXC_ COMP[2]	DECOMP _SC_EN[2]	RESERVED							
RE-SET:															
0	0	0	ID[19][1]	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. Reset value is taken from the indicated bit of the reset configuration word.
2. Available only on the MPC556.

Table 4-8 BBCMCR Bit Descriptions

Bit(s)	Name	Description
16:17	—	Reserved
18	BE	Burst enable 0 = BBC does not request burst accesses 1 = BBC requests burst accesses
19	ETRE	Exception table relocation enable 0 = Exception table relocation is off — the BBC does *not* map exception addresses 1 = Exception table relocation is on — the BBC maps exception addresses to a branch instruction table. Refer to **4.5 Exception Table Relocation**.
20	OERC	Other exceptions relocation control. 0 = All exceptions except reset are mapped to the internal memory base address. 1 = All exceptions except reset are mapped to the internal memory base address + 32 Kbytes.
21	EN_COMP	Enable COMPression – This bit enables the operation of the MPC556 in "Compression ON" mode. The default state is disabled. This bit is read only. 0 = "Decompression ON" mode is disabled. The MPC556 operates only in "Decompression OFF" mode. 1 = "Decompression ON" mode is enabled. The MPC556 may operates with both "Decompression ON" and "Decompression OFF" modes. The bit value is determined by reset configuration word, bit #21.
22	EXC_ COMP	Exception Compression – This bit determines the operation of the MPC556 with exceptions. If this bit is set, the MPC556 assumes that the all exception routines code is compressed; otherwise it is assumed that all exception routines code is not compressed. The reset value is determined by reset configuration word bit #22. 0 = The MPC556 assumes that exception routines are non-compressed 1 = The MPC556 assumes that ALL exception routines are compressed. This bit effects only when EN_COMP bit is set.

Table 4-8 BBCMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
23	DECOMP_ SC_EN	DECOMPression Show Cycle ENable - This bit determines the way the MPC556 executes instruction show-cycle. The reset value is determined by configuration word bit #21. For further details regarding show cycles execution in "Decompression ON" mode see **4.3.9 Decompression**. 0 = Decompression Show Cycle does not include the bit pointer. 1 = Decompression Show Cycles includes the bit pointer information on the data bus.
24:31	—	Reserved

NOTE

An ISYNC instruction is required immediately following any write to the BBCMCR.

SECTION 5
UNIFIED SYSTEM INTERFACE UNIT

The unified system interface unit (USIU) of the MPC555 / MPC556 controls system start-up, system initialization and operation, system protection, and the external system bus. The MPC555 / MPC556 USIU functions include the following:

- System configuration and protection
- Interrupt controller
- System reset monitoring and generation
- Clock synthesizer
- Power management
- External bus interface (EBI) control
- Memory controller
- Debug support

5.1 Module Overview

The system configuration and protection function controls the overall system configuration and provides various monitors and timers, including the bus monitor, software watchdog timer, periodic interrupt timer, PowerPC decrementer, time base, and real time clock. The interrupt controller and USIU are also included in the system configuration and protection function. Refer to **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** for details.

The reset controller receives input from a number of reset sources and takes appropriate actions, depending on the source. The reset status register (RSR) reflects the most recent source to cause a reset. Refer to **SECTION 7 RESET** for details.

The clock synthesizer generates the clock signals used by the SIU as well as the other modules and external devices. This circuitry can generate the system clock from a 4-MHz or 20-MHz crystal.

The SIU supports various low-power modes. Each supplies a different range of power consumption, functionality and wake-up time. Clock generation and low-power modes are described in **SECTION 8 CLOCKS AND POWER CONTROL**.

The external bus interface (EBI) handles the transfer of information between the internal busses and the memory or peripherals in the external address space. The MPC555 / MPC556 is designed to allow external bus masters to request and obtain mastership of the system bus, and if required access the on-chip memory and registers. **SECTION 9 EXTERNAL BUS INTERFACE** describes the bus operation.

The memory controller module provides a glueless interface to many types of memory devices and peripherals. It supports up to four memory banks, each with its own device

and timing attributes. See **SECTION 10 MEMORY CONTROLLER** for more information.

5.2 SIU Architecture

Figure 5-1 is a block diagram of the MPC555 / MPC556 USIU.

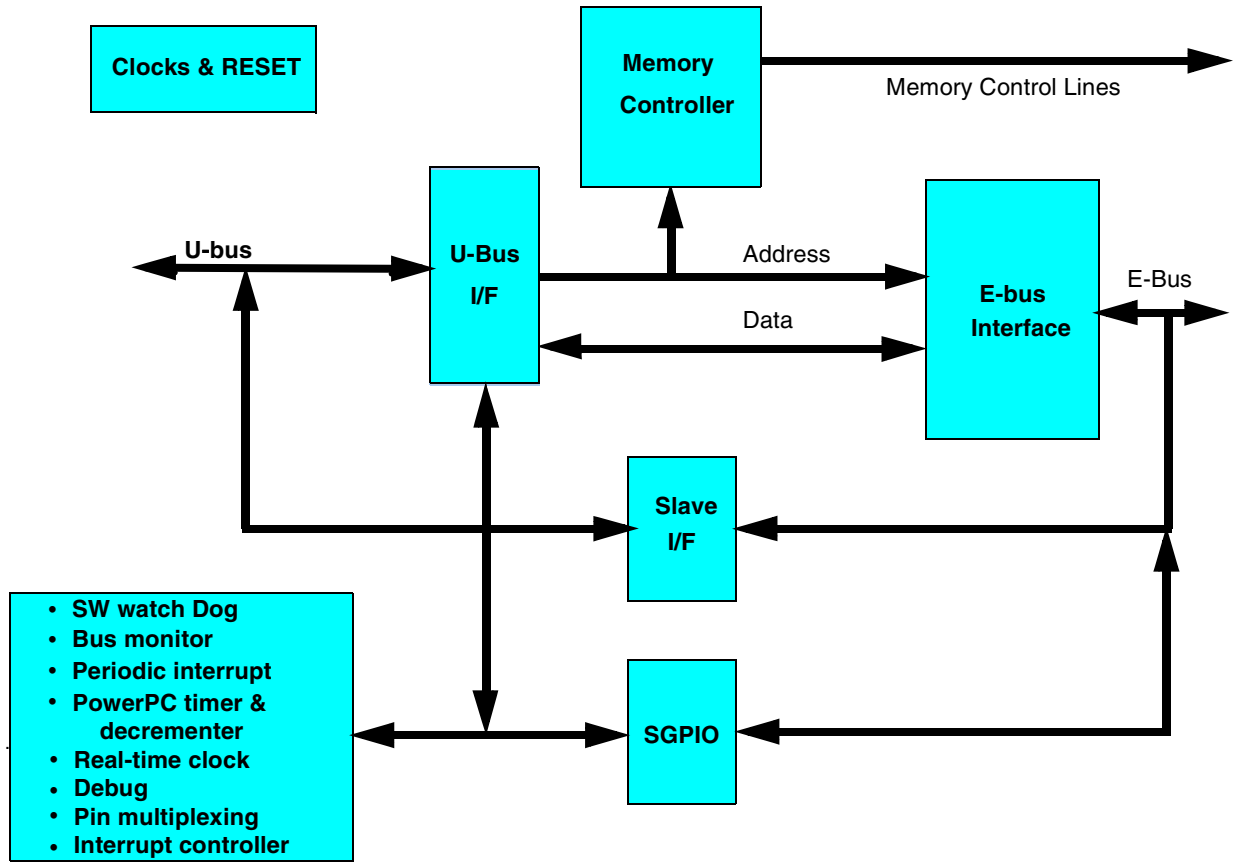


Figure 5-1 MPC555 / MPC556 USIU Block Diagram

5.3 USIU Address Map

Table 5-1 is an address map of the SIU registers. Where not otherwise noted, registers are 32 bits wide. The address shown for each register is relative to the base address of the MPC555 / MPC556 internal memory map. The internal memory block can reside in one of eight possible 4-Mbyte memory spaces. See **Figure 1-3** in **SECTION 1 OVERVIEW** for details.

Table 5-1 USIU Address Map

Address	Register
0x2F C000	SIU Module Configuration Register (SIUMCR) See **Table 6-5** for bit descriptions.
0x2F C004	System Protection Control Register (SYPCR) See **Table 6-13** for bit descriptions.
0x2F C008	Reserved
0x2F C00E[1]	Software Service Register (SWSR) See **Table 6-14** for bit descriptions.
0x2F C010	Interrupt Pending Register (SIPEND) See **6.13.2.1 SIPEND Register** for bit descriptions.
0x2F C014	Interrupt Mask Register (SIMASK) See **6.13.2.2 SIU Interrupt Mask Register (SIMASK)** for bit descriptions.
0x2F C018	Interrupt Edge Level Mask (SIEL) See **6.13.2.3 SIU Interrupt Edge Level Register (SIEL)** for bit descriptions.
0x2F C01C	Interrupt Vector (SIVEC) See **6.13.2.4 SIU Interrupt Vector Register** for bit descriptions.
0x2F C020	Transfer Error Status Register(TESR) See **Table 6-15** for bit descriptions.
0x2F C024	USIU General-Purpose I/O Data Register (SGPIODT1) See **Table 6-21** for bit descriptions.
0x2F C028	USIU General-Purpose I/O Data Register 2 (SGPIODT2) See **Table 6-22** for bit descriptions.
0x2F C02C	USIU General-Purpose I/O Control Register (SGPIOCR) See **Table 6-23** for bit descriptions.
0x2F C030	External Master Mode Control Register (EMCR) See **Table 6-12** for bit descriptions.
0x2F C03C[1]	Pads Module Configuration Register (PDMCR) See **Table 2-3** for bit descriptions.
0x2F C040 — 0x2F C0FC	Reserved
Memory Controller Registers	
0x2F C100	Base Register 0 (BR0) See **Table 10-7** for bit descriptions.
0x2F C104	Option Register 0 (OR0) See **Table 10-8** for bit descriptions.
0x2F C108	Base Register 1 (BR1) See **Table 10-7** for bit descriptions.
0x2F C10C	Option Register 1 (OR1) See **Table 10-8** for bit descriptions.
0x2F C110	Base Register 2 (BR2) See **Table 10-7** for bit descriptions.
0x2F C114	Option Register 2 (OR2) See **Table 10-8** for bit descriptions.
0x2F C118	Base Register 3 (BR3) See **Table 10-7** for bit descriptions.
0x2F C11C	Option Register 3 (OR3) See **Table 10-8** for bit descriptions.

Table 5-1 USIU Address Map (Continued)

Address	Register
0x2F C120 — 0x2F C13C	Reserved
0x2F C140	Dual-Mapping Base Register (DMBR) See **Table 10-9** for bit descriptions.
0x2F C144	Dual-Mapping Option Register (DMOR) See **Table 10-10** for bit descriptions.
0x2F C148 — 0x2F C174	Reserved
0x2F C178[1]	Memory Status (MSTAT) See **Table 10-6** for bit descriptions.
0x2F C17A — 0x2F C1FFC	Reserved
System Integration Timers	
0x2F C200	Time Base Status and Control (TBSCR) See **Table 6-16** for bit descriptions.
0x2F C204	Time Base Reference 0 (TBREF0) See **6.13.4.3 Time Base Reference Registers** for bit descriptions.
0x2F C208	Time Base Reference 1 (TBREF1) See **6.13.4.3 Time Base Reference Registers** for bit descriptions.
0x2F C20C — 0x2F C21C	Reserved
0x2F C220	Real-Time Clock Status and Control (RTCSC) See **Table 6-17** for bit descriptions.
0x2F C224	Real-Time Clock (RTC) See **6.13.4.6 Real-Time Clock Register (RTC)** for bit descriptions.
0x2F C228	Real-Time Alarm Seconds (RTSEC) — Reserved
0x2F C22C	Real-Time Alarm (RTCAL) See **6.13.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.
0x2F C230 — 0x2F C23C	Reserved
0x2F C240	PIT Status and Control (PISCR) See **Table 6-18** for bit descriptions.
0x2F C244	PIT Count (PITC) See **Table 6-19** for bit descriptions.
0x2F C248	PIT Register (PITR) See **Table 6-20** for bit descriptions.
0x2F C24C — 0x2F C27C	Reserved
Clocks and Reset	
0x2F C280	System Clock Control Register (SCCR) See **Table 8-9** for bit descriptions.
0x2F C284	PLL Low-Power and Reset Control Register (PLPRCR) See **Table 8-10** for bit descriptions.
0x2F C288[1]	Reset Status Register (RSR) See **Table 7-3** for bit descriptions.
0x2F C28C[1]	Change of Lock Interrupt Register (COLIR) See **Table 8-11** for bit descriptions.

Table 5-1 USIU Address Map (Continued)

Address	Register
0x2F C290[1]	VDDSRM Control Register (VSRMCR) See **Table 8-12** for bit descriptions.
0x2F C294 — 0x2F C2FC	Reserved
System Integration Timer Keys	
0x2F C300	Time Base Status and Control Key (TBSCRK) See **Table 8-8** for bit descriptions.
0x2F C304	Time Base Reference 0 Key (TBREF0K) See **Table 8-8** for bit descriptions.
0x2F C308	Time Base Reference 1 Key (TBREF1K) See **Table 8-8** for bit descriptions.
0x2F C30C	Time Base and Decrementer Key (TBK) See **Table 8-8** for bit descriptions.
0x2F C310 — 0x2F C31C	Reserved
0x2F C320	Real-Time Clock Status and Control Key (RTCSCK) See **Table 8-8** for bit descriptions.
0x2F C324	Real-Time Clock Key (RTCK) See **Table 8-8** for bit descriptions.
0x2F C328	Real-Time Alarm Seconds Key (RTSECK) See **Table 8-8** for bit descriptions.
0x2F C32C	Real-Time Alarm Key (RTCALK) See **Table 8-8** for bit descriptions.
0x2F C330 — 0x2F C33C	Reserved
0x2F C340	PIT Status and Control Key (PISCRIK) See **Table 8-8** for bit descriptions.
0x2F C344	PIT Count Key (PITCK) See **Table 8-8** for bit descriptions.
0x2F C348 — 0x2F C37C	Reserved
Clocks and Reset Keys	
0x2F C380	System Clock Control Key (SCCRK) See **Table 8-8** for bit descriptions.
0x2F C384	PLL Low-Power and Reset Control Register Key (PLPRCRK) See **Table 8-8** for bit descriptions.
0x2F C388	Reset Status Register Key (RSRK) See **Table 8-8** for bit descriptions.
0x2F C38C — 0x2F C3FC	Reserved

NOTES:
1. 16-bit register.

5.4 USIU PowerPC Memory Map

Table 5-2 lists the USIU PowerPC special-purpose registers (SPR). These registers can be accessed with the PowerPC **mtspr** and **mfspr** instructions, or from an external master (refer to **6.2 External Master Modes** for details). All registers are 32 bits wide.

Table 5-2 USIU Special-Purpose Registers

Internal Address[0:31]	Register	Decimal Address spr[5:9]:spr[0:4][1]
0x2C00	Decrementer (DEC)	22
0x1880	Time Base — Read (TB)	268
0x1A80	Time Base Upper — Read (TBU)	269
0x3880	Time Base — Write (TB)	284
0x3A80	Time Base Upper — Write (TBU)	285
0x3B30	Internal Memory Mapping Register (IMMR)	638

NOTES:
1. Bits [0:17] and [28:31] are all 0.

Table 5-3 shows the PowerPC special address range. For an external master accessing a PowerPC SPR, address bits [0:17] and [28:31] are compared to zeros to confirm that an SPR access is valid.

Table 5-3 PowerPC Address Range

0:17	18:27	28:31
0 . . . 0	spr[0:9]	0000

SECTION 6
SYSTEM CONFIGURATION AND PROTECTION

The MPC555 / MPC556 incorporates many system functions that normally must be provided in external circuits. In addition, it is designed to provide maximum system safeguards again hardware and/or software faults. The system configuration and protection sub-module provides the following features:

- **System Configuration** — The USIU allows the user to configure the system according to the particular requirements. The functions include control of show cycle operation, pin multiplexing, and internal memory map location. System configuration also includes a register containing part and mask number constants to identify the part in software.
- **Interrupt Configuration** — The interrupt controller receives interrupt requests from a number of internal and external sources and directs them on a single interrupt-request line to the RCPU.
- **General-Purpose I/O** — The USIU provides 64 pins for general-purpose I/O. The SGPIO pins are multiplexed with the address and data pins.
- **External Master Modes Support —** External master modes are special modes of operation that allow an alternate master on the external bus to access the internal modules for debugging and backup purposes.
- **Bus Monitor** — The SIU provides a bus monitor to watch internal to external accesses. It monitors the transfer acknowledge (TA) response time for internal to external transfers. A transfer error acknowledge ($\overline{TEA}$) is asserted if the TA response limit is exceeded. This function can be disabled.
- **Software Watchdog Timer (SWT)** — The SWT asserts a reset or non-maskable interrupt (as selected by the system protection control register) if the software fails to service the SWT for a designated period of time (e.g, because the software is trapped in a loop or lost). After a system reset, this function is enabled with a maximum time-out period and asserts a system reset if the time-out is reached. The SWT can be disabled or its time-out period can be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset.
- **Periodic Interrupt Timer (PIT)** — The SIU provides a timer to generate periodic interrupts for use with a real-time operating system or the application software. The PIT provides a period from 1 µs to 4 seconds with a 4-Mhz crystal or 200 ns to 0.8 ms with a 20-Mhz crystal. The PIT function can be disabled.
- **Power-PC Time Base Counter (TB)** — The TB is a 64-bit counter defined by the MPC555 / MPC556 architecture to provide a time base reference for the operating system or application software. The TB has four independent reference registers which can generate a maskable interrupt when the time-base counter reaches the value programmed in one of the four reference registers. The associated bit in the TB status register will be set for the reference register which generated the interrupt.
- **Power-PC Decrementer (DEC)** — The DEC is a 32-bit decrementing counter de-

fined by the MPC555 / MPC556 architecture to provide a decrementer interrupt. This binary counter is clocked by the same frequency as the time base (also defined by the MPC555 / MPC556 architecture). The period for the DEC when driven by a 4-Mhz oscillator is 4295 seconds, which is approximately 71.6 minutes.

- **Real-Time Clock (RTC)** — The RTC is used to provide time-of-day information to the operating system or application software. It is composed of a 45-bit counter and an alarm register. A maskable interrupt is generated when the counter reaches the value programmed in the alarm register. The RTC is clocked by the same clock as the PIT.
- **Freeze Support** — The SIU allows control of whether the SWT, PIT, TB, DEC and RTC should continue to run during the freeze mode.

Figure 6-1 shows a block diagram of the system configuration and protection logic.

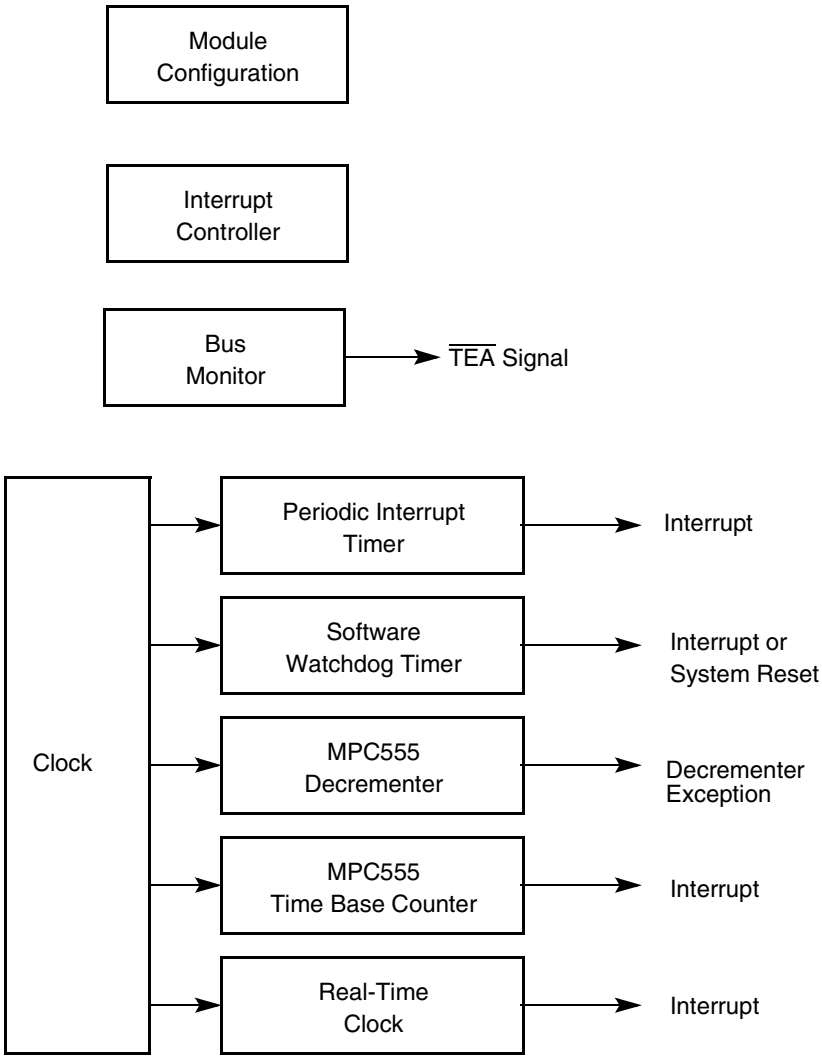


Figure 6-1 System Configuration and Protection Logic

6.1 System Configuration

The SIU allows the user to configure the system according to the particular requirements. The functions include control of show cycle operation, pin multiplexing, and internal memory map location. System configuration also includes a register containing part and mask number constants to identify the part in software.

System configuration registers include the system configuration register (SIUMCR), the internal memory mapping register (IMMR). Refer to **6.13 System Configuration and Protection Registers** for register diagrams and bit descriptions.

6.1.1 USIU Pins Multiplexing

Some of the functions defined in the various sections of the SIU (external bus interface, memory controller, and general-purpose I/O) share pins. **Table 6-1** summarizes how the pin functions of these multiplexed pins are assigned.

Table 6-1 USIU Pins Multiplexing Control

Pin Name	Multiplexing Controlled By:
$\overline{\text{IRQ0}}$/SGPIOC0 $\overline{\text{IRQ1}}$/$\overline{\text{RSV}}$/SGPIOC1 $\overline{\text{IRQ2}}$/$\overline{\text{CR}}$/SGPIOC2/$\overline{\text{MTS}}$ $\overline{\text{IRQ3}}$/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC3 $\overline{\text{IRQ4}}$/AT2/SGPIOC4 $\overline{\text{IRQ5}}$/SGPIOC5/MODCK1 $\overline{\text{IRQ6}}$/MODCK2 $\overline{\text{IRQ7}}$/MODCK3	At Power-on reset: MODCK[1:3] Otherwise: programmed in SIUMCR (See **6.13.1.1 SIU Module Configuration Register**.)
SGPIOC6/FRZ/PTR SGPIOC7/$\overline{\text{IRQ_OUT}}$/LWP0 $\overline{\text{BG}}$/VF0/LWP1 $\overline{\text{BR}}$/VF1/IWP2 $\overline{\text{BB}}$/VF2/IWP3 IWP[0:1]/VFLS[0:1] $\overline{\text{BI}}$/$\overline{\text{STS}}$ $\overline{\text{WE}}$(0:3)/$\overline{\text{BE}}$(0:3)/$\overline{\text{AT}}$(0:3) TDI/DSDI TCK/DSCK TDO/DSDO	Programmed in the SIUMCR and via the hard reset configuration. (See **6.13.1.1 SIU Module Configuration Register** and **7.5.2 Hard Reset Configuration Word**.)
DATA[0:31]/SGPIOD[0:31] ADDR[8:31]/SGPIOA[8:31]	Programmed in SIUMCR and hard reset configuration. (See **6.13.1.1 SIU Module Configuration Register** and **7.5.2 Hard Reset Configuration Word**.)
RSTCONF/TEXP	At Power-on reset: RSTCONF Otherwise: programmed in SIUMCR. (See **6.13.1.1 SIU Module Configuration Register**.)

6.1.2 Memory Mapping

The MPC555 / MPC556 internal memory space can be assigned to one of eight locations.

The internal memory map is organized as a single 4-Mbyte block. The user can assign this block to one of eight locations by programming the ISB field in the internal memory

mapping register (IMMR). The eight possible locations are the first eight 4-Mbyte memory blocks starting with address 0x0000 0000. (Refer to **Figure 6-2**.)

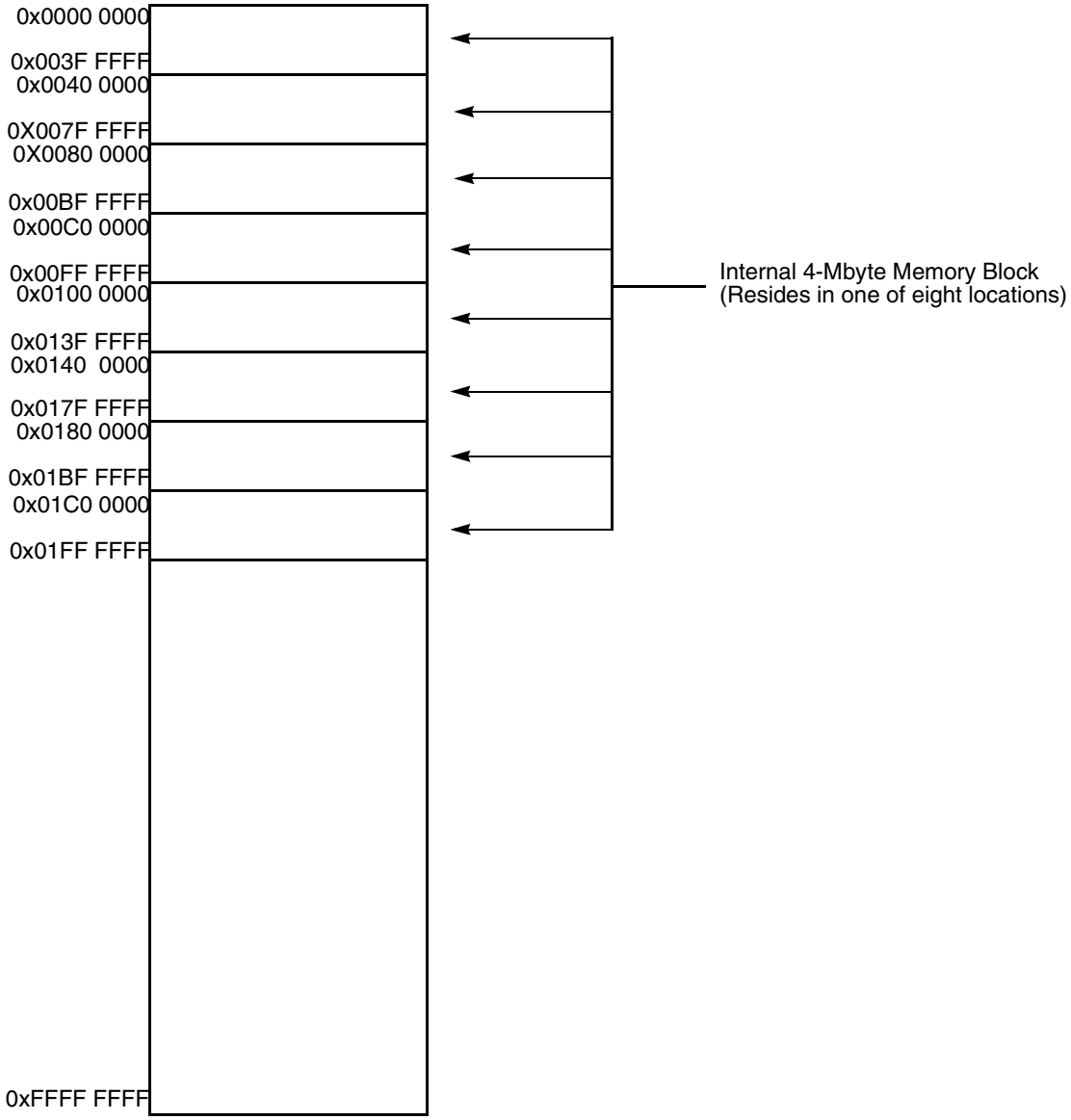


Figure 6-2 MPC555 / MPC556 Memory Map

6.1.3 Arbitration Support

Two bits in the SIUMCR control USIU bus arbitration. The external arbitration (EARB) bit determines whether arbitration is performed internally or externally. If EARB is cleared (internal arbitration), the external arbitration request priority (EARP) bit determines the priority of an external master's arbitration request. The operation of the internal arbiter is described in **9.5.6.4 Internal Bus Arbiter**.

6.2 External Master Modes

External master modes are special modes of operation that allow an alternate master on the external bus to access the internal modules for debugging and backup purposes. They provide access to the internal buses (U-bus and L-bus) and to the intermodule bus (IMB3).

There are two external master modes. Peripheral mode (enabled by setting PRPM in the EMCR) is a special slave mechanism in which the RCPU is shut down and an alternate master on the external bus can perform accesses to any internal bus slave. Slave mode (enabled by setting SLVM and clearing PRPM in the EMCR) enables an external master to access any internal bus slave while the RCPU is fully operational. Both modes can be enabled and disabled by software. In addition, peripheral mode can be selected from reset.

The internal bus is not capable of providing fair priority between internal RCPU accesses and external master accesses. If the bandwidth of external master accesses is large, it is recommended that the system forces gaps between external master accesses in order to avoid suspension of internal RCPU activity.

The MPC555 / MPC556 does not support burst accesses from an external master; only single accesses of 8, 16, or 32 bits can be performed. The MPC555 / MPC556 asserts burst inhibit ($\overline{BI}$) on any attempt to initiate a burst access to internal memory.

The MPC555 / MPC556 provides memory controller services for external master accesses (single and burst) to external memories. See **SECTION 10 MEMORY CONTROLLER** for details.

6.2.1 Operation of External Master Modes

The external master modes are controlled by the EMCR register, which contains the internal bus attributes. The default attributes in the EMCR enable the external master to configure EMCR with the required attributes, and then access the internal registers. The external master must be granted external bus ownership in order to initiate the external master access. The SIU compares the address on the external bus to the allocated internal address space. If the address is within the internal space, the access is performed with the internal bus. The internal address space is determined according to ISB (see **6.13.1.2 Internal Memory Map Register** for details). The external master access is terminated by the $\overline{TA}$, $\overline{TEA}$ or $\overline{RETRY}$ signal on the external bus.

A deadlock situation might occur if an internal-to-external access is attempted on the internal bus while an external master access is initiated on the external bus. In this case, the SIU will assert the $\overline{RETRY}$ on the external bus in order to relinquish and retry the external access until the internal access is completed. The internal bus will deny other internal accesses for the next eight clocks in order to complete the pending accesses and prevent additional internal accesses from being initiated on the internal bus. The SIU will also mask internal accesses to support consecutive external accesses if the delay between the external accesses is less than 4 clocks. The external master access and retry timings are described in **9.5.11 Bus Operation in External Master Modes**.

The external master may access the internal MPC555 / MPC556 special registers that are located outside the RCPU. In order to access one of these MPC555 / MPC556 registers, program the EMCR to MPC555 / MPC556 special register access (CONT = 1 and SUPU = 0 in EMCR). Next, access the register by providing the address according to the MPC555 / MPC556 address map. Only the first external master access that follows EMCR setting will be assigned to the special register map; the next accesses will be directed to the normal address map. This is done in order to enable the user to access the EMCR again after the required MPC555 / MPC556 special register access.

Peripheral mode does not require external bus arbitration between the external master and the internal RCPU, since the internal RCPU is disabled. The $\overline{BR}$ and $\overline{BB}$ signals should be connected to ground, and the internal bus arbitration should be selected in order to prevent the “slave” MPC555 / MPC556 from occupying the external bus. Internal bus arbitration is selected by clearing the EARB bit in the SIUMCR (see **6.13.1.1 SIU Module Configuration Register**).

6.2.2 Address Decoding for External Accesses

During an external master access, the USIU compares the external address with the internal address block to determine if MPC555 / MPC556 operation is required. Since only 24 of the 32 internal address bits are available on the external bus, the USIU assigns zeros to the most significant address bits (ADDR[0:7]).

The address compare sequence can be summarized as follows:

- Normal external access. If the CONT bit in EMCR is cleared, the address is compared to the internal address map.
- MPC555 / MPC556 special register external access. If the CONT bit in EMCR is set by the previous external master access, the address is compared to the MPC555 / MPC556 special address range. See **5.4 USIU PowerPC Memory Map** for a list of the SPRs in the USIU.
- Memory controller external access. If the first two comparisons do not match, the internal memory controller determines whether the address matches an address assigned to one of the regions. If it finds a match, the memory controller generates the appropriate chip select and attribute accordingly

When trying to fetch an MPC555 / MPC556 special register from an external master, the address might be aliased to one of the external devices on the external bus. If this device is selected by the MPC555 / MPC556 internal memory controller, this aliasing does not occur since the chip select is disabled. If the device has its own address decoding or is being selected by external logic, this case should be resolved.

6.3 USIU General-Purpose I/O

The USIU provides 64 general-purpose I/O (SGPIO) pins. The SGPIO pins are multiplexed with the address and data pins. In single-chip mode, where communicating with external devices is not required, the user can use all 64 SGPIO pins. In multiple-chip mode, only eight SGPIO pins are available. Another configuration allows the use of the address bus for instruction show cycles while the data bus is dedicated to SGPIO func-

tionality. The functionality of these pins is assigned by the single-chip (SC) bit in the SIUMCR. (See **6.13.1.1 SIU Module Configuration Register**.)

SGPIO pins are grouped as follows:

- Six groups of eight pins each, whose direction is set uniformly for the whole group
- 16 single pins whose direction is set separately for each pin

Table 6-2 describes the SGPIO signals, and all available configurations. The SGPIO registers are described in **6.13.5 General-Purpose I/O Registers**.

Table 6-2 SGPIO Configuration

SGPIO Group Name	Individual Pin Control	Direction Control	Available When SC = 00 (32-bit Port Size Mode)	Available When SC = 01 (16-bit Port Size Mode)	Available When SC = 10 (Single-Chip Mode with Trace)	Available When SC = 11 (Single-Chip Mode)
SGPIOD[0:7]		GDDR0			X	X
SGPIOD[8:15]		GDDR1			X	X
SGPIOD[16:23]		GDDR2		X	X	X
SGPIOD[24:31]	X	SDDRD[23:31]		X	X	X
SGPIOC[0:7][1]	X	SDDRC[0:7]				
SGPIOA[8:15]		GDDR3				X
SGPIOA[16:23]		GDDR4				X
SGPIOA[24:31]		GDDR5				X

NOTES:
1. SGPIOC[0:7] is selected according to GPC and MLRC fields in SIUMCR. See **6.13.1.1 SIU Module Configuration Register**.

Figure 6-3 illustrates the functionality of the SGPIO.

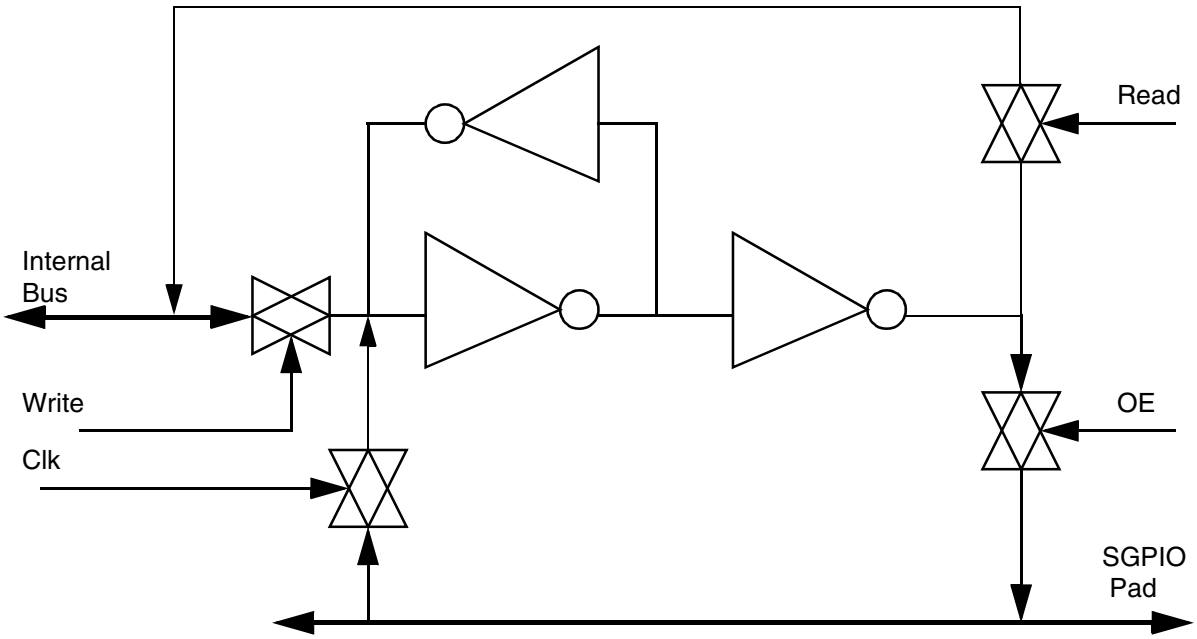


Figure 6-3 SGPIO Cell

6.4 Interrupt Controller

The USIU receives interrupts from internal sources (such as the PIT and RTC), from the IMB3 module (which has its own interrupt controller), and from external pins $\overline{\text{IRQ}}$[0:7]. An overview of the MPC555 / MPC556 interrupt structure is shown in **Figure 6-4**.

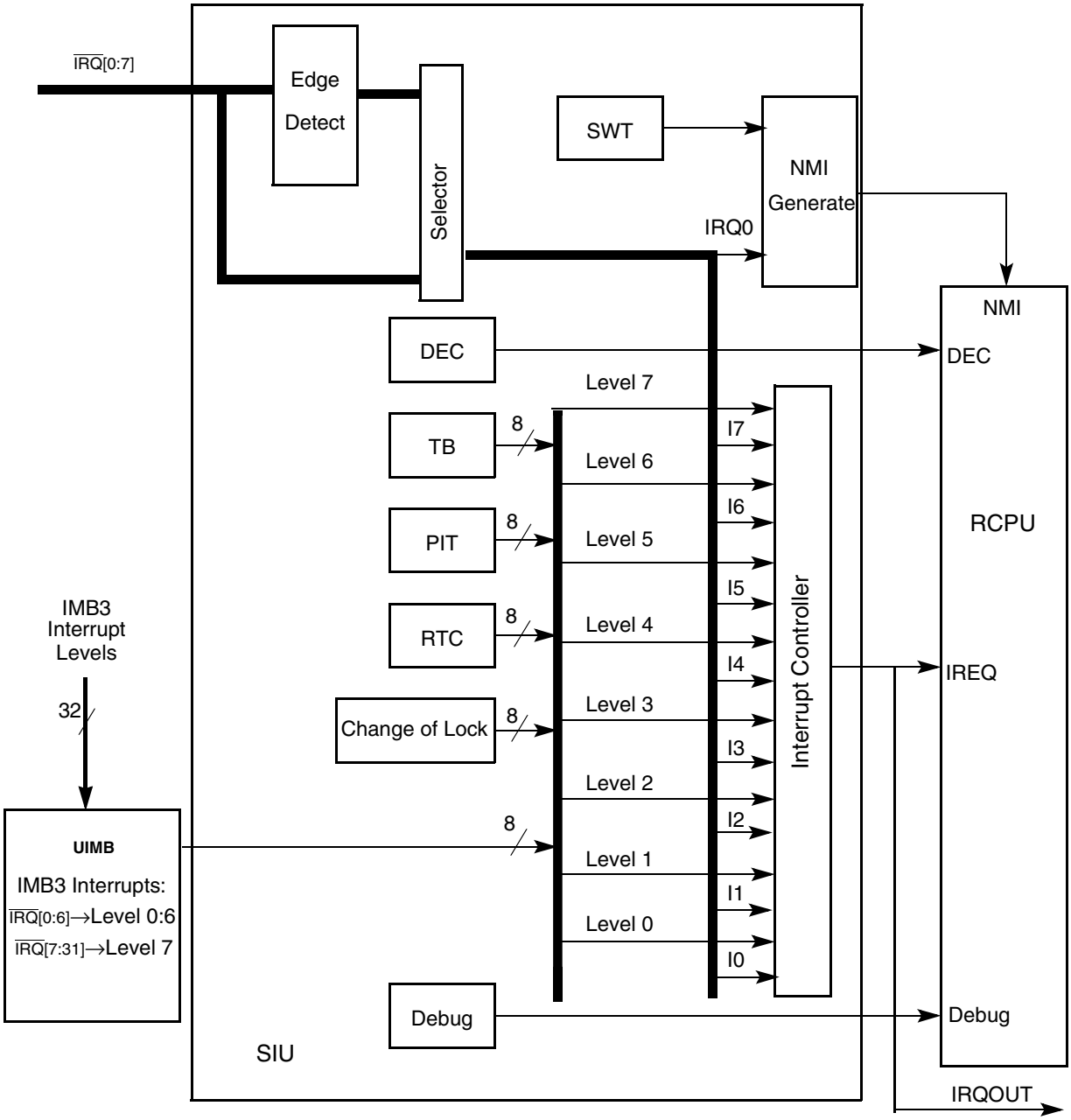


Figure 6-4 MPC555 / MPC556 Interrupt Structure

If programmed to generate interrupts, the SWT and external pin $\overline{\text{IRQ}}$[0] always generate a non-maskable interrupt (NMI) to the RCPU. Notice that the RCPU takes the system reset interrupt when an NMI is asserted and the external interrupt for any other interrupt asserted by the interrupt controller.

Each one of the external pins $\overline{\text{IRQ}}$[1:7] has its own dedicated assigned priority level. $\overline{\text{IRQ}}$[0] is also mapped but should be used only as a status bit indicating that $\overline{\text{IRQ}}$[0] was asserted and generated an NMI interrupt. There are eight additional interrupt pri-

ority levels. Each one of the SIU internal interrupt sources, as well as the interrupt requests generated by the IMB3 modules, can be assigned by the software to any one of those eight interrupt priority levels.

The same interrupt request signal that is generated within the RCPU is optionally driven on the IRQ_OUT pin. This pin may be used in peripheral mode, in which the internal processor is shut off and the internal modules are accessed externally.

The IMB3 interrupts are controlled by the UIMB. The IMB3 provides 32 interrupt levels. Any interrupt source can be configured to any IMB3 interrupt level. The 32-bit UIPEND register in the UIMB holds the pending IMB3 interrupt requests. IMB3 interrupt request levels zero to six are mapped to USIU interrupt levels zero to six, respectively. IMB3 interrupt request levels seven to 31 are mapped to USIU request level seven. The user must read the UIPEND register to determine the actual source of the interrupt. Refer to **12.4 Interrupt Operation** for more information.

NOTE

If the same interrupt level is assigned to more than one source, software must read the appropriate status bits in the appropriate UIMB3 registers to determine which interrupt was asserted.

Figure 6-5 illustrates the operation of the interrupt controller.

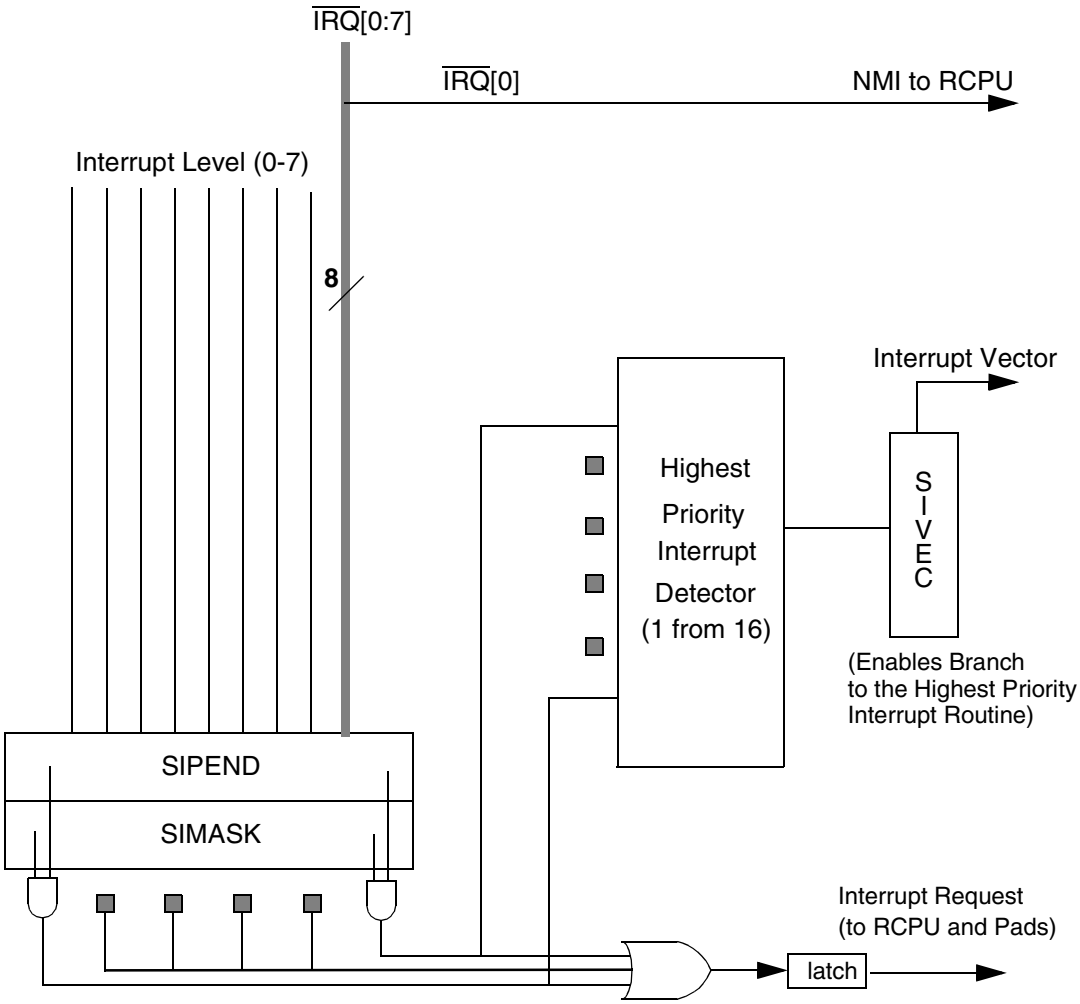


Figure 6-5 MPC555 / MPC556 Interrupt Configuration

6.4.1 SIU Interrupt Sources Priority

The SIU has 15 interrupt sources that assert just one interrupt request to the RCPU. There are eight external $\overline{\text{IRQ}}$ pins ($\overline{\text{IRQ}}$[0] should be masked since it generates a NMI) and eight interrupt levels. The priority between all interrupt sources is shown in **Table 6-3**.

Table 6-3 Priority of Interrupt Sources

Priority	Interrupt Source	Interrupt Code
0 (Highest)	IRQ0	00000000
1	Level 0	00000100
2	IRQ1	00001000
3	Level 1	00001100
4	IRQ2	00010000
5	Level 2	00010100
6	IRQ3	00011000
7	Level 3	00011100
8	IRQ4	00100000
9	Level 4	00100100
10	IRQ5	00101000
11	Level 5	00101100
12	IRQ6	00110000
13	Level 6	00110100
14	IRQ7	00111000
15 (Lowest)	Level 7	00111100
16-31	Reserved	—

6.5 Hardware Bus Monitor

The bus monitor ensures that each bus cycle is terminated within a reasonable period of time. The USIU provides a bus monitor option to monitor internal to external bus accesses on the external bus. The monitor counts from transfer start to transfer acknowledge and from transfer acknowledge to transfer acknowledge within bursts. If the monitor times out, transfer error acknowledge ($\overline{TEA}$) is asserted internally.

The bus monitor timing bit in the system protection control register (SYPCR) defines the bus monitor time-out period. The programmability of the time-out allows for variation in system peripheral response time. The timing mechanism is clocked by the system clock divided by eight. The maximum value is 2040 system clock cycles.

The bus monitor enable (BME) bit in the SYPCR enables or disables the bus monitor. The bus monitor is always enabled, however, when freeze is asserted or when a debug mode request is pending, regardless of the state of this bit.

6.6 MPC555 / MPC556 Decrementer

The decrementer (DEC) is a 32-bit decrementing counter defined by the MPC555 / MPC556 architecture to provide a decrementer interrupt. This binary counter is clocked by the same frequency as the time base (also defined by the MPC555 / MPC556 architecture). The operation of the time base and decrementer are therefore coherent. In the MPC555 / MPC556, the DEC is clocked by the TMBCLK clock. The decrementer period is computed as follows:

$$T_{dec} = \frac{2^{32}}{F_{tmbclk}}$$

The state of the DEC is not affected by any resets and should be initialized by software. The DEC runs continuously after power-up once the time base is enabled by setting the TBE bit of the TBSCR (see **Table 6-16**) (unless the clock module is programmed to turn off the clock). The decrementer continues counting while reset is asserted.

Loading from the decrementer has no effect on the counter value. Storing to the decrementer replaces the value in the decrementer with the value in the GPR.

Whenever bit zero (the MSB) of the decrementer changes from zero to one, a decrementer exception occurs. If software alters the decrementer such that the content of bit 0 is changed to a value of 1, a decrementer exception occurs.

A decrementer exception causes a decrementer interrupt request to be pending in the RCPU. When the decrementer exception is taken, the decrementer interrupt request is automatically cleared.

Table 6-4 illustrates some of the periods available for the decrementer, assuming a 4-MHz or 20-MHz crystal, and TBS = 0 which selects **tbclk** division to FOUR.

NOTE

Time base must be enabled to use the decrementer. See **6.13.4.4 Time Base Control and Status Register** for more information.

Table 6-4 Decrementer Time-Out Periods

Count Value	Time-Out @ 4 MHz	Time-Out @ 20 MHz
0	1.0 µs	0.2 µs
9	10 µs	2.0 µs
99	100 µs	20 µs
999	1.0 ms	200 µs
9999	10.0 ms	2 ms
999999	1.0 s	200 ms
9999999	10.0 s	2.0 s
99999999	100.0 s	20 s
999999999	1000. s	200 s
(hex) FFFFFFFF	4295 s	859 s

Refer to **3.9.5 Decrementer Register (DEC)** for more information.

6.7 MPC555 / MPC556 Time Base (TB)

The time base (TB) is a 64-bit free-running binary counter defined by the MPC555 / MPC556 architecture. The TB has two independent reference registers which can

generate a maskable interrupt when the time base counter reaches the value programmed in one of the two reference registers. The period of the TB depends on the driving frequency. In the MPC555 / MPC556, the TB is clocked by the TMBCLK clock. The period for the TB is:

$$T_{TB} = \frac{2^{64}}{F_{tmbclk}}$$

The state of the time base is not affected by any resets and should be initialized by software. Reads and writes of the TB are restricted to special instructions. Separate special-purpose registers are defined in the MPC555 / MPC556 architecture for reading and writing the time base. For the MPC555 / MPC556 implementation, it is not possible to read or write the entire TB in a single instruction. Therefore, the **mttb** and **mftb** instructions are used to move the lower half of the time base (TBL) while the **mttbu** and **mftbu** instructions are used to move the upper half (TBU).

Two reference registers are associated with the time base: TBREF0 and TBREF1. A maskable interrupt is generated when the TB count reaches to the value programmed in one of the two reference registers. Two status bits in the time base control and status register (TBSCR) indicate which one of the two reference registers generated the interrupt.

Refer to **6.13.4 System Timer Registers** for diagrams and bit descriptions of time base registers. Refer to **3.9.4 Time Base Facility (TB) — OEA** and to ***RCPU Reference Manual* (RCPURM/AD)** for additional information regarding the MPC555 / MPC556 time base.

6.8 Real-Time Clock (RTC)

The RTC is a 32-bit counter and pre-divider used to provide a time-of-day indication to the operating system and application software. It is clocked by the *pitrtclk* clock.The counter is not affected by reset and operates in all low-power modes. It is initialized by software. The RTC can be programmed to generate a maskable interrupt when the time value matches the value programmed in its associated alarm register. It can also be programmed to generate an interrupt once a second. A control and status register is used to enable or disable the different functions and to report the interrupt source.

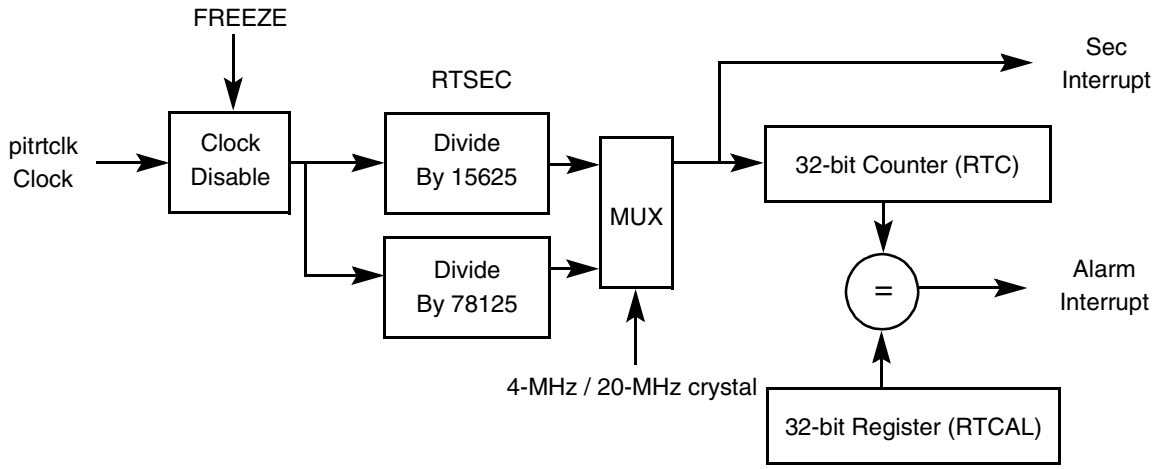


Figure 6-6 RTC Block Diagram

6.9 Periodic Interrupt Timer (PIT)

The periodic interrupt timer consists of a 16-bit counter clocked by the PITRCLK clock supplied by the clock module.

The 16-bit counter counts down to zero when loaded with a value from the PITC. After the timer reaches zero, the PS bit is set and an interrupt is generated if the PIE bit is is a logic one. The software service routine should read the PS bit and then write it to zero to terminate the interrupt request. At the next input clock edge, the value in the PITC is loaded into the counter, and the process starts over again.

When a new value is loaded into the PITC, the periodic timer is updated, the divider is reset, and the counter begins counting. If the PS bit is not cleared, an interrupt request is generated. The request remains pending until PS is cleared. If the PS bit is set again prior to being cleared, the interrupt remains pending until PS is cleared.

Any write to the PITC stops the current countdown, and the count resumes with the new value in PITC. If the PTE bit is not set, the PIT is unable to count and retains the old count value. Reads of the PIT have no effect on the counter value.

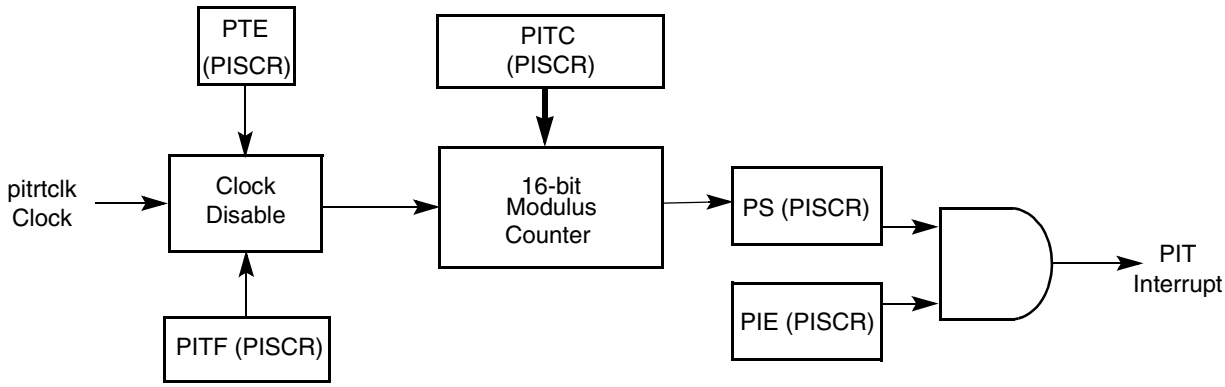


Figure 6-7 PIT Block Diagram

The timeout period is calculated as:

$$PIT_{period} = \frac{PITC + 1}{F_{pitrtclk}} = \frac{PITC + 1}{\left(\frac{ExternalClock}{4 or 256}\right)}$$

Solving this equation using a 4-MHz external clock and a pre-divider of 256 gives:

$$PITperiod = \frac{PITC + 1}{15625}$$

This gives a range from 64 microseconds, with a PITC of 0x0000, to 4.19 seconds, with a PITC of 0xFFFF. When a 20-MHz crystal is used with a pre-divider of 256, the range is between 12.8 microseconds to 0.84 seconds.

6.10 Software Watchdog Timer (SWT)

The software watchdog timer (SWT) prevents system lockout in case the software becomes trapped in loops with no controlled exit. The SWT is enabled after system reset to cause a system reset if it times out. it. The SWT requires a special service sequence to be executed on a periodic basis. If this periodic servicing action does not occur, the SWT times out and issues a reset or a non-maskable interrupt (NMI), depending on the value of the SWRI bit in the SYPCR.

The SWT can be disabled by clearing the SWE bit in the SYPCR. Once the SYPCR is written by software, the state of the SWE bit cannot be changed.

The SWT service sequence consists of the following two steps:

1. Write 0x556C to the Software Service Register (SWSR)
2. Write 0xAA39 to the SWSR

The service sequence clears the watchdog timer and the timing process begins again. If any value other than 0x556C or 0xAA39 is written to the SWSR, the entire sequence must start over.

Although the writes must occur in the correct order prior to time-out, any number of instructions may be executed between the writes. This allows interrupts and exceptions to occur, if necessary, between the two writes.

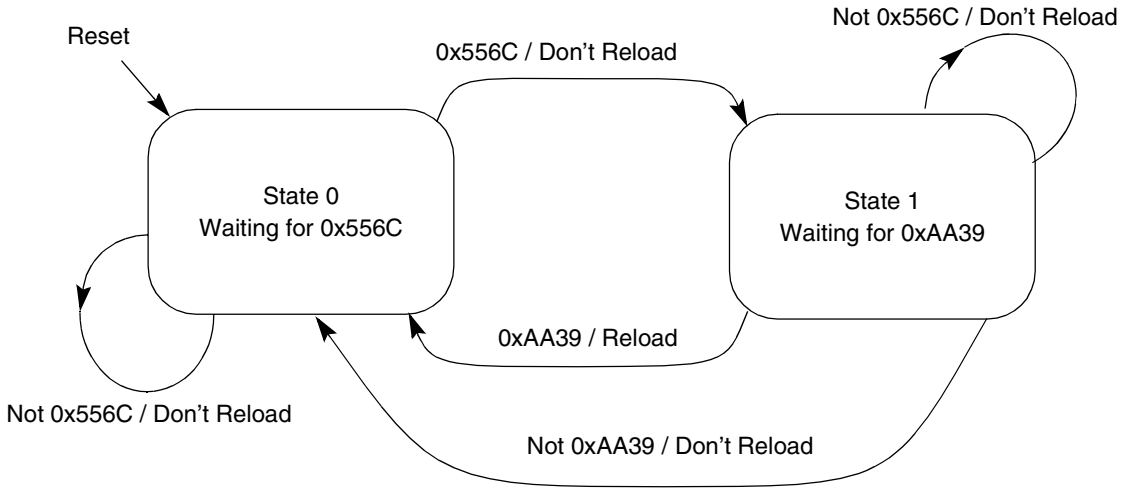


Figure 6-8 SWT Interrupts and Exceptions

Although most software disciplines support the watchdog concept, different systems require different time-out periods. For this reason, the software watchdog provides a selectable range for the time-out period.

In **Figure 6-9**, the range is determined by the value SWTC field. The value held in the SWTC field is then loaded into a 16-bit decrementer clocked by the system clock. An additional divide by 2048 prescaler is used if necessary. The decrementer begins counting when loaded with a value from the software watchdog timing count field (SWTC). After the timer reaches 0x0, a software watchdog expiration request is issued to the reset or NMI control logic.

Upon reset, the value in the SWTC is set to the maximum value and is again loaded into the software watchdog register (SWR), starting the process over. When a new value is loaded into the SWTC, the software watchdog timer is not updated until the servicing sequence is written to the SWSR. If the SWE is loaded with the value zero, the modulus counter does not count.

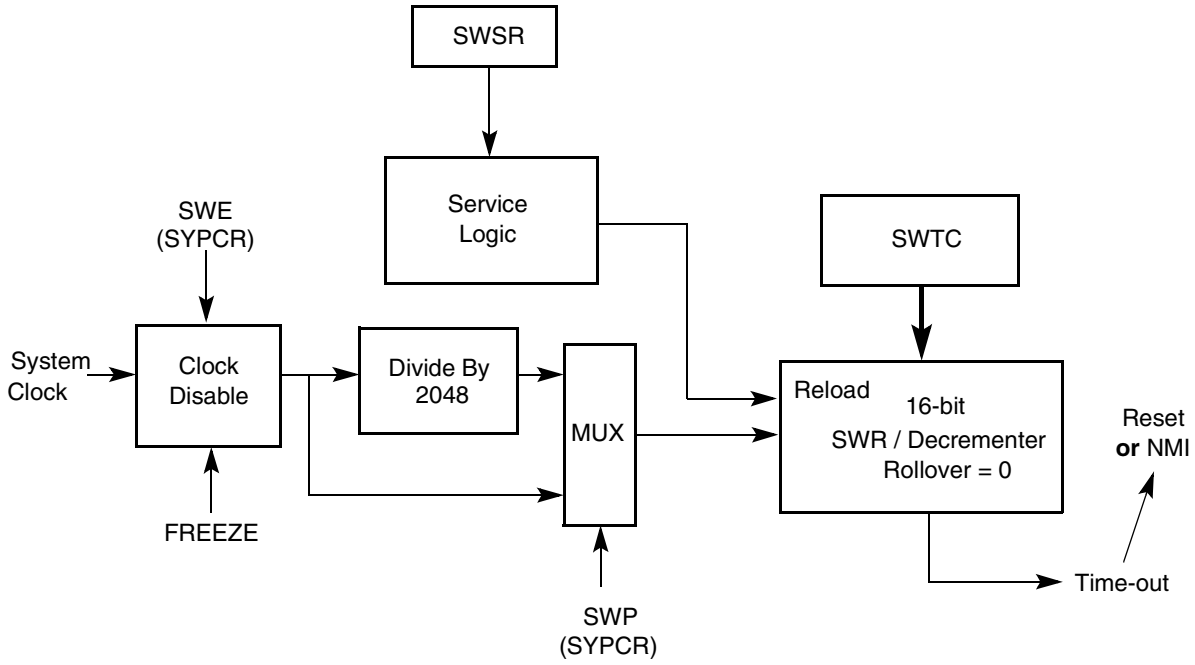


Figure 6-9 SWT Block Diagram

6.11 Freeze Operation

When the FREEZE line is asserted, the clocks to the software watchdog, the periodic interrupt timer, the real-time clock, the time base counter, and the decrementer can be disabled. This is controlled by the associated bits in the control register of each timer. If programmed to stop during FREEZE assertion, the counters maintain their values while FREEZE is asserted, unless changed by the software. The bus monitor, however, remains enabled regardless of this signal.

6.12 Low Power Stop Operation

When the processor is set in a low-power mode (doze, sleep, or deep sleep), the software watchdog timer is frozen. It remains frozen and maintain its count value until the processor exits this state and resumes executing instructions.

The periodic interrupt timer, decrementer, and time base are not affected by these low-power modes. They continue to run at their respective frequencies. These timers are capable of generating an interrupt to bring the MCU out of these low-power modes.

6.13 System Configuration and Protection Registers

This section provides diagrams and bit descriptions of the system configuration and protection registers.

6.13.1 System Configuration Registers

System configuration registers include the SIUMCR, the EMCR, and the IMMR.

6.13.1.1 SIU Module Configuration Register

The SIU module configuration register (SIUMCR) configures various aspects of SIU operation.

SIUMCR — SIU Module Configuration Register **0x2F C000**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	EARP			RESERVED				DSHW	DBGC		DBPC	ATWC	GPC		DLK
RESET:															
ID0*	0	0	0	0	0	0	0	0	ID[9:10]*		ID11*	ID12*	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RE-SERVED	SC		RCTX	MLRC		RESERVED		MTSC	RESERVED						
RESET:															
0	ID[17:18]*		0	0	0	0	0	0	0	0	0	0	0	0	0

* The reset value is a reset configuration word value, extracted from the indicated internal data bus lines.

WARNING

Software must not change any SIUMCR fields controlled by the reset configuration word while the functions that these fields control are active.

Table 6-5 SIUMCR Bit Descriptions

Bit(s)	Name	Description
0	EARB	External arbitration 0 = Internal arbitration is performed 1 = External arbitration is assumed
1:3	EARP	External arbitration request priority. This field defines the priority of an external master's arbitration request. This field is valid when EARB is cleared. Refer to **9.5.6.4 Internal Bus Arbiter** for details.
4:7	—	Reserved
8	DSHW	Data show cycles. This bit selects the show cycle mode to be applied to U-bus data cycles (data cycles to IMB modules and flash EEPROM). This field is locked by the DLK bit. Note that instruction show cycles are programmed in the ICTRL and L-bus data show cycles (to SRAM) are programmed in the L2UMCR. 0 = Disable show cycles for all internal data cycles 1 = Show address and data of all internal data cycles
9:10	DBGC	Debug pins configuration. Refer to **Table 6-6**.
11	DBPC	Debug port pins configuration. Refer to **Table 6-7**.
12	ATWC	Address write type enable configuration. This bit configures the pins to function as byte write enables or address types for debugging purposes. 0 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as $\overline{WE}$[0:3]/$\overline{BE}$[0:3][1] 1 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as AT[0:3]
13:14	GPC	This bit configures the pins as shown in **Table 6-8**.
15	DLK	Debug register lock 0 = Normal operation 1 = SIUMCR is locked and can be written only in test mode or when the internal freeze signal is asserted.
16	—	Reserved
17:18	SC	Single-chip select. This field configures the functionality of the address and data buses. Changing the SC field while external accesses are performed is not supported. Refer to **Table 6-9**.
19	RCTX	Reset configuration/timer expired. During reset the $\overline{RSTCONF}$/TEXP pin functions as $\overline{RSTCONF}$. After reset the pin can be configured to function as TEXP, the timer expired signal that supports the low-power modes. 0 = $\overline{RSTCONF}$/TEXP functions as $\overline{RSTCONF}$ 1 = $\overline{RSTCONF}$/TEXP functions as TEXP
20:21	MLRC	Multi-level reservation control. This field selects between the functionality of the reservation logic and IRQ pins, refer to **Table 6-10**.
22:23	—	Reserved
24	MTSC	Memory transfer start control. 0 = $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ functions according to the MLRC bits setting 1 = $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ functions as $\overline{MTS}$
25:31	—	Reserved

NOTES:
1. $\overline{WE}$/$\overline{BE}$ is selected per memory region by WEBS in the approprite BR register in the memory controller.

Table 6-6 Debug Pins Configuration

DBGC	Pin Function				
	IWP[0:1]/VFLS[0:1]	$\overline{BI}$/$\overline{STS}$	$\overline{BG}$/VF0/LWP1	$\overline{BR}$/VF1/IWP2	$\overline{BB}$/VF2/IWP3
00	VFLS[0:1]	$\overline{BI}$	$\overline{BG}$	$\overline{BR}$	$\overline{BB}$
01	IWP[0:1]	$\overline{STS}$	$\overline{BG}$	$\overline{BR}$	$\overline{BB}$
10	VFLS[0:1]	$\overline{STS}$	VF0	VF1	VF2
11	IWP[0:1]	$\overline{STS}$	LWP1	IWP2	IWP3

Table 6-7 Debug Port Pins Configuration

DBPC	Pin Function		
	TCK/DSCK	TDI/DSDI	TDO/DSDO
0	DSCK	DSDI	DSDO
1	TCK	TDI	TDO

Table 6-8 General Pins Configuration

GPC	Pin Function	
	FRZ/PTR/SGPIOC6	$\overline{IRQOUT}$/LWP0/SGPIOC7
00	PTR	$\overline{LWP0}$
01	SGPIOC6	SGPIOC7
10	FRZ	LWP0
11	FRZ	IRQOUT

Table 6-9 Single-Chip Select Field Pin Configuration

SC	Pin Function		
	DATA[0:15]/ SGPIOD[0:15]	DATA[16:31] SGPIOD[16:31]	ADDR[8:31]/ SGPIOA[8:31]
00 (multiple chip, 32-bit port size)	DATA[0:15]	DATA[16:31]	ADDR[8:31]
01 (multiple chip, 16-bit port size	DATA[0:15]	SPGIOD[16:31]	ADDR[8:31]
10 (single-chip with address show cycles for debugging)	SPGIOD[0:15]	SPGIOD[16:31]	ADDR[8:31]
11 (single-chip)	SPGIOD[0:15]	SPGIOD[16:31]	SPGIOA[8:31]

Table 6-10 Multi-Level Reservation Control Pin Configuration

MLRC	Pin Function					
	$\overline{IRQ}$[0]/ SGPIOC[0]	$\overline{IRQ}$[1]/$\overline{RSV}$/ SGPIOC[1]	$\overline{IRQ}$[2]/$\overline{CR}$/ SGPIOC[2]/ $\overline{MTS}$	$\overline{IRQ}$[3]/$\overline{KR}$/ $\overline{RETRY}$ / SGPIOC[3]	$\overline{IRQ}$[4]/AT[2]/ SGPIOC[4]	$\overline{IRQ}$[5]/SGPI-OC[5]/ MODCK[1][1]
00	$\overline{IRQ}$[0]	$\overline{IRQ}$[1]	$\overline{IRQ}$[2][2]	$\overline{IRQ}$[3]	$\overline{IRQ}$[4]	$\overline{IRQ}$[5] / MODCK[1]
01	$\overline{IRQ}$[0]	$\overline{RSV}$	$\overline{CR}$[2]	$\overline{KR}$/$\overline{RETRY}$	AT[2]	$\overline{IRQ}$[5] / MODCK[1]
10	SGPIOC[0]	SGPIOC[1]	SGPIOC[2][2]	SGPIOC[3]	SGPIOC[4]	SGPIOC[5]/ MODCK[1]
11	$\overline{IRQ}$[0]	$\overline{IRQ}$[1]	SGPIOC[2][2]	$\overline{KR}$/$\overline{RETRY}$	AT[2]	SGPIOC[5]/ MODCK[1]

NOTES:
1. Operates as MODCK[1] during reset.
2. This holds if MTSC bit is reset to 0. Otherwise $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ will function as $\overline{MTS}$.

6.13.1.2 Internal Memory Map Register

The internal memory map register (IMMR) is a special register located within the MPC555 / MPC556 special register space. The IMMR contains identification of a specific device as well as the base for the internal memory map. Based on the value read from this register, software can deduce availability and location of any on-chip system resources.

This register can be read by the **mfspr** instruction. The ISB field can be written by the **mtspr** instruction. The PARTNUM and MASKNUM fields are mask programmed and cannot be changed.

IMMR — Internal Memory Mapping Register **SPR 638**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PARTNUM								MASKNUM							
RESET:															
Read-Only Fixed Value								Read-Only Fixed Value							

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				FLEN	RESERVED		CLES	RESERVED				ISB			0
RESET:															
0	0	0	0	ID20*	0	0	ID23*	0	0	0	0	ID[28:30]*			0

* The reset value is a reset configuration word value extracted from the indicated bits of the internal data bus. Refer to **7.5.2 Hard Reset Configuration Word**.

Table 6-11 IMMR Bit Descriptions

Bit(s)	Name	Description
0:7	PARTNUM	This read-only field is mask programmed with a code corresponding to the part number of the part on which the SIU is located. It is intended to help factory test and user code which is sensitive to part changes. This changes when the part number changes. For example, it would change if any new module is added, if the size of any memory module is changed. It would not change if the part is changed to fix a bug in an existing module. The MPC555 / MPC556 chip has an ID of 0x30.
8:15	MASKNUM	This read-only field is mask programmed with a code corresponding to the mask number of the part. It is intended to help factory test and user code which is sensitive to part changes.
16:19	—	Reserved
20	FLEN	Flash enable is a read-write bit. The default state of FLEN is negated, meaning that the boot is performed from external memory. This bit can be set at reset by the reset configuration word. 0 = On-chip flash memory is disabled, and all internal cycles to the allocated flash address space are mapped to external memory 1 = On-chip flash memory is enabled
21:22	—	Reserved
23	CLES	Core little-endian swap 0 = Little-endian swap logic in the EBI is not activated for RCPU accesses after reset 1 = Little-endian swap logic in the EBI is activated for RCPU accesses after reset
24:27	—	Reserved
28:30	ISB	This read-write field defines the base address of the internal memory space. The initial value of this field can be configured at reset to one of eight addresses, and then can be changed to any value by software. Internal base addresses are as follows: 000 = 0x0000 0000 001 = 0x0040 0000 010 = 0x0080 0000 011 = 0x00C0 0000 100 = 0x0100 0000 101 = 0x0140 0000 110 = 0x0180 0000 111 = 0x01C0 0000
31	—	Reserved

6.13.1.3 External Master Control Register (EMCR)

The external master control register selects the external master modes and determines the internal bus attributes for external-to-internal accesses.

EMCR — External Master Control Register **0x2F C030**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
PRPM	SLVM	0	SIZE		SUPU	INST	RESERVED		RESV	CONT	0	TRAC	SIZEN	RESERVED	
RESET:															
ID16*	0	0	0	1	0	1	0	0	1	1	0	1	1	0	0

* The reset value is a reset configuration word value, extracted from the indicated internal data bus line. Refer to **7.5.2 Hard Reset Configuration Word**.

Table 6-12 EMCR Bit Descriptions

Bit(s)	Name	Description
0:15	—	Reserved
16	PRPM	Peripheral mode. In this mode, the internal RCPU core is shut off and an alternative master on the external bus can access any internal slave module. The reset value of this bit is determined by the reset configuration word bit 16. The bit can also be written by software. 0 = Normal operation 1 = Peripheral mode operation
17	SLVM	Slave mode (valid only if PRPM = 0). In this mode, an alternative master on the external bus can access any internal slave module while the internal RCPU core is fully operational. If PRPM is set, the value of SLVN is a “don’t care.” 0 = Normal operation 1 = Slave mode
18	—	Reserved
19:20	SIZE	Size attribute. If SIZEN = 1, the SIZE bits controls the internal bus attributes as follows: 00 = Double word (8 bytes) 01 = Word (4 bytes) 10 = Half word (2 bytes) 11 = Byte
21	SUPU	Supervisor/user attribute. SUPU controls the supervisor/user attribute as follows: 0 = Supervisor mode access permitted to all registers 1 = User access permitted to registers designated “user access”
22	INST	Instruction attribute. INST controls the internal bus instruction attribute as follows: 0 = Instruction fetch 1 = Operand or non-CPU access
23:24	—	Reserved
25	RESV	Reservation attribute. RESV controls the internal bus reservation attribute as follows: 0 = Storage reservation cycle 1 = Not a reservation
26	CONT	Control attribute. CONT drives the internal bus control bit attribute as follows: 0 = Access to MPC555 / MPC556 control register, or control cycle access 1 = Access to global address map
27	—	Reserved
28	TRAC	Trace attribute. TRAC controls the internal bus program trace attribute as follows: 0 = Program trace 1 = Not program trace
29	SIZEN	External size enable control bit. SIZEN determines how the internal bus size attribute is driven: 0 = Drive size from external bus signals TSIZE[0:1] 1 = Drive size from SIZE0, SIZE1 in EMCR
30:31	—	Reserved

6.13.2 SIU Interrupt Registers

The SIU interrupt controller contains the SIPEND, SIMASK, SIEL, and SIVEC registers.

6.13.2.1 SIPEND Register

Each of the 32 bits in the SIPEND register corresponds to an interrupt request. The bits associated with internal exceptions indicate, if set, that an interrupt service is requested (if not masked by the corresponding bit in the SIMASK register). Each bit reflects the status of the internal requestor device and is cleared when the appropriate actions are initiated by the software in the device itself. Writing to these bits while they are not set has no effect.

The bits associated with the $\overline{IRQ}$ pins have a different behavior depending on the sensitivity defined for them in the SIEL register. When the $\overline{IRQ}$ is defined as a “level” interrupt the corresponding bit behaves similar to the bits associated with internal interrupt sources. When the $\overline{IRQ}$ is defined as an “edge” interrupt and if the corresponding bit is set, it indicates that a falling edge was detected on the line and the bit can be reset by software by writing a 1 to it.

SIPEND — SIU Interrupt Pending Register **0x2F C010**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ0	LVL0	IRQ1	LVL1	IRQ2	LVL2	IRQ3	LVL3	IRQ4	LVL4	IRQ5	LVL5	IRQ6	LVL6	IRQ7	LVL7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.13.2.2 SIU Interrupt Mask Register (SIMASK)

The SIMASK is a 32-bit read/write register. Each bit corresponds to an interrupt request bit in the SIPEND register. Setting a bit in this register allows the interrupt request to reach the RCPU. SIMASK is updated by the software and cleared upon reset. It is the responsibility of the software to determine which of the interrupt sources are enabled at a given time.

SIMASK — SIU Interrupt Mask Register **0x2F C014**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRM0*	LVM0	IRM1	LVM1	IRM2	LVM2	IRM3	LVM3	IRM4	LVM4	IRM5	LVM5	IRM6	LVM6	IRM7	LVM7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*IRQ0 of the SIPEND register is not affected by the setting or clearing of the IRM0 bit of the SIMASK register. IRQ0 is a non-maskable interrupt.

6.13.2.3 SIU Interrupt Edge Level Register (SIEL)

The SIEL is a 32-bit read/write register. Each pair of bits corresponds to an external interrupt request. The EDx bit, if set, specifies that a falling edge in the corresponding $\overline{IRQ}$ line will be detected as an interrupt request. When the EDx bit is 0, a low logical level in the $\overline{IRQ}$ line will be detected as an interrupt request. The WMx (wake-up mask) bit, if set, indicates that an interrupt request detection in the corresponding line causes the MPC555 / MPC556 to exit low-power mode.

SIEL — SIU Interrupt Edge Level Register **0x2F C018**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ED0	WM0	ED1	WM1	ED2	WM2	ED3	WM3	ED4	WM4	ED5	WM5	ED6	WM6	ED7	WM7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.13.2.4 SIU Interrupt Vector Register

The SIVEC is a 32-bit read-only register that contains an 8-bit code representing the unmasked interrupt source of the highest priority level. The SIVEC can be read as either a byte, half word, or word. When read as a byte, a branch table can be used in which each entry contains one instruction (branch). When read as a half-word, each entry can contain a full routine of up to 256 instructions. The interrupt code is defined such that its two least significant bits are 0, thus allowing indexing into the table.

SIVEC — SIU Interrupt Vector **0x2F C01C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INTERRUPT CODE								RESERVED							
RESET:															
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.13.3 System Protection Registers

6.13.3.1 System Protection Control Register (SYPCR)

The system protection control register (SYPCR) controls the system monitors, the software watchdog period, and the bus monitor timing. This register can be read at any time, but can be written only once after system reset.

SYPCR — System Protection Control Register **0x2F C004**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SWTC															
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BMT								BME	RESERVED			SWF	SWE	SWRI	SWP
RESET:															
1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1

Table 6-13 SYPCR Bit Descriptions

Bit(s)	Name	Description
0:15	SWTC	Software watchdog timer count. This field contains the count value of the software watchdog timer.
16:23	BMT	Bus monitor timing. This field specifies the time-out period, in eight-system-clock resolution, of the bus monitor.
24	BME	Bus monitor enable 0 = Disable bus monitor 1 = Enable bus monitor
25:27	—	Reserved
28	SWF	Software watchdog freeze 0 = Software watchdog continues to run while FREEZE is asserted 1 = Software watchdog stops while FREEZE is asserted
29	SWE	Software watchdog enable. Software should clear this bit after a system reset to disable the SWT. 0 = Watchdog is disabled 1 = Watchdog is enabled

Table 6-13 SYPCR Bit Descriptions (Continued)

Bit(s)	Name	Description
30	SWRI	Software watchdog reset/interrupt select 0 = Software watchdog time-out causes a non-maskable interrupt to the RCPU 1 = Software watchdog time-out causes a system reset
31	SWP	Software watchdog prescale 0 = Software watchdog timer is not pre-scaled 1 = Software watchdog timer is prescaled by 2048

6.13.3.2 Software Service Register (SWSR)

The SWSR is the location to which the SWT servicing sequence is written. To prevent SWT time-out, the user should write a 0x556C followed by 0xAA39 to this register. The SWSR can be written at any time but returns all zeros when read.

SWSR — Software Service Register **0x2F C00E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SWSR															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-14 SWSR Bit Descriptions

Bit(s)	Name	Description
0:15	SWSR	SWT servicing sequence is written to this register. To prevent SWT time-out, the user should write a 0x556C followed by 0xAA39 to this register. The SWSR can be written at any time but returns all zeros when read.

6.13.3.3 Transfer Error Status Register (TESR)

The transfer error status register contains a bit for each exception source generated by a transfer error. A bit set to logic 1 indicates what type of transfer error exception occurred since the last time the bits were cleared by reset or by the normal software status bit-clearing mechanism. Note that these bits may be set due to canceled speculative accesses which do not cause an interrupt. The register has two identical sets of bit fields; one is associated with instruction transfers and the other with data transfers.

TESR — Transfer Error Status Register **0x2F C020**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED		IEXT	IBMT	RESERVED						DEXT	DBM	RESERVED			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-15 TESR Bit Descriptions

Bit(s)	Name	Description
0:17	—	Reserved
18	IEXT	Instruction external transfer error acknowledge. This bit is set if the cycle was terminated by an externally generated $\overline{\text{TEA}}$ signal when an instruction fetch was initiated.
19	IBMT	Instruction transfer monitor time out. This bit is set if the cycle was terminated by a bus monitor time-out when an instruction fetch was initiated.
20:25	—	Reserved
26	DEXT	Data external transfer error acknowledge. This bit is set if the cycle was terminated by an externally generated $\overline{\text{TEA}}$ signal when a data load or store is requested by an internal master.
27	DBM	Data transfer monitor time out. This bit is set if the cycle was terminated by a bus monitor time-out when a data load or store is requested by an internal master.
28:31	—	Reserved

6.13.4 System Timer Registers

The following sections describe registers associated with the system timers. These facilities are powered by the KAPWR and can preserve their value when the main power supply is off. Refer to **8.3.3 Pre-Divider** for details on the required actions needed in order to guarantee this data retention.

6.13.4.1 Decrementer Register

The 32-bit decrementer register is defined by the MPC555 / MPC556 architecture. The values stored in this register are used by a down counter to cause decrementer exceptions. The decrementer causes an exception whenever bit zero changes from a logic zero to a logic one. A read of this register always returns the current count value from the down counter.

Contents of this register can be read or written to by the **mfspr** or the **mtspr** instruction. The decrementer register is reset by PORESET. HRESET and SRESET do not affect this register. The decrementer is powered by standby power and can continue to count when standby power is applied.

DEC — Decrementer Register **SPR 22**

MSB 0																															LSB 31
Decrementing Counter																															
POREST																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HRESET/SRESET: UNCHANGED																															

Refer to **3.9.5 Decrementer Register (DEC)** for more information on this register.

6.13.4.2 Time Base SPRs

The TB is a 64-bit register containing a 64-bit integer that is incremented periodically. There is no automatic initialization of the TB; the system software must perform this

initialization. The contents of the register may be written by the **mttbl** or the **mttbu** instructions, see **3.9.4 Time Base Facility (TB) — OEA**.

Refer to **3.8 PowerPC VEA Register Set — Time Base** and **3.9.4 Time Base Facility (TB) — OEA** for more information on reading and writing the TBU and TBL registers.

TB — Time Base (Reading) **SPR 268, 269**

MSB 0	31 32	LSB 63
TBU	TBL	

RESET: UNCHANGED

TB — Time Base (Writing) **SPR 284, 285**

MSB 0	31 32	LSB 63
TBU	TBL	

RESET: UNCHANGED

6.13.4.3 Time Base Reference Registers

Two reference registers (TBREF0 and TBREF1) are associated with the lower part of the time base (TBL). Each is a 32-bit read/write register. Upon a match between the contents of TBL and the reference register, a maskable interrupt is generated.

TBREF0 — Time Base Reference Register 0 **0x2F C204**

MSB 0	LSB 31
TBREF0	

RESET:

TBREF1 — Time Base Reference Register 1 **0x2F C208**

MSB 0	LSB 31
TBREF1	

RESET:

6.13.4.4 Time Base Control and Status Register

The TBSCR is 16-bit read/write register. It controls the TB, decrementer count enable, and interrupt generation and is used for reporting the source of the interrupts. The register can be read anytime. A status bit is cleared by writing a one to it. (Writing a zero has no effect.) More than one bit can be cleared at a time.

TBSCR — Time Base Control and Status Register **0x2F C200**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TBIRQ								REFA	REFB	RESERVED		REFAE	REFBE	TBF	TBE
PORESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-16 TBSCR Bit Descriptions

Bit(s)	Name	Description
0:7	TBIRQ	Time base interrupt request. These bits determine the interrupt priority level of the time base. Refer to **6.4 Interrupt Controller** for interrupt level encodings.
8	REFA	Reference A (TBREF0) interrupt status. 0 = No match detected 1 = TBREF0 value matches value in TBL
9	REFB	Reference B (TBREF1) interupt status. 0 = No match detected 1 = TBREF1 value matches value in TBL
10:11	—	Reserved
12	REFAE	Reference A (TBREF0) interrupt enable. If this bit is set, the time base generates an interrupt when the REFA bit is set.
13	REFBE	Reference B (TBREF1) interrupt enable. If this bit is set, the time base generates an interrupt when the REFB bit is set.
14	TBF	Time base freeze. If this bit is set, the time base and decrementer stop while FREEZE is asserted.
15	TBE	Time base enable 0 = Time base and decrementer are disabled 1 = Time base and decrementer are enabled

6.13.4.5 Real-Time Clock Status and Control Register

The RTCSC is used to enable the different RTC functions and to report the source of the interrupts. The register can be read anytime. A status bit is cleared by writing it to a one. (Writing a zero does not affect a status bit's value.) More than one status bit can be cleared at a time. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep Alive Power Registers Lock Mechanism**.

RTCSC — Real-Time Clock Status and Control Register **0x2F C220**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RTCIRQ								SEC	ALR	Reserved	4M	SIE	ALE	RTF	RTE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	—	0	0	0	—

Table 6-17 RTCSC Bit Descriptions

Bit(s)	Name	Description
0:7	RTCIRQ	Real-time clock interrupt request. Thee bits determine the interrupt priority level of the RTC. Refer to **6.4 Interrupt Controller** for interrupt level encodings.
8	SEC	Once per second interrupt. This status bit is set every second. It should be cleared by the software.
9	ALR	Alarm interrupt. This status bit is set when the value of the RTC equals the value programmed in the alarm register.
10	—	Reserved
11	4M	Real-time clock source 0 = RTC assumes that it is driven by 20 MHz to generate the seconds pulse. 1 = RTC assumes that it is driven by 4 MHz
12	SIE	Second interrupt enable. If this bit is set, the RTC generates an interrupt when the SEC bit is set.
13	ALE	Alarm interrupt enable. If this bit is set, the RTC generates an interrupt when the ALR bit is set.
14	RTF	Real-time clock freeze. If this bit is set, the RTC stops while FREEZE is asserted.
15	RTE	Real-time clock enable 0 = RTC is disabled 1 = RTC is enabled

6.13.4.6 Real-Time Clock Register (RTC)

The real-time clock register is a 32-bit read write register. It contains the current value of the real-time clock. A write to the RTC resets the seconds timer to zero. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep Alive Power Registers Lock Mechanism**.

RTC —Real-Time Clock Register **0x2F C224**

MSB 0		LSB 31
	RTC	

RESET: UNCHANGED

6.13.4.7 Real-Time Clock Alarm Register (RTCAL)

The RTCAL is a 32-bit read/write register. When the value of the RTC is equal to the value programmed in the alarm register, a maskable interrupt is generated.

The alarm interrupt will be generated as soon as there is a match between the ALARM field and the corresponding bits in the RTC. The resolution of the alarm is 1 sec. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep Alive Power Registers Lock Mechanism**.

RTCAL — Real-Time Clock Alarm Register **0x2F C22C**

MSB 0		LSB 31
	ALARM	

RESET: UNCHANGED

6.13.4.8 Periodic Interrupt Status and Control Register (PISCR)

The PISCR contains the interrupt request level and the interrupt status bit. It also contains the controls for the 16 bits to be loaded into a modulus counter. This register can be read or written at any time.

PISCR — Periodic Interrupt Status and Control Register **0x2F C240**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIRQ								PS	RESERVED				PIE	PITF	PTE
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-18 PISCR Bit Descriptions

Bit(s)	Name	Description
0:7	PIRQ	Periodic interrupt request. These bits determine the interrupt priority level of the PIT. Refer to **6.4 Interrupt Controller** for interrupt level encodings.
8	PS	Periodic interrupt status. This bit is set if the PIT issues an interrupt. The PIT issues an interrupt after the modulus counter counts to zero. PS can be negated by writing a one to it. A write of zero has no affect.
9:12	—	Reserved
13	PIE	Periodic interrupt enable. If this bit is set, the time base generates an interrupt when the PS bit is set.
14	PITF	PIT freeze. If this bit is set, the PIT stops while FREEZE is asserted.
15	PTE	Periodic timer enable 0 = PIT stops counting and maintains current value 1 = PIT continues to decrement

6.13.4.9 Periodic Interrupt Timer Count Register (PITC)

The PITC register contains the 16 bits to be loaded in a modulus counter. This register is readable and writable at any time.

PITC — Periodic Interrupt Timer Count **0x2F C244**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PITC															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-19 PITC Bit Descriptions

Bit(s)	Name	Description
0:15	PITC	Periodic interrupt timing count. This field contains the 16-bit value to be loaded into the modulus counter that is loaded into the periodic timer. This register is readable and writeable at any time.
16:31	—	Reserved

6.13.4.10 Periodic Interrupt Timer Register (PITR)

The periodic interrupt register is a read-only register that shows the current value in the periodic interrupt down counter. Read or writing this register does not affect the register.

PITR — Periodic Interrupt Timer Register **0x2F C248**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PIT															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-20 PIT Bit Descriptions

Bit(s)	Name	Description
0:15	PIT	Periodic interrupt timing count — This field contains the current count remaining for the periodic timer. Writes have no effect on this field.
16:31	—	Reserved

6.13.5 General-Purpose I/O Registers

6.13.5.1 SGPIO Data Register 1 (SGPIODT1)

SGPIODT1 — SGPIO Data Register 1 **0x2F C024**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SGPIOD[0:7]								SGPIOD[8:15]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SGPIOD[16:23]								SGPIOD[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-21 SGPIODT1 Bit Descriptions

Bit(s)	Name	Description
0:7	SGPIOD[0:7]	SIU general-purpose I/O Group D[0:7]. This 8-bit register controls the data of general-purpose I/O pins SGPIOD[0:7]. The direction (input or output) of this group of pins is controlled by the GDDR0 bit in the SGPIO control register.
8:15	SGPIOD[8:15]	SIU general-purpose I/O Group D[8:15]. This 8-bit register controls the data of general-purpose I/O pins SGPIOD[8:15]. The direction (input or output) of this group of pins is controlled by the GDDR1 bit in the SGPIO control register.
16:23	SGPIOD[16:23]	SIU general-purpose I/O Group D[16:23]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOD[16:23]. The direction (input or output) of this group of pins is controlled by the GDDR2 bit in the SGPIO control register
24:31	SGPIOD[24:31]	SIU general-purpose I/O Group D[24:31]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOD[24:31]. The direction of SGPIOD[24:31] is controlled by eight dedicated direction control signals SDDRD[24:31]. Each pin in this group can be configured separately as general-purpose input or output.

6.13.5.2 SGPIO Data Register 2 (SGPIODT2)

SGPIODT2 — SGPIO Data Register 2 **0x2F C028**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SGPIOC[0:7]								SGPIOA[8:15]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SGPIOA[16:23]								SGPIOA[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-22 SGPIODT2 Bit Descriptions

Bit(s)	Name	Description
0:7	SGPIOC[0:7]	SIU general-purpose I/O Group C[0:7]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOC[0:7]. The direction of SGPIOC[0:7] is controlled by 8 dedicated direction control signals SDDRC[0:7] in the SGPIO control register. Each pin in this group can be configured separately as general-purpose input or output. **NOTE**: Bit 0 controls SGPIOC0, bit 1 controls SGPIOC1, etc.
8:15	SGPIOA[8:15]	SIU general-purpose I/O Group A[8:15]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[8:15]. The GDDR3 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.
16:23	SGPIOA [16:23]	SIU general-purpose I/O Group A[16:23]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[16:23]. The GDDR4 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.
24:31	SGPIOA [24:31]	SIU general-purpose I/O Group A[24:31]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[24:31]. The GDDR5 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.

6.13.5.3 SGPIO Control Register (SGPIOCR)

SGPIOCR — SGPIO Control Register **0x2F C02C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SDDRC[0:7]								RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
GDDR 0	GDDR 1	GDDR 2	GDDR 3	GDDR 4	GDDR 5	RESERVED		SDDRD[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-23 SGPIOCR Bit Descriptions

Bit(s)	Name	Description
0:7	SDDRC[0:7]	SGPIO data direction for SGPIOC[0:7]. Each SDDR bit zero to seven controls the direction of the corresponding SGPIOC pin zero to seven
8:15	—	Reserved
16	GDDR0	Group data direction for SGPIOD[0:7]
17	GDDR1	Group data direction for SGPIOD[8:15]
18	GDDR2	Group data direction for SGPIOD[16:23]
19	GDDR3	Group data direction for SGPIOA[8:15]
20	GDDR4	Group data direction for SGPIOA[16:23]
21	GDDR5	Group data direction for SGPIOA[24:31]
22:23	—	Reserved
24:31	SDDRD [24:31]	SGPIO data direction for SGPIOD[24:31]. Each SDDRD bit 24:31 controls the direction of the corresponding SGPIOD pin [24:31].

Table 6-24 describes the bit values for data direction control.

Table 6-24 Data Direction Control

SDDR/GDDR	Operation
0	SGPIO configured as input
1	SGPIO configured as output

SECTION 7
RESET

This section describes the MPC555 / MPC556 reset sources, operation, control, and status.

7.1 Reset Operation

The MPC555 / MPC556 has several inputs to the reset logic which include the following:

- Power on reset
- External hard reset pin ($\overline{\text{HRESET}}$)
- External soft reset pin ($\overline{\text{SRESET}}$)
- Loss of lock
- On-chip clock switch
- Software watchdog reset
- Checkstop reset
- Debug port hard reset
- Debug port soft reset
- JTAG reset

All of these reset sources are fed into the reset controller. The control logic determines the cause of the reset, synchronizes it if necessary, and resets the appropriate logic modules, depending on the source of the reset. The memory controller, system protection logic, interrupt controller, and parallel I/O pins are initialized only on hard reset. External soft reset initializes internal logic while maintaining system configuration.

The reset status register (RSR) reflects the most recent source to cause a reset.

7.1.1 Power On Reset

The power-on reset pin, $\overline{\text{PORESET}}$, is an active low input. In a system with power-down low-power mode, this pin should be activated only as a result of a voltage failure in the KAPWR pin. After detecting the assertion of $\overline{\text{PORESET}}$, the MPC555 / MPC556 enters the power-on reset state. During this state the MODCK[1:3] signals determine the oscillator frequency, PLL multiplication factor, and the PITRCLK and TMBCLK clock sources. In addition, the MPC555 / MPC556 asserts the $\overline{\text{SRESET}}$ and $\overline{\text{HRESET}}$ pins.

The $\overline{\text{PORESET}}$ pin should be asserted for a minimum time of 100,000 cycles of clock oscillator after a valid level has been reached on the KAPWR supply. After detecting the assertion of $\overline{\text{PORESET}}$, the MPC555 / MPC556 remains in the power-on reset state until the last of the following two events occurs:

- The Internal PLL enters the lock state and the system clock is active.
- The $\overline{\text{PORESET}}$ pin is negated.

If the MPC555 / MPC556 is in single-chip mode and limp mode is enabled, the internal PLL is not required to be locked before the chip exits power-on reset.

After exiting the power-on reset state, the MCU continues to drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins for 512 system clock cycles. When the timer expires (after 512 cycles), the configuration is sampled from data bus pins, if required (see **7.5.1 Hard Reset Configuration**) and the MCU stops driving the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins. In addition, the internal MODCK[1:3] values are sampled.

The $\overline{\text{PORESET}}$ pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{\text{PORESET}}$ signal asserts only if the $\overline{\text{PORESET}}$ pin asserts for more than 100 ns.

7.1.2 Hard Reset

$\overline{\text{HRESET}}$ (hard reset) is an active low, bi-directional I/O pin. The MPC555 / MPC556 can detect an external assertion of $\overline{\text{HRESET}}$ only if it occurs while the MCU is not asserting reset.

When the MPC555 / MPC556 detects assertion of the external $\overline{\text{HRESET}}$ pin or a cause to assert the internal $\overline{\text{HRESET}}$ line, is detected the chip starts to drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ for 512 cycles. When the timer expires (after 512 cycles) the configuration is sampled from data pins (refer to **7.5.1 Hard Reset Configuration**) and the chip stops driving the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins. An external pull-up resistor should drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins high. After detecting the negation of $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$, the MCU waits 16 clock cycles before testing the presence of an external hard or soft reset.

The $\overline{\text{HRESET}}$ pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{\text{HRESET}}$ will be asserted only if $\overline{\text{HRESET}}$ is asserted for more than 100 ns.

The $\overline{\text{HRESET}}$ is an open collector type pin.

7.1.3 Soft Reset

$\overline{\text{SRESET}}$ (soft reset) is an active low, bi-directional I/O pin. The MPC555 / MPC556 can only detect an external assertion of $\overline{\text{SRESET}}$ if it occurs while the MPC555 / MPC556 is not asserting reset.

When the MPC555 / MPC556 detects the assertion of external $\overline{\text{SRESET}}$ or a cause to assert the internal $\overline{\text{SRESET}}$ line, the chip starts to drive the $\overline{\text{SRESET}}$ for 512 cycles. When the timer expires (after 512 cycles) the debug port configuration is sampled from the DSDI and DSCK pins and the chip stops driving the $\overline{\text{SRESET}}$ pin. An external pull-up resistor should drive the $\overline{\text{SRESET}}$ pin high. After the MPC555 / MPC556 detects the negation of $\overline{\text{SRESET}}$, it waits 16 clock cycles before testing the presence of an external soft reset.

7.1.4 Loss of Lock

If the PLL detects a loss of lock, erroneous external bus operation will occur if synchronous external devices use the MPC555 / MPC556 input clock. Erroneous operation could also occur if devices with a PLL use the MPC555 / MPC556 CLKOUT signal. This source of reset can be optionally asserted if the LOLRE bit in the PLL, low-power, and reset control register (PLPRCR) is set. The enabled PLL loss of lock event generates an internal hard reset sequence. Refer to **SECTION 8 CLOCKS AND POWER CONTROL** for more information on loss of lock.

7.1.5 On-Chip Clock Switch

If the system clocked is switched to the backup clock or switched from backup clock to another clock source an internal hard reset sequence is generated. Refer to **SECTION 8 CLOCKS AND POWER CONTROL**.

7.1.6 Software Watchdog Reset

When the MPC555 / MPC556 software watchdog counts to zero, a software watchdog reset is asserted. The enabled software watchdog event generates an internal hard reset sequence.

7.1.7 Checkstop Reset

When the RCPU enters a checkstop state, and the checkstop reset is enabled (the CSR bit in the PLPRCR is set), a checkstop reset is asserted. The enabled checkstop event generates an internal hard reset sequence. Refer to the ***RCPU Reference Manual* (RCPURM/AD)** for more information.

7.1.8 Debug Port Hard Reset

When the development port receives a hard reset request from the development tool, an internal hard reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. In this case the development tool must reconfigure the debug port. Refer to **SECTION 21 DEVELOPMENT SUPPORT** for more information.

7.1.9 Debug Port Soft Reset

When the development port receives a soft reset request from the development tool, an internal soft reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. In this case the development tool must reconfigure the debug port. Refer to **SECTION 21 DEVELOPMENT SUPPORT** for more information.

7.1.10 JTAG Reset

When the JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. Refer to **SECTION 22 IEEE 1149.1-COMPLIANT INTERFACE (JTAG)** for more information.

7.2 Reset Actions Summary

Table 7-1 summarizes the action taken for each reset.

Table 7-1 Reset Action Taken For Each Reset Cause

Reset Source	Reset Logic and PLL States Reset	System Configuration Reset	Clock Module Reset	$\overline{\text{HRESET}}$ Pin Driven	Debug Port Configuration	Other Internal Logic Reset	$\overline{\text{SRESET}}$ Pin Driven
Power On Reset	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hard Reset Sources External Hard Reset Loss of Lock On-Chip Clock Switch Illegal Low-Power Mode Software Watchdog Checkstop Debug Port Hard Reset	No	Yes	Yes	Yes	Yes	Yes	Yes
Soft Reset Sources External Soft Reset Debug Port Soft Reset JTAG Reset	No	No	No	No	Yes	Yes	Yes

7.3 Data Coherency During Reset

The MPC555 / MPC556 supports data coherency and avoids data corruption while reset. If a cycle is to be executed when detecting any $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$ source, then the cycle will either complete or will not start before generating the corresponding reset control signal. There are reset sources, however, when the MPC555 / MPC556 generates an internal reset due to special internal situation where this protection is not supported. See **7.4 Reset Status Register**.

In the case of large operand size (32 or 16 bits) transaction to a smaller port size, the cycle is split into two 16-bit or four 8-bit cycles. In this case, data coherency is assured and data will not be corrupted.

In the case where the core executes an unaligned load/store cycle which is broken down into multiple cycles, data coherency is NOT assured between these cycles (i.e., data could be corrupted).

A contention on the data pins may occur while asserting external reset ($\overline{\text{EXT_RESET}}$) if the data coherency mechanism is required, and thus enables a cycle to complete, while external hardware drives the data for the configuration word. See **Table 7-2** for a description of the required $\overline{\text{EXT_RESET}}$ line source in a system.

Table 7-2 Reset Configuration Word and Data Corruption/Coherency

Reset Driven	Reset to Use for Data Coherency ($\overline{\text{EXT_RESET}}$)	Comments
$\overline{\text{HRESET}}$	$\overline{\text{SRESET}}$	
$\overline{\text{SRESET}}$	$\overline{\text{HRESET}}$	
$\overline{\text{HRESET}}$ & $\overline{\text{SRESET}}$	$\overline{\text{HRESET}}$ || $\overline{\text{SRESET}}$	Provided only one of them is driven into the MPC555 / MPC556 at a time

7.4 Reset Status Register

All of the reset sources are fed into the reset controller. The 16-bit reset status register (RSR) reflects the most recent source, or sources, of reset. (Simultaneous reset requests can cause more than one bit to be set at the same time.) This register contains one bit for each reset source. A bit set to logic one indicates the type of reset that occurred.

Once set, individual bits in the RSR remain set until software clears them. Can be cleared by writing a one to the bit. A write of zero has no effect on the bit. The register can be read at all times. The reset status register receives its default reset values during power-on reset. The RSR is powered by the KAPWR pin.

RSR — Reset Status Register **0x2F C288**

MSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EHRS	ESRS	LLRS	SWRS	CSRS	DB-HRS	DB-SRS	JTRS	OCCS	ILBC	GPOR	GH-RST	GSRST	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Table 7-3 Reset Status Register Bit Descriptions

Bit(s)	Name	Description
0	EHRS[1]	External hard reset status 0 = No external hard reset has occurred 1 = An external hard reset has occurred
1	ESRS[1]	External soft reset status 0 = No external soft reset has occurred 1 = An external soft reset has occurred
2	LLRS	Loss of lock reset status 0 = No enabled loss-of-lock reset has occurred 1 = An enabled loss-of-lock reset has occurred
3	SWRS	Software watchdog reset status 0 = No software watchdog reset has occurred 1 = A software watchdog reset has occurred
4	CSRS	Checkstop reset status 0 = No enabled checkstop reset has occurred 1 = An enabled checkstop reset has occurred
5	DBHRS	Debug port hard reset status 0 = No debug port hard reset request has occurred 1 = A debug port hard reset request has occurred
6	DBSRS	Debug port soft reset status 0 = No debug port soft reset request has occurred 1 = A debug port soft reset request has occurred
7	JTRS	JTAG reset status 0 = No JTAG reset has occurred 1 = A JTAG reset has occurred
8	OCCS	On-chip clock switch 0 = No on-chip clock switch reset has occurred 1 = An on-chip clock switch reset has occurred

Table 7-3 Reset Status Register Bit Descriptions (Continued)

Bit(s)	Name	Description
9	ILBC	Illegal bit change. This bit is set when the MPC555 / MPC556 changes any of the following bits when they are locked: LPM[0:1], locked by the LPML bit MF[0:11], locked by the MFPDL bit DIVF[0:4], locked by the MFPDL bit
10	GPOR	Glitch detected on $\overline{\text{PORESET}}$ pin. This bit is set when the $\overline{\text{PORESET}}$ pin is asserted for more than TBD ns 0 = No glitch was detected on the $\overline{\text{PORESET}}$ pin 1 = A glitch was detected on the $\overline{\text{PORESET}}$ pin
11	GHRST	Glitch detected on $\overline{\text{HRESET}}$ pin. This bit is set when the $\overline{\text{HRESET}}$ pin is asserted for more than TBD ns 0 = No glitch was detected on the $\overline{\text{HRESET}}$ pin 1 = A glitch was detected on the $\overline{\text{HRESET}}$ pin
12	GSRST	Glitch detected on $\overline{\text{SRESET}}$ pin. If the $\overline{\text{SRESET}}$ pin is asserted for more than TBD ns the GHRST bit will be set. If an internal or external SRESET is generated the $\overline{\text{SRESET}}$ pin is asserted and the GSRST bit will be set. The GSRST bit remains set until software clears it. The GSRST bit can be negated by writing a one to GSRST. A write of zero has no effect on this bit. 0 = No glitch was detected on $\overline{\text{SRESET}}$ pin 1 = A glitch was detected on $\overline{\text{SRESET}}$ pin.
13:15	—	Reserved

NOTES:

1. In the USIU RSR, if both EHRS and ESRS are set, the reset source is internal. The EHRS and ESRS bits in RSR register are set for any internal reset source in addition to external HRESET and external SRESET events. If both internal and external indicator bits are set, then the reset source is internal.

7.5 Reset Configuration

7.5.1 Hard Reset Configuration

When a hard reset event occurs, the MPC555 / MPC556 reconfigures its hardware system as well as the development port configuration The logical value of the bits that determine its initial mode of operation, are sampled from the following:

- The external data bus pins DATA[0:31]
- An internal default constant (0x0000 0000)
- An internal NVM register value (CMFCFIG)

If at the sampling time (at $\overline{\text{HRESET}}$ negation) $\overline{\text{RSTCONF}}$ is asserted, then the configuration is sampled from the data bus. If $\overline{\text{RSTCONF}}$ is negated and a valid NVM value exists (CMFCFIG bit $\overline{\text{HC}}$=0), then the configuration is sampled from the NVM register in the CMF module. If $\overline{\text{RSTCONF}}$ is negated and no valid NVM value exists (CMFCFIG bit $\overline{\text{HC}}$=1), then the configuration word is sampled from the internal default. $\overline{\text{HC}}$ will be “1” if the internal flash is erased. **Table 7-4** summarizes the reset configuration options.

NOTE

If the CMFCFIG reset config word is being used, then the flash is automatically enabled.

Table 7-4 Reset Configuration Options

$\overline{\text{RSTCONF}}$	Has Configuration (HC)	Internal Configuration Word
0	x	DATA[0:31] pins
1	0	NVM flash EEPROM register (CMFCFIG)
1	1	Internal data word default (0x0000 0000)

If the PRDS control bit in the PDMCR register is set and $\overline{\text{HRESET}}$ and $\overline{\text{RSTCONF}}$ are asserted, the MPC555 / MPC556 pulls the data bus low with a weak resistor. The user can overwrite this default by driving the appropriate bit high. See **Figure 7-1** for the basic reset configuration scheme.

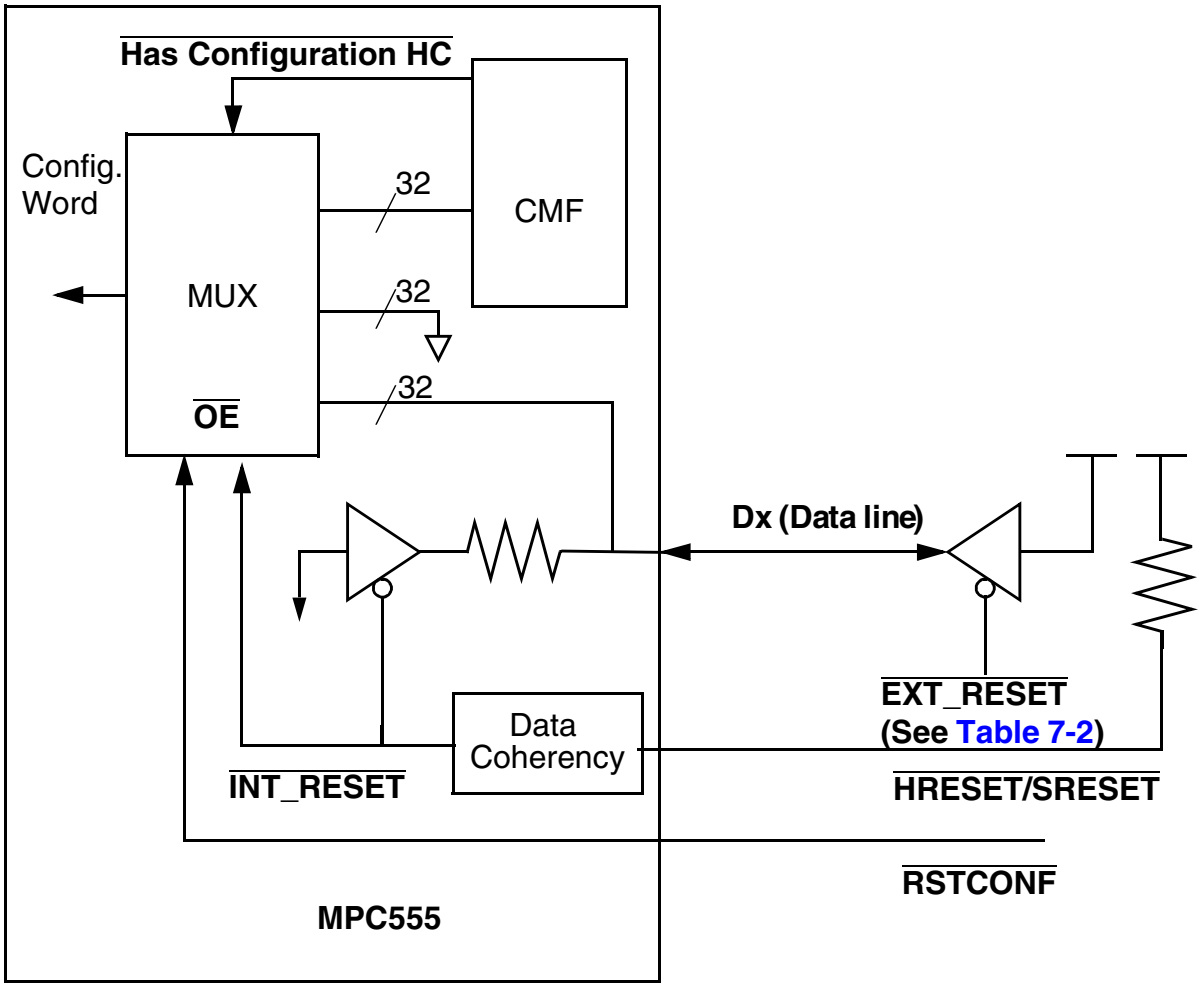


Figure 7-1 Reset Configuration Basic Scheme

During the assertion of the $\overline{\text{PORESET}}$ input signal, the chip assumes the default reset configuration. This assumed configuration changes if the input signal $\overline{\text{RSTCONF}}$ is asserted when the $\overline{\text{PORESET}}$ is negated or the CLKOUT starts to oscillate. To ensure

that stable data is sampled, the hardware configuration is sampled every eight clock cycles on the rising edge of CLKOUT with a double buffer. The setup time required for the data bus is approximately 15 cycles, and the maximum rise time of $\overline{\text{HRESET}}$ should be less than 6 clock cycles. In systems where an external reset configuration word and the TEXP output function are both required, $\overline{\text{RSTCONF}}$ should be asserted until $\overline{\text{SRESET}}$ is negated.

Figure 7-2 to **Figure 7-5** provide sample reset configuration timings.

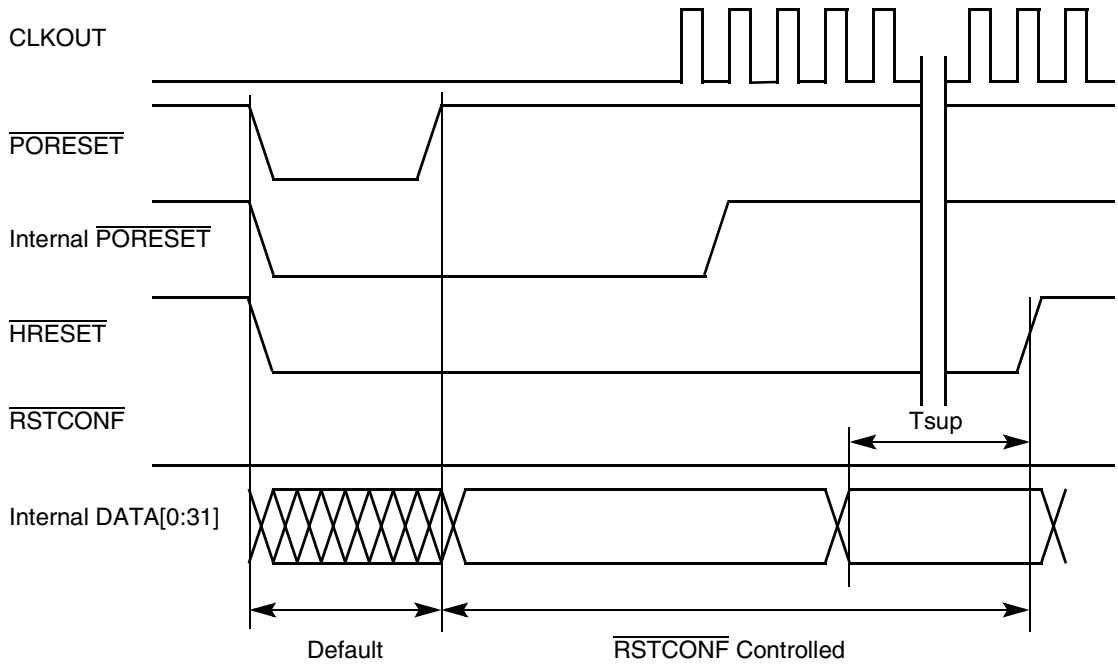


Figure 7-2 Reset Configuration Sampling Scheme For "Short" $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled

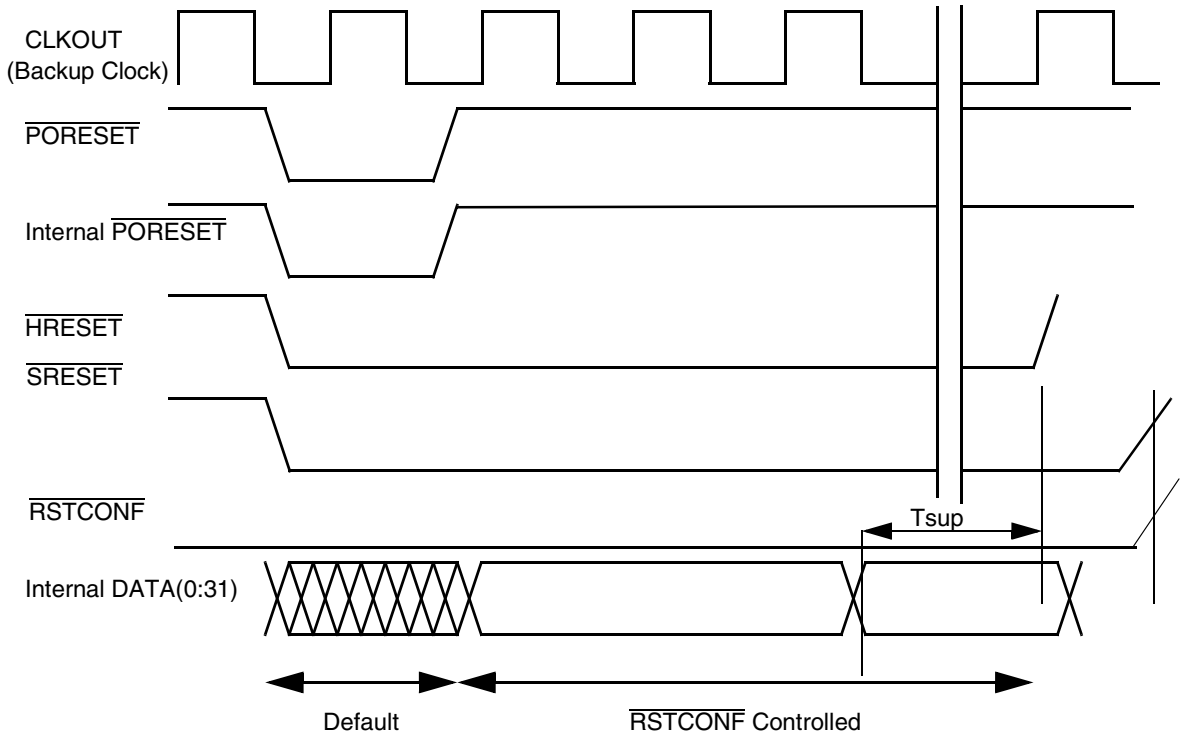


Figure 7-3 Reset Configuration Timing for "Short" POR̄ESET Assertion, Limp Mode Enabled

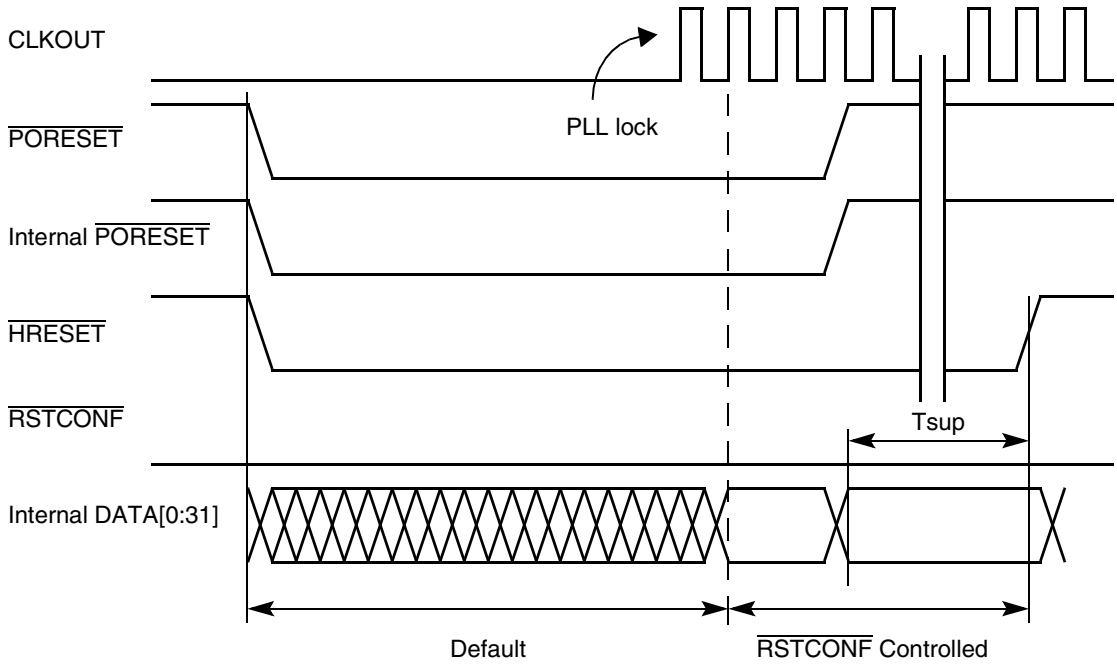


Figure 7-4 Reset Configuration Timing for

“Long” $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled

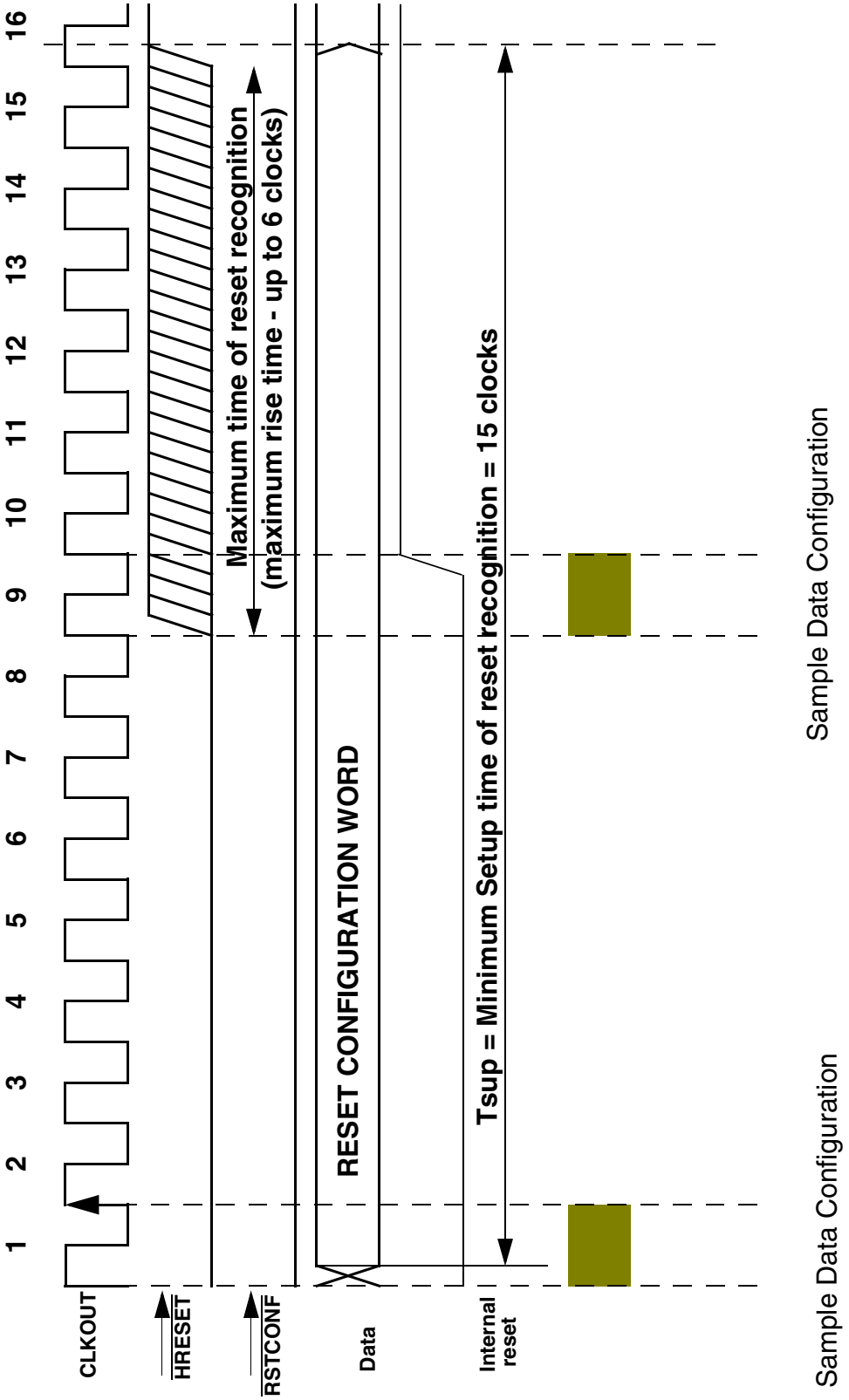


Figure 7-5 Reset Configuration Sampling Timing Requirements

7.5.2 Hard Reset Configuration Word

The hard reset configuration word, which is sampled from the internal data bus on the negation of $\overline{\text{HRESET}}$, is shown below. The reset configuration word is not a register in the memory map. Most of the bits in the configuration are located in registers in the USIU. The user should refer to the appropriate register definition for a detailed description of each control bit.

Hard Reset Configuration Word

MSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	IP	BDRV	BDIS	BPS		Reserved			DBGC		DBPC	ATWC	EBDF		Re-served
DEFAULT:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRPM	SC		ETRE	FLEN	EN_ COMP	EXC_ COMP	RE-SERVED	Reserved				ISB			DME
DEFAULT:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-5 Hard Reset Configuration Word Bit Descriptions

Bit(s)	Name	Description
0	EARB	External arbitration. Refer to **6.13.1.1 SIU Module Configuration Register** for a detailed bit definition. 0 = Internal arbitration is performed 1 = External arbitration is assumed
1	$\overline{\text{IP}}$	Initial interrupt prefix. This bit defines the initial value of the MSR[IP] bit immediately after reset. MSR[IP] defines the interrupt table location. 0 = MSR[IP] = 0 after reset 1 = MSR[IP] = 1 after reset
2	BDRV	Bus pins drive strength. This bit determines the driving capability of the bus pins (address, data, and control) and the CLKOUT pin. For details, refer to description of the COM bits in **8.12.1 System Clock Control Register (SCCR)**. The default value is full drive strength for the bus pins and CLKOUT. 0 = Full drive 1 = Reduced drive
3	BDIS	External boot disable. If a write to the OR0 register occurs after reset, this bit definition is ignored. 0 = Memory controller bank 0 is active and matches all addresses immediately after reset 1 = Memory controller is not activated after reset.
4:5	BPS	Boot port size. If a write to the OR0 register occurs after reset, this field definition is ignored. 00 = 32-bit port (default) 01 = 8-bit port 10 = 16-bit port 11 = Reserved
6:8	—	Reserved
9:10	DBGC	Debug pins configuration. See **6.13.1.1 SIU Module Configuration Register** for this field definition. The default value is for these pins to function as VFLS[0:1], $\overline{\text{BI}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{BB}}$.

Table 7-5 Hard Reset Configuration Word Bit Descriptions (Continued)

Bit(s)	Name	Description
11	DBPC	Debug port pins configuration. See **6.13.1.1 SIU Module Configuration Register** for this field definition. The default value is for these pins to function as development support pins.
12	ATWC	Address type write-enable configuration. Refer to **6.13.1.1 SIU Module Configuration Register** for this field definition. The default value is for these pins to function as write-enable pins.
13:14	EBDF	External bus division factor. This field defines the initial value of the external bus frequency. Refer to **8.12.1 System Clock Control Register (SCCR)** for details. The default value is that CLKOUT frequency is equal to that of the internal clock (divide by one).
15	—	Reserved
16	PRPM	Peripheral mode enable. This bit determines whether the chip is in peripheral mode. Refer to **6.13.1.3 External Master Control Register (EMCR)** for details. The default value is that peripheral mode is not enabled.
17:18	SC	Single chip select. Refer to **6.13.1.1 SIU Module Configuration Register** for details. 00 = Extended chip, 32 bits data 01 = Extended chip, 16 bits data 10 = Single chip and show cycles (address) 11 = Single chip
19	ETRE	Exception table relocation enable. This field defines whether the exception table relocation feature in the BBC is enabled or disabled. The default state is disabled. Refer to **SECTION 4 BURST BUFFER** for details.
20	FLEN	Flash Enable — This field determines whether the on-chip flash memory is enabled or disabled out of reset. The default state is disabled, which means that by default, the boot is from external memory. 0 = Flash disabled — boot is from external memory 1 = Flash enabled
21	EN_ COMP[1]	Enable Compression — This bit enables the operation of the MPC555 / MPC556 with compressed code. The default state is disabled. See **Table 4-8**.
22	EXC_ COMP[1]	Exception Compression — This bit determines the operation of the MPC555 with exceptions. If this bit is set, than the MPC555 assumes that ALL the exception routines are in compressed code. The default indicates the exceptions are all non-compressed. See **Table 4-8**.
23	—	This bit should not be high in the reset configuration word.
24:27	—	Reserved
28:30	ISB	Initial internal space base select. This field defines the initial value of the ISB field in the IMMR register. Refer to **6.13.1.2 Internal Memory Map Register** for details. The default state is that the internal memory map is mapped to start at address 0x0000 0000.
31	DME	Dual mapping enable. This bit determines whether dual mapping of the flash EEPROM module is enabled. Refer to **10.8.5 Dual Mapping Base Register (DMBR)** for details.The default value is for dual mapping to be disabled. 0 = Dual mapping disabled 1 = Dual mapping enabled

NOTES:
1. This bit is available only on the MPC555 / MPC556.

7.5.3 Soft Reset Configuration

When a soft reset event occurs, the MPC555 / MPC556 reconfigures the development port. Refer to **SECTION 21 DEVELOPMENT SUPPORT** for details.

SECTION 8
CLOCKS AND POWER CONTROL

8.1 Overview

The main timing reference for the MPC555 / MPC556 can monitor any of the following:

- A crystal with a frequency of four MHz or 20 MHz
- An external frequency source with a frequency of 4 MHz
- An external frequency source at the system frequency

The system operating frequency is generated through a programmable phase-locked loop, the system PLL (SPLL). The SPLL is programmable in integer multiples of the input oscillator frequency to generate the internal (VCO/2) operating frequency. A pre-divider before the SPLL enables the user to divide the high frequency crystal oscillator. The SPLL VCO is twice the system frequency. The internal operating SPLL frequency should be at least 30 MHz. It can be divided by a power-of-two divider to generate the system operating frequencies.

In addition to the system clock, the clocks submodule provides the following:

- TMBCLK to the time base (TB) and decrementer (DEC)
- PITRTCLK to the periodic interrupt timer (PIT) and real-time clock (RTC)

The oscillator, TB, DEC, RTC, and the PIT are powered from the keep alive power supply (KAPWR) pin. This allows the counters to continue to count (increment/decrement) at the oscillator frequency even when the main power to the MCU is off. While the power is off, the PIT may be used to signal to the power supply IC to enable power to the system at specific intervals. This is the power-down wake-up feature. When the chip is not in power-down low-power mode, the KAPWR is powered to the same voltage value as the voltage of the I/O buffers and logic.

The MPC555 / MPC556 clock module consists of the main crystal oscillator (OSCM), the SPLL, the low-power divider, the clock generator, the system low-power control block, and the limp mode control block. The clock module receives control bits from the system clock control register (SCCR), change of lock interrupt register (COLIR), the low-power and reset-control register (PLPRCR), and the PLL.

All of the MPC555 peripherals on the IMB bus derive its clock timing from the UIMB module. The UIMB runs on the main system clock, but can divide the system frequency in half. See **12.3 Clock Module**.

Figure 8-1 illustrates the functional block diagram of the clock unit.

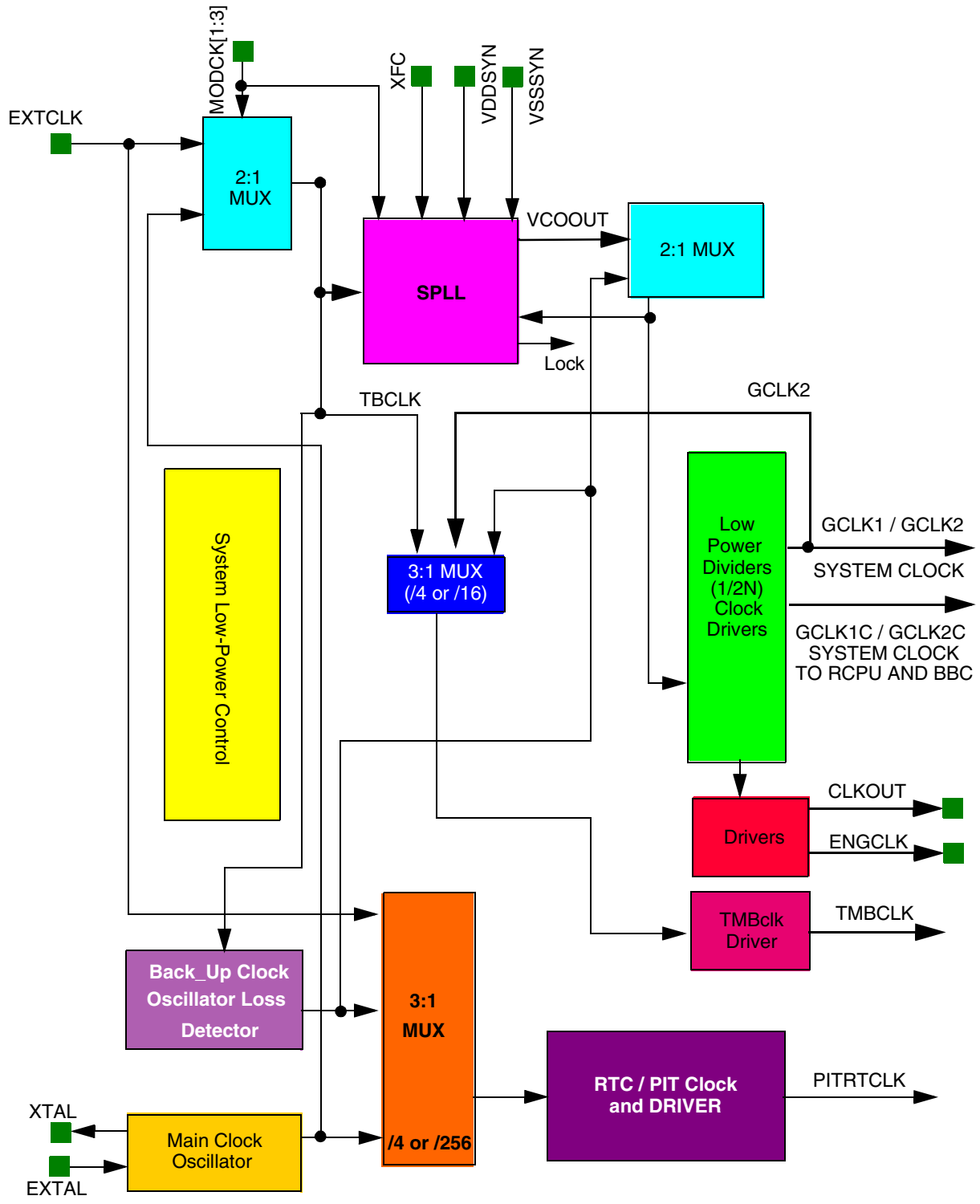


Figure 8-1 Clock Unit Block Diagram

8.2 System Clock Sources

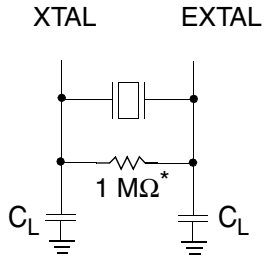
The system clock can be provided by the main system oscillator (OSCM), an external clock input, or the backup clock (BUCLK) on-chip ring oscillator, see **Figure 8-2**.

The OSCM uses either a 4-MHz or 20-MHz crystal to generate the PLL reference clock. When the main system oscillator output is the timing reference to the system PLL, skew elimination between the XTAL/EXTAL pins and CLKOUT is not guaranteed.

The external clock input receives a clock signal from an external source. The clock frequency must be either in the range of 3 MHz – 5 MHz or at the system frequency of at least 15 MHz (1:1 mode). When the external clock input is the timing reference to the system PLL skew elimination between the EXTCLK pin and the CLKOUT is less than ± 1 ns.

The backup clock on-chip ring oscillator enables the MCU to function with a less precise clock. When operating from the backup clock, the MCU is in **limp mode**. This enables the system to continue minimum functionality until the system is fixed. The BUCLK frequency is approximately 7 MHz (see **APPENDIX G ELECTRICAL CHARACTERISTICS** for the complete frequency range).

For normal operation, at least one clock source (EXTCLK or OSCM) must be active. A configuration with both clock sources active is possible as well. At this configuration EXTCLK provides the OSCCLK and OSCM provides the PITRTCLK. The input of an unused timing reference (EXTCLK or EXTAL) must be grounded.



*Resistor is not currently required on the board but space should be available for its addition in the future.

Figure 8-2 Main System Oscillator (OSCM)

8.3 System PLL

The PLL allows the processor to operate at a high internal clock frequency using a low frequency clock input, a feature which offers two benefits. Lower frequency clock input reduces the overall electromagnetic interference generated by the system, and the ability to oscillate at different frequencies reduces cost by eliminating the need to add an additional oscillator to a system.

The PLL can perform the following functions:

- Frequency multiplication

- Skew elimination
- Frequency division

8.3.1 Frequency Multiplication

The PLL can multiply the input frequency by any integer between one and 4096. The multiplication factor depends on the value of the MF[0:11] bits in the PLPRCR register. While any integer value from one to 4096 can be programmed, the resulting VCO output frequency must be at least 15 MHz. The multiplication factor is set to a predetermined value during power-on reset as defined in **Table 8-1**.

8.3.2 Skew Elimination

The PLL is capable of eliminating the skew between the external clock entering the chip (EXTCLK) and both the internal clock phases and the CLKOUT pin, making it useful for tight synchronous timings. Skew elimination is active only when the PLL is enabled and programmed with a multiplication factor of one or two (MF = 0 or 1). The timing reference to the system PLL is the external clock input.

8.3.3 Pre-Divider

A pre-divider before the phase comparator enables additional system clock resolution when the crystal oscillator frequency is 20 MHz. The division factor is determined by the DIVF[0:4] bits in the PLPRCR.

8.3.4 PLL Block Diagram

As shown in **Figure 8-3**, the reference signal, OSCCLK, goes to the phase comparator. The phase comparator controls the direction (up or down) that the charge pump drives the voltage across the external filter capacitor (XFC). The direction depends on whether the feedback signal phase lags or leads the reference signal. The output of the charge pump drives the VCO. The output frequency of the VCO is divided down and fed back to the phase comparator for comparison with the reference signal, OSCCLK. The MF values, zero to 4095, are mapped to multiplication factors of one to 4096. Note that when the PLL is operating in 1:1 mode (refer to **Table 8-1**), the multiplication factor is one (MF = 0). The PLL output frequency is twice the maximum system frequency. This double frequency is needed to generate GCLK1 and GCLK2 clocks. On power-up, with a four MHz or 20 MHz crystal and the default MF settings, System Frequency ($FREQ_{SYS}$) will be 40 MHz and the system clock will be 20 MHz. The equation for system frequency ($FREQ_{SYS}$) is shown below:

$$\text{System Frequency } (FREQ_{SYS}) = \frac{OSCCLK}{DIVF + 1} \times (MF + 1) \times 2 / 2$$

NOTE

When operating with the backup clock, the system clock (and CLKOUT) is one-half of the ring oscillator frequency. (i.e., the system clock is a nominal seven MHz). The time base and PIT clocks will be twice the system clock frequency.

The PLL maximum lock time is determined by the input clock to the phase detector. The PLL locks within 500 input clock cycles.

NOTE

Upon initial system power up and after KAPWR is lost, an external circuit must assert power on reset ($\overline{\text{PORESET}}$). If limp mode will be enabled during power-on reset, $\overline{\text{PORESET}}$ must be asserted for at least 100,000 cycles of input PLL clock after a valid level has been reached on the KAPWR supply. If limp mode will be disabled, $\overline{\text{PORESET}}$ should be asserted for approximately 3 µs after a valid level has been reached on the KAPWR supply.

Whenever power-on reset is asserted, the MF bits are set according to **Table 8-1**, and the DFNH and DFNL bits in SCCR are set to the value of 0 (÷1 and 2), respectively.

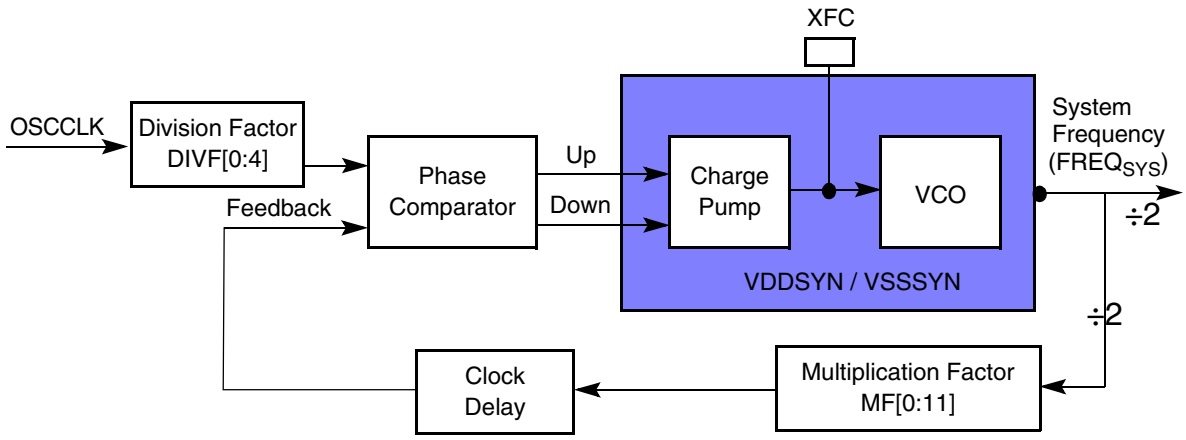


Figure 8-3 System PLL Block Diagram

8.3.5 PLL Pins

The following pins are dedicated to the PLL operation:

- **VDDSYN** — Drain voltage. This is the V_{DD} dedicated to the analog PLL circuits. The voltage should be well-regulated and the pin should be provided with an extremely low impedance path to the V_{DD} power rail. VDDSYN should be bypassed to VSSSYN by a 0.1 µF capacitor located as close as possible to the chip package.
- **VSSSYN** — Source voltage. This is the V_{SS} dedicated to the analog PLL circuits. The pin should be provided with an extremely low impedance path to ground. VSSSYN should be bypassed to VDDSYN by a 0.1 µF capacitor located as close as possible to the chip package.
- **XFC** — External filter capacitor. XFC connects to the off-chip capacitor for the PLL filter. One terminal of the capacitor is connected to XFC, and the other terminal is connected to VDDSYN.
  The off-chip capacitor must have the following values:

$0 < MF + 1 < 4$	$(680 \times (MF + 1) - 120)$ pF
$MF + 1 \geq 4$	$1100 \times (MF + 1)$ pF

Where MF = the value stored on MF[0:11]. This is one less than the desired frequency multiplication factor.

8.4 System Clock During PLL Loss of Lock

At reset, until the SPLL is locked, the SPLL output clock is disabled.

During normal operation (once the PLL has locked), either the oscillator or an external clock source is generating the system clock. In this case, if loss of lock is detected and the LOLRE (loss of lock reset enable) bit in the PLPRCR is cleared, the system clock source continues to function as the PLL's output clock. The USIU timers can operate with the input clock to the PLL, so that these timers are not affected by the PLL loss of lock. Software can use these timers to measure the loss-of-lock period. If the timer reaches the user-preset software criterion, the MCU can switch to the backup clock by setting the switch to backup clock (STBUC) bit in the SCCR, provided the limp mode enable (LME) bit in the SCCR is set.

If loss of lock is detected during normal operation, assertion of $\overline{\text{HRESET}}$ (for example, if LOLRE is set) disables the PLL output clock until the lock condition is met. During hard reset, the STBUC bit is set as long as the PLL lock condition is not met and clears when the PLL is locked. If STBUC and LME are both set, the system clock switches to the backup clock, and the chip operates in limp mode until STBUC is cleared.

Every change in the lock status of the PLL can generate a maskable interrupt.

NOTE

When the VCO is the system clock source, chip operation is unpredictable while the PLL is unlocked. Note further that a switch to the backup clock is possible only if the LME bit in the SCCR is set.

8.5 Low-Power Divider

The output of the PLL is sent to a low-power divider block. (In limp mode the BUCLK is sent to a low-power divider block.) This block generates all other clocks in normal operation, but has the ability to divide the output frequency of the VCO before it generates the general system clocks sent to the rest of the MPC555 / MPC556. The PLL System Frequency ($FREQ_{SYS}$) is always divided by at least 2.

The purpose of the low-power divider block is to allow the user to reduce and restore the operating frequencies of different sections of the MPC555 / MPC556 without losing the PLL lock. Using the low-power divider block, the user can still obtain full chip operation, but at a lower frequency. This is called gear mode. The selection and speed of gear mode can be changed at any time, with changes occurring immediately.

The low-power divider block is controlled in the system clock control register (SCCR). The default state of the low-power divider is to divide all clocks by one. Thus, for a 40-MHz system, the general system clocks are each 40 MHz.

8.6 MPC555 / MPC556 Internal Clock Signals

The internal clocks generated by the clocks module are shown in **Figure 8-4**. The clocks module also generates the CLKOUT and ENGCLK external clock signals. The PLL synchronizes these signals to each other. The PITRTCLK frequency and source are specified by the RTDIV and RTSEL bits in the SCCR. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source and is twice the MPC555 / MPC556 system clock.

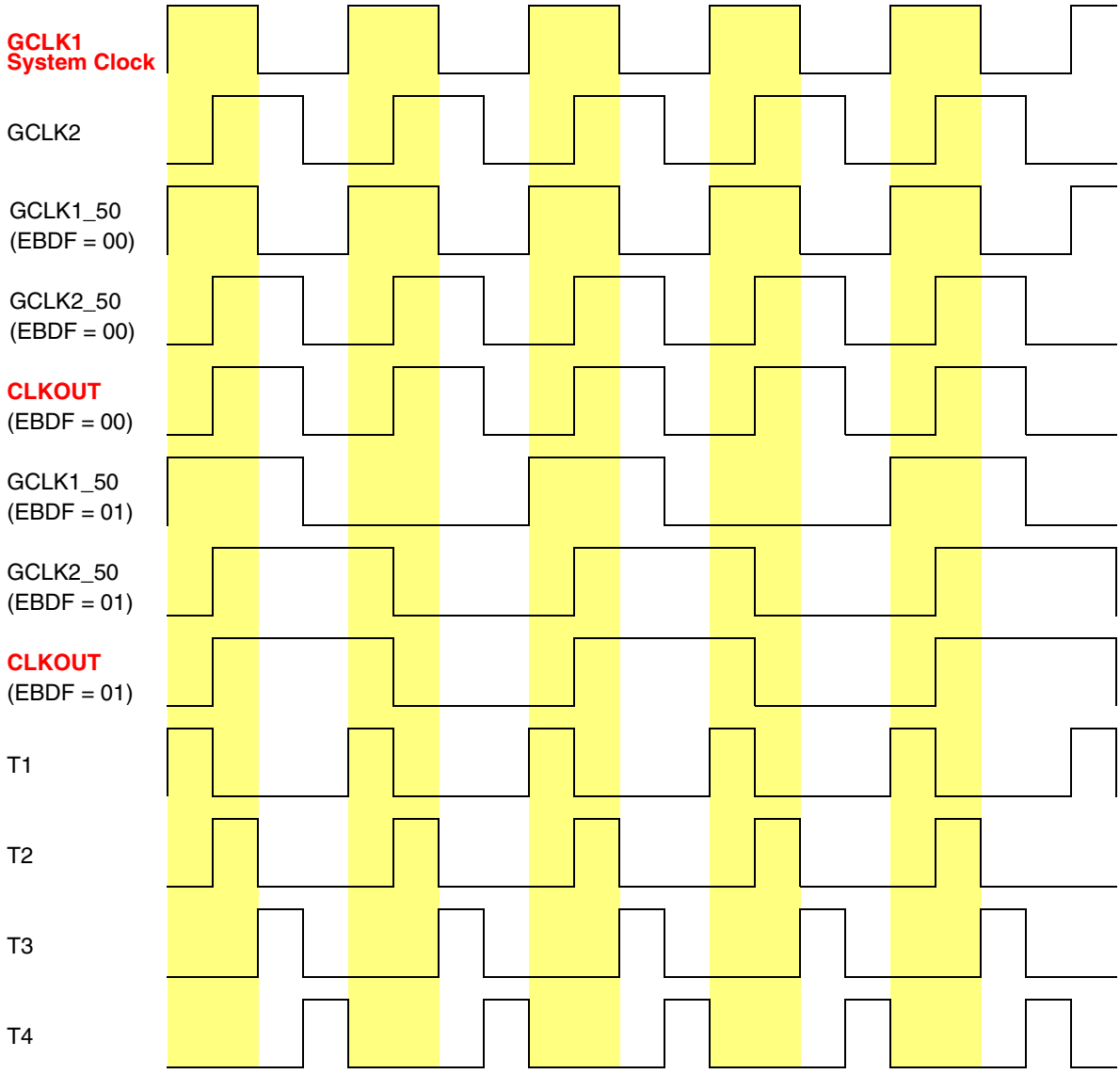


Figure 8-4 MPC555 / MPC556 Clocks

Note that GCLK1_50, GCLK2_50, and CLKOUT can have a lower frequency than GCLK1 and GCLK2. This is to enable the external bus operation at lower frequencies (controlled by EBDF in the SCCR). GCLK2_50 always rises simultaneously with

GCLK2. When DFNH = 0, GCLK2_50 has a 50% duty cycle. With other values of DFNH or DFNL, the duty cycle is less than 50%. Refer to **Figure 8-7**. GCLK1_50 rises simultaneously with GCLK1. When the MPC555 / MPC556 is not in gear mode, the falling edge of GCLK1_50 occurs in the middle of the high phase of GCLK2_50. EBDF determines the division factor between GCLK1/GCLK2 and GCLK1_50/GCLK2_50.

During power-on reset, the MOCCK1, MODCK2, and MODCK3 pins determine the clock source for the PLL and the clock drivers. These pins are latched on the positive edge of $\overline{\text{PORESET}}$. Their values must be stable as long as this line is asserted. The configuration modes are shown in **Table 8-1**. MODCK1 specifies the input source to the SPLL (OSCM or EXTCLK). MODCK1, MODCK2, and MODCK3 together determine the multiplication factor at reset and the functionality of limp mode.

If the configuration of PITRTCLK and TMBCLK and the SPLL multiplication factor is to remain unchanged in power-down low-power mode, the MODCK signals should not be sampled at wake-up from this mode. In this case the $\overline{\text{PORESET}}$ pin should remain negated and $\overline{\text{HRESET}}$ should be asserted during the power supply wake-up stage.

When MODCK1 is cleared, the output of the main oscillator (OSCM) is selected as the input to the SPLL. When MODCK1 is asserted, the external clock input (EXTCLK) is selected as the input to the SPLL. In all cases, the system clock frequency ($freq_{gclk2}$) can be reduced by the DFNH[0:2] bits in the SCCR. Note that $freq_{gclk2(max)}$ occurs when the DFNH bits are cleared.

The TBS bit in the SCCR selects the time base clock to be either the SPLL input clock or GCLK2. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

The PITRTCLK frequency and source are specified by the RTDIV and RTSEL bits in the SCCR. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

When the $\overline{\text{PORESET}}$ pin is negated (driven to a high value), the MODCK1, MODCK2, and MODCK3 values are not affected. They remain the same as they were defined during the most recent power-on reset.

Table 8-1 shows the clock configuration modes during power-on reset ($\overline{\text{PORESET}}$ asserted).

NOTE

The MODCK[1:3] are shared functions with $\overline{\text{IRQ}}$[5:7]. If $\overline{\text{IRQ}}$[5:7] are used as interrupts, the interrupt source should be removed during $\overline{\text{PORESET}}$ to insure the MODCK pins are in the correct state on the rising edge of $\overline{\text{PORESET}}$.

Table 8-1 Reset Clocks Source Configuration

MODCK[1:3][1]	LME	Default Values @ $\overline{\text{PORESET}}$			SPLL Options
		MF + 1	PITCLK Division	TMBCLK Division	
000	0	513	4	4	Used for testing purposes.
001	0	1	256	16	Normal operation, PLL enabled. Main timing reference is freq(OSCM) = 20 MHz. Limp mode disabled.
010	1	5	256	4	Normal operation, PLL enabled. Main timing reference is freq(OSCM) = 4 MHz. Limp mode enabled.
011	1	1	256	16	Normal operation, PLL enabled. Main timing reference is freq(OSCM) = 20 MHz. Limp mode enabled.
100 101	0	1	256	16	Normal operation, PLL enabled. 1:1 Mode freqclkout(max) = freq(EXTCLK) Limp mode disabled.
110	0	5	256	4	Normal operation, PLL enabled. Main timing reference is freq(EXTCLK) = 3-5 MHz. Limp mode disabled.
111	1	1	256	16	Normal operation, PLL enabled. 1:1 Mode freqclkout(max) = freq(EXTCLK) Limp mode enabled.

NOTES:
1. For other implementations in the MPC500 family, MODCK2 could be inverted.

NOTE

The reset value of the PLL pre-divider is 1.

The values of the PITRTCLK clock division and TMBCLK clock division can be changed by software. The RTDIV bit value in the SCCR register defines the division of PITRTCLK. All possible combinations of the TMBCLK divisions are listed in **Table 8-2**.

Table 8-2 TMBCLK Divisions

SCCR[TBS]	MF + 1	TMBCLK Division
1	—	16
0	1, 2	16
0	> 2	4

8.6.1 General System Clocks

The general system clocks (GCLK1C, GCLK2C, GCLK1, GCLK2, GCLK1_50, and GCLK2_50) are the basic clock supplied to all modules and sub-modules on the MPC555 / MPC556. GCLK1C and GCLK2C are supplied to the RCPU and to the BBC. GCLK1C and GCLK2C are stopped when the chip enters the doze-low power mode. GCLK1 and GCLK2 are supplied to the SIU and the clock module. The external bus clock GCLK2_50 is the same as CLKOUT. The general system clock defaults to VCO/

2 = 20 MHz (assuming a 20-MHz system frequency) with default power-on reset MF values.

The general system clock frequency can be switched between different values. The highest operational frequency can be achieved when the system clock frequency is determined by DFNH (CSRC bit in the PLPRCR is cleared) and DFNH = 0 (division by one). The general system clock can be operated at a low frequency (gear mode) or a high frequency. The DFNL bits in SCCR define the low frequency. The DFNH bits in SCCR define the high frequency.

The frequency of the general system clock can be changed dynamically with the system clock control register (SCCR), as shown in **Figure 8-5**.

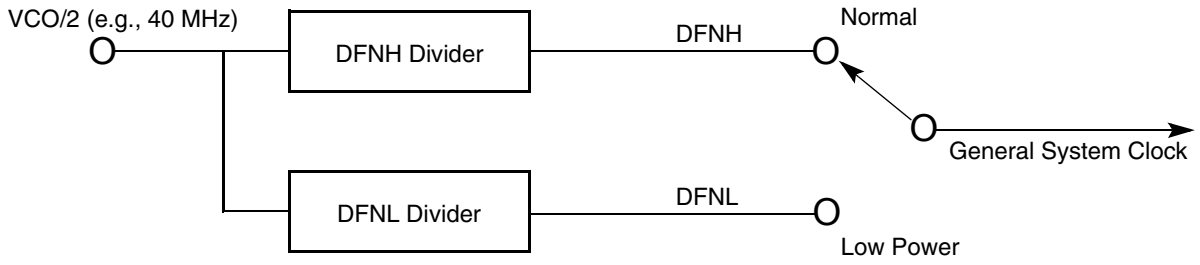


Figure 8-5 General System Clocks Select

The frequency of the general system clock can be changed “on the fly” by software. The user may simply cause the general system clock to switch to its low frequency. However, in some applications, there is a need for a high frequency during certain periods. Interrupt routines, for example, may require more performance than the low frequency operation provides, but must consume less power than in maximum frequency operation. The MPC555 / MPC556 provides a method to automatically switch between low and high frequency operation whenever one of the following conditions exists:

- There is a pending interrupt from the interrupt controller. This option is maskable by the PRQEN bit in the SCCR.
- The (POW) bit in the MSR is clear in normal operation. This option is maskable by the PRQEN bit in the SCCR.

When neither of these conditions exists and the CSRC bit in PLPRCR is set, the general system clock switches automatically back to the low frequency.

Abrupt changes in the divide ratio can cause linear changes in the operating currents of the MPC555 / MPC556. Insure that the proper power supply filtering is available to handle this change instantaneously.

When the general system clock is divided, its duty cycle is changed. One phase remains the same (e.g., 12.5 ns @ 40 MHZ) while the other become longer. Note that CLKOUT does not have a 50% duty cycle when the general system clock is divided. The CLKOUT waveform is the same as that of GCLK2_50.

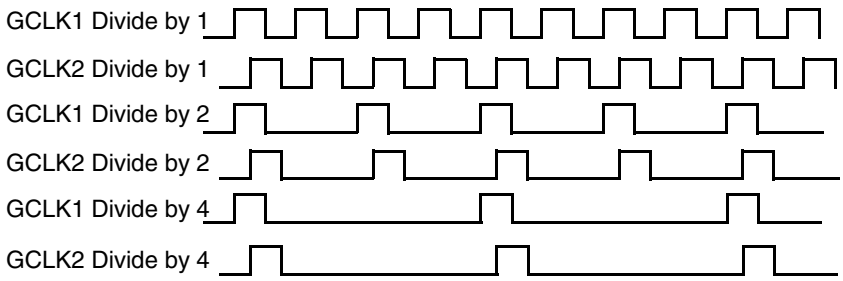


Figure 8-6 Divided System Clocks Timing Diagram

The system clocks GCLK1 and GCLK2 frequency is:

$$FREQ_{sys} = \frac{FREQsysmax}{(2^{DFNH})or(2^{DFNL+1})}$$

where FREQsysmax = System Frequency ($FREQ_{SYS}$)/2

Therefore, the complete equation for determining the system clock frequency is:

$$\text{System Frequency } (FREQ_{SYS}) = \frac{OSCCLK}{DIVF + 1} \times \frac{(MF + 1)}{(2^{DFNH} \text{ or } 2^{DFNL+1})} \times \frac{2}{2}$$

The clocks GCLK1_50 and GCLK2_50 frequency is:

$$FREQ_{50} = \frac{FREQsysmax}{(2^{DFNH})or(2^{DFNL+1})} \times \frac{1}{EBDF + 1}$$

Figure 8-7 shows the timing of USIU clocks when DFNH = 1 or DFNL = 0.

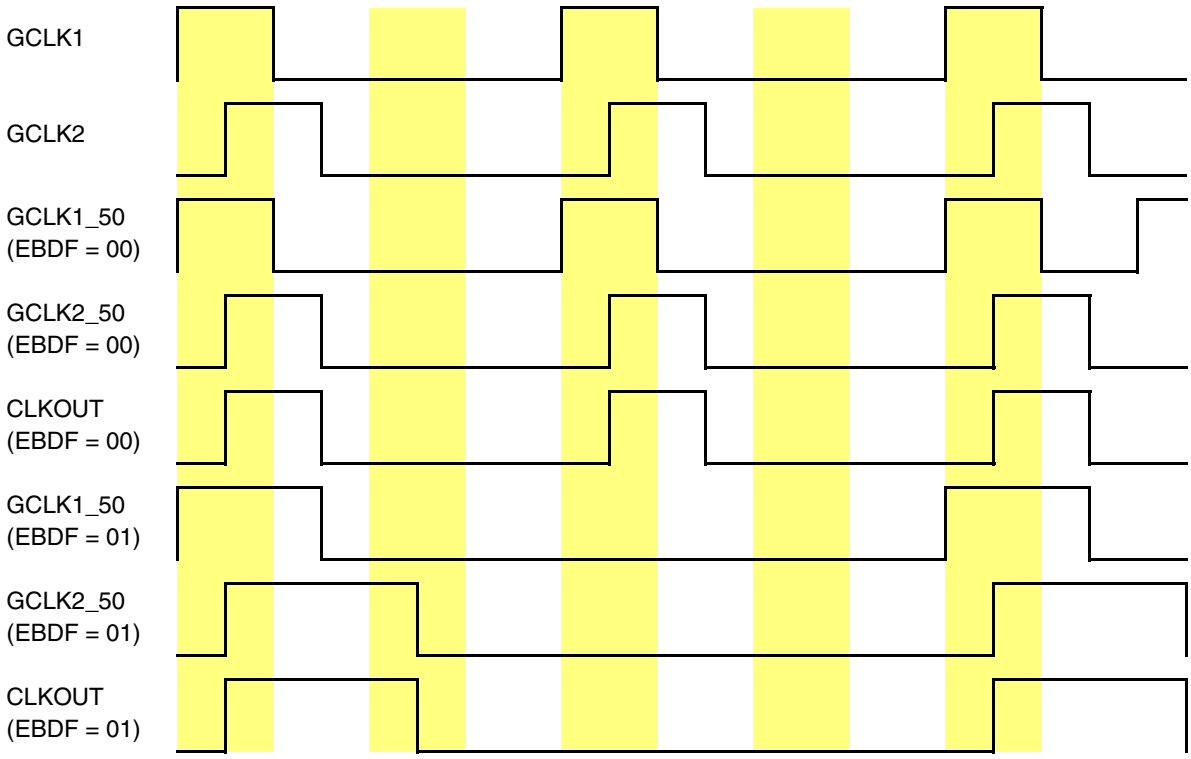


Figure 8-7 Clocks Timing For DFNH = 1 (or DFNL = 0)

8.6.2 CLKOUT

CLKOUT has the same frequency as the general system clock (GCLK2_50). Unlike the main system clock GCLK1/GCLK2 however, CLKOUT (and GCLK2_50) represents the external bus clock, and thus will be one-half of the main system clock if the external bus is running at half speed (EBDF = 0b01). The CLKOUT frequency defaults to VCO/2. CLKOUT can drive full- or half-strength or be disabled. The drive strength is controlled in the system clock and reset-control register (SCCR). Disabling or decreasing the strength of CLKOUT can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

When the PLL is acquiring lock, the CLKOUT signal is disabled and remains in the low state (provided that BUCS = 0).

8.6.3 Engineering Clock

ENGCLK is an output clock with a 50% duty cycle. Its frequency defaults to VCO/128[1], which is one-sixtyfourth of the main system frequency. ENGCLK frequency can be programmed to the main system frequency divided by a factor from one to 64, as controlled by the ENGDIV[0:5] bits in the SCCR. ENGCLK can drive full- or half-strength or be disabled (remaining in the high state). The drive strength is controlled by the EECLK[0:1] bits in the SCCR. Disabling ENGCLK can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

[1.] Mask sets prior to K62N default to VCO/4.

When the PLL is acquiring lock, the ENGCLK signal is disabled and remains in the low state (provided that BUCS = 0).

NOTE

Skew elimination between CLKOUT and ENGCLK is not guaranteed.

8.7 Clock Source Switching

For limp mode support, clock source switching is supported. If for any reason the clock source for the chip is not functioning, the user has the option to switch the system clock to the backup clock ring oscillator, BUCLK.

This circuit consists of a loss-of-clock detector, which sets the LOCS status bit and LOCSS sticky bit in the PLPRCR. If the LME bit in the SCCR is set, whenever LOCS is asserted the clock logic switches the system clock automatically to BUCLK and asserts hard reset to the chip. Switching the system clock to BUCLK is also possible by software setting the STBUC bit in SCCR. Switching from limp mode to normal system operation is accomplished by clearing STBUC and LOCSS bits. This operation also asserts hard reset to the chip.

At $\overline{\text{HRESET}}$ assertion, if the PLL output clock is not valid, the BUCLK will be selected until software clears LOCSS bit in SCCR. At $\overline{\text{HRESET}}$ assertion, if the PLL output clock is valid, the system will switch to oscillator/external clock. If during $\overline{\text{HRESET}}$ the PLL loses lock or the clock frequency becomes slower than the required value, the system will switch to the BUCLK. After $\overline{\text{HRESET}}$ negation, the PLL lock condition does not effect the system clock source selection.

If the LME bit is clear, the switch to the backup clock is disabled and assertion of STBUC bit is ignored. If the chip is in limp mode, clearing the LME bit switches the system to normal operation and asserts hard reset to the chip.

Figure 8-8 describes the clock switching control logic. **Table 8-3** summarizes the status and control for each state.

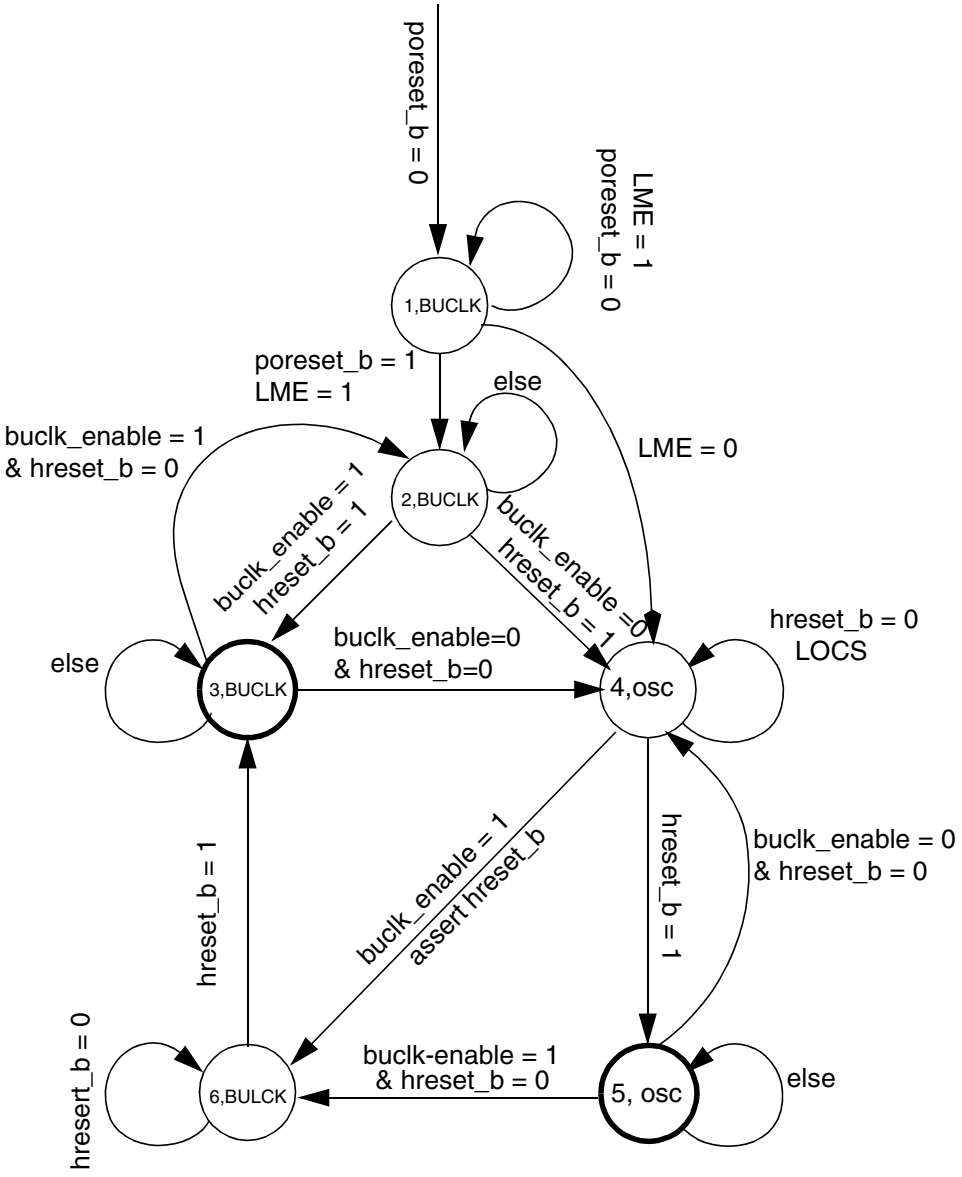


Figure 8-8 Clock Source Flow Chart

NOTES

BUCLK_ENABLE = (STBUC | LOC) & LME lock indicates loss of lock status bit (LOCS) for all cases and loss of clock sticky bit (LOCSS) when state 3 is active. When BUCLK_ENABLE is changed, the chip asserts $\overline{\text{HRESET}}$ to switch the system clock to BUCLK or PLL.

At $\overline{\text{PORESET}}$ negation, if the PLL is not locked, the loss-of-clock sticky bit (LOCSS) is asserted, and the chip should operate with BUCLK.

The switching from state three to state four is accomplished by clearing the STBUC and LOCSS bits. If the switching is done when the PLL is not locked, the system clock will not oscillate until lock condition is met.

Table 8-3 Status of Clock Source

STATE	$\overline{\text{PORESET}}$	$\overline{\text{HRESET}}$	LME	LOCS (status)	LOCSS (sticky)	STBUC	BUCS	Chip Clock Source
1	0	0	1	0	0	0	1	BUCLK
2	1	0	1	0/1	0	0	1	BUCLK
3[1]	1	1	1	x[2]	0/1	0/1	1	BUCLK
4	1	0	0/1	0	x[2]	0	0	Oscillator
5	1	1	0/1	0	x[2]	0	0	Oscillator
6	1	0	1	0/1	1	0/1	1	BUCLK

NOTES:
1. At least one of the two bits, LOCSS or BUCS, must be asserted (one) in this state.
2. X = don't care.

The default value of the LME bit is determined by MODCK[1:3] during assertion of the $\overline{\text{PORESET}}$ line. The configuration modes are shown in **Table 8-1**.

8.8 Low-Power Modes

The LPM and other bits in the PLPRCR are encoded to provide one normal operating mode and four low-power modes. In normal and doze modes the system can be in high state with frequency defined by the DFNH bits, or in the low state with frequency defined by the DFNL bits. The normal-high operating mode is the state out of reset. This is also the state of the bits after the low-power mode exit signal arrives.

There are four low-power modes:

- Doze mode
- Sleep mode
- Deep-sleep mode
- Power-down mode

8.8.1 Entering a Low-Power Mode

Low-power modes are enabled by setting the POW bit in the MSR and clearing the LPML (low-power mode lock) bit in the PLPRCR. Once enabled, a low-power mode is entered by setting the LPM bits to the appropriate value. This can be done only in one of the normal modes. The user cannot change the LPM or CSRC bits when the MCU is in doze mode.

Table 8-6 summarizes the control bit descriptions for the different clock power modes.

Table 8-4 Power Mode Control Bit Descriptions

Power Mode	LPM[0:1]	CSRC	TEXPS
Normal-high	00	0	X
Normal-low (“gear”)	00	1	X
Doze-high	01	0	X
Doze-low	01	1	X
Sleep	10	X	X
Deep-sleep	11	X	1
Power-down	11	X	0

8.8.2 Power Mode Descriptions

Table 8-5 describes the power consumption, clock frequency, and chip functionality for each power mode.

Table 8-5 Power Mode Descriptions

Operation Mode	SPLL	Clocks	Functionality
Normal-high	Active	Full frequency ÷ 2^{DFNH}	Full functions not in use are shut off
Normal-low (“gear”)	Active	Full frequency ÷ 2^{DFNL+1}	
Doze-high	Active	Full frequency ÷ 2^{DFNH}	Enabled: RTC, PIT, TB and DEC, memory controller Disabled: extended core (RCPU, BBC, FPU)
Doze-low	Active	Full frequency ÷ 2^{DFNL+1}	
Sleep	Active	Not active	Enabled: RTC, PIT, TB and DEC
Deep-sleep	Not active	Not active	
Power-down	Not active	Not active	
VDDSRAM	Not active	Not active	SRAM’s data retention

8.8.3 Exiting from Low-Power Modes

Exiting from low-power modes occurs through an asynchronous interrupt or a synchronous interrupt generated by the memory controller. Any enabled asynchronous interrupt clears the LPM bits but does not change the PLPRCR[CSRC] bit.

The exit from normal-low, doze-high, and low modes and sleep mode to normal-high mode is accomplished with the asynchronous interrupt. The sources of the asynchronous interrupt are:

- Asynchronous wake-up interrupt from the interrupt controller
- RTC, PIT, or time base interrupts (if enabled)
- Decrementer exception

The system response to asynchronous interrupts is fast. The wake-up time from normal-low, doze-high, doze-low, and sleep mode due to an asynchronous interrupt or

decrementer exception is only three to four clock cycles of maximum system frequency. In 40-MHz systems, this wake-up requires 75 to 100 ns. The asynchronous wake-up interrupt from the interrupt controller is level sensitive one. It will therefore be negated only after the reset of interrupt cause in the interrupt controller.

The timers (RTC, PIT, time base, or decrementer) interrupts indication set status bits in the PLPRCR (TMIST). The clock module considers this interrupt to be pending asynchronous interrupt as long as the TMIST is set. The TMIST status bit should be cleared before entering any low-power mode.

Table 8-7 summarizes wake-up operation for each of the low-power modes.

Table 8-6 Power Mode Wake-Up Operation

Operation Mode	Wake-up Method	Return Time from Wake-up Event to Normal-High
Normal-low (“gear”)	Software or Interrupt	Asynchronous interrupts: 3-4 maximum system cycles Synchronous interrupts: 3-4 actual system cycles
Doze-high	Interrupt	
Doze-low	Interrupt	
Sleep	Interrupt	3-4 maximum system clocks
Deep-sleep	Interrupt	< 500 Oscillator Cycles 125 µsec – 4 MHz 25 µsec – 20 MHz
Power-down	Interrupt	< 500 oscillator cycles + power supply wake-up
V_{DDSRAM}	External	Power-on sequence

8.8.3.1 Exiting from Normal-Low Mode

In normal mode (as well as doze mode), if the PLPRCR[CSRC] bit is set, the system toggles between low frequency (defined by PLPRCR[DFNL]) and high frequency (defined by PLPRCR[DFNH]. The system switches from normal-low mode to normal-high mode if either of the following conditions is met:

- An interrupt is pending from the interrupt controller; or
- The MSR[POW] bit is cleared (power management is disabled).

When neither of these conditions are met, the PLPRCR[CSRC] bit is set, and the asynchronous interrupt status bits are reset, the system returns to normal-low mode.

8.8.3.2 Exiting from Doze Mode

The system changes from doze mode to normal-high mode whenever an interrupt is pending from the interrupt controller.

8.8.3.3 Exiting from Deep-Sleep Mode

The system switches from deep-sleep mode to normal-high mode if any of the following conditions is met:

- An interrupt is pending from the interrupt controller
- An interrupt is requested by the RTC, PIT, or time base
- A decrementer exception

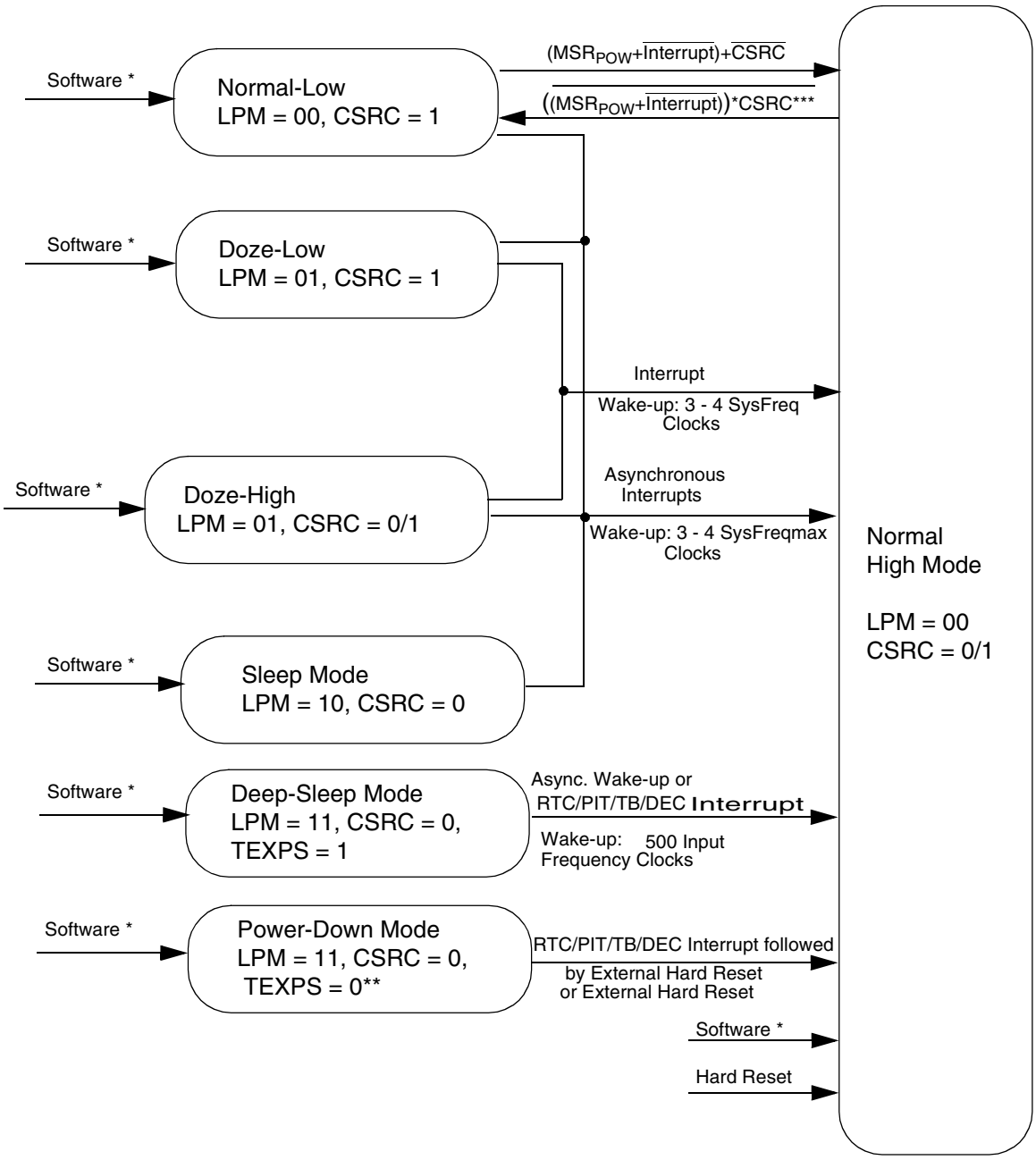
In deep-sleep mode the PLL is disabled. The wake-up time from this mode is up to 500 PLL input frequency clocks. In one-to-one mode the wake-up time may be up to 100 PLL input frequency clocks. For a PLL input frequency of 4 MHz, the wake-up time is less than 125 µs.

8.8.3.4 Exiting from Power-Down Mode

Exit from power-down mode is accomplished through hard reset. External logic should assert $\overline{\text{HRESET}}$ in response to the TEXPS bit being set and TEXP pin being asserted. The TEXPS bit is set by an enabled RTC, PIT, time base, or decrementer interrupt. The hard reset should be asserted for no longer than the time it takes for the power supply to wake-up in addition to the PLL lock time. When the TEXPS bit is cleared (and the TEXP signal is negated), assertion of hard reset sets the bit, causes the pin to be asserted, and causes an exit from power-down low-power mode. Refer to **8.9.3 Keep Alive Power** for more information.

8.8.3.5 Low-Power Modes Flow

Figure 8-9 shows the flow among the different power modes.



* Software is active only in normal-high/low modes
** TEXPS receives the zero value by writing one. Writing of zero has no effect on TEXPS.
*** The switch from normal-high to normal-low is enable only if the conditions to asynchronous interrupt are cleared

Figure 8-9 MPC555 / MPC556 Low-Power Modes Flow Diagram

8.9 Basic Power Structure

8.9.1 Clock Unit Power Supply

KAPWR and VSS power the following clock unit modules: oscillator, PITRTCLK and TMBCLK generation logic, timebase, decrementer, RTC, PIT, system clock control register (SCCR), low-power and reset-control register (PLPRCR), and reset status register (RSR). All other circuits are powered by the normal supply pins, VDDI, VDDL, VDDH and VSS. The power supply for each block is listed in **Table 8-7**.

Table 8-7 Clock Unit Power Supply

Circuit	Power Supply
CLKOUT SPLL (digital), System low-power control Internal logic Clock drivers	VDDL/VDDI
SPLL (analog)	VDDSYN
Main oscillator Reset machine Limp mode mechanism Register control SCCR, PLLRCR and RSR RTC, PIT, TB, and DEC	KAPWR
SRAM, VDDSRAM detector, VSRMCR	VDDSRAM

The following are the relations between different power supplies:

- VDDL = VDDI = VDDSYN = VDDF = 3.3 V ±10%
- KAPWR ≥ VDDL – 0.2 V (during normal operation)
- VDDSRAM ≥ VDDL – 0.3 V (during normal operation)
- VDDSRAM ≥ 1.4 V (during standby operation)
- VPP ≥ VDDL – 0.3 V, but VPP – VDDL < 4.0 volts

8.9.2 Chip Power Structure

The MPC555 / MPC556 provides a wide range of possibilities for power supply connections. **Figure 8-10** illustrates the different power supply sources for each of the basic units on the chip.

8.9.2.1 VDDL

The I/O buffers and logic are fed by a 3.3-V power supply.

8.9.2.2 VDDI

VDDI powers the internal logic of the MPC555 / MPC556, nominally 3.3 V.

8.9.2.3 VDDSYN, VSSSYN

The charge pump and the VCO of the SPLL are fed by a separate 3.3-V power supply (VDDSYN) in order to improve noise immunity and achieve a high stability in its output frequency. VSSSYN provides an isolated ground reference for the PLL.

8.9.2.4 KAPWR

The oscillator, time base counter, decrementer, periodic interrupt timer and the real-time clock are fed by the KAPWR rail. This allows the external power supply unit to disconnect all other sub-units of the MCU in low-power deep-sleep mode. The TEXP pin (fed by the same rail) can be used by the external power supply unit to switch between sources. The $\overline{\text{IRQ}}$[6:7]/MODCK[2:3], $\overline{\text{IRQ}}$[5]/MODCK1, XTAL, EXTAL, EXTCLK, $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$, and $\overline{\text{RSTCONF}}$/TEXP input pins are powered by KAPWR. Circuits, including pull-up resisters, driving these inputs should be powered by KAPWR.

8.9.2.5 VDDA, VSSA

VDDA supplies power to the analog subsystems of the QADC_A and QADC_B modules; it is nominally 5.0 V. VDDA is the ground reference for the analog subsystems.

8.9.2.6 VPP

VPP supplies the programming and erase voltage for the CMF flash modules. It is nominally 5.0 V for program or erase operations and can be lowered to a nominal 3.3 V for read operations.

8.9.2.7 VDDF, VSSF

VDDF provides internal power to the CMF flash module; it should be a nominal 3.3 V. VSSF provides an isolated ground for the CMF flash module.

8.9.2.8 VDDH

VDDH provides power for the 5-V I/O operations. It is a nominal 5.0 V.

8.9.2.9 VDDSRAM

VDDSRAM supplies power to the 26-Kbyte SRAM module and the DPTRAM. It can be used to keep the contents on the SRAM stable while the rest of the MPC555 / MPC556 is powered down for standby operation.

8.9.2.10 VSS

VSS provides the ground reference for the MPC555 / MPC556.

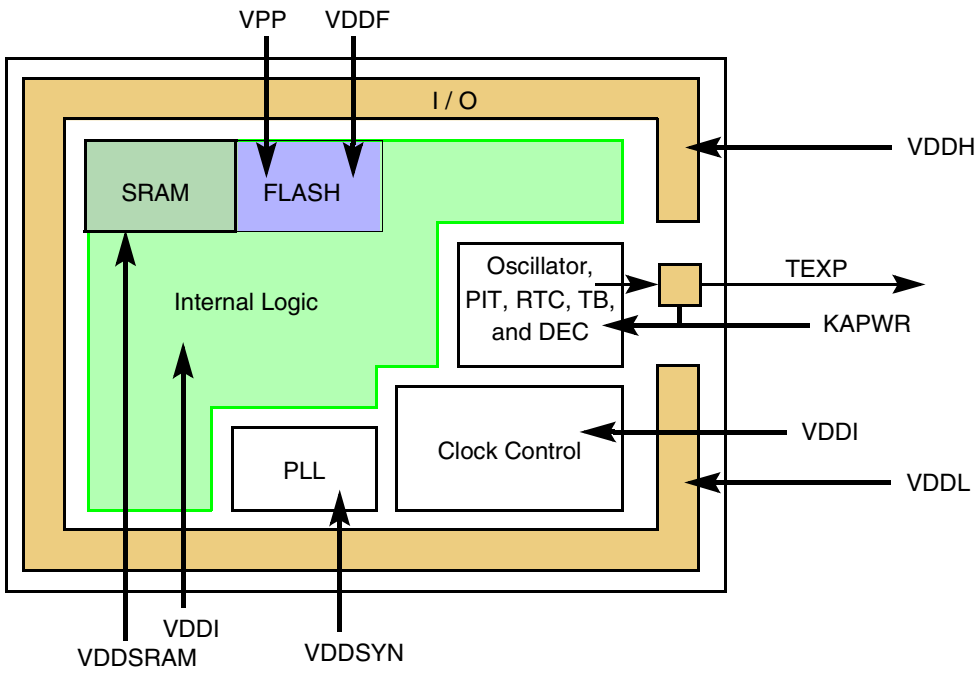


Figure 8-10 Basic Power Supply Configuration

8.9.3 Keep Alive Power

8.9.3.1 Keep Alive Power Configuration

Figure 8-11 illustrates an example of a switching scheme for an optimized low-power system. SW1 and SW2 can be unified in only one switch if VDDSYN and VDDI/VDDL are supplied by the same source.

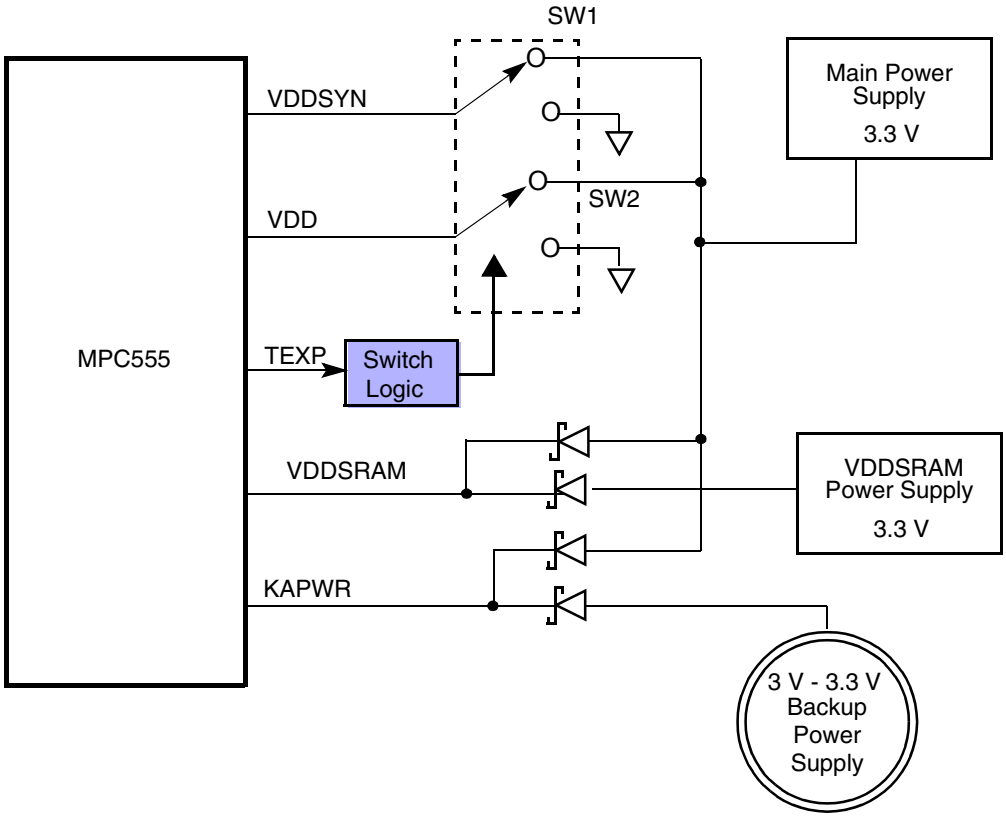


Figure 8-11 External Power Supply Scheme

The MPC555 / MPC556 asserts the TEXP signal, if enabled, when the RTC or TB time value matches the value programmed in the associated alarm register or when the PIT or DEC value reaches zero. The TEXP signal is negated when the TEXPS status bit is written to one.

The KAPWR power supply feeds the main crystal oscillator (OSCM). The condition for the main crystal oscillator stability is that the power supply value changes slowly. The maximum slope must be less than 5 mV per oscillation cycle ($\tau > 200\text{-}300/freq_{oscm}$).

8.9.3.2 Keep Alive Power Registers Lock Mechanism

The USIU timer, clocks, reset, power, decrementer, and time base registers are powered by the KAPWR supply. When the main power supply is disconnected after power-down mode is entered, the value stored in any of these registers is preserved. If power-down mode is not entered before power disconnect, there is a chance of data loss in these registers. To minimize the possibility of data loss, the MPC555 / MPC556 includes a key mechanism that ensures data retention as long as a register is locked. While a register is locked, writes to this register are ignored.

Each of the registers in the KAPWR region have a key that can be in one of two states: open or locked. At power-on reset the following keys are locked: RTC, RTSEC, RT-

CAL, and RTCSC. All other registers are unlocked. Each key has an address associated with it in the internal memory map.

A write of 0x55CCAA33 to the associated key register changes the key to the open state. A write of any other data to this location changes the key to the locked state.

NOTE

A read of a key register may be interpreted as a write of a lock value and may lock the associated KAPWR register.

The key registers are write-only. A read of the key register has undefined side effects and may be interpreted as a write that locks the associated register.

Table 8-8 lists the registers powered by KAPWR and the associated key registers.

Table 8-8 KAPWR Registers and Key Registers

KAPWR Register		Associated Key Register	
Address or SPR Number	**Register**	**Address**	**Register**
0x2F C200	Time Base Status and Control (TBSCR) See **Table 6-16** for bit descriptions.	0x2F C300	Time Base Status and Control Key (TBSCRK)
0x2F C204	Time Base Reference 0 (TBREF0) See **6.13.4.3 Time Base Reference Registers** for bit descriptions.	0x2F C304	Time Base Reference 0 Key (TBREF0K)
0x2F C208	Time Base Reference 1 (TBREF1) See **6.13.4.3 Time Base Reference Registers** for bit descriptions.	0x2F C308	Time Base Reference 1 Key (TBREF1K)
0x2F C220	Real Time Clock Status and Control (RTCSC) See **Table 6-17** for bit descriptions.	0x2F C320	Real Time Clock Status and Control Key (RTCSCK)
0x2F C224	Real Time Clock (RTC) See **6.13.4.6 Real-Time Clock Register (RTC)** for bit descriptions.	0x2F C324	Real Time Clock Key (RTCK)
0x2F C228	Real Time Alarm Seconds (RTSEC) Reserved	0x2F C328	Real Time Alarm Seconds Key (RTSECK)
0x2F C22C	Real Time Alarm (RTCAL) See **6.13.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.	0x2F C32C	Real Time Alarm Key (RTCALK)
0x2F C240	PIT Status and Control (PISCR) See **Table 6-18** for bit descriptions.	0x2F C340	PIT Status and Control Key (PISCRK)
0x2F C244	PIT Count (PITC) See **Table 6-19** for bit descriptions.	0x2F C344	PIT Count Key (PITCK)
0x2F C280	System Clock Control Register (SCCR) See **Table 8-9** for bit descriptions.	0x2F C380	System Clock Control Key (SCCRK)
0x2F C284	PLL Low-Power and Reset-Control Register (PLPRCR) See **Table 8-10** for bit descriptions.	0x2F C384	PLL Low-Power and Reset-Control Register Key (PLPRCRK)
0x2F C288	Reset Status Register (RSR) See **Table 7-3** for bit descriptions.	0x2F C388	Reset Status Register Key (RSRK)

Table 8-8 KAPWR Registers and Key Registers (Continued)

KAPWR Register		Associated Key Register	
Address or SPR Number	**Register**	**Address**	**Register**
SPR 22	Decrementer See **3.9.5 Decrementer Register (DEC)** for bit descriptions.	0x2F C30C	Time Base and Decrementer Key (TBK)
SPR 268, 269, 284, 285,	Time Base See **Table 3-11** and **Table 3-14** for bit descriptions.		

Figure 8-12 illustrates the process of locking or unlocking a register powered by KAPWR.

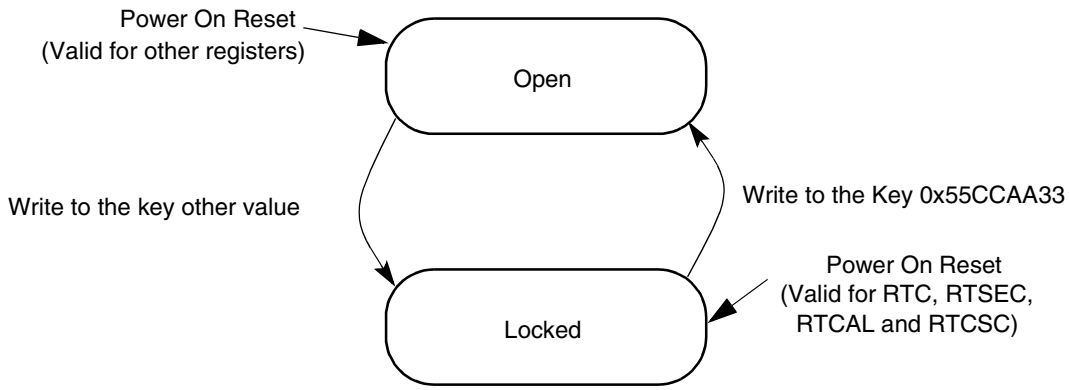


Figure 8-12 Keep Alive Register Key State Diagram

8.10 VDDSRAM Supply Failure Detection

A special circuit for VDDSRAM supply failure detection is provided. In the case of supply failure detection, the dedicated sticky bits LVSRS in the VSRMCR register are asserted. Software can read or clear these bits. The user should enable the detector and then clear these bits. If the user reads any of the LVSR bits as one, then a power failure of VDDSRAM has occurred. The circuit is capable of detecting supply failure below 2.6 V. Also, enable/disable control bit for the VDDSRAM detector may be used to disconnect the circuit and save the detector power consumption.

8.11 Power Up/Down Sequencing

Figure 8-13 and **Figure 8-14** detail the power-up sequencing for MPC555 / MPC556 during normal operation. Note that for each of the conditions detailing the voltage relationships the absolute bounds of the minimum and maximum voltage supply cannot be violated, i.e. the value of VDDL cannot fall below 3.0 V or exceed 3.6 V and the value of VDDH cannot fall below 4.5 V or exceed 5.5 V for normal operation. Further information detailing the functionality of the VPP signal for flash program and erase is outlined in **19.9.2 FLASH Program/Erase Voltage Conditioning**. Power consumption during power up sequencing can not be specified prior to evaluation and charac-

terization of production silicon. The goal is to keep the power consumption during power up sequencing below the operating power consumption.

During the power down sequence the user needs to assert $\overline{\text{PORESET}}$ while VDDI and VDDL are at a voltage equal or greater to 3 V. Below this voltage the power supply chip can be turned off. If the turn off voltage of the power supply chip is greater than 0.74 V for the 3-V supply and greater than 0.8 V for the 5-V supply, then the circuitry inside the MPC555 / MPC556 will act as a load to the respective supply and will discharge the supply line down to these values. Since the 3-V logic represents a larger load to the supply chip, the 3-V supply line will decay faster than the 5-V supply line.

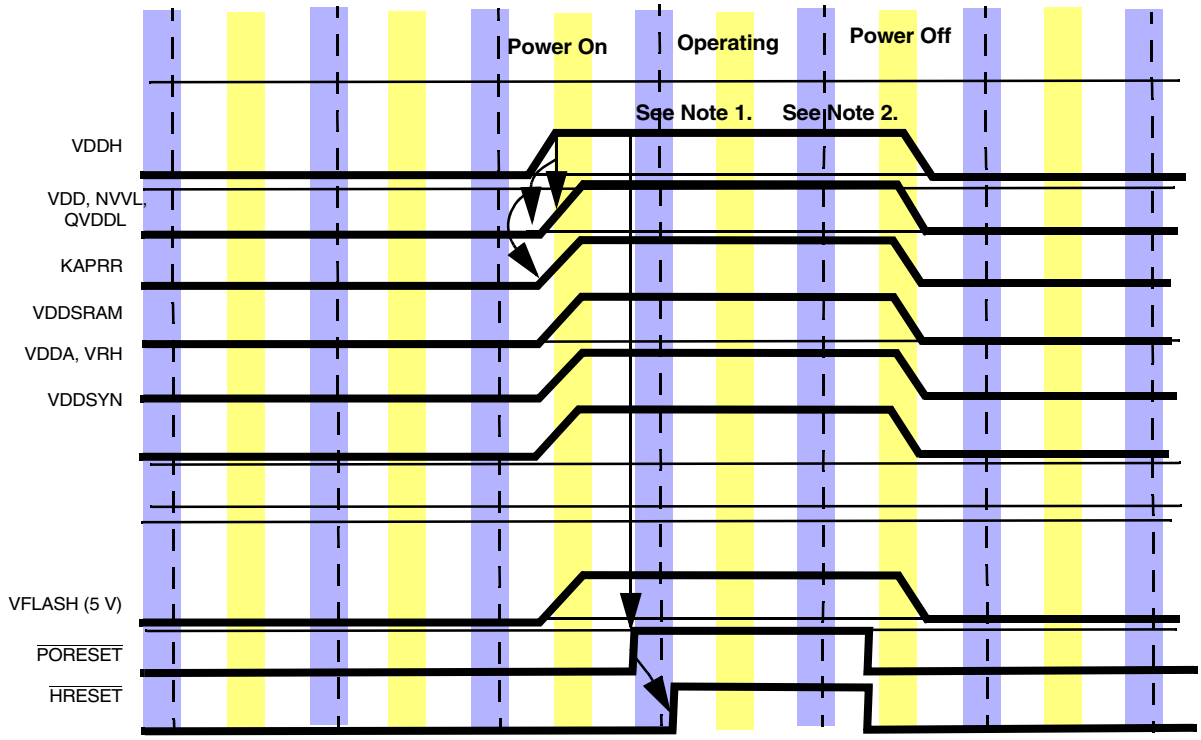


Figure 8-13 No Standby, No KAPWR, All System Power On/Off

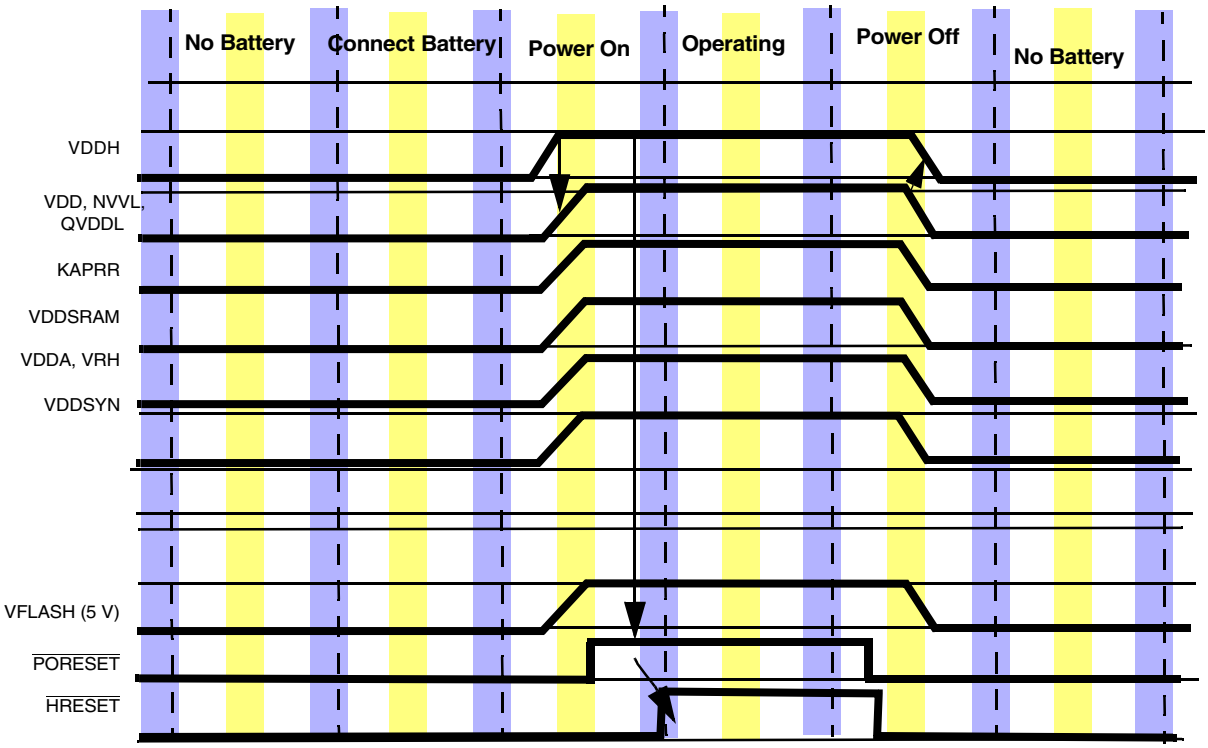


Figure 8-14 Standby and KAPWR, Other Power-On/Off

NOTE

The following notes apply to **Figure 8-13** and **Figure 8-14** above:

1. VDDH ≥ VDDL - 0.35 V (0.5 V max. at temperature extremes)
   VPP ≤ VDDH + 0.5 V AND VPP ≥ VDDL - 0.35 V
   (The delta VPP - VDDL must be ≤ 3.6 V during power on or off)

   VDDA can lag VDDH, and VDDSYN can lag VDDL, but both must be at a valid level before resets are negated.

2. If keep alive functions are NOT used, then when system power is on:

   KAPWR = VDDSRAM = VDDL ± 0.35 V

3. If keep alive functions ARE used, then

   KAPWR = VDDSRAM = VDDL = 3.3 V ± 0.35 V when system power is on
   VDDSRAM ≥ 1.8 V and optionally KAPWR = 3.3 V ± 0.3 V when system power is off

   Normal system power is defined as
   VDDL = VDDI = VDDF = VDDSYN = VPP = VDDSRAM = KAPWR = 3.3 ± 0.3 V and VDDA = VDDH = 5.0 ± 0.5 V

Flash programming requirements are the same as normal system power, except VPP = 5.0 ± 0.25 V

4. Do not hold the 3-V supplies at ground while VDDH/VDDA is ramping to 5 V.
5. If 5 V is applied before the 3-V supply, all 5-V outputs will be in indeterminate states until the 3-V supply reaches a level that allows reset to be distributed throughout the device

8.12 Clocks Unit Programming Model

8.12.1 System Clock Control Register (SCCR)

The SPLL has a 32-bit control register, SCCR, which is powered by keep-alive power.

SCCR — System Clock Control Register **0x2F C280**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DBCT	COM		DCSLR	MFPDL	LPML	TBS	RTDIV[2]	STBUC	RE-SERVED	PRQEN	RTSEL	BUCS	EBDF		LME
POWER-ON RESET:															
1	0	ID2[1]	0	0	0	0	1[2]	0	0	1	EQ2[3]		ID[13:14][1]		EQ3[4]
HARD RESET:															
U	0	ID2[1]	U	0	0	U	U	U	0	1	U	U	ID[13:14][1]		U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
EECLK		ENGDIV[5]						—	DFNL			—	DFNH		
POWER-ON RESET:															
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
HARD RESET:															
U	U	U	U	U	U	U	U		0	0	0	0	0	0	0

NOTES:
1. The hard reset value is a reset configuration word value, extracted from the indicated internal data bus lines. Refer to **7.5.2 Hard Reset Configuration Word**.

U = Unaffected by reset

2. RTDIV will be 0 if MODCK[1:3] = 0b000
3. EQ2 = MODCK1
4. EQ3 = ($\overline{\text{MODCK1}}$ & MODCK2 & $\overline{\text{MODCK3}}$) | ($\overline{\text{MODCK1}}$ & MODCK2 & MODCK3) | (MODCK1 & $\overline{\text{MODCK2}}$ & MODCK3). See **Table 8-1**.
5. On mask sets prior to K62N, ENGDIV defaults to 0b000001.

Table 8-9 SCCR Bit Descriptions

Bit(s)	Name	Description
0	DBCT	Disable backup clock for timers. The DBCT bit controls the timers clock source while the chip is in limp mode. If DBCT is set, the timers clock (tbclk, rtclk) source will not be the backup clock, even if the system clock source is the backup clock ring oscillator. The real-time clock source will be EXTAL or EXTCLK according to RTSEL bit (see description in bit 11 below), and the time base clocks source will be determined according to TBS bit and MODCK1. 0 = If the chip is in limp mode, the timer clock source is the backup (limp) clock 1 = The timer clock source is either the external clock or the crystal (depending on the current clock mode selected)
1:2	COM	Clock output mode. These bits control the output buffer strength of the CLKOUT and external bus pins. These bits can be dynamically changed without generating spikes on the CLKOUT and external bus pins. If CKLOUT is not connected to external circuits, set both bits (disabling CLKOUT) to minimize noise and power dissipation. COM1 is determined by the hard reset configuration word. 00 = Clock output enabled full-strength output buffer, bus pins full drive 01 = Clock output enabled half-strength output buffer, bus pins reduced drive 10 = Clock output disabled, bus pins full drive 11 = Clock output disabled, bus pins reduced drive
3	DCSLR	Disable clock switching at loss of lock during reset. When DCSLR is clear and limp mode is enabled, the chip will switch automaticaly to the backup clock if the PLL losses lock during $\overline{\text{HRESET}}$. When DCSLR is asserted, a PLL loss-of-lock event does not cause clock switching. If $\overline{\text{HRESET}}$ is asserted and DCSLR is set, the chip will not negate $\overline{\text{HRESET}}$ until the PLL aquires lock. 0 = Enable clock switching if the PLL loses lock during reset 1 = Disable clock switching if the PLL loses lock during reset
4	MFPDL	MF and pre-divider lock. Setting this control bit disables writes to the MF and DIVF bits. This helps prevent runaway software from changing the VCO frequency and causing the SPLL to lose lock. In addition, to protect against hardware interference, a hardware reset will be asserted if these fields are changed while LPML is asserted. This bit is writable once after power-on reset. 0 = MF and DIVF fields are writable 1 = MF and DIVF fields are locked
5	LPML	LPM lock. Setting this control bit disables writes to the LPM and CSRC control bits. In addition, for added protection, a hardware reset is asserted if any mode is entered other than normal-high mode. This protects against runaway software causing the MCU to enter low-power modes. (The MSR[POW] bit provides additional protection). LPML is writable once after power-on reset. 0 = LPM and CSRC bits are writable 1 = LPM and CSRC bits are locked and hard reset will occur if the MCU is not in normal-high mode
6	TBS	Time base source. Note that when the chip is operating in limp mode (BUCS = 1), TBS is ignored, and the backup clock is the time base clock source. 0 = Source is OSCCLK divided by either four or 16 1 = Source is system clock divided by 16
7	RTDIV	RTC (and PIT) clock divider. At power-on reset this bit is cleared if MODCK[1:3] are all low; otherwise the bit is set. 0 = RTC and PIT clock divided by four 1 = RTC and PIT clock divided by 256
8	STBUC	Switch to backup clock control. When software sets this bit, the system clock is switched to the on-chip backup clock ring oscillator, and the chip undergoes a hard reset. The STBUC bit is ignored if LME is cleared. 0 = Do not switch to the backup clock ring oscillator 1 = Switch to backup clock ring oscillator
9	—	Reserved

Table 8-9 SCCR Bit Descriptions (Continued)

Bit(s)	Name	Description
10	PRQEN	Power management request enable 0 = Remains in the lower frequency (defined by DFNL) even if the power management bit in the MSR is reset (normal operational mode) or if there is a pending interrupt from the interrupt controller 1 = Switches to high frequency (defined by DFNH) when the power management bit in the MSR is reset (normal operational mode) or there is a pending interrupt from the interrupt controller
11	RTSEL	RTC circuit input source select. At power-on reset RTSEL receives the value of the MODCK1 bit. Note that if the chip is operating in limp mode (BUCS = 0), the RTSEL bit is ignored, and the backup clock is the clock source for the RT and PIT clocks 0 = OSCM clock is selected as input to RTC and PIT 1 = EXTCLK clock is selected as the RTC and PIT clock source
12	BUCS	Backup clock status. This status bit indicates the current system clock source. When loss of clock is detected and the LME bit is set, the clock source is the backup clock and this bit is set. When the user sets the STBUC bit and LME bit is set, the system switches to the backup clock and BUCS is set. 0 = System clock is not the backup clock 1 = System clock is the backup clock
13:14	EBDF	External bus division factor. These bits define the frequency division factor between (GCLK1 and GCLK2) and (GCLK1_50 and GCLK2_50). CLKOUT is similar to GCLK2_50. The GCLK2_50 and GCKL1_50 are used by the external bus interface and memory controller in order to interface to the external system. The EBDF bits are initialized during hard reset using the hard reset configuration mechanism. 00 = CLKOUT is GCKL2 divided by 1 01 = CLKOUT is GCKL2 divided by 2 1x = Reserved
15	LME	Limp mode enable. When LME is set, the loss-of-clock monitor is enabled and any detection of loss of clock will switch the system clock automatically to backup clock. It is also possible to switch to the backup clock by setting the STBUC bit. If LME is cleared, the option of using limp mode is disabled. The loss of clock detector is not active, and any write to STBUC is ignored. The LME bit is writable once, by software, after power-on reset, when the system clock is not backup clock (BUCS = 0). During power-on reset, the value of LME is determined by the MODCK[1:3] bits. (Refer to **Table 8-1**.) 0 = Limp mode disabled 1 = Limp mode enabled
16:17	EECLK	Enable engineering clock. This field controls the output buffer strength of the ENGCLK pin. When both bits are set the ENGCLK pin is held in the high state. These bits can be dynamically changed without generating spikes on the ENGCLK pin. If ENGCLK is not connected to external circuits, set both bits (disabling ENGCLK) to minimize noise and power dissipation. For measurement purposes the backup clock (BUCLK) can be driven externally on the ENGCLK pin. 00 = Engineering clock enabled, full-strength output buffer 01 = Engineering clock enabled, half-strength output buffer 10 = BUCLK is the output on the ENGCLK full-strength output buffer 11 = Engineering clock disabled
18:23	ENGDIV	Engineering clock division factor. These bits define the frequency division factor between VCO/2 and ENGCLK. The divider ratio is ENGDIV+1. Division factor can be from 1 (ENGDIV = 0b000000) to 64 (ENGDIV = 0b111111). These bits can be read and written at any time. They are not affected by hard reset but are cleared during power-on reset. NOTE: If the engineering clock division factor is not a power of two, synchronization between the system and ENGCLK is not guaranteed. NOTE: The default (Power On Reset) value of ENGDIV will be 0b111111 on all mask sets after K62N. The default for previous mask sets (J76N, K02A, and K83H) is 0b000001.

Table 8-9 SCCR Bit Descriptions (Continued)

Bit(s)	Name	Description
24	—	Reserved
25:27	DFNL	Division factor low frequency. The user can load these bits with the desired divide value and the CSRC bit to change the frequency. Changing the value of these bits does not result in a loss of lock condition. These bits are cleared by power-on or hard reset. Refer to **8.6.1 General System Clocks** and **Figure 8-5** for details on using these bits. 000 = Divide by 2 001 = Divide by 4 010 = Divide by 8 011 = Divide by 16 100 = Divide by 32 101 = Divide by 64 110 = Reserved 111 = Divide by 256
28	—	Reserved
29:31	DFNH	Division factor high frequency. These bits determine the general system clock frequency during normal mode. Changing the value of these bits does not result in a loss of lock condition. These bits are cleared by power-on or hard reset. The user can load these bits at any time to change the general system clock rate. Note that the GCLKs generated by this division factor are not 50% duty cycle (i.e. CLKOUT). 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 101 = Divide by 32 110 = Divide by 64 111 = Reserved

8.12.2 PLL, Low-Power, and Reset-Control Register (PLPRCR)

The PLL, low-power, and reset-control register (PLPRCR) is a 32-bit register powered by the keep alive power supply.

PLPRCR — PLL, Low-Power, and Reset-Control Register **0x2F C284**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MF												RE-SERVED	LOCS	LOC-SS	SPLS
POWER-ON RESET:															
0 OR 4												0	0	0	0
HARD RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SPLSS	TEXPS	RE-SERVED	TMIST	RE-SERVED	CSRC	LPM		CSR	LOL-RE	RE-SERVED	DIVF				
POWER-ON RESET:															
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HARD RESET:															
U	1	U	0	U	0	0	0	U	U		U	U	U	U	U

U = Unaffected by reset

Table 8-10 PLPRCR Bit Descriptions

Bit(s)	Name	Description
0:11	MF	Multiplication factor bits. The output of the VCO is divided to generate the feedback signal to the phase comparator. The MF bits control the value of the divider in the SPLL feedback loop. The phase comparator determines the phase shift between the feedback signal and the reference clock. This difference results in either an increase or decrease in the VCO output frequency. The MF bits can be read and written at any time. However, this field can be write-protected by setting the MF and pre-divider lock (MFPDL) bit in the SCCR. Changing the MF bits causes the SPLL to lose lock. Also, the MF field should not be modified when entering or exiting from low power mode (LPM change), or when back-up clock is active. The normal reset value for the DFNH bits is zero (divide by one). When the PLL is operating in one-to-one mode, the multiplication factor is set to x1 (MF = 0).
12	—	Reserved
13	LOCS	Loss of clock status. When the oscillator or external clock source is not at the minimum frequency, the loss-of-clock circuit asserts the LOCS bit. This bit is cleared when the oscillator or external clock source is functioning normally. This bit is reset only on power-on reset. Writes to this bit have no effect. 0 = No loss of oscillator is currently detected 1 = Loss of oscillator is currently detected
14	LOCSS	Loss of clock sticky. If, after negation of $\overline{\text{PORESET}}$, the loss-of-clock circuit detects that the oscillator or external clock source is not at a minimum frequency, the LOCSS bit is set. LOCSS remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. The reset value is determined during hard reset. The STBUC bit will be set provided the PLL lock condition is not met when $\overline{\text{HRESET}}$ is asserted, and cleared if the PLL is locked when $\overline{\text{HRESET}}$ is asserted. 0 = No loss of oscillator has been detected 1 = Loss of oscillator has been detected

Table 8-10 PLPRCR Bit Descriptions (Continued)

Bit(s)	Name	Description
15	SPLS	System PLL lock status bit 0 = SPLL is currently not locked 1 = SPLL is currently locked
16	SPLSS	SPLL lock status sticky bit. An out-of-lock sets the SPLSS bit. The bit remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. The bit is cleared at power-on reset. This bit is not affected due to a software initiated loss-of-lock (MF change and entering deep-sleep or power-down mode). The SPLSS bit is not affected by hard reset. 0 = SPLL has remained in lock 1 = SPLL has gone out of lock at least once (not due to software-initiated loss of lock)
17	TEXPS	Timer expired status bit. This bit controls whether the chip negates the TEXP pin in deep-sleep mode, thus enabling external circuitry to switch off the VDD (power-down mode). When LPM = 11, CSRC = 0, and TEXPS is high, the TEXP pin remains asserted. When LPM = 11, CSRC = 0, and TEXPS is low, the TEXPS pin is negated. To enable automatic wake-up TEXPS is asserted when one of the following occurs: • The PIT is expired • The real-time clock alarm is set • The time base clock alarm is set • The decrementer exception occurs The bit remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. TEXPS is set by power-on or hard reset. 0 = TEXP is negated in deep-sleep mode 1 = TEXP pin remains asserted always
18	—	Reserved
19	TMIST	Timers interrupt status.TMIST is set when an interrupt from the RTC, PIT, TB or DEC occurs. The TMIST bit is cleared by writing a one to it. Writing a zero has no effect on this bit. The system clock frequency remains at its high frequency value (defined by DFNH) if the TMIST bit is set, even if the CSRC bit in the PLPRCR is set (DFNL enabled) and conditions to switch to normal-low mode do not exist. This bit is cleared during power-on or hard reset. 0 = No timer expired event was detected 1 = A timer expire event was detected
20	—	Reserved
21	CSRC	Clock source. This bit is cleared at hard reset. 0 = General system clock is determined by the DFNH value 1 = General system clock is determined by the DFNL value
22:23	LPM	Low-power mode select. These bits are encoded to provide one normal operating mode and four low-power modes. In normal and doze modes, the system can be in high state (frequency determined by the DFNH bits) or low state (frequency defined by the DFNL bits). The LPM field can be write-protected by setting the LPM and CSRC lock (LPML) bit in the PLPRCR Refer to **Table 8-4** and **Table 8-5**.
24	CSR	Checkstop reset enable. If this bit is set, then an automatic reset is generated when the RCPU signals that it has entered checkstop mode, unless debug mode was enabled at reset. If the bit is clear and debug mode is not enabled, then the USIU will not do anything upon receiving the checkstop signal from the RCPU. If debug mode is enabled, then the part enters debug mode upon entering checkstop mode. In this case, the RCPU will not assert the checkstop signal to the reset circuitry. This bit is writable once after soft reset. 0 = No reset will occur when checkstop is asserted 1 = Reset will occur when checkstop is asserted
25	LOLRE	Loss of lock reset enable 0 = Loss of lock does not cause $\overline{\text{HRESET}}$ assertion 1 = Loss of lock causes $\overline{\text{HRESET}}$ assertion Note: if limp mode is enabled, use the COLIR feature instead of setting the LOLRE bit. See **8.12.3 Change of Lock Interrupt Register (COLIR)**.

Table 8-10 PLPRCR Bit Descriptions (Continued)

Bit(s)	Name	Description
26	—	Reserved
27:31	DIVF	The DIVF bits control the value of the pre-divider in the SPLL circuit. The DIVF bits can be read and written at any time. However, the DIVF field can be write-protected by setting the MF and pre-divider lock (MFPDL) bit in the SCCR. Changing the DIVF bits causes the SPLL to lose lock.

8.12.3 Change of Lock Interrupt Register (COLIR)

The COLIR is 16-bit read/write register. It controls the change of lock interrupt generation, and is used for reporting a loss of lock interrupt source. It contains the interrupt request level and the interrupt status bit. This register is readable and writable at any time. A status bit is cleared by writing a one (writing a zero does not affect a status bit's value). The COLIR is memory mapped into the MPC555 / MPC556 USIU register map.

COLIR — Change of Lock Interrupt Register **0x2F C28C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
COLIRQ								COLIS	Re-served	COLIE	Reserved				
RESET:															
0	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U

U = Unaffected by reset

Table 8-11 COLIR Bit Descriptions

Bit(s)	Name	Description
0:7	COLIRQ	Change of lock interrupt request level. These bits determine the interrupt priority level of the change of lock. To specify certain level, the appropriate one of these bits should be set.
8	COLIS	If set ("one"), the bit indicates that a change in the PLL lock status was detected. The PLL was locked and lost lock, or the PLL was unlocked and got locked. The bit should be cleared by writing a one.
9	—	Reserved
10	COLIE	Change of Lock Interrupt enable. If COLIE bit is asserted, an interrupt will generate when the COLIS bit is asserted. 0 = Change of lock Interrupt disable 1 = Change of lock Interrupt enable
10:15	—	Reserved

8.12.4 VDDSRAM Control Register (VSRMCR)

This register contains control bits for enabling or disabling the VDDSRAM supply detection circuit. There are also four bits that indicate the failure detection. All four bits have the same function and are required to improve the detection capability in extreme cases.

VSRMCR — VDDSRAM Control Register **0x2F C290**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	LVSRS				VS-RDE	RESERVED									
	HARD RESET:														
	U	U	U	U	0										

U = Unaffected by reset

Table 8-12 VSRMCR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:4	LVSRS	Loss of VDDSRAM sticky. These status bits indicate whether a VDDSRAM supply failure occurred. In addition, when the power is turned on for the first time, VDDSRAM rises and these bits are set. The LVSRS bits are cleared by writing them to ones. A write of zero has no effect on these bits. 0 = No VDDSRAM supply failure was detected 1 = VDDSRAM supply failure was detected
5	VSRDE	VDDSRAM detector disable. 0 = VDDSRAM detection circuit is enabled 1 = VDDSRAM detection circuit is disabled
6:15	—	Reserved

SECTION 9
EXTERNAL BUS INTERFACE

The MPC555 / MPC556 bus is a synchronous, burstable bus. Signals driven on this bus are required to make the setup and hold time relative to the bus clock's rising edge. The bus has the ability to support multiple masters. The MPC555 / MPC556 architecture supports byte, half-word, and word operands allowing access to 8-, 16-, and 32-bit data ports through the use of synchronous cycles controlled by the size outputs (TSIZ0, TSIZ1). For accesses to 16- and 8-bit ports, the slave must be controlled by the memory controller.

9.1 Features

The external bus interface features are listed below.

- 32-bit address bus with transfer size indication (only 24 available on pins)
- 32-bit data bus
- Bus arbitration logic on-chip supports an external master
- Internal chip-select and wait state generation to support peripheral or static memory devices through the memory controller
- Supports various memory (SRAM, EEPROM) types: synchronous and asynchronous, burstable and non-burstable
- Supports non-wrap bursts
- Flash ROM programming support
- Compatible with PowerPC architecture
- Easy to interface to slave devices
- Bus is synchronous (all signals are referenced to rising edge of bus clock)
- Bus can operate at the same frequency as the MPC555 / MPC556 or half the frequency.

9.2 Bus Transfer Signals

The bus transfers information between the MPC555 / MPC556 and external memory of a peripheral device. External devices can accept or provide 8, 16, and 32 bits in parallel and must follow the handshake protocol described in this section. The maximum number of bits accepted or provided during a bus transfer is defined as the port width.

The MPC555 / MPC556 contains an address bus that specifies the address for the transfer and a data bus that transfers the data. Control signals indicate the beginning and type of the cycle, as well as the address space and size of the transfer. The selected device then controls the length of the cycle with the signal(s) used to terminate the cycle. A strobe signal for the address bus indicates the validity of the address and provides timing information for the data.

The MPC555 / MPC556 bus is synchronous. The bus and control input signals must be timed to setup and hold times relative to the rising edge of the clock. Bus cycles can be completed in two clock cycles.

For all inputs, the MPC555 / MPC556 latches the level of the input during a sample window around the rising edge of the clock signal. This window is illustrated in **Figure 9-1**, where **tsu** and **tho** are the input setup and hold times, respectively. To ensure that an input signal is recognized on a specific falling edge of the clock, that input must be stable during the sample window. If an input makes a transition during the window time period, the level recognized by the MPC555 / MPC556 is not predictable; however, the MPC555 / MPC556 always resolves the latched level to either a logic high or low before using it. In addition to meeting input setup and hold times for deterministic operation, all input signals must obey the protocols described in this section.

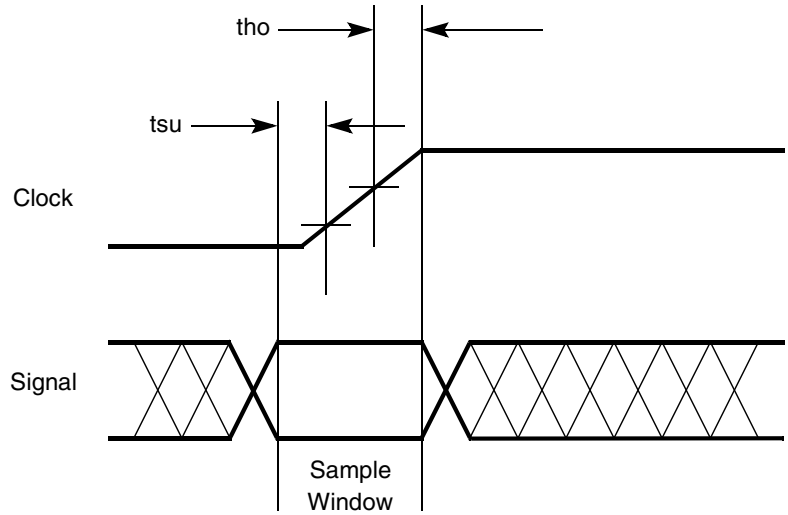


Figure 9-1 Input Sample Window

9.3 Bus Control Signals

The MPC555 / MPC556 initiates a bus cycle by driving the address, size, address type, cycle type, and read/write outputs. At the beginning of a bus cycle, TSIZ0 and TSIZ1 are driven with the address type signals. TSIZ0 and TSIZ1 indicate the number of bytes remaining to be transferred during an operand cycle (consisting of one or more bus cycles). These signals are valid at the rising edge of the clock in which the transfer start ($\overline{TS}$) signal is asserted.

The read/write (RD/$\overline{WR}$) signal determines the direction of the transfer during a bus cycle. Driven at the beginning of a bus cycle, RD/$\overline{WR}$ is valid at the rising edge of the clock in which $\overline{TS}$ is asserted. The logic level of RD/$\overline{WR}$ only changes when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for consecutive write cycles.

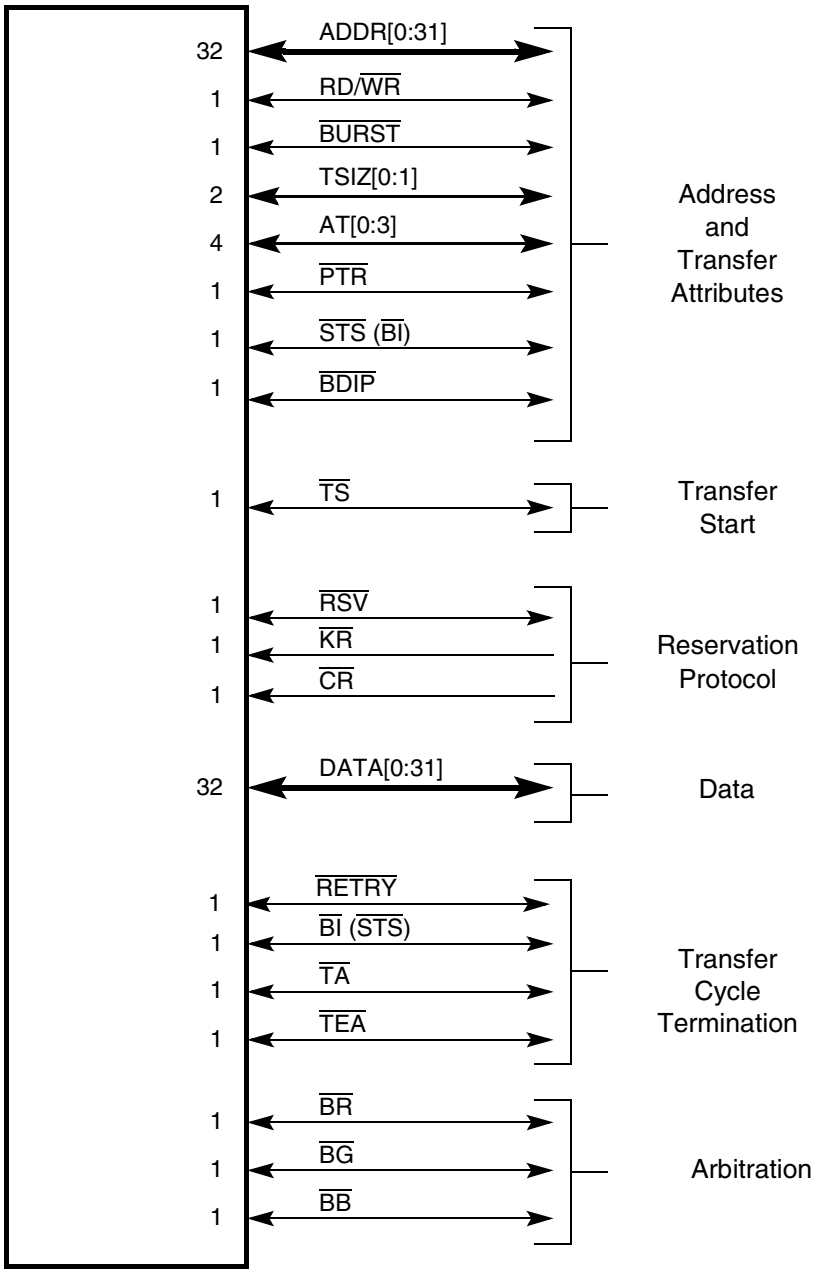


Figure 9-2 MPC555 / MPC556 Bus Signals

9.4 Bus Interface Signal Descriptions

Table 9-1 diatribes each signal in the bus interface unit. More detailed descriptions can be found in subsequent subsections.

Table 9-1 MPC555 / MPC556 SIU Signals

Signal Name	Pins	Active	I/O	Description
Address and Transfer Attributes				
ADDR[0:31] Address bus	24 (8:31)	High	O	Specifies the physical address of the bus transaction.
			I	Driven by an external bus master when it owns the external bus. Only for testing purposes.
RD/$\overline{\text{WR}}$ Read/write	1	HIgh	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Driven high indicates that a read access is in progress. Driven low indicates that a write access is in progress.
			I	Driven by an external master when it owns the external bus. Driven high indicates that a read access is in progress. Driven low indicates that a write access is in progress.
$\overline{\text{BURST}}$ Burst transfer	1	Low	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Driven low indicates that a burst transfer is in progress. Driven high indicates that the current transfer is not a burst.
			I	Driven by an external master when it owns the external bus. Driven low indicates that a burst transfer is in progress. Driven high indicates that the current transfer is not a burst. The MPC555 / MPC556 does not support burst accesses to internal slaves.
TSIZ[0:1] Transfer size	2	High	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Specifies the data transfer size for the transaction.
			I	Driven by an external master when it owns the external bus. Specifies the data transfer size for the transaction.
AT[0:3] Address type	3	High	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.
RSV Reservation transfer	1	Low	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.
PTR Program trace	1	High	O	Driven by the MPC555 / MPC556 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.

Table 9-1 MPC555 / MPC556 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
$\overline{\text{RETRY}}$	1	Low	I	In the case of regular transaction, this signal is driven by the slave device to indicate that the MPC555 / MPC556 must relinquish the ownership of the bus and retry the cycle.
			O	When an external master owns the bus and the internal MPC555 / MPC556 bus initiates access to the external bus at the same time, this signal is used to cause the external master to relinquish the bus for one clock to solve the contention.
$\overline{\text{BDIP}}$ Burst data in progress	1	Low	O	Driven by the MPC555 / MPC556 when it owns the external bus. It is part of the burst protocol. When $\overline{\text{BDIP}}$ is asserted, the second beat in front of the current one is requested by the master. This signal is negated prior to the end of a burst to terminate the burst data phase early.
			I	Driven by an external master when it owns the external bus. When $\overline{\text{BDIP}}$ is asserted, the second beat in front of the current one is requested by the master. This signal is negated prior to the end of a burst to terminate the burst data phase early. The MPC555 / MPC556 does not support burst accesses to internal slaves.
Transfer Start				
$\overline{\text{TS}}$ Transfer start	1	Low	O	Driven by the MPC555 / MPC556 when it owns the external bus. Indicates the start of a transaction on the external bus.
			I	Driven by an external master when it owns the external bus. It indicates the start of a transaction on the external bus or (in show cycle mode) signals the beginning of an internal transaction.
$\overline{\text{STS}}$ Special transfer start	1	Low	O	Driven by the MPC555 / MPC556 when it owns the external bus. Indicates the start of a transaction on the external bus or signals the beginning of an internal transaction in show cycle mode.
Reservation Protocol				
CR Cancel reservation	1	Low	I	Each PowerPC CPU has its own $\overline{\text{CR}}$ signal. Assertion of $\overline{\text{CR}}$ instructs the bus master to clear its reservation; some other master has touched its reserved space. This is a pulsed signal.
$\overline{\text{KR}}$ Kill reservation	1	Low	I	In case of a bus cycle initiated by a STWCX instruction issued by the RCPU to a non-local bus on which the storage reservation has been lost, this signal is used by the non-local bus interface to backoff the cycle. Refer to **9.5.9 Storage Reservation** for details.

Table 9-1 MPC555 / MPC556 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
Data				
DATA[0:31] Data bus	32	High		The data bus has the following byte lane assignments: Data Byte — Byte Lane DATA[0:7] — 0 DATA[8:15] — 1 DATA[16:23] — 2 DATA[24:31] — 3
			O	Driven by the MPC555 / MPC556 when it owns the external bus and it initiated a write transaction to a slave device. For single beat transactions, the byte lanes not selected for the transfer by ADDR[30:31] and TSIZ[0:1] do not supply valid data. In addition, the MPC555 / MPC556 drives DATA[0:31] when an external master owns the external bus and initiated a read transaction to an internal slave module.
			I	Driven by the slave in a read transaction. For single beat transactions, the MPC555 / MPC556 does not sample byte lanes that are not selected for the transfer by ADDR[30:31] and TSIZ[0:1]. In addition, an external master that owns the bus and initiated a write transaction to an internal slave module drives DATA[0:31].
Transfer Cycle Termination				
$\overline{TA}$ Transfer acknowledge	1	LOW	I	Driven by the slave device to which the current transaction was addressed. Indicates that the slave has received the data on the write cycle or returned data on the read cycle. If the transaction is a burst, $\overline{TA}$ should be asserted for each one of the transaction beats.
			O	Driven by the MPC555 / MPC556 when the slave device is controlled by the on-chip memory controller or when an external master initiated a transaction to an internal slave module.
$\overline{TEA}$ Transfer error acknowledge	1	Low	I	Driven by the slave device to which the current transaction was addressed. Indicates that an error condition has occurred during the bus cycle.
			O	Driven by the MPC555 / MPC556 when the internal bus monitor detected an erroneous bus condition, or when an external master initiated a transaction to an internal slave module and an internal error was detected.
$\overline{BI}$ Burst inhibit	1	Low	I	Driven by the slave device to which the current transaction was addressed. Indicates that the current slave does not support burst mode.
			O	Driven by the MPC555 / MPC556 when the slave device is controlled by the on-chip memory controller. the MPC555 / MPC556 also asserts $\overline{BI}$ for any external master burst access to internal MPC555 / MPC556 memory space.

Table 9-1 MPC555 / MPC556 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
ARBITRATION				
$\overline{BR}$ Bus request	1	Low	I	When the internal arbiter is enabled, $\overline{BR}$ assertion indicates that an external master is requesting the bus.
			O	Driven by the MPC555 / MPC556 when the internal arbiter is disabled and the chip is not parked.
$\overline{BG}$ Bus grant	1	Low	O	When the internal arbiter is enabled, the MPC555 / MPC556 asserts this signal to indicate that an external master may assume ownership of the bus and begin a bus transaction. The $\overline{BG}$ signal should be qualified by the master requesting the bus in order to ensure it is the bus owner: Qualified bus grant = $\overline{BG}$ & ~ $\overline{BB}$
			I	When the internal arbiter is disabled, $\overline{BG}$ is sampled and properly qualified by the MPC555 / MPC556 when an external bus transaction is to be executed by the chip.
$\overline{BB}$ Bus busy	1	Low	O	When the internal arbiter is enabled, the MPC555 / MPC556 asserts this signal to indicate that it is the current owner of the bus. When the internal arbiter is disabled, the MPC555 / MPC556 asserts this signal after the external arbiter has granted the ownership of the bus to the chip and it is ready to start the transaction.
			I	When the internal arbiter is enabled, the MPC555 / MPC556 samples this signal to get indication of when the external master ended its bus tenure ($\overline{BB}$ negated). When the internal arbiter is disabled, the $\overline{BB}$ is sampled to properly qualify the $\overline{BG}$ line when an external bus transaction is to be executed by the chip.

9.5 Bus Operations

This section provides a functional description of the system bus, the signals that control it, and the bus cycles provided for data transfer operations. It also describes the error conditions, bus arbitration, and reset operation.

The MPC555 / MPC556 generates a system clock output (CLKOUT). This output sets the frequency of operation for the bus interface directly. Internally, the MPC555 / MPC556 uses a phase-lock loop (PLL) circuit to generate a master clock for all of the CPU circuitry (including the bus interface) which is phase-locked to the CLKOUT output signal.

All signals for the MPC555 / MPC556 bus interface are specified with respect to the rising edge of the external CLKOUT and are guaranteed to be sampled as inputs or changed as outputs with respect to that edge. Since the same clock edge is referenced for driving or sampling the bus signals, the possibility of clock skew could exist between various modules in a system due to routing or the use of multiple clock lines. It

is the responsibility of the system to handle any such clock skew problems that could occur.

9.5.1 Basic Transfer Protocol

The basic transfer protocol defines the sequence of actions that must occur on the MPC555 / MPC556 bus to perform a complete bus transaction. A simplified scheme of the basic transfer protocol is illustrated in **Figure 9-3**.

Arbitration	Address Transfer	Data Transfer	Termination

Figure 9-3 Basic Transfer Protocol

The basic transfer protocol provides for an arbitration phase and an address and data transfer phase. The address phase specifies the address for the transaction and the transfer attributes that describe the transaction. The data phase performs the transfer of data (if any is to be transferred). The data phase may transfer a single beat of data (4 bytes or less) for nonburst operations, a 4-beat burst of data (4 x 4 bytes), an 8-beat burst of data (8 x 2 bytes) or a 16-beat burst of data (16 x 1 bytes).

9.5.2 Single Beat Transfer

During the data transfer phase, the data is transferred from master to slave (in write cycles) or from slave to master (on read cycles).

During a write cycle, the master drives the data as soon as it can, but never earlier than the cycle following the address transfer phase. The master has to take into consideration the “one dead clock cycle” switching between drivers to avoid electrical contentions. The master can stop driving the data bus as soon as it samples the $\overline{TA}$ line asserted on the rising edge of the CLKOUT.

During a read cycle, the master accepts the data bus contents as valid at the rising edge of the CLKOUT in which the $\overline{TA}$ signal is sampled/asserted.

9.5.2.1 Single Beat Read Flow

The basic read cycle begins with a bus arbitration, followed by the address transfer, then the data transfer. The handshakes are illustrated in the following flow and timing diagrams as applicable to the fixed transaction protocol.

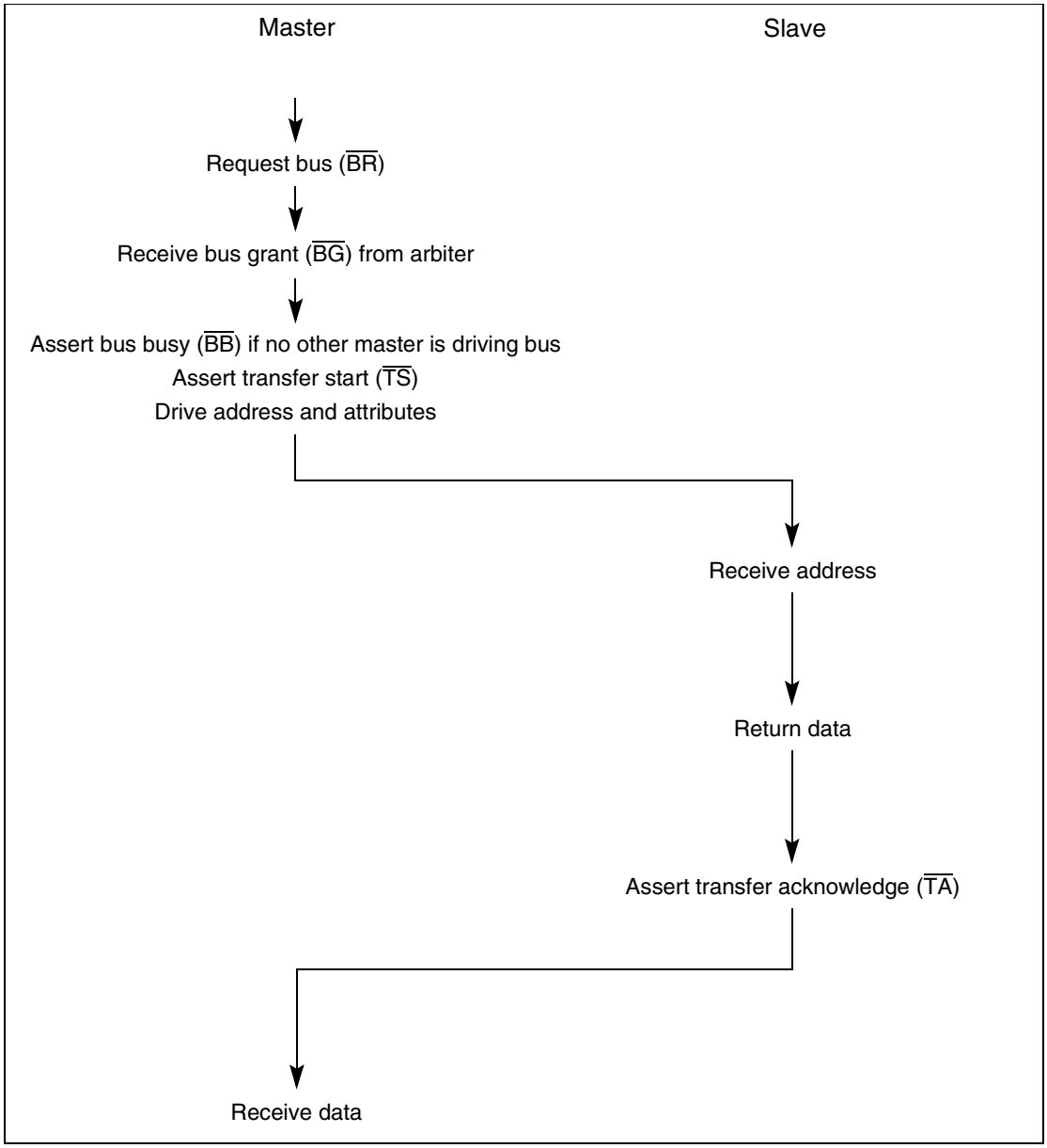


Figure 9-4 Basic Flow Diagram of a Single Beat Read Cycle

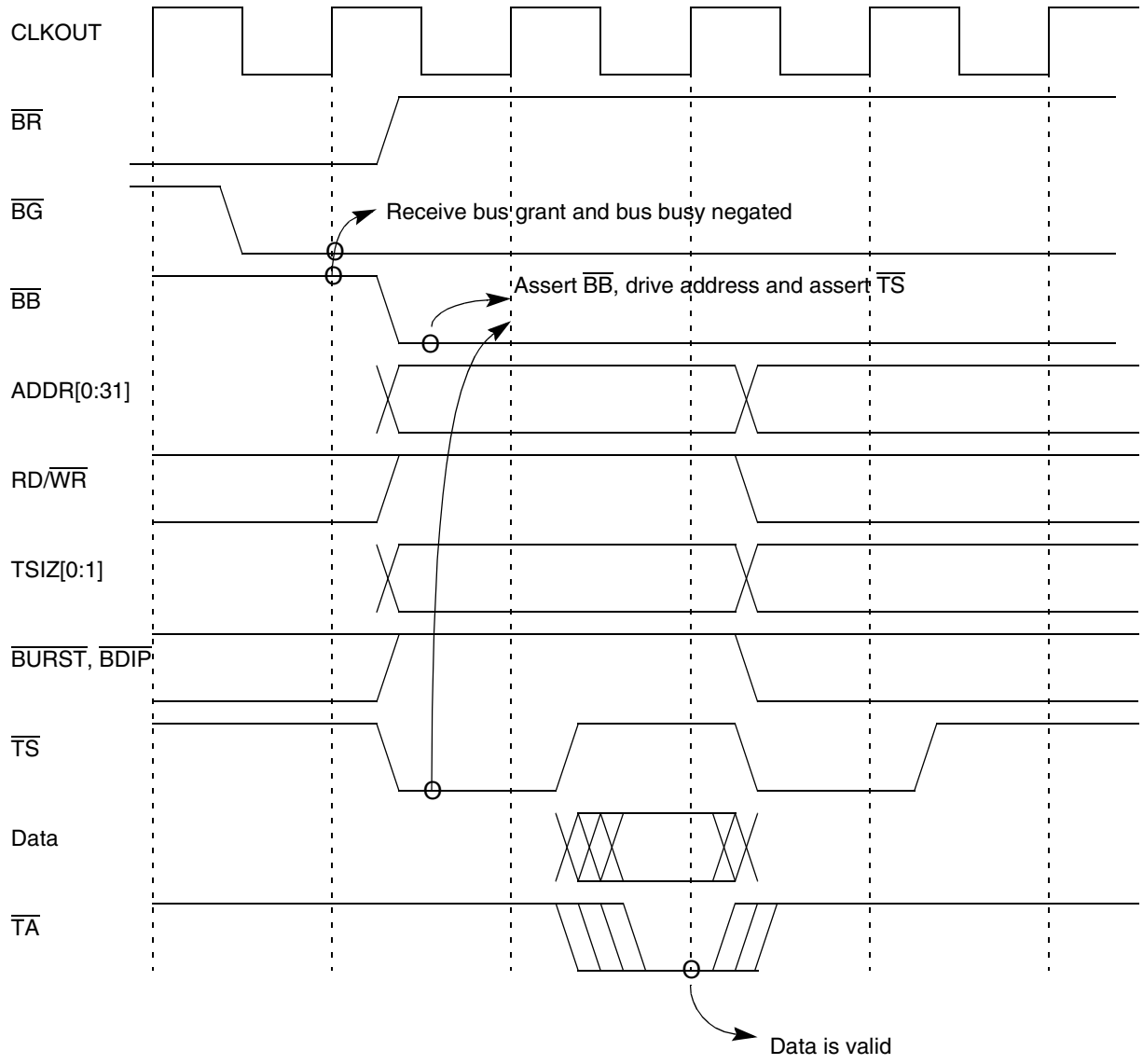


Figure 9-5 Single Beat Read Cycle–Basic Timing–Zero Wait States

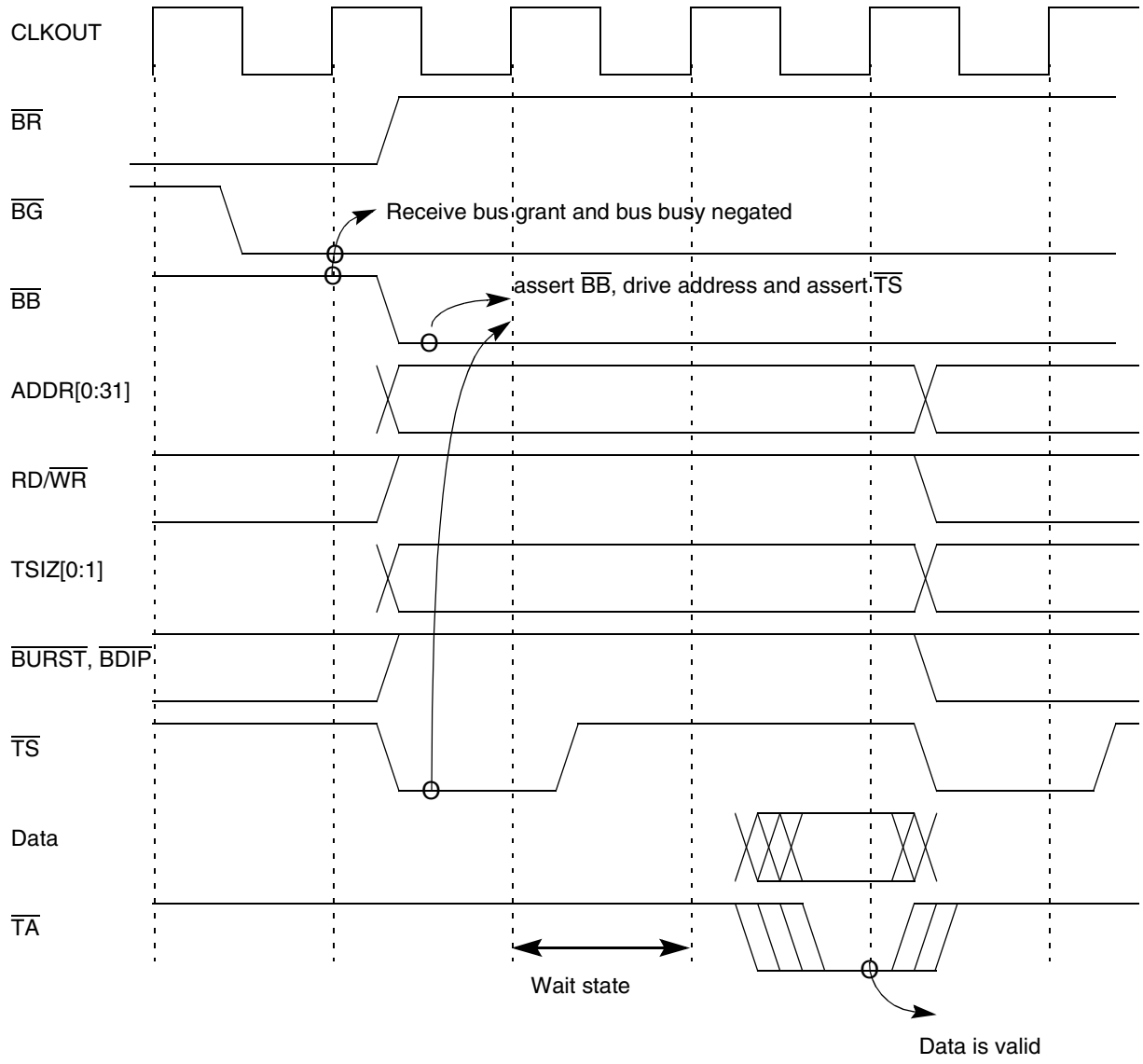


Figure 9-6 Single Beat Read Cycle–Basic Timing–One Wait State

9.5.2.2 Single Beat Write Flow

The basic write cycle begins with a bus arbitration, followed by the address transfer, then the data transfer. The handshakes are illustrated in the following flow and timing diagrams as applicable to the fixed transaction protocol.

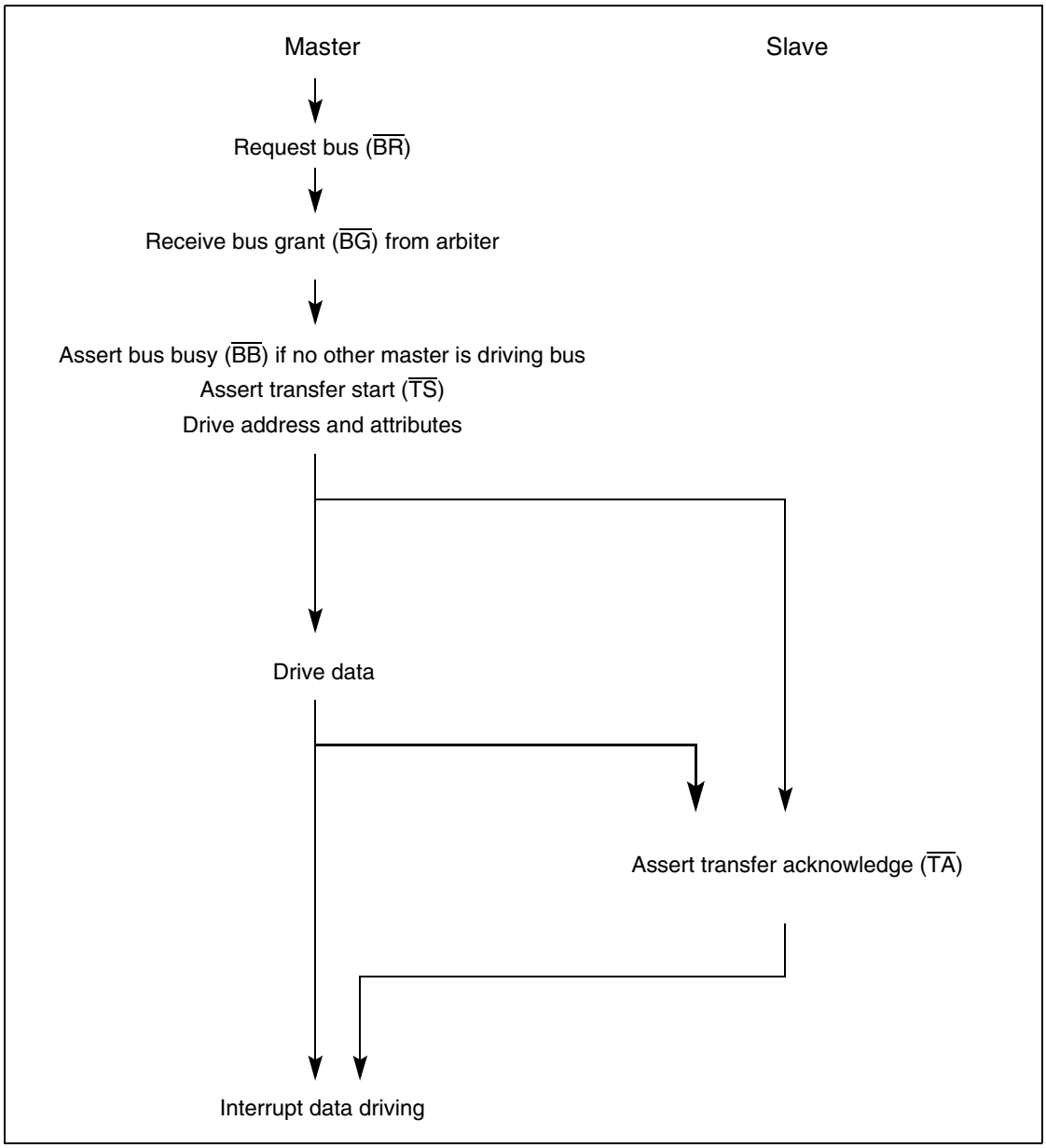


Figure 9-7 Basic Flow Diagram of a Single Beat Write Cycle

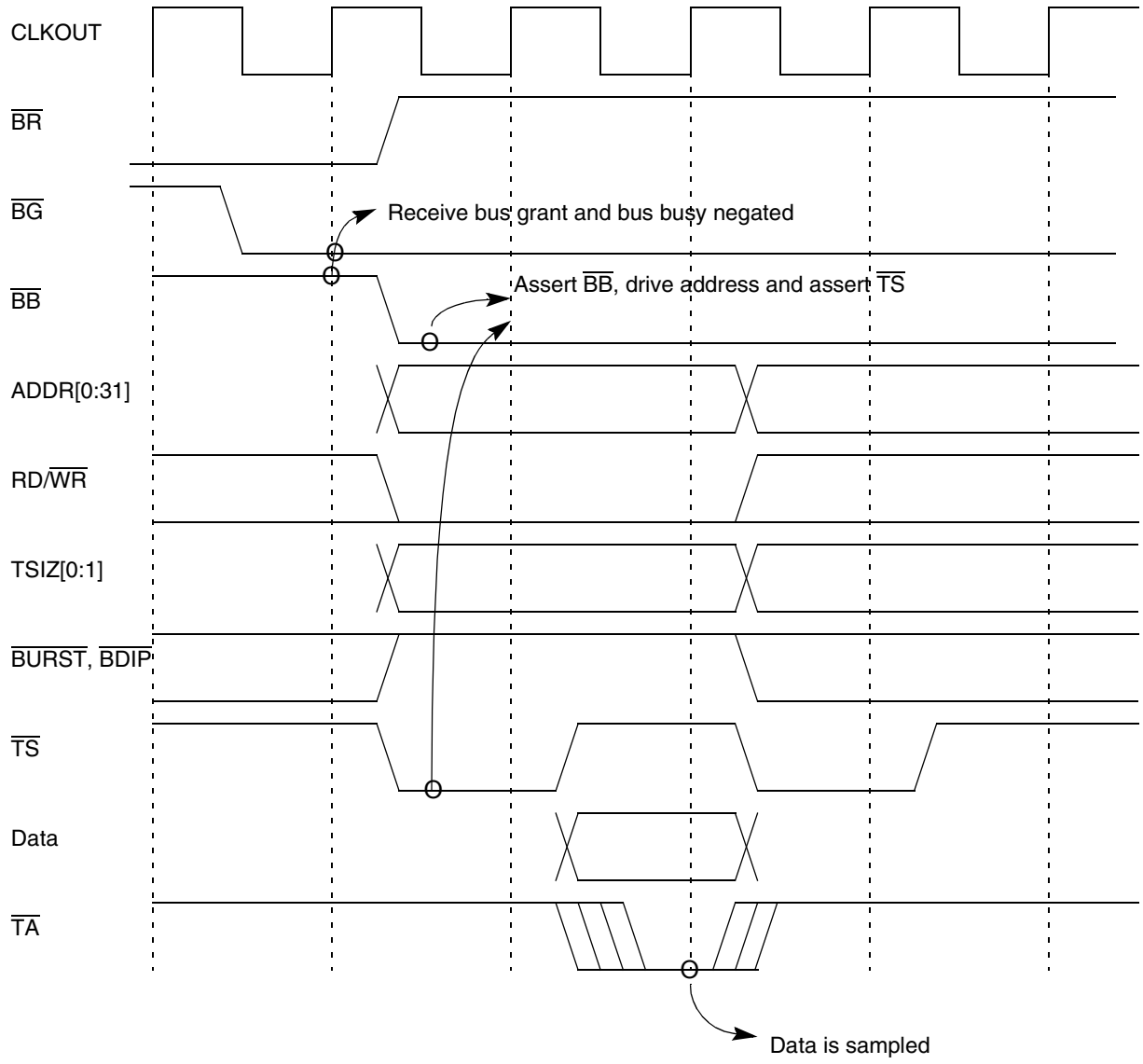


Figure 9-8 Single Beat Basic Write Cycle Timing, Zero Wait States

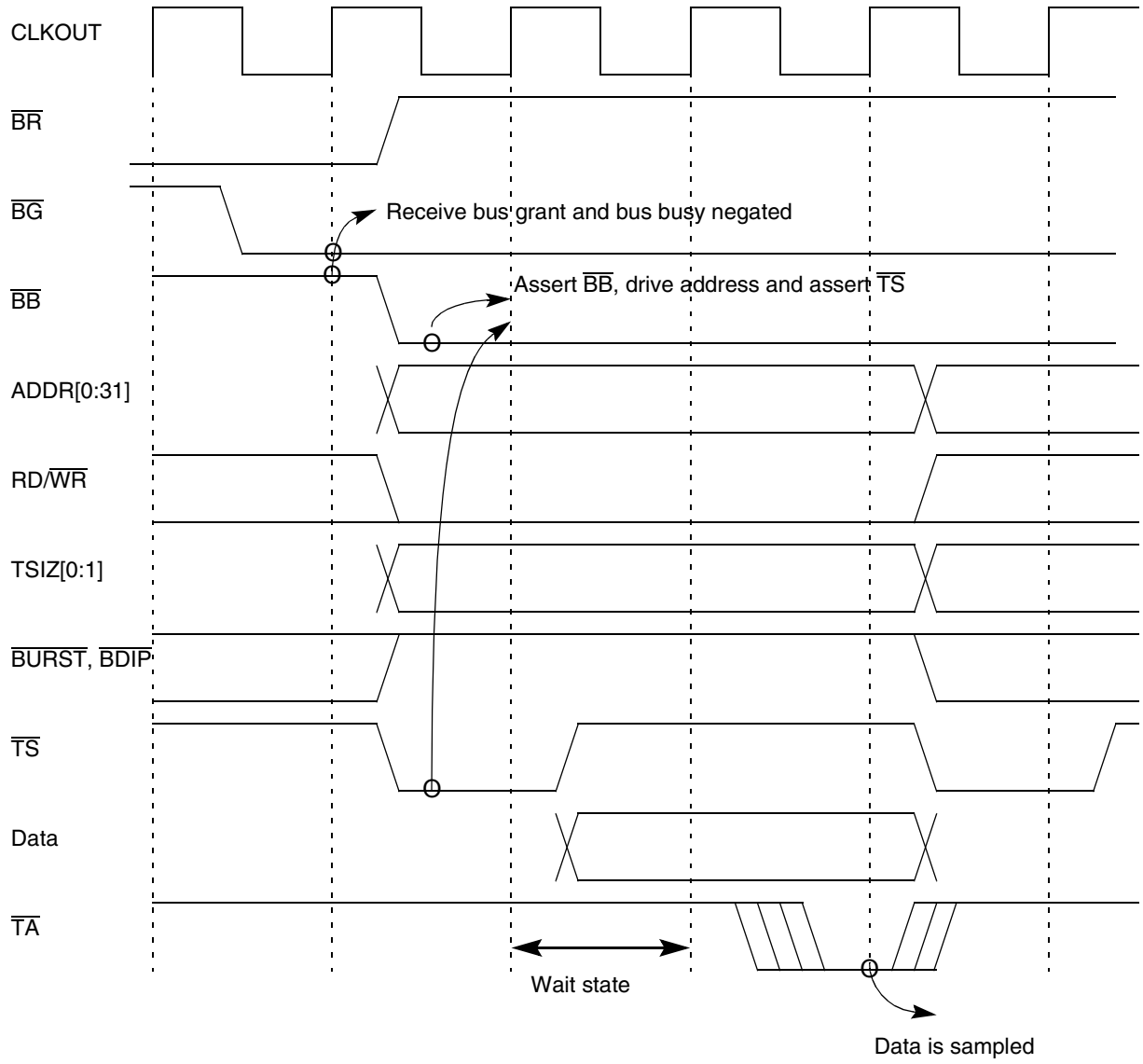


Figure 9-9 Single Beat Basic Write Cycle Timing, One Wait State

9.5.2.3 Single Beat Flow with Small Port Size

The general case of single beat transfers assumes that the external memory has a 32-bit port size. The MPC555 / MPC556 provides an effective mechanism for interfacing with 16-bit and 8-bit port size memories, allowing transfers to these devices when they are controlled by the internal memory controller.

In this case, the MPC555 / MPC556 attempts to initiate a transfer as in the normal case. If the bus interface receives a small port size (16 or 8 bits) indication before the transfer acknowledge to the first beat (through the internal memory controller), the MCU initiates successive transactions until the completion of the data transfer. Note that all the transactions initiated to complete the data transfer are considered to be part of an atomic transaction, so the MCU does not allow other unrelated master accesses

or bus arbitration to intervene between the transfers. If any of the transactions except the first is re-tried during an access to a small port, then an exception is generated to the RCPU.

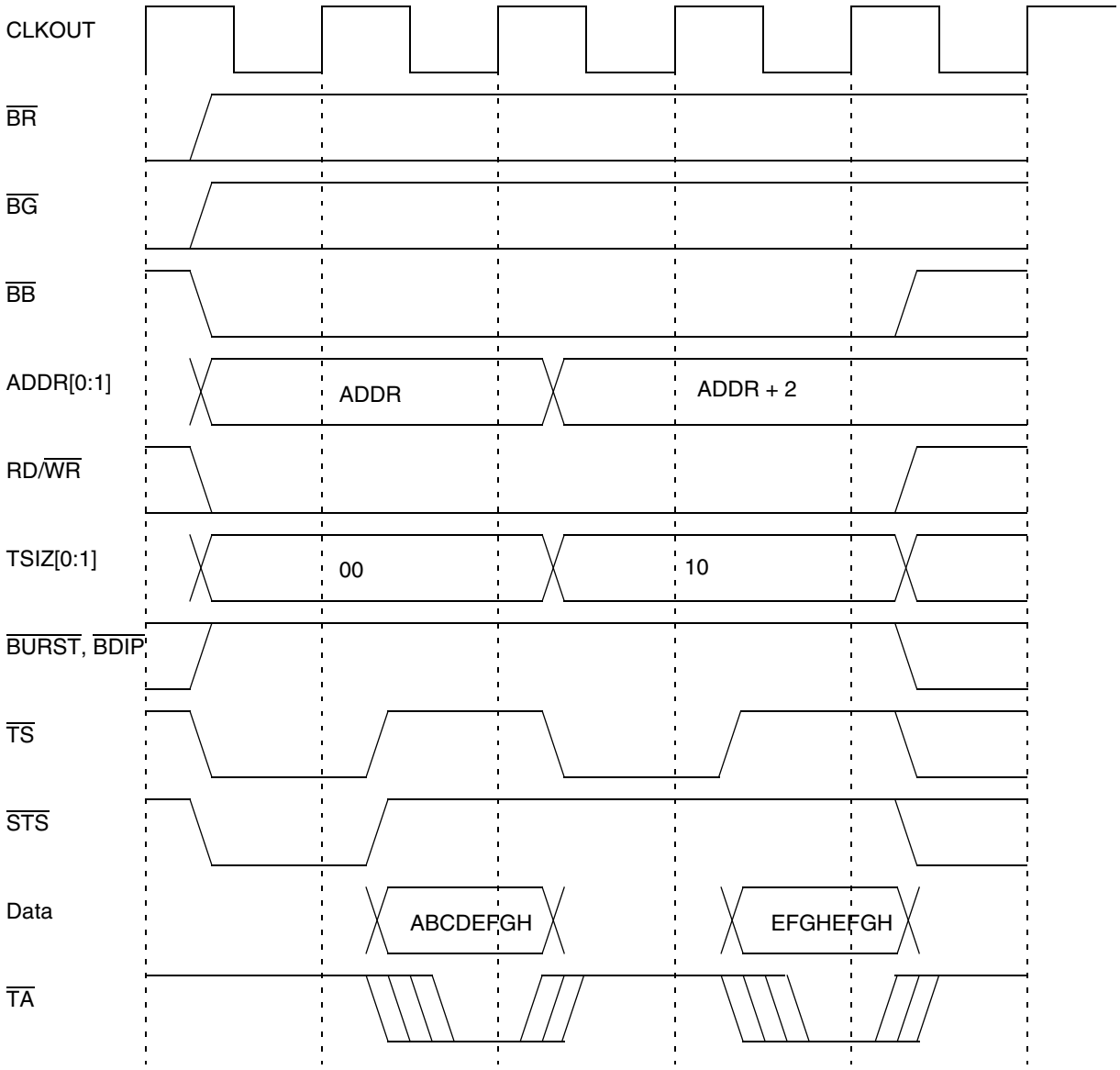


Figure 9-10 Single Beat 32-Bit Data Write Cycle Timing, 16 Bit-Port Size

9.5.3 Burst Transfer

The MPC555 / MPC556 uses non-wrapping burst transfers to access operands of up to 16 bytes (four words). A non-wrapping burst access stops accessing the external device when the word address is modulo four. The MPC555 / MPC556 begins the access by supplying a starting address that points to one of the words and requiring the memory device to sequentially drive or sample each word on the data bus. The select-

ed slave device must internally increment ADDR28 and ADDR29 (and ADDR30 in the case of a 16-bit port slave device, and also ADDR31 in the case of an 8-bit port slave device) of the supplied address for each transfer, causing the address to reach a four-word boundary, and then stop. The address and transfer attributes supplied by the MPC555 / MPC556 remain stable during the transfers. The selected device terminates each transfer by driving or sampling the word on the data bus and asserting $\overline{TA}$.

The MPC555 / MPC556 also supports burst-inhibited transfers for slave devices that are unable to support bursting. For this type of bus cycle, the selected slave device supplies or samples the first word the MPC555 / MPC556 points to and asserts the burst-inhibit signal with $\overline{TA}$ for the first transfer of the burst access. The MPC555 / MPC556 responds by terminating the burst and accessing the remainder of the 16-byte block. These remaining accesses use up to three read/write bus cycles (each one for a word) in the case of a 32-bit port width slave, up to seven read/write bus cycles in the case of a 16-bit port width slave, or up to fifteen read/write bus cycles in the case of a 8-bit port width slave.

The general case of burst transfers assumes that the external memory has a 32-bit port size. The MPC555 / MPC556 provides an effective mechanism for interfacing with 16-bit port size memories and 8-bit port size memories allowing bursts transfers to these devices when they are controlled by the internal memory controller.

In this case, the MPC555 / MPC556 attempts to initiate a burst transfer as in the normal case. If the memory controller signals to the bus interface that the external device has a small port size (8 or 16 bits), and if the burst is accepted, the bus interface completes a burst of 8 or 16 beats.

Each of the data beats of the burst transfers effectively only one or two bytes. Note that this burst of 8 or 16 beats is considered an atomic transaction, so the MPC555 / MPC556 does not allow other unrelated master accesses or bus arbitration to intervene between the transfers.

9.5.4 Burst Mechanism

In addition to the standard bus signals, the MPC555 / MPC556 burst mechanism uses the following signals:

- The $\overline{BURST}$ signal indicates that the cycle is a burst cycle.
- The burst data in progress ($\overline{BDIP}$) signal indicates the duration of the burst data.
- The burst inhibit ($\overline{BI}$) signal indicates whether the slave is burstable.

At the start of the burst transfer, the master drives the address, the address attributes, and the $\overline{BURST}$ signal to indicate that a burst transfer is being initiated, and asserts $\overline{TS}$. If the slave is burstable, it negates the burst-inhibit ($\overline{BI}$) signal. If the slave cannot burst, it asserts $\overline{BI}$.

During the data phase of a burst write cycle the master drives the data. It also asserts $\overline{BDIP}$ if it intends to drive the data beat following the current data beat. When the slave has received the data, it asserts the signal transfer acknowledge to indicate to the master that it is ready for the next data transfer. The master again drives the next data

and asserts or negates the $\overline{BDIP}$ signal. If the master does not intend to drive another data beat following the current one, it negates $\overline{BDIP}$ to indicate to the slave that the next data beat transfer is the last data of the burst write transfer.

$\overline{BDIP}$ has two basic timings: normal and late (see **Figure 9-13** and **Figure 9-14**). In the late timing mode, assertion of $\overline{BDIP}$ is delayed by the number of wait states in the first data beat. This implies that for zero-wait-state cycles, $\overline{BDIP}$ assertion time is identical in normal and late modes. Cycles with late $\overline{BDIP}$ generation can occur only during cycles for which the memory controller generates $\overline{TA}$ internally. Refer to **SECTION 10 MEMORY CONTROLLER** for more information.

In the MPC555 / MPC556, no internal master initiates write bursts. The MPC555 / MPC556 is designed to perform this kind of transaction in order to support an external master that is using the memory controller services. Refer to **10.7 Memory Controller External Master Support**.

During the data phase of a burst read cycle, the master receives data from the addressed slave. If the master needs more than one data beat, it asserts $\overline{BDIP}$. Upon receiving the second-to-last data beat, the master negates $\overline{BDIP}$. The slave stops driving new data after it receives the negation of the $\overline{BDIP}$ signal at the rising edge of the clock.

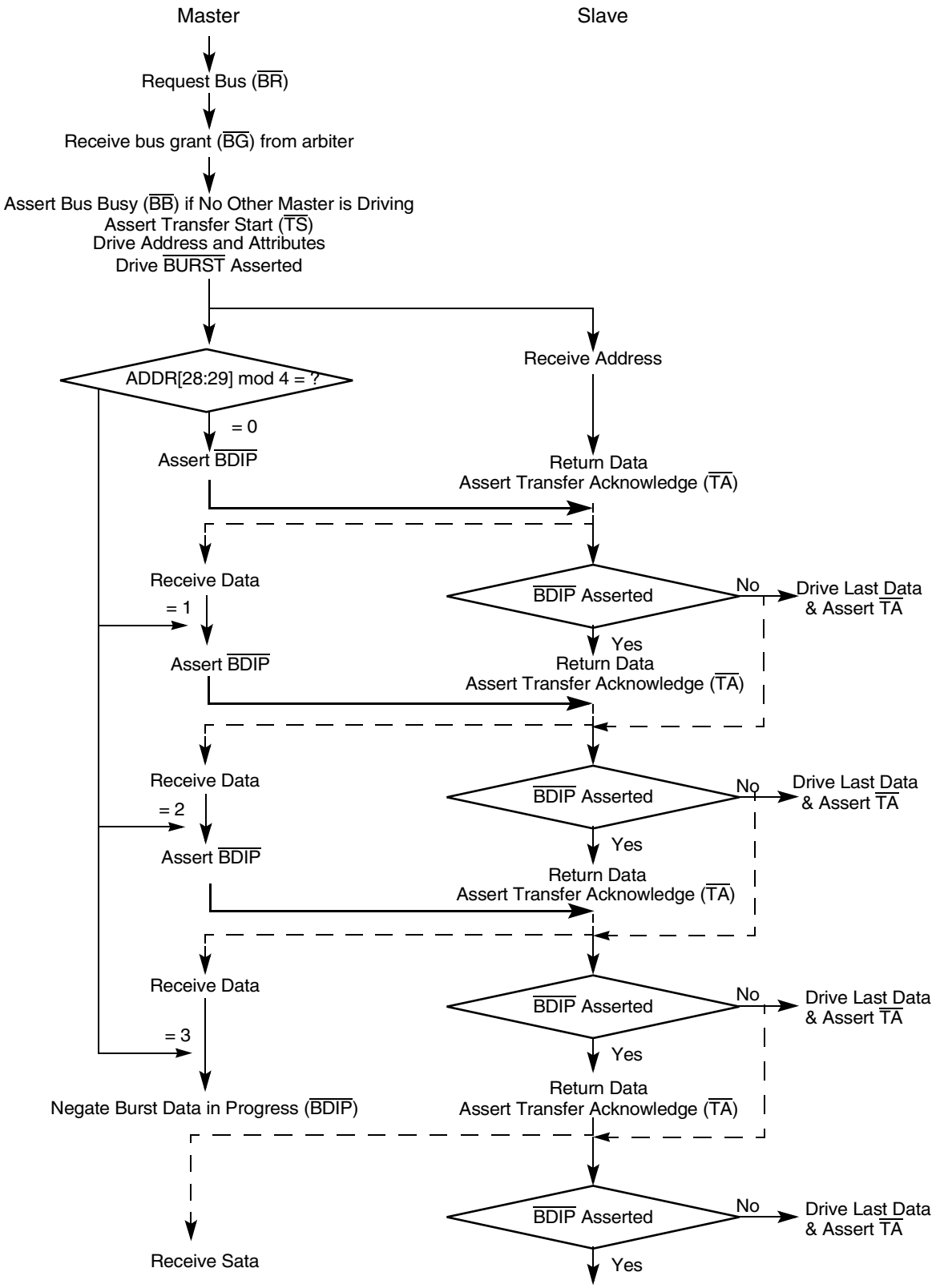


Figure 9-11 Basic Flow Diagram Of A Burst Read Cycle

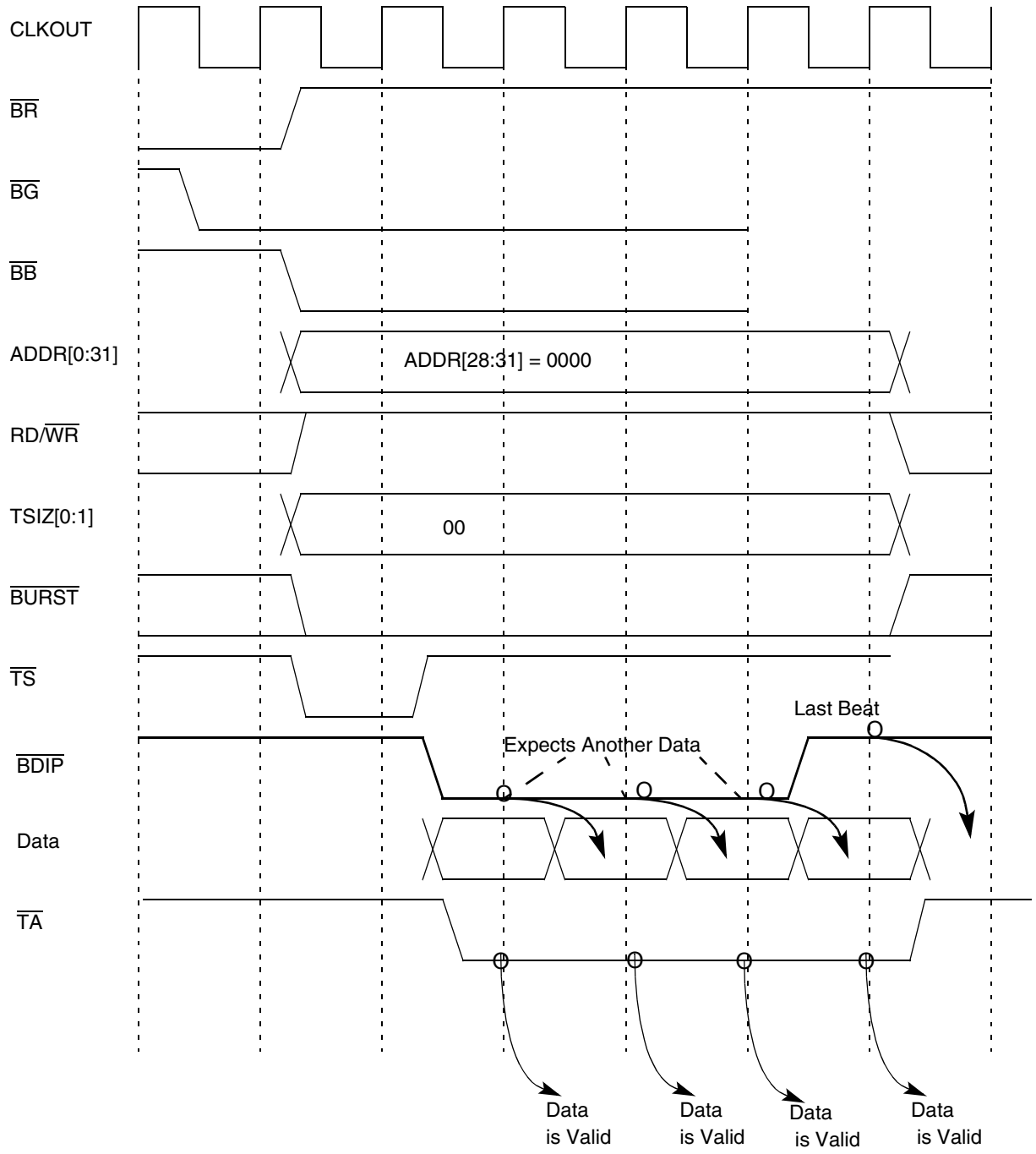


Figure 9-12 Burst-Read Cycle–32-Bit Port Size–Zero Wait State

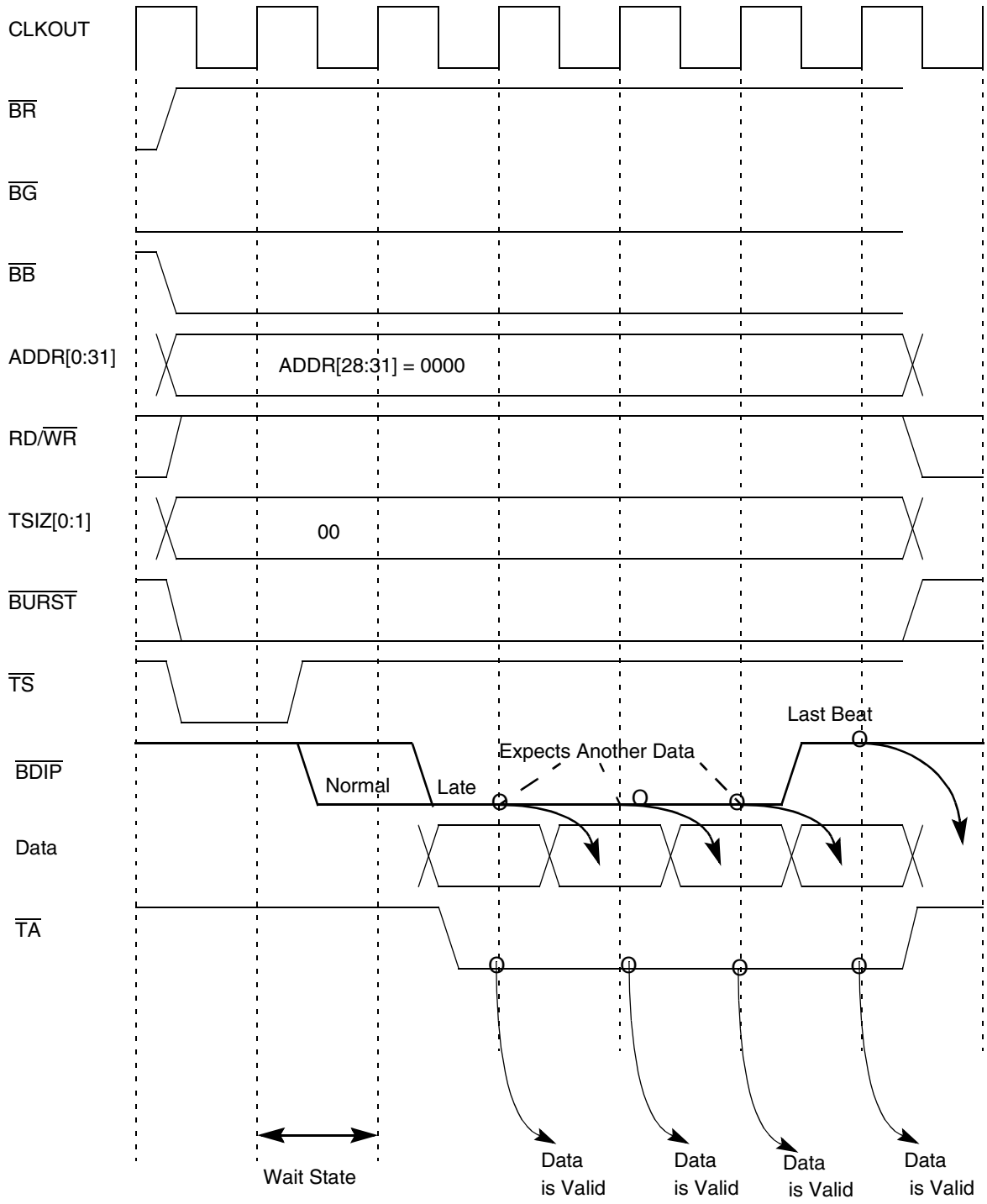


Figure 9-13 Burst-Read Cycle–32-Bit Port Size–One Wait State

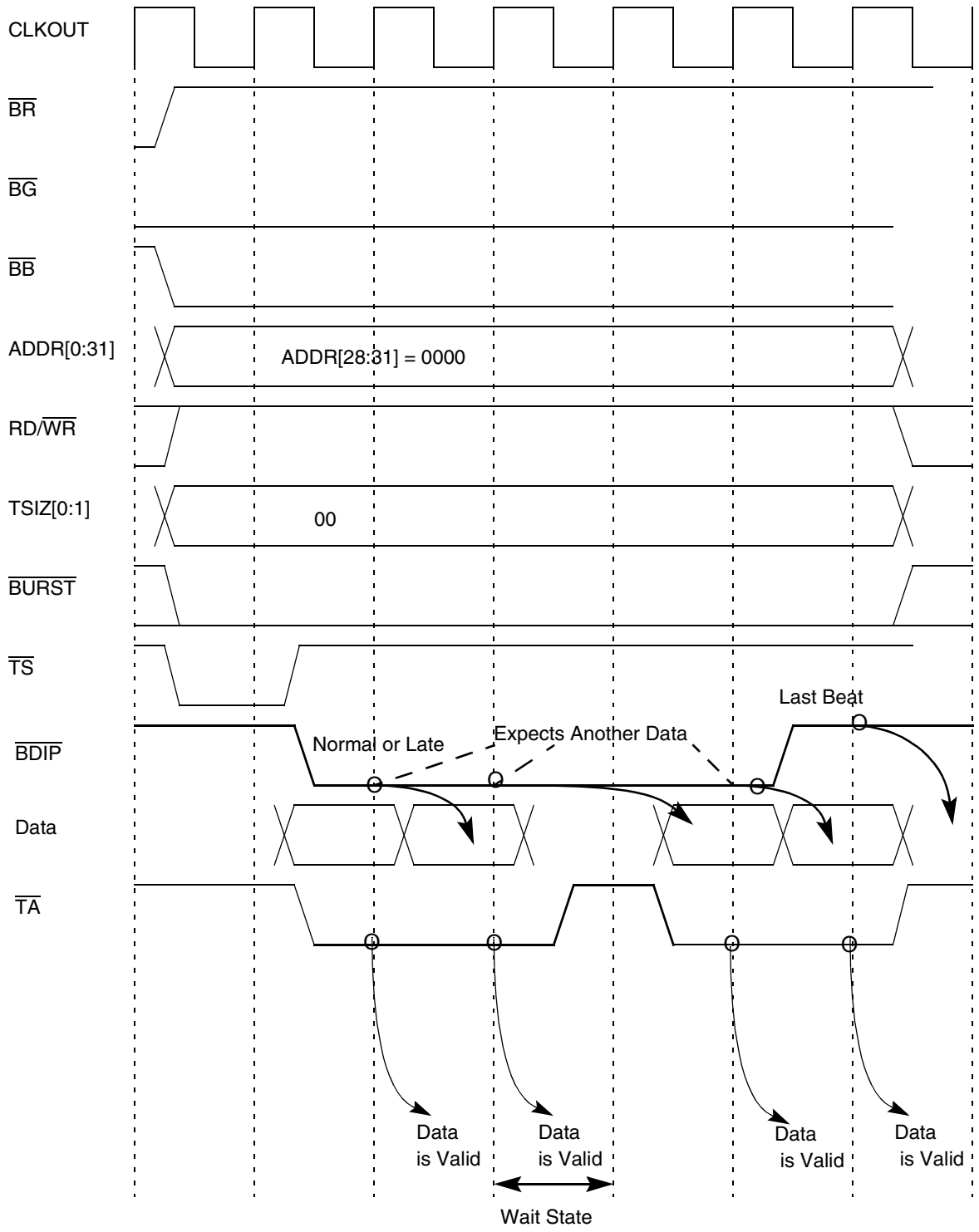


Figure 9-14 Burst-Read Cycle–32-Bit Port Size–Wait States Between Beats

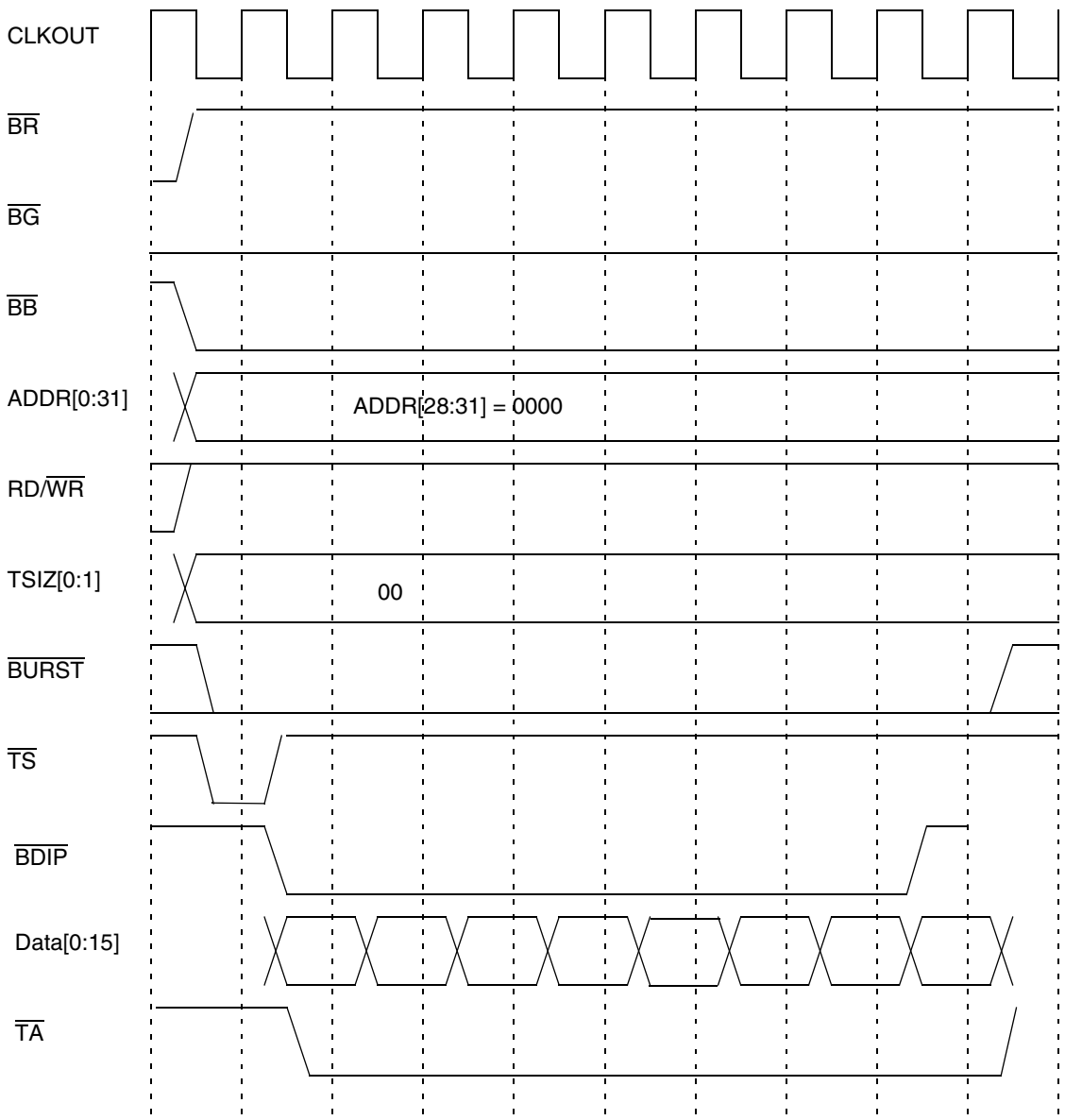


Figure 9-15 Burst-Read Cycle, 16-Bit Port Size

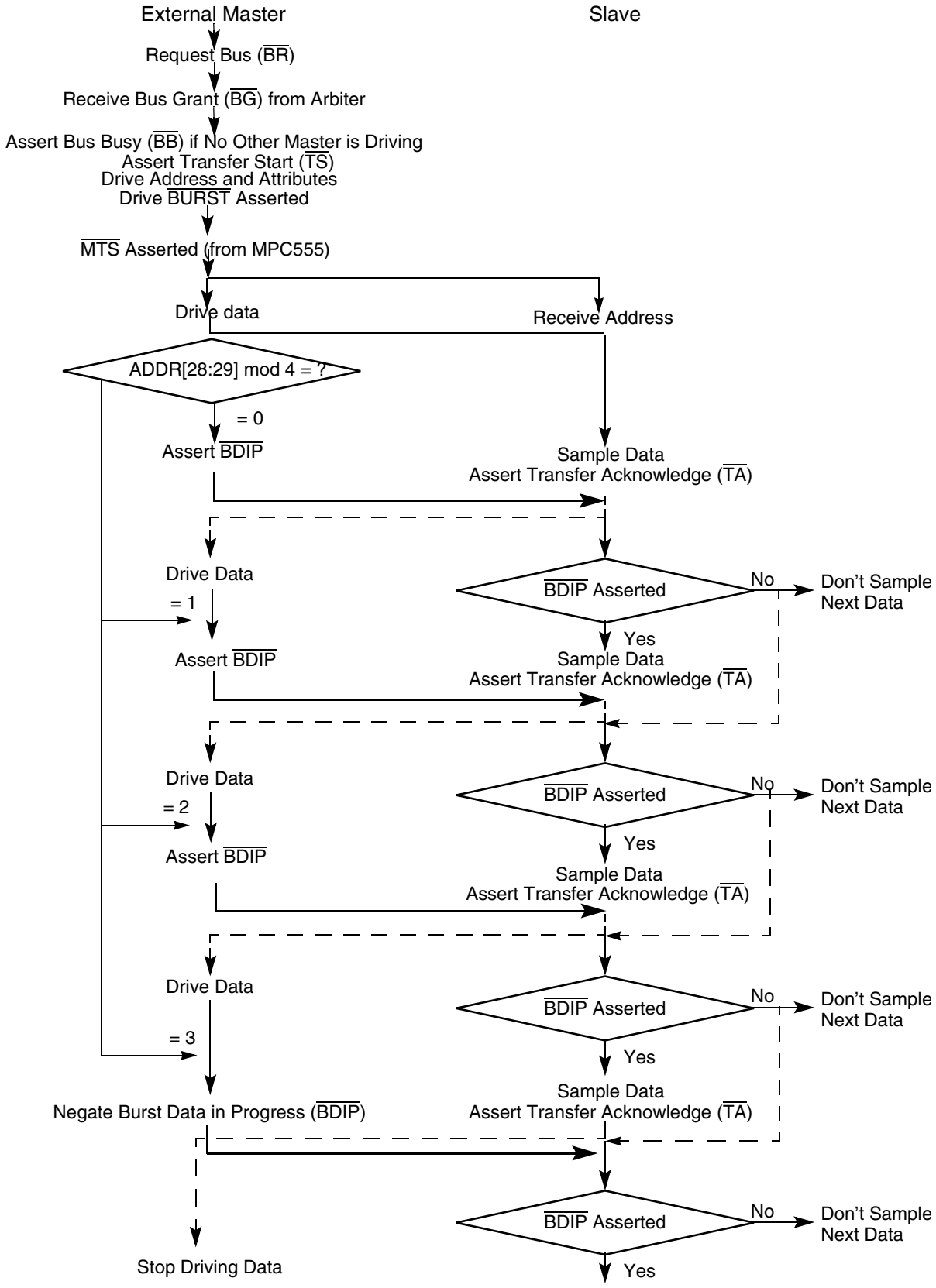


Figure 9-16 Basic Flow Diagram of a Burst Write Cycle

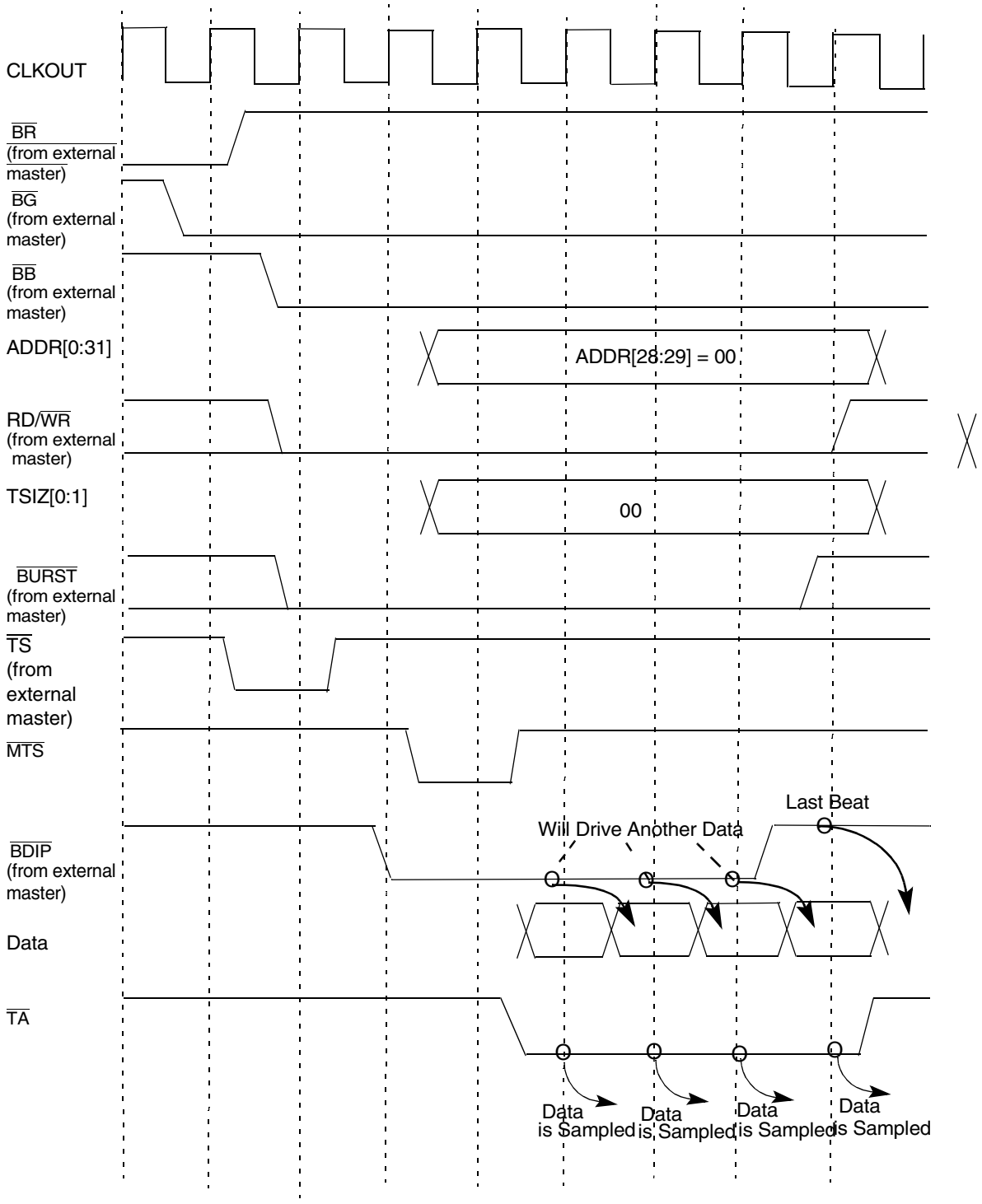
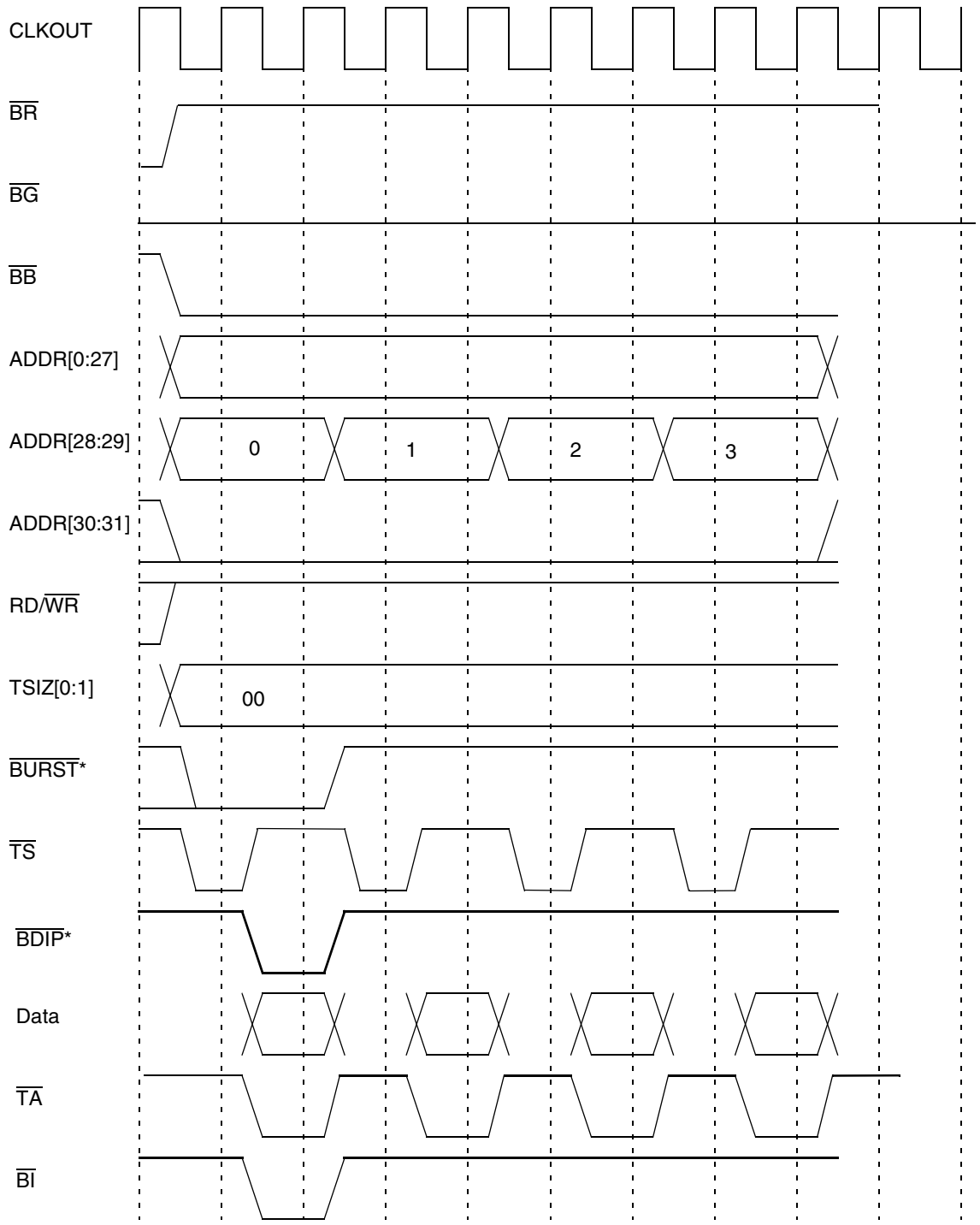


Figure 9-17 Burst-Write Cycle, 32-Bit Port Size, Zero Wait States



* $\overline{\text{BURST}}$ and $\overline{\text{BDIP}}$ will be asserted for one cycle if the RCPU core requests a burst, but the USIU splits it into a sequence of normal cycles.

Figure 9-18 Burst-Inhibit Cycle, 32-Bit Port Size (Emulated Burst)

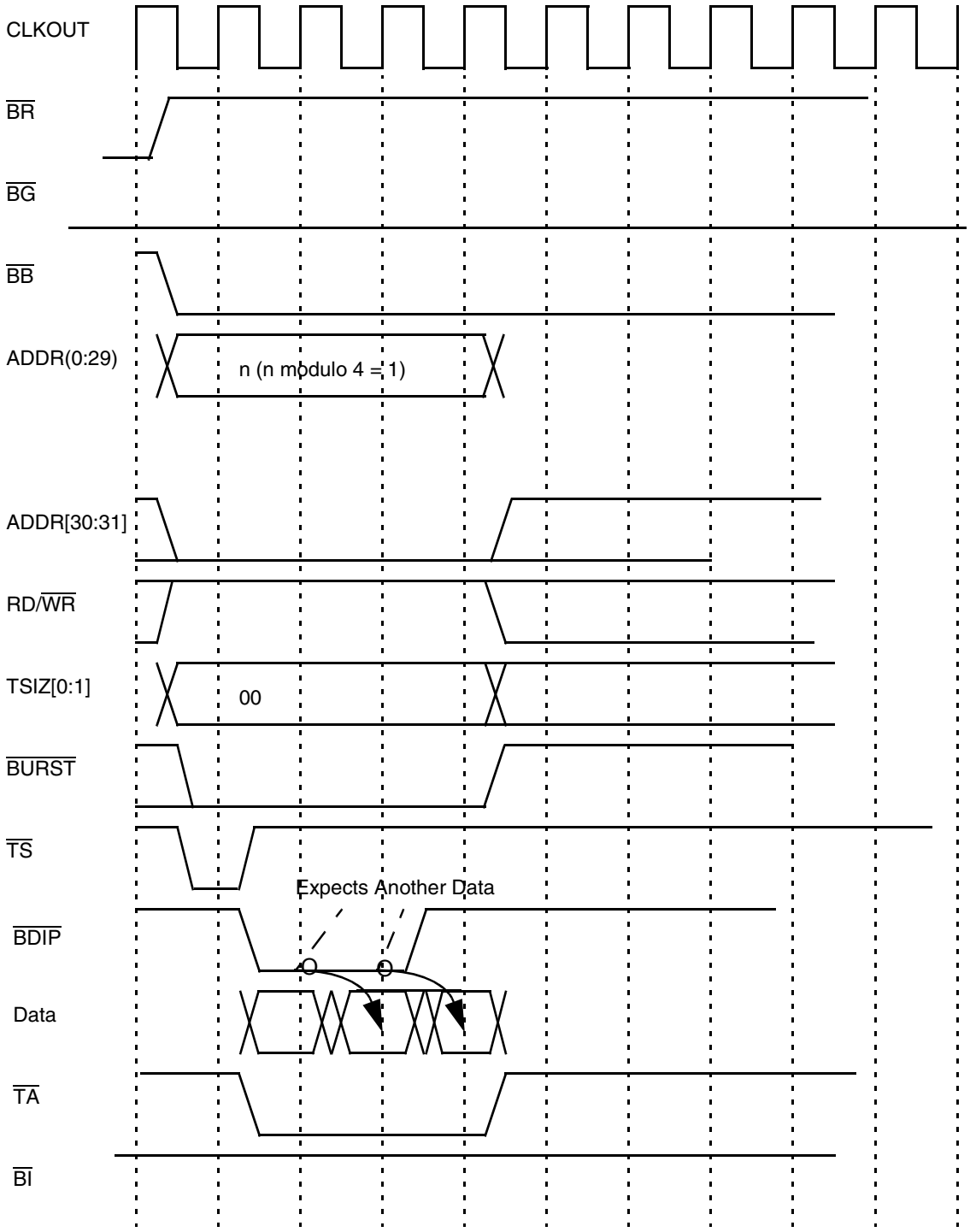


Figure 9-19 Non-Wrap Burst with Three Beats

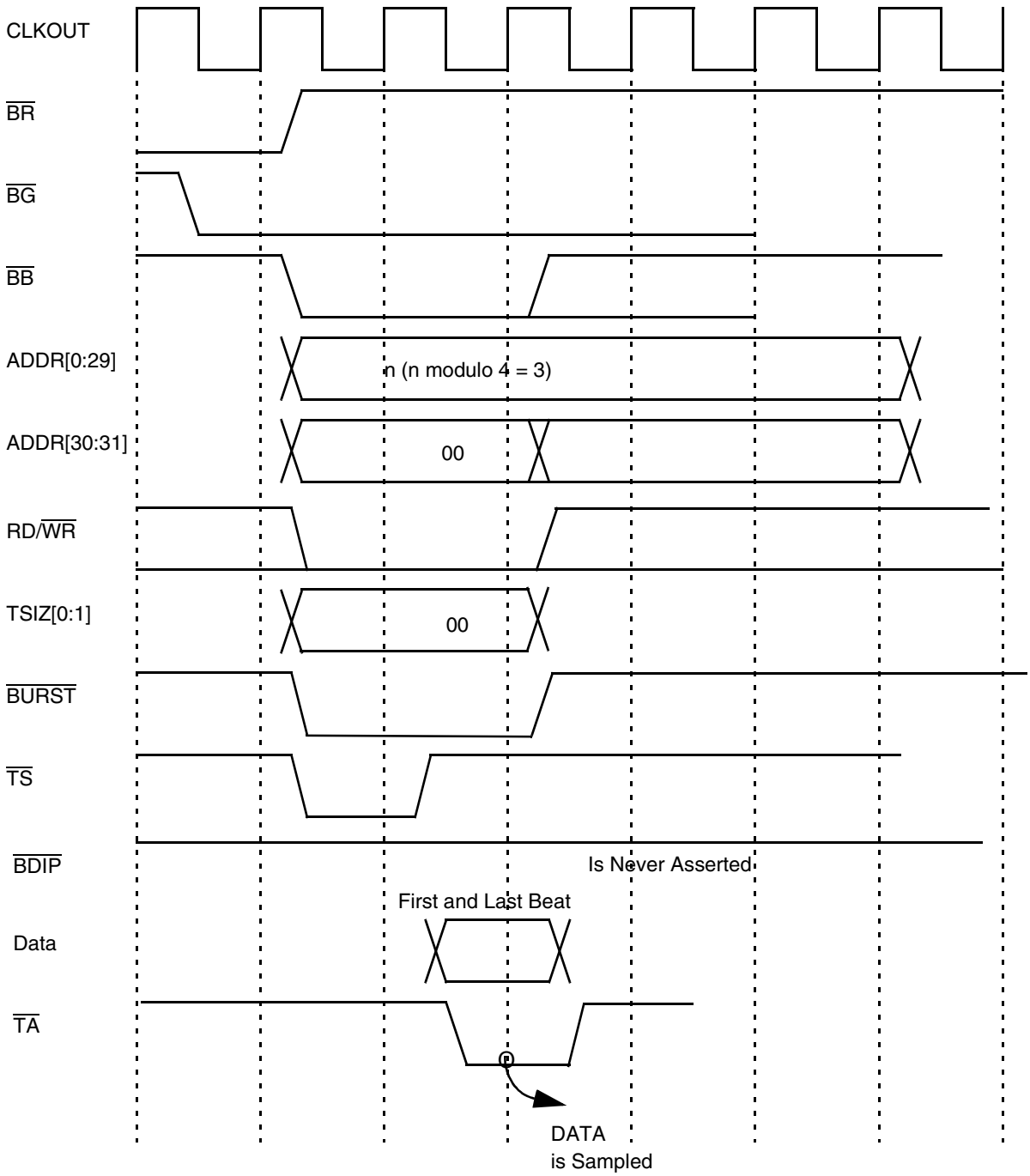


Figure 9-20 Non-Wrap Burst with One Data Beat

9.5.5 Alignment and Packaging of Transfers

The MPC555 / MPC556 external bus requires natural address alignment:

- Byte accesses allow any address alignment
- Half-word accesses require address bit 31 to equal zero
- Word accesses require address bits 30 – 31 to equal zero
- Burst accesses require address bits 30 – 31 to equal zero

The MPC555 / MPC556 performs operand transfers through its 32-bit data port. If the transfer is controlled by the internal memory controller, the MPC555 / MPC556 can support 8- and 16-bit data port sizes.

The bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on DATA[0:31], a 16-bit port must reside on DATA[0:15], and an 8-bit port must reside on DATA[0:7]. The MPC555 / MPC556 always tries to transfer the maximum amount of data on all bus cycles. For a word operation, it always assumes that the port is 32 bits wide when beginning the bus cycle.

In **Figure 9-21**, **Figure 9-22**, **Table 9-2**, and **Table 9-3**, the following conventions are used:

- OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.
- The two bytes of a half-word operand are either OP0 (most-significant) and OP1 or OP2 (most-significant) and OP3, depending on the address of the access.
- The single byte of a byte-length operand is OP0, OP1, OP2, or OP3, depending on the address of the access.

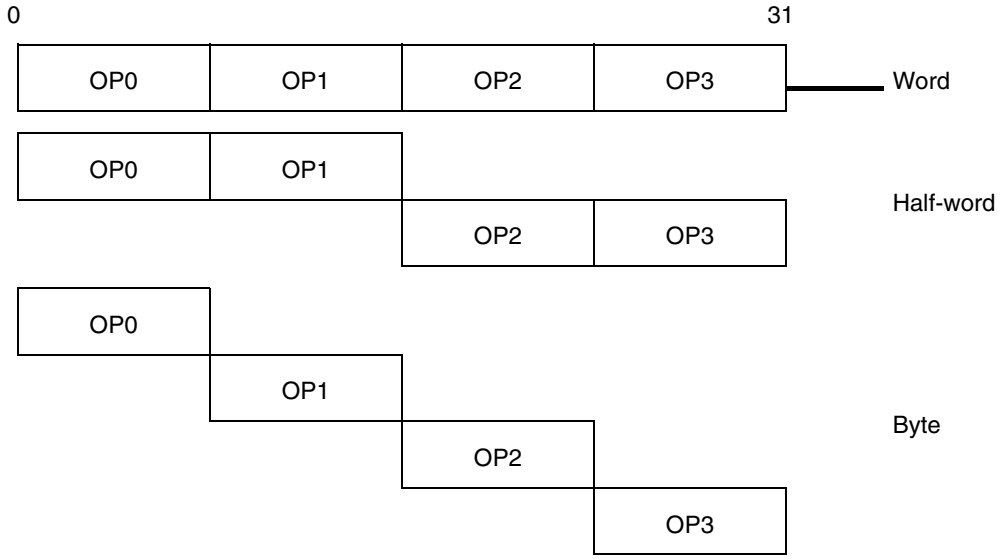


Figure 9-21 Internal Operand Representation

Figure 9-22 illustrates the device connections on the data bus.

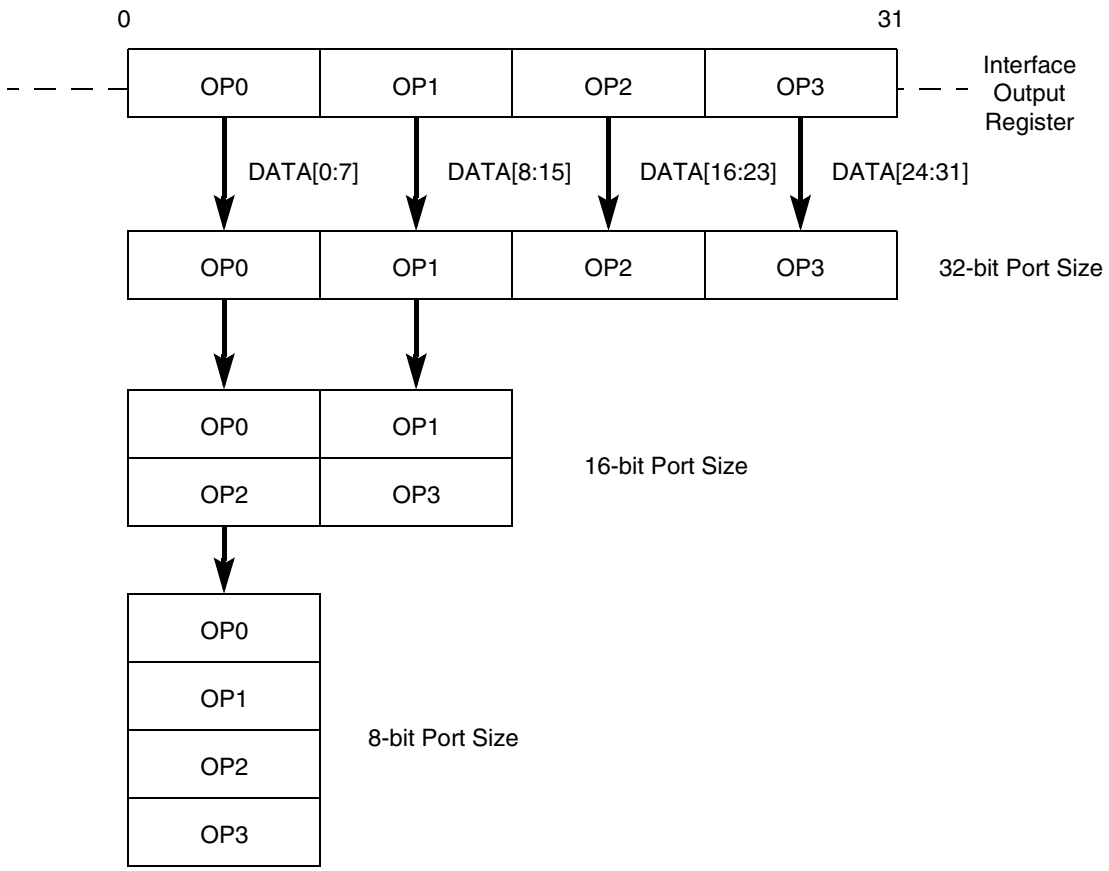


Figure 9-22 Interface To Different Port Size Devices

Table 9-2 lists the bytes required on the data bus for read cycles.

Table 9-2 Data Bus Requirements For Read Cycles

Transfer Size	TSIZE [0:1]	Address	32-bit Port Size				16-bit Port Size		8-bit Port Size
		ADDR [30:31]	DATA [0:7]	DATA [8:15]	DATA [16:23]	DATA [24:31]	DATA [0:7]	DATA [8:15]	DATA [0:7]
Byte	01	00	OP0	—	—	—	OP0	—	OP0
	01	01	—	OP1	—	—	—	OP1	OP1
	01	10	—	—	OP2	—	OP2	—	OP2
	01	11	—	—	—	OP3	—	OP3	OP3
Half-word	10	00	OP0	OP1	—	—	OP0	OP1	OP0
	10	10	—	—	OP2	OP3	OP2	OP3	OP2
Word	00	00	OP0	OP1	OP2	OP3	OP0	OP1	OP0

NOTE: "—" denotes a byte not required during that read cycle.

Table 9-3 lists the patterns of the data transfer for write cycles when the MPC555 / MPC556 initiates an access.

Table 9-3 Data Bus Contents for Write Cycles

Transfer Size	TSIZE[0:1]	Address	External Data Bus Pattern			
		ADDR [30:31]	DATA [0:7]	DATA [8:15]	DATA [16:23]	DATA [24:31]
Byte	01	00	OP0	—	—	—
	01	01	OP1	OP1	—	—
	01	10	OP2	—	OP2	—
	01	11	OP3	OP3	—	OP3
Half-word	10	00	OP0	OP1	—	—
	10	10	OP2	OP3	OP2	OP3
Word	00	00	OP0	OP1	OP2	OP3

NOTE: "—" denotes a byte not driven during that write cycle.

9.5.6 Arbitration Phase

The external bus design provides for a single bus master at any one time, either the MPC555 / MPC556 or an external device. One or more of the external devices on the bus can have the capability of becoming bus master for the external bus. Bus arbitration may be handled either by an external central bus arbiter or by the internal on-chip arbiter. In the latter case, the system is optimized for one external bus master besides the MPC555 / MPC556. The arbitration configuration (external or internal) is set at system reset.

Each bus master must have bus request ($\overline{BR}$), bus grant ($\overline{BG}$), and bus busy ($\overline{BB}$) signals. The device that needs the bus asserts $\overline{BR}$. The device then waits for the arbiter to assert $\overline{BG}$. In addition, the new master must look at $\overline{BB}$ to ensure that no other master is driving the bus before it can assert $\overline{BB}$ to assume ownership of the bus. Any time the arbiter has taken the bus grant away from the master and the master wants to ex-

ecute a new cycle, the master must re-arbitrate before a new cycle can be executed. The MPC555 / MPC556, however, guarantees data coherency for access to a small port size and for decomposed bursts. This means that the MPC555 / MPC556 will not release the bus before the completion of the transactions that are considered atomic. **Figure 9-23** describes the basic protocol for bus arbitration.

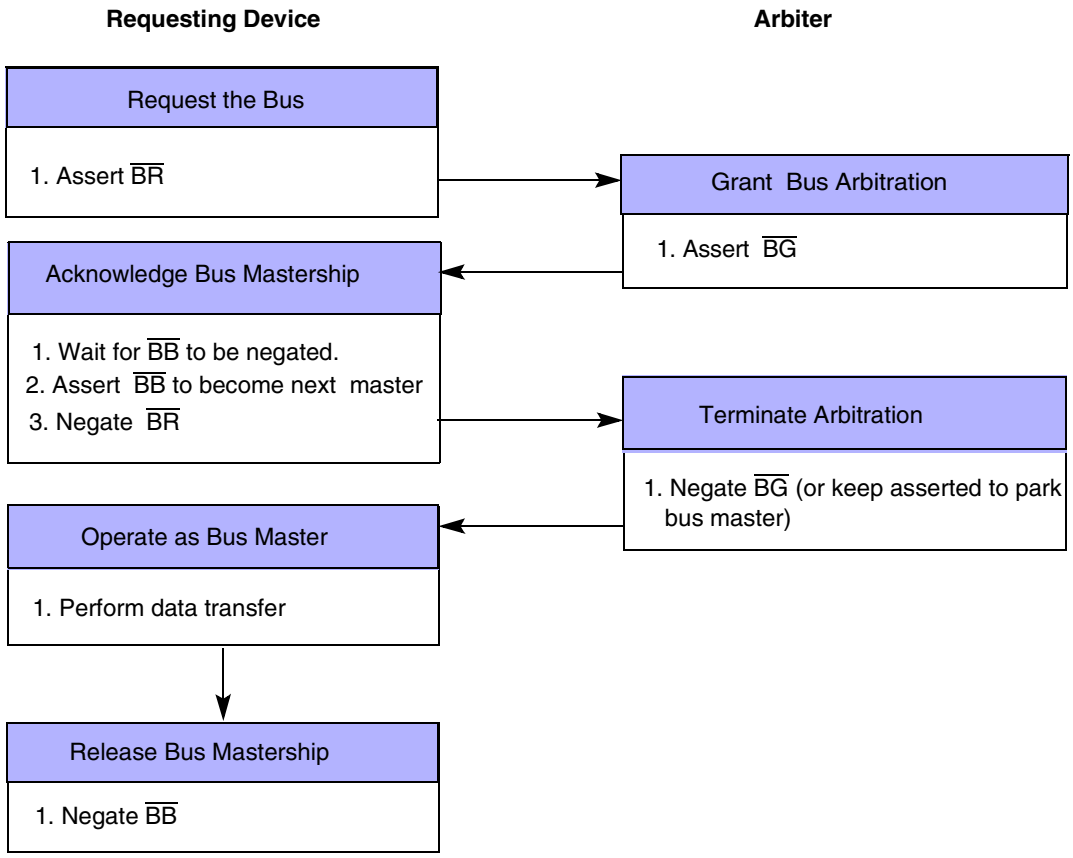


Figure 9-23 Bus Arbitration Flowchart

9.5.6.1 Bus Request

The potential bus master asserts $\overline{BR}$ to request bus mastership. $\overline{BR}$ should be negated as soon as the bus is granted, the bus is not busy, and the new master can drive the bus. If more requests are pending, the master can keep asserting its bus request as long as needed. When configured for external central arbitration, the MPC555 / MPC556 drives this signal when it requires bus mastership. When the internal on-chip arbiter is used, this signal is an input to the internal arbiter and should be driven by the external bus master.

9.5.6.2 Bus Grant

The arbiter asserts $\overline{BG}$ to indicate that the bus is granted to the requesting device. This signal can be negated following the negation of $\overline{BR}$ or kept asserted for the current master to park the bus.

When configured for external central arbitration, $\overline{BG}$ is an input signal to the MPC555 / MPC556 from the external arbiter. When the internal on-chip arbiter is used, this signal is an output from the internal arbiter to the external bus master.

9.5.6.3 Bus Busy

$\overline{BB}$ assertion indicates that the current bus master is using the bus. New masters should not begin transfer until this signal is negated. The bus owner should not relinquish or negate this signal until the transfer is complete. To avoid contention on the $\overline{BB}$ line, the master should three-state this signal when it gets a logical one value. This requires the connection of an external pull-up resistor to ensure that a master that acquires the bus is able to recognize the $\overline{BB}$ line negated, regardless of how many cycles have passed since the previous master relinquished the bus. Refer to **Figure 9-24**.

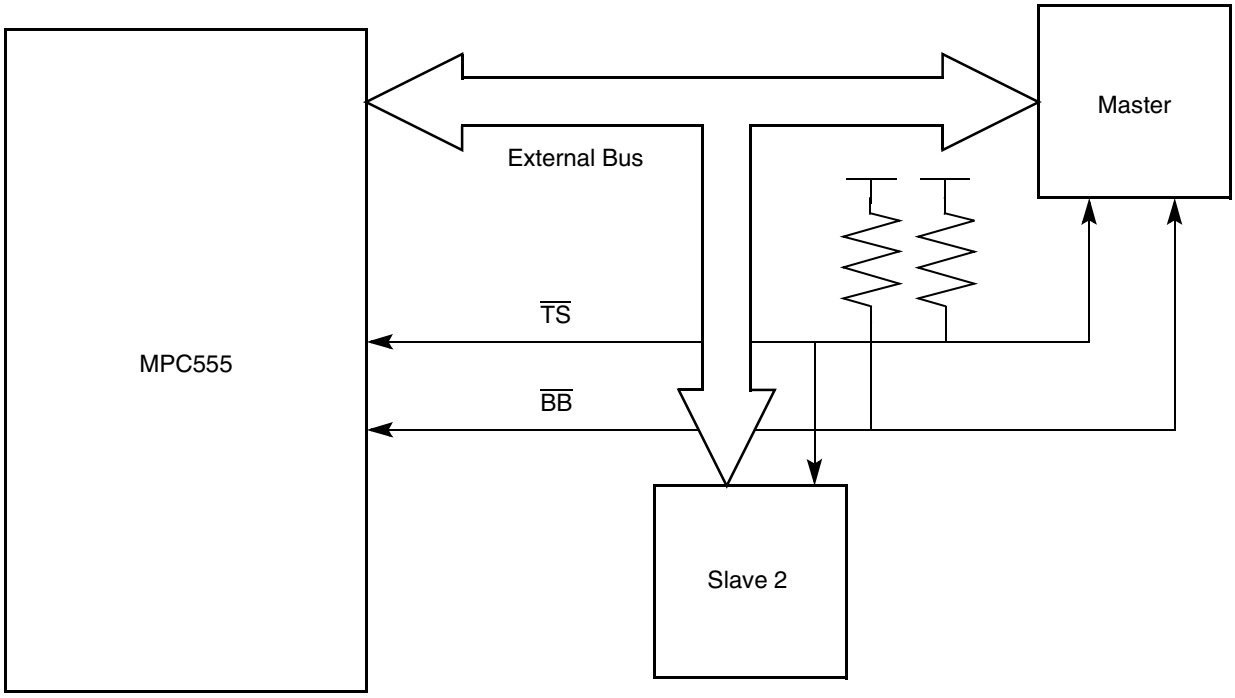


Figure 9-24 Masters Signals Basic Connection

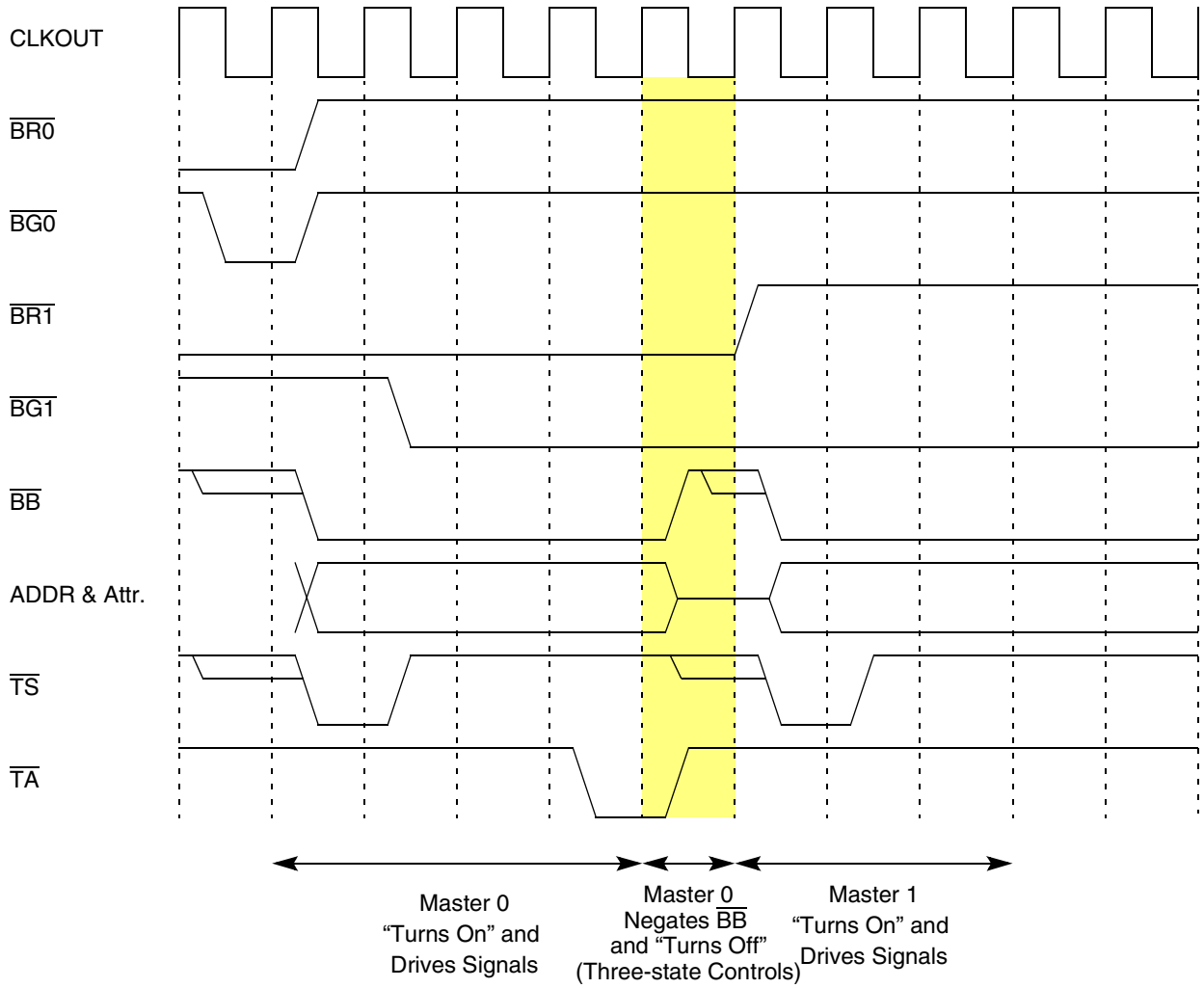


Figure 9-25 Bus Arbitration Timing Diagram

9.5.6.4 Internal Bus Arbiter

The MPC555 / MPC556 can be configured at system reset to use the internal bus arbiter. In this case, the MPC555 / MPC556 will be parked on the bus. The parking feature allows the MPC555 / MPC556 to skip the bus request phase, and if $\overline{BB}$ is negated, assert $\overline{BB}$ and initiate the transaction without waiting for $\overline{BG}$ from the arbiter.

The priority of the external device relative to the internal MPC555 / MPC556 bus masters is programmed in the SIU module configuration register. If the external device requests the bus and the MPC555 / MPC556 does not require it, or if the external device has higher priority than the current internal bus master, the MPC555 / MPC556 grants the bus to the external device.

Table 9-4 describes the priority mechanism used by the internal arbiter.

Table 9-4 Priority Between Internal and External Masters over External Bus[1]

Type	Direction	Priority
Parked access[2]	Internal → external	0
Instruction access	Internal → external	3
Data access	Internal → external	4
External access	external → external/internal	EARP[3] (could be programmed to 0 – 7)

NOTES:
1. External master will be granted external bus ownership if EARP is greater than the internal access priority.
2. Parked access is instruction or data access from the RCPU which is initiated on the internal bus without requesting it first in order to improve performance.
3. Refer to **6.13.1.1 SIU Module Configuration Register**.

Figure 9-26 illustrates the internal finite-state machine that implements the arbiter protocol.

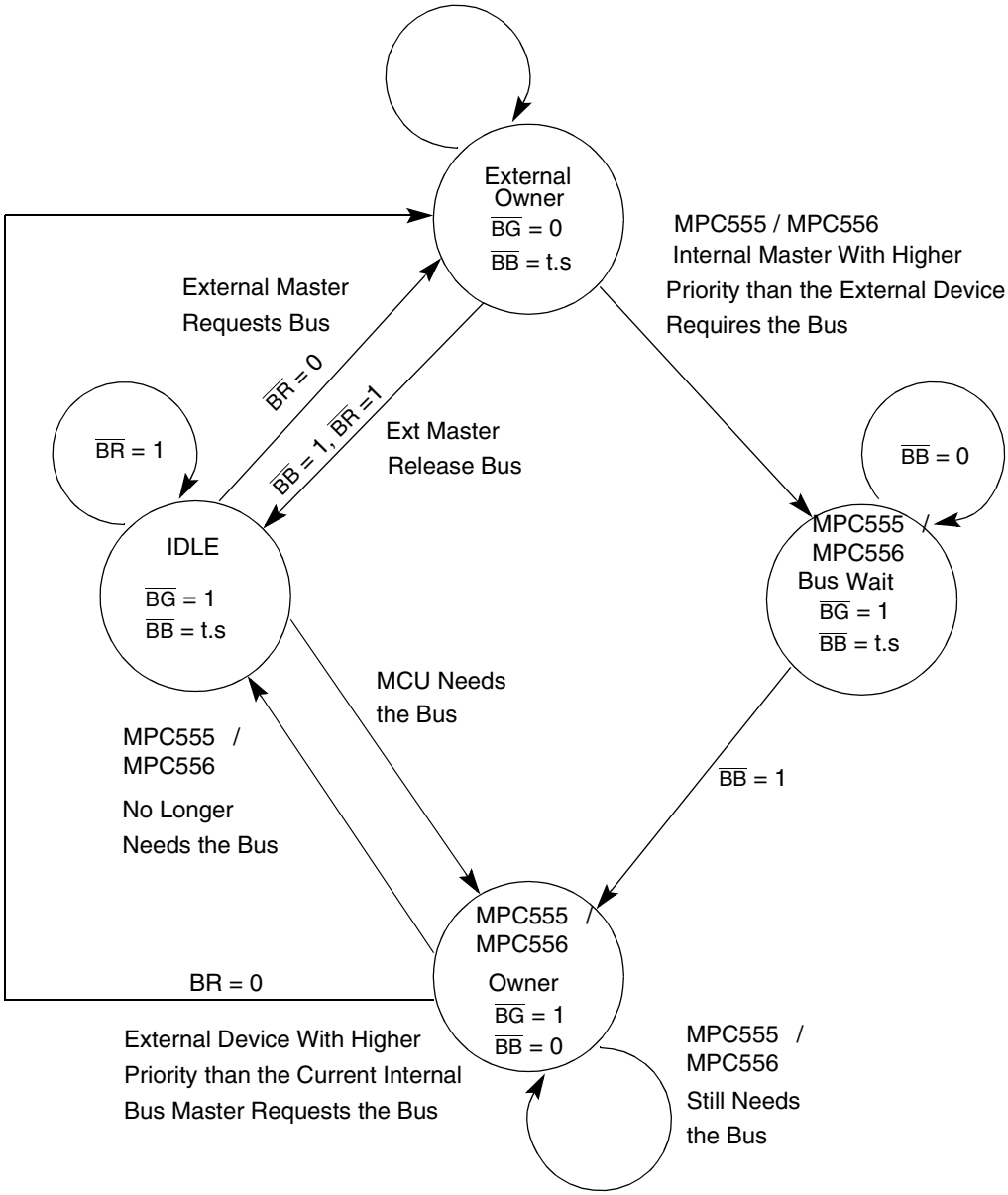


Figure 9-26 Internal Bus Arbitration State Machine

9.5.7 Address Transfer Phase Signals

Address transfer phase signals include the following:

- Transfer start
- Address bus
- Transfer attributes

Transfer attributes signals include RD/$\overline{\text{WR}}$, $\overline{\text{BURST}}$, TSIZ[0:1], AT[0:3], $\overline{\text{STS}}$, and $\overline{\text{BDIP}}$. With the exception of the $\overline{\text{BDIP}}$, these signals are available at the same time as the address bus.

9.5.7.1 Transfer Start

This signal ($\overline{TS}$) indicates the beginning of a transaction on the bus addressing a slave device. This signal should be asserted by a master only after the ownership of the bus was granted by the arbitration protocol. This signal is asserted for the first cycle of the transaction only and is negated in successive clock cycles until the end of the transaction. The master should three-state this signal when it relinquishes the bus to avoid contention between two or more masters in this line. This situation indicates that an external pull-up resistor should be connected to the $\overline{TS}$ signal to avoid having a slave recognize this signal as asserted when no master drives it. Refer to **Figure 9-24**.

9.5.7.2 Address Bus

The address bus consists of 32 bits, with ADDR[0] the most significant bit and ADDR[31] the least significant bit. The bus is byte-addressable, so each address can address one or more bytes. The address and its attributes are driven on the bus with the transfer start signal and kept valid until the bus master receives the transfer acknowledge signal from the slave. To distinguish the individual byte, the slave device must observe the TSIZ signals.

9.5.7.3 Read/Write

A high value on the RD/$\overline{WR}$ line indicates a read access. A low value indicates a write access.

9.5.7.4 Burst Indicator

$\overline{BURST}$ is driven by the bus master at the beginning of the bus cycle along with the address to indicate that the transfer is a burst transfer.

The MPC555 / MPC556 supports a non-wrapping, four-beat maximum, critical word first burst type. The maximum burst size is 16 bytes. For a 32-bit port, the burst includes four beats. For a 16-bit port, the burst includes 8 beats. For an 8-bit port, the burst includes 16 beats at most. Note that 8- and 16-bit ports must be controlled by the memory controller.

The actual size of the burst is determined by the address of the starting word of the burst. Refer to **Table 9-5** and **Table 9-6**.

Table 9-5 Burst Length and Order

Starting Address ADDR[28:29]	Burst Order (Assuming 32-bit Port Size)	Burst Length in Words (Beats)	Burst Length in Bytes	Comments
00	word 0 → word 1 → word 2 → word 3	4	16	
01	word 1 → word 2 → word 3	3	12	
10	word 2 → word 3	2	8	
11	word 3	1	4	$\overline{BDIP}$ never asserted

9.5.7.5 Transfer Size

The transfer size signals (TSIZ[0:1]) indicate the size of the requested data transfer. During each transfer, the TSIZ signals indicate how many bytes are remaining to be transferred by the transaction. The TSIZ signals can be used with $\overline{\text{BURST}}$ and ADDR[30:31] to determine which byte lanes of the data bus are involved in the transfer. For nonburst transfers, the TSIZ signals specify the number of bytes starting from the byte location addressed by ADDR[30:31]. In burst transfers, the value of TSIZ is always 00.

Table 9-6 $\overline{\text{BURST}}$/TSIZE Encoding

$\overline{\text{BURST}}$	TSIZ(0:1)	Transfer Size
Negated	01	Byte
Negated	10	Half-word
Negated	11	x
Negated	00	Word
Asserted	00	Burst (16 bytes)

9.5.7.6 Address Types

The address type (AT[0:3]), program trace ($\overline{\text{PTR}}$), and reservation transfer (RSV) signals are outputs that indicate one of 16 address types. These types are designated as either a normal or alternate master cycle, user or supervisor, and instruction or data type. The address type signals are valid at the rising edge of the clock in which the special transfer start ($\overline{\text{STS}}$) signal is asserted.

A special use of the $\overline{\text{PTR}}$ and $\overline{\text{RSV}}$ signals is for the reservation protocol described in **9.5.9 Storage Reservation.** Refer to **9.5.13 Show Cycle Transactions** for information on show cycles.

Table 9-7 summarizes the pins used to define the address type. **Table 9-8** lists all the definitions achieved by combining these pins.

Table 9-7 Address Type Pins

Pin	Function
$\overline{\text{STS}}$	0 = Special transfer 1 = Normal transfer
$\overline{\text{TS}}$	0 = Start of transfer 1 = No transfer
AT0	Must equal zero on MPC555 / MPC556
AT1	0 = Supervisor mode 1 = User mode
AT2	0 = Instruction 1 = Data
AT3	Reservation/Program Trace
$\overline{\text{PTR}}$	0 = Program trace 1 = No program trace
$\overline{\text{RSV}}$	0 = Reservation data 1 = No reservation

Table 9-8 Address Types Definition

$\overline{STS}$	$\overline{TS}$	AT0	AT1	AT2	AT3	$\overline{PTR}$	$\overline{RSV}$	Address Space Definitions
1	x	x	x	x	x	1	1	No transfer
0	0[1]	0	0	0	0	0	1	RCPU, normal instruction, program trace, supervisor mode
					1	1	1	RCPU, normal instruction, supervisor mode
				1	0	1	0	RCPU, reservation data, supervisor mode
					1	1	1	RCPU, normal data, supervisor mode
			1	0	0	0	1	RCPU, normal instruction, program trace, user mode
					1	1	1	RCPU, normal instruction, user mode
				1	0	1	0	RCPU, reservation data, user mode
					1	1	1	RCPU, normal data, user mode
		1	?	?	?	1	1	Reserved
	1	0	0	0	0	0	1	RCPU, show cycle address instruction, program trace, supervisor mode
					1	1	1	RCPU, show cycle address instruction, supervisor mode
				1	0	1	0	RCPU, reservation show cycle data, supervisor mode
					1	1	1	RCPU, show cycle data, supervisor mode
			1	0	0	0	1	RCPU, show cycle address instruction, program trace, user mode
					1	1	1	RCPU, show cycle address instruction, user mode
				1	0	1	0	RCPU, reservation show cycle data, user mode
					1	1	1	RCPU, show cycle data, user mode
		1	?	?	?	1	1	Reserved

NOTES:
1. Cases in which both $\overline{TS}$ and $\overline{STS}$ are asserted indicate normal cycles with the show cycle attribute.

9.5.7.7 Burst Data in Progress

This signal is sent from the master to the slave to indicate that there is a data beat following the current data beat. The master uses this signal to give the slave advance warning of the remaining data in the burst. $\overline{BDIP}$ can also be used to terminate the burst cycle early. Refer to **9.5.3 Burst Transfer** and **9.5.4 Burst Mechanism** for more information.

9.5.8 Termination Signals

The EBI uses three termination signals:

- Transfer acknowledge ($\overline{TA}$)
- Burst inhibit ($\overline{BI}$)
- Transfer error acknowledge ($\overline{TEA}$)

9.5.8.1 Transfer Acknowledge

Transfer acknowledge indicates normal completion of the bus transfer. During a burst cycle, the slave asserts this signal with every data beat returned or accepted.

9.5.8.2 Burst Inhibit

A slave sends the $\overline{BI}$ signal to the master to indicate that the addressed device does not have burst capability. If this signal is asserted, the master must transfer in multiple cycles and increment the address for the slave to complete the burst transfer. For a system that does not use the burst mode at all, this signal can be tied low permanently.

9.5.8.3 Transfer Error Acknowledge

The $\overline{TEA}$ signal terminates a bus cycle under one or more bus error conditions. The current bus cycle must be aborted. This signal overrides any other cycle termination signals, such as transfer acknowledge.

9.5.8.4 Termination Signals Protocol

The transfer protocol was defined to avoid electrical contention on lines that can be driven by various sources. To this end, a slave must not drive signals associated with the data transfer until the address phase is completed and it recognizes the address as its own. The slave must disconnect from signals immediately after it has acknowledged the cycle and no later than the termination of the next address phase cycle. This means that the termination signals must be connected to power through a pull-up resistor to avoid the situation in which a master samples an undefined value in any of these signals when no real slave is addressed.

Refer to **Figure 9-27** and **Figure 9-28**.

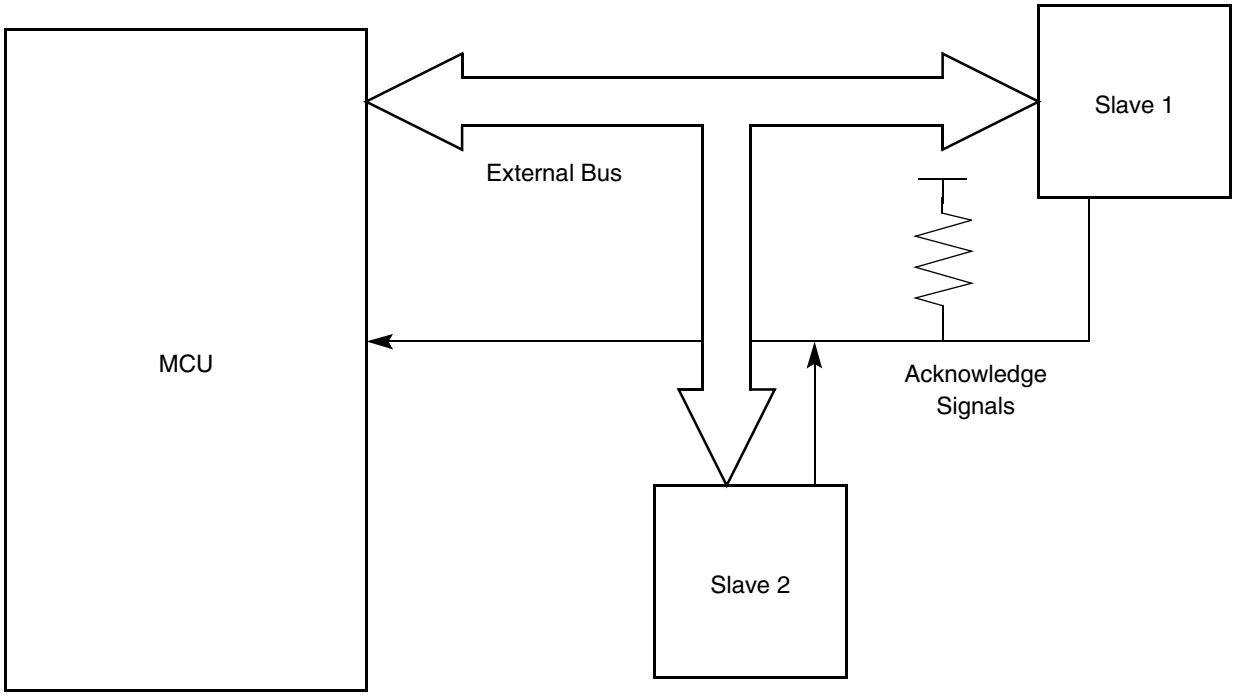


Figure 9-27 Termination Signals Protocol Basic Connection

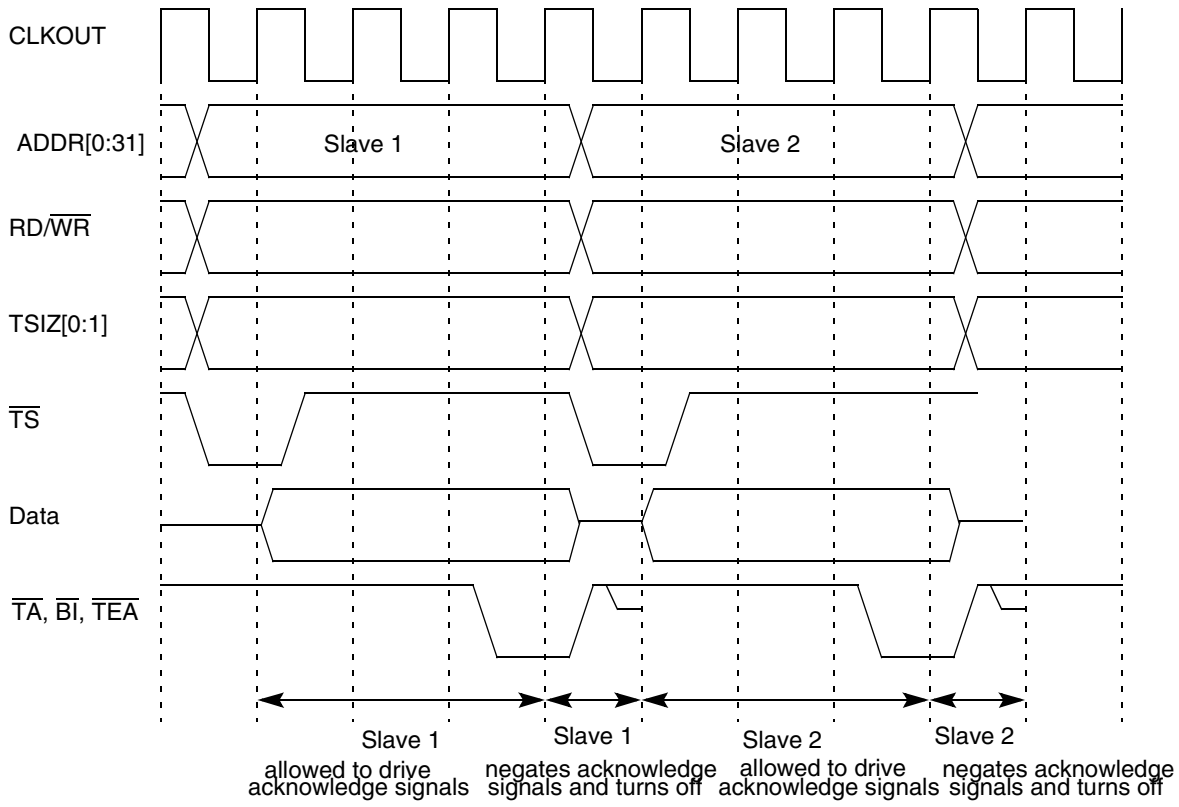


Figure 9-28 Termination Signals Protocol Timing Diagram

9.5.9 Storage Reservation

The MPC555 / MPC556 storage reservation protocol supports a multi-level bus structure. For each local bus, storage reservation is handled by the local reservation logic.

The protocol tries to optimize reservation cancellation such that a PowerPC processor is notified of storage reservation loss on a remote bus only when it has issued a **stwcx** cycle to that address. That is, the reservation loss indication comes as part of the **stwcx** cycle. This method avoids the need to have very fast storage reservation loss indication signals routed from every remote bus to every PowerPC master.

The storage reservation protocol makes the following assumptions:

- Each processor has, at most, one reservation flag
- **lwarx** sets the reservation flag
- **lwarx** by the same processor clears the reservation flag related to a previous **lwarx** instruction and again sets the reservation flag
- **stwcx** by the same processor clears the reservation flag
- Store by the same processor does *not* clear the reservation flag
- Some other processor (or other mechanism) store to the same address as an existing reservation clears the reservation flag
- In case the storage reservation is lost, it is guaranteed that **stwcx** will not modify the storage

The reservation protocol for a single-level (local) bus is illustrated in **Figure 9-29**. The protocol assumes that an external logic on the bus carries out the following functions:

- Snoops accesses to all local bus slaves
- Holds one reservation for each local master capable of storage reservations
- Sets the reservation when that master issues a load and reserve request
- Clears the reservation when some other master issues a store to the reservation address

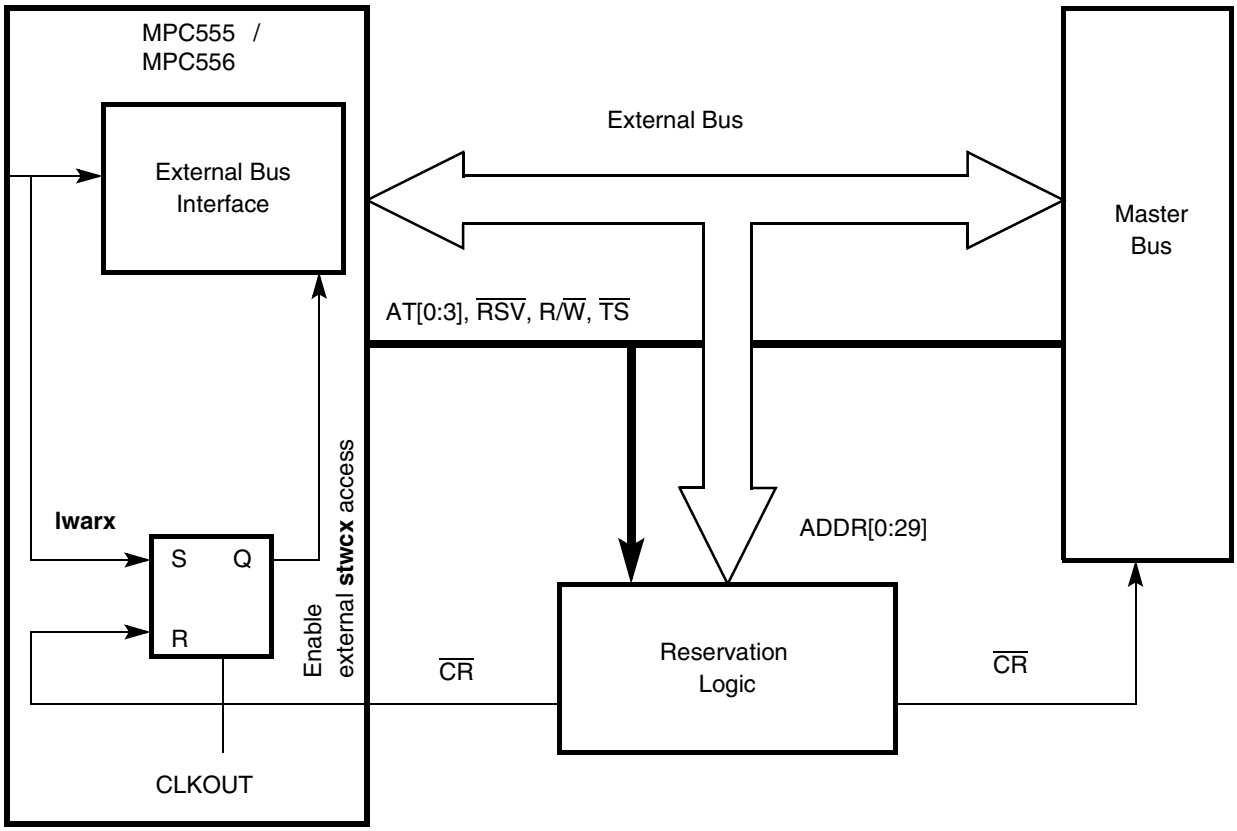


Figure 9-29 Reservation On Local Bus

The MPC555 / MPC556 samples the $\overline{CR}$ line at the rising edge of CLKOUT. When this signal is asserted, the reservation flag is reset.

The EBI samples the logical value of the reservation flag prior to externally starting a bus cycle initiated by the RCPU **stwcx** instruction. If the reservation flag is set, the EBI begins with the bus cycle. If the reservation flag is reset, no bus cycle is initiated externally, and this situation is reported to the RCPU.

The reservation protocol for a multi-level (local) bus is illustrated in **Figure 9-30**. The system describes the situation in which the reserved location is sited in the remote bus.

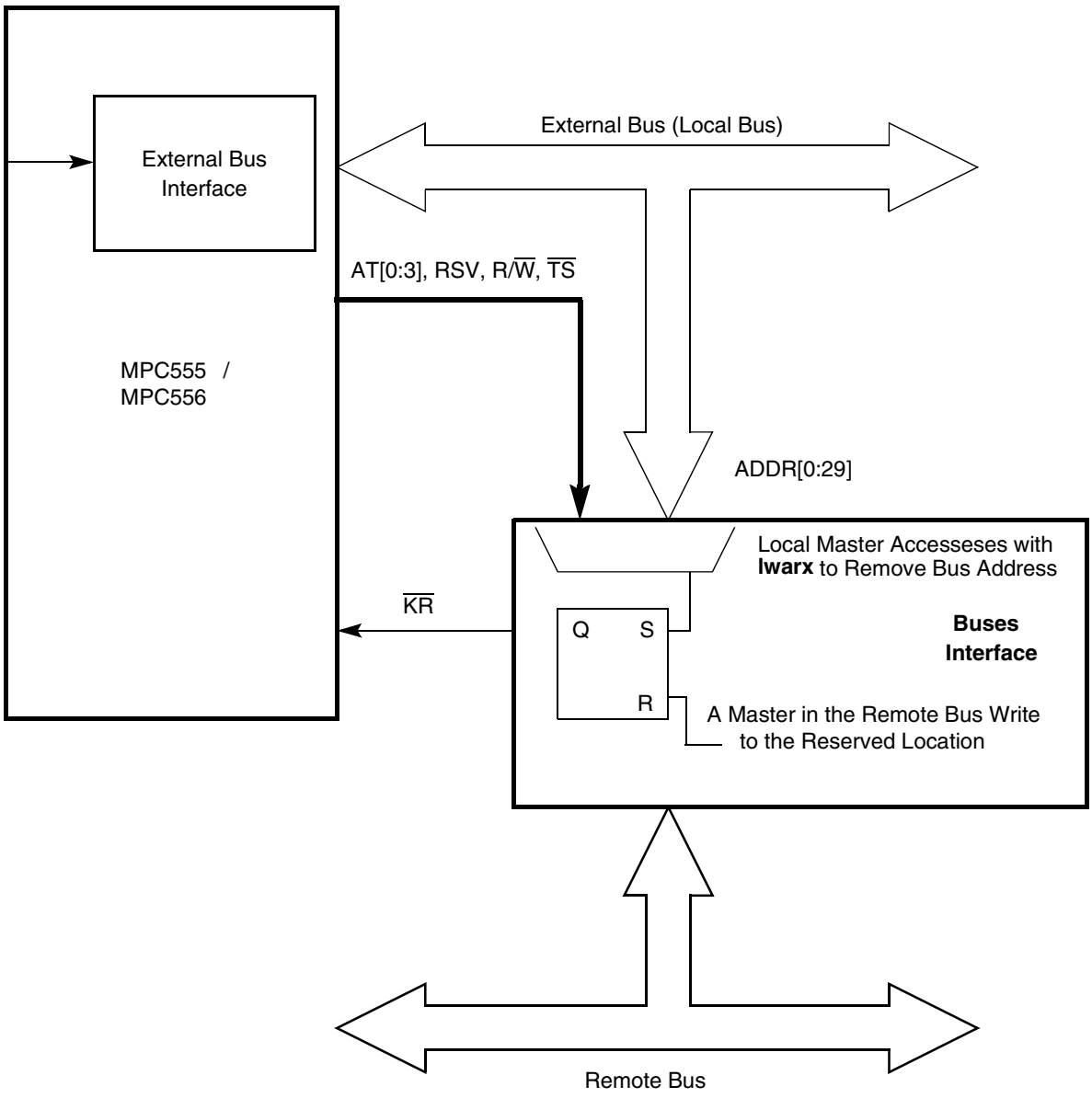


Figure 9-30 Reservation On Multilevel Bus Hierarchy

In this case, the bus interface block implements a reservation flag for the local bus master. The reservation flag is set by the bus interface when a load with reservation is issued by the local bus master and the reservation address is located on the remote bus. The flag is reset when an alternative master on the remote bus accesses the same location in a write cycle. If the MPC555 / MPC556 begins a memory cycle to the previously reserved address (located in the remote bus) as a result of an **stwcx** instruction, the following two cases can occur:

- If the reservation flag is set, the buses interface acknowledges the cycle in a normal way
- If the reservation flag is reset, the bus interface should assert the KR. However,

the bus interface should not perform the remote bus write-access or abort it if the remote bus supports aborted cycles. In this case the failure of the **stwcx** instruction is reported to the RCPU.

9.5.10 Bus Exception Control Cycles

The MPC555 / MPC556 bus architecture requires assertion of $\overline{TA}$ from an external device to signal that the bus cycle is complete. $\overline{TA}$ is not asserted in the following cases:

- The external device does not respond
- Various other application-dependent errors occur

External circuitry can provide $\overline{TEA}$ when no device responds by asserting $\overline{TA}$ within an appropriate period of time after the MPC555 / MPC556 initiates the bus cycle (it can be the internal bus monitor). This allows the cycle to terminate and the processor to enter exception-processing for the error condition (each one of the internal masters causes an internal interrupt under this situation). To properly control termination of a bus cycle for a bus error, $\overline{TEA}$ must be asserted at the same time or before $\overline{TA}$ is asserted. $\overline{TEA}$ should be negated before the second rising edge after it was sampled as asserted to avoid the detection of an error for the next initiated bus cycle. $\overline{TEA}$ is an open drain pin that allows the “wired-or” of any different sources of error generation.

9.5.10.1 Retrying a Bus Cycle

When an external device asserts the $\overline{RETRY}$ signal during a bus cycle, the MPC555 / MPC556 enters a sequence in which it terminates the current transaction, relinquishes the ownership of the bus, and retries the cycle using the same address, address attributes, and data (in the case of a write cycle).

Figure 9-31 illustrates the behavior of the MPC555 / MPC556 when the $\overline{RETRY}$ signal is detected as a termination of a transfer. As seen in this figure, in the case when the internal arbiter is enabled, the MPC555 / MPC556 negates $\overline{BB}$ and asserts $\overline{BG}$ in the clock cycle following the retry detection. This allows any external master to gain bus ownership. In the next clock cycle, a normal arbitration procedure occurs again. As shown in the figure, the external master did not use the bus, so the MPC555 / MPC556 initiates a new transfer with the same address and attributes as before.

In **Figure 9-32**, the same situation is shown except that the MPC555 / MPC556 is working with an external arbiter. In this case, in the clock cycle after the $\overline{RETRY}$ signal is detected asserted, $\overline{BR}$ is negated together with $\overline{BB}$. One clock cycle later, the normal arbitration procedure occurs again.

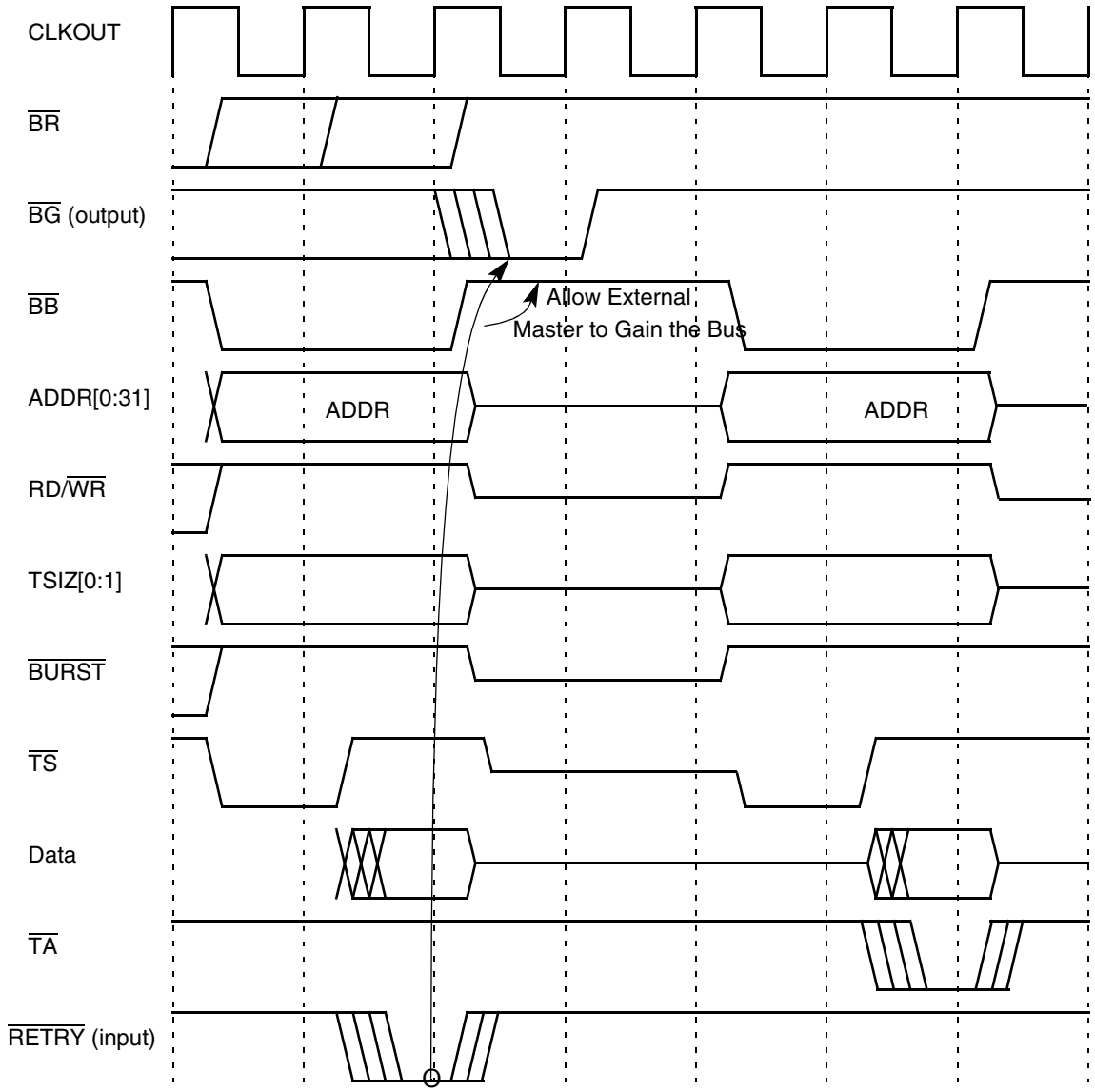


Figure 9-31 Retry Transfer Timing–Internal Arbiter

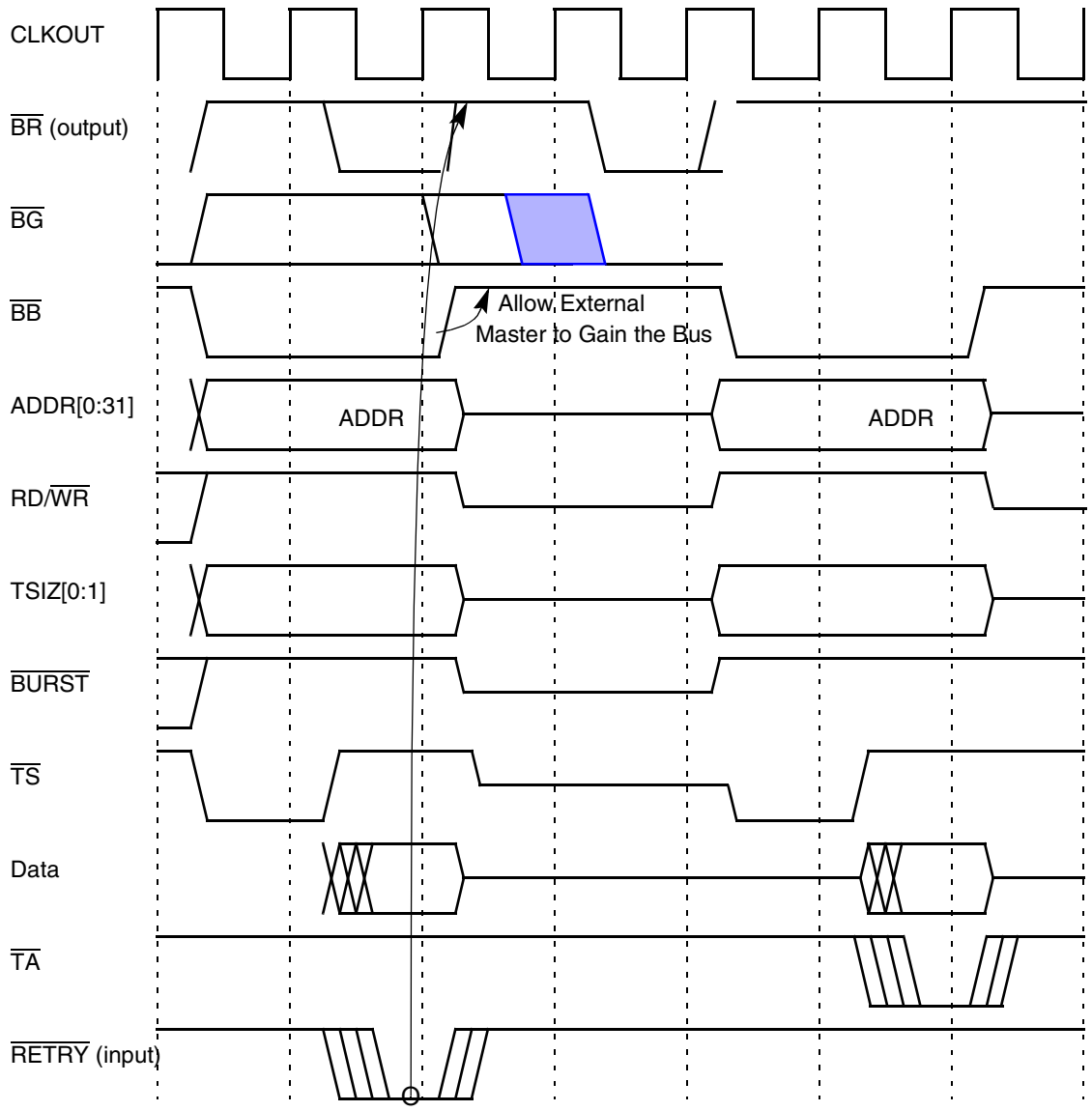


Figure 9-32 Retry Transfer Timing–External Arbiter

When the MPC555 / MPC556 initiates a burst access, the bus interface recognizes the RETRY assertion as a retry termination only if it detects it before the first data beat was acknowledged by the slave device. When the RETRY signal is asserted as a termination signal on any data beat of the access after the first (being the first data beat acknowledged by a normal TA assertion), the MPC555 / MPC556 recognizes RETRY as a transfer error acknowledge.

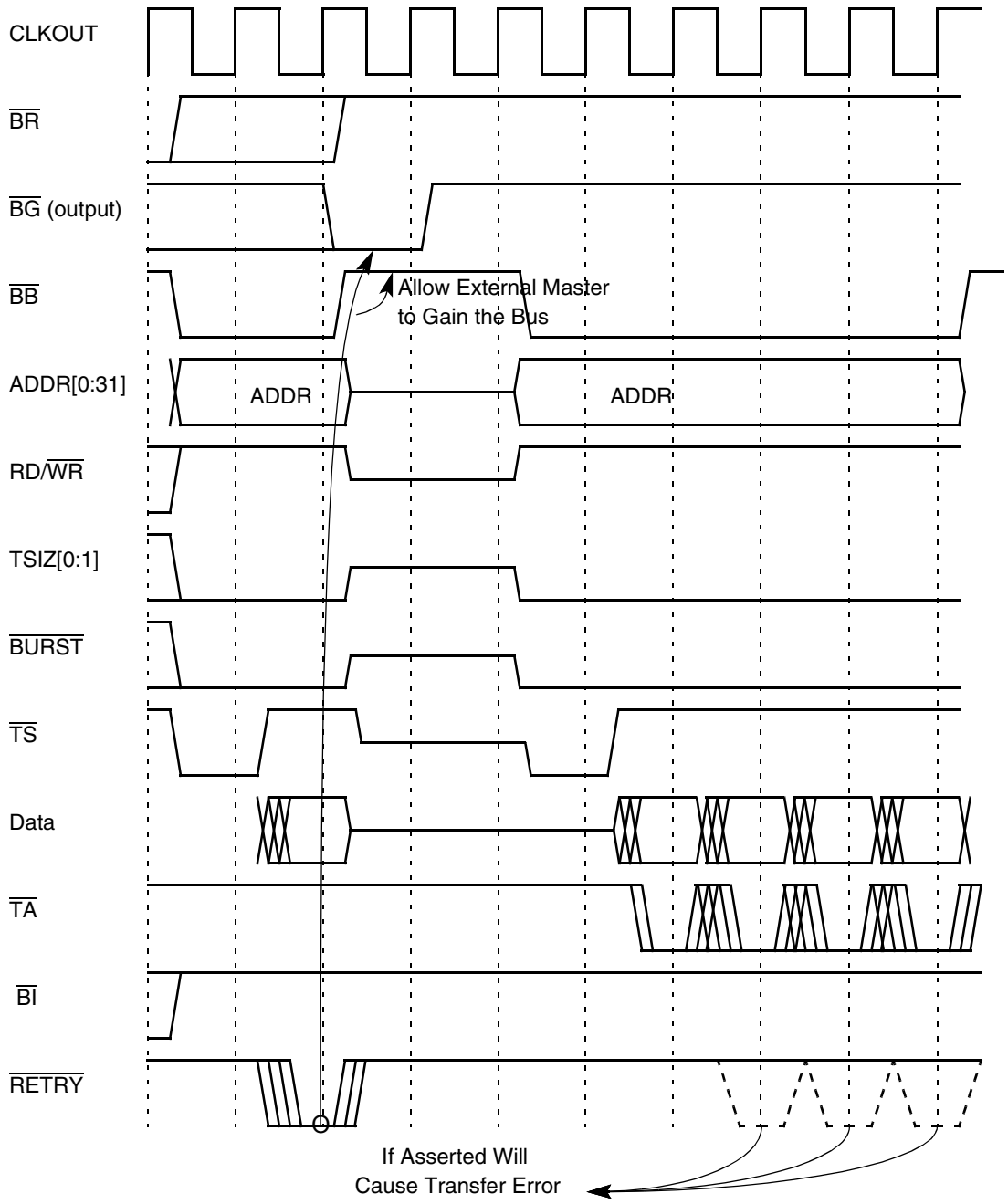


Figure 9-33 Retry On Burst Cycle

If a burst access is acknowledged on its first beat with a normal $\overline{TA}$ but with the $\overline{BI}$ signal asserted, the following single-beat transfers initiated by the MPC555 / MPC556 to complete the 16-byte transfer recognizes the $\overline{RETRY}$ signal assertion as a transfer error acknowledge.

In the case in which a small port size causes the MPC555 / MPC556 to break a bus transaction into several small transactions, terminating any transaction with $\overline{RETRY}$

causes a transfer error acknowledge. See **9.5.2.3 Single Beat Flow with Small Port Size**.

9.5.10.2 Termination Signals Protocol Summary

Table 9-9 summarizes how the MPC555 / MPC556 recognizes the termination signals provided by the slave device that is addressed by the initiated transfer.

Table 9-9 Termination Signals Protocol

$\overline{TEA}$	$\overline{TA}$	$\overline{RETRY}$	Action
Asserted	X	X	Transfer error termination
Negated	Asserted	X	Normal transfer termination
Negated	Negated	Asserted	Retry transfer termination

9.5.11 Bus Operation in External Master Modes

When an external master takes ownership of the external bus and the MPC555 / MPC556 is programmed for external master mode operation, the external master can access the internal space of the MPC555 / MPC556 (see **6.2 External Master Modes**). In an external master mode, the external master owns the bus, and the direction of most of the bus signals is inverted, relative to its direction when the MPC555 / MPC556 owns the bus.

The external master gets ownership of the bus and asserts $\overline{TS}$ in order to initiate an external master access. The access is directed to the internal bus only if the input address matches the internal address space. The access is terminated with one of the followings outputs: $\overline{TA}$, $\overline{TEA}$, or $\overline{RETRY}$. If the access completes successfully, the MPC555 / MPC556 asserts $\overline{TA}$, and the external master can proceed with another external master access or relinquish the bus. If an address or data error is detected internally, the MPC555 / MPC556 asserts $\overline{TEA}$ for one clock. $\overline{TEA}$ should be negated before the second rising edge after it is sampled asserted in order to avoid the detection of an error for the next bus cycle initiated. $\overline{TEA}$ is an open drain pin, and the negation timing depends on the attached pullup. The MPC555 / MPC556 asserts the $\overline{RETRY}$ signal for one clock in order to retry the external master access.

If the address of the external access does not match the internal memory space, the internal memory controller can provide the chip-select and control signals for accesses that belong to one of the memory controller regions. This feature is explained in **SECTION 10 MEMORY CONTROLLER**.

Figure 9-34 and **Figure 9-35** illustrate the basic flow of read and write external master accesses.

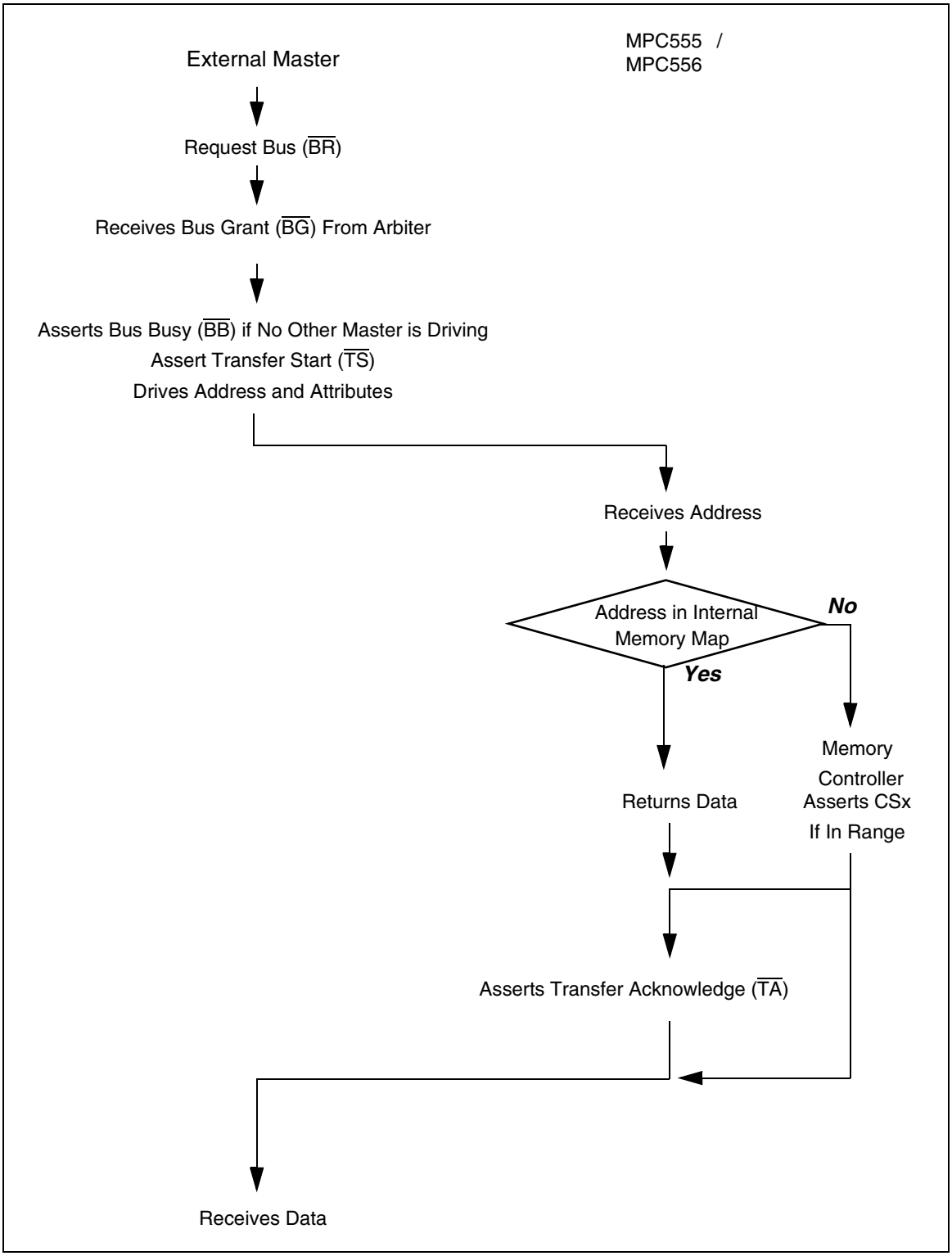


Figure 9-34 Basic Flow of an External Master Read Access

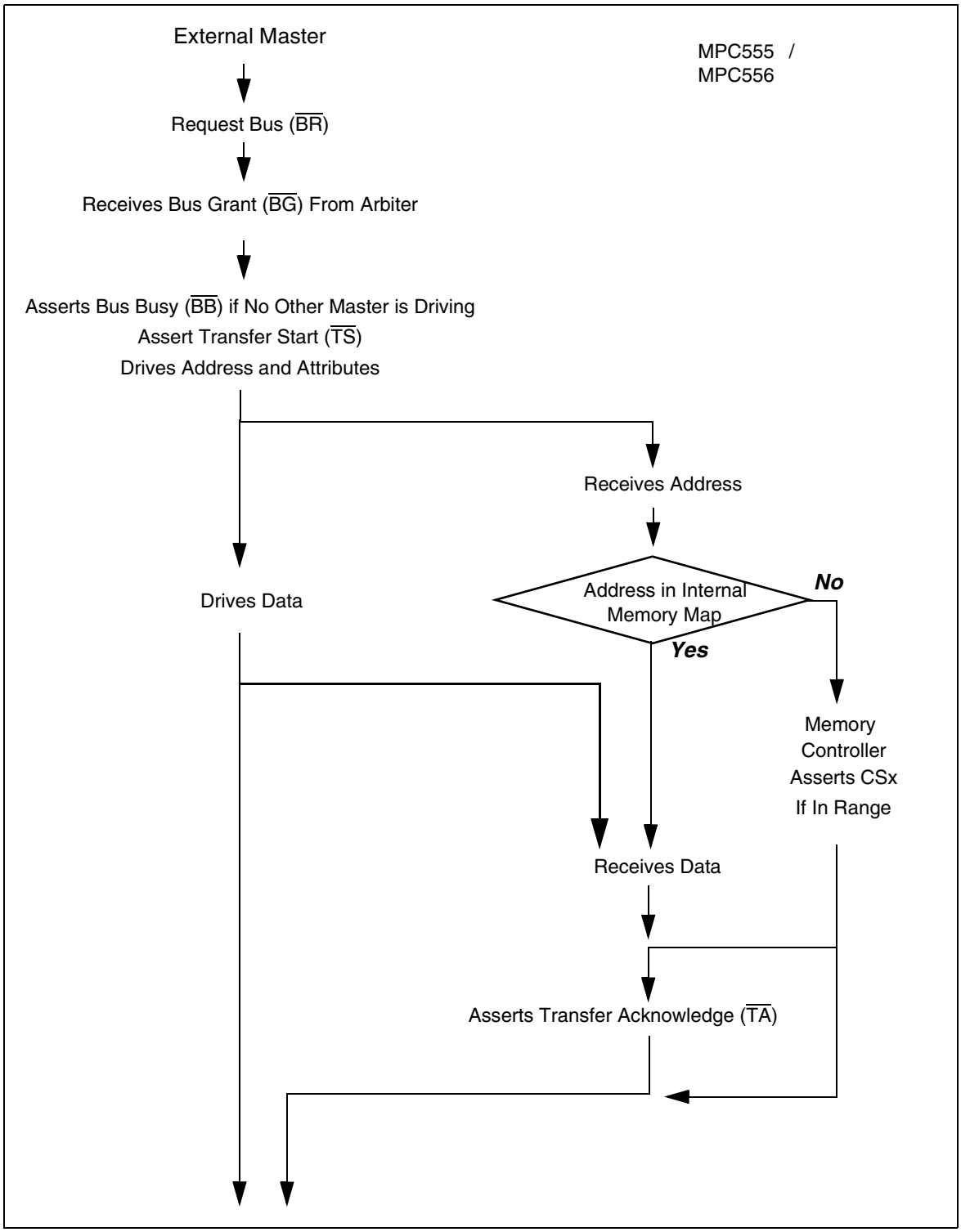


Figure 9-35 Basic Flow of an External Master Write Access

,**Figure 9-36**, **Figure 9-37** and **Figure 9-38** describe read and write cycles from an external master accessing internal space in the MPC555 / MPC556. Note that the mini-

mum number of wait states for such access is two clocks. The accesses in these figures are valid for both peripheral mode and slave mode.

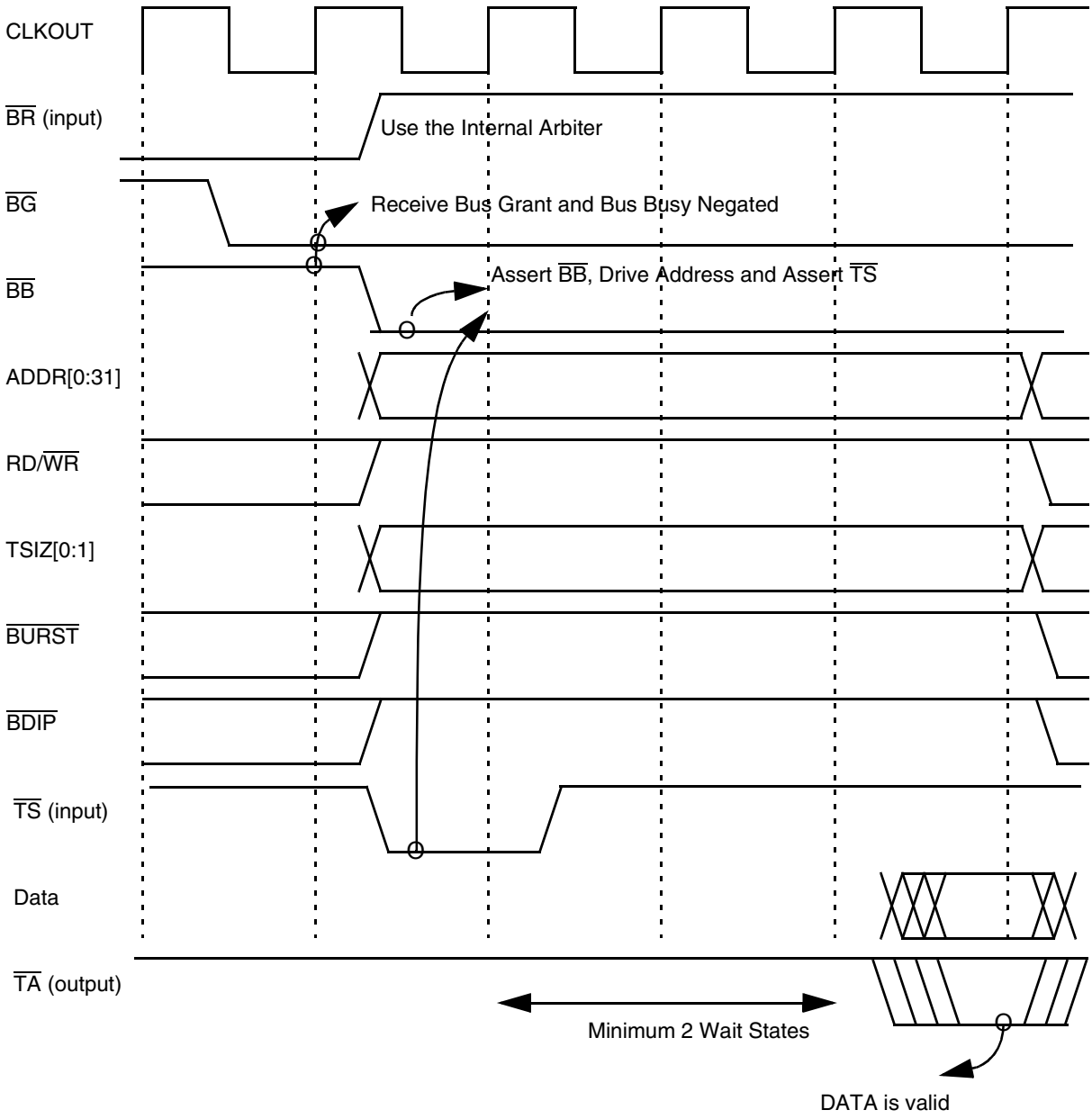


Figure 9-36 Peripheral Mode: External Master Reads from MPC555 / MPC556 — Two Wait States

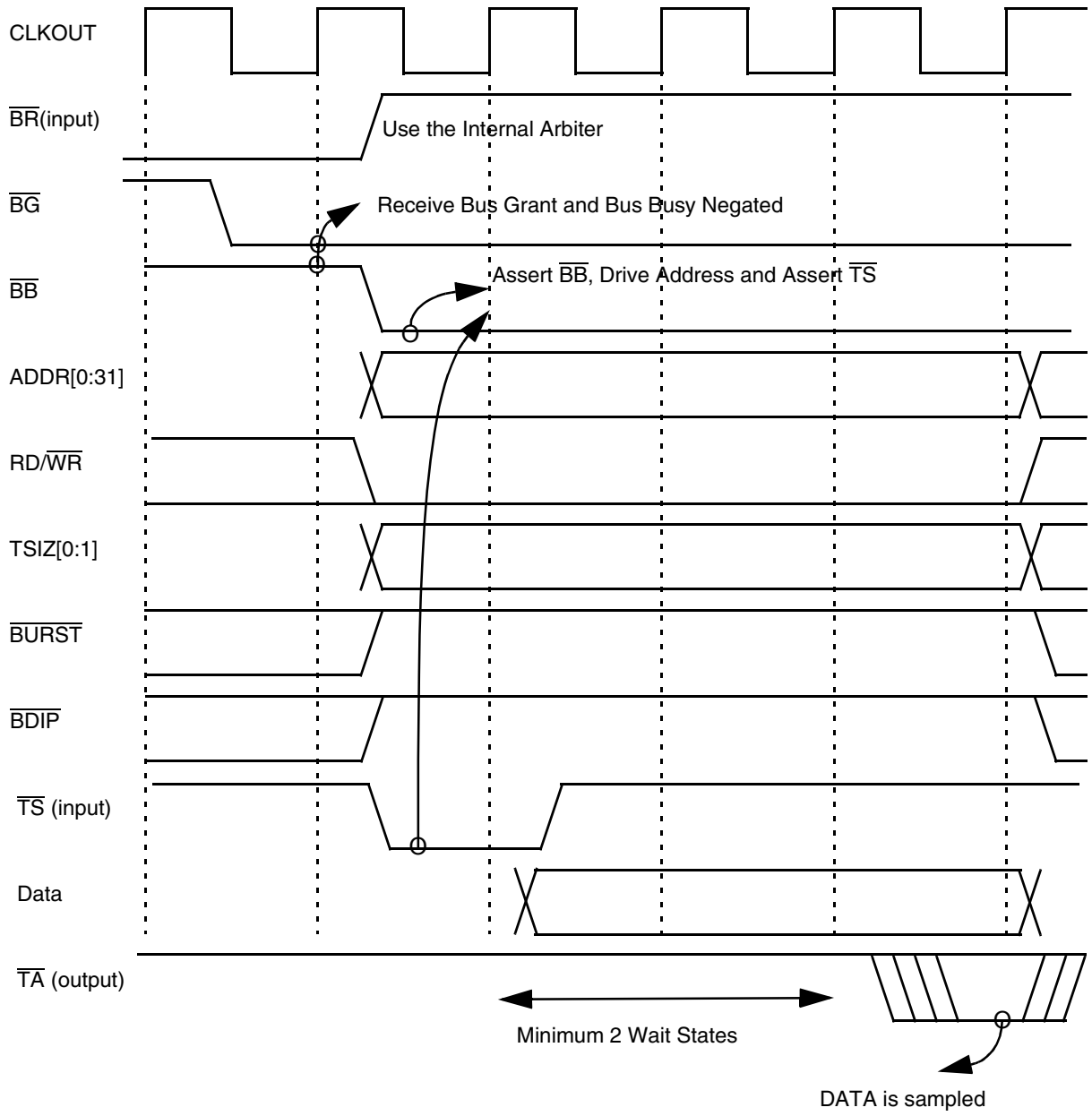


Figure 9-37 Peripheral Mode: External Master Writes to MPC555 / MPC556; Two Wait States

9.5.12 Contention Resolution on External Bus

When the MPC555 / MPC556 is in slave mode, external master access to the MPC555 / MPC556 internal bus can be terminated with relinquish and retry in order to allow a pending internal-to-external access to be executed. The $\overline{\text{RETRY}}$ signal functions as an output that signals the external master to release the bus ownership and retry the access after one clock.

Figure 9-38 describes the flow of an external master retried access. **Figure 9-39** shows the timing when an external access is retried and a pending internal-to-external access follows.

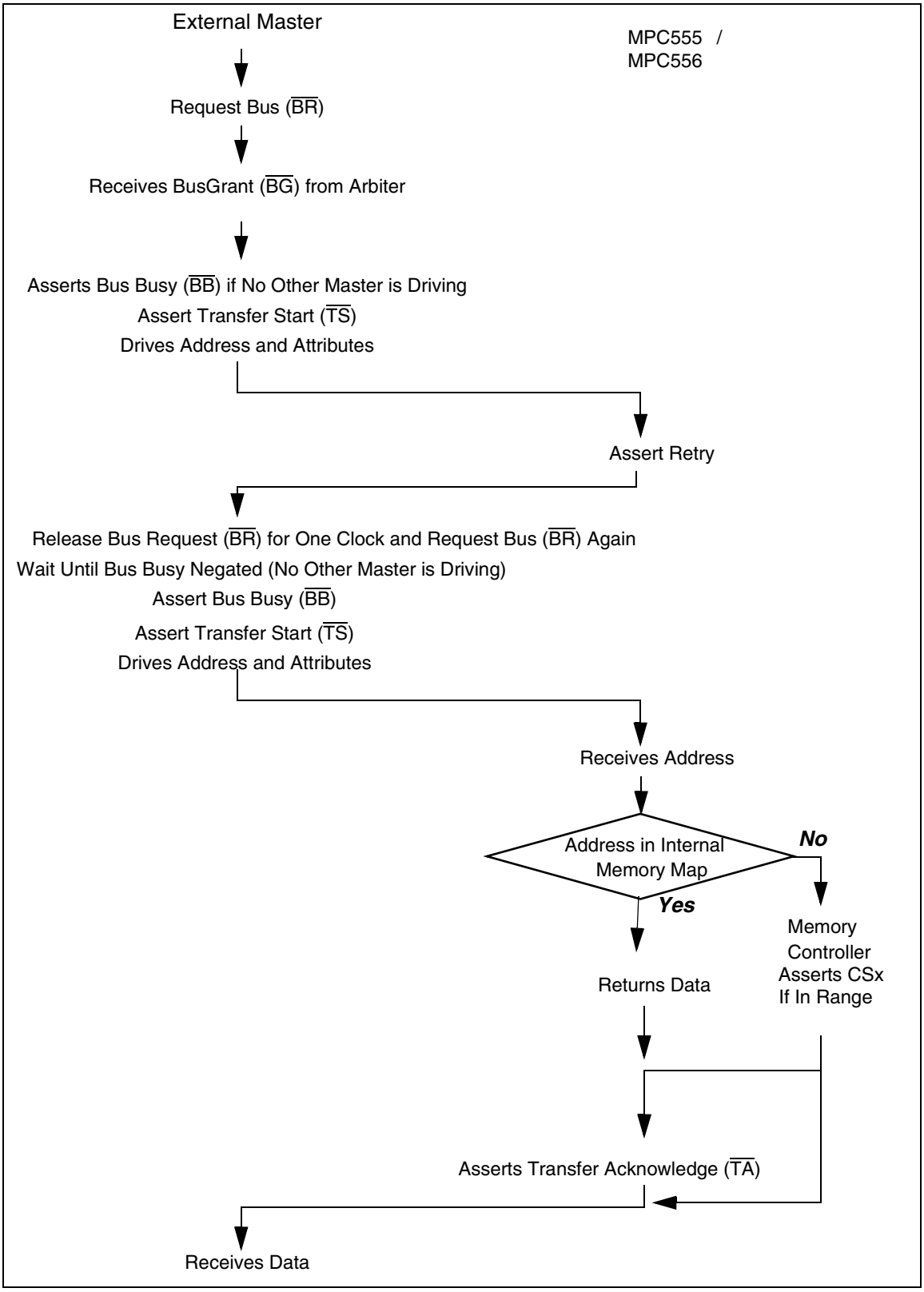
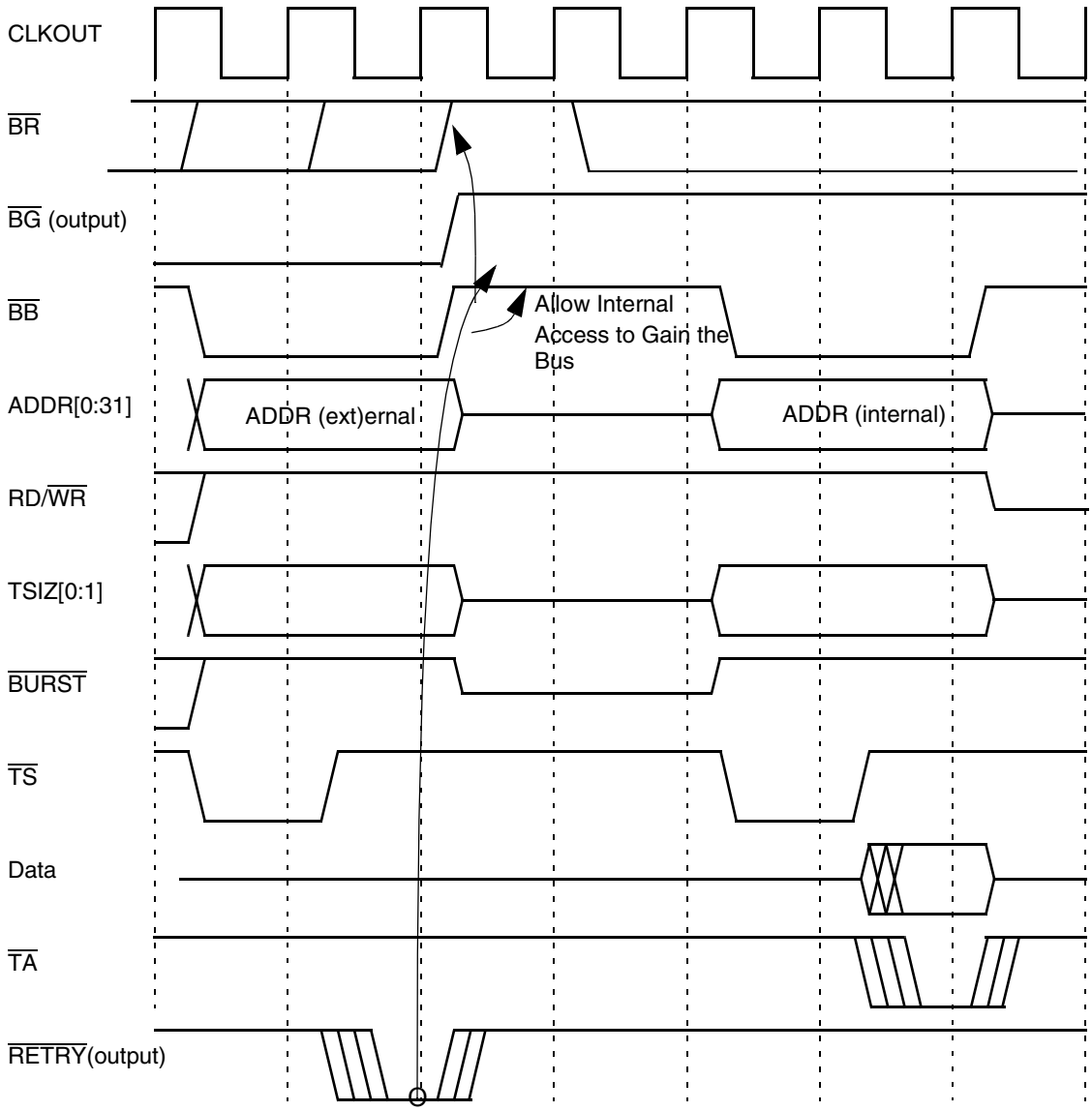


Figure 9-38 Flow of Retry of External Master Read Access



Note: the delay for the internal to external cycle may be one clock or greater.

Figure 9-39 Retry of External Master Access (Internal Arbiter)

9.5.13 Show Cycle Transactions

Show cycles are accesses to the CPU's internal bus devices. These accesses are driven externally for emulation, visibility, and debugging purposes. A show cycle can have one address phase and one data phase, or just an address phase in the case of instruction show cycles. The cycle can be a write or a read access. The data for both the read and write accesses should be driven by the bus master. (This is different from normal bus read and write accesses.) The address and data of the show cycle must each be valid on the bus for one clock. The data phase must not require a transfer ac-

knowledge to terminate the bus show cycle. In a burst show cycle only the first data beat is shown externally. Refer to **Table 9-8** for show cycle transaction encodings.

Instruction show cycle bus transactions have the following characteristics (see **Figure 9-40**):

- One clock cycle
- Address phase only
- $\overline{STS}$ assertion only (no $\overline{TA}$ assertion)

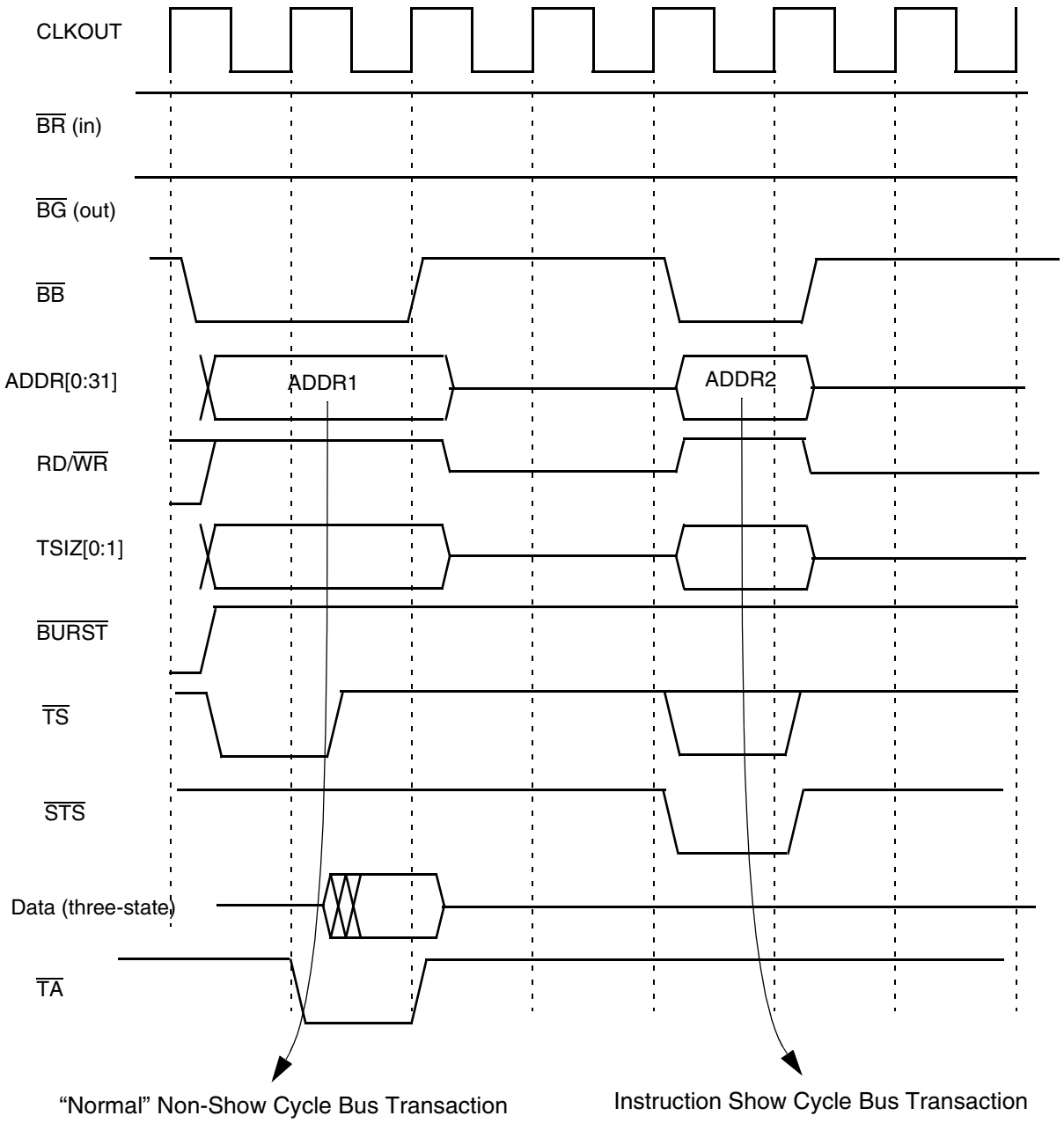


Figure 9-40 Instruction Show Cycle Transaction

Both read and write data show cycles have the following characteristics (see **Figure 9-41**):

- Two clock cycle duration
- Address valid for two clock cycles
- Data is valid only in the second clock cycle
- $\overline{STS}$ signal only is asserted (no $\overline{TA}$ or $\overline{TS}$)

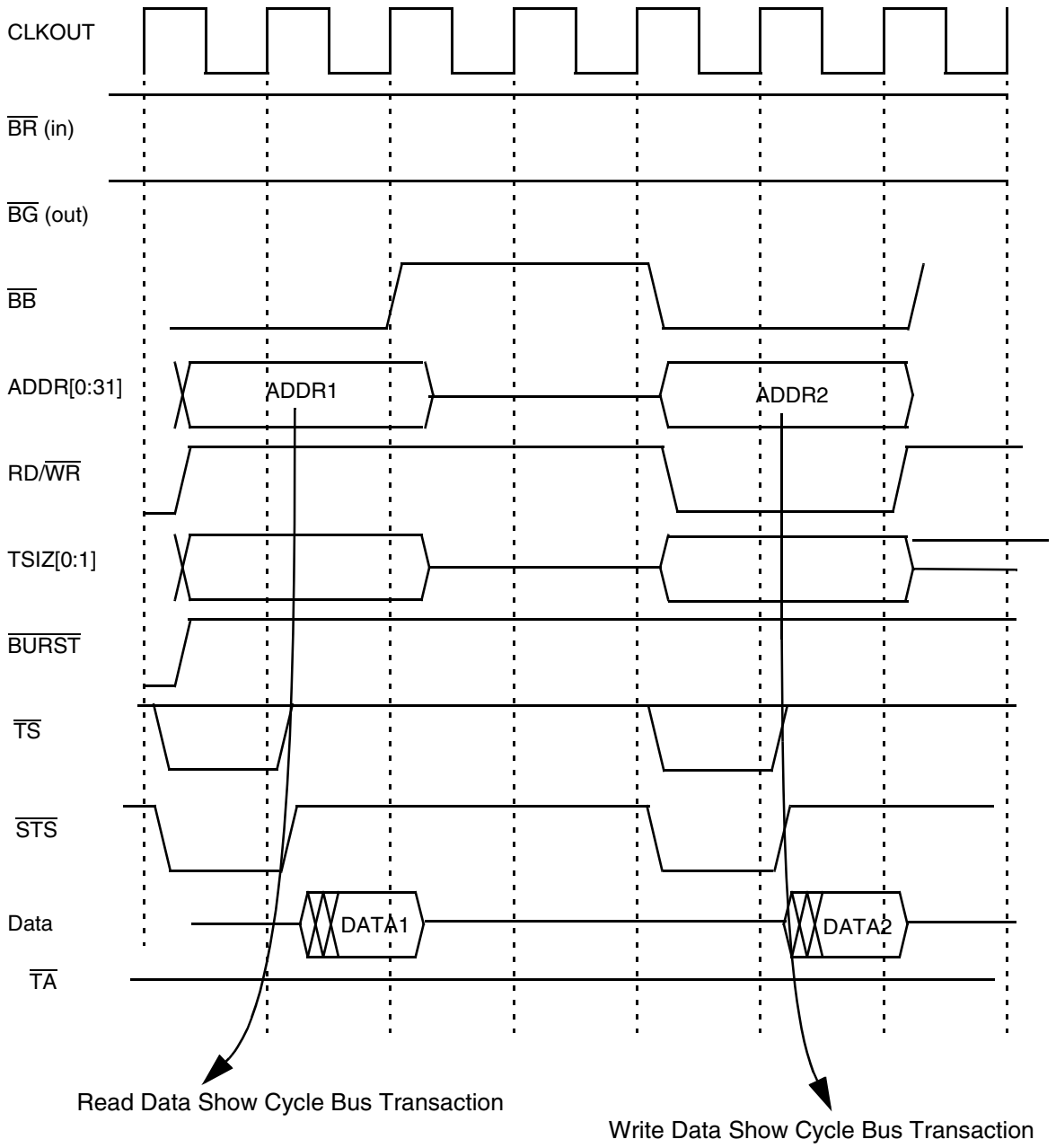


Figure 9-41 Data Show Cycle Transaction

SECTION 10
MEMORY CONTROLLER

The memory controller generates interface signals to support a glueless interface to external memory and peripheral devices. It supports four regions, each with its own programmed attributes. The four regions are reflected on four chip-select pins. Read and write strobes are also provided.

The memory controller operates in parallel with the external bus interface to support external cycles. When an access to one of the memory regions is initiated, the memory controller takes ownership of the external signals and controls the access until its termination. Refer to **Figure 10-1**.

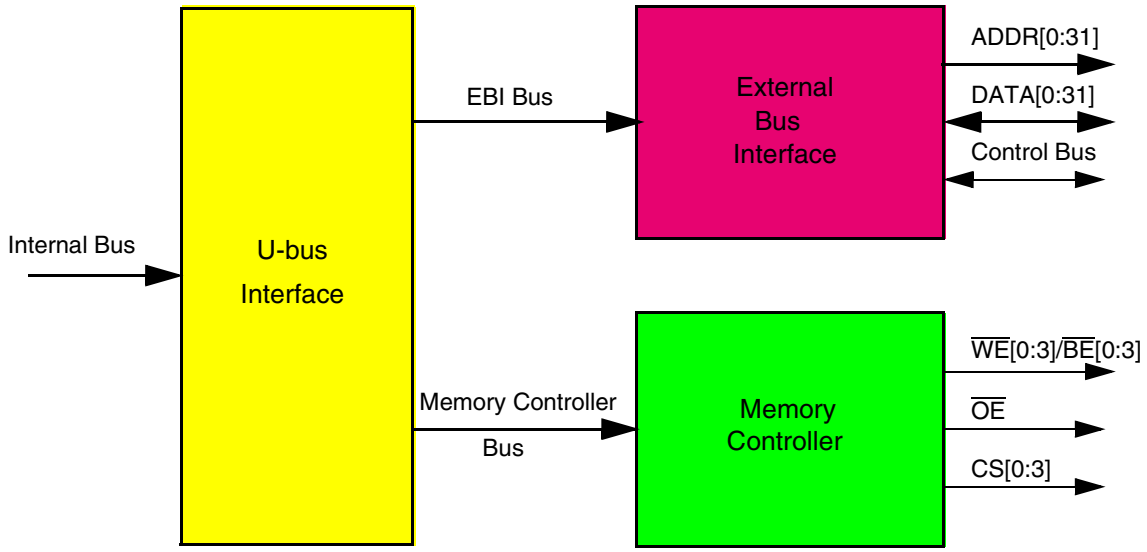


Figure 10-1 Memory Controller Function Within the USIU

10.1 Overview

The memory controller provides a glueless interface to EPROM, static RAM (SRAM), Flash EPROM (FEPROM), and other peripherals. The general-purpose chip-selects are available on lines $\overline{CS}$[0] through $\overline{CS}$[3]. $\overline{CS}$[0] also functions as the global (boot)

chip-select for accessing the boot flash EEPROM. The chip select allows zero to 30 wait states.

Figure 10-2 is a block diagram of the MPC555 / MPC556 memory controller.

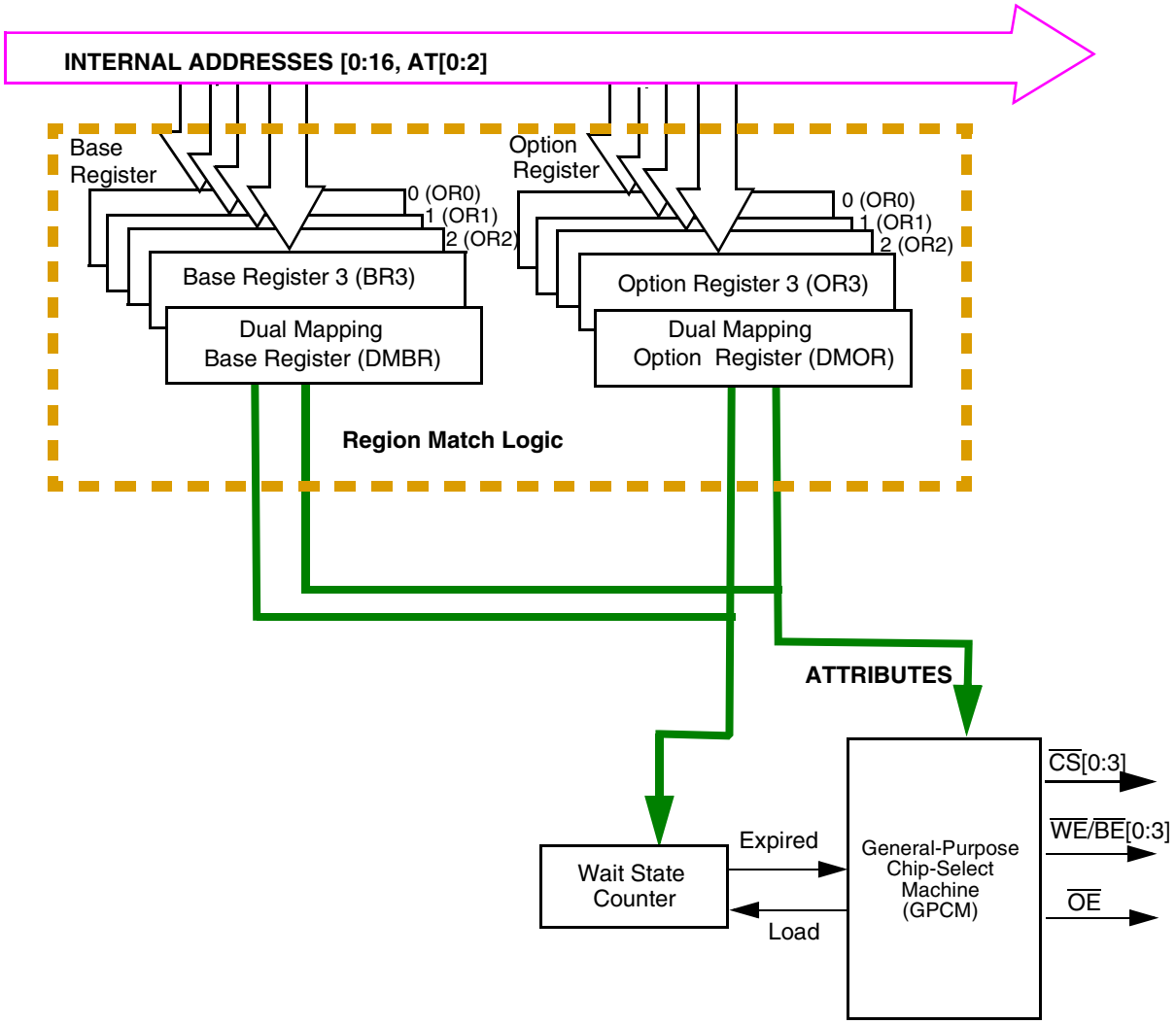


Figure 10-2 Memory Controller Block Diagram

Most memory controller features are common to all four banks. (For features unique to the $\overline{CS}$[0] bank, refer to **10.4 Global (Boot) Chip-Select Operation**.) A full 32-bit address decode for each memory bank is possible with 17 bits having address masking. The full 32-bit decode is available, even if all 32 address bits are not sent to the MPC555 / MPC556 pins.

Each memory bank includes a variable block size of 32 Kbytes, 64 Kbytes and up to 4 Gbytes. Each memory bank can be selected for read-only or read/write operation. The access to a memory bank can be restricted to certain address type codes for system protection. The address type comparison occurs with a mask option as well.

From zero to 30 wait states can be programmed with $\overline{TA}$ generation. Four byte-write and read-enable signals ($\overline{WE}/\overline{BE}$[0:3]) are available for each byte that is written to memory. An output enable ($\overline{OE}$) signal is provided to eliminate external glue logic. A memory transfer start ($\overline{MTS}$) strobe permits one master on a bus to access external memory through the chip selects on another.

The memory controller functionality allows MPC555 / MPC556-based systems to be built with little or no glue logic. A minimal system using no glue logic is shown in **Figure 10-3**. In this example $\overline{CS}$[0] is used for the 16-bit boot EPROM and $\overline{CS}$[1] is used for the 32-bit SRAM. The $\overline{WE}/\overline{BE}$[0:3] signals are used both to program the EPROM and to enable write access to various bytes in the RAM.

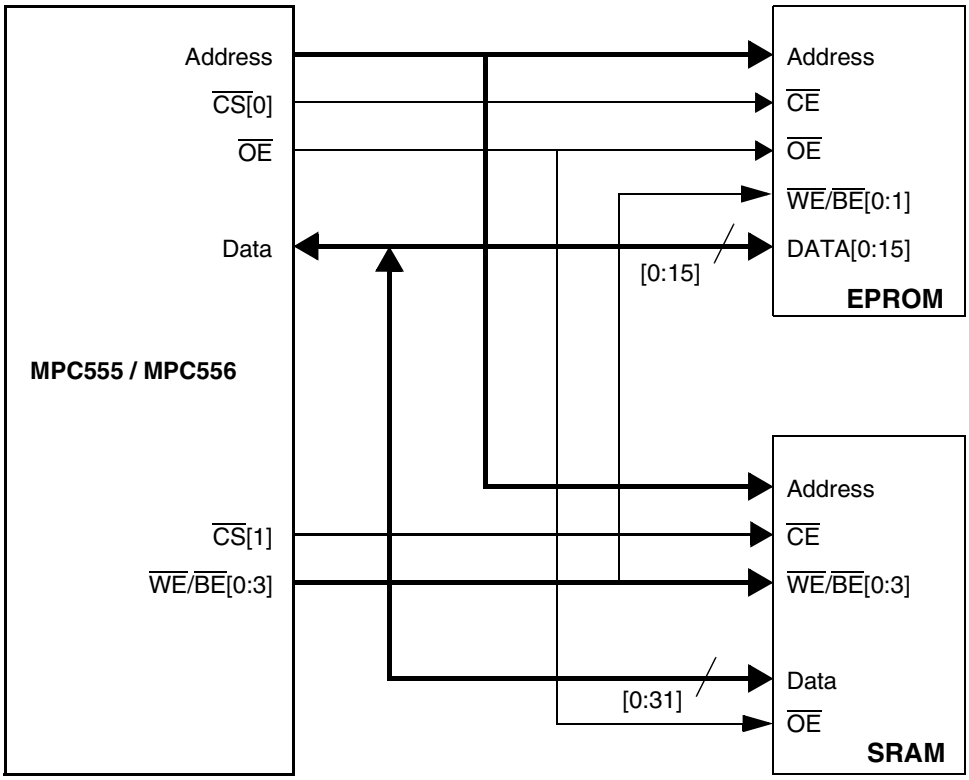


Figure 10-3 MPC555 / MPC556 Simple System Configuration

10.2 Memory Controller Architecture

The memory controller consists of a basic machine that handles the memory access cycle: the general-purpose chip-select machine (GPCM).

When a new access to external memory is requested by any of the internal masters, the address of the transfer (with 17 bits having mask) and the address type (with 3 bits having mask) are compared to each one of the valid banks defined in the memory controller. Refer to **Figure 10-4**.

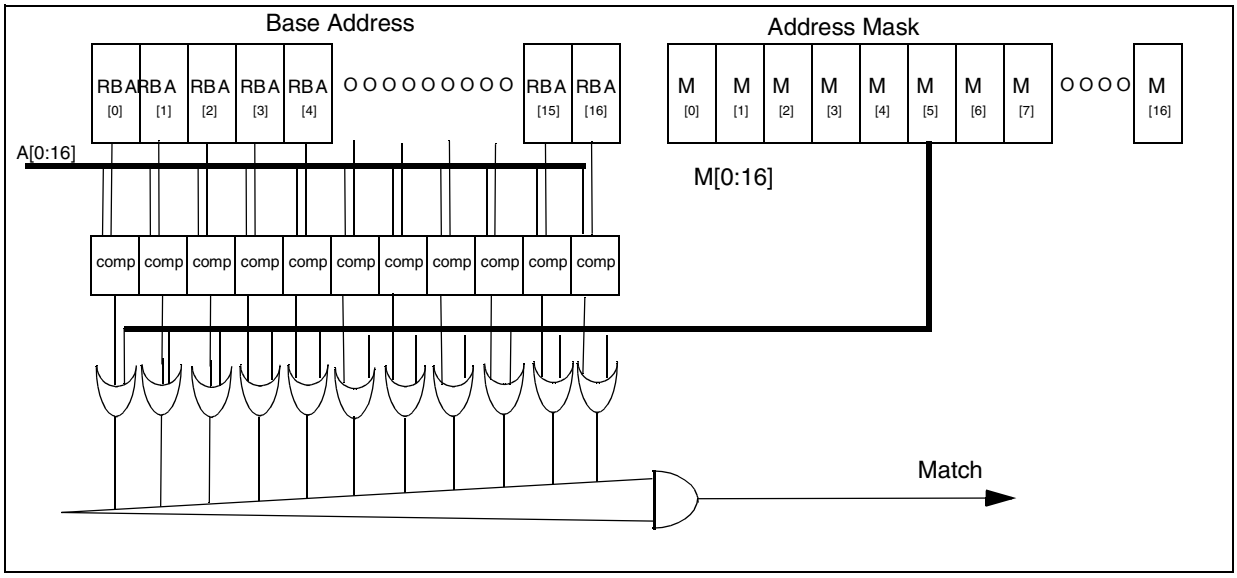


Figure 10-4 Bank Base Address and Match Structure

When a match is found on one of the memory banks, its attributes are selected for the functional operation of the external memory access:

- Read-only or read/write operation
- Number of wait states for a single memory access, and for any beat in a burst access
- Burst-inhibit indication. Internal burst requests are still possible during burst-inhibited cycles; the memory controller emulates the burst cycles
- Port size of the external device

Note that if more than one region matches the internal address supplied, then the lowest region is selected to provide the attributes and the chip select.

10.2.1 Associated Registers

Status bits for each memory bank are found in the memory control status register (MSTAT). The MSTAT reports write-protect violations for all the banks.

Each of the four banks has a base register (BR) and an option register (OR). The BRx and ORx registers contain the attributes specific to bank x. The base register contains a valid bit (V) that indicates that the register information for that chip select is valid.

10.2.2 Port Size Configuration

The memory controller supports dynamic bus sizing. Defined 8-bit ports can be accessed as odd or even bytes. Defined 16-bit ports, when connected to data bus lines zero to 15, can be accessed as odd bytes, even bytes, or even half-words. Defined 32-bit ports can be accessed as odd bytes, even bytes, odd half-words, even half-words,

or words on word boundaries. The port size is specified by the PS bits in the base register.

10.2.3 Write-Protect Configuration

The WP bit in each base register can restrict write access to its range of addresses. Any attempt to write this area results in the associated WPER bit being set in the MSTAT.

If an attempt to access an external device results in a write-protect violation, the memory controller considers the access to be no match. No chip-select line is asserted externally, and the memory controller does not terminate the cycle. The external bus interface generates a normal cycle on the external bus. Since the memory controller does not acknowledge the cycle internally, the cycle may be terminated by external logic asserting $\overline{TA}$ or by the on-chip bus monitor asserting $\overline{TEA}$.

10.2.4 Address and Address Space Checking

The base address is written to the BR. The address mask bits for the address are written to the OR. The address type access value, if desired, is written to the AT bits in the BR. The ATM bits in the OR can be used to mask this value. If address type checking is not desired, program the ATM bits to zero.

Each time an external bus cycle access is requested, the address and address type are compared with each one of the banks. If a match is found, the attributes defined for this bank in its BR and OR are used to control the memory access. If a match is found in more than one bank, the lowest bank matched handles the memory access (e.g., bank zero is selected over bank one). Note that when an external master accesses a slave on the bus, the internal AT[0:2] lines reaching the memory controller are forced to 100.

10.2.5 Burst Support

Burst support is for read only. The memory controller supports burst accesses of external burstable memory. To enable bursts, clear the $\overline{BI}$ in the appropriate base register.

Bursts are four beats and non-wrapping. That is, the memory controller executes up to four one-word accesses, but when a modulo four limit is reached, the burst is terminated (even if fewer than four words have been accessed).

When the SIU initiates a burst access, if no match is found in any of the memory controller's regions then a burst access is initiated to the external bus. The termination of each beat for this access is externally controlled (i.e., the user is responsible for terminating each data beat using the bus termination protocol).

To support different types of memory devices, the memory controller supports two types of timing for the $\overline{BDIP}$ signal: normal and late. Note that the $\overline{BDIP}$ pin itself is controlled by the external bus interface logic. Refer to **Figure 9-13** and **Figure 9-14** in **SECTION 9 EXTERNAL BUS INTERFACE**.

If the memory controller is used to support an external master accessing an external device with bursts, the $\overline{BDIP}$ input pin is used to indicate to the memory controller when the burst is terminated.

For addition details, refer to **9.5.3 Burst Transfer**.

10.3 Chip-Select Timing

The GPCM allows a glueless and flexible interface between the MPC555 / MPC556 and SRAM, EPROM, EEPROM, ROM devices and external peripherals. When an address and address type matches the values programmed in the BR and OR for one of the memory controller banks, the attributes for the memory cycle are taken from the OR and BR registers as well. These attributes include the following fields: CSNT, ACS, SCY, BSCY, WP, TRLX, BI, PS, and SETA.

Byte write and read-enable signals ($\overline{WE}$/$\overline{BE}$[0:3]) are available for each byte that is written to or read from memory. An output enable ($\overline{OE}$) signal is provided to eliminate external glue logic for read cycles. Upon system reset, a global (boot) chip select is available. This provides a boot ROM chip select before the system is fully configured. **Table 10-1** summarizes the chip-select timing options.

Table 10-1 Timing Attributes Summary

Timing Attribute	Bits/Fields	Description
Access speed	TRLX	The TRLX (timing relaxed) bit determines strobe timing to be fast or relaxed.
Intercycle space time	EHTR	The EHTR (extended hold time on read accesses) bit is provided for devices that have long disconnect times from the data bus on read accesses. EHTR specifies whether the next cycle is delayed one clock cycle following a read cycle, to avoid data bus contentions. EHTR applies to all cycles following a read cycle except for another read cycle to the same region.
Synchronous or asynchronous device	ACS, CSNT	The ACS (address-to-chip-select setup) and CSNT (chip-select negation time) bits cause the timing of the strobes to be the same as the address bus timing, or cause the strobes to have setup and hold times relative to the address bus.
Wait states	SCY, BSCY, SETA, TRLX	From zero to 15 wait states can be programmed for any cycle that the memory controller generates. The transfer is then terminated internally. In simplest case, the cycle length equals (2 + SCY) clock cycles, where SCY represents the programmed number of wait states (cycle length in clocks). The number of wait states is doubled if the TRLX bit is set. When the SETA (external transfer acknowledge) bit is set, $\overline{TA}$ must be generated externally, so that external hardware determines the number of wait states.

Note that when a bank is configured for $\overline{TA}$ to be generated externally (SETA bit is set) and the TRLX is set, the memory controller requires the external device to provide at least one wait state before asserting $\overline{TA}$ to complete the transfer. In this case, the minimum transfer time is three clock cycles.

The internal $\overline{TA}$ generation mode is enabled if the SETA bit in the OR register is negated. However, if the $\overline{TA}$ pin is asserted externally at least two clock cycles before the

wait states counter has expired, this assertion terminates the memory cycle. When SETA is cleared, it is forbidden to assert external $\overline{TA}$ less than two clocks before the wait states counter expires.

10.3.1 Memory Devices Interface Example

Figure 10-5 describes the basic connection between the MPC555 / MPC556 and a static memory device. In this case $\overline{CS}$x is connected directly to the chip enable ($\overline{CE}$) of the memory device. The $\overline{WE}$/$\overline{BE}$[0:3] lines are connected to the respective $\overline{W}$ in the memory device where each $\overline{WE}$/$\overline{BE}$ line corresponds to a different data byte.

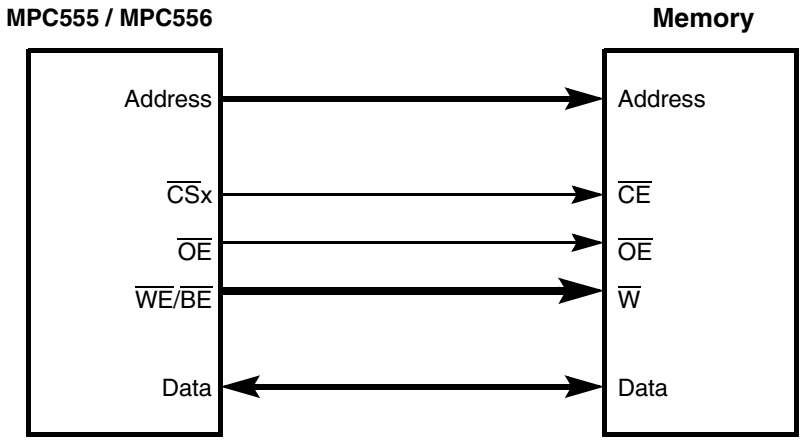
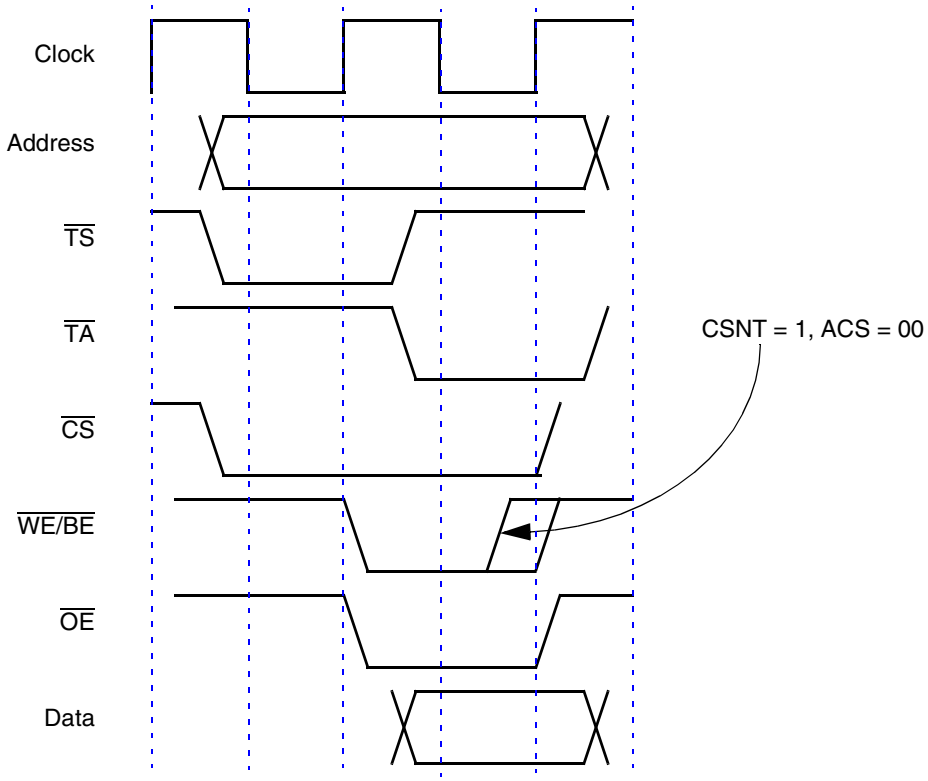


Figure 10-5 MPC555 / MPC556 GPCM–Memory Devices Interface

In **Figure 10-6**, the $\overline{CS}$x timing is the same as that of the address lines output. The strobes for the transaction are supplied by the $\overline{OE}$ and the $\overline{WE}$/$\overline{BE}$ lines (if programmed as $\overline{WE}$/$\overline{BE}$). Because the ACS bits in the corresponding ORx register = 00, $\overline{CS}$ is asserted at the same time that the address lines are valid. Note that because CSNT is set, the $\overline{WE}$ signal is negated a quarter of a clock earlier than normal.



Note: In this and subsequent timing diagrams in this section, the data bus refers to a read cycle. In a write cycle, the data immediately follows TS.

Figure 10-6 Memory Devices Interface Basic Timing (ACS = 00,TRLX = 0)

10.3.2 Peripheral Devices Interface Example

Figure 10-7 illustrates the basic connection between the MPC555 / MPC556 and an external peripheral device. In this case CSx is connected directly to the chip enable (CE) of the memory device and the R/W line is connected to the R/W in the peripheral device. The CSx line is the strobe output for the memory access.

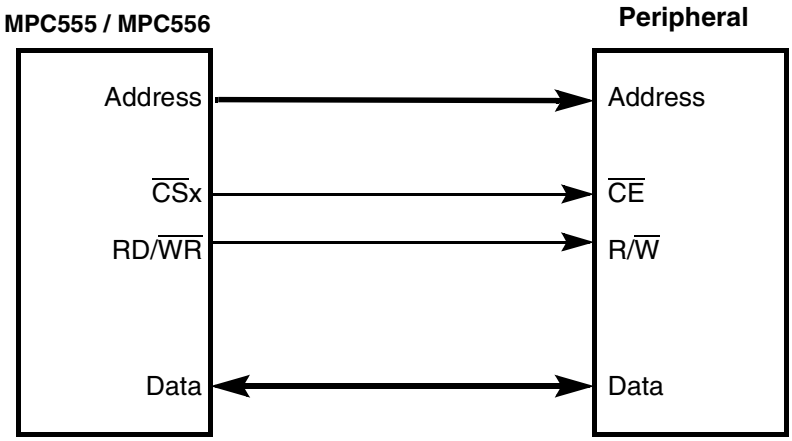


Figure 10-7 Peripheral Devices Interface

The CSx timing is defined by the setup time required between the address lines and the CE line. The memory controller allows the user to specify the CS timing to meet the setup time required by the peripheral device. This is accomplished through the ACS field in the base register. In **Figure 10-8**, the ACS bits are set to 11, so CSx is asserted half a clock cycle after the address lines are valid.

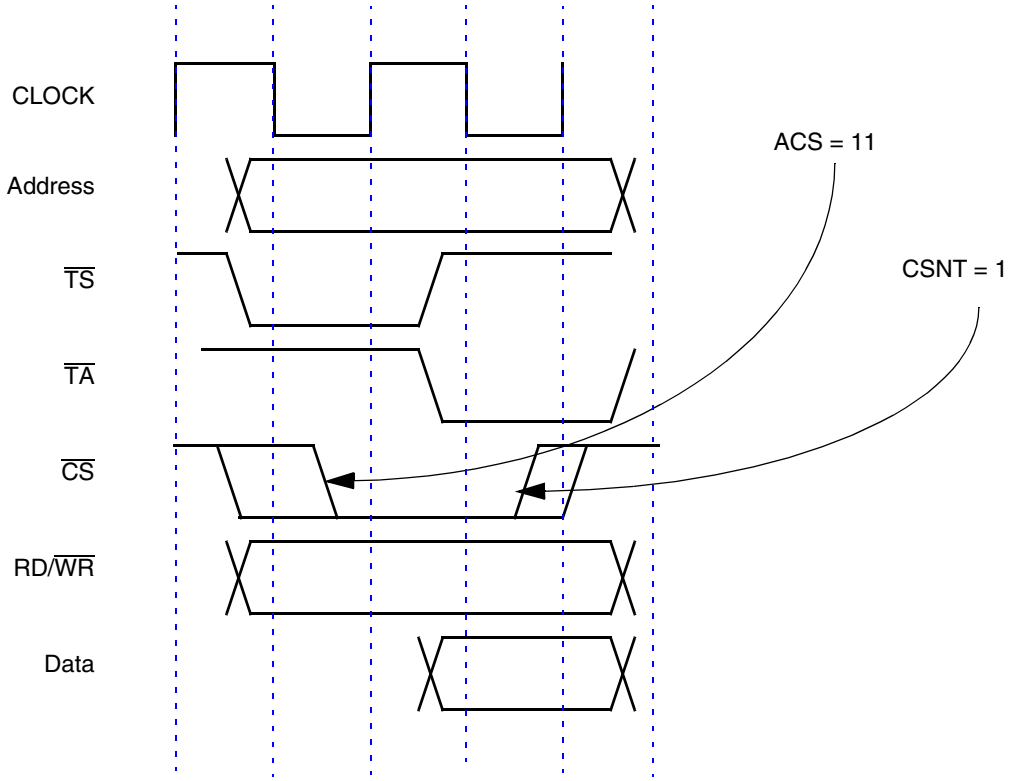


Figure 10-8 Peripheral Devices Basic Timing (ACS = 11,TRLX = 0)

10.3.3 Relaxed Timing Examples

The TRLX field is provided for memory systems that need a more relaxed timing between signals. When TRLX is set and ACS = 0b00, the memory controller inserts an additional cycle between address and strobes ($\overline{CS}$ line and $\overline{WE}$/$\overline{OE}$).

When TRLX and CSNT are both set in a write to memory, the strobe lines ($\overline{WE}$/$\overline{BE}$[0:3] and $\overline{CS}$, if ACS = 0b00) are negated one clock earlier than in the regular case.

Note that in the case of a bank selected to work with external transfer acknowledge (SETA = 1) and TRLX = 1, the memory controller does not support external devices providing $\overline{TA}$ to complete the transfer with zero wait states. The minimum access duration in this case equals three clock cycles.

Figure 10-9 shows a read access with relaxed timing. Note the following:

- Strobes ($\overline{OE}$ and $\overline{CS}$) assertion time is delayed one clock relative to address (TRLX bit set effect).
- Strobe ($\overline{CS}$) is further delayed (half-clock) relative to address due to ACS field being set to 11.
- Total cycle length = 5, is determined as follows:
  - — Two clocks for basic cycle
  - — SCY = 1 determines 1 wait state, which is multiplied by two due to TRLX being set.
  - — Extra clock is added due to TRLX effect on the strobes.

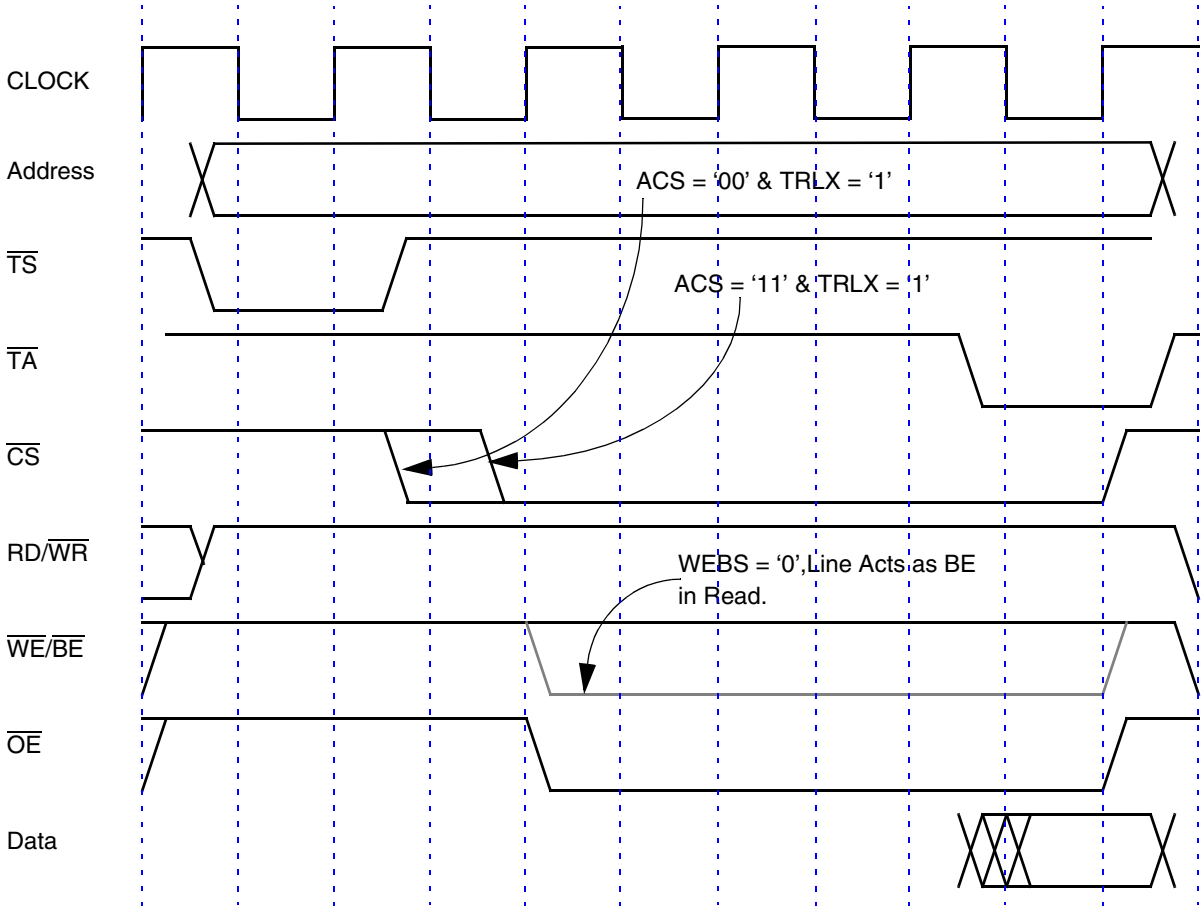


Figure 10-9 Relaxed Timing–Read Access (ACS = 11, SCY = 1, TRLX = 1)

Figure 10-10 through **Figure 10-12** are examples of write accesses using relaxed timing. In **Figure 10-10**, note the following points:

- Because TRLX is set, assertion of the $\overline{CS}$ and $\overline{WE}$ strobes is delayed by one clock cycle.
- $\overline{CS}$ assertion is delayed an additional one quarter clock cycle because ACS = 10.
- The total cycle length = three clock cycles, determined as follows:
  — The basic memory cycle requires two clock cycles.
  — An extra clock cycle is required due to the effect of TRLX on the strobes.

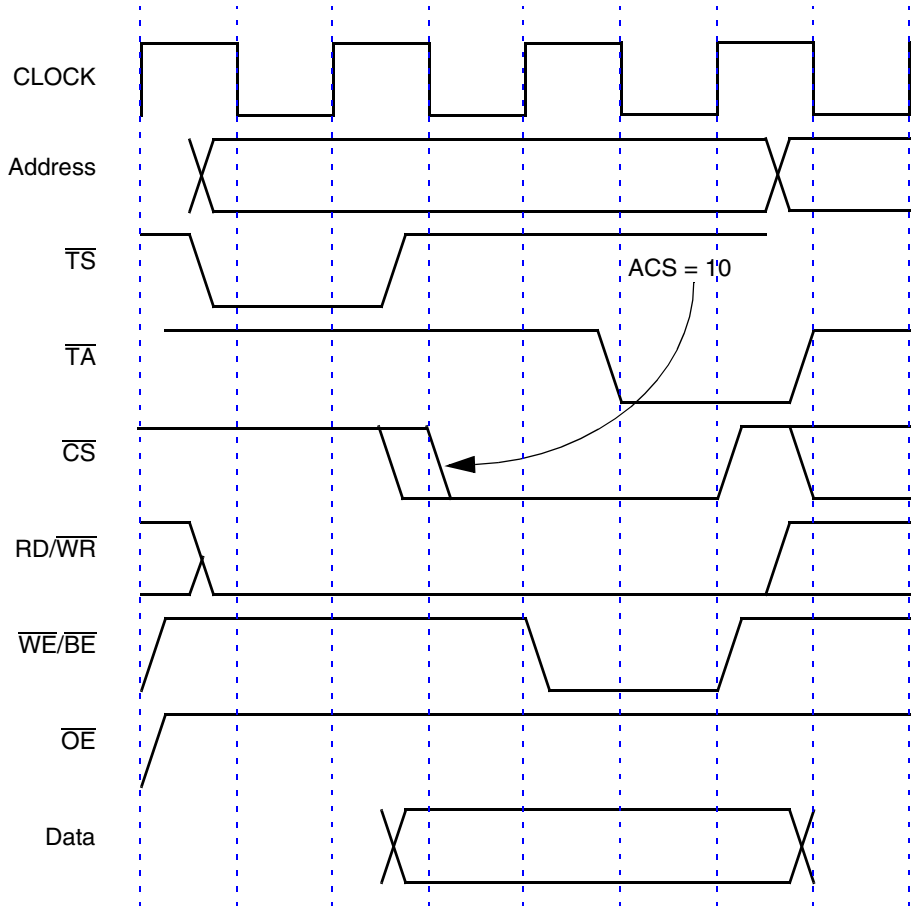


Figure 10-10 Relaxed Timing–Write Access (ACS = 10, SCY = 0, CSNT = 0, TRLX = 1)

In **Figure 10-11**, note the following:

- Because the TRLX bit is set, the assertion of the $\overline{CS}$ and $\overline{WE}$ strobes is delayed by one clock cycle.
- Because ACS = 11, the assertion of CS is delayed an additional half clock cycle.
- Because CSNT = 1, $\overline{WE}$ is negated one clock cycle earlier than normal. (Refer to **Figure 10-6**). The total cycle length is four clock cycles, determined as follows:
  — The basic memory cycle requires two clock cycles.
  — Two extra clock cycles are required due to the effect of TRLX on the assertion and negation of the $\overline{CS}$ and $\overline{WE}$ strobes.

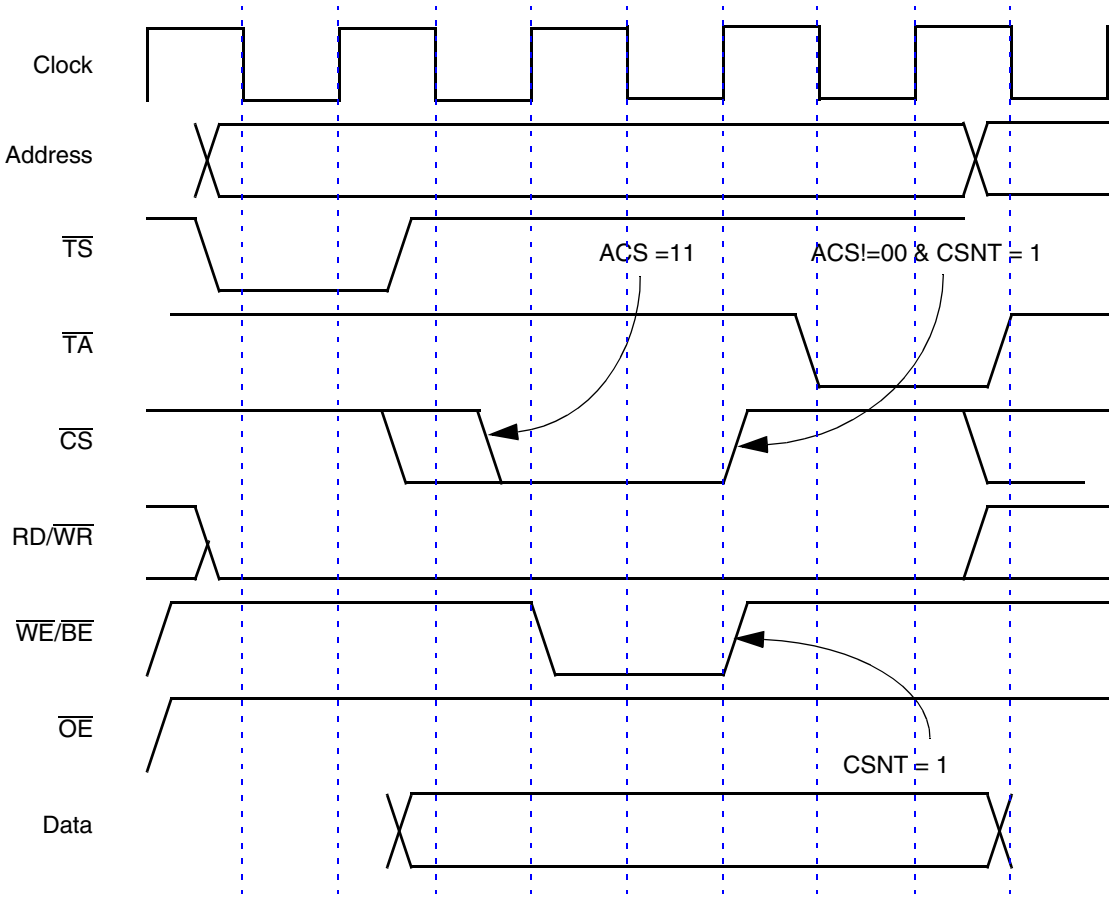


Figure 10-11 Relaxed Timing – Write Access (ACS = 11, SCY = 0, CSNT = 1, TRLX = 1)

In **Figure 10-12**, notice the following:

- Because ACS = 0, TRLX being set does not delay the assertion of the $\overline{CS}$ and $\overline{WE}$ strobes.
- Because CSNT = 1, $\overline{WE}/\overline{BE}$ is negated one clock cycle earlier than normal. (Refer to **Figure 10-6**).
- $\overline{CS}$ is not negated one clock cycle earlier, since ACS = 00.
- The total cycle length is three clock cycles, determined as follows:
  — The basic memory cycle requires two clock cycles.
  — One extra clock cycle is required due to the effect of TRLX on the negation of the $\overline{WE}/\overline{BE}$ strobes.

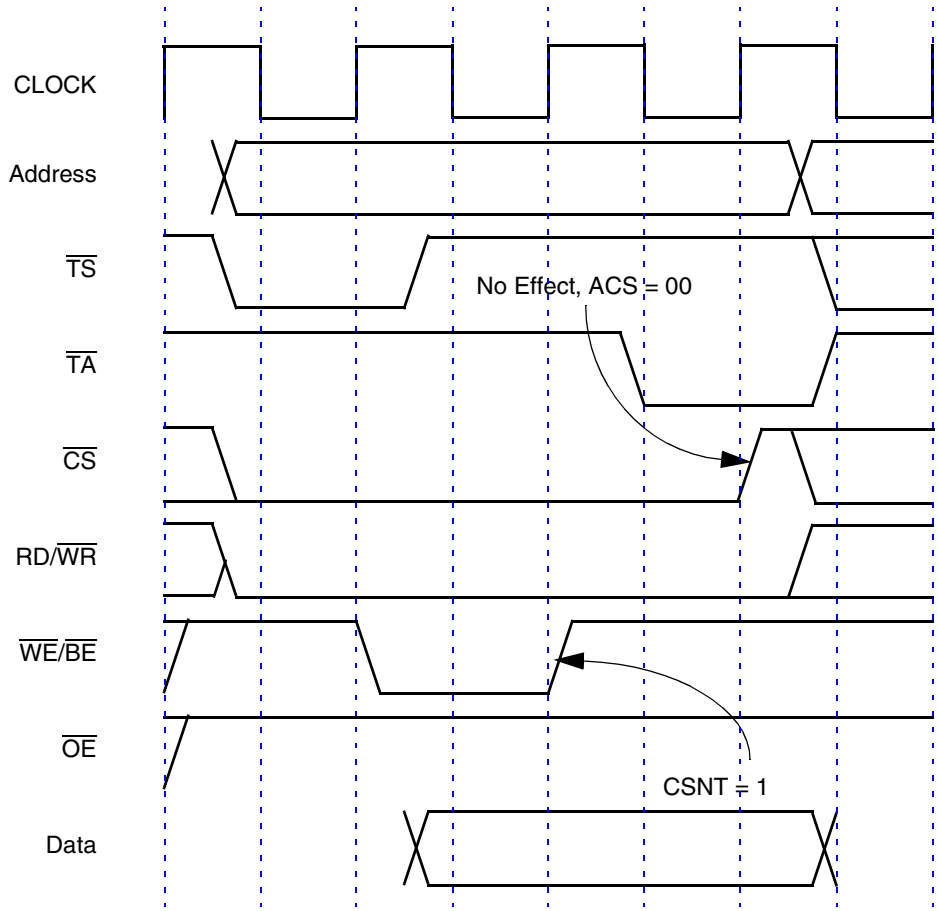


Figure 10-12 Relaxed Timing – Write Access (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1

10.3.4 Extended Hold Time on Read Accesses

For devices that require a long disconnection time from the data bus on read accesses, the bit EHTR in the corresponding OR register can be set. In this case any MPC555 / MPC556 access to the external bus following a read access to the referred memory bank is delayed by one clock cycle unless it is a read access to the same bank. **Figure**

10-13 through **Figure 10-16** show the effect of the EHTR bit on memory controller timing.

Figure 10-13 shows a write access following a read access. Because EHTR = 0, no extra clock cycle is inserted between memory cycles.

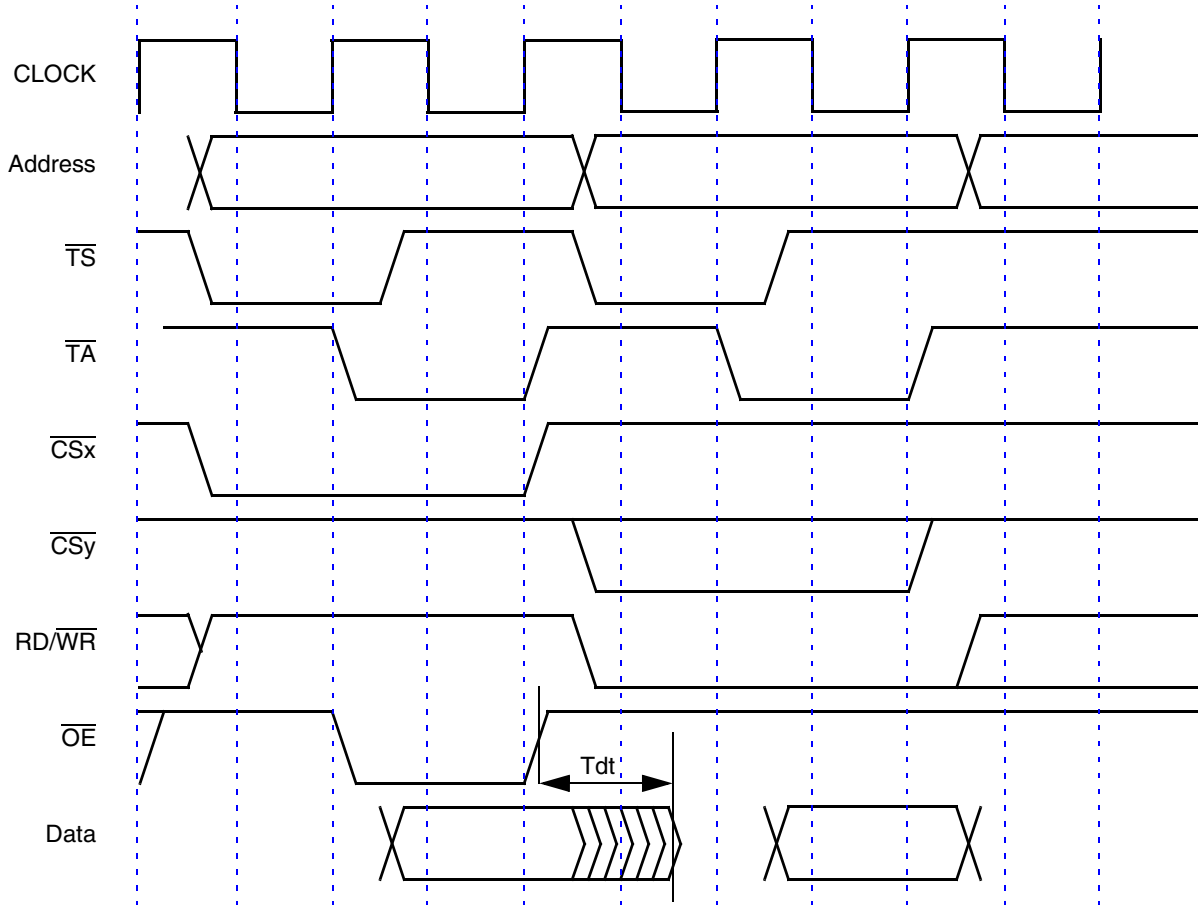


Figure 10-13 Consecutive Accesses (Write After Read, EHTR = 0)

Figure 10-14 shows a write access following a read access when EHTR = 1. An extra clock is inserted between the cycles. For a write cycle following a read, this is true regardless of whether both accesses are to the same region.

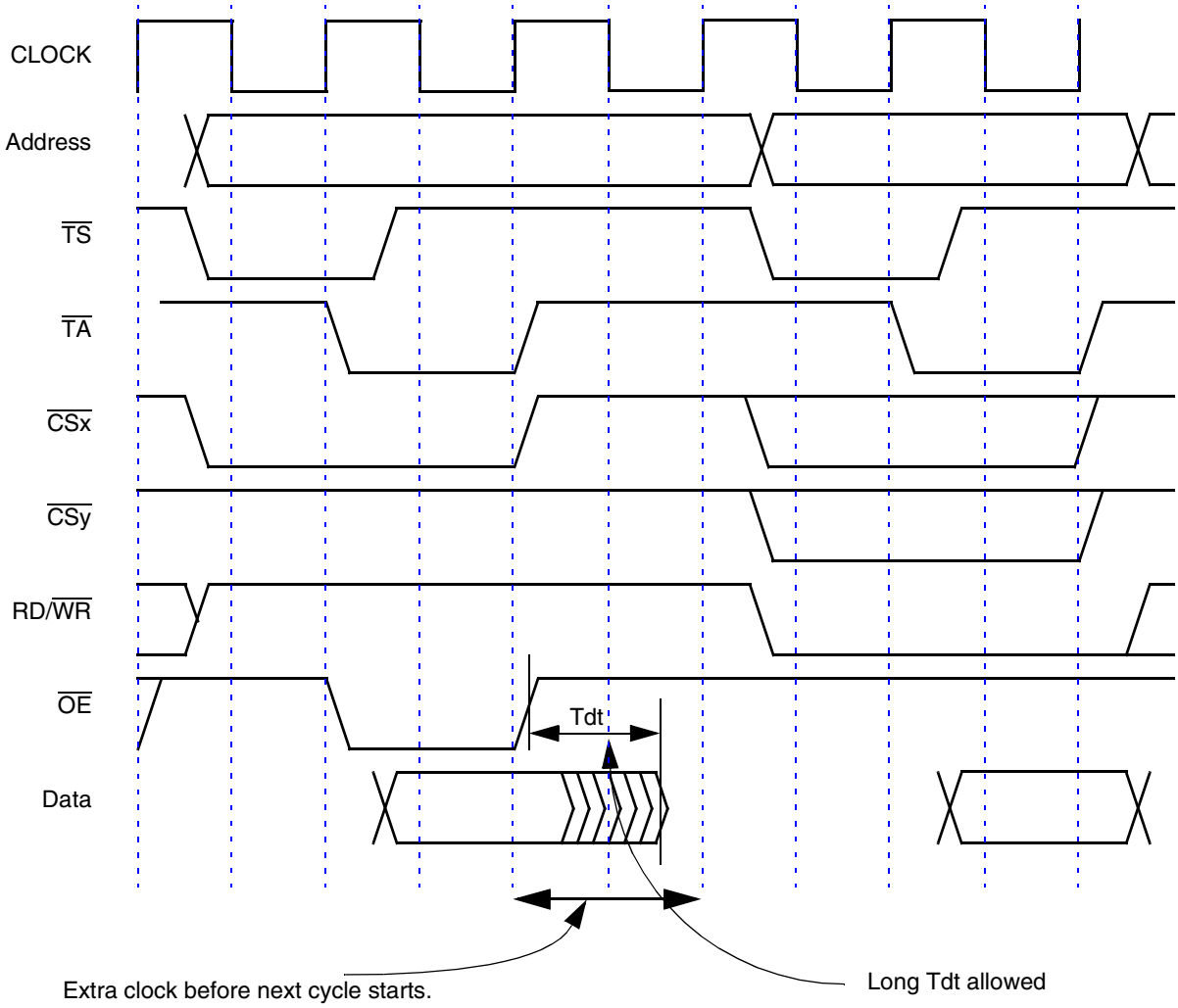


Figure 10-14 Consecutive Accesses (Write After Read, EHTR = 1)

Figure 10-15 shows consecutive accesses from different banks. Because EHTR = 1 (and the accesses are to different banks), an extra clock cycle is inserted.

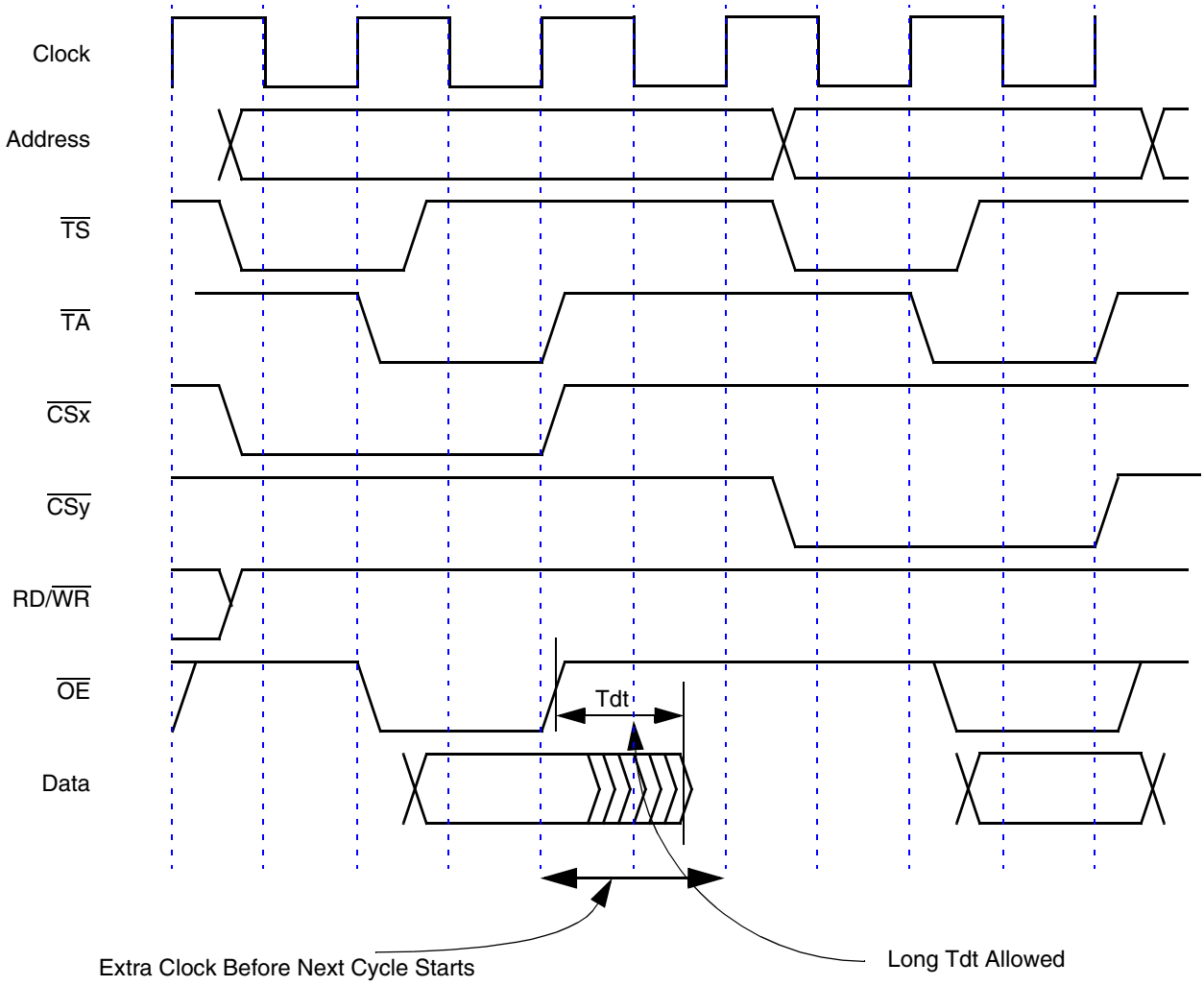


Figure 10-15 Consecutive Accesses (Read After Read From Different Banks, EHTR = 1)

Figure 10-16 shows two consecutive read cycles from the same bank. Even though EHTR = 1, no extra clock cycle is inserted between the memory cycles. (In the case of two consecutive read cycles to the same region, data contention is not a concern.)

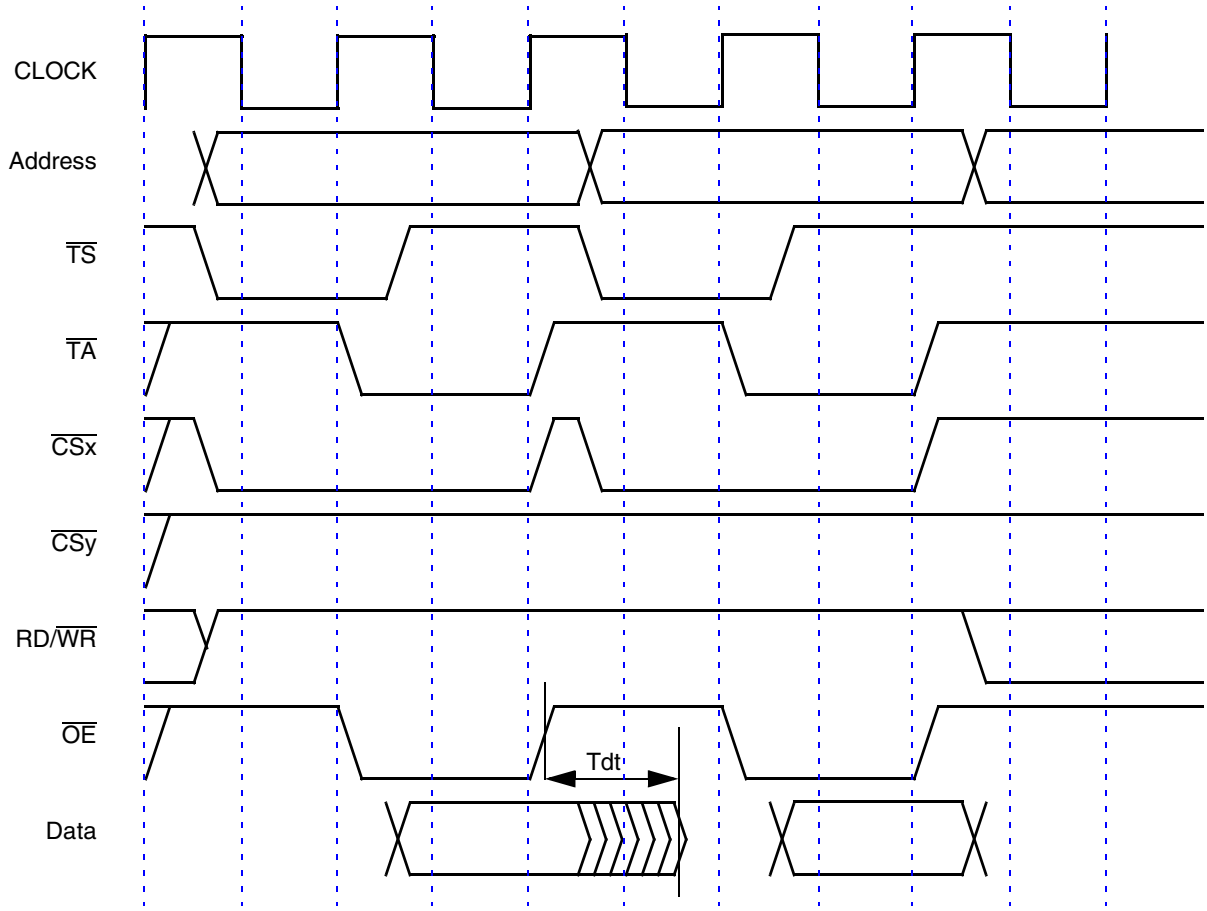


Figure 10-16 Consecutive Accesses (Read After Read From Same Bank, EHTR = 1)

10.3.5 Summary of GPCM Timing Options

Table 10-2 summarizes the different combinations of timing options.

Table 10-2 Programming Rules for Strobes Timing

TRLX	Access Type	ACS	CSNT	Address to $\overline{CS}$ Asserted	$\overline{CS}$ Negated to Add/Data Invalid	Address to $\overline{WE/BE}$ or $\overline{OE}$ Asserted	$\overline{WE/BE}$ Negated to Add/Data Invalid	$\overline{OE}$ Negated to Add/Data Invalid	Total Number of Cycles
0	Read	00	X	0	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	Read	10	X	1/4 * clock	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	Read	11	X	1/2 * clock	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	Write	00	0	0	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	Write	10	0	1/4 * clock	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	Write	11	0	1/2 * clock	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	Write	00	1	0	1/4 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
0	Write	10	1	1/4 * clock	1/2 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
0	Write	11	1	1/2 * clock	1/2 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
1	Read	00	X	0	1/4 * clock	3/4 clock	X	1/4 * clock	2 + 2 * SCY
1	Read	10	X	(1 + 1/4) * clock	1/4 * clock	(1 + 3/4) * clock	X	1/4 * clock	3 + 2 * SCY
1	Read	11	X	(1 + 1/2) * clock	1/4 * clock	(1 + 3/4) * clock	X	1/4 * clock	3 + 2 * SCY
1	Write	00	0	0	1/4 * clock	3/4 clock	1/4 * clock	X	2 + 2 * SCY
1	Write	10	0	(1 + 1/4) * clock	1/4 * clock	(1 + 3/4) * clock	1/4 * clock	X	3 + 2 * SCY
1	Write	11	0	(1 + 1/2) * clock	1/4 * clock	(1 + 3/4) clock	1/4 * clock	X	3 + 2 * SCY
1	Write	00	1	0	1/4 * clock	3/4 clock	(1 + 1/2) * clock	X	3 + 2 * SCY
1	Write	10	1	(1 + 1/4) * clock	(1 + 1/2) * clock	(1 + 3/4) clock	(1 + 1/2) * clock	X	4 + 2 * SCY
1	Write	11	1	(1 + 1/2) * clock	(1 + 1/2) * clock	(1 + 3/4) clock	(1 + 1/2) * clock	X	4 + 2 * SCY

NOTE: Timing in this table refers to the typical timing only. Consult the electrical characteristics for exact worst-case timing values. 1/4 clock actually means 0 to 1/4 clock, 1/2 clock means 1/4 to 1/2 clock.

Additional timing rules not covered in **Table 10-2** include the following:

- If SETA = 1, an external $\overline{TA}$ signal is required to terminate the cycle.
- If TRLX = 1 and SETA = 1, the minimum cycle length = 3 clock cycles (even if SCY = 0000)
- If TRLX = 1, the number of wait states = 2 * SCY & 2 * BSCY
- If EHTR = 1, an extra (idle) clock cycle is inserted between a read cycle and a following read cycle to another region, or between a read cycle and a following write cycle to any region.
- If LBDIP = 1 (late $\overline{BDIP}$ assertion), the $\overline{BDIP}$ pin is asserted only after the number of wait states for the first beat in a burst have elapsed. See **Figure 9-13** in **SECTION 9 EXTERNAL BUS INTERFACE** as well as **9.5.4 Burst Mechanism**. Note that this function can operate only when the cycle termination is internal, using the number of wait states programmed in one of the ORx registers

10.4 Global (Boot) Chip-Select Operation

Global (boot) chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{CS}$[0] is the global chip-select output. Its operation differs from that of the other external chip-select outputs following a system reset. When the RCPU begins accessing memory after a system reset, $\overline{CS}$[0] is asserted for every address, unless an internal device (register) is accessed.

The global chip select provides a programmable port size at system reset using the reset BPS pins ([4:5]) of the reset configuration word, allowing a boot ROM to be located anywhere in the address space. For more information, see **7.5.2 Hard Reset Configuration Word**. The global chip select does not provide write protection and responds to all address types. $\overline{CS}$[0] operates in this way until the first write to the $\overline{CS}$[0] option register (OR0). The pin can be programmed to continue decoding a range of addresses after this write, provided the preferred address range is first loaded into base register zero. After the first write to OR0, the global chip select can only be restarted with a system reset.

The memory controller operates in boot mode until the user modifies the values in OR0 and BR to the ones desired.

Table 10-3 shows the initial values of the “boot bank” in the memory controller.

Table 10-3 Boot Bank Fields Values After Hard Reset

Field	Value (Binary)
PS	From reset configuration
WP	0
V	From reset configuration
AM[0:16]	0 0000 0000 0000 0000
ATM[0:2]	000
CSNT	0
ACS[0:1]	00
BI	1
SCY[0:3]	1111
BSCY[0:2]	011
SETA	0
TRLX	0

NOTE

If the MPC555 / MPC556 is configured (in the reset configuration word) to use the internal flash EEPROM as boot memory $\overline{CS}$[0] is not asserted.

10.5 Write and Byte Enable Signals

The GPCM determines the timing and value of the $\overline{WE}/\overline{BE}$ signals if allowed by the port size of the accessed bank, the transfer size of the transaction and the address accessed.

The functionality of the $\overline{WE}/\overline{BE}$[0:3] pins depends upon the value of the write enable/ byte select (WEBS) bit in the corresponding BR register. Setting WEBS to 1 will enable these pins as $\overline{BE}$, while resetting it to zero will enable them as $\overline{WE}$. $\overline{WE}$ is asserted only during write access, while $\overline{BE}$ is asserted for both read and write accesses. The timing of the $\overline{WE}/\overline{BE}$ pins remains the same in either case, and is determined by the TRLX, ACS and CSNT bits.

The upper $\overline{WE}/\overline{BE}$ ($\overline{WE}$[0]/$\overline{BE}$[0]) indicates that the upper eight bits of the data bus (D0–D7) contains valid data during a write/read cycle. The upper-middle write byte enable ($\overline{WE}$[1]/$\overline{BE}$[1]) indicates that the upper-middle eight bits of the data bus (D8–D15) contains valid data during a write/read cycle. The lower-middle write byte enable ($\overline{WE}$[2]/$\overline{BE}$[2]) indicates that the lower-middle eight bits of the data bus (D16–D23) contains valid data during a write/read cycle. The lower write/read enable ($\overline{WE}$[3]/ $\overline{BE}$[3]) indicates that the lower eight bits of the data bus contains valid data during a write cycle.

The write/byte enable lines affected in a transaction for 32-bit port (PS = 00), a 16-bit port (PS = 10) and a 8-bit port (PS = 01) are shown in **Table 10-4**. This table shows which write enables are asserted (indicated with an ‘X’) for different combinations of port size and transfer size

Table 10-4 Write Enable/Byte Enable Signals Function

Transfer Size	TSIZ		Address		32-bit Port Size				16-bit Port Size				8-bit Port Size			
			A30	A31	$\overline{WE}$[0]/$\overline{BE}$[0]	$\overline{WE}$[1]/$\overline{BE}$[1]	$\overline{WE}$[2]/$\overline{BE}$[2]	$\overline{WE}$[3]/$\overline{BE}$[3]	$\overline{WE}$[0]/$\overline{BE}$[0]	$\overline{WE}$[1]/$\overline{BE}$[1]	$\overline{WE}$[2]/$\overline{BE}$[2]	$\overline{WE}$[3]/$\overline{BE}$[3]	$\overline{WE}$[0]/$\overline{BE}$[0]	$\overline{WE}$[1]/$\overline{BE}$[1]	$\overline{WE}$[2]/$\overline{BE}$[2]	$\overline{WE}$[3]/$\overline{BE}$[3]
Byte	0	1	0	0	X				X				X			
	0	1	0	1		X				X			X			
	0	1	1	0			X		X				X			
	0	1	1	1				X		X			X			
Half-Word	1	0	0	0	X	X			X	X			X			
	1	0	1	0			X	X	X	X			X			
Word	0	0	0	0	X	X	X	X	X	X			X			

10.6 Dual Mapping of the Internal Flash EEPROM Array

The user can enable mapping of the internal flash EEPROM (CMF) module to an external memory region controlled by the memory controller. Only one region can be programmed to be dual-mapped. When dual mapping is enabled (DME bit is set in DMBR), an internal address matches the dual-mapped address range (as programmed in the DMBR), and the cycle type matches AT/ATM field in DMBR/DMOR registers, then the following occur:

- The internal flash memory does not respond to that address

- The memory controller takes control of the external access
- The attributes for the access are taken from one of the base and option registers of the appropriate chip select
- The chip-select region selected is determined by the "$\overline{CS}$ line select" bit field (**10.8.5 Dual Mapping Base Register (DMBR)**).

Note that dual mapping can operate only for addresses within the FLASH pre-allocated address (up to 2 Mbytes). This is achieved by programming only six bits for the base address (11:16); The upper bits are always set as follows:

bus_addr[0:10]={0000000,isb[0:2],0}

Where ISB[0:2] represents the bit field in IMMR register that determines the location of the address map of the MPC555 / MPC556.

With dual mapping, aliasing of address spaces may occur. This happens when the user maps the dual-mapped region into a region which is also mapped into one of the four regions available in the memory controller. If the user writes code or data to the dual-mapped region, care must be taken to avoid overwriting this code or data by normal accesses of the chip-select region.

There is a match if:

bus_address[0:16] == {0000000,isb[0:2],0,dmbr_reg_value[1:6]}

Care must also be taken to avoid overwriting "normal" $\overline{CS}$x data with dual-mapped code or data.

One way to avoid this situation is by disabling the chip-select region and enabling only the dual-mapped region (DME = 1, but Vx = 0, where x = selected region, 0.3). **Figure 10-17** illustrates the phenomena.

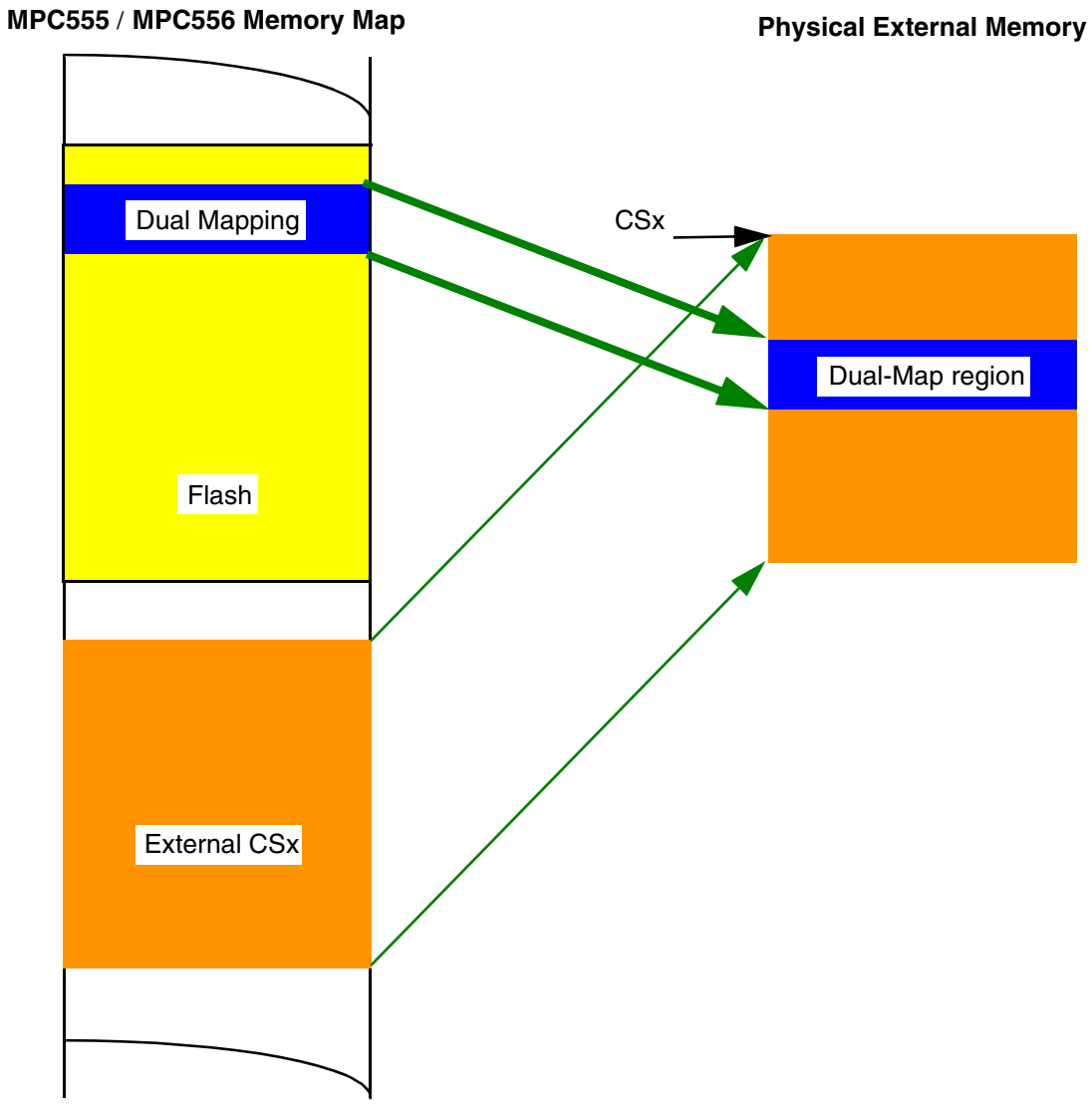


Figure 10-17 Aliasing Phenomena Illustration

The default state is to allow dual-mapping data accesses only; this means that dual mapping is possible only for data accesses on the internal bus. Also, the default state takes the lower two Mbytes of the MPC555 / MPC556 internal flash memory. Hence, caution should be taken to change the dual-mapping setup before the first data access.

NOTE

Dual mapping is *not* supported for an external master when the memory controller serves the access; In such a case, the MPC555 / MPC556 terminates the cycle by asserting $\overline{\text{TEA}}$.

10.7 Memory Controller External Master Support

The memory controller in the MPC555 / MPC556 supports accesses initiated by both internal and external bus masters to external memories. If the address of any master is mapped within the internal MPC555 / MPC556 address space, the access will be directed to the internal device, and will be ignored by the memory controller. If the address is not mapped internally, but rather mapped to one of the memory contoller regions, the memory controller will provide the appropriate chip select and strobes as programmed in the corresponding region (see **6.13.1.3 External Master Control Register (EMCR)**.

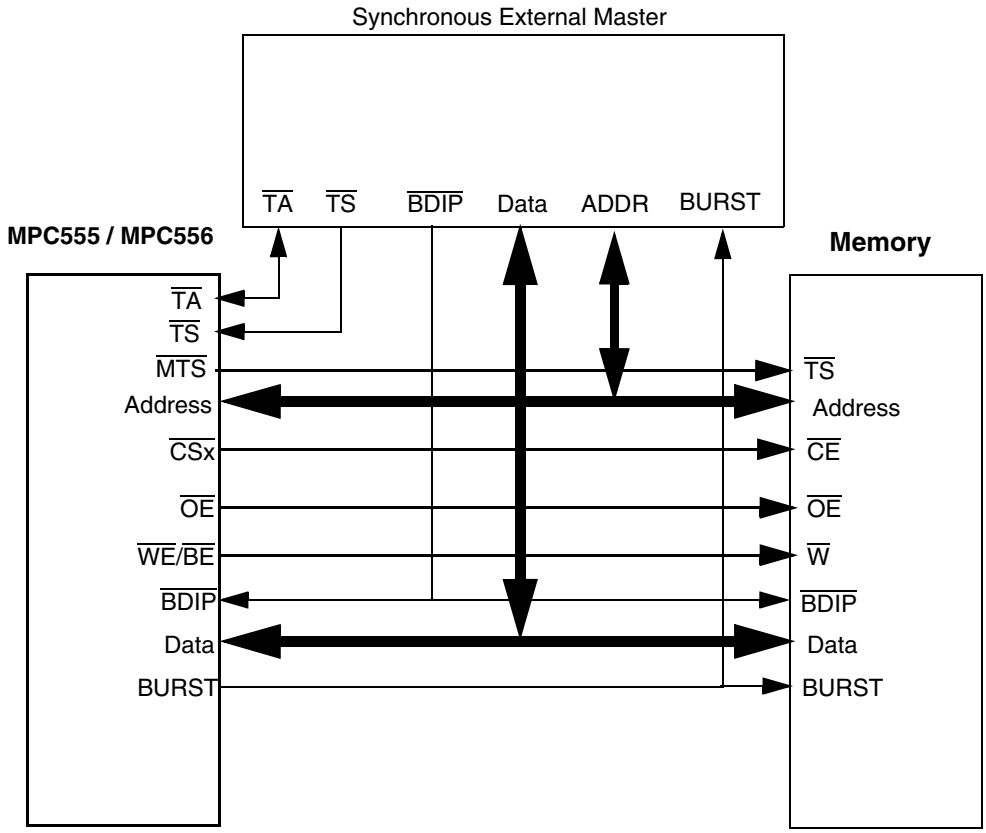
The MPC555 / MPC556 supports only synchronous external bus masters. This means that the external master works with CLKOUT and implements the MPC555 / MPC556 bus protocol to access a slave device.

A synchronous master initiates a transfer by asserting $\overline{TS}$. The ADDR[0:31] signals must be stable from the rising edge of CLKOUT during which $\overline{TS}$ is sampled, until the last $\overline{TA}$ acknowledges the transfer. Since the external master works synchronously with the MPC555 / MPC556, only setup and hold times around the rising edge of CLKOUT are important. Once the $\overline{TS}$ is detected/asserted, the memory controller compares the address with each one of its defined valid banks to find a possible match. But, since the external address space is shorter than the internal space, the actual addess that is used for comparing against the memory controller regions is in the format of: {00000000, bits 8:16 of the external address}. In the case where a match is found, the controls to the memory devices are generated and the transfer acknowledge indication ($\overline{TA}$) is supplied to the master.

Since it takes two clocks for the external address to be recognized and handled by the memory controller, the $\overline{TS}$ which is generated by the external master is ahead of the corresponding $\overline{CS}$ and strobes which are asserted by the memory controller. This 2-clock delay might cause problems in some synchronous memories. To overcome this, the memory controller generates the $\overline{MTS}$ (memory transfer start) strobe which can be used in the slave's memory instead of the external master's $\overline{TS}$ signal. As seen in **Figure 10-18**, the $\overline{MTS}$ strobe is synchronized to the assertion of $\overline{CS}$ by the memory controller so that the external memory can latch the external master's address correctly. To activate this feature, the MTSC bit must be set in the SIUMCR register. Refer to **6.13.1.1 SIU Module Configuration Register** for more information.

When the external master accesses the internal flash when it is disabled, then the access is terminated with transfer error acknowledge ($\overline{TEA}$ pin) asserted, and the memory controller does not support this access in any way.

When the memory controller serves an external master, the $\overline{BDIP}$ pin becomes an input pin. This pin is watched by the memory controller to detect when the burst is terminated.



Note that the memory controller's $\overline{\text{BDIP}}$ line is used as a burst_in_progress signal.

Figure 10-18 Synchronous External Master Configuration For GPCM–Handled Memory Devices

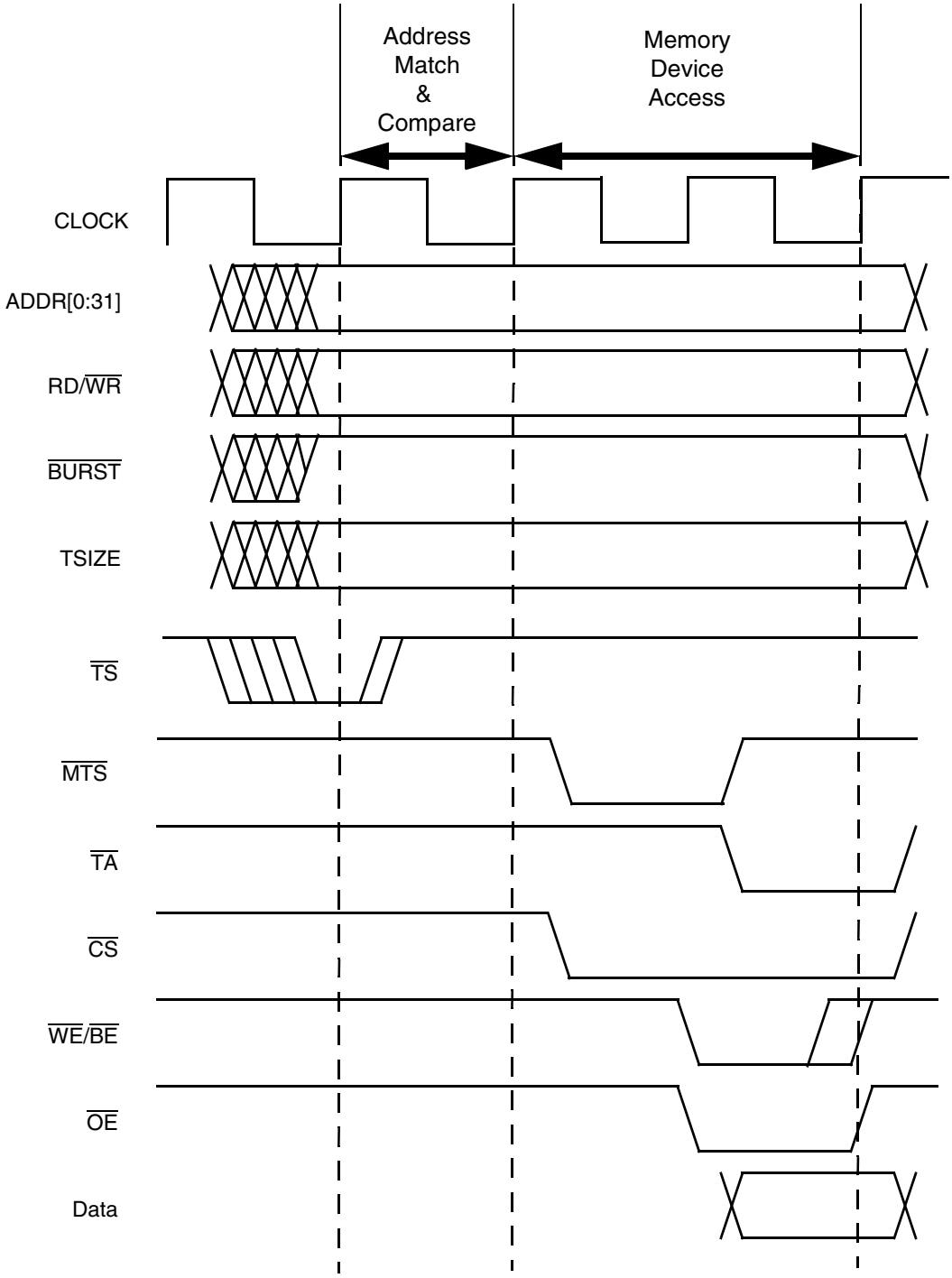


Figure 10-19 Synchronous External Master Basic Access (GPCM Controlled)

Note that since the MPC555 / MPC556 has only 24 address pins, the eight most significant internal address lines are driven as 0x0000_0000, and so compared in the memory controller's regions.

10.8 Programming Model

The following registers are used to control the memory controller.

Table 10-5 Memory Controller Address Map

Address	Register
0x2F C100	Base Register Bank 0 (BR0)
0x2F C104	Option Register Bank 0 (OR0)
0x2F C108	Base Register Bank 1 (BR1)
0x2F C10C	Option Register Bank 1 (OR1)
0x2F C110	Base Register Bank 2 (BR2)
0x2F C114	Option Register Bank 2 (OR2)
0x2F C118	Base Register Bank 3 (BR3)
0x2F C11C	Option Register Bank 3 (OR3)
0x2F C120 — 0x13F	Reserved
0x2F C140	Dual-Mapping Base Register (DMBR)
0x2F C144	Dual-Mapping Option Register (DMOR)
0x2F C148 — 0x2F C174	Reserved
0x2F C178	Memory Status Register (MSTAT)

Note:
In all subsequent registers bit tables, if two reset values are given: the upper is for $\overline{CS}$x, x = 1, 2, 3, and the lower is dedicated to $\overline{CS}$[0].

10.8.1 General Memory Controller Programming Notes

1. In the case of an external master that accesses an internal MPC555 / MPC556 module (in slave or peripheral mode), if that slave device address also matches one of the memory controller's regions, the memory controller will not issue any $\overline{CS}$ for this access, nor will it terminate the cycle. Thus, this practice should be avoided. Be aware also that any internal slave access prevents memory controller operation.
2. If the memory controller serves an external master, then it can support accesses to 32-bit port devices only. This is because the MPC555 / MPC556 external bus interface cannot initiate extra cycles to complete an access to a smaller port-size device as it does not own the external bus.
3. When the SETA bit in the base register is set, then the timing programming for the various strobes ($\overline{CS}$, $\overline{OE}$ and $\overline{WE}$/$\overline{BE}$) may become meaningless.

10.8.2 Memory Controller Status Registers (MSTAT)

MSTAT — Memory Controller Status Register **0x2F C178**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED								WPER 0	WPER 1	WPER 2	WPER 3	RESERVED			
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-6 MSTAT Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8:11	WPER0 – WPER3	Write protection error for bank x. This bit is asserted when a write-protect error occurs for the associated memory bank. A bus monitor (responding to $\overline{TEA}$ assertion) will, if enabled, prompt the user to read this register if $\overline{TA}$ is not asserted during a write cycle. WPERx is cleared by writing one to the bit or by performing a system reset. Writing a zero has no effect on WPER.
12:15	—	Reserved

10.8.3 Memory Controller Base Registers (BR0 – BR3)

BR0 – BR3 — Memory Controller Base Registers 0 – 3 **0x2F C100, C108, C110, C118**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BA															
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BA	AT			PS		RE-SERVED	WP	RESERVED		WEBS	TBDIP	LBDIP	SETA	BI	V
HRESET															
U	U	U	U	ID[4:5]*		0	0	0	0	0	0		0	1	$\overline{ID3}$**

* Reset value is determined by the value on the internal data bus during reset.

** The BR0 Reset value is determined by the value on the internal data bus during reset (reset-configuration word). The reset value of the V bit of BR1-3 = 0.

Table 10-7 BR0 – BR3 Bit Descriptions

Bit(s)	Name	Description
0:16	BA	Base address. These bits are compared to the corresponding unmasked address signals among ADDR[0:16] to determine if a memory bank controlled by the memory controller is being accessed by an internal bus master. (The address types are also compared.) These bits are used in conjunction with the AM[0:16] bits in the OR.
17:19	AT	Address type. This field can be used to require accesses of the memory bank to be limited to a certain address space type. These bits are used in conjunction with the ATM bits in the OR. Note that the address type field uses only AT[0:2] and does not need AT[3] to define the memory type space. For a full definition of address types, refer to **9.5.7.6 Address Types**.
20:21	PS	Port size 00 = 32-bit port 01 = 8-bit port 10 = 16-bit port 11 = Reserved
22	—	Reserved
23	WP	Write protect. An attempt to write to the range of addresses specified in a base address register that has this bit set can cause the $\overline{TEA}$ signal to be asserted by the bus-monitor logic (if enabled), causing termination of this cycle. 0 = Both read and write accesses are allowed 1 = Only read accesses are allowed. The $\overline{CS}$x signal and $\overline{TA}$ are not asserted by the memory controller on write cycles to this memory bank. WPER is set in the MSTAT register if a write to this memory bank is attempted
24:25	—	Reserved
26	WEBS	Write-enable/byte-select. This bit controls the functionality of the $\overline{WE}/\overline{BE}$ pads. 0 = The $\overline{WE}/\overline{BE}$ pads operate as $\overline{WE}$ 1 = The $\overline{WE}/\overline{BE}$ pads operate as $\overline{BE}$
27	TBDIP	Toggle-burst data in progress. TBDIP determines how long the $\overline{BDIP}$ strobe will be asserted for each data beat in the burst cycles.
28	LBDIP	Late-burst-data-in-progress (LBDIP). This bit determines the timing of the first assertion of the $\overline{BDIP}$ pin in burst cycles. Note: it is not allowed to set both LBDIP and TBDIP bits in a region's base registers; the behavior of the design in such cases is unpredictable. 0 = Normal timing for $\overline{BDIP}$ assertion (asserts one clock after negation of $\overline{TS}$ 1 = Late timing for $\overline{BDIP}$ assertion (asserts after the programmed number of wait states
29	SETA	External transfer acknowledge 0 = $\overline{TA}$ generated internally by memory controller 1 = $\overline{TA}$ generated by external logic. Note that programming the timing of $\overline{CS}/\overline{WE}/\overline{OE}$ strobes may have no meaning when this bit is set
30	BI	Burst inhibit 0 = Memory controller drives $\overline{BI}$ negated (high). The bank supports burst accesses. 1 = Memory controller drives $\overline{BI}$ asserted (low). The bank does not support burst accesses. Note: Following a system reset, the $\overline{BI}$ bit is set in OR0.
31	V	Valid bit. When set, this bit indicates that the contents of the base-register and option-register pair are valid. The $\overline{CS}$ signal does not assert until the V-bit is set. Note that an access to a region that has no V-bit set may cause a bus monitor timeout. Note also that following a system reset, the V-bit in BR0 reflects the value of $\overline{ID}$3 in the reset configuration word.

10.8.4 Memory Controller Option Registers (OR0 – OR3)

OR0 – OR3 — Memory Controller Option Registers 0 – 3 **0x2F C104, C10C, C114, C11C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AM*															
HRESET: (OR[1:3])															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HRESET (OR0)															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
AM*	ATM			CSNT	ACS		EHTR	SCY				BSCY			TRLX
HRESET: (OR[1:3]):															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HRESET (OR0)															
0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0

*It is recommended that this field would hold values that are the power of 2 minus 1 (e.g., - 2^3 - 1 = 7 [0b111]).

Table 10-8 OR0 – OR3 Bit Descriptions

Bit(s)	Name	Description
0:16	AM	Address mask. This field allows masking of any corresponding bits in the associated base register. Masking the address bits independently allows external devices of different size address ranges to be used. Any clear bit masks the corresponding address bit. Any set bit causes the corresponding address bit to be used in comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at anytime. Following a system reset, the AM bits are reset in OR0.
17:19	ATM	Address type mask. This field masks selected address type bits, allowing more than one address space type to be assigned to a chip-select. Any set bit causes the corresponding address type code bits to be used as part of the address comparison. Any cleared bit masks the corresponding address type code bit. Clear the ATM bits to ignore address type codes as part of the address comparison. Note that the address type field uses only AT[0:2] and does not need AT[3] to define the memory type space. Following a system reset, the ATM bits are reset in OR0.
20	CSNT	Chip-select negation time. Following a system reset, the CSNT bit is reset in OR0. 0 = $\overline{CS}/\overline{WE}$ are negated normally. 1 = $\overline{CS}/\overline{WE}$ are negated a quarter of a clock earlier than normal Following a system reset, the CSNT bit is reset in OR0.
21:22	ACS	Address to chip-select setup. Following a system reset, the ACS bits are reset in OR0. 00 = $\overline{CS}$ is asserted at the same time that the address lines are valid. 01 = Reserved 10 = $\overline{CS}$ is asserted a quarter of a clock after the address lines are valid. 11 = $\overline{CS}$ is asserted half a clock after the address lines are valid Following a system reset, the ACS bits are reset in OR0.

Table 10-8 OR0 – OR3 Bit Descriptions (Continued)

Bit(s)	Name	Description
23	EHTR	Extended hold time on read accesses. This bit, when asserted, inserts an idle clock cycle after a read access from the current bank and any MPC555 / MPC556 write accesses or read accesses to a different bank. 0 = Memory controller generates normal timing 1 = Memory controller generates extended hold timing
24:27	SCY	Cycle length in clocks. This four-bit value represents the number of wait states inserted in the single cycle, or in the first beat of a burst, when the GPCM handles the external memory access. Values range from from 0 (0b0000) to 15 (0b1111). This is the main parameter for determining the length of the cycle. The total cycle length may vary depending on the settings of other timing attributes. The total memory access length is (2 + SCY) x Clocks. If the user has selected an external $\overline{TA}$ response for this memory bank (by setting the SETA bit), then the SCY field is not used. **NOTE:** Following a system reset, the SCY bits are set to 0b1111 in OR0.
28:30	BSCY	Burst beats length in clocks. This field determines the number of wait states inserted in all burst beats except the first, when the GPCM starts handling the external memory access and thus using SCY[0:3] as the main parameter for determining the length of that cycle. The total cycle length may vary depending on the settings of other timing attributes. The total memory access length for the beat is is (1 + BSCY) x Clocks. If the user has selected an external $\overline{TA}$ response for this memory bank (by setting the SETA bit) then BSCY[0:3] are not used. 000 = 0-clock-cycle (1 clock per data beat) 001 = 1-clock-cycle wait states (2 clocks per data beat) 010 = 2-clock-cycle wait states (3 clocks per data beat) 011 = 3-clock-cycle wait states (4 clocks per data beat) 1xx = Reserved
31	TRLX	Timing relaxed. This bit, when set, modifies the timing of the signals that control the memory devices during a memory access to this memory region. Relaxed timing multiplies by two the number of wait states determined by the SCY and BSCY fields. Refer to **10.3.5 Summary of GPCM Timing Options** for a full list of the effects of this bit on pins timing. 0 = Normal timing is generated by the GPCM. 1 = Relaxed timing is generated by the GPCM Following a system reset, the TRLX bit is set in OR0.

10.8.5 Dual Mapping Base Register (DMBR)

DMBR — Dual Mapping Base Register **0x2F C140**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	BA						RESERVED			AT			RESERVED		
HARD RESET:															
0	U	U	U	U	U	U	0	0	0	0	0	1	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED												DMCS			DME
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ID31*

*The reset value is a reset configuration word value extracted from the indicated internal data bus lines.

Table 10-9 DMBR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:6	BA	Base address. The base address field is compared (along with the address type field) to the address of the address bus to determine whether an address should be dual-mapped by one of the memory banks controlled by the memory controller. These bits are used in conjunction with the AM[11:16] bits in the OR. Bit 10: is cleared at reset. That way, the default range for the dual mapping is 2 Mbytes. Note that by setting this bit, the range becomes 4 Mbyte, which includes memory space beyond the flash EEPROM memory.
7:9	—	Reserved
10:12	AT	Address type. This field can be used to specify that accesses involving the memory bank are limited to a certain address space type. These bits are used in conjunction with the ATM bits in the OR. The default value at reset is to map data only. For a full definition of address types, refer to **9.5.7.6 Address Types**.
13:27	—	Reserved
28:30	DMCS	Dual-mapping chip select. This field determines which chip-select pin is assigned for dual mapping. 000 = $\overline{CS}$[0] 001 = $\overline{CS}$[1] 010 = $\overline{CS}$[2] 011 = $\overline{CS}$[3] 1xx = Reserved
31	DME	Dual mapping enabled. This bit indicates that the contents of the dual-mapping registers and associated base and option registers are valid and enables the dual-mapping operation. The default value at reset comes from the internal data bus that reflects the reset configuration word. See **10.6 Dual Mapping of the Internal Flash EEPROM Array** for more information. 0 = Dual mapping is not active 1 = Dual mapping is active

10.8.6 Dual-Mapping Option Register

DMOR — Dual-Mapping Option Register **0x2F C144**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	AM*						RESERVED			ATM			RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*It is recommended that this field would hold values that are the power of 2 minus 1 (e.g., - 2^3 - 1 = 7 [0b111]).

Table 10-10 DMOR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:6	AM	Address mask. The address mask field of each option register provides for masking any of the corresponding bits in the associated base register. By masking the address bits independently, external devices of different size address ranges can be used. Any clear bit masks the corresponding address bit. Any set causes the corresponding address bit to be used in the comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at any time.
7:9	—	Reserved
10:12	ATM	Address type mask. This field can be used to mask certain address type bits, allowing more than one address space type to be assigned to a chip select. Any set bit causes the corresponding address type code bits to be used as part of the address comparison. Any cleared bit masks the corresponding address type code bit. To instruct the memory controller to ignore address type codes as part of the address comparison, clear the ATM bits. Note: Following a system reset, the ATM bits are cleared in DMOR, except the ATM2 bit. This means that only data accesses are dual mapped. Refer to the address types definition in **Table 9-8**.
13:31	—	Reserved

SECTION 11
L-BUS TO U-BUS INTERFACE (L2U)

The L-bus to U-bus interface unit (L2U) provides an interface between the load/store bus (L-bus) and the unified bus (U-bus). The L2U module includes the data memory protection unit (DMPU), which provides protection for data memory accesses.

The L2U is bi-directional. It allows load/store accesses not intended for the L-bus data RAM to go to the U-bus. It also allows code execution from the L-bus data RAM and read/write accesses from the U-bus to the L-bus.

The L2U directs bus traffic between the L-bus and the U-bus. When transactions start concurrently on both buses, the L2U interface arbitrates to select which transaction is handled. The top priority is assigned to U-bus to L-bus accesses; lower priority is assigned to the load/store accesses by the RCPU.

11.1 General Features

- Non-pipelined master and slave on U-bus
  - — Does not start two back-to-back accesses on the U-bus
  - — Supports the U-bus pipelining by starting a cycle on the U-bus when U-bus pipe depth is zero or one
  - — Does not accept back-to-back accesses from the U-bus master
- Non-pipelined master and slave on the L-bus
- Generates module selects for L-bus memory-mapped resources within a programmable, contiguous block of storage
- Programmable data memory protection unit (DMPU)
- L-bus and U-bus snoop logic for PowerPC reservation protocol
- L2U does not support dual mapping of L-bus or IMB3 space
- Show cycles for RCPU accesses to the SRAM (none, all, writes)
  - — Protection for SRAM accesses from the U-bus side (all accesses to the SRAM from the U-bus side are blocked once the SRAM protection bit is set)

11.2 DMPU Features

- Supports four memory regions whose base address and size can be programmed
  - — Available sizes are 4 Kbytes, 8 Kbytes, 16 Kbytes, 32 Kbytes, 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, 1 Mbyte, 2 Mbytes, 4 Mbytes, 8 Mbytes, and 16 Mbytes
  - — Region must start on the specified region size boundary
  - — Overlap between regions is allowed
- Each of the four regions supports the following attributes:
  - Access protection: user or supervisor
  - Guarded attribute: speculative or non-speculative
  - Enable/disable option
  - Read only option

- Supports a default global entry for memory space not covered by other regions:
  — Default access protection
  — Default guarded attribute
- Interrupt generated upon:
  — Access violation
  — Load from guarded region
  — Write to read-only region
- The PowerPC MSR[DR] bit (data relocate) controls DMPU protection on/off operation
- Programming is done using PowerPC's **mtspr/mfspr** instructions to/from implementation specific special purpose registers.
- No protection for accesses to the SRAM module on the L-bus (SRAM has its own protection options)

11.3 L2U Block Diagram

Figure 11-1 shows a block diagram of the L-bus to U-bus interface.

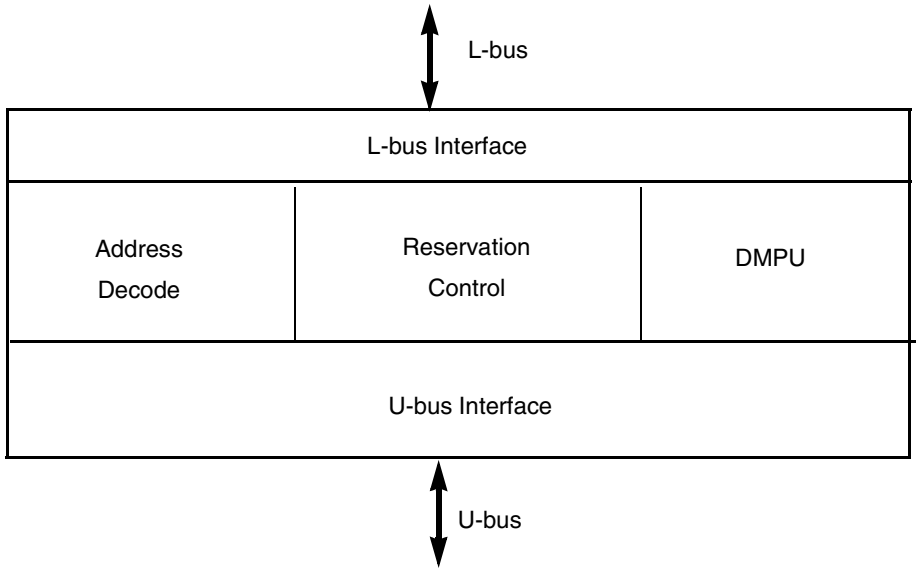


Figure 11-1 L2U Bus Interface Block Diagram

11.4 Modes Of Operation

The L2U Module can operate in the following modes:

- Normal Mode
- Reset Operation
- Factory Test Mode
- Peripheral Mode

11.4.1 Normal Mode

In normal mode (master or slave) the L2U module acts as a bi-directional protocol translator. In master mode the CPU is fully operational, and there is no external master access to the U-bus. Slave mode enables an external master to access any internal bus slave while the CPU is fully operational. The L2U transfers load/store accesses from the RCPU to the U-bus and the read/write accesses by the U-bus master to the L-bus.

In addition to the bus protocol translation, the L2U supports other functions such as show cycles, data memory protection and PowerPC reservation protocol.

When a load from the U-bus resource or store to the U-bus resource is issued by the RCPU, it is compared against the DMPU region access (address and attribute) comparators. If none of the access attributes are violated, the access is directed to the U-bus by the L2U module. If the DMPU detects an access violation, it informs the error status to the master initiating the cycle.

When show cycles are enabled, accesses to all of the L-bus resources by the RCPU are made visible on the U-bus side by the L2U.

The L2U is responsible for handling the effects of reservations on the L-bus and the U-bus. For the L-bus and the U-bus, the L2U detects reservation losses and updates the RCPU core with the reservation status.

11.4.2 Reset Operation

Upon soft reset assertion, the L2U goes to an idle state and all pending accesses are ignored. The L2U module control registers are not initialized on the assertion of a soft reset, keeping the system configuration unchanged.

Upon assertion of hard reset, the L2U control registers are initialized to their reset states.

While reset (hard or soft) is asserted on the U-bus, the L2U asserts the corresponding L-bus reset signals. The L2U also drives the reset configuration word from the U-bus to the L-bus upon assertion of hard reset.

11.4.3 Factory Test Mode

Factory test mode is a special mode of operation that allows access to the internal modules for testing. This mode is not intended for general use and is not supported for normal applications.

11.4.4 Peripheral Mode

In the peripheral mode of operation the RCPU is shut down and an alternative master on the external bus can perform accesses to any internal bus (U-bus and L-bus) slave.

The external master can also access the internal PowerPC special registers that are located in L2U. In order to access one of these PowerPC registers the EMCR[CONT] bit in the USIU must be cleared.

11.5 Data Memory Protection

The data memory protection unit (DMPU) in the L2U module provides access protection for the memory regions on the U-bus side from load/store accesses by the RCPU. (Only U-bus space is protected.) The DMPU does not protect PowerPC register accesses initiated by the RCPU on the L-bus. The user can assign up to four regions of access protection attributes and can assign global attributes to any space not included in the active regions. When it detects an access violation, the L2U generates an exception request to the CPU.

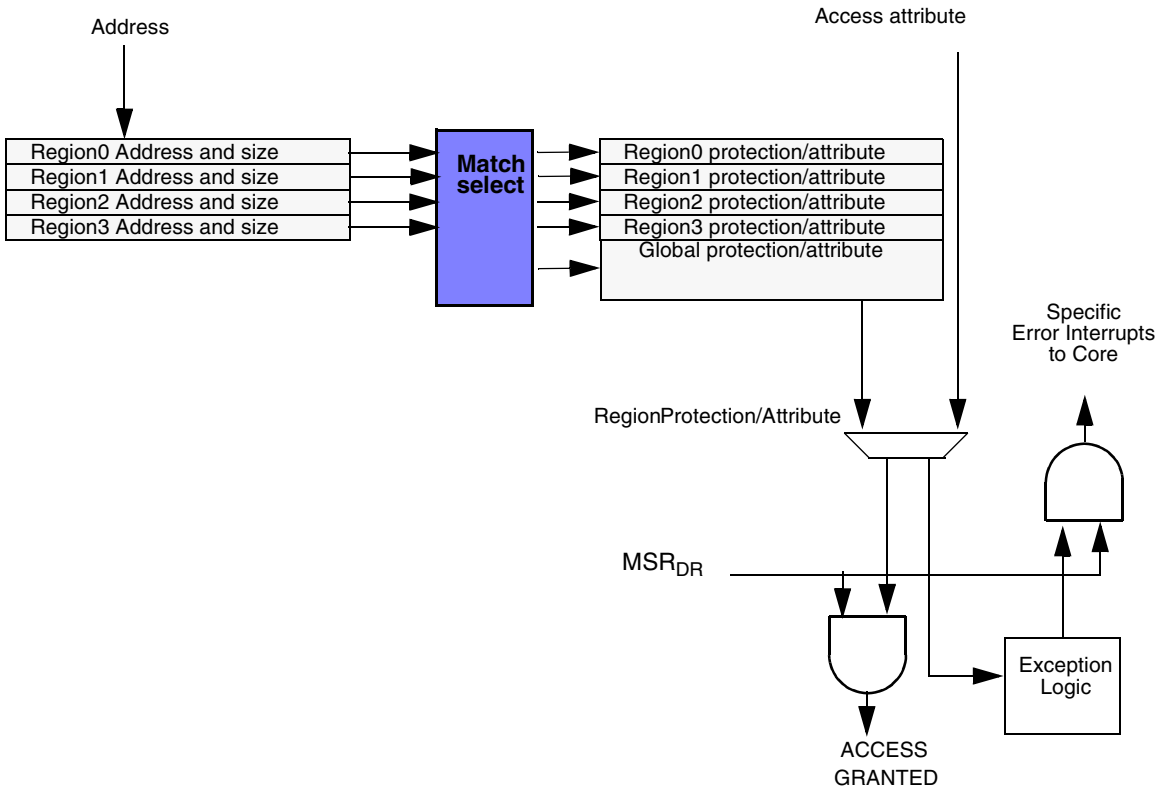


Figure 11-2 DMP Basic Functional Diagram

11.5.1 Functional Description

Data memory protection is assigned on a regional basis. Default manipulation of the DMPU is done on a global region. The DMPU has control registers which contain the following information: region protection on/off, region base address, region size, and the region's access permissions. Each region's protection attributes can be turned on/off by configuring the enable attribute bit (ENRx) located in the global region attribute register.

During each load/store access from the RCPU core to the U-bus, the address is compared to the value in the region base address register of each enabled region. Any access that matches the specific region within its appropriate size, as defined by the region size field (RS) of the region attribute register, sets a match indication.

When more than one match indication occurs, the effective region is the region with the highest priority. Priority is determined by region number; highest priority corresponds to the lowest region number.

When no match occurs, the effective region is the global region. The global region has the lowest priority.

The region attribute register also contains the region's protection fields. The protection field (PP) of the effective region is compared to the access attributes. If the attributes match, the access is permitted. When the access is permitted, a U-bus access may be generated according to the specific attribute of the effective region.

When the access by the RCPU is not permitted, the L2U module asserts a data memory storage exception to the RCPU.

For speculative load/store accesses from the RCPU to a region marked as guarded (G bit of region attribute register is set), the L2U asks the RCPU to retry the L-bus cycle until either the access is not speculative, or it is canceled by the RCPU.

In the case of attempted accesses to a guarded region together with any other protection violation (no access), the L2U retries the access. The L2U handles this event as a data storage violation only when the access becomes non-speculative.

Note that access protection is active only when the PowerPC's MSR[DR] = 1. When MSR[DR] = 0, DMPU exceptions are disabled, all accesses are considered to be to a guarded memory area, and no speculative accesses are allowed. In this case, if the L-bus master [RCPU] initiates a non-SRAM cycle (access through the L2U) that is marked speculative, the L2U asks the RCPU to retry the L-bus cycle until either the access is not speculative, or it is canceled by the RCPU core.

Note that the programmer must not overlap the SRAM memory space with any enabled region. Overlapping an enabled region with SRAM memory space disables the L2U data memory protection for that region.

If an enabled region overlaps with the L-bus space, the DMPU ignores all accesses to addresses within the L-bus space. If an enabled region overlaps with PowerPC register addresses, the DMPU ignores any access marked as a PowerPC access.

11.5.2 Associated Registers

The following registers are used to control the DMPU of the L2U module. All the registers are special purpose registers which are accessed via the PowerPC **mtspr**/**mfspr** instructions. The registers are also accessed by an external master when EMCR[CONT] = 0. See **11.8 L2U Programming Model** for register diagrams and bit descriptions.

Table 11-1 DMPU Registers

Name	Description
L2U_RBA0	Region Base Address Register 0
L2U_RBA1	Region Base Address Register 1
L2U_RBA2	Region Base Address Register 2
L2U_RBA3	Region Base Address Register 3
L2U_RA0	Region Attribute Register 0
L2U_RA1	Region Attribute Register 1
L2U_RA2	Region Attribute Register 2
L2U_RA3	Region Attribute Register 3
L2U_GRA	Global Region Attribute

CAUTION

The appropriate DMPU registers must be programmed before the MSR[DR] bit is set. Otherwise, DMPU operation is not guaranteed.

Program the region base address in the L2U_RBAx registers to the lower boundary of the region specified by the corresponding L2U_RAx[RS] field. If the region base address does not correspond to the boundary of the block size programmed in the L2U_RAx, the DMPU snaps the region base to the lower boundary of that block. For example, if the block size is programmed to 16 Kbytes for region zero (i.e. L2U_RA0[RS] = 0 x 3) and the region base address is programmed to 0x1FFF(i.e., L2U_RBA0[RBA] = 0 x 1), then the effective base address of region zero is 0 x 0. See **Figure 11-3**.

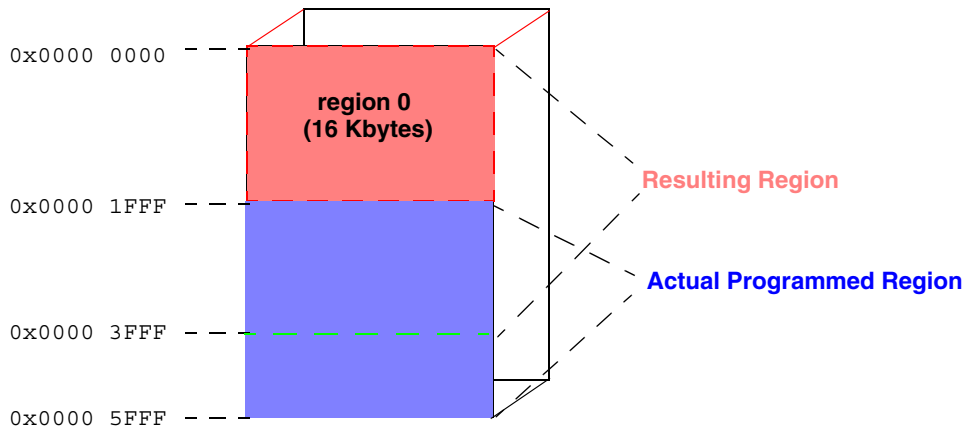


Figure 11-3 Region Base Address Example

It is the user's responsibility to program only legal region sizes. The L2U does not check whether the value is legal. If the user programs an illegal region size, the region calculation may not be successful.

11.5.3 L-bus Memory Access Violations

All L-bus slaves have their own access protection logic. For consistency, all storage access violations have the same termination result. Thus access violations for load/store accesses started by the RCPU always have the same termination from all slaves: assertion of the data storage exception. All other L-bus masters cause machine check exceptions.

11.6 Reservation Support

The RCPU storage reservation protocol supports a multi-level bus structure. For each local bus, storage reservation is handled by the local reservation logic. The protocol tries to optimize reservation cancellation such that a PowerPC processor (RCPU) is notified of storage reservation loss on a remote bus (U-bus, IMB or external bus) only when it has issued a **stwcx** cycle to that address. That is, the reservation loss indication comes as part of the **stwcx** cycle.

11.6.1 The Reservation Protocol

The reservation protocol operates under the following assumptions:

- Each processor has at most 1 reservation flag
- A **lwarx** instruction sets the reservation flag
- Another **lwarx** instruction by same processor clears the reservation flag related to a previous **lwarx** instruction and sets again the reservation flag
- A **stwcx** instruction by same processor clears the reservation flag
- A store instruction by same processor does not clear the reservation flag
- Some other processor (or other mechanism) store to an address with an existing reservation clears the reservation flag
- In case the storage reservation is lost, it is guaranteed that **stwcx** will not modify the storage

11.6.2 L2U Reservation Support

The L2U is responsible for handling the effects of reservations on the L-bus and the U-bus. For the L-bus and the U-bus, the L2U detects reservation losses.

The reservation logic in the L2U performs the following functions:

- Snoops accesses to all L-bus and U-bus slaves
- Holds one reservation (address) for the core
- Sets the reservation flag when the CPU issues a load-with-reservation request

The unit for reservation is one word. A byte or half-word store request by another master will clear the reservation flag.

A load-with-reservation request by the CPU updates the reservation address related to a previous load-with-reservation request and sets the reservation flag for the new location. A store-with-reservation request by the CPU clears the reservation flag. A store request by the CPU does not clear the flag. A store request by some other master to the reservation address clears the reservation flag.

If the storage reservation is lost, it is guaranteed that a store-with-reservation request by the CPU will not modify the storage.

The L2U does not start a store-with-reservation cycle on the U-bus if the reserved location on the U-bus has been touched by another master. The L2U drives the reservation status back to the core.

When the reserved location in the SRAM on the L-bus is touched by an alternate master, on the following clock, the L2U indicates to the CPU that the reservation has been touched. On assertion of the cancel-reservation signal, the RCPU clears the internal reservation bit. If an **stwcx** cycle has been issued at the same time, the RCPU aborts the cycle.

Storage reservation is set regardless of the termination status (address or data phase) of the **lwarx** access. Storage reservation is cleared regardless of the data phase termination status of the **stwcx** access if the address phase is terminated normally.

Storage reservation will be cleared regardless of the data phase termination status of the write requests by another master to the reserved address if the address phase of the write access is terminated normally on the destination (U-bus/L-bus) bus.

If the programmable memory map of the part is modified between a **lwarx** and a **stwcx** instruction, the reservation is not guaranteed.

11.6.3 Reserved Location (Bus) and Possible Actions

Once the CPU core reserves a memory location, the L2U module is responsible for snooping L-bus and U-bus for possible intrusion of the reserved location. Under certain circumstances, the L2U depends on the USIU or the UIMB to provide status of reservation on external bus and the IMB3 respectively.

Table 11-2 lists all reservation protocol cases supported by the L2U snooping logic.

Table 11-2 Reservation Snoop Support

Reserved Location On	Intruding Alternate Master	Action Taken on stwcx cycle
L-bus	L-Master	Request to cancel the reservation.[1]
	U-Master	Request to cancel the reservation.
U-bus	L-Master	Block **stwcx**[2]
	U-Master	Block **stwcx**
External Bus	L-Master	Block **stwcx**
	U-Master	Block **stwcx**
	Ext-Master	Transfer Status[3]
IMB3	L-Master	Block **stwcx**
	U-Master	Block **stwcx**
	IMB3-Master	Transfer Status

NOTES:

1. If the RCPU tries to modify (**stwcx**) that location, the L2U does not have enough time to stop the write access from completing. In this case, the L2U will drive cancel-reservation signal back to the core as soon as it comes to know that the alternate master on the U-bus has touched the reserved location.
2. If the RCPU tries to modify (**stwcx**) that location, the L2U does not start the cycle on the U-bus and it communicates to the core that the current write has been aborted by the slave with no side effects.
3. If the RCPU tries to modify (**stwcx**) that location, the L2U runs a write-cycle-with-reservation request on the U-bus. The L2U samples the status of the reservation along with the U-bus cycle termination signals and it communicates to the core if the current write has been aborted by the slave with no side effects.

11.7 L-Bus Show Cycle Support

The L2U module provides support for L-bus show cycles. L-bus show cycles are external visibility cycles that reflect activity on the L-bus that would otherwise not be visible to the external bus. L-bus show cycles are software controlled.

11.7.1 Programming Show Cycles

L-bus show cycles are disabled during reset and must be configured by writing the appropriate bits in the L2U_MCR control register. L-bus show cycles are programmed by setting the LSHOW[0:1] bits in the L2U_MCR. The **Table 11-3** shows the configurations of the LSHOW[0:1] bits.

Table 11-3 L2U_MCR LSHOW Modes

LSHOW	Action
00	Disable L-bus show cycles
01	Show address and data of all L-bus space write cycles
10	Reserved (Disable L-bus show cycles)
11	Show address and data of all L-bus space read and write cycles

11.7.2 Performance Impact

When show cycles are enabled in the L2U module, there is a performance penalty on the L-bus. This occurs because the L2U module does not support more than one access being processed at any time. To ensure that only one access at a time can be

processed, and not lose an L-bus access that would have been show cycled, the L2U module will arbitrate for the L-bus whenever it is processing any access. This L-bus arbitration will prevent any other L-bus master from starting a cycle that might turn out to be a qualifiable L-bus show cycle.

For L-bus show cycles, the minimum performance impact on the L-bus will be three clocks. This minimum impact assumes that the L-bus slave access is a 1-clock access, and the L2U module acquires immediate bus grant on the U-bus. The L2U has to wait two clocks before completing the show cycle on the U-Bus, thus using up five clocks for the complete process.

A retried access on the L-bus (no address acknowledge) that qualifies to be show cycled, will be accepted when it is actually acknowledged. This will cause a 1-clock delay before an L-bus master can retry the access on the L-bus, because the L2U module will release L-bus one clock later.

L2U asserts the internal bus request signal on the U-bus for a minimum of two clocks when starting a show cycle on the U-bus.

11.7.3 Show Cycle Protocol

The L2U module behaves as both a master and a slave on the U-bus during show cycles. The L2U starts the U-bus transfer as a a bus master and then completes the address phase and data phase of the cycle as a slave. The L2U follows U-bus protocol of in-order termination of the data phase.

The USIU can control the start of show cycles on the U-bus by asserting the no-show cycle indicator. This will cause the L2U module to release the U-bus for at least one clock before retrying the show cycle.

11.7.4 L-Bus Write Show Cycle Flow

The L2U performs the following sequence of actions for an L-bus-write show cycle.

1. Arbitrates for the L-bus to prevent any other L-bus cycles from starting
2. Latches the address and the data of the L-bus access, along with all address attributes
3. Waits for the termination of the L-bus access and latches the termination status (data error)
4. Arbitrate for the U-bus, and when granted, starts the U-bus access, asserting show cycle request on the U-bus, along with address, attributes and the write data. The L2U module provides address recognize and acknowledgment for the address phase. If the no-show cycle indicator from the U-bus is asserted, the L2U does not start the show cycle. The L2U module releases the U-bus until the no-show cycle indicator is negated and then arbitrates for the U-bus again.
5. When the L2U module has U-bus data bus grant, it drives the data phase termination handshakes on the U-bus.
6. Releases the L-bus

11.7.5 L-Bus Read Show Cycle Flow

The L2U performs the following sequence of actions for an L-bus read show cycle.

1. Arbitrates for the L-bus to prevent any other L-bus cycle from starting
2. Latches the address of the L-bus access, along with all address attributes
3. Waits for the data phase termination on the L-bus and latch the read data, and the termination status from the L-bus
4. Arbitrate for the U-bus, and when granted, starts the U-bus access, asserting the show cycle request on the U-bus, along with address attributes. The L2U module provides address recognize/acknowledgment for the address phase. If the no-show cycle indicator from the U-bus is asserted, the L2U does not start the show cycle. The L2U module releases the U-bus until the no-show cycle indicator is negated and then arbitrates for the U-bus again.
5. When the L2U module has U-bus data bus grant, it drives the read data and the data phase termination handshakes on the U-bus
6. Release the L-bus.

11.7.6 Show Cycle Support Guidelines

The following are the guidelines for L2U show cycle support:

- The L2U module provides address and data for all qualifying L-bus cycles when the appropriate mode bits are set in the L2U_MCR.
- The L2U-module-only show cycles L-bus activity that is not targeted for the U-bus or the L2U module internal registers, irrespective of the termination status of such activity.
- The L2U module does not show cycle any access to a PowerPC special purpose register.
- The L2U does not start a show cycle for an L-bus access that is retried. This decision to not start the show cycle causes a clock delay before the cycle can be retried, since the L2U module will have arbitrated away the L-bus immediately on detecting the show cycle, before the retry information is available.
- The L2U module does not show cycle any L-bus activity that is aborted.
- The L2U module backs off the U-bus if the USIU inhibits show cycle activity on the U-bus.
- The L2U does not show cycle any L-bus addresses that fall in the L-bus SRAM address space if the SRAM Protection [SP] bit is set in the L2U_MCR.

Table 11-4 summarizes the L2U show cycle support.

Table 11-4 L2U Show Cycle Support Chart

Case	Destination	LB AACK	LB ABORT	Comments
1	L-bus Slave[1]	No	X	Not show cycled [Cycle will be retried one clock later][4]
2	L2U [2]	X	X	Not show cycled
3	U-bus/E-bus [3]	X	X	Not show cycled
4	L-bus slave	Yes	No	Show cycled
5	L-bus slave	Yes	Yes	Not show cycled [L-bus will be released next clock]

1. L-bus slave includes all address in the L-bus address space.
2. L2U indicates L2U registers.
3. U-bus/E-bus refers to all destinations through the L2U interface.
4. There will be a 1-clock turnaround because the L-bus retry information is not available in time to negate the L-bus arbitration.

Note: X indicates don't care conditions.

11.8 L2U Programming Model

The L2U control registers control the L2U bus interface and the DMPU. They are accessible via the MPC555 / MPC556 **mtspr** and **mfspr** instructions. They are also accessible by an external master when EMCR[CONT] bit is cleared. L2U control registers are accessible from both the L-bus side and the U-bus side in one clock cycle. As with all SPRs, L2U registers are accessible in supervisor mode only.

Any unimplemented bits in L2U registers return 0's on a read, and the writes to those register bits are ignored.

The **Table 11-5** shows L2U registers along with their SPR numbers and hexadecimal addresses which are used to access L2U registers during a peripheral mode access.

Table 11-5 L2U (PPC) Register Decode

Name	SPR #	SPR5:9	SPR0:4	Address for External Master Access	Access	Description
L2U_MCR	568	10001	11000	0x0000_3110	SUPR	L2U Module Configuration Register
L2U_RBA0	792	11000	11000	0x0000_3180	SUPR	Region Base Address Register 0
L2U_RBA1	793	11000	11001	0x0000_3380	SUPR	Region Base Address Register 1
L2U_RBA2	794	11000	11010	0x0000_3580	SUPR	Region Base Address Register 2
L2U_RBA3	795	11000	11011	0x0000_3780	SUPR	Region Base Address Register 3
L2U_RA0	824	11001	11000	0x0000_3190	SUPR	Region Attribute Register 0
L2U_RA1	825	11001	11001	0x0000_3390	SUPR	Region Attribute Register 1
L2U_RA2	826	11001	11010	0x0000_3590	SUPR	Region Attribute Register 2
L2U_RA3	827	11001	11011	0x0000_3790	SUPR	Region Attribute Register 3
L2U_GRA	536	10000	11000	0x0000_3100	SUPR	Global Region Attribute

For these registers a bus cycle will be performed on the L-bus and the U-bus with the address as shown in**Table 11-6**.

Table 11-6 Hex Address For SPR Cycles

A0:17	A18:22	A23:27	A28:31
0	spr0:4	spr5:9	0

11.8.1 U-bus Access

The L2U registers are accessible from the U-bus side only if it is a supervisor mode data access and the register address is correct and it is indicated on the U-bus that it is a PPC register access.

A user mode access, or an access marked as instruction, to L2U registers from the U-bus side will cause a data error on the U-bus.

11.8.2 Transaction Size

All L2U registers are defined by PowerPC architecture as being 32-bit registers. There is no PowerPC instruction to access either a half word or a byte of the special purpose register. All L2U registers are only word accessible (read and write) in peripheral mode. A half-word or byte access in peripheral mode will result in a word transaction.

11.8.3 L2U Module Configuration Register (L2U_MCR)

The L2U module configuration register (L2U_MCR) is used to control the L2U module operation.

L2U_MCR — L2U Module Configuration Register **SPR 568**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SP	LSHOW		RESERVED												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7 L2U_MCR Bit Descriptions

Bit(s)	Name	Description
0	SP	SRAM Protection (SP) bit is used to protect the SRAM on the L-bus from U-bus accesses. This bit can be set or cleared from the L-bus side. It can be set or cleared from the U-bus side when factory test mode is enabled. When not in factory test mode, any attempt to set or clear the SP bit from the U-bus side has no affect. Once this bit is set, the L2U blocks all SRAM accesses initiated by the U-bus masters and the access is terminated with a data error on the U-bus. If L-bus show cycles are enabled, setting this bit will disable L-bus SRAM show cycles.
1:2	LSHOW	LSHOW bits are used to configure the show cycle mode for cycles accessing the L-bus slave e.g. SRAM 00 = Disable show cycles 01 = Show address and data of all L-bus space write cycles 10 = Reserved 11 = Show address and data of all L-bus space read and write cycles
3:31	—	Reserved

11.8.4 Region Base Address Registers (L2U_RBAx)

The region base address register defines the base address of a specific region protected by the data memory protection unit. There are four registers (x = 0...3), one for each supported region.

L2U_RBAx — L2U Region x Base Address Register **SPR 792 – 795**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RBA															
RESET:															
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RBA				RESERVED											
RESET:															
x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0

x = Undefined

Table 11-8 L2U_RBAx Bit Descriptions

Bit(s)	Name	Description
0:19	RBA	Region base address. The RBA field provides the base address of the region. The region base address should start on the block boundary for the corresponding block size attribute specified in the region attribute register (L2U_RAx).
20:31	—	Reserved

11.8.5 Region Attribute Registers (L2U_RAx)

Each region attribute register defines the protection attributes associated with a specific region protected by the data memory protection unit. There are four registers (x = 0...3), one for each supported region.

L2U_RAx — L2U Region X Attribute Register **SPR 824 – 827**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED								RS							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RS				PP		RESERVED			G	RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-9 L2U_RAx Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8:19	RS	Region size 0000_0000_0000 = 4 Kbytes 0000_0000_0001 = 8 Kbytes 0000_0000_0011 = 16 Kbytes 0000_0000_0111 = 32 Kbytes 0000_0000_1111 = 64 Kbytes 0000_0001_1111 = 128 Kbytes 0000_0011_1111 = 256 Kbytes 0000_0111_1111 = 512 Kbytes 0000_1111_1111 = 1 Mbyte 0001_1111_1111 = 2 Mbytes 0011_1111_1111 = 4 Mbytes 0111_1111_1111 = 8 Mbytes 1111_1111_1111 = 16 Mbytes
20:21	PP	Protection bits 00 = No supervisor access, no user access 01 = Supervisor read/write access, no user access 10 = Supervisor read/write access, user read-only access 11 = Supervisor read/write access, user read/write access
22:24	—	Reserved
25	G	Guarded attribute 0 = Not guarded from speculative accesses 1 = Guarded from speculative accesses
26:31	—	Reserved

11.8.6 Global Region Attribute Register

The global region attribute register defines the protection attributes associated with the memory region which is not protected under the four DMPU regions. This register also provides enable/disable control for the four DMPU regions.

L2U_GRA — L2U Global Region Attribute Register **SPR 536**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ENR0	ENR1	ENR2	ENR3	RESERVED											
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				PP		RESERVED			G	RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-10 L2U_GRA Bit Descriptions

Bit(s)	Name	Description
0	ENR0	Enable attribute for region 0 0 = Region attribute is off 1 = Region attribute is on
1	ENR1	Enable attribute for region 1 0 = Region attribute is off 1 = Region attribute is on
2	ENR2	Enable attribute for region 2 0 = Region attribute is off 1 = Region attribute is on
3	ENR3	Enable attribute for region 3 0 = Region attribute is off 1 = Region attribute is on
4:19	—	Reserved
20:21	PP	Protection bits 00 = No supervisor access, no user access 01 = Supervisor read/write access, no user access 10 = Supervisor read/write access, user read-only access 11 = Supervisor read/write access, user read/write access
22:24	—	Reserved
25	G	Guarded attribute 0 = Not guarded from speculative accesses 1 = Guarded from speculative accesses
26:31	—	Reserved

SECTION 12
U-BUS TO IMB3 BUS INTERFACE (UIMB)

The U-bus to IMB3 bus interface (UIMB) Interface structure is used to connect the CPU internal unified bus (U-bus) to the intermodule bus 3 (IMB3). It controls bus communication between the U-bus and the IMB3.

The UIMB interface consists of seven submodules that control bus interface timing, address decode, data multiplexing, intrasystem communication (interrupts), and clock generation to allow communication between U-bus and the IMB3. The seven submodules are:

- U-bus interface
- IMB3 interface
- Address decoder
- Data multiplexer
- Interrupt synchronizer
- Clock control
- Scan control

12.1 Features

- Provides complete interfacing between the U-bus and the IMB3:
  - — 15 bits (32 Kbytes) of address decode on IMB3
  - — 32-bit data bus
  - — Read/write access to IMB3 module registers[1]
  - — Interrupt synchronizer
  - — Monitoring of accesses to unimplemented addresses within UIMB interface address range
  - — Burst-inhibited accesses to the modules on IMB3
- Support of 32-bit and 16-bit BIUs for IMB3 modules
- Half and full speed operation of IMB3 bus with respect to U-bus
- Simple “slave only” U-bus interface implementation
  - — Supports alternate master on IMB3
  - — Transparent mode operation not supported
  - — Relinquish and retry not supported
- Supports scan control for modules on the IMB3 and on the U-bus

NOTE

Modules on the IMB3 bus can only be reset by $\overline{\text{SRESET}}$. Some modules may have a module reset, also.

[1]. The user should not perform instruction fetches from modules on the IMB.

12.2 UIMB Block Diagram

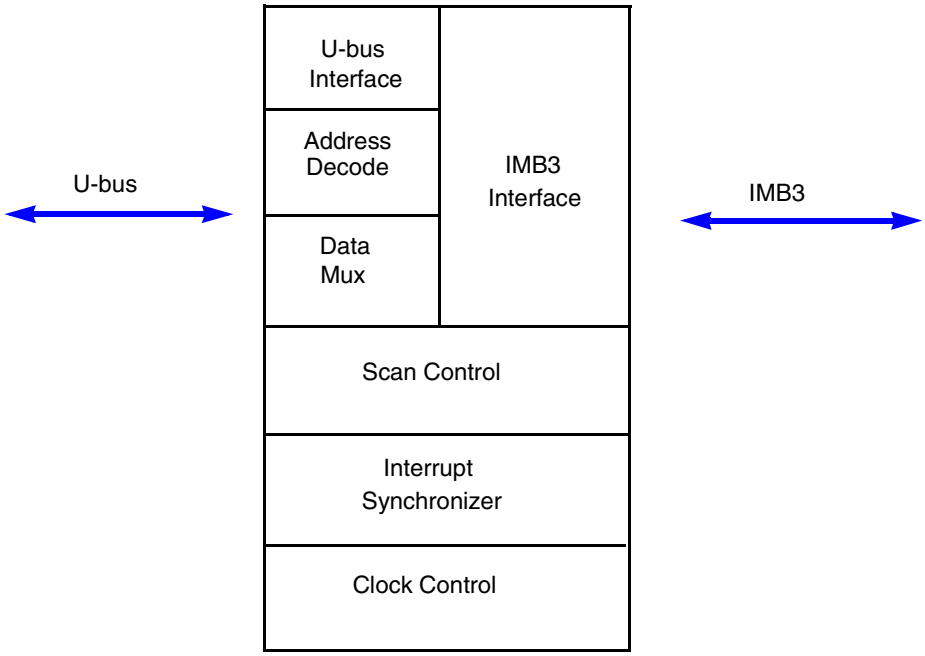


Figure 12-1 UIMB Interface Module Block Diagram

12.3 Clock Module

The clock module within the UIMB interface generates the IMB clock. The IMB clock is the main timing reference used within the IMB modules. The IMB clock is often referred to in the IMB module sections as F_{SYS} or IMB3 clock.

The IMB clock is generated based on the STOP and HSPEED bits in the UIMB module configuration register (UMCR). If the STOP bit is 1, the IMB clock is not generated. If the STOP bit is 0 and the HSPEED bit is 0, the IMB clock is generated as the inversion of the internal system clock. This is the same frequency as the CLKOUT if EBDF is 0b00 – full speed external bus. (See **Figure 12-2**.) If the HSPEED bit is 1, then the IMB clock is one-half of the internal system frequency. (See **Figure 12-3**.)

Table 12-1 STOP and HSPEED Bit Functionality

STOP	HSPEED	Functionality
0	0	IMB bus frequency is the same as U-bus frequency.
0	1	IMB bus frequency is half that of the U-bus frequency.
1	X	IMB clock is not generated.

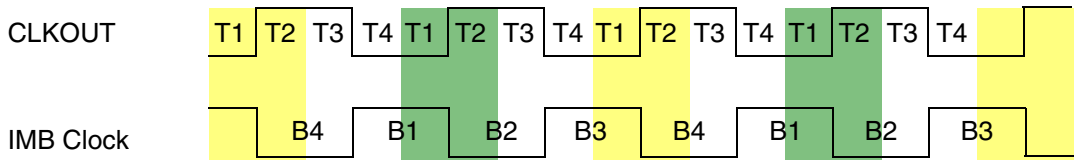


Figure 12-2 IMB Clock – Full-Speed IMB Bus

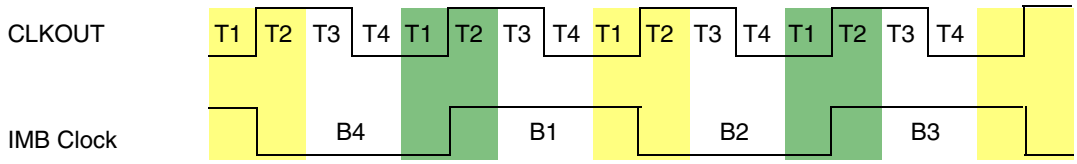


Figure 12-3 IMB Clock – Half-Speed IMB Bus

Table 12-2 shows the number of system clock cycles that the UIMB requires to perform each type of bus cycle. It is assumed in this table that the IMB3 is available to the UIMB at all times (fastest possible case).

Table 12-2 Bus Cycles and System Clock Cycles

Bus Cycle (from U-bus Transfer Start to U-bus Transfer Acknowledge)	Number of System Clock Cycles	
	Full Speed	Half Speed
Normal write	4	6
Normal read	4	6
Dynamically-sized write	6	10
Dynamically-sized read	6	10

NOTE

The UIMB interface dynamically interprets the port size of the addressed module during each bus cycle, allowing bus transfers to and from 16-bit and 32-bit IMB modules. During a bus transaction, the slave module on the IMB signals its port size (16- or 32-bit) via an internal port size signal.

12.4 Interrupt Operation

The interrupts from the modules on the IMB3 are propagated to the interrupt controller in the USIU through the UIMB interface. The UIMB interrupt synchronizer latches the Interrupts from the IMB3 and drives them onto the U-bus, where they are latched by the USIU interrupt controller.

12.4.1 Interrupt Sources and Levels on IMB

The IMB3 has eight interrupt lines. There can be a maximum of 32 levels of interrupts from the modules on IMB bus. A single module can be a source for more than one interrupt. For example, the QSMCM can generate two interrupts (one for QSCI1/QSCI2 and another for QSPI). In this case, the QSMCM has two interrupt sources. Each of these two sources can assert the interrupt on any of the 32 levels.

It is possible for multiple interrupt sources to assert the same interrupt level. To reduce the latency, it is a good practice for each interrupt source to assert an interrupt on a level on which no other interrupt source is mapped.

12.4.2 IMB Interrupt Multiplexing

The IMB has 10 lines for interrupt support. Eight lines are for interrupts and two for ILBS. These lines will transfer the 32 interrupt levels to the interrupt synchronizer. A diagram of the interrupt flow is shown in **Figure 12-4**.

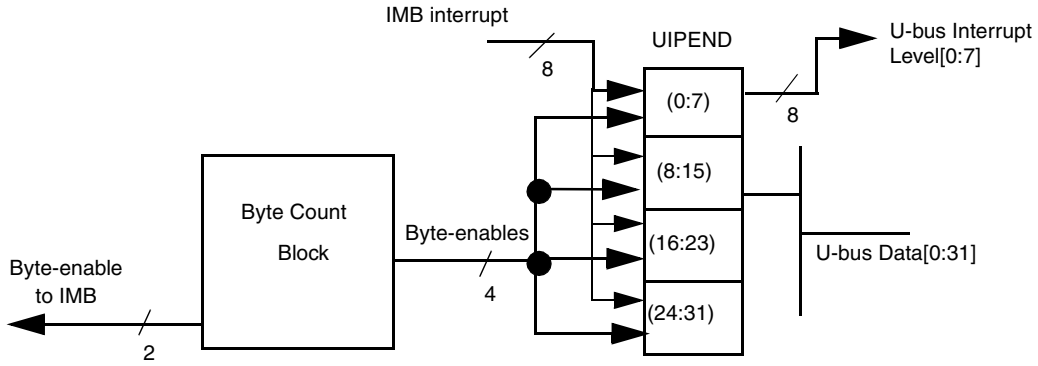


Figure 12-4 Interrupt Synchronizer Signal Flow

Latching 32 interrupt levels using eight IMB interrupt lines is accomplished with a 4:1 time-multiplexing scheme. The UIMB drives two signals (ILBS[0:1]) with a multiplexer select code that tells all interrupting modules on the IMB about which group of signals to drive during the next clock.

12.4.3 ILBS Sequencing

The IMB interface drives the ILBS signals continuously, incrementing through a code sequence (00, 01, 10, 11) once every clock The IRQMUX[0:1] bits in the IMB module configuration register select which type of multiplexing the Interrupt synchronizer will perform. The IRQMUX field can select time-multiplexing protocols for 8, 16, 24 or 32 interrupt sources. These protocols would take one, two, three or four clocks, respectively.

Table 12-4 shows ILBS sequencing. Programming IRQMUX[0:1] to 00 disables time multiplexing. In this case the ILBS lines remain at 00 at all times, as shown in Table 12-4. In this mode, no interrupts from IMB modules which assert on levels 8 through 31 are ever latched by the Interrupt synchronizer. Time multiplexing is disabled during reset, but the reset default value enables time multiplexing as soon as reset is released if the reset default value is not 00.

The timing for the scheme and the values of ILBS and the interrupt levels driven onto the IMB IRQ lines are shown in Figure 12-5. This scheme causes a maximum latency of four clocks and an average latency of two clocks before the interrupt request can reach the interrupt synchronizer.

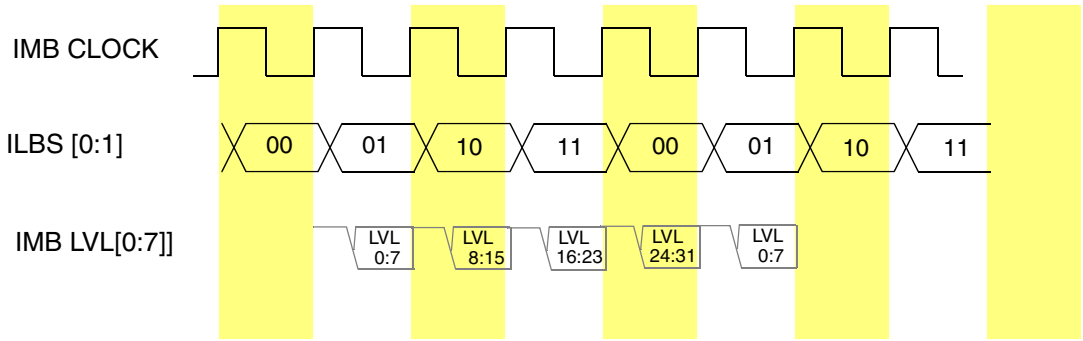


Figure 12-5 Time-Multiplexing Protocol for IRQ pins

Table 12-3 ILBS Signal functionality

ILBS[0:1]	Description
00	IMB interrupt sources mapped onto 0:7 levels will drive interrupts onto IMB $\overline{\text{IRQ}}$[0:7]
01	IMB interrupt sources mapped onto 8:15 levels will drive interrupts onto IMB $\overline{\text{IRQ}}$[0:7]
10	IMB interrupt sources mapped onto 16:23 levels will drive interrupts onto IMB $\overline{\text{IRQ}}$[0:7]
11	IMB interrupt sources mapped onto 24:31 levels will drive interrupts onto IMB $\overline{\text{IRQ}}$[0:7]

The IRQMUX bits determine how many levels of IMB interrupts are sampled. Refer to Table 12-4.

Table 12-4 IRQMUX Functionality

IRQMUX[0:1]	ILBS sequence	Description
00	00, 00, 00.....	Latch 0:7 IMB interrupt levels
01	00, 01, 00, 01....	Latch 0:15 IMB interrupt levels
10	00, 01, 10, 00, 01, 10,.....	Latch 0:23 IMB interrupt levels
11	00, 01, 10, 11, 00, 01, 10, 11,....	Latch 0:31 IMB interrupt levels

12.4.4 Interrupt Synchronizer

The interrupt synchronizer latches the 32 levels of interrupts from the IMB bus into a register which can be read by the CPU or other U-bus master. Since there are only eight lines for interrupts on the IMB and 32 levels of interrupts are possible, the 32 interrupt levels are multiplexed onto eight IMB interrupt lines. Apart from latching these interrupts in the register (UIPEND register), the interrupt synchronizer drives the interrupts onto the U-bus, where they are latched by the interrupt controller in the USIU.

If IMB modules drive interrupts on any of the 24 levels (levels eight through 31), they will be latched in the Interrupt pending register (UIPEND) in the UIMB. If any of the register bits 7 to 31 are set, then bit 7 will be set as well. Software must poll this register to find out which of the levels 7 to 31 are asserted.

The UIPEND register contains a status bit for each of the 32 interrupt levels. Each bit of the register is a read-only status bit, reflecting the current state of the corresponding interrupt signal. For each of the 32 interrupt levels, a corresponding bit of the UIPEND register is set.

Figure 12-4 shows how the eight interrupt lines are connected to the UIPEND register to represent 32 levels of interrupts. **Figure 12-6** shows the implementation of the interrupt synchronizer.

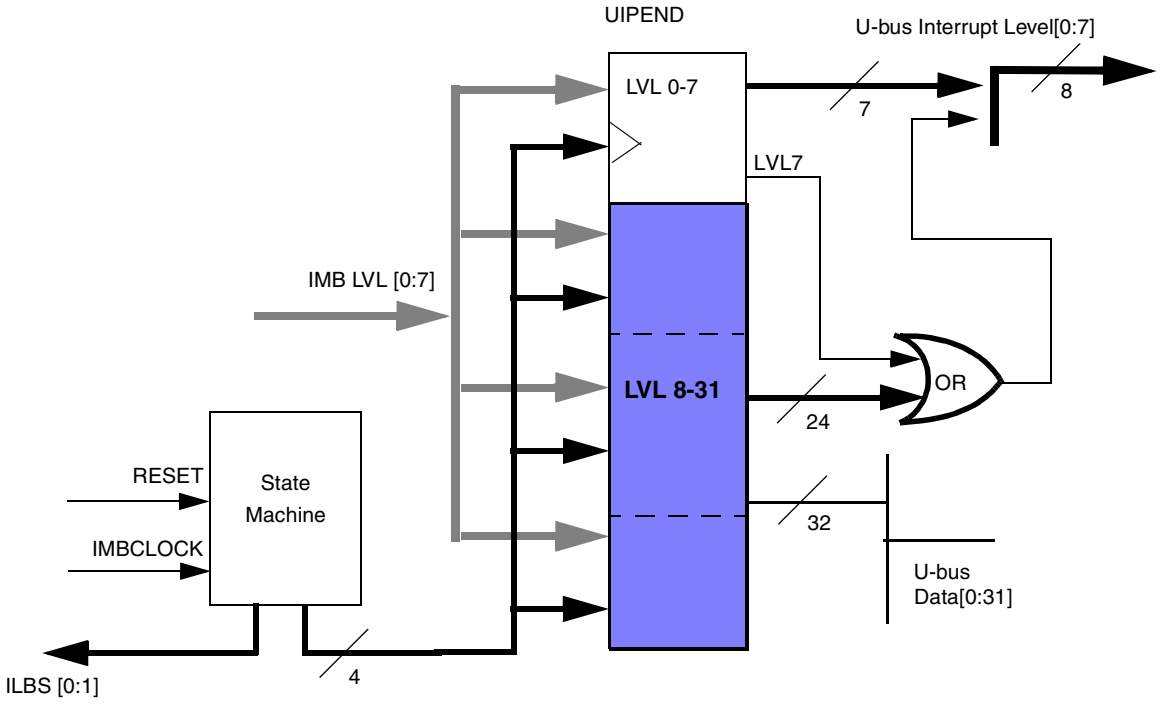


Figure 12-6 Interrupt Synchronizer Block diagram

12.5 Programming Model

Table 12-5 lists the registers used for configuring and testing the UIMB module. The address offset shown in this table is from the start of the block reserved for UIMB registers. As shown in **Figure 1-3** in **1.3 MPC555 / MPC556 Address Map**, this block begins at offset 0x30 7F80 from the start of the MPC555 / MPC556 internal memory map (the last 128-byte sub-block of the UIMB interface memory map).

Table 12-5 UIMB Interface Register Map

Access	Base Address	Register
S[1]	**0x30 7F80**	UIMB Module Configuration Register (UMCR) See **Table 12-6** for bit descriptions.
—	**0x30 7F84 — 0x30 7F8C**	Reserved
S/T	**0x30 7F90**	UIMB Test Control Register (UTSTCREG) Reserved
—	**0x30 7F94 — 0x30 7F9C**	Reserved
S	**0x30 7FA0**	Interrupt Request Pending (UIPEND) See **12.5.3 Pending Interrupt Request Register (UIPEND)** for bit descriptions.

NOTES:
1. S = Supervisor mode only, T = Test mode only

Any word, half-word or byte access to a 32-bit location within the UIMB interface register decode block that is unimplemented (defined as reserved) causes the UIMB interface to asserting a data error exception on the U-bus.The entire 32-bit location must be defined as reserved in order for a data error exception to be asserted.

Unimplemented bits in a register return zero when read.

12.5.1 UIMB Module Configuration Register (UMCR)

The UIMB module configuration register (UMCR) is accessible in supervisor mode only.

UMCR — UIMB Module Configuration Register **0x30 7F80**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	IRQMUX		HSPEED	RESERVED											
HRESET:															
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-6 UMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop enable. 0 = Enable system clock for IMB bus 1 = Disable IMB system clock To avoid complications at restart and data corruption, system software must stop each slave on the IMB before setting the STOP bit. Software must also ensure that all IMB interrupts have been serviced before setting this bit.
1:2	IRQMUX	Interrupt request multiplexing. These bits control the multiplexing of the 32 possible interrupt requests onto the eight IMB interrupt request lines. 00 = Disables the multiplexing scheme on the interrupt controller within this interface. What this means is that the IMB IRQ [0:7] signals are non-multiplexed, only providing 8 (0-7) interrupt request lines to the interrupt controller 01 = Enables the IMB IRQ control logic to perform a 2-to-1 multiplexing to allow transferring of 16 (0-15) interrupt sources 10 = Enables the IMB IRQ control logic to perform a 3-to-1 multiplexing to allow transferring of 24 (0-23) interrupt sources 11 = Enables the IMB IRQ control logic to perform a 4-to-1 multiplexing to allow transferring of 32 (0-31) interrupt sources
3	HSPEED	Half speed. The HSPEED bit controls the frequency at which the IMB3 runs with respect to the U-bus. This is a modify-once bit. Software can write the reset value of this bit any number of times. However, once logic 0 is written to this location, any attempt to rewrite this bit to a logic 1 will have no effect. 0 = IMB frequency is the same as that of the U-bus 1 = IMB frequency is one half that of the U-bus
4:31	—	Reserved

12.5.2 Test control register (UTSTCREG)

The UTSTCREG register is used for factory testing only.

12.5.3 Pending Interrupt Request Register (UIPEND)

The UIPEND register is a read-only status register which reflects the state of the 32 interrupt levels. The state of the IRQ0 is shown in bit 0, the state of IRQ1 is shown in bit 1 and so on. This register is accessible only in supervisor mode.

UIPEND — Pending Interrupt Request Register **0x30 7FA0**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LVL0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7	LVL8	LVL9	LVL0	LVL11	LVL12	LVL13	LVL14	LVL15
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
LVL16	IRQ17	LVL18	LVL19	LVL20	LVL21	LVL22	LVL23	LVL24	LVL25	LVL26	LVL27	LVL28	LVL29	LVL30	LVL31
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-7 UIPEND Bit Descriptions

Bit(s)	Name	Description
0:31	LVLx	Pending interrupt request level. Accessible only in supervisor mode. LVLx identifies the interrupt source as UIMB LVLx, where x is the interrupt number.

SECTION 13
QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64

The MPC555 / MPC556 includes two independent queued analog-to-digital converter (QADC64) modules. For details of QADC64 operation not included in this section, refer to the ***QADC Reference Manual* (QADCRM/AD)**.

13.1 Overview

The QADC64 consists of an analog front-end and a digital control subsystem, which includes an intermodule bus (IMB3) interface block. Refer to **Figure 13-1**.

The analog section includes input pins, channel selection logic, an analog multiplexer, and one sample-and-hold analog circuit. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array, a high-gain comparator, and a successive approximation register (SAR).

The digital control section contains the conversion sequencing logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result word table RAM.

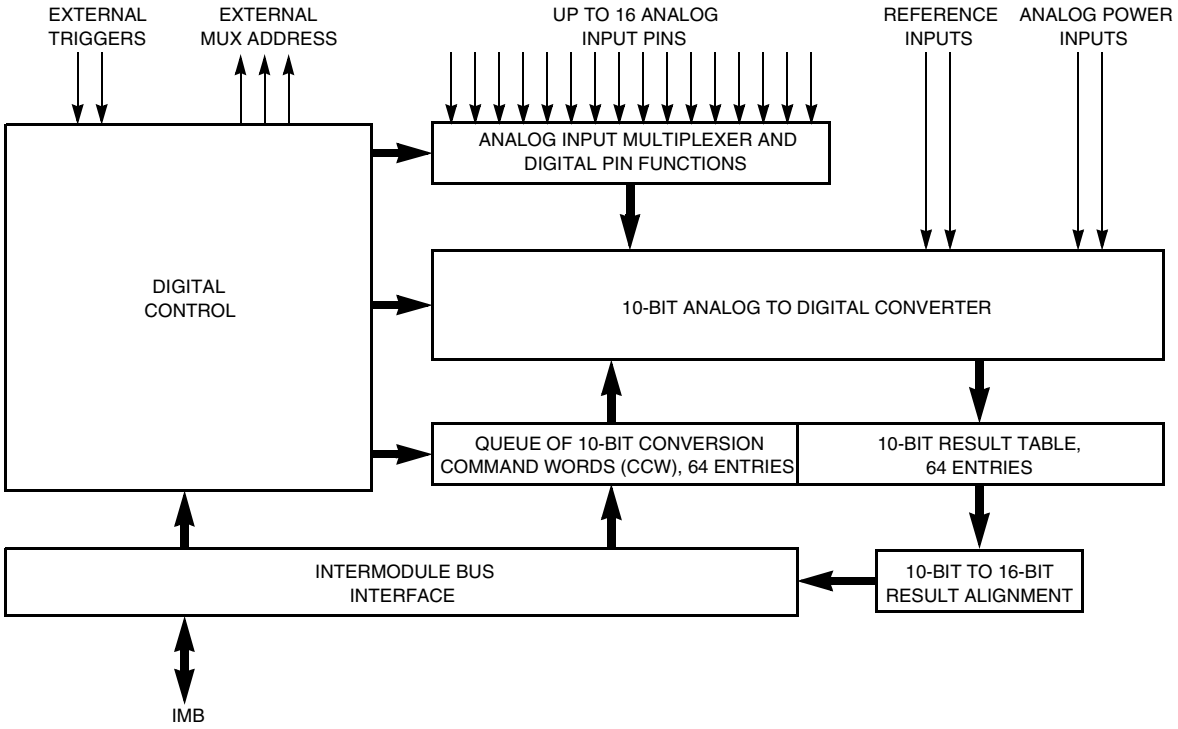


Figure 13-1 QADC64 Block Diagram

13.2 Features

Each QADC64 module offers the following features:

- Internal sample and hold
- Up to 16 analog input channels using internal multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 41 total input channels with internal and external multiplexing
- Programmable input sample time for various source impedances
- Two conversion command queues with a total of 64 entries
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Queue pointers indicate current location for each queue
- Automated queue modes initiated by:
  - — External edge trigger [queues 1 and 2] and gated mode [queue 1 only]
  - — Periodic/interval timer, within QADC64 module [queues 1 and 2]
  - — Software command [queues 1 and 2]
- Single-scan or continuous-scan of queues
- 64 result registers
- Output data readable in three formats:
  - — Right-justified unsigned
  - — Left-justified signed
  - — Left-justified unsigned
- Unused analog channels can be used as digital ports

13.3 QADC64 Pin Functions

The two QADC64 modules use the following 38 pins:

- Two analog reference pins, to which all analog input voltages are scaled (shared by the two modules)
- 32 analog input pins (16 per module, with three analog inputs per module multiplexed with multiplex address signals)
- Two analog power pins (shared by the two modules)
- Two external trigger pins (shared by the two modules)

The 16 channel/port pins in either module can support up to 41 channels when external multiplexing is used (including internal channels). All of the channel pins can also be used as general-purpose digital port pins.

The following paragraphs describe QADC64 pin functions. **Figure 13-2** shows the QADC64 module pins.

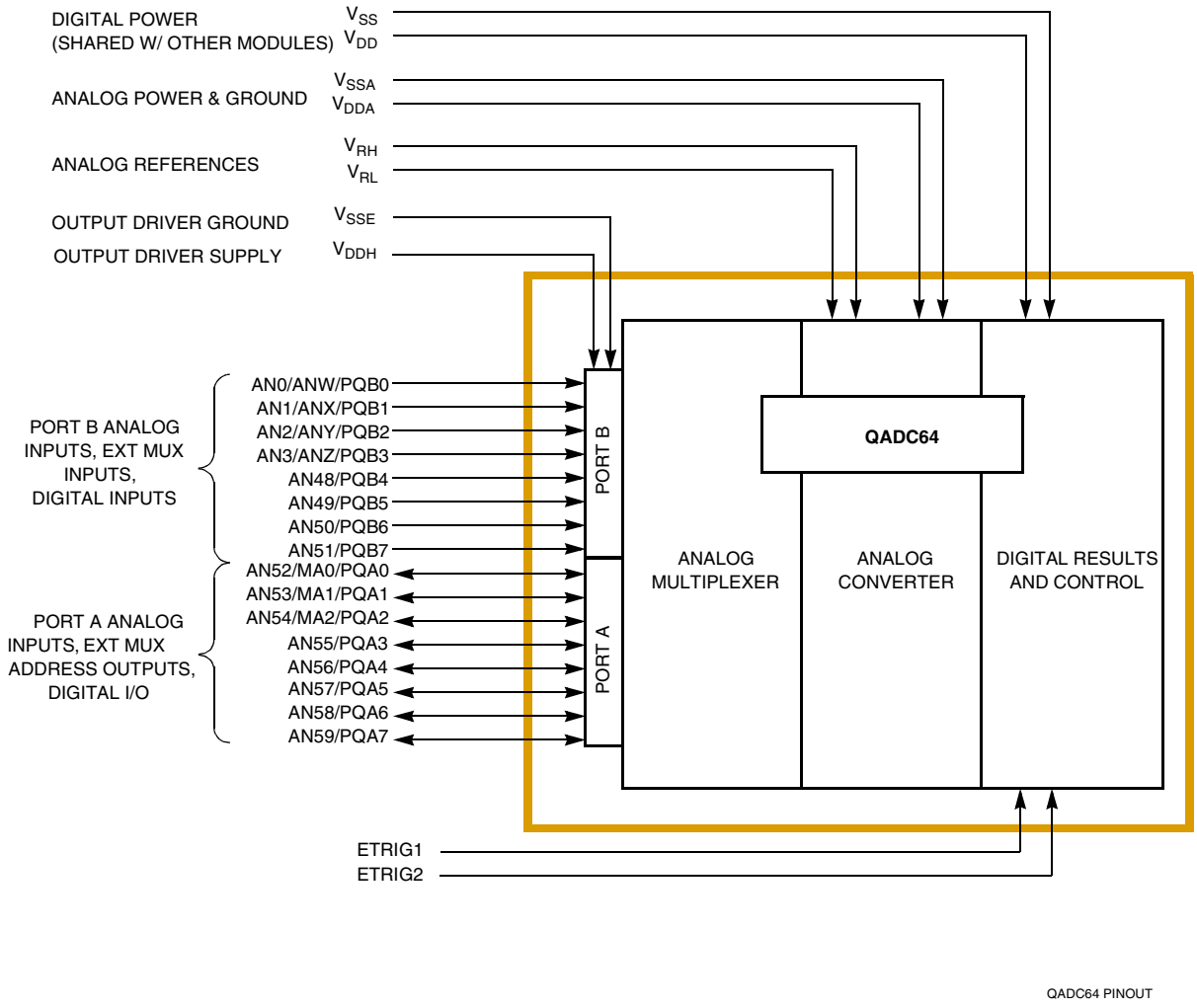


Figure 13-2 QADC64 Input and Output Signals

13.3.1 Port A Pin Functions

The eight port A pins can be used as analog inputs, or as a bi-directional 8-bit digital input/output port.

13.3.1.1 Port A Analog Input Pins

When used as analog inputs, the eight port A pins are referred to as AN[59:52]. Due to the digital output drivers associated with port A, the analog characteristics of port A are different from those of port B. All of the analog signal input pins may be used for at least one other purpose.

13.3.1.2 Port A Digital Input/Output Pins

Port A pins are referred to as PQA when used as a bidirectional 8-bit digital input/output port. These eight pins may be used for general-purpose digital input signals or digital output signals.

Port A pins are connected to a digital input synchronizer during reads and may be used as general purpose digital inputs. Since port A read captures the data on all pins, including those used for digital outputs or analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

Each port A pin is configured as an input or output by programming the port data direction register (DDRQA). Digital input signal states are read into the PORTQA data register when DDRQA specifies that the pins are inputs. Digital data in PORTQA is driven onto the port A pins when the corresponding bits in DDRQA specify outputs.

13.3.2 Port B Pin Functions

The eight port B pins can be used as analog inputs, or as an 8-bit digital input-only port. Refer to the following paragraphs for more information.

13.3.2.1 Port B Analog Input Pins

When used as analog inputs, the eight port B pins are referred to as AN[51:48]/AN[3:0]. Since port B functions as analog and digital input-only, the analog characteristics are different from those of port A. All of the analog signal input pins may be used for at least one other purpose.

13.3.2.2 Port B Digital Input Pins

Port B pins are referred to as PQB[7:0] when used as an 8-bit digital input-only port. In addition to functioning as analog input pins, the port B pins are also connected to the input of a synchronizer during reads and may be used as general-purpose digital inputs.

Since port B pins are input-only, there is no associated data direction register. Digital input signal states are read from the PORTQB data register. Since a port B read captures the data on all pins, including those used for analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

13.3.3 External Trigger Input Pins

The QADC64 has two external trigger pins (ETRIG[2:1]). Each of the two external trigger pins is associated with one of the scan queues. When a queue is in external trigger mode, the corresponding external trigger pin is configured as a digital input.

13.3.4 Multiplexed Address Output Pins

In non-multiplexed mode, the 16 channel pins are connected to an internal multiplexer which routes the analog signals into the A/D converter.

In externally multiplexed mode, the QADC64 allows automatic channel selection through up to four external 1-of-8 multiplexer chips. The QADC64 provides a 3-bit multiplexed address output to the external multiplexer chips to allow selection of one of eight inputs. The multiplexed address output signals MA[2:0] can be used as multiplex address output bits or as general-purpose I/O.

When externally-multiplexed mode is enabled, MA[2:0] are used as the address inputs for up to four 1-of-8 multiplexer chips (for example, the MC14051 and the MC74HC4051). Since MA[2:0] are digital outputs in multiplexed mode, the software programmed input/output direction and data for these pins in DDQA[2:0], DDRQA, and PQA[2:0] is ignored, and the value for MA[2:0] is taken from the currently executing CCW.

13.3.5 Multiplexed Analog Input Pins

In externally-multiplexed mode, four of the port B pins are redefined to each represent a group of eight input channels. Refer to **Table 13-1**.

The analog output of each external multiplexer chip is connected to one of the AN[w, x, y, z] inputs in order to convert a channel selected by the MA[2:0] multiplexed address outputs.

Table 13-1 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw	Even-numbered channels from 0 to 14
ANx	Odd-numbered channels from 1 to 15
ANy	Even-numbered channels from 16 to 30
ANz	Odd-numbered channels from 17 to 31

13.3.6 Voltage Reference Pins

V_{RH} and V_{RL} are the dedicated input pins for the high and low reference voltages. Separating the reference inputs from the power supply pins allows for additional external filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy.

13.3.7 Dedicated Analog Supply Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC64 module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply.

13.3.8 External Digital Supply Pin

Each port A pin includes a digital output driver, an analog input signal path, and a digital input synchronizer. The V_{SS} pin provides the ground level for the drivers on the port A pins. V_{DDH} provides the supply level for the drivers on port A pins.

13.3.9 Digital Supply Pins

V_{DD} and V_{SS} provide the power for the digital portions of the QADC64, and for all other digital MCU modules.

13.4 QADC64 Bus Interface

The QADC64 supports to 8-bit, 16-bit, and 32-bit data transfers, at even and odd addresses. Coherency of results read, (ensuring that all results read were taken consecutively in one scan) is not guaranteed. For example, if two consecutive 16-bit locations in a result area are read, the QADC64 could change one 16-bit location in the result area between the bus cycles. There is no holding register for the second 16-bit location. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. Depending on bus master protocol, these accesses could include misaligned and 32-bit accesses.

Normal reads-from and writes-to the QADC64 require two clock cycles. However, if the CPU tries to access locations that are also accessible to the QADC64 while the QADC64 is accessing them, the bus cycle will require additional clock cycles. The QADC64 may insert from one to four wait states in the process of a CPU read from or write to such a location.

13.5 Module Configuration

The QADC64 module configuration register (QADC64MCR) defines freeze and stop mode operation, supervisor space access, and interrupt arbitration priority. Unimplemented bits read zero and writes have no effect. QADC64MCR is typically written once when software initializes the QADC64, and not changed thereafter. Refer to **13.12.1 QADC64 Module Configuration Register** for register and bit descriptions.

13.5.1 Low-Power Stop Mode

When the STOP bit in QADC64MCR is set, the clock signal to the A/D converter is disabled, effectively turning off the analog circuitry. This results in a static, low power consumption, idle condition. Low-power stop mode aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off in low-power stop mode, the QADC64 requires some recovery time to stabilize the analog circuits after the STOP bit is cleared.

In low-power stop mode, the BIU state machine and logic do not shut down, and the QADC64MCR, the interrupt register (QADC64INT), and the test register (QADC64TEST) are fully accessible and are not reset. The data direction register (DDRQA), port data register (PORTQA/PORTQB), and control register zero (QACR0) are not reset and are read-only accessible. The RAM is not reset and is not accessible. Control register one (QACR1), control register two (QACR2), and the status registers (QASR0 and QASR1) are reset and are read-only accessible. In addition, the periodic/interval timer is held in reset during stop mode.

If the STOP bit is clear, low-power stop mode is disabled. The STOP bit must be clear to program CCWs into RAM or read results from RAM.

13.5.2 Freeze Mode

The QADC64 enters freeze mode when background debug mode is enabled and a breakpoint is processed. This is indicated by assertion of the FREEZE line on the

IMB3. The FRZ bit in QADC64MCR determines whether or not the QADC64 responds to an IMB FREEZE assertion. Freeze mode is useful when debugging an application.

When the IMB FREEZE line is asserted and the FRZ bit is set, the QADC64 finishes any conversion in progress and then freezes. Depending on when the FREEZE is asserted, there are three possible queue freeze scenarios:

- When a queue is not executing, the QADC64 freezes immediately
- When a queue is executing, the QADC64 completes the current conversion and then freezes
- If during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC64 freezes immediately

When the QADC64 enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock, QCLK, is held in reset and the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not captured. The BIU remains active to allow IMB access to all QADC64 registers and RAM. Although the QADC64 saves a pointer to the next CCW in the current queue, the software can force the QADC64 to execute a different CCW by writing new queue operating modes for normal operation. The QADC64 looks at the queue operating modes, the current queue pointer, and any pending trigger events to decide which CCW to execute.

If the FRZ bit is clear, assertion of the IMB FREEZE line is ignored.

13.5.3 Supervisor/Unrestricted Address Space

The QADC64 memory map is divided into two segments: supervisor-only data space and assignable data space. Access to supervisor-only data space is permitted only when the CPU is operating in supervisor mode. Assignable data space can have either restricted to supervisor-only data space access or unrestricted supervisor and user data space accesses. The SUPV bit in QADC64MCR designates the assignable space as supervisor or unrestricted.

Attempts to read or write supervisor-only data space when the CPU is not in supervisor mode cause the bus master to assert the internal transfer error acknowledge ($\overline{\text{TEA}}$) signal.

The supervisor-only data space segment contains the QADC64 global registers, which include QADC64MCR, QADC64TEST, and QADC64INT. The supervisor/unrestricted space designation for the CCW table, the result word table, and the remaining QADC64 registers is programmable.

13.6 General-Purpose I/O Port Operation

QADC64 port pins, when used as general-purpose input, are conditioned by a synchronizer with an enable feature. The synchronizer is not enabled until the QADC64 decodes an IMB bus cycle which addresses the port data register to minimize the high-

current effect of mid-level signals on the inputs used for analog signals. Digital input signals must meet the input low voltage (VIL) or input high voltage (VIH) specifications. If an analog input pin does not meet the digital input pin specifications when a digital port read operation occurs, an indeterminate state is read. To avoid reading inappropriate values on analog inputs, the user software should employ a “masking” operation.

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input (port A only). When the data direction bit specifies the pin to be an output, the content of the port data register is read. By reading the latch which drives the output pin, software instructions that read data, modify it, and write the result, like bit manipulation instructions, work correctly.

There is one special case to consider for digital I/O port operation. When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PQA[2:0], the three multiplexed address MA[2:0] output pins. The MA[2:0] pins are forced to be digital outputs, regardless of the data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the multiplexed address latches which drive MA[2:0], regardless of the data direction setting.

13.6.1 Port Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB). Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs AN[59:52] and external multiplexer address outputs MA[2:0].

Port B pins are referred to as PQB when used as an 8-bit input-only digital port. Port B can also be used for non-multiplexed AN[51:48]/AN[3:0] and multiplexed ANz, ANy, ANx, ANw analog inputs.

PORTQA and PORTQB are unaffected by reset. Refer to **13.12.4 Port A/B Data Register** for register and bit descriptions.

13.6.2 Port Data Direction Register

The port data direction register (DDRQA) is associated with the port A digital I/O pins. These bi-directional pins may have somewhat higher leakage and capacitance specifications.

Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. Software is responsible for ensuring that DDRQA bits are not set to one on pins used for analog inputs. When a DDRQA bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

Caution should be exercised when mixing digital and analog inputs. This should be minimized as much as possible. Input pin rise and fall times should be as large as possible to minimize AC coupling effects.

Since port B is input-only, a data direction register is not needed. Read operations on the reserved bits in DDRQA return zeros, and writes have no effect. Refer to **13.12.5 Port Data Direction Register** for register and bit descriptions.

13.7 External Multiplexing Operation

External multiplexers concentrate a number of analog signals onto a few inputs to the analog converter. This is helpful in applications that need to convert more analog signals than the A/D converter can normally provide. External multiplexing also puts the multiplexer closer to the signal source. This minimizes the number of analog signals that need to be shielded due to the close proximity of noisy, high-speed digital signals near the MCU.

The QADC64 can use from one to four external multiplexers to expand the number of analog signals that may be converted. Up to 32 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected from the channel field of the CCW table, the same as internally multiplexed channels.

All of the automatic queue features are available for externally and internally multiplexed channels. The software selects externally multiplexed mode by setting the MUX bit in QACR0.

Figure 13-3 shows the maximum configuration of four external multiplexers connected to the QADC64. The external multiplexers select one of eight analog inputs and connect it to one analog output, which becomes an input to the QADC64. The QADC64 provides three multiplexed address signals (MA[2:0]), to select one of eight inputs. These outputs are connected to all four multiplexers. The analog output of each multiplexer is each connected to one of four separate QADC64 inputs — ANw, ANx, ANy, and ANz.

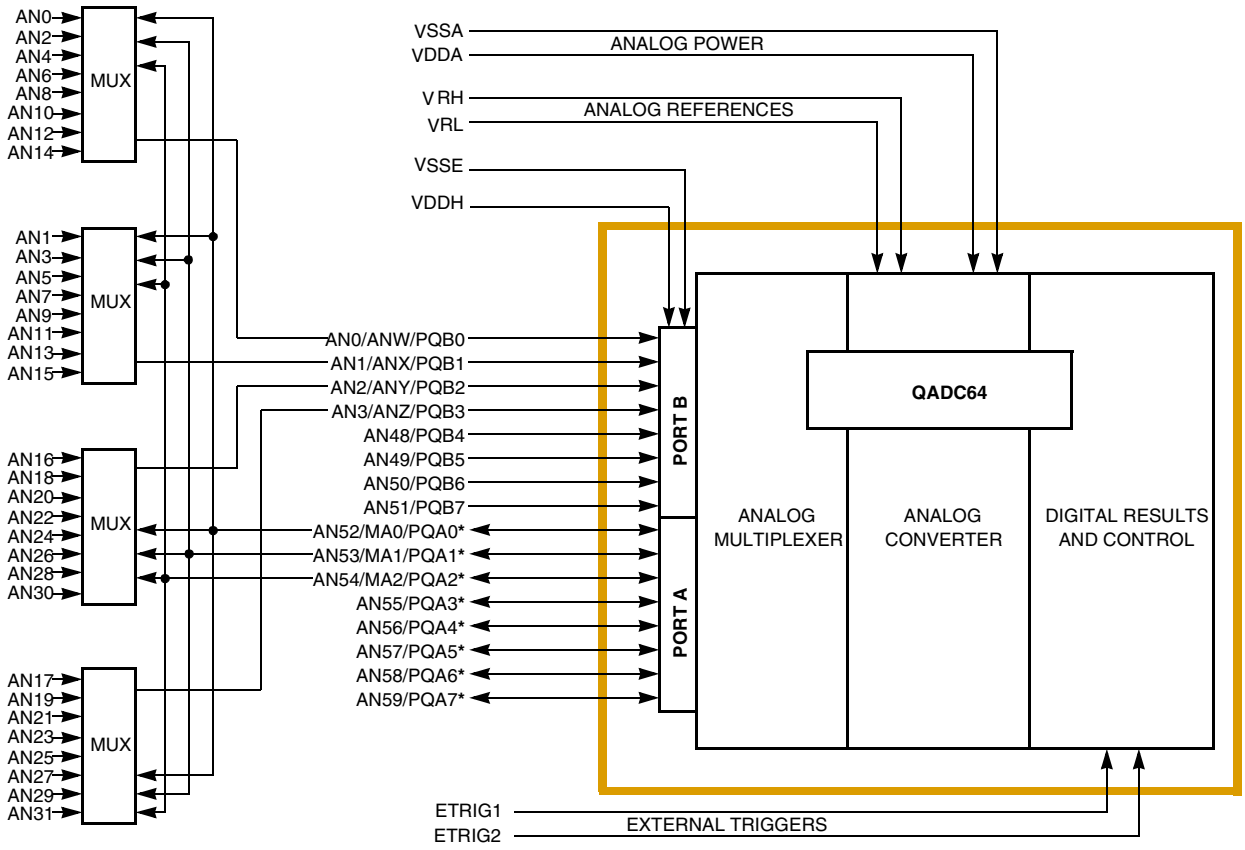


Figure 13-3 Example of External Multiplexing

When the external multiplexed mode is selected, the QADC64 automatically creates the MA[2:0] output signals from the channel number in each CCW. The QADC64 also converts the proper input channel (ANw, ANx, ANy, and ANz) by interpreting the CCW channel number. As a result, up to 32 externally multiplexed channels appear to the conversion queues as directly connected signals. Software simply puts the channel number of an externally multiplexed channel into a CCW.

Figure 13-3 shows that MA[2:0] may also be analog or digital input pins. When external multiplexing is selected, none of the MA[2:0] pins can be used for analog or digital inputs. They become multiplexed address outputs.

13.8 Analog Input Channels

The number of available analog channels varies, depending on whether or not external multiplexing is used. A maximum of 16 analog channels are supported by the internal multiplexing circuitry of the converter. **Table 13-2** shows the total number of analog input channels supported with zero to four external multiplexers.

Table 13-2 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels[1]				
No External MUX Chips	**One External MUX Chip**	**Two External MUX Chips**	**Three External MUX Chips**	**Four External MUX Chips**
16	12 + 8 = 20	11 + 16 = 27	10 + 24 = 34	9 + 32 = 41

NOTES:

1. When external multiplexing is used, three input channels become multiplexed address outputs, and for each external multiplexer chip, one input channel becomes a multiplexed analog input.

13.9 Analog Subsystem

The QADC64 analog subsystem includes a front-end analog multiplexer, a digital to analog converter (DAC) array, a comparator, and a successive approximation register (SAR).

The analog subsystem path runs from the input pins through the input multiplexing circuitry, into the DAC array, and through the analog comparator. The output of the comparator feeds into the SAR.

Figure 13-4 shows a block diagram of the QADC64 analog submodule.

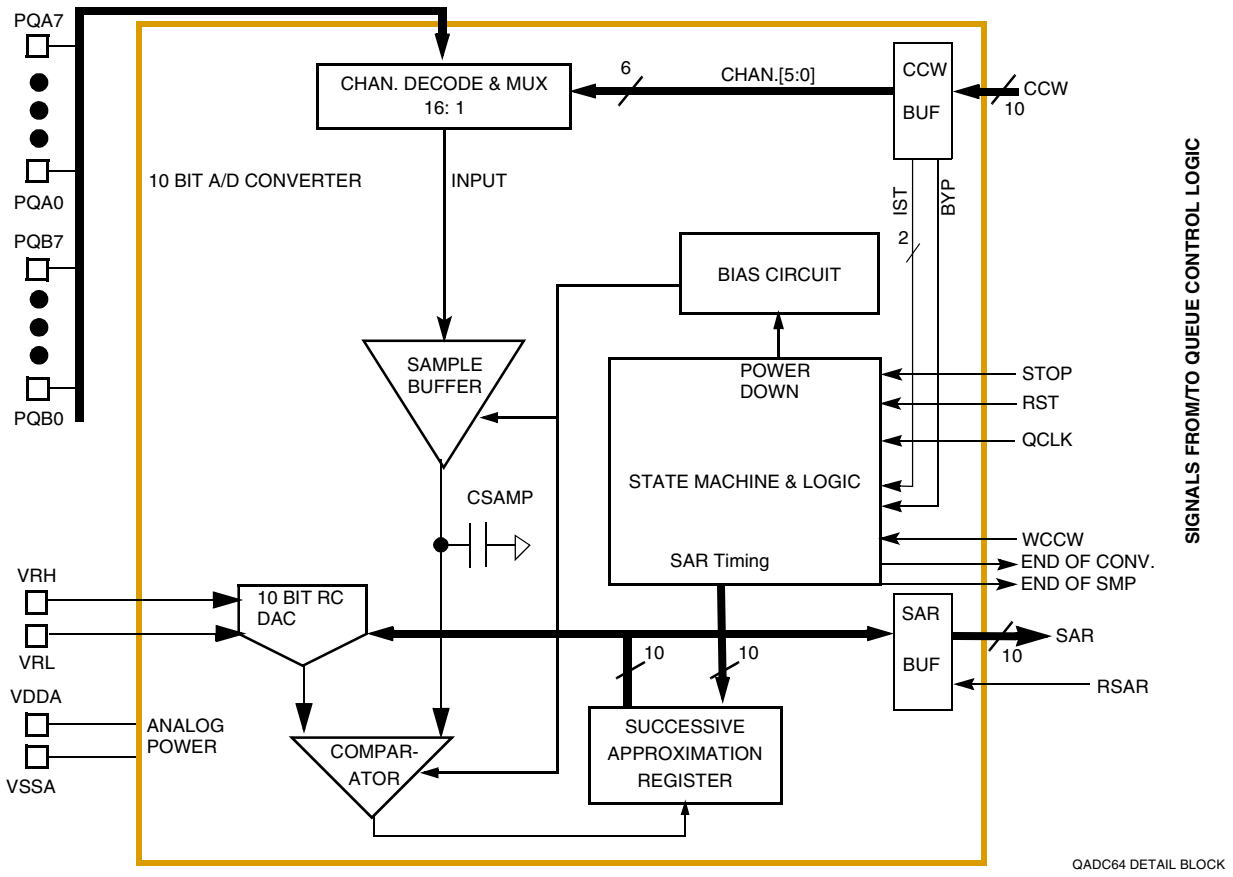


Figure 13-4 QADC64 Module Block Diagram

13.9.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is driven by the buffer amplifier onto the sample capacitor. The buffer amplifier can be disabled by means of the BYP bit in the CCW. During the final sampling period, amplifier is bypassed, and the multiplexer input charges the RC DAC array directly. During the resolution period, the voltage in the RC DAC array is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLK cycles. Final sample time can be 2, 4, 8, or 16 QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is ten QCLK cycles.

Sample and resolution require a minimum of 14 QCLK clocks (7 µs with a 2-MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 13.0 µs with a 2-MHz QCLK.

Figure 13-5 illustrates the timing for conversions. This diagram assumes a final sampling period of two QCLK cycles.

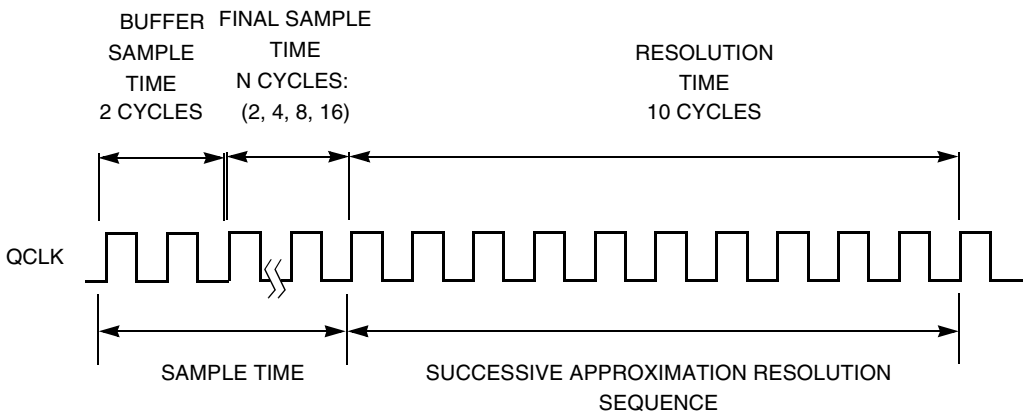


Figure 13-5 Conversion Timing

13.9.1.1 Amplifier Bypass Mode Conversion Timing

If the amplifier bypass mode is enabled for a conversion by setting the amplifier bypass (BYP) bit in the CCW, the timing changes to that shown in **Figure 13-6**. The buffered sample time is eliminated, reducing the potential conversion time by two QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. This results in no savings of QCLKs. When using the bypass mode, the external circuit should be of low source impedance, typically less than 10 kΩ. Also, the loading effects of the external circuitry by the QADC64 need to be considered, since the benefits of the sample amplifier are not present.

NOTE

Because of internal RC time constants, a sample time of two QCKLs in bypass mode for high frequency operation is not recommended.

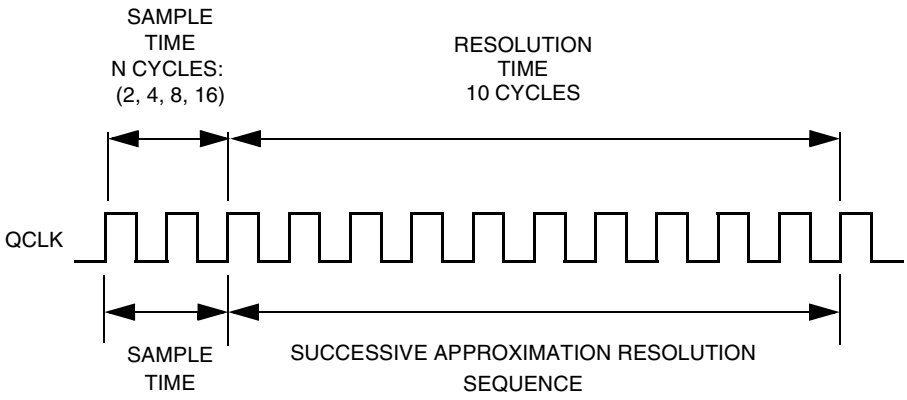


Figure 13-6 Bypass Mode Conversion Timing

13.9.2 Front-End Analog Multiplexer

The internal multiplexer selects one of the 16 analog input pins or one of three special internal reference channels for conversion. The following are the three special channels:

- V_{RH} — Reference Voltage High
- V_{RL} — Reference Voltage Low
- $(V_{RH} – V_{RL})/2$ — Mid-Reference Voltage

The selected input is connected to one side of the DAC capacitor array. The other side of the DAC array is connected to the comparator input. The multiplexer also includes positive and negative stress protection circuitry, which prevents other channels from affecting the present conversion when excessive voltage levels are applied to the other channels.

13.9.3 Digital-to-Analog Converter Array

The digital-to-analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The array serves two purposes:

- The array holds the sampled input voltage during conversion
- The resistor-capacitor array provides the mechanism for the successive approximation A/D conversion

Resolution begins with the MSB and works down to the LSB. The switching sequence is controlled by the SAR logic.

13.9.4 Comparator

The comparator is used during the approximation process to sense whether the digitally selected arrangement of the DAC array produces a voltage level higher or lower than the sampled input. The comparator output feeds into the SAR which accumulates the A/D conversion result sequentially, starting with the MSB.

13.9.5 Successive Approximation Register

The input of the successive approximation register (SAR) is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the ten bits of the conversion result, the SAR data is transferred by the queue control logic in the digital section to the appropriate result location, where it may be read by user software.

13.10 Digital Control Subsystem

The digital control subsystem includes the clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of the QADC64 conversions is the 64-entry CCW table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, sub-queues can

be created within the two queues. Each queue can be operated using several different scan modes. The scan modes for queue 1 and queue 2 are programmed in QACR1 and QACR2. Once a queue has been started by a trigger event (any of the ways to cause the QADC64 to begin executing the CCWs in a queue or sub-queue), the QADC64 performs a sequence of conversions and places the results in the result word table.

13.10.1 Queue Priority

Queue 1 has execution priority over queue 2 execution. **Table 13-3** shows the conditions under which queue 1 asserts its priority:

Table 13-3 Queue 1 Priority Assertion

Queue State	Result
Inactive	A trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
Queue 1 active/trigger event occurs for Queue 2	Queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are recorded as trigger overruns.
Queue 2 active/trigger event occurs for Queue 1	The current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while queue 2 is suspended are recorded as trigger overruns. Once queue 1 reaches the completion or the paused state, queue 2 begins executing again. The programming of the resume bit in QACR2 determines which CCW is executed in queue 2.
Simultaneous trigger events occur for Queue 1 and Queue 2	Queue 1 begins execution and the queue 2 status is changed to trigger pending.
sub-queues paused	The pause feature can be used to divide queue 1 and/or queue 2 into multiple sub-queues. A sub-queue is defined by setting the pause bit in the last CCW of the sub-queue.

Figure 13-7 shows the CCW format and an example of using pause to create sub-queues. Queue 1 is shown with four CCWs in each sub-queue and queue 2 has two CCWs in each sub-queue.

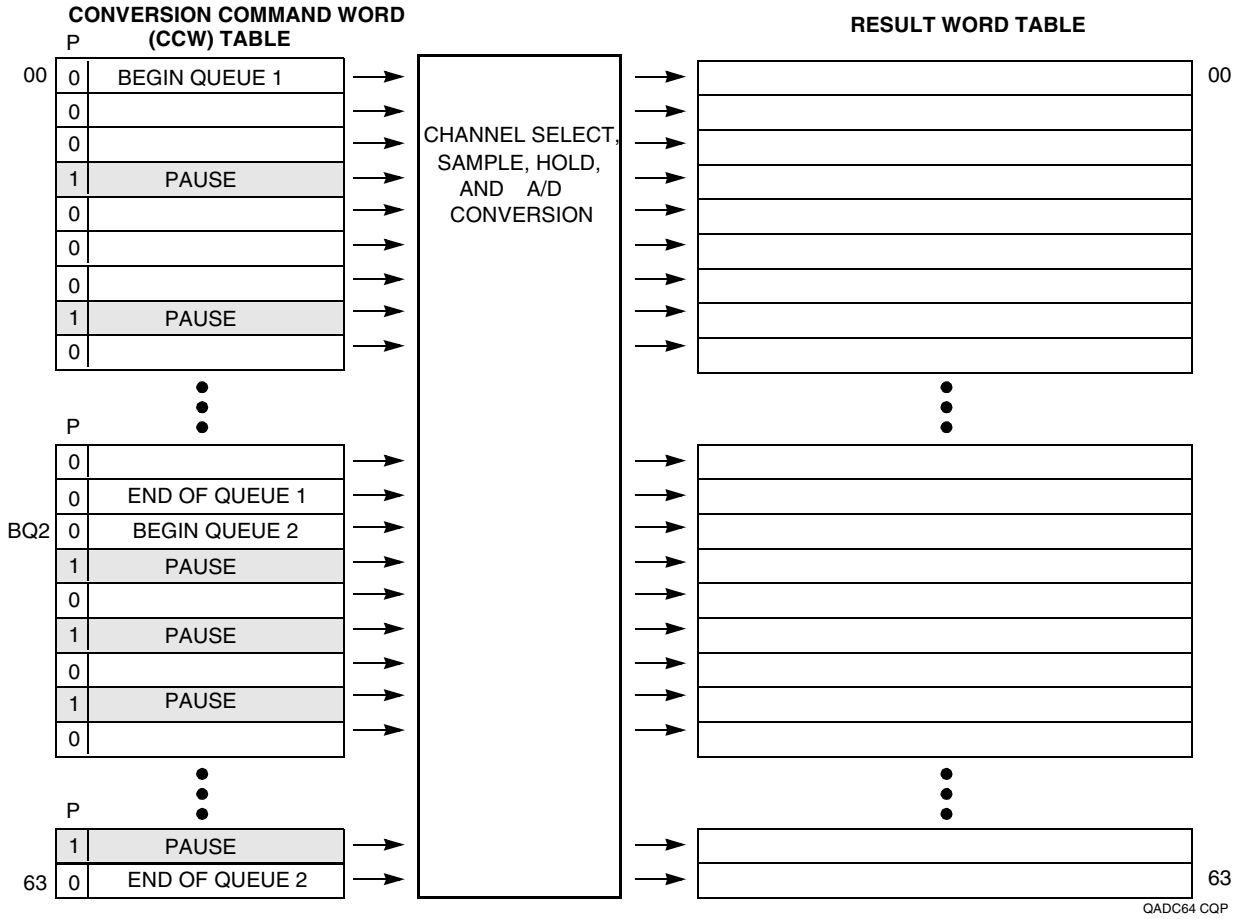


Figure 13-7 QADC64 Queue Operation with Pause

The queue operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the sub-queues within queue 1. Similarly, the queue operating mode for queue 2 determines the type of trigger event required to execute each of the sub-queues within queue 2.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each sub-queue. Once a sub-queue is initiated, each CCW is executed sequentially until the last CCW in the sub-queue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the next sub-queue. A sub-queue cannot be executed a second time before the overall queue execution has been completed.

Trigger events which occur during the execution of a sub-queue are ignored, except that the trigger overrun flag is set. When continuous-scan mode is selected, a trigger event occurring after the completion of the last sub-queue (after the queue completion flag is set), causes execution to continue with the first sub-queue, starting with the first CCW in the queue.

When the QADC64 encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set and a pause software interrupt may optionally be issued. The status of the queue is shown to be paused, indicating completion of a sub-queue. The QADC64 then waits for another trigger event to again begin execution of the next sub-queue.

13.10.2 Queue Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set at the end of the CCW table (63) and a trigger event occurs on queue 2. **13.12.8 QADC64 Control Register 2 (QACR2)** on BQ2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set beyond the end of the CCW table (64 - 127) and a trigger event occurs on queue 2. Refer to 7.6.3 Control Register two for information on BQ2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in the QADC64 behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC64 reads CCW0 and detects both end-of-queue conditions. The completion flag is set for queue 1 only and it becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, since the end-of-queue condition is recognized, the completion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 (EOQ) code is in CCW6
- The pause bit is set in CCW63
- During queue 1 operation, the pause bit is set in CCW14 and BQ2 points to CCW15

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC64 sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW0 and EOQ is programmed into CCW0
- During queue 1 operation, the pause bit is set in CCW20, which is also BQ2

13.10.3 Scan Modes

The QADC64 queuing mechanism provides several methods for automatically scanning input channels. In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed. The possible modes are:

- Disabled and reserved mode
- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode
- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

The following paragraphs describe the disabled/reserved, single-scan, and continuous-scan operations.

13.10.3.1 Disabled Mode

When the disabled mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, wait states are not encountered for IMB accesses of the RAM. When both queues are disabled, it is safe to change the QCLK prescaler values.

13.10.3.2 Reserved Mode

Reserved mode allows for future mode definitions. When the reserved mode is selected, the queue is not active.

CAUTION

Do not use a reserved mode. Unspecified operations may result.

13.10.3.3 Single-Scan Modes

When the application software wants to execute a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected. By programming the MQ field in QACR1 or QACR2, the following modes can be selected:

- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode

NOTE

Queue 2 can not be programmed for external gated single-scan mode.

In all single-scan queue operating modes, the software must also enable the queue to begin execution by writing the single-scan enable bit to a one in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2 respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a one during the write cycle, which selects the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The completion flag, completion interrupt, or queue status are used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC64 to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC64 resets the single-scan enable bit to zero. If the single-scan enable bit is written to a one or a zero by the software before the queue scan is complete, the queue is not affected. However, if the software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The conversion in progress is aborted and the new queue operating mode takes effect.

In the software initiated single-scan mode, the writing of a one to the single-scan enable bit causes the QADC64 to internally generate a trigger event and the queue execution begins immediately. In the other single-scan queue operating modes, once the single-scan enable bit is written, the selected trigger event must occur before the queue can start. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger single-scan mode and the interval timer single-scan mode.

In the interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After the queue execution is complete, the queue status is shown as idle. The software can restart the queue by setting the single-scan enable bit to a one. Queue execution begins with the first CCW in the queue.

Software Initiated Single-Scan Mode. Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting the software initiated single-scan mode, and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC64 immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC64 automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the software again sets the single-scan enable bit. While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused. The trigger overrun flag is never set while in the software initiated single-scan mode.

The software initiated single-scan mode is useful in the following applications:

- Allows software complete control of the queue execution
- Allows the software to easily alternate between several queue sequences

External Trigger Single-Scan Mode. The external trigger single-scan mode is a variation of the external trigger continuous-scan mode, and is also available with both queue 1 and queue 2. The software programs the polarity of the external trigger edge that is to be detected, either a rising or a falling edge. The software must enable the scan to occur by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC64 clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of the queue to be initiated by the next external trigger edge.

The external trigger single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though the software is not interested in data taken from every edge. The software can start the external trigger single-scan mode and get one set of data, and at a later time, start the queue again for the next set of samples.

When a pause bit is encountered during external trigger single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

The external trigger single-scan mode is also useful when the software needs to change the polarity of the external trigger so that both the rising and falling edges cause queue execution.

External Gated Single-Scan Mode. The QADC64 provides external gating for queue 1 only. When external gated single-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Software must enable the scan to occur by setting the single-scan enable bit for queue 1. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC64 sets the completion flag (CF1) and clears the

single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. Software can read the CWPQ1 to determine the last valid conversion in the queue. Software must set the single-scan enable bit again and should clear the PF1 bit before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Single-Scan Mode. Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128 to 128 Kbytes times the QCLK period in binary multiples. When the interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR1(2), the timer begins counting. When the time interval elapses, an internal trigger event is created to start the queue and the QADC64 begins execution with the first CCW.

The QADC64 automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval elapses again, and queue execution continues. When the queue execution reaches an end of queue situation the single-scan enable bit is cleared. Software may set the single-scan enable bit again, allowing another scan of the queue to be initiated by the interval timer.

The interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause, or may be considered a trigger overrun. Once the queue execution is completed, the single-scan enable bit must be set again to enable the timer to count again.

Normally only one queue will be enabled for interval timer single-scan mode and the timer will reset at the end of queue. However, if both queues are enabled for either single-scan or continuous interval timer mode, the end of queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end of queue.

The interval timer single-scan mode can be used in applications which need coherent results, for example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog pins
- When the interrupt rate in the periodic timer continuous-scan mode would be too high
- In sensitive battery applications, where the single-scan mode uses less power than the software initiated continuous-scan mode

13.10.3.4 Continuous-Scan Modes

When the application software wants to execute multiple passes through a sequence of conversions defined by a queue, a continuous-scan queue operating mode is se-

lected. By programming the MQ1(2) field in QACR1(2), the following software initiated modes can be selected:

- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of the software initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic timer continuous-scan mode.

After the queue execution is complete, the queue status is shown as idle. Since the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed to enable queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

In this version of QADC64, coherent samples can be guaranteed. The time between consecutive conversions has been designed to be consistent, provided the sample time bits in both the CCW and IST are identical. However, there is one exception. For queues that end with a CCW containing EOQ code (channel 63), the last queue conversion to the first queue conversion requires one additional CCW fetch cycle. Therefore continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion can not be guaranteed since it is a function of clock synchronization, programmable trigger events, queue priorities, and so on.

Software Initiated Continuous-Scan Mode. When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC64. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and then execution continues without pausing. When the end-of-queue is reached, another internal trigger event is generated, and queue execution begins again from the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is idle. The trigger overrun flag is never set while in the software initiated continuous-scan mode.

The software initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The software can always read the result table to get the latest converted value for each channel. The channels scanned are kept up-to-date by the QADC64 without software involvement. Software can read a result value at any time.

The software initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When the software initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything, or for the slow-to-change analog channels. Interrupts are normally not used with the software initiated continuous-scan mode. Rather, the software reads the latest conversion result from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel. Data read at different locations, however, may or may not be coherent (that is, from the same queue scan sequence).

External Trigger Continuous-Scan Mode. The QADC64 provides external trigger pins for both queues. When the external trigger software initiated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external trigger signal is programmable, so that the software can choose to begin queue execution on the rising or falling edge. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, the queue execution begins again automatically. Software initialization is not needed between trigger events.

When a pause bit is encountered in external trigger continuous-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, since interrupt response times vary.

External Gated Continuous-Scan Mode. The QADC64 provides external gating for queue 1 only. When external gated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, the queue execution automatically begins again from the beginning of the queue. Software initialization is not needed between trigger events. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

The purpose of external gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. To ensure consistent sample times in waveform digitizing, for example, the programmer must ensure that all CCWs have identical sample time settings in IST.

It is up to the programmer to ensure that the queue is large enough so that a maximum gate open time will not reach an end of queue. However it is useful to take advantage of a smaller queue in the manner described in the next paragraph.

In the event that the queue completes before the gate closes, a completion flag will be set and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the queue completes a second time, the trigger overrun flag will be set and the queue will roll-over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops and QADC64 sets the PF1 bit to indicate an incomplete queue. Software can read the CWPQ1 to determine the last valid conversion in the queue. In this mode, if the gate opens again execution of queue 1 begins again. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Continuous-Scan Mode. The QADC64 includes a dedicated periodic/interval timer for initiating a scan sequence on queue 1 and/or queue 2. Software selects a programmable timer interval ranging from 128 to 128 Kbytes times the QCLK period in binary multiples. The QCLK period is prescaled down from the intermodule bus (IMB) MCU clock.

When a periodic timer continuous-scan mode is selected for queue 1 and/or queue 2, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. Meanwhile, the QADC64 automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC64 waits for the periodic interval to expire again, then continues with the queue. Once end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in the queue.

The periodic timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause or queue completion, or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between trigger events.

Software enables the completion interrupt when using the periodic timer continuous-scan mode. When the interrupt occurs, the software knows that the periodically collected analog results have just been taken. The software can use the periodic interrupt to obtain non-analog inputs as well, such as contact closures, as part of a periodic look at all inputs.

13.10.4 QADC64 Clock (QCLK) Generation

Figure 13-8 is a block diagram of the clock subsystem. The QCLK provides the timing for the A/D converter state machine, which controls the timing of the conversion. The

QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (FQCLK) must be within a specified tolerance. See **APPENDIX G ELECTRICAL CHARACTERISTICS**.

Before using the QADC64, the software must initialize the prescaler with values that put the QCLK within the specified range. Though most software applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

NOTE

For software compatibility with earlier versions of QADC64, the definition of PSL, PSH, and PSA have been maintained. However, the requirements on minimum time and minimum low time no longer exist.

CAUTION

A change in the prescaler value while a conversion is in progress is likely to corrupt the result from any conversion in progress. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

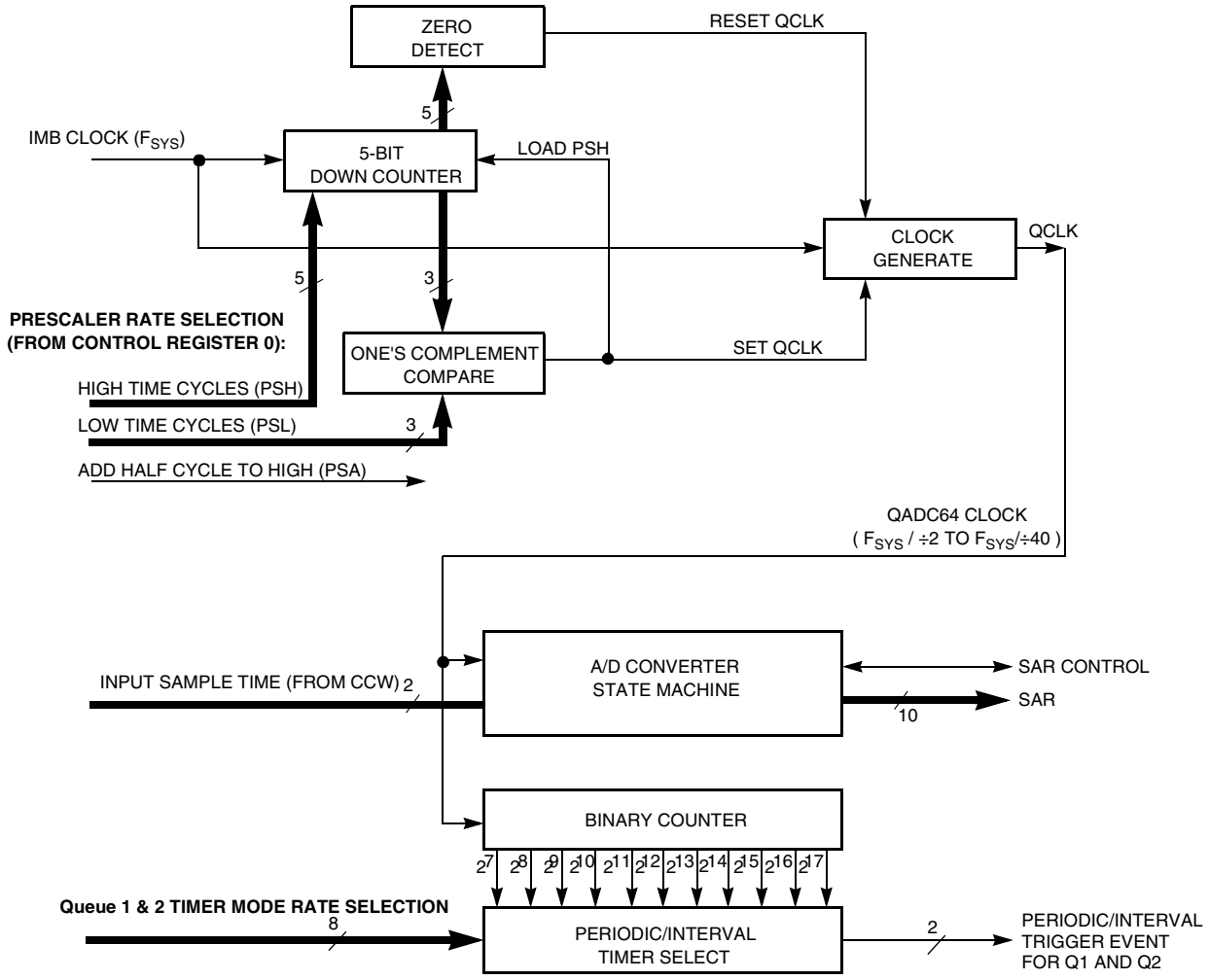


Figure 13-8 QADC64 Clock Subsystem Functions

To accommodate wide variations of the main MCU clock frequency (IMB clock — F_{SYS}), QCLK is generated by a programmable prescaler which divides the MCU IMB clock to a frequency within the specified QCLK tolerance range. To allow the A/D conversion time to be maximized across the spectrum of IMB clock frequencies, the QADC64 prescaler permits the frequency of QCLK to be software selectable. It also allows the duty cycle of the QCLK waveform to be programmable.

The software establishes the basic high phase of the QCLK waveform with the PSH (prescaler clock high time) field in QACR0, and selects the basic low phase of QCLK with the prescaler clock low time (PSL) field. The combination of the PSH and PSL parameters establishes the frequency of the QCLK.

NOTE

The guideline for selecting PSH and PSL is select is to maintain approximately 50% duty cycle. So for prescaler values less then 16, or PSH ~= PSL. For prescaler values greater than 16 keep PSL as large as possible.

Figure 13-8 shows that the prescaler is essentially a variable pulse width signal generator. A 5-bit down counter, clocked at the IMB clock rate, is used to create both the high phase and the low phase of the QCLK signal. At the beginning of the high phase, the 5-bit counter is loaded with the 5-bit PSH value. When the zero detector finds that the high phase is finished, the QCLK is reset. A 3-bit comparator looks for a one's complement match with the 3-bit PSL value, which is the end of the low phase of the QCLK. The PSA bit was maintained for software compatibility, but has no effect on QADC64.

The following equations define QCLK frequency:

$$\text{High QCLK Time} = (\text{PSH} + 1) \div F_{SYS}$$

$$\text{Low QCLK Time} = (\text{PSL} + 1) \div F_{SYS}$$

$$\text{FQCLK} = 1 \div (\text{High QCLK Time} + \text{Low QCLK Time})$$

Where:

- PSH = 0 to 31, the prescaler QCLK high cycles in QACR0
- PSL = 0 to 7, the prescaler QCLK low cycles in QACR0
- F_{SYS} = IMB clock frequency
- FQCLK = QCLK frequency

The following are equations for calculating the QCLK high/low phases in Example 1:

$$\text{High QCLK Time} = (11 + 1) \div 40 \times 10^6 = 300 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 40 \times 10^6 = 200 \text{ ns}$$

$$\text{FQCLK} = 1/(300 + 200) = 2 \text{ MHz}$$

The following are equations for calculating the QCLK high/low phases in Example 2:

$$\text{High QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$FQCLK = 1/(250 + 250) = 2\ MHz$$

Figure 13-9 and **Table 13-4** show examples of QCLK programmability. The examples include conversion times based on the following assumption:

- Input sample time is as fast as possible (IST = 0, 2 QCLK cycles).

Figure 13-9 and **Table 13-4** also show the conversion time calculated for a single conversion in a queue. For other MCU IMB clock frequencies and other input sample times, the same calculations can be made.

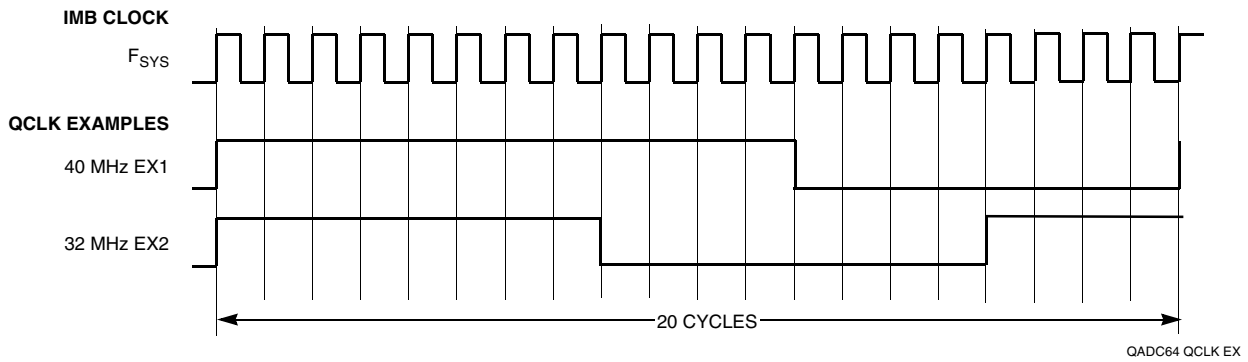


Figure 13-9 QADC64 Clock Programmability Examples

Table 13-4 QADC64 Clock Programmability

Control Register 0 Information					Input Sample Time (IST) =%00	
Example Number	Frequency	PSH	PSA	PSL	QCLK (MHz)	Conversion Time (µs)
1	40 MHz	11	0	7	2.0	7.0
2	32 MHz	7	0	7	2.0	7.0

NOTE

PSA is maintained for software compatibility but has no functional benefit to this version of the module.

The MCU IMB clock frequency is the basis of the QADC64 timing. The QADC64 requires that the IMB clock frequency be at least twice the QCLK frequency. The QCLK frequency is established by the combination of the PSH and PSL parameters in QACR0. The 5-bit PSH field selects the number of IMB clock cycles in the high phase of the QCLK wave. The 3-bit PSL field selects the number of IMB clock cycles in the low phase of the QCLK wave.

Example 1 in **Figure 13-9** shows that when PSH = 11, the QCLK remains high for twelve cycles of the IMB clock. It also shows that when PSL = 7, the QCLK remains low for eight IMB clock cycles. In Example 2, PSH = 7, the QCLK remains high for eight

cycles of the IMB clock. It also shows that when PSL = 7, the QCLK remains low for eight IMB clock cycles.

13.10.5 Periodic/Interval Timer

The on-chip periodic/interval timer is enabled to generate trigger events at a programmable interval, initiating execution of queue 1 and/or 2. The periodic/interval timer stays reset under the following conditions:

- Queue 1 and queue 2 are programmed to any queue operating mode which does not use the periodic/interval timer
- Interval timer single-scan mode is selected, but the single-scan enable bit is set to zero
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

Two other conditions which cause a pulsed reset of the timer are:

- Roll-over of the timer counter
- A queue operating mode change from one periodic/interval timer mode to another periodic/interval timer mode, depending on which queues are active in timer mode.

NOTE

The periodic/interval timer will not reset for a queue 2 operating mode change from one periodic/interval timer mode to another periodic/interval timer mode while queue 1 is in an active periodic/interval timer mode.

During the low power stop mode, the periodic/interval timer is held in reset. Since low power stop mode causes QACR1 and QACR2 to be reset to zero, a valid periodic or interval timer mode must be written after stop mode is exited to release the timer from reset.

When the IMB internal FREEZE line is asserted and a periodic or interval timer mode is selected, the timer counter is reset after the conversion in-progress completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, the freeze mode takes effect immediately, and the timer is held in reset. When the internal FREEZE line is negated, the timer counter starts counting from the beginning.

13.11 Interrupts

The QADC64 supports both polled and interrupt driven operation. Status bits in QASR reflect the operating condition of each queue and can optionally generate interrupts when enabled by the appropriate bits in QACR1 and/or QACR2.

Figure 13-10 displays the QADC64 interrupt flow.

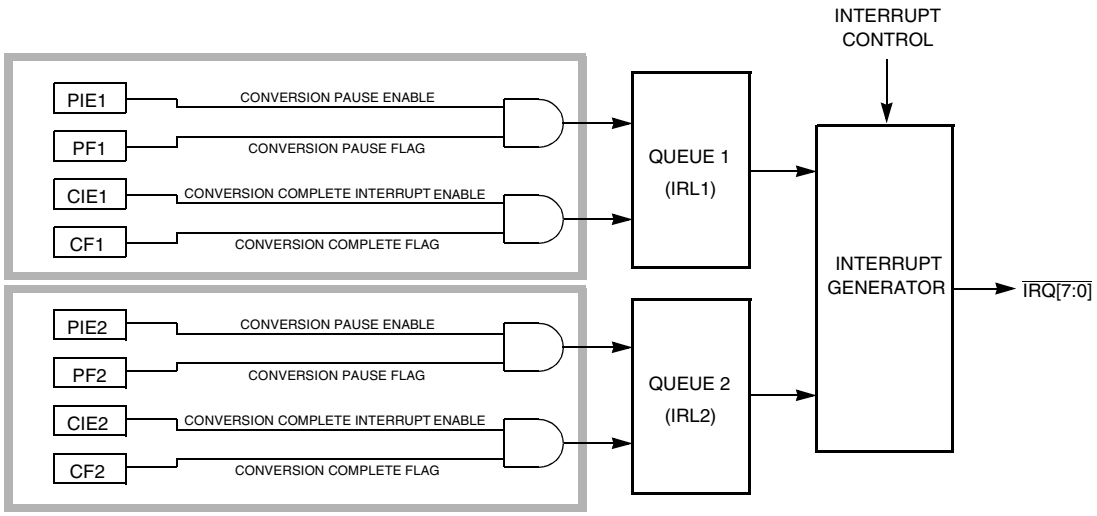


Figure 13-10 QADC64 Interrupt Flow Diagram

13.11.1 Interrupt Sources

The QADC64 has four interrupt service sources, each of which is separately enabled. Each time the result is written for the last CCW in a queue, the completion flag for the corresponding queue is set, and when enabled, an interrupt request is generated. In the same way, each time the result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated.

Table 13-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity.

Table 13-5 QADC64 Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written for the last CCW in queue 1	CF1	CIE1
	Result written for a CCW with pause bit set in queue 1	PF1	PIE1
Queue 2	Result written for the last CCW in queue 2	CF2	CIE2
	Result written for a CCW with pause bit set in queue 2	PF2	PIE2

Both polled and interrupt-driven QADC64 operations require that status flags must be cleared after an event occurs. Flags are cleared by first reading QASR with the appropriate flag bits set to one, then writing zeros to the flags that are to be cleared. A flag can be cleared only if the flag was a logic one at the time the register was read by the CPU. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

13.11.2 Interrupt Register

The QADC64 interrupt register QADC64INT specifies the priority level of QADC64 interrupt requests

The values contained in the IRL1 and IRL2 fields in QADC64INT determine the priority of QADC64 interrupt service requests.The interrupt levels for queue 1 and queue 2 may be different.

13.11.3 Interrupt Levels and Time Multiplexing

The QADC64 conditionally generates interrupts to the bus master via the IMB IRQ signals. When the QADC64 sets a status bit assigned to generate an interrupt, the QADC64 drives the IRQ bus. The value driven onto IRQ[7:0] represents the interrupt level assigned to the interrupt source. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 13-11** displays the interrupt levels on IRQ with ILBS.

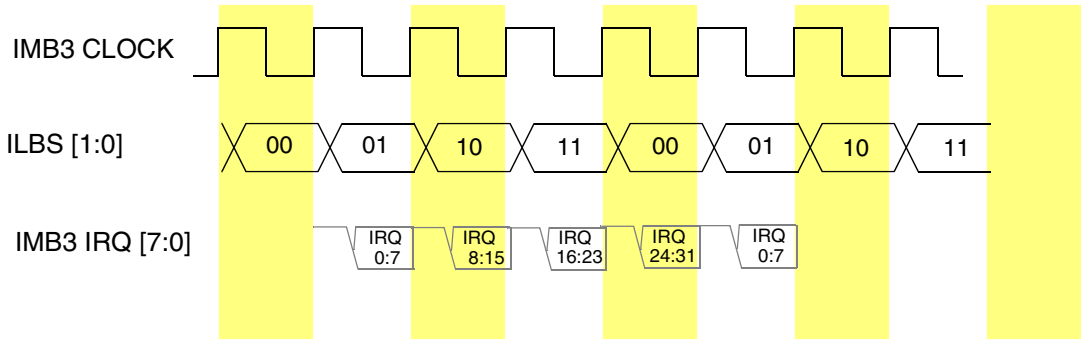


Figure 13-11 Interrupt Levels on IRQ with ILBS

13.12 Programming Model

Each QADC64 occupies 1 Kbyte (512 16-bit entries) of address space. The address space consists of ten 16-bit control, status, and port registers; 64 16-bit entries in the CCW table; and 64 16-bit entries in the result table. The result table occupies 192 16-bit address locations because the result data is readable in three data alignment formats.

Table 13-6 shows the QADC64 memory map. The lowercase “x” appended to each register name represents “A” or “B” for the QADC64_A or QADC64_B module, respectively. The address offset shown is from the base address of the module. Refer to **1.3 MPC555 / MPC556 Address Map** to locate each QADC64 module in the MPC555 / MPC556 memory map.

Table 13-6 QADC64 Address Map

Access	Address	MSB 0	LSB 15
S[1]	0x30 4800 0x30 4C00	QADC64 Module Configuration Register (QADC64MCR_x) See **Table 13-7** for bit descriptions.	
T[2]	0x30 4802 0x30 4C02	QADC64 Test Register (QADC64TEST_x)	
S	0x30 4804 0x30 4C04	Interrupt Register (QADC64INT_x) See **Table 13-8** for bit descriptions.	
S/U[3]	0x30 4806 0x30 4C06	Port A Data (PORTQA_x) See **Table 13-10** for bit descriptions.	Port B Data (PORTQB_x)
S/U	0x30 4808 0x30 4C08	Port A Data Direction Register (DDRQA_x) See **Table 13-10** for bit descriptions.	
S/U	0x30 480A 0x30 4C0A	QADC64 Control Register 0 (QACR0_x) See **Table 13-11** for bit descriptions.	
S/U	0x30 480C 0x30 4C0C	QADC64 Control Register 1 (QACR1_x) See **Table 13-12** for bit descriptions.	
S/U	0x30 480E 0x30 4C0E	QADC64 Control Register 2 (QACR2_x) See **Table 13-14** for bit descriptions.	
S/U	0x30 4810, 0x30 4C10	QADC64 Status Register 0 (QASR0_x) See **Table 13-16** for bit descriptions.	
S/U	0x30 4812, 0x30 4C12	QADC64 Status Register 1 (QASR1_x) See **Table 13-18** for bit descriptions.	
---	0x30 4814 – 0x30 49FE 0x30 4C14 – 0x30 4DFE	Reserved	
S/U	0x30 4A00 – 0x30 4A7E 0x30 4E00 – 0x30 4E7E	Conversion Command Word (CCW_x) Table See **Table 13-19** for bit descriptions.	
S/U	0x30 4A80 – 0x30 4AFE 0x30 4E80 – 0x30 4EFE	Result Word Table Right-Justified, Unsigned Result Register (RJURR_x) See **13.12.12** for bit descriptions.	
S/U	0x30 4B00 – 0x30 4B7E 0x30 4F00 – 0x30 4F7E	Result Word Table Left-Justified, Signed Result Register (LJSRR_x) See **13.12.12** for bit descriptions.	
S/U	0x30 4B80 – 0x30 4BFE 0x30 4F80 – 0x30 4FFE	Result Word Table Left-Justified, Unsigned Result Register (LJURR_x) See **13.12.12** for bit descriptions.	

NOTES:
1. S = Supervisor only
2. Access is restricted to supervisor only and factory test mode only.
3. S/U = Unrestricted or supervisor depending on the state of the SUPV bit in the QADC64MCR.

The QADC64 has three global registers for configuring module operation: the module configuration register (QADC64MCR), the interrupt register (QADC64INT), and a test register (QADC64TEST). The global registers are always defined to be in supervisor data space. The CPU allows software to establish the global registers in supervisor data space and the remaining registers and tables in user space.

All QADC64 analog channel/port pins that are not used for analog input channels can be used as digital port pins. Port values are read/written by accessing the port A and B data registers (PORTQA and PORTQB). Port A pins are specified as inputs or outputs by programming the port data direction register (DDRQA). Port B is an input-only port.

13.12.1 QADC64 Module Configuration Register

QADC64MCR — QADC64 Module Configuration Register **0x30 4800**
0x30 4C00

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	RESERVED						SUPV	RESERVED						
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 13-7 QADC64MCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. When the STOP bit is set, the clock signal to the QADC64 is disabled, effectively turning off the analog circuitry. 0 = Enable QADC64 clock 1 = Disable QADC64 clock
1	FRZ	FREEZE assertion response. The FRZ bit determines whether or not the QADC64 responds to assertion of the IMB3 FREEZE signal. 0 = QADC64 ignores the IMB3 FREEZE signal 1 = QADC64 finishes any current conversion, then freezes
2:7	—	Reserved
8	SUPV	Supervisor/unrestricted data space. The SUPV bit designates the assignable space as supervisor or unrestricted. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted 1 = All QADC64 registers and tables are designated as supervisor-only data space
9:15	—	Reserved

13.12.2 QADC64 Test Register

QADC64TEST — QADC64 Test Register **0x30 4802, 0x30 4C02**

Used for factory test only.

13.12.3 QADC64 Interrupt Register

QADC64INT — QADC64 Interrupt Register **0x30 4804**
0x30 4C04

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRL1					IRL2					RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-8 QADC64INT Bit Descriptions

Bit(s)	Name	Description
0:4	IRL1	Interrupt level for queue 1. A value of 0b00000 provides an interrupt level of 0; 0b11111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
5:9	IRL2	Interrupt level for queue 2. A value of 0b00000 provides an interrupt level of 0; 0b11111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
10:15	—	Reserved

13.12.4 Port A/B Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB).

PORTQA — Port QA Data Register **0x30 4806**
PORTQB — Port QB Data Register **0x30 4C06**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN3	AN2	AN1	AN0
MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

Table 13-9 PORTQA, PORTQB Bit Descriptions

Bit(s)	Name	Description
0:7	PQA[0:7]	Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs (AN[59:52]), and external multiplexer address outputs (MA[2:0]).
8:15	PQB[0:7]	Port B pins are referred to as PQB when used as an 8-bit input-only port. Port B can also be used for non-multiplexed (AN[51:48])/AN[3:0]) and multiplexed (ANz, ANy, ANx, ANw) analog inputs.

13.12.5 Port Data Direction Register

DDRQA — Port QA Data Direction Register **0x30 4808**
0x30 4C08

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
DDQA7	DDQA 6	DDQA 5	DDQA 4	DDQA 3	DDQA 2	DDQA 1	DDQA 0	RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-10 DDRQA Bit Descriptions

Bit(s)	Name	Description
0:7	DDQA[7:0]	Bits in this register control the direction of the port QA pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

13.12.6 QADC64 Control Register 0 (QACR0)

Control register zero establishes the QCLK with prescaler parameter fields and defines whether external multiplexing is enabled. All of the implemented control register fields can be read or written, reserved fields read zero and writes have no effect. They are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR0 — QADC64 Control Register 0 **0x30 480A**
0x30 4C0A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MUX	RESERVED		TRG	RESERVED			PSH					PSA	PSL		
RESET:															
0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1

Table 13-11 QACR0 Bit Descriptions

Bit(s)	Name	Description
0	MUX	Externally multiplexed mode. The MUX bit configures the QADC64 for externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[2:0] pins to be outputs. 0 = Internally multiplexed, 16 possible channels 1 = Externally multiplexed, 41 possible channels
1:2	—	Reserved
3	TRG	Trigger assignment. TRG allows the software to assign the ETRIG[2:1] pins to queue 1 and queue 2. 0 = ETRIG1 triggers queue 1; ETRIG2 triggers queue 2 1 = ETRIG1 triggers queue 2; ETRIG2 triggers queue 1
4:6	—	Reserved
7:11	PSH	Prescaler clock high time. The PSH field selects the QCLK high time in the prescaler. PSH value plus 1 represents the high time in IMB clocks
12	PSA	Note that this bit location is maintained for software compatibility with previous versions of the QADC64. It serves no functional benefit in the MPC555 / MPC556 and is not operational.
13:15	PSL	Prescaler clock low time. The PSL field selects the QCLK low time in the prescaler. PSL value plus 1 represents the low time in IMB clocks

13.12.7 QADC64 Control Register 1 (QACR1)

Control register 1 is the mode control register for the operation of queue 1. The applications software defines the queue operating mode for the queue, and may enable a completion and/or pause interrupt. All of the control register fields are read/write data. However, the SSE1 bit always reads as zero unless the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR1 — Control Register 1 **0x30 480C**
0x30 4C0C

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE1	PIE1	SSE1	MQ1					RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-12 QACR1 Bit Descriptions

Bit(s)	Name	Description
0	CIE1	Queue 1 completion interrupt enable. CIE1 enables completion interrupts for queue 1. The interrupt request is generated when the conversion is complete for the last CCW in queue 1. 0 = Queue 1 completion interrupts disabled 1 = Generate an interrupt request after completing the last CCW in queue 1
1	PIE1	Queue 1 pause interrupt enable. PIE1 enables pause interrupts for queue 1. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 1 pause interrupts disabled 1 = Generate an interrupt request after completing a CCW in queue 1 which has the pause bit set
2	SSE1	Queue 1 single-scan enable. SSE1 enables a single-scan of queue 1 after a trigger event occurs. The SSE1 bit may be set to a one during the same write cycle that sets the MQ1 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE1 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 1. The QADC64 clears SSE1 when the single-scan is complete.
3:7	MQ1	Queue 1 operating mode. The MQ1 field selects the queue operating mode for queue 1. **Table 13-13** shows the different queue 1 operating modes.
8:15	—	Reserved

Table 13-13 Queue 1 Operating Modes

MQ1	Operating Modes
0b00000	Disabled mode, conversions do not occur
0b00001	Software triggered single-scan mode (started with SSE1)
0b00010	External trigger rising edge single-scan mode
0b00011	External trigger falling edge single-scan mode
0b00100	Interval timer single-scan mode: time = QCLK period x 2^7
0b00101	Interval timer single-scan mode: time = QCLK period x 2^8
0b00110	Interval timer single-scan mode: time = QCLK period x 2^9
0b00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
0b01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
0b01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
0b01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
0b01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
0b01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
0b01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
0b01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
0b01111	External gated single-scan mode (started with SSE1)
0b10000	Reserved mode
0b10001	Software triggered continuous-scan mode
0b10010	External trigger rising edge continuous-scan mode
0b10011	External trigger falling edge continuous-scan mode
0b10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
0b10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
0b10110	Periodic timer continuous-scan mode: time = QCLK period x 2^9
0b10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
0b11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
0b11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
0b11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
0b11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
0b11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
0b11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
0b11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
0b11111	External gated continuous-scan mode

13.12.8 QADC64 Control Register 2 (QACR2)

Control register two is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit,

which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR2 — Control Register 2 **0x30 480E**
0x30 4C0E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE2	PIE2	SSE2	MQ2					RE-SUME	BQ2						
RESET:															
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 13-14 QACR2 Bit Descriptions

Bit(s)	Name	Description
0	CIE2	Queue 2 completion interrupt enable. CIE2 enables completion interrupts for queue 2. The interrupt request is generated when the conversion is complete for the last CCW in queue 2. 0 = Queue 2 completion interrupts disabled. 1 = Generate an interrupt request after completing the last CCW in queue 2.
1	PIE2	Queue 2 pause interrupt enable. PIE2 enables pause interrupts for queue 2. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 2 pause interrupts disabled. 1 = Generate an interrupt request after completing a CCW in queue 2 which has the pause bit set.
2	SSE2	Queue 2 single-scan enable bit. SSE2 enables a single-scan of queue 2 after a trigger event occurs. The SSE2 bit may be set to a one during the same write cycle that sets the MQ2 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE2 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 2. The QADC64 clears SSE2 when the single-scan is complete.
3:7	MQ2	Queue 2 operating mode. The MQ2 field selects the queue operating mode for queue 2. **Table 13-15** shows the bits in the MQ2 field which enable different queue 2 operating modes.
8	RESUME	Queue 2 resume. RESUME selects the resumption point after queue 2 is suspended by queue 1. If RESUME is changed during execution of queue 2, the change is not recognized until an end-of-queue condition is reached, or the queue operating mode of queue 2 is changed. 0 = After suspension, begin execution with the first CCW in queue 2 or the current sub-queue. 1 = After suspension, begin execution with the aborted CCW in queue 2.
9:15	BQ2	Beginning of queue 2. The BQ2 field indicates the location in the CCW table where queue 2 begins. The BQ2 field also indicates the end of queue 1 and thus creates an end-of-queue condition for queue 1. Setting BQ2 to any value ≥ 64 (0b1000000) allows the entire RAM space for queue 1 CCWs.

Table 13-15 Queue 2 Operating Modes

MQ2	Operating Modes
0b00000	Disabled mode, conversions do not occur
0b00001	Software triggered single-scan mode (started with SSE2)
0b00010	External trigger rising edge single-scan mode
0b00011	External trigger falling edge single-scan mode
0b00100	Interval timer single-scan mode: interval = QCLK period x 2^7
0b00101	Interval timer single-scan mode: interval = QCLK period x 2^8
0b00110	Interval timer single-scan mode: interval = QCLK period x 2^9
0b00111	Interval timer single-scan mode: interval = QCLK period x 2^{10}
0b01000	Interval timer single-scan mode: interval = QCLK period x 2^{11}
0b01001	Interval timer single-scan mode: interval = QCLK period x 2^{12}
0b01010	Interval timer single-scan mode: interval = QCLK period x 2^{13}
0b01011	Interval timer single-scan mode: interval = QCLK period x 2^{14}
0b01100	Interval timer single-scan mode: interval = QCLK period x 2^{15}
0b01101	Interval timer single-scan mode: interval = QCLK period x 2^{16}
0b01110	Interval timer single-scan mode: interval = QCLK period x 2^{17}
0b01111	Reserved mode
0b10000	Reserved mode
0b10001	Software triggered continuous-scan mode (started with SSE2)
0b10010	External trigger rising edge continuous-scan mode
0b10011	External trigger falling edge continuous-scan mode
0b10100	Periodic timer continuous-scan mode: period = QCLK period x 2^7
0b10101	Periodic timer continuous-scan mode: period = QCLK period x 2^8
0b10110	Periodic timer continuous-scan mode: period = QCLK period x 2^9
0b10111	Periodic timer continuous-scan mode: period = QCLK period x 2^{10}
0b11000	Periodic timer continuous-scan mode: period = QCLK period x 2^{11}
0b11001	Periodic timer continuous-scan mode: period = QCLK period x 2^{12}
0b11010	Periodic timer continuous-scan mode: period = QCLK period x 2^{13}
0b11011	Periodic timer continuous-scan mode: period = QCLK period x 2^{14}
0b11100	Periodic timer continuous-scan mode: period = QCLK period x 2^{15}
0b11101	Periodic timer continuous-scan mode: period = QCLK period x 2^{16}
0b11110	Periodic timer continuous-scan mode: period = QCLK period x 2^{17}
0b11111	Reserved mode

13.12.9 QADC64 Status Register 0 (QASR0)

QASR0 contains information about the state of each queue and the current A/D conversion. Except for the four flag bits (CF1, PF1, CF2, and PF2) and the two trigger overrun bits (TOR1 and TOR2), all of the status register fields contain read-only data.

The four flag bits and the two trigger overrun bits are cleared by writing a zero to the bit after the bit was previously read as a one.

QASR0 — QADC64 Status Register **0x30 4810**
0x30 4C10

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-16 QASR0 Bit Descriptions

Bit(s)	Name	Description
0	CF1	Queue 1 completion flag. CF1 indicates that a queue 1 scan has been completed. CF1 is set by the QADC64 when the conversion is complete for the last CCW in queue 1, and the result is stored in the result table. 0 = Queue 1 scan is not complete 1 = Queue 1 scan is complete
1	PF1	Queue 1 pause flag. PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC64 when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 1 has not reached a pause 1 = Queue 1 has reached a pause
2	CF2	Queue 2 completion flag. CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC64 when the conversion is complete for the last CCW in queue 2, and the result is stored in the result table. 0 = Queue 2 scan is not complete 1 = Queue 2 scan is complete
3	PF2	Queue 2 pause flag. PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC64 when the current queue 2 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 2 has not reached a pause 1 = Queue 2 has reached a pause
4	TOR1	— Queue 1 trigger overrun. TOR1 indicates that an unexpected queue 1 trigger event has occurred. TOR1 can be set only while queue 1 is active. A trigger event generated by a transition on ETRIG1/ETRIG2 may be recorded as a trigger overrun. TOR1 can only be set when using an external trigger mode. TOR1 cannot occur when the software initiated single-scan mode or the software initiated continuous-scan mode is selected. 0 = No unexpected queue 1 trigger events have occurred 1 = At least one unexpected queue 1 trigger event has occurred
5	TOR2	Queue 2 trigger overrun. TOR2 indicates that an unexpected queue 2 trigger event has occurred. TOR2 can be set when queue 2 is in the active, suspended, and trigger pending states. A trigger event generated by a transition depending on the value of TRG in QACR or ETRIG1/ETRIG2 or by the periodic/interval timer may be recorded as a trigger overrun. TOR2 can only be set when using an external trigger mode or a periodic/interval timer mode. Trigger overruns cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected. 0 = No unexpected queue 2 trigger events have occurred 1 = At least one unexpected queue 2 trigger event has occurred

Table 13-16 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
6:9	QS	Queue status. This 4-bit read-only field indicates the current condition of queue 1 and queue 2. QS[0:1] are associated with queue 1, and QS[2:3] are associated with queue 2. Since the queue priority scheme interlinks the operation of queue 1 and queue 2, the status bits should be considered as one 4-bit field. Table 13-17 shows the bit encodings of the QS field.
10:15	CWP	Command word pointer. CWP indicates which CCW is executing at present, or was last completed. The CWP is a read-only field; writes to it have no effect.

Table 13-17 Queue Status

QS	Description
0b0000	Queue 1 idle, queue 2 idle
0b0001	Queue 1 idle, queue 2 paused
0b0010	Queue 1 idle, queue 2 active
0b0011	Queue 1 idle, queue 2 trigger pending
0b0100	Queue 1 paused, queue 2 idle
0b0101	Queue 1 paused, queue 2 paused
0b0110	Queue 1 paused, queue 2 active
0b0111	Queue 1 paused, queue 2 trigger pending
0b1000	Queue 1 active, queue 2 idle
0b1001	Queue 1 active, queue 2 paused
0b1010	Queue 1 active, queue 2 suspended
0b1011	Queue 1 active, queue 2 trigger pending
0b1100	Reserved
0b1101	Reserved
0b1110	Reserved
0b1111	Reserved

13.12.10 QADC64 Status Register 1 (QASR1)

The QASR1 contains two fields: command word pointers for queue 1 and queue 2.

QASR1 — Status Register1 **0x30 4812**
0x30 4C12

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED		CWPQ1						RESERVED		CWPQ2					
RESET:															
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Table 13-18 QASR0 Bit Descriptions

Bit(s)	Name	Description
0:1	—	Reserved
2:7	CWPQ1	Command word pointer for queue 1. This field is a software read-only field, and write operations have no effect. CWPQ1 allows software to read the last executed CCW in queue 1, regardless which queue is active. The CWPQ1 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ1 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 1, both the result register is written and the CWPQ1 are updated. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, CWP points to BQ2 while CWPQ1 points to the last CCW queue 1. During the stop mode, the CWPQ1 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWPQ1 is unchanged; it points to the last executed CCW in queue 1.
8:9	—	Reserved
10:15	CWPQ2	Command word pointer for queue 2. This field is a software read-only field, and write operations have no effect. CWPQ2 allows software to read the last executed CCW in queue 2, regardless which queue is active. The CWPQ2 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ2 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 2, both the result register is written and the CWPQ2 are updated. During the stop mode, the CWPQ2 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW in queue 2.

13.12.11 Conversion Command Word Table

The CCW table is a RAM, 64 words long and 10 bits wide, which can be programmed by the software to request conversions of one or more analog input channels. The entries in the CCW table are 10-bit conversion command words. The CCW table is written by software and is not modified by the QADC64. Each CCW requests the conversion of an analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW.

The ten implemented bits of the CCW word are read/write data. They may be written when the software initializes the QADC64. Unimplemented bits are read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry. The QADC64 provides 64 CCW table entries.

The beginning of queue 1 is always the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, software must do the following:

- Program queue 2 to be in the disabled mode, and
- Program the beginning of BQ2 to ≥ 64.

To dedicate the entire CCW table to queue 2, software must do the following:

- Program queue 1 to be in the disabled mode
- Program BQ2 to be the first location in the CCW table.

Figure 13-12 illustrates the operation of the queue structure.

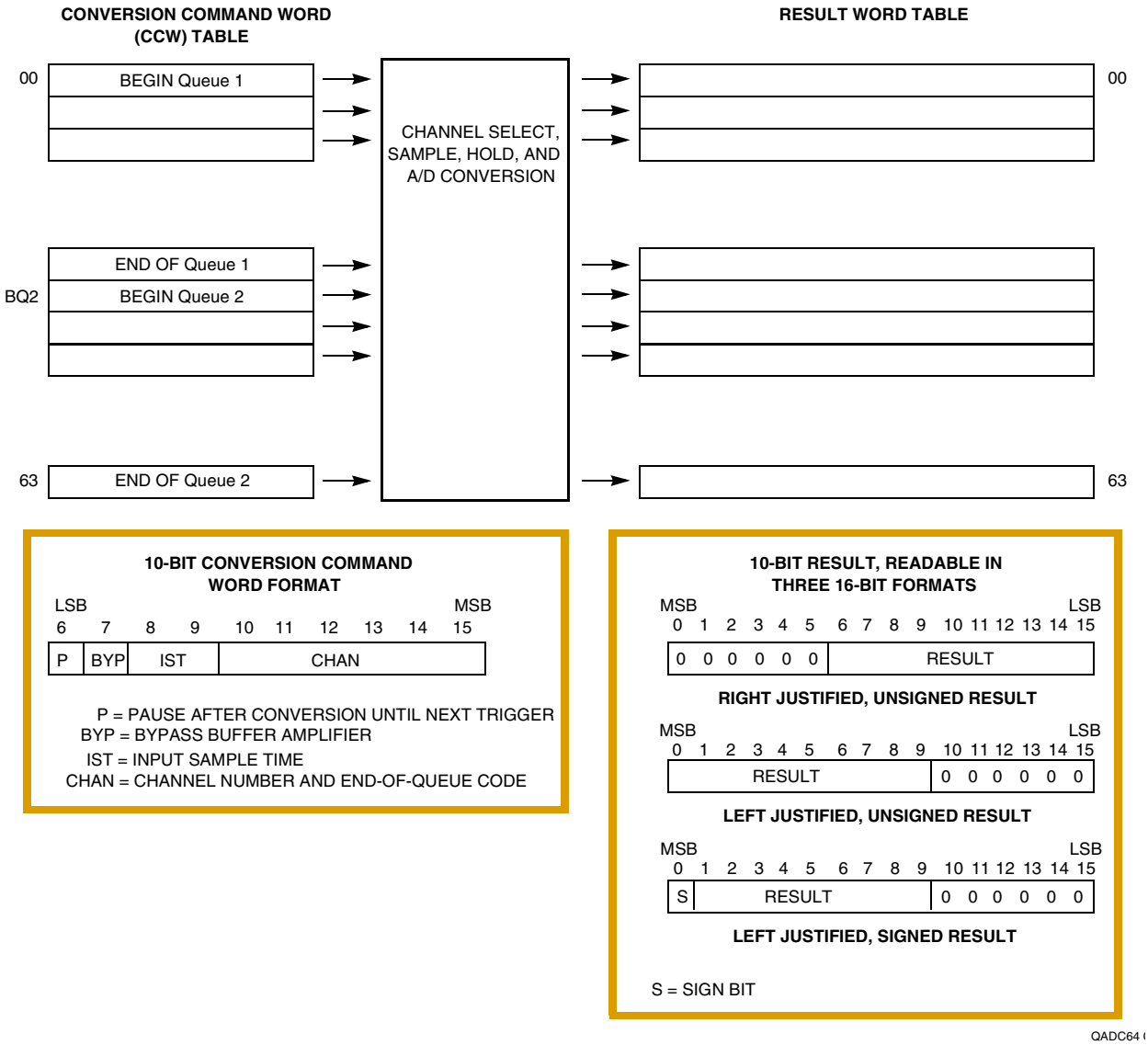


Figure 13-12 QADC64 Conversion Queue Operation

To prepare the QADC64 for a scan sequence, the software writes to the CCW table to specify the desired channel conversions. The software also establishes the criteria for initiating the queue execution by programming the queue operating mode. The queue operating mode determines what type of trigger event causes queue execution to begin. “Trigger event” refers to any of the ways to cause the QADC64 to begin executing the CCWs in a queue or sub-queue. An external trigger is only one of the possible trigger events.

A scan sequence may be initiated by the following:

- A software command

- Expiration of the periodic/interval timer
- External trigger signal
- External gated signal (queue 1 only)

The software also specifies whether the QADC64 is to perform a single pass through the queue or is to scan continuously. When a single-scan mode is selected, the software selects the queue operating mode and sets the single-scan enable bit. When a continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC64 completes each queue scan sequence.

During queue execution, the QADC64 reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample
- Resolution

During initial sample, a buffered version of the selected input channel is connected to the sample capacitor at the output of the sample buffer amplifier.

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of 2, 4, 8, or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC64 continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC64 stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, which may cause an interrupt to notify the software that the queue has reached the pause state. After the trigger event occurs, the paused state ends and the QADC64 continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

The following indicate the end-of-queue condition:

- The CCW channel field is programmed with 63 (0x3F) to specify the end of the queue
- The end of queue 1 is implied by the beginning of queue 2, which is specified in the BQ2 field in QACR2
- The physical end of the queue RAM space defines the end of either queue

When any of the end-of-queue conditions are recognized, a queue completion flag is set, and if enabled, an interrupt is issued to the software. The following situations prematurely terminate queue execution:

- Since queue 1 is higher in priority than queue 2, when a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and the queue 1 execution begins. When queue 1 execution is completed, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 sub-queue being ex-

ecuted when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit in QACR2 allows the software to select where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 queue and sub-queue CCWs, all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger event rate for queue 1 can prohibit the completion of queue 2. If this occurs, the software may choose to begin execution of queue 2 with the aborted CCW entry.

- Software can change the queue operating mode to disabled mode. Any conversion in progress for that queue is aborted. Putting a queue into the disabled mode does not power down the converter.
- Software can change the queue operating mode to another valid mode. Any conversion in progress for that queue is aborted. The queue restarts at the beginning of the queue, once an appropriate trigger event occurs.
- For low power operation, software can set the stop mode bit to prepare the module for a loss of clocks. The QADC64 aborts any conversion in progress when the stop mode is entered.
- When the freeze enable bit is set by software and the IMB internal FREEZE line is asserted, the QADC64 freezes at the end of the conversion in progress. When internal FREEZE is negated, the QADC64 resumes queue execution beginning with the next CCW entry.

CCW — Conversion Command Word Table **0x30 4A00 – 0x30 4A7E**
0x30 4E00 – 0x30 4E7E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						P	BYP	IST		CHAN					
RESET:															
0	0	0	0	0	0	U	U	U	U	U	U	U	U	U	U

Table 13-19 CCW Bit Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	P	Pause. The pause bit allows the creation of sub-queues within queue 1 and queue 2. The QADC64 performs the conversion specified by the CCW with the pause bit set, and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW. 0 = Do not enter the pause state after execution of the current CCW. 1 = Enter the pause state after execution of the current CCW.
7	BYP	Sample amplifier bypass. Setting BYP enables the amplifier bypass mode for a conversion, and subsequently changes the timing. Refer to **13.9.1.1 Amplifier Bypass Mode Conversion Timing** for more information. 0 = Amplifier bypass mode disabled. 1 = Amplifier bypass mode enabled.
8:9	IST	Input sample time. The IST field specifies the length of the sample window. Longer sample times permit more accurate A/D conversions of signals with higher source impedances, especially if BYP = 1. 00 = QCKL period x 2 01 = QCKL period x 4 10 = QCKL period x 8 11 = QCKL period x 16
10:15	CHAN	Channel number. The CHAN field selects the input channel number corresponding to the analog input pin to be sampled and converted. The analog input pin channel number assignments and the pin definitions vary depending on whether the QADC64 is operating in multiplexed or non-multiplexed mode. The queue scan mechanism sees no distinction between an internally or externally multiplexed analog input. If CHAN specifies a reserved channel number (channels 32 to 47) or an invalid channel number (channels 4 to 31 in non-multiplexed mode), the low reference level (VRL) is converted. Programming the channel field to channel 63 indicates the end of the queue. Channels 60 to 62 are special internal channels. When one of these channels is selected, the sample amplifier is not used. The value of VRL, VRH, or $(V_{RH} - V_{RL})/2$ is placed directly into the converter. Programming the input sample time to any value other than two for one of the internal channels has no benefit except to lengthen the overall conversion time. **Table 13-20** shows the channel number assignments for the non-multiplexed mode. **Table 13-21** shows the channel number assignments for the multiplexed mode.

Table 13-20 Non-Multiplexed Channel Assignments and Pin Designations

Non-multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type (I/O)**	**Binary**	**Decimal**
PQB0	AN0	—	I	000000	0
PQB1	AN1	—	I	000001	1
PQB2	AN2	—	I	000010	2
PQB3	AN3	—	I	000011	3
—	—	Invalid	—	000100 to 011111	4 to 31
—	—	Reserved	—	10XXXX	32 to 47
PQB4	AN48	—	I	110000	48
PQB5	AN49	—	I	110001	49
PQB6	AN50	—	I	110010	50
PQB7	AN51	—	I	110011	51
PQA0	AN52	—	I/O	110100	52
PQA1	AN53	—	I/O	110101	53
PQA2	AN54	—	I/O	110110	54
PQA3	AN55	—	I/O	110111	55
PQA4	AN56	—	I/O	111000	56
PQA5	AN57	—	I/O	111001	57
PQA6	AN58	—	I/O	111010	58
PQA7	AN59	—	I/O	111011	59
—	V_{RL}	—	I	111100	60
—	V_{RH}	—	I	111101	61
—	—	$(V_{RH} - V_{RL})/2$	—	111110	62
—	—	End of Queue Code	—	111111	63

Table 13-21 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type (I/O)**	**Binary**	**Decimal**
PQB0	ANw	—	I	00xxx0	0 to 14 even
PQB1	ANx	—	I	00xxx1	1 to 15 odd
PQB2	ANy	—	I	01xxx0	16 to 30 even
PQB3	ANz	—	I	01xxx1	17 to 31 odd
—	—	Reserved	—	10xxxx	32 to 47
PQB4	AN48	—	I	110000	48
PQB5	AN49	—	I	110001	49
PQB6	AN50	—	I	110010	50
PQB7	AN51	—	I	110011	51
PQA0	—	MA0	I/O	110100	52
PQA1	—	MA1	I/O	110101	53
PQA2	—	MA2	I/O	110110	54
PQA3	AN55	—	I/O	110111	55
PQA4	AN56	—	I/O	111000	56
PQA5	AN57	—	I/O	111001	57
PQA6	AN58	—	I/O	111010	58
PQA7	AN59	—	I/O	111011	59
—	V_{RL}	—	I	111100	60
—	V_{RH}	—	I	111101	61
—	—	(VRH -VRL)/2	—	111110	62
—	—	End of Queue Code	—	111111	63

13.12.12 Result Word Table

The result word table is a 64-word long, 10-bit wide RAM. The QADC64 writes a result word after completing an analog conversion specified by the corresponding CCW. The result word table can be read or written, but in normal operation, software reads the result word table to obtain analog conversions from the QADC64. Unimplemented bits are read as zeros, and write operations have no effect.

While there is only one result word table, the data can be accessed in three different alignment formats:

- Right-justified, with zeros in the higher order unused bits.
- Left-justified, with the most significant bit inverted to form a sign bit, and zeros in the unused lower order bits.
- Left-justified, with zeros in the unused lower order bits.

The left-justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The data is routed onto the IMB according to the selected format. The address used to access the table determines the data alignment format. All write operations to the result word table are right-justified.

RJURR — Right-Justified, Unsigned Result Register **0x30 4A80 – 0x30 4AFE**
0x30 4E80 – 0x30 4EFE

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						RESULT									
RESET:															
0	0	0	0	0	0										

The conversion result is unsigned, right-justified data. Unused bits return zero when read.

LJSRR — Left-Justified, Signed Result Register **0x30 4B00 – 0x30 4B7E**
0x30 4F00 – 0x30 4F7E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
S[1]	RESULT									RESERVED					
RESET:															
										0	0	0	0	0	0

NOTES:
1. S = Sign bit.

The conversion result is signed, left-justified data. Unused bits return zero when read.

LJURR — Left-Justified, Unsigned Result Register **0x30 4B80 – 0x30 4BFE**
0x30 4F80 – 0x30 4FFE

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESULT										RESERVED					
RESET:															
										0	0	0	0	0	0

The conversion result is unsigned, left-justified data. Unused bits return zero when read.

SECTION 14
QUEUED SERIAL MULTI-CHANNEL MODULE

14.1 Overview

The queued serial multi-channel module (QSMCM) provides three serial communication interfaces: the queued serial peripheral interface (QSPI) and two serial communications interfaces (SCI1 and SCI2). These submodules communicate with the CPU via a common slave bus interface unit (SBIU).

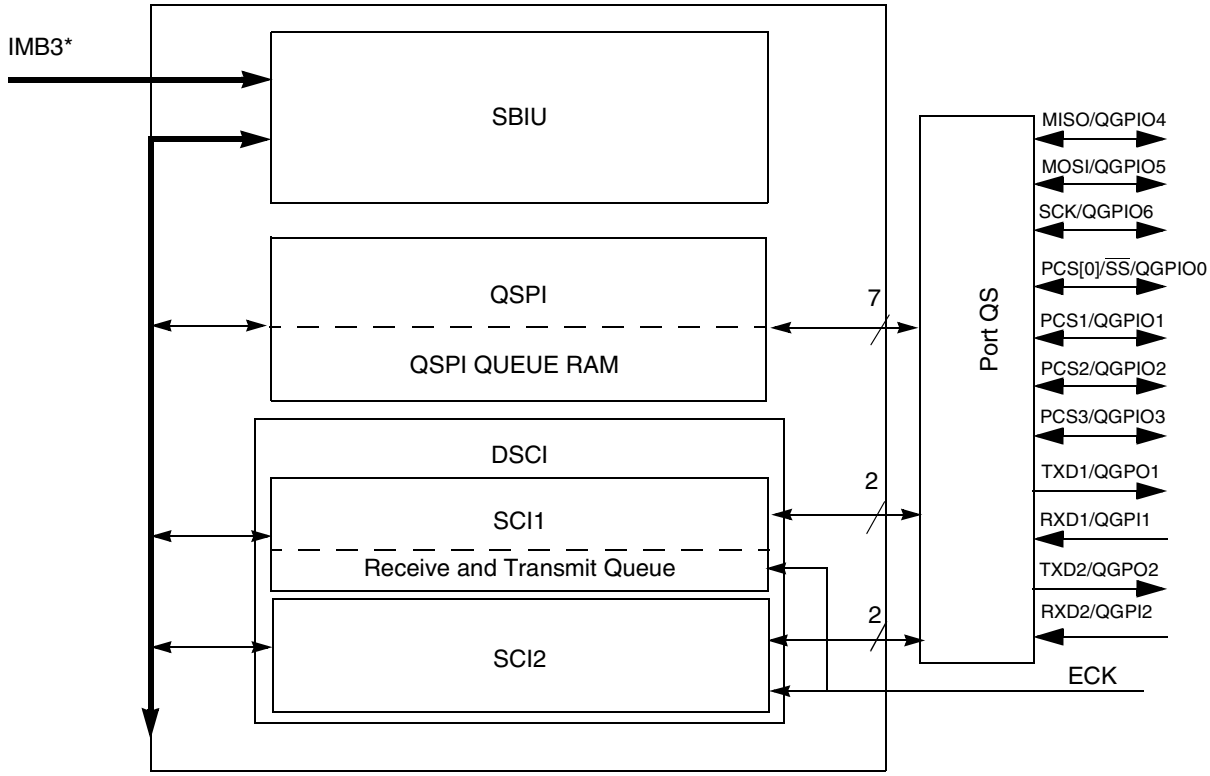
The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced from the original SPI in the QSMCM (queued serial module) to include a total of 160 bytes of queue RAM to accommodate more receive, transmit, and control information. The QSPI is fully compatible with the SPI systems found on other Motorola devices.

The dual, independent SCIs are used to communicate with external devices and other MCUs via an asynchronous serial bus. Each SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. The original QSMCM SCI is enhanced by the addition of an SCI and a common external baud clock source.

The SCI1 has the ability to use the resultant baud clock from SCI2 as the input clock source for the SCI1 baud rate generator. Also, the SCI1 has an additional mode of operation that allows queuing of transmit and receive data frames. If the queue feature is enabled, a set of 16 entry queues is allocated for the receive and/or transmit operation.

14.2 Block Diagram

Figure 14-1 depicts the major components of the QSMCM.



*Note: SBIU Bus and interface to IMB3 are each 16 bits wide.

Figure 14-1 QSMCM Block Diagram

14.3 Signal Descriptions

The QSMCM has 12 external pins, as shown in **Figure 14-1**. Seven of the pins, if not in use for their submodule function, can be used as general-purpose I/O port pins. The RXDx and TXDx pins can alternately serve as general-purpose input-only and output-only signals, respectively. ECK is a dedicated clock pin.

For detailed descriptions of QSMCM signals, refer to **14.6 QSMCM Pin Control Registers**, **14.7.3 QSPI Pins**, and **14.8.6 SCI Pins**.

14.4 Memory Map

The QSMCM memory map, shown in **Table 14-1**, includes the global registers, the QSPI and dual SCI control and status registers, and the QSPI RAM. The QSMCM memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **1.3 MPC555 / MPC556 Address Map** for a diagram of the MPC555 / MPC556 internal memory map.

Table 14-1 QSMCM Register Map

Access[1]	Address	MSB[2] 0	LSB 15
S	0x30 5000	QSMCM Module Configuration Register (QSMCMMCR) See **Table 14-4** for bit descriptions.	
T	0x30 5002	QSMCM Test Register (QTEST)	
S	0x30 5004	Dual SCI Interrupt Level (QDSCI_IL) See **Table 14-5** for bit descriptions.	Reserved
S	0x30 5006	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 14-6** for bit descriptions.
S/U	0x30 5008	SCI1Control Register 0 (SCC1R0) See **Table 14-23** for bit descriptions.	
S/U	0x30 500A	SCI1Control Register 1 (SCC1R1) See **Table 14-24** for bit descriptions.	
S/U	0x30 500C	SCI1 Status Register (SC1SR) See **Table 14-25** for bit descriptions.	
S/U	0x30 500E	SCI1 Data Register (SC1DR) See **Table 14-26** for bit descriptions.	
S/U	0x30 5010	Reserved	
S/U	0x30 5012	Reserved	
S/U	0x30 5014	Reserved	QSMCM Port Q Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
S/U	0x30 5016	QSMCM Pin Assignment Register (PQSPAR) See **Table 14-10** for bit descriptions.	QSMCM Data Direction Register (DDRQS) See **Table 14-11** for bit descriptions.
S/U	0x30 5018	QSPI Control Register 0 (SPCR0) See **Table 14-13** for bit descriptions.	
S/U	0x30 501A	QSPI Control Register 1 (SPCR1) See **Table 14-15** for bit descriptions.	
S/U	0x30 501C	QSPI Control Register 2 (SPCR2) See **Table 14-16** for bit descriptions.	
S/U	0x30 501E	QSPI Control Register 3 (SPCR3) See **Table 14-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-18** for bit descriptions.
S/U	0x30 5020	SCI2 Control Register 0 (SCC2R0)	
S/U	0x30 5022	SCI2 Control Register 1 (SCC2R1)	
S/U	0x30 5024	SCI2 Status Register (SC2SR)	
S/U	0x30 5026	SCI2 Data Register (SC2DR)	
S/U	0x30 5028	QSCI1 Control Register (QSCI1CR) See **Table 14-30** for bit descriptions.	
S/U	0x30 502A	QSCI1 Status Register (QSCI1SR) See **Table 14-31** for bit descriptions.	

Table 14-1 QSMCM Register Map (Continued)

Access[1]	Address	MSB[2] 0	LSB 15
S/U	0x30 502C – 0x30 504A	Transmit Queue Locations (SCTQ)	
S/U	0x30 504C – 0x30 506A	Receive Queue Locations (SCRQ)	
S/U	0x30 506C – 0x30 513F[3]	Reserved	
S/U	0x30 5140 – 0x30 517F	Receive Data RAM (REC.RAM)	
S/U	0x30 5180 – 0x30 51BF	Transmit Data RAM (TRAN.RAM)	
S/U	0x30 51C0 – 0x30 51DF	Command RAM (COMD.RAM)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.
3. Note that QRAM offsets have been changed from the original (modular family) QSMCM.

The supervisor-only data space segment contains the QSMCM global registers. These registers define parameters needed by the QSMCM to integrate with the MCU. Access to these registers is permitted only when the CPU is operating in supervisor mode.

Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user accesses. The supervisor (SUPV) bit in the QSMCM module configuration register (QSMCMMCR) designates the assignable data space as either supervisor or unrestricted. If SUPV is set, then the space is designated as supervisor-only space. Access is then permitted only when the CPU is operating in supervisor mode. If SUPV is clear, both user and supervisor accesses are permitted. To clear SUPV, the CPU must be in supervisor mode.

The QSMCM assignable data space segment contains the control and status registers for the QSPI and SCI submodules, as well as the QSPI RAM. All registers and RAM can be accessed on byte (8-bits), half-word (16-bits), and word (32-bit) boundaries. Word accesses require two consecutive IMB3 bus cycles.

14.5 QSMCM Global Registers

The QSMCM global registers contain system parameters used by the QSPI and SCI submodules for interfacing to the CPU and the intermodule bus. The global registers are listed in **Table 14-2 QSMCM Global Registers**

Table 14-2 QSMCM Global Registers

Access[1]	Address	MSB[2]	LSB
S	0x30 5000	QSMCM Module Configuration Register (QSMCMMCR) See Table 14-4 for bit descriptions.	
T	0x30 5002	QSMCM Test Register (QTEST)	
S	0x30 5004	Dual SCI Interrupt Level (QDSCI_IL) See Table 14-5 for bit descriptions.	Reserved
S	0x30 5006	Reserved	Queued SPI Interrupt Level (QSPI_IL) See Table 14-6 for bit descriptions.

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers reside on 8-bit boundaries. 16-bit registers reside on 16-bit boundaries.

14.5.1 Low-Power Stop Operation

When the STOP bit in QSMCMMCR is set, the IMB clock input to the QSMCM is disabled and the module enters a low-power operating state. QSMCMMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable in low-power stop mode. However, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be written by the CPU and is cleared by reset.

System software must bring each submodule to an orderly stop before setting STOP to avoid data corruption. The SCI receiver and transmitter should be disabled after transfers in progress are complete. The QSPI can be halted by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set.

14.5.2 Freeze Operation

The FRZ1 bit in QSMCMMCR determines how the QSMCM responds when the IMB3 FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debug mode. Setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. FREEZE causes the SCI1 transmit queue to halt on the first transfer boundary following FREEZE assertion.

14.5.3 Access Protection

The SUPV bit in the QMCR defines the assignable QSMCM registers as either supervisor-only data space or unrestricted data space.

When the SUPV bit is set, all registers in the QSMCM are placed in supervisor-only space. For any access from within user mode, the IMB3 address acknowledge ($\overline{AACK}$) signal is asserted and a bus error is generated.

Because the QSMCM contains a mix of supervisor and user registers, $\overline{AACK}$ is asserted for either supervisor or user mode accesses, and the bus cycle remains internal. If a supervisor-only register is accessed in user mode, the module responds as if an access had been made to an unauthorized register location, and a bus error is generated.

14.5.4 QSMCM Interrupts

The interrupt structure of the IMB3 supports a total of 32 interrupt levels that are time multiplexed on the $\overline{\text{IRQB}}$[0:7] lines as seen in **Figure 14-2**.

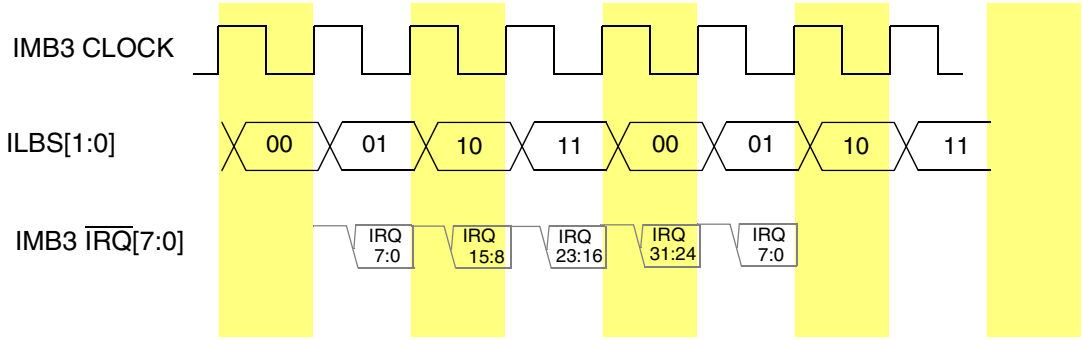


Figure 14-2 QSMCM Interrupt Levels

In this structure, all interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 14-3 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The QSMCM module is capable of generating one of the 32 possible interrupt levels on the IMB3. The levels that the interrupt will drive can be programmed into the interrupt request level (ILDSCI and ILQSPI) bits located in the interrupt configuration register (QDSCI_IL and QSPI_IL). This value determines which interrupt signal ($\overline{\text{IRQB}}$[0:7]) is driven onto the bus during the programmed time slot. **Figure 14-3** shows a block diagram of the interrupt hardware.

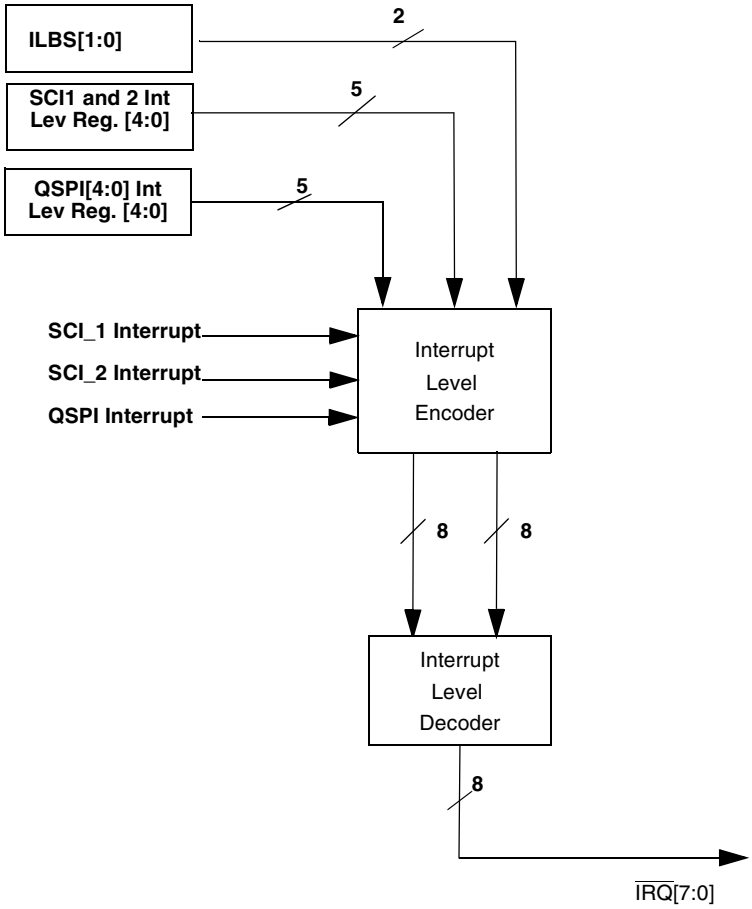


Figure 14-3 QSPI Interrupt Generation

14.5.5 QSMCM Configuration Register (QSMCMMCR)

The QSMCMMCR contains parameters for interfacing to the CPU and the intermodule bus. This register can be modified only when the CPU is in supervisor mode.

QSMCMMCR — QSMCM Configuration Register **0x30 5000**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ1	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 14-4 QSMCMMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop enable. Refer to **14.5.1 Low-Power Stop Operation**. 0 = Normal clock operation 1 = Internal clocks stopped
1	FRZ1	Freeze1 bit. Refer to **14.5.2 Freeze Operation**. 0 = Ignore the FREEZE signal 1 = Halt the QSMCM (on transfer boundary)
2:7	—	Reserved
8	SUPV	Supervisor /Unrestricted. Refer to **14.5.3 Access Protection**. 0 = Assigned registers are unrestricted (user access allowed) 1 = Assigned registers are restricted (only supervisor access allowed)
9:11	—	Reserved
12:15	IARB	This field currently has no effect. It is implemented for future interrupt arbitration schemes.

14.5.6 QSMCM Test Register (QTEST)

The QTEST register is used for factory testing of the MCU.

14.5.7 QSMCM Interrupt Level Registers (QDSCI_IL, QSPI_IL)

The QDSCI_ILI and QSPI_IL registers determine the interrupt level requested by the QSMCM. The two SCI submodules (DSCI) share a 5-bit interrupt level field, ILDSCI. The QSPI uses a separate field, ILQSPI. The level value is used to determine which interrupt is serviced first when two or more modules or external peripherals simultaneously request an interrupt. The user can select among 32 levels. This register can be accessed only when the CPU is in supervisor mode.

QDSCI_IL — QSM2 Dual SCI Interrupt Level Register **0x30 5004**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Reserved			ILDSCI					RESERVED							
RESET:															
0	0	0	0	0	0	0	0								

Table 14-5 QDSCI_IL Bit Descriptions

Bit(s)	Name	Description
0:2	—	Reserved
3:7	ILDSCI	Interrupt level of SCIs 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)
8:15	—	Reserved

QSPI_IL — QSPI Interrupt Level Register **0x30 5006**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED											ILQSPI				
RESET:															
0	0	0	0	0	0	0	0								

Table 14-6 QSPI_IL Bit Descriptions

Bit(s)	Name	Description
0:10	—	Reserved
11:15	ILQSPI	Interrupt level of SPI 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)

14.6 QSMCM Pin Control Registers

Table 14-7 lists the three QSMCM pin control registers.

Table 14-7 QSMCM Pin Control Registers

Address	Register
0x30 5014	QSMCM Port Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
0x30 5016	PORTQS Pin Assignment Register (PQSPAR) See **Table 14-11** for bit descriptions.
0x30 5017	PORTQS Data Direction Register (DDRQS) See **Table 14-11** for bit descriptions.

The QSMCM uses 12 pins. Eleven of the pins, when not being used by the serial subsystems, form a parallel port on the MCU. (The ECK pin is a dedicated external clock source.)

The port QS pin assignment register (PQSPAR) governs the usage of QSPI pins. Clearing a bit assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI.

PQSPAR does not affect operation of the SCI. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. When the SCIx transmitter or receiver is enabled, the associated TXDx or RXDx pin is assigned its SCI function.

The port QS data direction register (DDRQS) determines whether QSPI pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table 14-10** summarizes the effect of DDRQS bits on QSPI pin function.

DDRQS does not affect SCI pin function. TXDx pins are always outputs, and RXDx pins are always inputs, regardless of whether they are functioning as SCI pins or as PORTQS pins.

The port QS data register (PORTQS) latches I/O data. PORTQS writes drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, write the first data to PORTQS before configuring DDRQS.

Table 14-8 Effect of DDRQS on QSPI Pin Function

QSMCM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQS0	0	Serial data input to QSPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from QSPI
MOSI	Master	DDQS1	0	Disables data output
			1	Serial data output from QSPI
	Slave		0	Serial data input to QSPI
			1	Disables data input
SCK[1]	Master	DDQS2	—	Clock output from QSPI
	Slave		—	Clock input to QSPI
PCS[0]/$\overline{SS}$	Master	DDQS3	0	Assertion causes mode fault
			1	Chip-select output
	Slave		0	QSPI slave select input
			1	Disables slave select input
PCS[1:3]	Master	DDQS[4:6]	0	Disables chip-select output
			1	Chip-select output
	Slave		0	Inactive
			1	Inactive

NOTES:
1. SCK/QGPIO6 is a digital I/O pin unless the SPI is enabled (SPE set in SPCR1), in which case it becomes the QSPI serial clock SCK.

14.6.1 Port QS Data Register (PORTQS)

PORTQS determines the actual input or output value of a QSMCM port pin if the pin is defined as general-purpose input or output. All QSMCM pins except the ECK pin can be used as general-purpose input and/or output. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. Writes to this register affect the pins defined as outputs; reads of this register return the actual value of the pins.

PORTQS — Port QS Data Register **0x30 5014**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED				QDRX D2	QDTX D2	QDRX D1	QDTX D1	0	QDPC S3	QDPC S2	QDPC S1	QDPC S0	QD-SCK	QD-MOSI	QDMI-SO
RESET:															
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

14.6.2 PORTQS Pin Assignment Register (PQSPAR)

PQSPAR determines which of the QSPI pins, with the exception of the SCK pin, are used by the QSPI submodule, and which pins are available for general-purpose I/O. Pins may be assigned on a pin-by-pin basis. If the QSPI is disabled, the SCK pin is automatically assigned its general-purpose I/O function (QGPIO6).

QSPI pins designated by PQSPAR as general-purpose I/O pins are controlled only by PQSDDR and PQSPDR; the QSPI has no effect on these pins. PQSPAR does not affect the operation of the SCI submodule.

Table 14-9 summarizes the QSMCM pin functions.

Table 14-9 QSMCM Pin Functions

PORTQS Function	QSMCM Function
QGPI2	RXD2
QGPO2	TXD2
QGPI1	RXD1
QGPO1	TXD1
QGPIO6	SCK
QGPIO5	MOSI
QGPIO4	MISO
QGPIO3	PCS[3]
QGPIO2	PCS[2]
QGPIO1	PCS[1]
QGPIO0	PCS[0]

PQSPAR — PORTQS Pin Assignment Register **0x30 5016**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	QPAP CS3	QPAP CS2	QPAP CS1	QPAP CS0	0	QPA-MOSI	QPAM ISO	DDRQS*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-11**.

Table 14-10 PQSPAR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	QPAPCS3	0 = Pin is assigned QGPIO3 1 = Pin is assigned PCS3 function
2	QPAPCS2	0 = Pin is assigned QGPIO2 1 = Pin is assigned PCS2 function
3	QPAPCS1	0 = Pin is assigned QGPIO1 1 = Pin is assigned PCS1 function
4	QPAPCS0	0 = Pin is assigned QGPIO0 1 = Pin is assigned PCS[0] function
5	—	Reserved
6	QPAMOSI	0 = Pin is assigned QGPIO5 1 = Pin is assigned MOSI function
7	QPAMISO	0 = Pin is assigned QGPIO4 1 = Pin is assigned MISO function
8:15	DDRQS	PORSTQS data direction register. See **14.6.3 PORTQS Data Direction Register (DDRQS)**.

14.6.3 PORTQS Data Direction Register (DDRQS)

DDRQS assigns QSPI pin as an input or an output regardless of whether the QSPI submodule is enabled or disabled. All QSPI pins are configured during reset as general-purpose inputs.

This register does not affect SCI operation. The TXD1 and TXD2 remain output pins dedicated to the SCI submodules, and the RXD1, RXD2 and ECK pins remain input pins dedicated to the SCI submodules.

DDRQS — PORTQS Data Direction Register **0x30 5016**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PQSPAR*								0	QDD-PCS3	QDD-PCS2	QDD-PCS1	QDD-PCS0	QDD-SCK	QDD-MOSI	QDD-MISO
RE-SET:								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 14-10**.

Table 14-11 DDRQS Bit Descriptions

Bit(s)	Name	Description
0:7	PQSPAR	PORTSQS pin assignment register. See **14.6.2 PORTQS Pin Assignment Register (PQSPAR)**.
8	—	Reserved
9	QDDPCS3	QSPI pin data direction for the pin PCS3 0 = Pin direction is input 1 = Pin direction is output
10	QDDPCS2	QSPI pin data direction for the pin PCS2 0 = Pin direction is input 1 = Pin direction is output
11	QDDPCS1	QSPI pin data direction for the pin PCS1 0 = Pin direction is input 1 = Pin direction is output
12	QDDPCS0	QSPI pin data direction for the pin PCS[0] 0 = Pin direction is input 1 = Pin direction is output
13	QDDSCK	QSPI pin data direction for the pin SCK 0 = Pin direction is input 1 = Pin direction is output
14	QPDMOSI	QSPI pin data direction for the pin MOSI 0 = Pin direction is input 1 = Pin direction is output
15	QPDMISO	QSPI pin data direction for the pin MISO 0 = Pin direction is input 1 = Pin direction is output

14.7 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) is used to communicate with external devices through a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The QSPI can perform full duplex three-wire or half duplex two-wire transfers. Several transfer rates, clocking, and interrupt-driven communication options are available. **Figure 14-4** is a block diagram of the QSPI.

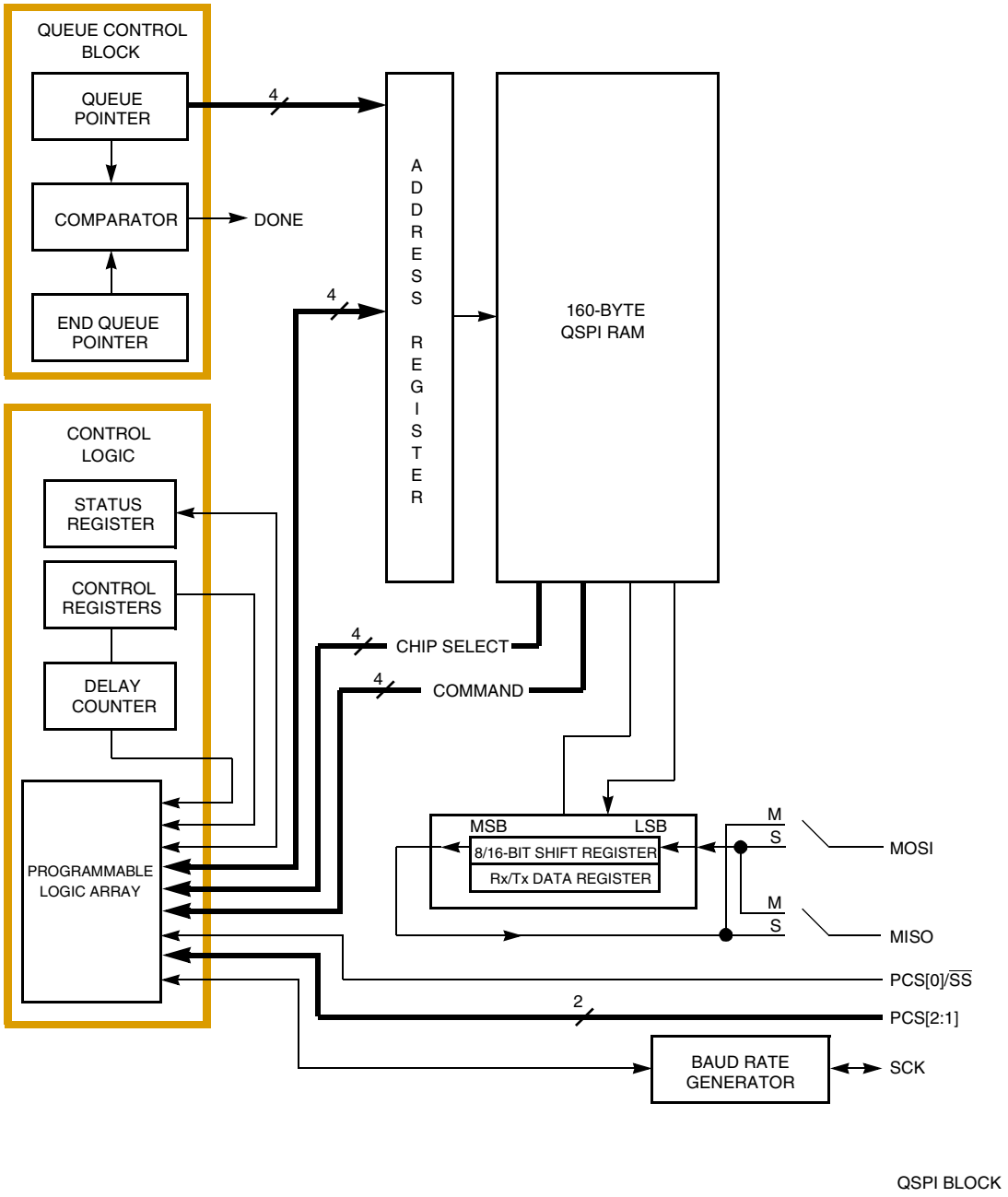


Figure 14-4 QSPI Block Diagram

Serial transfers of eight to 16 bits can be specified. Programmable transfer length simplifies interfacing to devices that require different data lengths.

An inter-transfer delay of approximately 0.8 to 204 µs (using a 40-MHz IMB clock) can be programmed. The default delay is 17 clocks (0.425 µs at 40 MHz). Programmable delay simplifies the interface to devices that require different delays between transfers.

A dedicated 160-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. This allows serial peripherals to be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 32 serial transfers without CPU intervention. Each queue entry contains all the information needed by the QSPI to independently complete one serial transfer.

A pointer identifies the queue location containing the data and command for the next serial transfer. Normally, the pointer address is incremented after each serial transfer, but the CPU can change the pointer value at any time. Support for multiple-tasks can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. The chip-select signals simplify interfacing by reducing CPU intervention. If the chip-select signals are externally decoded, 16 independent select signals can be generated.

Wrap-around mode allows continuous execution of queued commands. In wrap-around mode, newly received data replaces previously received data in the receive RAM. Wrap-around mode can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows transfer of an uninterrupted bit stream. From 8 to 512 bits can be transferred without CPU intervention. Longer transfers are possible, but minimal intervention is required to prevent loss of data. A standard delay of 17 IMB clocks (0.8 µs with a 40-MHz IMB clock) is inserted between the transfer of each queue entry.

14.7.1 QSPI Registers

The QSPI memory map, shown in **Table 14-12**, includes the QSMCM global and pin control registers, four QSPI control registers (SPCR[0:3]), the status register (SPSR), and the QSPI RAM. Registers and RAM can be read and written by the CPU. The memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **1.3 MPC555 / MPC556 Address Map** for a diagram of the MPC555 / MPC556 internal memory map.

Table 14-12 QSPI Register Map

Access[1]	Address	MSB[2]	LSB
S/U	0x30 5018	QSPI Control Register 0 (SPCR0) See **Table 14-13** for bit descriptions.	
S/U	0x30 501A	QSPI Control Register 1 (SPCR1) See **Table 14-15** for bit descriptions.	
S/U	0x30 501C	QSPI Control Register 2 (SPCR2) See **Table 14-16** for bit descriptions.	
S/U	0x30 501E/ 0x30 501F	QSPI Control Register 3 (SPCR3) See **Table 14-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-18** for bit descriptions.
S/U	0x30 5140 – 0x30 517F	Receive Data RAM (32 half-words)	
S/U	0x30 5180 – 0x30 51BF	Transmit Data RAM (32 half-words)	
S/U	0x30 51C0 – 0x30 51DF	Command RAM (32 bytes)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.

To ensure proper operation, set the QSPI enable bit (SPE) in SPCR1 only after initializing the other control registers. Setting this bit starts the QSPI.

Rewriting the same value to a control register does not affect QSPI operation with the exception of writing NEWQP in SPCR2. Rewriting the same value to these bits causes the RAM queue pointer to restart execution at the designated location.

Before changing control bits, the user should halt the QSPI. Writing a different value into a control register other than SPCR2 while the QSPI is enabled may disrupt operation. SPCR2 is buffered, preventing any disruption of the current serial transfer. After the current serial transfer is completed, the new SPCR2 value becomes effective.

14.7.1.1 QSPI Control Register 0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU has read/write access to SPCR0, but the QSPI has read access only. SPCR0 must be initialized before QSPI operation begins. Writing a new value to SPCR0 while the QSPI is enabled disrupts operation.

SPCR0 — QSPI Control Register 0 **0x30 5018**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Table 14-13 SPCR0 Bit Descriptions

Bit(s)	Name	Description
0	MSTR	Master/slave mode select 0 = QSPI is a slave device and only responds to externally generated serial transfers. 1 = QSPI is the system master and can initiate transmission to external SPI devices.
1	WOMQ	Wired-OR mode for QSPI pins. This bit controls the QSPI pins regardless of whether they are used as general-purpose outputs or as QSPI outputs, and regardless of whether the QSPI is enabled or disabled. 0 = Pins designated for output by DDRQS operate in normal mode. 1 = Pins designated for output by DDRQS operate in open drain mode.
2:5	BITS	Bits per transfer. In master mode, when BITSE is set in a command RAM byte, BITS determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred regardless of the value in BITS. In slave mode, the BITS field always determines the number of bits the QSPI will receive during each transfer before storing the received data. Data transfers from 8 to 16 bits are supported. Illegal (reserved) values default to eight bits. **Table 14-14** shows the number of bits per transfer.
6	CPOL	Clock polarity. CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices. 0 = The inactive state of SCK is logic zero. 1 = The inactive state of SCK is logic one.
7	CPHA	Clock phase. CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. 0 = Data is captured on the leading edge of SCK and changed on the trailing edge of SCK. 1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK
8:15	SPBR	Serial clock baud rate. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU IMB clock. Baud rate is selected by writing a value from 2 to 255 into SPBR. The following equation determines the SCK baud rate: $\text{SCK Baud Rate} = \frac{f_{SYS}}{2 \times \text{SPBR}}$ Refer to **14.7.5.2 Baud Rate Selection** for more information.

Table 14-14 Bits Per Transfer

BITS[3:0]	Bits per Transfer
0000	16
0001 to 0111	Reserved (defaults to 8)
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

14.7.1.2 QSPI Control Register 1

SPCR1 enables the QSPI and specifies transfer delays. The CPU has read/write access to SPCR1, but the QSPI has read access only to all bits except SPE. SPCR1 must be written last during initialization because it contains SPE. The QSPI automatically clears this bit after it completes all serial transfers or when a mode fault occurs. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPCR1 — QSPI Control Register 1 **0x30 501A**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPE	DSCKL							DTL							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 14-15 SPCR1 Bit Descriptions

Bit(s)	Name	Description
0	SPE	QSPI enable. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O. 1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.
1:7	DSCKL	Delay before SCK. When the DSCK bit is set in a command RAM byte, this field determines the length of the delay from PCS valid to SCK transition. The following equation determines the actual delay before SCK: $\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$ where DSCKL equals is in the range of 1 to 127. Refer to **14.7.5.3 Delay Before Transfer** for more information.
8:15	DTL	Length of delay after transfer. When the DT bit is set in a command RAM byte, this field determines the length of the delay after a serial transfer. The following equation is used to calculate the delay: $\text{Delay after Transfer} = \frac{32 \times DTL}{f_{SYS}}$ where DTL is in the range of 1 to 255. A zero value for DTL causes a delay-after-transfer value of $8192 \div F_{SYS}$ (204.8 µs with a 40-MHz IMB clock). Refer to **14.7.5.4 Delay After Transfer** for more information.

14.7.1.3 QSPI Control Register 2

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSPI has read access only. Writes to this register are buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not the buffer.

SPCR2 — QSPI Control Register 2 **0x30 501C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPIFIE	WREN	WRTO	ENDQP					Reserved			NEWQP				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-16 SPCR2 Bit Descriptions

Bit(s)	Name	Description
0	SPIFIE	SPI finished interrupt enable. Refer to **14.7.4.2 QSPI Interrupts**. 0 = QSPI interrupts disabled 1 = QSPI interrupts enabled
1	WREN	Wrap enable. Refer to **14.7.5.7 Master Wraparound Mode**. 0 = Wraparound mode disabled. 1 = Wraparound mode enabled.
2	WRTO	Wrap to. When wraparound mode is enabled and after the end of queue has been reached, WRTO determines which address the QSPI executes next. The end of queue is determined by an address match with ENDQP. 0 = Wrap to pointer address 0x0 1 = Wrap to address in NEWQP
3:7	ENDQP	Ending queue pointer. This field determines the last absolute address in the queue to be completed by the QSPI. After completing each command, the QSPI compares the queue pointer value of the just-completed command with the value of ENDQP. If the two values match, the QSPI sets SPIF to indicate it has reached the end of the programmed queue. Refer to **14.7.4 QSPI Operation** for more information.
8:10	—	Reserved
11:15	NEWQP	New queue pointer value. This field contains the first QSPI queue address. Refer to **14.7.4 QSPI Operation** for more information.

14.7.1.4 QSPI Control Register 3

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enable, and the halt control bit. The CPU has read/write access to SPCR3, but the QSPI has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation.

SPCR3 — QSPI Control Register **0x30 501E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Reserved					LOOP Q	HMIE	HALT	SPSR*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-18**.

Table 14-17 SPCR3 Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5	LOOPQ	QSPI loop mode. LOOPQ controls feedback on the data serializer for testing. 0 = Feedback path disabled. 1 = Feedback path enabled.
6	HMIE	HALTA and MODF interrupt enable. HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR. 0 = HALTA and MODF interrupts disabled. 1 = HALTA and MODF interrupts enabled.
7	HALT	Halt QSPI. When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI operates normally. 1 = QSPI is halted for subsequent restart.
8:15	—	SPSR. See**Table 14-18** for bit descriptions.

14.7.1.5 QSPI Status Register

The SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in this register. To clear status flags, the CPU reads SPSR with the flags set and then writes the SPSR with zeros in the appropriate bits. Writes to CPTQP have no effect.

SPSR — QSPI Status Register **0x30 501E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPCR3*								SPIF	MODF	HAL-TA	CPTQP				
								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 14-17**.

Table 14-18 SPSR Bit Descriptions

Bit(s)	Name	Description
0:7	SPCR3	See bit descriptions in **Table 14-17**.
8	SPIF	QSPI finished flag. SPIF is set after execution of the command at the address in ENDQP in SPCR2. If wraparound mode is enabled (WREN = 1), the SPIF is set, after completion of the command defined by ENDQP, each time the QSPI cycles through the queue. 0 = QSPI is not finished 1 = QSPI is finished
9	MODF	Mode fault flag. The QSPI asserts MODF when the QSPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver. Refer to **14.7.8 Mode Fault** for more information. 0 = Normal operation 1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).
10	HALTA	Halt acknowledge flag. HALTA is set when the QSPI halts in response to setting the HALT bit in SPCR3. HALTA is also set when the IMB3 FREEZE signal is asserted, provided the FRZ1 bit in the QSMCMMCR is set. To prevent undefined operation, the user must not modify any QSPI control registers or RAM while the QSPI is halted. If HMIE in SPCR3 is set the QSPI sends interrupt requests to the CPU when HALTA is asserted. 0 = QSPI is not halted. 1 = QSPI is halted
11:15	CPTQP	Completed queue pointer. CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value 0x0 or a pointer to the last command completed in the previous queue. If the QSPI is halted, CPTQP may be used to determine which commands have not been executed. The CPTQP may also be used to determine which locations in the receive data segment of the QSPI RAM contain valid received data.

14.7.2 QSPI RAM

The QSPI contains a 160-byte block of dual-ported static RAM that can be accessed by both the QSPI and the CPU. Because of this dual access capability, up to two wait states may be inserted into CPU access time if the QSPI is in operation.

The size and type of access of the QSPI RAM by the CPU affects the QSPI access time. The QSPI allows byte, half-word, and word accesses. Only word accesses of the RAM by the CPU are coherent because these accesses are an indivisible operation. If the CPU makes a coherent access of the QSPI RAM, the QSPI cannot access the QSPI RAM until the CPU is finished. However, a word or misaligned word access is not coherent because the CPU must break its access of the QSPI RAM into two parts, which allows the QSPI to access the QSPI RAM between the two accesses by the CPU.

The RAM is divided into three segments: receive data RAM, transmit data RAM, and command data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored for transmission to an external device. Command data defines transfer parameters. **Figure 14-5** shows RAM organization.

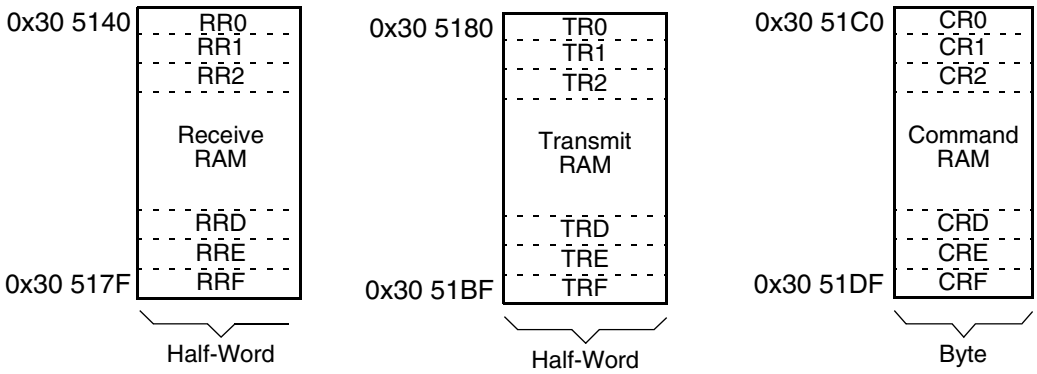


Figure 14-5 QSPI RAM

14.7.2.1 Receive RAM

Data received by the QSPI is stored in this segment, to be read by the CPU. Data stored in the receive RAM is right-justified,(i.e., the least significant bit is always in the right-most bit position within the word regardless of the serial transfer length). Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, half-word, or word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

14.7.2.2 Transmit RAM

Data that is to be transmitted by the QSPI is stored in this segment. The CPU normally writes one word of data into this segment for each queue command to be executed. If the corresponding peripheral, such as a serial input port, is used solely to input data, then this segment does not need to be initialized.

Data must be written to transmit RAM in a right-justified format. The QSPI cannot modify information in the transmit RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

14.7.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 32 bytes. Each byte is divided into two fields. The peripheral chip-select field, enables peripherals for transfer. The command control field provides transfer options.

A maximum of 32 commands can be in the queue. These bytes are assigned an address from 0x00 to 0x1F. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CR[0:F] — Command RAM **0x30 51C0 – 0x30 51DF**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
Command Control				Peripheral Chip Select			

NOTES:
1. The PCS[0] bit represents the dual-function PCS[0]/$\overline{SS}$.

Table 14-19 Command RAM Bit Descriptions

Bit(s)	Name	Description
0	CONT	Continue 0 = Control of chip selects returned to PORTQS after transfer is complete. 1 = Peripheral chip selects remain asserted after transfer is complete.
1	BITSE	Bits per transfer enable 0 = Eight bits 1 = Number of bits set in BITS field of SPCR0.
2	DT	Delay after transfer 0 = Delay after transfer is 17 ÷ F_{SYS}. 1 = SPCR1 DTL[7:0] specifies delay after transfer PCS valid to SCK.
3	DSCK	PCS to SCK Delay 0 = PCS valid to SCK delay is one-half SCK. 1 = SPCR1 DSCKL[6:0] specifies delay from PCS valid to SCK.
4:7	PCS[3:0]	Peripheral chip selects. Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select may be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided proper fanout is observed. PCS[0] shares a pin with the slave select ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

Refer to **14.7.5 Master Mode Operation** for more information on the command RAM.

14.7.3 QSPI Pins

Seven pins are associated with the QSPI. When not needed by the QSPI, they can be configured for general-purpose I/O. **Table 14-20** identifies the QSPI pins and their functions. Register DDRQS determines whether the pins are designated as input or output. The user must initialize DDRQS for the QSPI to function correctly.

Table 14-20 QSPI Pin Functions

Pin Names	Mnemonic	Mode	Function
Master in slave out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master out slave in	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial clock	SCK[1]	Master Slave	Clock output from QSPI clock Input to QSPI
Peripheral chip selects	PCS[1:3]	Master	Outputs select peripheral(s)
Peripheral chip select[2] Slave select[3]	PCS[0]/ $\overline{SS}$	Master Slave	Output selects peripheral(s) Input selects the QSPI
Slave select[4]	$\overline{SS}$	Master	May cause mode fault

NOTES:
1. All QSPI pins (except SCK) can be used as general-purpose I/O if they are not used by the QSPI while the QSPI is operating. SCK can only be used for general-purpose I/O if the QSPI is disabled.
2. An output (PCS[0]) when the QSPI is in master mode.
3. An input ($\overline{SS}$) when the QSPI is in slave mode.
4. An input ($\overline{SS}$) when the QSPI is in master mode; useful in multimaster systems.

14.7.4 QSPI Operation

The QSPI uses a dedicated 160-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments: 32 command control bytes, 64 transmit data bytes, and 64 receive data bytes.

Once the CPU has set up a queue of QSPI commands, written the transmit data segment with information to be sent, and enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating completion, and then either interrupts the CPU or waits for CPU intervention.

QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to each queue entry, 0x0 to 0x2F.

The CPU initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU or waits for intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

The internal pointer is initialized to the same value as NEWQP. During normal operation, the command pointed to by the internal pointer is executed, the value in the internal pointer is copied into CPTQP, the internal pointer is incremented, and then the sequence repeats. Execution continues at the internal pointer address unless the

NEWQP value is changed. After each command is executed, ENDQP and CPTQP are compared. When a match occurs, the SPIF flag is set and the QSPI stops and clears SPE, unless wraparound mode is enabled.

At reset, NEWQP is initialized to 0x0. When the QSPI is enabled, execution begins at queue address 0x0 unless another value has been written into NEWQP. ENDQP is initialized to 0x0 at reset but should be changed to the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When NEWQP changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to 0x0 transfers only the data in transmit RAM location 0x0.

14.7.4.1 Enabling, Disabling, and Halting the SPI

The SPE bit in the SPCR1 enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS[0]/$\overline{SS}$ pin to respond to the external initialization of a serial transfer.

When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access to all 160 bytes of the QSPI RAM. The QSPI can read-only the transmit data segment and the command control segment and can write-only the receive data segment of the QSPI RAM.

The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS[0]/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS[0]/$\overline{SS}$ is driven low externally.

Setting the HALT bit in SPCR3 stops the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is set, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip select pins with the value designated by the last command before the halt. If CONT was cleared, the QSPI drives the peripheral chip-select pins to the value in register PORTQS.

If HALT is set during the last command in the queue, the QSPI completes the last command, sets both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF or clear SPE. QSPI execution continues when the CPU clears HALT.

To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE can then be safely cleared, providing an orderly method of shutting down the QSPI quickly after the current serial transfer is completed. The CPU can disable the QSPI immediately by clearing SPE. However, loss of data from a current serial transfer may result and confuse an external SPI device.

14.7.4.2 QSPI Interrupts

The QSPI has three possible interrupt sources but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the user must ascertain the interrupt cause by reading the SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the interrupt source.

The SPIFIE bit in SPCR2 enables the QSPI to generate an interrupt request upon assertion of the SPIF status flag. Because it is buffered, the value written to SPIFIE applies only upon completion of the queue (the transfer of the entry indicated by ENDPQ). Thus, if a single sequence of queue entries is to be transferred (i.e., no WRAP), then SPIFIE should be set to the desired state before the first transfer.

If a sub-queue is to be used, the same CPU write that causes a branch to the sub-queue may enable or disable the SPIF interrupt for the sub-queue. The primary queue retains its own selected interrupt mode, either enabled or disabled.

The SPIF interrupt must be cleared by clearing SPIF. Subsequent interrupts may then be prevented by clearing SPIFIE. Clearing SPIFIE does not immediately clear an interrupt already caused by SPIF.

14.7.4.3 QSPI Flow

The QSPI operates in either master or slave mode. Master mode is used when the MCU initiates data transfers. Slave mode is used when an external device initiates transfers. Switching between these modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSMCM and QSPI registers must be initialized properly.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from the transmit RAM and received by the receive RAM.

In slave mode, operation proceeds in response to SS pin assertion by an external SPI bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multi-master operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

Figure 14-6 shows QSPI initialization. **Figure 14-7** through **Figure 14-11** show QSPI master and slave operation. The CPU must initialize the QSMCM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation. The command queue must be written before the QSPI is enabled for master

mode operation. Any data to be transmitted should be written into transmit RAM before the QSPI is enabled. During wraparound operation, data for subsequent transmissions can be written at any time.

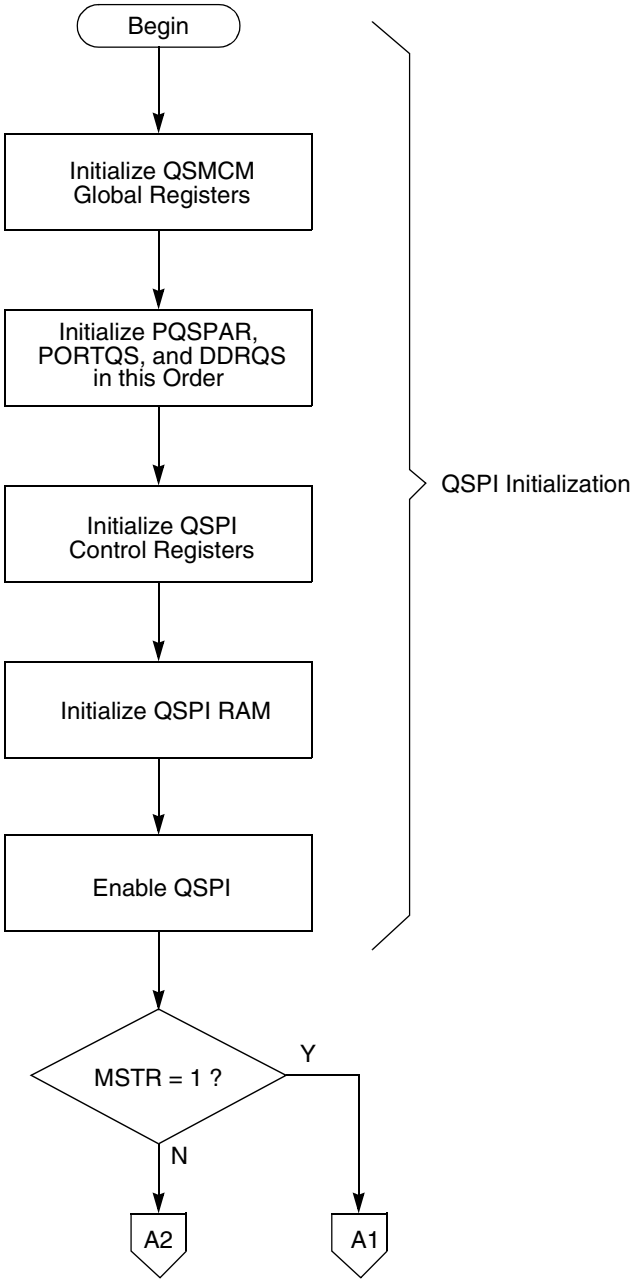


Figure 14-6 Flowchart of QSPI Initialization Operation

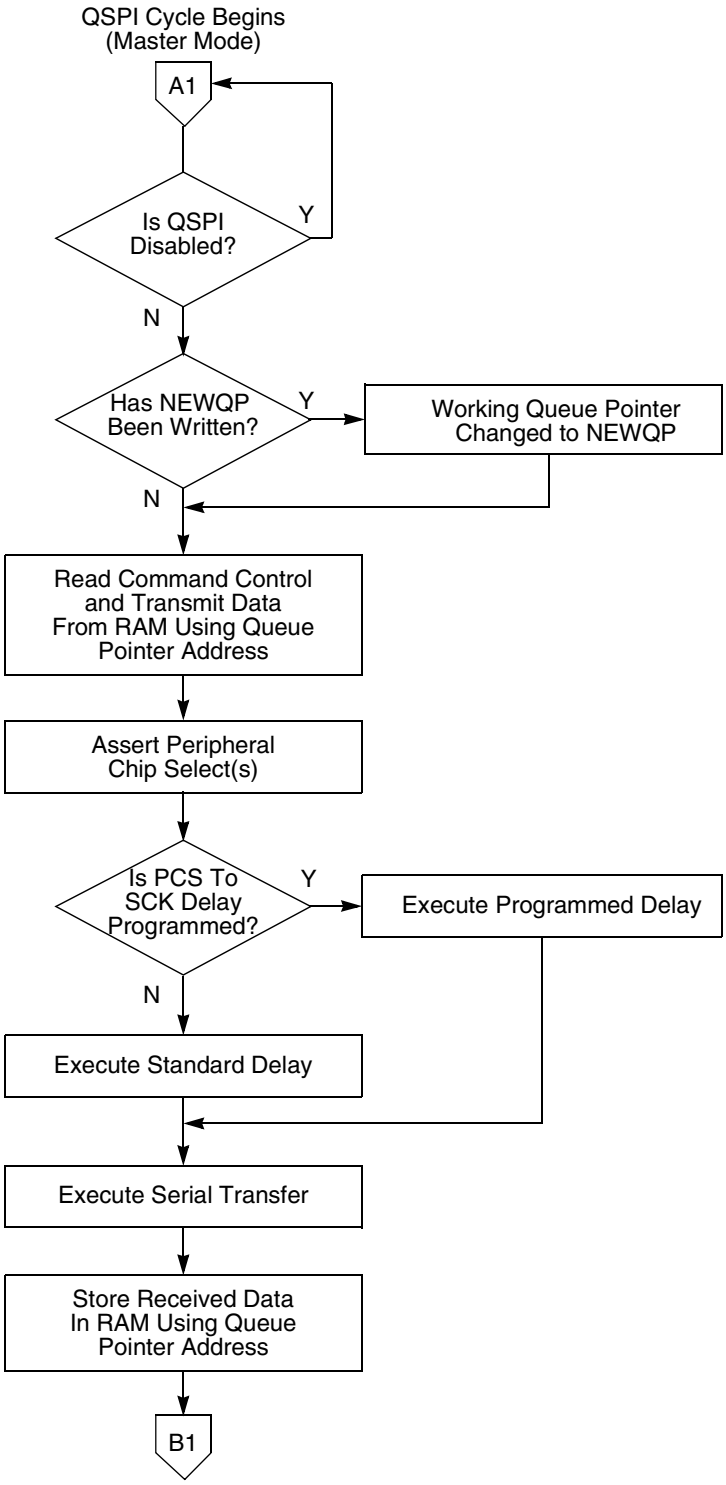


Figure 14-7 Flowchart of QSPI Master Operation (Part 1)

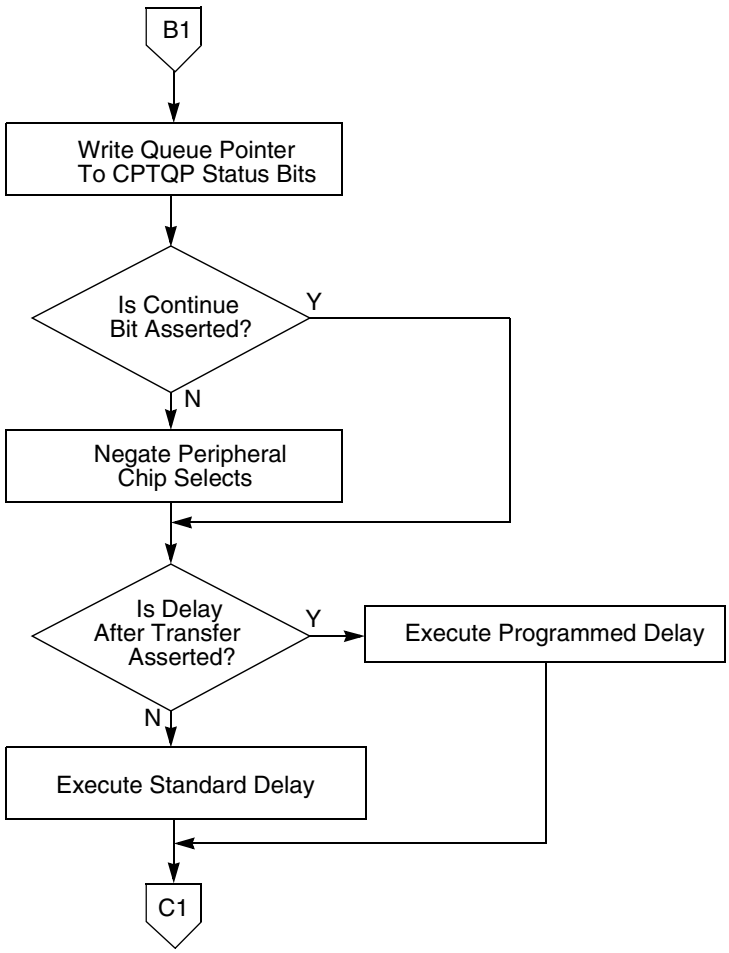


Figure 14-8 Flowchart of QSPI Master Operation (Part 2)

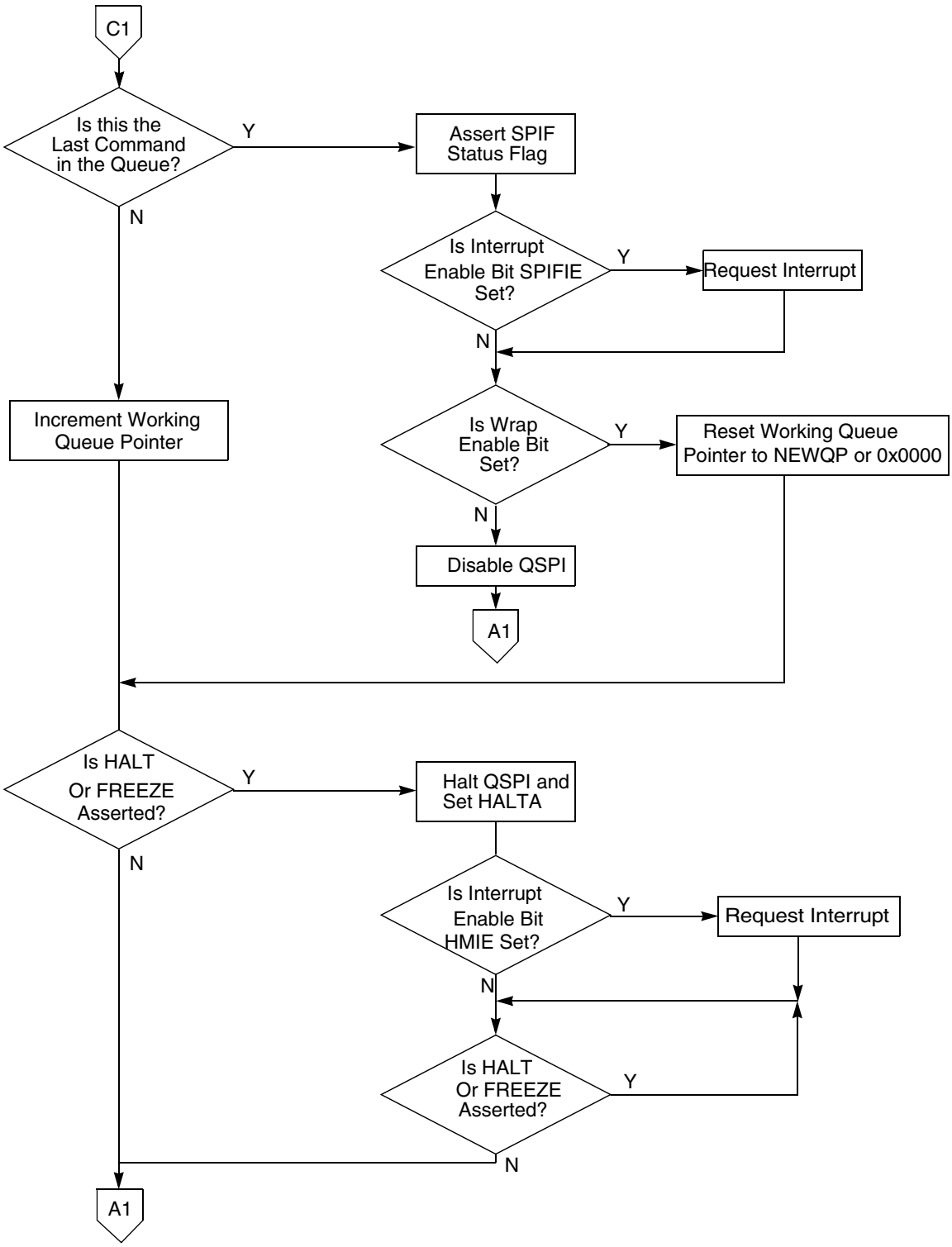


Figure 14-9 Flowchart of QSPI Master Operation (Part 3)

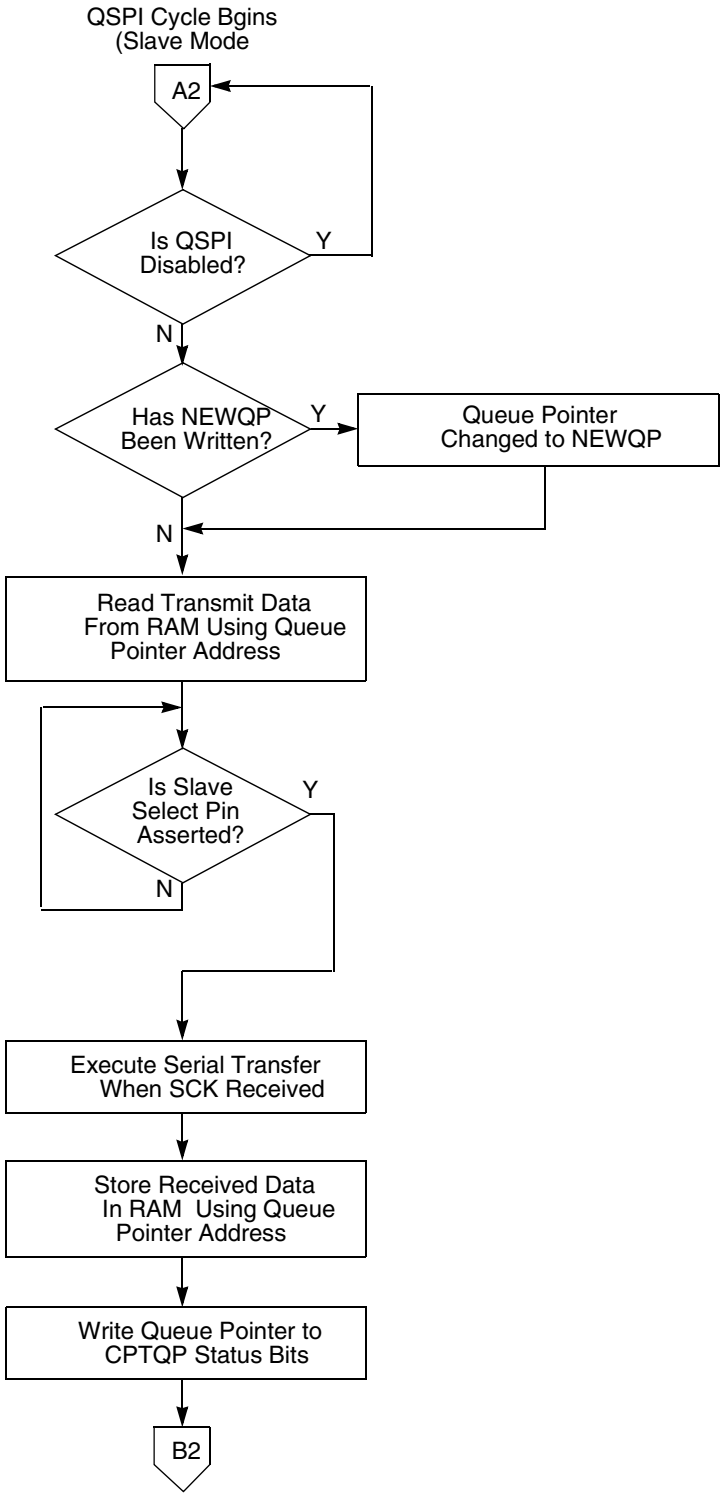


Figure 14-10 Flowchart of QSPI Slave Operation (Part 1)

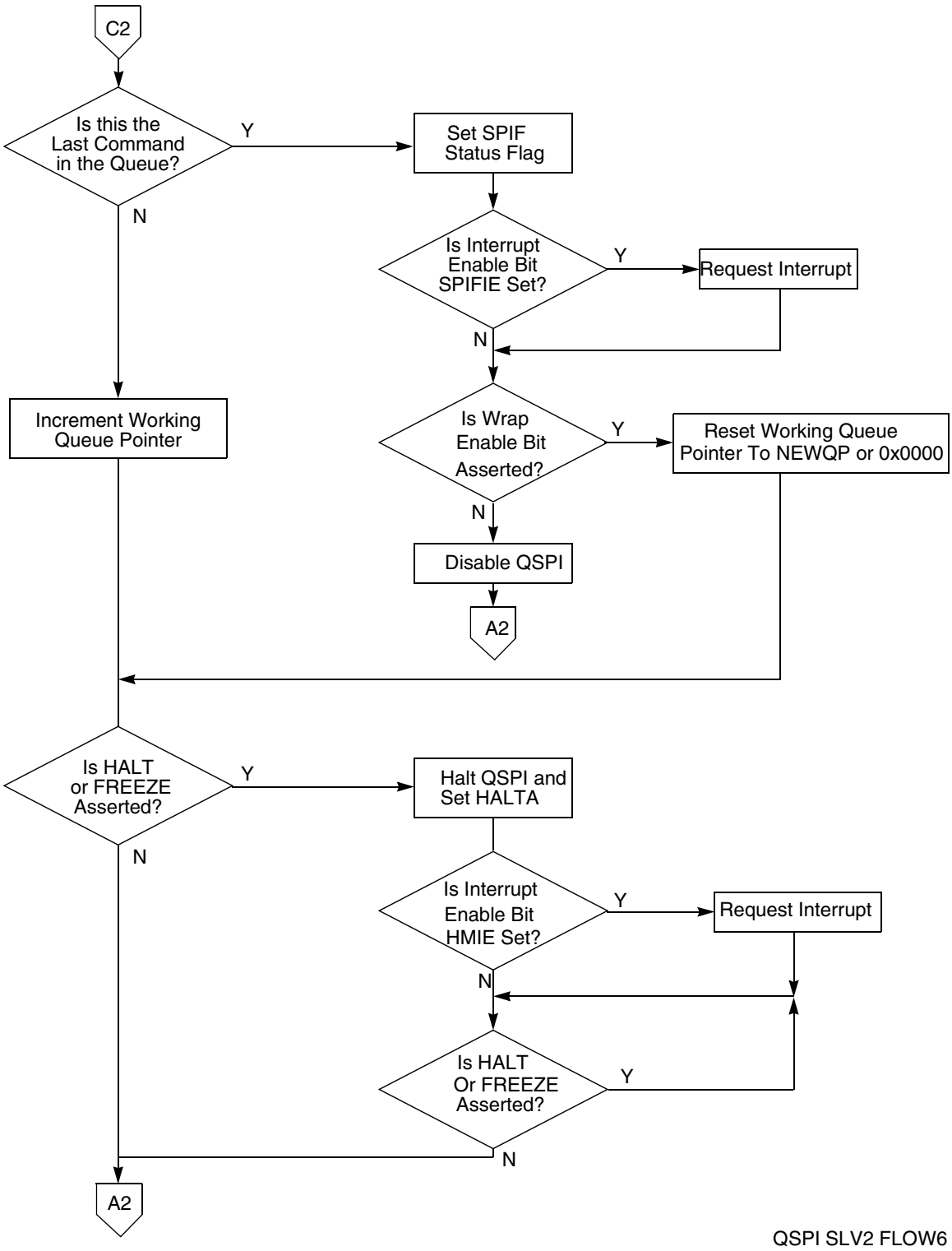


Figure 14-11 Flowchart of QSPI Slave Operation (Part 2)

Normally, the SPI bus performs synchronous bi-directional transfers. The serial clock on the SPI bus master supplies the clock signal SCK to time the transfer of data. Four

possible combinations of clock phase and polarity can be specified by the CPHA and CPOL bits in SPCR0.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but can be set to any value from eight to sixteen bits by writing a value into the BITS field in SPCR0 and setting BITSE in command RAM.

Typically, SPI bus outputs are not open drain unless multiple SPI masters are in the system. If needed, the WOMQ bit in SPCR0 can be set to provide wired-OR, open drain outputs. An external pull-up resistor should be used on each output line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

14.7.5 Master Mode Operation

Setting the MSTR bit in SPCR0 selects master mode operation. In master mode, the QSPI can initiate serial transfers, but cannot respond to externally initiated transfers. When the slave select input of a device configured for master mode is asserted, a mode fault occurs.

Before QSPI operation begins, PQSPAR must be written to assign the necessary pins to the QSPI. The pins necessary for master mode operation are MISO, MOSI, SCK, and one or more of the chip-select pins. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. SCK is the serial clock output in master mode and must be assigned to the QSPI for proper operation.

The PORTQS data register must next be written with values that make the QGPIO[6]/SCK (bit 13 QDSCK of PORTQS) and QGPIO[3:0]/PCS[3:0] (bits 12:9 QDPCS[3:0] of PORTQS) outputs inactive when the QSPI completes a series of transfers. Pins allocated to the QSPI by PQSPAR are controlled by PORTQS when the QSPI is inactive. PORTQS I/O pins driven to states opposite those of the inactive QSPI signals can generate glitches that momentarily enable or partially clock a slave device.

For example, if a slave device operates with an inactive SCK state of logic one (CPOL = 1) and uses active low peripheral chip-select PCS[0], the QDSCK and QDPCS0 bits in PORTQS must be set to 0b11. If QDSCK and QDPCS0 = 0b00, falling edges will appear on QGPIO[6]/SCK and GPIO[0]/PCS[0] as the QSPI relinquishes control of these pins and PORTQS drives them to logic zero from the inactive SCK and PCS[0] states of logic one.

Before master mode operation is initiated, QSMCM register DDRQS is written last to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0] as outputs. The MISO pin must be configured as an input.

After pins are assigned and configured, write appropriate data to the command queue. If data is to be transmitted, write the data to transmit RAM. Initialize the queue pointers as appropriate.

QSPI operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in a command RAM byte is set, PCS pins are continuously driven to specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

14.7.5.1 Clock Phase and Polarity

In master mode, data transfer is synchronized with the internally-generated serial clock SCK. Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

14.7.5.2 Baud Rate Selection

Baud rate is selected by writing a value from two to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU IMB clock.

The following expressions apply to the SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{SYS}}{2 \times \text{SPBR}}$$

or

$$\text{SPBR} = \frac{f_{SYS}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state. At reset, the SCK baud rate is initialized to one eighth of the IMB clock frequency.

Table 14-21 provides some example SCK baud rates with a 40-MHz IMB clock.

Table 14-21 Example SCK Frequencies with a 40-MHz IMB Clock

Division Ratio	SPBR Value	SCK Frequency
4	2	10.00 MHz
6	3	6.67 MHz
8	4	5.00 MHz
14	7	2.86 MHz
28	14	1.43 MHz
58	29	689 kHz
280	140	143 kHz
510	255	78.43 kHz

14.7.5.3 Delay Before Transfer

The DSCK bit in each command RAM byte inserts either a standard (DSCK = 0) or user-specified (DSCK = 1) delay from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the length of the user-defined delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$$

where DSCKL is in the range from 1 to 127.

NOTE

A zero value for DSCKL causes a delay of 128 IMB clocks, which equals 3.2 µs for a 40-MHz IMB clock. Because of design limits, a DSCKL value of one defaults to the same timing as a value of two.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the SCK period.

14.7.5.4 Delay After Transfer

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in each command RAM byte determines whether the standard delay period

(DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \text{ ¥ } DTL}{f_{SYS}}$$

where DTL is in the range from one to 255.

A zero value for DTL causes a delay-after-transfer value of 8192 ÷ IMB clock frequency (204.8 µs with a 40-MHz IMB clock).

If DT is zero in a command RAM byte, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = \frac{17}{f_{SYS}}$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the IMB clock is operating at a slower rate, the delay between transfers must be increased proportionately.

14.7.5.5 Transfer Length

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value from eight (0b1000) to 16 (0b0000) bits, inclusive. Reserved values (from 0b0001 to 0b0111) default to eight bits. The programmed value must be written into the BITS field in SPCR0. The BITSE bit in each command RAM byte determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used.

14.7.5.6 Peripheral Chip Selects

Peripheral chip-select signals are used to select an external device for serial data transfer. Chip-select signals are asserted when a command in the queue is executed. Signals are asserted at a logic level corresponding to the value of the PCS[3:0] bits in each command byte. More than one chip-select signal can be asserted at a time, and more than one external device can be connected to each PCS pin, provided proper fanout is observed. PCS[0] shares a pin with the slave select $\overline{SS}$ signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

To configure a peripheral chip select, set the appropriate bit in PQSPAR, then configure the chip-select pin as an output by setting the appropriate bit in DDRQS. The value of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If the base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared during reset. If no new data is written to PORTQS before pin assignment and configuration as an output, the base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

14.7.5.7 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

Wraparound mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

14.7.6 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external SPI bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSMCM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO, MOSI, SCK, and PCS[0]/$\overline{\text{SS}}$. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the serial clock input in slave mode and must be assigned to the QSPI for proper operation. Assertion of the active-low slave select signal $\overline{\text{SS}}$ initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS[0]/$\overline{\text{SS}}$ pins as inputs. The MISO pin must be configured as an output.

After pins are assigned and configured, write data to be transmitted into transmit RAM. Command RAM is not used in slave mode, and does not need to be initialized. Set the queue pointers, as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select PCS[0]/$\overline{SS}$ pin begins slave mode operation at the address indicated by NEWQP. Data that is received is stored at the pointer address in receive RAM. Data is simultaneously loaded into the data serializer from the pointer address in transmit RAM and transmitted. Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine upon which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI pin.

Because the command RAM is not used in slave mode, the CONT, BITSE, DT, DSCK, and peripheral chip-select bits have no effect. The PCS[0]/$\overline{SS}$ pin is used only as an input.

The SPBR, DT and DSCKL fields in SPCR0 and SPCR1 bits are not used in slave mode. The QSPI drives neither the clock nor the chip-select pins and thus cannot control clock rate or transfer delay.

Because the BITSE option is not available in slave mode, the BITS field in SPCR0 specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS[3:0] has been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads new transmit data from transmit RAM into the data serializer. The working queue pointer address is used the next time PCS[0]/$\overline{SS}$ is asserted, unless the RCPU writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If $\overline{SS}$ goes high before the number of bits specified by the BITS field is transferred, the QSPI resumes operation at the same pointer address the next time $\overline{SS}$ is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before $\overline{SS}$ is negated, pointers are incremented and operation continues.

The QSPI transmits as many bits as it receives at each queue address, until the BITS value is reached or $\overline{SS}$ is negated. $\overline{SS}$ does not need to go high between transfers as the QSPI transfers data until reaching the end of the queue, whether $\overline{SS}$ remains low or is toggled between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2. Slave wraparound operation is identical to master wraparound operation.

14.7.6.1 Description of Slave Operation

After reset, the QSMCM registers and the QSPI control registers must be initialized as described above. Although the command control segment is not used, the transmit and receive data segments may, depending upon the application, need to be initialized. If meaningful data is to be sent out from the QSPI, the user should write the data to the transmit data segment before enabling the QSPI.

If SPE is set and MSTR is not set, a low state on the slave select (PCS[0]/$\overline{SS}$) pin commences slave mode operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the ad-dress indicated by NEWQP. Data is transferred in response to an external slave clock input at the SCK pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The QSPI does not drive any of the four peripheral chip-selects as outputs. PCS[0]/$\overline{SS}$ is used as an input.

Although CONT cannot be used in slave mode, a provision is made to enable receipt of more than 16 data bits. While keeping the QSPI selected (PCS[0]/$\overline{SS}$ is held low), the QSPI stores the number of bits, designated by BITS, in the current receive data segment address, increments NEWQP, and continues storing the remaining bits (up to the BITS value) in the next receive data segment address.

As long as PCS[0]/$\overline{SS}$ remains low, the QSPI continues to store the incoming bit stream in sequential receive data segment addresses, until either the value in BITS is reached or the end-of-queue address is used with wraparound mode disabled.

When the end of the queue is reached, the SPIF flag is asserted, optionally causing an interrupt. If wraparound mode is disabled, any additional incoming bits are ignored.

If wraparound mode is enabled, storing continues at either address 0x0 or the address of NEWQP, depending on the WRTO value. When using this capability to receive a long incoming data stream, the proper delay between transfers must be used. The QSPI requires time, approximately 0.425 µs with a 40-MHz IMB clock, to prefetch the next transmit RAM entry for the next transfer. Therefore, the user may select a baud rate that provides at least a 0.6-µs delay between successive transfers to ensure no loss of incoming data. If the IMB clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Because the BITSE option in the command control segment is no longer available, BITS sets the number of bits to be transferred for all transfers in the queue until the CPU changes the BITS value. As mentioned above, until PCS[0]/$\overline{SS}$ is negated (brought high), the QSPI continues to shift one bit for each pulse of SCK. If PCS[0]/$\overline{SS}$ is negated before the proper number of bits (according to BITS) is received, the next time the QSPI is selected it resumes storing bits in the same receive-data segment address where it left off. If more than 16 bits are transferred before negating the PCS[0]/$\overline{SS}$, the QSPI stores the number of bits indicated by BITS in the current receive data segment address, then increments the address and continues storing as described

above. Note that PCS[0]/$\overline{SS}$ does not necessarily have to be negated between transfers.

Once the proper number of bits (designated by BITS) are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the new transmit data from the transmit data segment into the data serializer. The internal working queue pointer address is used the next time PCS[0]/$\overline{SS}$ is asserted, unless the CPU writes to the NEWQP first.

The DT and DSCK command control bits are not used in slave mode. As a slave, the QSPI does not drive the clock line nor the chip-select lines and, therefore, does not generate a delay.

In slave mode, the QSPI shifts out the data in the transmit data segment. The transmit data is loaded into the data serializer (refer to **Figure 14-1**) for transmission. When the PCS[0]/$\overline{SS}$ pin is pulled low the MISO pin becomes active and the serializer then shifts the 16 bits of data out in sequence, most significant bit first, as clocked by the incoming SCK signal. The QSPI uses CPHA and CPOL to determine which incoming SCK edge the MOSI pin uses to latch incoming data, and which edge the MISO pin uses to drive the data out.

The QSPI transmits and receives data until reaching the end of the queue (defined as a match with the address in ENDQP), regardless of whether PCS[0]/$\overline{SS}$ remains selected or is toggled between serial transfers. Receiving the proper number of bits causes the received data to be stored. The QSPI always transmits as many bits as it receives at each queue address, until the BITS value is reached or PCS[0]/$\overline{SS}$ is negated.

14.7.7 Slave Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag, whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is gen-erated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE bit can be found in 4.3.3 QSPI Control Register 2 (SPCR2).

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF, it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE.

As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR with SPIF asserted, followed by a write to SPSR with zero in SPIF (clear SPIF). Execution continues in wraparound mode even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer is incremented to the next address and the commands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data located in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wraparound mode by clearing WREN. The next time end of the queue is reached, the QSPI sets SPIF, clears SPE, and stops; and, b) The CPU sets HALT. This second method halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended, as it causes the QSPI to abort a serial transfer in process.

14.7.8 Mode Fault

MODF is asserted by the QSPI when the QSPI is the serial master (MSTR = 1) and the slave select (PCS[0]/$\overline{SS}$) input pin is pulled low by an external driver. This is possible only if the PCS[0]/$\overline{SS}$ pin is configured as input by QDDR. This low input to $\overline{SS}$ is not a normal operating condition. It indicates that a multimaster system conflict may exist, that another MCU is requesting to become the SPI network master, or simply that the hardware is incorrectly affecting PCS[0]/$\overline{SS}$. SPE in SPCR1 is cleared, disabling the QSPI. The QSPI pins revert to control by QPDR. If MODF is set and HMIE in SPCR3 is asserted, the QSPI generates an interrupt to the CPU.

The CPU may clear MODF by reading SPSR with MODF asserted, followed by writing SPSR with a zero in MODF. After correcting the mode fault problem, the QSPI can be re-enabled by asserting SPE.

The PCS[0]/$\overline{SS}$ pin may be configured as a general-purpose output instead of input to the QSPI. This inhibits the mode fault checking function. In this case, MODF is not used by the QSPI.

14.8 Serial Communication Interface

The dual, independent, serial communication interface (DSCI) communicates with external devices through an asynchronous serial bus. The two SCI modules are functionally equivalent, except that the SCI1 also provides 16-deep queue capabilities for the transmit and receive operations. The SCIs are fully compatible with other Motorola SCI systems. The DSCI has all of the capabilities of previous SCI systems as well as several significant new features.

Figure 14-12 is a block diagram of the SCI transmitter. **Figure 14-13** is a block diagram of the SCI receiver.

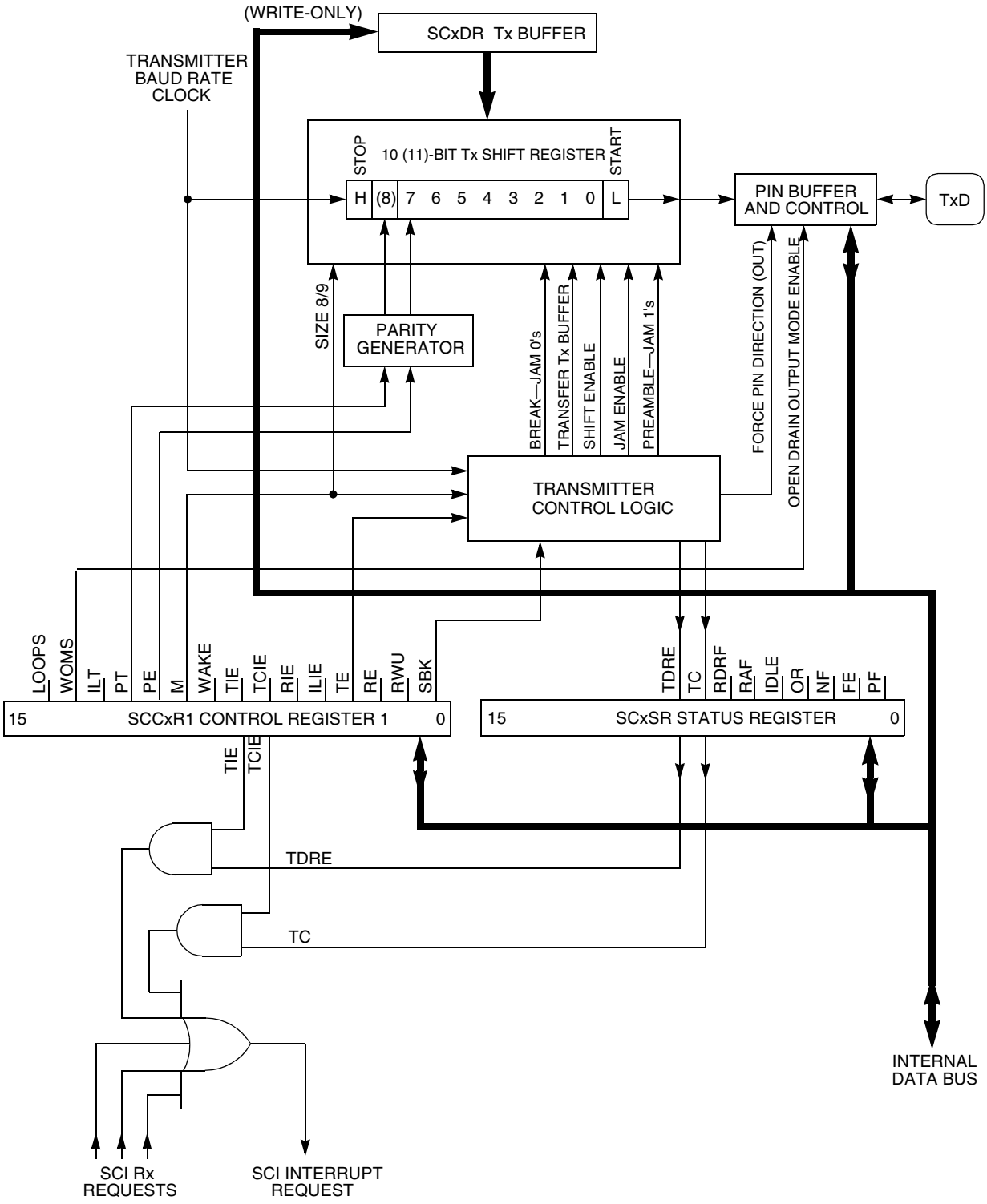


Figure 14-12 SCI Transmitter Block Diagram

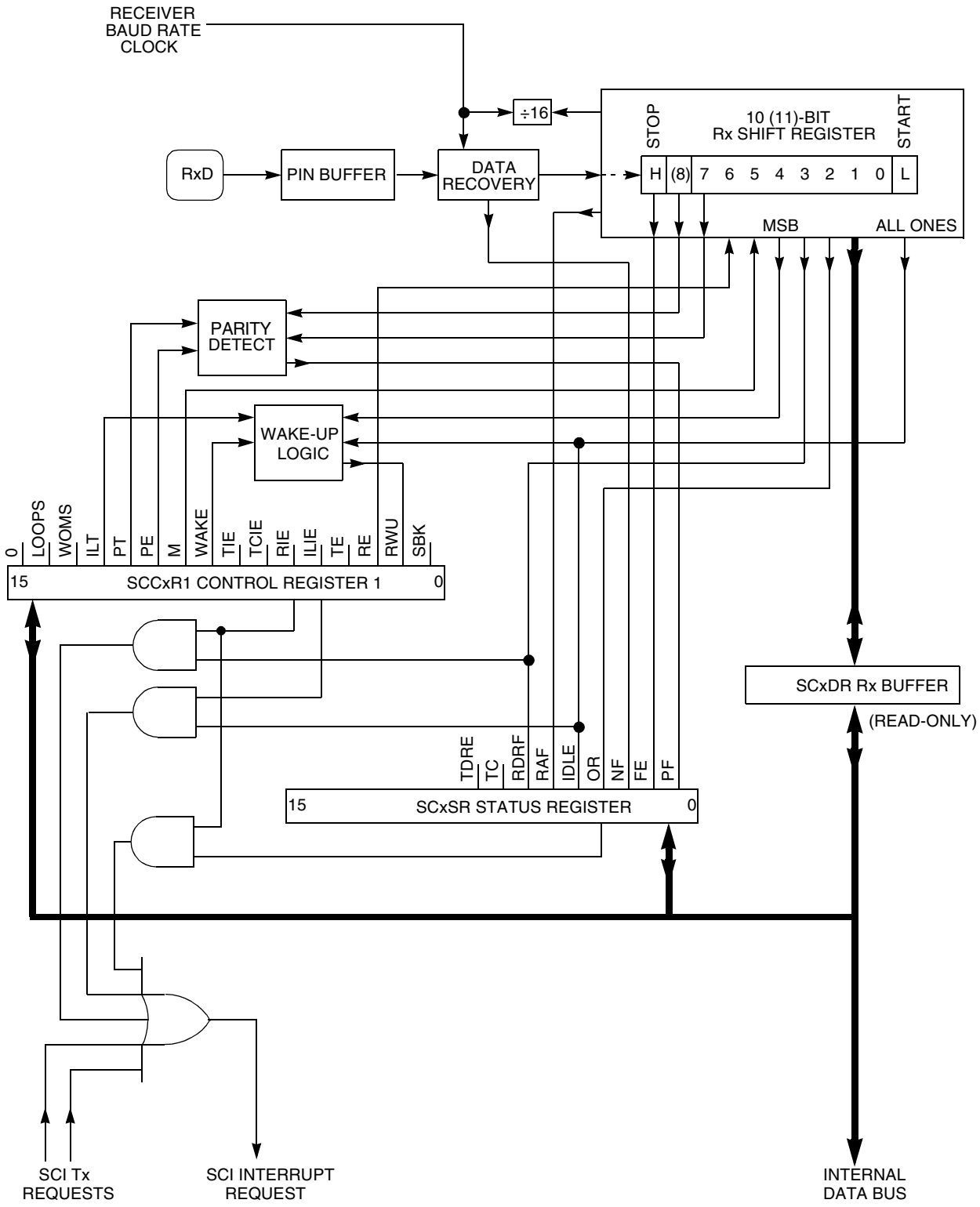


Figure 14-13 SCI Receiver Block Diagram

14.8.1 SCI Registers

The SCI programming model includes the QSMCM global and pin control registers and the DSCI registers.

The DSCI registers, listed in **Table 14-22**, consist of five control registers, three status registers, and 34 data registers. All registers may be read or written at any time by the CPU. Rewriting the same value to any DSCI register does not disrupt operation; however, writing a different value into a DSCI register when the DSCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in register SCxSR can be cleared at any time.

Table 14-22 SCI Registers

Address	Name	Usage
0x30 5008	SCC1R0	SCI1 Control Register 0 See **Table 14-23** for bit descriptions.
0x30 500A	SCC1R1	SCI1 Control Register 1 See **Table 14-24** for bit descriptions.
0x30 500C	SC1SR	SCI1 Status Register See **Table 14-25** for bit descriptions.
0x30 500E (non-queue mode only)	SC1DR	SCI1 Data Register Transmit Data Register (TDR1)* Receive Data Register (RDR1)* See **Table 14-26** for bit descriptions.
0x30 5020	SCC2R0	SCI2 Control Register 0
0x30 5022	SCC2R1	SCI2 Control Register 1
0x30 5024	SC2SR	SCI2 Status Register
0x30 5026	SC2DR	SCI2 Data Register Transmit Data Register (TDR2)* Receive Data Register (RDR2)*
0x30 5028	QSCI1CR	QSCI1 Control Register Interrupts, wrap, queue size and enables for receive and transmit, QTPNT. See **Table 14-30** for bit descriptions.
0x30 502A	QSCI1SR	QSCI1 Status Register OverRun error flag, queue status flags, QRPNT, and QPEND. See **Table 14-31** for bit descriptions.
0x30 502C — 0x30 504A	QSCI1 Transmit Queue Memory Area	QSCI1 Transmit Queue Data locations (on half-word boundary)
0x30 504C-6A	QSCI1 Receive Queue Memory Area	QSCI1 Receive Queue Data locations (on half-word boundary)

*Reads access the RDRx; writes access the TDRx.

During SCIx initialization, two bits in the SCCxR1 should be written last: the transmitter enable (TE) and receiver enable (RE) bits, which enable SCIx. Registers SCCxR0 and SCCxR1 should both be initialized at the same time or before TE and RE are asserted. A single half-word write to SCCxR1 can be used to initialize SCIx and enable the transmitter and receiver.

14.8.2 SCI Control Register 0

SCCxR0 contains the SCIx baud rate selection field and two bits controlling the clock source. The baud rate must be set before the SCI is enabled. The CPU can read and write SCCxR0 at any time.

Changing the value of SCCxR0 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR0 — SCI Control Register 0 **0x30 5008**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
OTHR	LNK-BD	0	SCxBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 14-23 SCCxR0 Bit Descriptions

Bit(s)	Name	Description
0	OTHR	This bit is reserved and should always be programmed to 0.
1	LNKBD	This bit is reserved and should always be programmed to 0.
2	—	Reserved
3:15	SCxBR	SCI baud rate. The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCxBR disables the baud rate generator. Baud clock rate is calculated as follows: $\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$ where SCxBR is in the range of 1 to 8191. Refer to **14.8.7.3 Baud Clock** for more information.

14.8.3 SCI Control Register 1

SCCxR1 contains SCIx configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read or write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCCxR1 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR1 — SCI Control Register 1 **0x30 500A, 0x30 5022**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-24 SCCxR1 Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	LOOPS	Loop mode 0 = Normal SCI operation, no looping, feedback path disabled. 1 = SCI test operation, looping, feedback path enabled.
2	WOMS	Wired-OR mode for SCI Pins 0 = If configured as an output, TXD is a normal CMOS output. 1 = If configured as an output, TXD is an open drain output.
3	ILT	Idle-line detect type. Refer to **14.8.7.8 Idle-Line Detection**. 0 = Short idle-line detect (start count on first one). 1 = Long idle-line detect (start count on first one after stop bit(s)).
4	PT	Parity type. Refer to **14.8.7.4 Parity Checking**. 0 = Even parity. 1 = Odd parity.
5	PE	Parity enable. Refer to **14.8.7.4 Parity Checking**. 0 = SCI parity disabled. 1 = SCI parity enabled.
6	M	Mode select. Refer to **14.8.7.2 Serial Formats**. 0 = 10-bit SCI frame. 1 = 11-bit SCI frame.
7	WAKE	Wakeup by address mark. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = SCI receiver awakened by idle-line detection. 1 = SCI receiver awakened by address mark (last bit set).
8	TIE	Transmit interrupt enable 0 = SCI TDRE interrupts disabled. 1 = SCI TDRE interrupts enabled.
9	TCIE	Transmit complete interrupt enable 0 = SCI TC interrupts disabled. 1 = SCI TC interrupts enabled.
10	RIE	Receiver interrupt enable 0 = SCI RDRF and OR interrupts disabled. 1 = SCI RDRF and OR interrupts enabled.
11	ILIE	Idle-line interrupt enable 0 = SCI IDLE interrupts disabled. 1 = SCI IDLE interrupts enabled.
12	TE	Transmitter enable 0 = SCI transmitter disabled (TXD pin can be used as general-purpose output) 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).
13	RE	Receiver Enable 0 = SCI receiver disabled (RXD pin can be used as general-purpose input). 1 = SCI receiver enabled (RXD pin is dedicated to SCI receiver).

Table 14-24 SCCxR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
14	RWU	Receiver wakeup. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = Normal receiver operation (received data recognized). 1 = Wakeup mode enabled (received data ignored until receiver is awakened).
15	SBK	Send break 0 = Normal operation. 1 = Break frame(s) transmitted after completion of current frame.

14.8.4 SCI Status Register (SCxSR)

SCxSR contains flags that show SCI operating conditions. These flags are cleared either by SCIx hardware or by a read/write sequence. The sequence consists of reading the SCxSR (either the upper byte, lower byte, or the entire half-word) with a flag bit set, then reading (or writing, in the case of flags TDRE and TC) the SCxDR (either the lower byte or the half-word).

The contents of the two 16-bit registers SCxSR and SCxDR appear as upper and lower half-words, respectively, when the SCxSR is read into a 32-bit register. An upper byte access of SCxSR is meaningful only for reads. Note that a word read can simultaneously access both registers SCxSR and SCxDR. This action clears the receive status flag bits that were set at the time of the read, but does not clear the TDRE or TC flags. To clear TC, the SCxSR read must be followed by a write to register SCxDR (either the lower byte or the half-word). The TDRE flag in the status register is read-only.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits but before the CPU has read or written the SCxDR, the newly set status bit is not cleared. Instead, SCxSR must be read again with the bit set and SCxDR must be read or written before the status bit is cleared.

NOTE

None of the status bits are cleared by reading a status bit while it is set and then writing zero to that same bit. Instead, the procedure outlined above must be followed. Note further that reading either byte of SCxSR causes all 16 bits to be accessed, and any status bits already set in either byte are armed to clear on a subsequent read or write of SCxDR.

SCxSR — SCIx Status Register **0x30 500C, 0x30 5024**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED							TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:															
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 14-25 SCxSR Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	TDRE	Transmit data register empty. TDRE is set when the byte in TDRx is transferred to the transmit serial shifter. If this bit is zero, the transfer is yet to occur and a write to TDRx will overwrite the previous value. New data is not transmitted if TDRx is written without first clearing TDRE. 0 = Transmit data register still contains data to be sent to the transmit serial shifter. 1 = A new character can now be written to the transmit data register. For transmit queue operation, this bit should be ignored by software.
8	TC	Transmit complete. TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle-line), or queued breaks (logic zero). 0 = SCI transmitter is busy. 1 = SCI transmitter is idle. For transmit queue operation, TC is cleared when SCxSR is read with TC set, followed by a write to SCTQ[0:15].
9	RDRF	Receive data register full. RDRF is set when the contents of the receive serial shifter are transferred to register RDRx. If one or more errors are detected in the received word, the appropriate flag(s) (NF, FE, or PF) are set within the same clock cycle. 0 = Receive data register is empty or contains previously read data. 1 = Receive data register contains new data. For receiver queue operation, this bit should be ignored by software.
10	RAF	Receiver active flag. RAF indicates whether the receiver is busy. This flag is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters. 0 = SCI receiver is idle. 1 = SCI receiver is busy.
11	IDLE	Idle line detected. IDLE is set when the receiver detects an idle-line condition (reception of a minimum of 10 or 11 consecutive ones as specified by ILT in SCCxR1). This bit is not set by the idle-line condition when RWU in SCCxR1 is set. Once cleared, IDLE is not set again until after RDRF is set (after the line is active and becomes idle again). If a break is received, RDRF is set, allowing a subsequent idle line to be detected again. Under certain conditions, the IDLE flag may be set immediately following the negation of RE in SCCxR1. System designs should ensure this causes no detrimental effects. 0 = SCI receiver did not detect an idle-line condition. 1 = SCI receiver detected an idle-line condition. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15].
12	OR	Overrun error. OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDRx, and RDRx is already full (RDRF is still set). Data transfer is inhibited until OR is cleared. Previous data in RDRx remains valid, but additional data received during an overrun condition (including the byte that set OR) is lost. Note that whereas the other receiver status flags (NF, FE, and PF) reflect the status of data already transferred to RDRx, the OR flag reflects an operational condition that resulted in a loss of data to RDRx. 0 = RDRF is cleared before new data arrives. 1 = RDRF is not cleared before new data arrives.

Table 14-25 SCxSR Bit Descriptions (Continued)

Bit(s)	Name	Description
13	NF	Noise error flag. NF is set when the receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit(s). It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times for noise. If the three samples are not at the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with NF, an interrupt can be generated with RDRF, and the interrupt handler can check NF. 0 = No noise detected in the received data. 1 = Noise detected in the received data. For receiver queue operation NF is cleared when SCxSR is read with NF set, followed by a read of SCRQ[0:15].
14	FE	Framing error. FE is set when the receiver detects a zero where a stop bit (one) was expected. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. FE is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with FE, an interrupt can be generated with RDRF, and the interrupt handler can check FE. 0 = No framing error detected in the received data. 1 = Framing error or break detected in the received data.
15	PF	Parity error. PF is set when the receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with PF, an interrupt can be generated with RDRF, and the interrupt handler can check PF. 0 = No parity error detected in the received data. 1 = Parity error detected in the received data.

14.8.5 SCI Data Register (SCxDR)

The SCxDR consists of two data registers located at the same address. The receive data register (RDRx) is a read-only register that contains data received by the SCI serial interface. Data is shifted into the receive serial shifter and is transferred to RDRx. The transmit data register (TDRx) is a write-only register that contains data to be transmitted. Data is first written to TDRx, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

SCxDR — SCI Data Register **0x30 500E, 0x30 5026**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED							R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

Table 14-26 SCxSR Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7:15	R[8:0]/ T[8:0]	R[7:0]/T[7:0] contain either the eight data bits received when SCxDR is read, or the eight data bits to be transmitted when SCxDR is written. R8/T8 are used when the SCI is configured for nine-bit operation (M = 1). When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect. Accesses to the lower byte of SCxDR triggers the mechanism for clearing the status bits or for initiating transmissions whether byte, half-word, or word accesses are used.

14.8.6 SCI Pins

The RXD1 and RXD2 pins are the receive data pins for the SCI1 and SCI2, respectively. TXD1 and TXD2 are the transmit data pins for the two SCI modules. An external clock pin, ECK, is common to both SCIs. The pins and their functions are listed in **Table 14-27**.

Table 14-27 SCI Pin Functions

Pin Names	Mnemonic	Mode	Function
Receive Data	RXD1, RXD2	Receiver disabled Receiver enabled	General purpose input Serial data input to SCI
Transmit Data	TXD1, TXD2	Transmitter disabled Transmitter enabled	General purpose output Serial data output from SCI
External Clock	ECK	Receiver disabled Receiver enabled Transmitter disabled Transmitter enabled	Not used Alternate input source to baud Not used Alternate input source to baud

14.8.7 SCI Operation

The SCI can operate in polled or interrupt-driven mode. Status flags in SCxSR reflect SCI conditions regardless of the operating mode chosen. The TIE, TCIE, RIE, and ILIE bits in SCCxR1 enable interrupts for the conditions indicated by the TDRE, TC, RDRF, and IDLE bits in SCxSR, respectively.

14.8.7.1 Definition of Terms

- Bit-time — The time required to transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- Start bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time samples of logic one.
- Stop bit— One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle frame — A frame that consists of consecutive ones. An idle frame has no start bit.

- Break frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

14.8.7.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both 10-bit and 11-bit frames. The M bit in SCCxR1 specifies the number of bits per frame.

The most common data frame format for NRZ (non-return to zero) serial interfaces is one start bit, eight data bits (LSB first), and one stop bit (ten bits total). The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and 11-bit frames are shown in **Table 14-28**.

Table 14-28 Serial Frame Formats

10-bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

14.8.7.3 Baud Clock

The SCI baud rate is programmed by writing a 13-bit value to the SCxBR field in SCI control register zero (SCCxR0). The baud rate is derived from the MCU IMB clock by a modulus counter. Writing a value of zero to SCxBR[12:0] disables the baud rate generator. The baud rate is calculated as follows:

$$\textit{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \textit{SCxBR}}$$

or

$$\textit{SCxBR} = \frac{f_{SYS}}{32 \times \textit{SCI Baud Rate Desired}}$$

where SCxBR is in the range {1, 2, 3, ..., 8191}.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receive time sampling clock with a frequency 16 times that of the SCI baud rate. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period.

Table 14-29 shows possible baud rates for a 40-MHz IMB clock. The maximum baud rate with this IMB clock speed is 1250 Kbaud.

Table 14-29 Examples of SCIx Baud Rates[1]

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCxBR
1,250,000.00	1,250,000.00	0.00	1
57,600.00	56,818.18	-1.36	22
38,400.00	37,878.79	-1.36	33
32,768.00	32,894.74	0.39	38
28,800.00	29,069.77	0.94	43
19,200.00	19,230.77	0.16	65
14,400.00	14,367.81	-0.22	87
9,600.00	9,615.38	0.16	130
4,800.00	4,807.69	0.16	260
2,400.00	2,399.23	-0.03	521
1,200.00	1,199.62	-0.03	1042
600.00	600.09	0.02	2083
300.00	299.98	-0.01	4167

NOTES:
1. These rates are based on a 40-MHz IMB clock.

14.8.7.4 Parity Checking

The PT bit in SCCxR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCxR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame (i.e., the bit preceding the stop bit) is used for the parity function. For transmitted data, a parity bit is generated. For received data, the parity bit is checked. When parity checking is enabled, the PF bit in the SCI status register (SCxSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 14-24** shows possible data and parity formats.

14.8.7.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDRx while other data is shifted out. The TE bit in SCCxR1 enables (TE = 1) and disables (TE = 0) the transmitter.

The shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in

SCCxR1 determines whether TXD is an open drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function, regardless of whether the pin is used by the SCI or as a general-purpose output pin.

Data to be transmitted is written to SCxDR, then transferred to the serial shifter. Before writing to TDRx, the user should check the transmit data register empty (TDRE) flag in SCxSR. When TDRE = 0, the TDRx contains data that has not been transferred to the shifter. Writing to SCxDR again overwrites the data. If TDRE = 1, then TDRx is empty, and new data may be written to TDRx, clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new data frame is in TDRx (TDRE = 0), then the new data is transferred from TDRx to the transmit serial shifter and TDRE is set automatically. An interrupt may optionally be generated at this point.

The transmission complete (TC) flag in SCxSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCxSR while TC is set, then writing new data to SCxDR, or writing to SCTQ[0:15] for transmit queue operation.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The SBK bit in SCCxR1 is used to insert break frames in a transmission. A non-zero integer number of break frames are transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To ensure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

When TE is cleared, the transmitter is disabled after all pending idle, data, and break frames are transmitted. The TC flag is set, and control of the TXD pin reverts to PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output, then write a one to either QDTX1 or QDTX2 of the PORTQS register. See **14.6.1**. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place non-listening receivers in wake-up mode between transmissions, or to signal a re-transmission by forcing an idle-line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter marks idle. Otherwise, normal transmission of the next sequence begins.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCxR1. Service routines can load the last data frame in a sequence into SCxDR, then terminate the transmission when a TDRE interrupt occurs.

Two SCI messages can be separated with minimum idle time by using a preamble of 10 bit-times (11 if a 9-bit data format is specified) of marks (logic ones). Follow these steps:

1. Write the last data frame of the first message to the TDRx
2. Wait for TDRE to go high, indicating that the last data frame is transferred to the transmit serial shifter
3. Clear TE and then set TE back to one. This queues the preamble to follow the stop bit of the current transmission immediately.
4. Write the first data frame of the second message to register TDRx

In this sequence, if the first data frame of the second message is not transferred to TDRx prior to the finish of the preamble transmission, then the transmit data line (TXDx pin) marks idle (logic one) until TDRx is written. In addition, if the last data frame of the first message finishes shifting out (including the stop bit) and TE is clear, TC goes high and transmission is considered complete. The TXDx pin reverts to being a general-purpose output pin.

14.8.7.6 Receiver Operation

The receiver can be divided into two segments. The first is the receiver bit processor logic that synchronizes to the asynchronous receive data and evaluates the logic sense of each bit in the serial stream. The second receiver segment controls the functional operation and the interface to the CPU including the conversion of the serial data stream to parallel access by the CPU.

Receiver Bit Processor — The receiver bit processor contains logic to synchronize the bit-time of the incom-ing data and to evaluate the logic sense of each bit. To accomplish this an RT clock, which is 16 times the baud rate, is used to sample each bit. Each bit-time can thus be divided into 16 time periods called RT1–RT16. The receiver looks for a possible start bit by watching for a high-to-low transition on the RXDx pin and by assigning the RT time labels appropriately.

When the receiver is enabled by writing RE in SCCxR1 to one, the receiver bit processor logic begins an asynchronous search for a start bit. The goal of this search is to gain synchronization with a frame. The bit-time synchronization is done at the beginning of each frame so that small differences in the baud rate of the receiver and transmitter are not cumulative. SCIx also synchronizes on all one-to-zero transitions

in the serial data stream, which makes SCIx tolerant to small frequency variations in the received data stream.

The sequence of events used by the receiver to find a start bit is listed below.

1. Sample RXDx input during each RT period and maintain these samples in a serial pipeline that is three RT periods deep.
2. If RXDx is low during this RT period, go to step 1.
3. If RXDx is high during this RT period, store sample and proceed to step 4.
4. If RXDx is low during this RT period, but not high for the previous three RT periods (which is noise only), set an internal working noise flag and go to step 1, since this transition was not a valid start bit transition.
5. If RXDx is low during this RT period and has been high for the previous three RT periods, call this period RT1, set RAF, and proceed to step 6.
6. Skip RT2 but place RT3 in the pipeline and proceed to step 7.
7. Skip RT4 and sample RT5. If both RT3 and RT5 are high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT5 in the pipeline and proceed to step 8.
8. Skip RT6 and sample RT7. If any two of RT3, RT5, or RT7 is high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT7 in the pipeline and proceed to step 9.
9. A valid start bit is found and synchronization is achieved. From this point on until the end of the frame, the RT clock will increment starting over again with RT1 on each one-to-zero transition or each RT16. The beginning of a bit-time is thus defined as RT1 and the end of a bit-time as RT16.

Upon detection of a valid start bit, synchronization is established and is maintained through the reception of the last stop bit, after which the procedure starts all over again to search for a new valid start bit. During a frame's reception, SCIx resynchronizes the RT clock on any one-to-zero transitions.

Additional logic in the receiver bit processor determines the logic level of the re-ceived bit and implements an advanced noise-detection function. During each bit-time of a frame (including the start and stop bits), three logic-sense samples are taken at RT8, RT9, and RT10. The logic sense of the bit-time is decided by a majority vote of these three samples. This logic level is shifted into register RDRx for every bit except the start and stop bits.

If RT8, RT9, and RT10 do not all agree, an internal working noise flag is set. Additionally for the start bit, if RT3, RT5, and RT7 do not all agree, the internal working noise flag is set. If this flag is set for any of the bit-times in a frame, the NF flag in SCxSR is set concurrently with the RDRF flag in SCxSR when the data is transferred to register RDRx. The user must determine if the data received with NF set is valid. Noise on the RXDx pin does not necessarily corrupt all data.

The operation of the receiver bit processor is shown in **Figure 14-14**. This example demonstrates the search for a valid start bit and the synchronization procedure as outlined above. The possibilities of noise durations greater than one bit-time are not considered in this examples.

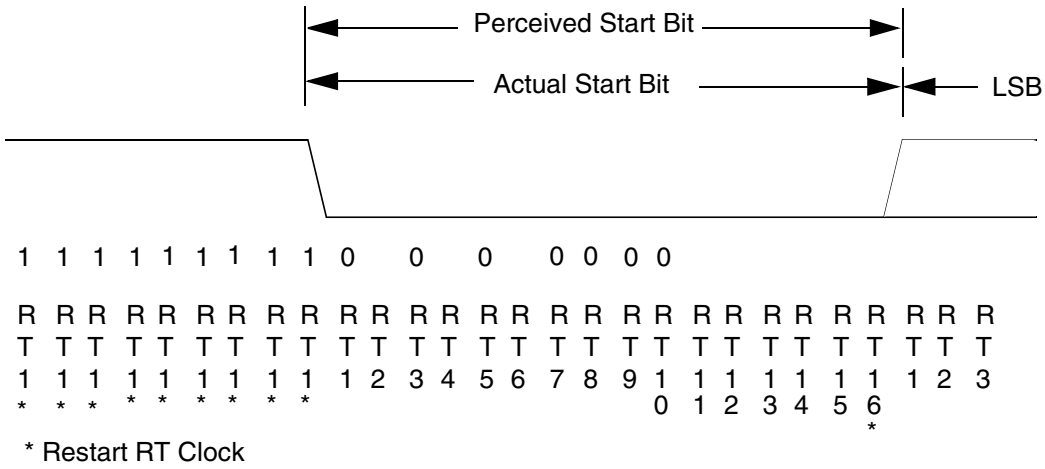


Figure 14-14 Start Search Example

14.8.7.7 Receiver Functional Operation

The RE bit in SCCxR1 enables (RE = 1) and disables (RE = 0) the receiver. The receiver contains a receive serial shifter and a parallel receive data register (RDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in the RDRx while other data is shifted in.

Receiver bit processor logic drives a state machine that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. A receive time clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the receive time clock with the incoming data stream. From this point on, data movement is synchronized with the MCU IMB clock. Operation of the receiver state machine is detailed in the ***Queued Serial Module Reference Manual*** **(QSMRM/AD)**.

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDRx. The receiver data register flag (RDRF) is set when the data is transferred.

The stop bit is always a logic one. If a logic zero is sensed during this bit-time, the FE flag in SCxSR is set. A framing error is usually caused by mismatched baud rates between the receiver and transmitter or by a significant burst of noise. Note that a framing error is not always detected; the data in the expected stop bit-time may happen to be a logic one.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise

flag (NF), the parity flag (PF), and the framing error (FE) flag in SCxSR are not set until data is transferred from the serial shifter to the RDRx.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in SCxSR is set. OR indicates that the RDRx needs to be serviced faster. When OR is set, the data in the RDRx is preserved, but the data in the serial shifter is lost.

When a completed frame is received into the RDRx, either the RDRF or OR flag is always set. If RIE in SCCxR1 is set, an interrupt results whenever RDRF is set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF, as appropriate. These receiver flags are never set with OR because the flags apply only to the data in the receive serial shifter. The receiver status flags do not have separate interrupt enables, since they are set simultaneously with RDRF and must be read by the user at the same time as RDRF.

When the CPU reads SCxSR and SCxDR in sequence, it acquires status and data, and also clears the status flags. Reading SCxSR acquires status and arms the clearing mechanism. Reading SCxDR acquires data and clears SCxSR.

14.8.7.8 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronically and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCxR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle-line type (ILT) bit in SCCxR1 determines which type of detection is used. When an idle-line condition is detected, the IDLE flag in SCxSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle-line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, software overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the ILIE bit in SCCxR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCxSR and SCxDR in sequence. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15]. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

14.8.7.9 Receiver Wake-Up

The receiver wake-up function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCxR1. While RWU is set, receiver status flags and interrupts are disabled. Although the software can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCxR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up allows a receiver to sleep until an idle line is detected. When an idle line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address mark wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

14.8.7.10 Internal Loop Mode

The LOOPS bit in SCCxR1 controls a feedback path in the data serial shifter. When LOOPS is set, the SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

14.9 SCI Queue Operation

14.9.1 Queue Operation of SCI1 for Transmit and Receive

The SCI1 serial module allows for queueing on transmit and receive data frames. In the standard mode, in which the queue is disabled, the SCI1 operates as previously defined (i.e. transmit and receive operations done via SC1DR). However, if the SCI1 queue feature is enabled (by setting the QTE and/or QRE bits within QSCI1CR) a set of 16 entry queues is allocated for the receive and/or transmit operation. Through soft-

ware control the queue is capable of continuous receive and transfer operations within the SCI1 serial unit.

14.9.2 Queued SCI1 Status and Control Registers

The SCI1 queue uses the following registers:

- QSCI1 control register (QSCI1CR, address offset 0x28)
- QSCI1 status register (QSCI1SR, address offset 0x2A)

14.9.2.1 QSCI1 Control Register

QSCI1CR — QSCI1 Control Register **0x30 5028**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QTPNT				QTH-FI	QBH-FI	QTHEI	QB-HEI	0	QTE	QRE	QTWE	QTSZ			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-30 QSCI1CR Bit Descriptions

Bit(s)	Name	Description
0:3	QTPNT	Queue transmit pointer. QTPNT is a 4-bit counter used to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This feature allows for ease of testability. This field is writable in test mode only; otherwise it is read-only.
4	QTHFI	Receiver queue top-half full interrupt. When set, QTHFI enables an SCI1 interrupt whenever the QTHF flag in QSCI1SR is set. The interrupt is blocked by negating QTHFI. This bit refers to the queue locations SCRQ[0:7]. 0 = QTHF interrupt inhibited 1 = Queue top-half full (QTHF) interrupt enabled
5	QBHFI	Receiver queue bottom-half full interrupt. When set, QBHFI enables an SCI1 interrupt whenever the QBHF flag in QSCI1SR is set. The interrupt is blocked by negating QBHFI. This bit refers to the queue locations SCRQ[8:15]. 0 = QBHF interrupt inhibited 1 = Queue bottom-half full (QBHF) interrupt enabled
6	QTHEI	Transmitter queue top-half empty interrupt. When set, QTHEI enables an SCI1 interrupt whenever the QTHE flag in QSCI1SR is set. The interrupt is blocked by negating QTHEI. This bit refers to the queue locations SCTQ[0:7]. 0 = QTHE interrupt inhibited 1 = Queue top-half empty (QTHE) interrupt enabled
7	QBHEI	Transmitter queue bottom-half empty interrupt. When set, QBHEI enables an SCI1 interrupt whenever the QBHE flag in QSCI1SR is set. The interrupt is blocked by negating QBHEI. This bit refers to the queue locations SCTQ[8:15]. 0 = QBHE interrupt inhibited 1 = Queue bottom-half empty (QBHE) interrupt enabled
8	—	Reserved
9	QTE	Queue transmit enable. When set, the transmit queue is enabled and the TDRE bit should be ignored by software. The TC bit is redefined to indicate when the entire queue is finished transmitting. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QTE. 0 = Transmit queue is disabled 1 = Transmit queue is enabled
10	QRE	Queue receive enable. When set, the receive queue is enabled and the RDRF bit should be ignored by software. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QRE. 0 = Receive queue is disabled 1 = Receive queue is enabled
11	QTWE	Queue transmit wrap enable. When set, the transmit queue is allowed to restart transmitting from the top of the queue after reaching the bottom of the queue. After each wrap of the queue, QTWE is cleared by hardware. 0 = Transmit queue wrap feature is disabled 1 = Transmit queue wrap feature is enabled
12:15	QTSZ	Queue transfer size. The QTSZ bits allow programming the number of data frames to be transmitted. From 1 (QTSZ = 0b0000) to 16 (QTSZ = 0b1111) data frames can be specified. QTSZ is loaded into QPEND initially or when a wrap occurs.

14.9.2.2 QSCI1 Status Register

QSCI1SR — QSCI1 Status Register **0x30 502A**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED			QOR	QTHF	QBHF	QTHE	QBHE	QRPNT				QPEND			
RESET:															
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Table 14-31 QSCI1SR Bit Descriptions

Bit(s)	Name	Description
0:2	—	Reserved
3	QOR	Receiver queue overrun error. The QOR is set when a new data frame is ready to be transferred from the SC1DR to the queue and the queue is already full (QTHF or QBHF are still set). Data transfer is inhibited until QOR is cleared. Previous data transferred to the queue remains valid. Additional data received during a queue overrun condition is not lost provided the receive queue is re-enabled before OR (SC1SR) is set. The OR flag is set when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost. 0 = The queue is empty before valid data is in the SC1DR 1 = The queue is not empty when valid data is in the SC1DR
4	QTHF	Receiver queue top-half full. QTHF is set when the receive queue locations SCRQ[0:7] are completely filled with new data received via the serial shifter. QTHF is cleared when register QSCI1SR is read with QTHF set, followed by a write of QTHF to zero. 0 = The queue locations SCRQ[0:7] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[0:7] are completely full of newly received data
5	QBHF	Receiver queue bottom-half full. QBHF is set when the receive queue locations SCRQ[8:15] are completely filled with new data received via the serial shifter. QBHF is cleared when register QSCI1SR is read with QBHF set, followed by a write of QBHF to zero. 0 = The queue locations SCRQ[8:15] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[8:15] are completely full of newly received data
6	QTHE	Transmitter queue top-half empty. QTHE is set when all the data frames in the transmit queue locations SCTQ[0:7] have been transferred to the transmit serial shifter. QTHE is cleared when register QSCI1SR is read with QTHE set, followed by a write of QTHE to zero. 0 = The queue locations SCTQ[0:7] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[0:7]
7	QBHE	Transmitter queue bottom-half empty. QBHE is set when all the data frames in the transmit queue locations SCTQ[8:15] has been transferred to the transmit serial shifter. QBHE is cleared when register QSCI1SR is read with QBHE set, followed by a write of QBHE to zero. 0 = The queue locations SCTQ[8:15] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[8:15]
8:11	QRPNT	Queue receive pointer. QRPNT is a 4-bit counter used to indicate the position where the next valid data frame will be stored within the receive queue. This field is writable in test mode only; otherwise it is read-only.
12:15	QPEND	Queue pending. QPEND is a 4-bit decrementer used to indicate the number of data frames in the queue that are awaiting transfer to the SC1DR. This field is writable in test mode only; otherwise it is read-only. From 1 (QPEND = 0b0000) to 16 (or done, QPEND = 1111) data frames can be specified.

14.9.3 QSCI1 Transmitter Block Diagram

The block diagram of the enhancements to the SCI transmitter is shown in **Figure 14-15**.

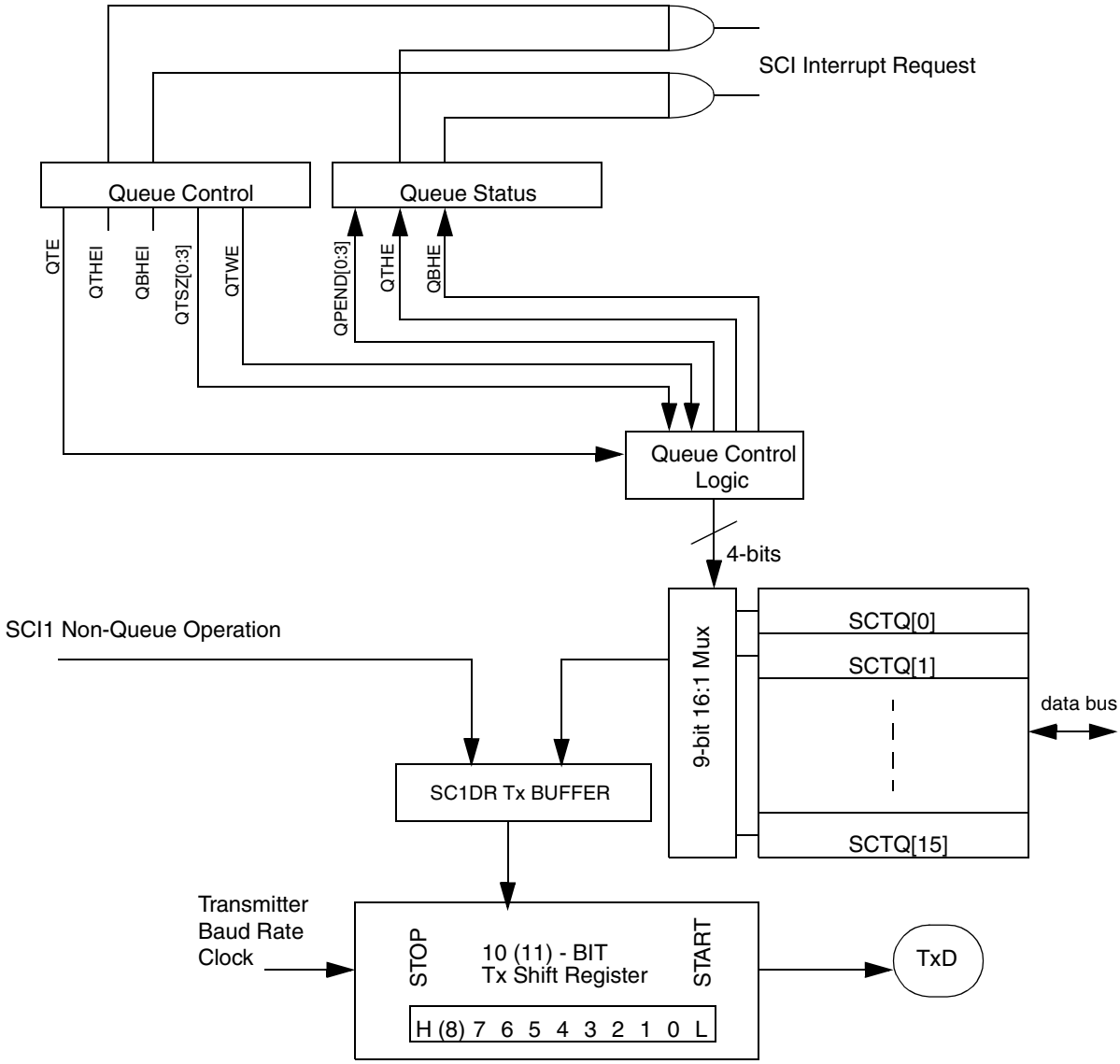


Figure 14-15 Queue Transmitter Block Enhancements

14.9.4 QSCI1 Additional Transmit Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue transmit enable (QTE) bit set by software. When enabled, (QTE = 1) the TDRE bit should be ignored by software and the TC bit is redefined (as described later).
- When the queue is disabled (QTE = 0), the SCI functions in single buffer transfer mode where the queue size is set to one (QTSZ = 0000), and TDRE and TC function as previously defined. Locations SCTQ[0:15] can be used as general purpose 9-bit registers. All other bits pertaining to the queue should be ignored by software.
- Programmable queue up to 16 transmits (SCTQ[0:15]) which may allow for infinite and continuous transmits.
- Available transmit wrap function to prevent message breaks for transmits greater

than 16. This is achieved by the transmit wrap enable (QTWE) bit. When QTWE is set, the hardware is allowed to restart transmitting from the top of the queue (SCTQ[0]). After each wrap, QTWE is cleared by hardware.

— Transmissions of more than 16 data frames must be performed in multiples of 16 (QTSZ = 0b1111) except for the last set of transmissions. For any single non-continuous transmissions of 16 or less or the last transmit set composed of 16 or fewer data frames, the user is allowed to program QTSZ to the corresponding value of 16 or less where QTWE = 0.

- Interrupt generation when the top half (SCTQ[0:7]) of the queue has been emptied (QTHE) and the bottom half (SCTQ[8:15]) of the queue has been emptied (QBHE). This may allow for uninterrupted and continuous transmits by indicating to the CPU that it can begin refilling the queue portion that is now emptied.
  - — The QTHE bit is set by hardware when the top half is empty or the transmission has completed. The QTHE bit is cleared when the QSCI1SR is read with QTHE set, followed by a write of QTHE to zero.
  - — The QBHE bit is set by hardware when the bottom half is empty or the transmission has completed. The QBHE bit is cleared when the QSCI1SR is read with QBHE set, followed by a write of QBHE to zero.
  - — In order to implement the transmit queue, QTE must be set (QSCI1CR), TE must be set (SCC1R1), QTHE must be cleared (QSCI1SR), and TDRE must be set (SC1SR).
- Enable and disable options for the interrupts QTHE and QBHE as controlled by QTHEI and QBHEI respectfully.
- Programmable 4-bit register queue transmit size (QTSZ) for configuring the queue to any size up to 16 transfers at a time. This value may be rewritten after transmission has started to allow for the wrap feature.
- 4-bit status register to indicate the number of data transfers pending (QPEND). This register counts down to all 0's where the next count rolls over to all 1's. This counter is writable in test mode; otherwise it is read-only.
- 4-bit counter (QTPNT) is used as a pointer to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This counter is writable in test mode; otherwise it is read-only.
- A transmit complete (TC) bit re-defined when the queue is enabled (QTE = 1) to indicate when the entire queue (including when wrapped) is finished transmitting. This is indicated when QPEND = 1111 and the shifter has completed shifting data out. TC is cleared when the SCxSR is read with TC = 1 followed by a write to SCTQ[0:15]. If the queue is disabled (QTE = 0), the TC bit operates as originally designed.
- When the transmit queue is enabled (QTE = 1), writes to the transmit data register (SC1DR) have no effect.

14.9.5 QSCI1 Transmit Flow Chart Implementing the Queue

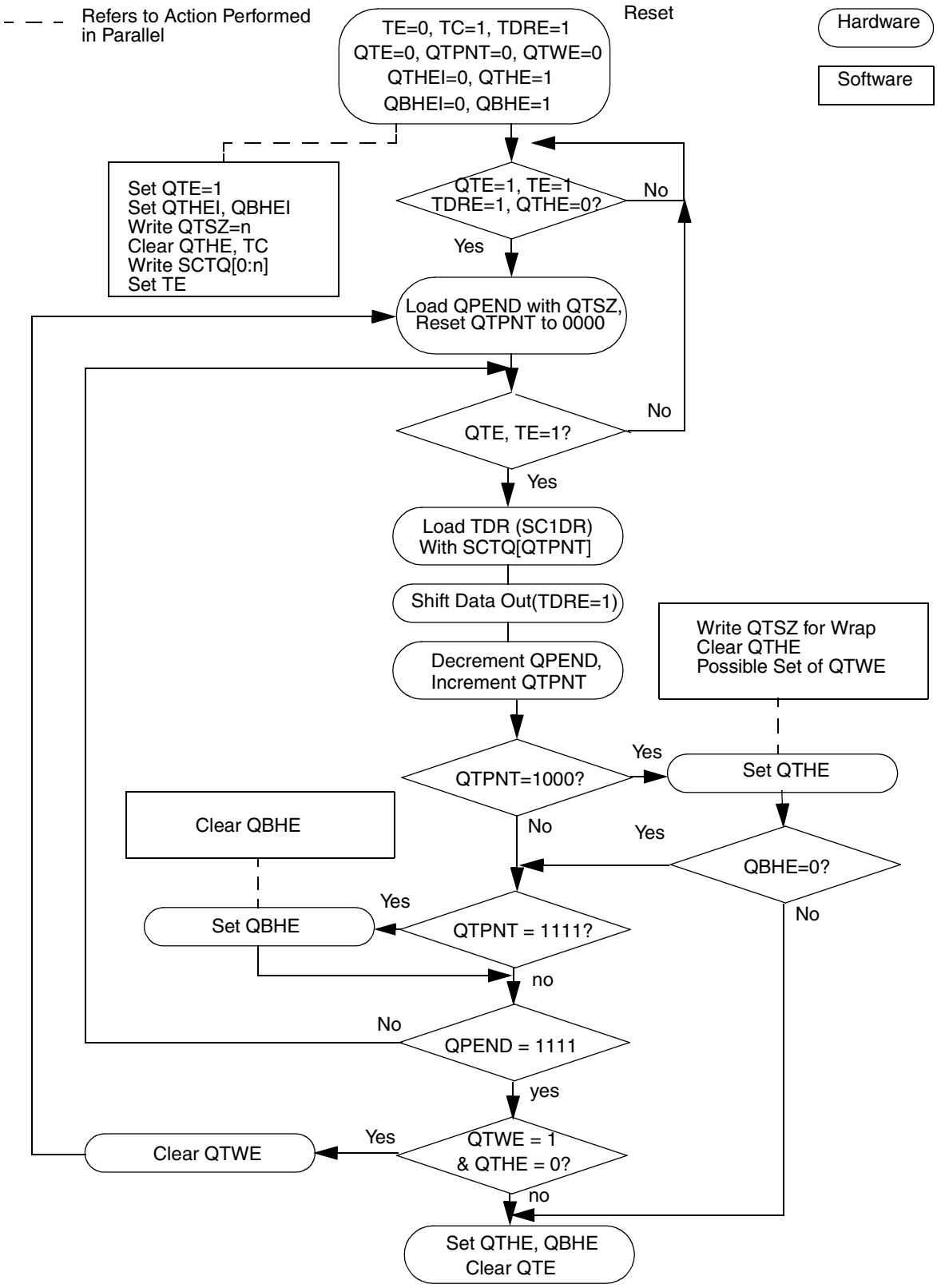


Figure 14-16 Queue Transmit Flow

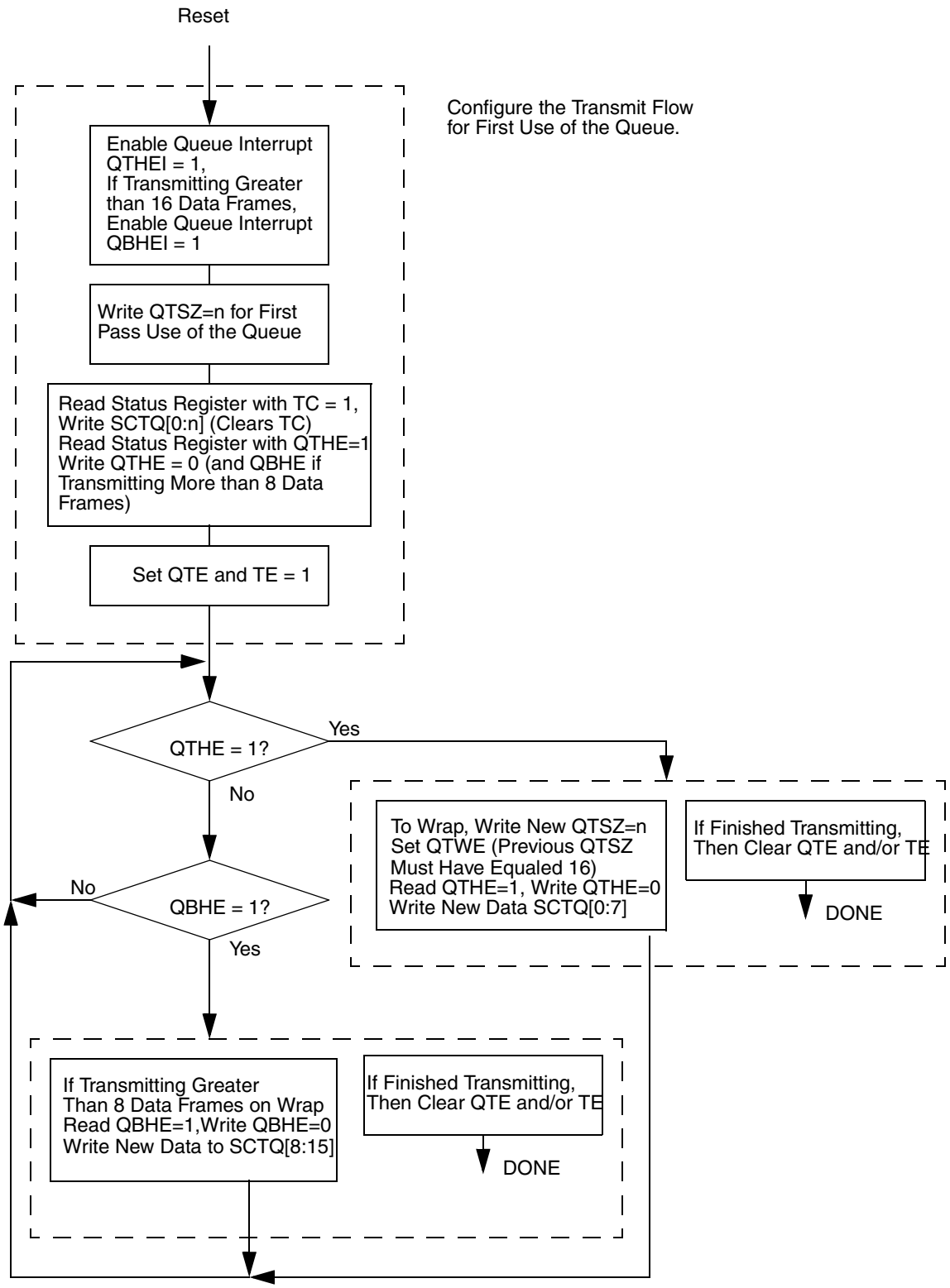


Figure 14-17 Queue Transmit Software Flow

14.9.6 Example QSCI1 Transmit for 17 Data Bytes

Figure 14-18 below shows a transmission of 17 data frames. The bold type indicates the current value for QTPNT and QPEND. The italic type indicates the action just performed by hardware. Regular type indicates the actions that should be performed by software before the next event.

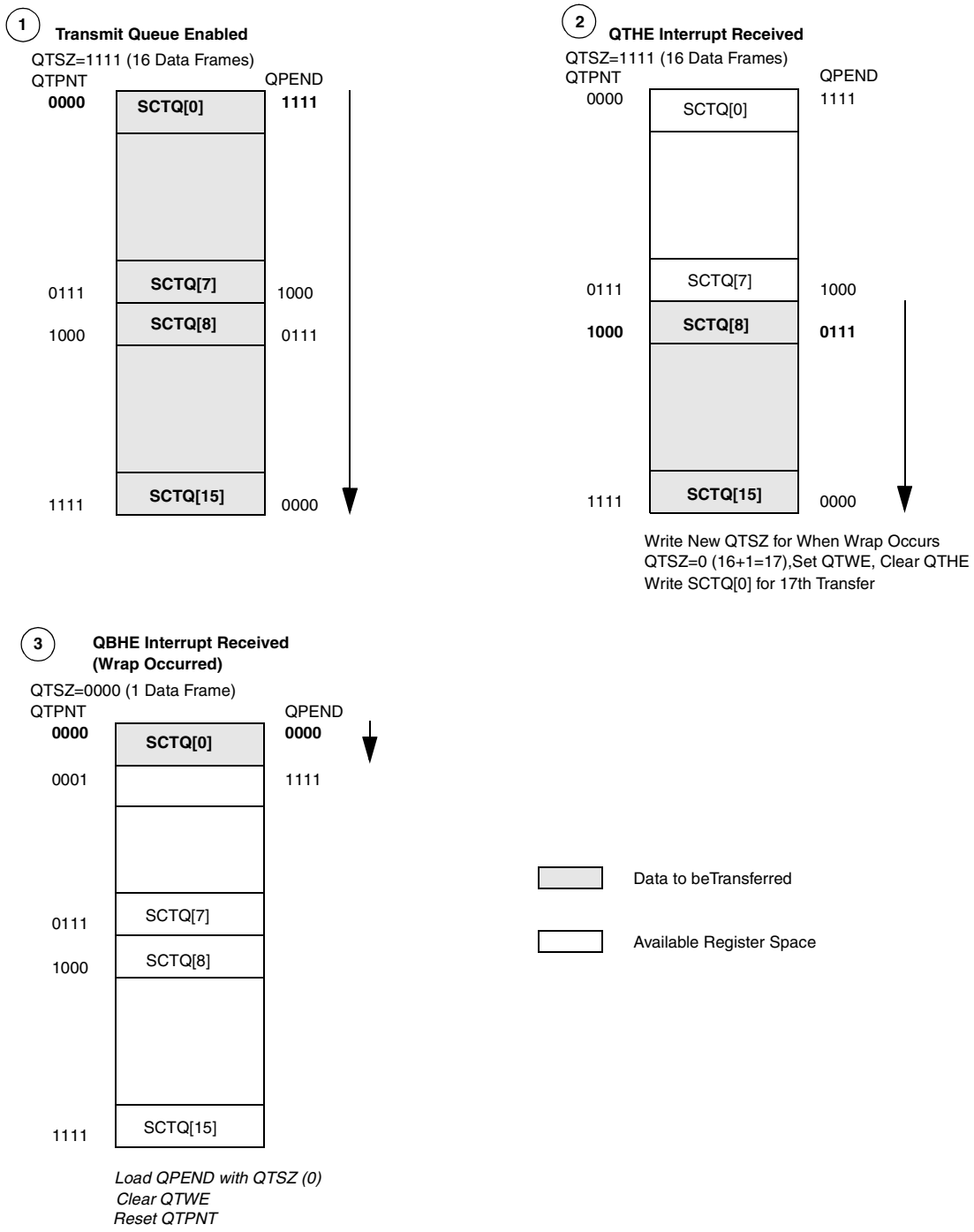


Figure 14-18 Queue Transmit Example for 17 Data Bytes

14.9.7 Example SCI Transmit for 25 Data Bytes

Figure 14-19 below is an example of a transmission of 25 data frames.

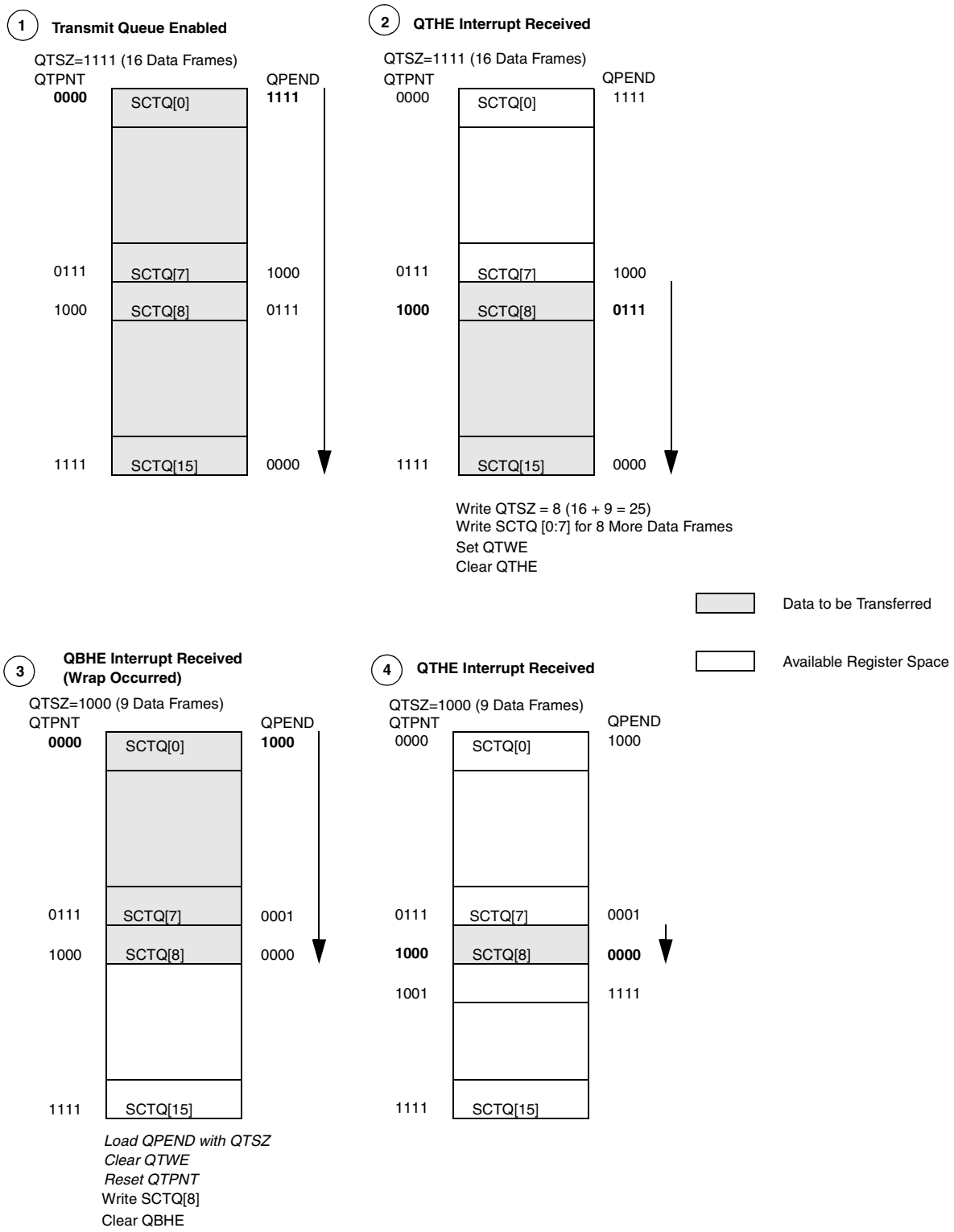


Figure 14-19 Queue Transmit Example for 25 Data Frames

14.9.8 QSCI1 Receiver Block Diagram

The block diagram of the enhancements to the SCI receiver is shown below in **Figure 14-20**.

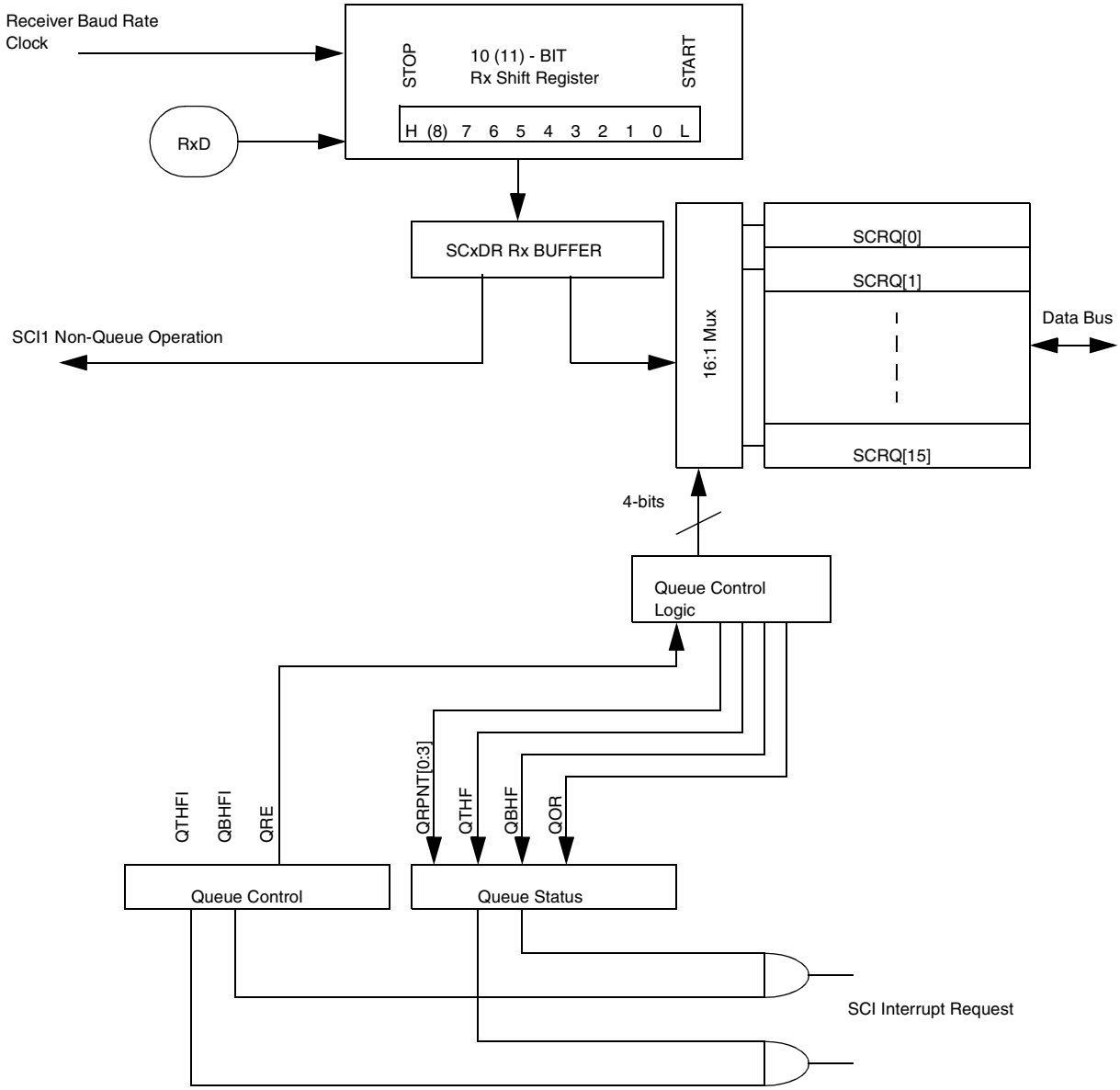


Figure 14-20 Queue Receiver Block Enhancements

14.9.9 QSCI1 Additional Receive Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue receiver enable (QRE) bit set by software. When the queue is enabled, software should ignore the RDRF bit.
- When the queue is disabled (QRE = 0), the SCI functions in single buffer receive mode (as originally designed) and RDRF and OR function as previously defined.

Locations SCRQ[0:15] can be used as general purpose 9-bit registers. Software should ignore all other bits pertaining to the queue.

- Only data that has no errors (FE and PF both false) is allowed into the queue. The status flags FE and PF, if set, reflect the status of data not allowed into the queue. The receive queue is disabled until the error flags are cleared via the original SCI mechanism and the queue is re-initialized. The pointer QRPNT indicates the queue location where the data frame would have been stored.
- Queue size capable to receive up to 16 data frames (SCRQ[0:15]) which may allow for infinite and continuous receives.
- Interrupt generation can occur when the top half (SCRQ[0:7]) of the queue has been filled (QTHF) and the bottom half (SCRQ[8:15]) of the queue has been filled (QBHF). This may allow for uninterrupted and continuous receives by indicating to the CPU to start reading the queue portion that is now full.
  - The QTHF bit is set by hardware when the top half is full. The QTHF bit is cleared when the SCxSR is read with QTHF set, followed by a write of QTHF to zero.
  - The QBHF bit is set by hardware when the bottom half is full. The QBHF bit is cleared when the SCxSR is read with QBHF set, followed by a write of QBHF to zero.
- In order to implement the receive queue, the following conditions must be met: QRE must be set (QSCI1CR); RE must be set (SCC1R1); QOR and QTHF must be cleared (QSCI1SR); and OR, PF, and FE must be cleared (SC1SR).
- Enable and disable options for the interrupts QTHF and QBHF as controlled by the QTHFI and QBHFI, respectfully.
- 4-bit counter (QRPNT) is used as a pointer to indicate where the next valid data frame will be stored.
- A queue overrun error flag (QOR) to indicate when the queue is already full when another data frame is ready to be stored into the queue (similar to the OR bit in single buffer mode). The QOR bit can be set for QTHF = 1 or QBHF = 1, depending on where the store is being attempted.
- The queue can be exited when an idle line is used to indicate when a group of serial transmissions is finished. This can be achieved by using the ILIE bit to enable the interrupt when the IDLE flag is set. The CPU can then clear QRE and/or RE allowing the receiver queue to be exited.
- For receiver queue operation, IDLE is cleared when SC1SR is read with IDLE set, followed by a read of SCRQ[0:15].
- For receiver queue operation, NF is cleared when the SC1SR is read with NF set, followed by a read of SCRQ[0:15]. When noise occurs, the data is loaded into the receive queue, and operation continues unaffected. However, it may not be possible to determine which data frame in the receive queue caused the noise flag to be asserted.
- The queue is successfully filled (16 data frames) if error flags (FE and PF) are clear, QTHF and QBHF are set, and QRPNT is reset to all zeroes.
- QOR indicates that a new data frame has been received in the data register (SC1DR), but it cannot be placed into the receive queue due to either the QTHF or QBHF flag being set (QSCI1SR). Under this condition, the receive queue is disabled (QRE = 0). Software may service the receive queue and clear the appropri-

ate flag (QTHF, QBHF). Data is not lost provided that the receive queue is re-enabled before OR (SC1SR) is set, which occurs when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost.

14.9.10 QSCI1 Receive Flow Chart Implementing The Queue

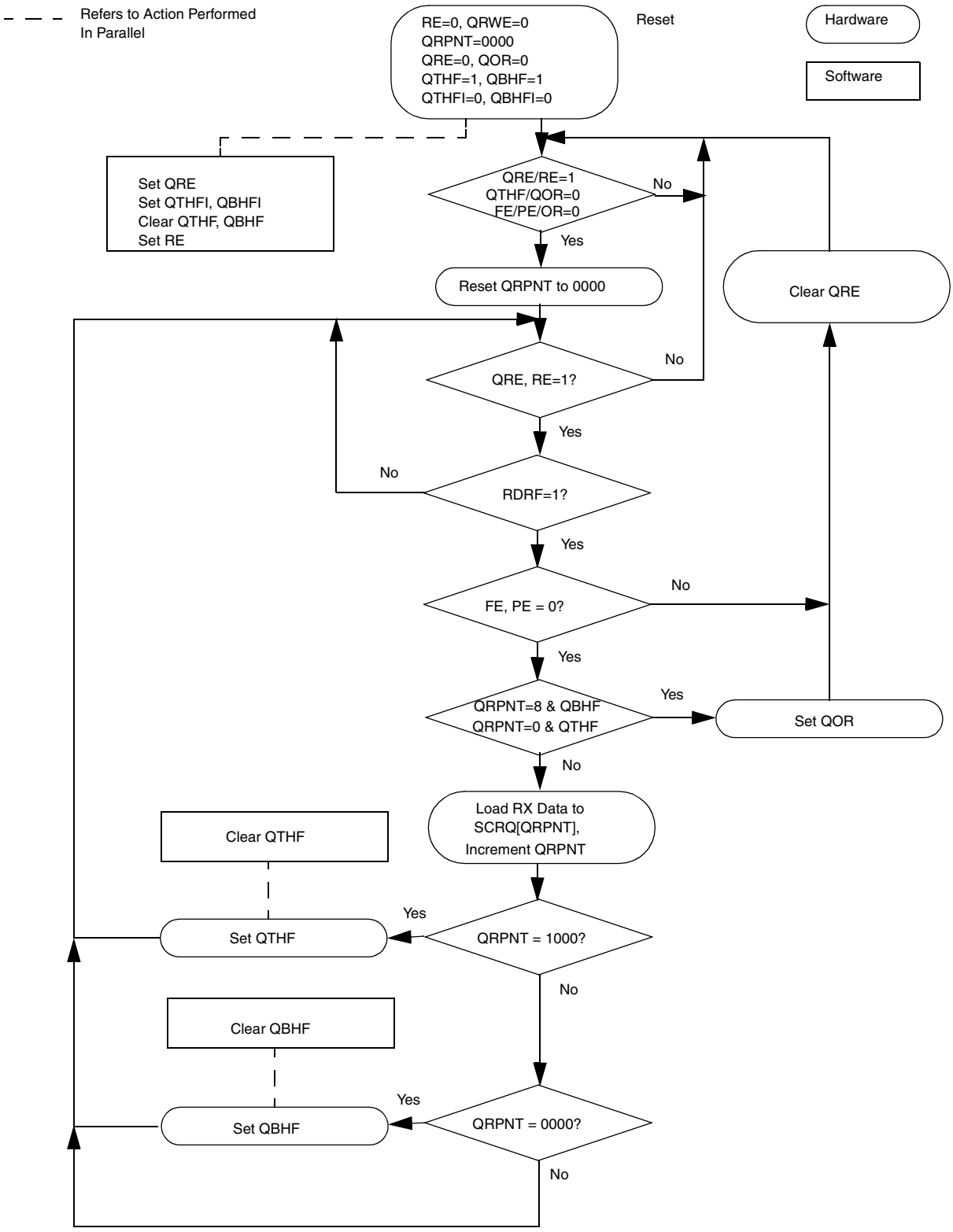


Figure 14-21 Queue Receive Flow

14.9.11 QSCI1 Receive Queue Software Flow Chart

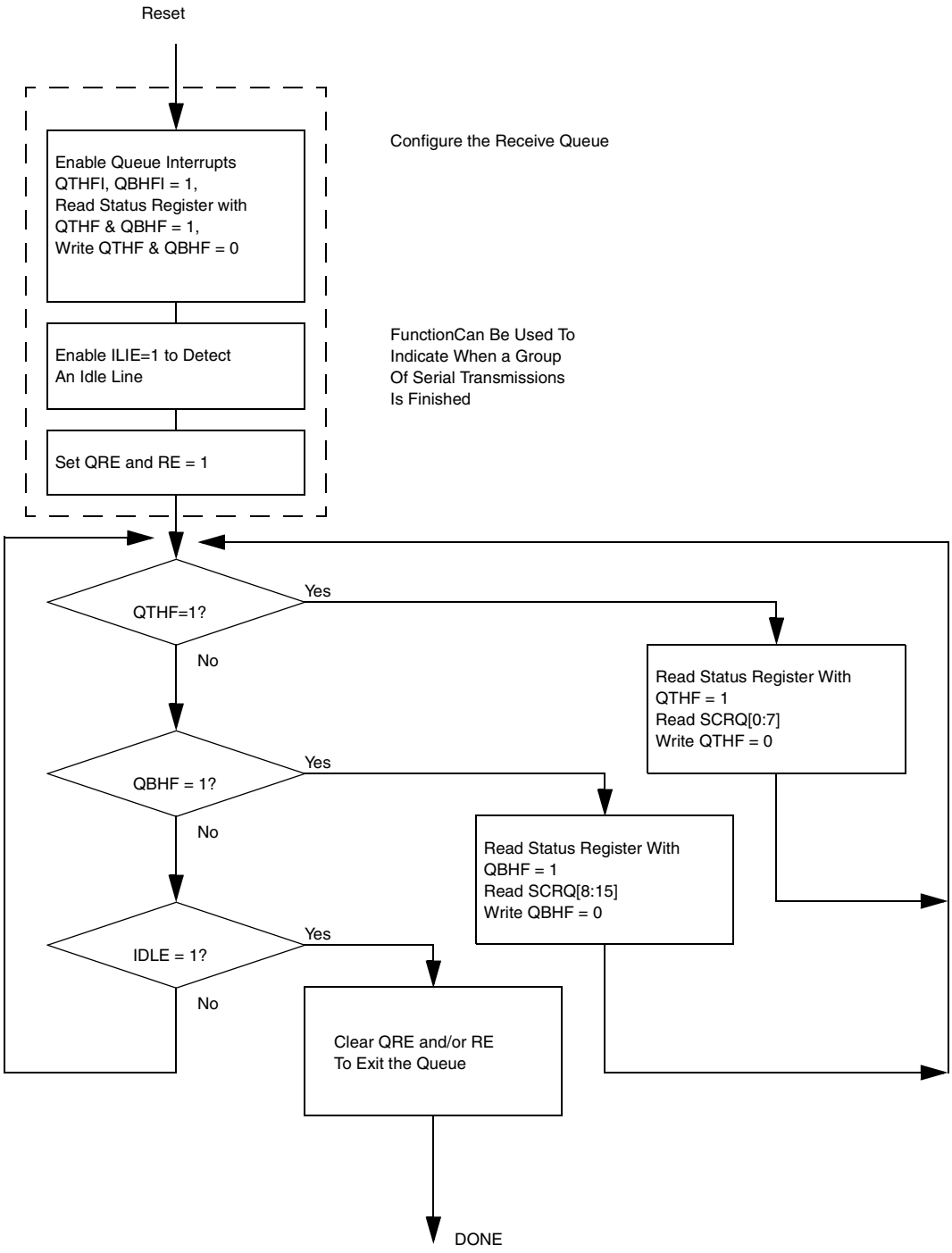


Figure 14-22 Queue Receive Software Flow

14.9.12 Example QSCI1 Receive Operation of 17 Data Frames

Figure 14-23 shows an example receive operation of 17 data frames. The bold type indicates the current value for the QRPNT. Action of the queue may be followed by starting at the top of the figure and going left to right and then down the page.

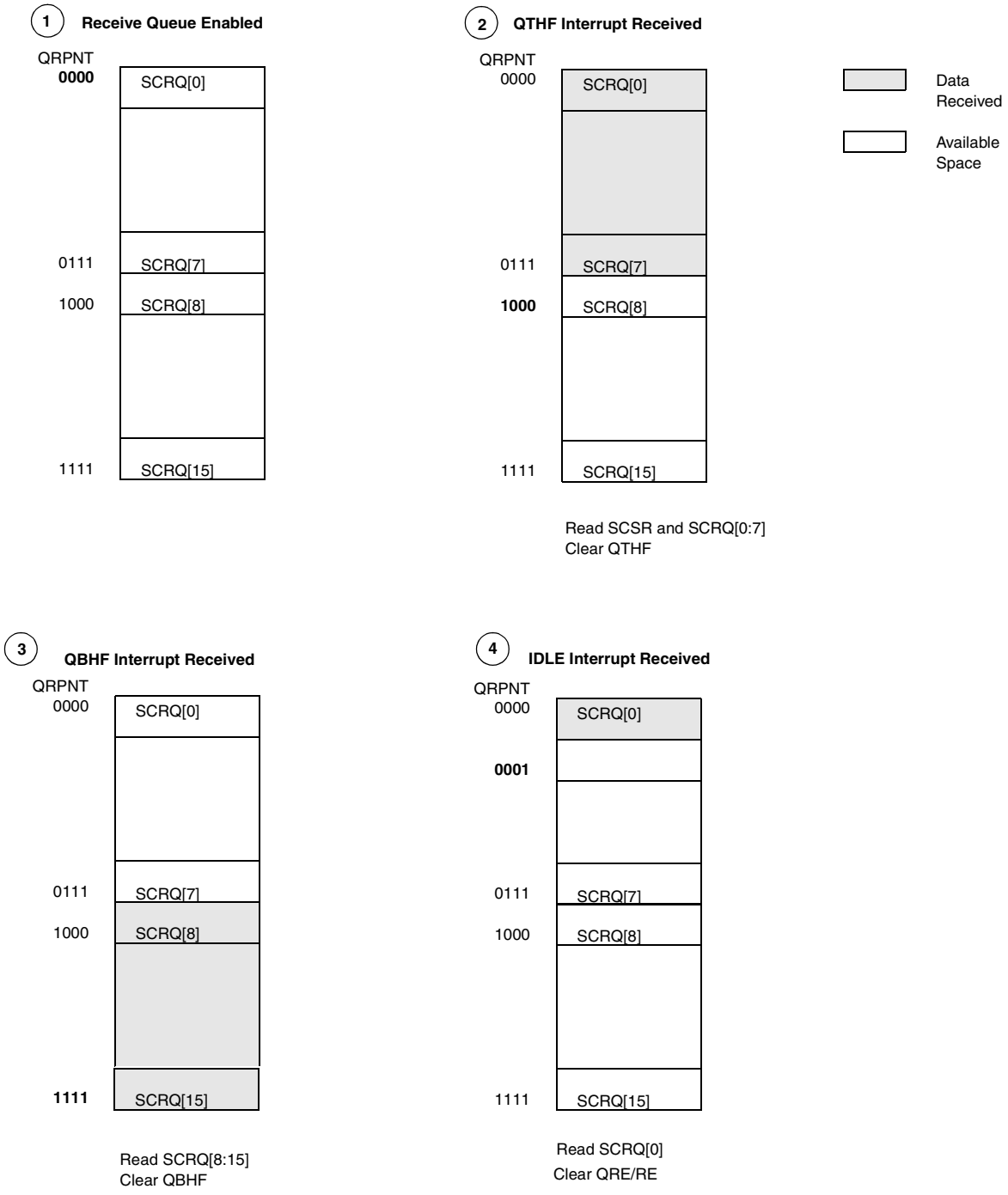


Figure 14-23 Queue Receive Example for 17 Data Bytes

SECTION 15
MODULAR INPUT/OUTPUT SUBSYSTEM (MIOS1)

The modular I/O system (MIOS) consists of a library of flexible I/O and timer functions including I/O port, counters, input capture, output compare, pulse and period measurement, and PWM. Because the MIOS is composed of submodules, it is easily configurable for different kinds of applications. MIOS1 is the implementation of the MIOS architecture used in the MPC555 / MPC556.

The MIOS1 is composed of the following submodules:

- One MIOS bus interface submodule (MBISM)
- One MIOS counter prescaler submodule (MCPSM)
- Two MIOS modulus counter submodules (MMCSM)
- 10 MIOS double action submodules (MDASM)
- Eight MIOS pulse width modulation submodules (MPWMSM)
- One MIOS 16-bit parallel port I/O submodule (MPIOSM)
- Two MIOS interrupt request submodules (MIRSM)

15.1 MIOS1 Features

The basic features of the MIOS1 are as follows:

- Modular architecture at the silicon implementation level
- Disable capability in each submodule to allow power saving when its function is not needed
- Two 16-bit buses to allow action submodules to use counter data
- When not used for timing functions, every channel pin can be used as a port pin: I/O, output only or input only, depending on the channel function
- Submodules pin status bits:
- MIOS counter prescaler submodule (MCPSM):
  - — Centralized counter clock generator
  - — Programmable 4-bit modulus down-counter
  - — Wide range of possible division ratios: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16
  - — Count inhibit under software control
- Two MIOS modulus counter submodules (MMCSM), each with these features:
  - — Programmable 16-bit modulus up-counter with built-in programmable 8-bit prescaler clocked by MCPSM output
  - — Maximum increment frequency of the counter:
    - clocked by the internal Counter Clock: F_{SYS} /2
    - clocked by the external pin: F_{SYS} /4
  - — Flag setting and possible interrupt generation on overflow of the up-counter
  - — Time counter on internal clock with interrupt capability after a pre-determined time

— Optional pin usable as an external event counter (pulse accumulator) with overflow and interrupt capability after a pre-determined number of external events
— Usable as a regular free-running up-counter
— Capable of driving a dedicated 16-bit counter bus to provide timing information to action submodules — the value driven is the contents of the 16-bit up-counter register
— Optional pin to externally force a load of the counter with modulus value

- Ten MIOS double action submodules (MDASM), each with these features:
  — Versatile 16-bit dual action unit allowing two events to occur before software intervention is required
  — Six software selectable modes allowing the MDASM to perform pulse width and period measurements, PWM generation, single input capture and output compare operations as well as port functions
  — Software selection of one of the two possible 16-bit counter buses used for timing operations
  — Flag setting and possible interrupt generation after MDASM action completion
  — Software selection of output pulse polarity
  — Software selection of totem-pole or open-drain output
  — Software readable output pin status
- Eight MIOS pulse width modulation submodules (MPWMSM), each with these features:
  — Output pulse width modulated (PWM) signal generation with no software involvement
  — Built-in 8-bit programmable prescaler clocked by the MCPSM
  — PWM period and pulse width values provided by software:
    - Double-buffered for glitch-free period and pulse width changes
    - 2-cycle minimum output period/pulse-width increment (50 ns at f_{SYS} = 40 MHz)
    - 50% duty-cycle output maximum frequency: 10 MHz
    - Up to 16 bits output pulse width resolution
    - Wide range of periods:
      — 16 bits of resolution: period range from 3.27 ms (with 50 ns steps) to 6.71 s (with 102.4 µs steps)
      — 8 bits of resolution: period range from 12.8 µs (with 50 ns steps) to 26.2 ms (with 102.4 µs steps)
    - Wide range of frequencies:
      — Maximum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-2 prescaler selection: 305 Hz (3.27 ms.)
      — Minimum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-4096 prescaler selection: 0.15 Hz (6.7 s.)
      — Maximum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-2 prescaler selection: 78125 Hz (12.8 µs.)
      — Minimum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-4096 prescaler selection: 38.14 Hz (26.2 ms.)
  — Programmable duty cycle from 0% to 100%
  — Possible interrupt generation after every period

— Software selectable output pulse polarity
— Software readable output pin status
— Possible use of pin as I/O port when PWM function is not needed

- MIOS 16-bit parallel port I/O submodule (MPIOSM):
  — 16 parallel input/output pins
  — Simple data direction register (DDR) concept for selection of pin direction

15.2 Submodule Numbering, Naming and Addressing

A block is a group of four 16-bit registers. Each of the blocks within the MIOS1 addressing range is assigned a block number. The first block is located at the base address of the MIOS1. The blocks are numbered sequentially starting from 0.

Every submodule instantiation is also assigned a number. The number of a given submodule is the block number of the first block of this submodule.

A submodule is assigned a name made of its acronym followed by its submodule number. For example, if submodule number 18 were an MPWMSM, it would be named MPWMSM18.

This numbering convention does not apply to the MBISM, the MCPSM and the MIRSMs. The MBISM and the MCPSM are unique in the MIOS1 and do not need a number. The MIRSMs are numbered incrementally starting from zero.

The MIOS1 base address is defined at the chip level and is referred to as the “MIOS1 base address.” The MIOS1 addressable range is four Kbytes.

The base address of a given implemented submodule within the MIOS1 is the sum of the base address of the MIOS1 and the submodule number multiplied by eight. (Refer to **Table 15-36**.)

This does not apply to the MBISM, the MCPSM and the MIRSMs. For these submodules, refer to the MIOS1 memory map (**Figure 15-2**).

15.3 MIOS1 Signals

The MIOS1 requires 34 pins: 10 MDASM pins, eight MPWMSM pins and 16 MPIOSM pins. The usage of these pins is shown in the block diagram of **Figure 15-1** and in the configuration description of **Table 15-36**. In the figure, MDASM pins have a prefix MDA, MPWMSM pins have a prefix of MPWM and the port pins have a prefix of MPIO. The modulus counter clock and load pins are multiplexed with MDASM pins.

The MIOS1 input and output pin names are composed of five fields according to the following convention:

- “M”
- <submodule short_prefix>
- <submodule number>
- <pin attribute suffix> (optional)
- <bit number> (optional)

The pin prefix and suffix for the different MIOS submodules are as follows:

- MMCSM:
  - — submodule short_prefix: “MC”
  - — pin attribute suffix: C for the Clock pin
  - — pin attribute suffix: L for the Load pin
  - — For example, an MMCSM placed as submodule number n would have its corresponding input clock pin named MMCnC and its input load pin named MMCnL. On the MPC555 / MPC556 MMC6C is input on MDA11 and MMC22C is input on MDA13. The MMC6L is input on MDA12 and MMC22C is input on MDA14.
- MDASM:
  - — submodule short_prefix: “DA”
  - — pin attribute suffix: none
  - — For example a MDASM placed as submodule number n would have its corresponding channel I/O pin named MDAn
- MPWMSM:
  - — submodule short_prefix: “PWM”
  - — pin attribute suffix: none
  - — For example a MPWMSM placed as submodule number n would have its corresponding channel I/O pin named MPWMn
- MPIOSM:
  - — submodule short_prefix: “PIO”
  - — pin attribute suffix: B
  - — For example a MPIOSM placed as submodule number n would have its corresponding I/O pins named MPIOnB0 to MPIOnB15 for bit-0 to bit-15, respectively.

In the MIOS1, some pins are multiplexed between submodules using the same pin names for the inputs and outputs which are connected as shown in **Table 15-36**.

15.4 Block Diagram

Figure 15-1 is a block diagram of the MIOS1.

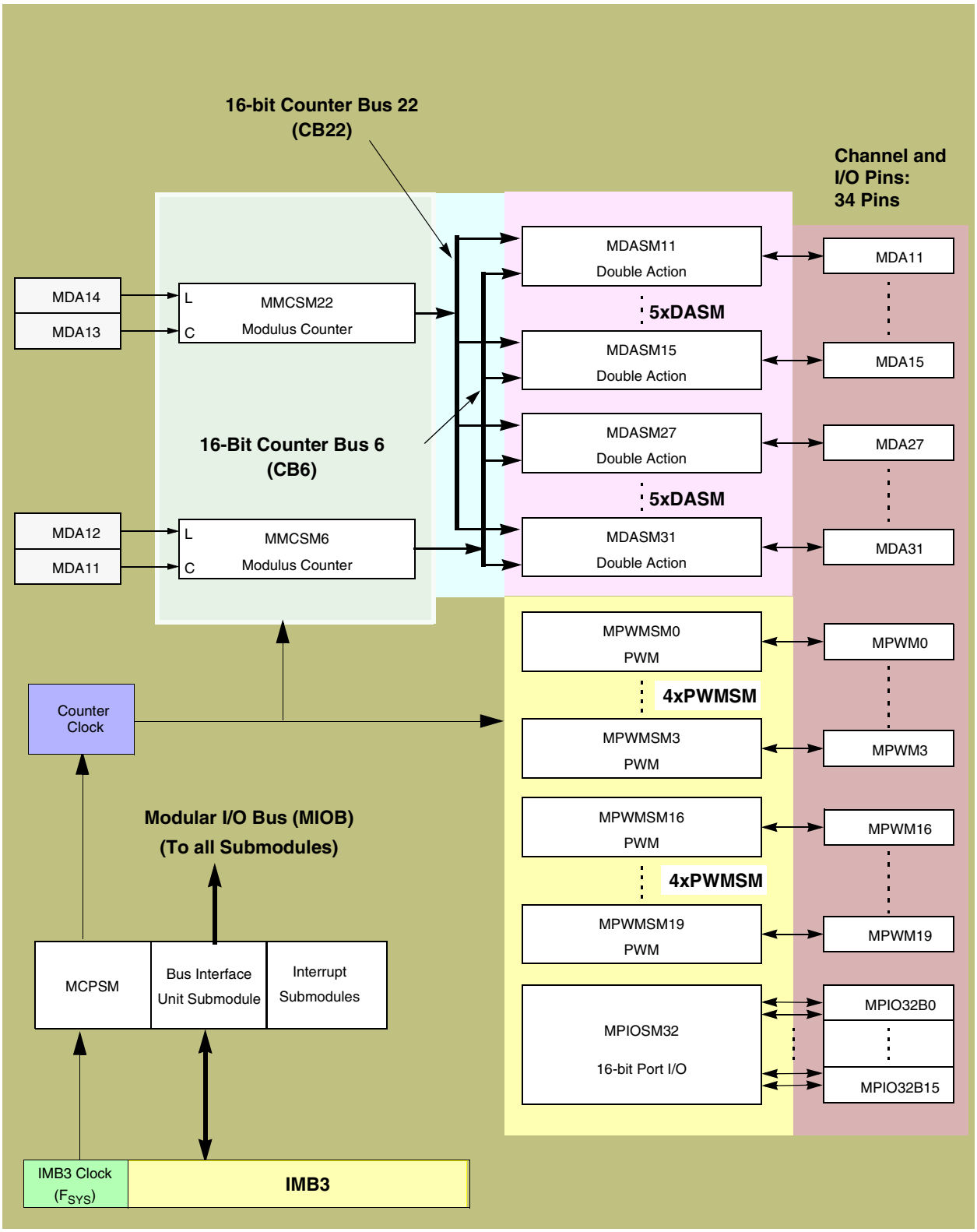


Figure 15-1 MIOS1 Block Diagram

15.5 MIOS1 Bus System

The internal bus system within the MIOS1 is called the modular I/O bus (MIOB). The MIOB makes communications possible between any submodule and the IMB3 bus master through the MBISM.

The MIOB is divided into three dedicated buses:

- The read/write and control bus
- The request bus
- The counter bus set

15.5.1 Read/Write and Control Bus

The read/write and control bus (RWCB) allows read and write data transfers to and from any I/O submodule through the MBISM. It includes signals for data and addresses as well as control signals. The control signals allow 16-bit simple synchronous single master accesses and supports fast or slow master accesses.

15.5.2 Request Bus

The request bus (RQB) provides interrupt request signals along with I/O submodule identification and priority information to the MBISM.

Note that some submodules do not generate interrupts and are therefore independent of the RQB.

15.5.3 Counter Bus Set

The 16-bit counter bus set (CBS) is a set of two 16-bit counter buses. The CBS makes it possible to transfer information between submodules. Typically, counter submodules drive the CBS, while action submodules process the data on these buses. Note, however, that some submodules are self-contained and therefore independent of the counter bus set.

15.6 MIOS1 Programmer's Model

The address space of the MIOS1 consist of four Kbytes starting at the base address of the module. The MIOS1 base address is a multiple of the addressable range. The overall address map organization is shown in **Figure 15-2**.

To find the base address of a given implementation, refer to **1.3 MPC555 / MPC556 Address Map**. To find the submodule base address, refer to **Table 15-36**.

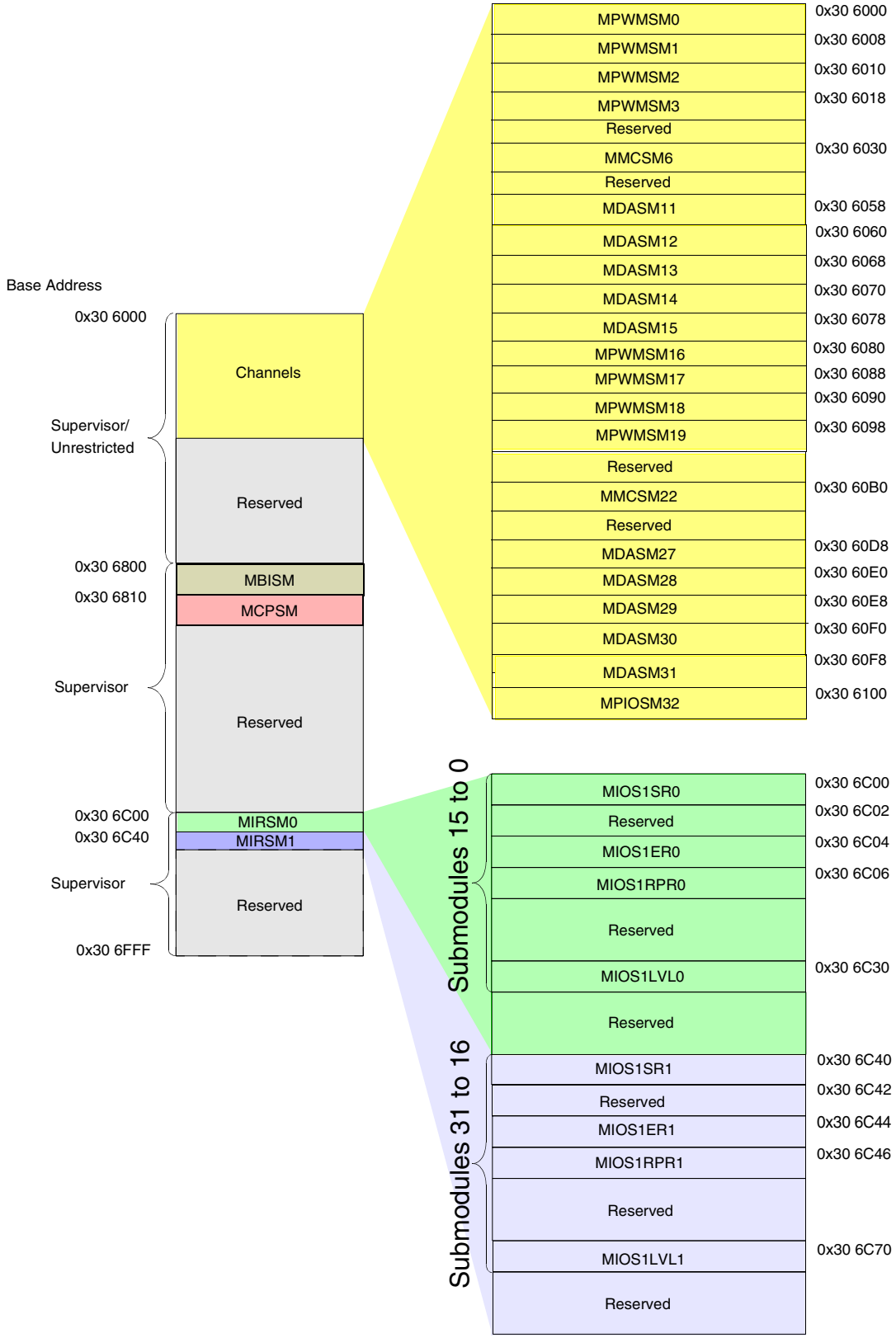


Figure 15-2 MIOS1 Memory Map

If a supervisor privilege address space is accessed in user mode, the module returns a bus error.

All MIOS1 unimplemented locations within the addressable range, return a logic 0 when accessed. In addition, the internal $\overline{\text{TEA}}$ (transfer error acknowledge) signal is asserted.

All unused bits within MIOS1 registers return a 0 when accessed.

15.7 MIOS1 I/O Ports

Each pin of each submodule can be used as an input, output, or I/O port:

Table 15-1 MIOS1 I/O Ports

Submodule	Number	Type
MPIOSM	16	I/O
MMCSM	2	I
MDASM	1	I/O
MPWMSM	1	I/O

15.8 MIOS Bus Interface Submodule (MBISM)

The MIOS bus interface submodule (MBISM) is used as an interface between the MIOB (modular I/O bus) and the IMB3. It allows the CPU to communicate with the MIOS1 submodules.

15.8.1 MIOS Bus Interface (MBISM) Registers

Table 15-2 is the address map for the MBISM submodule.

Table 15-2 MBISM Address Map

Address	Register
0x30 6800	MIOS1 Test and Pin Control Register (MIOS1TPCR) See **Table 15-3** for bit descriptions.
0x30 6802	Reserved (MIOS1 Vector Register in some implementations)
0x30 6804	MIOS1 Module Version Number Register (MIOS1VNR) See **Table 15-4** for bit descriptions.
0x30 6806	MIOS1 Module Control Register (MIOS1MCR) See **Table 15-4** for bit descriptions.
0x30 6808 – 0x30 680E	Reserved

15.8.1.1 MIOS1 Test and Pin Control Register

MIOS1TPCR — Test and Pin Control Register **0x30 6800**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TEST	RESERVED													VF	VFLS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is used for MIOS1 factory testing and selecting between the MIOS1 pin functions for the MPIO32B[0:3] and the developement support pin functions of VFLS[0:1] and VF[0-2].

Table 15-3 MIOS1TPCR Bit Descriptions

Bit(s)	Name	Description
0	TEST	This bit is reserved for factory testing of the MIOS1. The test mode is disabled by reset.
1:13	—	Reserved
14	VF	Pin multiplex. This bit is used to determine the usage of the MIOS1 pins. Refer to the pad-ring specification of the chip for details about the usage of this bit. This bit is set to 0 by reset. 0 = the concerned pins are dedicated to the MIOS1. 1 = alternate function
15	VFLS	Pin multiplex. This bit is used to determine the usage of the MIOS1 pins. Refer to the pad-ring specification of the chip for details about the usage of this bit. This bit is set to 0 by reset. 0 = the concerned pins are dedicated to the MIOS1. 1 = alternate function

15.8.1.2 MIOS1 Vector Register

This register is used only in MCUs that use vectored interrupts. The MPC555 / MPC556 does not use this register.

15.8.1.3 MIOS1 Module and Version Number Register

This read-only register contains the hard-coded values of the module and version number.

MIOS1VNR — MIOS1 Module/Version Number Register **0x30 6804**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MN								VN							

Table 15-4 MIOS1VNR Bit Descriptions

Bit(s)	Name	Description
0:7	MN	Module number = 1 on the MPC555 / MPC556. The MPC555 / MPC556 implements the MIOS1 module.
8:15	VN	Version number

15.8.1.4 MIOS1 Module Configuration Register

MIOS1MCR — MIOS1 Module Configuration Register **0x30 6806**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	0	FRZ	RST	RESERVED				SUPV	RESERVED			RESERVED (IARB)			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-5 MIOS1MCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop enable. Setting the STOP bit activates the MIOB freeze signal regardless of the state of the IMB3 FREEZE signal. The MIOB freeze signal is further validated in some submodules with internal freeze enable bits in order for the submodule to be stopped. The MBISM continues to operate to allow the CPU access to the submodule's registers. The MIOB freeze signal remains active until reset or until the STOP bit is written to zero by the CPU (via the IMB3). The STOP bit is cleared by reset. 0 = Enables MIOS1 operation. 1 = Selectively stops MIOS1 operation.
1	—	Reserved
2	FRZ	Freeze enable. Setting the FRZ bit, activates the MIOB freeze signal only when the IMB3 FREEZE signal is active. The MIOB freeze signal is further validated in some submodules with internal freeze enable bits in order for the submodule to be frozen. The MBISM continues to operate to allow the CPU access to the submodule's registers. The MIOB freeze signal remains active until the FRZ bit is written to zero or the IMB3 FREEZE signal is negated. The FRZ bit is cleared by reset. 0 = Ignores the FREEZE signal on the IMB3, allowing MIOS1 operation. 1 = Selectively stops MIOS1 operation when the FREEZE signal appears on the IMB3.
3	RST	Module reset. The RST bit always returns 0 when read and can be written to 1. When the RST bit is written to 1, the MBISM activates the reset signal on the MIOB. This completely stops the operation of the MIOS1 and resets all the values in the submodules registers that are affected by reset. This bit provides a way of resetting the complete MIOS1 module regardless of the reset state of the CPU. The RST bit is cleared by reset. 0 = Writing a 0 to RST has no effect. 1 = Reset the MIOS1 submodules
4:7	—	Reserved
8	SUPV	Supervisor data space selector. The SUPV bit specifies whether the address space from 0x0000 to 0x07FF in the MIOS1 is accessed at the supervisor privilege level. When SUPV is cleared, these addresses are accessed at the Unrestricted privilege level. The SUPV bit is cleared by reset. 0 = Unrestricted data space. 1 = Supervisor data space.
9:15	—	Reserved. In implementations that use hardware interrupt arbitration, bits 12:15 represent the IARB field.

15.8.2 MBISM Interrupt Registers

Table 15-6 shows the MBISM interrupt registers.

Table 15-6 MBISM Interrupt Registers Address Map

Address	Register
0x30 6C30	MIOS1 Interrupt Level Register 0 (MIOS1LVL0) See **Table 15-7** for bit descriptions.
0x30 6C70	MIOS1 Interrupt Level Register 1 (MIOS1LVL1) See **Table 15-8** for bit descriptions.

15.8.2.1 MIOS1 Interrupt Level Register 0 (MIOS1LVL0)

This register contains the interrupt level that applies to the submodules number 15 to zero.

MIOS1LVL0 — MIOS1 Interrupt Level Register 0 **0x30 6C30**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED					LVL			TM		RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-7 MIOS1LVL0 Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	LVL	Interrupt request level. This field represents one of eight possible levels.
8:9	TM	Time multiplexing. This field determines the multiplexed time slot
10:15	—	Reserved

15.8.2.2 MIOS1 Interrupt Level Register 1 (MIOS1LVL1)

This register contains the interrupt level that applies to the submodules number 31 to 16.

MIOS1LVL1 — MIOS1 Interrupt Level 1 Register **0x30 6C70**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED					LVL			TM		RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-8 MIOS1LVL1 Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	LVL	Interrupt request level. This field represents one of eight possible levels.
8:9	TM	Time multiplexing. This field determines the multiplexed time slot.
10:15	—	Reserved

15.8.3 Interrupt Control Section (ICS)

The interrupt control section delivers the interrupt level to the CPU. The interrupt control section adapts the characteristics of the MIOB request bus to the characteristics of the interrupt structure of the IMB3.

When at least one of the flags is set on an enabled level, the ICS receives a signal from the corresponding IRQ pending register. This signal is the result of a logical “OR” between all the bits of the IRQ pending register.

The signal received from the IRQ pending register is associated with the interrupt level register within the ICS. This level is coded on five bits in this register: three bits represent one of eight levels and the two other represent the four time multiplex slots. Ac-

cording to this level, the ICS sets the correct $\overline{IRQ}$[7:0] lines with the correct ILBS[1:0] time multiplex lines on the peripheral bus. The CPU is then informed as to which of the thirty-two interrupt levels is requested.

Based on the interrupt level requested, the software must determine which submodule requested the interrupt. The software may use a find-first-one type of instruction to determine, in the concerned MIRSM, which of the bits is set. The CPU can then serve the requested interrupt.

15.9 MIOS Counter Prescaler Submodule (MCPSM)

The MIOS counter prescaler submodule (MCPSM) divides the MIOS1 clock (F_{SYS}) to generate the counter clock. It is designed to provide all the submodules with the same division of the main MIOS1 clock (division of F_{SYS}). It uses a 4-bit modulus counter. The clock signal is prescaled by loading the value of the clock prescaler register into the prescaler counter every time it overflows. This allows all prescaling factors between two and 16. Counting is enabled by asserting the PREN bit in the control register. The counter can be stopped at any time by negating this bit, thereby stopping all submodules using the output of the MCPSM (counter clock).

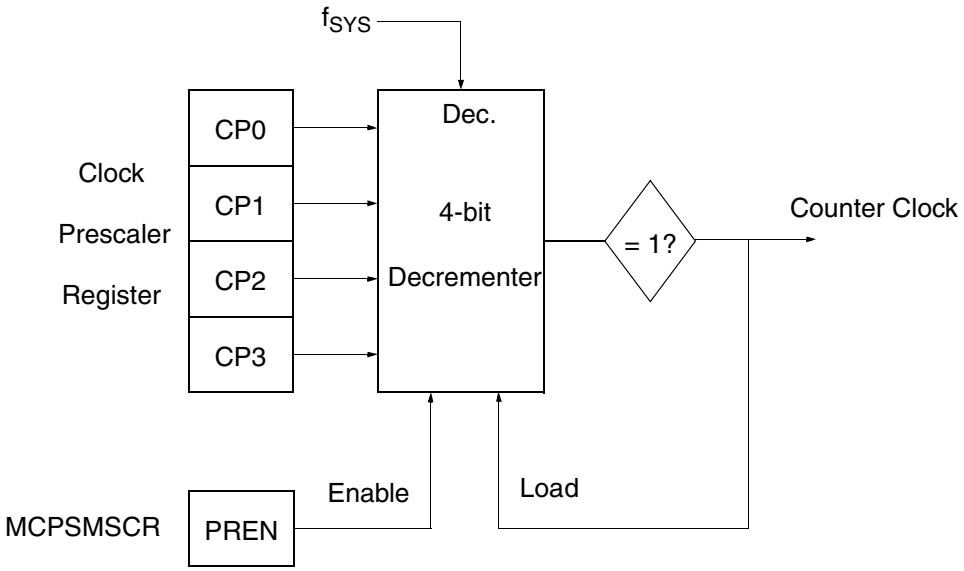


Figure 15-3 MCPSM Block Diagram

15.9.1 MIOS Counter Prescaler Submodule (MCPSM) Registers

Table 15-9 is the address map for the MCPSM submodule.

Table 15-9 MCPSM Address Map

Address	Register
0x30 6810 – 0x30 6814	Reserved
0x30 6816	MCPSM Status/Control Register (MCPSMSCR) See **Table 15-10** for bit descriptions.

15.9.1.1 MCPSM Status/Control Register (MCPSMCSCR)

This register contains status and control information for the MCPSM.

MCPSMSCR — MCPSM Status/Control Register **0x30 6816**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PREN	FREN	RESERVED										PSL			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-10 MCPSMSCR Bit Descriptions

Bit(s)	Name	Description
0	PREN	Prescaler enable. This active high read/write control bit enables the MCPSM counter. The PREN bit is cleared by reset. 0 = MCPSM counter disabled. 1 = MCPSM counter enabled.
1	FREN	Freeze enable. This active high read/write control bit when set make possible a freeze of the MCPSM counter if the MIOB freeze line is activated. Note that this line is active when the MIOS1MCR STOP bit is set or when the MIOS1MCR FREN bit and the IMB3 FREEZE line are set. When the MCPSM is frozen, it stops counting. Then when the FREN bit is reset or when the freeze condition on the MIOB is negated, the counter restarts from where it was before being frozen. The FREN bit is cleared by reset. 0 = MCPSM counter not frozen. 1 = Selectively stops MIOS1 operation when the FREEZE signal appears on the IMB3.
2:11	—	Reserved
12:15	PSL	Clock prescaler. This 4-bit read/write data register stores the modulus value for loading into the clock prescaler. The new value is loaded into the counter on the next time the counter equals one or when disabled (PREN bit = 0). Divide ratios are as follows: 0000 = 16 0001 = No counter clock output 0010 = 2 0011 = 3 . . . 1110 = 14 1111 = 15

15.10 MIOS Modulus Counter Submodule (MMCSM)

The MMCSM is a versatile counter submodule capable of performing complex counting and timing functions, including modulus counting, in a wide range of applications. The MMCSM may also be configured as an event counter, allowing the overflow flag to be set after a predefined number of events (internal clocks or external events), or

as a time reference for other submodules. Note that the MMCSM can also operate as a free running counter by loading the modulus value of zero.

The main components of the MMCSM are an 8-bit prescaler counter, an 8-bit prescaler register, a 16-bit up-counter register, a 16-bit modulus latch register, counter loading and interrupt flag generation logic.

The contents of the modulus latch register is transferred to the counter under the following three conditions:

1. When an overflow occurs
2. When an appropriate transition occurs on the external load pin
3. When the program writes to the counter register. In this case, the value is first written into the modulus register and immediately transferred to the counter.

Software can also write a value to the modulus register for later loading into the counter with one of the two first criteria.

A software control register selects whether the clock input to the counter is the prescaler output or the corresponding input pin. The polarity of the external input pin is also programmable.

Refer to **Table 15-36** for the MMCSM relative I/O pin implementation.

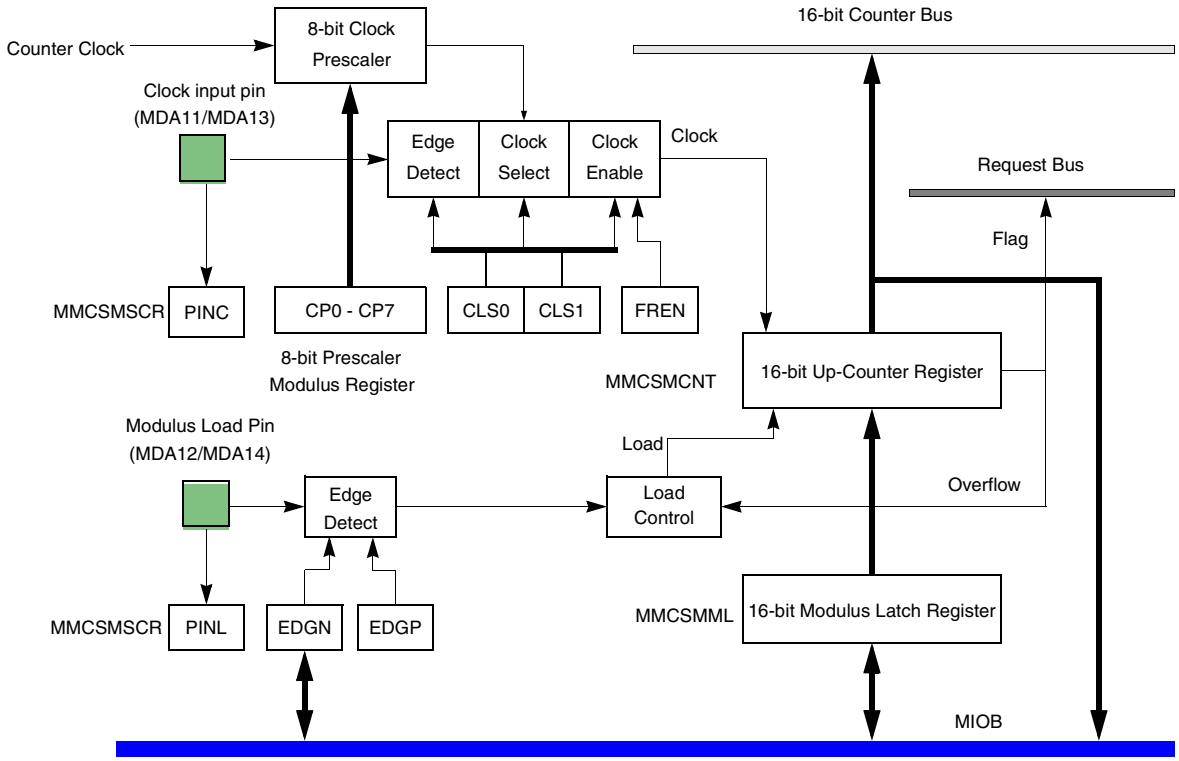


Figure 15-4 MMCSM Block Diagram

15.10.1 MIOS Modulus Counter Submodule (MMCSM) Registers

Each of the two MMCSM submodules in the MPC555 / MPC556 includes the register set shown in **Table 15-11**.

Table 15-11 MMCSM Address Map

Address	Register
MMCSM6	
0x30 6030	MMCSM6 Up-Counter Register (MMCSMCNT) See **Table 15-12** for bit descriptions.
0x30 6032	MMCSM6 Modulus Latch Register (MMCSMML) See **Table 15-13** for bit descriptions.
0x30 6034	MMCSM6 Status/Control Register Duplicated (MMCSMSCRD) See **15.10.1.3 MMCSM Status/Control Register (Duplicated)** for bit descriptions.
0x30 6036	MMCSM6 Status/Control Register (MMCSMSCR)
MMCSM22	
0x30 60B0	MMCSM Up-Counter Register (MMCSMCNT)
0x30 60B2	MMCSM Modulus Latch Register (MMCSMML)
0x30 60B4	MMCSM Status/Control Register Duplicated (MMCSMSCRD)
0x30 60B6	MMCSM Status/Control Register (MMCSMSCR)

15.10.1.1 MMCSM Up-Counter Register (MMCSMCNT)

The MMCSMCNT register contains the 16-bit value of the up counter. Note that writing to MMCSMCNT simultaneously writes to MMCSMML.

MMCSMCNT — MMCSM Up-Counter Register **0x30 6030**
0x30 60B0

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CNT															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

U = Unaffected by reset

Table 15-12 MMCSMCNT Bit Descriptions

Bit(s)	Name	Description
0:15	CNT	Counter value. These read/write data bits represent the 16-bit value of the up-counter. CNT contains the value that is driven onto the 16-bit counter bus.

15.10.1.2 MMCSM Modulus Latch Register (MMCSMML)

The MMCSMML is a read/write register containing the 16-bit value of the up-counter.

MMCSMML — MMCSM Modulus Latch Register **0x30 6032**
0x30 60B2

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
ML															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

U = Unaffected by reset

Table 15-13 MMCSMML Bit Descriptions

Bit(s)	Name	Description
0:15	ML	Modulus latches. These bits are read/write data bits containing the 16-bit modulus value to be loaded into the up-counter. The value loaded in this register must be the two's complement of the desired modulus count. The up-counter increments from this two's complement value up to 0xFFFF to get the correct number of steps before an overflow is generated to reload the modulus value into the up-counter. A value of 0x0000 should be used for a free-running counter.

15.10.1.3 MMCSM Status/Control Register (Duplicated)

The MMCSMSCRD and the MMCSMSCR are the same registers accessed at two different addresses. Reading or writing to one of these two addresses has exactly the same effect.

NOTE

The user should not write directly to the address of the MMCSM-SCRD. This register's address may be reserved for future use and should not be accessed by the software to assure future software compatibility.

MMCSMSCRD — MMCSM Status/Control Register (Duplicated) **0x30 6034**
0x30 60B4

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PINC	PINL	FREN	EDGN	EDGP	CLS		—	CP							
RESET:															
—	—	0	0	0	0	0	0	U	U	U	U	U	U	U	U

15.10.1.4 MMCSM Status/Control Register (MMCSMSCR)

This register contains both read-only status bits and read/write control bits.

MMCSMSCR — MMCSM Status/Control Register **0x30 6036**
0x30 60B6

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PINC	PINL	FREN	EDGN	EDGP	CLS		—	CP							
RESET:															
—	—	0	0	0	0	0	0	U	U	U	U	U	U	U	U

Table 15-14 MMCSMSCR Bit Descriptions

Bit(s)	Name	Description
0	PINC	Clock input pin status. This read-only status bit reflects the logic state of the clock input pin MMCnC (MDA11 or MDA13).
1	PINL	Modulus load input pin status. This read-only status bit reflects the logic state of the modulus load pin MMCnL (MDA12 or MDA14).
2	FREN	Freeze enable. This active high read/write control bit enables the MMCSM to recognize the MIOB freeze signal.
3:4	EDGN, EDGP	Modulus load falling edge/rising edge sensitivity. These active high read/write control bits set falling-edge and rising edge sensitivity, respectively, for the MMCnL pin (MDA12 or MDA14). 00 = Disabled 01 = MMCSMCNT load on rising edges 10 = MMCSMCNT load on falling edges 11 = MMCSMCNT load on rising and falling edges
5:6	CLS	Clock select. These read/write control bits select the clock source for the modulus counter. 00 = Disabled 01 = Falling edge of MMCnC (MDA11 or MDA13) pin 10 = Rising edge of MMCnC (MDA11 or MDA13) pin 11 = MMCSM clock prescaler
7	—	—
8:15	CP	Clock prescaler. This 8-bit read/write data register stores the two's complement of the desired modulus value for loading into the built-in 8-bit clock prescaler. The new value is loaded into the prescaler counter when the next counter overflow occurs or when the CLS bits are set to select the clock prescaler as the clock source. **Table 15-15** gives the clock divide ratio according to the CP values

Table 15-15 MMCSMCR CP and MPWMSMSCR CP Values

Prescaler Value (CP in hex)	MIOS Prescaler Clock Divided by
FF	1
FE	2
FD	3
FC	4
FB	5
FA	6
F9	7
F8	8
.....	.....
02	254 (2^8 -2)
01	255 (2^8 -1)
00	256 (2^8)

15.11 MIOS Double Action Submodule (MDASM)

The MIOS double action submodule (MDASM) provides two consecutive 16-bit input captures or two consecutive 16-bit output compare functions that can occur automatically without software intervention. The input edge detector is programmable to trigger the capture function to occur on the desired edge. The output flip-flop is set by one of the output compares and is reset by the other one. In all modes except disable mode, an optional interrupt is available to the software. Software selection is provided to select which of the incoming 16-bit counter buses is used for the input capture or the output compare.

The MDASM has six different software selectable modes:

- Disable mode
- Pulse width measurement
- Period measurement
- Input capture mode
- Single pulse generation
- Continuous pulse generation

The MDASM has three data registers that are accessible to the software from the various modes. For some of the modes, two of the registers are cascaded together to provide double buffering. The value in one register is transferred to the other register automatically at the correct time so that the minimum pulse (measurement or generation) is just one 16-bit counter bus count.

Refer to **Table 15-36** for the MDASM relative I/O pin implementation.

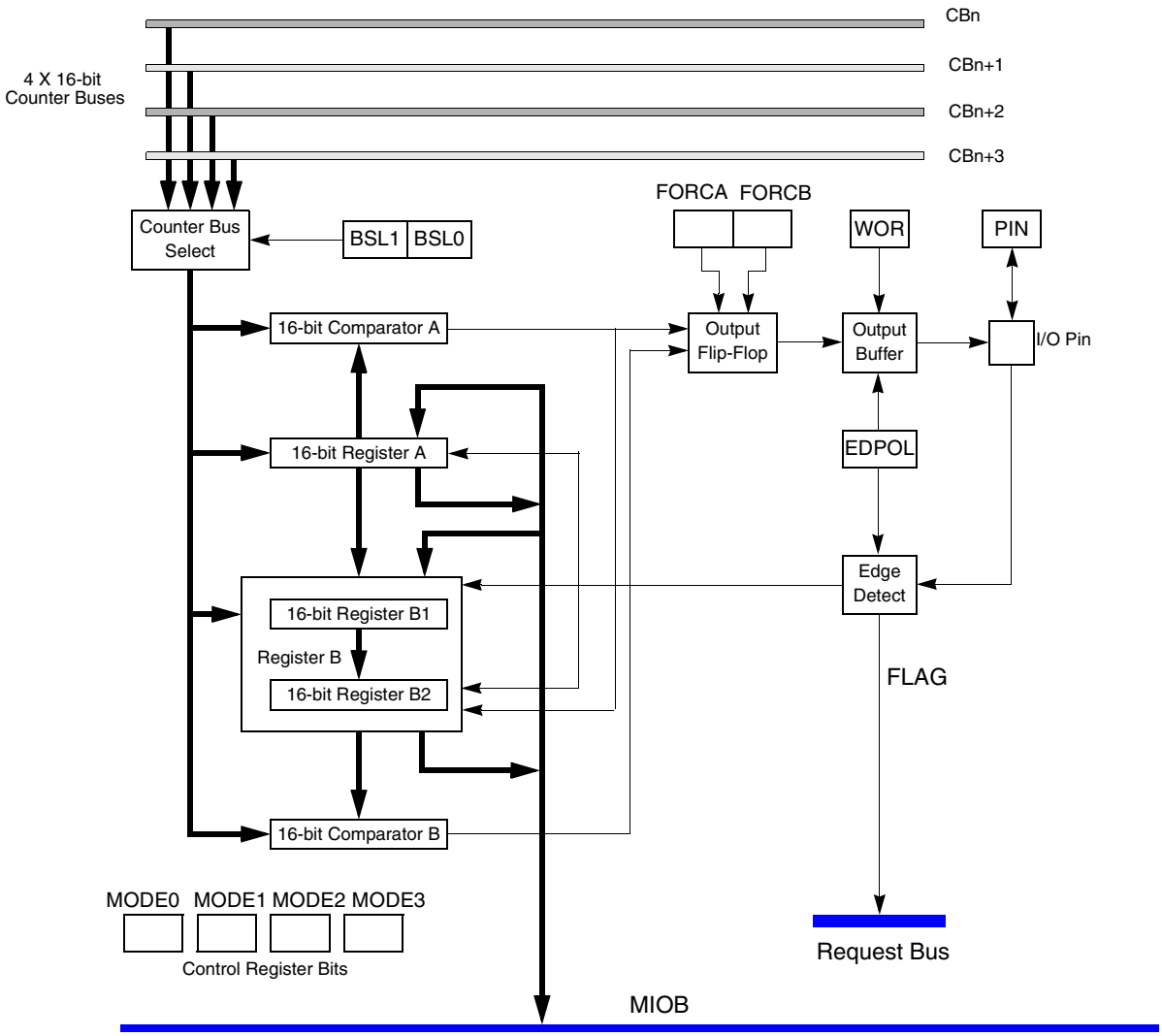


Figure 15-5 MDASM Block Diagram

15.11.1 MIOS Double Action Submodule (MDASM) Registers

One set of registers is associated with each MDASM submodule. The base address of the particular submodule is shown in the table below.

Table 15-16 MDASM Address Map

Address	Register
MDASM11	
0x30 6058	MDASM11 Data A Register (MDASMAR) See **15.11.1.1 MDASM Data A Register** for bit descriptions.
0x30 605A	MDASM11 Data B Register (MDASMBR) See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.
0x30 605C	MDASM11 Status/Control Register Duplicated (MDASMSCRD) See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.
0x30 605E	MDASM11 Status/Control Register (MDASMSCR) See **Table 15-17** for bit descriptions.
MDASM12	
0x30 6060	MDASM12 Data A Register (MDASMAR)
0x30 6062	MDASM12 Data B Register (MDASMBR)
0x30 6064	MDASM12 Status/Control Register Duplicated (MDASMSCRD)
0x30 6066	MDASM12 Status/Control Register (MDASMSCR)
MDASM13	
0x30 6068	MDASM13 Data A Register (MDASMAR)
0x30 606A	MDASM13 Data B Register (MDASMBR)
0x30 606C	MDASM13 Status/Control Register Duplicated (MDASMSCRD)
0x30 606E	MDASM13 Status/Control Register (MDASMSCR)
MDASM14	
0x30 6070	MDASM14 Data A Register (MDASMAR)
0x30 6072	MDASM14 Data B Register (MDASMBR)
0x30 6074	MDASM14 Status/Control Register Duplicated (MDASMSCRD)
0x30 6076	MDASM14 Status/Control Register (MDASMSCR)
MDASM15	
0x30 6078	MDASM15 Data A Register (MDASMAR)
0x30 607A	MDASM15 Data B Register (MDASMBR)
0x30 607C	MDASM15 Status/Control Register Duplicated (MDASMSCRD)
0x30 607E	MDASM15 Status/Control Register (MDASMSCR)
MDASM27	
0x30 60D8	MDASM27 Data A Register (MDASMAR)
0x30 60DA	MDASM27 Data B Register (MDASMBR)
0x30 60DC	MDASM27 Status/Control Register Duplicated (MDASMSCRD)
0x30 60DE	MDASM27 Status/Control Register (MDASMSCR)
MDASM28	
0x30 60E0	MDASM28 Data A Register (MDASMAR)
0x30 60E2	MDASM28 Data B Register (MDASMBR)
0x30 60E4	MDASM28 Status/Control Register Duplicated (MDASMSCRD)
0x30 60E6	MDASM28 Status/Control Register (MDASMSCR)
MDASM29	
0x30 60E8	MDASM29 Data A Register (MDASMAR)

Table 15-16 MDASM Address Map (Continued)

Address	Register
0x30 60EA	MDASM29 Data B Register (MDASMBR)
0x30 60EC	MDASM29 Status/Control Register Duplicated (MDASMSCRD)
0x30 60EE	MDASM29 Status/Control Register (MDASMSCR)
MDASM30	
0x30 60F0	MDASM30 Data A Register (MDASMAR)
0x30 60F2	MDASM30 Data B Register (MDASMBR)
0x30 60F4	MDASM30 Status/Control Register Duplicated (MDASMSCRD)
0x30 60F6	MDASM30 Status/Control Register (MDASMSCR)
MDASM31	
0x30 60F8	MDASM31 Data A Register (MDASMAR)
0x30 60FA	MDASM31 Data B Register (MDASMBR)
0x30 60FC	MDASM31 Status/Control Register Duplicated (MDASMSCRD)
0x30 60FE	MDASM31 Status/Control Register (MDASMSCR)

15.11.1.1 MDASM Data A Register

MDASMAR is the data register associated with channel A. Its use varies with the mode of operation:

- In the DIS mode, MDASMAR can be accessed to prepare a value for a subsequent mode selection
- In the IPWM mode, MDASMAR contains the captured value corresponding to the trailing edge of the measured pulse
- In the IPM and IC modes, MDASMAR contains the captured value corresponding to the most recently detected dedicated edge (rising or falling edge)
- In the OCB and OCAB modes, MDASMAR is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to MDASMAR in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
- In the OPWM mode, MDASMAR is loaded with the value corresponding to the leading edge of the PWM pulse to be generated

MDASMAR — MDASM Data A Register **0x30 6058***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
AR															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

* Refer to **Table 15-16** for a complete list of all the base addresses for the MDASM registers.

15.11.1.2 MDASM Data B Register (MDASMBR)

MDASMBR is the data register associated with channel B. Its use varies with the mode of operation. Depending on the mode selected, software access is to register B1 or register B2.

- In the DIS mode, MDASMBR can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value when register B1 is written.
- In the IPWM mode, MDASMBR contains the captured value corresponding to the leading edge of the measured pulse. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.
- In the IPM and IC modes, MDASMBR contains the captured value corresponding to the most recently detected period edge (rising or falling edge). In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.
- In the OCB and OCAB modes, MDASMBR is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to MDASMBR in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.
- In the OPWM mode, MDASMBR is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed; buffer register B2 is hidden and cannot be accessed.

MDASMBR — MDASM Data B Register **0x30 605A***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BR															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

* Refer to **Table 15-16** for a complete list of all the base addresses for the MDASM registers.

15.11.1.3 MDASM Status/Control Register (Duplicated)

The MDASMSCRD and the MDASMSCR are the same registers accessed at two different addresses. Reading or writing to either of these two addresses has exactly the same effect.

NOTE

The user should not write directly to the address of the MDASMSCRD. This register's address may be reserved for future use and should not be accessed by the software to assure future software compatibility.

MDASMSCRD — MDASM Status/Control Register (Duplicated) **0x30 605C***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIN	WOR	FREN	0	ED-POL	FORC A	FORC B	RESERVED		BSL		0	MOD			
RESET:															
—	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Refer to **Table 15-16** for a complete list of all the base addresses for the MDASM registers.

15.11.1.4 MDASM Status/Control Register

The status/control register contains a read-only bit reflecting the status of the MDASM pin as well as read/write bits related to its control and configuration.

MDASMSCR — MDASM Status/Control Register **0x30 605E***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIN	WOR	FREN	0	ED-POL	FORCA	FORCB	RESERVED		BSL		0	MOD			
RESET:															
—	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Refer to **Table 15-16** for a complete list of all the base addresses for the MDASM registers.

Table 15-17 MDASMSCR Bit Descriptions

Bit(s)	Name	Description
0	PIN	Pin input status. The pin input status bit reflects the status of the corresponding pin.
1	WOR	Wired-OR. In the DIS, IPWM, IPM and IC modes, the WOR bit is not used; reading this bit returns the value that was previously written. In the OCB, OCAB and OPWM modes, the WOR bit selects whether the output buffer is configured for open-drain or totem-pole operation. 0 = Output buffer is totem-pole. 1 = Output buffer is open-drain.
2	FREN	Freeze enable. This active high read/write control bit enables the MDASM to recognize the MIOB freeze signal. 0 = The MDASM is not frozen even if the MIOB freeze line is active. 1 = The MDASM is frozen if the MIOB freeze line is active.
3	—	0
4	EDPOL	Polarity. In the DIS mode, this bit is not used; reading it returns the last value written. In the IPWM mode, this bit is used to select the capture edge sensitivity of channels A and B. 0 = Channel A captures on a rising edge. Channel B captures on a falling edge. 1 = Channel A captures on a falling edge. Channel B captures on a rising edge. In the IPM and IC modes, the EDPOL bit is used to select the input capture edge sensitivity of channel A. 0 = Channel A captures on a rising edge. 1 = Channel A captures on a falling edge. In the OCB, OCAB and OPWM modes, the EDPOL bit is used to select the voltage level on the output pin. 0 = The output flip-flop logic level appears on the output pin: a compare on channel A sets the output pin, a compare on channel B resets the output pin. 1 = The complement of the output flip-flop logic level appears on the output pin: a compare on channel A resets the output pin; a compare on channel B sets the output pin.
5	FORCA	Force A. In the OCB, OCAB and OPWM modes, the FORCA bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel A (except that the FLAG line is not activated). Writing a one to FORCA sets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, the FORCA bit is not used and writing to it has no effect. FORCA is cleared by reset and is always read as zero. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.

Table 15-17 MDASMSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
6	FORCB	Force B. In the OCB, OCAB and OPWM modes, the FORCB bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel B (except that the FLAG line is not activated). Writing a one to FORCB resets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, the FORCB bit is not used and writing to it has no effect. FORCB is cleared by reset and is always read as zero. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.
7:8	—	Reserved
9:10	BSL	Bus select.These bits are used to select which of the four possible 16-bit counter bus passing nearby is used by the MDASM. Refer to **Table 15-36** to see how the MDASM is connected to the 16-bit counter buses in the MIOS1. **NOTE**: In the MPC555 / MPC556, only 0b00 (CB6) and 0b01 (CB22) are implemented.
11	—	0
12:15	MOD	Mode select. These four mode select bits select the mode of operation of the MDASM. To avoid spurious interrupts, it is recommended that MDASM interrupts are disabled before changing the operating mode. It is also imperative to go through the disable mode before changing the operating mode. See **Table 15-18** for details.

Table 15-18 MDASM Mode Selects

MDASM Control Register Bits	Bits of Resolution	Counter Bus Bits Ignored	MDASM Mode of Operation
MOD			
0000	—	—	DIS – Disabled
0001	16	—	IPWM – Input pulse width measurement
0010	16	—	IPM – Input period measurement
0011	16	—	IC – Input capture
0100	16	—	OCB – Output compare, flag on B compare
0101	16	—	OCAB – Output compare, flag on A and B compare
0110	—	—	Reserved
0111	—	—	Reserved
1000	16	—	OPWM – Output pulse width modulation
1001	15	0	OPWM – Output pulse width modulation
1010	14	0,1	OPWM – Output pulse width modulation
1011	13	0-2	OPWM – Output pulse width modulation
1100	12	0-3	OPWM – Output pulse width modulation
1101	11	0-4	OPWM – Output pulse width modulation
1110	9	0-6	OPWM – Output pulse width modulation
1111	7	0-8	OPWM – Output pulse width modulation

15.12 MIOS Pulse Width Modulation Submodule (MPWMSM)

The purpose of the MIOS pulse width modulation submodule (MPWMSM) is to create a variable pulse width output signal at a wide range of frequencies, independent of other MIOS1 output signals. The MPWMSM includes its own 8-bit prescaler and counter and, thus, does not use the MIOS1 16-bit counter buses.

The MPWMSM pulse width can vary from 0.0% to 100.0%, with up to 16 bits of resolution. The finest output resolution is the MCU IMB clock time divided by two (for a F_{SYS} of 40.0 MHz, the finest output pulse width resolution is 50 ns). With the full sixteen bits of resolution and the overall prescaler divide ratio varying from divide-by-2 to divide-by-4096, the period of the PWM output can range from 3.28 ms to 6.7 s (assuming a f_{SYS} of 40 MHz). By reducing the counting value, the output signal period can be reduced. The period can be as fast as 205 µs (4.882 KHz) with twelve bits of resolution, as fast as 12.8 µs (78.125 KHz) with eight bits of resolution and as fast as 3.2 µs (312.500 KHz) with six bits of resolution (still assuming a f_{SYS} of 40 MHz and a first stage prescaler divide-by-2 clock selection).

Refer to **Table 15-36** for the MPWMSM relative I/O pin implementation.

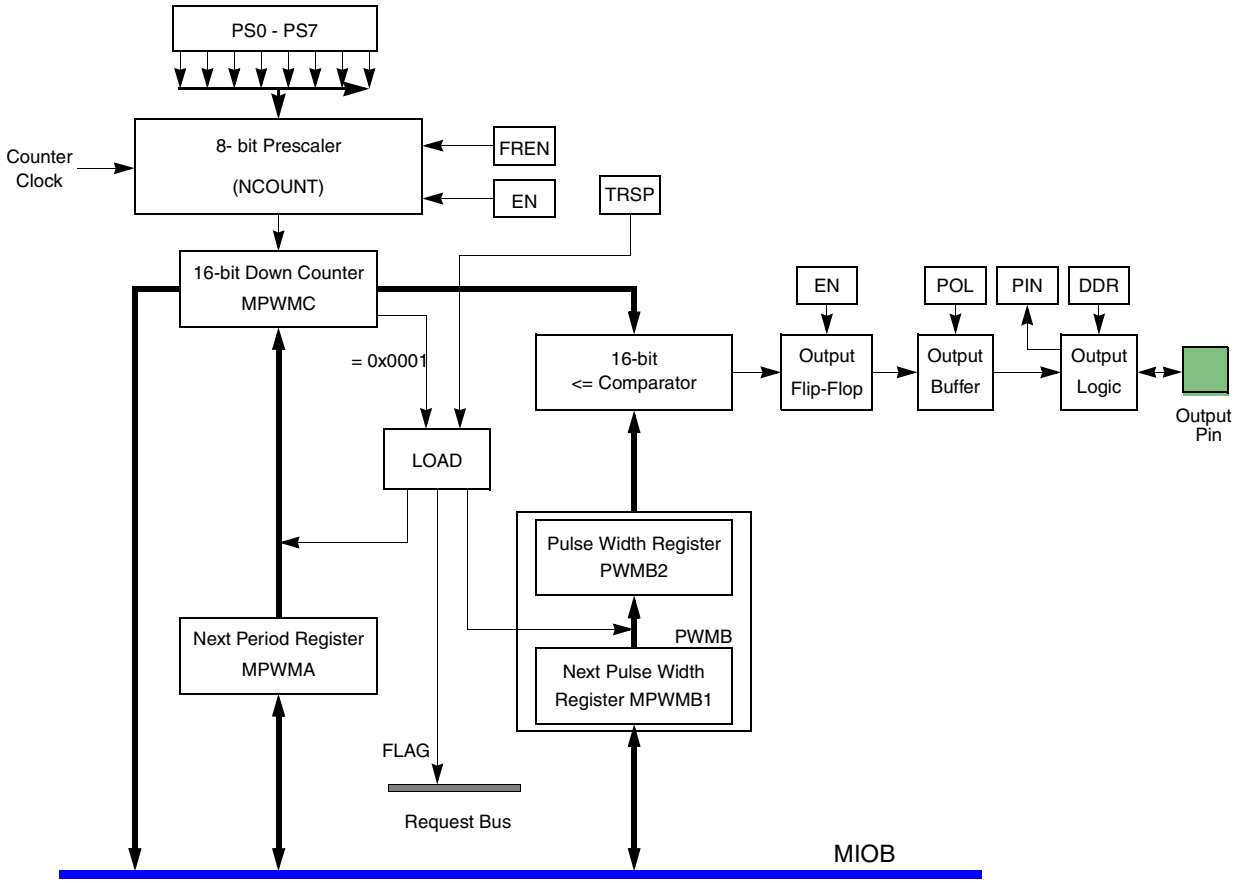


Figure 15-6 MPWMSM Block Diagram

15.12.1 MIOS Pulse Width Modulation Submodule (MPWMSM) Registers

One set of registers is associated with each MPWMSM submodule. The base address is given in the table below.

Table 15-19 MPWMSM Address Map

Address	Register
MPWMSM0	
0x30 6000	MPWMSM0 Period Register (MPWMSMPERR) See **Table 15-20** for bit descriptions.
0x30 6002	MPWMSM0 Pulse Register (MPWMSMPULR) See **Table 15-21** for bit descriptions.
0x30 6004	MPWMSM0 Count Register (MPWMSMCNTR) See **Table 15-22** for bit descriptions.
0x30 6006	MPWMSM0 Status/Control Register (MPWMSMSCR) See **Table 15-23** for bit descriptions.
MPWMSM1	
0x30 6008	MPWMSM1 Period Register (MPWMSMPERR)
0x30 600A	MPWMSM1 Pulse Register (MPWMSMPULR)
0x30 600C	MPWMSM1 Count Register (MPWMSMCNTR)
0x30 600E	MPWMSM1 Status/Control Register (MPWMSMSCR)
MPWMSM2	
0x30 6010	MPWMSM2 Period Register (MPWMSMPERR)
0x30 6012	MPWMSM2 Pulse Register (MPWMSMPULR)
0x30 6014	MPWMSM2 Count Register (MPWMSMCNTR)
0x30 6016	MPWMSM2 Status/Control Register (MPWMSMSCR)
MPWMSM3	
0x30 6018	MPWMSM3 Period Register (MPWMSMPERR)
0x30 601A	MPWMSM3 Pulse Register (MPWMSMPULR)
0x30 601C	MPWMSM3 Count Register (MPWMSMCNTR)
0x30 601E	MPWMSM3 Status/Control Register (MPWMSMSCR)
MPWMSM16	
0x30 6080	MPWMSM16 Period Register (MPWMSMPERR)
0x30 6082	MPWMSM16 Pulse Register (MPWMSMPULR)
0x30 6084	MPWMSM16 Count Register (MPWMSMCNTR)
0x30 6086	MPWMSM16 Status/Control Register (MPWMSMSCR)
MPWMSM17	
0x30 6088	MPWMSM17 Period Register (MPWMSMPERR)
0x30 608A	MPWMSM17 Pulse Register (MPWMSMPULR)
0x30 608C	MPWMSM17 Count Register (MPWMSMCNTR)
0x30 608E	MPWMSM17 Status/Control Register (MPWMSMSCR)
MPWMSM18	
0x30 6090	MPWMSM18 Period Register (MPWMSMPERR)
0x30 6092	MPWMSM18 Pulse Register (MPWMSMPULR)
0x30 6094	MPWMSM18 Count Register (MPWMSMCNTR)
0x30 6096	MPWMSM18 Status/Control Register (MPWMSMSCR)

Table 15-19 MPWMSM Address Map (Continued)

Address	Register
MPWMSM19	
0x30 6098	MPWMSM19 Period Register (MPWMSMPERR)
0x30 609A	MPWMSM19 Pulse Register (MPWMSMPULR)
0x30 609C	MPWMSM19 Count Register (MPWMSMCNTR)
0x30 609E	MPWMSM19 Status/Control Register (MPWMSMSCR)

15.12.1.1 MPWMSM Period Register (MPWMSMPERR)

The period register contains the binary value corresponding to the period to be generated.

MPWMSMPERR — MPWMSM Period Register **0x30 6000***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PER															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

* Refer to **Table 15-19** for a complete list of all the base addresses for the MPWMSM registers.

Table 15-20 MPWMSMPERR Bit Descriptions

Bit(s)	Name	Description
0:15	PER	Period. These bits contain the binary value corresponding to the period to be generated.

15.12.1.2 MPWMSM Pulse Width Register (MPWMSMPULR)

This register contains the binary value of the pulse width to be generated.

MPWMSMPULR — MPWMSM Pulse Width Register **0x30 6002***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PUL															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

* Refer to **Table 15-19** for a complete list of all the base addresses for the MPWMSM registers.

Table 15-21 MPWMSMPULR Bit Descriptions

Bit(s)	Name	Description
0:15	PUL	Pulse width. These bits contain the binary value of the pulse width to be generated.

15.12.1.3 MPWMSM Counter Register (MPWMSMCNTR)

This register reflects the actual value of the MPWMSM counter.

MPWMSMCNTR — MPWMSM Counter Register **0x30 6004***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CNT															
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

* Refer to **Table 15-19** for a complete list of all the base addresses for the MPWMSM registers. A write to the MPWMSMCNTR register also writes the same value to MPWMSMPERR.

Table 15-22 MPWMSMCNTR Bit Descriptions

Bit(s)	Name	Description
0:15	CNT	Counter. These bits reflect the actual value of the MPWMSM counter.

15.12.1.4 MPWMSM Status/Control Register(MPWMSMCR)

This register contains read-only status bits and read/write control bits.

MPWMSMSCR — MPWMSM Status/Control Register **0x30 6006***

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIN	DDR	FREN	TRSP	POL	EN	RESERVED		CP							
RESET:															
—	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U

* Refer to **Table 15-19** for a complete list of all the base addresses for the MPWMSM registers.

Table 15-23 MPWMSMSCR Bit Descriptions

Bit(s)	Name	Description
0	PIN	Pin input status. The PIN bit reflects the state present on the MPWMSM pin. The software can thus monitor the signal on the pin. The PIN bit is a read-only bit. Writing to the PIN bit has no effect.
1	DDR	Data direction register. The DDR bit indicates the direction for the pin when the PWM function is not used (disable mode). Note that when the PWM function is used, the DDR bit has no effect. **Table 15-24** lists the different uses for the polarity (POL) bit, the enable (EN) bit and the data direction register (DDR) bit. 0 = Pin is an input. 1 = Pin is an output.
2	FREN	Freeze enable. This active high read/write control bit enables the MPWMSM to recognize the freeze signal on the MIOB. 0 = MPWMSM not frozen even if the MIOB freeze line is active. 1 = MPWMSM frozen if the MIOB freeze line is active.
3	TRSP	Transparent mode. The TRSP bit indicates that the MPWMSM double buffers are transparent: when the software writes to either the MPWMA or MPWMB1 register the value written is immediately transferred to respectively the counter or register MPWMB2. 0 = Transparent mode de-activated. 1 = Transparent mode activated.
4	POL	Output polarity control. The POL bit works in conjunction with the EN bit and controls whether the MPWMSM drives the pin with the true or the inverted value of the output flip-flop **Table 15-24** lists the different uses for the polarity (POL) bit, the enable (EN) bit and the data direction register (DDR) bit.
5	EN	Enable PWM signal generation. The EN bit defines whether the MPWMSM generates a PWM signal or is used as an I/O channel: **Table 15-24** lists the different uses for the polarity (POL) bit, the enable (EN) bit and the data direction register (DDR) bit. 0 = PWM generation disabled (pin can be used as I/O). 1 = PWM generation enabled (pin is output only).
6:7	—	Reserved
8:15	CP	Clock Prescaler. This 8-bit read/write register stores the two's complement of the desired modulus value for loading into the built-in 8-bit clock prescaler. The value loaded defines the divide ratio for the signal that clocks the MPWMSM period counter. **Table 15-15** gives the clock divide ratio according to the CP values.

Table 15-24 PWMSM Output Pin Polarity Selection

Control Bits			Pin Direction (I/O)	Pin State	Periodic Edge	Variable Edge	Optional Interrupt On
POL	EN	DDR					
0	0	0	I	Input	—	—	—
0	0	1	O	Always Low	—	—	—
0	1	—	O	High Pulse	Falling Edge	Rising Edge	Falling Edge
1	0	0	I	Input	—	—	—
1	0	1	O	Always High	—	—	—
1	1	—	O	Low Pulse	Rising Edge	Falling Edge	Rising Edge

15.13 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM)

An MIOS 16-bit parallel port I/O submodule (MPIOSM) can handle up to 16 input/output pins. Its control register is composed of two 16-bit registers: the data register (DR) and the data direction register (DDR). Each pin of the MPIOSM may be programmed as an input or an output under software control. The direction of a pin is determined by the state of the corresponding bit in the DDR.

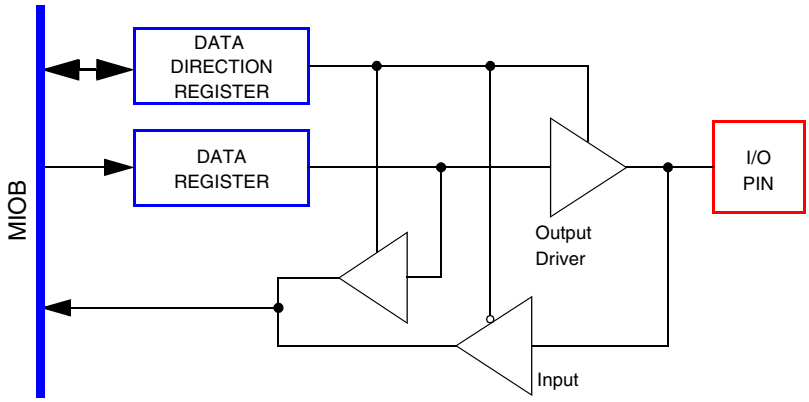


Figure 15-7 MPIOSM One-Bit Block Diagram

Refer to **Table 15-36** for the MPIOSM relative I/O pin implementation.

15.13.1 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM) Registers

One set of registers is associated with the MPIOSM submodule. The base addresses of the submodules are given in the table below.

Table 15-25 MPIOSM Address Map

Address	Register
0x30 6100	MPIOSM Data Register (MPIOSMDR) See **Table 15-26** for bit descriptions.
0x30 6102	MPIOSM Data Direction Register (MPIOSMDDR) See **Table 15-27** for bit descriptions.
0x30 6104	Reserved
0x30 6106	Reserved

15.13.1.1 MPIOSM Data Register (MPIOSMDR)

This read/write register defines the value to be driven to the pad in output mode, for each implemented I/O pin of the MPIOSM.

MPIOSMDR — MPIOSM Data Register **0x30 6100**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 15-26 MPIOSMDR Bit Descriptions

Bit(s)	Name	Description
0:15	D[15:0]	Data BITS. These bits are read/write data bits that define the value to be driven to the pad in output mode for each implemented I/O pin of the MPIOSM. While in output mode, a read returns the value of the pad. Note that, when little-endian bit ordering is used, bit 0 corresponds to D15 and bit 15 corresponds to D0.

NOTE

D[0:4] controls the signals MPIO32B[0:4]. These functions are shared on the MPC555 / MPC556 pins VF[0:2]/MPIO32B[0:2] VFLS[0:1]/MPIO32B[3:4] and can be configured as the alternate function (VF[0:2] and VFLS[0:1]). See **15.8.1.1 MIOS1 Test and Pin Control Register**.

15.13.1.2 MPIOSM Data Direction Register (MPIOSMDDR)

This read/write register defines the data direction for each implemented I/O pin of the MPIOSM.

MPIOSMDDR — MPIOSM Data Direction Register **0x30 6102**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
DDR1	DDR1	DDR1	DDR1	DDR1	DDR10	DDR9	DDR8	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-27 MPIOSMDDR Bit Descriptions

Bit(s)	Name	Description
0:15	DDR[15:0]	Data direction. These bits are read/write data bits that define the data direction status for each implemented I/O pin of the MPIOSM. Note that, when little-endian bit ordering is used, bit 0 corresponds to D15 and bit 15 corresponds to D0. 0 = Corresponding pin is input. 1 = Corresponding pin is output.

15.14 MIOS1 Interrupts

The MIOS1 and its submodules are capable of generating interrupts to be transmitted to the CPU via the IMB3. Inside the MIOS1, all the information required for requesting and servicing the interrupts are treated in two different blocks:

- The interrupt control section (ICS)

- The MIOS interrupt request submodules (MIRSM).

The MIOS interrupt request submodule gathers service request flags from each group of up to 16 submodules and transfers those requests to the MIOS1 interrupt control section (ICS). **Figure 15-8** shows a block diagram of the interrupt architecture.

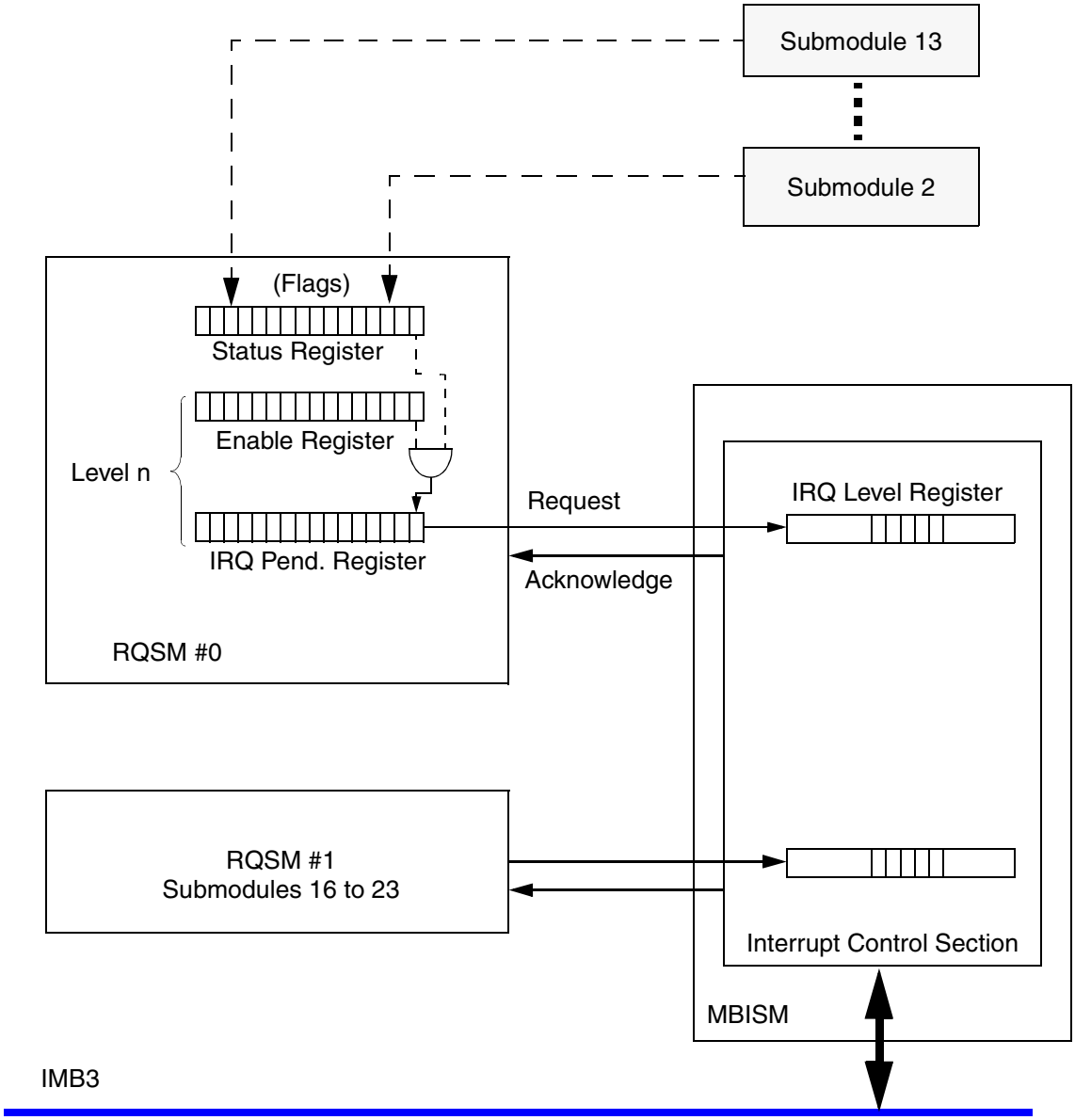


Figure 15-8 MIOS Interrupt Structure

15.14.1 MIOS Interrupt Request Submodule (MIRSM)

Each submodule that is capable of generating an interrupt can assert a flag line when an event occurs. In the MIOS1 configuration, there are eighteen flag lines and two MIRSMs are needed.

Within the MIOS1, each MIRSM includes:

- One 16-bit status register (for the flags)
- One 16-bit enable register
- One 16-bit IRQ pending register

One bit position in each of the above registers is associated with one submodule. Note that if a submodule in a group of 16 cannot generate interrupts, then its corresponding flag bit in the status register is inactive and reads as zero.

When an event occurs in a submodule that activates a flag line, the corresponding flag bit in the status register is set. The status register is read/write, but a flag bit can be reset only if it has previously been read as a one. Writing a one to a flag bit has no effect. When the software intends to clear only one flag bit within a status register, the software must write an 16-bit value of all ones except for a zero in the bit position to be cleared.

The enable register is initialized by the software to indicate whether each interrupt request is enabled for the level defined in the ICS.

Each bit in the IRQ pending register is the result of a logical “AND” between the corresponding bits in the status and in the enable registers. If a flag bit is set and the level enable bit is also set, then the IRQ pending bit is set and the information is transferred to the interrupt control section that is in charge of sending the corresponding level to the CPU. The IRQ pending register is read only.

NOTE

When the enable bit is not set for a particular submodule, the corresponding status register bit is still set when the corresponding flag is set. This allows the traditional software approach of polling the flag bits to see which ones are set. The status register makes flag polling easy, since up to sixteen flag bits are contained in one register.

The submodule number of an interrupting source defines the corresponding MIRSM number and the bit position in the status registers. To find the MIRSM number and bit position of an interrupting source, divide the interrupting submodule number by 16. The integer result of the division gives the MIRSM number. The remainder of the division gives the bit position.

Refer to **15.14.2 MIOS Interrupt Request Submodule 0 (MIRSM0) Registers** and to **15.14.3 MIOS Interrupt Request Submodule 1 (MIRSM1) Registers** for details about the registers in the MIRSM.

15.14.2 MIOS Interrupt Request Submodule 0 (MIRSM0) Registers

Table 15-28 shows the registers associated with the MIRSM0 submodule.

Table 15-28 MIRSM0 Address Map

Address	Register
0x30 6C00	MIRSM0 Interrupt Status Register (MIOS1SR0) See **Table 15-29** for bit descriptions.
0x30 6C02	Reserved
0x30 6C04	MIRSM0 Interrupt Enable Register (MIOS1ER0) See **Table 15-30** for bit descriptions.
0x30 6C06	MIRSM0 Request Pending Register (MIOS1RPR0) See **Table 15-31** for bit descriptions.

15.14.2.1 MIRSM0 Interrupt Status Register (MIOS1SR0)

This register contains flag bits that are set when the associated submodule generates an interrupt. Each bit corresponds to a submodule.

When an event occurs in a submodule that activates a flag line, the corresponding flag bit in the status register is set. The status register is read/write, but a flag bit can be reset only if it has previously been read as a one. Writing a one to a flag bit has no effect. When the software intends to clear only one flag bit within a status register, the software must write an 16-bit value of all ones except for a zero in the bit position to be cleared.

MIOS1SR0 — RQSM0 Interrupt Status Register **0x30 6C00**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
FLG15	FLG14	FLG13	FLG12	FLG11	RESERVED				FLG6	RESERVED		FLG3	FLG2	FLG1	FLG0
RESET:															
U	U	U	U	U	0	0	0	0	U	U	U	U	U	U	U

Table 15-29 MIOS1SR0 Bit Descriptions

Bit(s)	Name	Description
0	FLG15	MDASM15 flag bit
1	FLG14	MDASM14 flag bit
2	FLG13	MDASM13 flag bit
3	FLG12	MDASM12 flag bit
4	FLG11	MDASM11 flag bit
5:8	—	Reserved
9	FLG6	MMCSM6 flag bit
10:11	—	Reserved
12	FLG3	MPWMSM3 flag bit
13	FLG2	MPWMSM2 flag bit
14	FLG1	MPWMSM1 flag bit
15	FLG0	MPWMSM0 flag bit

15.14.2.2 MIRSM0 Interrupt Enable Register (MIOS1ER0)

This read/write register contains interrupt enable bits. Each bit corresponds to a submodule.

MIOS1ER0 — MIRSM0 Interrupt Enable Register **0x30 6C04**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
EN15	EN14	EN13	EN12	EN11	RESERVED				EN6	RESERVED		EN3	EN2	EN1	EN0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-30 MIOS1ER0 Bit Descriptions

Bit(s)	Name	Description
0	EN15	MDASM15 interrupt enable bit
1	EN14	MDASM14 interrupt enable bit
2	EN13	MDASM13 interrupt enable bit
3	EN12	MDASM12 interrupt enable bit
4	EN11	MDASM11 interrupt enable bit
5:8	—	Reserved
9	EN6	MMCSM6 interrupt enable bit
10:11	—	Reserved
12	EN3	MPWMSM3 interrupt enable bit
13	EN2	MPWMSM2 interrupt enable bit
14	EN1	MPWMSM1 interrupt enable bit
15	EN0	MPWMSM0 interrupt enable bit

15.14.2.3 MIRSM0 Request Pending Register (MIOS1RPR0)

This read-only register contains interrupt pending bits. Each bit corresponds to a submodule. A bit that is set indicates that the associated submodule set its flag and that the corresponding enable bit was set.

MIOS1RPR0 — MIRSM0 Request Pending Register **0x30 6C06**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRP15	IRP14	IRP13	IRP12	IRP11	RESERVED				IRP6	RESERVED		IRP3	IRP2	IRP1	IRP0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-31 MIOS1RPR0 Bit Descriptions

Bit(s)	Name	Description
0	IRP15	MDASM15 IRQ pending bit
1	IRP14	MDASM14 IRQ pending bit
2	IRP13	MDASM13 IRQ pending bit
3	IRP12	MDASM12 IRQ pending bit
4	IRP11	MDASM11 IRQ pending bit
5:8	—	Reserved
9	IRP6	MMCSM6 IRQ pending bit
10:11	—	Reserved
12	IRP3	MPWMSM3 IRQ pending bit
13	IRP2	MPWMSM2 IRQ pending bit
14	IRP1	MPWMSM1 IRQ pending bit
15	IRP0	MPWMSM0 IRQ pending bit

15.14.3 MIOS Interrupt Request Submodule 1 (MIRSM1) Registers

Table 15-32 shows the base addresses of the registers associated with the MIRSM1 submodule.

Table 15-32 MIRSM1 Address Map

Address	Register
0x30 6C40	MIRSM1 Interrupt Status Register (MIOS1SR1) See **Table 15-33** for bit descriptions.
0x30 6C42	Reserved
0x30 6C44	MIRSM1 Interrupt Enable Register (MIOS1ER1) See **Table 15-34** for bit descriptions.
0x30 6C46	MIRSM1 Request Pending Register (MIOS1PR1) See **Table 15-35** for bit descriptions.

15.14.3.1 MIRSM1 Interrupt Status Register (MIOS1SR1)

This register contains flag bits that are set when the associated submodule generates an interrupt. Each bit corresponds to a submodule.

MIOS1SR1 — MIRSM1 Interrupt Status Register **0x30 6C40**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
FLG31	FLG30	FLG29	FLG28	FLG27	RESERVED				FLG22	RESERVED		FLG19	FLG18	FLG17	FLG16
RESET:															
U	U	U	U	U	0	0	0	0	U	0	0	U	U	U	U

Table 15-33 MIOS1SR1 Bit Descriptions

Bit(s)	Name	Description
0	FLG31	MDASM31 flag bit
1	FLG30	MDASM30 flag bit
2	FLG29	MDASM29 flag bit
3	FLG28	MDASM28 flag bit
4	FLG27	MDASM27 flag bit
5:8	—	Reserved
9	FLG22	MMCSM22 flag bit
10:11	—	Reserved
12	FLG19	MPWMSM19 flag bit
13	FLG18	MPWMSM18 flag bit
14	FLG17	MPWMSM17 flag bit
15	FLG16	MPWMSM16 flag bit

15.14.3.2 MIRSM1 Interrupt Enable Register (MIOS1ER1)

This read/write register contains interrupt enable bits. Each bit corresponds to a submodule.

MIOS1ER1 — Interrupt Enable Register **0x30 6C44**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
EN31	EN30	EN129	EN28	EN27	RESERVED				EN22	RESERVED		EN19	EN18	EN17	EN16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-34 MIOS1ER1 Bit Descriptions

Bit(s)	Name	Description
0	EN31	MDASM31 interrupt enable bit
1	EN30	MDASM30 interrupt enable bit
2	EN29	MDASM29 interrupt enable bit
3	EN28	MDASM28 interrupt enable bit
4	EN27	MDASM27 interrupt enable bit
5:8	—	Reserved
9	EN22	MMCSM22 interrupt enable bit
10:11	—	Reserved
12	EN19	MPWMSM19 interrupt enable bit
13	EN18	MPWMSM18 interrupt enable bit
14	EN17	MPWMSM17 interrupt enable bit
15	EN16	MPWMSM16 interrupt enable bit

15.14.3.3 MIRSM1 Request Pending Register (MIOS1RPR1)

This read-only register contains interrupt pending bits. Each bit corresponds to a submodule. A bit that is set indicates that the associated submodule set its flag and that the corresponding enable bit was set.

MIOS1RPR1 — MIRSM1 Request Pending Register **0x30 6C46**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRP31	IRP30	IRP29	IRP28	IRP27	RESERVED				IRP22	RESERVED		IRP19	IRP18	IRP17	IRP16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-35 MIOS1RPR1 Bit Descriptions

Bit(s)	Name	Description
0	IRP31	MDASM31 IRQ pending bit
1	IRP30	MDASM30 IRQ pending bit
2	IRP29	MDASM29 IRQ pending bit
3	IRP28	MDASM28 IRQ pending bit
4	IRP27	MDASM27 IRQ pending bit
5:8	—	Reserved
9	IRP22	MMCSM22 IRQ pending bit
10:11	—	Reserved
12	IRP19	MPWMSM19 IRQ pending bit
13	IRP18	MPWMSM18 IRQ pending bit
14	IRP17	MPWMSM17 IRQ pending bit
15	IRP16	MPWMSM16 IRQ pending bit

15.15 MIOS1 Function Examples

The versatility of the MIOS1 timer architecture is based on multiple counters and capture/compare channel units interconnected on 16-bit counter buses. This section includes some typical application examples to show how the submodules can be interconnected to form timing functions. The diagrams used to illustrate these examples show only the blocks utilized for that function.

To illustrate the timing range of the MIOS1 in different applications, many of the following paragraphs include time intervals quoted in microseconds and seconds. The assumptions used are that f_{SYS} is at 40 MHz with minimum overall prescaling (50 ns cycle) and with the maximum overall prescaling (32 µs cycle). For other f_{SYS} clock cycle rates and prescaler choices, the times mentioned in these paragraphs scale appropriately.

15.15.1 MIOS1 Input Double Edge Pulse Width Measurement

To measure the width of an input pulse, the MIOS double action submodule (MDASM) has two capture registers so that only one interrupt is needed after the second edge. The software can read both edge samples and subtract them to get the pulse width. The leading edge sample is double latched so that the software has the time of one full period of the input signal to read the samples to be sure that nothing is lost. Depending on the prescaler divide ratio, pulse width from 50 ns to 6.7 s can be measured. Note that a software option is provided to also generate an interrupt after the first edge.

In the example shown in **Figure 15-9**, a counter submodule is used as the time-base for a MDASM configured in the input pulse width measurement mode. When the leading edge (programmed for being either rising or falling) of the input signal occurs, the state of the 16-bit counter bus is saved in register B1. When the trailing edge occurs, the 16-bit counter bus is latched into register A and the content of register B1 is transferred to register B2. This operation leaves register B1 free for the next leading edge to occur on the next clock cycle. When enabled, an interrupt is provided after the trailing edge, to notify the software that pulse width measurement data is available for a new pulse. After the trailing edge, the software has one cycle time of the input signal to obtain the values for each edge. When software attention is not needed for every pulse, the interrupt can be disabled. The software can read registers A and B2 coherently (using a 32-bit read instruction) at any time, to get the latest edge measurements. The software work is less than half that needed with a timer that requires the software to read one edge and save the value and then wait for the second edge.

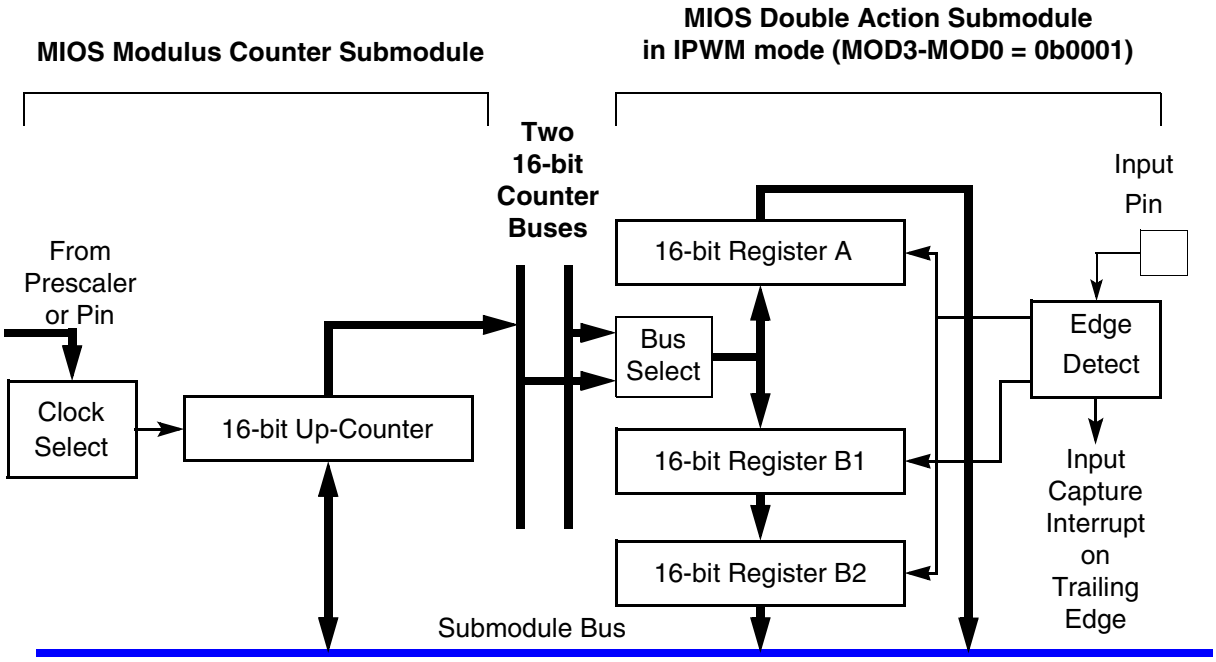


Figure 15-9 MIOS1 Example: Double Capture Pulse Width Measurement

15.15.2 MIOS1 Input Double Edge Period Measurement

Two samples are available to the software from an MIOS double action submodule for period measurement. The software can read the previous and the current edge samples and subtract them. As with pulse width measurement, the software can be sure not to miss samples by ensuring that the interrupt response time is faster than the fastest input period. Alternately, when the software is just interested in the latest period measurement, one 32-bit coherent read instruction can get both the current and the previous samples. Depending on the prescaler divide ratio, period times can be measured from 50 ns to 6.7 s.

Figure 15-10 shows a counter submodule and a DASM combination as an example of period measurement. The software designates whether the rising or falling edge of the input signal is to be used for the measurements. When the edge is detected, the state of the 16-bit counter bus is stored in register A and the content of register B1 is transferred to register B2. After register B2 is safely latched, the content of register A is transferred to register B1. This procedure gives the software coherent current and previous samples in registers A and B2 at all times. An interrupt is available for the cases where the software needs to be aware of each new sample. Note that a software option is provided to also generate an interrupt after the first edge.

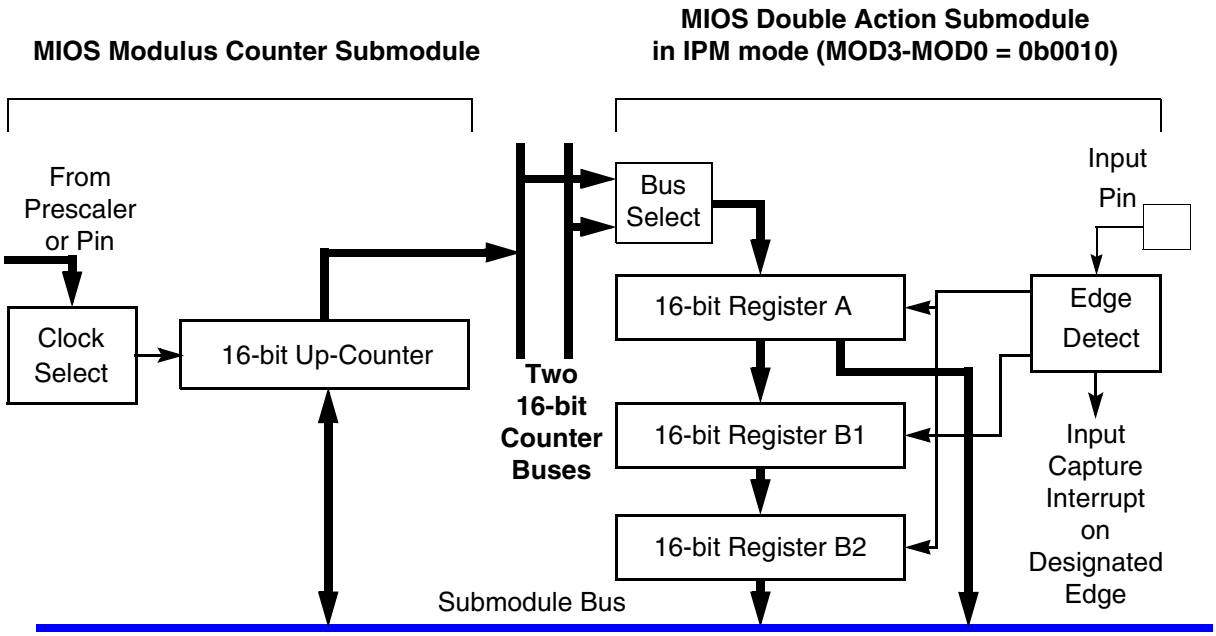


Figure 15-10 MIOS1 Example: Double Capture Period Measurement

15.15.3 MIOS1 Double Edge Single Output Pulse Generation

Software can initialize the MIOS1 to generate both the rising and the falling edge of an output pulse. With a MDASM, pulses as narrow as 50 ns can be generated since software action is not needed between the edges. Pulses as long as 2.1 s can be generated. When an interrupt is desired, it can be selected to occur on every edge or only after the second edge.

Figure 15-11 shows how a counter submodule and a MDASM can be used to generate both edges of a single output pulse. The software puts the compare value for one edge in register A and the other one in register B2. The MDASM automatically creates both edges and the pulse can be selected by software to be a high-going or a low-going. After the trailing edge, the MDASM stops to await further commands from the software. Note that a single edge output can be generated by writing to only one register.

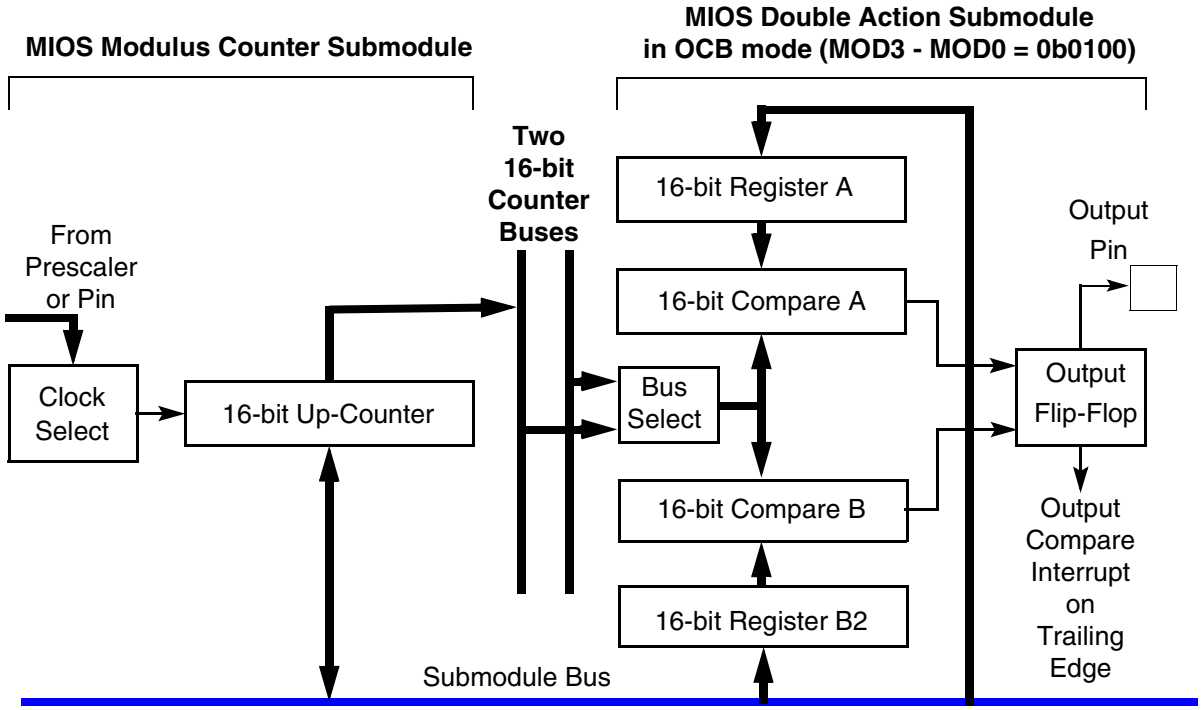


Figure 15-11 MIOS1 Example: Double Edge Output Compare

15.15.4 MIOS1 Output Pulse Width Modulation With MDASM

Output waveforms can be generated with any duty cycle without software involvement. The software sets up a MDASM with the compare times for the rising and falling edges and they are automatically repeated. The software does not need to respond to interrupts to generate continuous pulses. The frequency may be selected as the frequency of a free-running counter time-base, times a binary multiplier selected in the MDASM. Multiple PWM outputs can be created from multiple MDASMs and share one counter submodule, provided that the frequencies of all of the output signals are a binary multiple of the time-base and that the counter submodule is operating in a free-running mode. Each MDASM has a software selectable “don’t care” on high-order bits of the time-base comparison so that the frequency of one output can be a binary multiple of another signal. Masking the time-base serves to multiply the frequency of the time-base by a binary number to form the frequency of the output waveform. The duty cycle can vary from one cycle to 64-Kbyte cycles. The frequency can range from 0.48 Hz to 156 KHz, though the resolution decreases at the higher frequencies to as low as seven bits. The generation of output square wave signals is of course the special case where the high and low times are equal.

When an MMCSM is used to drive the time-base, the modulus value is the period of the output PWM signal. **Figure 15-12** shows such an example. The polarity of the leading edge of an output waveform is programmable for a rising or a falling edge. The software selects the period of the output signal by programming the MMCSM with a modulus value. The leading edge compare value is written into register A by software and the trailing edge time is written into register B1. When the leading edge value is reached, the content of register B1 is transferred to register B2, to form the next trailing edge value. Subsequent changes to the output pulse width are made by writing a new time into register B1. Updates to the pulse width are always synchronized to the leading edge of the waveform.

It is typical to use the pulse width modulation mode of the MDASM without interrupts, although an interrupt can be enabled to occur on the leading edge. When the output is an unchanging repetitive waveform, the MDASM continuously generates the signal without any software intervention. When the software needs to change the pulse width, a new trailing edge time is written to the MDASM. The output is changed on the next full pulse. When the software needs to change the output at a regular rate, such as an acceleration curve, the leading edge interrupt gives the software one period time to update the new trailing edge time.

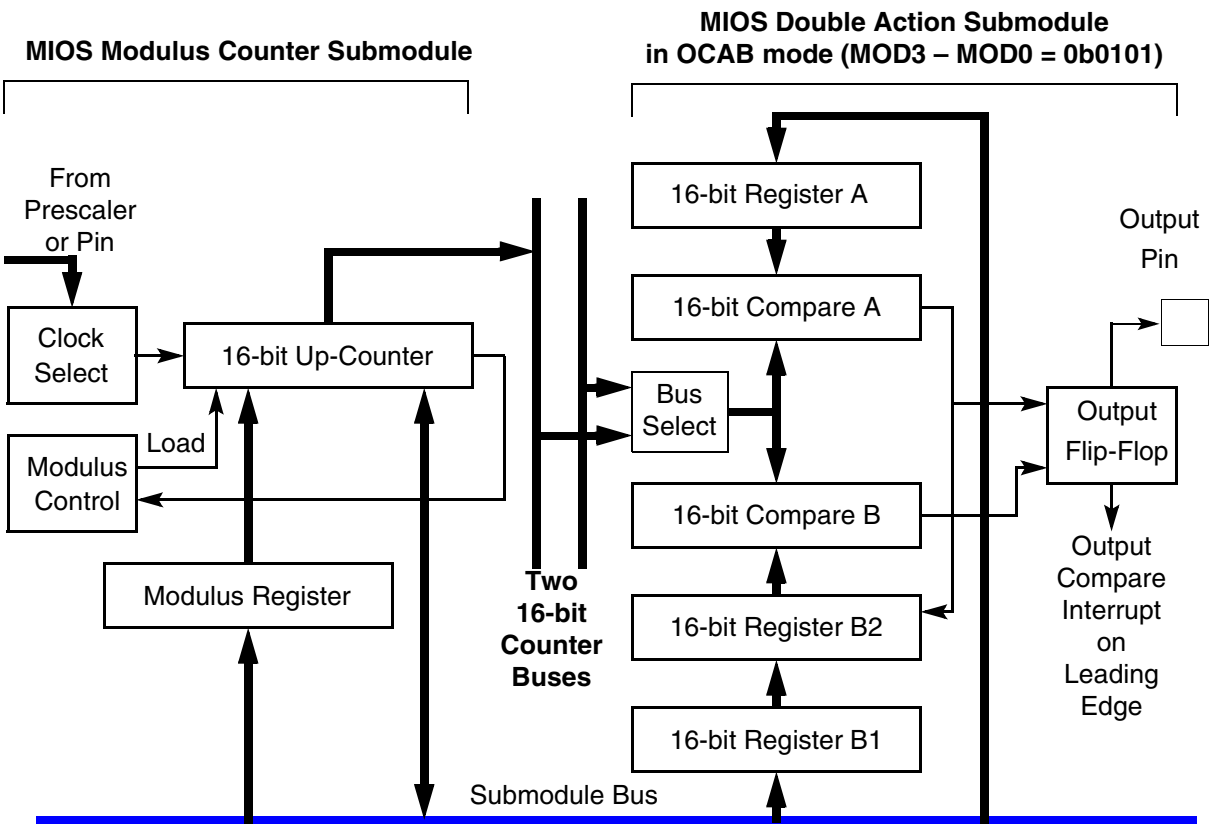


Figure 15-12 MIOS1 Example: Pulse Width Modulation Output

15.15.5 MIOS1 Input Pulse Accumulation

Counting the number of pulses on an input signal is another capability of the MIOS1. Pulse accumulation uses an MMCSM. Since the counters in the counter submodules are software accessible, pulse accumulation does not require the use of an action submodule. The pulse accumulation can operate continuously, interrupting only on binary overflow of the 16-bit counter. When an MMCSM is used, an interrupt can instead be created when the pulse accumulation reaches a preprogrammed value. To do that, the two's complement of the value is put in the modulus register and the interrupt occurs when the counter overflows.

15.16 MIOS1 Configuration

The complete MIOS1 submodule and pin configuration is shown in **Table 15-36**.

Table 15-36 MIOS1 Configuration

Submodule type	Submodule Number	connected to:				RQSM Number	RQSM Bit Position	Base Address	Pin Function	Input Pin Name	Output Pin Name	Alternate Pin Function
		CBA	CBB	CBC	CBD							
		BSL=00	BSL=01	BSL=10	BSL=11							
MPWMSM	0					0	0	0x30 6000	PWM, I/O	MPWM0	MPWM0	
MPWMSM	1					0	1	0x30 6008	PWM, I/O	MPWM1	MPWM1	
MPWMSM	2					0	2	0x30 6010	PWM, I/O	MPWM2	MPWM2	
MPWMSM	3					0	3	0x30 6018	PWM, I/O	MPWM3	MPWM3	
Reserved	4-5											
MMCSM	6	CB6				0	6	0x30 6030	Clock In	MDA11		
									Load In	MDA12		
Reserved	7-10											
MDASM	11	CB6	CB22			0	11	0x30 6058	Channel I/O	MDA11	MDA11	
MDASM	12	CB6	CB22			0	12	0x30 6060	Channel I/O	MDA12	MDA12	
MDASM	13	CB6	CB22			0	13	0x30 6068	Channel I/O	MDA13	MDA13	
MDASM	14	CB6	CB22			0	14	0x30 6070	Channel I/O	MDA14	MDA14	
MDASM	15	CB6	CB22			0	15	0x30 6078	Channel I/O	MDA15	MDA15	
MPWMSM	16					1	0	0x30 6080	PWM, I/O	MPWM16	MPWM16	
MPWMSM	17					1	1	0x30 6088	PWM, I/O	MPWM17	MPWM17	
MPWMSM	18					1	2	0x30 6090	PWM, I/O	MPWM18	MPWM18	
MPWMSM	19					1	3	0x30 6098	PWM, I/O	MPWM19	MPWM19	
Reserved	20-21											
MMCSM	22		CB22			1	6	0x30 60B0	Clock In	MDA13		
									Load In	MDA14		
Reserved	23-26											
MDASM	27	CB6	CB22			1	11	0x30 60D8	Channel I/O	MDA27	MDA27	
MDASM	28	CB6	CB22			1	12	0x30 60E0	Channel I/O	MDA28	MDA28	
MDASM	29	CB6	CB22			1	13	0x30 60E8	Channel I/O	MDA29	MDA29	
MDASM	30	CB6	CB22			1	14	0x30 60F0	Channel I/O	MDA30	MDA30	
MDASM	31	CB6	CB22			1	15	0x30 60F8	Channel I/O	MDA31	MDA31	
MPIOSM	32							0x30 6100	GP I/O[1]	MPIO32B0	MPIO32B0	VF0
									GP I/O	MPIO32B1	MPIO32B1	VF1
									GP I/O	MPIO32B2	MPIO32B2	VF2
									GP I/O	MPIO32B3	MPIO32B3	VFLS0
									GP I/O	MPIO32B4	MPIO32B4	VFLS1
									GP I/O	MPIO32B5	MPIO32B5	
									GP I/O	MPIO32B6	MPIO32B6	
									GP I/O	MPIO32B7	MPIO32B7	
									GP I/O	MPIO32B8	MPIO32B8	
									GP I/O	MPIO32B9	MPIO32B9	

Table 15-36 MIOS1 Configuration (Continued)

Submodule type	Submodule Number	connected to:				RQSM Number	RQSM Bit Position	Base Address	Pin Function	Input Pin Name	Output Pin Name	Alternate Pin Function
		CBA	CBB	CBC	CBD							
		BSL=00	BSL=01	BSL=10	BSL=11							
									GP I/O	MPIO32B10	MPIO32B10	
									GP I/O	MPIO32B11	MPIO32B11	
									GP I/O	MPIO32B12	MPIO32B12	
									GP I/O	MPIO32B13	MPIO32B13	
									GP I/O	MPIO32B14	MPIO32B14	
									GP I/O	MPIO32B15	MPIO32B15	
Reserved	33-255											
MBISM	256							0x30 6800				
Reserved	257											
MCPSM	258							0x30 6810				
Reserved	259											
RQSM0	384-391							0x30 6C00				
RQSM1	392-399							0x30 6C40				
Reserved	400-511											

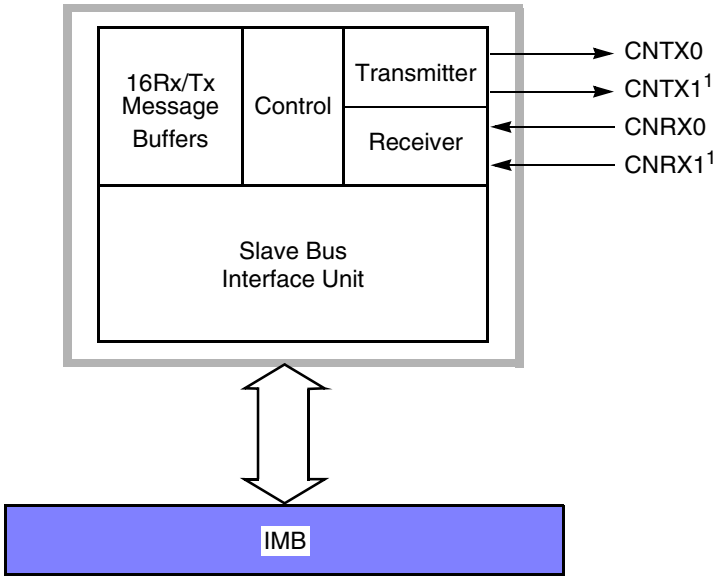
NOTES:
1. GP = General purpose.

SECTION 16
CAN 2.0B CONTROLLER MODULE

The MPC555 / MPC556 contains two CAN 2.0B controller modules (TouCAN). Each TouCAN is a communication controller that implements the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1 Mbit/sec), short distance, priority based protocol that can run over a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The TouCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B.

Each TouCAN module contains 16 message buffers, which are used for transmit and receive functions. It also contains message filters, which are used to qualify the received message IDs when comparing them to the receive buffer identifiers.

Figure 16-1 shows a block diagram of a TouCAN module.



1. In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.

Figure 16-1 TouCAN Block Diagram

16.1 Features

Each TouCAN module provides these features:

- Full implementation of CAN protocol specification, version 2.0 A/B
  — Standard data and remote frames (up to 109 bits long)

— Extended data and remote frames (up to 127 bits long)
— 0 to 8 bytes data length
— Programmable bit rate up to 1 Mbit/sec

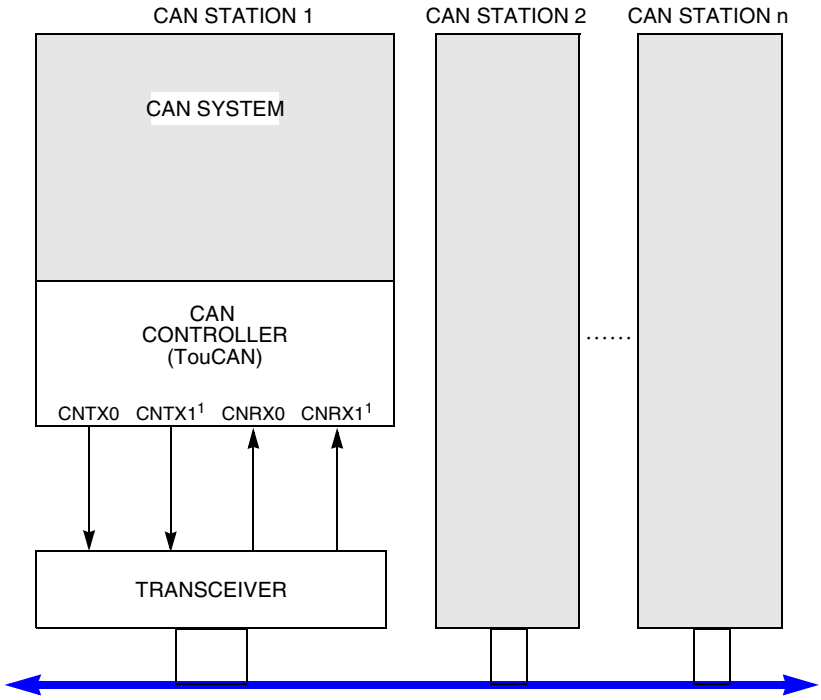
- 16 RX/TX message buffers of 0-8 bytes data length
- Content-related addressing
- No read/write semaphores required
- Three programmable mask registers: global (for message buffers zero through 13), special for message buffer 14, and special for message buffer 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- “Time stamp”, based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity
- Outputs have open drain drivers
- Can be used to implement recommended practices SAE J1939 and SAE J2284
- Can also be used to implement popular industrial automation standards such as DeviceNet™ and Smart Distributed System
- Motorola IMB-family modular architecture

16.2 External Pins

The TouCAN module interface to the CAN bus consists of four pins: CANTX0 and CANTX1, which transmit serial data, and CANRX0 and CANRX1, which receive serial In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.

NOTE

In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.



1. In the MPC555, the CNTX1 and CNRX1 signals are not available.

Figure 16-2 Typical CAN Network

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the TouCAN caused by a defective CAN bus or a defective CAN station.

16.3 TouCAN Architecture

The TouCAN module uses a flexible design that allows each of its 16 message buffers to be designated either a transmit (Tx) buffer or a receive (Rx) buffer. In addition, to reduce the CPU overhead required for message handling, each message buffer is assigned an interrupt flag bit to indicate that the transmission or reception completed successfully.

16.3.1 TX/RX Message Buffer Structure

Figure 16-3 displays the extended (29-bit) ID message buffer structure. **Figure 16-4** displays the standard (11-bit) ID message buffer structure.

	MSB 0	7	8	11	12			LSB 15	
0x0	TIME STAMP		CODE		LENGTH				CONTROL/STATUS
0x2	ID[28-18]			SRR	IDE	ID[17-15]			ID_HIGH
0x4	ID[14-0]							RTR	ID_LOW
0x6	DATA BYTE 0			DATA BYTE 1					
0x8	DATA BYTE 2			DATA BYTE 3					
0xA	DATA BYTE 4			DATA BYTE 5					
0xC	DATA BYTE 6			DATA BYTE 7					
0xE	RESERVED[1]								

NOTES:
1. The reading of a reserved location in memory may cause an RCPU exception.

Figure 16-3 Extended ID Message Buffer Structure

	MSB 0	7	8	11	12			LSB 15	
0x0	TIME STAMP		CODE		LENGTH				CONTROL/STATUS
0x2	ID[28:18]			RTR	0	0	0	0	ID_HIGH
0x4	16-BIT TIME STAMP								ID_LOW
0x6	DATA BYTE 0			DATA BYTE 1					
0x8	DATA BYTE 2			DATA BYTE 3					
0xA	DATA BYTE 4			DATA BYTE 5					
$C	DATA BYTE 6			DATA BYTE 7					
0xE	RESERVED[1]								

NOTES:
1. The reading of a reserved location in memory may cause an RCPU exception.

Figure 16-4 Standard ID Message Buffer Structure

16.3.1.1 Common Fields for Extended and Standard Format Frames

Table 16-1 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 16-1 Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to **Table 16-2** and **Table 16-3**.
Rx Length	Length (in bytes) of the Rx data stored in offset 0x6 through 0xD of the buffer.This field is written by the TouCAN module, copied from the DLC (data length code) field of the received frame.
Tx Length	Length (in bytes) of the data to be transmitted, located in offset 0x6 through 0xD of the buffer. This field is written by the CPU and is used as the DLC field value. If RTR (remote transmission request) = 1, the frame is a remote frame and will be transmitted without data field, regardless of the value in Tx length.
Data	This field can store up to eight data bytes for a frame. For Rx frames, the data is stored as it is received from the bus. For Tx frames, the CPU provides the data to be transmitted within the frame.
Reserved	The CPU controls access to this word entry field (16 bits).

Table 16-2 Message Buffer Codes for Receive Buffers

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0b0000	NOT ACTIVE — message buffer is not active.	—	—
0b0100	EMPTY — message buffer is active and empty.	0b0010	—
0b0010	FULL — message buffer is full.	0b0110	If a CPU read occurs before the new frame, new receive code is 0010.
0b0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0b0XY1[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0b0010	An empty buffer was filled (XY was 10).
		0b0110	A full/overrun buffer was filled (Y was 1).

NOTES:
1. For Tx message buffers, upon read, the BUSY bit should be ignored.

Table 16-3 Message Buffer Codes for Transmit Buffers

RTR	Initial Tx Code	Description	Code After Successful Transmission
X	0b1000	Message buffer not ready for transmit.	—
0	0b1100	Data frame to be transmitted once, unconditionally.	0b1000
1	0b1100	Remote frame to be transmitted once, and message buffer becomes an RX message buffer for data frames.	0b0100
0	0b1010[1]	Data frame to be transmitted only as a response to a remote frame, always.	0b1010
0	0b1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame, always.	0b1010

NOTES:
1. When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

16.3.1.2 Fields for Extended Format Frames

Table 16-4 describes the message buffer fields used only for extended identifier format frames.

Table 16-4 Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one by the user for Tx buffers. It will be stored as received on the CAN bus for Rx buffers.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 = Data Frame, 1 = Remote Frame.

16.3.1.3 Fields for Standard Format Frames

Table 16-5 describes the message buffer fields used only for standard identifier format frames.

Table 16-5 Standard Format Frames

Field	Description
16-bit Time Stamp	The ID LOW word, which is not needed for standard format, is used in a standard format buffer to store the 16-bit value of the free-running timer which is captured at the beginning of the identifier field of the frame on the CAN bus.
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the TouCAN.
RTR	This bit is located in the ID HIGH word of the message buffer; 0 = data frame, 1 = remote frame.
RTR/SRR Bit Treatment	If the TouCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the TouCAN transmits this bit as a zero and is receives it as a one, a bit error is indicated. If the TouCAN transmits a value and receives a matching response, a successful bit transmission is indicated.

16.3.1.4 Serial Message Buffers

To allow double buffering of messages, the TouCAN has two shadow buffers called serial message buffers. The TouCAN uses these two buffers for buffering both received messages and messages to be transmitted. Only one serial message buffer is active at a time, and its function depends upon the operation of the TouCAN at that time. At no time does the user have access to or visibility of these two buffers.

16.3.1.5 Message Buffer Activation/Deactivation Mechanism

Each message buffer must be activated once the user configures it for the desired operation. A buffer is activated by writing the appropriate code to the control/status word for that buffer. Once the buffer is activated, it will begin participating in the normal transmit and receive processes.

A buffer is deactivated by writing the appropriate deactivation code to the control/status word for that buffer. A buffer is typically deactivated when the user desires to reconfigure the buffer (for example to change the buffer's function from Rx to Tx or Tx to Rx). The buffer should also be deactivated before changing a receive buffer's message identifier or before loading a new message to be transmitted into a transmit buffer.

For more details on activation and deactivation of message buffers and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.1.6 Message Buffer Lock/Release/Busy Mechanism

In addition to the activation/deactivation mechanism, the TouCAN also uses a lock/release/busy mechanism to ensure data coherency during the receive process. The mechanism includes a lock status for each message buffer and uses the two serial message buffers to facilitate frame transfers within the TouCAN.

Reading the control/status word of a receive message buffer triggers the lock for that buffer. While locked, a received message cannot be transferred into that buffer from one of the serial message buffers.

If a message transfer between the message buffer and a serial message buffer is in progress when the control/status word is read, the BUSY status is indicated in the code field, and the lock is not activated.

The user can release the lock on a message buffer in one of two ways. Reading the control/status word of another message buffer locks that buffer, releasing the previously locked buffer. A global release can also be performed on any locked message buffer by reading the free-running timer.

Once a lock is released, any message transfers between a serial message buffer and a message buffer that were delayed due to that buffer being locked will take place. For more details on the message buffer locking mechanism, and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.2 Receive Mask Registers

The receive mask registers are used as acceptance masks for received frame IDs. The following masks are defined:

- A global mask, used for receive buffers 0-13
- Two separate masks for buffers 14 and 15

The value of the mask registers should not be changed during normal operation. If the mask register data is changed after the masked identifier of a received message is

matched to a locked message buffer, that message will be transferred into that message buffer once it is unlocked, regardless of whether that message's masked identifier still matches the receive buffer identifier. **Table 16-6** shows mask bit values.

Table 16-6 Receive Mask Register Bit Values

Mask Bit	Values
0	The corresponding incoming ID bit is "don't care"
1	The corresponding ID bit is checked against the incoming ID bit to see if a match exists

Table 16-7 shows mask examples for normal and extended messages. Refer to **16.7 Programmer's Model** for more information on Rx mask registers.

Table 16-7 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	1 1 1 1 1 1 1 1 0 0 0	0	—	—
MB3	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB4	0 0 0 0 0 0 0 1 1 1 1	0	—	—
MB5	0 0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB14	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
RX Global Mask	1 1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1	—
RX Message In	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
	1 1 1 1 1 1 1 1 0 0 1	0	—	2[2]
	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	__[3]
	0 1 1 1 1 1 1 1 0 0 0	0	—	__[4]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	__[5]
RX 14 Mask	0 1 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0	—
RX Message In	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	__[6]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

16.3.3 Bit Timing

The TouCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers one and two (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the IMB clock. The time

quanta clock operates at the S-clock frequency. **Table 16-8** provides examples of IMB clock, CAN bit rate, and S-clock bit timing parameters. Refer to **16.7 Programmer's Model** for more information on the bit timing registers.

Table 16-8 Example IMB Clock, CAN Bit Rate and S-Clock Frequencies

IMB Clock Frequency (MHz)	CAN Bit Rate (MHz)	Possible S-Clock Frequency (MHz)	Possible Number of Time Quanta/Bit	PRESDIV Value + 1
25	1	25	25	1
20	1	10, 20	10, 20	2, 1
16	1	8, 16	8, 16	2, 1
25	0.125	1, 1.25, 2.5	8,10, 20	25, 20,10
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8
16	0.125	1, 2	8,16	16, 8

16.3.3.1 Configuring the TouCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single IMB clock per one time quantum, then the PSEG2 field in CANCTRL2 register must not be programmed to zero.
- If the programmed PRESDIV value results in a single IMB clock per one time quantum, then the information processing time (IPT) equals three time quanta; otherwise it equals two time quanta. If PSEG2 equals two, then the TouCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than 10 IMB clock periods per CAN bit time and the CAN bus loading is 100%, then any time the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the TouCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The TouCAN bit time must be programmed to be greater than or equal to nine IMB clocks, or correct operation is not guaranteed.

16.3.4 Error Counters

The TouCAN has two error counters, the transmit (Tx) error counter and the receive (Rx) error counter. Refer to **16.7 Programmer's Model** for more information on error counters.The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the TouCAN. Each counter has the following features:

- 8-bit up/down-counter
- Increment by eight (Rx error counter also increments by one)
- Decrement by one
- Avoid decrement when equal to zero

- Rx error counter reset to a value between 119 and 127 inclusive, when the TouCAN transitions from error passive to error active
- Following reset, both counters reset to zero
- Detect values for error passive, bus off and error active transitions
- Cascade usage of Tx error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read-only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the Tx error counter or Rx error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the Tx error counter or Rx error counter decrements to a value less than or equal to 127 while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the Tx error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the Tx error counter is reset to zero.
- If the TouCAN is in the bus off state, the Tx error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the Tx error counter is first reset to zero, and then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the Tx error counter is incremented by one and the internal counter is reset to zero. When the Tx error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero but does not affect the Tx error counter value.
- If only one node is operating in a system, the Tx error counter is incremented with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors never cause the TouCAN to change from the error passive state to the bus off state.
- If the Rx error counter increments to a value greater than 127, it stops incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

16.3.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp is stored in the time stamp entry of the receive message buffer at the time the message is written into

that buffer. For a message being transmitted, the time stamp entry is written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed.

16.4 TouCAN Operation

The basic operation of the TouCAN can be divided into three areas:

- Reset and initialization of the module
- Transmit message handling
- Receive message handling

Example sequences for performing each of these processes is given in the following paragraphs.

16.4.1 TouCAN Reset

The TouCAN can be reset in two ways:

- Hard reset, using one of the IMB3 reset lines
- Soft reset, using the SOFTRST bit in the module configuration register

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

16.4.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes

   a. Initialize the transmit and receive pin modes in control register 0 (CANCTRL0)

   b. Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2])

   c. Select the S-clock rate by programming the PRESDIV register

   d. Select the internal arbitration mode (LBUF bit in CANCTRL1)
2. Initialize message buffers

a. The control/status word of all message buffers must be written either as an active or inactive message buffer.

b. All other entries in each message buffer should be initialized as required

3. Initialize mask registers for acceptance mask as needed
4. Initialize TouCAN interrupt handler

a. Initialize the interrupt configuration register (CANICR) with a specific request level

b. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt

5. Negate the HALT bit in the module configuration register

a. At this point, the TouCAN attempts to synchronize with the CAN bus

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

16.4.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. For the user, this involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare or change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = 0b1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active Tx code, Tx length)

NOTE

Steps one and four are mandatory to ensure data coherency.

Once an active transmit code is written to a transmit message buffer, that buffer begins participating in an internal arbitration process as soon as the receiver senses that the CAN bus is free, or at the inter-frame space. If there are multiple messages awaiting transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

The TouCAN transmits no more than eight data bytes, even if the transmit length contains a value greater than eight.

At the end of a successful transmission, the value of the free-running timer (which was captured at the beginning of the identifier field on the CAN bus), is written into the time stamp field in the message buffer. The code field in the control/status word of the message buffer is updated and a status flag is set in the IFLAG register.

16.4.3.1 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the user deactivates a transmit message buffer while a message is being transferred from it to a serial message buffer, the message is not transmitted.

If the user deactivates the transmit message buffer after the message is transferred to the serial message buffer, the message is transmitted, but no interrupt is requested, and the transmit code is not updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

16.4.3.2 Reception of Transmitted Frames

The TouCAN receives a frame it has transmitted if an empty message buffer with a matching identifier exists.

16.4.4 Receive Process

During the receive process, the following events occur:

- The user configures the message buffers for reception
- The TouCAN transfers received messages from the serial message buffers to the receive message buffers with matching IDs
- The user retrieves these messages

The user should prepare or change a message buffer for frame reception by executing the following steps.

1. Write the control/status word to hold the receive buffer inactive (code = 0b0000)
2. Write the ID_HIGH and ID_LOW words
3. Write the control/status word to mark the receive message buffer as active and empty

NOTE

Steps one and three are mandatory for data coherency.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time the TouCAN receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one. If a match is detected, the following actions occur:

1. The frame is transferred to the first (lowest entry) matching receive message buffer
2. The value of the free-running timer (captured at the beginning of the identifier field on the CAN bus) is written into the time stamp field in the message buffer
3. The ID field, data field, and RX length field are stored
4. The code field is updated
5. The status flag is set in the IFLAG register

The user should read a received frame from its message buffer in the following order:

1. Control/status word (mandatory, as it activates the internal lock for this buffer)
2. ID (optional, since it is needed only if a mask was used)
3. Data field word(s)
4. Free-running timer (optional, as it releases the internal lock)

If the free running timer is not read, that message buffer remains locked until the read process starts for another message buffer. Only a single message buffer is locked at a time. When a received message is read, the only mandatory read operation is that of the control/status word. This ensures data coherency.

If the BUSY bit is set in the message buffer code, the CPU should defer accessing that buffer until this bit is negated. Refer to **Table 16-2**.

NOTE

The user should check the status of a message buffer by reading the status flag in the IFLAG register and not by reading the control/status word code field for that message buffer. This prevents the buffer from being locked inadvertently.

Because the received identifier field is always stored in the matching receive message buffer, the contents of the identifier field in a receive message buffer may change if one or more of the ID bits are masked.

16.4.4.1 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data must never be written into a receive message buffer. If this occurs while a message is being transferred from a serial message buffer, the control/status word will reflect a full or overrun condition, but no interrupt is requested.

16.4.4.2 Locking and Releasing Message Buffers

The lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the the lock/release/busy mechanism affects TouCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame which matches the message buffer cannot be written into this message buffer while it is locked.
2. To release a locked message buffer, the CPU either locks another message buffer by reading its control/status word or globally releases any locked message buffer by reading the free-running timer.
3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame is not immediately transferred into that message buffer, but remains in the serial message buffer. There is no indication when this occurs.
4. When a locked message buffer is released, if a frame with a matching identifier exists within the serial message buffer, then this frame is transferred to the matching message buffer.
5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication when this occurs.
6. If the user reads the control/status word of a receive message buffer while a frame is being transferred from a serial message buffer, the BUSY code is indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word does not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should be followed by a read of the control/status word of another buffer, or by a read of the free-running timer, to ensure that the locked buffer is unlocked.

16.4.5 Remote Frames

The remote frame is a message frame that is transmitted to request a data frame. The TouCAN can be configured to transmit a data frame automatically in response to a remote frame, or to transmit a remote frame and then wait for the responding data frame to be received.

To transmit a remote frame, the user initializes a message buffer as a transmit message buffer with the RTR bit set to one. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer, with the same ID as the remote frame that was transmitted.

When the TouCAN receives a remote frame, it compares the remote frame ID to the IDs of all transmit message buffers programmed with a code of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit in the matching transmit message buffer is set, the TouCAN transmits a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

16.4.6 Overload Frames

The TouCAN does not initiate overload frame transmissions unless it detects the following conditions on the CAN bus:

- A dominant bit in the first or second bit of intermission
- A dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames
- A dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter

16.5 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto power save mode

16.5.1 Debug Mode

Debug mode is entered when the FRZ1 bit in CANMCR is set and one of the following events occurs:

- The HALT bit in the CANMCR is set; or
- The IMB3 FREEZE line is asserted

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. Once this happens, the following events occur:

- The TouCAN stops transmitting or receiving frames
- The prescaler is disabled, thus halting all CAN bus communication
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set
- The CPU is allowed to read and write the error counter registers

After engaging one of the mechanisms to place the TouCAN in debug mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the TouCAN; otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

Once debug mode is exited, the TouCAN resynchronizes with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

16.5.2 Low-Power Stop Mode

Before entering low-power stop mode, the TouCAN waits for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The TouCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Then the following events occur:

- The TouCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set

To exit low-power stop mode:

- Reset the TouCAN either by asserting one of the IMB3 reset lines or by asserting the SOFTRST bit CANMCR.
- Clear the STOP bit in CANMCR.
- The TouCAN module can optionally exit low-power stop mode via the self wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the TouCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the TouCAN clears the STOP bit in CANMCR and its clocks begin running.

When the TouCAN is in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event generates an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self wake mechanism is activated, the TouCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit.) It will not arbitrate for the CAN bus at this time.
- If the STOP bit is set while the TouCAN is in the bus off state, then the TouCAN enters low-power stop mode and stops counting recessive bit times. The count continues when STOP is cleared.
- To place the TouCAN in low-power stop mode with the self wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, and then wait

for the TouCAN to set the STOPACK bit.

- To take the TouCAN out of low-power stop mode when the self wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, and then wait for the TouCAN to clear the STOPACK bit.
- The SELFWAKE bit should not be set after the TouCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the TouCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the TouCAN awakes from low-power stop mode via the self wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the TouCAN in low-power stop mode.
- If the TouCAN is in debug mode when the STOP bit is set, the TouCAN assumes that debug mode should be exited. As a result, it tries to synchronize with the CAN bus, and only then does it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the TouCAN does not come out of reset in low-power stop mode. The basic TouCAN initialization procedure should be executed before placing the module in low-power stop mode. (Refer to **16.4.2 TouCAN Initialization**.)
- If the TouCAN is in low-power stop mode with the self wake mechanism engaged and is operating with a single IMB clock per time quantum, there can be extreme cases in which TouCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. TouCAN synchronization is shifted one time quantum from the wake-up event. This shift lasts until the next recessive-to-dominant edge, which resynchronizes the TouCAN to be in conformance with the CAN protocol. The same holds true when the TouCAN is in auto power save mode and awakens on a recessive to dominant edge.

16.5.3 Auto Power Save Mode

Auto power save mode enables normal operation with optimized power savings. Once the auto power save (APS) bit in CANMCR is set, the TouCAN looks for a set of conditions in which there is no need for the clocks to be running. If these conditions are met, the TouCAN stops its clocks, thus saving power. The following conditions activate auto power save mode.

- No Rx/Tx frame in progress
- No transfer of Rx/Tx frames to and from a serial message buffer, and no Tx frame awaiting transmission in any message buffer
- No CPU access to the TouCAN module
- The TouCAN is not in debug mode, low-power stop mode, or the bus off state

While its clocks are stopped, if the TouCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The TouCAN then continues to monitor these conditions and stops or restarts its clocks accordingly.

16.6 Interrupts

The TouCAN can generate one interrupt level on the IMB. This level is programmed into the priority level bits in the interrupt configuration register (CANICR). This value determines which interrupt signal is driven onto the bus when an interrupt is requested.

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between transmit and receive interrupts for a particular buffer. Each of the buffers is assigned a bit in the IFLAG register. An IFLAG bit is set when the corresponding buffer completes a successful transmission/reception. An IFLAG bit is cleared when the CPU reads IFLAG while the associated bit is set, and then writes it back as zero (and no new event of the same type occurs between the read and the write actions).

The other three interrupt sources (bus off, error and wake up) act in the same way, and have flag bits located in the error and status register (ESTAT). The bus off and error interrupt mask bits (BOFFMSK and ERRMSK) are located in CANCTRL0, and the wake up interrupt mask bit (WAKEMSK) is located in the module configuration register. Refer to **16.7 Programmer's Model** for more information on these registers.

The TouCAN module is capable of generating one of the 32 possible interrupt levels on the IMB3. The 32 interrupt levels are time multiplexed on the IMB3 $\overline{\text{IRQ}}$[0:7] lines. All interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 16-9 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The level that the TouCAN will drive onto $\overline{\text{IRQ}}$[7:0] is programmed in the three interrupt request level (IRL) bits located in the interrupt configuration register. The two ILBS bits in the ICR register determine on which slot the TouCAN should drive its interrupt signal. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 16-5** displays the interrupt levels on IRQ with ILBS.

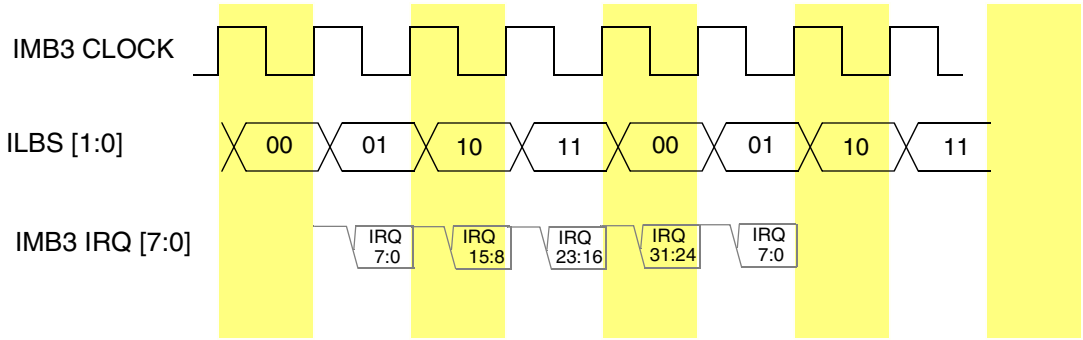


Figure 16-5 Interrupt levels on IRQ with ILBS

16.7 Programmer's Model

Table 16-10 shows the TouCAN address map. The lowercase "x" appended to each register name represents "A" or "B" for the TouCAN_A or TouCAN_B module, respectively. Refer to **1.3 MPC555 / MPC556 Address Map** to locate each TouCAN module in the MPC555 / MPC556 address map.

The column labeled "Access" indicates the privilege level at which the CPU must be operating to access the register. A designation of "S" indicates that supervisor mode is required. A designation of "S/U" indicates that the register can be programmed for either supervisor mode access or unrestricted access.

The address space for each TouCAN module is split, with 128 bytes starting at the base address, and an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures. Of the lower 128 bytes, some are not used. Registers with bits marked as "reserved" should always be written as logic 0.

Typically, the TouCAN control registers are programmed during system initialization, before the TouCAN becomes synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the TouCAN module. This is done by setting the HALT bit in the TouCAN module configuration register (CANMCR). The TouCAN responds by asserting the CANMCR NOTRDY bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

NOTE

The TouCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Table 16-10 TouCAN Register Map

Access	Offset	MSB 0	LSB 15
S	0x30 7080, 0x30 7480	TouCAN Module Configuration Register (TCNMCR_x) See **Table 16-11** for bit descriptions.	
S	0x30 7082, 0x30 7482	TouCAN Test Register (CANTCR_x)	
S	0x30 7084, 0x30 7484	TouCAN Interrupt Register (CANICR_x)	
S/U	0x30 7086, 0x30 7486	Control Register 0 (CANCTRL0_x) See **Table 16-13** and **Table 16-16** for bit descriptions.	Control Register 1 (CANCTRL1_x)
S/U	0x30 7088, 0x30 7488	Control and Prescaler Divider Register (PRESDIV_x) See **Table 16-17** and **Table 16-18** for bit descriptions.	Control Register 2 (CTRL2_x)
S/U	0x30 708A, 0x30 748A	Free-Running Timer Register (TIMER_x) See **Table 16-19** for bit descriptions.	
—	0x30 708C, 0x30 748C — 0x30 708E, 0x30 748E	Reserved	
S/U	0x30 7090, 0x30 7490	Receive Global Mask – High (RXGMSKHI_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7092, 0x30 7492	Receive Global Mask – Low (RXGMSKLO_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7094, 0x30 7494	Receive Buffer 14 Mask – High (RX14MSKHI_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7096, 0x30 7496	Receive Buffer 14 Mask – Low (RX14MSKLO_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7098, 0x30 7498	Receive Buffer 15 FMask – High (RX15MSKHI_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
S/U	0x30 709A, 0x30 749A	Receive Buffer 15 Mask – Low (RX15MSKLO_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
—	0x30 709C, 0x30 749C — 0x30 709E, 0x30 749E	Reserved	
S/U	0x30 70A0, 0x30 74A0	Error and Status Register (ESTAT_x) See **Table 16-21** for bit descriptions.	
S/U	0x30 70A2, 0x30 74A2	Interrupt Masks (IMASK_x) See **Table 16-24** for bit descriptions.	
S/U	0x30 70A4, 0x30 74A4	Interrupt Flags (IFLAG_x) See **Table 16-25** for bit descriptions.	
S/U	0x30 70A6, 0x30 74A6	Receive Error Counter (RXECTR_x) See **Table 16-26** for bit descriptions.	Transmit Error Counter (TXECTR_x)

Table 16-10 TouCAN Register Map (Continued)

Access	Offset	MSB 0	LSB 15
S/U	0x30 7100 — 0x30 710F(A) 0x30 7500 — 0x30 750F(B)	MBUFF0[1] TouCAN_A Message Buffer 0. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7110 — 0x30 711F(A) 0x30 7510 — 0x30 751F(B)	MBUFF1[1] TouCAN_A Message Buffer 1. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7120 — 0x30 712F(A) 0x30 7520 — 0x30 752F(B)	MBUFF2[1] TouCAN_A Message Buffer 2. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7130 — 0x30 713F(A) 0x30 7530 — 0x30 753F(B)	MBUFF3[1] TouCAN_A Message Buffer 3. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7140 — 0x30 714F(A) 0x30 7540 — 0x30 754F(B)	MBUFF4[1] TouCAN_A Message Buffer 4. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7150 — 0x30 715F(A) 0x30 7550 — 0x30 755F(B)	MBUFF5[1] TouCAN_A Message Buffer 5. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7160 — 0x30 716F(A) 0x30 7560 — 0x30 756F(B)	MBUFF6[1] TouCAN_A Message Buffer 6. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x307170 — 0x30717F(A) 0x30 7570 — 0x30 757F(B)	MBUFF7[1] TouCAN_A Message Buffer 7. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7180 — 0x30 718F(A) 0x30 7580 — 0x30 758F(B)	MBUFF8[1] TouCAN_A Message Buffer 8. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7190 — 0x30 719F(A) 0x30 7590 — 0x30 759F(B)	MBUFF9[1] TouCAN_A Message Buffer 9. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71A0 — 0x30 71AF(A) 0x30 75A0 — 0x30 75AF(B)	MBUFF10[1] TouCAN_A Message Buffer 10. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71B0 — 0x30 71BF(A) 0x30 75B0 — 0x30 75BF(B)	MBUFF11[1] TouCAN_A Message Buffer 11. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71C0 — 0x30 71CF(A) 0x30 75C0 — 0x30 75CF(B)	MBUFF12[1] TouCAN_A Message Buffer 12. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71D0 — 0x30 71DF(A) 0x30 75D0 — 0x30 75DF(B)	MBUFF13[1] TouCAN_A Message Buffer 13. See **Table 16-3** and **Table 16-4** for message buffer definitions.	

Table 16-10 TouCAN Register Map (Continued)

Access	Offset	MSB 0	LSB 15
S/U	0x30 71E0 — 0x30 71EF(A) 0x30 75E0 — 0x30 75EF(B)	MBUFF14[1] TouCAN_A Message Buffer 14. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71F0 — 0x30 71FF(A) 0x30 75F0 — 0x30 75FF(B)	MBUFF15[1] TouCAN_A Message Buffer 15. See **Table 16-3** and **Table 16-4** for message buffer definitions.	

NOTES:
1. The last word of each of the the MBUFF arrays (address 0x....E) is reserved and may cause a RCPU exception if read.

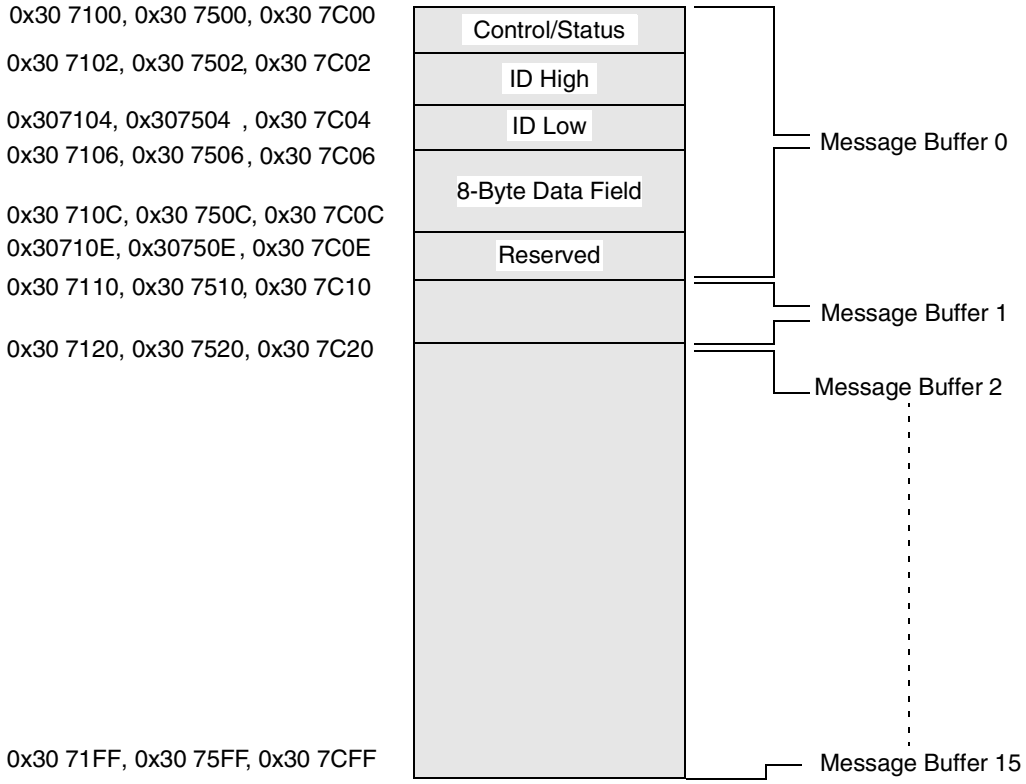


Figure 16-6 TouCAN Message Buffer Memory Map

16.7.1 TouCAN Module Configuration Register

TCNMCR — TouCAN Module Configuration Register **0x30 7080**
0x30 7480

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	NOT USED	HALT	NOT RDY	WAKE MSK	SOFT RST	FRZ ACK	SUPV	SELF WAKE	APS	STOP ACK	RESERVED			
RESET:															
0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Table 16-11 TCNMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the TouCAN, if the SELFWAKE bit is set. 0 = Enable TouCAN clocks 1 = Disable TouCAN clocks
1	FRZ	FREEZE assertion response. When FRZ = 1, the TouCAN can enter debug mode when the IMB3 FREEZE line is asserted or the HALT bit is set. Clearing this bit field causes the TouCAN to exit debug mode. Refer to **16.5.1 Debug Mode** for more information. 0 = TouCAN ignores the IMB3 FREEZE signal and the HALT bit in the module configuration register. 1 = TouCAN module enabled to enter debug mode.
2	—	Reserved
3	HALT	Halt TouCAN S-Clock. Setting the HALT bit has the same effect as assertion of the IMB3 FREEZE signal on the TouCAN without requiring that FREEZE be asserted. This bit is set to one after reset. It should be cleared after initializing the message buffers and control registers. TouCAN message buffer receive and transmit functions are inactive until this bit is cleared. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 = The TouCAN operates normally 1 = TouCAN enters debug mode if FRZ = 1
4	NOTRDY	TouCAN not ready. This bit indicates that the TouCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the TouCAN enters low-power stop mode or debug mode. It is cleared once the TouCAN exits either mode, either by synchronization to the CAN bus or by the self wake mechanism. 0 = TouCAN has exited low-power stop mode or debug mode. 1 = TouCAN is in low-power stop mode or debug mode.
5	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 = Wake up interrupt is disabled 1 = Wake up interrupt is enabled
6	SOFTRST	Soft reset. When this bit is asserted, the TouCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal TouCAN circuitry to completely reset before executing another access to CANMCR. The TouCAN clears this bit once the internal reset cycle is completed. 0 = Soft reset cycle completed 1 = Soft reset cycle initiated
7	FRZACK	TouCAN disable. When the TouCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the TouCAN has entered debug mode. When debug mode is exited, this bit is negated once the TouCAN prescaler is enabled. This is a read-only bit. 0 = The TouCAN has exited debug mode and the prescaler is enabled 1 = The TouCAN has entered debug mode, and the prescaler is disabled
8	SUPV	Supervisor/user data space. The SUPV bit places the TouCAN registers in either supervisor or user data space. 0 = Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode 1 = Registers with access controlled by the SUPV bit are restricted to supervisor mode

Table 16-11 TCNMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
9	SELFWAKE	Self wake enable. This bit allows the TouCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the TouCAN enters low-power stop mode, the TouCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the TouCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to **16.5.2 Low-Power Stop Mode** for more information on entry into and exit from low-power stop mode. 0 = Self wake disabled 1 = Self wake enabled
10	APS	Auto power save. The APS bit allows the TouCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 = Auto power save mode disabled; clocks run normally 1 = Auto power save mode enabled; clocks stop and restart as needed
11	STOPACK	Stop acknowledge. When the TouCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the TouCAN has entered low-power stop mode. When the TouCAN exits low-power stop mode, the STOPACK bit is cleared once the TouCAN's clocks are running. 0 = The TouCAN is not in low-power stop mode and its clocks are running 1 = The TouCAN has entered low-power stop mode and its clocks are stopped
12:15	—	Reserved

16.7.2 TouCAN Test Configuration Register

CANTCR — TouCAN Test Configuration Register **0x30 7082, 0x30 7482**

This register is used for factory test only.

16.7.3 TouCAN Interrupt Configuration Register

CANICR — TouCAN Interrupt Configuration Register **0x30 7084**
0x30 7484

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED					IRL			ILBS		RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 16-12 CANICR Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	IRL	Interrupt request level. When the TouCAN generates an interrupt request, this field determines which of the interrupt request signals is asserted.
8:9	ILBS	Interrupt level byte select. This field selects one of four time-multiplexed slots during which the interrupt request is asserted. The ILBS and IRL fields together select one of 32 effective interrupt levels. 00 = Levels 0 to7 01 = Levels 8 to 15 10 = Levels 16 to 23 11 = Levels 24 to 31
10:15	—	Reserved

16.7.4 Control Register 0

CANCTRL0 — Control Register 0 **0x30 7086**
0x30 7486

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BOFF MSK	ERR MSK	RESERVED		RXMOD		TXMODE		CANCTRL1							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-13 CANCTRL0 Bit Descriptions

Bit(s)	Name	Description
0	BOFFMSK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 = Bus off interrupt disabled 1 = Bus off interrupt enabled
1	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 = Error interrupt disabled 1 = Error interrupt enabled
2:3	—	
4:5	RXMODE	Receive pin configuration control. These bits control the configuration of the CANRX0 and CANRX1 pins. Refer to the **Table 16-14**.
6:7	TXMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTX0 and CANTX1 pins. Refer to **Table 16-15**.
8:15	CANCTRL1	See **Table 16-16**.

Table 16-14 RX MODE[1:0] Configuration

Pin	RX1	RX0	Receive Pin Configuration
CNRX1[1]	0	X	A logic zero on the CNRX1 pin is interpreted as a dominant bit; a logic one on the CNRX1 pin is interpreted as a recessive bit
	1	X	A logic one on the CNRX1 pin is interpreted as a dominant bit; a logic zero on the CNRX1 pin is interpreted as a recessive bit
CNRX0	X	0	A logic zero on the CNRX0 pin is interpreted as a dominant bit; a logic one on the CNRX0 pin is interpreted as a recessive bit
	X	1	A logic one on the CNRX0 pin is interpreted as a dominant bit; a logic zeroon the CNRX0 pin is interpreted as a recessive bit

NOTES:
1. The CNRX1 signal is not available on the MPC555 / MPC556.

Table 16-15 Transmit Pin Configuration

TXMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CNTX0 = 0, CNTX1[2] = 1 is a dominant level)[2]
01	Full CMOS; negative polarity (CNTX0 = 1, CNTX1[2] = 0 is a dominant level)
1X	Open drain[3]; positive polarity

NOTES:
1. Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.
2. The CNTX1 signal is not available on the MPC555 / MPC556.
3. Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CNTX0 and CNTX1 pins are disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both Tx mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

16.7.5 Control Register 1

CANCTRL1 — Control Register 1 **0x30 7086**
0x30 7486

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CANCTRL0								SAMP	Re-served	TSYNC	LBUF	0D	PROPSE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16 CANCTRL1 Bit Descriptions

Bit(s)	Name	Description
0:7	CANCTRL0	See **Table 16-13**
8	SAMP	Sampling mode. The SAMP bit determines whether the TouCAN module will sample each received bit one time or three times to determine its value. 0 = One sample, taken at the end of phase buffer segment one, is used to determine the value of the received bit. 1 = Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock.
9	—	Reserved
10	TSYNC	Timer synchronize mode. The TSYNC bit enables the mechanism that resets the free-running timer each time a message is received in message buffer zero. This feature provides the means to synchronize multiple TouCAN stations with a special “SYNC” message (global network time). 0 = Timer synchronization disabled. 1 = Timer synchronization enabled. Note: there can be a bit clock skew of four to five counts between different TouCAN modules that are using this feature on the same network.
11	LBUF	Lowest buffer transmitted first. The LBUF bit defines the transmit-first scheme. 0 = Message buffer with lowest ID is transmitted first. 1 = Lowest numbered buffer is transmitted first.
12	—	Reserved
13:15	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are zero to seven. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

16.7.6 Prescaler Divide Register

PRESDIV — Prescaler Divide Register **0x30 7088**
0x30 7488

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								CANCTRL2							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-17 PRESDIV Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the IMB clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $$\text{S-clock} = \frac{f_{SYS}}{\text{PRESDIV} + 1}$$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the IMB clock. The valid programmed values are 0 through 255.
8:15	CANCTRL2	See **Table 16-18**.

16.7.7 Control Register 2

CANCTRL2 — Control Register 2 **0x30 7088**
0x30 7488

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-18 CANCTRL2 Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	See **Table 16-17**.
8:9	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are zero through three. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
10:12	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment one in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
13:15	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment two in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment two is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

16.7.8 Free Running Timer

TIMER — Free Running Timer Register **0x30 708A**
0x30 748A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TIMER															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-19 TIMER Bit Descriptions

Bit(s)	Name	Description
0:15	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the TouCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

16.7.9 Receive Global Mask Registers

RXGMSKHI — Receive Global Mask Register High **0x30 7090, 0x30 7490**
RXGMSKLO — Receive Global Mask Register Low **0x30 7092, 0x30 7492**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 16-20 RXGMSKHI, RXGMSKLO Bit Descriptions

Bit(s)	Name	Description
0:31	MIDx	The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14 and 15, which have their own specific mask registers. Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and zero) are always zero, regardless of any write to these bits. The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

16.7.10 Receive Buffer 14 Mask Registers

RX14MSKHI — Receive Buffer 14 Mask Register High **0x30 7094, 0x30 7494**
RX14MSKLO — Receive Buffer 14 Mask Register Low **0x30 7096, 0x30 7496**

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

16.7.11 Receive Buffer 15 Mask Registers

RX15MSKHI — Receive Buffer 15 Mask Register High **0x30 7098, 0x30 7498**
RX15MSKLO — Receive Buffer 15 Mask Register Low **0x30 709A, 0x30 749A**

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.

16.7.12 Error and Status Register

ESTAT — Error and Status Register **0x30 70A0**
0x30 74A0

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BITERR		ACK ERR	CRC ERR	FORM ERR	STUFF ERR	TX WARN	RX WARN	IDLE	TX/RX	FCS		0	BOFF INT	ERR INT	WAKE INT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register reflects various error conditions, general status, and has the enable bits for three of the TouCAN interrupt sources. The reported error conditions are those which have occurred since the last time the register was read. A read clears these bits to zero.

Table 16-21 ESTAT Bit Descriptions

Bit(s)	Name	Description
0:1	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. Refer to **Table 16-22**. **NOTE**: The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
2	ACKERR	Acknowledge error. The ACKERR bit indicates whether an acknowledgment has been correctly received for a transmitted message. 0 = No ACK error was detected since the last read of this register 1 = An ACK error was detected since the last read of this register
3	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 0 = No CRC error was detected since the last read of this register 1 = A CRC error was detected since the last read of this register
4	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 = No format error was detected since the last read of this register 1 = A format error was detected since the last read of this register
5	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 0 = No bit stuffing error was detected since the last read of this register 1 = A bit stuffing error was detected since the last read of this register
6	TXWARN	Transmit error status flag. The TXWARN status flag reflects the status of the TouCAN transmit error counter. 0 = Transmit error counter $<$ 96 1 = Transmit error counter $\geq$ 96
7	RXWARN	Receiver error status flag. The RXWARN status flag reflects the status of the TouCAN receive error counter. 0 = Receive error counter $<$ 96 1 = Receive error counter $\geq$ 96
8	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 = The CAN bus is not idle 1 = The CAN bus is idle
9	TX/$\overline{\text{RX}}$	Transmit/receive status. The TX/RX bit indicates when the TouCAN module is transmitting or receiving a message. TX/RX has no meaning when IDLE = 1. 0 = The TouCAN is receiving a message if IDLE = 0 1 = The TouCAN is transmitting a message if IDLE = 0
10:11	FCS	Fault confinement state. The FCS[1:0] field describes the state of the TouCAN. Refer to **Table 16-23**. If the SOFTRST bit in CANMCR is asserted while the TouCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the TouCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to**16.3.4 Error Counters** for more information on entry into and exit from the various fault confinement states.
12	—	Reserved
13	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the TouCAN enters the bus off state. 0 = No bus off interrupt requested 1 = When the TouCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.

Table 16-21 ESTAT Bit Descriptions (Continued)

Bit(s)	Name	Description
14	ERRINT	Error Interrupt. The ERRINT bit is used to request an interrupt when the TouCAN detects a transmit or receive error. 0 = No error interrupt request 1 = If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated. To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.
15	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the TouCAN module is in low-power stop mode. 0 = No wake interrupt requested 1 = When the TouCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

Table 16-22 Transmit Bit Error Status

BITERR[1:0]	Bit Error Status
00	No transmit bit error
01	At least one bit sent as dominant was received as recessive
10	At least one bit sent as recessive was received as dominant
11	Not used

Table 16-23 Fault Confinement State Encoding

FCS[1:0]	Bus State
00	Error active
01	Error passive
1X	Bus off

16.7.13 Interrupt Mask Register

IMASK — Interrupt Mask Register **0x30 70A2, 0x30 74A2**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IMASKH								IMASKL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-24 IMASK Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IMASKH, IMASKL	IMASK contains two 8-bit fields, IMASKH and IMASKL. IMASK can be accessed with a 16-bit read or write, and IMASKH and IMASKL can be accessed with byte reads or writes. IMASK contains one interrupt mask bit per buffer. It allows the CPU to designate which buffers will generate interrupts after successful transmission/reception. Setting a bit in IMASK enables interrupt requests for the corresponding message buffer. **NOTE**: Bit 15 (LSB) corresponds to message buffer 0. Bit 0 (MSB) corresponds to mesage buffer 15.

16.7.14 Interrupt Flag Register

IFLAG — Interrupt Flag Register **0x30 70A4**
0x30 74A4

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IFLAGH								IFLAGL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-25 IFLAG Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IFLAGH, IFLAGL	IFLAG contains two 8-bit fields, IFLAGH and IFLAGL. IFLAG can be accessed with a 16-bit read or write, and IFLAGH and IFLAGL can be accessed with byte reads or writes. IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a zero. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only. **NOTE**: Bit 15 (LSB) corresponds to message buffer 0. Bit 0 (MSB) corresponds to mesage buffer 15.

16.7.15 Error Counters

RXECTR — Receive Error Counter **0x30 70A6, 0x30 74A6**
TXECTR — Transmit Error Counter

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RXECTR								TXECTR							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-26 RXECTR, TXECTR Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	RXECTR, TXECTR	Both counters are read only, except when the TouCAN is in test or debug mode.

SECTION 17
TIME PROCESSOR UNIT 3

The time processor unit 3 (TPU3), an enhanced version of the original TPU, is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU3 is fully compatible to the TPU2. Operating simultaneously with the CPU, the two TPU3 modules process micro-instructions, schedules and processes real-time hardware events, performs input and output, and accesses shared data without CPU intervention. Consequently, for each timer event, the CPU setup and service times are minimized or eliminated.

The MPC555 / MPC556 contains two independent TPU3s. **Figure 17-1** is a simplified block diagram of a single TPU3.

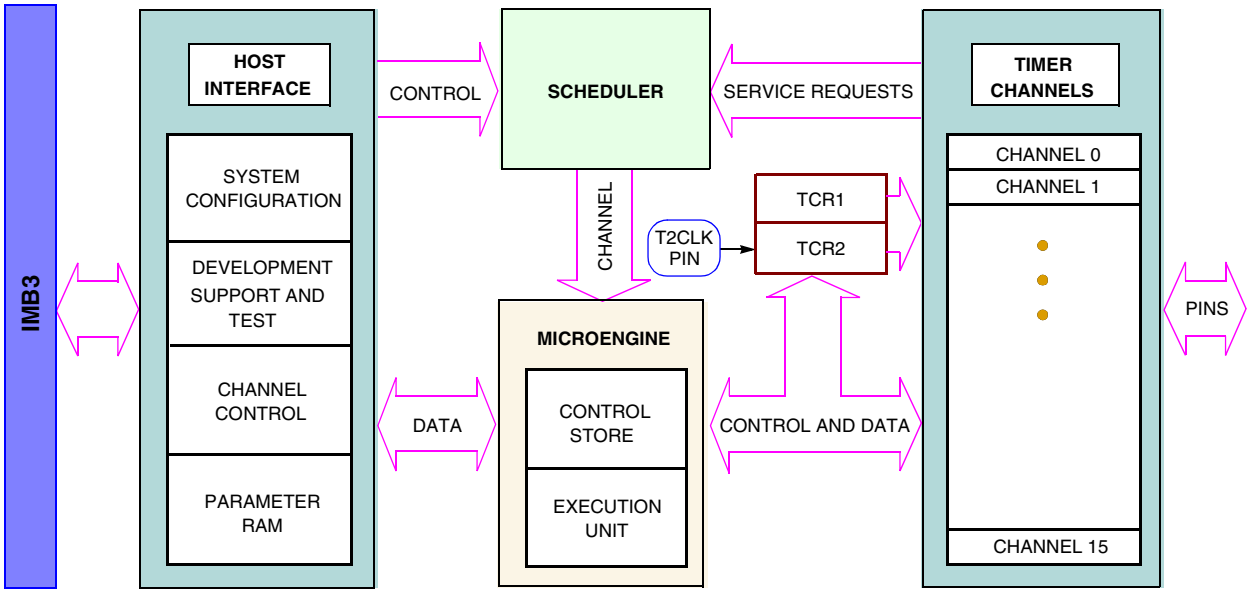


Figure 17-1 TPU3 Block Diagram

17.1 Overview

The TPU3 can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require CPU interrupt service.

The microcode ROM TPU3 functions that are available in the MPC555 / MPC556 are described in **APPENDIX D TPU ROM FUNCTIONS**.

17.2 TPU3 Components

The TPU3 consists of two 16-bit time bases, 16 independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-ported parameter RAM is used to pass parameters between the module and the CPU.

17.2.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the CPU via bit fields in the TPU3 module configuration register (TPUMCR) and TPU module configuration register two (TPUMCR2). Timer count registers TCR1 and TCR2 provide access to the current counter values. TCR1 and TCR2 can be read by TPU microcode but are not directly available to the CPU. The TCR1 clock is always derived from the IMB clock. The TCR2 clock can be derived from the IMB clock or from an external input via theT2CLK clock pin. The duration between active edges on the T2CLK clock pin must be at least nine IMB clocks.

17.2.2 Timer Channels

The TPU3 has 16 independent channels, each connected to an MCU pin. The channels have identical hardware and are functionally equivalent in operation. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

17.2.3 Scheduler

When a service request is received, the scheduler determines which TPU3 channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

17.2.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the CPU. Microcode can also be executed from the dual-port RAM (DPTRAM) module instead of the control store. The DPTRAM allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **17.3.6 Emulation Support** for more information.

17.2.5 Host Interface

The host interface registers allow communication between the CPU and the TPU3, both before and during execution of a time function. The registers are accessible from the IMB through the TPU3 bus interface unit. Refer to **17.4 Programming Model** for register bit/field definitions and address mapping.

17.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Channels zero through 15 each have eight parameters. The parameter RAM address map in **17.4.18 TPU3 Parameter RAM** shows how parameter words are organized in memory.

The CPU specifies function parameters by writing to the appropriate RAM address. The TPU3 reads the RAM to determine channel operation. The TPU3 can also store information to be read by the CPU in the parameter RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this manual. Refer to the ***TPU Reference Manual*** **(TPURM/AD)** and the Motorola **TPU Literature Package (TPULITPAK/D)** for more information.

17.3 TPU Operation

All TPU3 functions are related to one of the two 16-bit time bases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU3 can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneous match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

17.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. The time needed to respond to and service an event is determined by which channels and the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU3 performance in a given application. Latency can be closely estimated. For more information, refer to the ***TPU Reference Manual*** **(TPURM/AD)**.

17.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU3 channels contain identical hardware and are functionally equivalent in oper-

ation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

17.3.3 Interchannel Communication

The autonomy of the TPU3 is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

17.3.4 Programmable Channel Service Priority

The TPU3 provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

17.3.5 Coherency

For data to be coherent, all available portions of the data must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

17.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU3 provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in TPUMCR. In emulation mode, an auxiliary bus connection is made between the DPTRAM and the TPU3, and access to DPTRAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, DPTFLASH module access timing remains consistent with access timing of the TPU microcode ROM control store.

To support changing TPU application requirements, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Motorola Programming Note ***Using the TPU Function Library and TPU Emulation Mode, (*****TPUPN00/D),** for information about developing custom functions and accessing the

TPU function library. Refer to the Motorola **TPU Literature Package (TPULITPAK/D)** for more information about specific functions.

17.3.7 TPU3 Interrupts

Each of the TPU3 channels can generate an interrupt service request. Interrupts for each channel must be enabled by writing to the appropriate control bit in the channel interrupt enable register (CIER). The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions set the flags. Setting a flag bit causes the TPU3 to make an interrupt service request if the corresponding channel interrupt enable bit is set.

The TPU3 can generate one of 32 possible interrupt request levels on the IMB3. The value driven onto $\overline{\text{IRQ}}$[7:0] represents the interrupt level programmed in the IRL field of the TPU interrupt configuration register (TICR). Under the control of the ILBS bits in the ICR, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 17-2** displays the interrupt level scheme.

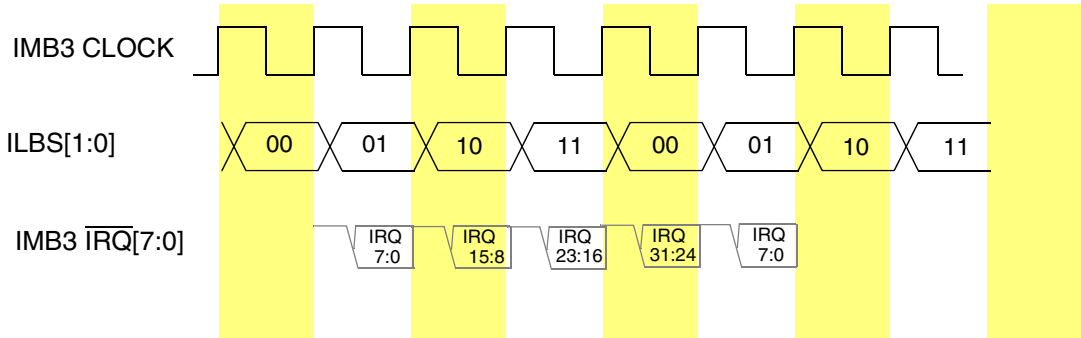


Figure 17-2 TPU3 Interrupt Levels

17.3.8 Prescaler Control for TCR1

Timer count register 1 (TCR1) is clocked from the output of a prescaler. The following fields control TCR1:

- The PSCK and TCR1P fields in TPUMCR
- The DIV2 field in TPUMCR2
- The EPSCKE and EPSCK fields in TPUMCR3.

The rate at which TCR1 is incremented is determined as follows:

- The user selects either the standard prescaler (by clearing the enhanced prescaler enable bit, EPSCKE, in TPUMCR3) or the enhanced prescaler (by setting EPSCKE).

— If the standard prescaler is selected (EPSCKE = 0), the the PSCK bit determines whether the standard prescaler divides the IMB clock input by 32 (PSCK = 0) or four (PSCK = 1)

— If the enhanced prescaler is selected (EPSCKE = 1), the EPSCK bits select a value by which the IMB clock is divided. The lowest frequency for TCR1 clock is IMB clock divided by 64x8. The highest frequency for TCR1 clock is IMB clock divided by two (2x1). See **Table 17-1**.

Table 17-1 Enhanced TCR1 Prescaler Divide Values

EPSCK Value	Divide IMB Clock By
0x00	2
0x01	4
0x02	6
0x03	8
0x04, 0x05,...0x1d	10,12,...60
0x1e	62
0x1f	64

— The output of either the standard prescaler or the enhanced prescaler is then divided by 1, 2, 4, or 8, depending on the value of the TCR1P field in the TPUMCR.

Table 17-2 TCR1 Prescaler Values

TCR1P Value	Divide by
0b00	1
0b01	2
0b10	4
0b11	8

— If the DIV2 bit is one, the TCR1 counter increments at a rate of the internal clock divided by two. If DIV2 is zero, the TCR1 increment rate is defined by the output of the TCR1 prescaler (which, in turn, takes as input the output of either the standard or enhanced prescaler).

Figure 17-3 shows a diagram of the TCR1 prescaler control block.

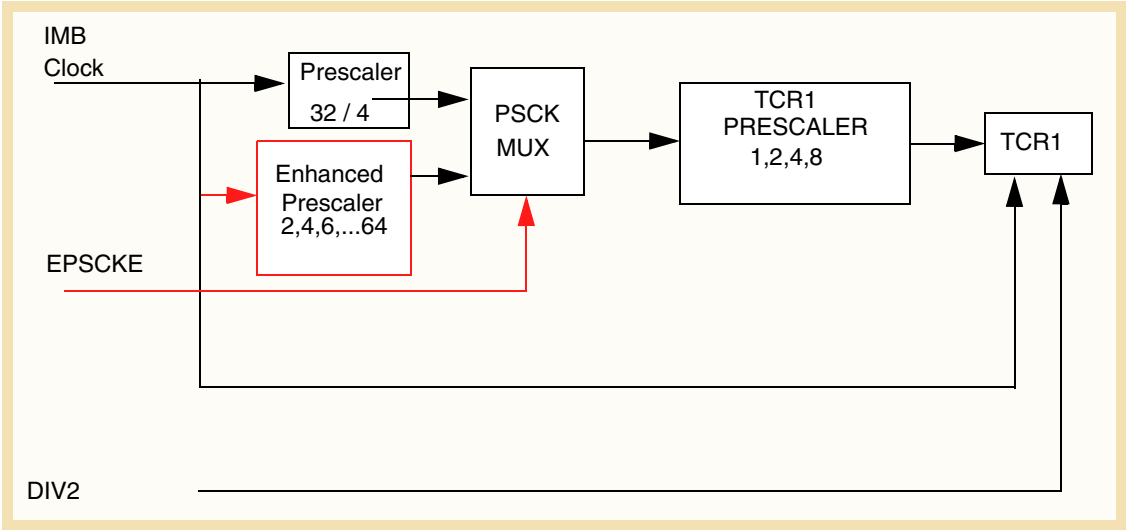


Figure 17-3 TCR1 Prescaler Control

17.3.9 Prescaler Control for TCR2

Timer count register 2 (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit and the T2CSL (TCR2 counter clock edge) bit in TPUMCR determine T2CR2 pin functions. Refer to **Table 17-3**.

Table 17-3 TCR2 Counter Clock Source

T2CSL	T2CG	TCR2 Clock
0	0	Rise transition T2CLK
0	1	Gated IMB clock
1	0	Fall transition T2CLK
1	1	Rise & fall transition T2CLK

The function of the T2CG bit is shown in **Figure 17-4**.

When T2CG is set, the external T2CLK pin functions as a gate of the DIV8 clock (the TPU3 IMB clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2. The duration between active edges on the T2CLK clock pin must be at least nine IMB clocks.

The TCR2PSCK2 bit in TPUMCR3 determines whether the clock source is divided by two before it is fed into the TCR2 prescaler. The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to re-

solve down to the TPU3 IMB clock divided by eight. **Figure 17-4** illustrates the TCR2 pre-divider and pre-scaler control.

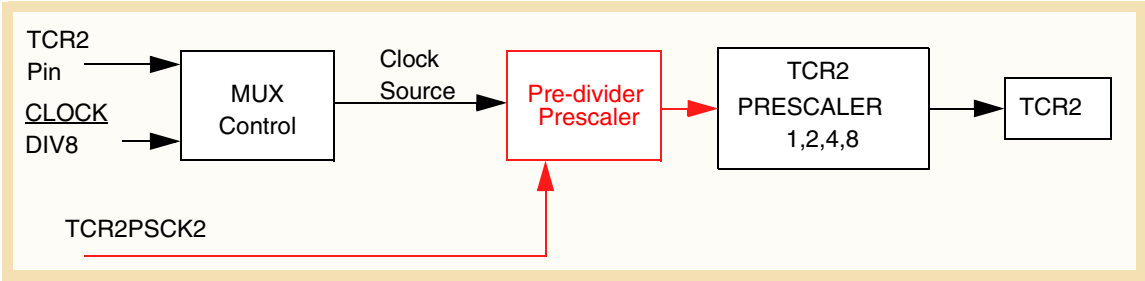


Figure 17-4 TCR2 Prescaler Control

Table 17-4 is a summary of prescaler output (assuming a divide-by-one value for the pre-divider prescaler.

Table 17-4 TCR2 Prescaler Control

TCR2 Value	Internal Clock Divide Ratio		External Clock Divide Ratio	
	TCR2PSCK2 = 0	TCR2PSCK2 = 1	TCR2PSCK2 = 0	TCR2PSCK2 = 1
0b00	8	8	1	1
0b01	16	24	2	3
0b10	32	56	4	7
0b11	64	120	8	15

17.4 Programming Model

The TPU3 memory map contains three groups of registers:

- System configuration registers
- Channel control and status registers
- Development support and test verification registers

All registers except the channel interrupt status register (CISR) must be read or written by means of half-word (16-bit) or word (32-bit) accesses. The address space of the TPU3 memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

Table 17-5 shows the TPU3 address map.

Table 17-5 TPU3 Register Map

Address	MSB 0 Register LSB 15
0x30 4000 0x30 4400	TPU3 Module Configuration Register (TPUMCR) See **Table 17-6** for bit descriptions.
0x30 4002 0x30 4402	TPU3 Test Configuration Register (TCR)
0x30 4004 0x30 4404	Development Support Control Register (DSCR) See **Table 17-7** for bit descriptions.
0x30 4006 0x30 4406	Development Support Status Register (DSSR) See **Table 17-8** for bit descriptions.
0x30 4008 0x30 4408	TPU3 Interrupt Configuration Register (TICR) See **Table 17-9** for bit descriptions.
0x30 400A 0x30 440A	Channel Interrupt Enable Register (CIER) See **Table 17-10** for bit descriptions.
0x30 400C 0x30 440C	Channel Function Selection Register 0 (CFSR0) See **Table 17-11** for bit descriptions.
0x30 400E 0x30 440E	Channel Function Selection Register 1 (CFSR1) See **Table 17-11** for bit descriptions.
0x30 4010 0x30 4410	Channel Function Selection Register 2 (CFSR2) See **Table 17-11** for bit descriptions.
0x30 4012 0x30 4412	Channel Function Selection Register 3 (CFSR3) See **Table 17-11** for bit descriptions.
0x30 4014 0x30 4414	Host Sequence Register 0 (HSQR0) See **Table 17-12** for bit descriptions.
0x30 4016 0x30 4416	Host Sequence Register 1 (HSQR1) See **Table 17-12** for bit descriptions.
0x30 4018 0x30 4418	Host Service Request Register 0 (HSRR0) See **Table 17-13** for bit descriptions.
0x30 401A 0x30 441A	Host Service Request Register 1 (HSRR1) See **Table 17-13** for bit descriptions.
0x30 401C 0x30 441C	Channel Priority Register 0 (CPR0) See **Table 17-14** for bit descriptions.
0x30 401E 0x30 441E	Channel Priority Register 1 (CPR1) See **Table 17-14** for bit descriptions.
0x30 4020 0x30 4420	Channel Interrupt Status Register (CISR) See **Table 17-16** for bit descriptions.
0x30 4022 0x30 4422	Link Register (LR)
0x30 4024 0x30 4424	Service Grant Latch Register (SGLR)
0x30 4026 0x30 4426	Decoded Channel Number Register (DCNR)
0x30 4028 0x30 4428	TPU Module Configuration Register 2 (TPUMCR2) See **Table 17-17** for bit descriptions.
0x30 402A 0x30 442A	TPU Module Configuration 3 (TPUMCR3) See **Table 17-20** for bit descriptions.
0x30 402C 0x30 442C	Internal Scan Data Register (ISDR)

Table 17-5 TPU3 Register Map (Continued)

Address	MSB 0 Register
0x30 402E 0x30 442E	Internal Scan Control Register (ISCR)
0x30 4100 – 0x30 410F 0x30 4500 – 0x30 450F	Channel 0 Parameter Registers
0x30 4110 – 0x30 411F 0x30 4510 – 0x30 451F	Channel 1 Parameter Registers
0x30 4120 – 0x30 412F 0x30 4520 – 0x30 452F	Channel 2 Parameter Registers
0x30 4130 – 0x30 413F 0x30 4530 – 0x30 453F	Channel 3 Parameter Registers
0x30 4140 – 0x30 414F 0x30 4540 – 0x30 454F	Channel 4 Parameter Registers
0x30 4150 – 0x30 415F 0x30 4550 – 0x30 455F	Channel 5 Parameter Registers
0x30 4160 – 0x30 416F 0x30 4560 – 0x30 456F	Channel 6 Parameter Registers
0x30 4170 – 0x30 417F 0x30 4570 – 0x30 457F	Channel 7 Parameter Registers
0x30 4180 – 0x30 418F 0x30 4580 – 0x30 458F	Channel 8 Parameter Registers
0x30 4190 – 0x30 419F 0x30 4590 – 0x30 459F	Channel 9 Parameter Registers
0x30 41A0 – 0x30 41AF 0x30 45A0 – 0x30 45AF	Channel 10 Parameter Registers
0x30 41B0 – 0x30 41BF 0x30 45B0 – 0x30 45BF	Channel 11 Parameter Registers
0x30 41C0 – 0x30 41CF 0x30 45C0 – 0x30 45CF	Channel 12 Parameter Registers
0x30 41D0 – 0x30 41DF 0x30 45D0 – 0x30 45DF	Channel 13 Parameter Registers
0x30 41E0 – 0x30 41EF 0x30 45E0 – 0x30 45EF	Channel 14 Parameter Registers
0x30 41F0 – 0x30 41FF 0x30 45F0 – 0x30 45FF	Channel 15 Parameter Registers

17.4.1 TPU Module Configuration Register

TPUMCR — TPU Module Configuration Register **0x30 4000**
0x30 4400

MSB LSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	TPU3	T2CSL	RESERVED			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 17-6 TPUMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. If the STOP bit in TPUMCR is set, the TPU3 shuts down its internal clocks, shutting down the internal microengine. TCR1 and TCR2 cease to increment and retain the last value before the stop condition was entered. The TPU3 asserts the stop flag (STF) in TPUMCR to indicate that it has stopped. 0 = Enable TPU3 clocks 1 = Disable TPU3 clocks
1:2	TCR1P	Timer count register 1 prescaler control. TCR1 is clocked from the output of a prescaler. The prescaler divides its input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **17.3.8 Prescaler Control for TCR1** for more information.
3:4	TCR2P	Timer count register 2 prescaler control. TCR2 is clocked from the output of a prescaler. The prescaler divides this input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **17.3.9 Prescaler Control for TCR2** for more information.
5	EMU	Emulation control. In emulation mode, the TPU3 executes microinstructions from DPTRAM exclusively. Access to the DPTRAM via the IMB3 is blocked, and the DPTRAM is dedicated for use by the TPU3. After reset, this bit can be written only once. 0 = TPU3 and DPTRAM operate normally 1 = TPU3 and DPTRAM operate in emulation mode
6	T2CG	TCR2 clock/gate control 0 = TCR2 pin used as clock source for TCR2 1 = TCR2 pin used as gate of DIV8 clock for TCR2 Refer to **17.3.9 Prescaler Control for TCR2** for more information.
7	STF	Stop flag. 0 = TPU3 is operating normally 1 = TPU3 is stopped (STOP bit has been set)
8	SUPV	Supervisor data space 0 = Assignable registers are accessible from user or supervisor privilege level 1 = Assignable registers are accessible from supervisor privilege level only
9	PSCK	Standard prescaler clock. Note that this bit has no effect if the extended prescaler is selected (EPSCKE = 1). 0 = $f_{SYS} \div 32$ is input to TCR1 prescaler, if standard prescaler is selected 1 = $f_{SYS} \div 4$ is input to TCR1 prescaler, if standard prescaler is selected
10	TPU3	TPU3 enable. The TPU3 enable bit provides compatibility with the TPU. If running TPU code on the TPU3, the microcode size should not be greater than two Kbytes and the TPU3 enable bit should be cleared to zero. The TPU3 enable bit is write-once after reset. The reset value is one, meaning that the TPU3 will operate in TPU3 mode. 0 = TPU mode; zero is the TPU reset value 1 = TPU3 mode; one is the TPU3 reset value **NOTE**: The programmer should not change this value unless necessary when developing custom TPU microcode.
11	T2CSL	TCR2 counter clock edge. This bit and the T2CG control bit determine the clock source for TCR2. Refer to **17.3.9 Prescaler Control for TCR2** for details.
12:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TPU3 implementations that use hardware interrupt arbitration.

17.4.2 TPU3 Test Configuration Register

TCR — TPU3 Test Configuration Register **0x30 4002, 0x30 4402**

Used for factory test only.

17.4.3 Development Support Control Register

DSCR — Development Support Control Register **0x30 4004**
0x30 4404

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HOT4	RESERVED				BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
RESET:															
0					0	0	0	0	0	0	0	0	0	0	0

Table 17-7 DSCR Bit Descriptions

Bit(s)	Name	Description
0	HOT4	Hang on T4 0 = Exit wait on T4 state caused by assertion of HOT4 1 = Enter wait on T4 state
1:4	—	Reserved
5	BLC	Branch latch control 0 = Latch conditions into branch condition register before exiting halted state 1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period
6	CLKS	Stop clocks (to TCRs) 0 = Do not stop TCRs 1 = Stop TCRs during the halted state
7:8	FRZ	FREEZE assertion response. The FRZ bits specify the TPU microengine response to the IMB3 FREEZE signal 00 = Ignore freeze 01 = Reserved 10 = Freeze at end of current microcycle 11 = Freeze at next time-slot boundary
9	CCL	Channel conditions latch. CCL controls the latching of channel conditions (MRL and TDL) when the CHAN register is written. 0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction 1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction
10	BP	µPC breakpoint enable 0 = Breakpoint not enabled 1 = Break if µPC equals µPC breakpoint register
11	BC	Channel breakpoint enable 0 = Breakpoint not enabled 1 = Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode
12	BH	Host service breakpoint enable 0 = Breakpoint not enabled 1 = Break if host service latch is asserted at beginning of state
13	BL	Link service breakpoint enable 0 = Breakpoint not enabled 1 = Break if link service latch is asserted at beginning of state
14	BM	MRL breakpoint enable 0 = Breakpoint not enabled 1 = Break if MRL is asserted at beginning of state
15	BT	TDL breakpoint enable 0 = Breakpoint not enabled 1 = Break if TDL is asserted at beginning of state

17.4.4 Development Support Status Register

DSSR — Development Support Status Register **0x30 4006**
0x30 4406

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED								BKPT	PCBK	CHBK	SRBK	TPUF	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-8 DSSR Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8	BKPT	Breakpoint asserted flag. If an internal breakpoint caused the TPU3 to enter the halted state, the TPU3 asserts the BKPT signal on the IMB and sets the BKPT flag. BKPT remains set until the TPU3 recognizes a breakpoint acknowledge cycle, or until the IMB FREEZE signal is asserted.
9	PCBK	µPC breakpoint flag. PCBK is asserted if a breakpoint occurs because of a µPC (microprogram counter) register match with the µPC breakpoint register. PCBK is negated when the BKPT flag is cleared.
10	CHBK	Channel register breakpoint flag. CHBK is asserted if a breakpoint occurs because of a CHAN register match with the CHAN register breakpoint register. CHBK is negated when the BKPT flag is cleared.
11	SRBK	Service request breakpoint flag. SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is cleared.
12	TPUF	TPU3 FREEZE flag. TPUF is set whenever the TPU3 is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU3 exits the halted state because of FREEZE being negated.
13:15	—	Reserved

17.4.5 TPU3 Interrupt Configuration Register

TICR — TPU3 Interrupt Configuration Register **0x30 4008**
0x30 4408

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED					CIRL			ILBS		RESERVED					
RESET:															
					0	0	0	0	0	0	0				

Table 17-9 TICR Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	CIRL	Channel interrupt request level. This three-bit field specifies the interrupt request level for all channels. This field is used in conjunction with the ILBS field to determine the request level of TPU3 interrupts.
8:9	ILBS	Interrupt level byte select. This field and the CIRL field determine the level of TPU3 interrupt requests. 00 = $\overline{IRQ}$[0:7] selected 01 = $\overline{IRQ}$[8:15] selected 10 = $\overline{IRQ}$[16:23] selected 11 = $\overline{IRQ}$[24:31] selected
10:15	—	Reserved. Note that bits 10:11 represent channel interrupt base vector (CIBV) bits in some TPU3 implementations.

17.4.6 Channel Interrupt Enable Register

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU3 channels to request interrupt service. Setting the appropriate bit in the register enables a channel to make an interrupt service request; clearing a bit disables the interrupt.

CIER — Channel Interrupt Enable Register **0x30 400A**
0x30 440A

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-10 CIER Bit Descriptions

Bit(s)	Name	Description
0:15	CH[15:0]	Channel interrupt enable/disable 0 = Channel interrupts disabled 1 = Channel interrupts enabled **Note:** The MSB (bit 0 in big-endian mode) represents CH15, and the LSB (bit 15 in big-endian mode) represents CH0.

17.4.7 Channel Function Select Registers

Encoded 4-bit fields within the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions will be provided in a subsequent draft of this document.

CFSR0 — Channel Function Select Register 0 **0x30 400C**
0x30 440C

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15				CH 14				CH 13				CH 12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 **0x30 400E**
0x30 440E

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 11				CH 10				CH 9				CH 8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 **0x30 4010**
0x30 4410

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7				CH 6				CH 5				CH 4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 **0x30 4012**
0x30 4412

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 3				CH 2				CH 1				CH 0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-11 CFSRx Bit Descriptions

Name	Description
CH[15:0]	Encoded time function for each channel. Encoded four-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

17.4.8 Host Sequence Registers

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Meanings of host sequence bits and host service request bits for predefined time functions will be provided in a subsequent draft of this document.

HSQR0 — Host Sequence Register 0 **0x30 4014**
0x30 4414

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSQR1 — Host Sequence Register 1 **0x30 4016**
0x30 4416

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-12 HSQRx Bit Descriptions

Name	Description
CH[15:0]	Encoded host sequence. The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

17.4.9 Host Service Request Registers

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits is determined by time function microcode. Refer to the ***TPU Reference Manual* (TPURM/AD)** and the Motorola **TPU Literature Package (TPULITPAK/D)** for more information.

HSRR0 — Host Service Request Register 0 **0x30 4018**
0x30 4418

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSRR1 — Host Service Request Register 1 **0x30 401A**
0x30 441A

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-13 HSSRx Bit Descriptions

Name	Description
CH[15:0]	Encoded type of host service. The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. A host service request field cleared to 0b00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three non-zero states. The CPU must monitor the host service request register until the TPU3 clears the service request to 0b00 before any parameters are changed or a new service request is issued to the channel.

17.4.10 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel.

CPR0 — Channel Priority Register 0 **0x30 401C**
0x30 441C

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1 **0x30 401E**
0x30 441E

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-14 CPRx Bit Descriptions

Name	Description
CH[15:0]	Encoded channel priority levels. **Table 17-15** indicates the number of time slots guaranteed for each channel priority encoding.

Table 17-15 Channel Priorities

CHx[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

17.4.11 Channel Interrupt Status Register

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU3 to make an interrupt service request if the corresponding CIER bit is set. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU3 register that can be accessed on a byte basis.

CISR — Channel Interrupt Status Register **0x30 4020**
0x30 4420

MSB LSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-16 CISR Bit Descriptions

Bit(s)	Name	Description
0:15	CH[15:0]	Channel interrupt status 0 = Channel interrupt not asserted 1 = Channel interrupt asserted

17.4.12 Link Register

LR — Link Register **0x30 4022, 0x30 4422**

Used for factory test only.

17.4.13 Service Grant Latch Register

SGLR — Service Grant Latch Register **0x30 4024, 0x30 4424**

Used for factory test only.

17.4.14 Decoded Channel Number Register

DCNR — Decoded Channel Number Register **0x30 4026, 0x30 4426**

Used for factory test only.

17.4.15 TPU3 Module Configuration Register 2

TPUMCR2 — TPU Module Configuration Register 2 **0x30 4028**
0x30 4428

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED							DIV2	SOFT RST	ETBANK		FPSCK			T2CF	DTPU
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-17 TPUMCR2 Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	DIV2	Divide by 2 control. When asserted, the DIV2 bit, along with the TCR1P bit and the PSCK bit in the TPUMCR, determines the rate of the TCR1 counter in the TPU3. If set, the TCR1 counter increments at a rate of two IMB clocks. If negated, TCR1 increments at the rate determined by control bits in the TCR1P and PSCK fields. 0 = TCR1 increments at rate determined by control bits in the TCR1P and PSCK fields of the TPUMCR register 1 = Causes TCR1 counter to increment at a rate of the IMB clock divided by two
8	SOFT RST	Soft reset. The TPU3 performs an internal reset when both the SOFT RST bit in the TPUMCR2 and the STOP bit in TPUMCR are set. The CPU must write zero to the SOFT RST bit to bring the TPU3 out of reset. The SOFT RST bit must be asserted for at least nine clocks. 0 = Normal operation 1 = Puts TPU3 in reset until bit is cleared **NOTE**: Do not attempt to access any other TPU3 registers when this bit is asserted. When this bit is asserted, it is the only accessible bit in the register.
9:10	ETBANK	Entry table bank select. This field determines the bank where the microcoded entry table is situated. After reset, this field is 0b00. This control bit field is write once after reset. ETBANK is used when the microcode contains entry tables not located in the default bank 0. To execute the ROM functions on this MCU, ETBANK[1:0] must be 0b0. Refer to **Table 17-18**. **NOTE**: This field should not be modified by the programmer unless necessary because of custom microcode.
11:13	FPSCK	Filter prescaler clock. The filter prescaler clock control bit field determines the ratio between IMB clock frequency and minimum detectable pulses. The reset value of these bits is zero, defining the filter clock as four IMB clocks. Refer to **Table 17-19**.
14	T2CF	T2CLK pin filter control. When asserted, the T2CLK input pin is filtered with the same filter clock that is supplied to the channels. This control bit is write once after reset. 0 = Uses fixed four-clock filter 1 = T2CLK input pin filtered with same filter clock that is supplied to the channels
15	DTPU	Disable TPU3 pins. When the disable TPU3 control pin is asserted, pin TP15 is configured as an input disable pin. When the TP15 pin value is zero, all TPU3 output pins are three-stated, regardless of the pins function. The input is not synchronized. This control bit is write once after reset. 0 = TP15 functions as normal TPU3 channel 1 = TP15 pin configured as output disable pin. When TP15 pin is low, all TPU3 output pins are in a high-impedance state, regardless of the pin function.

Table 17-18 Entry Table Bank Location

ETBANK	Bank
00	0
01	1
10	2
11	3

Table 17-19 IMB Clock Frequency/Minimum Guaranteed Detected Pulse

Filter Control FPSCK	Divide By	20 MHz	33 MHz	40 MHz
000	4	200 ns	121 ns	100 ns
001	8	400 ns	242 ns	200 ns
010	16	800 ns	485 ns	400 ns
011	32	1.6 µs	970 ns	800 ns
100	64	3.2 µs	1.94 µs	1.60 µs
101	128	6.4 µs	3.88 µs	3.20 µs
110	256	12.8 µs	7.76 µs	6.40 µs
111	512	25.6 µs	15.51 µs	12.80 µs

17.4.16 TPU Module Configuration Register 3

TPUMCR3 — TPU Module Configuration Register 3 **0x30 402A**
0x30 442A

MSB LSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED							PWOD	TCR2 PCK2	EP-SCKE	Re-served	EPSCK				
RESET:															
					0	0	0	0	0	0	0				

Table 17-20 TPUMCR3 Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	PWOD	Prescaler write-once disable bit. The PWOD bit does not lock the EPSCK field and the EPSCKE bit. 0 = Prescaler fields in MCR are write-once 1 = Prescaler fields in MCR can be written anytime
8	TCR2PSCK2	TCR2 prescaler 2 0 = Prescaler clock source is divided by one. 1 = Prescaler clock is divided. See divider definitions in **Table 17-4**.
9	EPSCKE	Enhanced pre-scaler enable 0 = Disable enhanced prescaler (use standard prescaler) 1 = Enable enhanced prescaler. IMB clock will be divided by the value in EPSCK field.

Table 17-20 TPUMCR3 Bit Descriptions (Continued)

Bit(s)	Name	Description
10	—	Reserved
11:15	EPSCK	Enhanced prescaler value that will be loaded into the enhanced prescaler counter. Prescaler value = (EPSCK + 1) x 2. Refer to **17.3.8 Prescaler Control for TCR1** for details.

17.4.17 TPU3 Test Registers

The following TPU3 registers are used for factory test only:

- Internal scan data register (ISDR, address offset 0x30 402C, 0x30 442C)
- Internal scan control register (ISCR, address offset 0x30 402E, 0x30 442E)

17.4.18 TPU3 Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 15 have eight parameters. The parameter registers constitute a shared work space for communication between the CPU and the TPU3. The TPU3 can only access data in the parameter RAM. Refer to **Table 17-21**.

Table 17-21 Parameter RAM Address Offset Map[1]

Channel Number	Parameter							
	0	1	2	3	4	5	6	7
0	100 500	102 502	104 504	106 506	108 508	10A 50A	10C 50C	10E 50E
1	110 510	112 512	114 514	116 516	118 518	11A 51A	11C 51C	11E 51E
2	120 520	122 522	124 524	126 526	128 528	12A 52A	12C 52C	12E 52E
3	130 530	132 532	134 534	136 536	138 538	13A 53A	13C 53C	13E 53E
4	140 540	142 542	144 544	146 546	148 548	14A 54A	14C 54C	14E 54C
5	150 550	152 552	154 554	156 556	158 558	15A 55A	15C 55C	15E 55E
6	160 560	162 562	164 564	166 566	168 568	16A 56A	16C 56C	16E 56E
7	170 570	172 572	174 574	176 576	178 578	17A 57A	17C 57C	17E 57E
8	180 580	182 582	184 585	186 586	188 588	18A 58A	18C 58C	18E 58E
9	190 590	192 592	194 594	196 596	198 598	19A 59A	19C 59C	19E 59E
10	1A0 5A0	1A2 5A2	1A4 5A4	1A6 5A6	1A8 5A8	1AA 5AA	1AC 5AC	1AE 5AE
11	1B0 5B0	1B2 5B2	1B4 5B4	1B6 5B6	1B8 5B8	1BA 5BA	1BC 5BC	1BE 5BE
12	1C0 5C0	1C2 5C2	1C4 5C4	1C6 5C6	1C8 5C8	1CA 5CA	1CC 5CC	1CE 5CE
13	1D0 5D0	1D2 5D2	1D4 5D4	1D6 5D6	1D8 5D8	1DA 5DA	1DC 5DC	1DE 5DE

Table 17-21 Parameter RAM Address Offset Map[1]

Channel Number	Parameter							
	0	1	2	3	4	5	6	7
14	1E0 5E0	1E2 5E2	1E4 5E4	1E6 5E6	1E8 5E8	1EA 5EA	1EC 5EC	1EE 5EE
15	1F0 5F0	1F2 5F2	1F4 5F4	1F6 5F6	1F8 5F8	1FA 5FA	1FC 5FC	1FE 5FE

NOTES:

1. These addresses should be added to 0x30 4000 to derive the complete parameter address.

17.5 Time Functions

Descriptions of the MPC555 / MPC556 pre-programmed time functions are shown in **APPENDIX D TPU ROM FUNCTIONS**.

SECTION 18
DUAL-PORT TPU RAM (DPTRAM)

The dual-port RAM module with TPU microcode storage support (DPTRAM) consists of a control register block and a 6-Kbyte array of static RAM, which can be used either as a microcode storage for TPU or as a general-purpose memory.

The DPTRAM module acts as a common memory on the IMB3 and allows the transfer of data to the two TPU3 modules. Therefore, the DPTRAM interface includes an IMB3 bus interface and two TPU3 interfaces. When the RAM is being used in microcode mode, the array is only accessible to the TPU3 via a separate local bus, and not via the IMB3.

The dual-port TPU3 RAM (DPTRAM) is intended to serve as fast, two-clock access, general-purpose RAM memory for the MCU. When used as general-purpose RAM, this module is accessed via the MCU's internal bus.

The DPTRAM module is powered by VDDL in normal operation. The entire array may be used as standby RAM if standby power is supplied via the VDDSRAM pin of the MCU. VDDSRAM must be supplied by an external source.

The DPTRAM may also be used as the microcode control store for up to two TPU3 modules when placed in a special emulation mode. In this mode the DPTRAM array may only be accessed by either or both of the TPU3 units simultaneously via separate emulation buses, and not via the IMB3.

The DPTRAM contains a multiple input signature calculator (MISC) in order to provide RAM data corruption checking. The MISC reads each RAM address and generates a 32-bit data-dependent signature. This signature can then be checked by the host.

The DPTRAM supports soft defects detection (SDD).

NOTE

The RCPU can not perform instruction fetches from any module on the IMB (including the DPTRAM). Only data accesses are permitted.

18.1 Features

- Six Kbytes of static RAM
- Only accessible by the CPU if neither TPU3 is in emulation mode
- Low-power stop operation
  - — Entered by setting the STOP bit in the DPTMCR
  - — Applies only to IMB3 accesses and not to accesses from either TPU3 interface
- TPU microcode mode
  - — The DPTRAM array acts as a microcode storage for the TPU module. This provides a means executing TPU code out of DPTRAM instead of programming it in the TPU ROM.

- Includes built in check logic which scans the array contents and calculates the RAM signature
- IMB3 bus interface
- Two TPU3 interface units
- Bytes, half-word or word accessible

18.2 DPTRAM Configuration and Block Diagram

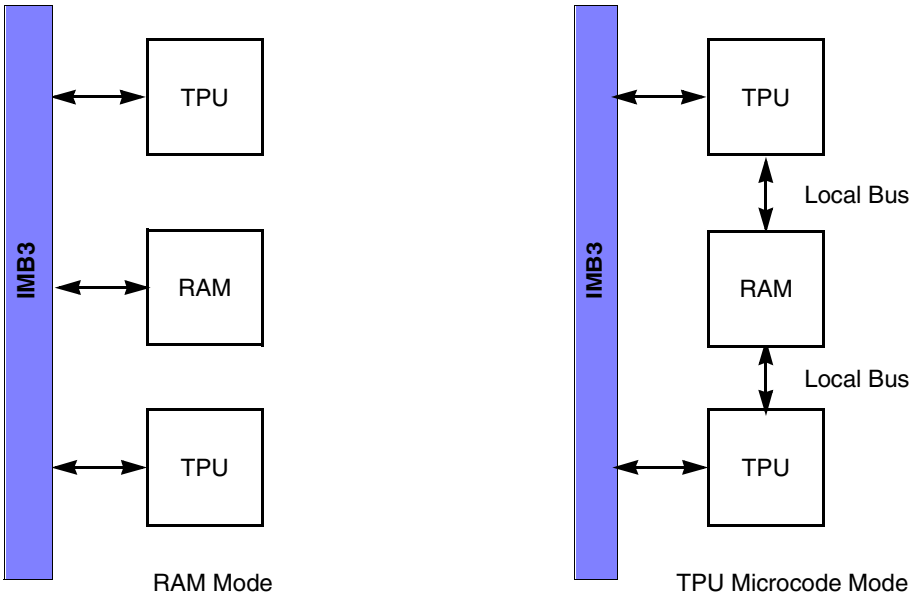


Figure 18-1 DPTRAM Configuration

18.3 Programming Model

The DPTRAM module consists of two separately addressable sections. The first is a set of memory-mapped control and status registers used for configuration (DPTMCR, RAMBAR, MISRH, MISRL, MISCNT) and testing (DPTTCR) of the DPTRAM array. The second section is the array itself.

All DPTRAM module control and status registers are located in supervisor data space. User reads or writes of these will result in a bus error.

When the TPU3 is using the RAM array for microcode control store, none of these control registers have any effect on the operation of the RAM array.

All addresses within the 64-byte control block will respond when accessed properly. Unimplemented addresses will return zeros for read accesses. Likewise, unimplemented bits within registers will return zero when read and will not be affected by write operations.

Table 18-1 shows the DPTRAM control and status registers. The addresses shown are offsets from the base address for the module. Refer to **1.3 MPC555 / MPC556 Ad-**

dress Map to locate the DPTRAM control block in the MPC555 / MPC556 address map.

Table 18-1 DPTRAM Register Map

R/W Access	Address	Register	Reset Value
Supv R/W	0x30 0000	DPT RAM Module Configuration Register (DPTRMCR) See **Table 18-2** for bit descriptions.	0x0100
Test	0x30 0002	Test Configuration Register (DPTTCR)	0x0000
Supv R/W	0x30 0004	RAM Base Address Register (RAMBAR) See **Table 18-3** for bit descriptions.	0x0001
Supv Read Only	0x30 0006	Multiple Input Signature Register High (MISRH) See **18.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0x30 0008	Multiple Input Signature Register Low (MISRL) See **18.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0x30 000A	Multiple Input Signature Counter (MISCNT) See **18.3.5 MISC Counter (MISCNT)** for bit descriptions.	Last memory address

The DPTRAM array occupies the 6-Kbyte block. In the MPC555 / MPC556, the array must be located at the address 0x30 2000. Refer to **Figure 1-3** and **Figure 18-2**.

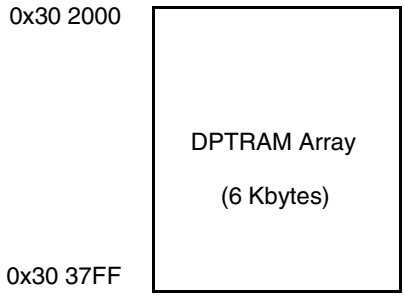


Figure 18-2 DPTRAM Memory Map

18.3.1 DPTRAM Module Configuration Register (DPTMCR)

This register defines the basic configuration of the DPTRAM module. The DPTMCR contains bits to configure the DPT RAM module for stop operation and for proper access rights to the array. The register also contains the MISC control bits.

DPTMCR — DPT Module Configuration Register **0x30 0000**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	NOT USED				MISF	MIS-EN	RASP	Reserved							
RESET:															
0					0	0	1	0	0	0	0	0	0	0	0

Table 18-2 DPTMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low power stop (sleep) mode 0 = DPTRAM clocks running 1 = DPTRAM clocks shut down Only the STOP bit in the DPTMCR may be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior. Refer to **18.4.4 Stop Operation** for more information.
1:4	—	Reserved
5	MISF	Multiple input signature flag. MISF is readable at any time. This flag bit should be polled by the host to determine if the MISC has completed reading the RAM. If MISF is set, the host should read the MISRH and MISRL registers to obtain the RAM signature. 0 = First signature not ready 1 = MISC has read entire RAM. Signature is latched in MISRH and MISRL and is ready to be read.
6	MISEN	Multiple input signature enable. MISEN is readable and writable at any time. The MISC will only operate when this bit is set and the MPC555 / MPC556 is in TPU3 emulation mode. When enabled, the MISC will continuously cycle through the RAM addresses, reading each and adding the contents to the MISR. In order to save power, the MISC can be disabled by clearing the MISEN bit. 0 = MISC disabled 1 = MISC enabled
7	RASP	Ram area supervisor/user program/data. The RAM array may be placed in supervisor or unrestricted Space. When placed in supervisor space, (RASP = 1), only a supervisor may access the array. If a supervisor program is accessing the array, normal read/write operation will occur. If a user program is attempting to access the array, the access will be ignored and the address may be decoded externally. 0 = Both supervisor and user access to RAM allowed 1 = Supervisor access only to RAM allowed
8:15	—	Reserved

18.3.2 DPTRAM Test Register

RAMTST — Test Register **0x30 0002**

RAMTST is used only during factory testing of the MCU.

18.3.3 Ram Base Address Register (RAMBAR)

The RAMBAR register is used to specify the 16 MSBs of the starting DPT RAM array location in the memory map. In the MPC555 / MPC556, this register must be programed to the value 0xFFA0.

This register can be written only once after a reset. This prevents runaway software from inadvertently re-mapping the array. Since the locking mechanism is triggered by the first write after reset, the base address of the array should be written in a single operation. Writing only one half of the register will prevent the other half from being written.

Soft reset has no effect on this register.

RAMBAR — RAM Array Base Address Register **0x30 0004**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	Reserved				RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 18-3 RAMBAR Bit Descriptions

Bit(s)	Name	Description
0:10	A[8:18]	RAM array base address. These bits specify the 11 high-order bits (address lines ADDR[8:18] in little-endian notation) of the 24-bit base address of the RAM array. This allows the array to be placed on a 8-Kbyte boundary anywhere in the memory map. It is the users responsibility not to overlap the RAM array memory map with other modules on the chip. On the MPC555 / MPC556 the value 0xFFA0 must be used.
11:14	—	Reserved. (Bits 11:12 represent A[19:20] in DPTRAM implementations that require them.
15	RAMDS	RAM disabled. RAMDS is a read-only status bit. The RAM array is disabled after a master reset since the RAMBAR register may be incorrect. When the array is disabled, it will not respond to any addresses on the IMB3. Access to the RAM control register block is not affected when the array is disabled. RAMDS is cleared by the DPTRAM module when a base address is written to the array address field of RAMBAR. RAMDS = 0: RAM enabled RAMDS = 1: RAM disabled

18.3.4 MISR High (MISRH) and MISR Low (MISRL)

The MISRH and MISRL together contain the 32-bit RAM signature calculated by the MISC. These registers are read-only and should be read by the host when the MISF bit in the MCR is set. Note that the naming of the D[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode results in the reset of both MISRH and MISRL

MISRH — Multiple Input Signature Register High **0x30 0006**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15

MISRH — Multiple Input Signature Register High **0x30 0006**

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISRL — Multiple Input Signature Register Low **0x30 0008**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.3.5 MISC Counter (MISCNT)

The MISCNT contains the address of the current MISC memory access. This registers is read-only. Note that the naming of the A[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode or clearing the MISEN bit in the DPTMCR results in the reset of this register.

MISCNT — MISC Counter **0x30 000A**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED			A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET:															
			Last Memory Address												

18.4 Operation

The DPTRAM module has several modes of operation. The following sections describe DPTRAM operation in each of these modes.

18.4.1 Normal Operation

In normal operation, the DPTRAM is powered by V_{DDL} and may be accessed via the IMB3 by a bus master.

Read or write accesses of 8, 16, or 32 bits are supported. In normal operation, neither TPU3 accesses the array, nor do they have any effect on the operation of the DPTRAM module.

18.4.2 Standby Operation

The DPTRAM array uses a separate power supply VDDSRAM to maintain the contents of the DPTRAM array during a power-down phase.

When the RAM array is powered by the VDDSRAM pin of the MCU, access to the RAM array is blocked. Data read from the RAM array during this condition cannot be guaranteed. Data written to the DPTRAM may be corrupted if switching occurs during a write operation.

In order to guarantee valid DPTRAM data during power-down, external low voltage inhibit circuitry (external to the MCU) must be designed to force the RESET pin of the MCU into the active state before VDDL drops below its normal limit. This is necessary to inhibit spurious writes to the DPTRAM during power-down.

18.4.3 Reset Operation

When a synchronous reset occurs, a bus master is allowed to complete the current access. Thus a write bus cycle (byte or half word) that is in progress when a synchronous reset occurs will be completed without error. Once a write already in progress has been completed, further writes to the RAM array are inhibited.

NOTE

A word (32-bit) write will be completed coherently only if the reset occurs during the second (16-bit) write bus cycle. If reset occurs during the first write bus cycle, only the first half word will be written to the RAM array and the second write will not be allowed to occur. In this case, the word data contained in the DPTRAM will not be coherent. The first half word will contain the most significant half of the new word information and the second half word will contain the least significant half of the old word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the RAM array are not guaranteed. (Refer to **SECTION 7 RESET** for a description of MPC555 / MPC556 reset sources, operation, control, and status.)

Reset will also reconfigure some of the fields and bits in the DPTRAM control registers to their default reset state. See the description of the control registers to determine the effect of reset on these registers.

18.4.4 Stop Operation

Setting the STOP control bit in the DPTMCR causes the module to enter its lowest power-consuming state. The DPTMCR can still be written to allow the STOP control bit to be cleared.

In stop mode, the DPTRAM array cannot be read or written. All data in the array is retained. The BIU continues to operate to allow the CPU to access the STOP bit in the DPTMCR. The system clock remains stopped until the STOP bit is cleared or the DPTRAM module is reset.

The STOP bit is initialized to logical zero during reset. Only the STOP bit in the DPTMCR can be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior.

Switching to VDDSRAM occurs if VDDL drops below its specified value when the RAM module is in stop mode.

The DPTRAM will not enter stop mode if either or both of the TP1EMM or TP2EMM signals are asserted, indicating TPU3 emulation mode.

18.4.5 Freeze Operation

The FREEZE line on the IMB3 has no effect on the DPTRAM module. When the freeze line is set, the DPTRAM module will operate in its current mode of operation. If the DPTRAM module is not disabled, (RAMDS = 0), it may be accessed via the IMB3. If the DPTRAM array is being used by the TPU in emulation mode, the DPTRAM will still be able to be accessed by the TPU microengine.

18.4.6 TPU3 Emulation Mode Operation

To emulate TPU3 time functions, the user stores the microinstructions required for all time functions to be used, in the RAM array. This must be done with the DPTRAM in its normal operating mode and accessible from the IMB3. After the time functions are stored in the array, the user places one or both of the TPU3 units in emulation mode. The RAM array is then controlled by the TPU3 units and disconnected from the IMB3.

To use the DPTRAM for microcode accesses, set the EMU bit in the corresponding TPU3 module configuration register. Through the auxiliary buses, the TPU3 units can access word instructions simultaneously at a rate of up to 40 MHz.

When the RAM array is being used by either or both of the TPU3 units, all accesses via the IMB3 are disabled. The control registers have no effect on the RAM array. Accesses to the array are ignored, allowing an external RAM to replace the function of the general-purpose RAM array.

The contents of the RAM are validated using a multiple input signature calculator (MISC). MISC reads of the RAM are performed only when the MPC555 / MPC556 is in emulation mode and the MISC is enabled (MISEN = 1 in the DPTMCR).

Refer to **17.3.6 Emulation Support** for more information in TPU3 and DPTRAM operation in emulation mode.

18.5 Multiple Input Signature Calculator (MISC)

The integrity of the RAM data is ensured through the use of a MISC. The RAM data is read in reverse address order and a unique 32-bit signature is generated based on the output of these reads. MISC reads are performed when one of the TPU3 modules does not request back-to-back accesses to the RAM provided that the MISEN bit in the DPTMCR is set.

The MISC generates the DPTRAM signature based on the following polynomial:

$$G(x) = 1 + x + x^2 + x^{22} + x^{31}$$

After the entire RAM has been read and a signature has been calculated, the MISC sets the MISF bit in the DPTMCR. The host should poll this bit and enter a handling routine when the bit is found to be set.

The signature should be then read from the MISRH and MISRL registers and the host determines if it matches the predetermined signature.

The MISRH and MISRL registers are updated each time the MISC completes reading the entire RAM regardless of whether or not the previous signature has been read or not. This ensures that the host reads the most recently generated signature.

The MISC can be disabled by clearing the MISEN bit in the DPTMCR. Note that the reset state of the MPC555 / MPC556 MISEN is disabled.

SECTION 19
CDR MoneT FLASH EEPROM

19.1 Introduction

The two CDR MoneT flash EEPROM modules (CMF) serve as electrically programmable and erasable non-volatile memory (NVM) to store system program and data. The modules are designed to be used with the unified bus (U-bus). The CMF arrays use Motorola's one-transistor (MoneT) bit cell technology. The MPC555 / MPC556's total 448-Kbytes of flash EEPROM non-volatile memory are distributed between two CMF EEPROM modules: a 256-Kbyte array and a 192-Kbyte array. The erase block size is 32 Kbytes.

Each CMF EEPROM module is arranged into two major sections. The first section is the flash EEPROM array used to store system program and data. The second section is the bus interface unit (BIU) that controls access and operation of the array through a standard U-bus interface and the external signals EPEE (external program or erase enable) and VPP (supply program or erase power).

Each CMF EEPROM module array is divided into blocks to allow for independent erase, access state, and protection from program and erase for each block. Information is transferred to the CMF EEPROM through the U-bus a word (32 bits), half-word (16 bits), or byte at a time.

The BIU accesses 32 bytes of information in the array at a time. These bytes are copied into a read-page buffer aligned to the low order addresses, ADDR[27:31]. Each CMF module contains two non-overlapping page buffers. The first page buffer is associated with array blocks zero to three. The second page is associated with array blocks four to seven (for CMF Module A), or blocks four to five (for CMF Module B).

Read access time for data in the page buffers (on-page read) is one system clock. The read access time for a new page of data (off-page read) is two system clocks. To prevent the BIU from accessing an unnecessary page from the array, the CMF EEPROM monitors the U-bus address to determine whether the required information is in one of the two current pages and whether the access is valid for the module.

Burst accesses are not supported by the CMF EEPROM. In normal operation, write accesses to the CMF array are not recognized.

The CMF EEPROM module requires an external program or erase voltage (VPP) to program or erase the array or any of its control register shadow bits. Special hardware interlocks and the external signal EPEE protect the array from accidental enabling of program and erase operation. The program and erase algorithms are implemented by a series of writes to the CMF EEPROM registers and are under software control.

Up to eight unique 64-byte pages are programmed simultaneously in eight separate array blocks of each of the two CMF modules. Each of the pages being programmed simultaneously is located at the same block offset address, ADDR[17:25].

Erasing is performed on one or more of the selected block(s) simultaneously.

19.1.1 MPC555 / MPC556 CMF Features

- Motorola's one-transistor (MoneT) bit cell
- Reset configuration stored in special FLASH NVM locations
- Array sizes of 256 Kbytes and 192 Kbytes
- Arrays distributed in 32-Kbyte blocks
  - — Erase by block(s)
  - — Block protection for program and erase operations
  - — Block access state control
    - Select between supervisor and supervisor/user spaces
    - Select between data and instruction/data spaces
  - — 32-bit word length
- Page mode read
  - — Retains two separate pages per CMF module
  - — Page size of 32 bytes (eight words)
  - — Off-page read access time of two system clocks
  - — On-page read access time of one system clock
- Supports U-bus pipelined accesses to a pipe depth of two
- Program up to 512 bytes at a time per CMF module
  - — Program up to eight unique 64-byte pages of data in eight separate blocks simultaneously (CMF Module A)
  - — Program up to six unique 64-byte pages of data in six separate blocks simultaneously (CMF Module B)
  - — Program CMF Module A and CMF Module B at the same time
  - — Pages located at the same offset address
- Self-timed program and erase pulses
  - — Internal pulse width timing control generates pulses from 4.00 µs to 2.73 s using system clock frequencies from 8.0 MHz to 40.0 MHz
- Censored access mode with a user bypass for uncensored access per CMF module
- External 4.75 to 5.25-V VPP program and erase power supply
- External program or erase enable signal (EPEE)
- CMF arrays are hardware mapped to the first 512 Kbytes of the 4-Mbyte internal address space of the U-bus
- Supports external emulation

19.1.2 Glossary of Terms for the CMF EEPROM

- **Array block** — 32-Kbyte contiguous block of information. Each array block may be erased independently.
- **BIU** — Bus interface unit controls access and operation of the CMF array through a standard U-bus interface.
- **Cleared censorship** — CENSOR[0:1] = 00. The CMF EEPROM can change to

either no censorship or information censorship without modifying the CMF array contents. Cleared censorship will prevent CMF array accesses when the device is censored and ACCESS = 0.

- **Erase interlock write** — A write to any CMF array address after initializing the erase sequence.
- **Erase margin read** — Special off-page read of the CMF array where the CMF EEPROM hardware adjusts the reference of the sense amplifier to check for correct erase operation. All CMF array off-page read accesses between the erase interlock write and clearing the SES bit are erase margin reads.
- **Information censorship** — CENSOR[0:1] = 11. Requires an erase of the CMF EEPROM to change CENSOR[0:1]. Information censorship will prevent CMF array accesses when the device is censored and ACCESS = 0. Information stored in the CMF array is made invalid while clearing CENSOR[0:1].
- **Initialize program/erase sequence** — The write to the high voltage control register that changes the SES bit from a zero to a one.
- **MoneT** — The CMF EEPROM's FLASH bit cell.
- **No censorship** — CENSOR[0:1] = 10 or 01, The CMF EEPROM can change to information censorship without modifying the CMF array contents. No censorship allows all CMF array accesses.
- **Off-page read** — Array read operation that requires two clocks and updates a page buffer.
- **On-page read** — Array read operation that accesses information in one of the read page buffers and requires one clock.
- **Over-programmed** — By exceeding the specified programming time and/or voltage a CMF bit may be over-programmed. This bit will cause erased bits on the same column in the same array block to read as programmed.
- **Programming write** — A word write to a CMF array address to transfer information into a program page buffer. The CMF EEPROM accepts programming writes from after initializing the program sequence until the EHV bit is changed from a zero to a one.
- **Program margin read** — Special off-page read of the CMF array where the CMF EEPROM hardware adjusts the reference of the sense amplifier to check for correct program operation. All CMF array off-page read accesses between the first programming write and clearing the SES bit are program margin reads.
- **Program page buffer** — 64 bytes of information used to program the CMF array. This information is aligned to a 64-byte boundary within the CMF array block. Each CMF module has one program page buffer for each array block.
- **Read page buffer** — 32-byte block of information that is read from the CMF array. This information is aligned to a 32-byte boundary within the CMF array. Each CMF module has two read page buffers.
- **Shadow information** — An extra row (256 bytes) of the CMF array used to provide reset configuration information. This row may be accessed by setting the SIE bit in the module configuration register and accessing the CMF array. The shadow information is always in the lowest array block of the CMF array.

19.2 Programming Model

The CMF EEPROM Module consists of two addressed sections. The first is the 32-byte control registers used to configure, program, erase and test the CMF EEPROM array while, the second is the array.

Figure 19-1 shows the part of the MPC555 / MPC556 memory map involving the CMF arrays and control registers. Refer to **1.3 MPC555 / MPC556 Address Map** for the complete memory map.

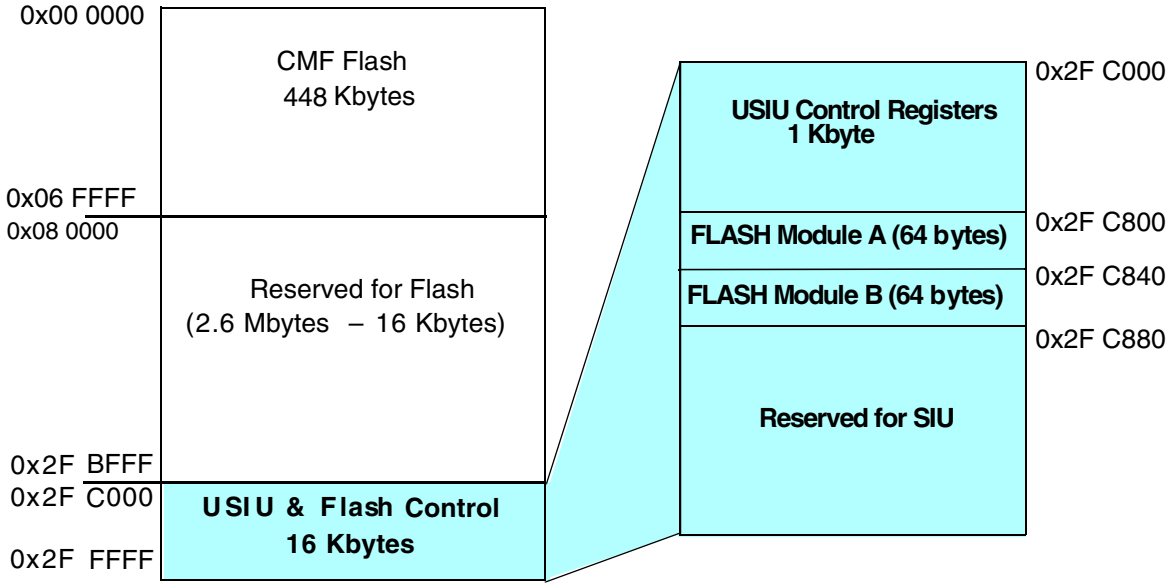


Figure 19-1 CMF Array and Control Register Addressing

19.2.1 CMF EEPROM Control Registers

The control registers are used to control CMF EEPROM module operation. They reside in supervisor data space. On master reset the registers are loaded with default reset information. Some of the registers are special CMF NVM registers which retain their state when power is removed from the CMF EEPROM. These special FLASH NVM registers are identified in the individual register field and control bit descriptions.

The CMF EEPROM control registers are accessible for read or write operation at all times while the device is powered up except during master reset, soft reset or erase interlock write.

The access time of a CMF register is one system clock for both read and write accesses. Accesses to reserved registers will cause the BIU to generate a data error exception.

Table 19-1 is a programming model for each set of CMF EEPROM control registers. The address offset is from the start of the control register block for each CMF module. (See **Figure 19-1**.)

Table 19-1 CMF Register Programmer's Model

Address	MSB 0 Register LSB 31
Control Registers (Located in Supervisor Data Space)	
0x2F C800 0x2F C840	CMF Module Configuration Register (CMFMCR) See **Table 19-2** for bit descriptions.
0x2F C804 0x2F C844	CMF EEPROM Test Register (CMFTST) See **Table 19-3** for bit descriptions.
0x2F C808 0x2F C848	High Voltage Control Register (CMFCTL) See **Table 19-6** for bit descriptions.
0x2F C80C — 0x2F C81C 0x2F C84C — 0x2F C85C	Reserved
CMF Flash Array	
0x00 0000 – 0x03 FFFF	CMF_A RAM Array
0x04 0000 – 0x06 FFFF	CMF_B RAM Array

19.2.1.1 CMF EEPROM Configuration Register (CMFMCR)

The CMF EEPROM module configuration register is used to control the operation of the CMF EEPROM array and BIU. Two bits (the Censor bits) of the CMFMCR bits are special FLASH NVM registers. The factory default state of the Censor bits is either 0b01 or 0b10.

CMFMCR — CMF EEPROM Configuration Register **0x2F C800 0x2F C840**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LOCK	0	0	FIC	SIE	AC-CESS	CENSOR[1]		SUPV[0:7]							
RESET:															
1	0	0	0	0	0			1	1	1	1	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
DATA[0:7]								PROTECT[0:7]							
RESET:															
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

1. The reset state of bits 6:7 are defined by special FLASH NVM registers. The factory default state is either 0b01 or 0b10.

Table 19-2 CMFMCR Bit Descriptions

Bit(s)	Name	Description
0	$\overline{\text{LOCK}}$	Lock control. When the $\overline{\text{LOCK}}$ control bit is cleared, the following bits are locked: FIC, SUPV[0:7], DATA[0:7] and PROTECT[0:7]. Writes to these bits will have no effect. In normal operation, once the $\overline{\text{LOCK}}$ bit is cleared, the write-lock can only be disabled again by a master reset. The $\overline{\text{LOCK}}$ bit is writable if the device is in background debug mode and CSC = 0. 0 = Write-locked registers are protected 1 = Write-lock is disabled (reset state) **Warning**: If the lock protection mechanism is enabled ($\overline{\text{LOCK}}$ = 0) before the PROTECT[0:7] bits are cleared, the device must use background debug mode to program or erase the CMF array.
1:2	—	Reserved
3	FIC	Force information censorship for access development. Refer to **19.8 Censored and Non-Censored Accesses** for details. The FIC bit is write protected by the $\overline{\text{LOCK}}$. If FIC = 1 it cannot be cleared except by a hard reset. 0 = Normal CMF censorship operation 1 = Forces the CMF into information censorship mode, unless ACCESS = 1
4	SIE	Shadow information enable. Refer to **19.3 Shadow Information** for details. The SIE bit is write protected by the SES bit for programming operation. Writes have no effect if (SES = 1 and PE = 0). The SIE bit can be read whenever the registers are enabled. 0 = Normal array access 1 = Disables normal array access and selects the shadow information
5	ACCESS	Enable uncensored access. Refer to **19.8 Censored and Non-Censored Accesses** for details. Writes to this bit have no effect when CSC = 1. This bit can be set only when the MCU is in uncensored mode. 0 = Censored CMF array access allowed only if the CMF censorship is no censorship, (FIC = 0 and CENSOR[0] ≠ CENSOR[1]) 1 = Allows all CMF array access.
6:7	CENSOR	Censor accesses. The value of these bits is determined by the state of two NVM bits in two special NVM fuses. Refer to **19.8 Censored and Non-Censored Accesses** for details. The default reset state of CENSOR is user defined by the FLASH NVM register bits. 00 = Cleared censorship, CMF array access allowed only if device is in uncensored mode or ACCESS = 1 01 = No censorship, All CMF array accesses allowed 10 = No censorship, All CMF array accesses allowed 11 = Information censorship, CMF array access allowed only if device is in uncensored mode or ACCESS = 1
8:15	SUPV[0:7]	Supervisor space. Each array block can be mapped into supervisor or unrestricted address space. When an array block is mapped into supervisor address space, only supervisor accesses are allowed. A user access to a location in supervisor address space will result in a data error exception. When an array block is mapped into unrestricted address space, both supervisor and user accesses are allowed. The SUPV[0:7] bits are write protected by the $\overline{\text{LOCK}}$ and CSC bits. Writes will have no effect if $\overline{\text{LOCK}}$=0 or CSC=1. 0 = Array block M is placed in unrestricted address space 1 = Array block M is placed in supervisor address space (reset value)

Table 19-2 CMFMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
16:23	DATA[0:7]	Data space. Each array block can be mapped into data or both data and Instruction address space. When an array block is mapped into data address space (DATA[M] = 1) only data accesses are allowed. An instruction access to a location in data address space will result in a data error exception. When an array block is mapped into both data and instruction address space (DATA[M] = 0), both data and instruction accesses are allowed. The DATA[0:7] bits are write protected by the $\overline{\text{LOCK}}$ and CSC bits. Writes have no effect if $\overline{\text{LOCK}}$ = 0 or CSC = 1. 0 = Array block M is placed in both data and instruction address spaces (reset value) 1 = Array block M is placed in data address space
24:31	PROTECT [0:7]	Block protect. Each array block of the CMF EEPROM can be protected from program and erase operation by setting PROTECT[M] = 1. The CMF BIU will perform all programming and erase interlocks except the program and erase voltages will not be applied to MoneT locations within the protected array block(s). Writes to PROTECT[0:7] have no effect if $\overline{\text{LOCK}}$ = 0 or CSC = 1 or SES = 1. 0 = Array block M is unprotected 1 = Array block M is protected (default value) **Warning**: If a CMF EEPROM enables the lock protection mechanism ($\overline{\text{LOCK}}$ = 0) before PROTECT is cleared, the device must use background debug mode to program or erase the CMF EEPROM.

19.2.1.2 CMF EEPROM Test Register (CMFTST)

The CMF EEPROM test register (CMFTST) is used to control the test operation of the CMF array. Only six bits [20:23, 25:26] are readable or writeable in normal operation.

CMFTST — CMF EEPROM Test Register **0x2F C804**
0x2F C844

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				NVR[1]	PAWS			RE-SERVED	STE[1,2]	GDB[1]	RESERVED				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. The NVR, STE, and GDB bits are not accessible in all revisions of the MPC555 / MPC556 (2K02A mask sets and earlier).
2. The STE bit should always be programmed as a 0.

Table 19-3 CMFTST Bit Descriptions

Bit(s)	Name	Description
0:19	—	Reserved
20	NVR	Negative voltage range. This bit switches between the low and high voltage range of the negative charge pump in programming and erasing the CMF Flash module when GDB = 0b1. This bit is writeable when HVS = 0b0. 0 = High Range (more negative) 1 = Low Range
21:23	PAWS [0:2]	Program amplitude/width modulation select. The PAWS bits, in conjunction with the NVR and GDB bits can be used to select the programming voltage applied to the drain or gate of the EEPROM bitcell. These bits should be left set to 0b000, unless advised by Motorola. For information about PAWS programming modes, see **Table 19-4** and **Table 19-5**.
24	—	Reserved
25	STE	This bit is reserved for Motorola factory testing and should always be programmed to 0b0. 0 = Normal Operation 1 = Factory test mode use only. This setting could disturb contents of flash.
26	GDB	Gate/drain bias select. This bit works in conjunction with the PAWS bits to select between positive and negative ramped voltages for programing and erasing. This bit is writeable when SES = 0b0. 0 = Positive voltage ramp selected on the bitcell drain. 1 = Negative voltage ramp selected on the bitcell gate.
27:31	—	Reserved

Table 19-4 CMF Programming Algorithm (v6 and Later)

No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
4	256 µs	1	100	1	Mode 4NL	Negative gate ramp (low range)
4	256 µs	1	101	1	Mode 5NL	
4	256 µs	1	110	1	Mode 6NL	
4	256 µs	1	111	1	Mode 7NL	
20	50 µs	0	100	1	Mode 4NL	Negative gate ramp (high range)
20	50 µs	0	101	1	Mode 5NL	
20	50 µs	0	110	1	Mode 6NL	
max. 48,000	50 µs	0	111	1	Mode 7NL	

Table 19-5 CMF Erase Algorithm (v6)

No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
1	100 ms[1]	1	100	1	Mode 4NL	Negative gate ramp (low range)
1	100 ms[1]	1	101	1	Mode 5NL	
1	100 ms[1]	1	110	1	Mode 6NL	
1	100 ms[1]	1	111	1	Mode 7NL	
1	100 ms[1]	0	100	1	Mode 4NL	Negative gate ramp (high range)
1	100 ms[1]	0	101	1	Mode 5NL	
1	100 ms[1]	0	110	1	Mode 6NL	
20	100 ms[2]	0	111	1	Mode 7NL	

NOTES:
1. No margin read after pulse.
2. Do margin read after each pulse.

19.2.1.3 CMF EEPROM High Voltage Control Register (CMFCTL)

The CMF EEPROM high voltage control register is used to control the program and erase operations of the CMF EEPROM module and setting and clearing CENSOR[0:1] fuses. Refer to **19.7 Voltage Control for Programming and Erasing** for more information on this register.

CMFCTL — CMF EEPROM High Voltage Control Register **0x2F C808** **0x2F C848**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HVS	0	SCLKR			0	CLKPE		0	CLKPM						
POR ESET/HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BLOCK[0:7]								0	CSC	EPEE	0	0	PE	SES	EHV
POR ESET/HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 19-6 CMFCTL Bit Descriptions

Bit(s)	Name	Description
0	HVS	High voltage status. During a program or erase pulse this bit is set while the pulse is active or during recovery. The BIU does not acknowledge an access to an array location if HVS = 1. While HVS = 1, SES cannot be changed. This bit is read only; writes have no effect. 0 = Program or erase pulse is not applied to the CMF array or shadow information 1 = Program or erase pulse is applied to the CMF array or shadow information
1	—	Reserved
2:4	SCLKR	System clock range. These bits are write protected by the SES bit. Writes to CMFCTL do not change SCLKR[0:2] if SES = 1. The default reset state of SCLKR[0:2] = 000 for a clock scaling of 1. 000 = Clock scaling of 1 (Not for Customer use.) 001 = Clock scaling of 1 010 = Clock scaling of 3/2 011 = Clock scaling of 2 100 = Clock scaling of 3 101 = Clock scaling of 4 110 = Reserved 111 = Reserved Refer to **19.7.3 System Clock Scaling** for instructions on selecting a clock scaling factor.
5	—	Reserved
6:7	CLKPE	Clock period exponent. The CLKPE, CSC, and PE fields determine the value of the exponential clock multiplier, N. Refer to **19.7.4 Exponential Clock Multiplier** for details. The CLKPE bits are write protected by the SES bit. Writes to CMFCTL will not change CLKPE if SES = 1.The default reset state of CLKPE is 00.
8	—	Reserved
9:15	CLKPM	Clock period multiple. This field determines the linear clock multiplier, M, according to the following equation: M = 1 + CLKPM[0:6] The CLKPM bits are write protected by the SES bit. Writes to CMFCTL will not change CLKPM if SES = 1. The reset state of CLKPM = 0, for a multiplier of 1. Refer to **19.7.5 Linear Clock Multiplier** for more information.
16:23	BLOCK [0:7]	Block program and erase select. The CMF EEPROM array blocks that are selected to be programmed or erased are the blocks for which BLOCK[M] = 1. Bit 16 controls block 0 and bit 23 controls block 7. On the 192-Kbyte array (Flash Module B), blocks 6 and 7 are not available, but these bits need to be set when doing a clear censor operation. Warning: The block bit must be set only for the blocks currently being programed. If the block bits are set for blocks that are not being programmed, the contents of the other blocks could be disturbed. The BLOCK[0:7] bits are write protected by the SES bit. Writes to CMFCTL will not change BLOCK[0:7] if SES = 1. BLOCK[0:7] default reset state is 0x00, not selected for program or erase. 0 = Array block M is not selected for program or erase 1 = Array block M is selected for program or erase
24	—	Reserved
25	CSC	Censor set or clear. CSC configures the CMF EEPROM for setting or clearing the CENSOR bits. If CSC=1 then CENSOR is configured for setting if PE = 0 or clearing if PE = 1. For more information on setting or clearing the CENSOR bits see section **19.8.4 Setting and Clearing Censor**. The CSC bit is write protected by the SES bit. Writes to CMFCTL will not change CSC if SES = 1. 0 = Configure for normal operation (default value) 1 = Configure to set or clear the CENSOR bits

Table 19-6 CMFCTL Bit Descriptions (Continued)

Bit(s)	Name	Description
26	EPEE	EPEE pin status bit. The EPEE bit monitors the state of the external program/erase enable (EPEE) pin. EPEE has a digital filter that requires two consecutive samples to be equal before the output of the filter changes. The CMF samples EPEE when EHV is asserted and holds the EPEE state until EHV is negated. EPEE is a read-only bit; writes have no effect. 0 = High voltage operations are not possible 1 = High voltage operations are possible Refer to **19.9.1 E_{PEE} Signal** for more information.
27:28	—	Reserved
29	PE	Program or erase select. PE configures the CMF EEPROM for programming or erasing. When PE = 0, the array is configured for programming and if SES = 1 the SIE bit will be write locked. When PE = 1, the array is configured for erasing and SES will not write lock the SIE bit. The PE bit is write protected by the SES bit. Writes to CMFCTL will not change PE if SES = 1. 0 = Configure for program operation (default value) 1 = Configure for erase operation
30	SES	Start-end program or erase sequence. The SES bit is write protected by the HVS and EHV bits, unless the PAW bits are set to 0b1xx. Writes to CMFCTL will not change SES if HVS = 1 or EHV = 1. Refer to **19.7.7 Starting and Ending a Program or Erase Sequence** for more information. 0 = CMF EEPROM not configured for program or erase operation 1 = Configure CMF EEPROM for program or erase operation
31	EHV	Enable high voltage. EHV can be asserted only after the SES bit has been asserted and a valid programming write(s) or erase hardware interlock write has occurred. If an attempt is made to assert EHV when SES is negated, or if a valid programming write(s) or erase hardware interlock write has not occurred since SES was asserted, EHV will remain negated. 0 = Program or erase pulse disabled 1 = Program or erase pulse enabled

19.2.2 CMF EEPROM Array Addressing

The CMF EEPROM array is addressed when an internal access has been initialized and ADDR[10:13] matches the array hardware mapping address. The CMF array location selected is determined by ADDR[14:29] and the bytes are selected by ADDR[30:31] and internal SIZE[0:1] information. **Table 19-6** and **Table 19-7** show the internal mapping of the flash array.

Information in the array is accessed in 32-byte pages. For each CMF module, two read page buffers are assigned to the low order addresses (ADDR[27:31]). The first page buffer is assigned to blocks zero to three; the second to blocks four to seven (for CMF Module A) or four to five (for CMF Module B).

Access time for data in the read page buffers is one system clock; access time for an off-page read is two system clocks. To prevent the BIU from accessing an unnecessary page from the array, the CMF EEPROM monitors the U-bus address to determine whether the required information is within one of the two read page buffers and the access is valid for the module. This strategy allows the CMF EEPROM to have a two-clock read for an off-page access and one clock for an on-page access.

The BIU does not recognize write accesses to the CMF array.

Table 19-7 EEPROM Array Addressing

0 1 2 3 4 5 6	7 8 9	10 11 12 13	14 15 16	17 18 19 20 21 22 23	24 25 26 27 28 29	30 31
0000000	USIU Internal Mapping	Array Hardware Mapping	Block Address	Row Address	Column Address	Byte Addr

Table 19-8 CMF EEPROM Array Address Fields

Bit(s)	Field	Description
0:6	—	The seven high-order address bits of a CMF EEPROM array access (or any MPC555 / MPC556 internal access) must equal zero.
7:9	USIU Internal Mapping	These bits (programmed in the USIU internal memory map register) specify one of eight locations for the MPC555 / MPC556 internal memory map.
10:13	Array Hardware Mapping	These bits determine the location of each array within the MPC555 / MPC556 internal memory map. Values are as follows: Flash module A = 0000 Flash Module B = 0001
14:16	Block Address	These three bits specify one of eight 32-Kbyte blocks within CMF Module A (000 to 111), or one of six 32-Kbyte blocks within CMF Module B (000 to 101)
17:23	Row Address	These seven bits select one of 128 rows within the 32-Kbyte block.
24:29	Column Address	These six bits select one of 64 (word-length) columns within the row. Note also the following: ADDR[24:26] select a 32-byte read page. ADDR[27:29] represent the read page word address. ADDR[24:25] select a 64-byte program page. ADDR[26:29] represent the program page word address.
30:31	Byte Address	Bits 30:31 select a byte within the column.

19.2.2.1 Read Page Buffers

Each CMF array has two 32-byte read page buffers. The fully independent buffers are located in two separate read sections of the array. Each page buffer status and address are monitored in the BIU. The status of the read page buffers is made invalid by any of the following operations:

- Reset
- Programming write
- Erase interlock write
- Setting EHV
- Clearing SES
- Setting or clearing SIE

Each access to the CMF EEPROM array determines whether the requested location is within the current pages. If the requested location is not within the read page buffers, the correct read page buffer is made invalid, and a new page of information is fetched from the array. The page buffer address is updated and status is made valid. If the requested location is within one of the current page buffers or has been fetched from the array, the selected bytes are transferred to the U-bus, completing the access. CMF EEPROM array accesses that make the page buffer(s) invalid (off-page reads) require two system clocks. CMF EEPROM array accesses that do not make the page buffer(s)

invalid (on-page reads) require one system clock. Read buffer management is transparent to user software and is taken care of in hardware.

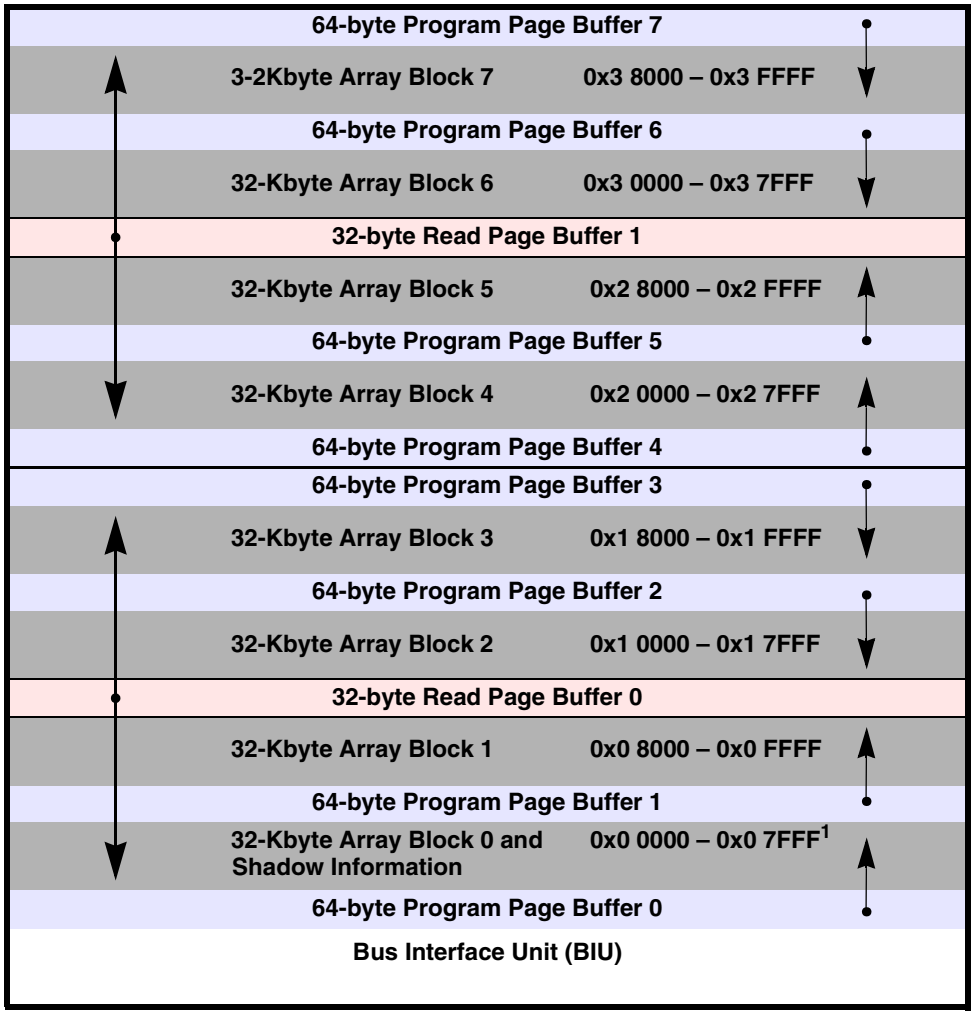
19.2.2.2 Program Page Buffers

The CMF EEPROM modules A and B can program up to eight and six 64-byte pages at one time, respectively. Each program page buffer is located in one program section. All program page buffers within a CMF module share the same block offset address. The block offset address is extracted from the address of the first programming write. To select the CMF EEPROM array block that will be programmed, the program page buffers use the CMF EEPROM array configuration and BLOCK[0:7].

The array block that will be programmed is selected by the BLOCK bit that is set. If BLOCK[M] = 1 then program buffer[M] is active and array block[M] is selected for programming.

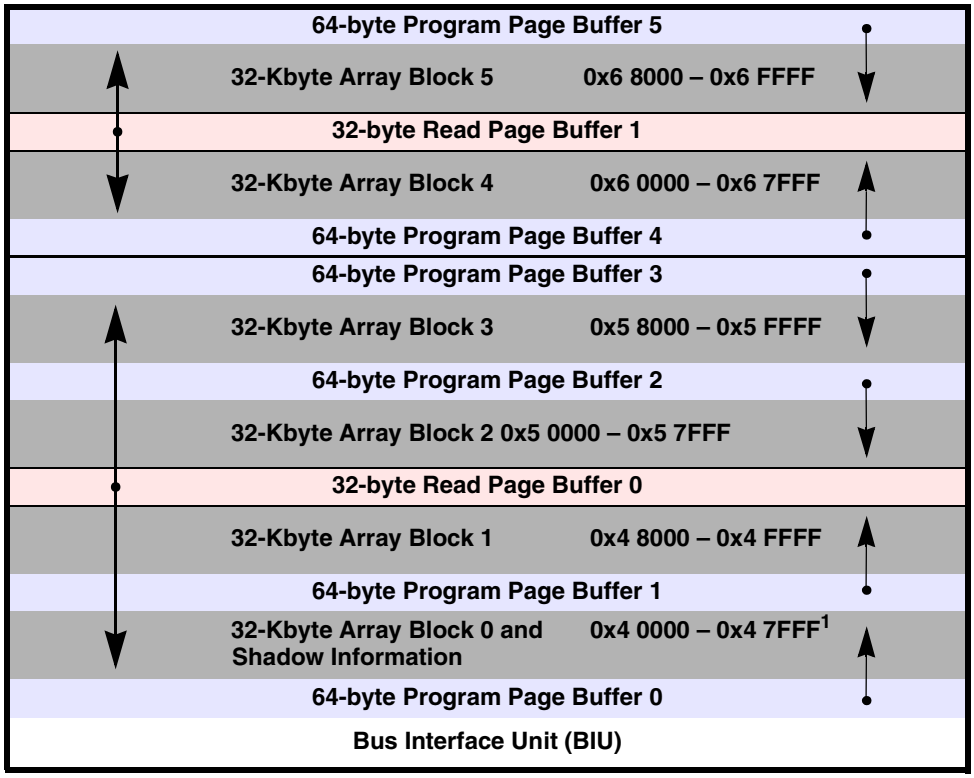
Bits in the program page buffers select the non-program state if SES = 0. During a program margin read, the program buffers update bits to the non-program state for bits that correspond to array bits that the program margin read has determined are programmed.

19.2.2.3 Array Configuration for CMF Module A



1·If SIE=1, then the shadow row is enabled instead of the flash block. Shadow locations 0x00 to 0x03 are reserved for the Internal Flash Reset Configuration word. Shadow locations 0x04 to 0x0F are Reserved by Motorola for possible future use.

19.2.2.4 Array Configuration for CMF Module B



1. If SIE=1, then the shadow row is enabled instead of the flash block. Locations 0x4 00000 to 0x4 000F of the shadow row are Reserved by Motorola for possible future use.

19.3 Shadow Information

Programming the shadow information uses the same procedure as programming the array, except that there are only 256 bytes available in the shadow row. Before starting the program sequence SIE must equal one.

The SIE bit is write protected by the SES bit for programming operation. Writes will have no effect if (SES = 1 and PE = 0). The SIE bit can be read whenever the registers are enabled.

When SIE = 1, normal array accesses are disabled and the shadow information is enabled. When an array location is read using supervisor data in this mode, the shadow information is read from a location determined by the column, 32-byte read page select and read page word addresses (ADDR[24:29]) of the access. Accessing the CMF control block registers accesses the registers and not the shadow information. The read page buffer address monitor is reset whenever SIE is modified, making the next CMF array access an off-page access.

The default reset state of SIE is normal array access (SIE = 0).

NOTE

When SIE = 1, only program page buffer zero can be programmed. All programming writes to the CMF will be directed to program page buffer 0 if SIE = 1. In addition, the other program page buffers cannot be accessed and will not apply any programming voltages to their CMF array blocks while programming the shadow information.

19.3.1 Address Range of Shadow Information

The address range of the shadow information is the entire address range of the CMF EEPROM array but the high order array addresses, ADDR[14:23], are not used to encode the location. The first 16 bytes (ADDR[24:29] = 0x00 to 0x0F) of the 256 bytes of shadow locations are withheld by Motorola for the reset configuration word and future applications. The remaining 240 bytes are available as supervisor data. This is shown in **Figure 19-2**.

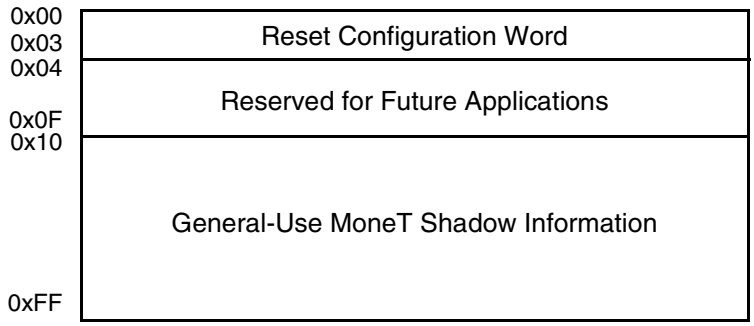


Figure 19-2 Shadow Information

19.3.2 Reset Configuration Word (CMFCFIG)

The CMF EEPROM reset configuration word is implemented in the first word (ADDR[24:29] = 0x00) of the special shadow locations. The reset configuration word along with the rest of the shadow information words is located in supervisor data address space. The purpose of the reset configuration word is to provide the system with an alternative internal source for the reset configuration.

Note that with the exception of bit 20, the bits in the CMFCFIG are identical to those in the USIU hard reset configuration word. Refer to **7.5.2 Hard Reset Configuration Word** for descriptions of these bits.

CMFCFIG — Hard Reset Configuration Word

MSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	IP	BDRV	BDIS	BPS		RESERVED			DBGC		DBPC	ATWC	EBDF		Re-served

LSB

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRPM	SC		ETRE	FLEN	EN_ COMP	EXC_ COMP	Reserved					ISB			DME

During reset the $\overline{HC}$ bit (“has configuration,” bit 20) and the USIU configure the CMF EEPROM module to provide CMFCFIG. If $\overline{HC}$ = 0 and the USIU requests internal configuration during reset the reset configuration word will be provided by CMFCFIG.

The default reset state of the CMFCFIG after an erase operation of the CMF module A block 0 is no configuration word available ($\overline{HC}$ = 1).

19.4 Array Read Operation

The CMF EEPROM array is available for read operation under most conditions while the device is powered up. Reads of the array are not allowed under any of the following conditions:

- During master or soft reset
- ACCESS = 0 and CENSOR[0:1] = 11 or 00
- While the CMF EEPROM is disabled

During programming and erase operations, while the high voltage is applied to the array, the BIU does not acknowledge a CMF array read. At certain points, as defined in the program or erase sequence, reading the array results in a margin read. These margin reads return the status of the program or erase operation and not the data in the array.

The type of CMF EEPROM array read is determined by comparing the address of the requested information with the address of the read page buffers. If the requested address is not within one of the read page buffers or if the read page buffer has been made invalid, an off-page read results. This read updates the read page buffer address of the selected array block, copies the information from the array into the read page buffer, and drives a word onto the data bus. The off-page read requires a minimum of two clocks, while margin off-page reads require additional clocks.

If the address of the requested information is within the address ranges of either of the read page buffers, an on-page read is performed. This requires one clock to transfer information from the read page buffer onto the data bus. See section **19.2.2 CMF EEPROM Array Addressing** for more information on array accesses.

19.5 Programming the CMF Array

To modify the charge stored in the isolated element of the CMF bit from a logic one state to a logic zero state, a programming operation is required. This programming operation applies the required voltages to change the charge state of the selected bits without changing the logic state of any other bits in the CMF array. The program operation cannot change the logic zero state to a logic one state; this must be done by the erase operation. Programming uses a set of program buffers of 64 bytes each to store the required data, an address offset buffer to store the starting address of the block(s) to be programmed and a block select buffer that stores information on which block(s) are to be programmed. Any number of the array blocks may be programmed at one time.

WARNING

Do not program any page more than once after a successful erase operation. While this will not physically damage the array it will cause an increased partial disturb time for the unselected bits on the row and columns that are not programmed. If this happens, a full erase of all blocks being programmed must be done before the CMF EEPROM can be used reliably.

If block M of the CMF EEPROM is protected (PROTECT[M] = 1), it will not be programmed. Also, if EPEE = 0, no programming voltages will be applied to the array. Software should verify the state of EPEE prior to programming (programming will fail if EPEE = 0). The user should also insure that the programming voltage (5.0 ± 0.25 volts) is applied to VPP.

19.5.1 Program Sequence

The CMF EEPROM module requires a sequence of writes to the high voltage control register (CMFCTL) and to the programming page buffer(s) in order to enable the high voltage to the array or shadow information for program operation. See **Table 19-4** for the programming algorithm bit settings.

The required program sequence follows.

1. Write PROTECT[0:7] to disable protection on blocks to be programmed.
2. Write PAWS to 0b100, write NVR = 1, write GDB = 1.
3. Using **19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM**, program the following fields:
   — Pulse width timing control fields for a program pulse
   — BLOCK[0:7] to select the array blocks to be programmed
   — PE = 0 in the CMFCTL register
4. Write SES = 1 in the CMFCTL register.

NOTE

Step 4 can be accomplished with the same write as that in step 3. It is listed as a separate step in the sequence for looping.

5. Write to the 64-byte array locations to be programmed. This updates the programming page buffer(s) with the information to be programmed. The last write to a word within the program page buffer will be saved for programming. All accesses of the array after the first write are to the same block offset address (ADDR[17:25]) regardless of the address provided. Thus the locations accessed after the first programming write are limited to the page locations to be programmed. Off-page read accesses of the CMF array after the first programming write are program margin reads. (See section **19.5.2 Program Margin Reads**.)

   To select the CMF EEPROM array block(s) to be programmed, the program page buffers use the CMF EEPROM array configuration and BLOCK[0:7]. Subsequent writes fill in the programming page buffers using the block address to select the program page buffer and the page word address (ADDR[26:29]) to select the word in the page buffer.

6. Write EHV = 1 in the CMFCTL register.

   NOTE

   If a program buffer word has not received a programming write no programming voltages will be applied to the drain of the corresponding word in the array. Also, at this point writes to the program page buffers are disabled until SES has been cleared and set.

7. Read the CMFCTL register until HVS = 0.

8. Write EHV = 0.

9. To verify the programming, read the words of the pages that are being programmed. These are program margin reads. (See **19.5.2 Program Margin Reads**.) If any bit is a 1 after reading all of the locations that are being programmed, then another pulse needs to be applied to the these locations. If all the locations verify as programmed go to step 11.

   WARNING

   After a program pulse, read at least one location with ADDR[26] = 0 and one location with ADDR[26] = 1 on each programmed page. Failure to do so may result in the loss of information in the CMF EEPROM array. While this will not physically damage the array a full erase of all blocks being programmed must be done before the CMF EEPROM can be used reliably. For more information see **19.5.3 Over-Programming**.

   To reduce the time for verification, read two locations in each program page that is being programmed after reading a non-programmed bit. The first location must be a location with ADDR[26] = 0, while the second must use ADDR[26] = 1. In addition, after a location has been fully verified (all bits are programmed) it is not necessary to verify the location again, since no further programming voltages will be applied to the drain of the corresponding bits. This will reduce the time required to program the array.

10. If the margin read is successful, then write SES = 0 in the CMFCTL register, otherwise do the following:

    a. Write new pulse width parameters (if required per **Table 19-4**) - SCLKR, CLKPE, CLKPM.

    b. Write new values for PAWS, NVR, and GDB (if required per **Table 19-4**).

    c. Go back to step 6 to apply additional programming pulses.

11. If more information needs to be programmed, go back to step 2.

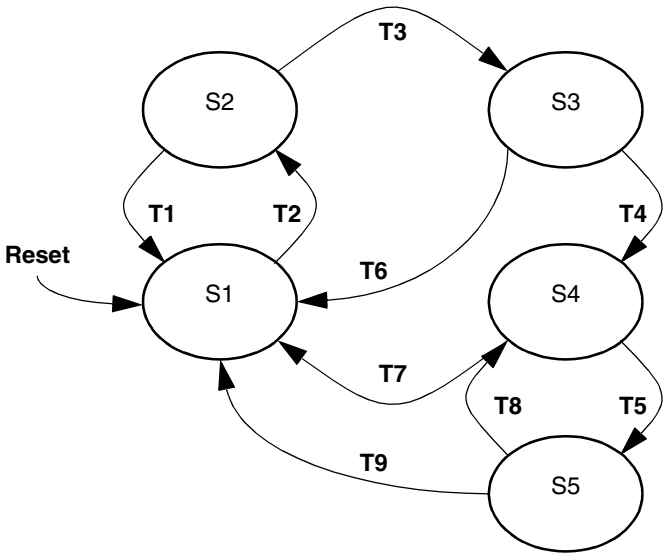


Figure 19-3 Program State Diagram

Table 19-9 Program Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses. The block protect information and pulse width timing control can be modified.	S2	T2	Write PE = 0, SES = 1
S2	First Program Hardware Interlock Write: Normal read operation still occurs. The array will accept programming writes. Accesses to the registers are normal register accesses. A write to CMFCTL can not change EHV at this time. If the write is to a register, no data is stored in the program page buffers, and the CMF remains in state S2.	S1	T1	Write SES = 0 or a master reset
		S3	T3	Hardware Interlock A successful write to any CMF array location. This programming write latches the selected word of data into the programming page buffer and the address is latched to select the location to be programmed. Once a bit has been written then it will remain in the program buffer until another write to the word or a write of SES = 0 or a program margin read determines that the state of the bit needs no further modification by the program operation. If the write is to a register no data will be stored in the program page buffers and the CMF will remain in state S2.
S3	Expanded Program Hardware Interlock Operation: Program margin reads will occur. Programming writes are accepted so that all program pages may be programmed. These writes may be to any CMF array location. The program page buffers will be updated using only the data, the lower address (ADDR[26:29]) and the block address. Accesses to the registers are normal register accesses. A write to CMFCTL can change EHV. If the write is to a register, no data is stored in the program page buffer.	S1	T6	Write SES = 0 or a master reset
		S4	T4	Write EHV = 1
S4	Program Operation: High voltage is applied to the array or shadow information to program the CMF bit cells. The pulse width timer is active if SCLKR[0:2] $\neq$ 0 and HVS can be polled to time the program pulse. No further programming writes are accepted. During programming the array does not respond to any access. Accesses to the registers are allowed. A write to CMFCTL can change EHV only.	S1	T7	Master reset
		S5	T5	Write EHV = 0, disable the internal memory map or a soft reset.

Table 19-9 Program Interlock State Descriptions (Continued)

State	Mode	Next State	Transition Requirement	
S5	Program Margin Read Operation: These reads determines if the state of the bits on the selected page needs further modification by the program operation. Once a bit is fully programmed, the data stored in the program page is updated. No further programming occurs for that bit, and the value read is a 0. While it is not necessary to read all words on a page to determine if another program pulse needs to be applied, all pages being programmed must be read once after each program pulse.	S4	T8	Write EHV = 1
		S1	T9	Write SES = 0 or a master reset.

19.5.2 Program Margin Reads

The CMF EEPROM provides a program margin read with electrical margin for the program state. Program margin reads provide sufficient margin to assure specified data retention. The program margin read is enabled when SES = 1 and a programming write has occurred. To increase the access time of the program margin read, the off-page access time is four clocks instead of the usual two-clock off-page read access time. The program margin read and subsequent on-page program verify reads return a one for any bit that has not been completely programmed. Bits that the programming write left in the non-programmed state return zero when read. Bits that have completed programming return zero when read and update the data in the programming page buffer so that no further programming of those bits will occur. The program margin read occurs during the off-page read. A program margin read must be performed for all pages that are being programmed after each program pulse.

Table 19-10 Results of Programming Margin Read

Current Data in the Program Page Buffer[1]	Current State of Bit	Data Read During Margin Read[2]	New Data for the Program Page Buffer[1]
0	Programmed (0)	0	1
0	Erased (1)	1	0
1	Programmed (0)	0	1
1	Erased (1)	0	1

NOTES:
1. 0 = bit needs further programming
   1 = bit does not need further programming
2. A “0” read during the margin read means that the bit does NOT need further programming. A “1” means the bit needs to be programmed further.

CAUTION

Failure to read each page that is being programmed after each program pulse may result in the loss of information in the CMF EEPROM array. While this will not physically damage the array a full erase of all blocks being programmed must be performed before the CMF EEPROM can be used reliably. For more information, see **19.5.3 Over-Programming**.

19.5.3 Over-Programming

Either of the following events results in an over-programmed state:

- Programming a CMF bit without a program margin read after each program pulse
- Exceeding the specified program times or voltages

Once a CMF bit has been over-programmed, data in the array block (32 Kbytes) that is located in the same column is lost, since the over-programmed bit causes the entire column to appear programmed. To restore an array block with an over-programmed bit, the block must be erased.

19.6 Erasing CMF Array Blocks

To modify the charge stored in the isolated element of the CMF bit from a logic zero state to a logic one state, an erase operation is required. The erase operation cannot change the logic one state to a logic zero state; this is accomplished by the program operation. In the CMF EEPROM, erase is a bulk operation that affects the stored charge of all the isolated elements in an array block.

To make the CMF module block-erasable, the array is divided into blocks that are physically isolated from each other. Each of the array blocks may be erased in isolation or in any combination. The CMF array block size is fixed for all blocks in the module at 32 Kbytes. CMF module A consists of eight array blocks; CMF module B consists of six blocks. Array blocks of the CMF EEPROM that are protected (PROTECT[M] = 1) are not erased. In addition, if EPEE = 0 no erase voltages are applied to the array.

The array blocks selected for erase operation are determined by BLOCK[0:7] and the array configuration.

19.6.1 Erase Sequence

The CMF EEPROM module requires a sequence of writes to the high voltage control register (CMFCTL) and an erase interlock write in order to enable the high voltage to the array and shadow information for erase operation. See **Table 19-5** for erase algorithm bit settings.The erase sequence follows.

1. Write PROTECT[0:7] to disable protect for the blocks to be erased.
2. Write PAWS to 0b100, NVR to 1, and GDB to 1.
3. Using **19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM**, write the pulse width timing control fields for an erase pulse, BLOCK[0:7] to select the blocks to be erased, PE = 1 and SES = 1 in the CMFCTL register. Set the initial

pulse width bit settings per **Table 19-5**.

4. Execute an erase interlock write to any CMF array location.
5. Write EHV = 1 in the CMFCTL register.
6. Read the CMFCTL register until HVS = 0.
7. Write EHV = 0 in the CMFCTL register.
8. To verify the erase operation, read all locations that are being erased, including the shadow information if the block containing it is erased. Off-page reads are erase margin reads that update the read page buffer. (See section **19.6.2 Erase Margin Reads**.) If all the locations read as erased, go to step 9.

NOTE

Do not perform erase margin reads until reaching the condition PAWS=0b111, NVR=0 and GDB=1.

9. To reduce the time used for erase margin reads, upon the first read of a zero, do the following:

   a. Write new pulse width parameters, SCLKR, CLKPE, and CLKPM (if required per **Table 19-5**).

   b. Write new PAWS value (if required per **Table 19-5**).

   c. Write new values for NVR and GDB (if required per **Table 19-5**).

   d. Go back to step 5 to apply additional erase pulses.

NOTE

After a location has been verified (all bits erased), it is not necessary to verify the location after subsequent erase pulses.

10. Write SES = 0 in the CMFCTL register.

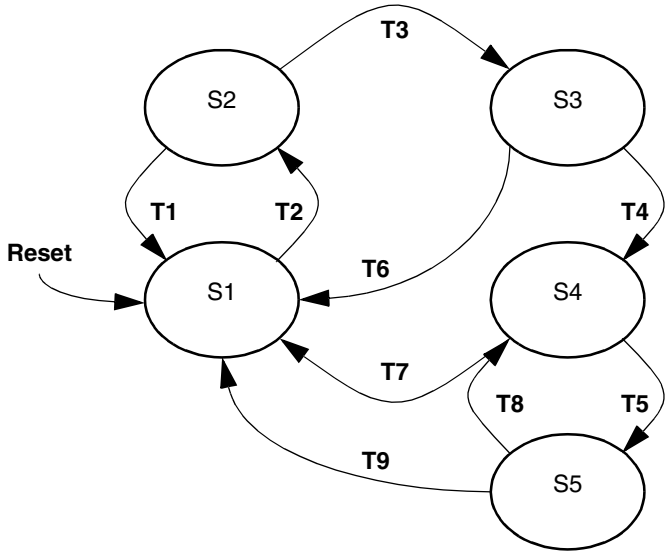


Figure 19-4 Erase State Diagram

Table 19-11 Erase Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses. The Block protect information and pulse width timing control can be modified.	S2	T2	Write PE = 1, SES = 1.
S2	Erase Hardware Interlock Write: Normal read operation still occurs. The CMF accepts the erase hardware interlock write. This write may be to any CMF array location. Accesses to the registers are normal register accesses. A write to CMFCTL can not set EHV at this time. A write to the register is not an erase hardware interlock write, and the CMF remains in state S2.	S1	T1	Write SES = 0 or a master reset
		S3	T3	Hardware Interlock A successful write to any CMF array location is the erase interlock write. If the write is to a register the erase hardware interlock write has not been done and the CMF will remain in state S2.
S3	High Voltage Write Enable Erase margin reads will occur. Accesses to the registers are normal register accesses. A write to CMFCTL can change EHV.	S1	T6	Write SES = 0 or a master reset
		S4	T4	Write EHV=1
S4	Erase Operation: High voltage is applied to the array blocks to erase the CMF bit cells. The pulse width timer is active if SCLKR[0:2] $\neq$ 0, and HVS can be polled to time the erase pulse. During the erase operation, the array does not respond to any address. Accesses to the registers are allowed. A write to CMFCTL can change EHV only.	S1	T7	Master reset
		S5	T5	Write EHV = 0, disable the internal memory map or a soft reset
S5	Erase Margin Read Operation: These reads determine whether the state of the bits in the selected blocks needs further modification by the erase operation. Once a bit is fully erased it returns one when read. All words within the blocks being erased must be read to determine whether the erase operation is completed.	S4	T8	Write EHV = 1
		S1	T9	Write SES = 0 or a master reset

19.6.2 Erase Margin Reads

The CMF EEPROM provides an erase margin read with electrical margin for the erase state. Erase margin reads provide sufficient margin to ensure specified data retention. The erase margin read is enabled when SES = 1 and the erase write has occurred. The erase margin read and subsequent on-page erase verify reads return a zero for any bit that has not been completely erased. Bits that have completed erasing return one when read. To increase the access time of the erase margin read, the off-page access time is 16 clocks instead of the usual two clock off-page read access time. The erase margin read occurs during an off-page read. All locations within the block(s) being erased must return one when read to determine that no more erase pulses are required.

19.6.3 Erasing Shadow Information Words

The shadow information words are erased with CMF array block zero. To verify that the shadow information words are erased, the SIE bit in CMFMCR must be set to one

during the erase margin read while the shadow information is read. For the erase operation to be completed, block zero must also be fully verified.

NOTE

Setting the SIE bit disables normal array access. SIE should be cleared after verifying the shadow information.

19.7 Voltage Control for Programming and Erasing

Bits for controlling the voltage during programming and erasing are found in the CMFCTL register.

19.7.1 Pulse Status

During a program or erase pulse, the HVS bit is set while the pulse is active or during recovery. The BIU does not acknowledge an access to an array location if HVS = 1. While HVS = 1, SES cannot be changed. The program or erase pulse becomes active by setting the EHV bit and is terminated by clearing EHV or by the pulse width timing control.

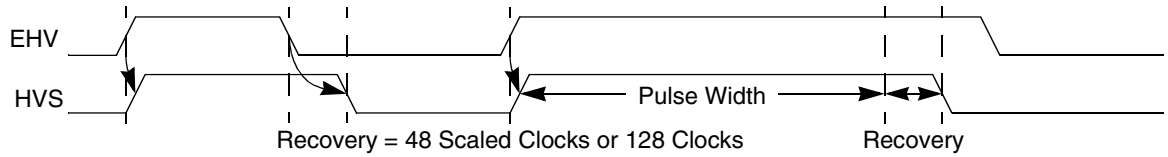


Figure 19-5 Pulse Status Timing

The recovery time is the time required for the CMF EEPROM to remove the program or erase voltage from the array or shadow information before switching to another mode of operation. The recovery time is determined by the system clock range (SCLKR[0:2]) and the PE bit. If SCLKR = 000, the recovery time is 128 clocks. Otherwise, the recovery time is 48 periods of the scaled clock.

Once reset is completed HVS will indicate no program or erase pulse (HVS = 0).

19.7.2 Pulse Width Timing Equation

To control the pulse widths for program and erase operations, the CMF EEPROM uses the system clock and the timing control in CMFCTL. The total pulse time is defined by the following pulse width equation:

$$\text{Pulse Width} = \text{System Clock Period} \bullet R \bullet 2^N \bullet M$$

Where: R = Clock Scaling
N = 5 + CLKPE[0:1] + ((PE | CSC) • 10)
M = 1 + CLKPM[0:6]

The following subsections explain how the values for R, N, and M are determined.

19.7.3 System Clock Scaling

The first term of the pulse width timing equation is the clock scaling, R. The value of R is determined by the system clock range (SCLKR) field. SCLKR defines the pulse timer's base clock using the system clock. Use **Table 19-12** to set SCLKR based upon the system clock frequency. The system clock period is multiplied by the clock scaling value to generate a 83.3-ns to 125-ns scaled clock. This scaled clock is used to run the charge pump submodule and the next functional block of the timing control.

Table 19-12 System Clock Range

SCLKR[0:2]	System Clock Frequency (MHz)		Clock Scaling (R)
	Minimum	Maximum	
000	**Not for customer use**. Program and erase timing control not specified and pulse is not terminated by the timer control. Recovery time is specified to be 128 clocks.		1
001	8	12	1
010	12	18	3/2
011	18	24	2
100	24	36	3
101	36	40	4
110 and 111	Reserved by Motorola for future use		

NOTE

The minimum specified system clock frequency for performing program and erase operations is 8.0 MHz. The CMF EEPROM does not have any means to monitor the system clock frequency and will not prevent program or erase operation at frequencies below 8.0 MHz. Attempting to program or erase the CMF EEPROM at system clock frequencies lower than 8.0 MHz will not damage the device if the maximum pulse times and total times are not exceeded. While some bits in the CMF EEPROM array may change state if programmed or erased at system clock frequencies below 8.0 MHz, the full program or erase transition is not ensured.

WARNING

Never stop the U-bus clock or alter its frequency during a program or erase operation. Changing the clock frequency during a program or erase operation results in inaccurate pulse widths and variations in the charge pump output. This includes loss of system clock/PLL.

19.7.4 Exponential Clock Multiplier

The second term in the pulse width timing equation is the exponential clock multiplier, N. The program pulse number (pulse), clock period exponent (CLKPE), CSC, and PE define the exponent in the 2^N multiply of the clock period. The exponent, N, is defined by the equation:

$$N = 5 + CLKPE[0:1] + ((PE \mid CSC) \bullet 10)$$

All of the exponents are shown in **Table 19-12**.

Table 19-13 Clock Period Exponent and Pulse Width Range

PE | CSC	CLKPE[0:1]	Exponent (N)	Pulse Width Range for all System Clock Frequencies from 8.0 MHz to 40.0 MHz.					
			Minimum Pulse Width			Maximum Pulse Width		
			8 MHz[1] 2^N•1.25E-7	10 MHz[1] 2^N•1E-7	12 MHz[1] 2^N•0.833E-7	8 MHz[1] 2^N•1.25E-07	10 MHz[1] 2^N•1E-7	12 MHz[1] 2^N•0.833E-7
0	00	5	4 µs	3.2 µs	2.7 µs	512 µs	409.6 µs	341.3 µs
	01	6	8 µs	6.4 µs	5.3 µs	1.024 ms	819.2 µs	682.7 µs
	10	7	16 µs	12.8 µs	10.7 µs	2.048 ms	1.6384 ms	1.365 ms
	11	8	32 µs	25.6 µs	21.3 µs	4.096 ms	3.2768 ms	2.731 ms
1	00	15	4.096 ms	3.28 ms	2.73 ms	524.29 ms	419.43 ms	349.5 ms
	01	16	8.192 ms	6.55 ms	5.46 ms	1.05 s	838.86 ms	699.1 ms
	10	17	16.384 ms	13.11 ms	10.92 ms	2.10 s	1.68 s	1.398 s
	11	18	32.768 ms	26.21 ms	21.85 ms	4.19 s	3.35 s	2.796 s

NOTES:
1. CMF clock frequency after SCKLR scaling. Example: A 40 MHz system clock scaled by 4 (SCLKR[0:2] = 0b101) results in an equivalent CMF clock of 10 MHz.

19.7.5 Linear Clock Multiplier

The third term of the pulse width timing equation is the linear clock multiplier, M. The clock period multiplier, CLKPM[0:6], defines a linear multiplier for the program or erase pulse. The multiplier, M, is defined by the equation:

$$M = 1 + CLKPM[0:6]$$

This allows for the program/erase pulse to be from one to 128 times the pulse set by the system clock period, SCLKR[0:2] and CLKPE[0:1].

The default reset state of CLKPM[0:6] = 000 0000 for a multiplier of one.

19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM

The following example determines the values of the SCLKR, CLKPE, and CLKPM fields for a 25.6 µs program pulse, PE = 0, in a system with a 40 MHz system clock.

In this example system clock frequency = 40 MHz; the system clock period is therefore 25 ns.

1. Determine SCLKR:
   From **Table 19-12** a 40 MHz system clock uses SCLKR[0:2] = 0b101, R = 4.
2. Determine CLKPE:
   From **Table 19-13** a 25.6 µs program pulse, PE = 0, can be generated by exponents in the range of N = 5, 6, 7, or 8. While any of these values can be selected CLKPE[0:1] = 0b00, N = 5, will be used for the example.
3. Determine CLKPM:
   Using the selected values of N and R in the pulse width equation and solving for M yields M = 8. Therefore, CLKPM[0:6] = 0x7 (0b0000111).
4. Check the results:
   Pulse Width = System Clock Period • R • 2^N • M
   Using SCLKR[0:2] = 0b101, CLKPE[0:1] = 0b00, CLKPM[0:6] = 0b0000111 and PE = 0 at 40 MHz system clock. Pulse Width = 25 ns • 4 • 2^5 • 8 = 25.6 µs program pulse.

19.7.7 Starting and Ending a Program or Erase Sequence

The SES bit is used to signal the start and end of a program or erase sequence. At the start of a program or erase sequence, SES is set (written to a one). This locks PROTECT[0:7], SCLKR[0:2], CLKPE[0:1], CLKPM[0:6], BLOCK[0:7], CSC and PE. If PE = 0 and SES = 1, SIE is write-locked. At this point the CMF EEPROM is ready to receive either the programming writes or the erase interlock write.

NOTE

The erase interlock write is a write to any CMF EEPROM array location after SES is set and PE = 1.

If the PE bit is a zero, the CMF BIU accepts programming writes to the CMF array address for programming. The first programming write selects the program page offset address (ADDR[17:25]) to be programmed along with the data for the programming buffers at the location written. All programming writes after the first will update the program buffers using the lower address (ADDR[26:29]) and the block address (ADDR[14:16]) to select the program page buffers to receive the data. For further information see section **19.2.2.2 Program Page Buffers**. After the data has been written to the program buffers the EHV bit is set (written to a one) to start the programming pulse and lock out further programming writes.

If the PE bit = 1, the CMF BIU accepts writes to any CMF array address as an erase-interlock write. An erase interlock write is required before the EHV bit can be set.

At the end of the program or erase operation the SES bit must be cleared (written to a zero) to return to normal operation and release the program buffers, PROTECT[0:7], SCLKR[0:2], CLKPE[0:1], CLKPM[0:6], BLOCK[0:7], CSC and PE.

The default reset state of SES is not configured for program or erase operation (SES = 0).

19.7.8 Controlling the Program/Erase Voltage

The external program or erase enable pin (EPEE) and EHV are used to control the application of the program or erase voltage to the CMF EEPROM module. High voltage operations to the CMF EEPROM array, special MoneT shadow locations or FLASH NVM registers can occur only if EHV = 1 and EPEE = 1.

Only after the correct hardware and software interlocks have been applied to the CMF EEPROM can EHV be set. Once EHV is set SES cannot be changed and attempts to read the array will not be acknowledged.

The default reset state of EHV disables program or erase pulses (EHV = 0). A master reset while EHV = 1 terminates the high voltage operation, and CMF generates the required sequence to disable the high voltage without damage to the high voltage circuits. A soft reset or disabling the internal memory map clears EHV, terminating the high voltage pulse.

19.8 Censored and Non-Censored Accesses

The MPC555 / MPC556 always operates in one of two modes: censored or uncensored.

19.8.1 Uncensored Mode

Uncensored mode provides no censorship. In uncensored mode the FIC, ACCESS, and CENSOR[0:1] bits are irrelevant. The MPC555 / MPC556 operates in uncensored mode unless a specific event occurs to place the device in censored mode.

19.8.2 Censored Mode

The MPC555 / MPC556 enters censored mode in response to any of the following events:

- Booting from external memory
- Any CMF array access from an external master
- Entering background debug mode.

The CMF EEPROM censorship mechanism provides several censorship levels. Four bits in CMFMCR are used to configure the CMF censorship level. These bits are listed in **Table 19-14**.

Table 19-14 Censorship Control Bits

ACCESS	Enables a CMF EEPROM to bypass the censorship
FIC	Overrides CENSOR[0:1] to force information censorship if ACCESS = 0
CENSOR[0:1]	Determine the censorship level of the CMF

In censored mode, the ACCESS and CENSOR bits work together according to **Table 19-15**.

Table 19-15 Levels of Censorship

ACCESS	CENSOR[0:1]	Description
0	11	Information censorship, No CMF array accesses allowed
0	01 or 10	No censorship, CMF array accesses allowed
0	00	Cleared censorship, No CMF array accesses allowed
1	XX	No censorship, CMF array accesses allowed

There are two states of censorship: information censorship (CENSOR[0:1] = 11) and cleared censorship (CENSOR[0:1] = 00). In the information censorship state the entire CMF array must be erased to clear CENSOR[0:1]. In the cleared censorship or no censorship states the bits in CENSOR[0:1] may be set without modifying the information in the CMF array. When FIC=1, the CENSOR bits have no effect upon censorship.

While the device is in uncensored mode, ACCESS may be set to allow the device to enter censored mode and still access the CMF array. ACCESS may not be set while the device is in censored mode but may be cleared.

The default reset state of ACCESS is zero, so that FIC and CENSOR[0:1] control the level of censorship to the CMF EEPROM array. All accesses to the CMF EEPROM array are allowed if ACCESS=1.

If an access is attempted when the device is in censored mode and the following condition holds, the CMF EEPROM module disallows access to the array and signals a bus error:

$$((CENSOR[0] = CENSOR[1])|(FIC = 1)) \text{ AND } (ACCESS = 0)$$

If CENSOR[0:1] is in the no-censorship state, however (CENSOR[0]≠CENSOR[1]), the CMF EEPROM module recognizes accesses to its address space.

When FIC = 1, the CENSOR bits have no effect upon censorship. If ((FIC = 1) and (ACCESS = 0)) the CMF is in information censorship mode. If ((FIC = 1) and (ACCESS = 1)), the CMF is in normal access mode. This arrangement aids in the development of custom techniques for controlling the ACCESS bit without setting CENSOR[0:1] to the information censorship state. Using FIC to force information censorship allows testing of the hardware and software for setting ACCESS without setting CENSOR[0:1] = 11.

The default reset state of FIC is normal censorship operation (FIC = 0).

19.8.3 Device Modes and Censorship Status

Table 19-16 summarizes the various combinations of censorship mode and states of the ACCESS, FIC, and CENSOR[0:1] bits.

When booting from the internal flash, the default state is #8 unless BDM was entered, or a slave access to the MPC555 / MPC556 occurred. When any of these three conditions occur, then the state of CENSOR[0:1] determine whether the flash array can be accessed:

1. BDM is active
2. Accessing the MPC555 / MPC556 flash via a slave mode read
3. Booting from an external memory

Table 19-16 CMF EEPROM Devices Modes and Censorship Status

Device Mode	Censored							Uncensored			
ACCESS	0					1		0		1	
FIC	0			1		0	1	0	1	0	1
CENSOR[0:1]	00	01 or 10	11	00, 01 or 10	11	00, 01, 10 or 11					
CMF EEPROM Status	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**	**#10**	**#11**
#1	CMF array can not be accessed. ACCESS can not be changed. FIC can be set. CENSOR[0:1] can be set. CENSOR[0:1] can not be cleared.										
#2	CMF array can be accessed. ACCESS can not be changed. FIC can be set. CENSOR[0:1] can be set. CENSOR[0:1] can be cleared.										
#3	CMF array can not be accessed. ACCESS can not be changed. FIC can be set. CENSOR[0:1] can not be cleared.										
#4	CMF array can not be accessed. ACCESS can not be changed. FIC can not be changed. CENSOR[0:1] can be set. CENSOR[0:1] can not be cleared.										
#5	CMF array can not be accessed. ACCESS can not be changed. FIC can not be changed. CENSOR[0:1] can not be cleared.										
#6	CMF array can be accessed. ACCESS can be cleared. FIC can be set. CENSOR[0:1] can be changed.										
#7	CMF array can be accessed. ACCESS can be cleared. FIC can not be changed. CENSOR[0:1] can be changed.										
#8	CMF array can be accessed. ACCESS can be changed. FIC can be set. CENSOR[0:1] can be changed.										
#9	CMF array can be accessed. ACCESS can be changed. FIC can not be changed. CENSOR[0:1] can be changed.										
#10	CMF array can be accessed. ACCESS can be changed. FIC can be set. CENSOR[0:1] can be changed.										
#11	CMF array can be accessed. ACCESS can be changed. FIC can not be changed. CENSOR[0:1] can be changed.										

= Indicates that the CMF array can not be accessed.

The only way CENSOR[0:1] can be changed is by setting or clearing the FLASH NVM fuses. In the information censorship state, CENSOR[0:1] must be cleared to the cleared censorship state before CENSOR[0:1] can be put into the no-censorship state.

CAUTION

Clearing the CENSOR[0:1] bits causes the entire CMF array to be erased.

19.8.4 Setting and Clearing Censor

The value of each bit in CENSOR[0:1] is determined by the state of two NVM bits in a special NVM fuse as shown in **Table 19-17**. These two NVM bits are not part of the main flash array. The NVM fuse is not writable but instead may be set or cleared. The two NVM bits in the NVM fuse are programmed and erased simultaneously to change

the value of the NVM fuse. Reading CENSOR[0:1] while setting or clearing with the high voltage applied (CSC = 1 and EHV = 1) will return zeroes.

Table 19-17 NVM Fuse States

NVM bit 0	NVM bit 1	NVM Fuse Bit Value
Erased	Erased	Undefined
Programmed	Erased	Set (1)
Erased	Programmed	Cleared (0)
Programmed	Programmed	Undefined

The set operation changes the state in an NVM fuse from a zero to a one by programming NVM bit 0 and erasing NVM bit 1 simultaneously in the NVM fuse. This set operation can be performed without changing the contents of the CMF array.

To set one or both of the bits in CENSOR[0:1],

1. Using section **19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM**, write the pulse width timing control fields for an erase pulse, CSC = 1, PE = 0 and SES = 1 in the CMFCTL register.
2. Write a one to the CENSOR bit(s) to be set.
3. Write EHV = 1 in the CMFCTL register. This will apply the programming voltages to NVM bit 0 and the erase voltages to NVM bit 1 simultaneously.
4. Read the CMFCTL register until HVS = 0.
5. Write EHV = 0 in the CMFCTL register.
6. Read the CMFMCR CENSOR bit(s) that are being set. If any bit selected for set is a 0 go to step 3.
7. Write SES = 0 and CSC = 0.

The clear operation changes the state in an NVM fuse from a one to a zero by erasing NVM bit 0 and programming NVM bit 1 simultaneously in the NVM fuse. This clear operation can be done only while erasing the entire CMF array and shadow information. To clear CENSOR[0:1],

1. Write PROTECT[0:7] = 0x00 to enable the entire array for erasure.
2. Using section **19.7.6 A Technique to Determine SCLKR, CLKPE, and CLKPM**, write the pulse width timing control fields for an erase pulse, BLOCK[0:7] = 0xFF, CSC = 1, PE = 1 and SES = 1 in the CMFCTL register.
3. Perform an erase interlock write.
4. Write EHV = 1 in the CMFCTL register. This will apply the erase voltages to the entire CMF array and NVM bit 0 and the programming voltages to NVM bit 1 simultaneously.
5. Read the CMFCTL register until HVS = 0.
6. Write EHV = 0 in the CMFCTL register.
7. Read the entire CMF array and the shadow information words. If any bit equals zero, go to step 4.
8. Read CENSOR[0:1]. If CENSOR[0:1] ≠ 0 go to step 4.
9. Write SES = 0 and CSC = 0.

19.8.5 Switching the CMF EEPROM Censorship

There are three levels of censorship that CENSOR[0:1] can select: cleared censorship, no censorship (two states) and information censorship. These three levels, state values, transitions and level of censorship are shown in **Figure 19-6**.

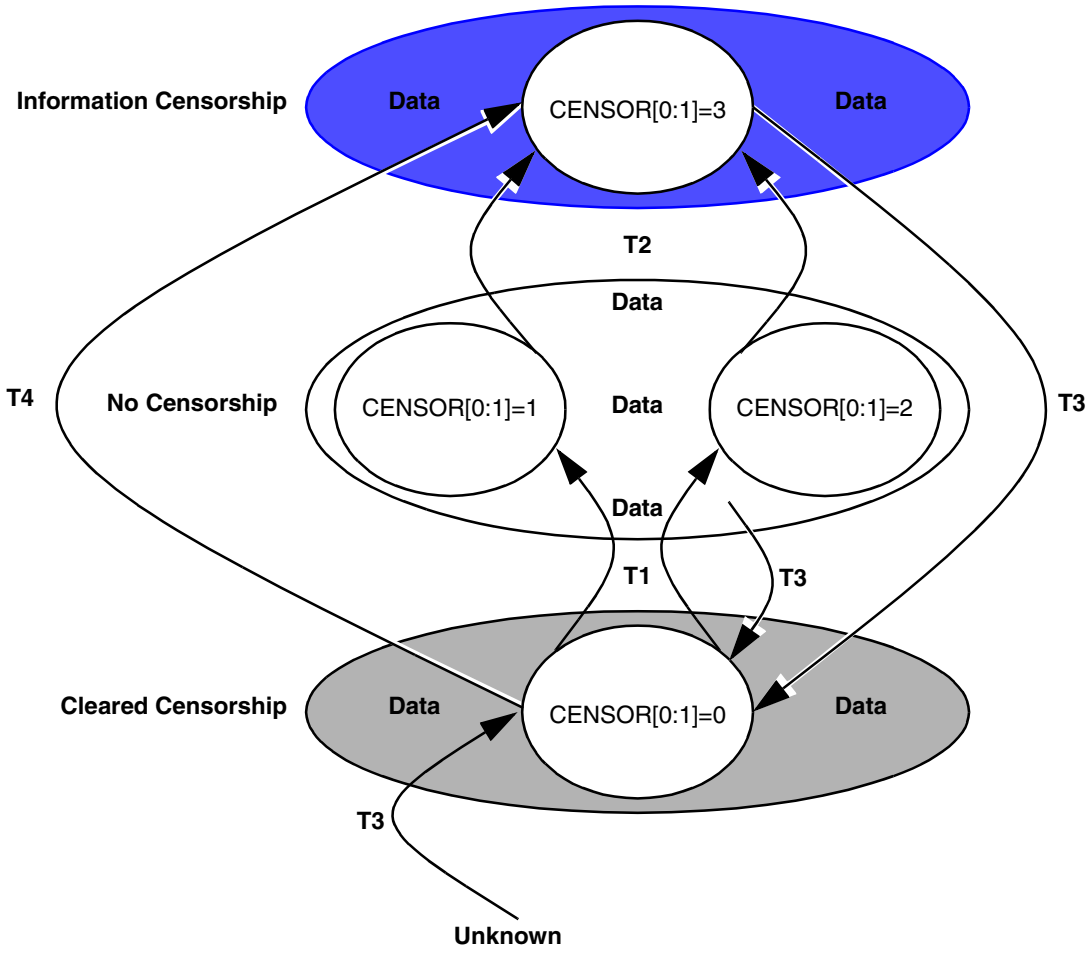


Figure 19-6 Censorship States and Transitions

Figure 19-6 illustrates the following CENSOR[0:1] transitions:

- T1: Cleared censorship to no censorship
  Set CENSOR[0] or CENSOR[1].
- T2: No censorship to information censorship
  Set CENSOR[0] and CENSOR[1].
- T3: Information censorship, no censorship or unknown to cleared censorship
  clear CENSOR[0:1]. This is done only while the entire CMF array is erased.
- T4: Cleared censorship to information censorship
  Set both CENSOR[0] and CENSOR[1].

19.9 Pin Descriptions

The CMF modules use the following pins:

- E_{PEE}
- V_{PP}
- V_{DDF}
- V_{SSF}

19.9.1 E_{PEE} Signal

The EPEE bit monitors the state of the external program/erase enable, E_{PEE} pin. E_{PEE} has a digital filter that requires two consecutive samples to be equal before the output of the filter changes. The CMF samples E_{PEE} when EHV is asserted and holds the E_{PEE} state until EHV is negated. This is shown in **Figure 19-7**.

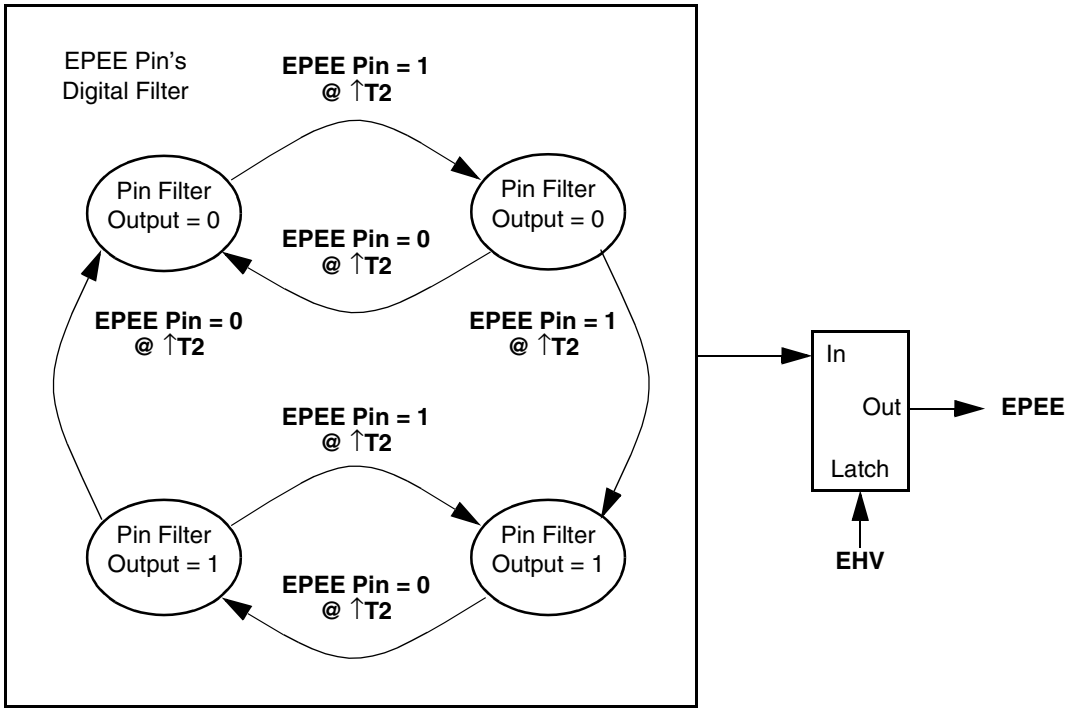


Figure 19-7 EPEE Digital Filter and Latch

If EPEE = 1 when EHV is asserted, high voltage operations such as program or erase are enabled. If EPEE = 0 when EHV is asserted, high voltage operations are disabled.

The EPEE pin uses a two-period clock synchronizer that switches the internal signal CMF_EPEE after two consecutive constant values of EPEE, as shown in **Figure 19-8**. At first the value of CMF_EPEE is unknown, as the prior information for the EPEE pin is not provided. One high or low clock of EPEE does not cause CMF_EPEE to switch.

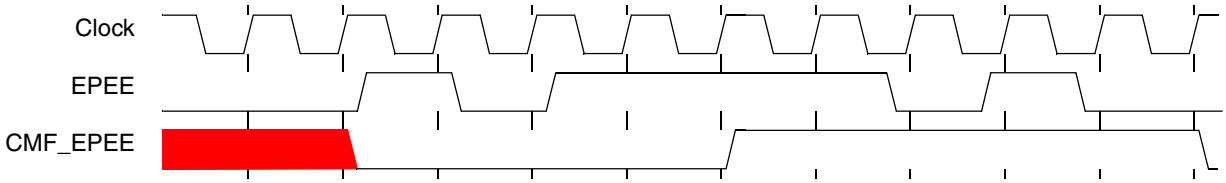
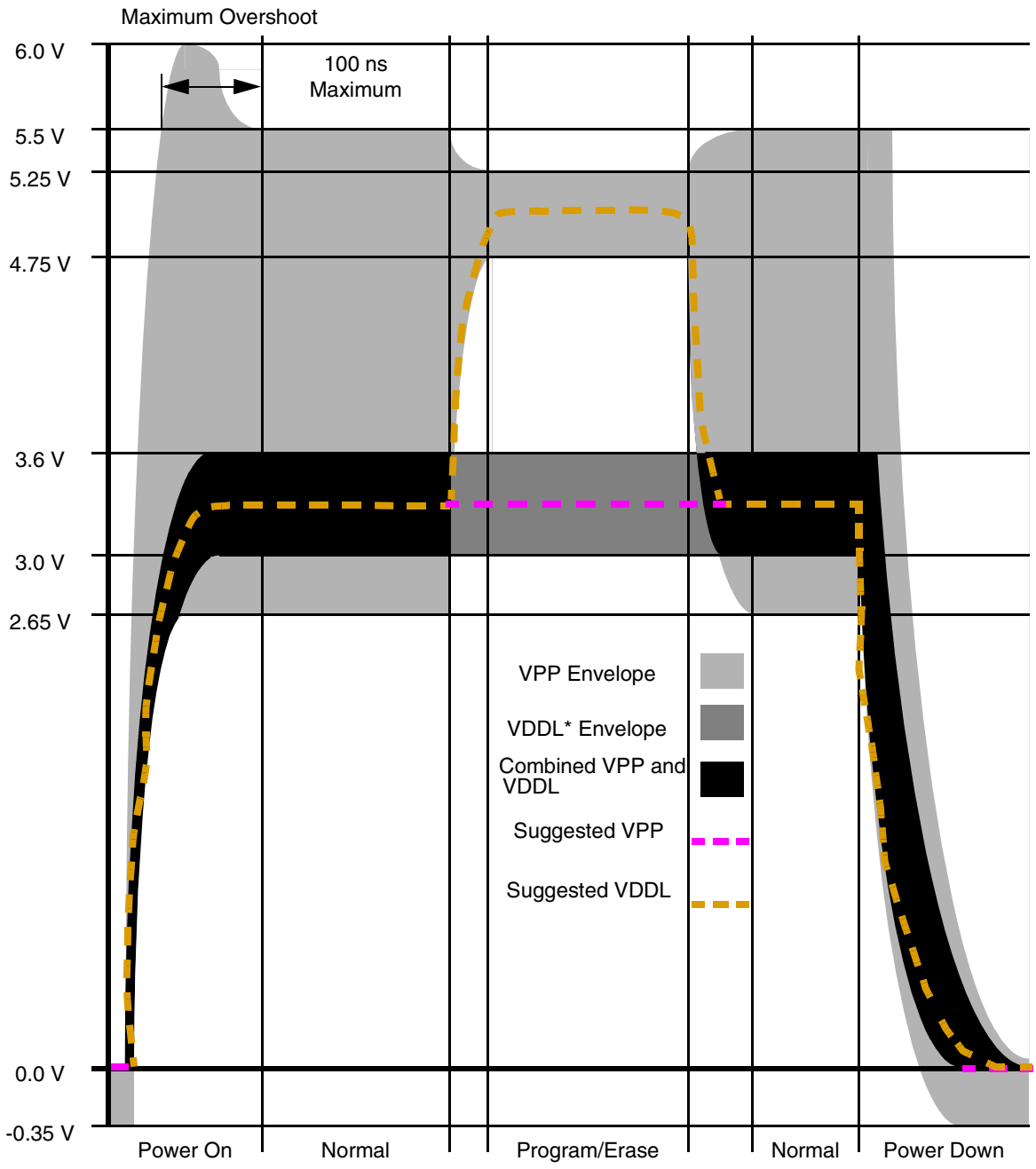


Figure 19-8 CMF_EPEE Timing Diagram

19.9.2 FLASH Program/Erase Voltage Conditioning

A voltage of at least (VDDL – 0.35 V) must be applied at all times to the VPP pins or damage to the FLASH module can occur. FLASH modules can be damaged by power on and power off VPP transients. VPP must not rise to programming level while VDDL is below 1.0 volts, and must not fall below the minimum specified value while VDDL is applied. **Figure 19-9** shows the VPP and VDDL operating envelope.

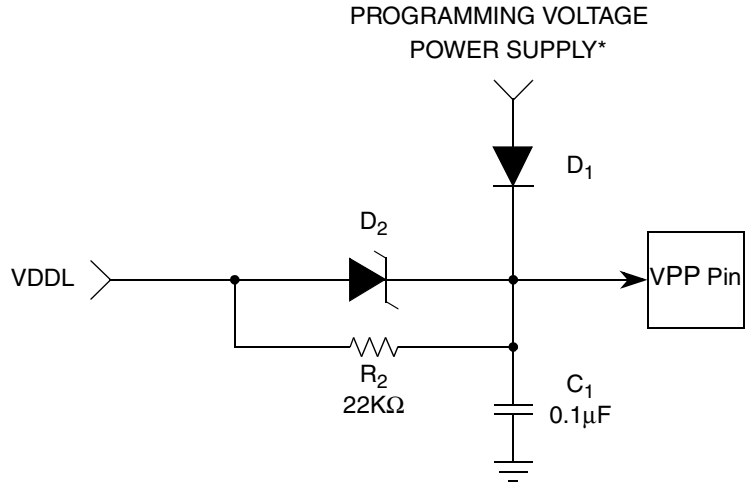


*This assumes that VDDL = VDDI = VDDF = VDDSYN.

Figure 19-9 VPP and VDDL Power Switching

Use of an external circuit to condition VPP is recommended. **Figure 19-10** shows a simple circuit that maintains required voltages and filters transients. VPP is pulled up to VDDL via Schottky diode D2, protecting VDDL from excessive reverse current. D2 also protects the FLASH from damage should the programming voltage go to zero. Programming power supply voltage must be adjusted to compensate for the forward-

bias drop across D1. Capacitor C1 filters transients, while R2 provides a discharge bleed path for C1. Allow for RC charge and discharge time constants when applying and removing power. When using this circuit, keep leakage from external devices connected to the VPP pin low, to minimize diode voltage drop.



*The VPP voltage specification is the voltage at the VPP pin, not the input to diode D1.

Figure 19-10 VPP Conditioning Circuit

19.10 Reset Operation

19.10.1 Master Reset

The MPC555 / MPC556 signals a master reset (both $\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$) to the CMF EEPROM when a full reset is required. A master reset is the highest priority operation for the CMF EEPROM and will terminate all other operations. The CMF EEPROM module uses master reset to initialize all register bits to their reset values. If the CMF EEPROM is in program or erase operation (EHV = 1) and a master reset is generated, the module will perform the needed interlocks to disable the high voltage without damage to the high voltage circuits. Master reset will terminate any other mode of operation and force the CMF EEPROM BIU to a state ready to receive U-bus accesses within 10 clocks of the end of master reset.

If the $\overline{\text{HC}}$ bit of the reset configuration word = 0 and the SIU requests internal configuration during reset, the CMF EEPROM will provide the reset configuration word to the device from CMFRC.

19.10.2 Soft Reset

A soft reset forces the BIU into a state ready to receive U-bus accesses and clear the EHV bit. All other register bits remain unaltered by a soft reset.

19.10.3 Emulation Operation

The CMF EEPROM supports externally mapped access for emulation operation. When the SIU indicates an externally mapped access to the CMF EEPROM, the CMF does not respond to the address, even though it may be a valid CMF access. Refer to **10.6 Dual Mapping of the Internal Flash EEPROM Array** for details.

19.11 Disabling the CMF Module

The CMF EEPROM can be disabled when the internal memories are disabled. Disabling the internal memories is controlled by the FLEN bit (bit 20) in the USIU internal memory map register. The default reset enable/disable state of the internal memories is user defined with the reset configuration word bit 20.

CAUTION

The reset configuration word from an erased CMF must be generated external to the CMF, i.e., from the default reset configuration word off the external reset configuration word. See **7.5 Reset Configuration**.

EHV is reset to 0 when the CMF is disabled and can not be set until the CMF is enabled, see section **19.7.8 Controlling the Program/Erase Voltage** When disabled, the power used by the CMF is reduced.

NOTE

Although the program and erase operations can be suspended (EHV = 0) by disabling the internal memory, it is not recommended that program or erase be suspended in this manner.

SECTION 20
STATIC RANDOM ACCESS MEMORY (SRAM)

The MPC555 / MPC556 contains two static random access memory (SRAM) modules: a 16-Kbyte module and a 10-Kbyte module. The SRAM modules provide the micro-controller unit (MCU) with fast (one cycle access), general-purpose memory. The SRAM can be read or written as either bytes, half words, or words.

Each SRAM module is built with a series of 4-Kbyte blocks and occupies a continuous block of memory. For a RAM size array block of less than 4 Kbytes (e.g., the 2-Kbyte array in the 10-Kbyte SRAM module), the remaining 2 Kbytes are unimplemented and unusable.

The SRAM modules are accessible to the CPU and other bus masters via the L-bus on the CPU chip. To improve access time, each SRAM module resides on a separate bus interface unit (BIU). Each BIU has its own module control register.

20.1 Features

- One-cycle access
- Byte, half-word, or word read/write accesses
- Individual protection control bits provided for 4-Kbyte block boundaries
  — read only region
  — data only region
  — user/supervisor
- Two-cycle access for power savings
- Low power standby operation for data retention
  — V_{DDI} = 0, no read/writes to the SRAM; VDDSRAM = 3.3 V to retain data
- Supports pipelining

20.2 Block Diagram

Figure 20-1 shows the major components of an SRAM module.

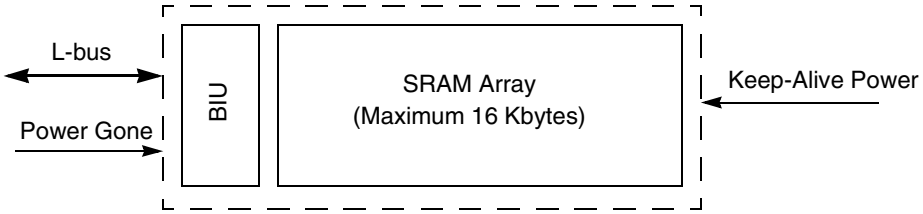


Figure 20-1 SRAM Block Diagram

20.3 Programming Model

The SRAM modules consist of two separately addressable sections the array itself, and a set of registers used for configuration and testing of the SRAM array. The registers are located in the SRAM control register block, shown in **Figure 20-2**. See also **Figure 1-3** for the entire MPC555 / MPC556 memory map.

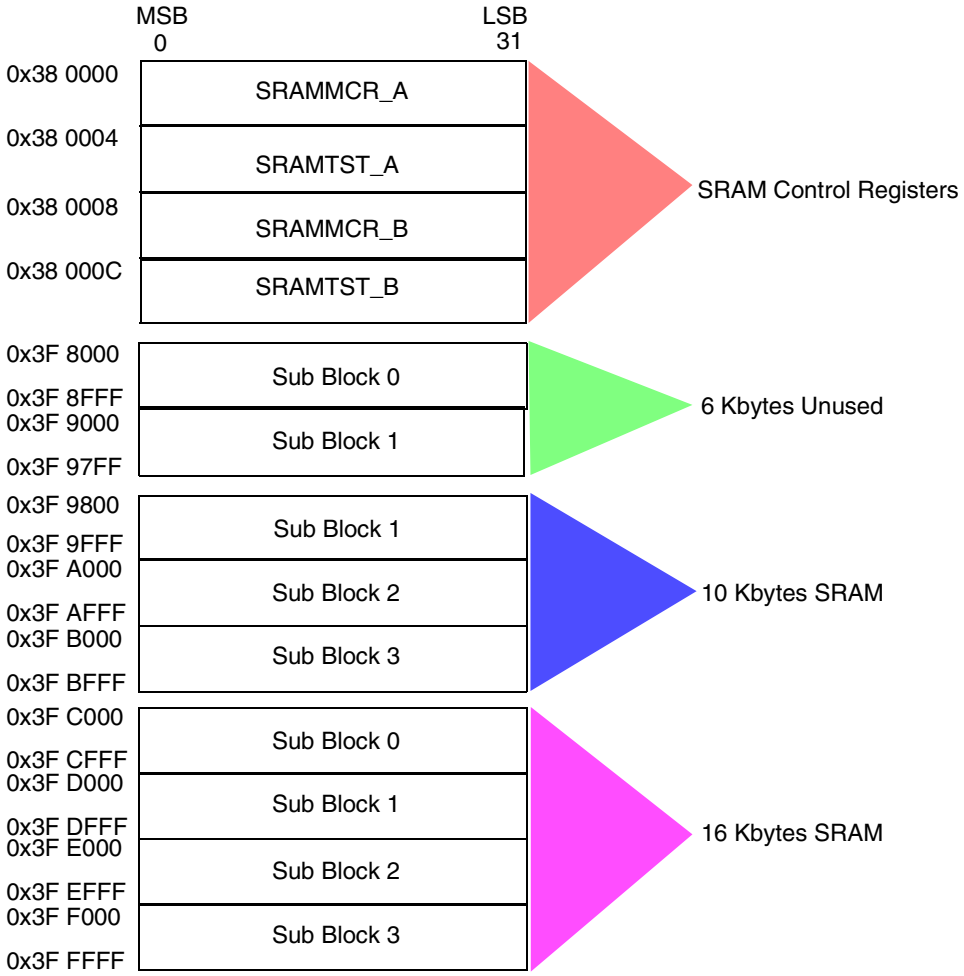


Figure 20-2 SRAM Memory Map

The control block for each of the two SRAM modules contains one control register for configuring the array and one control register for use in testing.

20.3.1 SRAM Module Configuration Register (SRAMMCR)

Each SRAM module configuration register contains bits for setting access rights to the array. **Table 20-1** provides definitions for the bits.

SRAMMCR — SRAM Module Configuration Register **0x38 0000**
0x38 0008

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LCK	DIS	2CY	RESERVED												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				R0	D0	S0	R1	D1	S1	R2	D2	S2	R3	D3	S3
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-1 SRAMMCR Bit Descriptions

Bit(s)	Name	Description
0	LCK	Lock bit. This bit can be set only once and cleared only by reset. 0 = Writes to the SRAMMCR are accepted 1 = Writes to the SRAMMCR are ignored
1	DIS	Module disable 0 = SRAM module is enabled 1 = SRAM module is disabled. Module can be subsequently re-enabled by software setting this bit or by reset. Attempts to read SRAM array when it is disabled result in internal $\overline{\text{TEA}}$ assertion.
2	2CY	Two-cycle mode 0 = SRAM module is in single-cycle mode (normal operation) 1 = SRAM module is in two-cycle mode. In this mode, the first cycle is used for decoding the address, and the second cycle is used for accepting or providing data. This mode provides some power savings while keeping the memory active.
3:19	—	Reserved
20, 23, 26, 29	Rx (x = 0, 1, 2, 3)	Read only. R0 controls the highest 4-Kbyte block (lowest address) of the SRAM array; R3 controls the lowest block (highest address). 0 = 4-Kbyte block is readable and writable 1 = 4-Kbyte block is read only. Attempts to write to this space result in internal $\overline{\text{TEA}}$ assertion.
21, 24, 27, 30	Dx (x = 0, 1, 2, 3)	Data only. D0 controls the highest 4-Kbyte block (lowest address) of the SRAM array; D3 controls the lowest block (highest address). 0 = 4-Kbyte block can contain data or instructions 1 = 4-Kbyte block contains data only. Attempts to load instructions from this space result in internal $\overline{\text{TEA}}$ assertion.
22, 25, 28, 31	Sx (x = 0, 1, 2, 3)	Supervisor only. S0 controls the highest 4-Kbyte block (lowest address) of the SRAM array; S3 controls the lowest block (highest address). 0 = 4-Kbyte block is placed in unrestricted space 1 = 4-Kbyte block is placed in supervisor space. Attempts to access this space from the user privilege level result in internal $\overline{\text{TEA}}$ assertion.

20.3.2 SRAM Test Register (SRAMTST)

SRAMTST — SRAM Test Register **0x38 0004, 0x38 000C**

The SRAM test register is used for factory testing only.

SECTION 21
DEVELOPMENT SUPPORT

21.1 Overview

The visibility and controllability requirements of emulators and bus analyzers are in opposition to the trend of modern microcomputers and microprocessors where many bus cycles are directed to internal resources and are not visible externally.

In order to enhance the development tool visibility and controllability, some of the development support functions are implemented in silicon. These functions include program flow tracking, internal watchpoint, breakpoint generation, and emulation while in debug mode.

This section covers program flow tracking support, breakpoint/watchpoint support, development system interface support (debug mode) and software monitor debugger support. These features allow the user to efficiently debug systems based on the MPC555 / MPC556.

21.2 Program Flow Tracking

The mechanism described below allows tracking of program instruction flow with almost no performance degradation. The information provided may be compressed and captured externally and then parsed by a post-processing program using the microarchitecture defined below.

The program instructions flow is visible on the external bus when the MPC555 / MPC556 is programmed to operate in serial mode and show all fetch cycles on the external bus. This mode is selected by programming the ISCT_SER (instruction fetch show cycle control) field in the I-bus support control register (ICTRL), as shown in **Table 21-21**. In this mode, the processor is fetch serialized, and all internal fetch cycles appear on the external bus. Processor performance is, therefore, much lower than when working in regular mode.

These features, together with the fact that most fetch cycles are performed internally (e.g., from the I-cache), increase performance but make it very difficult to provide the user with the real program trace.

In order to reconstruct a program trace, the program code and the following additional information from the MCU are needed:

- A description of the last fetched instruction (stall, sequential, branch not taken, branch direct taken, branch indirect taken, exception taken)
- The addresses of the targets of all indirect flow change. Indirect flow changes include all branches using the link and count registers as the target address, all exceptions, and **rfi**, **mtmsr and mtspr** (to some registers) because they may cause a context switch.

- The number of instructions canceled each clock

Instructions are fetched sequentially until branches (direct or indirect) or exceptions appear in the program flow or some stall in execution causes the machine not to fetch the next address. Instructions may be architecturally executed, or they may be canceled in some stage of the machine pipeline.

The following sections define how this information is generated and how it should be used to reconstruct the program trace. The issue of data compression that could reduce the amount of memory needed by the debug system is also mentioned.

21.2.1 Program Trace Cycle

To allow visibility of the events happening in the machine a few dedicated pins are used and a special bus cycle attribute, program trace cycle, is defined.

The program trace cycle attribute is attached to all fetch cycles resulting from indirect flow changes. When program trace recording is needed, the user can make sure these cycles are visible on the external bus.

The VSYNC indication, when asserted, forces all fetch cycles marked with the program trace cycle attribute to be visible on the external bus even if their data is found in one of the internal devices. To enable the external hardware to properly synchronize with the internal activity of the CPU, the assertion and negation of VSYNC forces the machine to synchronize. The first fetch after this synchronization is marked as a program trace cycle and is visible on the external bus. For more information on the activity of the external hardware during program trace refer to **21.2.4 The External Hardware**.

In order to keep the pin count of the chip as low as possible, VSYNC is not implemented as one of the chip's external pins. It is asserted and negated using the serial interface implemented in the development port. For more information on this interface refer to **21.5 Development Port**

Forcing the CPU to show all fetch cycles marked with the program trace cycle attribute can be done either by asserting the VSYNC pin (as mentioned above) or by programming the fetch show cycle bits in the instruction support control register, ICTRL. For more information refer to **21.2.5 Instruction Fetch Show Cycle Control**

When the VSYNC indication is asserted, all fetch cycles marked with the program trace cycle attribute are made visible on the external bus. These cycles can generate regular bus cycles (address phase and data phase) when the instructions reside only in one of the external devices. Or, they can generate address-only cycles when the instructions reside in one of the internal devices (internal memory, etc.).

When VSYNC is asserted, some performance degradation is expected due to the additional external bus cycles. However, since this performance degradation is expected to be very small, it is possible to program the machine to *show all indirect flow changes*. In this way, the machine will always perform the additional external bus cycles and maintain exactly the same behavior both when VSYNC is asserted and when it is negated. For more information refer to **21.7.6 I-Bus Support Control Register**.

The status pins are divided into two groups and one special case listed below:

21.2.1.1 Instruction Queue Status Pins — VF [0:2]

Instruction queue status pins denote the type of the last fetched instruction or how many instructions were flushed from the instruction queue. These status pins are used for both functions because queue flushes only happen in clocks that there is no fetch type information to be reported.

Possible instruction types are defined in **Table 21-1**.

Table 21-1 VF Pins Instruction Encodings

VF[0:2]	Instruction Type	VF Next Clock Will Hold
000	None	More instruction type information
001	Sequential	More instruction type information
010	Branch (direct or indirect) **not** taken	More instruction type information
011	VSYNC was asserted/negated and therefore the next instruction will be marked with the indirect change-of-flow attribute	More instruction type information
100	Exception taken — the target will be marked with the indirect change-of-flow attribute	Queue flush information[1]
101	Branch indirect taken, **rfi**, **mtmsr**, **isync** and in some cases **mtspr** to CMPA-F, ICTRL, ECR, or DER — the target will be marked with the indirect change-of-flow attribute[2]	Queue flush information[1]
110	Branch direct taken	Queue flush information[1]
111	Branch (direct or indirect) **not** taken	Queue flush information[1]

NOTES:
1. Unless next clock VF=111. See below.
2. The sequential instructions listed here affect the machine in a manner similar to indirect branch instructions. Refer to **21.2.3 Sequential Instructions Marked as Indirect Branch**.

Table 21-2 shows VF[0:2] encodings for instruction queue flush information.

Table 21-2 VF Pins Queue Flush Encodings

VF[0:2]	Queue Flush Information
000	0 instructions flushed from instruction queue
001	1 instruction flushed from instruction queue
010	2 instructions flushed from instruction queue
011	3 instructions flushed from instruction queue
100	4 instructions flushed from instruction queue
101	5 instructions flushed from instruction queue
110	Reserved
111	Instruction type information[1]

NOTES:
1. Refer to **Table 21-1**.

21.2.1.2 History Buffer Flushes Status Pins— VFLS [0..1]

The history buffer flushes status pins denote how many instructions are flushed from the history buffer this clock due to an exception.**Table 21-3** shows VFLS encodings.

Table 21-3 VFLS Pin Encodings

VFLS[0:1]	History Buffer Flush Information
00	0 instructions flushed from history queue
01	1 instruction flushed from history queue
10	2 instructions flushed from history queue
11	Used for debug mode indication (FREEZE). Program trace external hardware should ignore this setting.

21.2.1.3 Queue Flush Information Special Case

There is one special case when although queue flush information is expected on the VF pins, (according to the last value on the VF pins), regular instruction type information is reported. The only instruction type information that can appear in this case is VF = 111, branch (direct or indirect) NOT taken. Since the maximum queue flushes possible is five, it is easy to identify this special case.

21.2.2 Program Trace when in Debug Mode

When entering debug mode an *interrupt/exception taken* is reported on the VF pins, (VF = 100) and a cycle marked with the program trace cycle is made visible externally.

When the CPU is in debug mode, the VF pins equal ‘000’ and the VFLS pins equal ‘11’. For more information on debug mode refer to **21.4 Development System Interface**

If VSYNC is asserted/negated while the CPU is in debug mode, this information is reported as the first VF pins report when the CPU returns to regular mode. If VSYNC was not changed while in debug mode. the first VF pins report will be of an indirect branch taken (VF = 101), suitable for the **rfi** instruction that is being issued. In both

cases the first instruction fetch after debug mode is marked with the program trace cycle attribute and therefore is visible externally.

21.2.3 Sequential Instructions Marked as Indirect Branch

There are cases when non-branch (sequential) instructions may effect the machine in a manner similar to indirect branch instructions. These instructions include **rfi**, **mtmsr**, **isync** and **mtspr** to CMPA-F, ICTRL, ECR and DER.

These instructions are marked by the CPU as indirect branch instructions (VF = 101) and the following instruction address is marked with the same program trace cycle attribute as if it were an indirect branch target. Therefore, when one of these special instructions is detected in the CPU, the address of the following instruction is visible externally. In this way the reconstructing software is able to evaluate correctly the effect of these instructions.

21.2.4 The External Hardware

When program trace is needed, the external hardware needs to sample the status pins (VF and VFLS) each clock cycle and the address of all cycles marked with the program trace cycle attribute.

Program trace can be used in various ways. Below are two examples of how program trace can be used:

- **Back trace —** Back trace is useful when a record of the program trace *before* some event occurred is needed. An example of such an event is some system failure.
  In case back trace is needed the external hardware should start sampling the status pins (VF and VFLS) and the address of all cycles marked with the program trace cycle attribute immediately when reset is negated. If *show cycles* is programmed out of reset to *show all,* all cycles marked with program trace cycle attribute are visible on the external bus. VSYNC should be asserted sometime after reset and negated when the programmed event occurs. If *no show* is programmed for *show cycles*, make sure VSYNC is asserted before the Instruction show cycles programming is changed from *show all*.
  Note that in case the timing of the programmed event is unknown it is possible to use cyclic buffers.
  After VSYNC is negated the trace buffer will contain the program flow trace of the program executed before the programmed event occurred.
- **Window trace —** Window trace is useful when a record of the program trace between two events is needed. In case window trace is needed the VSYNC pin should be asserted between these two events.
  After the VSYNC pin is negated the trace buffer will contain information describing the program trace of the program executed *between* the two events.

21.2.4.1 Synchronizing the Trace Window to the CPU Internal Events

The assertion/negation of VSYNC is done using the serial interface implemented in the development port. In order to synchronize the assertion/negation of VSYNC to an in-

ternal event of the CPU, it is possible to use the internal breakpoints together with debug mode. This method is available only when debug mode is enabled. For more information on debug mode refer to **21.4 Development System Interface**

The following is an example of steps that enable the user to synchronize the trace window to the CPU internal events:

1. Enter debug mode, either immediately out of reset or using the debug mode request
2. Program the hardware to break on the event that marks the start of the trace window using the control registers defined in **21.3 Watchpoints and Breakpoints Support**
3. Enable debug mode entry for the programmed breakpoint in the debug enable register (DER). See **21.7.12 Debug Enable Register (DER)**)
4. Return to the regular code run (see **21.4.1.6 Exiting Debug Mode**)
5. The hardware generates a breakpoint when the programmed event is detected and the machine enters debug mode (see **21.4.1.2 Entering Debug Mode**)
6. Program the hardware to break on the event that marks the end of the trace window
7. Assert VSYNC
8. Return to the regular code run. The first report on the VF pins is a VSYNC (VF = 011).
9. The external hardware starts sampling the program trace information upon the report on the VF pins of VSYNC
10. The hardware generates a breakpoint when the programmed event is detected and the machine enters debug mode
11. Negate VSYNC
12. Return to the regular code run (issue an **rfi**). The first report on the VF pins is a VSYNC (VF = 011)
13. The external hardware stops sampling the program trace information upon the report on the VF pins of VSYNC

21.2.4.2 Detecting the Trace Window Start Address

When using *back trace*, latching the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle, should start immediately after the negation of reset. The start address is the first address in the program trace cycle buffer.

When using *window trace*, latching the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle, should start immediately after the first VSYNC is reported on the VF pins. The start address of the trace window should be calculated according to first two VF pins reports.

Assuming that VF1 and VF2 are the two first VF pins reports and T1 and T2 are the two addresses of the first two cycles marked with the program trace cycle attribute that were latched in the trace buffer, use the following table to calculate the trace window start address.

Table 21-4 Detecting the Trace Buffer Start Point

VF1	VF2	Starting point	Description
011 **VSYNC**	001 sequential	T1	**VSYNC** asserted followed by a sequential instruction. The start address is T1
011 **VSYNC**	110 branch direct taken	T1 - 4 + offset (T1 - 4)	**VSYNC** asserted followed by a taken direct branch. The start address is the target of the direct branch
011 **VSYNC**	101 branch indirect taken	T2	**VSYNC** asserted followed by a taken indirect branch. The start address is the target of the indirect branch

21.2.4.3 Detecting the Assertion/Negation of VSYNC

Since the VF pins are used for reporting both instruction type information and queue flush information, the external hardware must take special care when trying to detect the assertion/negation of VSYNC. When VF = 011 it is a VSYNC assertion/negation report only if the previous VF pins value was one of the following values: 000, 001, or 010.

21.2.4.4 Detecting the Trace Window End Address

The information on the status pins that describes the last fetched instruction and the last queue/history buffer flushes, changes every clock. Cycles marked as program trace cycle are generated on the external bus only when possible (when the SIU wins the arbitration over the external bus). Therefore, there is some delay between the information reported on the status pins that a cycle marked as program trace cycle will be performed on the external bus and the actual time that this cycle can be detected on the external bus.

When VSYNC is negated by the user (through the serial interface of the development port), the CPU delays the report of the of the assertion/negation of VSYNC on the VF pins (VF = 011) until all addresses marked with the program trace cycle attribute were visible externally. Therefore, the external hardware should stop sampling the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle immediately after the VSYNC report on the VF pins.

The last two instructions reported on the VF pins are not always valid. Therefore at the last stage of the reconstruction software, the last two instructions should be ignored.

21.2.4.5 Compress

In order to store all the information generated on the pins during program trace (five bits per clock + 30 bits per show cycle) a large memory buffer may be needed. However, since this information includes events that were canceled, compression can be very effective. External hardware can be added to eliminate all canceled instructions and report only on branches (taken and not taken), indirect flow change, and the number of sequential instructions after the last flow change.

21.2.5 Instruction Fetch Show Cycle Control

Instruction fetch show cycles are controlled by the bits in the ICTRL and the state of VSYNC. The following table defines the level of fetch show cycles generated by the CPU. For information on the fetch show cycles control bits refer to **Table 21-5**

Table 21-5 Fetch Show Cycles Control

VSYNC	ISCTL Instruction Fetch Show Cycle Control Bits ISCTRL[ISCT_SER]	Show Cycles Generated
X	x00	All fetch cycles
X	x01	All change of flow (direct & indirect)
X	x10	All indirect change of flow
0	x11	No show cycles are performed
1	x11	All indirect change of flow

NOTE

A cycle marked with the program trace cycle attribute is generated for any change in the VSYNC state (assertion or negation).

21.3 Watchpoints and Breakpoints Support

Watchpoints, when detected, are reported to the external world on dedicated pins but do not change the timing and the flow of the machine. Breakpoints, when detected, force the machine to branch to the appropriate exception handler. The CPU supports internal watchpoints, internal breakpoints, and external breakpoints.

Internal watchpoints are generated when a user programmable set of conditions are met. Internal breakpoints can be programmed to be generated either as an immediate result of the assertion of one of the internal watchpoints, or after an internal watchpoint is asserted for a user programmable times. Programming a certain internal watchpoint to generate an internal breakpoint can be done either in software, by setting the corresponding software trap enable bit, or on the fly using the serial interface implemented in the development port to set the corresponding development port trap enable bit.

External breakpoints can be generated by any of the peripherals of the system, including those found on the MPC555 / MPC556 or externally, and also by an external development system. Peripherals found on the external bus use the serial interface of the development port to assert the external breakpoint.

In the CPU, as in other RISC processors, saving/restoring machine state on the stack during exception handling, is done mostly in software. When the software is in the middle of saving/restoring machine state, the MSR[RI] bit is cleared. Exceptions that occur and that are handled by the CPU when the MSR[RI] bit is clear result in a non-restartable machine state. For more information refer to **3.15.4 Interrupts**

In general, breakpoints are recognized in the CPU is only when the MSR[RI] bit is set, which guarantees machine restartability after a breakpoint. In this working mode breakpoints are said to be *masked*. There are cases when it is desired to enable

breakpoints even when the MSR[RI] bit is clear, with the possible risk of causing a non-restartable machine state. Therefore internal breakpoints have also a programmable *non-masked* mode, and an external development system can also choose to assert a *non-maskable* external breakpoint.

Watchpoints are not *masked* and therefore always reported on the external pins, regardless of the value of the MSR[RI] bit. The counters, although counting watchpoints, are part of the internal breakpoints logic and therefore are not decremented when the CPU is operating in the masked mode and the MSR[RI] bit is clear.

The following figure illustrates the watchpoints and breakpoints support of the CPU.

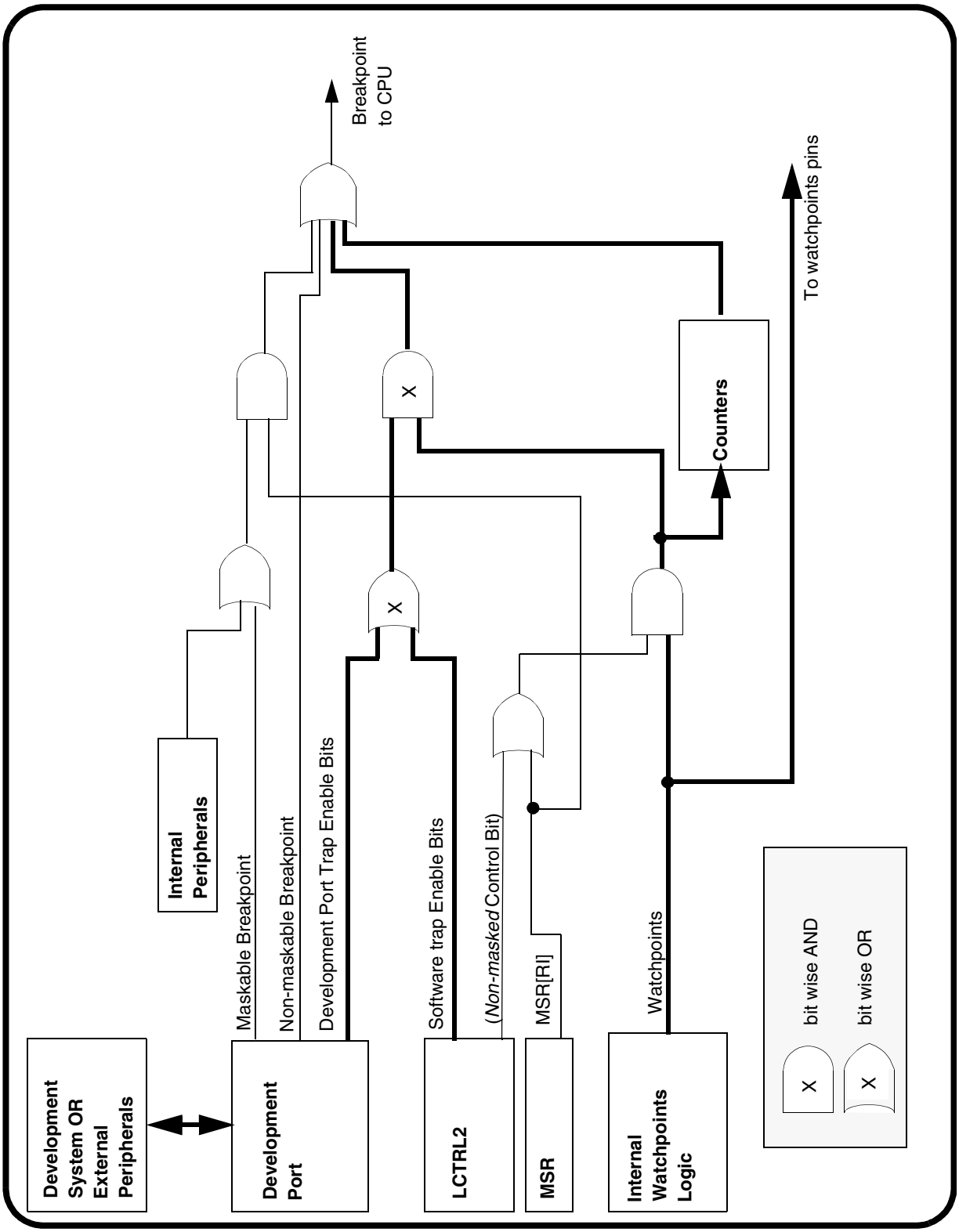


Figure 21-1 Watchpoints and Breakpoint Support in the CPU

21.3.1 Internal Watchpoints and Breakpoints

This section describes the internal breakpoints and watchpoints support of the CPU. For information on external breakpoints support refer to **21.4 Development System Interface**

Internal breakpoint and watchpoint support is based on eight comparators comparing information on instruction and load/store cycles, two counters, and two AND-OR logic structures. The comparators perform compare on the Instruction address (I-address), on the load/store address (L-address) and on the load/store data (L-data).

The comparators are able to detect the following conditions: equal, not equal, greater than, less than (greater than or equal and less than or equal are easily obtained from these four conditions, for more information refer to **21.3.1.6 Generating Six Compare Types**). Using the AND-OR logic structures “in range” and “out of range” detections (on address and on data) are supported. Using the counters, it is possible to program a breakpoint to be recognized after an event was detected a predefined number of times.

The L-data comparators can operate on fix point data of load or store. When operating on fix point data the L-data comparators are able to perform compare on bytes, half-words and words and can treat numbers either as signed or as unsigned values.

The comparators generate match events. The match events enter the instruction AND-OR logic where the instruction watchpoints and breakpoint are generated. The instruction watchpoints, when asserted, may generate the instruction breakpoint. Two of them may decrement one of the counters. If one of the instruction watchpoints expires in a counter that is counting, the instruction breakpoint is asserted.

The instruction watchpoints and the load/store match events (address and data) enter the load/store AND-OR logic where the load/store watchpoints and breakpoint are generated. The load/store watchpoints, when asserted, may generate the load/store breakpoint or they may decrement one of the counters. When a counter that is counting one of the load/store watchpoints expires, the load/store breakpoint is asserted.

Watchpoints progress in the machine and are reported on retirement. Internal breakpoints progress in the machine until they reach the top of the history buffer when the machine branches to the breakpoint exception routine.

In order to enable the user to use the breakpoint features without adding restrictions on the software, the address of the load/store cycle that generated the load/store breakpoint is not stored in the DAR (data address register), like other load/store type exceptions. In case of a load/store breakpoint, the address of the load/store cycle that generated the breakpoint is stored in an implementation-dependent register called the BAR (breakpoint address register).

Key features of internal watchpoint and breakpoint support are:

- Four I-address comparators (each supports equal, not equal, greater than, less than)
- Two L-address comparators (each supports equal, not equal, greater than, less

than) including least significant bits masking according to the size of the bus cycle for the byte and half-word working modes. Refer to **21.3.1.2 Byte and Half-Word Working Modes**

- Two L-data comparators (each supports equal, not equal, greater than, less than) including byte, half-word and word operating modes and four byte mask bits for each comparator. Can be used for fix point data. Match is detected only on the valid part of the data bus (according to the cycle's size and the two address least significant bits).
- No internal breakpoint/watchpoint matching support for unaligned words and half-words
- The L-data comparators can be programmed to treat fix point numbers as signed values or as unsigned values
- Combine comparator pairs to detect in and out of range conditions (including either signed or unsigned values on the L-data)
- A programmable AND-OR logic structure between the four instruction comparators results with five outputs, four instruction watchpoints and one instruction breakpoint
- A programmable AND-OR logic structure between the four instruction watchpoints and the four load/store comparators results with three outputs, two load/store watchpoints and one load/store breakpoint
- Five watchpoint pins, three for the instruction and two for the load/store
- Two dedicated 16-bit down counters. Each can be programmed to count either an instruction watchpoint or an load/store watchpoint. Only *architecturally executed* events are counted, (count up is performed in case of recovery).
- On the fly trap enable programming of the different internal breakpoints using the serial interface of the development port (refer to **21.5 Development Port**). Software control is also available.
- Watchpoints do not change the timing of the machine
- Internal breakpoints and watchpoints are detected on the instruction during instruction fetch
- Internal breakpoints and watchpoints are detected on the load/store during load/store bus cycles
- Both instruction and load/store breakpoints and watchpoints are handled and reported on retirement. Breakpoints and watchpoints on recovered instructions (as a result of exceptions, interrupts or miss prediction) are not reported and do not change the timing of the machine.
- Instructions with instruction breakpoints are not executed. The machine branches to the breakpoint exception routine BEFORE it executes the instruction.
- Instructions with load/store breakpoints are executed. The machine branches to the breakpoint exception routine AFTER it executes the instruction. The address of the access is placed in the BAR (breakpoint address register).
- Load/store multiple and string instructions with load/store breakpoints first finish execution (all of it) and then the machine branches to the breakpoint exception routine.
- Load/store data compare is done on the load/store, AFTER swap in store accesses and BEFORE swap in load accesses (as the data appears on the bus).
- Internal breakpoints may operate either in *masked* mode or in *non-masked* mode.

- Both “go to x” and “continue” working modes are supported for the instruction breakpoints.

21.3.1.1 Restrictions

There are cases when the same watchpoint can be detected more than once during the execution of a single instruction, e.g. a load/store watchpoint is detected on more than one transfer when executing a load/store multiple/string or a load/store watchpoint is detected on more than one byte when working in byte mode. In all these cases only one watchpoint of the same type is reported for a single instruction. Similarly, only one watchpoint of the same type can be counted in the counters for a single instruction.

Since watchpoint events are reported upon the retirement of the instruction that caused the event, and more than one instruction can retire from the machine in one clock, consequent events may be reported in the same clock. Moreover the same event, if detected on more than one instruction (e.g., tight loops, range detection), in some cases will be reported only once. Note that the internal counters count correctly in these cases.

Do not put a breakpoint on an **mtspr ICTRL** instruction. When a breakpoint is set on an **mtspr ICTRL** Rx instruction and the value of bit 28 (IFM) is one, the result will be unpredictable. A breakpoint can be taken or not on the instruction and the value of the IFM bit can be either zero or one. Also, do not put a breakpoint on an mtspr ICTRL Rx instruction when Rx contains one in bit 28.

21.3.1.2 Byte and Half-Word Working Modes

The CPU watchpoints and breakpoints support enables the user to detect matches on bytes and half-words even when accessed using a load/store instruction of larger data widths, for example when loading a table of bytes using a series of load word instructions. In order to use this feature, the user needs to program the byte mask for each of the L-data comparators and to write the needed match value to the correct half-word of the data comparator when working in half-word mode and to the correct bytes of the data comparator when working in byte mode.

Since bytes and half-words can be accessed using a larger data width instruction, it is impossible for the user to predict the exact value of the L-address lines when the requested byte/half-word is accessed, (e.g., if the matched byte is byte two of the word and it is accessed using a load word instruction), the L-address value will be of the word (byte zero). Therefore, the CPU masks the two least-significant bits of the L-address comparators whenever a word access is performed and the least-significant bit whenever a half-word access is performed.

Address range is supported only when aligned according to the access size. (See examples)

21.3.1.3 Examples

- A fully supported scenario:
  Looking for:
  Data size: Byte
  Address: 0x00000003
  Data value: greater than 0x07 and less than 0x0c
  Programming options:
  One L-address comparator = 0x00000003 and program for equal
  One L-data comparator = 0x00000007 and program for greater than
  One L-data comparator = 0x0000000c and program for less than
  Both byte masks = 0xe
  Both L-data comparators program to byte mode
  Result: The event will be correctly detected regardless of the load/store instruction the compiler chooses for this access
- A fully supported scenario:
  Looking for:
  Data size: half-word
  Address: greater than 0x00000000 and less than 0x0000000c
  Data value: greater than 0x4e204e20 and less than 0x9c409c40
  Programming option:
  One L-address comparator = 0x00000000 and program for greater than
  One L-address comparator = 0x0000000c and program for less than
  One L-data comparator = 0x4e204e20 and program for greater than
  One L-data comparator = 0x9c409c40 and program for less than
  Both byte masks = 0x0
  Both L-data comparators program to half-word mode
  Result: The event will be correctly detected as long as the compiler does not use a load/store instruction with data size of byte.
- A **partially** supported scenario:
  Looking for:
  Data size: half-word
  Address: greater than or equal 0x00000002 and less than 0x0000000e
  Data value: greater than 0x4e204e20 and less than 0x9c409c40
  Programming option:
  One L-address comparator = 0x00000001 and program for greater than
  One L-address comparator = 0x0000000e and program for less than
  One L-data comparator = 0x4e204e20 and program for greater than
  One L-data comparator = 0x9c409c40 and program for less than
  Both byte masks = 0x0
  Both L-data comparators program to half-word mode or to word mode
  Result: The event will be correctly detected if the compiler chooses a load/store instruction with data size of half-word. If the compiler chooses load/store instructions with data size greater than half-word (word, multiple), there might be some false detections.

These can be ignored only by the software that handles the breakpoints. The following figure illustrates this **partially** supported scenario.

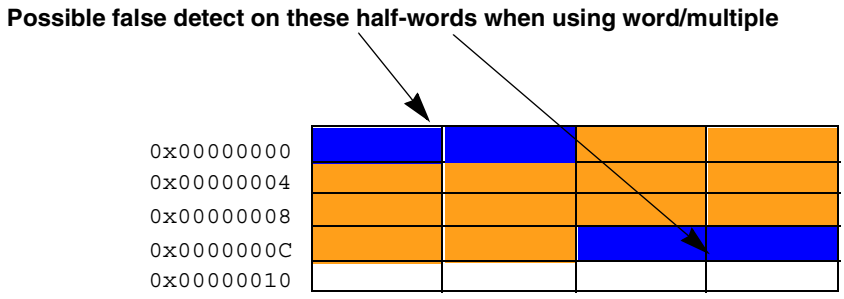


Figure 21-2 Partially Supported Watchpoint/Breakpoint Example

21.3.1.4 Context Dependent Filter

The CPU can be programmed to either recognize internal breakpoints only when the recoverable interrupt bit in the MSR is set (*masked* mode) or it can be programmed to always recognize internal breakpoints (*non-masked* mode).

When the CPU is programmed to recognize internal breakpoints only when MSRRI = 1, it is possible to debug all parts of the code except when the machine status save/ restore registers (SRR0 and SRR1), DAR (data address register) and DSISR (data storage interrupt status register) are busy and, therefore, MSRRI = 0, (in the prologues and epilogues of interrupt/exception handlers).

When the CPU is programmed always to recognize internal breakpoints, it is possible to debug all parts of the code. However, if an internal breakpoint is recognized when MSRRI = 0 (SRR0 and SRR1 are busy), the machine enters into a non-restartable state. For more information refer to **3.15.4 Interrupts**

When working in the *masked* mode, all internal breakpoints detected when MSRRI = 0 are lost. Watchpoints detected in this case are not counted by the debug counters. Watchpoints detected are always reported on the external pins, regardless of the value of the MSRRI bit.

Out of reset, the CPU is in *masked* mode. Programming the CPU to be in *non-masked* mode is done by setting the BRKNOMSK bit in the LCTRL2 register. Refer to **21.7.8 L-Bus Support Control Register 2** The BRKNOMSK bit controls all internal breakpoints (I-breakpoints and L-breakpoints).

21.3.1.5 Ignore First Match

In order to facilitate the debugger utilities “continue” and “go from x”, the ignore first match option is supported for instruction breakpoints. When an instruction breakpoint is first enabled (as a result of the first write to the instruction support control register or as a result of the assertion of the MSRRI bit when operating in the *masked* mode), the first instruction will not cause an instruction breakpoint if the ignore first match (IFM) bit in the instruction support control register (ICTRL) is set (used for “continue”).

When the IFM bit is clear, every matched instruction can cause an instruction breakpoint (used for “go from x”). This bit is set by the software and cleared by the hardware after the first instruction breakpoint match is ignored. Load/store breakpoints and all counter generated breakpoints (instruction and load/store) are not affected by this mode.

21.3.1.6 Generating Six Compare Types

Using the four compare types mentioned above (equal, not equal, greater than, less than) it is possible to generate also two more compare types: greater than or equal and less than or equal.

- Generating the greater than or equal compare type can be done by using the greater than compare type and programming the comparator to the needed value minus 1.
- Generating the less than or equal compare type can be done by using the less than compare type and programming the comparator to the needed value plus 1.

This method does not work for the following boundary cases:

- Less than or equal of the largest unsigned number (1111...1)
- Greater than or equal of the smallest unsigned number (0000...0)
- Less than or equal of the maximum positive number when in signed mode (0111...1)
- Greater than or equal of the maximum negative number when in signed mode (1000...)

These boundary cases need no special support because they all mean ‘always true’ and can be programmed using the ignore option of the load/store watchpoint programming (refer to **21.3 Watchpoints and Breakpoints Support**).

21.3.2 Instruction Support

There are four instruction address comparators A,B,C, and D. Each is 30 bits long, generating two output signals: equal and less than. These signals are used to generate one of the following four events: equal, not equal, greater than, less than.

The instruction watchpoints and breakpoint are generated using these events and according to user programming. Note that using the OR option enables “out of range” detect.

Table 21-6 Instruction Watchpoints Programming Options

Name	Description	Programming options
IWP0	First instruction watchpoint	Comparator A Comparators (A&B)
IWP1	Second instruction watchpoint	Comparator B Comparator (A | B)
IWP2	Third instruction watchpoint	Comparator C Comparators (C&D)
IWP3	Fourth instruction watchpoint	Comparator D Comparator (C | D)

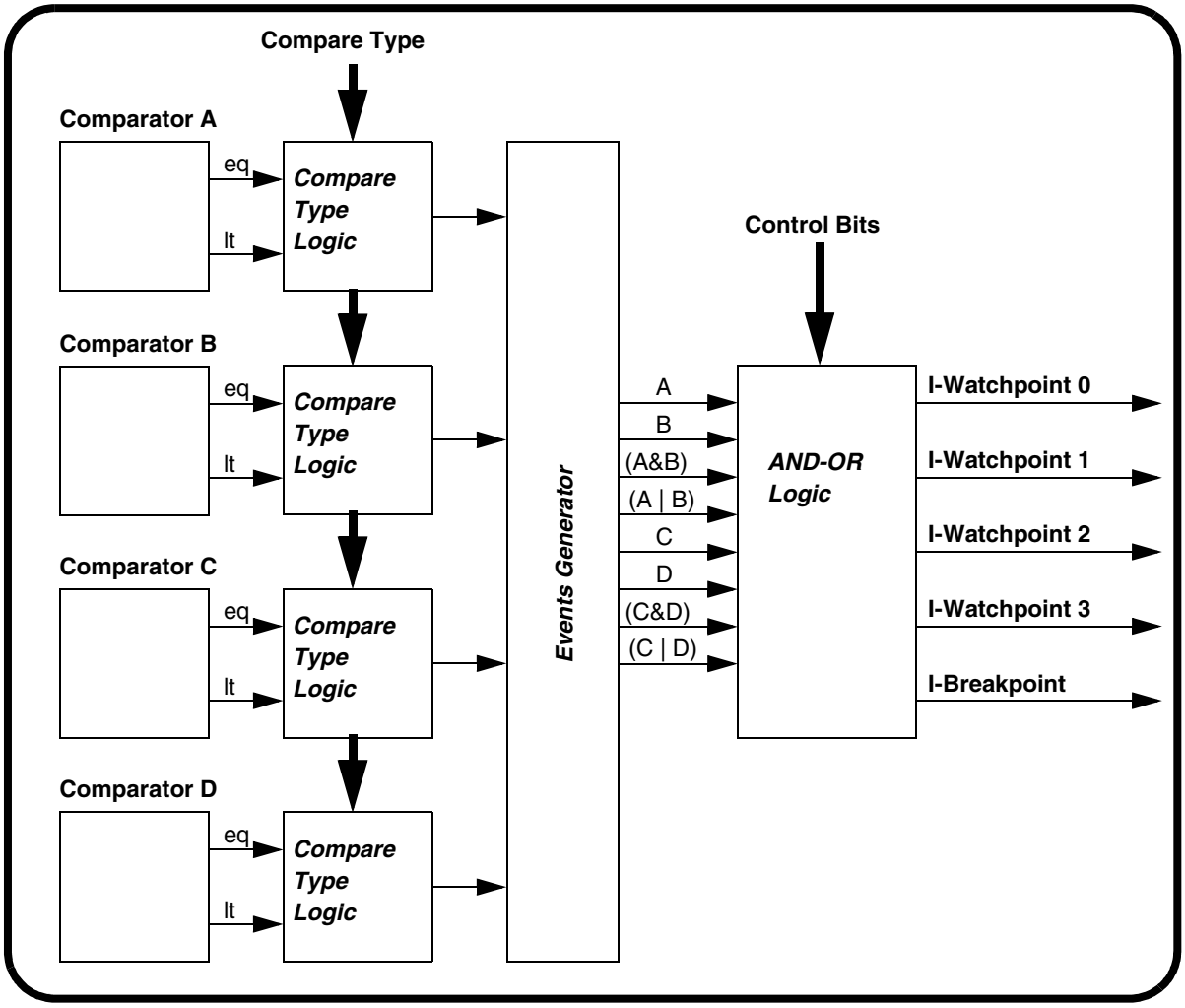


Figure 21-3 Instruction Support General Structure

21.3.2.1 Load/Store Support

There are two load/store address comparators E, and F. Each compares the 32 address bits and the cycle's attributes (read/write). The two least-significant bits are

masked (ignored) whenever a word is accessed and the least-significant bit is masked whenever a half-word is accessed. (For more information refer to **21.3.1.2 Byte and Half-Word Working Modes**). Each comparator generates two output signals: equal and less than. These signals are used to generate one of the following four events (one from each comparator): equal, not equal, greater than, less than.

There are two load/store data comparators (comparators G,H) each is 32 bits wide and can be programmed to treat numbers either as signed values or as unsigned values. Each data comparator operates as four independent byte comparators. Each byte comparator has a mask bit and generates two output signals: equal and less than, if the mask bit is not set. Therefore, each 32 bit comparator has eight output signals.

These signals are used to generate the “equal and less than” signals according to the compare size programmed by the user (byte, half-word, word). When operating in byte mode all signals are significant, when operating in half-word mode only four signals from each 32 bit comparator are significant. When operating in word mode only two signals from each 32 bit comparator are significant.

From the *new* “equal and less than” signals and according to the compare type programmed by the user one of the following four match events are generated: equal, not equal, greater than, less than. Therefore, from the two 32-bit comparators eight match indications are generated: Gmatch[0:3], Hmatch[0:3].

According to the lower bits of the address and the size of the cycle, only match indications that were detected on bytes that have valid information are validated, the rest are negated. Note that if the cycle executed has a smaller size than the compare size (e.g., a byte access when the compare size is word or half-word) no match indication will be asserted.

Using the match indication signals four load/store data events are generated in the following way.

Table 21-7 Load/Store Data Events

Event Name	Event Function[1]
G	(Gmatch0 | Gmatch1 | Gmatch2 | Gmatch3)
H	(Hmatch0 | Hmatch1 | Hmatch2 | Hmatch3)
(G&H)	((Gmatch0 & Hmatch0) | (Gmatch1 & Hmatch1) | (Gmatch2 & Hmatch2) | (Gmatch3 & Hmatch3))
(G | H)	((Gmatch0 | Hmatch0) | (Gmatch1 | Hmatch1) | (Gmatch2 | Hmatch2) | (Gmatch3 | Hmatch3))

NOTES:
1. ‘&’ denotes a logical AND, ‘|’ denotes a logical OR

The four load/store data events together with the match events of the load/store address comparators and the instruction watchpoints are used to generate the load/store watchpoints and breakpoint according to the users programming.

Table 21-8 Load/Store Watchpoints Programming Options

Name	Description	Instruction Events Programming Options	L-address Events Programming Options	L-data Events Programming Options
LWP0	First Load/store watch-point	IWP0, IWP1, IWP2, IWP3, ignore instruction events	Comparator E Comparator F Comparators (E&F) Comparators (E | F) ignore L-addr events	Comparator G Comparator H Comparators (G&H) Comparators (G | H) ignore L-data events
LWP1	Second Load/store watch-point	IWP0, IWP1, IWP2, IWP3, ignore instruction events	Comparator E Comparator F Comparators (E&F) Comparators (E | F) ignore I-addr events	Comparator G Comparator H Comparators (G&H) Comparators (G | H) ignore L-data events

Note that when programming the load/store watchpoints to ignore L-addr events and L-data events, it does not reduce the load/store watchpoints detection logic to be instruction watchpoint detection logic since the instruction must be a load/store instruction for the load/store watchpoint event to trigger.

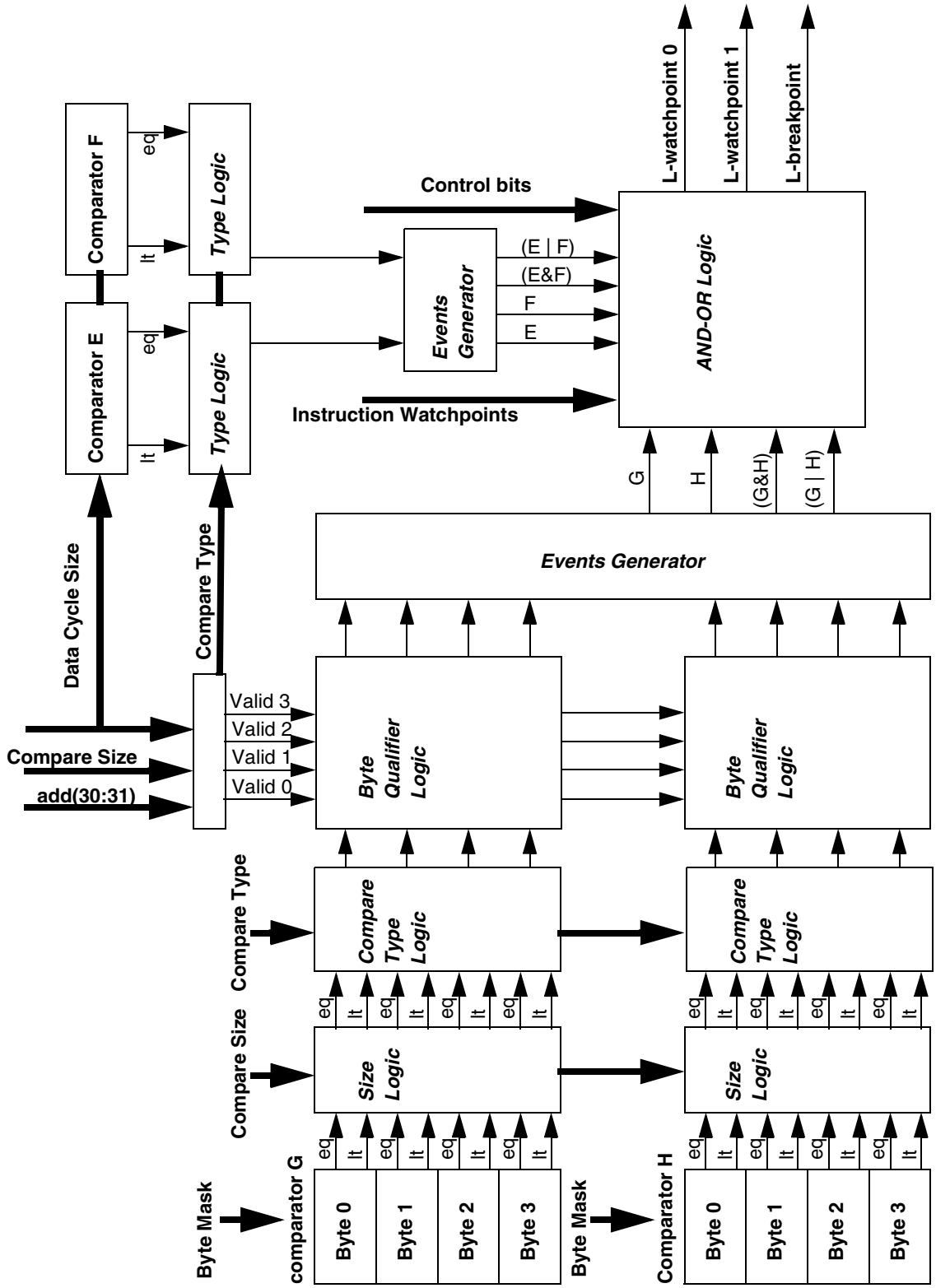


Figure 21-4 Load/Store Support General Structure

21.3.3 Watchpoint Counters

There are two 16-bit watchpoint counters. Each counter is able to count one of the instruction watchpoints or one of the load/store watchpoints. Both generate the corresponding breakpoint when they reach ZERO.

When working in the *masked* mode, the counters do **not** count watchpoints detected when MSRRI = 0. See **21.3.1.4 Context Dependent Filter**

The counters value when counting watchpoints programmed on the actual instructions that alter the counters, are not predictable. Reading values from the counters when they are active, must be synchronized by inserting a sync instruction before the actual read is performed.

NOTE

When programmed to count instruction watchpoints, the last instruction which decrements the counter to ZERO is treated like any other instruction breakpoint in the sense that it is not executed and the machine branches to the breakpoint exception routine BEFORE it executes this instruction. As a side effect of this behavior, the value of the counter inside the breakpoint exception routine equals ONE and not ZERO as might be expected.

When programmed to count load/store watchpoints, the last instruction which decrements the counter to ZERO is treated like any other load/store breakpoint in the sense that it is executed and the machine branches to the breakpoint exception routine AFTER it executes this instruction. Therefore, the value of the counter inside the breakpoint exception routine equals ZERO.

21.3.3.1 Trap Enable Programming

The trap enable bits can be programmed by regular software (only if MSRPR = 0) using THE **mtspr** instruction or “on the fly” using the special development port interface. For more information refer to section **21.5.6.5 Development Port Serial Communications — Trap Enable Mode**.

The value used by the breakpoints generation logic is the bit wise **OR** of the software trap enable bits, (the bits written using the **mtspr**) and the development port trap enable bits (the bits serially shifted using the development port).

All bits, the software trap enable bits and the development port trap enable bits, can be read from ICTRL and the LCTRL2 using **mfspr**. For the exact bits placement refer to **21.7.6 I-Bus Support Control Register** and to **21.7.8 L-Bus Support Control Register 2**

21.4 Development System Interface

When debugging an existing system, it is sometimes desirable to be able to do so without the need to insert any changes in the existing system. In some cases it is not de-

sired, or even impossible, to add load to the lines connected to the existing system. The development system interface of the CPU supports such a configuration.

The development system interface of the CPU uses a dedicated serial port (the development port) and, therefore, does not need any of the regular system interfaces. Controlling the activity of the system from the development port is done when the CPU is in the debug mode. The development port is a relatively economical interface (three pins) that allows the development system to operate in a lower frequency than the frequency of the CPU. Note that it is also possible to debug the CPU using monitor debugger software, for more information refer to **21.6 Software Monitor Debugger Support**.

Debug mode is a state where the CPU fetches all instructions from the development port. In addition, when in debug mode, data can be read from the development port and written to the development port. This allows memory and registers to be read and modified by a development tool (emulator) connected to the development port.

For protection purposes, two possible working modes are defined: debug mode enable and debug mode disable. These working modes are selected only during reset. For more information refer to **21.4.1.1 Debug Mode Enable vs. Debug Mode Disable**

The user can work in debug mode starting from reset or the CPU can be programmed to enter debug mode as a result of a predefined list of events. These events include all possible interrupts and exceptions in the CPU system, including the internal breakpoints, together with two levels of development port requests (*masked* and *non-masked*) and one peripheral breakpoint request that can be generated by any one of the peripherals of the system (including internal and external modules). Each event can be programmed either to be treated as a regular interrupt that causes the machine to branch to its interrupt vector, or to be treated as a special interrupt that causes debug mode entry.

When in debug mode an **rfi** instruction will return the machine to its regular work mode.

The relationship between the debug mode logic to the rest of the CPU chip is shown in the following figure.

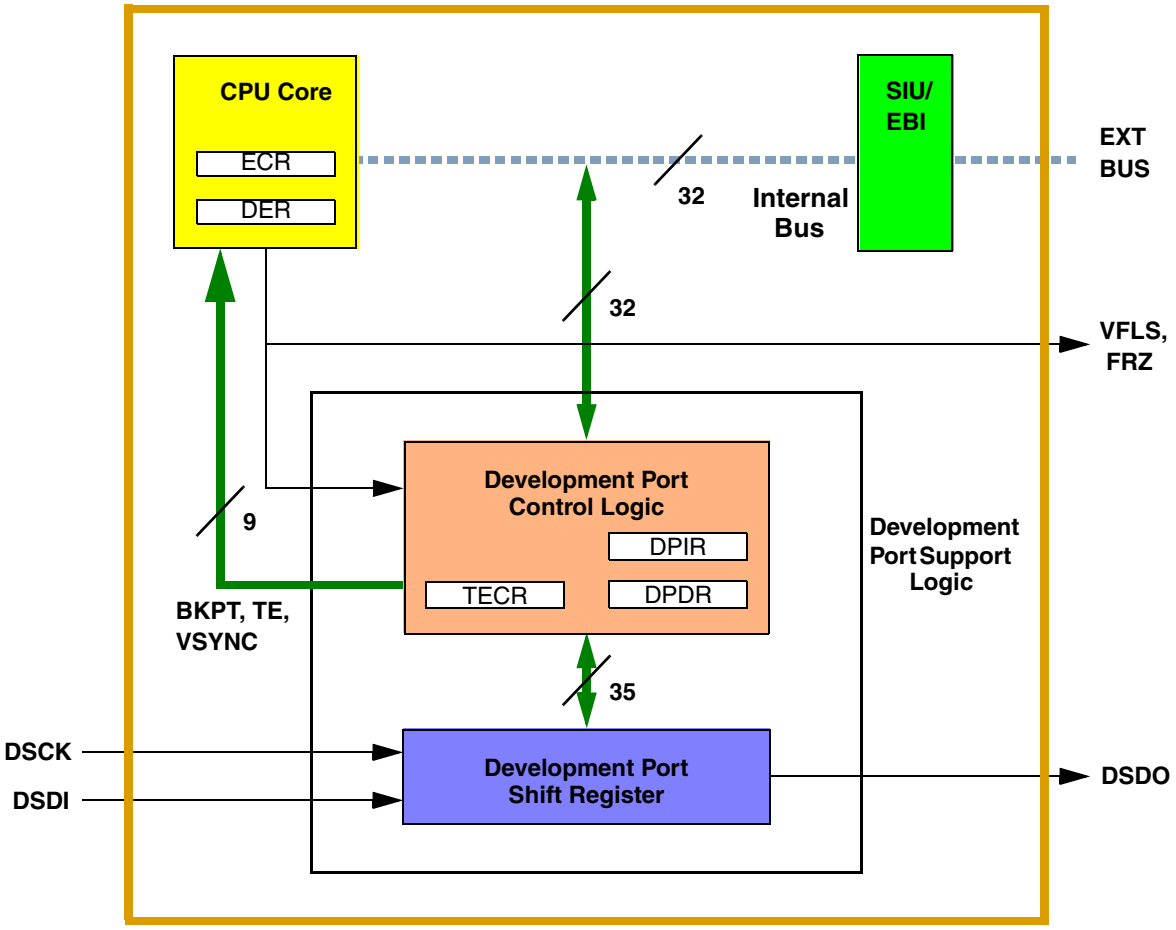


Figure 21-5 Functional Diagram of MPC555 / MPC556 Debug Mode Support

The development port provides a full duplex serial interface for communications between the internal development support logic of the CPU and an external development tool. The development port can operate in two working modes: the trap enable mode and the debug mode.

The trap enable mode is used in order to shift into the CPU internal development support logic the following control signals:

1. Instruction trap enable bits, used for on the fly programming of the instruction breakpoint
2. Load/store trap enable bits, used for on the fly programming of the load/store breakpoint
3. Non-maskable breakpoint, used to assert the non-maskable external breakpoint
4. Maskable breakpoint, used to assert the maskable external breakpoint
5. VSYNC, used to assert and negate VSYNC

In debug mode the development port controls also the debug mode features of the CPU. For more information **21.5 Development Port**

21.4.1 Debug Mode Support

The debug mode of the CPU provides the development system with the following basic functions:

- Gives an ability to control the execution of the processor and maintain control on it under all circumstances. The development port is able to force the CPU to enter to the debug mode even when external interrupts are disabled.
- It is possible to enter debug mode immediately out of reset thus allowing the user even to debug a ROM-less system.
- The user can selectively define, using an enable register, the events that will cause the machine to enter into the debug mode.
- When in debug mode the user can detect the reason upon which the machine entered debug mode by reading a cause register.
- Entering into the debug mode in all regular cases is restartable in the sense that the user is able to continue to run his regular program from the location where it entered the debug mode.
- When in debug mode all instructions are fetched from the development port but load/store accesses are performed on the real system memory.
- Data Register of the development port is accessed using **mtspr** and **mfspr** instructions via special load/store cycles. (This feature together with the last one enables easy memory dump & load).
- Upon entering debug mode, the processor gets into the privileged state (MSRPR = 0). This allows execution of any instruction, and access to any storage location.
- An OR signal of all exception cause register (ECR) bits (ECR_OR) enables the development port to detect pending events while already in debug mode. An example is the ability of the development port to detect a debug mode access to a non existing memory space.

The following figure illustrates the debug mode logic implemented in the CPU.

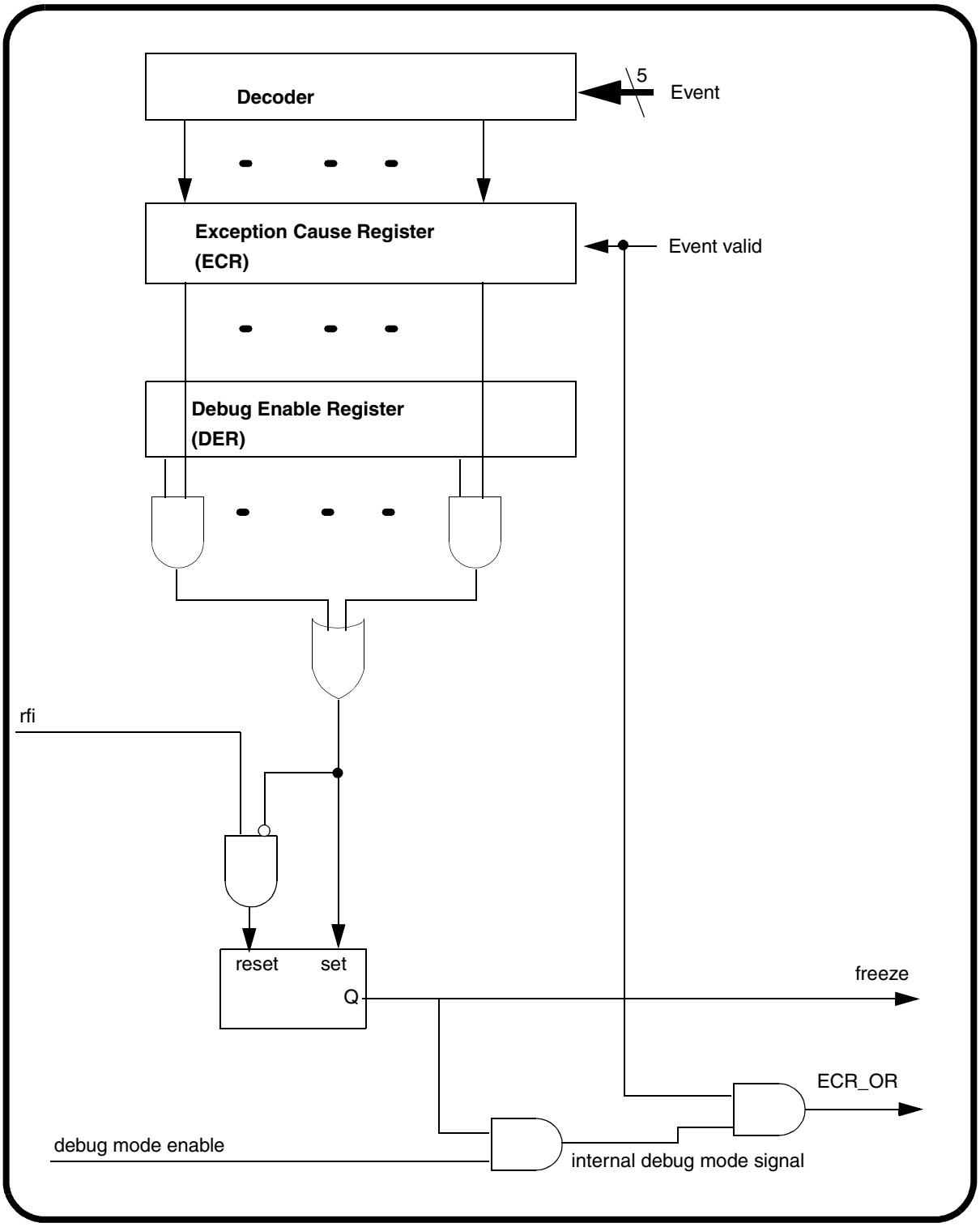


Figure 21-6 Debug Mode Logic

21.4.1.1 Debug Mode Enable vs. Debug Mode Disable

For protection purposes two possible working modes are defined: debug mode enable and debug mode disable. These working modes are selected only during reset.

Debug mode is enabled by asserting the DSCK pin during reset. The state of this pin is sampled three clocks before the negation of $\overline{\text{SRESET}}$.

NOTE

Since $\overline{\text{SRESET}}$ negation is done by an external pull up resistor any reference here to $\overline{\text{SRESET}}$ negation time refers to the time the MPC555 / MPC556 releases $\overline{\text{SRESET}}$. If the actual negation is slow due to large resistor, set up time for the debug port signals should be set accordingly.

If the DSCK pin is sampled negated, debug mode is disabled until a subsequent reset when the DSCK pin is sampled in the asserted state. When debug mode is disabled the internal watchpoint/breakpoint hardware will still be operational and may be used by a software monitor program for debugging purposes.

When working in debug mode disable, all development support registers (see list in **Table 21-14**) are accessible to the supervisor code (MSRPR = 0) and can be used by a monitor debugger software. However, the processor *never* enters debug mode and, therefore, the exception cause register (ECR) and the debug enable register (DER) are used only for asserting and negating the freeze signal. For more information on the software monitor debugger support refer to **21.6 Software Monitor Debugger Support**.

When working in debug mode enable, all development support registers are accessible *only* when the CPU is in debug mode. Therefore, even supervisor code that may be still under debug cannot prevent the CPU from entering debug mode. The development system has full control of all development support features of the CPU through the development port. Refer to **Table 21-16**

21.4.1.2 Entering Debug Mode

Entering debug mode can be a result of a number of events. All events have a programmable enable bit so the user can selectively decide which events result in debug mode entry and which in regular interrupt handling.

Entering debug mode is also possible immediately out of reset, thus allowing the user to debug even a ROM-less system. Using this feature is possible by special programming of the development port during reset. If the DSCK pin continues to be asserted following $\overline{\text{SRESET}}$ negation (after enabling debug mode) the processor will take a breakpoint exception and go directly to debug mode instead of fetching the reset vector. To avoid entering debug mode following reset, the DSCK pin must be negated no later than seven clock cycles after $\overline{\text{SRESET}}$ negates. In this case, the processor will jump to the reset vector and begin normal execution. When entering debug mode immediately after reset, bit 31 (development port interrupt) of the exception cause register (ECR) is set.

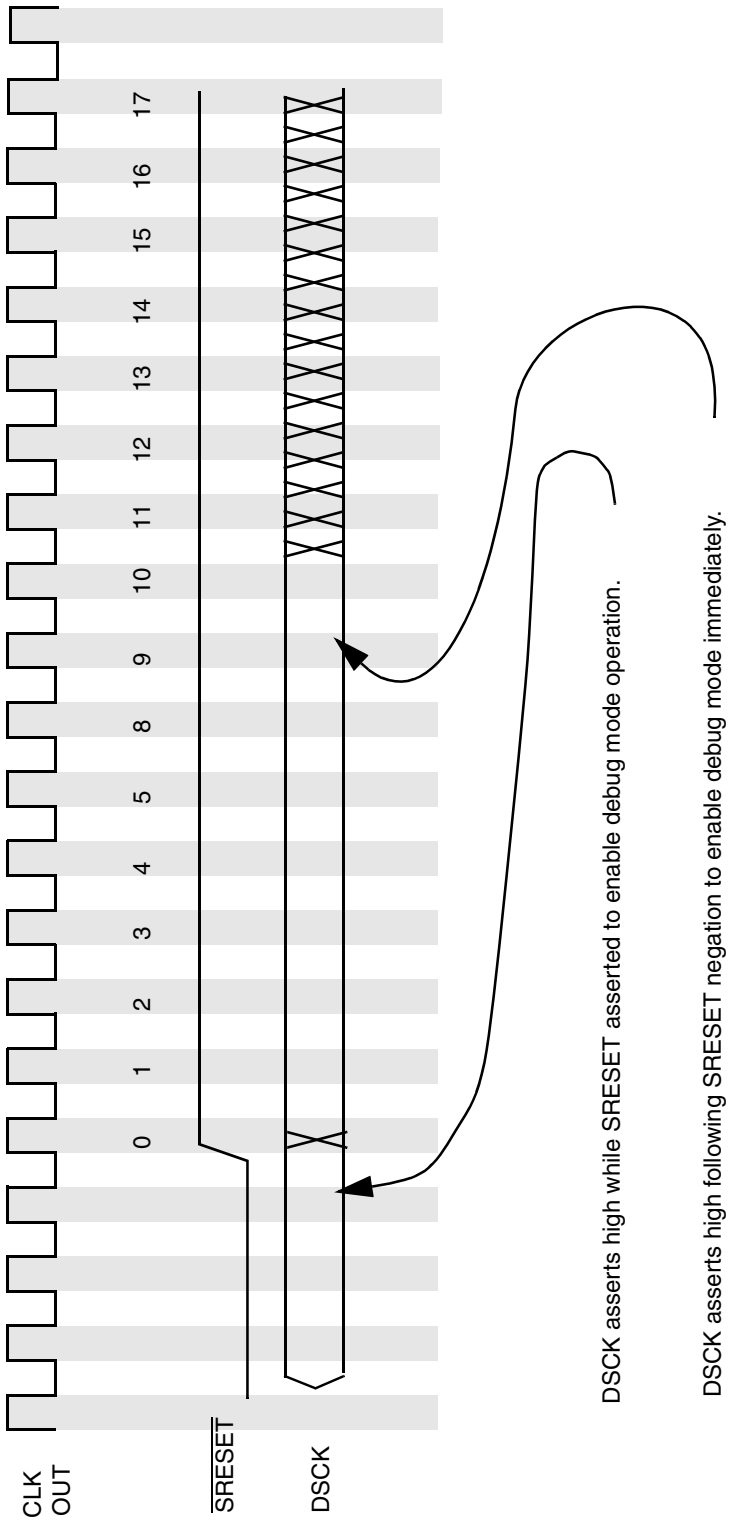


Figure 21-7 Debug Mode Reset Configuration

When debug mode is disabled all events result in regular interrupt handling.

The internal freeze signal is asserted whenever an enabled event occurs, regardless if debug mode is enabled or disabled. The internal freeze signal is connected to all relevant internal modules. These modules can be programmed to stop all operations in response to the assertion of the freeze signal. Refer to **21.6.1 Freeze Indication**.

The freeze indication is negated when exiting debug mode. Refer to **21.4.1.6 Exiting Debug Mode**

The following list contains the events that can cause the CPU to enter debug mode. Each event results in debug mode entry if debug mode is enabled and the corresponding enable bit is set. The reset values of the enable bits let the user, in most cases, to use of the debug mode features without the need to program the debug enable register (DER). For more information refer to **21.7.12 Debug Enable Register (DER)**.

- NMI exception as a result of the assertion of the IRQ0_B pin. For more information refer to **3.15.4.1 System Reset Interrupt**
- Check stop. Refer to **21.4.1.3 The Check Stop State and Debug Mode**
- Machine check exception
- Implementation specific instruction protection error
- Implementation specific data protection error
- External interrupt, recognized when MSREE = 1
- Alignment interrupt
- Program interrupt
- Floating point unavailable exception
- Floating point assist exception
- Decrementer exception, recognized when MSREE = 1
- System call exception
- Trace, asserted when in single trace mode or when in branch trace mode (refer to **3.15.4.10 Trace Interrupt**)
- Implementation dependent software emulation exception
- Instruction breakpoint, when breakpoints are *masked* (BRKNOMSK bit in the LCTRL2 is clear) recognized only when MSRRI = 1, when breakpoints are not *mask*ed (BRKNOMSK bit in the LCTRL2 is set) always recognized
- Load/store breakpoint, when breakpoints are *masked* (BRKNOMSK bit in the LCTRL2 is cleared) recognized only when MSRRI = 1, when breakpoints are not *masked* (BRKNOMSK bit in the LCTRL2 is set) always recognized
- Peripherals breakpoint, from the development port, internal and external modules. are recognized only when MSRRI = 1.
- Development port non-maskable interrupt, as a result of a debug station request. Useful in some catastrophic events like an endless loop when MSRRI = 0. As a result of this event the machine may enter a non-restartable state, for more information refer to **3.15.4 Interrupts**.

The processor enters into the debug mode state when at least one of the bits in the exception cause register (ECR) is set, the corresponding bit in the debug enable register (DER) is enabled and debug mode is enabled. When debug mode is enabled and an enabled event occurs, the processor waits until its pipeline is empty and then starts fetching the next instructions from the development port. For information on the exact value of machine status save/restore registers (SRR0 and SRR1) refer to **3.15.4 Interrupts**

When the processor is in debug mode the freeze indication is asserted thus allowing any peripheral that is programmed to do so to stop. The fact that the CPU is in debug mode is also broadcast to the external world using the value b11 on the VFLS pins.

NOTE

The freeze signal can be asserted by software when debug mode is disabled.

The development port should read the value of the exception cause register (ECR) in order to get the cause of the debug mode entry. *Reading the exception cause register (ECR) clears all its bits*.

21.4.1.3 The Check Stop State and Debug Mode

The CPU enters the check stop state if the machine check interrupt is disabled (MSRME = 0) and a machine check interrupt is detected. However, if a machine check interrupt is detected when MSRME = 0, debug mode is enabled and the check stop enable bit in the debug enable register (DER) is set, the CPU enters debug mode rather then the check stop state.

The different actions taken by the CPU when a machine check interrupt is detected are shown in the following table.

Table 21-9 The Check Stop State and Debug Mode

MSR_{ME}	Debug Mode Enable	CHSTPE[1]	MCIE[2]	Action Performed by the CPU when Detecting a Machine Check Interrupt	Exception Cause Register (ECR) Value
0	0	X	X	Enter the check stop state	0x20000000
1	0	X	X	Branch to the machine check interrupt	0x10000000
0	1	0	X	Enter the check stop state	0x20000000
0	1	1	X	Enter Debug Mode	0x20000000
1	1	X	0	Branch to the machine check interrupt	0x10000000
1	1	X	1	Enter Debug Mode	0x10000000

NOTES:
1. Check stop enable bit in the debug enable register (DER)
2. Machine check interrupt enable bit in the debug enable register (DER)

21.4.1.4 Saving Machine State upon Entering Debug Mode

If entering debug mode was as a result of any load/store type exception, and therefore the DAR (data address register) and DSISR (data storage interrupt status register)

have some significant value, these two registers must be saved before any other operation is performed. Failing to save these registers may result in loss of their value in case of another load/store type exception inside the development software.

Since exceptions are treated differently when in debug mode (refer to **21.4.1.5 Running in Debug Mode**), there is no need to save machine status save/restore zero register (SRR0) and machine status save/restore one register (SRR1).

21.4.1.5 Running in Debug Mode

When running in debug mode all fetch cycles access the development port regardless of the actual address of the cycle. All load/store cycles access the real memory system according to the cycle's address. The data register of the development port is mapped as a special control register therefore it is accessed using **mtspr** and **mfspr** instructions via special load/store cycles (refer to **21.7.13 Development Port Data Register (DPDR)**).

Exceptions are treated differently when running in debug mode. When already in debug mode, upon recognition of an exception, the exception cause register (ECR) is updated according to the event that caused the exception, a special error indication (ECR_OR) is asserted for one clock cycle to report to the development port that an exception occurred and execution continues in debug mode without any change in SRR0 and SRR1. ECR_OR is asserted before the next fetch occurs to allow the development system to detect the excepting instruction.

Not all exceptions are recognized when in debug mode. Breakpoints and watchpoints are not generated by the hardware when in debug mode (regardless of the value of MSRRI). Upon entering debug mode MSREE is cleared by the hardware thus forcing the hardware to ignore external and decrementer interrupts.

Setting the MSREE bit while in debug mode, (by the debug software), is strictly forbidden. The reason for this restriction is that the external interrupt event is a level signal, and since the CPU only reports exceptions while in debug mode but do not treat them, the CPU does not clear the MSREE bit and, therefore, this event, if enabled, is recognized again every clock cycle.

When the ECR_OR signal is asserted the development station should investigate the exception cause register (ECR) in order to find out the event that caused the exception.

Since the values in SRR0 and SRR1 do not change if an exception is recognized while already in debug mode, they only change once when entering debug mode, saving them when entering debug mode is not necessary.

21.4.1.6 Exiting Debug Mode

The **rfi** instruction is used to exit from debug mode in order to return to the normal processor operation and to negate the freeze indication. The development system may monitor the freeze status to make sure the MPC555 / MPC556 is out of debug mode. It is the responsibility of the software to read the exception cause register (ECR) before

performing the **rfi**. Failing to do so will force the CPU to immediately re-enter to debug mode and to re-assert the freeze indication in case an asserted bit in the interrupt cause register (ECR) has a corresponding enable bit set in the debug enable register (DER).

21.5 Development Port

The development port provides a full duplex serial interface for communications between the internal development support logic including debug mode and an external development tool.

The relationship of the development support logic to the rest of the CPU chip is shown in **Figure 21-5**. The development port support logic is shown as a separate block for clarity. It is implemented as part of the SIU module.

21.5.1 Development Port Pins

The following development port pin functions are provided:

1. Development serial clock (DSCK)
2. Development serial data in (DSDI)
3. Development serial data out (DSDO)

21.5.2 Development Serial Clock

The development serial clock (DSCK) is used to shift data into and out of the development port shift register. At the same time, the new most significant bit of the shift register is presented at the DSDO pin. In all further discussions references to the DSCK signal imply the internal synchronized value of the clock. The DSCK input must be driven either high or low at all times and not allowed to float. A typical target environment would pull this input low with a resistor.

The clock may be implemented as a free running clock or as gated clock. As discussed in section **21.5.6.5 Development Port Serial Communications — Trap Enable Mode** and section **21.5.6.8 Development Port Serial Communications — Debug Mode**, the shifting of data is controlled by ready and start signals so the clock does not need to be gated with the serial transmissions.

The DSCK pin is also used at reset to enable debug mode and immediately following reset to optionally cause immediate entry into debug mode following reset.

21.5.3 Development Serial Data In

Data to be transferred into the development port shift register is presented at the development serial data in (DSDI) pin by external logic. To be sure that the correct value is used internally. When driven asynchronous (synchronous) with the system clock, the data presented to DSDI must be stable a setup time before the rising edge of DSCK (CLKOUT) and a hold time after the rising edge of DSCK (CLKOUT).

The DSDI pin is also used at reset to control the overall chip configuration mode and to determine the development port clock mode. See section **21.5.6.4 Development Port Serial Communications — Clock Mode Selection** for more information.

21.5.4 Development Serial Data Out

The debug mode logic shifts data out of the development port shift register using the development serial data out (DSDO) pin. All transitions on DSDO are synchronous with DSCK or CLKOUT depending on the clock mode. Data will be valid a setup time before the rising edge of the clock and will remain valid a hold time after the rising edge of the clock.

Refer to **Table 21-12** for DSDO data meaning.

21.5.5 Freeze Signal

The freeze indication means that the processor is in debug mode (i.e., normal processor execution of user code is frozen). On the MPC555 / MPC556, the freeze state can be indicated by three different pins. The FRZ signal is generated synchronously with the system clock. This indication may be used to halt any off-chip device while in debug mode as well as a handshake means between the debug tool and the debug port. The internal freeze status can also be monitored through status in the data shifted out of the debug port.

21.5.5.1 SGPIO6/FRZ/$\overline{\text{PTR}}$ Pin

The SGPIO6/FRZ/$\overline{\text{PTR}}$ pin powers up as the $\overline{\text{PTR}}$ function and its function is controlled by the GPC bits in the SIUMCR.

21.5.5.2 IWP[0:1]/VFLS[0:1] Pins

The IWP[0:1]/VFLS[0:1] pins power up as the VFLS[0:1] function and their function can be changed via the DBGC bits in the SIUMCR (see **6.13.1.1 SIU Module Configuration Register**). They can also be set via the reset configuration word (See **7.5.2 Hard Reset Configuration Word)**. The FRZ state is indicated by the value b11 on the VFLS[0:1] pins.

21.5.5.3 VFLS[0:1]_MPIO32B[3:4] Pins

The VFLS[0:1]_MPIO32B[3:4] Pins power up as the MPIO32B[3:4] function and their function can be changed via the VFLS bit in the MIOS1TPCR register (see section 15.15.1.1). The FRZ state is indicated by the value b11 on the VFLS[0:1] pins.

21.5.6 Development Port Registers

The development port consists logically of the three registers: development port instruction register (DPIR), development port data register (DPDR), and trap enable control register (TECR). These registers are physically implemented as two registers, development port shift register and trap enable control register. The development port shift register acts as both the DPIR and DPDR depending on the operation being per-

formed. It is also used as a temporary holding register for data to be stored into the TECR. These registers are discussed below in more detail.

21.5.6.1 Development Port Shift Register

The development port shift register is a 35-bit shift register. Instructions and data are shifted into it serially from DSDI using DSCK (or CLKOUT depending on the debug port clock mode, refer to **21.5.6.4 Development Port Serial Communications — Clock Mode Selection**) as the shift clock. These instructions or data are then transferred in parallel to the CPU, the trap enable control register (TECR). When the processor enters debug mode it fetches instructions from the DPIR which causes an access to the development port shift register. These instructions are serially loaded into the shift register from DSDI using DSCK (or CLKOUT) as the shift clock. In a similar way, data is transferred to the CPU by moving it into the shift register which the processor reads as the result of executing a “move from special purpose register DPDR” instruction. Data is also parallel-loaded into the development port shift register from the CPU by executing a “move to special purpose register DPDR” instruction. It is then shifted out serially to DSDO using DSCK (or CLKOUT) as the shift clock.

21.5.6.2 Trap Enable Control Register

The trap enable control register is a 9-bit register that is loaded from the development port shift register. The contents of the control register are used to drive the six trap enable signals, the two breakpoint signals, and the VSYNC signal to the CPU. The “transfer data to trap enable control register” commands will cause the appropriate bits to be transferred to the control register.

The trap enable control register is not accessed by the CPU, but instead supplies signals to the CPU. The trap enable bits, VSYNC bit, and the breakpoint bits of this register are loaded from the development port shift register as the result of trap enable mode transmissions. The trap enable bits are reflected in ICTRL and LCTRL2 special registers. See **21.7.6 I-Bus Support Control Register** and **21.7.8 L-Bus Support Control Register 2**.

21.5.6.3 Development Port Registers Decode

The development port shift register is selected when the CPU accesses DPIR or DPDR. Accesses to these two special purpose registers occur in debug mode and appear on the internal bus as an address and the assertion of an address attribute signal indicating that a special purpose register is being accessed. The DPIR register is read by the CPU to fetch all instructions when in debug mode and the DPDR register is read and written to transfer data between the CPU and external development tools. The DPIR and DPDR are pseudo registers. Decoding either of these registers will cause the development port shift register to be accessed. The debug mode logic knows whether the CPU is fetching instructions or reading or writing data. If what the CPU is expecting and what the register receives from the serial port do not match (instruction instead of data) the mismatch is used to signal a sequence error to the external development tool.

21.5.6.4 Development Port Serial Communications — Clock Mode Selection

All of the serial transmissions are clock transmissions and are therefore synchronous communications. However, the transmission clock may be either synchronous or asynchronous with the system clock (CLKOUT). The development port allows the user to select two methods for clocking the serial transmissions. The first method allows the transmission to occur without being externally synchronized with CLKOUT, in this mode a serial clock DSCK must be supplied to the MPC555 / MPC556. The other communication method requires a data to be externally synchronized with CLKOUT.

The first clock mode is called “asynchronous clock” since the input clock (DSCK) is asynchronous with CLKOUT. To be sure that data on DSDI is sampled correctly, transitions on DSDI must occur a setup time ahead and a hold time after the rising edge of DSCK. This clock mode allows communications with the port from a development tool which does not have access to the CLKOUT signal or where the CLKOUT signal has been delayed or skewed. Refer to the timing diagram in **Figure 21-8**

The second clock mode is called “synchronous self clock”. It does not require an input clock. Instead the port is timed by the system clock. The DSDI input is required to meet setup and hold time requirements with respect to CLKOUT rising edge. The data rate for this mode is always the same as the system clock. Refer to the timing diagram in **Figure 21-9**.

The selection of clock or self clock mode is made at reset. The state of the DSDI input is latched eight clocks after $\overline{\text{SRESET}}$ negates. If it is latched low, asynchronous clock mode is enabled. If it is latched high then synchronous self clock mode is enabled.

Since DSDI is used to select the development port clocking scheme, it is necessary to prevent any transitions on DSDI during this time from being recognized as the start of a serial transmission. The port will not begin scanning for the start bit of a serial transmission until 16 clocks after the negation of $\overline{\text{SRESET}}$. If DSDI is asserted 16 clocks after $\overline{\text{SRESET}}$ negation, the port will wait until DSDI is negated to begin scanning for the start bit.

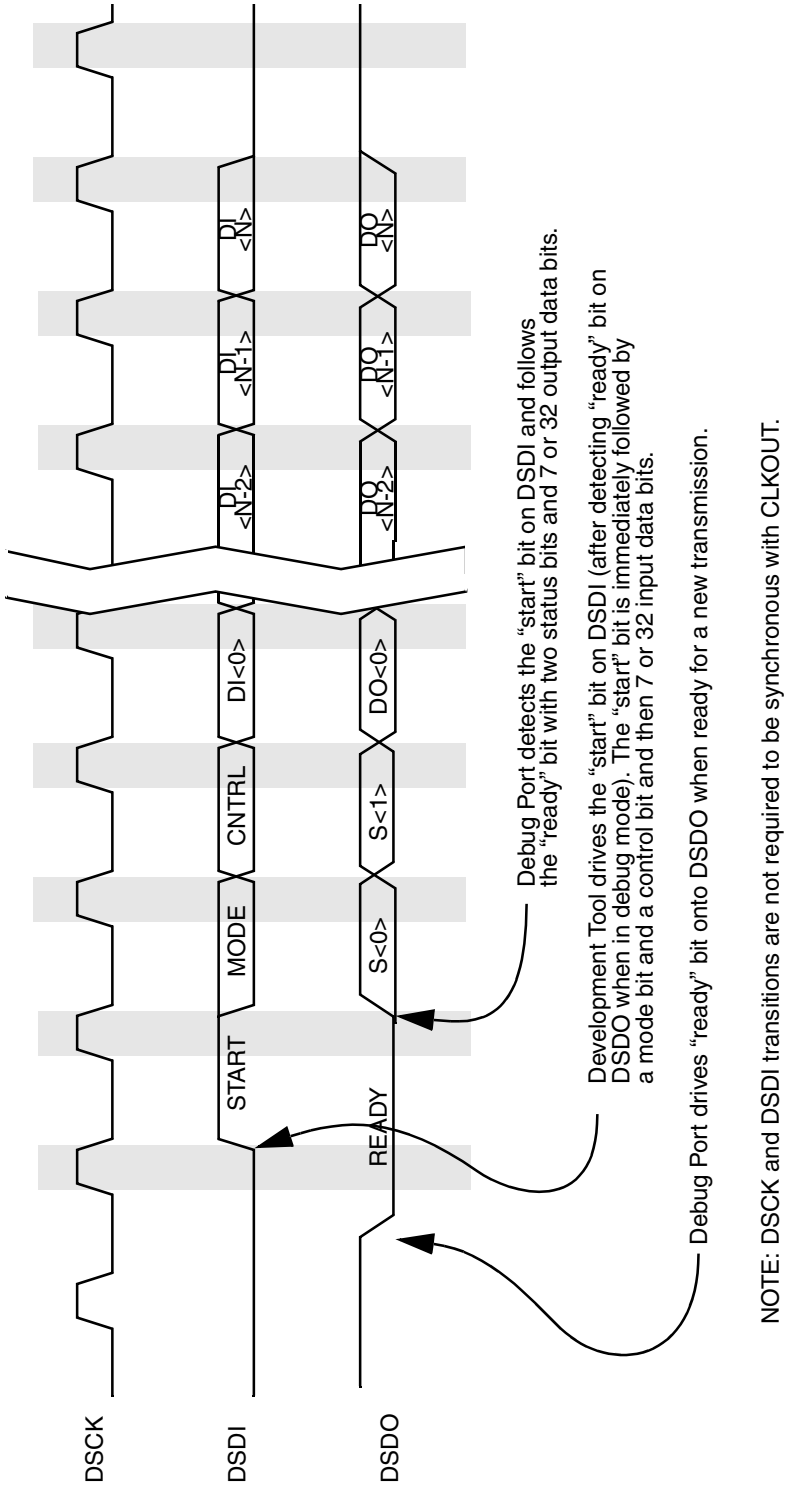


Figure 21-8 Asynchronous Clock Serial Communications

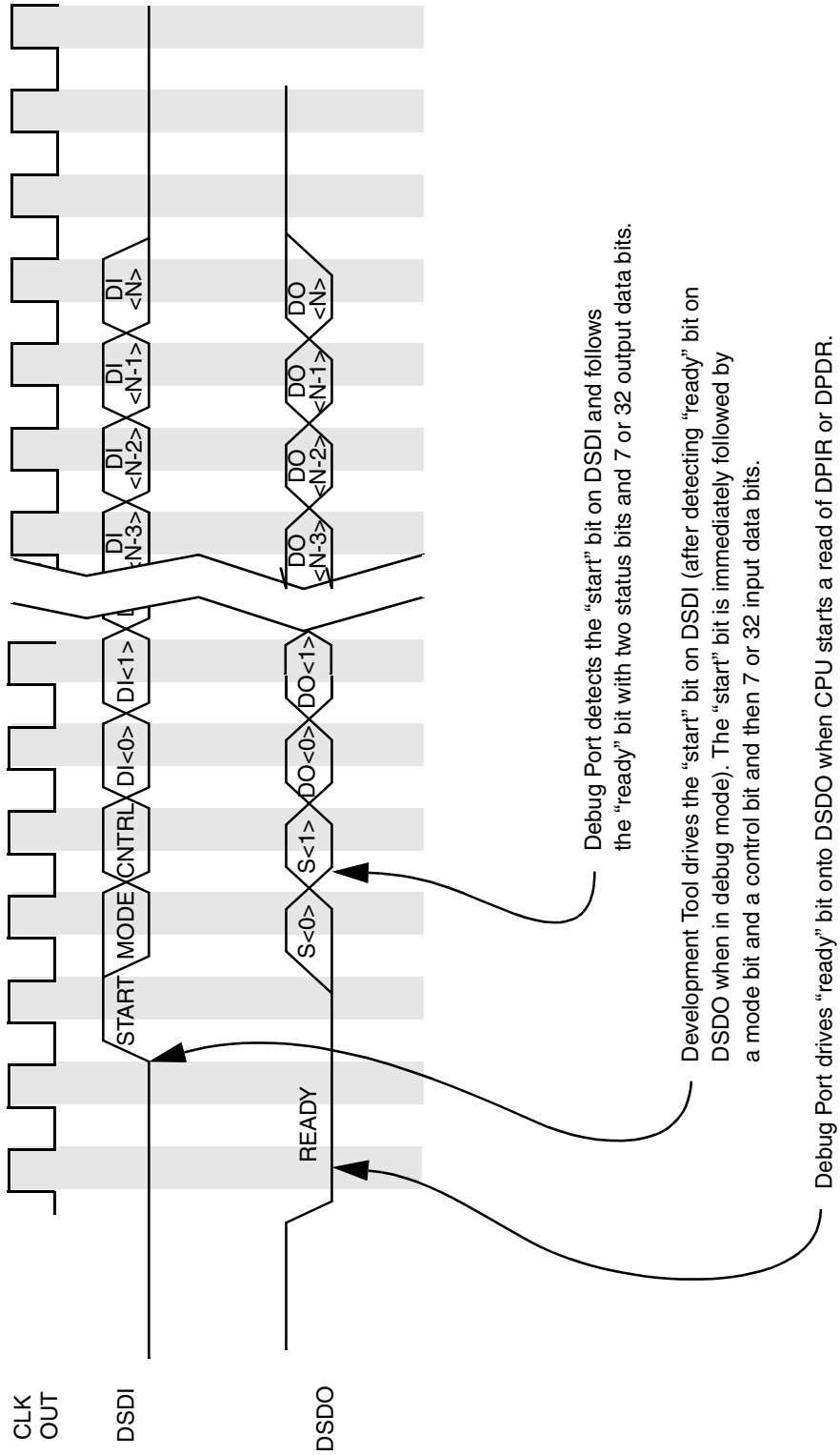


Figure 21-9 Synchronous Self Clock Serial Communication

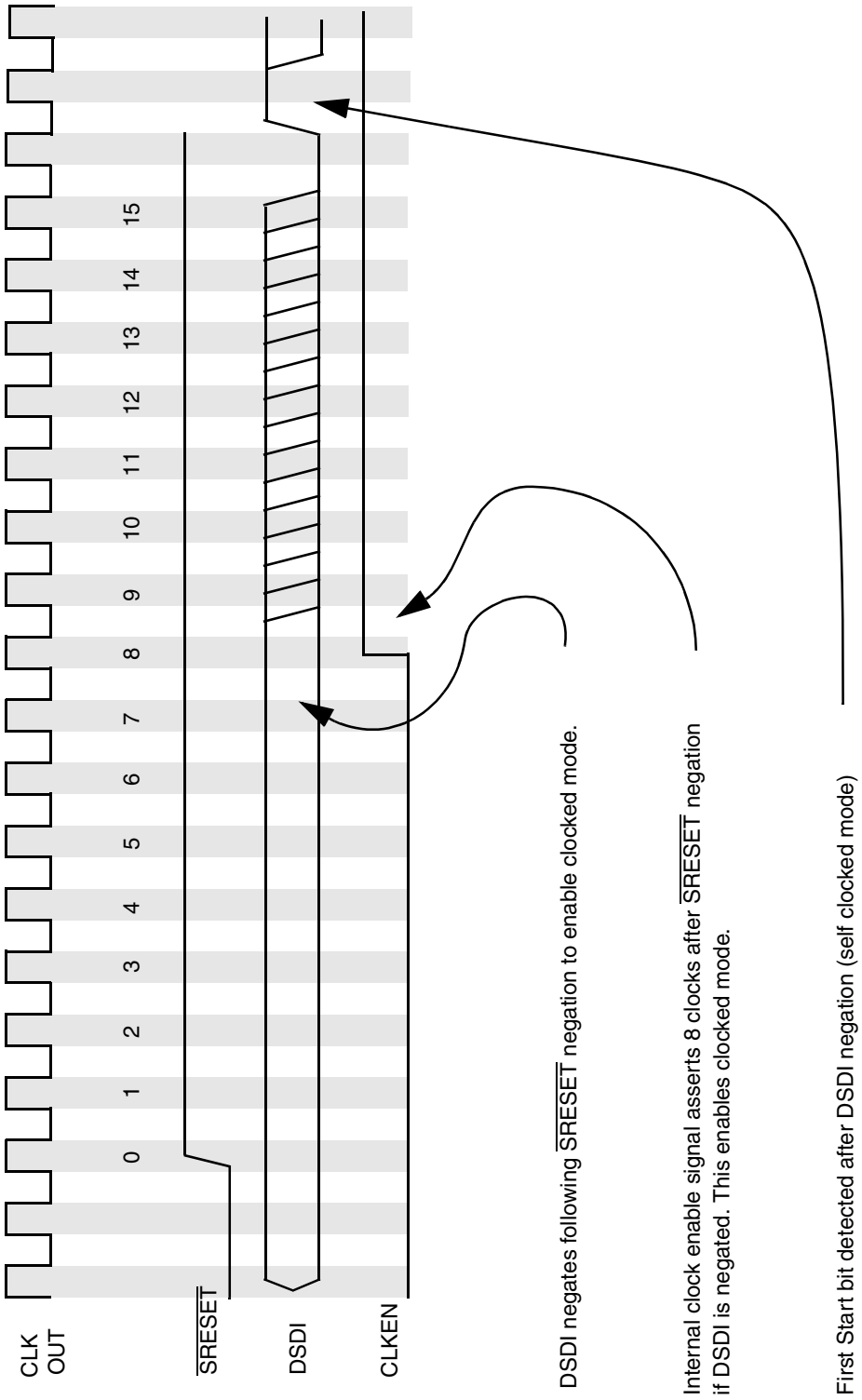


Figure 21-10 Enabling Clock Mode Following Reset

21.5.6.5 Development Port Serial Communications — Trap Enable Mode

When in not in debug mode the development port starts communications by setting DSDO (the MSB of the 35-bit development port shift register) low to indicate that all activity related to the previous transmission are complete and that a new transmission may begin. The start of a serial transmission from an external development tool to the development port is signaled by a start bit. A mode bit in the transmission defines the transmission as either a trap enable mode transmission or a debug mode transmission. If the mode bit is set the transmission will only be 10 bits long and only seven data bits will be shifted into the shift register. These seven bits will be latched into the TECR. A control bit determines whether the data is latched into the trap enable and VSYNC bits of the TECR or into the breakpoints bits of the TECR.

21.5.6.6 Serial Data into Development Port — Trap Enable Mode

The development port shift register is 35 bits wide but trap enable mode transmissions only use the start/ready bit, a mode/status bit, a control/status bit, and the seven least significant data bits. The encoding of data shifted into the development port shift register (through the DSDI pin) is shown in **Table 21-10** and **Table 21-11** below:

Table 21-10 Trap Enable Data Shifted into Development Port Shift Register

Start	Mode	Control	1st	2nd	3rd	4th	1st	2nd	VSYNC	Function
			- - - - - - Instruction- - - - - -				- - Data- -			
			Watchpoint Trap Enables							
1	1	0	0 = disabled; 1 = enabled							Transfer Data to Trap Enable Control Register

Table 21-11 Debug Port Command Shifted Into Development Port Shift Register

Start	Mode	Control	Extended Opcode		Major Opcode	Function
1	1	1	x	x	00000	NOP
					00001	Hard Reset request
					00010	Soft Reset request
			0	x	00011	Reserved
			1	0	00011	End Download procedure
			1	1	00011	Start Download procedure
			x	x	00100... 11110	Reserved
			x	0	11111	Negate Maskable breakpoint.
			x	1	11111	Assert Maskable breakpoint.
			0	x	11111	Negate Non Maskable breakpoint.
			1	x	11111	Assert Non Maskable breakpoint.

The watchpoint trap enables and VSYNC functions are described in section **21.3 Watchpoints and Breakpoints Support** and section **21.2 Program Flow Tracking**.

The debug port command function allows the development tool to either assert or negate breakpoint requests, reset the processor, activate or deactivate the fast download procedure.

21.5.6.7 Serial Data Out of Development Port — Trap Enable Mode

In trap enable mode the only response out of the development port is “sequencing error.”

Data that can come out of the development port is shown in **Table 21-12**. “Valid data from CPU” and “CPU interrupt” status cannot occur in trap enable mode.

Table 21-12 Status / Data Shifted Out of Development Port Shift Register

Ready	Status [0:1]		Data			Function
			Bit 0	Bit 1	Bits 2:31 or 2:6 — (Depending on Input Mode)	
(0)	0	0	Data			Valid Data from CPU
(0)	0	1	Freeze status[1]	Download Procedure in progress[2]	1’s	Sequencing Error
(0)	1	0			1’s	CPU Interrupt
(0)	1	1			1’s	Null

NOTES:
1. The “Freeze” status is set to (1) when the CPU is in debug mode and to (0) otherwise.
2. The “Download Procedure in progress” status is asserted (0) when Debug port in the Download procedure and is negated (1) otherwise.

When not in debug mode the sequencing error encoding indicates that the transmission from the external development tool was a debug mode transmission. When a sequencing error occurs the development port will ignore the data shifted in while the sequencing error was shifting out. It will be treated as a NOP function.

Finally, the null output encoding is used to indicate that the previous transmission did not have any associated errors.

When not in debug mode, ready will be asserted at the end of each transmission. If debug mode is not enabled and transmission errors can be guaranteed not to occur, the status output is not needed.

21.5.6.8 Development Port Serial Communications — Debug Mode

When in debug mode the development port starts communications by setting DSDO low to indicate that the CPU is trying to read an instruction from DPIR or data from DPDR. When the CPU writes data to the port to be shifted out the ready bit is not set. The port waits for the CPU to read the next instruction before asserting ready. This allows duplex operation of the serial port while allowing the port to control all transmissions from the external development tool. After detecting this ready status the external development tool begins the transmission to the development port with a start bit (logic high) on the DSDI pin.

21.5.6.9 Serial Data Into Development Port

In debug mode the 35 bits of the development port shift register are interpreted as a start/ready bit, a mode/status bit, a control/status bit, and 32 bits of data. All instructions and data for the CPU are transmitted with the mode bit cleared indicating a 32-bit data field. The encoding of data shifted into the development port shift register (through the DSDI pin) is shown below in **Table 21-13**

Table 21-13 Debug Instructions / Data Shifted Into Development Port Shift Register

Start	Mode	Control	Instruction / Data (32 Bits)		Function
			Bits 0:6	Bits 7:31	
1	0	0	CPU Instruction		Transfer Instruction to CPU
1	0	1	CPU Data		Transfer Data to CPU
1	1	0	Trap enable[1]	Not exist	Transfer data to Trap Enable Control Register
1	1	1	0011111	Not exist	Negate breakpoint requests to the CPU.
1	1	1	0	Not exist	nop

NOTES:
1. Refer to **Table 21-10**

Data values in the last two functions other than those specified are reserved.

All transmissions from the debug port on DSDO begin with a “0” or “ready” bit. This indicates that the CPU is trying to read an instruction or data from the port. The external development tool must wait until it sees DSDO go low to begin sending the next transmission.

The control bit differentiates between instructions and data and allows the development port to detect that an instruction was entered when the CPU was expecting data and vice versa. If this occurs a sequence error indication is shifted out in the next serial transmission.

The trap enable function allows the development tool to transfer data to the trap enable control register.

The debug port command function allows the development tool to either negate breakpoint requests, reset the processor, activate or deactivate the fast down load procedure.

The NOP function provides a null operation for use when there is data or a response to be shifted out of the data register and the appropriate next instruction or command will be determined by the value of the response or data shifted out.

21.5.6.10 Serial Data Out of Development Port

The encoding of data shifted out of the development port shift register in debug mode (through the DSDO pin) is the same as for trap enable mode and is shown in **Table 21-12**.

Valid data encoding is used when data has been transferred from the CPU to the development port shift register. This is the result of an instruction to move the contents of a general purpose register to the debug port data register (DPDR). The valid data encoding has the highest priority of all status outputs and will be reported even if an interrupt occurs at the same time. Since it is not possible for a sequencing error to occur and also have valid data there is no priority conflict with the sequencing error status. Also, any interrupt that is recognized at the same time that there is valid data is not related to the execution of an instruction. Therefore, a valid data status will be output and the interrupt status will be saved for the next transmission.

The sequencing error encoding indicates that the inputs from the external development tool are not what the development port and/or the CPU was expecting. Two cases could cause this error:

1. The processor was trying to read instructions and there was data shifted into the development port, or
2. The processor was trying to read data and there was instruction shifted into the development port. The port will terminate the read cycle with a bus error.

This bus error will cause the CPU to signal that an interrupt (exception) occurred. Since a status of sequencing error has a higher priority than exception, the port will report the sequencing error first, and the CPU interrupt on the next transmission. The development port will ignore the command, instruction, or data shifted in while the sequencing error or CPU interrupt is shifted out. The next transmission after all error status is reported to the port should be a new instruction, trap enable or command (possibly the one that was in progress when the sequencing error occurred).

The interrupt-occurred encoding is used to indicate that the CPU encountered an interrupt during the execution of the previous instruction in debug mode. Interrupts may occur as the result of instruction execution (such as unimplemented opcode or arithmetic error), because of a memory access fault, or from an unmasked external interrupt. When an interrupt occurs the development port will ignore the command, instruction, or data shifted in while the interrupt encoding was shifting out. The next transmission to the port should be a new instruction, trap enable or debug port command.

Finally, the null encoding is used to indicate that no data has been transferred from the CPU to the development port shift register.

21.5.6.11 Fast Download Procedure

The download procedure is used to download a block of data from the debug tool into system memory. This procedure can be accomplished by repeating the following sequence of transactions from the development tool to the debug port for the number of data words to be down loaded:

```
INIT:Save RX, RY
RY <- Memory Block address- 4

•••

repeat:mfsprRX, DPDR
DATA word to be moved to memory
stwuRX, 0x4(RY)
until here


•••

Restore RX,RY
```

Figure 21-11 Download Procedure Code Example

For large blocks of data this sequence may take significant time to complete. The “fast download procedure” of the debug port may be used to reduce this time. This time reduction is achieved by eliminating the need to transfer the instructions in the loop to the debug port. The only transactions needed are those required to transfer the data to be placed in system memory. **Figure 21-12** and **Figure 21-13** illustrate the time benefit of the “fast download procedure”.

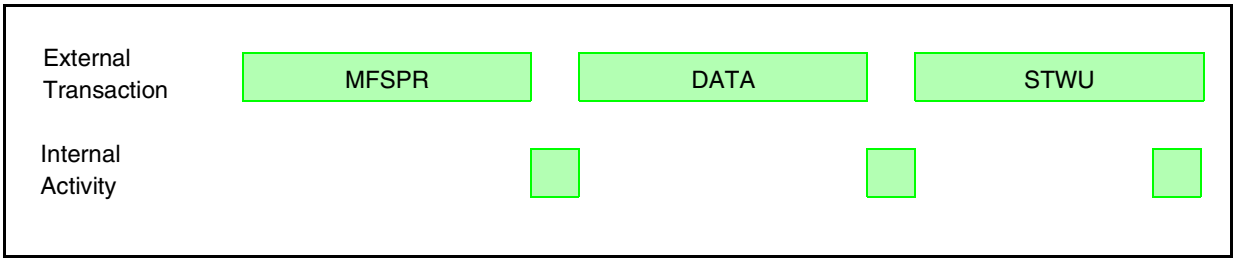


Figure 21-12 Slow Download Procedure Loop

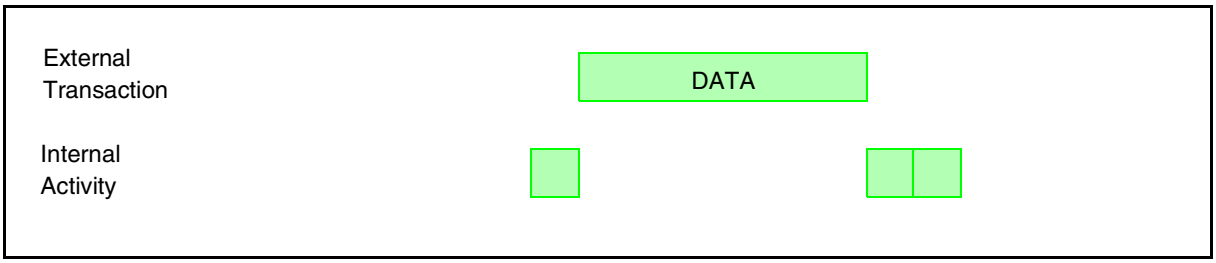


Figure 21-13 Fast Download Procedure Loop

The sequence of the instructions used in the “fast download procedure” is the one illustrated in **Figure 21-11** with RX = r31 and RY = r30. This sequence is repeated infinitely until the “end download procedure” command is issued to the debug port.

Note that, the internal general purpose register 31 is used for temporary storage data value. Before beginning the “fast download procedure” by the “start download procedure command”, The value of the first memory block address, – 4, must be written to the general purpose register 30.

To end a download procedure, an “end download procedure” command should be issued to the debug port, and then, additional DATA transaction should be sent by the development tool. This data word will NOT be placed into the system memory, but it is needed to stop the procedure gracefully.

21.6 Software Monitor Debugger Support

When in debug mode disable, a software monitor debugger can make use of all of the development support features defined in the CPU. When debug mode is disabled all events result in regular interrupt handling, i.e. the processor resumes execution in the corresponding interrupt handler. The exception cause register (ECR) and the debug enable register (DER) only influence the assertion and negation of the freeze signal.

21.6.1 Freeze Indication

The internal freeze signal is connected to all relevant internal modules. These modules can be programmed to stop all operations in response to the assertion of the freeze signal. In order to enable a software monitor debugger to broadcast the fact that the debug software is now executed, it is possible to assert and negate the internal freeze signal also when debug mode is disabled.

The assertion and negation of the freeze signal when in debug mode disable is controlled by the exception cause register (ECR) and the debug enable register (DER) as described in **Figure 21-6**. In order to assert the freeze signal the software needs to program the relevant bits in the debug enable register (DER). In order to negate the freeze line the software needs to read the exception cause register (ECR) in order to clear it and perform an **rfi** instruction.

If the exception cause register (ECR) is not cleared before the **rfi** is performed the freeze signal is not negated. Therefore it is possible to nest inside a software monitor debugger without affecting the value of the freeze line although **rfi** may be performed a few times. Only before the last **rfi** the software needs to clear the exception cause register (ECR).

The above mechanism enables the software to accurately control the assertion and the negation of the freeze signal.

21.7 Development Support Registers

Table 21-14 lists the registers used for development support. The registers are accessed with the **mtspr** and **mfspr** instructions.

Table 21-14 Development Support Programming Model

SPR Number (Decimal)	Name
144	Comparator A Value Register (CMPA) See **Table 21-17** for bit descriptions.
145	Comparator B Value Register (CMPB) See **Table 21-17** for bit descriptions.
146	Comparator C Value Register (CMPC) See **Table 21-17** for bit descriptions.
147	Comparator D Value Register (CMPD) See **Table 21-17** for bit descriptions.
148	Exception Cause Register (ECR) See **Table 21-27** for bit descriptions.
149	Debug Enable Register (DER) See **Table 21-28** for bit descriptions.
150	Breakpoint Counter A Value and Control Register (COUNTA) See **Table 21-25** for bit descriptions.
151	Breakpoint Counter B Value and Control Register (COUNTB) See **Table 21-26** for bit descriptions.
152	Comparator E Value Register (CMPE) See **Table 21-18** for bit descriptions.
153	Comparator F Value Register (CMPF) See **Table 21-18** for bit descriptions.
154	Comparator G Value Register (CMPG) See **Table 21-20** for bit descriptions.
155	Comparator H Value Register (CMPH) See **Table 21-20** for bit descriptions.
156	L-Bus Support Control Register 1 (LCTRL1) See **Table 21-23** for bit descriptions.
157	L-Bus Support Control Register 2 (LCTRL2) See **Table 21-24** for bit descriptions.
158	I-Bus Support Control Register (ICTRL) See **Table 21-21** for bit descriptions.
159	Breakpoint Address Register (BAR) See **Table 21-19** for bit descriptions.
630	Development Port Data Register (DPDR) See **21.7.13 Development Port Data Register (DPDR)** for bit descriptions.

21.7.1 Register Protection

Table 21-15 and **Table 21-16** summarize protection features of development support registers during read and write accesses, respectively.

Table 21-15 Development Support Registers Read Access Protection

MSR[PR]	Debug Mode Enable	In Debug Mode	Result
0	0	X	Read is performed. ECR is cleared when read. Reading DPDR yields indeterminate data.
0	1	0	Read is performed. ECR is *not* cleared when read. Reading DPDR yields indeterminate data.
0	1	1	Read is performed. ECR is cleared when read.
1	X	X	Program exception is generated. Read is not performed. ECR is *not* cleared when read.

Table 21-16 Development Support Registers Write Access Protection

MSR[PR]	Debug Mode Enable	In Debug Mode	Result
0	0	X	Write is performed. Write to ECR is ignored. Writing to DPDR is ignored.
0	1	0	Write is *not* performed. Writing to DPDR is ignored.
0	1	1	Write is performed. Write to ECR is ignored.
1	X	X	Write is *not* performed. Program exception is generated.

21.7.2 Comparator A–D Value Registers (CMPA–CMPD)

CMPA–CMPD — Comparator A–D Value Register **SPR 144 – SPR 147**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CMPAD															

RESET: UNAFFECTED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPAD														RESERVED	

RESET: UNAFFECTED

Table 21-17 CMPA-CMPD Bit Descriptions

Bits	Mnemonic	Description
0:29	CMPAD	Address bits to be compared
30:31	—	Reserved

These registers are unaffected by reset.

21.7.3 Comparator E–F Value Registers

CMPE–CMPF — Comparator E–F Value Registers **SPR 152, 153**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPEF																															

RESET: UNAFFECTED

Table 21-18 CMPE-CMPF Bit Descriptions

Bits	Mnemonic	Description
0:31	CMPV	Address bits to be compared

These registers are unaffected by reset.

21.7.4 Breakpoint Address Register (BAR)

BAR — Breakpoint Address Register **SPR 159**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPEF																															

RESET: UNAFFECTED

Table 21-19 BAR Bit Descriptions

Bits	Mnemonic	Description
0:31	BARV[0:31]	The address of the load/store cycle that generated the breakpoint

21.7.5 Comparator G–H Value Registers (CMPG–CMPH)

CMPG–CMPH — Comparator G–H Value Registers **SPR 154, 155**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPGH																															

RESET: UNAFFECTED

Table 21-20 CMPG-CMPH Bit Descriptions

Bits	Mnemonic	Description
0:31	CMPGH	Data bits to be compared

These registers are unaffected by reset.

21.7.6 I-Bus Support Control Register

ICTRL — I-Bus Support Control Register **SPR 158**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CTA			CTB			CTC			CTD			IWP0		IWP1	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IWP2		IWP3		SIWP0 EN	SIWP1 EN	SIWP2 EN	SIWP3 EN	DIWP0 EN	DIWP 1 EN	DIWP 2 EN	DIWP 3 EN	IIFM	ISCT_SER*		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Changing the instruction show cycle programming starts to take effect only from the second instruction after the actual **mtspr** to ICTRL.

If the processor aborts a fetch of the target of a direct branch (due to an exception), the target is not always visible on the external pins. Program trace is not affected by this phenomenon.

Table 21-21 ICTRL Bit Descriptions

Bits	Mnemonic	Description	Function
0:2	CTA	Compare type of comparator A	0xx = not active (reset value) 100 = equal 101 = less than 110 = greater than 111 = not equal
3:5	CTB	Compare type of comparator B	
6:8	CTC	Compare type of comparator C	
9:11	CTD	Compare type of comparator D	
12:13	IWP0	I-bus 1st watchpoint programming	0x = not active (reset value) 10 = match from comparator A 11 = match from comparators (A&B)
14:15	W1	I-bus 2nd watchpoint programming	0x = not active (reset value) 10 = match from comparator B 11 = match from comparators (A | B)
16:17	IWP2	I-bus 3rd watchpoint programming	0x = not active (reset value) 10 = match from comparator C 11 = match from comparators (C&D)
18:19	IWP3	I-bus 4th watchpoint programming	0x = not active (reset value) 10 = match from comparator D 11 = match from comparators (C | D)
20	SIWP0EN	Software trap enable selection of the 1st I-bus watchpoint	0 = trap disabled (reset value) 1 = trap enabled
21	SIWP1EN	Software trap enable selection of the 2nd I-bus watchpoint	
22	SIWP2EN	Software trap enable selection of the 3rd I-bus watchpoint	
23	SIWP3EN	Software trap enable selection of the 4th I-bus watchpoint	
24	DIWP0EN	Development port trap enable selection of the 1st I-bus watchpoint (read only bit)	0 = trap disabled (reset value) 1 = trap enabled
25	DIWP1EN	Development port trap enable selection of the 2nd I-bus watchpoint (read only bit)	
26	DIWP2EN	Development port trap enable selection of the 3rd I-bus watchpoint (read only bit)	
27	DIWP3EN	Development port trap enable selection of the 4th I-bus watchpoint (read only bit)	
28	IIFM	Ignore first match, only for I-bus breakpoints	0 = Do not ignore first match, used for “go to x” (reset value) 1 = Ignore first match (used for “continue”)
29:31	ISCT_SER	Instruction fetch show cycle and RCPU serialize control	These bits control serialization and instruction fetch show cycles. See **Table 21-5** for the bit definitions. **NOTE**: Changing the instruction show cycle programming starts to take effect only from the second instruction after the actual **mtspr** to ICTRL.

Table 21-22 ISCT_SER Bit Descriptions

Serialize (SER)	Instruction Fetch Show Cycle) Control (ISCTL)	Functions Selected
0	00	RCPU is fully serialized and show cycle will be performed for all fetched instructions (reset value)
0	01	RCPU is fully serialized and show cycle will be performed for all changes in the program flow
0	10	RCPU is fully serialized and show cycle will be performed for all indirect changes in the program flow
0	11	RCPU is fully serialized and no show cycles will be performed for fetched instructions
1	00	Illegal. This mode should not be selected.
1	01	RCPU is not serialized (normal mode) and show cycle will be performed for all changes in the program flow
1	10	RCPU is not serialized (normal mode) and show cycle will be performed for all indirect changes in the program flow
1	11	RCPU is not serialized (normal mode) and no show cycles will be performed for fetched instructions

21.7.7 L-Bus Support Control Register 1

LCTRL1 — L-Bus Support Control Register 1 **SPR 156**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CTE			CTF			CTG			CTH			CRWE		CRWF	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CSG		CSH		SUSG	SUSH	CGBMSK				CHBMSK				UNUSED	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-23 LCTRL1 Bit Descriptions

Bits	Mnemonic	Description	Function
0:2	CTE	Compare type, comparator E	0xx = not active (reset value) 100 = equal 101 = less than 110 = greater than 111 = not equal
3:5	CTF	Compare type, comparator F	
6:8	CTG	Compare type, comparator G	
9:11	CTH	Compare type, comparator H	
12:13	CRWE	Select match on read/write of comparator E	0X = don't care (reset value) 10 = match on read 11 = match on write
14:15	CRWF	Select match on read/write of comparator F	
16:17	CSG	Compare size, comparator G	00 = reserved 01 = word 10 = half word 11 = byte (Must be programmed to word for floating point compares)
18:19	CSH	Compare size, comparator H	
20	SUSG	Signed/unsigned operating mode for comparator G	0 = unsigned 1 = signed (Must be programmed to signed for floating point compares)
21	SUSH	Signed/unsigned operating mode for comparator H	
22:25	CGBMSK	Byte mask for 1st L-data comparator	0000 = all bytes are **not** masked 0001 = the last byte of the word is masked . . . 1111 = all bytes are masked
26:29	CHBMSK	Byte mask for 2nd L-data comparator	
30:31	—	Reserved	—

LCTRL1 is cleared following reset.

21.7.8 L-Bus Support Control Register 2

LCTRL2 — L-Bus Support Control Register 2 **SPR 157**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW0EN	LW0IA		LW0 IADC	LW0LA		LW0 LADC	LW0LD		LW0 LDDC	LW1EN	LW1IA		LW1 IADC	LW1LA	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LW1 LADC	LW1LD		LW1 LDDC	BRK NOMSK	RESERVED							DLW0 EN	DLW1 EN	SLW0 EN	SLW1 EN
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-24 LCTRL2 Bit Descriptions

Bits	Mnemonic	Description	Function
0	LW0EN	1st L-bus watchpoint enable bit	0 = watchpoint not enabled (reset value) 1 = watchpoint enabled
1:2	LW0IA	1st L-bus watchpoint I-addr watchpoint selection	00 = first I-bus watchpoint 01 = second I-bus watchpoint 10 = third I-bus watchpoint 11 = fourth I-bus watchpoint
3	LW0IADC	1st L-bus watchpoint care/don't care I-addr events	0 = Don't care 1 = Care
4:5	LW0LA	1st L-bus watchpoint L-addr events selection	00 = match from comparator E 01 = match from comparator F 10 = match from comparators (E&F) 11 = match from comparators (E | F)
6	LW0LADC	1st L-bus watchpoint care/don't care L-addr events	0 = Don't care 1 = Care
7:8	LW0LD	1st L-bus watchpoint L-data events selection	00 = match from comparator G 01 = match from comparator H 10 = match from comparators (G&H) 11 = match from comparators (G | H)
9	LW0LDDC	1st L-bus watchpoint care/don't care L-data events	0 = Don't care 1 = Care
10	LW1EN	2nd L-bus watchpoint enable bit	0 = watchpoint not enabled (reset value) 1 = watchpoint enabled
11:12	LW1IA	2nd L-bus watchpoint I-addr watchpoint selection	00 = first I-bus watchpoint 01 = second I-bus watchpoint 10 = third I-bus watchpoint 11 = fourth I-bus watchpoint
13	LW1IADC	2nd L-bus watchpoint care/don't care I-addr events	0 = Don't care 1 = Care
14:15	LW1LA	2nd L-bus watchpoint L-addr events selection	00 = match from comparator E 01 = match from comparator F 10 = match from comparators (E&F) 11 = match from comparators (E | F)
16	LW1LADC	2nd L-bus watchpoint care/don't care L-addr events	0 = Don't care 1 = Care
17:18	LW1LD	2nd L-bus watchpoint L-data events selection	00 = match from comparator G 01 = match from comparator H 10 = match from comparators (G&H) 11 = match from comparator (G | H)
19	LW1LDDC	2nd L-bus watchpoint care/don't care L-data events	0 = Don't care 1 = Care
20	BRKNOMSK	Internal breakpoints non-mask bit	0 = masked mode; breakpoints are recognized only when MSR[RI]=1 (reset value) 1 = non-masked mode; breakpoints are always recognized
21:27	—	Reserved	—

Table 21-24 LCTRL2 Bit Descriptions (Continued)

Bits	Mnemonic	Description	Function
28	DLW0EN	Development port trap enable selection of the 1st L-bus watch-point (read only bit)	0 = trap disabled (reset value) 1 = trap enabled
29	DLW1EN	Development port trap enable selection of the 2nd L-bus watchpoint (read only bit)	
30	SLW0EN	Software trap enable selection of the 1st L-bus watchpoint	
31	SLW1EN	Software trap enable selection of the 2nd L-bus watchpoint	

LCTRL2 is cleared following reset.

For each watchpoint, three control register fields (LWxIA, LWxLA, LWxLD) must be programmed. For a watchpoint to be asserted, all three conditions must be detected.

21.7.9 Breakpoint Counter A Value and Control Register

COUNTA — Breakpoint Counter A Value and Control Register **SPR 150**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTV															

RESET: UNAFFECTED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED														CNTC	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-25 Breakpoint Counter A Value and Control Register (COUNTA)

Bit(s)	Name	Description
0:15	CNTV	Counter preset value
16:29	—	Reserved
30:31	CNTC	Counter source select 00 = not active (reset value) 01 = I-bus first watchpoint 10 =L-bus first watchpoint 11 = Reserved

COUNTA[16:31] are cleared following reset; COUNTA[0:15] are unaffected by reset.

21.7.10 Breakpoint Counter B Value and Control Register

COUNTB — Breakpoint Counter B Value and Control Register **SPR 151**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTV															
RESET: UNAFFECTED															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED														CNTC	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-26 Breakpoint Counter B Value and Control Register (COUNTB)

Bit(s)	Name	Description
0:15	CNTV	Counter preset value
16:29	—	Reserved
30:31	CNTC	Counter source select 00 = not active (reset value) 01 = I-bus second watchpoint 10 = L-bus second watchpoint 11 = Reserved

COUNTB[16:31] are cleared following reset; COUNTB[0:15] are unaffected by reset.

21.7.11 Exception Cause Register (ECR)

The ECR indicates the cause of entry into debug mode. All bits are set by the hardware and cleared when the register is read when debug mode is disabled, or if the processor is in debug mode. Attempts to write to this register are ignored. When the hardware sets a bit in this register, debug mode is entered only if debug mode is enabled and the corresponding mask bit in the DER is set.

All bits are cleared to zero following reset.

ECR — Exception Cause Register SPR 148

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RST	CHSTP	MCE	RESERVED		EXTI	ALE	PRE	FPUVE	DECE	RESERVED		SYSE	TR	FPASE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	SEE	RE-SERVED	ITL-BER	RE-SERVED	DTL-BER	RESERVED						LBRK	IBRK	EBRKD	DPI
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-27 ECR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	RST	Reset interrupt bit. This bit is set when the system reset pin is asserted.
2	CHSTP	Checkstop bit. Set when the processor enters checkstop state.
3	MCE	Machine check interrupt bit. Set when a machine check exception (other than one caused by a data storage or instruction storage error) is asserted.
4:5	—	Reserved
6	EXTI	External interrupt bit. Set when the external interrupt is asserted.
7	ALE	Alignment exception bit. Set when the alignment exception is asserted.
8	PRE	Program exception bit. Set when the program exception is asserted.
9	FPUVE	Floating point unavailable exception bit. Set when the program exception is asserted.
10	DECE	Decrementer exception bit. Set when the decrementer exception is asserted.
11:12	—	Reserved
13	SYSE	System call exception bit. Set when the system call exception is asserted.
14	TR	Trace exception bit. Set when in single-step mode or when in branch trace mode.
15	FPASE	Floating point assist exception bit. Set when the floating point assist exception occurs.
16	—	Reserved
17	SEE	Software emulation exception. Set when the software emulation exception is asserted.
18	—	Reserved
19	ITLBER	Implementation specific instruction protection error This bit is set as a result of an instruction protection error. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
20	—	Reserved
21	DTLBER	Implementation specific data protection error This bit is set as a result of an data protection error. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
22:27	—	Reserved

Table 21-27 ECR Bit Descriptions (Continued)

Bit(s)	Name	Description
28	LBRK	L-bus breakpoint exception bit. This bit is set as a result of the assertion of a load/store breakpoint. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
29	IBRK	I-bus breakpoint exception bit. This bit is set as a result of the assertion of an Instruction breakpoint. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
30	EBRK	External breakpoint exception bit. Set when an external breakpoint is asserted (by an on-chip IMB or L-bus module, or by an external device or development system through the development port). This bit is set as a result of the assertion of an external breakpoint. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
31	DPI	Development port interrupt bit. Set by the development port as a result of a debug station non-maskable request or when debug mode is entered immediately out of reset.

21.7.12 Debug Enable Register (DER)

This register enables the user to selectively mask the events that may cause the processor to enter into debug mode.

DER — Debug Enable Register **SPR 149**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RSTE	CHSTPE	MCEE	RESERVED		EXTIE	ALEE	PREE	FPU-VEE	DE-CEE	RESERVED		SY-SEE	TRE	FPASE
RESET:															
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RE-SERVED	SEEE	RE-SERVED	ITL-BERE	RE-SERVED	DTL-BERE	RESERVED						LBRKE	IBRKE	EBRKE	DPIE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 21-28 DER Bit Descriptions

Bit(s)	Name	Description
0:1	—	Reserved
1	RSTE	Reset enable 0 = Debug entry is disabled (reset value) 1 = Debug entry is enabled
2	CHSTPE	Checkstop enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
3	MCEE	Machine check exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
4:5	—	Reserved

Table 21-28 DER Bit Descriptions (Continued)

Bit(s)	Name	Description
6	EXTIE	External interrupt enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
7	ALEE	Alignment exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
8	PREE	Program exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
9	FPUVEE	Floating point unavailable exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
10	DECEE	Decrementer exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
11:12	—	Reserved
13	SYSEE	System call exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
14	TRE	Trace exception enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
15	FPASEE	Floating point assist exception enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
16	—	Reserved
17	SEEE	Software emulation exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
18	—	Reserved
19	ITLBERE	Implementation specific instruction protection error enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
20	—	Reserved
21	DTLBERE	Implementation specific data protection error enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
22:27	—	Reserved
28	LBRKE	Load/store breakpoint enable bit. 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
29	IBRKE	Instruction breakpoint interrupt enable bit. 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
30	EBRKE	External breakpoint interrupt enable bit (development port, internal or external modules). 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
31	DPIE	Development port interrupt enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)

21.7.13 Development Port Data Register (DPDR)

This 32-bit special purpose register physically resides in the development port logic. It is used for data interchange between the core and the development system. An access to this register is initiated using **mtspr** and **mfspr** (SPR 630) and implemented using a special bus cycle on the internal bus.

SECTION 22
IEEE 1149.1-COMPLIANT INTERFACE (JTAG)

The MPC555 / MPC556 includes dedicated user-accessible test logic that is fully compatible with the *IEEE 1149.1-1990 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high-density circuit boards have led to development of this standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The MPC555 / MPC556 supports circuit-board test strategies based on this standard.

This section is intended to be used with the supporting IEEE 1149.1-1990 standard. The scope of this description includes those items required by the standard to be defined and, in certain cases, provides additional information specific to the implementation. For internal details and applications of the standard, refer to the IEEE 1149.1-1990 document.

An overview of the JTAG pins on the MPC555 / MPC556 is shown in **Figure 22-1**.

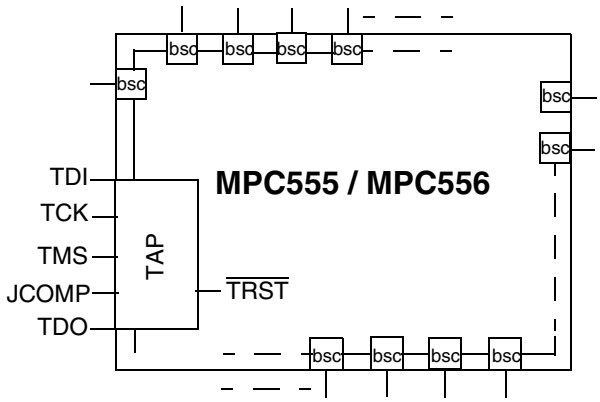


Figure 22-1 JTAG Pins

Boundary scan cells (BSC) are placed at the digital boundary of the chip (normally the package pins). The boundary scan cells are chained together to form a boundary scan register (BSR). The data is serially shifted in through the serial port (TDI) and serially shifted out through the output port (TDO).

22.1 JTAG Interface Block Diagram

A block diagram of the MPC555 / MPC556 implementation of the IEEE 1149.1-1990 test logic is shown in **Figure 22-2**.

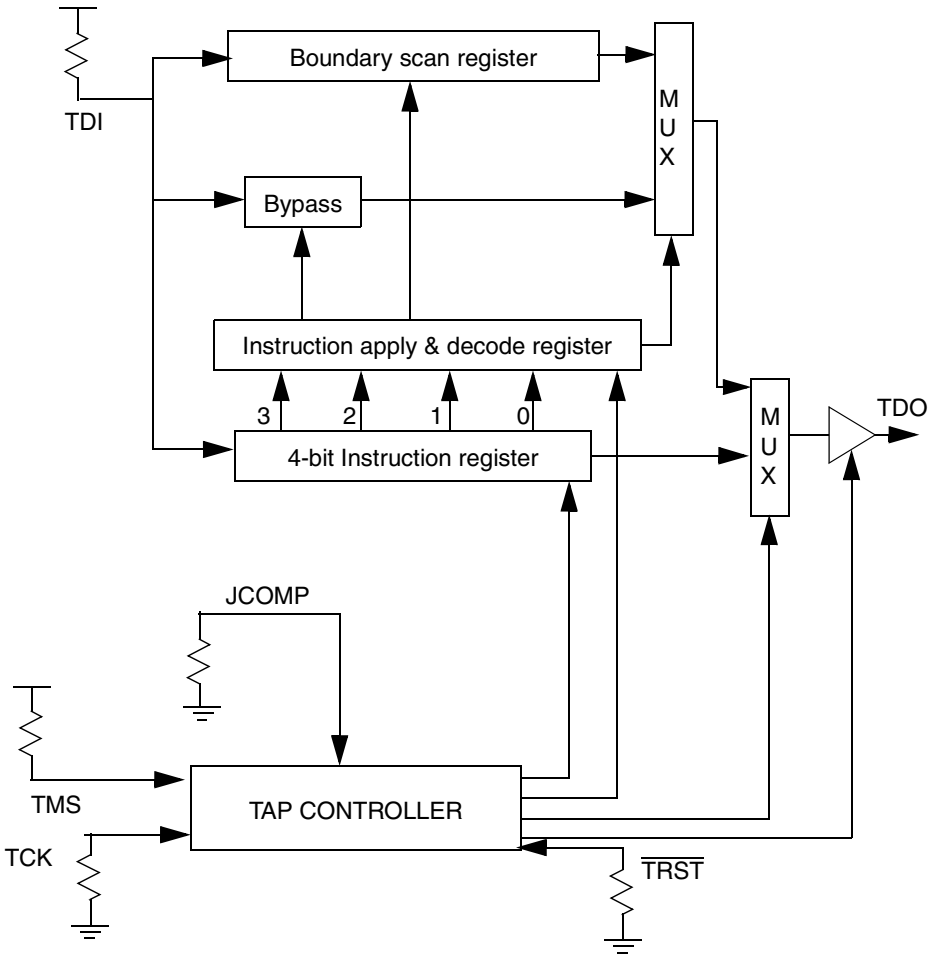


Figure 22-2 Test Logic Block Diagram

22.2 JTAG Signal Descriptions

The MPC555 / MPC556 has five dedicated JTAG pins, which are described in **Table 22-1**. The TDI and TDO scan ports are used to scan instructions as well as data into the various scan registers for JTAG operations. The scan operation is controlled by the test access port (TAP) controller, which in turn is controlled by the TMS input sequence.

To enable JTAG on reset for board test, bit 11 (DGPC select JTAG pins) and bit 16 (PRPM peripheral mode enable) of the reset configuration word should be held high during the rising edge of reset (see **7.5.2 Hard Reset Configuration Word**). These need to be configurable on the user board to allow JTAG test of a board. To allow normal operation of the board these bits need to be low in the reset configuration word.

Table 22-1 JTAG Interface Pin Descriptions

Signal Name	Input/ Output (I/O)	Internal Pull-Up/ Pull-Down Provided	Description
TDI	Input	Pull-up	Test data input pin. Sampled on the rising edge of TCK. Has a pull-up resistor.
TDO	Output	None	Test data output pin. Actively driven during the Shift-IR and Shift-DR controller states. Changes on the falling edge of TCK. Can be placed in high-impedance state.
TMS	Input	Pull-up	Test mode select pin. Sampled on the rising edge of TCK to sequence the test controller's state machine. Has a pull-up resistor.
TCK	Input	Pull-down	Test clock input to synchronize the test logic. Has a pull-down resistor.
$\overline{\text{TRST}}$	Input	Pull-up	TAP controller asynchronous reset. Provides initialization of the TAP controller and other logic as required by the standard. Has a pull-up resistor.

22.3 Operating Frequency

The TCK frequency must be between 5 MHz and 10 MHz. This pin is internally driven to a low value when disconnected.

22.4 TAP Controller

$\overline{\text{TRST}}$ is used to reset the TAP controller asynchronously. The $\overline{\text{TRST}}$ pin ensures that the JTAG logic does not interfere with the normal operation of the chip. This pin is optional in the JTAG specification.

The TAP controller changes state either on the rising edge of TCK or when $\overline{\text{TRST}}$ is asserted.

The TDO signal remains in a high-impedance state except during the Shift-DR or Shift-IR controller states. During these controller states, TDO is updated on the falling edge of TCK.

The TAP controller states are designed to meet the IEEE 1149.1 standard. Refer to **Figure 22-3**.

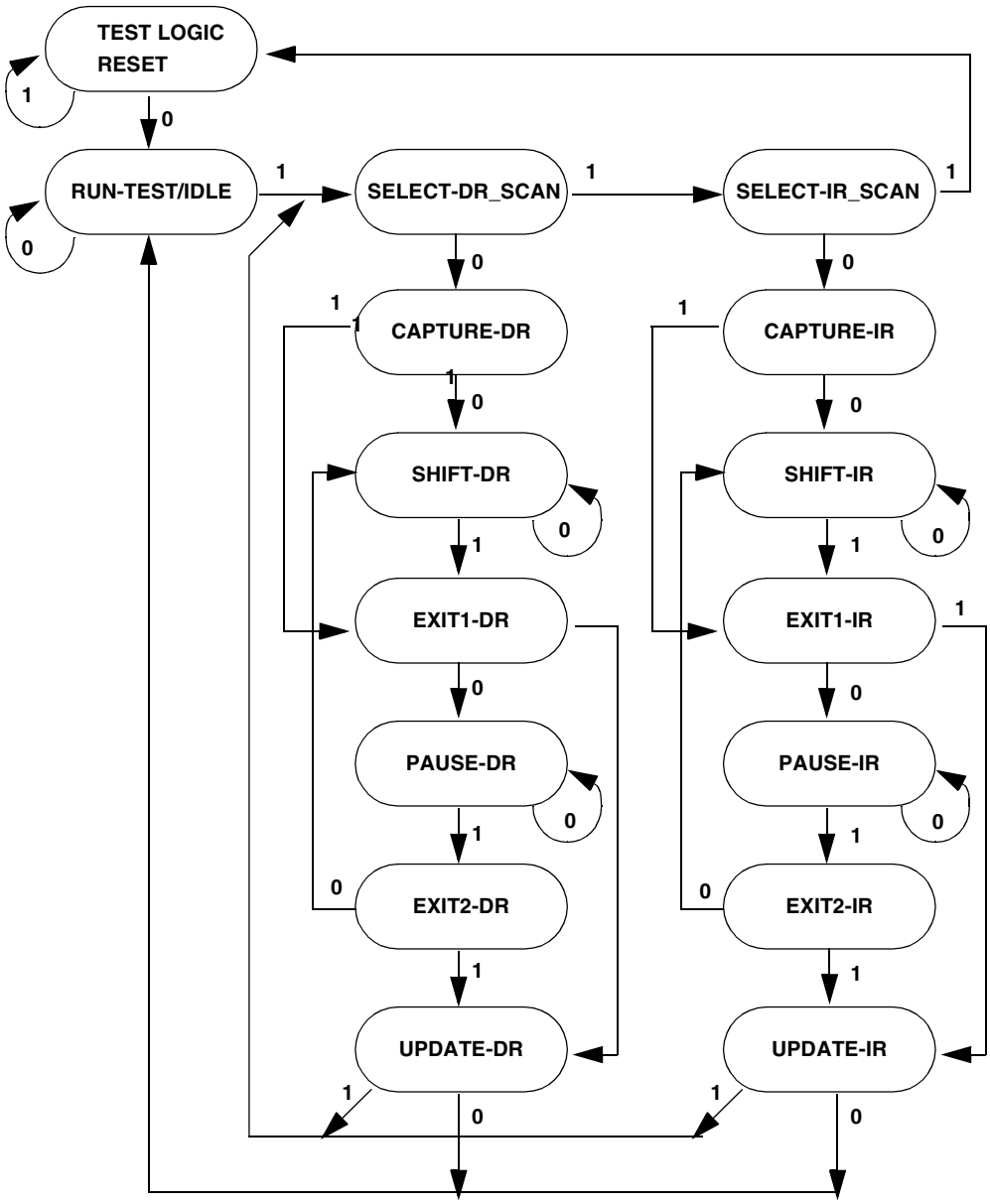


Figure 22-3 TAP Controller State Machine

22.5 Instruction Register

The MPC555 / MPC556 JTAG implementation includes the public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS), and also supports the CLAMP instruction. One additional public instruction (HI-Z) provides the capability for disabling all device output drivers. The MPC555 / MPC556 includes a 4-bit instruction register without parity consisting of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The four bits are used to decode the five unique instructions listed in **Table 22-2**.

Table 22-2 Instruction Decoding

Code				Instruction
B3	B2	B1	B0	
0	0	0	0	EXTEST
0	0	0	1	SAMPLE/PRELOAD
0	X	1	X	BYPASS
0	1	0	0	HI-Z
0	1	0	1	CLAMP and BYPASS

The parallel output of the instruction register is reset to all ones in the test-logic-reset controller state. Note that this preset state is equivalent to the BYPASS instruction.

During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with the CLAMP command code.

22.5.1 EXTEST

The external test (EXTEST) instruction selects the 346-bit boundary scan register. EXTEST also asserts internal reset for the MPC555 / MPC556 system logic to force a predictable beginning internal state while performing external boundary scan operations.

By using the TAP, the register is capable of scanning user-defined values into the output buffers, capturing values presented to input pins and controlling the output drive of three-state output or bi-directional pins. For more details on the function and use of EXTEST, refer to the scan chain document.

22.5.2 SAMPLE/PRELOAD

The SAMPLE/PRELOAD instruction initializes the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction. The SAMPLE/PRELOAD instruction also provides a means to obtain a snapshot of system data and control signals.

NOTE

Since there is no internal synchronization between the scan chain clock (TCK) and the system clock (CLKOUT), the user must provide some form of external synchronization to achieve meaningful results.

22.5.3 BYPASS

The BYPASS instruction selects the single-bit bypass register as shown in **Figure 22-4**. This creates a shift register path from TDI to the bypass register and, finally, to TDO, circumventing the 463-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the MPC555 / MPC556 becomes the device under test.

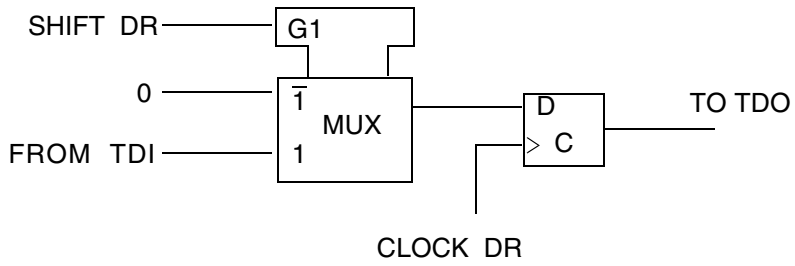


Figure 22-4 Bypass Register

When the bypass register is selected by the current instruction, the shift register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

22.5.4 CLAMP

The CLAMP instruction selects the single-bit bypass register as shown in **Figure 22-4**, and the state of all signals driven from system output pins is completely defined by the data previously shifted into the boundary scan register (e.g., using the SAMPLE/PRELOAD instruction).

22.5.5 HI-Z

The HI-Z instruction is provided as a manufacturer's optional public instruction to prevent having to backdrive the output pins during circuit-board testing. When HI-Z is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register.

22.6 Restrictions

The MPC555 / MPC556 provides flexible control of external signals using the boundary scan register and EXTEST or CLAMP instructions. As a result, the circuit board test environment must be designed to avoid signal contention which may result in device destruction.

22.7 Low-Power Stop Mode

The MPC555 / MPC556 features a low-power stop mode. The interaction of the scan chain interface with low-power stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller in the test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.

2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to VDD or ground, although TCK pin is internally connected to ground.
3. The TMS, TDI and $\overline{\text{TRST}}$ pins include on-chip pullup resistors. In low-power stop mode, these three pins should remain either unconnected or connected to VDD to achieve minimal power consumption.

NOTE

For proper reset of the scan chain test logic, the best approach is to assert $\overline{\text{TRST}}$ at power on reset ($\overline{\text{PORESET}}$).

22.8 Non-IEEE 1149.1-1990 Operation

In non-IEEE 1149.1-1990 operation, the IEEE 1149.1-1990 test logic must be kept transparent to the system logic by forcing and holding the TAP controller into the **test-logic-reset** controller state. There are two methods of forcing and holding the controller to this state. The first is to assert the $\overline{\text{TRST}}$ signal, forcing the TAP into the **test-logic-reset** controller state. The second is to provide at least five TCK pulses with TMS held high.

The best approach is to connect a pull down resistor to $\overline{\text{TRST}}$, or to connect it to $\overline{\text{PORESET}}$ with a resistor. If bounday scan is required, the JTAG controller should drive $\overline{\text{TRST}}$ to the negated state (“1” value) following $\overline{\text{PORESET}}$.

22.9 Boundary Scan Register

The MPC555 / MPC556 scan chain implementation has a 346-bit boundary scan register. This register contains bits for all device signal and clock pins and associated control signals. The XTAL, EXTAL and XFC pins are associated with analog signals and are not included in the boundary scan register.

An IEEE-1149.1 compliant boundary scan register has been included on the MPC555 / MPC556. This 346-bit boundary scan register can be connected between TDI and TDO when EXTEST or SAMPLE/PRELOAD instructions are selected. This register is used for capturing signal pin data on the input pins, forcing fixed values on the output signal pins, and selecting the direction and drive characteristics (a logic value or high impedance) of the bidirectional and three-state signal pins. **Figure 22-5** through **Figure 22-8** depict the various cell types.

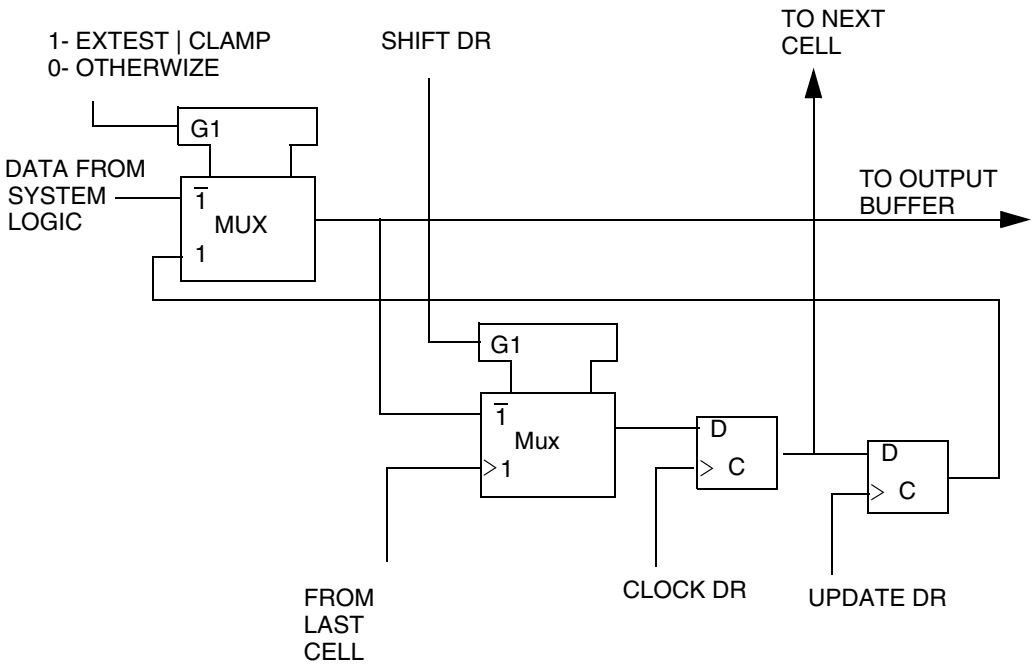


Figure 22-5 Output Pin Cell (O.pin)

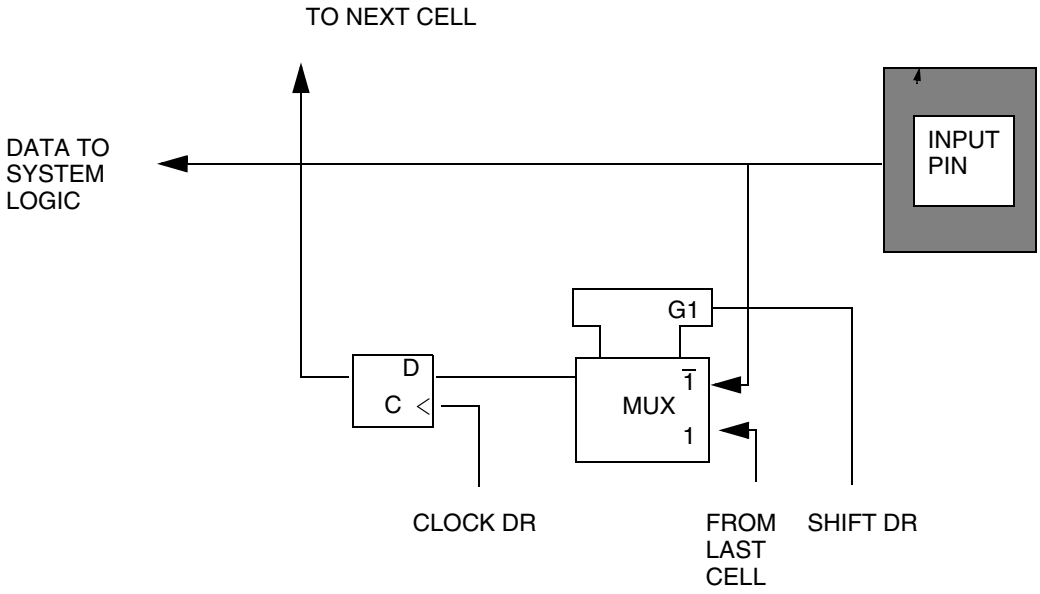


Figure 22-6 Observe-Only Input Pin Cell (I.Obs)

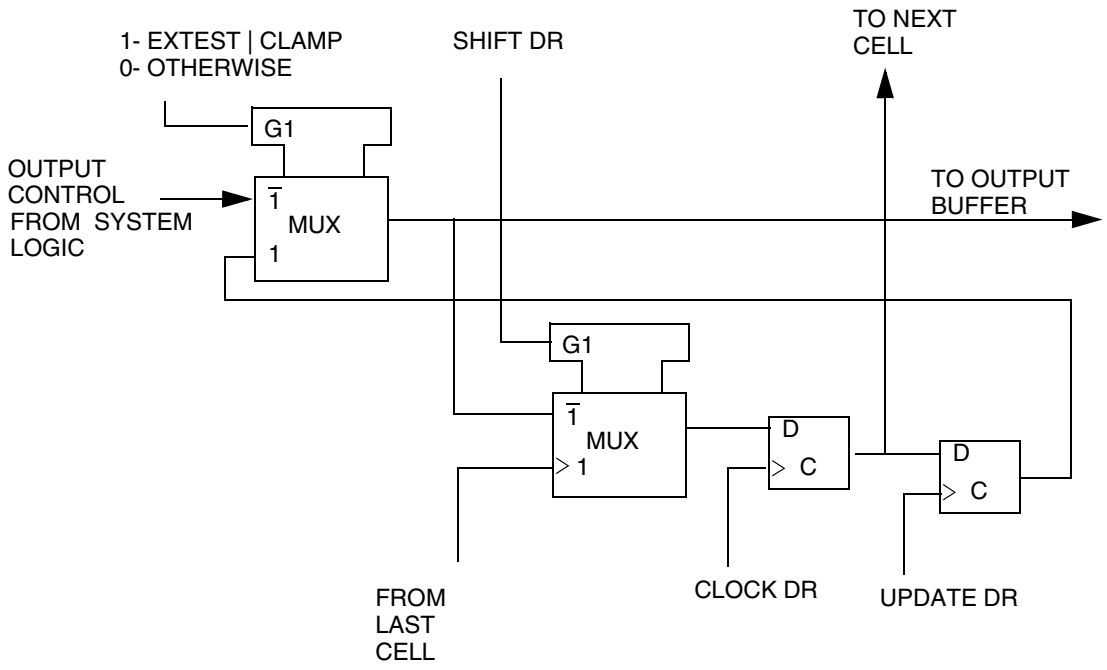


Figure 22-7 Output Control Cell (IO.CTL)

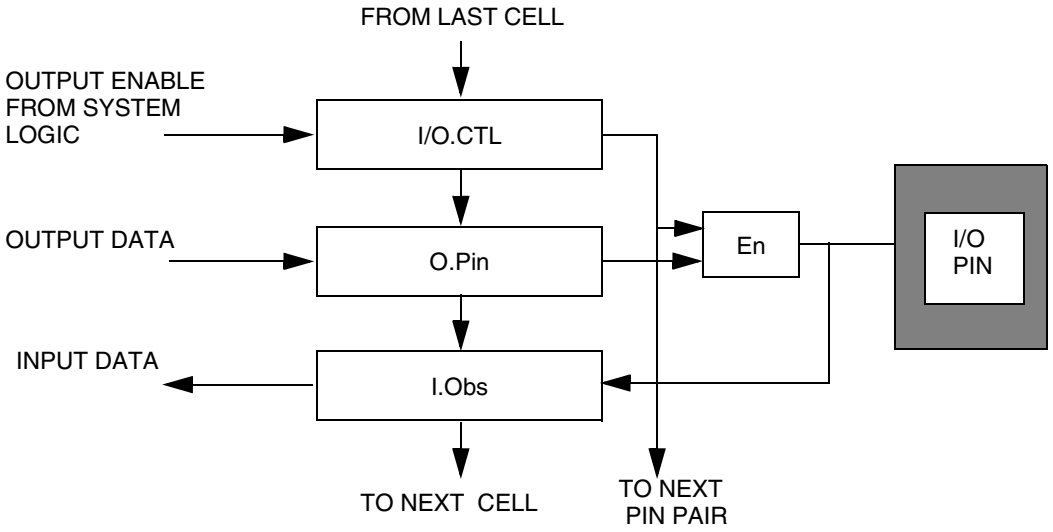


Figure 22-8 General Arrangement of Bidirectional Pin Cells

The key to using the boundary scan register is knowing the boundary scan bit order and the pins that are associated with them. Below in **Table 22-3** is the bit order starting

from the TDI input and going to the TDO output. This table uses the “long” pin names of the MPC555 / MPC556. See **Table 2-6** for a translation of the long names to the short names found on the pinout information in **Figure 2-2**.

The first column in the table defines the bit’s ordinal position in the boundary scan register. The shift register cell nearest TDI (i.e., first to be shifted in) is defined as bit 1; the last bit to be shifted in is 345.

The second column references one of the three MPC555 / MPC556 cell types depicted in **Figure 22-5** through **Figure 22-8**, which describe the cell structure for each type.

The third column lists the pin name for all pin-related cells or defines the name of bidirectional control register bits. The fourth column lists the pin type. The last column indicates the associated boundary scan register control bit for bi-directional output pins.

Bi-directional pins include two scan cells for data (IO.Cell) as depicted in **Figure 22-8**. These bits are controlled by the cell shown in **Figure 22-7**. The value of the control bit controls the output function of the bidirectional pin. One or more bidirectional data cells can be serially connected to a control cell.

Table 22-3 Boundary Scan Bit Definition

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
0	i.obs	b_cnrx0	i	—
1	o.pin	b_cntx0	o	—
2	IO.PIN	b_tpuch0	io	g251.ctl
3	IO.ctl	g251.ctl	—	—
4	IO.PIN	b_tpuch1	io	g252.ctl
5	IO.ctl	g252.ctl	—	—
6	IO.PIN	b_tpuch2	io	g253.ctl
7	IO.ctl	g253.ctl	—	—
8	IO.PIN	b_tpuch3	io	g254.ctl
9	IO.ctl	g254.ctl	—	—
10	IO.PIN	b_tpuch4	io	g255.ctl
11	IO.ctl	g255.ctl	—	—
12	IO.PIN	b_tpuch5	io	g256.ctl
13	IO.ctl	g256.ctl	—	—
14	IO.PIN	b_tpuch6	io	g257.ctl
15	IO.ctl	g257.ctl	—	—
16	IO.PIN	b_tpuch7	io	g258.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
17	IO.ctl	g258.ctl	—	—
18	IO.PIN	b_tpuch8	io	g259.ctl
19	IO.ctl	g259.ctl	—	—
20	IO.PIN	b_tpuch9	io	g260.ctl
21	IO.ctl	g260.ctl	—	—
22	IO.PIN	b_tpuch10	io	g261.ctl
23	IO.ctl	g261.ctl	—	—
24	IO.PIN	b_tpuch11	io	g262.ctl
25	IO.ctl	g262.ctl	—	—
26	IO.PIN	b_tpuch12	io	g263.ctl
27	IO.ctl	g263.ctl	—	—
28	IO.PIN	b_tpuch13	io	g264.ctl
29	IO.ctl	g264.ctl	—	—
30	IO.PIN	b_tpuch14	io	g265.ctl
31	IO.ctl	g265.ctl	—	—
32	IO.PIN	b_tpuch15	io	g266.ctl
33	IO.ctl	g266.ctl	—	—
34	IO.PIN	b_t2clk	io	g267.ctl
35	IO.ctl	g267.ctl	—	—
36	IO.PIN	a_t2clk	io	g268.ctl
37	IO.ctl	g268.ctl	—	—
38	IO.PIN	a_tpuch0	io	g269.ctl
39	IO.ctl	g269.ctl	—	—
40	IO.PIN	a_tpuch1	io	g302.ctl
41	IO.ctl	g302.ctl	—	—
42	IO.PIN	a_tpuch2	io	g303.ctl
43	IO.ctl	g303.ctl	—	—
44	IO.PIN	a_tpuch3	io	g304.ctl
45	IO.ctl	g304.ctl	—	—
46	IO.PIN	a_tpuch4	io	g305.ctl
47	IO.ctl	g305.ctl	—	—

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
48	IO.PIN	a_tpuch5	io	g306.ctl
49	IO.ctl	g306.ctl	—	—
50	IO.PIN	a_tpuch6	io	g307.ctl
51	IO.ctl	g307.ctl	—	—
52	IO.PIN	a_tpuch7	io	g308.ctl
53	IO.ctl	g308.ctl	—	—
54	IO.PIN	a_tpuch8	io	g309.ctl
55	IO.ctl	g309.ctl	—	—
56	IO.PIN	a_tpuch9	io	g310.ctl
57	IO.ctl	g310.ctl	—	—
58	IO.PIN	a_tpuch10	io	g311.ctl
59	IO.ctl	g311.ctl	—	—
60	IO.PIN	a_tpuch11	io	g312.ctl
61	IO.ctl	g312.ctl	—	—
62	IO.PIN	a_tpuch12	io	g313.ctl
63	IO.ctl	g313.ctl	—	—
64	IO.PIN	a_tpuch13	io	g314.ctl
65	IO.ctl	g314.ctl	—	—
66	IO.PIN	a_tpuch14	io	g315.ctl
67	IO.ctl	g315.ctl	—	—
68	IO.PIN	a_tpuch15	io	g316.ctl
69	IO.ctl	g316.ctl	—	—
70	i.obs	a_an0_anw_pqb0	i	—
71	i.obs	a_an1_anx_pqb1	i	—
72	i.obs	a_an2_any_pqb2	i	—
73	i.obs	a_an3_anz_pqb3	i	—
74	i.obs	a_an48_pqb4	i	—
75	i.obs	a_an49_pqb5	i	—
76	IO.PIN	a_an50_pqb6	io	g333.ctl
77	IO.ctl	g333.ctl	—	—
78	IO.PIN	a_an51_pqb7	io	g334.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
79	IO.ctl	g334.ctl	—	—
80	IO.PIN	a_an52_ma0_pqa0	io	g335.ctl
81	IO.ctl	g335.ctl	—	—
82	IO.PIN	a_an53_ma1_pqa1	io	g336.ctl
83	IO.ctl	g336.ctl	—	—
84	IO.PIN	a_an54_ma2_pqa2	io	g337.ctl
85	IO.ctl	g337.ctl	—	—
86	IO.PIN	a_an55_pqa3	io	g338.ctl
87	IO.ctl	g338.ctl	—	—
88	IO.PIN	a_an56_pqa4	io	g339.ctl
89	IO.ctl	g339.ctl	—	—
90	IO.PIN	a_an57_pqa5	io	g340.ctl
91	IO.ctl	g340.ctl	—	—
92	IO.PIN	a_an58_pqa6	io	g341.ctl
93	IO.ctl	g341.ctl	—	—
94	IO.PIN	a_an59_pqa7	io	g342.ctl
95	IO.ctl	g342.ctl	—	—
96	i.obs	b_an0_anw_pqb0	i	—
97	i.obs	b_an1_anx_pqb1	i	—
98	i.obs	b_an2_any_pqb2	i	—
99	i.obs	b_an3_anz_pqb3	i	—
100	i.obs	b_an48_pqb4	i	—
101	i.obs	b_an49_pqb5	i	—
102	IO.PIN	b_an50_pqb6	io	g349.ctl
103	IO.ctl	g349.ctl	—	—
104	IO.PIN	b_an51_pqb7	io	g350.ctl
105	IO.ctl	g350.ctl	—	—
106	IO.PIN	b_an52_ma0_pqa0	io	g351.ctl
107	IO.ctl	g351.ctl	—	—
108	IO.PIN	b_an53_ma1_pqa1	io	g352.ctl
109	IO.ctl	g352.ctl	—	—

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
110	IO.PIN	b_an54_ma2_pqa2	io	g353.ctl
111	IO.ctl	g353.ctl	—	—
112	IO.PIN	b_an55_pqa3	io	g354.ctl
113	IO.ctl	g354.ctl	—	—
114	IO.PIN	b_an56_pqa4	io	g355.ctl
115	IO.ctl	g355.ctl	—	—
116	IO.PIN	b_an57_pqa5	io	g356.ctl
117	IO.ctl	g356.ctl	—	—
118	IO.PIN	b_an58_pqa6	io	g357.ctl
119	IO.ctl	g357.ctl	—	—
120	IO.PIN	b_an59_pqa7	io	g358.ctl
121	IO.ctl	g358.ctl	—	—
122	i.obs	etrig2	i	—
123	i.obs	etrig1	i	—
124	IO.PIN	mda11	io	g365.ctl
125	IO.ctl	g365.ctl	—	—
126	IO.PIN	mda12	io	g366.ctl
127	IO.ctl	g366.ctl	—	—
128	IO.PIN	mda13	io	g367.ctl
129	IO.ctl	g367.ctl	—	—
130	IO.PIN	mda14	io	g368.ctl
131	IO.ctl	g368.ctl	—	—
132	IO.PIN	mda15	io	g369.ctl
133	IO.ctl	g369.ctl	—	—
134	IO.PIN	mda27	io	g370.ctl
135	IO.ctl	g370.ctl	—	—
136	IO.PIN	mda28	io	g371.ctl
137	IO.ctl	g371.ctl	—	—
138	IO.PIN	mda29	io	g372.ctl
139	IO.ctl	g372.ctl	—	—
140	IO.PIN	mda30	io	g403.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
141	IO.ctl	g403.ctl	—	—
142	IO.PIN	mda31	io	g404.ctl
143	IO.ctl	g404.ctl	—	—
144	IO.PIN	mpwm0	io	g405.ctl
145	IO.ctl	g405.ctl	—	—
146	IO.PIN	mpwm1	io	g406.ctl
147	IO.ctl	g406.ctl	—	—
148	IO.PIN	mpwm2	io	g407.ctl
149	IO.ctl	g407.ctl	—	—
150	IO.PIN	mpwm3	io	g408.ctl
151	IO.ctl	g408.ctl	—	—
152	IO.PIN	mpwm16	io	g409.ctl
153	IO.ctl	g409.ctl	—	—
154	IO.PIN	mpwm17	io	g410.ctl
155	IO.ctl	g410.ctl	—	—
156	IO.PIN	mpwm18	io	g411.ctl
157	IO.ctl	g411.ctl	—	—
158	IO.PIN	mpwm19	io	g412.ctl
159	IO.ctl	g412.ctl	—	—
160	IO.PIN	mpio32b5	io	g413.ctl
161	IO.ctl	g413.ctl	—	—
162	IO.PIN	mpio32b6	io	g414.ctl
163	IO.ctl	g414.ctl	—	—
164	IO.PIN	mpio32b7	io	g415.ctl
165	IO.ctl	g415.ctl	—	—
166	IO.PIN	mpio32b8	io	g416.ctl
167	IO.ctl	g416.ctl	—	—
168	IO.PIN	mpio32b9	io	g417.ctl
169	IO.ctl	g417.ctl	—	—
170	IO.PIN	mpio32b10	io	g418.ctl
171	IO.ctl	g418.ctl	—	—

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
172	IO.PIN	mpio32b11	io	g419.ctl
173	IO.ctl	g419.ctl	—	—
174	IO.PIN	mpio32b12	io	g420.ctl
175	IO.ctl	g420.ctl	—	—
176	IO.PIN	mpio32b13	io	g421.ctl
177	IO.ctl	g421.ctl	—	—
178	IO.PIN	mpio32b14	io	g422.ctl
179	IO.ctl	g422.ctl	—	—
180	IO.PIN	mpio32b15	io	g423.ctl
181	IO.ctl	g423.ctl	—	—
182	IO.PIN	vf0_mpio32b0	io	g424.ctl
183	IO.ctl	g424.ctl	—	—
184	IO.PIN	vf1_mpio32b1	io	g425.ctl
185	IO.ctl	g425.ctl	—	—
186	IO.PIN	vf2_mpio32b2	io	g426.ctl
187	IO.ctl	g426.ctl	—	—
188	IO.PIN	vfls0_mpio32b3	io	g427.ctl
189	IO.ctl	g427.ctl	—	—
190	IO.PIN	vfls1_mpio32b4	io	g428.ctl
191	IO.ctl	g428.ctl	—	—
192	o.pin	a_cntx0	o	—
193	i.obs	a_cnrx0	i	—
194	IO.PIN	pcs0_ss_b_qgpio0	io	g435.ctl
195	IO.ctl	g435.ctl	—	—
196	IO.PIN	pcs1_qgpio1	io	g436.ctl
197	IO.ctl	g436.ctl	—	—
198	IO.PIN	pcs2_qgpio2	io	g437.ctl
199	IO.ctl	g437.ctl	—	—
200	IO.PIN	pcs3_qgpio3	io	g438.ctl
201	IO.ctl	g438.ctl	—	—
202	IO.PIN	miso_qgpio4	io	g439.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
203	IO.ctl	g439.ctl	—	—
204	IO.PIN	mosi_qgpio5	io	g440.ctl
205	IO.ctl	g440.ctl	—	—
206	IO.PIN	sck_qgpio6	io	g441.ctl
207	IO.ctl	g441.ctl	—	—
208	i.obs	eck	i	—
209	o.pin	txd1_qgpo1	o	—
210	o.pin	txd2_qgpo2	o	—
211	i.obs	rxd1_qgpi1	i	—
212	i.obs	rxd2_qgpi2	i	—
213	i.obs	epee	i	—
214	o.pin	engclk_buclk	o	—
215	i.obs	extclk	i	—
216	o.pin	clkout	o	—
217	i.obs	poreset_b	i	—
218	i.obs	sreset_b	io	—
219	o.pin	sreset_b	io	g465.ctl
220	IO.ctl	g465.ctl	—	—
221	i.obs	hreset_b	io	—
222	o.pin	hreset_b	io	g466.ctl
223	IO.ctl	g466.ctl	—	—
224	IO.PIN	rstconf_b_texp	io	g467.ctl
225	IO.ctl	g467.ctl	—	—
226	i.obs	irq7_b_modck3	i	—
227	i.obs	irq6_b_modck2	i	—
228	IO.PIN	irq5_b_sgpioc5_modck1	io	g503.ctl
229	IO.ctl	g503.ctl	—	—
230	IO.PIN	data_sgpiod[16]	io	g112.ctl
231	IO.ctl	g112.ctl	—	—
232	IO.PIN	data_sgpiod[17]	io	g112.ctl
233	IO.PIN	data_sgpiod[18]	io	g112.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
234	IO.PIN	data_sgpiod[14]	io	g111.ctl
235	IO.PIN	data_sgpiod[15]	io	g111.ctl
236	IO.PIN	data_sgpiod[19]	io	g112.ctl
237	IO.PIN	data_sgpiod[20]	io	g112.ctl
238	IO.PIN	data_sgpiod[12]	io	g111.ctl
239	IO.PIN	data_sgpiod[13]	io	g111.ctl
240	IO.PIN	data_sgpiod[21]	io	g112.ctl
241	IO.PIN	data_sgpiod[10]	io	g111.ctl
242	IO.PIN	data_sgpiod[11]	io	g111.ctl
243	IO.PIN	data_sgpiod[22]	io	g112.ctl
244	IO.PIN	data_sgpiod[23]	io	g112.ctl
245	IO.PIN	data_sgpiod[8]	io	g111.ctl
246	IO.ctl	g111.ctl	—	—
247	IO.PIN	data_sgpiod[9]	io	g111.ctl
248	IO.PIN	data_sgpiod[24]	io	g524.ctl
249	IO.ctl	g524.ctl	—	—
250	IO.PIN	data_sgpiod[25]	io	g525.ctl
251	IO.ctl	g525.ctl	—	—
252	IO.PIN	data_sgpiod[6]	io	g110.ctl
253	IO.PIN	data_sgpiod[7]	io	g110.ctl
254	IO.PIN	data_sgpiod[26]	io	g528.ctl
255	IO.ctl	g528.ctl	—	—
256	IO.PIN	data_sgpiod[27]	io	g529.ctl
257	IO.ctl	g529.ctl	—	—
258	IO.PIN	data_sgpiod[4]	io	g110.ctl
259	IO.PIN	data_sgpiod[5]	io	g110.ctl
260	IO.PIN	data_sgpiod[28]	io	g534.ctl
261	IO.ctl	g534.ctl	—	—
262	IO.PIN	data_sgpiod[29]	io	g535.ctl
263	IO.ctl	g535.ctl	—	—
264	IO.PIN	data_sgpiod[2]	io	g110.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
265	IO.PIN	data_sgpiod[3]	io	g110.ctl
266	IO.PIN	data_sgpiod[30]	io	g540.ctl
267	IO.ctl	g540.ctl	—	—
268	IO.PIN	data_sgpiod[0]	io	g110.ctl
269	IO.ctl	g110.ctl	—	—
270	IO.PIN	data_sgpiod[1]	io	g110.ctl
271	IO.PIN	data_sgpiod[31]	io	g545.ctl
272	IO.ctl	g545.ctl	—	—
273	IO.PIN	addr_sgpioa[29]	io	g102.ctl
274	IO.PIN	addr_sgpioa[25]	io	g102.ctl
275	IO.PIN	addr_sgpioa[26]	io	g102.ctl
276	IO.PIN	addr_sgpioa[27]	io	g102.ctl
277	IO.PIN	addr_sgpioa[28]	io	g102.ctl
278	IO.PIN	addr_sgpioa[24]	io	g102.ctl
279	IO.ctl	g102.ctl	—	—
280	IO.PIN	addr_sgpioa[23]	io	g101.ctl
281	IO.PIN	addr_sgpioa[22]	io	g101.ctl
282	IO.PIN	addr_sgpioa[30]	io	g102.ctl
283	IO.PIN	addr_sgpioa[21]	io	g101.ctl
284	IO.PIN	addr_sgpioa[20]	io	g101.ctl
285	IO.PIN	addr_sgpioa[8]	io	g100.ctl
286	IO.ctl	g100.ctl	—	—
287	IO.PIN	addr_sgpioa[31]	io	g102.ctl
288	IO.PIN	addr_sgpioa[19]	io	g101.ctl
289	IO.PIN	addr_sgpioa[18]	io	g101.ctl
290	IO.PIN	addr_sgpioa[9]	io	g100.ctl
291	IO.PIN	addr_sgpioa[17]	io	g101.ctl
292	IO.PIN	addr_sgpioa[16]	io	g101.ctl
293	IO.ctl	g101.ctl	—	—
294	IO.PIN	addr_sgpioa[10]	io	g100.ctl
295	IO.PIN	addr_sgpioa[15]	io	g100.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
296	IO.PIN	addr_sgpioa[14]	io	g100.ctl
297	IO.PIN	addr_sgpioa[13]	io	g100.ctl
298	IO.PIN	addr_sgpioa[11]	io	g100.ctl
299	IO.PIN	addr_sgpioa[12]	io	g100.ctl
300	IO.PIN	bi_b_sts_b	io	g205.ctl
301	IO.ctl	g205.ctl	—	—
302	IO.PIN	burst_b	io	g131.ctl
303	IO.PIN	bdip_b	io	g207.ctl
304	IO.ctl	g207.ctl	—	—
305	IO.PIN	ta_b	io	g208.ctl
306	IO.ctl	g208.ctl	—	—
307	IO.PIN	ts_b	io	g131.ctl
308	IO.PIN	tsiz1	io	g130.ctl
309	IO.PIN	tsiz0	io	g130.ctl
310	IO.ctl	g130.ctl	—	—
311	IO.PIN	tea_b	io	g214.ctl
312	IO.ctl	g214.ctl	—	—
313	o.pin	oe_b	o	—
314	IO.PIN	rd_wr_b	io	g131.ctl
315	IO.ctl	g131.ctl	—	—
316	o.pin	cs3_b	o	—
317	o.pin	cs2_b	o	—
318	o.pin	cs1_b	o	—
319	o.pin	cs0_b	o	—
320	o.pin	we_b_at[3]	o	—
321	o.pin	we_b_at[2]	o	—
322	o.pin	we_b_at[1]	o	—
323	o.pin	we_b_at[0]	o	—
324	IO.PIN	br_b_vf1_iwp2	io	g227.ctl
325	IO.ctl	g227.ctl	—	—
326	IO.PIN	bg_b_vf0_lwp1	io	g228.ctl

Table 22-3 Boundary Scan Bit Definition (Continued)

Bit	Cell Type	Pin/Cell Name	Pin Type	Output CTL Cell
327	IO.ctl	g228.ctl	—	—
328	IO.PIN	bb_b_vf2_iwp3	io	g229.ctl
329	IO.ctl	g229.ctl	—	—
330	IO.PIN	sgpioc7_irqout_b_lwp0	io	g230.ctl
331	IO.ctl	g230.ctl	—	—
332	IO.PIN	irq1_b_rsv_b_sgpioc1	io	g231.ctl
333	IO.ctl	g231.ctl	—	—
334	IO.PIN	irq0_b_sgpioc0	io	g232.ctl
335	IO.ctl	g232.ctl	—	—
336	IO.PIN	irq2_b_cr_b_sgpioc2	io	g233.ctl
337	IO.ctl	g233.ctl	—	—
338	IO.PIN	irq4_b_at2_sgpioc4	io	g234.ctl
339	IO.ctl	g234.ctl	—	—
340	IO.PIN	irq3_b_kr_b_retry_b_sgpio c3	io	g237.ctl
341	IO.ctl	g237.ctl	—	—
342	o.pin	iwp0_vfls0	o	—
343	o.pin	iwp1_vfls1	o	—
344	IO.PIN	sgpioc6_frz_ptr_b	io	g240.ctl
345	IO.ctl	g240.ctl	—	—

APPENDIX A
MPC555 / MPC556 INTERNAL MEMORY MAP

The tables below use the following notations.

In the Access column:

S = Supervisor Access Only

U = User Access

T = Test Access

In the Reset column:

S = $\overline{\text{SRESET}}$

H = $\overline{\text{HRESET}}$

M = $\overline{\text{Module Reset}}$

POR = $\overline{\text{Power-On Reset}}$

U = Unchanged

X = Unknown

The codes in the Reset column indicate which reset has an effect on register values.

INDEX of MEMORY MAP TABLES

Table A-1 SPR (Special Purpose Registers)

Table A-2 CMF (CDR MoneT Flash EEPROM) Flash Array

Table A-3 USIU (Unified System Interface Unit)

Table A-4 CMF (CDR MoneT Flash EEPROM)

Table A-5 DPTRAM (Dual-Port TPU RAM)

Table A-6 DPTRAM Array

Table A-7 TPU3 (Time Processor Unit)

Table A-8 QADC64 (Queued Analog-to-Digital Converter)

Table A-9 QSMCM (Queued Serial Multi-Channel Module)

Table A-10 MIOS1 (Modular Input/Output Subsystem)

Table A-11 TouCAN (CAN 2.0B Controller)

Table A-12 UIMB (U-Bus to IMB3 Bus Interface)

Table A-13 SRAM (Static RAM Access Memory)

Table A-14 SRAM (Static RAM Access Memory) Array

Table A-1 SPR (Special Purpose Registers)

Address	Access	Symbol	Register	Size	Reset
MSR	S	MSR	Machine State Register. See **Table 3-12** for bit descriptions	32	S
SPR 1	U	XER	Integer Exception Register. See **Table 3-10** for bit descriptions	32	U
SPR 8	U	LR	Link Register. See **3.7.6 Link Register (LR)** for bit descriptions.	32	U
SPR 9	U	CTR	Count Register. See **3.7.7 Count Register (CTR)** for bit descriptions.	32	U
SPR 18	S	DSISR	DAE/Source Instruction Service Register. See **3.9.2 DAE/Source Instruction Service Register (DSISR)** for bit descriptions.	32	U
SPR 19	S	DAR	Data Address Register. See **3.9.3 Data Address Register (DAR)** for bit descriptions.	32	U
SPR 22	S	DEC	Decrementer Register. See **3.9.5 Decrementer Register (DEC)** for bit descriptions.	32	U
SPR 26	S	SRR0	Machine Status Save/Restore Register 0. See **3.9.6 Machine Status Save/Restore Register 0 (SRR0)** for bit descriptions.	32	U
SPR 27	S	SRR1	Machine Status Save/Restore Register 1. See **3.9.7 Machine Status Save/Restore Register 1 (SRR1)** for bit descriptions.	32	U
SPR 80	S	EIE	External Interrupt Enable Register See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.	32	—
SPR 81	S	EID	External Interrupt Disable Register See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.	32	—
SPR 82	S	NRI	Non-Recoverable Interrupt Register See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.	32	—
SPR 144	S	CMPA	Comparator A Value Register. See **Table 21-17** for bit descriptions.	32	U
SPR 145	S	CMPB	Comparator B Value Register. See **Table 21-17** for bit descriptions.	32	U
SPR 146	S	CMPC	Comparator C Value Register. See **Table 21-17** for bit descriptions.	32	U
SPR 147	S	CMPD	Comparator D Value Register. See **Table 21-17** for bit descriptions.	32	U
SPR 148	S	ECR	Exception Cause Register See **Table 21-27** for bit descriptions.	32	S
SPR 149	S	DER	Debug Enable Register See **Table 21-28** for bit descriptions.	32	S
SPR 150	S	COUNTA	Breakpoint Counter A Value and Control Register See **Table 21-25** for bit descriptions.	32	U

Table A-1 SPR (Special Purpose Registers) (Continued)

Address	Access	Symbol	Register	Size	Reset
SPR 151	S	COUNTB	Breakpoint Counter B Value and Control Register See **Table 21-26** for bit descriptions.	32	U
SPR 152	S	CMPE	Comparator G Value Register. See **Table 21-20** for bit descriptions.	32	U
SPR 153	S	CMPF	Comparator H Value Register. See **Table 21-20** for bit descriptions.	32	U
SPR 154	S	CMPG	Comparator E Value Register. See **Table 21-18** for bit descriptions.	32	U
SPR 155	S	CMPH	Comparator F Value Register. See **Table 21-18** for bit descriptions.	32	U
SPR 156	S	LCTRL1	L-Bus Support Control Register 1 See **Table 21-23** for bit descriptions.	32	S
SPR 157	S	LCTRL2	L-Bus Support Control Register 2 See **Table 21-24** for bit descriptions.	32	S
SPR 158	S	ICTRL	I-Bus Support Control Register. See **Table 21-21** for bit descriptions.	32	S
SPR 159	S	BAR	Breakpoint Address Register. See **Table 21-19** for bit descriptions.	32	U
SPR 268, 269	U read only	TB	Time Base (Read Only). See **Table 3-11** for bit descriptions.	64	U
SPR 272	S	SPRG0	General Special Purpose Registers. See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.	32	U
SPR 273	S	SPRG1	General Special Purpose Registers. See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.	32	U
SPR 274	S	SPRG2	General Special Purpose Registers. See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.	32	U
SPR 275	S	SPRG3	General Special Purpose Registers. See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.	32	U
SPR 284, 285	S write only	TB	Time Base Register (Write Only). See **Table 3-14** for bit descriptions.	64	U
SPR 287	S read only	PVR	Processor Version Registers. See **Table 3-16** for bit descriptions.	32	U
SPR 528	S	MI_GRA	Global Region Attribute Register. See **Table 4-7** for bit descriptions.	32	H
SPR 536	S	L2U_GRA	L2U Global Region Attribute Register. See **Table 11-10** for bit descriptions.	32	POR, H
SPR 560	S	BBCMCR	BBC Module Configuration Register. See **Table 4-8** for bit descriptions.	32	U
SPR 568	U	L2U_MCR	L2U Module Configuration Register. See **Table 11-7** for bit descriptions.	32	POR, H
SPR 630	S	DPDR	Development Port Data Register.	32	U
SPR 638	S	IMMR	Internal Memory Mapping Register. See **Table 6-11** for bit descriptions.	32	H

Table A-1 SPR (Special Purpose Registers) (Continued)

Address	Access	Symbol	Register	Size	Reset
SPR 784	S	MI_RBA0	Region Address Register 0. See **Table 4-5** for bit descriptions.	32	U
SPR 785	S	MI_RBA1	Region Address Register 1. See **Table 4-5** for bit descriptions.	32	U
SPR 786	S	MI_RBA2	Region Address Register 2. See **Table 4-5** for bit descriptions.	32	U
SPR 787	S	MI_RBA3	Region Address Register 3. See **Table 4-5** for bit descriptions.	32	U
SPR 792	S	L2U_RBA0	L2U Region 0 Address Register. See **Table 11-8** for bit descriptions.	32	POR, H
SPR 793	S	L2U_RBA1	L2U Region 1 Address Register. See **Table 11-8** for bit descriptions.	32	POR, H
SPR 794	S	L2U_RBA2	L2U Region 2 Address Register. See **Table 11-8** for bit descriptions.	32	POR, H
SPR 795	S	L2U_RBA3	L2U Region 3Address Register. See **Table 11-8** for bit descriptions.	32	POR, H
SPR 816	S	MI_RA0	Region Attribute Register 0. See **Table 4-6** for bit descriptions.	32	U
SPR 817	S	MI_RA1	Region Attribute Register 1. See **Table 4-6** for bit descriptions.	32	U
SPR 818	S	MI_RA2	Region Attribute Register 2. See **Table 4-6** for bit descriptions.	32	U
SPR 819	S	MI_RA3	Region Attribute Register 3. See **Table 4-6** for bit descriptions.	32	U
SPR 824	S	L2U_RA0	L2U Region 0 Attribute Register. See **Table 11-9** for bit descriptions.	32	POR, H
SPR 825	S	L2U_RA1	L2U Region 1 Attribute Register. See **Table 11-9** for bit descriptions.	32	POR, H
SPR 826	S	L2U_RA2	L2U Region 2 Attribute Register. See **Table 11-9** for bit descriptions.	32	POR, H
SPR 827	S	L2U_RA3	L2U Region 3 Attribute Register. See **Table 11-9** for bit descriptions.	32	POR, H
SPR 1022	S	FPECR	Floating-Point Exception Cause Register. See **3.9.10.2 Floating-Point Exception Cause Register (FPECR)** for bit descriptions.	32	S

Table A-2 CMF (CDR MoneT Flash EEPROM) Flash Array

Address	Access	Symbol	Register	Size	Reset
0x00 0000 – 0x03 FFFF			CMF_A Flash Array	8, 16, 32	—
0x04 0000 – 0x06 FFFF			CMF_B Flash Array	8, 16, 32	—

Table A-3 USIU (Unified System Interface Unit)

Address	Access	Symbol	Register	Size	Reset
0x2F C000	U[1]	SIUMCR	SIU Module Configuration Register. See **Table 6-5** for bit descriptions.	32	H
0x2F C004	U[2]	SYPCR	System Protection Control Register. See **Table 6-13** for bit descriptions.	32	H
0x2F C008	—	—	Reserved	—	—
0x2F C00E	U, write only	SWSR	Software Service Register. See **Table 6-14** for bit descriptions.	16	S
0x2F C010	U	SIPEND	Interrupt Pending Register. See **6.13.2.1 SIPEND Register** for bit descriptions.	32	S
0x2F C014	U	SIMASK	Interrupt Mask Register. See **6.13.2.2 SIU Interrupt Mask Register (SIMASK)** for bit descriptions.	32	S
0x2F C018	U	SIEL	Interrupt Edge Level Mask. See **6.13.2.3 SIU Interrupt Edge Level Register (SIEL)** for bit descriptions.	32	H
0x2F C01C	U, read only	SIVEC	Interrupt Vector. See **6.13.2.4 SIU Interrupt Vector Register** for bit descriptions.	32	—
0x2F C020	U	TESR	Transfer Error Status Register. See **Table 6-15** for bit descriptions.	32	S
0x2F C024	U	SGPIODT1	USIU General-Purpose I/O Data Register 1. See **Table 6-21** for bit descriptions.	32	H
0x2F C028	U	SGPIODT2	USIU General-Purpose I/O Data Register 2. See **Table 6-22** for bit descriptions.	32	H
0x2F C02C	U	SGPIOCR	USIU General-Purpose I/O Control Register. See **Table 6-23** for bit descriptions.	32	H
0x2F C030	U	EMCR	External Master Mode Control Register. See **Table 6-12** for bit descriptions.	32	H
0x2F C03C	U	PDMCR	Pads Module Configuration Register. See **Table 2-3** for bit descriptions.	32	H
0x2F C040 – 0x2F C0FC	—	—	Reserved	—	—
Memory Controller Registers					
0x2F C100	U	BR0	Base Register 0. See **Table 10-7** for bit descriptions.	32	H
0x2F C104	U	OR0	Option Register 0. See **Table 10-8** for bit descriptions.	32	H
0x2F C108	U	BR1	Base Register 1. See **Table 10-7** for bit descriptions.	32	H
0x2F C10C	U	OR1	Option Register 1. See **Table 10-8** for bit descriptions.	32	H
0x2F C110	U	BR2	Base Register 2. See **Table 10-7** for bit descriptions.	32	H
0x2F C114	U	OR2	Option Register 2. See **Table 10-8** for bit descriptions.	32	H
0x2F C118	U	BR3	Base Register 3. See **Table 10-7** for bit descriptions.	32	H

Table A-3 USIU (Unified System Interface Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x2F C11C	U	OR3	Option Register 3. See **Table 10-8** for bit descriptions.	32	H
0x2F C120 – 0x2F C13C	—	—	Reserved	—	—
0x2F C140	U	DMBR	Dual-Mapping Base Register. See **Table 10-9** for bit descriptions.	32	H
0x2F C144	U	DMOR	Dual-Mapping Option Register. See **Table 10-10** for bit descriptions.	32	H
0x2F C148 – 0x2F C174	—	—	Reserved	—	—
0x2F C178	U	MSTAT	Memory Status. See **Table 10-6** for bit descriptions.	16	H
System Integration Timers					
0x2F C200	U[3]	TBSCR	Time Base Status and Control. See **Table 6-16** for bit descriptions.	16	H
0x2F C204	U[3]	TBREF0	Time Base Reference 0. See **6.13.4.3 Time Base Reference Registers** for bit descriptions.	32	U
0x2F C208	U[3]	TBREF1	Time Base Reference 1. See **6.13.4.3 Time Base Reference Registers** for bit descriptions.	32	U
0x2F C20C – 0x2F C21C	—	—	Reserved	—	—
0x2F C220	U[4]	RTCSC	Real Time Clock Status and Control. See **Table 6-17** for bit descriptions.	16	H
0x2F C224	U[4]	RTC	Real Time Clock. See **6.13.4.6 Real-Time Clock Register (RTC)** for bit descriptions.	32	U
0x2F C228	T[4]	RTSEC	Real Time Alarm Seconds, reserved.	32	—
0x2F C22C	U[4]	RTCAL	Real Time Alarm. See **6.13.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.	32	U
0x2F C230 – 0x2F C23C	—	—	Reserved	—	—
0x2F C240	U[3]	PISCR	PIT Status and Control. See **Table 6-18** for bit descriptions.	16	H
0x2F C244	U[3]	PITC	PIT Count. See **Table 6-19** for bit descriptions.	32 (half reserved)	U
0x2F C248	U, read only	PITR	PIT Register. See **Table 6-20** for bit descriptions.	32 (half reserved)	U
0x2F C24C – 0x2F C27C		—	Reserved	—	—
Clocks and Reset					
0x2F C280	U[2]	SCCR	System Clock Control Register. See **Table 8-9** for bit descriptions.	32	H

Table A-3 USIU (Unified System Interface Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x2F C284	U[3,5,6]	PLPRCR	PLL Low Power and Reset Control Register. See **Table 8-10** for bit descriptions.	32	H
0x2F C288	U[3]	RSR	Reset Status Register. See **Table 7-3** for bit descriptions.	16	POR
0x2F C28C	U	COLIR	Change of Lock Interrupt Register. See **Table 8-11** for bit descriptions.	16	U
0x2F C290	U	VSRMCR	VDDSRM Control Register. See **Table 8-12** for bit descriptions.	16	U
0x2F C294 – 0x2F C2FC	—	—	Reserved	—	—
System Integration Timer Keys					
0x2F C300	U	TBSCRK	Time Base Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C304	U	TBREF0K	Time Base Reference 0 Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C308	U	TBREF1K	Time Base Reference 1 Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C30C	U	TBK	Time Base and Decrementer Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C310 – 0x2F C31C	—	—	Reserved	—	—
0x2F C320	U	RTCSCK	Real-Time Clock Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C324	U	RTCK	Real-Time Clock Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C328	U	RTSECK	Real-Time Alarm Seconds Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C32C	U	RTCALK	Real-Time Alarm Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C330 – 0x2F C33C	—	—	Reserved	—	—
0x2F C340	U	PISCRIK	PIT Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C344	U	PITCK	PIT Count Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C348 – 0x2F C37C	—	—	Reserved	—	—
Clocks and Reset Keys					
0x2F C380	U	SCCRK	System Clock Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C384	U	PLPRCRK	PLL Low-Power and Reset Control Register Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C388	U	RSRK	Reset Status Register Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C38C – 0x2F C3FC		—	Reserved	—	—

NOTES:

1. Entire register is locked if bit 15 (DLK) is set.
2. Write once after power on reset (POR).
3. Must use the key register to unlock if it has been locked by a key register, see **8.9.3.2 Keep Alive Power Registers Lock Mechanism**.
4. Locked after Power on Reset (POR). A write of 0x55CCAA33 must performed to the key register to unlock. See **8.9.3.2 Keep Alive Power Registers Lock Mechanism**.
5. Can have bits 0:11 (MF bits) write-protected by setting bit 4 (MFPDL) in the SCCR register to 1. Bit 21 (CSRC) and bits 22:23 (LPM) can be locked by setting bit 5 (LPML) of the SCCR register to 1.
6. Bit 24 (CSR) is write-once after soft reset.

Table A-4 CMF (CDR MoneT Flash EEPROM)

Address	Access	Symbol	Register	Size	Reset
CMF_A					
0x2F C800	S[1]	CMFMCR	CMF_A EEPROM Configuration Register. See **Table 19-2** for bit descriptions.	32	POR, H
0x2F C804	S	CMFTST	CMF_A EEPROM Test Register. See **Table 19-3** for bit descriptions.	32	POR, H
0x2F C808	S	CMFCTL	CMF_A EEPROM High Voltage Control Register. See **Table 19-6** for bit descriptions.	32	POR, H
CMF_B					
0x2F C840	S[1]	CMFMCR	CMF_B EEPROM Configuration Register. See **Table 19-2** for bit descriptions.	32	POR, H
0x2F C844	S	CMFTST	CMF_B EEPROM Test Register. See **Table 19-3** for bit descriptions.	32	POR, H
0x2F C848	S	CMFCTL	CMF_B EEPROM High Voltage Control Register. See **Table 19-6** for bit descriptions.	32	POR, H

NOTES:

1. Bit 3 (FIC) is write-once. Bit 0 ($\overline{\text{LOCK}}$) is write-once unless in freeze or test mode.

Table A-5 DPTRAM (Dual-Port TPU RAM)

Address	Access	Symbol	Register	Size	Reset
0x30 0000	S	DPTMCR	DPT Module Configuration Register. See **Table 18-2** for bit descriptions.	16	S
0x30 0002	T	RAMTST	Test register, factory test only.	16	S
0x30 0004	S[1]	RAMBAR	RAM Array Address Register. See **Table 18-3** for bit descriptions.	16	S
0x30 0006	S, read only	MISRH	Multiple Input Signature Register High. See **18.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	S
0x30 0008	S, read only	MISRL	Multiple Input Signature Register Low. See **18.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	S
0x30 000A	S, read only	MISCNT	MISC Counter. See **18.3.5 MISC Counter (MISCNT)** for bit descriptions.	16	S

NOTES:
1. Entire register is write-once.

Table A-6 DPTRAM Array

Address	Access	Symbol	Register	Size	Reset
0x30 2000 – 0x30 37FF	U, S[1]		DPTRAM Array	—	—

NOTES:
1. Access to the DPTRAM array through the IMB3 bus is disabled once bit 5 (EMU) of either TPUMCR is set.

Table A-7 TPU3 (Time Processor Unit)

Address	Access	Symbol	Register	Size	Reset
TPU_A (Note: Bit descriptions apply to TPU_B as well)					
0x30 4000	S[1]	TPUMCR_A	TPU3_A Module Configuration Register. See **Table 17-6** for bit descriptions.	16 only	S, M
0x30 4002	T	TCR_A	TPU3_A Test Configuration Register.	16	S, M
0x30 4004	S	DSCR_A	TPU3_A Development Support Control Register. See **Table 17-7** for bit descriptions.	16[2]	S, M
0x30 4006	S	DSSR_A	TPU3_A Development Support Status Register. See **Table 17-8** for bit descriptions.	16[2]	S, M
0x30 4008	S	TICR_A	TPU3_A Interrupt Configuration Register. See **Table 17-9** for bit descriptions.	16[2]	S, M
0x30 400A	S	CIER_A	TPU3_A Channel Interrupt Enable Register. See **Table 17-10** for bit descriptions.	16[2]	S, M
0x30 400C	S	CFSR0_A	TPU3_A Channel Function Selection Register 0. See **Table 17-11** for bit descriptions.	16[2]	S, M
0x30 400E	S	CFSR1_A	TPU3_A Channel Function Selection Register 1. See **Table 17-11** for bit descriptions.	16[2]	S, M
0x30 4010	S	CFSR2_A	TPU3_A Channel Function Selection Register 2. See **Table 17-11** for bit descriptions.	16[2]	S, M
0x30 4012	S	CFSR3_A	TPU_A Channel Function Selection Register 3. See **Table 17-11** for bit descriptions.	16[2]	S, M
0x30 4014	S/U[3]	HSQR0_A	TPU_A Host Sequence Register 0. See **Table 17-12** for bit descriptions.	16[2]	S, M
0x30 4016	S/U[3]	HSQR1_A	TPU_A Host Sequence Register 1. See **Table 17-12** for bit descriptions.	16[2]	S, M
0x30 4018	S/U[3]	HSRR0_A	TPU_A Host Service Request Register 0. See **Table 17-13** for bit descriptions.	16[2]	S, M
0x30 401A	S/U[3]	HSRR1_A	TPU_A Host Service Request Register 1. See **Table 17-13** for bit descriptions.	16[2]	S, M
0x30 401C	S	CPR0_A	TPU_A Channel Priority Register 0. See **Table 17-14** for bit descriptions.	16[2]	S, M
0x30 401E	S	CPR1_A	TPU_A Channel Priority Register 1. See **Table 17-14** for bit descriptions.	16[2]	S, M
0x30 4020	S	CISR_A	TPU_A Channel Interrupt Status Register. See **Table 17-16** for bit descriptions.	16	S, M
0x30 4022	T	LR_A	TPU_A Link Register	16[2]	S, M
0x30 4024	T	SGLR_A	TPU_A Service Grant Latch Register	16[2]	S, M
0x30 4026	T	DCNR_A	TPU_A Decoded Channel Number Register	16[2]	S, M
0x30 4028	S[4]	TPUMCR2_A	TPU_A Module Configuration Register 2. See **Table 17-17** for bit descriptions.	16[2]	S, M

Table A-7 TPU3 (Time Processor Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 402A	S	TPUMCR3_A	TPU_A Module Configuration Register 3. See Table 17-20 for bit descriptions.	16[2]	S, M
0x30 402C	T	ISDR_A	TPU_A Internal Scan Data Register	16, 32[2]	
0x30 402E	T	ISCR_A	TPU_A Internal Scan Control Register	16, 32[2]	
0x30 4100 – 0x30 410F	S/U[3]	—	TPU_A Channel 0 Parameter Registers	16, 32[2]	
0x30 4110 – 0x30 411F	S/U[3]	—	TPU_A Channel 1 Parameter Registers	16, 32[2]	
0x30 4120 – 0x30 412F	S/U[3]	—	TPU_A Channel 2 Parameter Registers.	16, 32[2]	
0x30 4130 – 0x30 413F	S/U[3]	—	TPU_A Channel 3 Parameter Registers.	16, 32[2]	
0x30 4140 – 0x30 414F	S/U[3]	—	TPU_A Channel 4 Parameter Registers	16, 32[2]	
0x30 4150 – 0x30 415F	S/U[3]	—	TPU_A Channel 5 Parameter Registers	16, 32[2]	
0x30 4160 – 0x30 416F	S/U[3]	—	TPU_A Channel 6 Parameter Registers	16, 32[2]	
0x30 4170 – 0x30 417F	S/U[3]	—	TPU_A Channel 7 Parameter Registers	16, 32[2]	
0x30 4180 – 0x30 418F	S/U[3]	—	TPU_A Channel 8 Parameter Registers	16, 32[2]	
0x30 4190 – 0x30 419F	S/U[3]	—	TPU_A Channel 9 Parameter Registers	16, 32[2]	
0x30 41A0 – 0x30 41AF	S/U[3]	—	TPU_A Channel 10 Parameter Registers	16, 32[2]	
0x30 41B0 – 0x30 41BF	S/U[3]	—	TPU_A Channel 11 Parameter Registers	16, 32[2]	
0x30 41C0 – 0x30 41CF	S/U[3]	—	TPU_A Channel 11 Parameter Registers	16, 32[2]	
0x30 41D0 – 0x30 41DF	S/U[3]	—	TPU_A Channel 11 Parameter Registers	16, 32[2]	
0x30 41E0 – 0x30 41EF	S/U[3]	—	TPU_A Channel 14 Parameter Registers	16, 32[2]	
0x30 41F0 – 0x30 41FF	S/U[3]	—	TPU_A Channel 15 Parameter Registers	16, 32[2]	
TPU_B					
0x30 4400[1]	S[1]	TPUMCR_B	TPU3_B Module Configuration Register	16 only	S, M
0x30 4402	T	TCR_B	TPU3_B Test Configuration Register	16	S, M
0x30 4404	S	DSCR_B	TPU3_B Development Support Control Register	16[2]	S, M
0x30 4406	S	DSSR_B	TPU3_B Development Support Status Register	16[2]	S, M
0x30 4408	S	TICR_B	TPU3_B Interrupt Configuration Register	16[2]	S, M
0x30 440A	S	CIER_B	TPU3_B Channel Interrupt Enable Register	16[2]	S, M

Table A-7 TPU3 (Time Processor Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 440C	S	CFSR0_B	TPU3_B Channel Function Selection Register 0	16[2]	S, M
0x30 440E	S	CFSR1_B	TPU3_B Channel Function Selection Register 1	16[2]	S, M
0x30 4410	S	CFSR2_B	TPU3_B Channel Function Selection Register 2	16[2]	S, M
0x30 4412	S	CFSR3_B	TPU_B Channel Function Selection Register 3	16[2]	S, M
0x30 4414	S/U[3]	HSQR0_B	TPU_B Host Sequence Register 0	16[2]	S, M
0x30 4416	S/U[3]	HSQR1_B	TPU_B Host Sequence Register 1	16[2]	S, M
0x30 4418	S/U[3]	HSRR0_B	TPU_B Host Service Request Register 0	16[2]	S, M
0x30 441A	S/U[3]	HSRR1_B	TPU_B Host Service Request Register 1	16[2]	S, M
0x30 441C	S	CPR0_B	TPU_B Channel Priority Register 0	16[2]	S, M
0x30 441E	S	CPR1_B	TPU_B Channel Priority Register 1	16[2]	S, M
0x30 4420	S	CISR_B	TPU_B Channel Interrupt Status Register	16	S, M
0x30 4422	T	LR_B	TPU_B Link Register	16[2]	S, M
0x30 4424	T	SGLR_B	TPU_B Service Grant Latch Register	16[2]	S, M
0x30 4426	T	DCNR_B	TPU_B Decoded Channel Number Register	16[2]	S, M
0x30 4428	S[4]	TPUMCR2_B	TPU_B Module Configuration Register 2	16[2]	S, M
0x30 442A	S	TPUMCR3_B	TPU_B Module Configuration Register 3	16, 32[2]	S, M
0x30 442C	T	ISDR_B	TPU_B Internal Scan Data Register	16, 32[2]	
0x30 442E	T	ISCR_B	TPU_B Internal Scan Control Register	16, 32[2]	
0x30 4500 – 0x30 450E	S/U[3]	—	TPU_B Channel 0 Parameter Registers	16, 32[2]	
0x30 4510 – 0x30 451E	S/U[3]	—	TPU_B Channel 1 Parameter Registers	16, 32[2]	
0x30 4520 – 0x30 452E	S/U[3]	—	TPU_B Channel 2 Parameter Registers	16, 32[2]	
0x30 4530 – 0x30 453E	S/U[3]	—	TPU_B Channel 3 Parameter Registers	16, 32[2]	
0x30 4540 – 0x30 454E	S/U[3]	—	TPU_B Channel 4 Parameter Registers	16, 32[2]	
0x30 4550 – 0x30 455E	S/U[3]	—	TPU_B Channel 5 Parameter Registers	16, 32[2]	
0x30 4560 – 0x30 456E	S/U[3]	—	TPU_B Channel 6 Parameter Registers	16, 32[2]	
0x30 4570 – 0x30 457E	S/U[3]	—	TPU_B Channel 7 Parameter Registers	16, 32[2]	
0x30 4580 – 0x30 458E	S/U[3]	—	TPU_B Channel 8 Parameter Registers	16, 32[2]	
0x30 4590 – 0x30 459E	S/U[3]	—	TPU_B Channel 9 Parameter Registers	16, 32[2]	
0x30 45A0 – 0x30 45AE	S/U[3]	—	TPU_B Channel 10 Parameter Registers	16, 32[2]	

Table A-7 TPU3 (Time Processor Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 45B0 – 0x30 45BF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	
0x30 45C0 – 0x30 45CF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	
0x30 45D0 – 0x30 45DF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	
0x30 45E0 – 0x30 45EF	S/U[3]	—	TPU_B Channel 14 Parameter Registers	16, 32[2]	
0x30 45F0 – 0x30 45FF	S/U[3]	—	TPU_B Channel 15 Parameter Registers	16[2]	

NOTES:

1. Bit 10 (TPU3) and bit 11 (T2CSL) are write-once. Bits 1:2 (TCR1P) and bits 3:4 (TCR2P) are write-once if PWOD is not set in the TPUMCR3 register. This register cannot be accessed with a 32-bit read. It can only be accessed with an 8- or 16-bit read.
2. Some TPU registers can only be read or written with 16- or 32-bit accesses. 8-bit accesses are not allowed.
3. S/U = Supervisor accessible only if SUPV = 1 or unrestricted if SUPV = 0. Unrestricted registers allow both user and supervisor access. The SUPV bit is in the TPUMCR register.
4. Bits 9:10 (ETBANK), 14 (T2CF), and 15 (DTPU) are write-once.

Table A-8 QADC64 (Queued Analog-to-Digital Converter)

Address	Access	Symbol	Register	Size	Reset
QADC_A (Note: Bit descriptions apply to QADC_B as well)					
0x30 4800	S	QADC64MCR_A	QADC64 Module Configuration Register. See **Table 13-7** for bit descriptions.	16	S
0x30 4802	T	QADC64TEST_A	QADC64 Test Register	16	—
0x30 4804	S	QADC64INT_A	Interrupt Register. See **Table 13-8** for bit descriptions.	16	S
0x30 4806	S/U	PORTQA_A/ PORTQB_A	Port A and Port B Data. See **Table 13-9** for bit descriptions.	16	U
0x30 4808	S/U	DDRQA_A/ DDRQB_A	Port A Data and Port B Direction Register. See **Table 13-10** for bit descriptions.	16	S
0x30 480A	S/U	QACR0_A	QADC64 Control Register 0. See **Table 13-11** for bit descriptions.	16	S
0x30 480C	S/U[1]	QACR1_A	QADC64 Control Register 1. See **Table 13-12** for bit descriptions.	16	S
0x30 480E	S/U[1]	QACR2_A	QADC64 Control Register 2. See **Table 13-14** for bit descriptions.	16	S
0x30 4810	S/U	QASR0_A	QADC64 Status Register 0. See **Table 13-16** for bit descriptions.	16	S
0x30 4812	S/U	QASR1_A	QADC64 Status Register 1. See **Table 13-18** for bit descriptions.	16	S
0x30 4814 – 0x30 49FE	—	—	Reserved	—	—
0x30 4A00 – 0x30 4A7E	S/U	CCW_A	Conversion Command Word Table. See **Table 13-19** for bit descriptions.	16	U

Table A-8 QADC64 (Queued Analog-to-Digital Converter) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 4A80 – 0x30 4AFE	S/U	RJURR_A	Result Word Table Right-Justified, Unsigned Result Register. See 13.12.12 for bit descriptions.	16	X
0x30 4B00 – 0x30 4B7E	S/U	LJSRR_A	Result Word Table Left-Justified, Signed Result Register. See 13.12.12 for bit descriptions.	16	X
0x30 4B80 – 0x30 4BFE	S/U	LJURR_A	Result Word Table Left-Justified, Unsigned Result Register. See 13.12.12 for bit descriptions.	16	X
QADC_B					
0x30 4C00	S	QADC64MCR_B	QADC64 Module Configuration Register	16	S
0x30 4C02	T	QADC64TEST_B	QADC64 Test Register	16	—
0x30 4C04	S	QADC64INT_B	Interrupt Register	16	S
0x30 4C06	S/U	PORTQA_B/ PORTQB_B	Port A and Port B Data	16	U
0x30 4C08	S/U	DDRQA_B/ DDRQB_B	Port A Data and Port B Direction Register	16	S
0x30 4C0A	S/U	QACR0_B	QADC64 Control Register 0	16	S
0x30 4C0C	S/U[1]	QACR1_B	QADC64 Control Register 1	16	S
0x30 4C0E	S/U[1]	QACR2_B	QADC64 Control Register 2	16	S
0x30 4C10	S/U	QASR0_B	QADC64 Status Register 0	16	S
0x30 4C12	S/U	QASR1_B	QADC64 Status Register 1	16	S
0x30 4C14 – 0x30 4DFE	—	—	Reserved	—	—
0x30 4E00 – 0x30 4E7E	S/U	CCW_B	Conversion Command Word Table	16	U
0x30 4E80 – 0x30 4EFE	S/U	RJURR_B	Result Word Table. Right-Justified, Unsigned Result Register.	16	X
0x30 4F00 – 0x30 4F7E	S/U	LJSRR_B	Result Word Table. Left-Justified, Signed Result Register.	16	X
0x30 4F80 – 0x30 4FFE	S/U	LJURR_B	Result Word Table. Left-Justified, Unsigned Result Register.	16	X

NOTES:
1. Bit 3 (SSEx) is readable in test mode only.

Table A-9 QSMCM (Queued Serial Multi-Channel Module)

Address	Access	Symbol	Register	Size	Reset
0x30 5000	S	QSMCMMCR	QSMCM Module Configuration Register. See **Table 14-4** for bit descriptions.	16	S
0x30 5002	T	QTEST	QSMCM Test Register	16	S
0x30 5004	S	QDSCI_IL	Dual SCI Interrupt Level. See **Table 14-5** for bit descriptions.	16	S
0x30 5006	S	QSPI_IL	Queued SPI Interrupt Level. See **Table 14-6** for bit descriptions.	16	S
0x30 5008	S/U	SCC1R0	SCI1Control Register 0. See **Table 14-23** for bit descriptions.	16	S
0x30 500A	S/U	SCC1R1	SCI1Control Register 1. See **Table 14-24** for bit descriptions.	16	S
0x30 500C	S/U	SC1SR	SCI1 Status Register. See **Table 14-25** for bit descriptions.	16	S
0x30 500E	S/U	SC1DR	SCI1 Data Register. See **Table 14-26** for bit descriptions.	16	S
0x30 5010 — 0x30 5012	—	—	Reserved	—	—
0x30 5014	S/U	PORTQS	QSMCM Port QS Data Register. See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.	16	S
0x30 5016	S/U	PQSPAR/ DDRQST	QSMCM Port QS PIn Assignment Register/ QSMCM Port QS Data Direction Register. See **Table 14-11** for bit descriptions.	16	S
0x30 5018	S/U	SPCR0	QSPI Control Register 0. See **Table 14-13** for bit descriptions.	16	S
0x30 501A	S/U	SPCR1	QSPI Control Register 1. See **Table 14-15** for bit descriptions.	16	S
0x30 501C	S/U	SPCR2	QSPI Control Register 2. See **Table 14-16** for bit descriptions.	16	S
0x30 501E	S/U	SPCR3	QSPI Control Register 3. See **Table 14-17** for bit descriptions.	8	S
0x30 501F	S/U	SPSR	QSPI Status Register 3. See **Table 14-18** for bit descriptions.	8	S
0x30 5020	S/U	SCC2R0	SCI2 Control Register 0	16	S
0x30 5022	S/U	SCC2R1	SCI2 Control Register 1	16	S
0x30 5024	S/U	SC2SR	SCI2 Status Register	16	S
0x30 5026	S/U	SC2DR	SCI2 Data Register	16	S
0x30 5028	S/U[1]	QSCI1CR	QSCI1 Control Register. See **Table 14-30** for bit descriptions.	16	S
0x30 502A	S/U[2]	QSCI1SR	QSCI1 Status Register. See **Table 14-31** for bit descriptions.	16	S
0x30 502C – 0x30 504A	S/U	SCTQ	Transmit Queue Locations	16	S
0x30 504C – 0x30 506A	S/U	SCRQ	Receive Queue Locations	16	S
0x30 506C – 0x30 5013F	—	—	Reserved	—	—

Table A-9 QSMCM (Queued Serial Multi-Channel Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 5140 – 0x30 517F	S/U	RECRAM	Receive Data RAM	16	S
0x30 5180 – 0x30 51BF	S/U	TRAN.RAM	Transmit Data RAM	16	S
0x30 51C0 – 0x30 51DF	S/U	COMD.RAM	Command RAM	16	S

NOTES:
1. Bits 0–3 writeable only in test mode, otherwise read only.
2. Bits 3–11 writeable only in test mode, otherwise read only.

Table A-10 MIOS1 (Modular Input/Output Subsystem)

Address	Access	Symbol	Register	Size	Reset
MPWMSM0 (MIOS Pulse Width Modulation Submodule 0)					
0x30 6000	S/U	MPWMSMPERR	MPWMSM0 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 6002	S/U	MPWMSMPULR	MPWMSM0 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 6004	S/U	MPWMSMCNTR	MPWMSM0 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 6006	S/U	MPWMSMSCR	MPWMSM0 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MPWMSM1 (MIOS Pulse Width Modulation Submodule 1)					
0x30 6008	S/U	MPWMSMPERR	MPWMSM1 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 600A	S/U	MPWMSMPULR	MPWMSM1 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 600C	S/U	MPWMSMCNTR	MPWMSM1 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 600E	S/U	MPWMSMSCR	MPWMSM1 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MPWMSM2 (MIOS Pulse Width Modulation Submodule 2)					
0x30 6010	S/U	MPWMSMPERR	MPWMSM2 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 6012	S/U	MPWMSMPULR	MPWMSM2 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 6014	S/U	MPWMSMCNTR	MPWMSM2 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 6016	S/U	MPWMSMSCR	MPWMSM2 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MPWMSM3 (MIOS Pulse Width Modulation Submodule 3)					
0x30 6018	S/U	MPWMSMPERR	MPWMSM3 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 601A	S/U	MPWMSMPULR	MPWMSM3 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 601C	S/U	MPWMSMCNTR	MPWMSM3 Count Register. See **Table 15-22** for bit descriptions.	16	X

Table A-10 MIOS1 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 601E	S/U	MPWMSMSCR	MPWMSM3 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MMCSM6 (MIOS Modulus Counter Submodule 6)					
0x30 6030	S/U	MMCSMCNT	MMCSM6 Up-Counter Register. See **Table 15-12** for bit descriptions.	16	X
0x30 6032	S/U	MMCSMML	MMCSM6 Modulus Latch Register. See **Table 15-13** for bit descriptions.	16	X
0x30 6034	S/U	MMCSMSCRD	MMCSM6 Status/Control Register Duplicated. See **15.10.1.3 MMCSM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 6036	S/U	MMCSMSCR	MMCSM6 Status/Control Register. See **Table 15-14** for bit descriptions.	16	S
MDASM11 (MIOS Double Action Submodule 11)					
0x30 6058	S/U	MDASMAR	MDASM11 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 605A	S/U	MDASMBR	MDASM11 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 605C	S/U	MDASMSCRD	MDASM11 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 605E	S/U	MDASMSCR	MDASM11 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM12 (MIOS Double Action Submodule 12)					
0x30 6060	S/U	MDASMAR	MDASM12 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 6062	S/U	MDASMBR	MDASM12 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 6064	S/U	MDASMSCRD	MDASM12 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 6066	S/U	MDASMSCR	MDASM12 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM13 (MIOS Double Action Submodule 13)					
0x30 6068	S/U	MDASMAR	MDASM13 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 606A	S/U	MDASMBR	MDASM13 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 606C	S/U	MDASMSCRD	MDASM13 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S

Table A-10 MIOS1 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 606E	S/U	MDASMSCR	MDASM13 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM14 (MIOS Double Action Submodule 14)					
0x30 6070	S/U	MDASMAR	MDASM14 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 6072	S/U	MDASMBR	MDASM14 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 6074	S/U	MDASMSCRD	MDASM14 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 6076	S/U	MDASMSCR	MDASM14 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM15 (MIOS Double Action Submodule 15)					
0x30 6078	S/U	MDASMAR	MDASM15 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 607A	S/U	MDASMBR	MDASM15 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 607C	S/U	MDASMSCRD	MDASM15 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 607E	S/U	MDASMSCR	MDASM15 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MPWMSM16 (MIOS Pulse Width Modulation Submodule 16)					
0x30 6080	S/U	MPWMSMPERR	MPWMSM16 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 6082	S/U	MPWMSMPULR	MPWMSM16 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 6084	S/U	MPWMSMCNTR	MPWMSM16 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 6086	S/U	MPWMSMSCR	MPWMSM16 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MPWMSM17 (MIOS Pulse Width Modulation Submodule 17)					
0x30 6088	S/U	MPWMSMPERR	MPWMSM17 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 608A	S/U	MPWMSMPULR	MPWMSM17 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 608C	S/U	MPWMSMCNTR	MPWMSM17 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 608E	S/U	MPWMSMSCR	MPWMSM17 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S

Table A-10 MIOS1 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
MPWMSM18 (MIOS Pulse Width Modulation Submodule 18)					
0x30 6090	S/U	MPWMSMPERR	MPWMSM18 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 6092	S/U	MPWMSMPULR	MPWMSM18 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 6094	S/U	MPWMSMCNTR	MPWMSM18 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 6096	S/U	MPWMSMSCR	MPWMSM18 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MPWMSM19 (MIOS Pulse Width Modulation Submodule 19)					
0x30 6098	S/U	MPWMSMPERR	MPWMSM19 Period Register. See **Table 15-20** for bit descriptions.	16	X
0x30 609A	S/U	MPWMSMPULR	MPWMSM19 Pulse Register. See **Table 15-21** for bit descriptions.	16	X
0x30 609C	S/U	MPWMSMCNTR	MPWMSM19 Count Register. See **Table 15-22** for bit descriptions.	16	X
0x30 609E	S/U	MPWMSMSCR	MPWMSM19 Status/Control Register. See **Table 15-23** for bit descriptions.	16	S
MMCSM22 (MIOS Modulus Counter Submodule 22)					
0x30 60B0	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 15-12** for bit descriptions.	16	X
0x30 60B2	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 15-12** for bit descriptions.	16	X
0x30 60B4	S/U	MMCSMSCRD	MMCSM Status/Control Register Duplicated. See **15.10.1.3 MMCSM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60B6	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 15-14** for bit descriptions.	16	S
MDASM27 (MIOS Double Action Submodule 27)					
0x30 60D8	S/U	MDASMAR	MDASM27 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 60DA	S/U	MDASMBR	MDASM27 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 60DC	S/U	MDASMSCRD	MDASM27 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60DE	S/U	MDASMSCR	MDASM27 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM28 (MIOS Double Action Submodule 28)					
0x30 60E0	S/U	MDASMAR	MDASM28 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 60E2	S/U	MDASMBR	MDASM28 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X

Table A-10 MIOS1 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 60E4	S/U	MDASMSCRD	MDASM28 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60E6	S/U	MDASMSCR	MDASM28 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM29 (MIOS Double Action Submodule 29)					
0x30 60E8	S/U	MDASMAR	MDASM29 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 60EA	S/U	MDASMBR	MDASM29 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 60EC	S/U	MDASMSCRD	MDASM29 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60EE	S/U	MDASMSCR	MDASM29 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM30 (MIOS Double Action Submodule 30)					
0x30 60F0	S/U	MDASMAR	MDASM30 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 60F2	S/U	MDASMBR	MDASM30 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 60F4	S/U	MDASMSCRD	MDASM30 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60F6	S/U	MDASMSCR	MDASM30 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MDASM31 (MIOS Double Action Submodule 31)					
0x30 60F8	S/U	MDASMAR	MDASM31 Data A Register. See **15.11.1.1 MDASM Data A Register** for bit descriptions.	16	X
0x30 60FA	S/U	MDASMBR	MDASM31 Data B Register. See **15.11.1.2 MDASM Data B Register (MDASMBR)** for bit descriptions.	16	X
0x30 60FC	S/U	MDASMSCRD	MDASM31 Status/Control Register Duplicated. See **15.11.1.3 MDASM Status/Control Register (Duplicated)** for bit descriptions.	16	S
0x30 60FE	S/U	MDASMSCR	MDASM31 Status/Control Register. See **Table 15-17** for bit descriptions.	16	S
MPIOSM (MIOS 16-bit Parallel Port I/O Submodule)					
0x30 6100	S/U	MPIOSMDR	MPIOSM Data Register. See **Table 15-26** for bit descriptions.	16	X
0x30 6102	S/U	MPIOSMDDR	MPIOSM Data Direction Register. See **Table 15-27** for bit descriptions.	16	S

Table A-10 MIOS1 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 6104 — 0x30 6106	S/U	—	Reserved	—	—
MBISM (MIOS Bus Interface Submodule)					
0x30 6800	S[1]	MIOS1TPCR	MIOS1 Test and Pin Control Register. See **Table 15-3** for bit descriptions.	16	S
0x30 6802	S	—	Reserved	—	—
0x30 6804	S, read only	MIOS1VNR	MIOS1 Module Version Number Register. See **Table 15-4** for bit descriptions.	16	X
0x30 6806	S	MIOS1MCR	MIOS1 Module Control Register. See **Table 15-4** for bit descriptions.	16	S
0x30 6808 – 0x30 680E	S	—	Reserved	—	—
MCPSM (MIOS Counter Prescaler Submodule)					
0x30 6810 – 0x30 6814	S	—	Reserved	—	S
0x30 6816	S	MCPSMSCR	MCPSM Status/Control Register. See **Table 15-10** for bit descriptions.	16	S
MIRSM0 (MIOS Interrupt Request Submodule 0)					
0x30 6C00	S	MIOS1SR0	MIRSM0 Interrupt Status Register. See **Table 15-29** for bit descriptions.	16	X
0x30 6C02	S	—	Reserved	—	—
0x30 6C04	S	MIOS1ER0	MIRSM0 Interrupt Enable Register. See **Table 15-30** for bit descriptions.	16	S
0x30 6C06	S, read only	MIOS1RPR0	MIRSM0 Request Pending Register. See **Table 15-31** for bit descriptions.	16	S
MIRSM (MIOS Interrupt Request Submodule)					
0x30 6C30	S	MIOS1LVL0	MIOS1 Interrupt Level Register 0. See **Table 15-7** for bit descriptions.	16	S
MIRSM1 (MIOS Interrupt Request Submodule 1)					
0x30 6C40	S	MIOS1SR1	MIRSM1 Interrupt Status Register. See **Table 15-33** for bit descriptions.	16	X
0x30 6C42	S	—	Reserved	—	—
0x30 6C44	S	MIOS1ER1	MIRSM1 Interrupt Enable Register. See **Table 15-34** for bit descriptions.	16	S
0x30 6C46	S, read only	MIOS1RPR1	MIRSM1 Request Pending Register. See **Table 15-35** for bit descriptions.	16	S
MIRSM (MIOS Interrupt Request Submodule)					
0x30 6C70	S	MIOS1LVL1	MIOS1 Interrupt Level Register 1. See **Table 15-8** for bit descriptions.	16	S

NOTES:
1. Bit 0 (TEST) is reserved for factory testing.

Table A-11 TouCAN (CAN 2.0B Controller)

Address	Access	Symbol	Register	Size	Re-set
TouCAN_A (Note: Bit descriptions apply to TouCAN_B as well)					
0x30 7080	S	TCNMCR_A	TouCAN_A Module Configuration Register. See **Table 16-11** for bit descriptions.	16	S
0x30 7082	T	CANTCR_A	TouCAN_A Test Register	16	S
0x30 7084	S	CANICR_A	TouCAN_A Interrupt Configuration Register. See **Table 16-12** for bit descriptions.	16	S
0x30 7086	S/U	CANCTRL0_A/ CANCTRL1_A	TouCAN_A Control Register 0/ TouCAN_A Control Register 1. See **Table 16-13** and **Table 16-16** for bit descriptions.	16	S
0x30 7088	S/U	PRESDIV_A/ CTRL2_A	TouCAN_A Control and Prescaler Divider Register/ TouCAN_A Control Register 2. See **Table 16-17** and **Table 16-18** for bit descriptions.	16	S
0x30 708A	S/U	TIMER_A	TouCAN_A Free-Running Timer Register. See **Table 16-19** for bit descriptions.		S
0x30 708C — 0x30 708E	—	—	Reserved	—	—
0x30 7090	S/U	RXGMSKHI_A	TouCAN_A Receive Global Mask High. See **Table 16-20** for bit descriptions.	16	S
0x30 7092	S/U	RXGMSKLO_A	TouCAN_A Receive Global Mask Low. See **Table 16-20** for bit descriptions.	16	S
0x30 7094	S/U	RX14MSKHI_A	TouCAN_A Receive Buffer 14 Mask High. See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	16	S
0x30 7096	S/U	RX14MSKLO_A	TouCAN_A Receive Buffer 14 Mask Low. See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	16	S
0x30 7098	S/U	RX15MSKHI_A	TouCAN_A Receive Buffer 15 Mask High. See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	16	S
0x30 709A	S/U	RX15MSKLO_A	TouCAN_A Receive Buffer 15 Mask Low. See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	16	S
0x30 709C — 0x30 709E	—	—	Reserved	—	—
0x30 70A0	S/U	ESTAT_A	TouCAN_A Error and Status Register. See **Table 16-21** for bit descriptions.	16	S
0x30 70A2	S/U	IMASK_A	TouCAN_A Interrupt Masks. See **Table 16-24** for bit descriptions.	16	S
0x30 70A4	S/U	IFLAG_A	TouCAN_A Interrupt Flags. See **Table 16-25** for bit descriptions.	16	S
0x30 70A6	S/U	RXECTR_A/ TXECTR_A	TouCAN_A Receive Error Counter/ TouCAN_A Transmit Error Counter. See **Table 16-26** for bit descriptions.	16	S

Table A-11 TouCAN (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x307100 — 0x30710F	S/U	MBUFF0_A	TouCAN_A Message Buffer 0. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307110 — 0x30711F	S/U	MBUFF1_A	TouCAN_A Message Buffer 1. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307120 — 0x30712F	S/U	MBUFF2_A	TouCAN_A Message Buffer 2. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307130 — 0x30713F	S/U	MBUFF3_A	TouCAN_A Message Buffer 3. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307140 — 0x30714F	S/U	MBUFF4_A	TouCAN_A Message Buffer 4. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307150 — 0x30715F	S/U	MBUFF5_A	TouCAN_A Message Buffer 5. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307160 — 0x30716F	S/U	MBUFF6_A	TouCAN_A Message Buffer 6. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307170 — 0x30717F	S/U	MBUFF7_A	TouCAN_A Message Buffer 7. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307180 — 0x30718F	S/U	MBUFF8_A	TouCAN_A Message Buffer 8. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307190 — 0x30719F	S/U	MBUFF9_A	TouCAN_A Message Buffer 9. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071A0 — 0x3071AF	S/U	MBUFF10_A	TouCAN_A Message Buffer 10. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071B0 — 0x3071BF	S/U	MBUFF11_A	TouCAN_A Message Buffer 11. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071C0 — 0x3071CF	S/U	MBUFF12_A	TouCAN_A Message Buffer 12. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071D0 — 0x3071DF	S/U	MBUFF13_A	TouCAN_A Message Buffer 13. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071E0 — 0x3071EF	S/U	MBUFF14_A	TouCAN_A Message Buffer 14. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x3071F0 — 0x3071FF	S/U	MBUFF15_A	TouCAN_A Message Buffer 15. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U

Table A-11 TouCAN (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Re-set
TouCAN_B					
0x30 7480	S	TCNMCR_B	TouCAN_B Module Configuration Register	16	S
0x30 7482	T	CANTCR_B	TouCAN_B Test Register	16	S
0x30 7484	S	CANICR_B	TouCAN_B Interrupt Configuration Register	16	S
0x30 7486	S/U	CANCTRL0_B/ CANCTRL1_B	TouCAN_B Control Register 0/ TouCAN_B Control Register 1	16	S
0x30 7488	S/U	PRESDIV_B/ CTRL2_B	TouCAN_B Control and Prescaler Divider Register/ TouCAN_B Control Register 2	16	S
0x30 748A	S/U	TIMER_B	TouCAN_B Free-Running Timer Register		S
0x30 748C — 0x30 748E	—	—	Reserved	—	—
0x30 7490	S/U	RXGMSKHI_B	TouCAN_B Receive Global Mask High	16	S
0x30 7492	S/U	RXGMSKLO_B	TouCAN_B Receive Global Mask Low	16	S
0x30 7494	S/U	RX14MSKHI_B	TouCAN_B Receive Buffer 14 Mask High	16	S
0x30 7496	S/U	RX14MSKLO_B	TouCAN_B Receive Buffer 14 Mask Low	16	S
0x30 7498	S/U	RX15MSKHI_B	TouCAN_B Receive Buffer 15 Mask High	16	S
0x30 749A	S/U	RX15MSKLO_B	TouCAN_B Receive Buffer 15 Mask Low	16	S
0x30 749C — 0x30 749E	—	—	Reserved	—	—
0x30 74A0	S/U	ESTAT_B	TouCAN_B Error and Status Register	16	S
0x30 74A2	S/U	IMASK_B	TouCAN_B Interrupt Masks	16	S
0x30 74A4	S/U	IFLAG_B	TouCAN_B Interrupt Flags	16	S
0x30 74A6	S/U	RXECTR_B/ TXECTR_B	TouCAN_B Receive Error Counter/ TouCAN_B Transmit Error Counter	16	S
0x307500 — 0x30750F	S/U	MBUFF0_B	TouCAN_B Message Buffer 0.	—	U
0x307510 — 0x30751F	S/U	MBUFF1_B	TouCAN_B Message Buffer 1.	—	U
0x307520 — 0x30752F	S/U	MBUFF2_B	TouCAN_A Message Buffer 2.	—	U
0x307530 — 0x30753F	S/U	MBUFF3_B	TouCAN_B Message Buffer 3.	—	U
0x307540 — 0x30754F	S/U	MBUFF4_B	TouCAN_B Message Buffer 4.	—	U
0x307550 — 0x30755F	S/U	MBUFF5_B	TouCAN_B Message Buffer 5.	—	U
0x307560 — 0x30756F	S/U	MBUFF6_B	TouCAN_B Message Buffer 6.	—	U
0x307570 — 0x30757F	S/U	MBUFF7_B	TouCAN_B Message Buffer 7.	—	U
0x307580 — 0x30758F	S/U	MBUFF8_B	TouCAN_B Message Buffer 8.	—	U
0x307590 — 0x30759F	S/U	MBUFF9_B	TouCAN_B Message Buffer 9.	—	U

Table A-11 TouCAN (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Re-set
0x3075A0 — 0x3075AF	S/U	MBUFF10_B	TouCAN_B Message Buffer 10.	—	U
0x3075B0 — 0x3075BF	S/U	MBUFF11_B	TouCAN_B Message Buffer 11.	—	U
0x3075C0 — 0x3075CF	S/U	MBUFF12_B	TouCAN_B Message Buffer 12.	—	U
0x3075D0 — 0x3075DF	S/U	MBUFF13_B	TouCAN_B Message Buffer 13.	—	U
0x3075E0 — 0x3075EF	S/U	MBUFF14_B	TouCAN_B Message Buffer 14.	—	U
0x3075F0 — 0x3075FF	S/U	MBUFF15_B	TouCAN_B Message Buffer 15.	—	U

Table A-12 UIMB (U-Bus to IMB3 Bus Interface)

Address	Access	Symbol	Register	Size	Reset
0x30 7F80	S[1]	UMCR	UIMB Module Configuration Register. See **Table 12-6** for bit descriptions.	32	H
0x30 7F90	S/T	UTSTCREG	Test Register — Reserved	32	—
0x30 7FA0	S, read only	UIPEND	Pending Interrupt Request Register. See **Table 12-7** for bit descriptions.	32	H

NOTES:
1. Bit 3 (HSPEED) is write-once.

Table A-13 SRAM (Static RAM Access Memory)

Address	Access	Symbol	Register	Size	Reset
SRAM_A					
0x38 0000	S[1]	SRAMMCR_A	SRAM_A Module Configuration Register. See **Table 20-1** for bit descriptions.	16	S,H, POR
0x38 0004	T	SRAMTST_A	SRAM_A Test Register.	16	S,H, POR
SRAM_B					
0x38 0008	S[1]	SRAMMCR_B	SRAM_B Module Configuration Register. See **Table 20-1** for bit descriptions.	16	S,H, POR
0x38 000C	T	SRAMTST_B	SRAM_B Test Register.	16	S,H, POR

NOTES:
1. Bit 0 (LCK) locks the register (write-protected except in test mode) and is write once.

Table A-14 SRAM (Static RAM Access Memory) Array

Address	Access	Symbol	Register	Size	Reset
0x3F 8000 — 0x3F 97FF	—		Reserved	—	—
0x3F 9800 — 0x3F BFFF	U, S		SRAM_A RAM Array (10 K Bytes)	8, 16, 32	—
0x3F C000 — 0x3F FFFF	U, S		SRAM_B RAM Array (16 K Bytes)	8, 16, 32	—

APPENDIX B
REGISTER GENERAL INDEX

–A–

Associated registers 10-4

–B–

BAR (breakpoint address register) 21-46
BBC
    global region attribute register description (MI_GRA) 4-23
    module configuration register (BBCMCR) 4-24
    region attribute registers MI_RAx description (MI_RAx) 4-21
    region base address registers (MI_RBAx) 4-21
BBCMCR (BBC module configuration register) 4-24
BR0 (BR3 - memory controller base registers 0 - 3) 10-28
Breakpoint address register (BAR) 21-46
Breakpoint counter A value and control register (COUNTA) 21-52
Breakpoint counter B value and control register (COUNTB) 21-53

–C–

CANCTRL0 (control register 0) 16-25
CANCTRL1 (control register 1) 16-26
CANCTRL2 (control register 2) 16-28
CANICR (TouCAN interrupt configuration register) 16-24
CANMCR (TouCAN module configuration register) 16-22
CCW (conversion command word table) 13-46
CFSR0 (TPU3 channel function select register 0) 17-16
CFSR1 (TPU3 channel function select register 1) 17-16
CFSR2 (TPU3 channel function select register 2) 17-16
CFSR3 (TPU3 channel function select register 3) 17-16
CIER (TPU3 channel interrupt enable register) 17-15
CISR (TPU3 channel interrupt status register) 17-19
CMF
    EEPROM configuration register (CMFMCR) 19-5
    EEPROM control registers 19-4
    EEPROM high voltage control register (CMFCTL) 19-9
CMFCFIG (hard reset configuration word) 19-17
CMFCTL (CMF EEPROM high voltage control register) 19-7, 19-9
CMFMCR (CMF EEPROM configuration register) 19-5
CMPA-CMPD (comparator A-D value register) 21-45
CMPE-CMPF (comparator E-F value registers) 21-46
CMPG-CMPH (comparator G-H value registers) 21-46
COLIR
    change of lock interrupt register 8-35
COLIR (change of lock register) 8-35
Comparator A-D value registers (CMPA-CMPD) 21-45
Comparator E-F value registers (CMPE-CMPF) 21-46
Comparator G-H value registers (CMPG-CMPH) 21-46
COUNTA (breakpoint counter A value and control register) 21-52
COUNTB (breakpoint counter B value and control register) 21-53
CPR0 (TPU3 channel priority register 0) 17-18
CPR1 (TPU3 channel priority register 1) 17-18
CR (condition register) 3-16

CTR (count register) 3-19

–D–

DAR (data address register) 3-22
DDRQA (port QA data direction register) 13-35
DDRQS (PORTQS data direction register) 14-12
Debug enable register (DER) 21-55
DEC (decrementer register) 3-24, 6-28
Decrementer register (DEC) 6-28
DER (debug enable register) 21-55
Development port data register (DPDR) 21-57
DMBR (dual mapping base register) 10-31
DMOR (dual mapping option register) 10-32
DPTMCR (DPTRAM module configuration register) 18-4
DPTRAM
    module configuration register (DPTMCR) 18-3
    ram base address register (RAMBAR) 18-5
    test register 18-4
DSCR (TPU3 development support control register) 17-12
DSISR (dae/source instruction service register) 3-22
DSSR (TPU3 development support status register) 17-14
Dual mapping base register (DMBR) 10-31
Dual mapping option register 10-32

–E–

ECR (exception cause register) 21-54
EMCR (external master control register) 6-22
ESTAT (error and status register) 16-30
Exception cause register (ECR) 21-53
External master control register (EMCR) 6-22

–F–

FPRs - (floating-point registers) 3-12
FPSCR (floating-point status and control register) 3-13

–G–

General-Purpose I/O registers 6-34
GPRs (general-purpose registers) 3-12

–H–

HSQR0 (TPU3 host sequence register 0) 17-17
HSQR1 (TPU3 host sequence register 1) 17-17
HSSR0 (TPU3 host service request register 0) 17-17
HSSR1 (TPU3 host service request register 1) 17-18

–I–

I-Bus support control register (ICTRL) 21-47
ICTRL (i-bus support control register) 21-47
IFLAG (interrupt flag register) 16-33
IMASK (interrupt mask register) 16-32
IMMR (internal memory mapping register) 6-21
Internal memory map register 6-21

–K–

Keep alive power registers lock mechanism 8-23

–L–

L2U
- global region attribute register (L2U_GRA) 11-15
- module configuration register (L2U_MCR) 11-13
- region attribute registers (L2U_RAx) 11-15
- region base address registers (L2U_RBAx) 11-14

L2U_GRA (L2U global region attribute register) 11-16
L2U_MCR (L2U module configuration register) 11-13
L2U_RAx (L2U region X attribute register) 11-15
L2U_RBAx (L2U region x base address register) 11-14
L-Bus support control register 1 (LCTRL1) 21-49
L-Bus support control register 2 (LCTRL2) 21-50
LCTRL1 (l-bus support control register 1) 21-49
LCTRL2 (l-bus support control register 2) 21-50
LJSRR (left justified, signed result register) 13-49
LJURR (left justified, unsigned result register) 13-50
LR (link register) 3-18

–M–

MBISM
- interrupt registers 15-10

MCPSM
- status/control register (MCPSMCSCR) 15-13

MCPSMSCR (MCPSM status/control register) 15-13
MDASM
- data A register (MDASMAR) 15-21
- data B register (MDASMBR) 15-21
- status/control register - duplicated (MDASMSCRD) 15-22
- status/control register (MDASMSCR) 15-23

MDASMAR (MDASM data A register) 15-21
MDASMBR (MDASM data B register) 15-22
MDASMSCR (MDASM status/control register) 15-23
MDASMSCRD (MDASM status/control register - duplicated) 15-22
Memory controller base registers (BR0 - BR3) 10-28
Memory controller option registers (OR0 - OR3) 10-30
Memory controller status registers (MSTAT) 10-28
MI_GRA (global region attribute register) 4-23
MIOS
- 16-bit parallel port I/O submodule (MPIOSM) Registers 15-30
- bus interface (MBISM) Registers 15-8
- counter prescaler submodule (MCPSM) Registers 15-12
- double action submodule (MDASM) Registers 15-19
- interrupt request submodule 0 (MIRSM0) Registers 15-33
- interrupt request submodule 1 (MIRSM1) Registers 15-36
- modulus counter submodule (MMCSM) Registers 15-15
- pulse width modulation submodule (MPWMSM) Registers 15-26

MIOS1
- interrupt level register 0 (MIOSLVL0) (MIOS1LVL0) 15-10
- interrupt level register 1 (MIOSLVL1) (MIOS1LVL1) 15-11
- module and version number register (MIOS1VNR) 15-9
- module configuration register (MIOS1MCR) 15-9
- test and pin control register 15-8
- vector register 15-9

MIOS1ER0 (MIRSM0 interrupt enable register) 15-35
MIOS1ER1 (interrupt enable register) 15-37
MIOS1LVL0 (MIOS1 interrupt level register 0) 15-11
MIOS1LVL1 (MIOS1 interrupt level 1 register) 15-11
MIOS1MCR (MIOS1 module configuration register) 15-9
MIOS1RPR0 (MIRSM0 request pending register) 15-35

MIOS1RPR1 (MIRSM1 request pending register) 15-38
MIOS1SR0 (MIRSM0 interrupt status register) 15-34
MIOS1SR1 (MIRSM1 interrupt status register) 15-36
MIOS1TPCR ((test and pin control register) 15-8
MIOS1VNR (MIOS1 module/version number register) 15-9
MIRSM0
    interrupt enable register (MIOS1ER0) 15-35
    interrupt status register (MIOS1SR0) 15-34
    request pending register (MIOS1RPR0) 15-35
MIRSM1
    interrupt enable register (MIOS1ER1) 15-37
    interrupt status register (MIOS1SR1) 15-36
    request pending register (MIOS1RPR1) 15-37
MISCNT (MISC counter) 18-6
MISRH (multiple input signature register high) 18-5
MISRL (multiple input signature register low) 18-6
MMCSM
    modulus latch register (MMCSMML) 15-16
    status/control register - duplicated (MMCSMSCRD) 15-16
    status/control register (MMCSMSCR) 15-17
    up-counter register (MMCSMCNT) 15-16
MMCSMCNT (MMCSM up-counter register) 15-16
MMCSMML (MMCSM modulus latch register) 15-16
MMCSMSCR (MMCSM status/control register) 15-17
MMCSMSCRD (MMCSM status/control register - duplicated) 15-17
MPIOSM
    data direction register (MPIOSMDDR) 15-31
    data register (MPIOSMDR) 15-30
MPIOSMDDR (MPIOSM data direction register) 15-31
MPIOSMDR (MPIOSM data register) 15-31
MPWMSM
    counter register (MPWMSMCNTR) 15-28
    period register (MPWMSMPERR) 15-27
    pulse width register (MPWMSMPULR) 15-27
    status/control register (MPWMSMCR) 15-28
MPWMSMCNTR (MPWMSM counter register) 15-28
MPWMSMPERR (MPWMSM period register) 15-27
MPWMSMPULR (MPWMSM pulse width register) 15-27
MPWMSMSCR (MPWMSM status/control register) 15-28
MSR (machine state register) 3-20
MSTAT (memory controller status register) 10-28

–O–

OR0 (OR3 - memory controller option registers 0 - 3) 10-30

–P–

PDMCR (Pad module configuration register) 2-29
Periodic interrupt status and control register (PISCR) 6-32
Periodic interrupt timer count register (PITC) 6-32
Periodic interrupt timer register (PITR) 6-33
PISCR (periodic interrupt status and control register) 6-32
PITC (periodic interrupt timer count) 6-32
PITR (periodic interrupt timer register) 6-33
PLL,
    low power, and reset control register (PLPRCR) 8-33
PLPRCR (PLL, low power, and reset control register) 8-33
PORTQA (port QA data register) 13-34
PORTQB (port QB data register) 13-34
PORTQS (port QS data register) 14-11

PQSPAR (PORTQS pin assignment register) 14-11
PRESDIV (prescaler divide register) 16-27
PVR (processor version register) 3-25

–Q–

QACR0 (QADC64 control register 0) 13-35
QACR1 (QADC64 control register 1) 13-36
QACR2 (QADC64 control register 2) 13-39
QADC64
- control register 0 (QACR0) 13-35
- control register 1 (QADC64CR1) 13-36
- control register 2 (QADC64CR2) 13-38
- interrupt register (QADC64INT) 13-33
- module configuration register (QADC64MCR) 13-33
- port A/B data register (PORTQA/B) 13-34
- port data direction register (DDRQA) 13-35
- status register 0 (QADC64SR0) 13-40
- status register 1 (QADC64SR1) 13-42
- successive approximation register (SAR) 13-14
- test register (QADC64TEST) 13-33

QADC64INT (QADC64 interrupt register) 13-33
QADC64MCR (QADC64 module configuration register) 13-33
QASR0 (QADC64 status register 0) 13-41, 13-42
QDSCI_IL (QSM2 dual SCI interrupt level register) 14-8
QSCI1CR (QSCI1 control register) 14-59
QSCI1SR (QSCI1 status register) 14-61
QSMCM
- configuration register (QMCR) 14-7
- interrupt level registers (QDSCI_IL, QSPI_IL) 14-8
- port QS data register (PORTQS) 14-10
- PORTQS data direction register (DDRQS) 14-12
- PORTQS pin assignment register (PQSPAR) 14-11
- QSCI1 control register (QSCI1CR) 14-59
- QSCI1 status register (QSCI1SR) 14-61
- QSPI command RAM (CRx) 14-22
- QSPI control register 0 (SPCR0) 14-16
- QSPI control register 1 (SPCR1) 14-18
- QSPI control register 2 (SPCR2) 14-18
- QSPI control register 3 (SPCR3) 14-19
- QSPI registers 14-15
- QSPI status register (SPSR) 14-20
- queued SCI1 status and control registers 14-59
- SCI control register 0 (SCCxR0) 14-45
- SCI control register 1 (SCCxR1) 14-45
- SCI data register (SCxDR) 14-49
- SCI registers 14-44
- SCI status register (SCxSR) 14-47
- test register (QTEST) 14-8

QSMCMMCR (QSMCM module configuration register) 14-7
QSPI_IL (QSPI interrupt level register) 14-9

–R–

RAMBAR (ram array base address register) 18-5
RCPU
- additional implementation-specific registers 3-27
- condition register (CR) 3-15
- condition register CR0 field definition 3-16
- condition register CR1 field definition 3-16
- condition register crn field - compare instruction 3-17

count register (CTR) 3-19
dae/source instruction service register (DSISR) 3-22
data address register (DAR) 3-22
decrementer register (DEC) 3-23
EIE, EID, and NRI special-purpose registers 3-26
floating-point exception cause register (FPECR) 3-26
floating-point registers (FPRs) 3-12
floating-point status and control register (FPSCR) 3-12
general special-purpose registers (SPRG0-SPRG3) 3-25
general-purpose registers (GPRs) 3-12
implementation-specific special-purpose registers 3-26
integer exception register (XER) 3-17
link register (LR) 3-18
machine state register (MSR) 3-20
machine status save/restore register 0 (SRR0) 3-24
machine status save/restore register 1 (SRR1) 3-24
powerpc OEA register set 3-20
powerpc UISA register set 3-11
powerpc VEA register set - time base 3-19
processor version register (PVR) 3-25
Real-Time clock alarm register (RTCAL) 6-31
Real-Time clock register (RTC) 6-31
Real-Time clock status and control register (RTCSC) 6-30
Region attribute registers (0 - 3) 4-21
Reset status register (RSR) 7-5
RJURR (right justified, unsigned result register) 13-49
RSR (reset status register) 7-5
RTC (real time clock alarm register) 6-31
RTC (real time clock register) 6-31
RTCSC (real time clock status and control register) 6-30
RXECTR (receive error counter) 16-33
RXGMSKHI (receive global mask register high) 16-29

–S–

SCCR (system clock control register) 8-30
SCCxR0 (QSMCM SCI control register 0) 14-45
SCCxR1 (QSMCM SCI control register 1) 14-46
SCDR (QSMCM SCI data register) 14-49
SCxSR (QSMCM SCIx status register) 14-47
SGPIO
control register (SGPIOCR) 6-35
data register 1 (SGPIODT1) 6-34
data register 2 (SGPIODT2) 6-34
SGPIOCR (SGPIO control register) 6-35
SGPIODT1 (SGPIO data register 1) 6-34
SGPIODT2 (SGPIO data register 2) 6-34
SIEL (SIU interrupt edge level register) 6-25
SIMASK (SIU interrupt mask register) 6-24
SIPEND
register (SIPEND) 6-23
SIPEND (SIU interrupt pending register) 6-24
SIU
interrupt edge level register (SIEL) 6-25
interrupt mask register (SIMASK) 6-24
interrupt registers 6-23
interrupt vector register (SIVEC) 6-25
module configuration register (SIUMCR) 6-18
SIUMCR (SIU module configuration register) 6-18
SIVEC (SIU interrupt vector) 6-25
Software service register (SWSR) 6-26

SPCR0 (QSPI control register 0) 14-16
SPCR1 (QSPI control register 1) 14-18
SPCR2 (QSPI control register 2) 14-19
SPCR3 (QSPI control register) 14-19
SPRG0-SPRG3 (general special-purpose registers 0-3) 3-25
SPSR (QSPI status register) 14-20
SRAM
    module configuration register (SRAMMCR) 20-2
    test register (SRAMTST) 20-3
SRAMMCR (SRAM module configuration register) 20-3
SRR0 (machine status save/restore register 0) 3-24
SRR1 (machine status save/restore register 1) 3-25
SWSR (software service register) 6-27
SYPCR (system protection control register) 6-26
System clock control register (SCCR) 8-30
System configuration and protection registers 6-18
System configuration registers 6-18
System protection control register (SYPCR) 6-26
System protection registers 6-26
System timer registers 6-28

–T–

TB (time base) 3-19, 3-23, 6-29
TBREF0 (time base reference register 0) 6-29
TBREF1 (time base reference register 1) 6-29
TBSCR (time base control and status register) 6-30
TESR (transfer error status register) 6-27
TICR (TPU3 interrupt configuration register) 17-14
Time base control and status register (TBSCR) 6-29
Time base reference registers (TBREF0) 6-29
TIMER (free running timer register) 16-29
TouCAN
    control register 0 (CANCTRL0) 16-25
    control register 1 (CANCTRL1) 16-26
    control register 2 (CANCTRL2) 16-28
    error and status register (ESTAT) 16-30
    interrupt configuration register (CANICR) 16-24
    interrupt flag register (IFLAG) 16-33
    interrupt mask register (IMASK) 16-32
    module configuration register (CANMCR) 16-22
    prescaler divide register (PRESDIV) 16-27
    receive buffer 14 mask registers 16-30
    receive buffer 15 mask registers 16-30
    receive global mask registers (RXGMSKHI) 16-29
    receive mask registers 16-7
    test configuration register 16-24
TPU3
    channel function select registers (CFSRx) 17-15
    channel interrupt enable register (CIER) 17-15
    channel interrupt status register (CISR) 17-19
    channel priority registers (CPRx) 17-18
    decoded channel number register (DCNR) 17-19
    development support control register (DSCR) 17-12
    development support status register (DSSR) 17-14
    host sequence registers (HSQRx) 17-16
    host service request registers (HSSRx) 17-17
    interrupt configuration register (TICR) 17-14
    link register (LR) 17-19
    module configuration register (TPUMCR) 17-10
    module configuration register 2 (TPUMCR2) 17-20

module configuration register 3 (TPUMCR3) 17-21
service grant latch register (SGLR) 17-19
test configuration register (TCR) 17-12
test registers (ISDR, ISCR) 17-22
TPUMCR (TPU3 module configuration register) 17-10
TPUMCR2 (TPU3 module configuration register 2) 17-20
TPUMCR3 (TPU3 module configuration register 3) 17-21
Transfer error status register (TESR) 6-27

–U–

UIMB
module configuration register (UMCR) 12-7
pending interrupt request register (UIPEND) 12-8
test control register (UTSTCREG) 12-8
UIPEND (UIMB pending interrupt reqiuest register) 12-8
UMCR (UIMB module configuration register) 12-7

–V–

VDDSRM
sensor register (VSRMSR) 8-36
VSRMSR (VDDSRM control register) 8-36

–X–

XER (integer exception register) 3-17

APPENDIX C
REGISTER DIAGRAM INDEX

–B–

BAR (breakpoint address register) 21-46
BBCMCR (BBC module configuration register) 4-24
BR0 (BR3 - memory controller base registers 0 - 3) 10-28

–C–

CANCTRL0 (control register 0) 16-25
CANCTRL1 (control register 1) 16-26
CANCTRL2 (control register 2) 16-28
CANICR (TouCAN interrupt configuration register) 16-24
CANMCR (TouCAN module configuration register) 16-22
CCW (conversion command word table) 13-46
CFSR0 (TPU3 channel function select register 0) 17-16
CFSR1 (TPU3 channel function select register 1) 17-16
CFSR2 (TPU3 channel function select register 2) 17-16
CFSR3 (TPU3 channel function select register 3) 17-16
CIER (TPU3 channel interrupt enable register) 17-15
CISR (TPU3 channel interrupt status register) 17-19
CMFCFIG (hard reset configuration word) 19-17
CMFCTL (CMF EEPROM high voltage control register) 19-7, 19-9
CMFMCR (CMF EEPROM configuration register) 19-5
CMPA-CMPD (comparator A-D value register) 21-45
CMPE-CMPF (comparator E-F value registers) 21-46
CMPG-CMPH (comparator G-H value registers) 21-46
COLIR (change of lock interrupt register) 8-35
COUNTA (breakpoint counter A value and control register) 21-52
COUNTB (breakpoint counter B value and control register) 21-53
CPR0 (TPU3 channel priority register 0) 17-18
CPR1 (TPU3 channel priority register 1) 17-18
CR (condition register) 3-16
CTR (count register) 3-19

–D–

DAR (data address register) 3-22
DDRQA (port QA data direction register) 13-35
DDRQS (PORTQS data direction register) 14-12
DEC (decrementer register) 3-24, 6-28
DER (debug enable register) 21-55
DMBR (dual mapping base register) 10-31
DMOR (dual mapping option register) 10-32
DPTMCR (DPTRAM module configuration register) 18-4
DSCR (TPU3 development support control register) 17-12
DSISR (dae/source instruction service register) 3-22
DSSR (TPU3 development support status register) 17-14

–E–

ECR (exception cause register) 21-54
EMCR (external master control register) 6-22
ESTAT (error and status register) 16-30

–F–

FPRs - (floating-point registers) 3-12
FPSCR (floating-point status and control register) 3-13

–G–

GPRs (general-purpose registers) 3-12

–H–

HSQR0 (TPU3 host sequence register 0) 17-17
HSQR1 (TPU3 host sequence register 1) 17-17
HSSR0 (TPU3 host service request register 0) 17-17
HSSR1 (TPU3 host service request register 1) 17-18

–I–

ICTRL (i-bus support control register) 21-47
IFLAG (interrupt flag register) 16-33
IMASK (interrupt mask register) 16-32
IMMR (internal memory mapping register) 6-21

–L–

L2U_GRA (L2U global region attribute register) 11-16
L2U_MCR (L2U module configuration register) 11-13
L2U_RAx (L2U region X attribute register) 11-15
L2U_RBAx (L2U region x base address register) 11-14
LCTRL1 (l-bus support control register 1) 21-49
LCTRL2 (l-bus support control register 2) 21-50
LJSRR (left justified, signed result register) 13-49
LJURR (left justified, unsigned result register) 13-50
LR (link register) 3-18

–M–

MCPSMSCR (MCPSM status/control register) 15-13
MDASMAR (MDASM data A register) 15-21
MDASMBR (MDASM data B register) 15-22
MDASMSCR (MDASM status/control register) 15-23
MDASMSCRD (MDASM status/control register - duplicated) 15-22
MI_GRA (global region attribute register) 4-23
MIOS1ER0 (MIRSM0 interrupt enable register) 15-35
MIOS1ER1 (interrupt enable register) 15-37
MIOS1LVL0 (MIOS1 interrupt level register 0) 15-11
MIOS1LVL1 (MIOS1 interrupt level 1 register) 15-11
MIOS1MCR (MIOS1 module configuration register) 15-9
MIOS1RPR0 (MIRSM0 request pending register) 15-35
MIOS1RPR1 (MIRSM1 request pending register) 15-38
MIOS1SR0 (MIRSM0 interrupt status register) 15-34
MIOS1SR1 (MIRSM1 interrupt status register) 15-36
MIOS1TPCR (test and pin control register) 15-8
MIOS1VNR (MIOS1 module/version number register) 15-9
MISCNT (MISC counter) 18-6
MISRH (multiple input signature register high) 18-5
MISRL (multiple input signature register low) 18-6
MMCSMCNT (MMCSM up-counter register) 15-16
MMCSMML (MMCSM modulus latch register) 15-16
MMCSMSCR (MMCSM status/control register) 15-17
MMCSMSCRD (MMCSM status/control register - duplicated) 15-17
MPIOSMDDR (MPIOSM data direction register) 15-31
MPIOSMDR (MPIOSM data register) 15-31

MPWMSMCNTR (MPWMSM counter register) 15-28
MPWMSMPERR (MPWMSM period register) 15-27
MPWMSMPULR (MPWMSM pulse width register) 15-27
MPWMSMSCR (MPWMSM status/control register) 15-28
MSR (machine state register) 3-20
MSTAT (memory controller status register) 10-28

–O–

OR0 (OR3 - memory controller option registers 0 - 3) 10-30

–P–

PDMCR (Pad module configuration register) 2-29
PISCR (periodic interrupt status and control register) 6-32
PITC (periodic interrupt timer count) 6-32
PITR (periodic interrupt timer register) 6-33
PLPRCR (PLL, low power, and reset control register) 8-33
PORTQA (port QA data register) 13-34
PORTQB (port QB data register) 13-34
PORTQS (port QS data register) 14-11
PQSPAR (PORTQS pin assignment register) 14-11
PRESDIV (prescaler divide register) 16-27
PVR (processor version register) 3-25

–Q–

QACR0 (QADC64 control register 0) 13-35
QACR1 (QADC64 control register 1) 13-36
QACR2 (QADC64 control register 2) 13-39
QADC64INT (QADC64 interrupt register) 13-33
QADC64MCR (QADC64 module configuration register) 13-33
QASR0 (QADC64 status register 0) 13-41, 13-42
QDSCI_IL (QSM2 dual SCI interrupt level register) 14-8
QSCI1CR (QSCI1 control register) 14-59
QSCI1SR (QSCI1 status register) 14-61
QSMCMMCR (QSMCM module configuration register) 14-7
QSPI_IL (QSPI interrupt level register) 14-9

–R–

RAMBAR (ram array base address register) 18-5
Regionattribute registers (0 - 3) 4-21
RJURR (right justified, unsigned result register) 13-49
RSR (reset status register) 7-5
RTC (real time clock) 6-31
RTCAL (real time clock alarm) 6-31
RTCSC (real time clock status and control register) 6-30
RXECTR (receive error counter) 16-33
RXGMSKHI (receive global mask register high) 16-29

–S–

SCCR (system clock control register) 8-30
SCCxR0 (QSMCM SCI control register 0) 14-45
SCCxR1 (QSMCM SCI control register 1) 14-46
SCDR (QSMCM SCI data register) 14-49
SCxSR (QSMCM SCIx status register) 14-47
SGPIOCR (SGPIO control register) 6-35
SGPIODT1 (SGPIO data register 1) 6-34
SGPIODT2 (SGPIO data register 2) 6-34
SIEL (SIU interrupt edge level register) 6-25
SIMASK (SIU interrupt mask register) 6-24

SIPEND (SIU interrupt pending register) 6-24
SIUMCR (SIU module configuration register) 6-18
SIVEC (SIU interrupt vector) 6-25
SPCR0 (QSPI control register 0) 14-16
SPCR1 (QSPI control register 1) 14-18
SPCR2 (QSPI control register 2) 14-19
SPCR3 (QSPI control register) 14-19
SPRG0-SPRG3 (general special-purpose registers 0-3) 3-25
SPSR (QSPI status register) 14-20
SRAMMCR (SRAM module configuration register) 20-3
SRR0 (machine status save/restore register 0) 3-24
SRR1 (machine status save/restore register 1) 3-25
SWSR (software service register) 6-27
SYPCR (system protection control register) 6-26

–T–

TB (time base) 3-19, 3-23, 6-29
TBREF0 (time base reference register 0) 6-29
TBREF1 (time base reference register 1) 6-29
TBSCR (time base control and status register) 6-30
TESR (transfer error status register) 6-27
TICR (TPU3 interrupt configuration register) 17-14
TIMER (free running timer register) 16-29
TPUMCR (TPU3 module configuration register) 17-10
TPUMCR2 (TPU3 module configuration register 2) 17-20
TPUMCR3 (TPU3 module configuration register 3) 17-21

–U–

UIPEND (UIMB pending interrupt reqiuest register) 12-8
UMCR (UIMB module configuration register) 12-7

–V–

VSRMSR (VDDSRM control register) 8-36

–X–

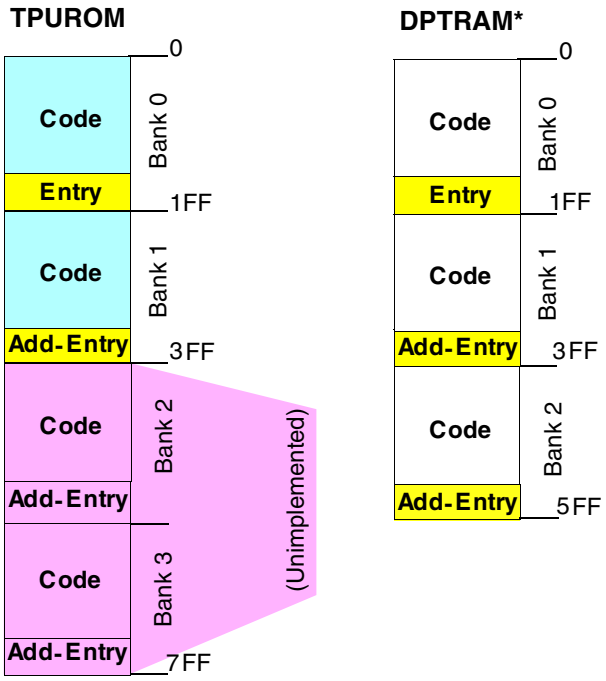
XER (integer exception register) 3-17

APPENDIX D
TPU ROM FUNCTIONS

The following pages provide brief descriptions of the pre-programmed functions in the TPU3. For detailed descriptions, refer to the programming note for the individual function. The Motorola **TPU Literature Pack, TPULITPAK/D**, provides a list of available programming notes.

D.1 Overview

The TPU3 contains four Kbytes of microcode ROM. This appendix defines the functions that are in the standard ROM on the MPC555 / MPC556. The TPU3 can have up to eight Kbytes of memory and a maximum of four entry tables (see **Figure D-1**).



*The DPTRAM is located at 0x30 2000 until it is switched to emulation mode. In emulation mode, the DPTRAM is accessible by the TPUs only.

Figure D-1 TPU3 Memory Map

The TPU3 can address up to eight Kbytes of memory at any one time. It has four Kbytes of internal ROM, located in Bank 0 and Bank 1, and six Kbytes of dual-ported SRAM (DPTRAM), located in Bank 0, Bank 1and Bank 2. As only one type of memory can be used at a time, the TPU3 must either use the internal ROM or the SRAM. Functions from both memory types cannot be used in conjunction.

A new feature of the TPU3 microcode ROM is the existence of two entry tables in the four Kbytes of internal ROM. Each entry table has a set of sixteen functions that define which of the two tables the TPU3 will be able to access. Only one table can be used at a time and functions from the two entry tables cannot be mixed. The default entry table is located in Bank 0. This table is identical to the standard microcode ROM in the TPU2, so any CPU code written for the TPU2 will work unchanged on the TPU3. The TPU2 and TPU3 ROMs are different than the original TPU ROM. The functions in the default entry table in Bank 0 are listed in **Table D-1**.

Table D-1 Bank 0 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	MCPWM	Multi-Channel Pulse Width Modulation
6	FQD	Fast Quadrature Decode
5	PPWA	Period/Pulse Width Accumulator
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	SPWM	Synchronized Pulse Width Modulation
0	SIOP	Serial Input/output Port

The functions in the entry table in bank one are listed in **Table D-2**.

Table D-2 Bank 1 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	Reserved	
6	FQD	Fast Quadrature Decode
5	ID	Identification
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	RWTPIN	Read/Write Timers and Pin
0	SIOP	Serial Input/output Port

The functions in the Bank 1 entry table are identical to the Bank 0 entry table functions with three exceptions. Function 1, SPWM, has been replaced by RWTPIN. This is a function that allows reads and writes to the TPU time bases and corresponding pin. Function 5, PPWA, is now an identification function in **Table D-2**. The microcode ROM revision number is provided by this function. Finally, Function 7, MCPWM, has been removed and left open for future use.

The CPU selects which entry table to use by setting the ETBANK field in the TPUMCR2 register. This register is write once after reset. Although one entry table is specified at start-up, it is possible, in some cases, to use functions from both tables without resetting the microcontroller. A customer may, for example, wish to use the ID function from Bank 1 to verify the TPU microcode version but then use the MCPWM function from Bank 0. As a customer will typically only run the ID function during system configuration, and not again after that, the Bank 1 entry table can be changed to the Bank 0 entry table using the soft reset feature of the TPU3. The procedure should be:

1. Set ETBANK field in TPUMCR2 to 0b01 to select the entry table in Bank 1
2. Run the ID function
3. Stop the TPU3 by setting the STOP bit in the TPUMCR to one.
4. Reset the TPU3 by setting the SOFTRST bit in the TPUMCR2 register
5. Wait at least nine clocks
6. Clear the SOFTRST bit in the TPUMCR2 register

The TPU3 stays in reset until the CPU clears the SOFTRST bit. After the SOFTRST bit has been cleared the TPU3 will be reset and the entry table in Bank 0 will be selected by default. To select the Bank 0 entry table, write 0b00 to the ETBANK field in

TPUMCR 2. It is good practice always to initialize any write once register to ensure an incorrect value is not accidentally written.

The descriptions below document the functions listed in **Table D-1** (Bank 0) and **Table D-2** (Bank 1) of the TPU3 ROM module.

D.2 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The period accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request.

From one to 255 period measurements can be accumulated before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability. See Motorola TPU Progamming Note ***Programmable Time Accumulator TPU Function (PTA),*** **(TPUPN06/D)**.

Figure D-2 shows all of the host interface areas for the PTA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PTA Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – High Time Accumulate 01 – Low Time Accumulate 10 – Period Accumulate, Rising 11 – Period Accumulate, Falling	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Not Used 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	MAX_COUNT	PERIOD_COUNT
0x30YYW4	LAST_TIME	
0x30YYW6	ACCUM	
0x30YYW8	HW	
0x30YYWA	LW	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-2 PTA Parameters

D.3 Queued Output Match TPU Function (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin. See Motorola TPU Progamming Note ***Queued Output Match TPU Function (QOM),*** **(TPUPN01/D)**.

Figure D-3 shows all of the host interface areas for the QOM function. The bit encodings shown in **Table D-3** describe the corresponding fields in parameter RAM.

Table D-3 QOM Bit Encoding

A	Timebase Selection
0	Use TCR1 as Timebase
1	Use TCR2 as Timebase

	Edge Selection
0	Falling Edge at Match
1	Rising Edge at Match

B:C	Reference for First Match
00	Immediate TCR Value
01	Last Event Time
10	Value Pointed to by REF_ADDR
11	Last Event Time

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	COM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Single-Shot Mode 01 – Loop Mode 10 – Continuous Mode 11 – Continuous Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize, No Pin Change 10 – Initialize, Pin Low 11 – Initialize, Pin High	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8	7 6 5 4 3 2 1	0
0x30YYW0	REF_ADDR	B	LAST_OFF_ADDR	A
0x30YYW2	LOOP_CNT		OFF_PTR	C
	(LAST_MATCH_TM)			
0x30YYW4	OFFSET_1			;
0x30YYW6	OFFSET_2			;
0x30YYW8	OFFSET_3			;
0x30YYWA	OFFSET_4			;
0x30YYWC	OFFSET_5*			;
0x30YYWE	OFFSET_6*			;
0x30YY(W=1)0	OFFSET_7*			;
0x30YY(W=1)2	OFFSET_8*			;
	: :			:
0x30YY(W=1)14	OFFSET_14*			;

*Not Available On All Channels.

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-3 QOM Parameters

D.4 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing full defininition of the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. See Motorola TPU Progamming Note ***Table Stepper Motor TPU Function (TSM),*** **(TPUPN04/D)**.

Figure D-4 and **Figure D-5** show all of the host interface areas for the TSM function when operating in master and slave mode, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	TSM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	x0 – Rotate Pin_Sequence Once Between Steps x1 – Split Mode Acceleration Table 0x – Rotate Pin_Sequence Once Between Steps 1x – Rotate Pin_Sequence Twice Between Steps	0x30YY14 – 0x30YY16
1 0	Host Service Reques	00 – No Host Service (Reset Condition) 01 – Initialize, Pin Low 10 – Initialize, Pin High 11 – Move Request (Master Only)	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1	0
0x30YYW0	DESIRED_POSITION		
0x30YYW2	CURRENT_POSITION		
0x30YYW4	TABLE_SIZE	TABLE_INDEX	
0x30YYW6	SLEW_PERIOD		S
0x30YYW8	START_PERIOD		A
0x30YYWA	PIN_SEQUENCE		
0x30YYWC			
0x30YYWE			

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-4 TSM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	TSM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	x0 – Rotate Pin_Sequence Once Between Steps x1 – Split Mode Acceleration Table 0x – Rotate Pin_Sequence Once Between Steps 1x – Rotate Pin_Sequence Twice Between Steps	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Initialize, Pin Low 10 – Initialize, Pin High 11 – Move Request (Master Only)	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YY(W+1)0	ACCEL_RATIO_2	ACCEL_RATIO_1
0x30YY(W+1)2	ACCEL_RATIO_4	ACCEL_RATIO_3
0x30YY(W+1)4	ACCEL_RATIO_6	ACCEL_RATIO_5
0x30YY(W+1)6	ACCEL_RATIO_8	ACCEL_RATIO_7
0x30YY(W+1)8	ACCEL_RATIO_10	ACCEL_RATIO_9
0x30YY(W+1)A	ACCEL_RATIO_12	ACCEL_RATIO_11
0x30YY(W+1)C*	ACCEL_RATIO_14*	ACCEL_RATIO_13*
:	:	:
0x30YY(W+3)A*	ACCEL_RATIO_36*	ACCEL_RATIO_35*

*Optional Additional Parameters not Available in all cases.
Refer to Motorola Programming Note TPUPN04/D for details

= Written By CPU
= Written by CPU and TPU
W = Channel Number
YY = 41 for TPU_A and 44 for TPU_B
= Written By TPU
= Unused Parameters

Figure D-5 TSM Parameters — Slave Mode

D.5 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.See Motorola TPU Progamming Note ***Frequency Measurement TPU Function (FQM),* (TPUPN03/D)**.

Figure D-6 shows all of the host interface areas for the FQM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Begin with Falling Edge, Single-Shot Mode 01 – Begin with Falling Edge, Continuous Mode 10 – Begin with Rising Edge, Single-Shot Mode 11 – Begin with Rising Edge, Continuous Mode	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize 11 –Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	
0x30YYW2	
0x30YYW4	CHANEL_CONTROL
0x30YYW6	WINDOW_SIZE
0x30YYW8	PULSE_COUNT
0x30YYWA	IN_WINDOW_ACCUMULATOR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Primary Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-6 FQM Parameters

D.6 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from one to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bi-directional UART channels running in excess of 9600 baud could be implemented on the TPU. See Motorola TPU Progamming Note ***Asynchronous Serial Interface TPU Function (UART),*** **(TPUPN07/D)**.

Figure D-7 and **Figure D-8** show all of the host interface areas for the UART function in transmitting and receiving modes, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	UART Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – No Parity 01 – No Parity 10 – Even Parity 11 – Odd Parity	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Transmit 11 – Receive	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	PARITY_TEMP
0x30YYW2	MATCH_RATE
0x30YYW4	TDRE | TRANSMIT_DATA_REG
0x30YYW6	DATA_SIZE
0x30YYW8	ACTUAL_BIT_COUNT
0x30YYWA	SHIFT_REGISTER
0x30YYWC	
0x30YYWA	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-7 UART Transmitter Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	UART Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – No Parity 01 – No Parity 10 – Even Parity 11 – Odd Parity	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Transmit 11 – Receive	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	PARITY_TEMP															
0x30YYW2	MATCH_RATE															
0x30YYW4	PE	RE	TRANSMIT_DATA_REG													
0x30YYW6	DATA_SIZE															
0x30YYW8	ACTUAL_BIT_COUNT															
0x30YYWA	SHIFT_REGISTER															
0x30YYWC																
0x30YYWA																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-8 UART Receiver Parameters

D.7 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels. See Motorola TPU Progamming Note ***New Input Capture/Input Transition Counter TPU Function (NITC),*** **(TPUPN08/D)**.

Figure D-9 shows all of the host interface areas for the NITC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	NITC Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Single-Shot Mode, No Links 01 – Continuous Mode, No Links 10 – Single-Shot Mode, Links 11 – Continuous Mode, Links	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize TCR Mode 10 – Initialize Parameter Mode 11 – Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12	11 10 9	8 7 6 5 4 3 2 1	0
0x30YYW0			CHANNEL_CONTROL	
0x30YYW2	START_LINK_CHANNEL	LINK_CHANNEL_COUNT	PARAM_ADDR	0
0x30YYW4	MAX_COUNT			
0x30YYW6	TRANS_COUNT			
0x30YYW8	FINAL_TRANS_TIME			
0x30YYWA	LAST_TRANS_TIME			
0x30YYWC				
0x30YYWE				

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-9 NITC Parameters

D.8 Multiphase Motor Commutation (COMM)

The COMM function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds. See Motorola TPU Progamming Note ***Multiphase Motor Commutation TPU Function (COMM),* (TPUPN09/D)**.

Figure D-10 and **Figure D-11** show all of the host interface areas for the COMM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Sensorless Match Update Mode 01 – Sensorless Match Update Mode 10 – Sensorless Link Update Mode 11 – Sensorled Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize or Force State 11 – Initialize or Force Immediate State Test	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0					START_LINK_CHANNEL				COUNTER_ADDR							
0x30YYW2	NO_OF_STATES								STATE_NO							
0x30YYW4	OFFSET															
0x30YYW6	UPDATE_PERIO															
0x30YYW8	UPPER															
0x30YYWA	LOWER															
0x30YYWC																
0x30YYWE																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-10 COMM Parameters (Part 1 of 2)

	CONTROL BITS 15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YY(W + 1)0	LENGTH STATE	0 PIN_CONFIG
0x30YY(W + 1)2	LENGTH STATE	1 PIN_CONFIG
0x30YY(W + 1)4	LENGTH STATE	2 PIN_CONFIG
0x30YY(W + 1)6	LENGTH STATE	3 PIN_CONFIG
0x30YY(W + 1)8	LENGTH STATE	4 PIN_CONFIG
0x30YY(W + 1)A	LENGTH STATE	5 PIN_CONFIG
0x30YY(W + 1)C	LENGTH STATE	6* PIN_CONFIG
0x30YY(W + 1)E	LENGTH STATE	7* PIN_CONFIG
0x30YY(W + 2)0	LENGTH STATE	8* PIN_CONFIG
0x30YY(W + 2)2	LENGTH STATE	9* PIN_CONFIG
•	•	
•	•	
•	•	
0x30YY(W + 3)A	LENGTH STATE	21* PIN_CONFIG

*Not Available on all Channels

= Written By CPU

= Written by CPU and TPU

W = Master Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-11 COMM Parameters (Part 2 of 2)

D.9 Hall Effect Decode (HALLD)

The HALLD function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding “option” switches. See Motorola TPU Progamming Note ***Hall Effect Decode TPU Function (HALLD),*** **(TPUPN10/D)**.

Figure D-12 shows all of the host interface areas for the HALLD function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Channel A 01 – Channel B 10 – Channel B 11 – Channel C (3-Channel Mode Only)	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize, 2-Channel Mode 11 – Initialize, 3-Channel Mode	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	x – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		
0x30YYW2		
0x30YYW4		
0x30YYW6	DIRECTION (1)	
0x30YYW8		STATE_NO_ADDR (2)
0x30YYWA	PINSTATE	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

NOTES:
1. Channel A Only.
2. One Channel Only (Channel B in 2-Channel Mode, Channel C in 3-Channel Mode.

Figure D-12 HALLD Parameters

D.10 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge-aligned mode uses n + 1 TPU channels for n PWMs; center-aligned mode uses 2n + 1 channels. Center-aligned mode allows a user-defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications. See Motorola TPU Progamming Note ***Multichannel PWM TPU Function (MCPWM),*** **(TPUPN05/D)**.

Figure D-13 through **Figure D-18** show the host interface areas for the MCPWM function in each mode.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center-Aligned Mode 10 – Slave B Type Center-Aligned Mode 11 – Slave C Type Center-Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize as Slave (Inverted) 10 – Initialize, as Slave (Normal) 11 – Initialize as Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	IRQ_RATE	PERIOD_COUNT
0x30YYW4	LAST_RISE_TIME	
0x30YYW6	LAST_FALL_TIME	
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-13 MCPWM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	PERIOD
0x30YYW2	HIGH_TIME
0x30YYW4	
0x30YYW6	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR
0x30YYWA	FALL_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-14 MCPWM Parameters — Slave Edge-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1A
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	NXT_B_RISE_TIME	
0x30YYW4	NXT_B_FALL_TIME	
0x30YYW6	DEAD_TIME	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	HIGH_TIME
0x30YYW2	CURRENT_HIGH_TIME
0x30YYW4	TEMP_STORAGE
0x30YYW6	
0x30YYW8	B_FALL_TIME_PTR
0x30YYWA	B_RISE_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1A
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	NXT_B_RISE_TIME	
0x30YYW4	NXT_B_FALL_TIME	
0x30YYW6	DEAD_TIME	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	HIGH_TIME
0x30YYW2	CURRENT_HIGH_TIME
0x30YYW4	TEMP_STORAGE
0x30YYW6	
0x30YYW8	B_FALL_TIME_PTR
0x30YYWA	B_RISE_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-18 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

D.11 Fast Quadrature Decode TPU Function (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free-running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function. See Motorola TPU Progamming Note ***Fast Quadrature Decode TPU Function (FQD),*** **(TPUPN02/D)**.

Figure D-19 and **Figure D-20** show the host interface areas for the FQD function for primary and secondary channels, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Primary Channel (Normal Mode) 01 – Secondary Channel (Normal Mode) 10 – Primary Channel (Fast Mode) 11 – Secondary Channel (Fast Mode)	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Read TCR1 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	x – Not Used	0x30YY0A
0	Channel Interrupt Status	xx – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	EDGE_TIME
0x30YYW2	POSITION_COUNT
0x30YYW4	TCR1_VALUE
0x30YYW6	CHAN_PINSTATE
0x30YYW8	CORR_PINSTATE_ADDR
0x30YYWA	EDGE_TIME_LSB_ADDR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-19 FQD Parameters — Primary Channel

	NAME	CONTROL BITS OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Primary Channel (Normal Mode) 01 – Secondary Channel (Normal Mode) 10 – Primary Channel (Fast Mode) 11 – Secondary Channel (Fast Mode)	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Read TCR1 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	x – Not Used	0x30YY0A
0	Channel Interrupt Status	xx – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	
0x30YYW2	
0x30YYW4	TCR1_VALUE
0x30YYW6	CHAN_PINSTATE
0x30YYW8	CORR_PINSTATE_ADDR
0x30YYWA	EDGE_TIME_LSB_ADDR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-20 FQD Parameters — Secondary Channel

D.12 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator (PPWA) algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from one to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation.

Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From one to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (one to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter.

By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels. See Motorola TPU Progamming Note ***Period/Pulse-Width Accumulator TPU Function (PPWA),*** **(TPUPN11/D)**.

Figure D-21 shows the host interface areas and parameter RAM for the PPWA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PPWA Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Accumulate 24-Bit Periods, No Links 01 – Accumulate 16-Bit Periods, Links 10 – Accumulate 24-Bit Pulse Widths, No Links 11 – Accumulate 16-Bit Pulse Widths, Links	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Initialize 11 – Not Used	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	START_LINK_CHANNEL	LINK_CHANNEL_COUNT	CHANNEL_CONTROL
0x30YYW2	MAX_COUNT		PERIOD_COUNT
0x30YYW4	LAST_ACCUM		
0x30YYW6	ACCUM		
0x30YYW8	ACCUM_RATE		PPWA_UB
0x30YYWA	PPWA_LW		
0x30YYWC			
0x30YYWE			

= Written By CPU

= Written by CPU and TPU

= Written By TPU

= Unused Parameters

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

NOTES:

1. The TPU does not check the value of LINK_CHANNEL_COUNT. If this parameter is NOT >0 and ≤ 8, results are unpredictible.
2. MAX_COUNT may be written at any time by the host CPU, but if the value written is ≤ PERIOD_COUNT, a period or pulse-width accumulation is terminated. If this happens, the number of periods over which the accumulation is done will not correspond to MAX_COUNT.

Figure D-21 PPWA Parameters

D.13 Output Compare (OC)

The output compare (OC) function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time
2. At a programmable delay time from a user-specified time
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} \times \text{Ratio}$$

where RATIO is a parameter supplied.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins. See Motorola TPU Progamming Note ***Output Compare TPU Function (OC),*** **(TPUPN12/D)**.

Figure D-22 shows the host interface areas and parameter RAM for the OC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	OC Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	0x – Matches and Pulses Scheduled x1 – Only Read TCR1, TCR2	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service Request 01 – Host-Initiated Pulse 10 – Not Used 11 – Initialize, Continuous Pulses	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1	0
0x30YYW0		CHANNEL_CONTROL	
0x30YYW2	OFFSET		
0x30YYW4	RATIO	REF_ADDR1	0
0x30YYW6	REF_ADDR2	0 REF_ADDR3	0
0x30YYW8	REF_TIME		
0x30YYWA	ACTUAL_MATCH_TIME		
0x30YYWC	TCR1		
0x30YYWE	TCR2		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-22 OC Parameters

D.14 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation (PWM) waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin. See Motorola TPU Progamming Note ***Pulse-Width Modulation TPU Function (PWM),*** **(TPUPN17/D)**.

Figure D-23 shows the host interface areas and parameter RAM for the PWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	xx – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Immediate Update of PWM 10 – Initialize 11 – Not Used	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	OLDRIS	
0x30YYW4	PWMHI (1, 3)	
0x30YYW6	PWMPER (2, 3)	
0x30YYW8	PWMRIS	
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B.

= Written By TPU

= Unused Parameters

Figure D-23 PWM Parameters

D.15 Discrete Input/Output (DIO)

The DIO function allows a TPU channel to be used as a digital I/O pin.

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter:

1. when a transition occurs
2. when the CPU makes a request, or
3. when a rate specified in another parameter is matched

When a pin is used as a discrete output, it is set high or low only upon request by the CPU. See Motorola TPU Progamming Note ***Discrete Input/Output TPU Function (DIO),*** **(TPUPN18/D)**.

Figure D-24 shows the host interface areas for the DIO function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	DIO Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Update on Transition 01 – Update at Match Rate 10 – Update on HSR 11 11 – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Drive Pin High 10 – Drive Pin Low 11 – Initialize	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	PIN_LEVEL	
0x30YYW4	MATCH_RATE	
0x30YYW6		
0x30YYW8		
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-24 DIO Parameters

D.16 Synchronized Pulse-Width Modulation (SPWM)

The SPWM function generates a pulse-width modulated waveform (PWM). The CPU can change the period or high time of the waveform at any time. Three different operating modes allow the function to maintain complex timing relationships between channels without CPU intervention.

The SPWM output waveform duty cycle excludes 0% and 100%. If a PWM does not need to maintain a time relationship to another PWM, the PWM function should be used instead. See Motorola TPU Progamming Note ***Synchronized Pulse-Width Modulation TPU Function (SPWM),*** **(TPUPN19/D)**.

Figure D-25 and **Figure D-26** show all of the host interface areas for the SPWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	SPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Mode 0 01 – Mode 1 10 – Mode 2 11 – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service Request 01 – Not Used 10 – Initialize 11 – Immediate Update (Mode 1)	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	LASTRISE	CHANNEL_CONTROL
0x30YYW2	NEXTRISE	
0x30YYW4	HIGH_TIME	
0x30YYW6	PERIOD	
0x30YYW8		REF_ADDR1
0x30YYWA	DELAY	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-25 SPWM Parameters, Part 1 of 2

PARAMETER RAM (MODE 1)

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	LASTRISE	CHANNEL_CONTROL
0x30YYW2	NEXTRISE	
0x30YYW4	HIGH_TIME	
0x30YYW6	DELAY	
0x30YYW8	REF_ADDR1	REF_ADDR2
0x30YYWA	REF_VALUE	
0x30YYWC		
0x30YYWE		

PARAMETER RAM (MODE 2)

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
0x30YYW0	LASTRISE	CHANNEL_CONTROL	
0x30YYW2	NEXTRISE		
0x30YYW4	HIGH_TIME		
0x30YYW6	PERIOD		
0x30YYW8	START_LINK_ CHANNEL	LINK_CHANNEL _COUNT	REF_ADDR1
0x30YYWA	DELAY		
0x30YYWC			
0x30YYWE			

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-26 SPWM Parameters, Part 2 of 2

D.17 Read / Write Timers and Pin TPU Function (RWTPIN)

The RWTPIN TPU function enables the CPU to read both the TCR1 and TCR2 timer counters via locations in PRAM and selectively load TCR1 or TCR2 with a CPU supplied value contained in PRAM. The function also allows control of the pin state and direction of the RWTPIN channel.

A pin state parameter is maintained in PRAM and is updated upon every service request. It can contain a value of the current pinstate whether the pin is programmed as an input or output.

The function also receives links, and upon receipt will read the two TCRs into PRAM, updating the pinstate parameter. A maskable interrupt request to the CPU is generated.

The CPU can control the channel pin, the channel pin and the TCRs, or just the TCRs. To control the pin only, the 'read TCR' option is used and the values returned ignored. Controlling the TCRs without effect on the pin allows this function to be run on a TPU channel whose pin is being controlled by a different function running on another channel (e.g., a slave stepper motor channel). See Motorola TPU Progamming Note ***Using The TPU Function Library And TPU Emulation Mode,*** **(TPUPN00/D)**.

Figure D-27 shows all of the host interface areas for the PTA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	Read/Write Timers and Pin Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	XX – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Action 01 – Read TCRs and read/write pin 10 – Write TCR1, read TCRs and read/write pin 11 – Write TCR2, read TCRs and read/write pin	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		
0x30YYW2		
0x30YYW4		CHANNEL_CONTROL
0x30YYW6	CHANNEL_PIN_STATE	
0x30YYW8	TCR1_VALUE	
0x30YYWA	TCR2_VALUE	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-27 RWTPIN Parameters

D.18 ID TPU Function (ID)

This is a simple function that returns the version of the TPU ROM on the current device.

Figure D-28 shows all of the host interface areas for the ID function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	Identification Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	XX – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Action 01 – Read Read TPU ROM version 10 – Not Used 11 – Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	TPU3_ID	ROM_REVISION
0x30YYW2		
0x30YYW4		
0x30YYW6		
0x30YYW8		
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-28 ID Parameters

D.19 Serial Input/Output Port (SIOP)

The serial input/output port (SIOP) TPU function uses two or three TPU channels to form a uni- or bi-directional synchronous serial port that can be used to communicate with a wide variety of devices. Features such as baud rate and transfer size are user programmable. The function can also produce a clock-only, when it uses just one channel.

The SIOP TPU function has been designed to closely resemble the SIOP hardware port found on some Motorola MCUs and can be used to add serial capabilities to a device without a serial port, or extend the capabilities of one with a hardware synchronous port.

SIOP operates in master mode (i.e., the TPU always generates the clock) and the following features are programmable:

1. Choice of clock-only (one channel), clock + transmit (two channels), clock + receive (two channels) or clock + transmit + receive (three channels) operating modes
2. Baud rate period is freely programmable over a 15-bit range of TCR1 counts
3. Selection of msb or lsb first shift direction
4. Variable transfer size from one to 16 bits
5. Clock polarity is programmable

When a transfer of data is complete the SIOP function notifies the host CPU by issuing an interrupt request. The arrangement of the multiple SIOP channels is fixed: the data out channel is the channel above the clock channel and the data in channel is the channel below the clock channel. In clock-only or uni-directional mode, the unused TPU channels are free to run other TPU functions. Two possible SIOP configurations are show in **Figure D-29**.

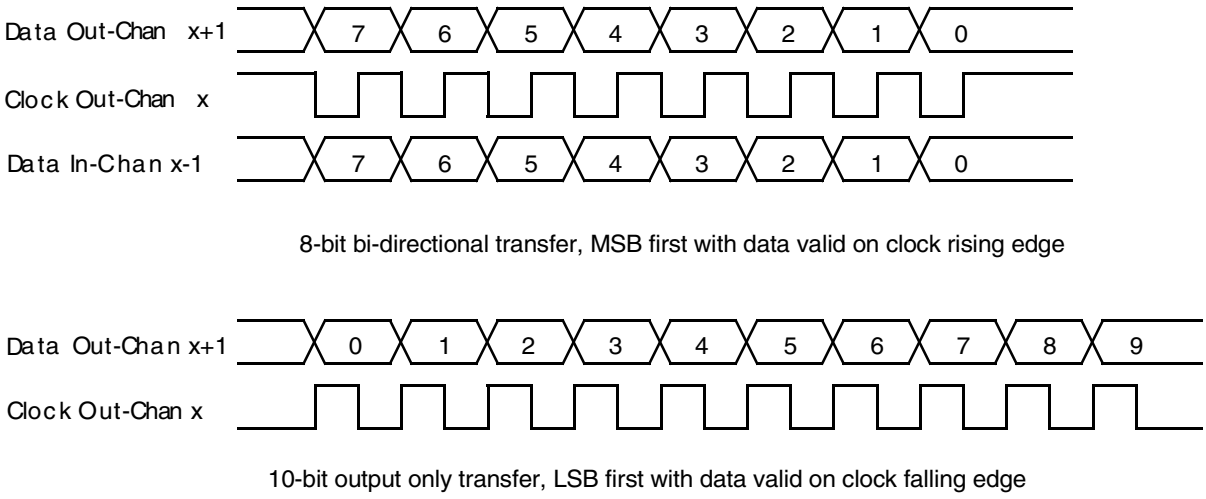


Figure D-29 Two Possible SIOP Configurations

D.19.1 Parameters

Figure D-30 shows the host interface areas and parameter RAM for the SIOP function. The following sections describe these parameters. Note that only the clock channel requires any programming — the data in and out channels are entirely under TPU microcode control.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	SIOP Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Clock Channel Active Only, No Data Transfer 01 – D_{OUT} Channels Active, No Data Receive 10 – Clock and D_{IN} Channels Active, No Data Transmit 11 – Full Bidirectional Tansmit and Receive	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – No Action 10 – No Action 11 – Initialize Clock Channel and Start Transfer	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	S							CHANNEL_CONTROL								
0x30YYW2	HALF-PERIOD															
0x30YYW4													BIT_COUNT			
0x30YYW6													XFER_SIZE			
0x30YYW8	DATA															
0x30YYWA																
0x30YYWC																
0x30YYWE																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 for TPU_A and 44 for TPU_B

= Written By TPU

= Unused Parameters

Figure D-30 SIOP Parameters

D.19.1.1 CHAN_CONTROL

This 9-bit CPU written parameter is used to setup the clock polarity for the SIOP data transfer. The valid values for CHAN_CONTROL for this function are given in the table below. CHAN_CONTROL must be written by the host prior to issuing the host service request (HSR) to initialize the function.

Table D-4 SIOP Function Valid CHAN_Control Options

CHAN_CONTROL[1] 8 7 6 5 4 3 2 1 0	Resulting Action
0 1 0 0 0 1 1 0 1	Data valid on clock Falling edge.
0 1 0 0 0 1 1 1 0	Data valid on clock Rising edge.

NOTES:
1. Other values of CHAN_CONTROL may result in indeterminate operation.

D.19.1.2 BIT_D

BIT_D is a CPU written bit that determines the direction of shift of the SIOP data. If BIT_D is zero then SIOP_DATA is right shifted (lsb first). If BIT_D is one then SIOP_DATA is left shifted (msb first).

D.19.1.3 HALF_PERIOD

This CPU-written parameter defines the baud rate of the SIOP function. The value contained in HALF_PERIOD is the number of TCR1 counts for a half SIOP clock period (e.g., for a 50 KHz baud rate, with a TCR1 period of 240 ns, the value [(1/50 KHz)/2]/240 ns = 42 should be written to HALF_PERIOD. The range for HALF_PERIOD is 1 to 0x8000, although the minimum value in practice will be limited by other system conditions. See notes on use and performance of SIOP function.

D.19.1.4 BIT_COUNT

This parameter is used by the TPU to count down the number bits remaining while a transfer is in progress. During the SIOP initialization state, BIT_COUNT is loaded with the value contained in XFER_SIZE. It is then decremented as the data is transferred and when it reaches zero, the transfer is complete and the TPU issues an interrupt request to the CPU.

D.19.1.5 XFER_SIZE

This CPU-written parameter determines the number of bits that make up a data transfer. During initialization, XFER_SIZE is copied into BIT_COUNT. XFER_SIZE is shown as a 5-bit parameter to match the maximum size of 16 bits in SIOP_DATA, although the TPU uses the whole word location. For normal use, XFER_SIZE should be in the range 1-to-16.

D.19.1.6 SIOP_DATA

This parameter is the data register for all SIOP transfers. Data is shifted out of one end of SIOP_DATA and shifted in at the other end, the shift direction being determined

by the value of BIT_D. In output only mode, zero will be shifted into SIOP_DATA and in input only mode, the data shifted out is ignored. In clock-only mode SIOP_DATA is still shifted. Note that no 'justifying' of SIOP_DATA is performed by the TPU, (e.g., if an 8-bit bi-directional transfer is made, shifting lsb first, then the bottom byte of SIOP_DATA will be shifted out and the input data will be shifted into the upper byte of SIOP_DATA).

NOTE

SIOP_DATA is not buffered. The CPU should only access it between completion of one transfer and the start of the next.

D.19.2 Host CPU Initialization of the SIOP Function

The CPU initializes the SIOP function by:

1. Disabling the channel by clearing the two channel priority bits
2. Selecting the SIOP function on the channel by writing the assigned SIOP function number to the function select bits
3. Writing CHAN_CONTROL in the clock channel parameter RAM
4. Writing HALF_PERIOD, BIT_D and XFER_SIZE in the clock channel parameter RAM to determine the speed, shift direction and size of the transfer
5. Writing SIOP_DATA if the data output is to be used
6. Selecting the required operating mode via the two host sequence bits
7. Issuing a host service request type 0b11
8. Enabling service by assigning H, M or L priority to the clock channel via the two channel priority bits

The TPU then starts the data transfer, and issues an interrupt request when the transfer is complete.

Once the function has been initialized, the CPU only needs to write SIOP_DATA with the new data and issue a HSR 0b11 to initiate a new transfer. In input or clock-only modes, just the HSR 0b11 is required.

D.19.3 SIOP Function Performance

Like all TPU functions, the performance limit of the SIOP function in a given application is dependent to some extent on the service time (latency) associated with other active TPU channels. This is due to the operational nature of the scheduler. Where two channels are being used for a uni-directional system, and no other TPU channels are active, the maximum baud rate is approximately 230 KHz at a bus speed of 16.77 MHz. A three-channel bi-directional system under the same conditions has a maximum baud rate of approximately 200 KHz. When more TPU channels are active, these performance figures will be degraded, however, the scheduler assures that the worst case latency in any TPU application can be closely approximated. It is recommended that the guidelines given in the TPU reference manual be used along with the information given in the SIOP state timing table to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

Table D-5 SIOP State Timing[1]

State Number and Name	Max. CPU Clock Cycles	Number of RAM Accesses by TPU
S1 SIOP_INIT		
HSQ = X0	28	7
X1	38	7
S2 DATA_OUT		
HSQ = X0	14	4
X1	24	4
S3 DATA_IN		
HSQ = 0X	14	4
1X	28	6

NOTES:
1. Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks).

D.19.3.1 XFER_SIZE Greater Than 16

XFER_SIZE is normally programmed to be in the range 1-to-16 to match the size of SIOP_DATA, and has thus been shown as a 5-bit value in the host interface diagram. However, the TPU actually uses all 16 bits of the XFER_SIZE parameter when loading BIT_COUNT. In some unusual circumstances this can be used. If an input device is producing a data stream of greater than 16 bits then manipulation of XFER_SIZE will allow selective capturing of the data. In clock-only mode, the extended XFER_SIZE can be used to generate up to 0xFFFF clocks.

D.19.3.2 Data Positioning

As stated above, no 'justifying' of the data position in SIOP_DATA is performed by the TPU. This means that in the case of a byte transfer, the data output will be sourced from one byte and the data input will shift into the other byte. This rule holds for all data size options except 16 bits when the full SIOP_DATA register is used for both data output and input.

D.19.3.3 Data Timing

In the example given in **Figure D-31**, the data output transitions are shown as being completely synchronous with the relevant clock edge and it is assumed that the data input is latched exactly on the opposite clock edge. This is the simplest way to show the examples, but is not strictly true. Since the TPU is a multi-tasking system, and the data channels are manipulated directly by microcode software while servicing the clock edge, there is a finite delay between the relevant clock edge and the data-out being valid or the data-in being latched. This delay is equivalent to the latency in servicing the clock channel due to other TPU activity and is shown as 'Td' in the timing diagram. Td is the delay between the clock edge and the next output data being valid and also the delay between the opposite clock edge and the input data being read. For the vast majority of applications, the delay Td will not present a problem and can be ignored. Only for a system which heavily loads the TPU should calculations be made for the worst case latency for the SIOP clock channel + actual SIOP service time (= Td) and ensure that the baud rate is chosen such that HALF_PERIOD - Td is not less

that the minimum setup time of the receiving device. A transmitting device must also hold data valid for a minimum time of Td after the clock.

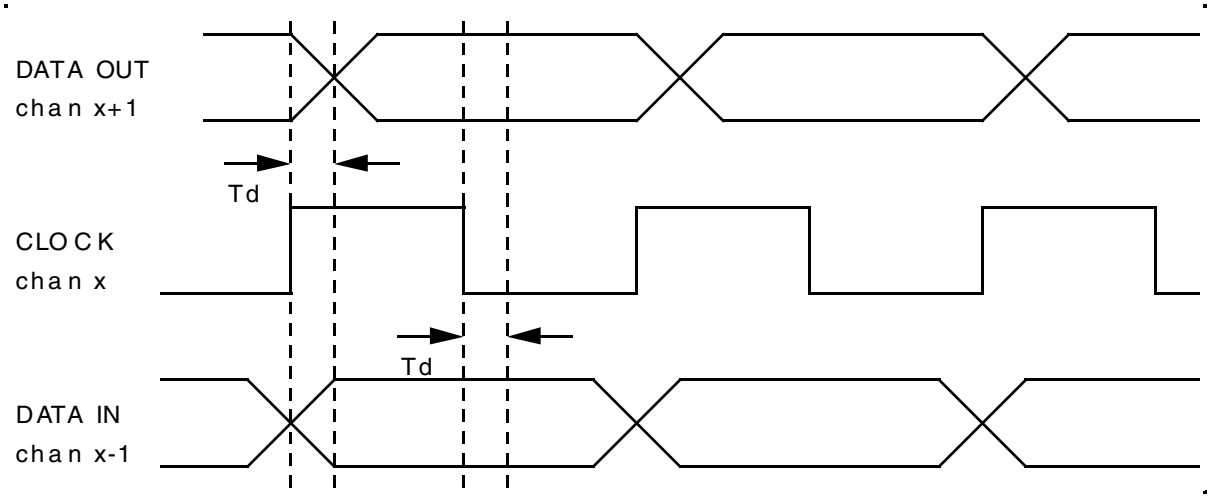


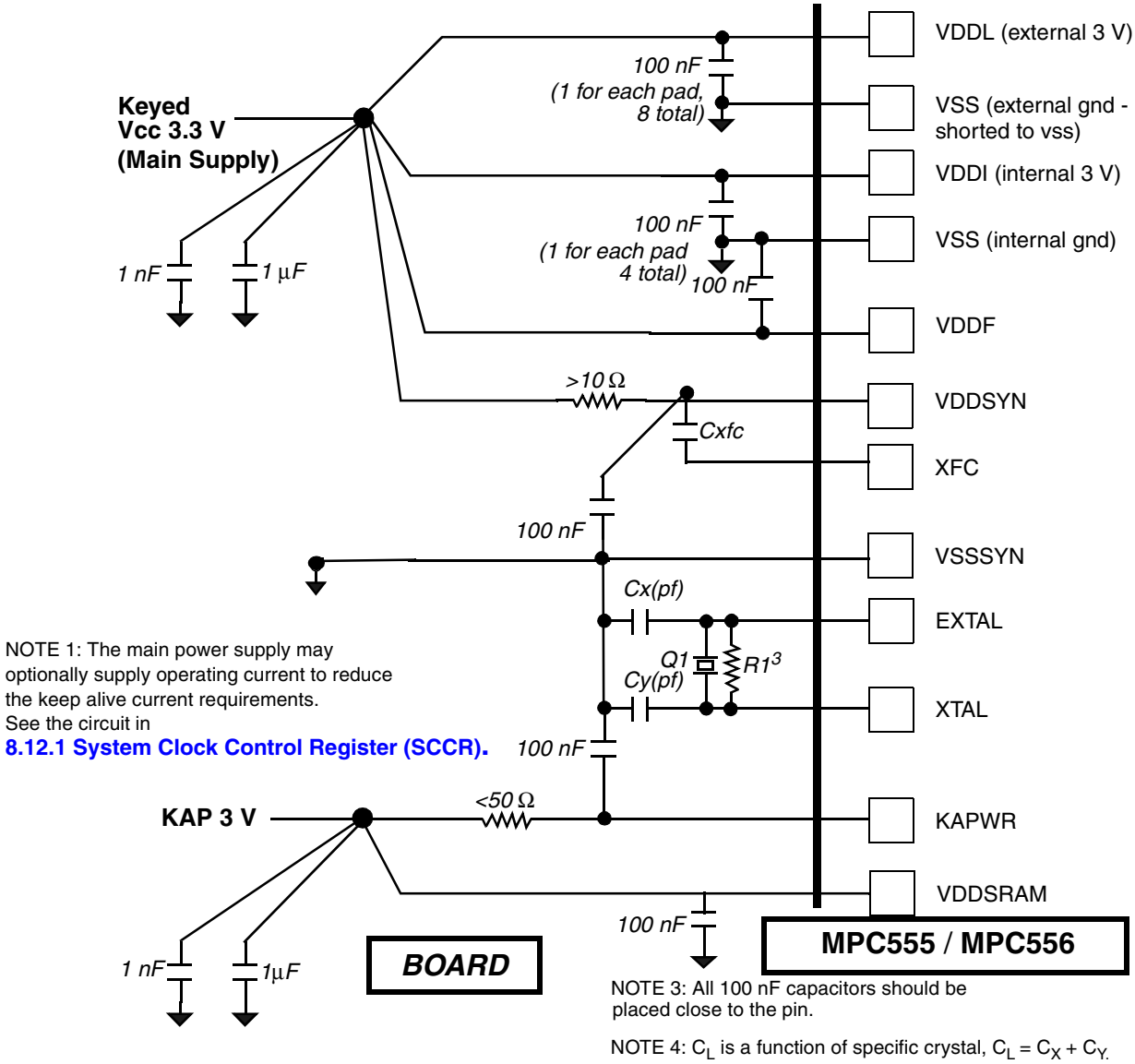
Figure D-31 SIOP Function Data Transition Example

APPENDIX E
CLOCK AND BOARD GUIDELINES

E.1 INTRODUCTION

The MPC555 / MPC556 built-in PLL, oscillator, and other analog and sensitive circuits, require that the board design follow special layout guidelines to ensure proper operation of the chip clocks. This appendix describes how the clock supplies and external components should be connected in the board. These guidelines must be fulfilled to reduce switching noise which is generated on internal and external buses during operation. Any noise injected into the sensitive clock and PLL logic reduces clock performance. The USIU maintains a PLL loss-of-lock warning indication that can be used to determine the clock stability in the MPC555 / MPC556.

E.2 MPC555 / MPC556 Family Power Distribution



NOTE 1: The main power supply may optionally supply operating current to reduce the keep alive current requirements. See the circuit in **8.12.1 System Clock Control Register (SCCR).**

NOTE 2: Resistor R_1 is currently not required. Space should be left on the board to add it in the future if necessary.

NOTE 3: All 100 nF capacitors should be placed close to the pin.

NOTE 4: C_L is a function of specific crystal, $C_L = C_X + C_Y$.

Figure E-1 MPC555 / MPC556 Family Power Distribution Diagram — 3 V

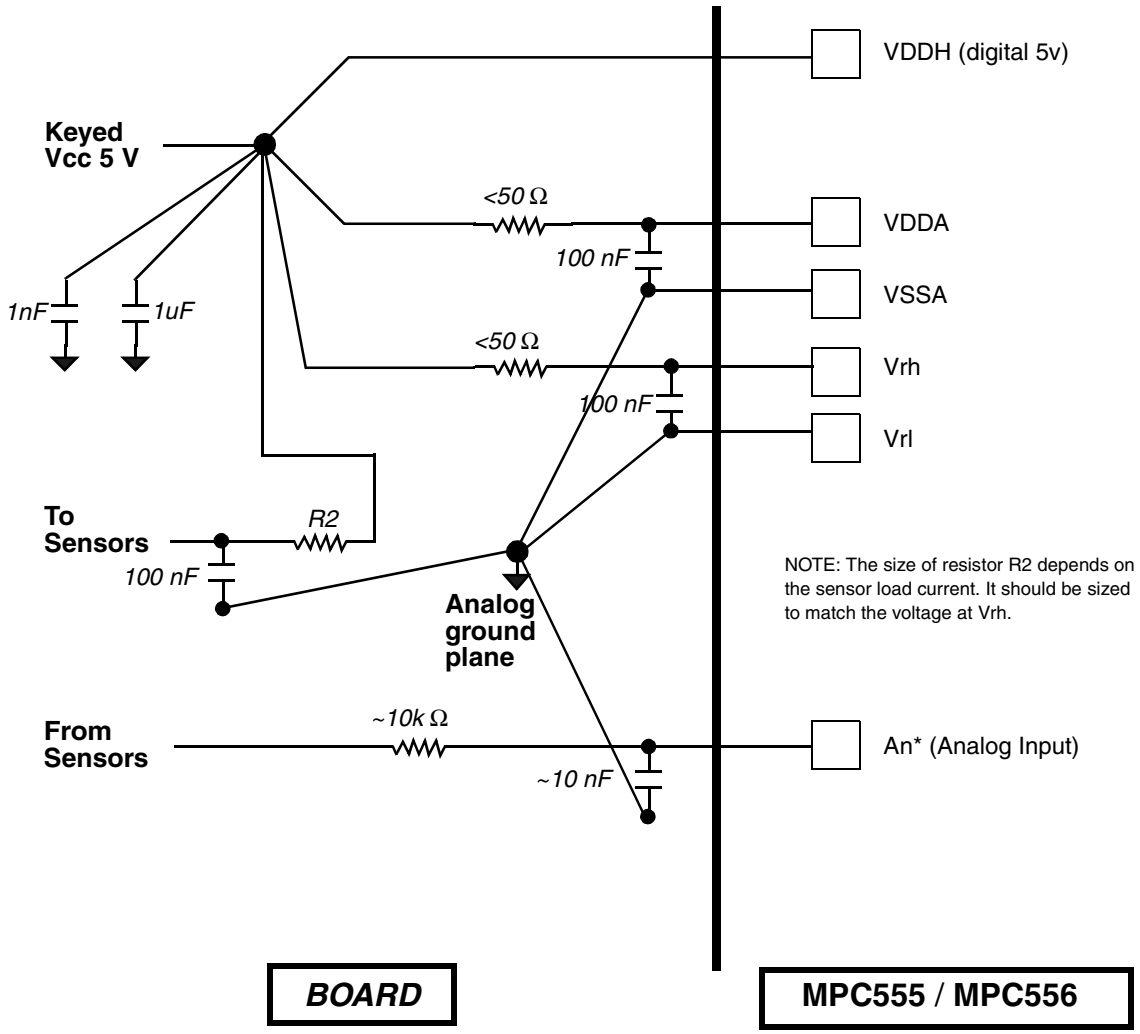


Figure E-2 MPC555 / MPC556 Family Power Distribution Diagram — 5 V and Analog

E.3 PLL and Crystal Oscillator External Components

E.3.1 Crystal Oscillator External Components

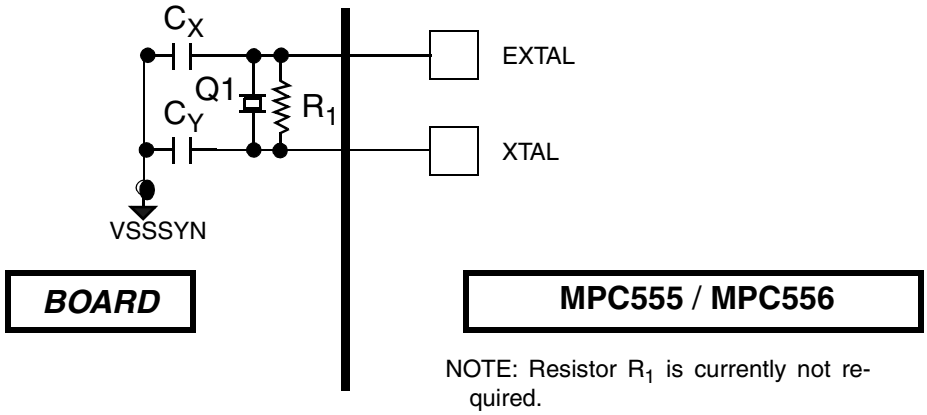


Figure E-3 Crystal Oscillator Circuit

Table E-1 External Components Value For Different Crystals (Q1)

Component	NDK CP32C 20 MHz	KINSEKI CX-11F 20 MHz	MURATA CCSTC 4 MHz	Units
C_L[1]	6	14	—	pF
R_1[3]	1MEG[3]	1MEG[3]	1MEG[3]	Ohm
C_X	6	16	—[2]	pF
C_Y	6	16	—	pF

NOTES:
1. C_L according to crystal specification, $C_L = C_X + C_Y$.
2. The Murata ceramic resonator includes the load capacitors. (8pF should be selected)
3. Resistor R_1 is currently not required. Space should be left on the board to add it in the future if necessary.

Load capacitances specified in the table include all stray capacitance.

Tolerance of the capacitors are ±10%.

The oscillator capacitors were calculated as follows:

$$C_{XX} = C_{YY} = 2C_L$$

$$C_{XX} = C_X + C_{pad} + C_{socket}$$

$$C_{YY} = C_Y + C_{pad} + C_{socket}$$

Where C_X is “real” capacitor

C_{pad} is pad capacitance

C_{socket} is socket and trace capacitance

C_L is load capacitance

Capacitance of the socket $C \leq 1pF$

Capacitance of the board trace $C \leq 1pF$. It should be low since the crystal must be located very close to the chip.

Capacitance of the MPC555 / MPC556 XTAL pin is $C_{pad} \sim 7pF$

Capacitance of the MPC555 / MPC556 EXTAL pin is $C_{pad} \sim 7pF$

Tolerance of the capacitors taken into account is ±10%

E.3.2 KAPWR Filtering

KAPWR pin is the MPC555 / MPC556 keep-alive power. KAPWR is used for the crystal oscillator circuit, and should be isolated from the noisy supplies. It is recommended that an RC filter be used on KAPWR, or bypass capacitors which are located as close as possible to the part.

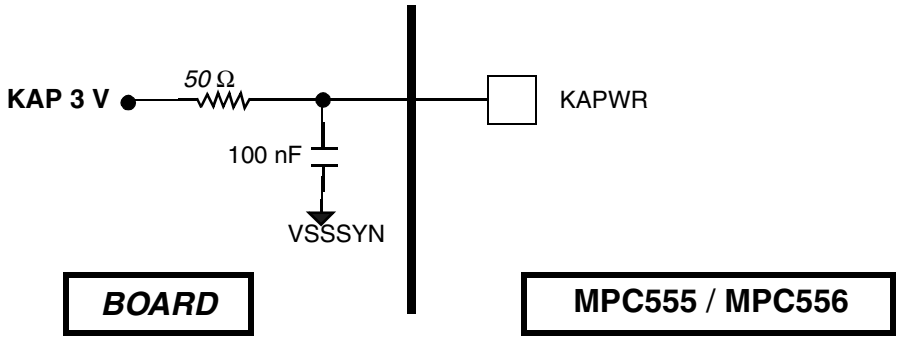


Figure E-4 RC Filter Example

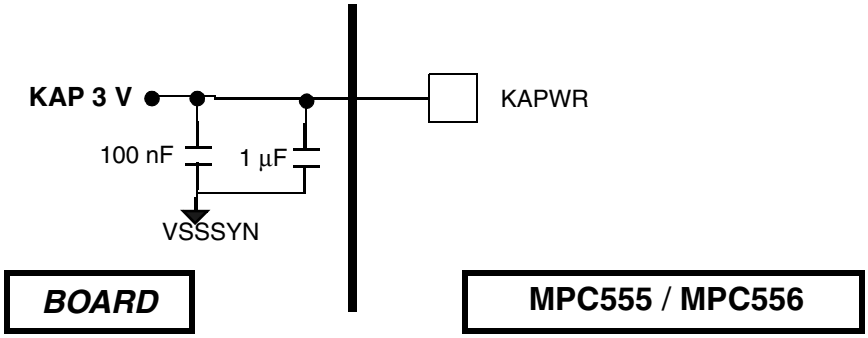


Figure E-5 Bypass Capacitors Example (Alternative)

E.3.3 PLL External Components

VDDSYN and VSSSYN are the PLL dedicated power supplies. These supplies must be used only for the PLL and isolated from all other noisy signals in the board. VDDSYN could be isolated with RC filter (see **Figure E-1**), or LC filter. The maximum noise allowed on VDDSYN, and VSSSYN is 50 mV with typical cut-off frequency of 500 Hz.

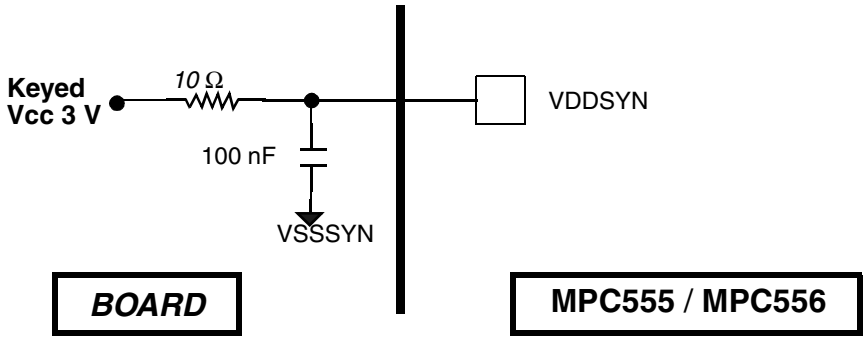


Figure E-6 RC Filter Example

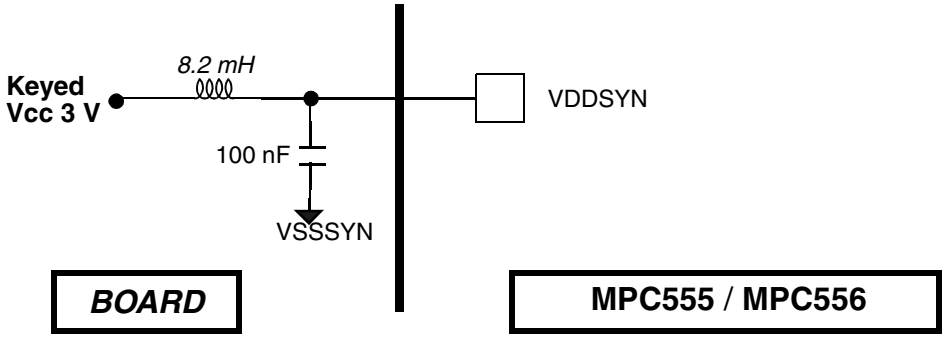


Figure E-7 LC Filter Example (Alternative)

E.3.4 PLL Off-Chip Capacitor C_{XFC}

C_{XFC} is the PLL feedback capacitor. It must be located as close as possible to the XFC and VDDSYN pads. The maximum noise allowed on XFC is 50 mV peak to peak with typical cut-off frequency of 500 Hz.

The required value for C_{FXC} are determined by the following two cases. MF is the multiplication factor as defined in the PLPRCR register (refer to **Table 8-10**).

1. $0 < (MF + 1) < 4$ $\quad C_{XFC} = (680 \times (MF + 1) - 120)$ pF
2. $(MF + 1) \geq 4$ $\quad C_{XFC} = 1100 \times (MF + 1)$ pF

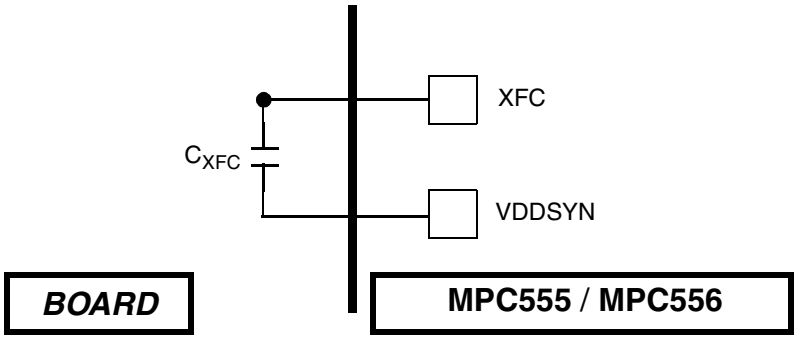


Figure E-8 PLL Off-Chip Capacitor Example

E.4 Clock Oscillator and PLL External Components Layout Requirements

E.4.1 Traces and Placement

Traces connecting capacitors, crystal, resistor should be as short as possible. Therefore, the components (crystal, resistor and capacitors) should be placed as close to the oscillator pins of the MPC555 / MPC556 as possible.

The voltage to the VDDSYN pin should be well regulated and the pin should be provided with an extremely low impedance path from the VDDSYN filter to the VDDSYN pad.

The VSSSYN pin should be provided with an extremely low impedance path in the board. All the filters for the supplies should be located as close as possible to the chip package. It is recommended to design individual VSSSYN plane to improve VSSSYN quietness.

E.4.2 Grounding/Guarding

The traces from the oscillator pins and PLL pins of the MPC555 / MPC556 should be guarded from all other traces to reduce crosstalk. It can be provided by keeping other traces away from the oscillator circuit and placing a ground plane around the components and traces.

APPENDIX F
MEMORY ACCESS TIMING

F.1 Introduction

Table F-1 lists all possible memory access timing to internal and external memory combinations. The clock values show the number of clocks from the moment an address is valid on a specific bus, until data is back on that same bus. The following assumptions were used when compiling the information:

- The arbitration time was ignored. The values assume that the bus (or buses) involved in a transaction was in the IDLE state when the transaction needs that bus.
- The UIMB works in a mode of 1:1. This is relevant for IMB accesses values. In the case of 2:1 mode, the clock latency for a cycle on the IMB should be doubled. (each IMB access takes two clocks.)
- The basic delay of external bus to U-bus is four clocks (external master case).
- All IMB accesses are assumed to be 16-bit accesses only. If 32-bit accesses are used, then each such IMB access is split into two separate 16-bit cycles with normal IMB performance for each.

Table F-1 Memory Access Times Using Different Buses

	INTERNAL				EXTERNAL RAM/ FLASH		SHOW CYCLE	
	FLASH	RAM	IMB	SIU	Internal Memory Mapped External	Non-mapped Internal Memory	Write	Read
RCPU Load/Store	3/4[1]	1	6	5	4+N[2]	4+N	2	2
RCPU Instruction Fetches	2-1-1-1-1...	3[3]	-	-	2+N	2+N	-	1[4]
Peripheral Mode (ony ext. master is active)	4/5	6	7	6				
Slave Mode (both ext. & int. CPUs are active)	5/6	7	8	7				

NOTES:
1. "/" comes for on/off page flash access.
2. N is the number of clocks from external address valid till external data valid in the case of read cycle. In the case of zero wait states, N = 2.
3. Assuming BBC is parked on U-BUS.
4. Until address is valid on external pins

Table F-2 Timing Examples

Access	# Clock													Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	
Load/Store -> Ebus	L	U	E											6[1]
				E	U	L								
Load/Store -> IMB 16 bits	L	U	IMB											6
				IMB	U	L								
Instruction Fetch-> cmf new page 3 concecutive accesses	C,U													2
		U[2]												
		C,U												1
		U												
			C,U											1
			U											
Instruction Fetch-> cmf new page Load/Store -> IMB	C,U													2
		U												
	L	U	IMB											6
				IMB	U	L								
Instruction Fetch-> cmf new page Load/Store -> IMB		C	U											6
						U								
	L	U	IMB											6
				IMB	U	L								
External Bus-> cmf new page	E		U											5
				U	E									
External Bus-> IMB	E		U	IMB										7
					IMB	U	E							
Instruction Fetch-> cmf 2 concecutive accesses & External Bus-> cmf		C,U												2
			U											
			C	-[3]	-	-	-	-	-	-	-	U		11
													U	
	E		retry	E[4]		U								8
							U	E						

NOTES:

1. N is the number of clocks from external address valid until external data valid in the case of read cycle. In the case of zero wait states, N = 2.
2. Core instruction fetch data bus is usualy the UBUS
3. 8 clocks are dedicated for external access, and internal accesses are denied.
4. Assuming the external master immediately retries

LEGEND

Shaded areas = address phase ; Non-shaded areas = data phase

APPENDIX G
ELECTRICAL CHARACTERISTICS

This section contains detailed information on power considerations, DC/AC electrical characteristics, and AC timing characteristics of the MPC555 / MPC556. The MPC555 / MPC556 is designed to operate at 40 MHz with nominal 3.3-V and 5.0-V power supplies.

G.1 Absolute Maximum Ratings (V_{SS} = 0 V)

Table G-1 Absolute Maximum Ratings

Rating	Symbol	Min. Value	Max. Value	Unit
3-V Supply Voltage[1,2]	V_{DDL}/V_{DDI}	-6.0	4.0	V
Flash Supply Voltages[3]	V_{PP}	-6.0	6.0	V
Flash Core Voltage[1]	V_{DDF}	-6.0	4.0	V
Oscillator, Keep Alive Reg. Supply Voltage[1]	KAPWR	-6.0	4.0	V
SRAM Supply Voltage[1]	V_{DDSRAM}	-6.0	4.0	V
Clock Synthesizer Supply Voltage[1]	V_{DDSYN}	-6.0	4.0	V
QADC Supply Voltage[4]	V_{DDA}	-6.0	6.0	V
5-V Supply Voltage	V_{DDH}	-0.3	6.0	V
DC Input Voltages[5]	V_{IN}	V_{SS} -0.3	V_{DDH} +0.3	V
Reference Supply V_{RH}, with Reference to V_{RL}	$V_{RH} – V_{RL}$	-0.3	6.0	V
V_{SS} Differential Voltage	$V_{SS} – V_{SSA}$	-0.1	0.1	V
V_{DD} Differential Voltage[6]	$V_{DDL} – V_{DDA}$	-6.0	4.0	V
V_{RL} to V_{SSA} Differential Voltage	$V_{RL} – V_{SSA}$	-6.0	0.3	V
Maximum Input Current per pin [7, 8, 9]	I_{MA}	-25	25	mA
QADC Maximum Input Current per Pin	I_{max}	-25	25	mA
Operating Temperature Range (Packaged)	T_A	-40 (T_L)	+125 (T_H)	°C
Operating Temperature Range (Die Form) (Maximum junction temperature for packaged devices)	T_J	-40	+150	°C
Storage Temperature Range	T_{STG}	-55	+155	°C
Maximum Solder Temperature[10]	T_{sdr}		220	°C
Moisture Sensitivity Level[11]	MSL		3	

NOTES:
1. For internal digital supply of V_{DDL} = 3.3 V typical.
2. V_{DDL} and V_{DDI} should always be connected to the same potential with no differential voltage.

3. During program/erase operation the value of V_{PP} must be 5.0 V ± 5%.
4. V_{DDA} = 5.0 V ±10%.
5. All 3-V input pins are 5-V tolerant. This applies to all input pins.
6. Refers to allowed random sequencing of power supplies.
7. Maximum continuous current on I/O pins provided the overall power dissipation is below the power dissipation of the package. Proper operation is not guaranteed at this condition.
8. Condition applies to one pin at a time.
9. Transitions within the limit do not affect device reliability or cause permanent damage. Exceeding limit may cause permanent conversion error on stressed channels and on unstressed channels.
10. Solder profile per CDF-AEC-Q100, current revision.
11. Moisture sensitivity per JEDEC test method A112.

Functional operating conditions are given in **G.7 DC Electrical Characteristics**. Absolute maximum ratings are stress ratings only, and functional operation at the maximum is not guaranteed. Stress beyond those listed may affect device reliability or cause permanent damage to the device.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

G.2 Target Failure Rate

Target failure rate of **TBD** ppm pending characterization and evaluation of qualifiable silicon.

G.3 Package

The MPC555 / MPC556 is available in two forms, packaged and die. The package is a 272-ball PBGA, Motorola case outline 1135A-01 (See **Figure 2-1** of the MPC555 User's Manual for a case drawing or contact Motorola.) For die characteristics, contact the Motorola factory.

G.4 EMI Characteristics

G.4.1 Reference Documents

The documents referenced for the EMC testing of MPC555 / MPC556 are listed below.

1. SAE J1752/3 Issued 1995-03
2. VDE UK 767.14/ZVEI-Ad-Hoc-HL-AK Version1.0 May 1994

G.4.2 Definitions and Acronyms

EMC — Electromagnetic compatibility

EMI — Electromagnetic interference

TEM cell — Transverse electromagnetic mode cell

G.4.3 Testing Characteristics

1. Scan range: 150 kHz — 1000 MHz
2. Operating frequency: 20 MHz, 40 MHz
3. Operating voltages: 3.3 V, 5.0 V
4. Max spikes: 50 dBuV
5. I/O port waveforms: 50% duty cycle @ 100 µs period
6. Temperature: 25°C (-40°C, 125°C if available)

G.5 Thermal Characteristics

Table G-2 Thermal Characteristics

Characteristic	Symbol	Value	Unit
BGA Package Thermal Resistance, Junction to Ambient — Natural Convection	$R_{\theta JA}$	42.8[1,2]	°C/W
BGA Package Thermal Resistance, Junction to Ambient — Four layer (2s2p) board, natural convection	$R_{\theta MA}$	30.4[3,4]	°C/W
BGA Package Thermal Resistance, Junction to Board	$R_{\theta JB}$	19.9[5]	°C/W
BGA Package Thermal Resistance, Junction to Case (top)	$R_{\theta JC}$	6.3[6]	°C/W
BGA Package Thermal Resistance, Junction to Package Top, Natural Convection	θ_{JT}	2.7[7]	°C/W

NOTES:

1. Junction temperature is a function of on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and the board thermal resistance.
2. Per SEMI G38-87 and JESD51-2 with the board horizontal.
3. Junction temperature is a function of on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and the board thermal resistance.
4. Per JESD51-6 with the board horizontal.
5. Thermal resistance between the die and the printed circuit board (Four layer [2s2p] board, natural convection).
6. Indicates the thermal resistance between the die and the case top surface as measured by the cold plate method (MIL SPEC-883 Method 1012.1) with the cold plate temperature used for the case temperature.
7. Thermal characterization parameter indicating the temperature difference between package top and the junction temperature per EIA/JESD51-2.

An estimation of the chip junction temperature, T_J, in °C can be obtained from the equation:

$$T_J = T_A + (R_{\theta JA} \times P_D)$$

where:

T_A = ambient temperature (°C)

$R_{\theta JA}$ = package junction to ambient resistance (°C/W)

P_D = power dissipation in package

The junction to ambient thermal resistance is an industry standard value which provides a quick and easy estimation of thermal performance. Unfortunately, the answer is only an estimate; test cases have demonstrated that errors of a factor of two are possible. As a result, more detailed thermal characterization is supplied.

Historically, the thermal resistance has frequently been expressed as the sum of a junction to case thermal resistance and a case to ambient thermal resistance:

$$R_{\theta JA} = R_{\theta JC} + R_{\theta CA}$$

where:

$R_{\theta JA}$ = junction to ambient thermal resistance (°C/W)

$R_{\theta JC}$ = junction to case thermal resistance (°C/W)

$R_{\theta JA}$ = case to ambient thermal resistance (°C/W)

$R_{\theta JC}$ is device related and cannot be influenced by the user. The user controls the thermal environment to change the case to ambient thermal resistance, $R_{\theta CA}$. For instance, the user can change the air flow around the device, add a heat sink, change the mounting arrangement on printed circuit board, or change the thermal dissipation on the printed circuit board surrounding the device. This description is most useful for ceramic packages with heat sinks where some 90% of the heat flow is through the case to the heat sink to ambient. For most packages, a better model is required.

The simplest thermal model of a package which has demonstrated reasonable accuracy (about 20%) is a two resistor model consisting of a junction to board and a junction to case thermal resistance. The junction to case covers the situation where a heat sink will be used or where a substantial amount of heat is dissipated from the top of the package. The junction to board thermal resistance describes the thermal performance when most of the heat is conducted to the printed circuit board. It has been observed that the thermal performance of most plastic packages and especially PBGA packages is strongly dependent on the board. temperature.

If the board temperature is known, an estimate of the junction temperature in the environment can be made using the following equation:

$$T_J = T_B + (R_{\theta JB} \times P_D)$$

where:

T_B = board temperature (°C)

$R_{\theta JB}$ = package junction to board resistance (°C/W)

P_D = power dissipation in package (W)

If the board temperature is known and the heat loss from the package case to the air can be ignored, acceptable predictions of junction temperature can be made. For this method to work, the board and board mounting must be similar to the test board used to determine the junction to board thermal resistance, namely a 2s2p (board with a power and a ground plane) and vias attaching the thermal balls to the ground plane.

When the board temperature is not known, a thermal simulation of the application is needed. The simple two resistor model can be used with the thermal simulation of the application [2], or a more accurate and complex model of the package can be used in the thermal simulation. Consultation on the creation of the complex model is available.

To determine the junction temperature of the device in the application after prototypes are available, the thermal characterization parameter (Ψ_{JT}) can be used to determine the junction temperature with a measurement of the temperature at the top center of the package case using the following equation:

$$T_J = T_T + (\Psi_{JA} \times P_D)$$

where:

T_T = thermocouple temperature on top of package (°C)

$R_{\theta JA}$ = thermal characterization parameter

P_D = power dissipation in package

The thermal characterization parameter is measured per JESD51-2 characteristic published by JEDEC using a 40 gauge type T thermocouple epoxied to the top center of the package case. The thermocouple should be positioned so that the thermocouple junction rests on the package. A small amount of epoxy is placed over the thermocouple junction and over about 1 mm of wire extending from the junction. The thermocouple wire is placed flat against the package case to avoid measurement errors caused by cooling effects of the thermocouple wire.

G.5.1 Thermal References:

Semiconductor Equipment and Materials International
805 East Middlefield Rd
Mountain View, CA 94043
(415) 964-5111

MIL-SPEC and EIA/JESD (JEDEC) characteristics are available from Global Engineering Documents at 800-854-7179 or 303-397-7956.

JEDEC characteristics are available on the WEB at: **http://www.jedec.org**

1. C.E. Triplett and B. Joiner, “An Experimental Characterization of a 272 PBGA Within an Automotive Engine Controller Module,” Proceedings of SemiTherm, San Diego, 1998, pp. 47-54.
2. B. Joiner and V. Adams, “Measurement and Simulation of Junction to Board Thermal Resistance and Its Application in Thermal Modeling,” Proceedings of SemiTherm, San Diego, 1999, pp. 212-220.

G.6 ESD Protection

Table G-3 ESD Protection

Characteristics	Symbol	Value	Units
ESD Target for Human Body Model (HBM)		2000	V
HBM Circuit Description	R1	1500	Ω
	C	100	pF
ESD Target for Machine Model (MM)		200	V
MM Circuit Description	R1	0	Ω
	C	200	pF
Number of Pulses Per Pin Positive Pulses (MM) Negative Pulses (MM) Positive Pulses (HBM) Negative Pulses (HBM)	—	 3 3 1 1	—
Interval of Pulses	—	1	Second

Notes:
1. All ESD testing is in conformity with CDF-AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits.
2. A device will be defined as a failure if after exposure to ESD pulses the device no longer meets the device characteristic requirements. Complete DC parametric and functional testing shall be performed per applicable device characteristic at room temperature followed by hot temperature, unless specified otherwise in the device characteristic.

G.7 DC Electrical Characteristics

Table G-4 DC Electrical Characteristics

(V_{DDL} = 3.3 V ± 0.3 V, V_{DDH} = 5.0 V ± 0.5 V, $T_A = T_L$ to T_H)

Characteristic	Symbol	Min	Max	Unit
3-V only Input High Voltage[1] Except EXTAL and EXTCLK	V_{IH3}	2.0	V_{DDH} +0.3	V
3-V Input High Voltage EXTCLK	V_{IHC}	2.4	V_{DDH} +0.3	V
5-V Input Only High Voltage [2]	V_{IH5}	0.7*V_{DDH}	V_{DDH} +0.3	V
5-V Input High Voltage (QADC PQA, PQB)	V_{IHA5}	0.7*V_{DDA}	V_{DDA} +0.3	V
Muxed 3-V/ 5-V Pins (GPIO Muxed with Addr. (Port A), Data (Port D), and Control (Port C)) 3-V Input High Voltage Addr. (Port A), Data (Port D), Control (Port C) 5-V Input High Voltage (GPIO)	V_{IH3M} V_{IH5M}	2.0 0.7*V_{DDH}	V_{DDH} +0.3 V_{DDH} +0.3	V
3-V Input Low Voltage Except EXTCLK	V_{IL3}	V_{SS} -0.3	0.8	V
3-V Input Low Voltage EXTCLK	V_{IL3C}	V_{SS} -0.3	0.4	V
5-V Input Low Voltage	V_{IL5}	V_{SS} -0.3	0.4*V_{DDH}	V
5-V Input Low Voltage (QADC PQA, PQB)	V_{ILA5}	V_{SSA} -0.3	0.4*V_{DDA}	V
Muxed 3-V/ 5-V Pins (GPIO Muxed with Addr. (Port A), Data (Port D), and Control (Port C)) 3-V Input Low Voltage (Addr. (Port A), Data (Port D), Control (Port C)) 5-V Input Low Voltage (GPIO)	V_{IL3M} V_{IL5M}	V_{SS} -0.3 V_{SS} -0.3	0.8 0.4*V_{DDH}	V V
QADC Analog Input Voltage[3]	V_{INDC}	V_{SSA} -0.3	V_{DDA} +0.3	V
3-V Mode Select Current Pull-up @ 0 V to V_{IL3}, Pull-down @ V_{IH3} to V_{DDL}	$I_{act3\ V}$	20	130	µA
5-V Mode Select Current Pull-up @ 0 to V_{IL5}, Pull-down @ V_{IH5} to V_{DDH}	$I_{act5\ V}$	20	130	µA
3-V Input Leakage Current Pull-up/down Inactive	$I_{inact3\ V}$	—	1.0	µA
5-V Input Leakage Current Pull-up/down Inactive	$I_{inact5\ V}$	—	1.0	µA
QADC64 Input Current, Channel Off [4] PQA PQB	I_{OFF}	-200 -150	200 150	nA

Table G-4 DC Electrical Characteristics (Continued)

(V_{DDL} = 3.3 V ± 0.3 V, V_{DDH} = 5.0 V ± 0.5 V, $T_A = T_L$ to T_H)

Characteristic	Symbol	Min	Max	Unit
3-V Output High Voltage $V_{DD} = V_{DDL}$ (I_{OH} = -2 mA)	V_{OH3}	2.4	—	V
5-V Output High Voltage $V_{DD} = V_{DDH}$ (I_{OH} = -2 mA) All 5-V Only Outputs Except TPU.	V_{OH5}	V_{DDH} -0.7	—	V
5-V Output High Voltage $V_{DD} = V_{DDH}$ (I_{OH} = -5 mA) For TPU Pins Only	V_{OHTP5}	V_{DDH} -0.65	—	V
Muxed 3-V/ 5-V Pins (GPIO Muxed with Addr. (Port A), Data (Port D), and Control (Port C)) 3-V Output High Voltage (I_{OH} = -2 mA) 5-V Output High Voltage (I_{OH} = -2 mA)	V_{OH3M} V_{OH5M}	2.4 V_{DDH} -0.7	—	V
3-V Output Low voltage $V_{DD} = V_{DDL}$ (I_{OL} = 3.2 mA)	V_{OL3}	—	0.5	V
5-V Output Low Voltage $V_{DD} = V_{DDH}$ (I_{OL} = 2 mA) All 5-V Only Outputs Except TPU.	V_{OL5}	—	0.45	V
5-V Output Low Voltage $V_{DD} = V_{DDH}$ For TPU Pins Only IOL = 10 mA IOL = 2 mA	V_{OLTP5}	—	1.0 0.45	V
Muxed 3-V/ 5-V Pins (GPIO Muxed with Addr. (Port A), Data (Port D), and Control (Port C)) 3-V Output Low Voltage (I_{OL} = 3.2 mA) 5-V Output Low Voltage (I_{OL} = 2 mA)	V_{OL3M} V_{OL5M}	—	0.5 0.45	V
Output Low Current CLKOUT @ V_{OL} = 0.5 V	I_{OL}	2.0	—	mA
Output High Current CLKOUT @ V_{OH} = 2.4 V	I_{OH}	2.0	—	mA
CLKOUT Capacitance (@ 40 MHz) COM[1:0] of SCCR = 0b01 COM[1:0] of SCCR = 0b00	C_{clk}	—	30[5] 90	pF
ENGCLK Capacitance@20Mhz EECLK[1:0] of SCCR = 0b01 EECLK[1:0] of SCCR = 0b00	C_{eng}	—	25[5] 50[6]	pF
Capacitance for Input, Output, and Bidirectional V_{in} = 0 V, f = 1 MHz (except QADC)	C_{in}	—	7	pF
Load Capacitance for Bus Pins Only[7] COM[1:0] of SCCR = 0bX1 COM[1:0] of SCCR = 0bX0	C_L	—	25 50	pF
QADC Total Input Capacitance PQA Not Sampling PQB Not Sampling Incremental Capacitance Added During Sampling	C_{IN}	— — —	15 10 5	pF
Hysteresis (Only IRQ, TPU, MIOS, GPIO, QADC [Digital Inputs] and $\overline{PORESET}$, $\overline{HRESET}$, $\overline{SRESET}$)[8]	V_H	0.5	—	V

Table G-4 DC Electrical Characteristics (Continued)

(V_{DDL} = 3.3 V ± 0.3 V, V_{DDH} = 5.0 V ± 0.5 V, T_A = T_L to T_H)

Characteristic	Symbol	Min	Max	Unit
Operating Current (3-V Supplies) @ 33 MHz Expanded V_{DDL} / V_{DDI} KAPWR V_{DDSRAM} V_{DDSYN} (Crystal Frequency: 20 Mhz) V_{DDF}[9] Single Chip V_{DDL} / V_{DDI} KAPWR V_{DDSRAM} V_{DDSYN} (Crystal Frequency: 20 Mhz) V_{DDF}[9]	 I_{DDL} I_{DDKAP} I_{DDSRM} I_{DDSYN} I_{DDF} I_{DDL} I_{DDKAP} I_{DDSRM} I_{DDSYN} I_{DDF}		 206 6.6 1.7 1.7 8.3 206 6.6 1.7 1.7 8.3	mA
Operating Current (5-V Supplies)@ 33 MHz V_{DDH} V_{DDA} V_{PP}[10]	 I_{DDH5} I_{DDA} I_{DDPP}		 16.5 5 30	mA
Operating Current (3-V Supplies)@ 40 MHz[11] Expanded V_{DDL} / V_{DDI} KAPWR V_{DDSRAM} V_{DDSYN} (Crystal Frequency: 20 Mhz) V_{DDF}[9] Single Chip V_{DDL} / V_{DDI} KAPWR V_{DDSRAM} V_{DDSYN} (Crystal Frequency: 20 Mhz) V_{DDF}[9]	 I_{DDL} I_{DDKAP} I_{DDSRM} I_{DDSYN} I_{DDF} I_{DDL} I_{DDKAP} I_{DDSRM} I_{DDSYN} I_{DDF}		 250 8.0 2.0 2.0 10 250 8.0 2.0 2 10	mA
Operating Current (5-V Supplies)@ 40 MHz V_{DDH} V_{DDA}[12] V_{PP}[10]	 I_{DDH5} I_{DDA} I_{DDVPP}		 20 5.0 30	mA
QADC64 Low Power Stop Mode (V_{DDA})	I_{DDA}		10	µA
Low Power Current @ 40 MHz (VDDI) DOZE, Active PLL and Active Clocks SLEEP, Active PLL with Clocks off DEEP SLEEP[13] PLL and Clocks off	 I_{DDDZ} I_{DDSLP} $I_{DDDPSLP}$		 100 10 4	 mA mA mA
V_{DDL}, V_{DDI}, V_{DDF} Operating Voltage	V_{DDL}, V_{DDI}, V_{DDF}	3.0	3.6	V
V_{PP} Flash Operating Voltage	V_{PP}	V_{DDF} -0.35 V	5.50	V

Table G-4 DC Electrical Characteristics (Continued)

(V_{DDL} = 3.3 V ± 0.3 V, V_{DDH} = 5.0 V ± 0.5 V, $T_A = T_L$ to T_H)

Characteristic	Symbol	Min	Max	Unit
V_{PP} Flash Programming Voltage	V_{PP}	4.75	5.25	V
Oscillator, Keep Alive Registers Operating Voltage during normal operation (V_{DDL} powered-up)[13,14]	KAPWR	V_{DDL} -0.2	V_{DDL} +0.2	V
Oscillator, Keep Alive Registers Operating Voltage during powered-down operation	KAPWR	3.0	3.6	V
SRAM Operating Voltage during normal operation (V_{DDL} powered-up)[14]	V_{DDSRAM}	V_{DDL} -0.2	V_{DDL} + 0.2	V
V_{DDH} Operating Voltage	V_{DDH}	4.5	5.5	V
QADC Operating Voltage	V_{DDA}	4.5	5.5	V
Clock Synthesizer Operating Voltage[13]	V_{DDSYN}	V_{DDL} -0.2	V_{DDL} +0.2	V
V_{SS} Differential Voltage	$V_{SS} - V_{SSA}$	-100	100	mV
QADC64 Reference Voltage Low[15]	V_{RL}	V_{SSA}	V_{SSA} +0.1	V
QADC64 Reference Voltage High[16]	V_{RH}	V_{DDA} -0.3	V_{DDA}	V
QADC64 V_{REF} Differential Voltage	$V_{RH} - V_{RL}$	4.5	5.5	V
QADC64 Reference Supply Current, DC QADC64 Reference Supply Current, Transient Measured on V_{RH}	I_{REF} I_{REF}	— —	500 4.0	µA mA
Standby Supply Current KAPWR Only V_{DDSRAM} Only (RAM Standby Current) @ T_J = 90°C V_{DDSRAM} Only (RAM Standby Current) @ T_J = 90°C with Low Voltage Protection Circuitry V_{DDSRAM} Only (RAM Standby Current) @ T_J = 150°C			4 100 150 250	mA µA µA µA
RAM Standby Voltage for Data Retention (Powered-down Mode) Specified V_{DD} Applied ($V_{DD} = V_{SS}$)	V_{DDSRAM}	1.4[17]	3.6	V
DC Injection Current per Pin GPIO, TPU, MIOS, QSM, EPEE and 5 V [18,19]	I_{IC5}	-1.0	1.0	mA
DC Injection Current Per Pin 3 V [18, 19]	I_{IC3}	-1.0	1.0	mA
QADC64 Disruptive Input Current [18, 20]	I_{NA}	-3	3	mA
Power Dissipation -40 MHz 33 MHz	PD		1 0.8	W

NOTES:
1. This spec is for 3-V output and 5-V input friendly pins.
2. This spec is for 5-V output and 5-V input pins.
3. Within this range, no significant injection will be seen. See QADC64 disruptive input current (I_{NA}).
4. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 8 – 12°C, in the ambient temperature range of 50 – 125°C.
5. 45 pF maximum for mask sets prior to K62N

6. 90 pF maximum for mask sets prior to K62N
7. All bus pins support two drive strengths capabilities, 25 pF and 50 pF. Current drive is less at the 25-pF capacitive load. Both modes achieve 40-MHz timing.
8. Only IRQ, TPU, MIOS, GPIO, QADC (when digital inputs) and RESET pins have hysteresis, thus there is no hysteresis characteristic required for all other pins
9. The worst case V_{DDF} occurs during $\overline{HRESET}$ active (booting), other modules will not be running.
10. Maximum occurs during programming and erase. Read I_{PP} is lower.
11. All power consumption characteristics assume 50-pF loads and running a typical application.The power consumption of some modules could go up is they are exercised heavier, but the power consumption of other modules would decrease.
12. Current measured at maximum system clock frequency with QADC active.
13. This parameter is periodically sampled rather than 100% tested.
14. KAPWR and V_{DDSRAM} are powered up prior to any other supply.
15. To obtain full-range results, $V_{SSA} \leq V_{RL} \leq V_{INDC} \leq V_{RH} \leq V_{DDA}$
16. To obtain full-range results, $V_{SSA} \leq V_{RL} \leq V_{INDC} \leq V_{RH} \leq V_{DDA}$
17. The voltage at which the LVSRS bits in the VSRMCR register will be set ranges from 1.5 – 2.4 V.
18. All injection current is transferred to the V_{DDH}. An external load is required to dissipate this current to maintain the power supply within the specified voltage range.
19. Total injection current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.
20. Below disruptive current conditions, the channel being stressed has conversion values of 0x3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.

G.8 Oscillator and PLL Electrical Characteristics

Table G-5 Oscillator and PLL

Characteristic	Symbol	Min	Typical	Max	Unit
Oscillator Startup Time (For Typical Crystal Capacitive Load) 4-MHz Crystal 20-MHz Crystal	 OSCstart4 OSCstart20			 10 10	mS
PLL Lock Time	T_{LOCK}			500	Input Clocks
PLL Operating Range	F_{VCOOUT}	30		80	MHz
Crystal Operating Range, MODCK[1:3]=0b010, or 0b110 MODCK[1:3] = 0b001, 0b011, 0b100, 0b101, 0b111	$F_{CRYSTAL}$	 2 15		 5 25	MHz
PLL Jitter PLL Jitter (averaged over 10 µs) MF < 20	F_{JIT} F_{JIT10}	-1% -0.3		+1% +0.3	—
Limp Mode Clock Frequency	F_{LIMP}	3[1]	7	12[1]	MHz
Oscillator Bias Current (XTAL) 4 MHz 20 MHz	I_{BIAS}	 — —		 -0.4[1] -0.8[1]	mA
Oscillator Drive (XTAL)	Iosc	8[1]		25[1]	mA
Oscillator Bias Resistor	R_{OSC}	0.72[1]	1.1	1.93[1]	MΩ

NOTES:
1. Values to be evaluated upon further characterization.

G.9 Power Up/Down Sequencing

See **SECTION 8 CLOCKS AND POWER CONTROL**.

G.10 FLASH Electrical Characteristics

NOTE

See **APPENDIX H FLASH ELECTRICAL CHARACTERISTICS FOR ALL J76N MASK SETS AND 0K02A AND 1K02A ONLY** for flash electrical characteristics for all J76N, 0K02A, and 1K02A mask sets. Contact Motorola for flash electrical characteristics for all J12F mask sets.

Table G-6 Program and Erase Characteristics

(V_{DDF} = 3.3 V ± 0.3 V, V_{PP}= 4.75 V to 5.25 V, $T_A = T_L$ to T_H)

Symbol	Meaning	Value			Units
		Minimum	Typical	Maximum	
E_{PULSE}	Number of Erase Pulses	8	8	27	
T_{ERASE}	Erase Pulse Time	98	100	102	mS
$P_{PULSE(4.75\ Vpp)}$	Number of Program Pulses @V_{PP} = 4.75	—	45000	48000[1,2]	Pulses
$P_{PULSE(5.0\ Vpp)}$	Number of Program Pulses @V_{PP} = 5.00	—	800	7000	Pulses
$P_{PULSE(5.25\ Vpp)}$	Number of Program Pulses @V_{PP} = 5.25	—	250	2000[3]	Pulses
T_{PROG}	Program Pulse Time	48	50	256.5	µS
$C_{PULSE(4.75\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 4.75	47	87	700	Pulses
$C_{PULSE(5.0\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 5.00	11	17	57	Pulses
$C_{PULSE(5.25\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 5.25	8	10	37	Pulses
T_{CLEAR}	CENSOR Clear Pulse Time	98	100	102	mS
$S_{PULSE(4.75\ Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 4.75	47	87	700	Pulses
$S_{PULSE(5.0Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 5.00	11	17	57	Pulses
$S_{PULSE(5.25\ Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 5.25	8	10	37	Pulses
T_{SET}	CENSOR Set Pulse Time	98	100	102	mS

NOTES:
1. The worst case programming time occurs at V_{PP} = 4.75 V and T_A = -40 °C.
2. This value is based on initial device characterization and may not be tested in production.
3. The best case (fastest) programming time of < 50 pulses is at V_{PP} = 5.25 V and T_A = 125°C.

Table G-7 CMF AC and DC Power Supply Characteristics

Symbol	Meaning	Min. Value	Max Value	Unit
V_{DDF}	Operating Voltage Read, Program or Erase	3.0	3.6	V
I_{DDF}	Operating Current at 40.0 MHz, V_{DDF} = 3.3 V for a 256-Kbyte Module Read, Program or Erase Operation Disabled	 — —	 10 5	mA
V_{PP}	External Program or Erase Voltage Read Program or Erase	 V_{DDF} -0.35 4.75	 5.5 5.25	V
I_{DDPP}	External Program and Erase Current[1] Read, V_{PP} = 5 V Program, V_{PP} = 5.25 V Erase, V_{PP} = 5.25 V		 <100 30[1] 30[1]	 µA mA mA

NOTES:
1. Average current is less than 30 mA when programming both modules simultaneously.

G.10.1 Flash Module Life

Table G-8 Flash Module Life

Symbol	Meaning	Value
P/E Cycles[1]	Maximum Number of Program/Erase Cycles[2] to Guarantee Data Retention Array Blocks	100[3,4]
P/E Cycles[1]	Maximum Number of Program/Erase Cycles[2] to Guarantee Data Retention Censor Bits	10^4
Retention	Data Retention at Average Operating Temperature of 85 °C	Minimum 10 years

NOTES:
1. Target failure rate at specified number of program/erase cycles of 2 ppm pending characterization of production silicon.
2. A program/erase cycle is defined as switching the bits from 1 → 0 → 1.
3. Reprogramming of a CMF array block prior to erase is not required.
4. Number of program/erase cycles to be adjusted pending characterization of production silicon.

G.10.2 Programming and Erase Algorithm

Table G-9 CMF Programming Algorithm (v6 and Later)

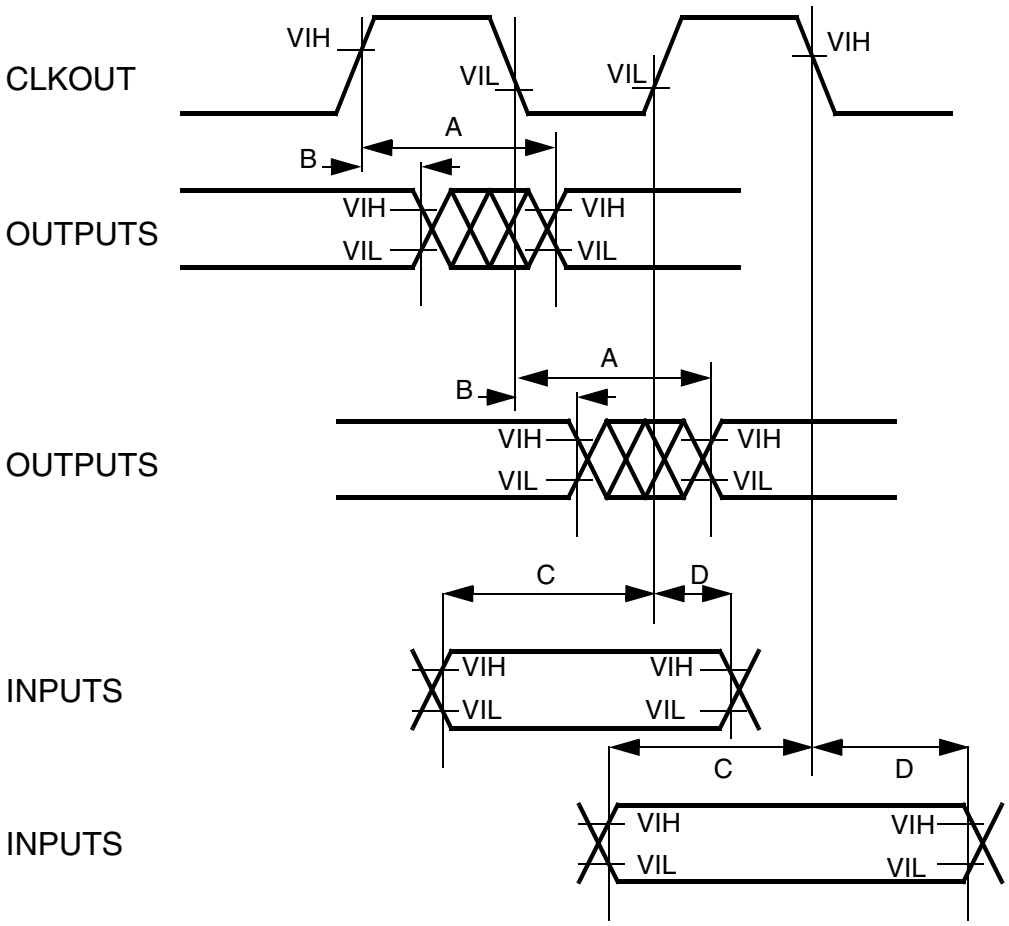
No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
4	256 µs	1	100	1	Mode 4NL	Negative gate ramp (low range)
4	256 µs	1	101	1	Mode 5NL	
4	256 µs	1	110	1	Mode 6NL	
4	256 µs	1	111	1	Mode 7NL	
20	50 µs	0	100	1	Mode 4NL	Negative gate ramp (high range)
20	50 µs	0	101	1	Mode 5NL	
20	50 µs	0	110	1	Mode 6NL	
max. 48,000	50 µs	0	111	1	Mode 7NL	

Table G-10 CMF Erase Algorithm (v6)

No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
1	100 ms[1]	1	100	1	Mode 4NL	Negative gate ramp (low range)
1	100 ms[1]	1	101	1	Mode 5NL	
1	100 ms[1]	1	110	1	Mode 6NL	
1	100 ms[1]	1	111	1	Mode 7NL	
1	100 ms[1]	0	100	1	Mode 4NL	Negative gate ramp (high range)
1	100 ms[1]	0	101	1	Mode 5NL	
1	100 ms[1]	0	110	1	Mode 6NL	
20	100 ms[2]	0	111	1	Mode 7NL	

NOTES:
1. No margin read after pulse.
2. Do margin read after each pulse.

G.11 Generic Timing



A. Maximum Output Delay Characteristic

B. Minimum Output Hold Time

C. Minimum input Setup Time Characteristic

D. Minimum input Hold Time Characteristic

Figure G-1 CLKOUT Timing

Table G-11 Bus Operation Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
1	CLKOUT Period	TC	30.30		25		ns
1a	ENGCLK[2]			20		20	MHz
2	Clock Pulse Width Low		15 – 2%		12.5 – 2%		ns
3	Clock Pulse Width High		15 – 2%		12.5 – 2%		ns
4	CLKOUT Rise Time			5		5	ns
4a	ENGCLK Rise Time			20		20	ns
5	CLKOUT Fall Time			5		5	ns
5a	ENGCLK Fall Time			20		20	ns
6	Circuit Parameter	TCC	8		7.75		ns
7	CLKOUT to Signal Invalid (Hold Time) A[0:31] RD/$\overline{\text{WR}}$ BURST D[0:31]	0.2TC – 1.0	5		4		ns
7a	CLKOUT to Signal Invalid: (Hold Time) TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] $\overline{\text{BDIP}}$	0.2TC – 1.0	5		4		ns
7b	CLKOUT to Signal Invalid (Hold Time)[3] $\overline{\text{BR}}$ $\overline{\text{BG}}$ FRZ VFLS[0:1] VF[0:2] IWP[0:2] LWP[0:1] $\overline{\text{PTR}}$ $\overline{\text{RETRY}}$ $\overline{\text{STS}}$[4]	0.2TC – 1.0	5		4		ns
7c	Slave mode CLKOUT to Signal Invalid D[0:31]	0.25TC + TCC	5		4		ns
8	CLKOUT to Signal Valid A[0:31] RD/$\overline{\text{WR}}$ BURST	0.25TC + TCC	7.5	14	6.25	14	ns

Table G-11 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
8a	CLKOUT to Signal Valid TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] $\overline{\text{BDIP}}$	0.25TC + TCC	7.5	15	6.25	15	ns
8b	CLKOUT to Signal Valid[3] $\overline{\text{BR}}$ $\overline{\text{BG}}$ VFLS[0:1] VF[0:2] IWP[0:2] FRZ LWP[0:1] $\overline{\text{PTR}}$ $\overline{\text{RETRY}}$ $\overline{\text{MTS}}$	0.25TC + TCC	7.5	15	6.25	14	ns
8c	Slave Mode CLKOUT to Signal Valid D[0:31]	0.25TC + TCC + 4	7.5	15	6.25	14	ns
8d	CLKOUT to Signal Valid D[0:31]	0.25TC + TCC	7.5	14	6.25	14	ns
9	CLKOUT to Signal invalid A[0:31] RD/$\overline{\text{WR}}$ $\overline{\text{BURST}}$ D[0:31] TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] $\overline{\text{PTR}}$ $\overline{\text{RETRY}}$	0.25TC + TCC	7.5	15	6.25	13	ns
10	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ Assertion	0.25TC + TCC	7.5	15	6.25	14	ns
10a	CLKOUT to $\overline{\text{TA}}$, $\overline{\text{BI}}$ Assertion (When Driven by the Memory Controller)	—		10		10	ns
10b	CLKOUT to $\overline{\text{RETRY}}$ Assertion (When Driven by the Memory Controller)	—		10		10	ns
11	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ Negation	0.25TC + TCC	7.5	15	6.25	14	ns
11a	CLKOUT to $\overline{\text{TA}}$, $\overline{\text{BI}}$ Negation (When Driven by the Memory Controller)	—		11		11	ns
11b	CLKOUT to $\overline{\text{RETRY}}$ Negation (When Driven by the Memory Controller)	—		11		11	ns
12	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ Signal invalid	0.25TC + 14	7.5	21	6.25	20	ns

Table G-11 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
12a	CLKOUT to $\overline{TA}$, $\overline{BI}$ Signal invalid (When Driven by the Memory Controller)			15		15	ns
13	CLKOUT to $\overline{TEA}$ Assertion			11		11	ns
14	CLKOUT to $\overline{TEA}$ Signal invalid			15		15	ns
15	Input Valid to CLKOUT (Setup Time) $\overline{TA}$ $\overline{TEA}$ $\overline{BI}$[3]		13		12		ns
15a	Input Valid to CLKOUT (Setup Time) $\overline{KR}$ $\overline{CR}$ $\overline{RETRY}$		11		10		ns
15b	Input Valid to CLKOUT (Setup Time) $\overline{BB}$ $\overline{BG}$ $\overline{BR}$[5]		10		8		ns
16	CLKOUT to Signal Invalid (Hold Time) $\overline{TA}$ $\overline{TEA}$ $\overline{BI}$ $\overline{BB}$ $\overline{BG}$ $\overline{BR}$ [4]		2		2		ns
16a	CLKOUT to Signal Invalid (Hold Time) $\overline{RETRY}$ $\overline{KR}$ $\overline{CR}$		2		2		ns
17	Signal Valid to CLKOUT Rising Edge (Setup Time) D[0:31][6]		7		6		ns
18	CLKOUT Rising Edge to Signal Invalid (Hold Time) D[0:31][6]		1		1		ns
19	CLKOUT Rising Edge to $\overline{CS}$ asserted -GPCM- ACS = 00	0.25TC + TCC + 1	7.5	16	6.25	14	ns
19a	CLKOUT Falling Edge to $\overline{CS}$ Asserted -GPCM- ACS = 10, TRLX = 0 or 1	TCC + 1		9		8	ns
19b	CLKOUT Falling Edge to $\overline{CS}$ Asserted -GPCM- ACS = 11, TRLX = 0 or 1	0.25TC + TCC + 1	7.5	16	6.25	14	ns

Table G-11 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
19c	CLKOUT Falling Edge to $\overline{CS}$ Asserted -GPCM- ACS = 11, TRLX = 0, EBDF = 1	0.25TC + TCC + 4	7.5	20	6.25	17	ns
20	CLKOUT Rising Edge to $\overline{CS}$ Negated -GPCM- Read Access or Write Access When CSNT = 0 or Write Access When CSNT = 1 and ACS = 00	TCC + 1	1	9	1	8	ns
21	A[0:31] to $\overline{CS}$ Asserted -GPCM- ACS = 10, TRLX = 0		2		1.5		ns
21a	A[0:31] to $\overline{CS}$ Asserted -GPCM- ACS = 11, TRLX = 0		8		7		ns
22	CLKOUT Rising Edge to $\overline{OE}$, $\overline{WE}$[0:3]/ $\overline{BE}$[0:3] Asserted			11		9	ns
23	CLKOUT Rising Edge to $\overline{OE}$ Negated		2	11	0	8	ns
24	A[0:31] to $\overline{CS}$ Asserted -GPCM- ACS = 10, TRLX = 1		30		25		ns
24a	A[0:31] to $\overline{CS}$ Asserted -GPCM- ACS = 11, TRLX = 1		38		32		ns
25	CLKOUT Rising Edge to $\overline{WE}$[0:3]/ $\overline{BE}$[0:3] Negated -GPCM-Write Access CSNT = ‘0‘			9		8	ns
25a	CLKOUT Falling Edge to $\overline{WE}$[0:3]/ $\overline{BE}$[0:3] Negated -GPCM-Write Access TRLX = ‘0’ or ‘1’, CSNT = ‘1, EBDF = 0’.	0.25TC + TCC + 1	7.5	6.25	5	14	ns
25b	CLKOUT Falling Edge to $\overline{CS}$ Negated -GPCM-Write Access TRLX = ‘0’ or ‘1’, CSNT = ‘1’, ACS = ‘10’ or ACS=’11’, EBDF = 0	0.25TC + TCC + 1	7.5	16	5	14	ns
25c	CLKOUT Falling Edge to $\overline{WE}$[0:3]/ $\overline{BE}$[0:3] Negated -GPCM-Write Access TRLX = ‘0’, CSNT = ‘1, EBDF = 1’.	0.25TC +TCC + 4	7.5	20	6.25	17	ns
25d	CLKOUT Falling Edge to $\overline{CS}$ Negated -GPCM-Write Access TRLX = ‘0’, CSNT = ‘1’, ACS = ‘10’ or ACS=’11’, EBDF = 1	0.25TC +TCC + 4	7.5	20	6.25	17	ns
26	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to D[0:31] Signal invalid -GPCM- Write Access, CSNT = ‘0’		0		0		ns
26a	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = ‘0’, CSNT = ‘1’, EBDF = 0		5		4		ns

Table G-11 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
26b	$\overline{CS}$ Negated to D[0:31], Signal invalid -GPCM- Write Access, ACS = '00', TRLX = '0' & CSNT = '0'		0		0		ns
26c	$\overline{CS}$ Negated to D[0:31], Signal invalid -GPCM- Write Access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0		5		4		ns
26d	$\overline{WE}$[0:3] Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', EBDF = 0		38		29		ns
26e	$\overline{CS}$ Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0		38		29		ns
26f	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to D[0:31] Signal Invalid -GPCM- Write Access, TRLX = '0', CSNT = '1', EBDF = 1		12		5		ns
26g	$\overline{CS}$ Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1		12		5		ns
26h	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', EBDF = 1		30		24		ns
26i	$\overline{CS}$ Negated to D[0:31] Signal invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1		30		24		ns
27	$\overline{CS}$, $\overline{WE}$[0:3] Negated to A[0:31] Invalid -GPCM- Write Access[7]		0		0		ns
27a	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '0', CSNT = '1'. $\overline{CS}$ Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '0', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 0		18		4		ns

Table G-11 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression[1]	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
27b	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '1', CSNT = '1'. $\overline{CS}$ Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', ACS = 10, ACS = ='11', EBDF = 0		38		29		ns
27c	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '0', CSNT = '1'. $\overline{CS}$ Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '0', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 1		5		4		ns
27d	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '1', CSNT = '1'. CS Negated to A[0:31] Invalid -GPCM- Write Access, TRLX = '1', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 1		30		24		ns
28	A[0:31], TSIZ[0:1], RD/$\overline{WR}$, $\overline{BURST}$, $\overline{BDIP}$ Valid to CLKOUT Rising Edge. (Slave Mode Setup Time)		11		9		ns
28a	Slave Mode D[0:31] Valid to CLKOUT Rising Edge				7		ns
29	$\overline{TS}$ Valid to CLKOUT Rising Edge (Setup Time)		9		7		ns
30	CLKOUT Rising Edge to $\overline{TS}$ Valid (Hold Time).		2		2		ns

NOTES:

1. Expressions are approximate equations only.
2. This is the maximum frequency at which ENGCLK will meet output drive and rise/fall time specifications.
3. The timing for $\overline{BR}$ ouput is relevant when the MPC555 / MPC556 is selected to work with external bus arbiter. The timing for $\overline{BG}$ output is relevant when the MPC555 / MPC556 is selected to work with internal bus arbiter.
4. The setup times required for $\overline{TA}$, $\overline{TEA}$, and $\overline{BI}$ are relevant only when they are supplied by the external device (and not the memory controller).
5. The timing required for $\overline{BR}$ input is relevant when the MPC500 family microcontroller is selected to work with internal bus arbiter. The timing for $\overline{BG}$ is relevant when the MPC500 is selected to work with external bus arbiter.
6. The D[0:31] input timings 17 and 18 refer to the rising edge of the CLKOUT in which the $\overline{TA}$ input signal is asserted.
7. The timing 27 refers to $\overline{CS}$ when ACS = '00' and to $\overline{WE}$[0:3] when CSNT = '0'.

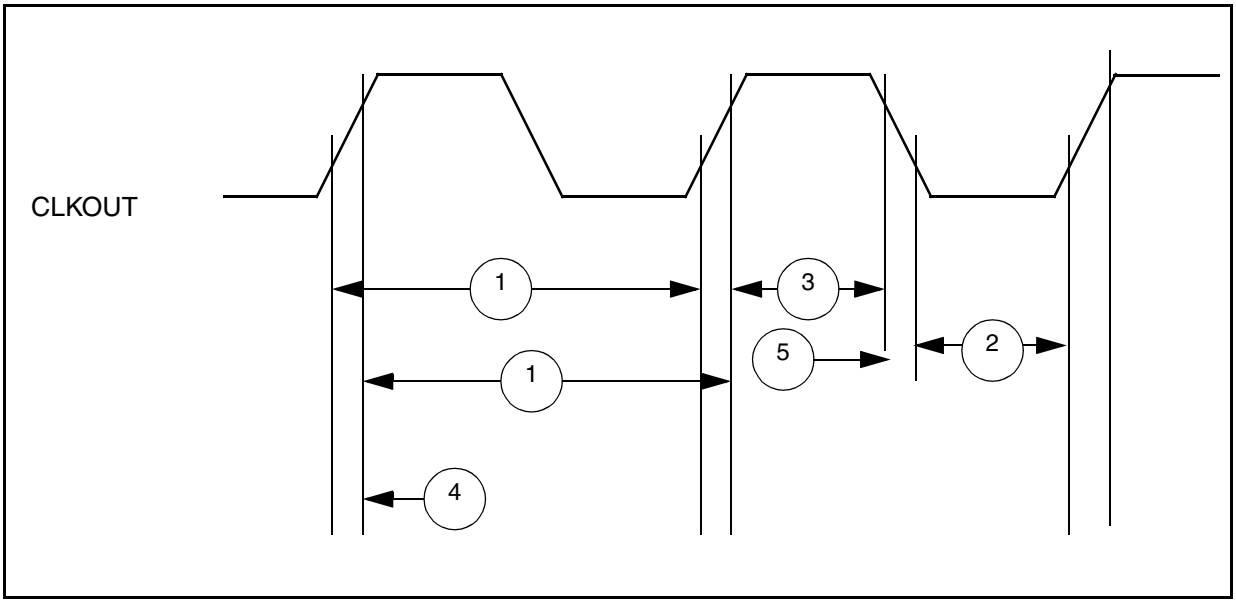


Figure G-2 External Clock Timing

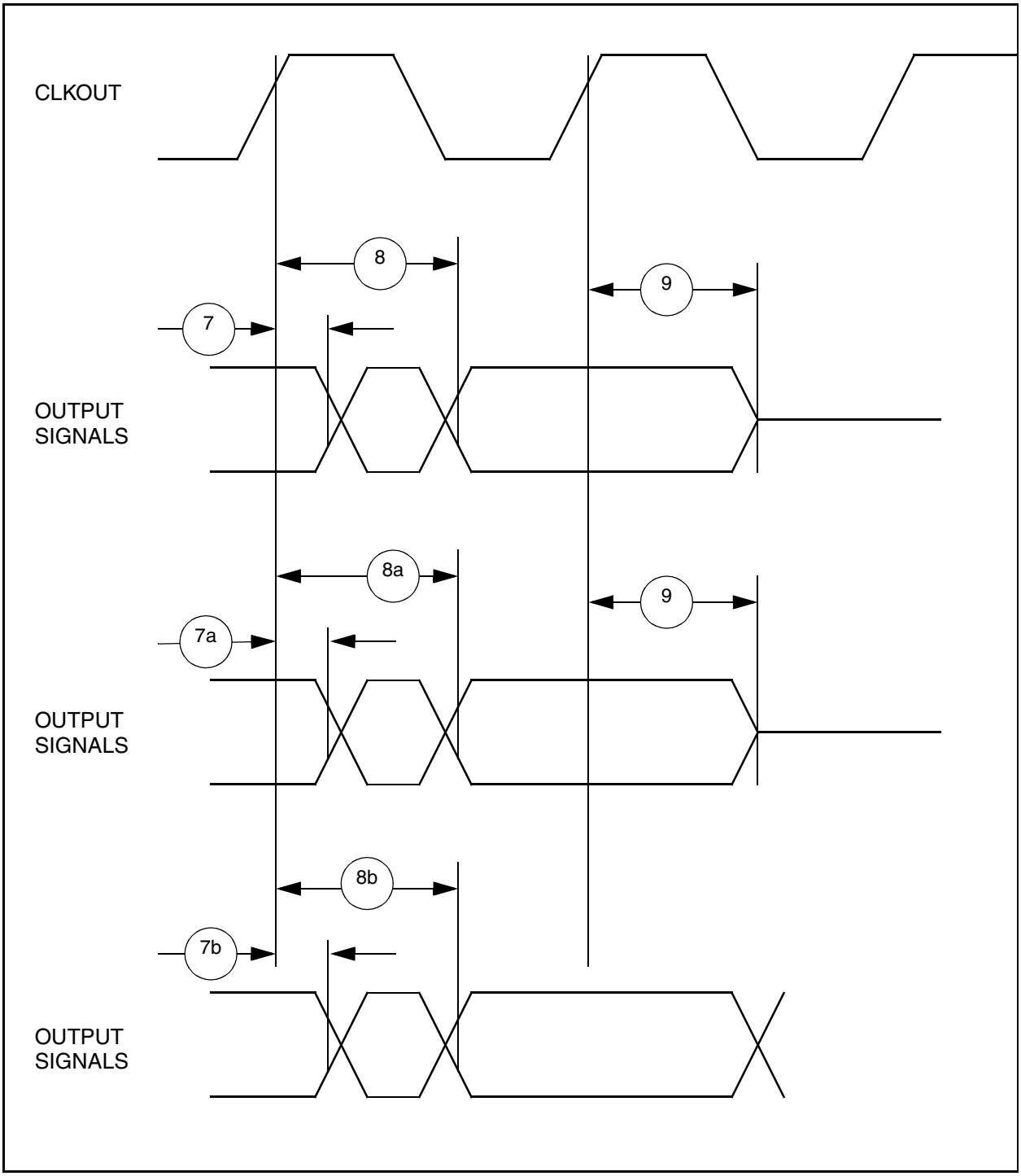


Figure G-3 Synchronous Output Signals Timing

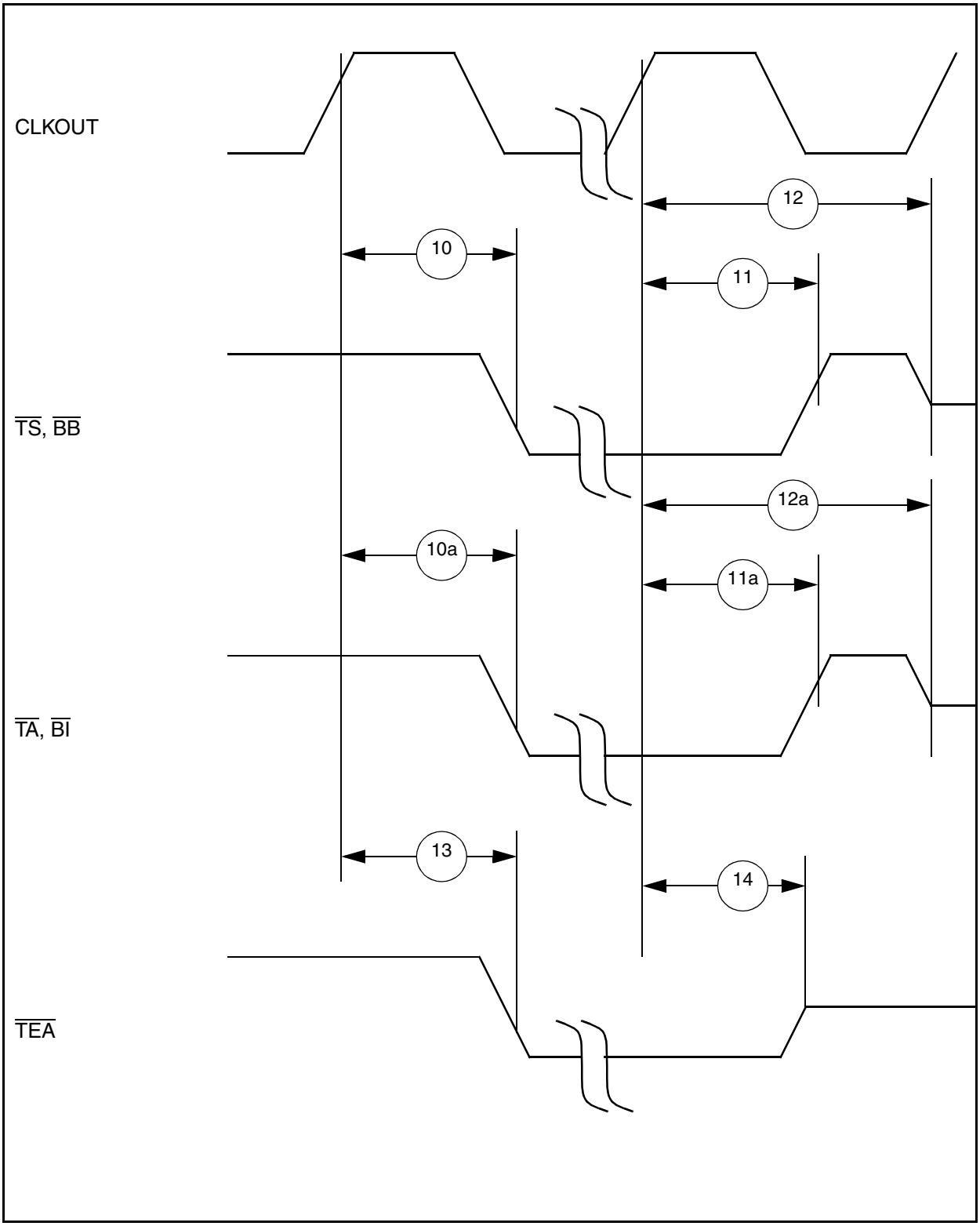


Figure G-4 Synchronous Active Pull-Up and Open Drain Outputs Signals Timing

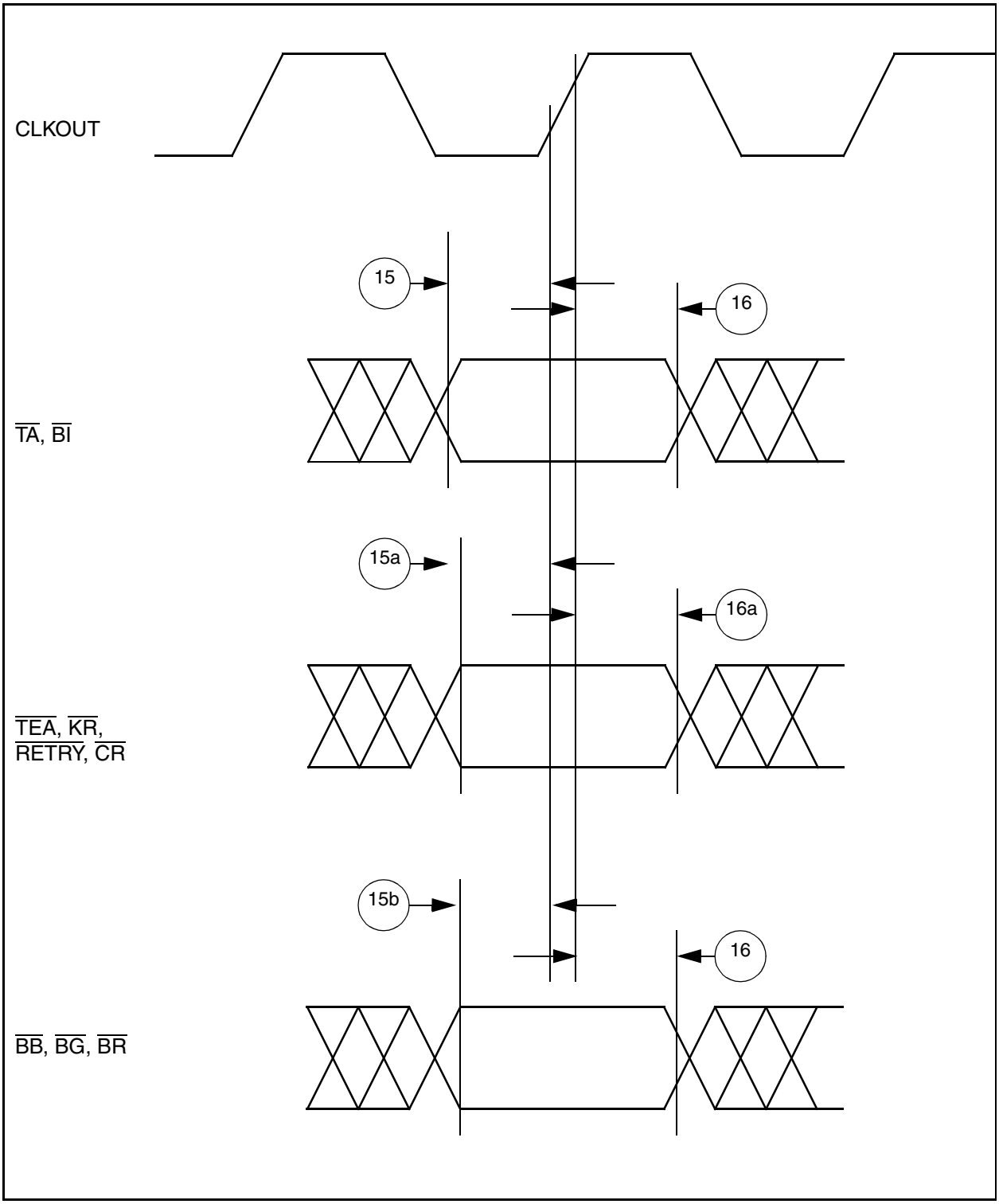


Figure G-5 Synchronous Input Signals Timing

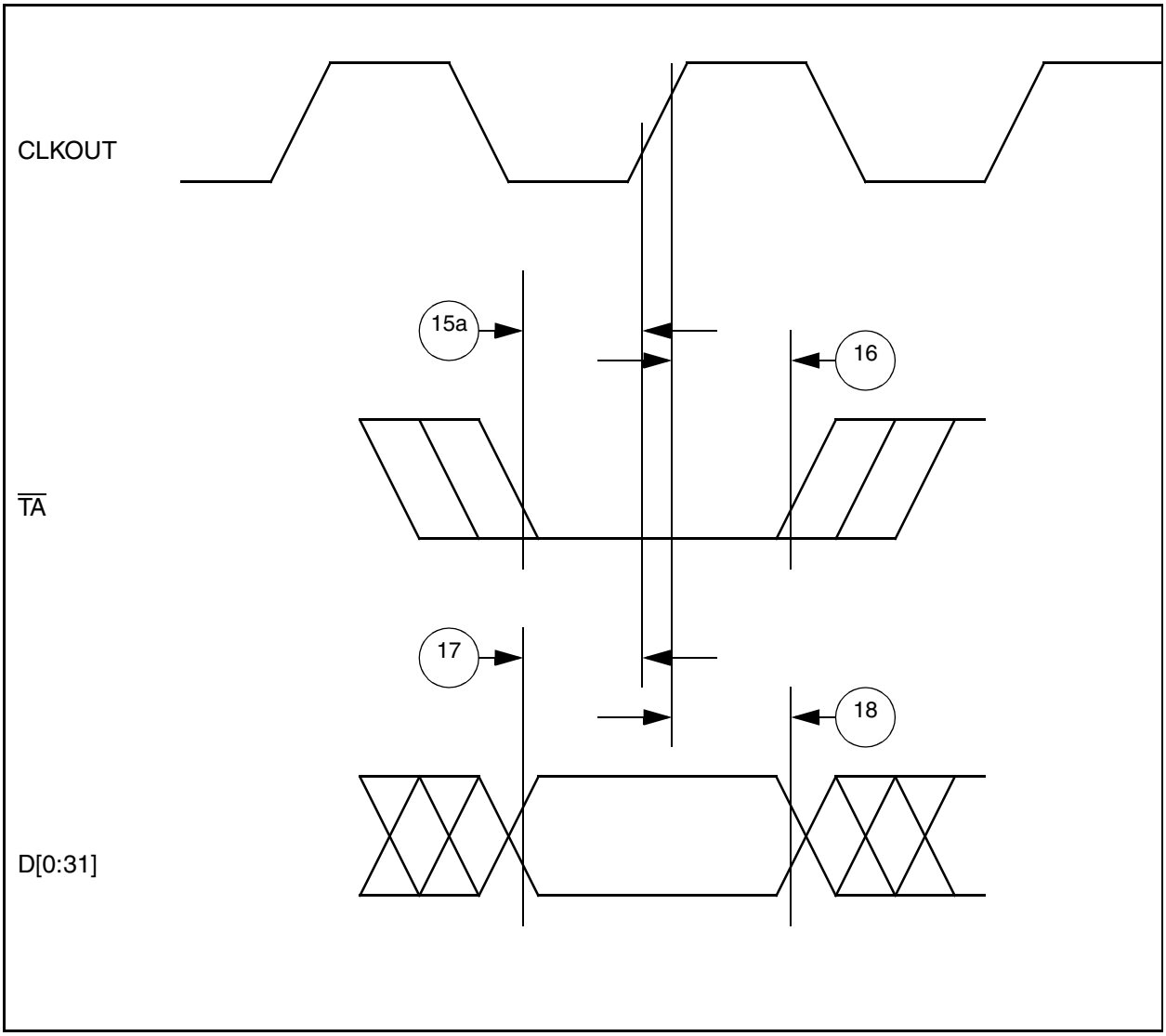


Figure G-6 Input Data Timing in Normal Case

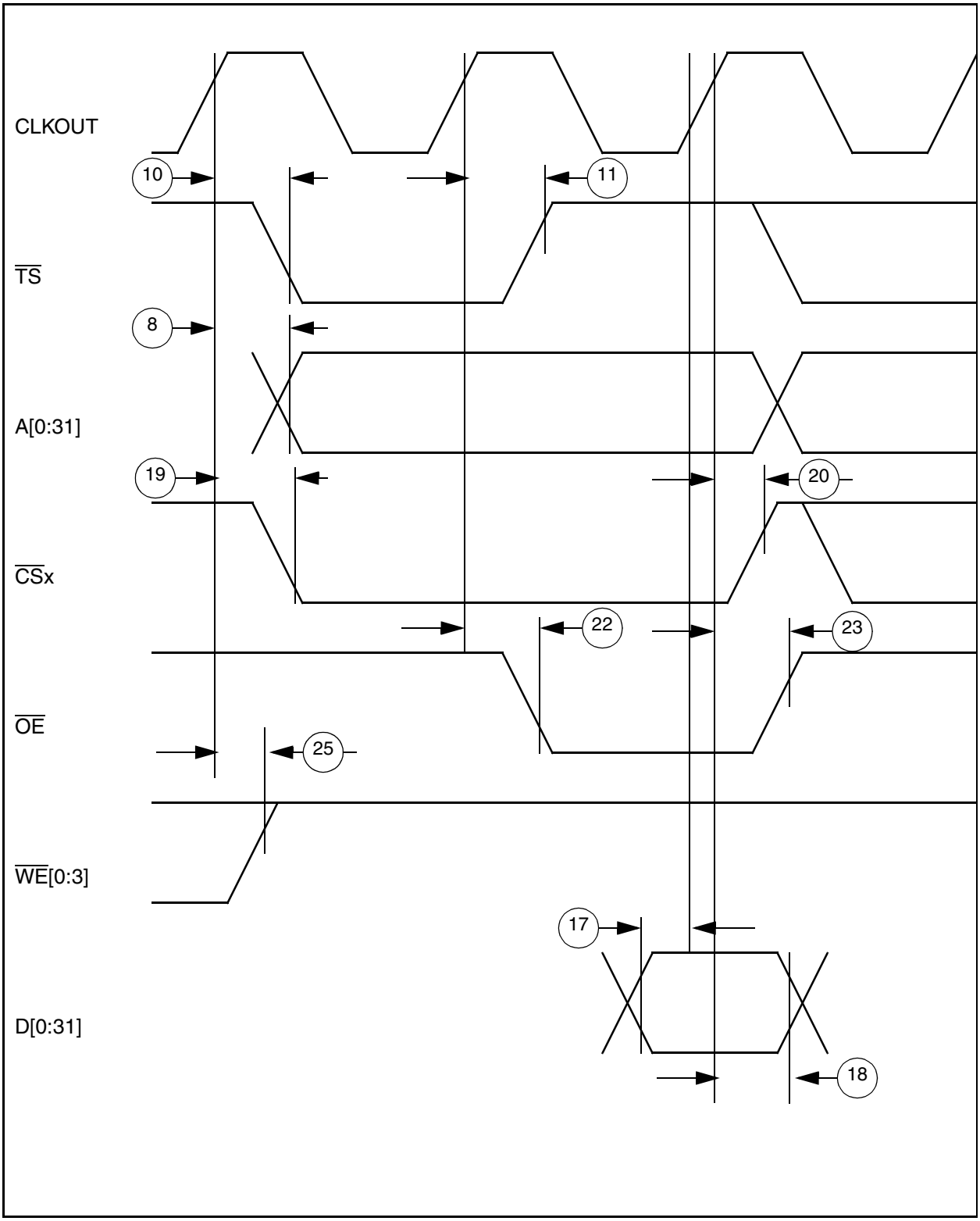


Figure G-7 External Bus Read Timing (GPCM Controlled — ACS = ‘00’)

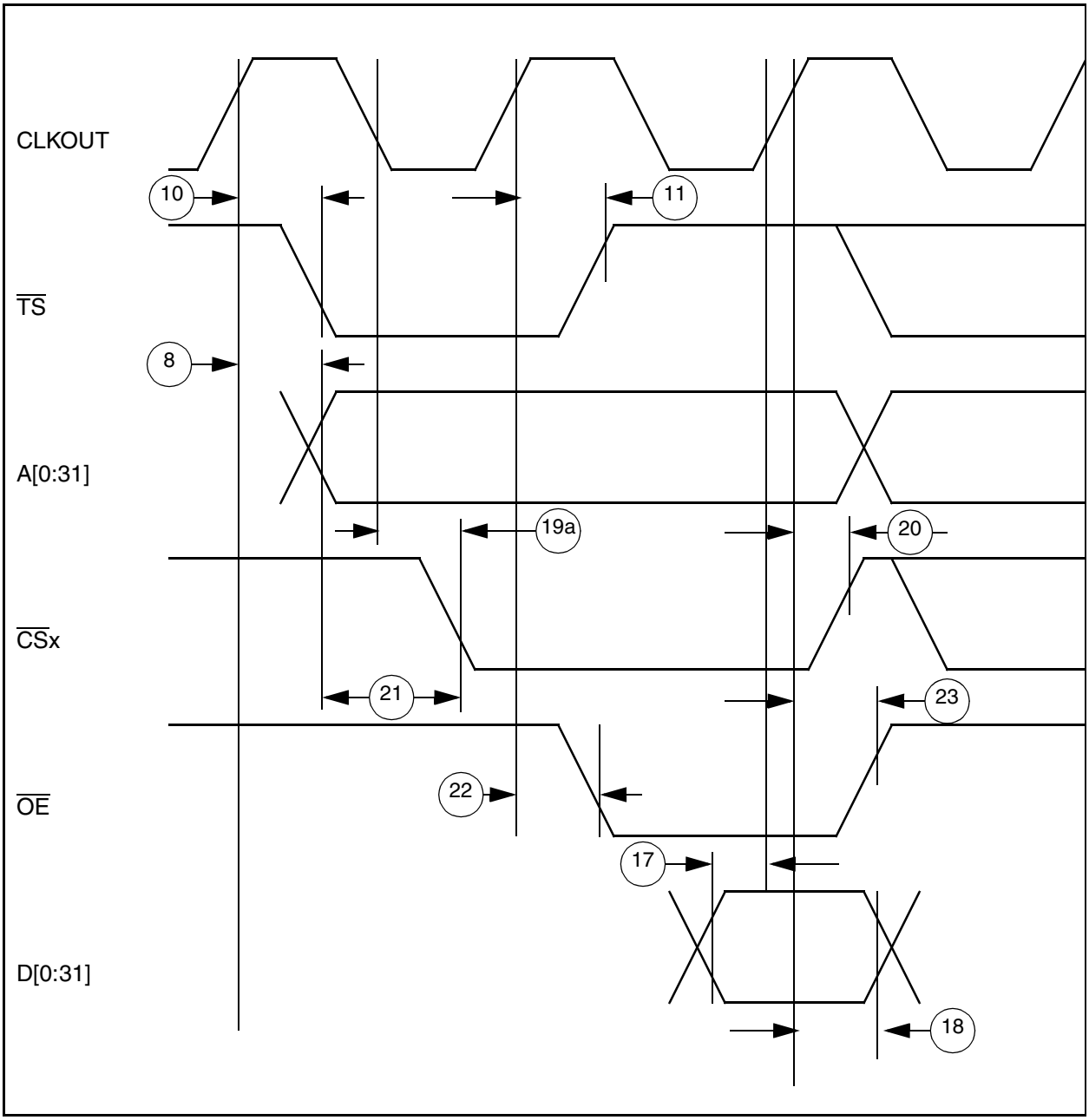


Figure G-8 External Bus Read Timing (GPCM Controlled — TRLX = ‘0’ ACS = ‘10’)

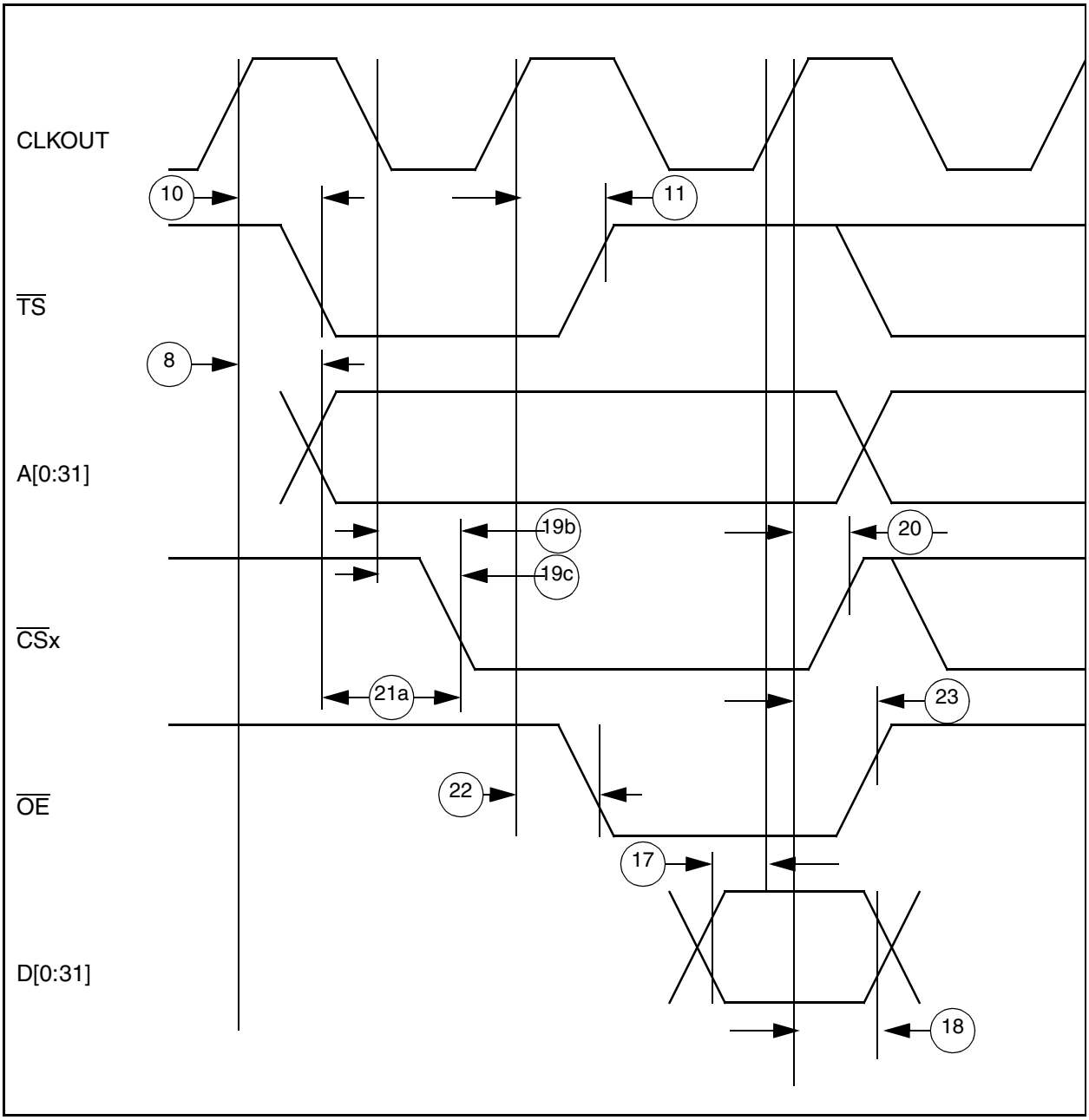


Figure G-9 External Bus Read Timing (GPCM Controlled — TRLX = ‘0’ ACS = ‘11’)

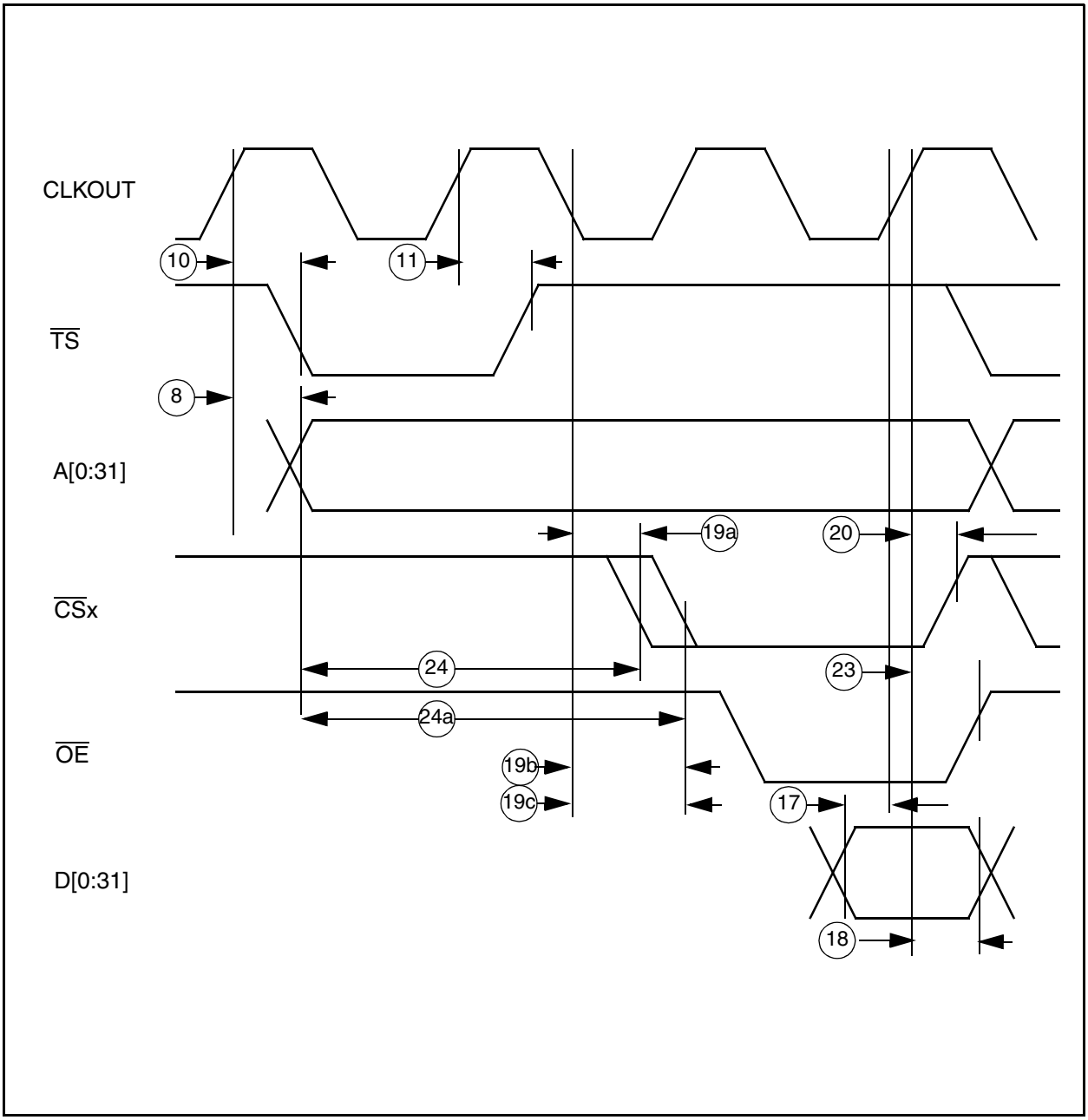


Figure G-10 External Bus Read Timing
(GPCM Controlled — TRLX = ‘1’, ACS = ‘10’, ACS = ‘11’)

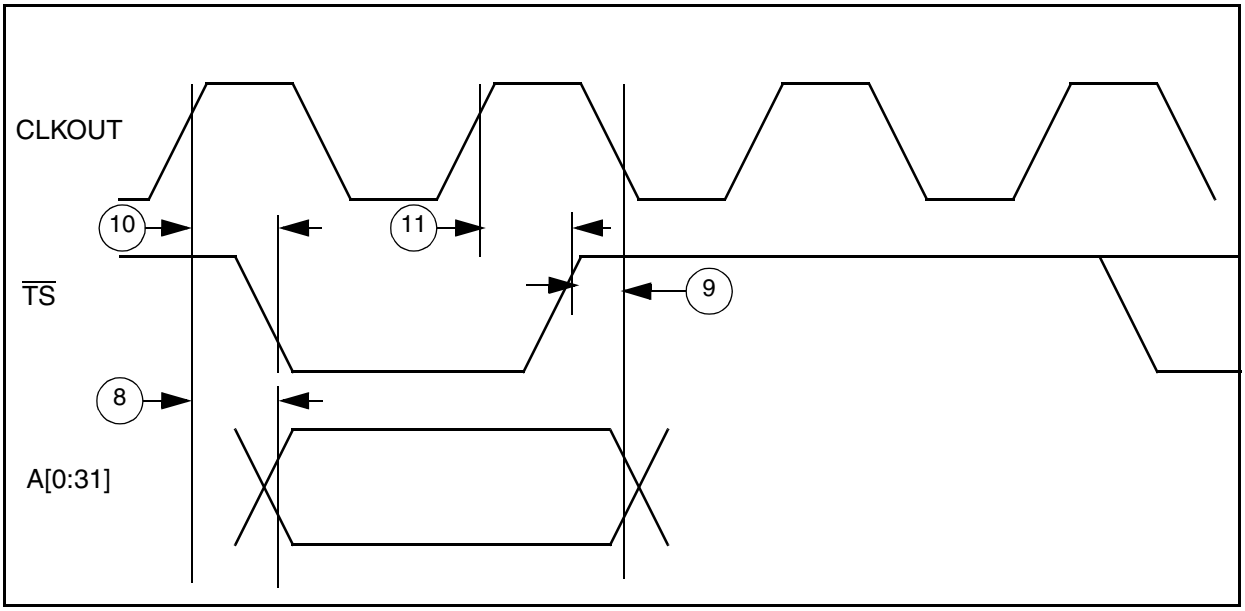


Figure G-11 Address Show Cycle Bus Timing

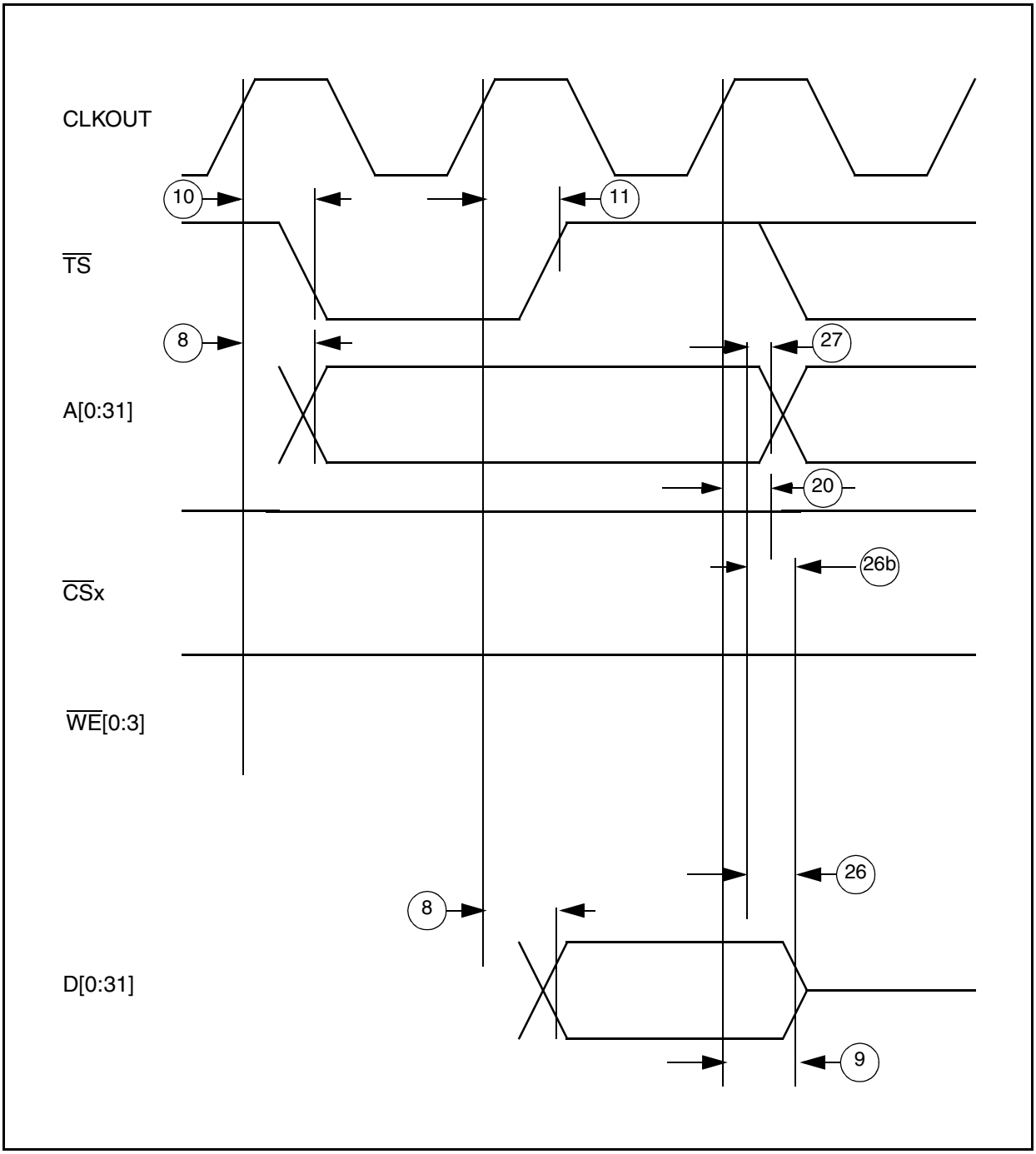


Figure G-12 Address and Data Show Cycle Bus Timing

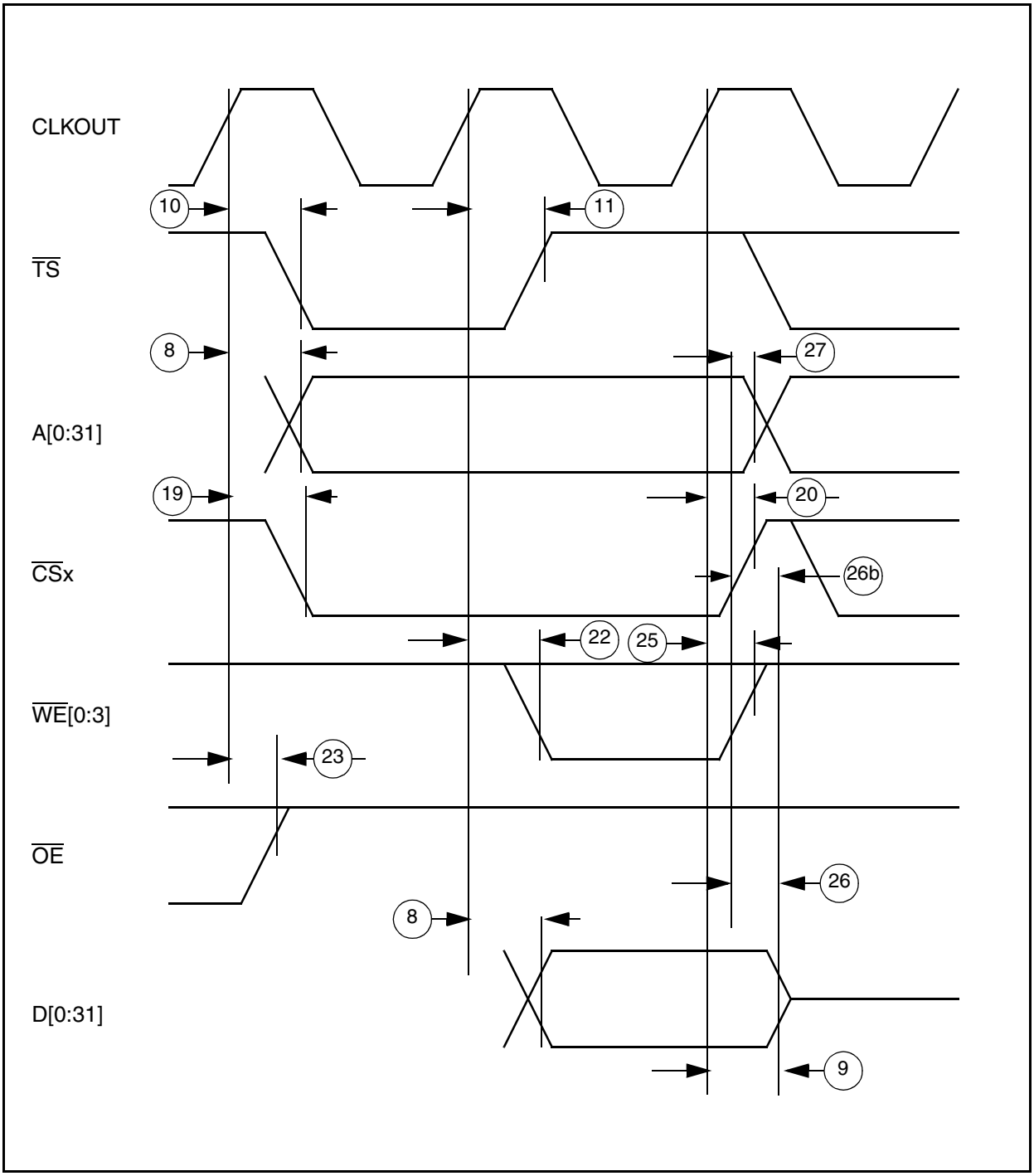


Figure G-13 External Bus Write Timing (GPCM Controlled — TRLX = ‘0’, CSNT = ‘0’)

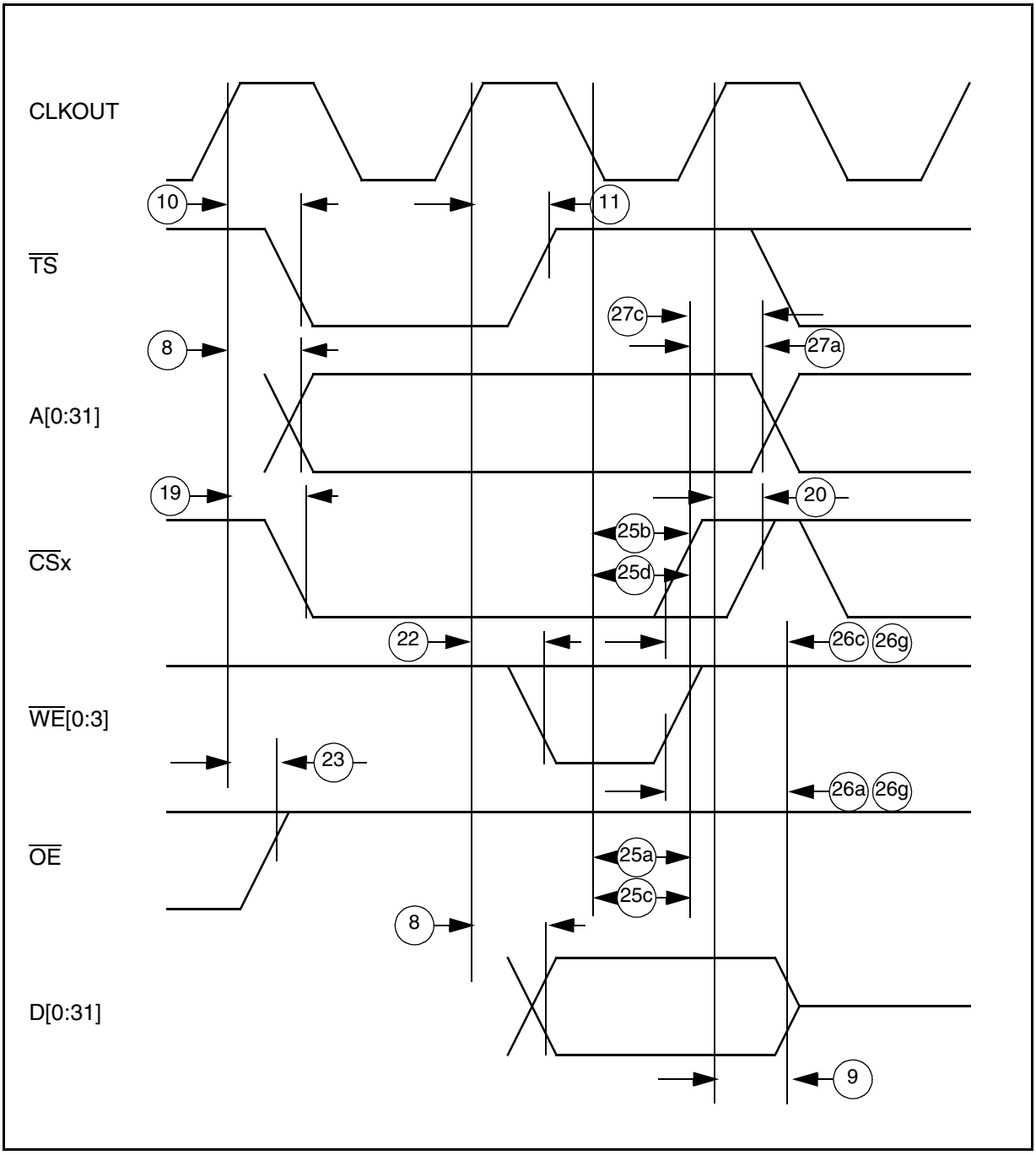


Figure G-14 External Bus Write Timing (GPCM Controlled — TRLX = ‘0’, CSNT = ‘1’)

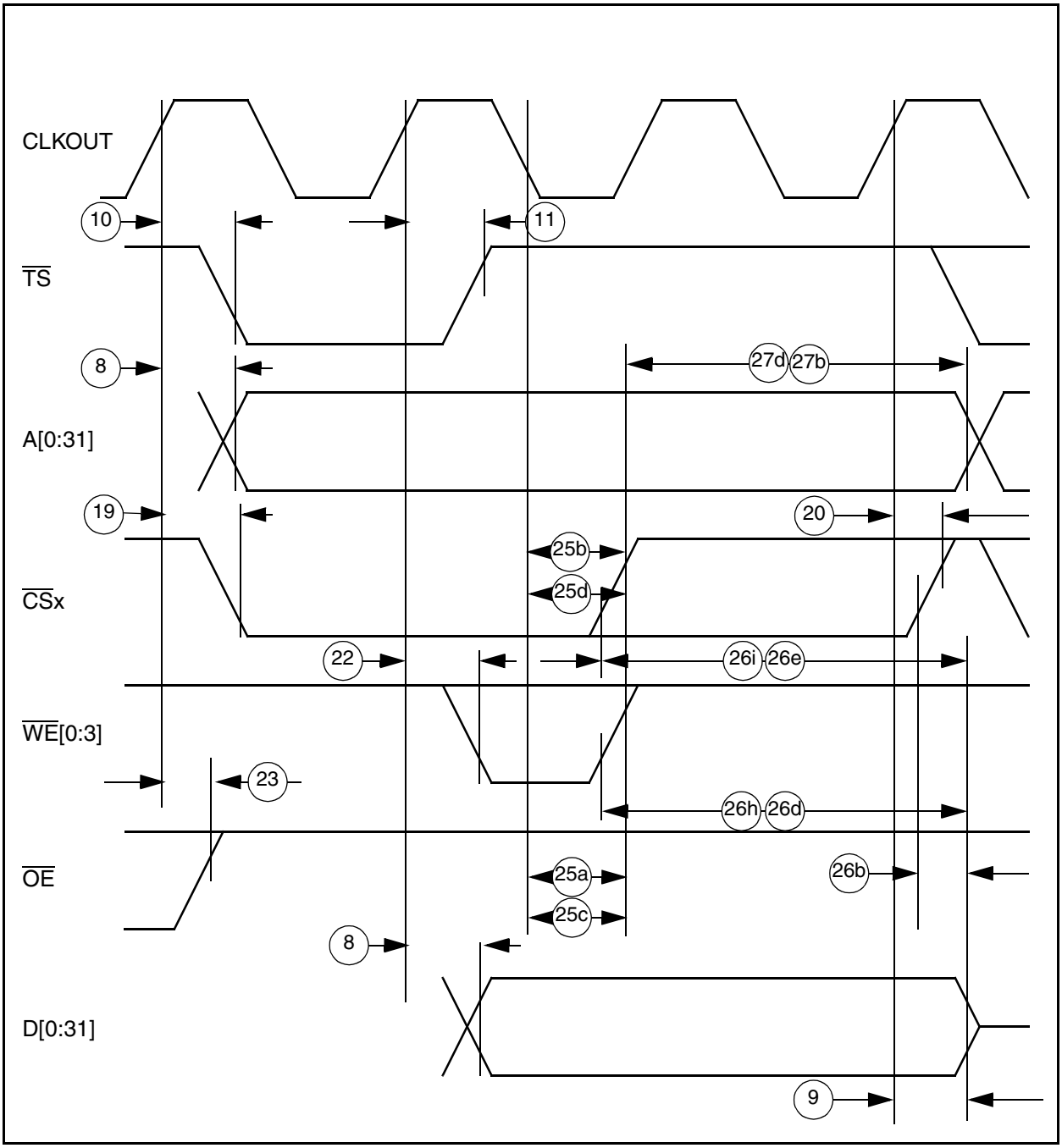


Figure G-15 External Bus Write Timing (GPCM Controlled — TRLX = ‘1’, CSNT = ‘1’)

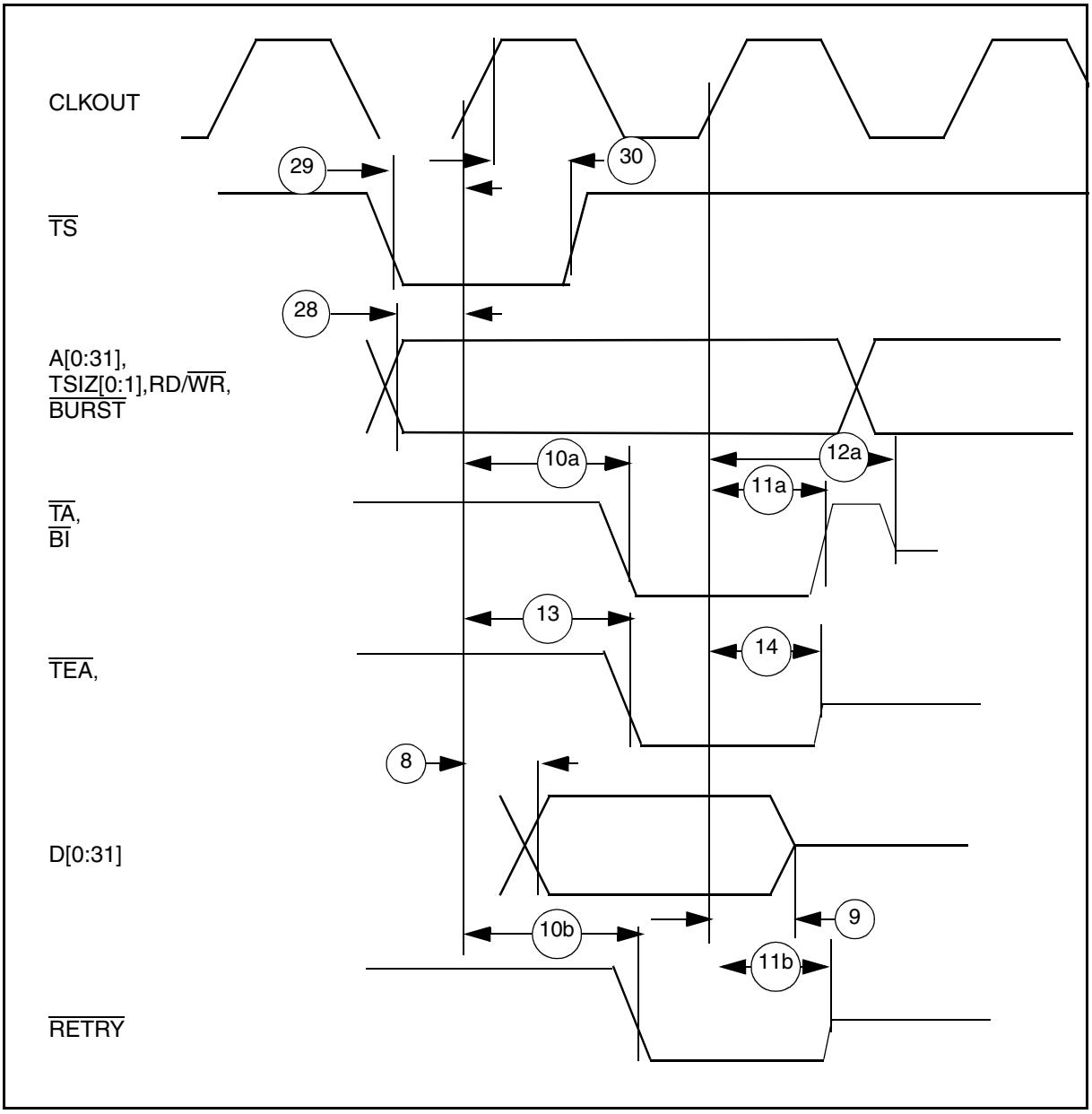


Figure G-16 External Master Read from Internal Registers Timing

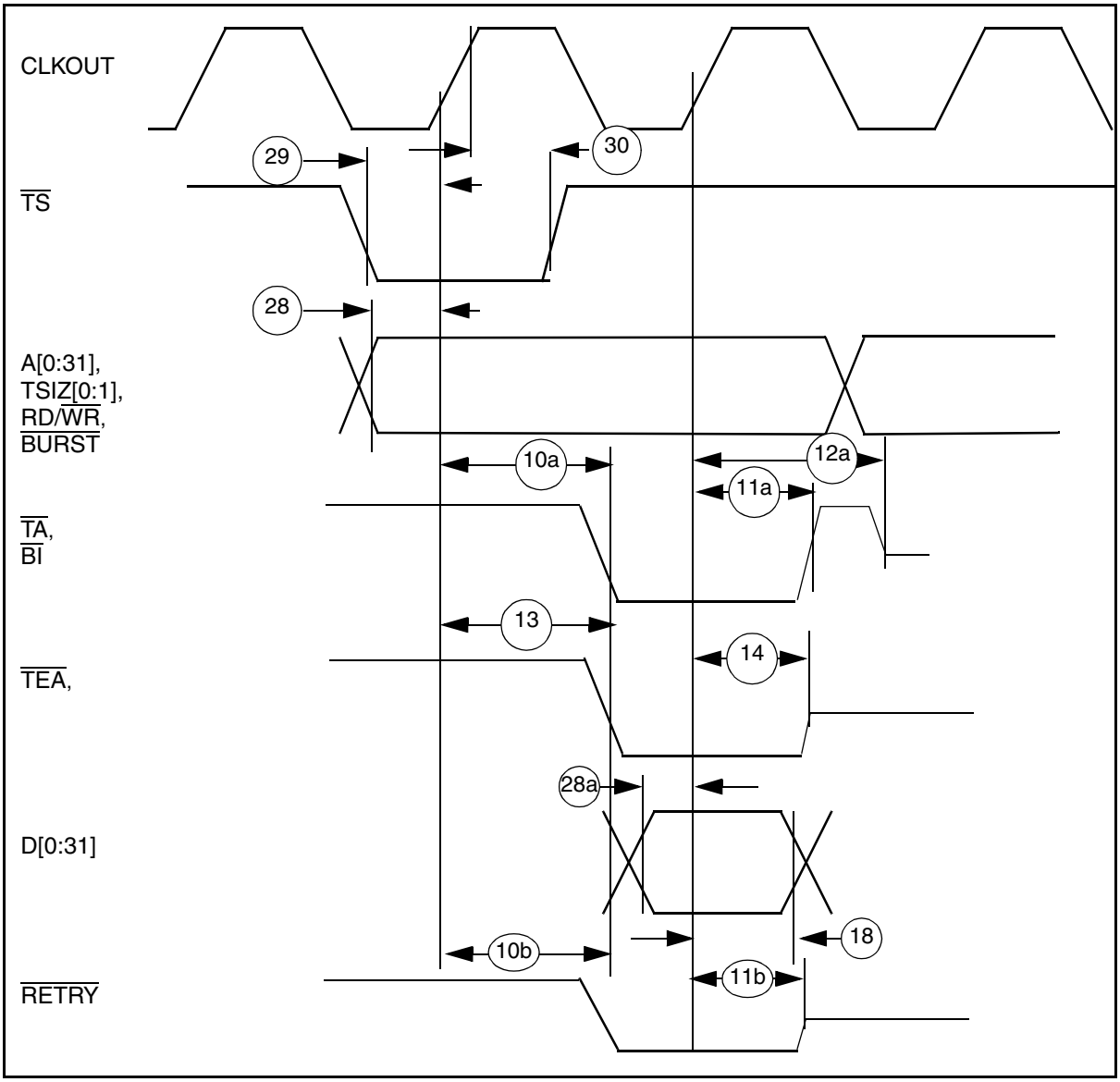


Figure G-17 External Master Write to Internal Registers Timing

G.12 Interrupt Timing

Table G-12 Interrupt Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	33MHz		40MHz		Unit
			Min	Max	Min	Max	
31	$\overline{IRQ}$x Valid to CLKOUT Rising Edge (Setup Time)[1]		10		10		ns
32	$\overline{IRQ}$x Hold Time After CLKOUT[1]		2		2		ns
33	$\overline{IRQ}$x Pulse Width Low		3		3		ns
34	$\overline{IRQ}$x Pulse Width High		3		3		ns
35	$\overline{IRQ}$x Edge to Edge Time	4*TC	121		100		ns

NOTES:

1. The timings 31 and 32 describe the testing conditions under which the $\overline{IRQ}$ lines are tested when being defined as level sensitive. The $\overline{IRQ}$ lines are synchronized internally and do not have to be asserted or negated with reference to the CLKOUT.

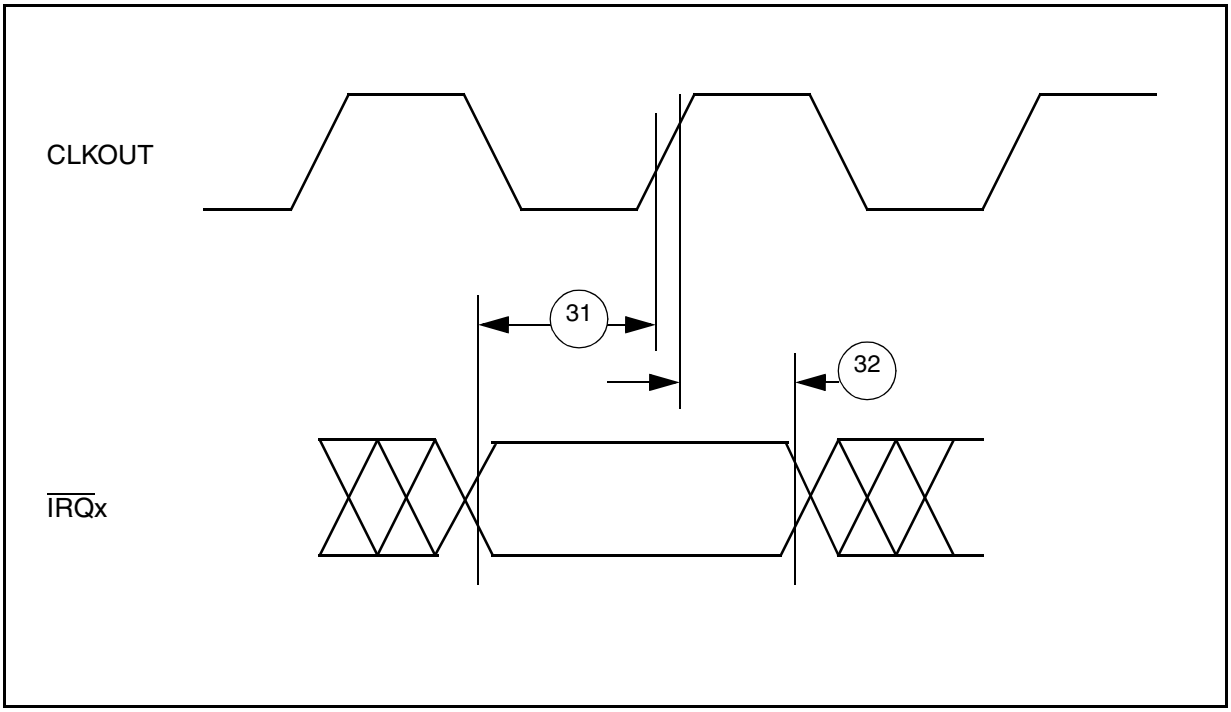


Figure G-18 Interrupt Detection Timing for External Level Sensitive Lines

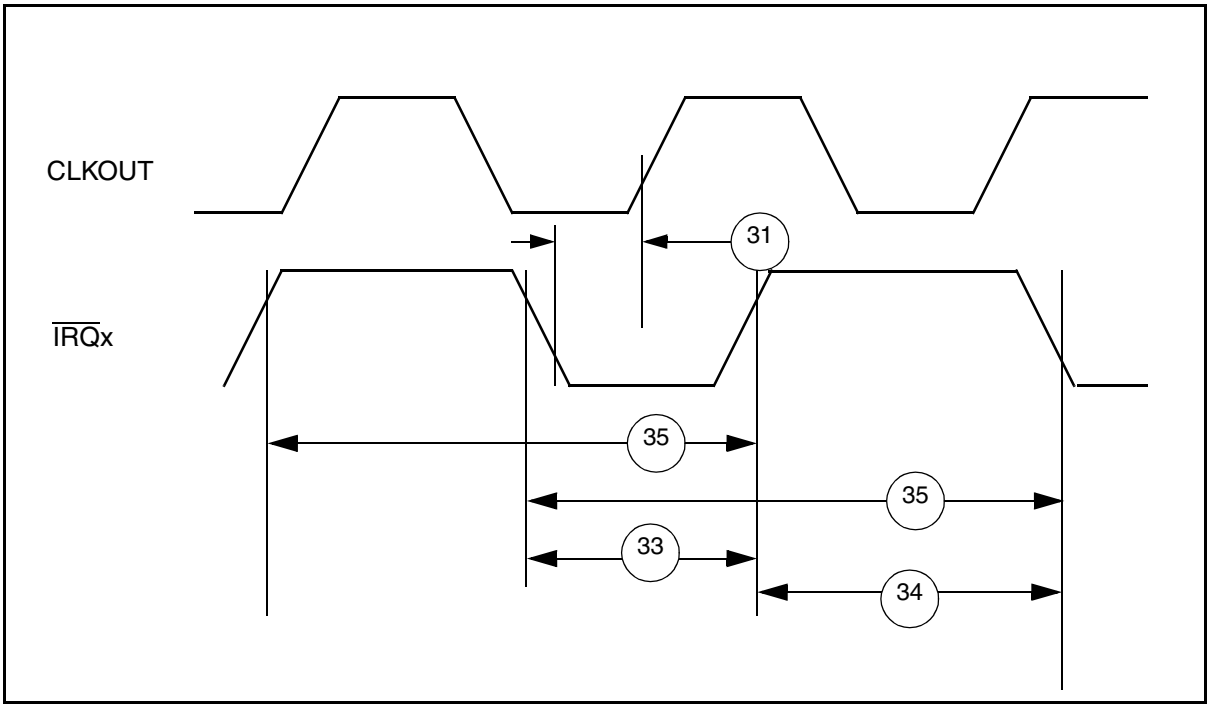


Figure G-19 Interrupt Detection Timing for External Edge Sensitive Lines

G.13 Debug Port Timing

Table G-13 Debug Port timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	33MHz		40MHz		Unit
			Min	Max	Min	Max	
36	DSCK Cycle Time		120	—	60	—	ns
37	DSCK Clock Pulse Width		50	—	25	—	ns
38	DSCK Rise and Fall Times		0	3	0	3	ns
39	DSDI Input Data Setup Time		12	—	12	—	ns
40	DSDI Data Hold Time		5	—	5	—	ns
41	DSCK low to DSDO Data Valid		0	18	0	18	ns
42	DSCK low to DSDO Invalid		0	—	0	—	ns

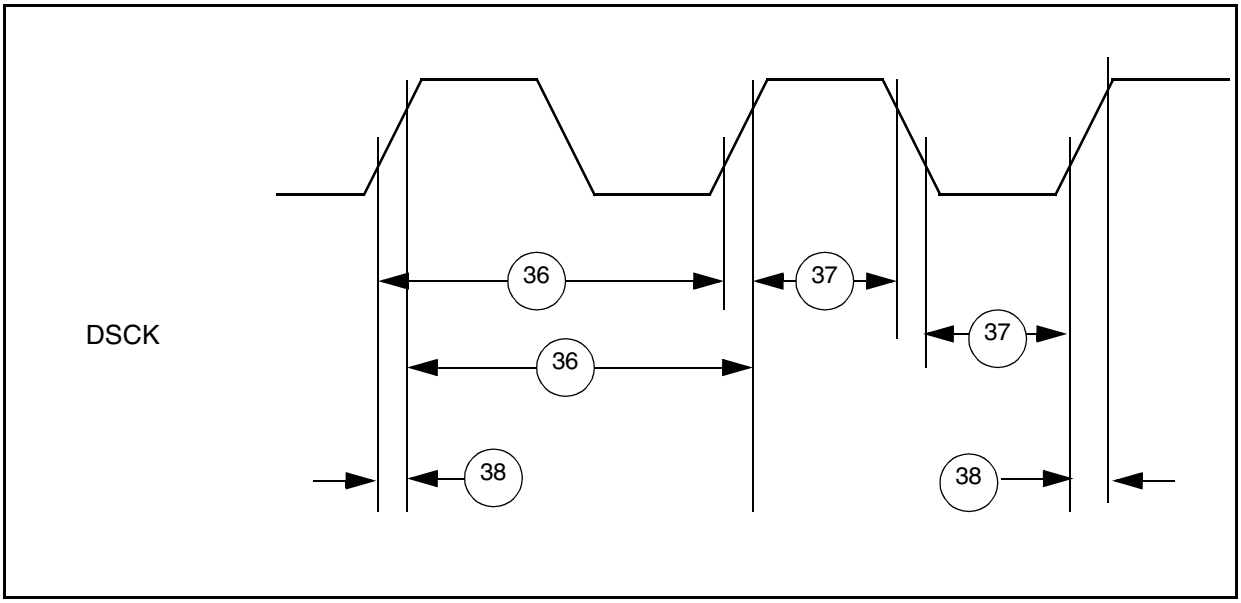


Figure G-20 Debug Port Clock Input Timing

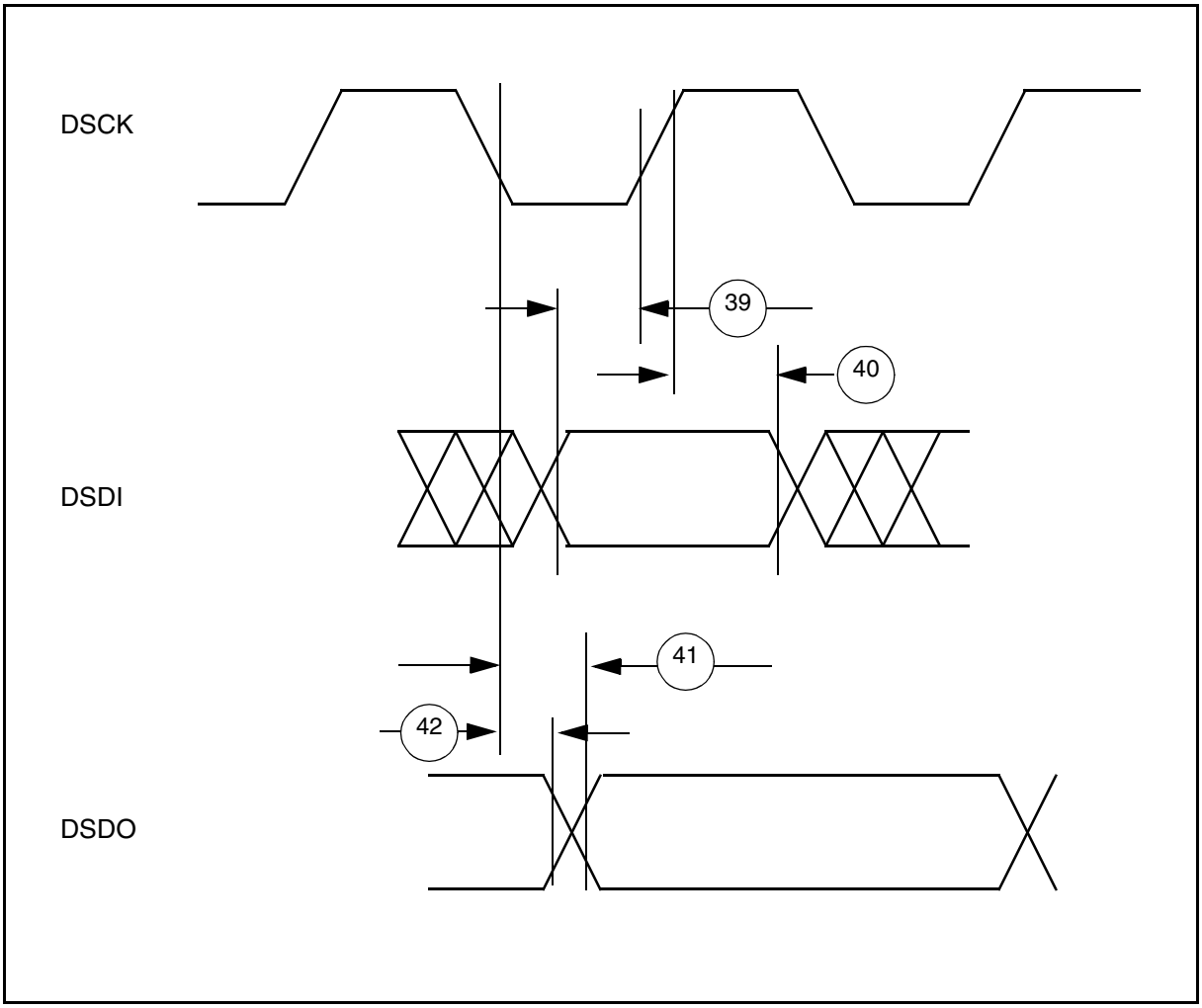


Figure G-21 Debug Port Timings

G.14 Reset Timing

Table G-14 RESET Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	33MHz		40MHz		Unit
			Min	Max	Min	Max	
43	CLKOUT to $\overline{\text{HRESET}}$ High Impedance			20		20	ns
44	CLKOUT to $\overline{\text{SRESET}}$ High Impedance			20		20	ns
45	$\overline{\text{RSTCONF}}$ Pulse Width	17*TC	515		425		ns
46	Configuration Data to HRESET Rising Edge Setup Time	15*TC + TCC	463		382		ns
47	Configuration Data to $\overline{\text{RSTCONF}}$ Rising Edge Set Up Time	15*TC + TCC	463		382		ns
48	Configuration Data Hold Time After $\overline{\text{RSTCONF}}$ Negation		0		0		ns
49	Configuration Data Hold Time After $\overline{\text{HRESET}}$ Negation		0		0		ns
49a	$\overline{\text{RSTCONF}}$ Hold Time After $\overline{\text{HRESET}}$ Negation[1]		60		50		ns
50	$\overline{\text{HRESET}}$ and $\overline{\text{RSTCONF}}$ Asserted to Data Out Drive		25		25		ns
51	$\overline{\text{RSTCONF}}$ Negated to Data Out High Impedance.		25		25		ns
52	CLKOUT of Last Rising Edge Before Chip Tristates $\overline{\text{HRESET}}$ to Data Out High Impedance.		25		25		ns
53	DSDI, DSCK Set Up	3*TC	91		75		ns
54	DSDI, DSCK hold time		0		0		ns
55	$\overline{\text{SRESET}}$ Negated to CLKOUT Rising Edge for DSDI and DSCK Sample	8*TC	243		200		ns
55a	$\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$, $\overline{\text{PORESET}}$ Pulse Width [2]		100		100		ns

NOTES:

1. Weak pullups and pulldowns used for Reset timing will comply with the 130-µA mode select current outlined in **Table G-4**.The simplest way to insure meeting this requirement in systems that require the use of the TEXP function, is to connect $\overline{\text{RSTCONF}}$/TEXP to $\overline{\text{SRESET}}$. The maximum rise time of $\overline{\text{HRESET}}$ should be less than six clock cycles.
2. $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{PORESET}}$ have a glitch detector to ensure that spikes less than 20 ns are rejected. The internal $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{PORESET}}$ will assert only if these signals are asserted for more than 100 ns.

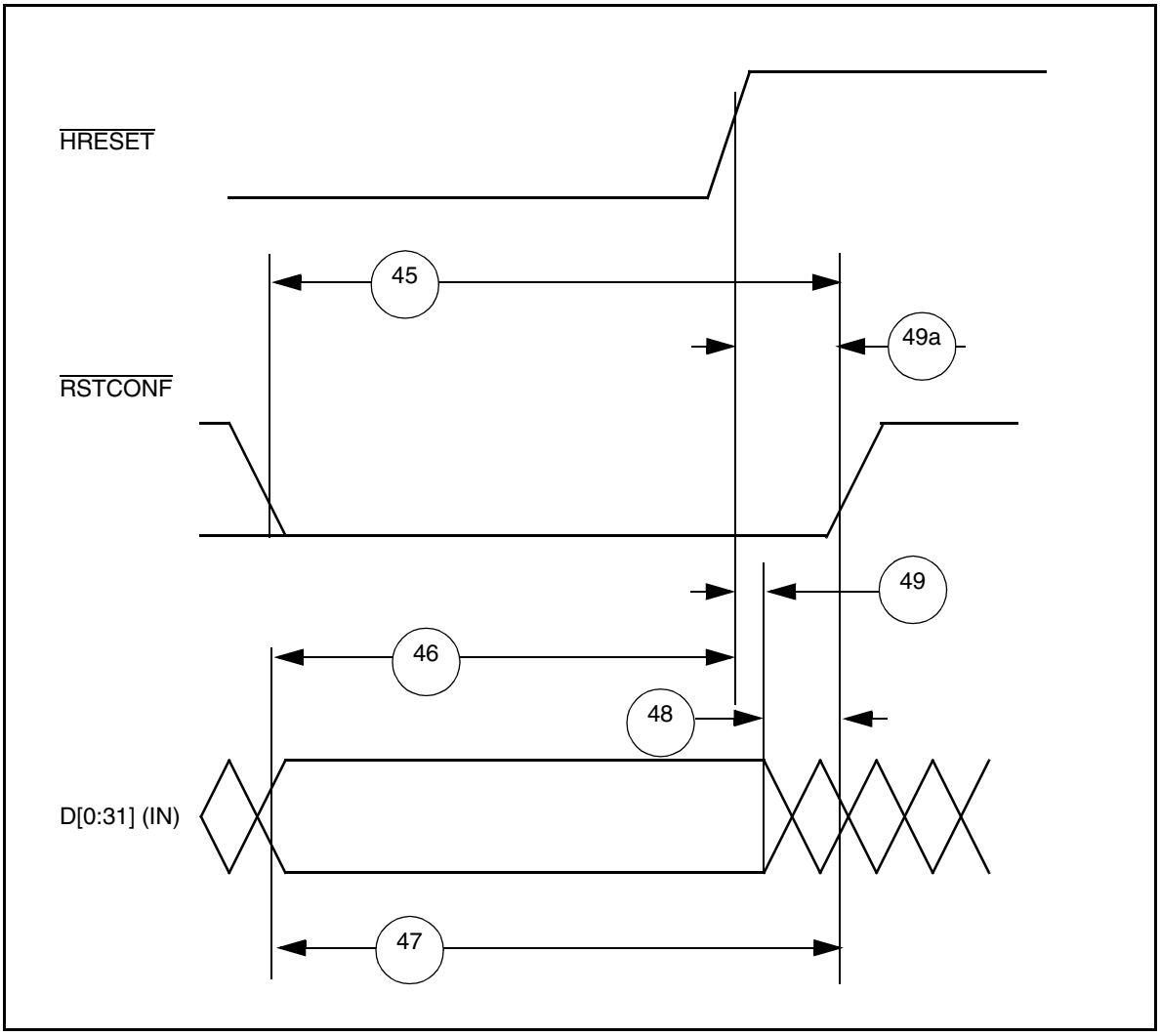


Figure G-22 Reset Timing — Configuration from Data Bus

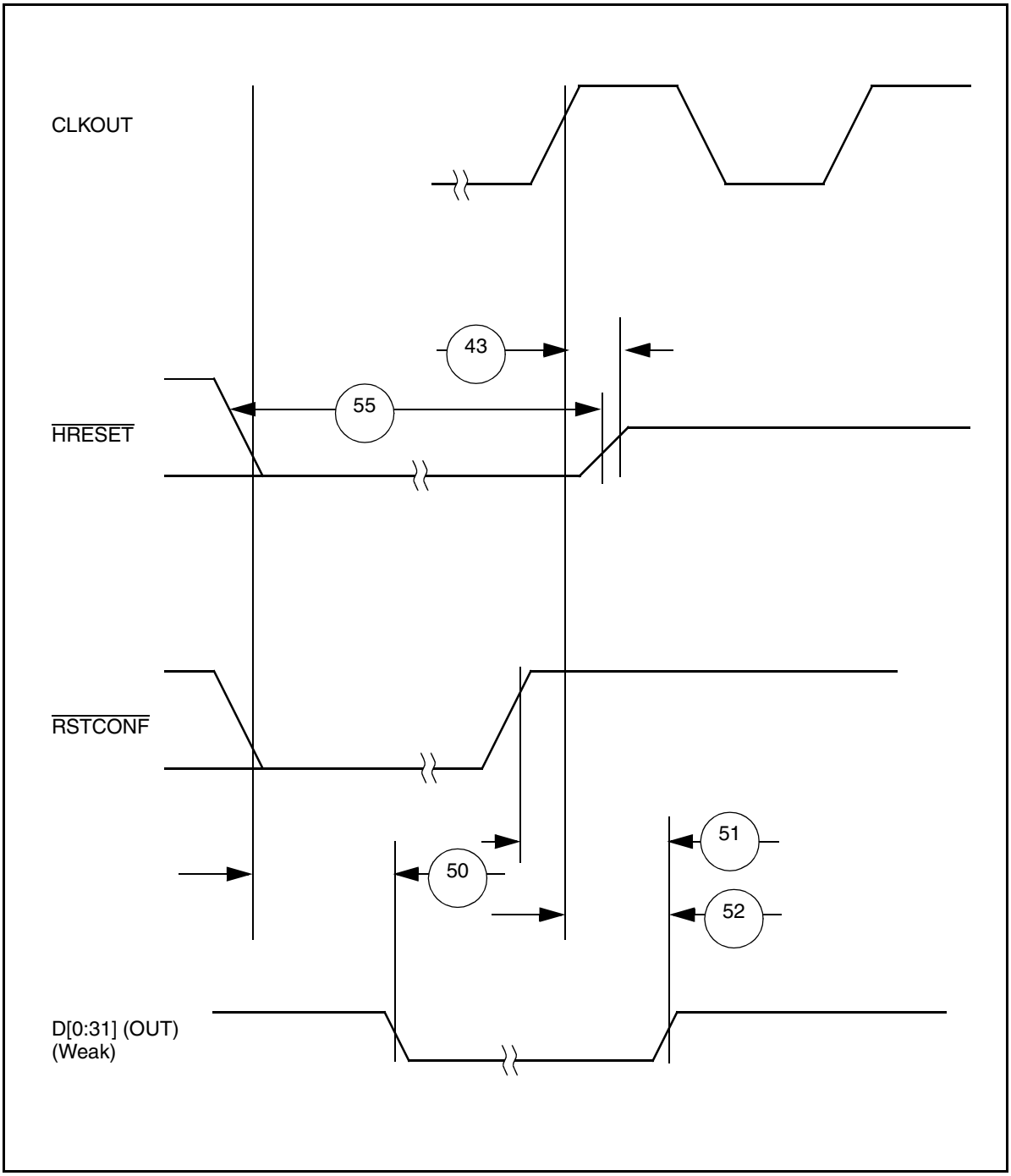


Figure G-23 Reset Timing — Data Bus Weak Drive During Configuration

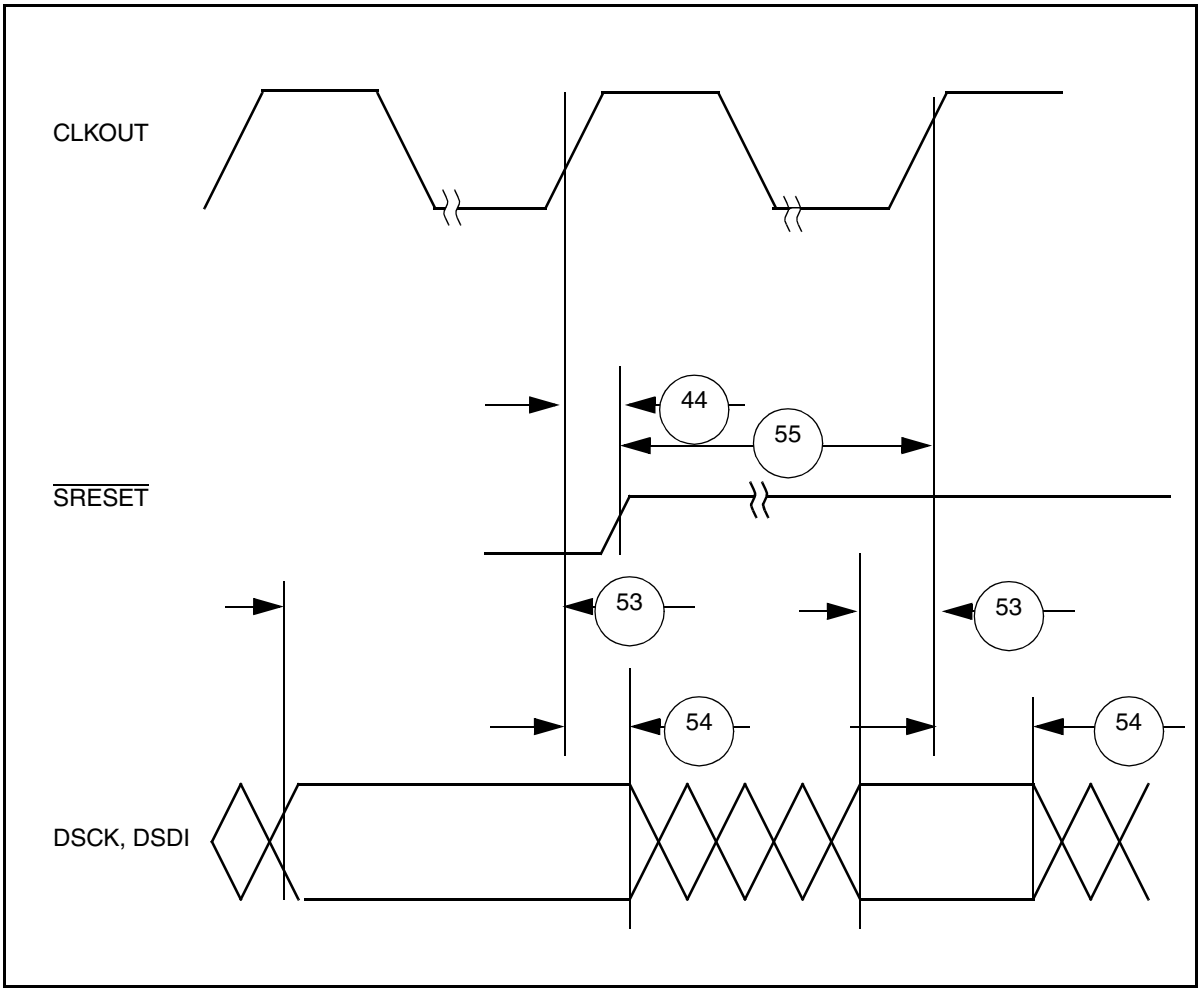


Figure G-24 Reset Timing — Debug Port Configuration

G.15 IEEE 1149.1 Electrical Characteristics

Table G-15 JTAG Timing[1]

($T_A = T_L$ to T_H)

	Characteristic	Expression	33 MHz		40 MHz		Unit
			Min	Max	Min	Max	
56	TCK Cycle Time		100	—	100	—	ns
57	TCK Clock Pulse Width Measured at $V_{DD}/2$		40	—	40	—	ns
58	TCK Rise and Fall Times		0	10	0	10	ns
59	TMS, TDI Data Setup Time		5	—	5	—	ns
60	TMS, TDI Data Hold Time		25	—	25	—	ns
61	TCK Low to TDO Data Valid		—	20	—	20	ns
62	TCK Low to TDO Data Invalid		0	—	0	—	ns
63	TCK Low to TDO High Impedance		—	20	—	20	ns
64	$\overline{\text{TRST}}$ Assert Time		100	—	100	—	ns
65	$\overline{\text{TRST}}$ Setup Time to TCK Low		40	—	40	—	ns
66	TCK Falling Edge to Output Valid		—	50	—	50	ns
67	TCK Falling Edge to Output Valid out of High Impedance		—	50	—	50	ns
68	TCK Falling Edge to Output High Impedance		—	50	—	50	ns
69	Boundary Scan Input Valid to TCK Rising Edge		50	—	50	—	ns
70	TCK Rising Edge to Boundary Scan Input Invalid		50	—	50	—	ns

NOTES:
1. JTAG timing is only tested at 10 MHz

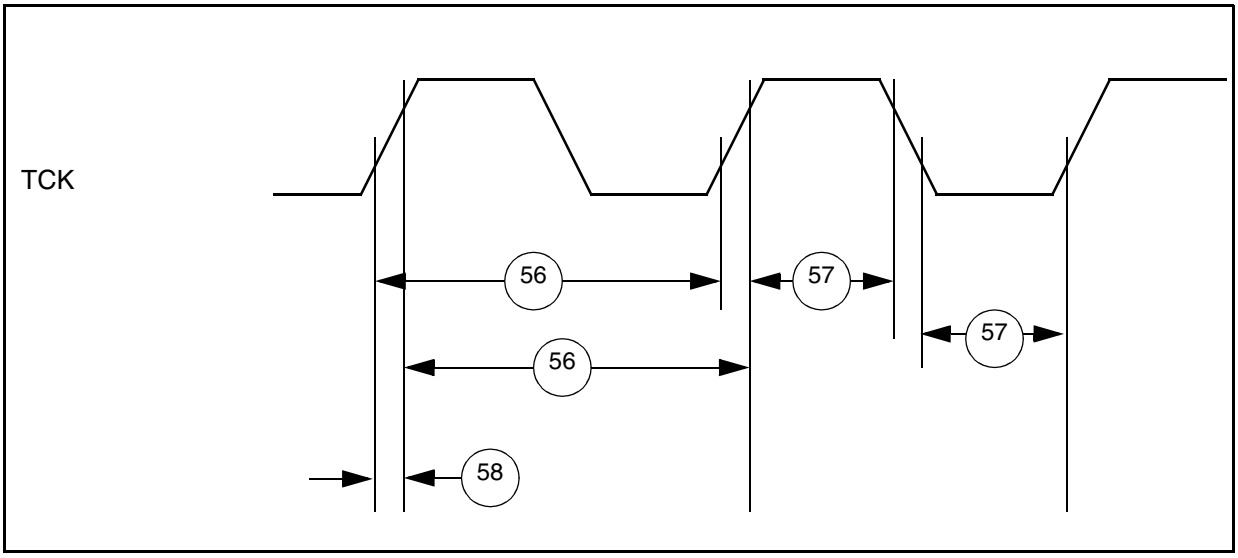


Figure G-25 JTAG Test Clock Input Timing

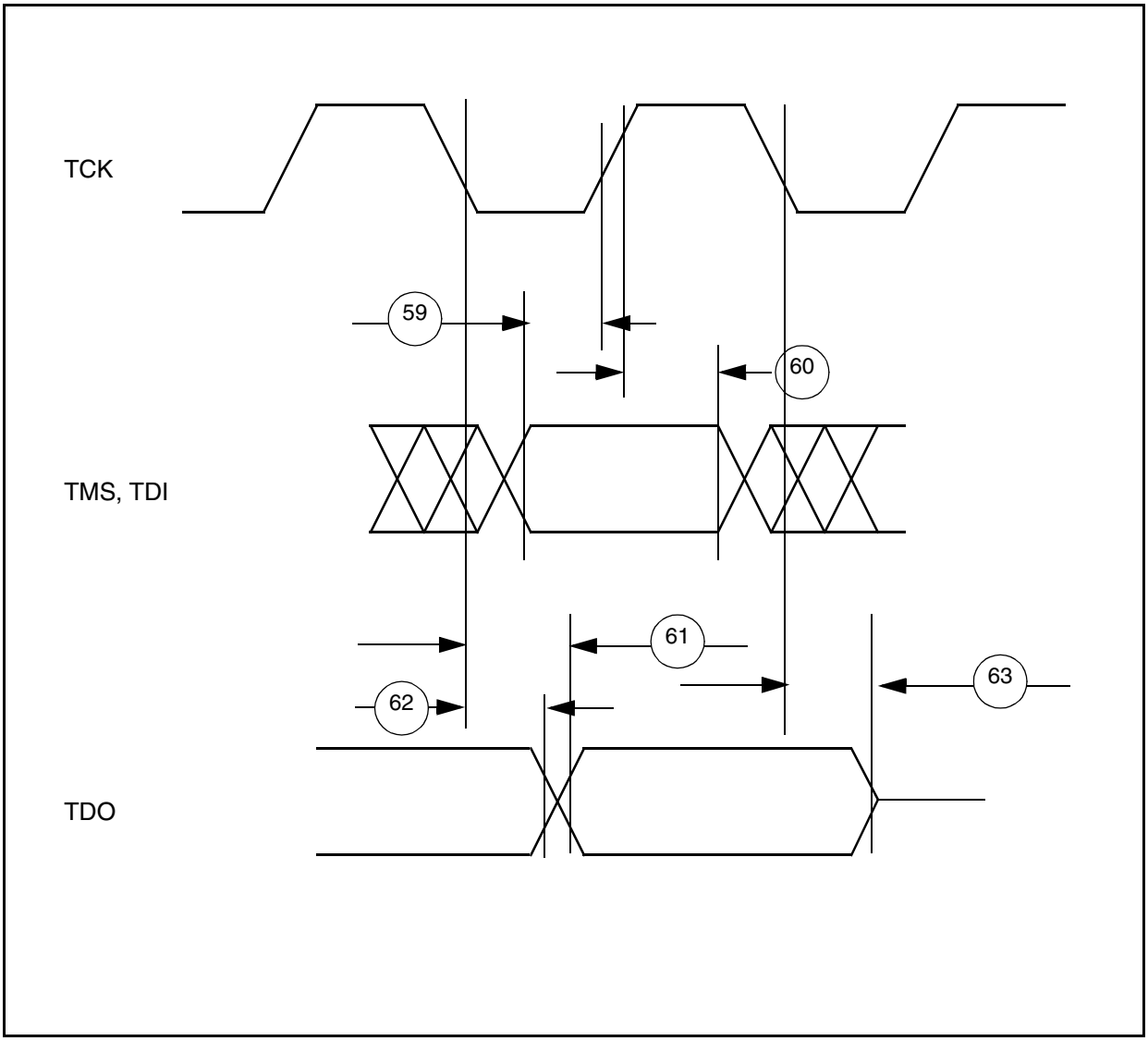


Figure G-26 JTAG — Test Access Port Timing Diagram

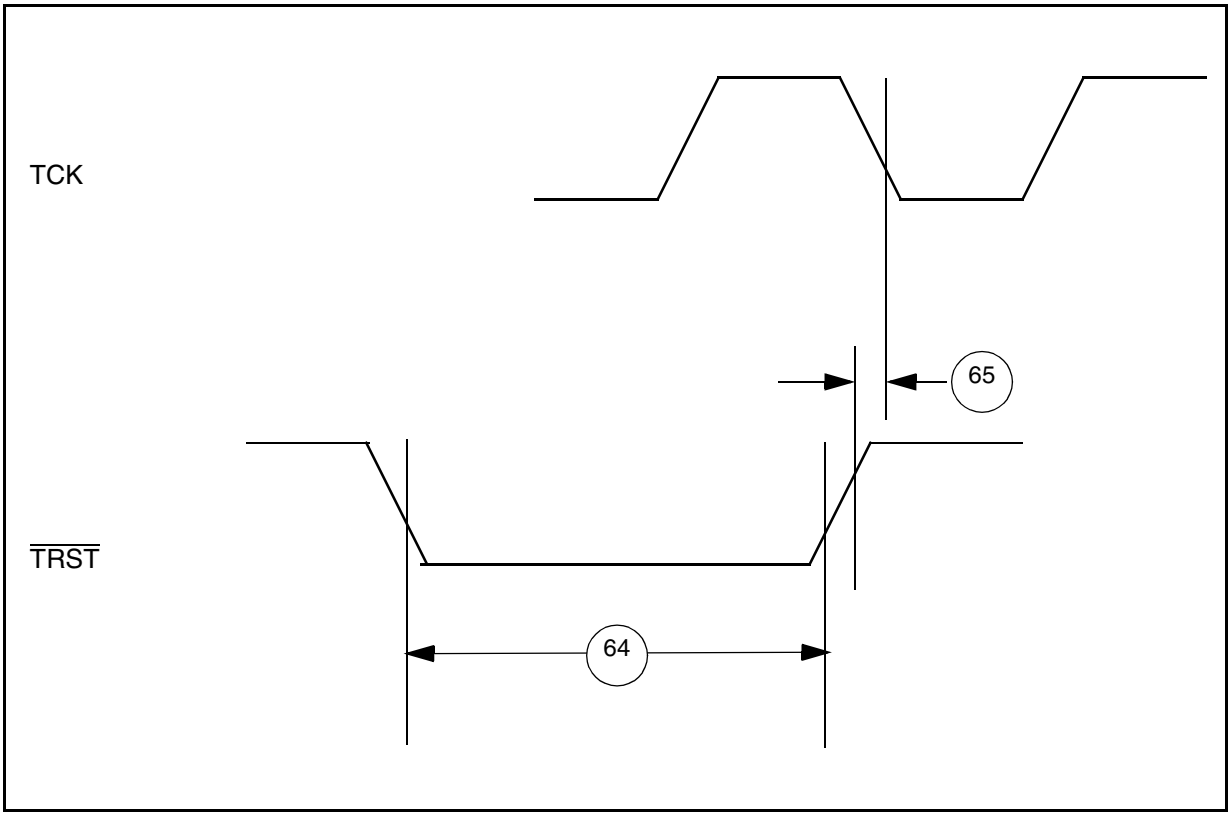


Figure G-27 JTAG — $\overline{\text{TRST}}$ Timing Diagram

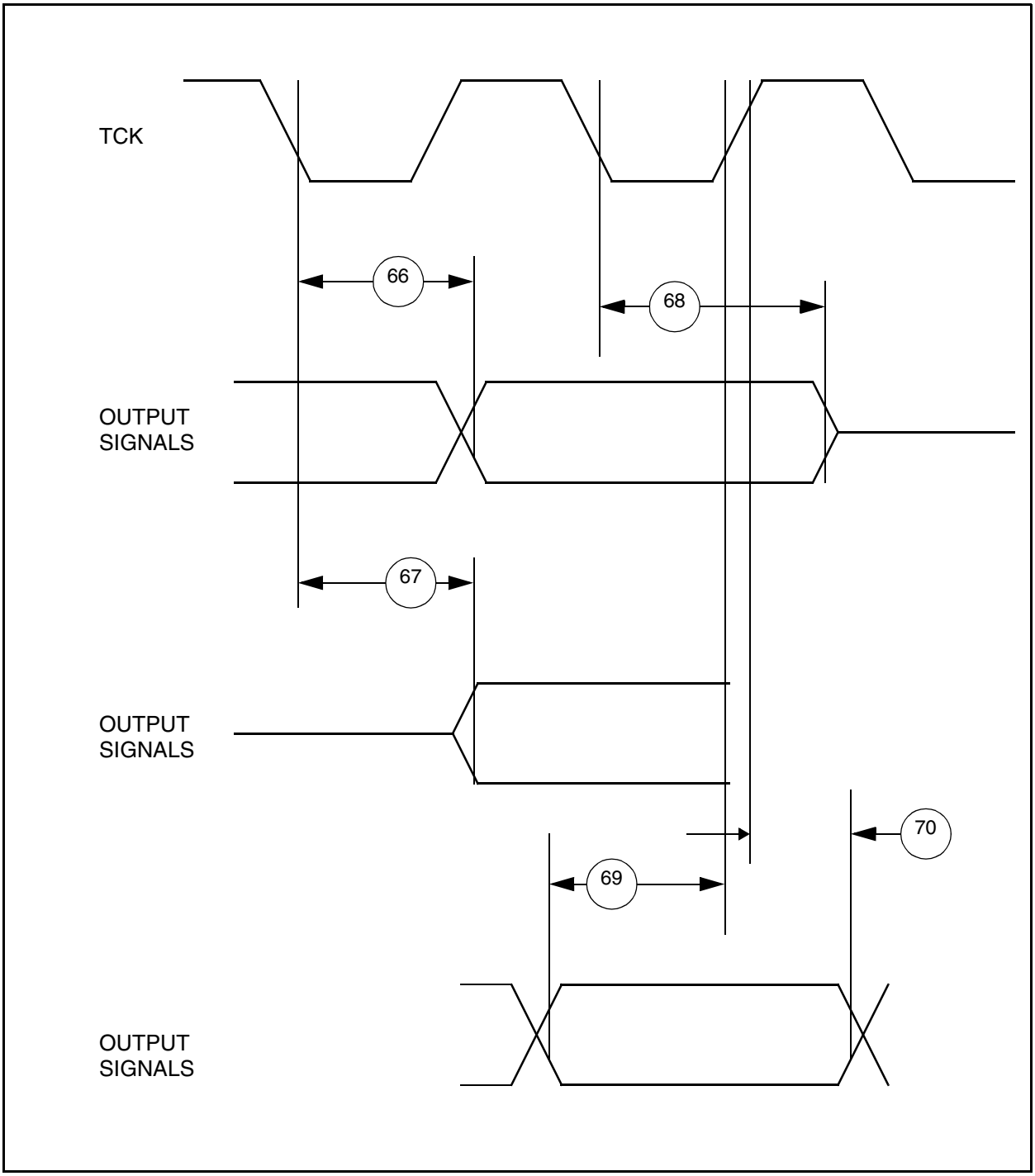


Figure G-28 Boundary Scan (JTAG) Timing Diagram

G.16 QADC64 Electrical Characteristics

Table G-16 QADC64 Conversion Characteristics (Operating)

($T_A = T_L$ to T_H)

Num	Parameter	Symbol	Min	Max	Units
97	QADC Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	3.0	MHz
98	Conversion Cycles[2]	CC	14	28	QCLK cycles
99	Conversion Time[3] F_{QCLK} = 2.0 MHz[1,4] Min = CCW/IST = 0b00 Max = CCW/IST = 0b11	T_{CONV}	7.0	14.0	µs
100	Stop Mode Recovery Time	T_{SR}	—	10	µs
101	Resolution[5]	—	5	—	mV
102	Absolute (Total Unadjusted) Error[6, 7, 8, 9] F_{QCLK} = 2.0 MHz[3], 2 Clock Input Sample Time	AE	-2.0	2.0	Counts
103	DC Disruptive Input Injection Current[10, 11, 12, 13, 14]	I_{INJ}[15] I_{INJ}[16]	-3 -1	3 1	mA
104	Current Coupling Ratio[17] PQA PQB	K	— —	$8x10^{-5}$ $8x10^{-5}$	
105	Incremental Error Due to Injection Current All Channels Have Same 10 KΩ < Rs < 100 KΩ Channel Under Test has Rs = 10 KΩ, I_{INJ} = ±3 mA	E_{INJ}		±1.0 ±1.0	Counts
106	Source Impedance At Input [18]	R_S	—	100	KΩ
107	Incremental Capacitance During Sampling [19]	C_{SAMP}	—	5	pF

NOTES:

1. Conversion characteristics vary with F_{QCLK} rate. Reduced conversion accuracy occurs at max F_{QCLK} rate.
2. The number of conversion cycles is dependent on the IST bits in the CCW register.
3. Assumes that f_{SYS} = 40 MHz
4. Assumes F_{QCLK} = 2.00 MHz, with clock prescaler values of:
   QACR0: PSH =0b01111, PSA = 0b0, PSL = 0b011)
   CCW: BYP = 0b0
5. At $V_{RH} - V_{RL}$ = 5.12 V, one count = 5 mV (at 5.0 V, one count = 4.8875 mV).
6. Accuracy tested and guaranteed at $V_{RH} - V_{RL}$ = 5.0 V ± 0.5 V
7. This parameter is periodically sampled rather than 100% tested.
8. Absolute error includes 1/2 count (~2.5 mV) of inherent quantization error and circuit (differential, integral, and offset) error. Characteristic assumes that adequate low-pass filtering is present on analog input pins — capacitive filter with 0.01 µF to 0.1 µF capacitor between analog input and analog ground, typical source isolation impedance of 10 Kbytes.
9. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals may affect the conversion accuracy of other channels.
10. Below disruptive current conditions, the channel being stressed has conversion values of 0x3FF for analog inputs greater than V_{RH} and 0x000 for values less than V_{RL}. This assumes that $V_{RH} <= V_{DDA}$ and $V_{RL} <= V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
11. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.

12. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using $V_{POSCLAMP}$ = (the lower of V_{DDA} or V_{DDH}) + 0.3 V and $V_{NEGCLAMP}$ = -0.3 V, then use the larger of the calculated values. The diode drop voltage is a function of current and varies approximately 0.4 to 0.8 volts over temperature
13. This parameter is periodically sampled rather 100% tested.
14. Derate linearly to 0.3 mA if V_{DDH} - V_{DDA} = 1 volt. This characteristic is preliminary and may change after further characterization.
15. Condition applies to two adjacent pins.
16. Condition applies to all analog channels.
17. Current coupling ratio, K, is defined as the ratio of the output current, I_{out}, measured on the pin under test to the injection current, I_{inj}, when both adjacent pins are overstressed with the specified injection current. $K = I_{out}/ I_{inj}$ The input voltage error on the channel under test is calculated as $V_{err} = I_{inj} * K * R_S$
18. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.
    Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{errj}):
    $V_{errj} = R_S * I_{OFF}$
    where I_{OFF} is a function of operating temperature.
    Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the filtering capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance
19. For a maximum sampling error of the input voltage <= 1LSB, then the external filter capacitor, C_f >= 1024 * C_{SAMP}. The value of C_{SAMP} in the new design may be reduced.

G.17 QSMCM Electrical Characteristics

Table G-17 QSPI Timing

($T_A = T_L$ to T_H, 50 pF load on all QSPI pins except as noted)

Name	Function	Symbol	Min	Max	Unit
108	Operating Frequency[1] Master Slave	f_{op}	DC DC	$F_{SYS}/4$ $F_{SYS}/4$	Hz
109	Cycle Time Master Slave	t_{qcyc}	4*TC 4*TC	510*TC[2] —	ns
110	Enable Lead Time Master Slave	t_{lead}	2*TC 2*TC	128*TC —	ns
111	Enable Lag Time Master Slave	t_{lag}	— 2*TC	SCK/2 —	ns
112	Clock (SCK) High or Low Time Master Slave[3]	t_{sw}	2*TC – 60 2*TC – n	255*TC —	ns
113	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{td}	17*TC 13*TC	8192*TC —	ns
114	Data Setup Time (Inputs) Master Slave	t_{su}	30 20	— —	ns

Table G-17 QSPI Timing (Continued)

($T_A = T_L$ to T_H, 50 pF load on all QSPI pins except as noted)

Name	Function	Symbol	Min	Max	Unit
115	Data Hold Time (Inputs) Master Slave	t_{hi}	 0 20	 — —	ns
116	Slave Access Time	t_a	—	TC	ns
117	Slave MISO Disable Time	t_{dis}	—	2*TC	ns
118	Data Valid (After SCK Edge) Master Slave	t_v	 — —	 50 50	ns
119	Data Hold Time (Outputs) Master Slave	t_{ho}	 0 0	 — —	ns
120	Rise Time Input Output up to 50 pF Load, SLRC Bit of PDMCR = “0” up to 200 pF Load, SLRC Bit of PDMCR = “0” up to 50 pF, SLRC Bit of PDMCR = “1”	 t_{ri} t_{ro}	 — 10 20 2	 1 50 100 25	 µs ns ns ns
121	Fall Time Input Output up to 50 pF Load, SLRC Bit of PDMCR = “0” up to 200 pF Load, SLRC Bit of PDMCR = “0” up to 50 pF, SLRC Bit of PDMCR = “1”	 t_{fi} t_{fo}	 — 10 20 2	 1 50 100 25	 µs ns ns ns

NOTES:

1. All AC timing is tested to the 5-V levels outlined in **Table G-4**.
2. TC is defined to be the clock period of f_{SYS} (IMB Clock).
3. For high time, n = External SCK rise time; for low time, n = External SCK fall time.

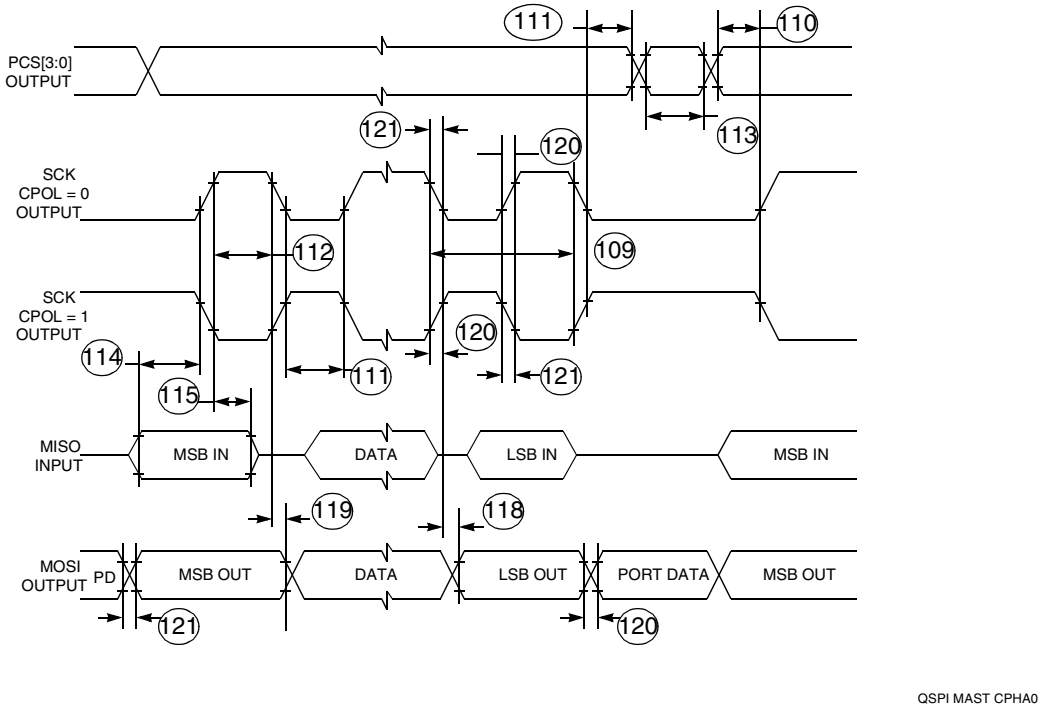


Figure G-29 QSPI Timing — Master, CPHA = 0

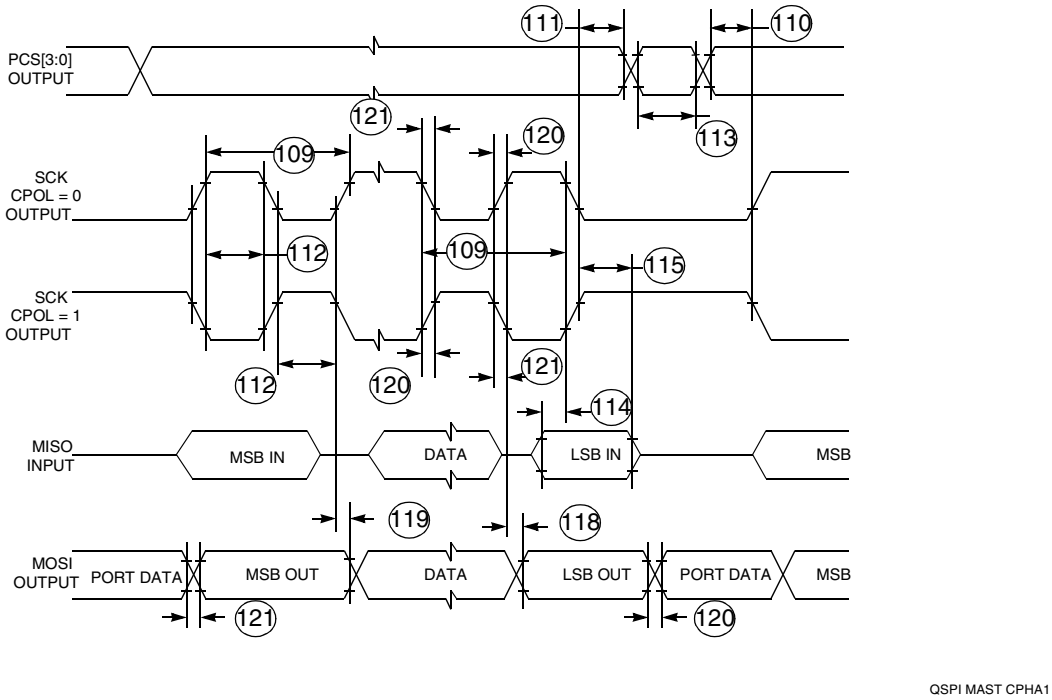


Figure G-30 QSPI Timing — Master, CPHA = 1

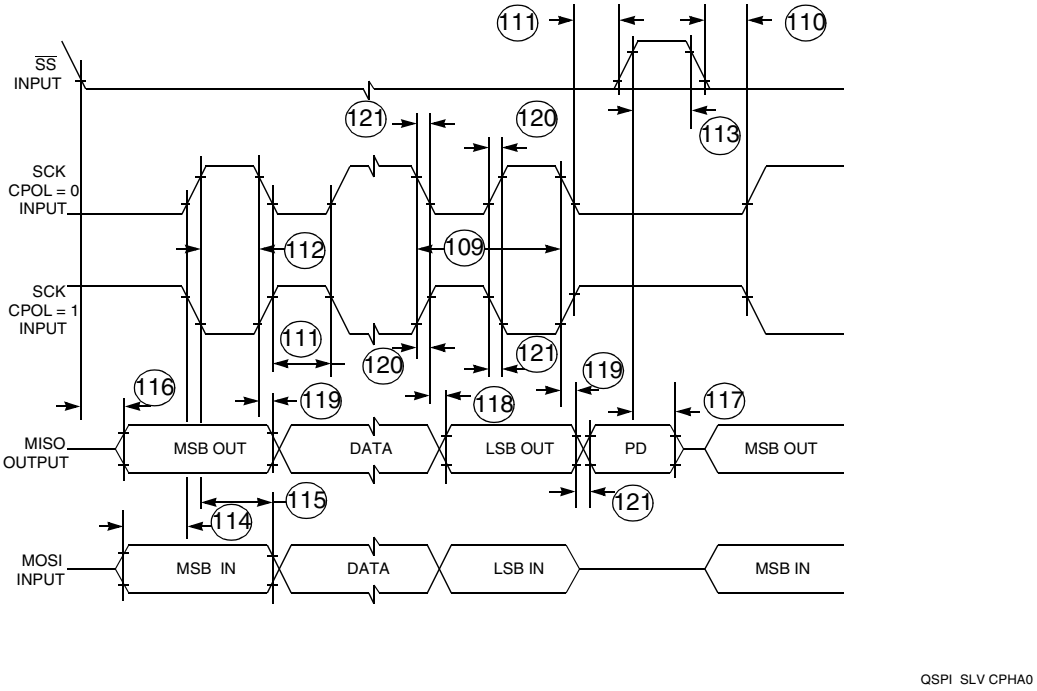


Figure G-31 QSPI Timing — Slave, CPHA = 0

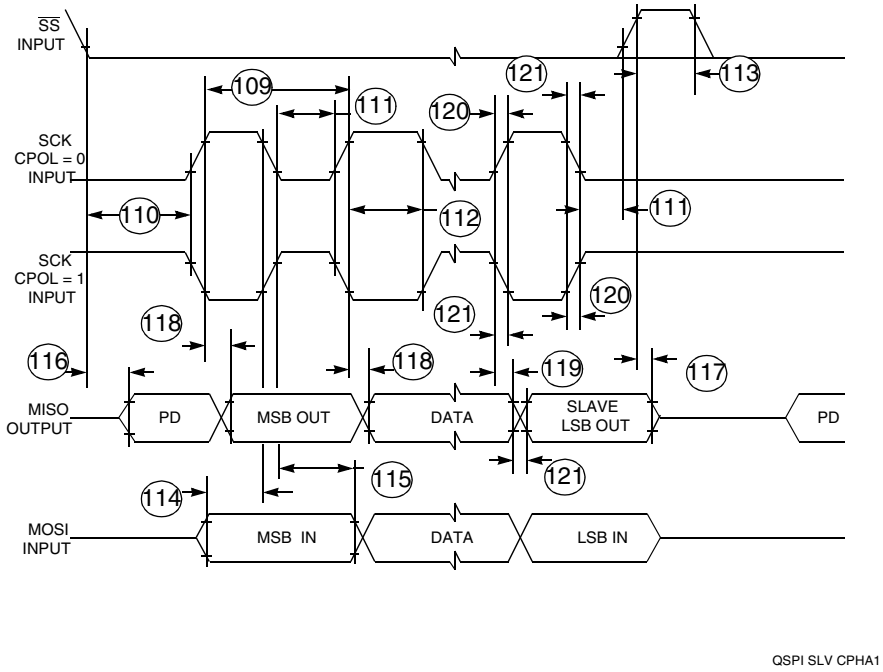


Figure G-32 QSPI Timing — Slave, CPHA = 1

G.18 GPIO Electrical Characteristics

Table G-18 GPIO Timing[1]

($T_A = T_L$ to T_H)

Num	Rating	Symbol	Min	Max	Unit
122	Rise Time Input Output[2] (SLR0 of PDMCR = 0), 50 pF to 200 pF Load Output[2] (SLR0 of PDMCR = 0), up to 20 nF Load[3] Output (SLR0 of PDMCR = 1), up to 50 pF Load	t_{ri} t_{ro}	— 90 2000 3	1 600 7500 25	µs ns ns ns
123	Fall Time Input Output[2](SLR0 of PDMCR = 0), 50 pF to 200 pF Load Output[2] (SLR0 of PDMCR = 0), up to 20 nF Load[3] Output (SLR0 of PDMCR = 1), up to 50 pF Load	t_{fi} t_{fo}	— 90 2000 3	1 600 7500 25	µs ns ns ns

NOTES:
1. GPIO applies to all pins used as GPIO: SGPIOA[8:31], SQPIOD[0:31], SGPIOC[0:7], QGPIO[0:6], QGPO[1:2], MPIO[0:15], A_PQA[0:7], B_PQA[0:7], A_PQB[0:7] (inputs only), B_PQB[0:7] (inputs only).
2. This parameter is tested during initial characterization and is not tested in production.
3. Care should be taken to insure that the total power dissipation of the device remain below the absolute maximum rating under this condition. See **Table G-1**. With a capacitive load > 20 nF (up to 100 nF maximum), the user must insure that the pin is always configured as an output and set to slow slew rate mode (SLR0 of PDMCR = 0). Do not change SLR0 of PDMCR to a 1 under these conditions.

G.19 TPU3 Electrical Characteristics

Table G-19 TPU3 Timing

($T_A = T_L$ to T_H)

Num	Rating	Symbol	Min	Max	Unit
124	Slew Rate of TPU Output Channel Valid[1,2] (SLR0 of PDMCR = 0, 50 pF to 200 pF Load) (SLR0 of PDMCR = 0, up to 20 nF Load)[3] (SLR0 of PDMCR = 1, up to 50 pF Load)	t_{CHTOV}	92 2000 3	650 7550 25	ns
125	CLKOUT High to TPU Output Channel Hold	t_{CHTOH}	0	15	ns
126	TPU Input Channel Pulse Width[4]	t_{TIPW}	4	—	t_{cyc}

NOTES:
1. AC timing is shown with respect to 10% V_{DDH} and 90% V_{DDH} levels. Total slew rate from 0 to V_{DDH} will be larger.
2. Timing not valid for external T2CLK input.
3. Care should be taken to insure that the total power dissipation of the device remain below the absolute maximum rating under this condition. See **Table G-1**. With a capacitive load > 20 nF, the user must insure that the pin is always configured as an output.
4. t_{cyc} is defined as the IMB Clock Period.

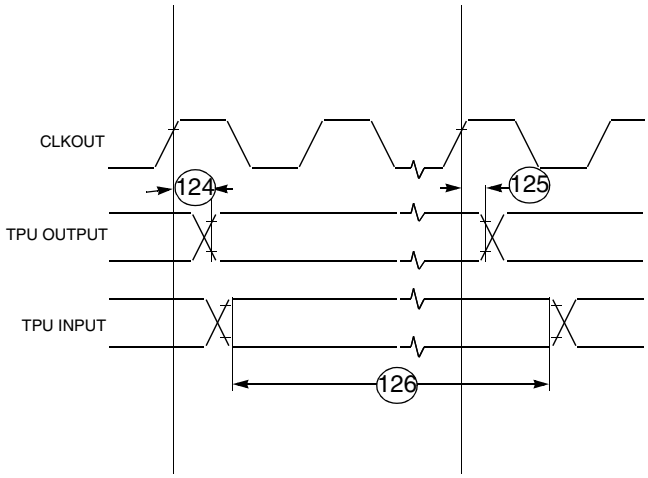


Figure G-33 TPU3 Timing

G.20 TouCAN Electrical Characteristics

Table G-20 TouCAN Timing[1]

($T_A = T_L$ to T_H)

Num	Rating	Symbol	Min	Max	Unit
127	CNTX0 (Delay from ICLOCK)	t_{CNTX0}	19	—	ns
128	CNRX0 (Set-Up to ICLOCK Rise)	t_{CNRX0}	0	—	ns
129	Rise Time Input Output Up to 50 pF Load, SLRC Bit of PDMCR = “0” Up to 200 pF Load, SLRC Bit of PDMCR = “0” Up to 50 pF Load, SLRC Bit of PDMCR = “1”	 t_{ri} t_{ro}	 — 10 20 2	 1 50 100 25	 µs ns ns ns
130	Fall Time Input Output Up to 50 pF Load, SLRC Bit of PDMCR = “0” Up to 200 pF Load, SLRC bit of PDMCR = “0” Up to 50 pF Load, SLRC Bit of PDMCR = “1”	 t_{fi} t_{fo}	 — 10 20 2	 1 50 100 25	 µs ns ns ns
	Serial Pins	t_f	—	1	Mhz

NOTES:
1. AC timing is shown is tested to the 5-V levels outlined in **Table G-4**.

G.21 MIOS Timing Characteristics

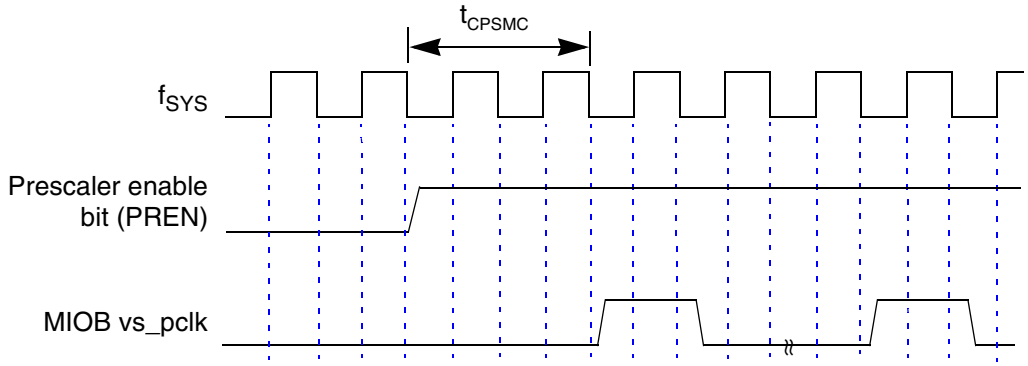
All MIOS output pins are slew rate controlled. Slew rate control circuitry adds 90 ns as minimum to the output timing and 650 ns as a maximum. This slew rate is from 10% V_{DDH} to 90% V_{DDH}, an additional 100 ns should be added for total 0 to V_{DDH} slew rate.

Table G-21 MCPSM Timing Characteristics

Characteristic	Symbol	Delay	Unit
MCPSM Enable to vs_pclk Pulse[1]	t_{CPSMC}	(MCPSMSCR_PSL[3:0]) -1[2]	IMB Clock Cycles

NOTES:

1. The MCPSM clock prescaler value (MCPSMSCR_PSL[3:0]) should be written to the MCPSMSCR (MCPSM status/control register) before rewriting the MCPSMSCR to set the enable bit (MCPSMSCR_PREN). If this is not done the prescaler will start with the old value in the MCPSMSCR_PSL[3:0] before reloading the new value into the counter.
   vs_pclk is the MIOS prescaler clock which is distributed to all the counter (e.g., MPWMSM and MMCSM) submodules.
2. After reset MCPSMSCR_PSL[3:0] is set to 0b0000.



Note 1: f_{SYS} is the internal system clock for the IMB3 bus.
Note 2: The numbers associated with the f_{SYS} ticks refer to the IMB3 internal state.
Note 3: vs_pclk is the MIOS prescaler clock which is distributed around the MIOS to counter modules such as the MMCSM and MPWMSM.

Figure G-34 MCPSM Enable to vs_pclk Pulse Timing Diagram

G.21.1 MPWMSM Timing Characteristics

Table G-22 MPWMSM Timing Characteristics

(All delays are in IMB clock periods)

Characteristic	Symbol	Min	Max
PWMSM Output Resolution	t_{PWMR}	– [1]	2.0 [2]
PWM Output Pulse[3]	t_{PWMO}	2.0	—
mpwmi Input Pin to MPWMSCR_PIN Status Set	t_{PIN}	1	2
CPSM Enable To Output Set[4]	t_{PWMP}	(MPWMPERR – MPWMPULR+ 1) * (256 – MPWMSCR_CP) * MCPSMSCR_PSL +1	
MPWMSM Enable to Output Set (MIN)[5]	t_{PWME}	(MPWMPERR – MPWMPULR) * (256 – MPWMSCR_CP) * MCPSMSCR_PSL +3 + (255 – MPWMSCR_CP) * MCPSMSCR_PSL [6]	
MPWMSM Enable to Output Set (MAX)[5]	t_{PWME}	t_{PWME}(MIN) + MCPSMSCR_PSL – 1 [6]	
Interrupt Flag to Output Pin Reset (Period Start)[7]	t_{FLGP}	(256 – MPWMSCR_CP) * MCPSMSCR_PSL – 1 [6]	

NOTES:
1. Minimum output resolution depends on MPWMSM and MCPSM prescaler settings.
2. Maximum resolution is obtained by setting CPSMPSL[3:0] =0x2 and MPWMSCR_CP[7:0] =0xFF.
3. Excluding the case where the output is always “0”.
4. With MPWMSM enabled before enabling the MCPSM. Please also see Note *1 on the MCPSM timing information.
5. The exact timing from mpwmsm enable to the pin being set depends on the timing of the register write and the MCPSM vs_pclk.
6. When MCPSMSCR_PSL = 0x0000, this gives a prescale value of 16 and it is 16 which should be used in these calculations. When MCPSMSCR_PSL = 0x0001, the CPSM is inactive.
7. Note: the interrupt is set before the output pin is reset (Signifying the start of a new period).

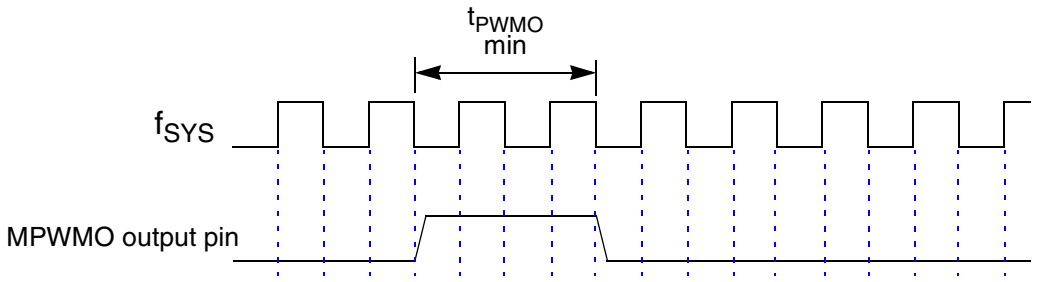
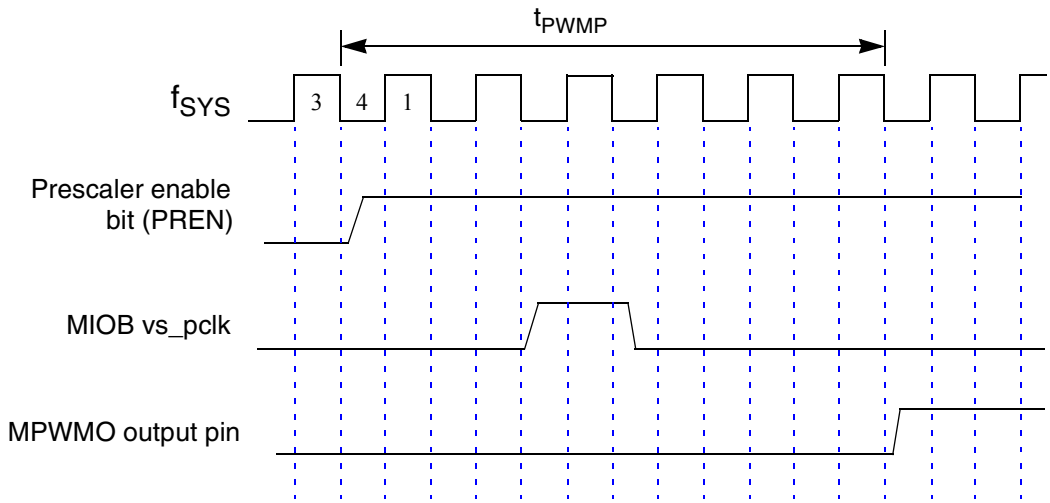


Figure G-35 MPWMSM Minimum Output Pulse Example Timing Diagram



Note: F_{SYS} is the internal IMB clock for the IMB3 bus.

Figure G-36 MCPSM Enable to MPWMO Output Pin Rising Edge Timing Diagram

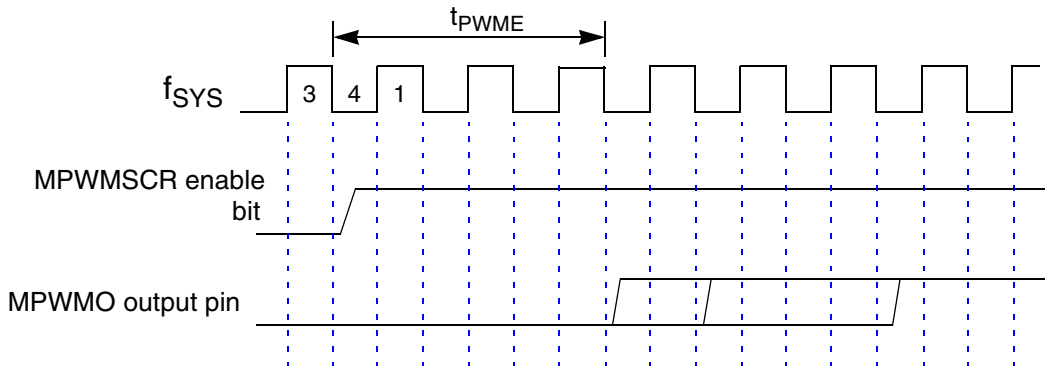


Figure G-37 MPWMSM Enable to MPWMO Output Pin Rising Edge Timing Diagram

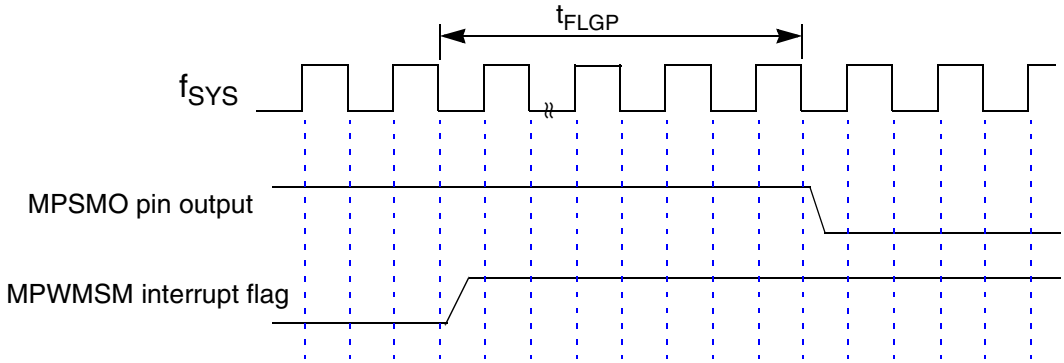


Figure G-38 MPWMSM Interrupt Flag to MPWMO Output Pin Falling Edge Timing Diagram

G.21.2 MMCSM Timing Characteristics

Table G-23 MMCSM Timing Characteristics

(All delays are in IMB clock periods.)

Characteristic	Symbol	Min	Max
MMCSM Input Pin Period	t_{PPER}	4	—
MMCSM Pin Low Time	t_{PLO}	2	—
MMCSM Pin High Time	t_{PHI}	2	—
Clock Pin to Counter Bus Increment	t_{PCCB}	1	2
Load Pin to New Counter Bus Value	t_{PLCB}	1	2
Clock Pin to PINC Delay	t_{PINC}	1	2
Load Pin to PINL Delay	t_{PINL}	1	2
Counter Bus Resolution	t_{CBR}	–[1]	2[2]
Counter Bus Overflow Reload to Interrupt Flag	t_{CBFLG}	1	
MCPSM Enable to Counter Bus Increment	t_{MCMP}	(256 – MMCSMSCR_CP) * MCPSMSCR_PSL + 2[3]	
MMCSM Enable to Counter Bus Increment (MIN)[4]	t_{MCME}	4 + MCPSMSCR_PSL* (255 – MMCSMSCR_CP)[3]	
MMCSM Enable to Counter Bus Increment (MAX)[4]	t_{MCME}	4 + MCPSMSCR_PSL * (255 – MMCSMSCR_CP) + (MCPSMSCR_PSL – 1)[3]	

NOTES:
1. Minimum output resolution depends on MMCSM and MCPSM prescaler settings.
2. Maximum resolution is obtained by setting CPSMPSL[3:0] = 0x2 and MMCSMSCR_CP[7:0] = 0xFF.
3. When MCPSMSCR_PSL = 0x0000, this gives a prescale value of 16 and it is 16 which should be used in these calculations. When MCPSMSCR_PSL = 0x0001, the CPSM is inactive.
4. The exact timing from MMCSM enable to the pin being set depends on the timing of the MMCSMSCR register write and the MCPSM vs_pclk. The MMCSM enable is taken to mean the MMCSMSCR_CLS[1:0] being written to 2‘b11.

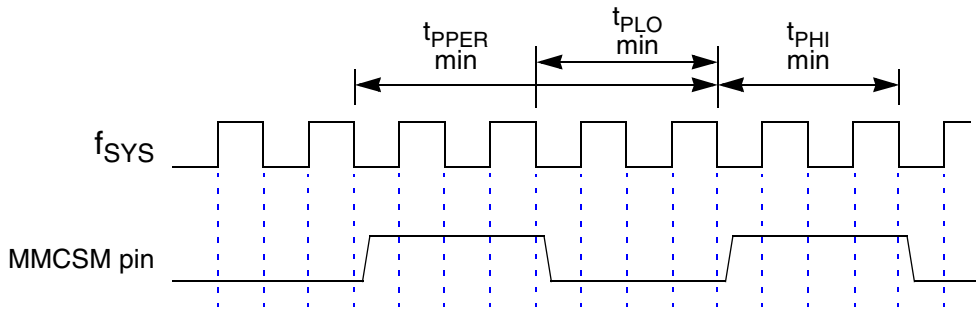
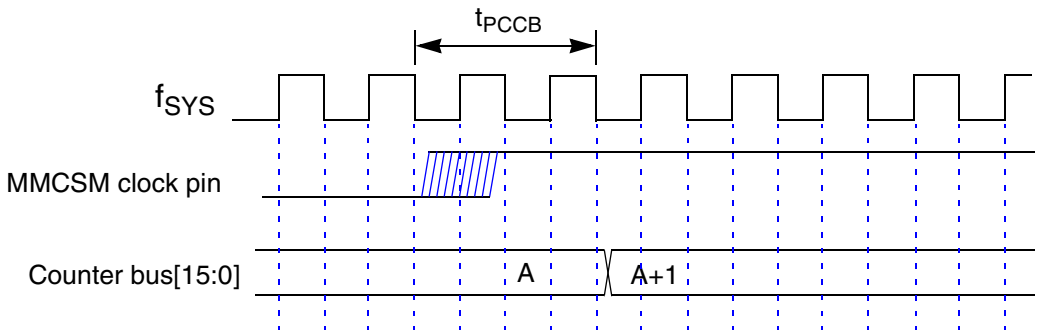


Figure G-39 MMCSM Minimum Input Pin (Either Load or Clock) Timing Diagram



Note: F_{SYS} is the internal IMB clock for the IMB3 bus.

Figure G-40 MMCSM Clock Pin to Counter Bus Increment Timing Diagram

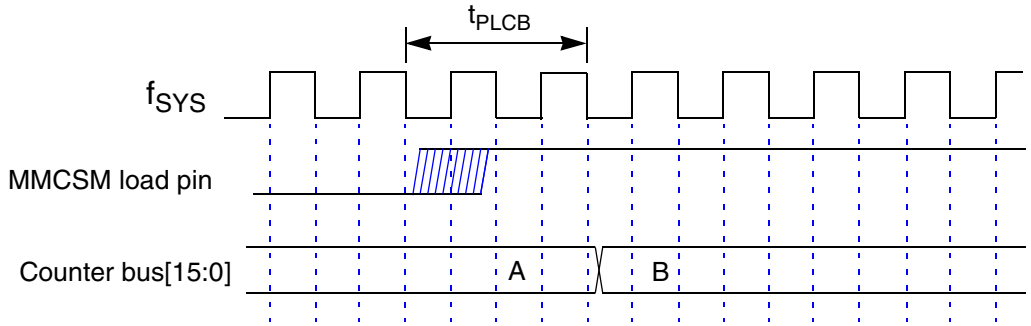


Figure G-41 MMCSM Load Pin to Counter Bus Reload Timing Diagram

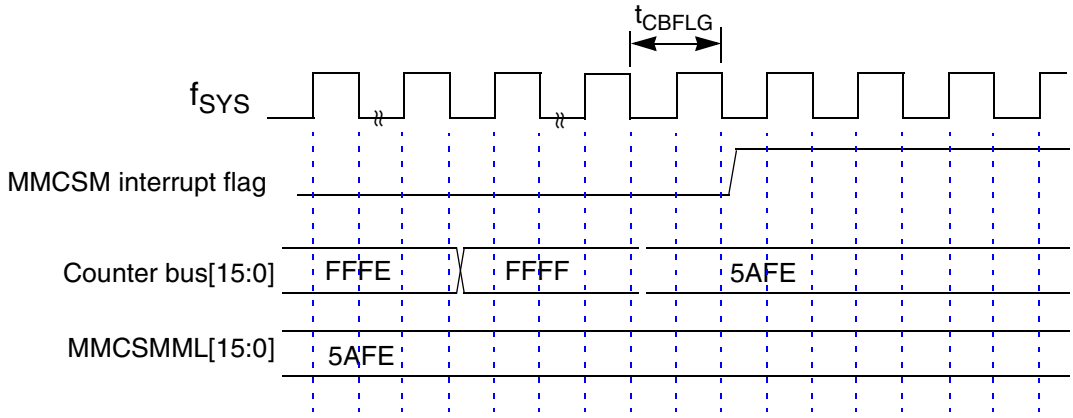


Figure G-42 MMCSM Counter Bus Reload to Interrupt Flag Setting Timing Diagram

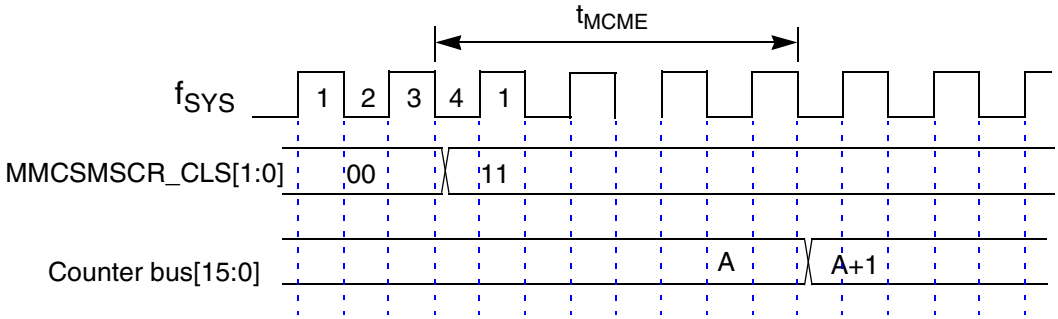


Figure G-43 MMCSM Prescaler Clock Select to Counter Bus Increment Timing Diagram

G.21.3 MDASM Timing Characteristics

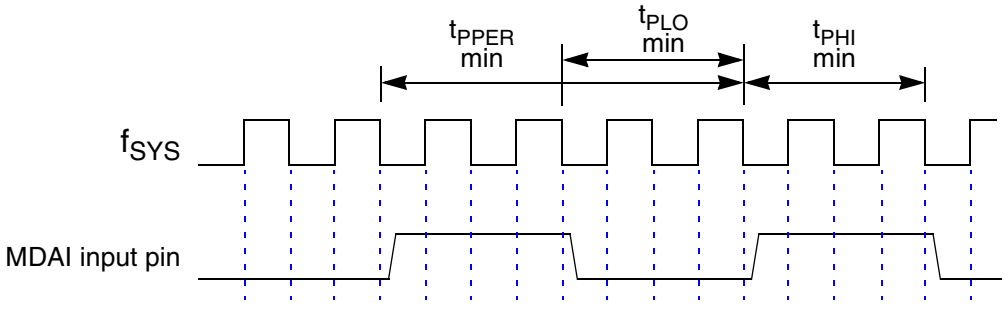
Table G-24 MDASM Timing Characteristics

(All delays are in IMB clock periods.)

Characteristic	Symbol	Min	Max
Input Modes: (IPWM, IPM, IC, DIS)			
MDASM Input Pin Period	t_{PPER}	4	—
MDASM Pin Low Time	t_{PLO}	2	—
MDASM Pin High Time	t_{PHI}	2	—
Input Capture Resolution	t_{CAPR}	—	2
Input Pin to Counter Bus Capture Delay	t_{PCAP}	1	3[1]
Input Pin to Interrupt Flag Delay	t_{PFLG}	2	3
Input Pin to PIN Delay	t_{PIN}	1	2
Counter Bus Resolution	t_{CBR}	—	2[2]
Output Modes: (OC, OPWM)			
Output Pulse Width[3]	t_{PULW}	2	—
Compare Resolution	t_{COMR}	—	2[2]
Counter Bus to Pin Change	t_{CBP}	3	
Counter Bus to Interrupt Flag Set.	t_{CBFLG}	3	

NOTES:

1. If the counter bus capture occurs when the counter bus is changing then the capture is delayed one cycle. In situations where the counter bus is stable when the input capture occurs the t_{PCAP} has a maximum delay of 2 cycles. (the 1 cycle uncertainty is due to the synchronizer).
2. Maximum resolution is obtained by setting CPSMPSL[3:0] = 0x2 and MDASMSCR_CP[7:0] = 0xFF.
3. Maximum output resolution and pulse width depends on counter (e.g., MMCSM) and MCPSM prescaler settings.



f_{SYS} is the internal IMB clock for the IMB3 bus.

Figure G-44 MDASM Minimum Input Pin Timing Diagram

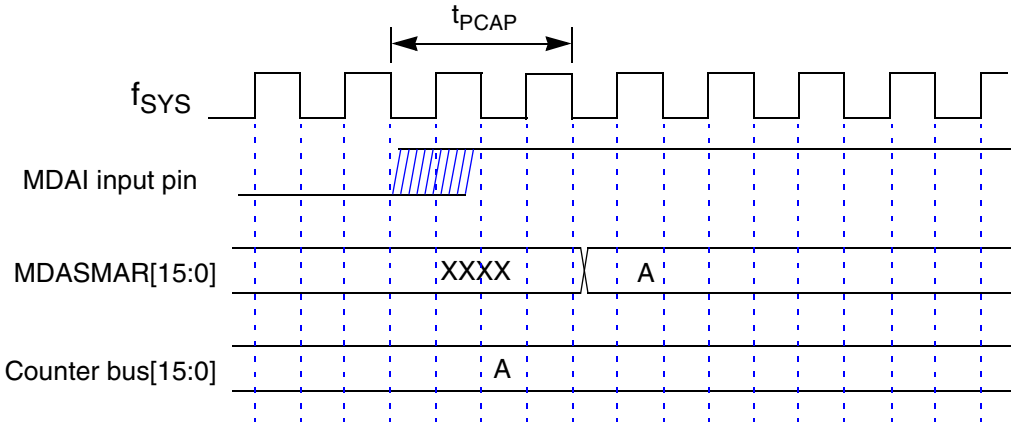


Figure G-45 MDASM Input Pin to Counter Bus Capture Timing Diagram

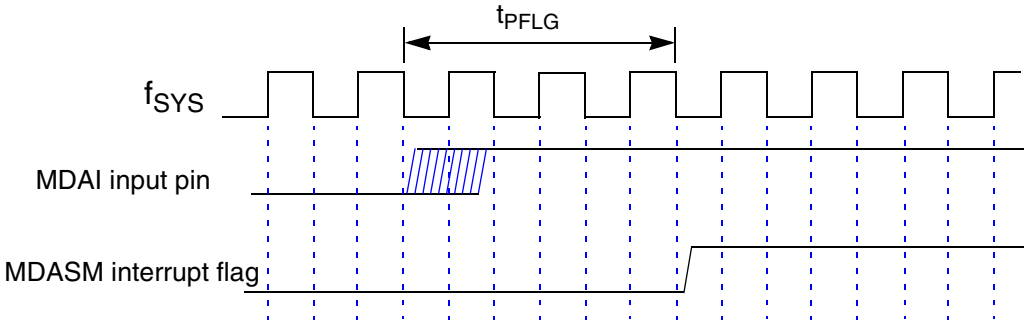


Figure G-46 MDASM Input Pin to MDASM Interrupt Flag Timing Diagram

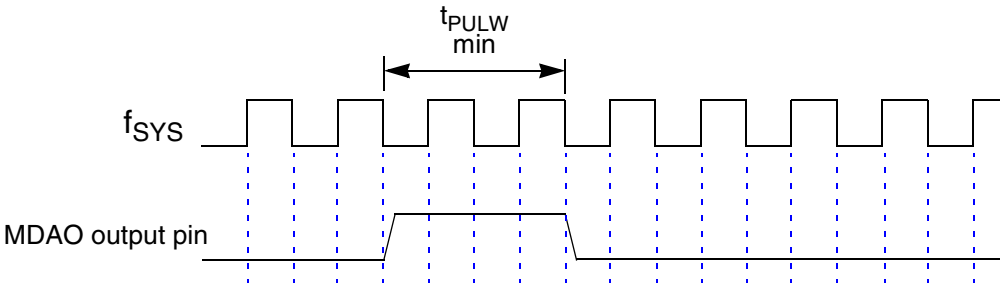


Figure G-47 MDASM Minimum Output Pulse Width Timing Diagram

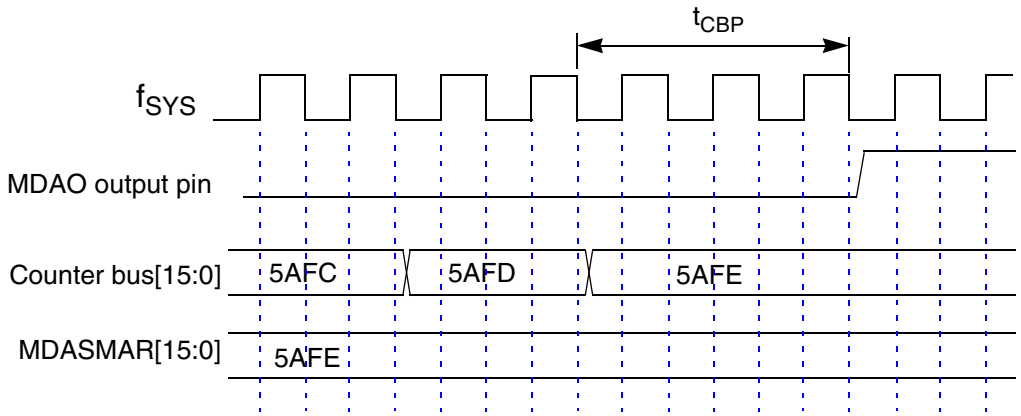


Figure G-48 Counter Bus to MDASM Output Pin Change Timing Diagram

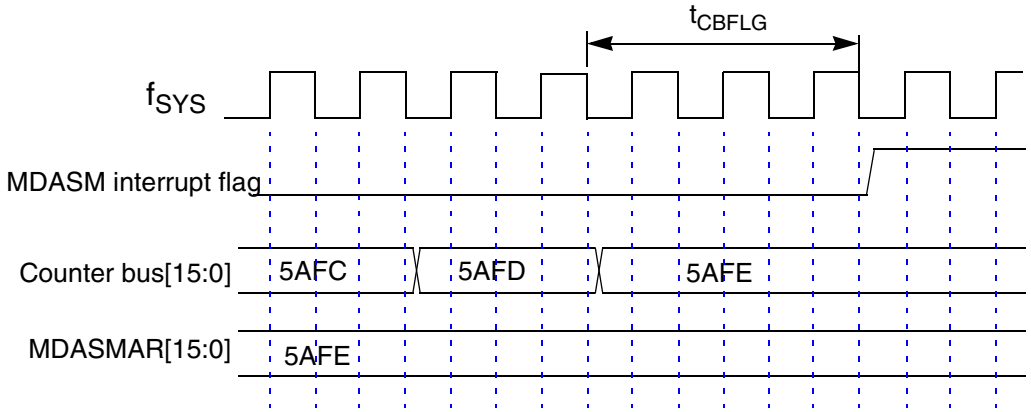


Figure G-49 Counter Bus to MDASM Interrupt Flag Setting Timing Diagram

G.21.4 MPIOSM Timing Characteristics

Table G-25 MPIOSM Timing Characteristics

(All delays are in IMB clock periods.)

Characteristic	Symbol	Min	Max
Input Mode			
MPIOSM Input Pin Period	t_{PPER}	—[1]	—
MPIOSM Pin Low Time	t_{PLO}	—[1]	—
MPIOSM Pin High Time	t_{PHI}	—[1]	—
Input Pin to MPIOSM_DR Delay	t_{PDR}	0	1
Output Mode			
Output Pulse Width[2]	t_{PULW}	—[2]	—

NOTES:

1. The minimum input pin period, pin low and pin high times depend on the rate at which the MPIOSM_DR register is polled.
2. The minimum output pulse width depends on how quickly the CPU updates the value inside the MIOPSM_DR register. The MPC555 RCPU core takes six clock cycles to access the MIOPSM_DR register, therefore the minimum output pulse will be 12 IMB clocks.

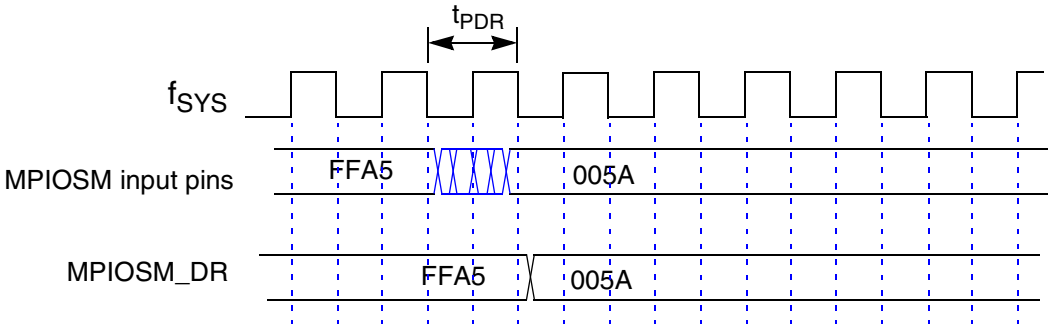


Figure G-50 MPIOSM Input Pin to MPIOSM_DR (Data Register) Timing Diagram

APPENDIX H
FLASH ELECTRICAL CHARACTERISTICS FOR ALL J76N MASK SETS AND 0K02A AND 1K02A ONLY

H.1 Electrical Characteristics

Table H-1 Program and Erase Characteristics

(V_{DDF} = 3.3 V ± 0.3 V, V_{PP}= 4.75 V to 5.25 V, $T_A = T_L$ to T_H)

Symbol	Meaning	Value			Units
		Minimum	Typical	Maximum	
E_{PULSE}	Number of Erase Pulses	1	1	1	
T_{ERASE}	Erase Pulse Time	0.9	1	1.1	S
$P_{PULSE(4.75\ Vpp)}$	Number of Program Pulses @V_{PP} = 4.75	—	1200[1]	3500[2,3,5]	Pulses
$P_{PULSE(5.0\ Vpp)}$	Number of Program Pulses @V_{PP} = 5.00	—	300[4]	1000[5]	Pulses
$P_{PULSE\ (5.25\ Vpp)}$	Number of Program Pulses[6] @V_{PP} = 5.25	—	125[7]	500[5]	Pulses
T_{PROG}	Program Pulse Time	21.2	25.6	32.0	µS
$C_{PULSE(4.75\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 4.75	1	1	3[8]	Pulses
$C_{PULSE(5.0\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 5.00	1	1	3[8]	Pulses
$C_{PULSE(5.25\ Vpp)}$	Number of CENSOR Clear Pulses @V_{PP} = 5.25	1	1	3[8]	Pulses
T_{CLEAR}	CENSOR Clear Pulse Time	0.9[8]	1[8]	1.1[8]	S
$S_{PULSE(4.75\ Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 4.75	1	1	3[8]	Pulses
$S_{PULSE(5.0\ Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 5.00	1	1	3[8]	Pulses
$S_{PULSE(5.25\ Vpp)}$	Number of CENSOR Set Pulses @V_{PP} = 5.25	1	1	3[8]	Pulses
T_{SET}	CENSOR Set Pulse Time	0.9[8]	1[8]	1.1[8]	S

NOTES:
1. The typical number of pulses at V_{PP} = 4.75 V and T_A = 25°C.
2. The worst case programming time occurs at V_{PP} = 4.75 V and T_A = -40 °C.
3. This value is based on initial device characterization and is not tested in production.
4. The typical number of pulses is at V_{PP} = 5.00 v and T_A = 25°C.
5. Assumes pulse width = 25.6 µs.
6. The best case (fastest) programming time of < 50 pulses is at V_{PP} = 5.25 V and T_A = 125°C.
7. The typical number of pulses is at V_{PP} = 5.25 V and T_A = 25°C.
8. After characterization this value may be improved.

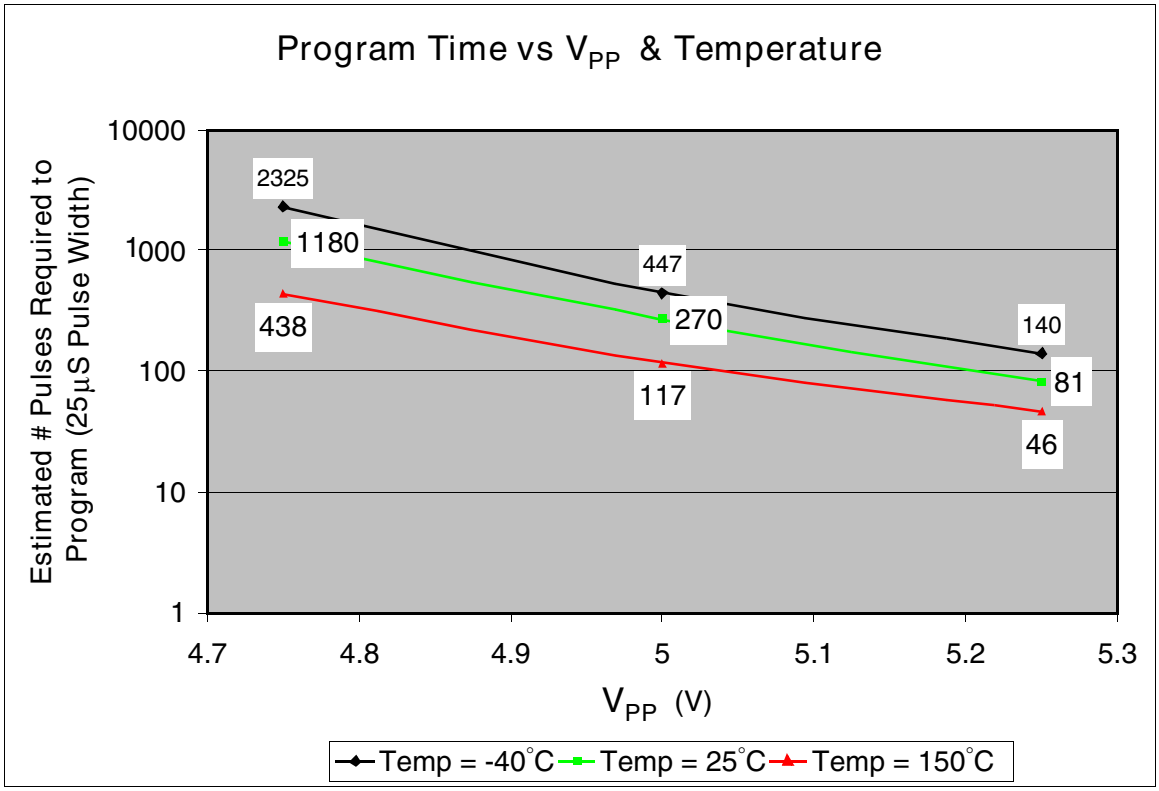


Figure H-1 Typical Program Time vs. V_{PP} and Temperature (for CDR1 "Target" Process)

Table H-2 CMF AC and DC Power Supply Characteristics

Symbol	Meaning	Min. Value	Max Value	Unit
V_{DDF}	Operating Voltage Read, Program or Erase	3.0	3.6 V	V
I_{DDF}	Operating Current at 40.0MHz, V_{DDF} = 3.3 V for a 256K-byte Module Read, Program or Erase Operation Disabled	 — —	 22 5	mA
V_{PP}	External Program or Erase voltage Read Program or Erase	 V_{DDF} -0.35 4.75	 5.5 5.25	V
I_{DDPP}	External Program and Erase Current[1] Read, V_{PP} = 5 V Program, V_{PP} = 5.25 V Erase, V_{PP} = 5.25 V		 <100[1] 30[1] 30[1]	 µA[1] mA[1] mA[1]

NOTES:
1. Average current is less than 30 mA when programming both modules simultaneously.

H.1.1 Flash Module Life

Table H-3 Flash Module Life

Symbol	Meaning	Value
P/E Cycles[1]	Maximum Number of Program/ Erase cycles[2] to Guarantee Data Retention	100[3,4]
Retention	Data Retention at Average Operating Temperature of 85 °C	Minimum 10 years

NOTES:
1. Target failure rate at specified number of program/erase cycles of 2ppm pending characterization of production silicon.
2. A program/erase cycle is defined as switching the bits from 1 → 0 → 1.
3. Reprogramming of a CMF array block prior to erase is not required.
4. Number of program/erase cycles to be adjusted pending characterization of production silicon.

H.2 Programming and Erase Algorithm

Table H-4 CMF Programming Algorithm (v5)

No. of Pulses (Maximum)	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
3500	25.6 µs	X	000	X	Normal	

Table H-5 CMF Erase Algorithm (v5)

No. of Pulses (Maximum)	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
10	1 s	X	111	X	Mode 7P	Positive drain ramp

INDEX

–A–

A(0
    31), 9-4
ACKERR 16-31
Acknowledge error (ACKERR) 16-31
Address
    -mark wakeup 14-58
    space 13-7
address type (AT0-AT3), 9-37
ALE 21-54
ALEE 21-56
alignment exception, 3-46
ALU–BFU 3-5
AN 13-3, 13-5
Analog
    front-end multiplexer 13-14
    input
        multiplexed 13-5
        port A 13-3
        port B 13-4
    section contents 13-1
    submodule block diagram 13-12
    supply pins 13-5
arbitration, 9-30
AT(0
    3), 9-4
atomic update primitives, 3-42
atomic, 9-31

–B–

BAR 3-52
Base ID mask bits 16-30
Baud
    clock 14-51
BB, 9-7
BDIP, 9-5
BE bit 3-21
Beginning of queue 2 (BQ2) 13-39
BG, 9-7
BI, 9-6
Binary
    divider 13-25
    -weighted capacitors 13-14
Bit stuff error (STUFFERR) 16-31
BITERR 16-31
BITS 14-17
Bits per transfer
    enable (BITSE) 14-23
    field (BITS) 14-17
BITSE 14-23, 14-38
Bit-time 14-50
BKPT (TPU asserted) 17-14
BLC 17-13
BOFFINT 16-31
Boundary conditions 13-17
Boundary scan
    cells 22-1
    descriptor language 22-7
    register 22-1
BPU 3-5
BQ2 13-17, 13-39
BR, 9-7
Branch
    prediction 3-5
    processing unit 3-5
    trace enable 3-21
Branch latch control (BLC) 17-13
Branch processing unit 3-5
Break frame 14-51
Breakpoint
    asserted flag (BKPT) 17-14
    flag (PCBK) 17-14
Breakpoint counter A value and control register 21-52
Breakpoint counter B value and control register 21-53
BRKNOMSK 21-51
BSC 22-1
BSR 22-1
burst indicator (BURST), 9-36
burst inhibit (BI), 9-39
burst read cycle (illustration), 9-18
burst transfer, 9-15
burst write cycle (illustration), 9-23
BURST, 9-4
Bus
    monitor 6-12
    off interrupt
        (BOFFINT) 16-31
bus busy (BB), 9-32
bus exception control cycles, 9-43
bus grant (BG), 9-32
bus interface
    bus control signals, 9-2
    bus operation
        address transfer phase related signals, 9-35
        arbitration phase, 9-30
        basic transfer protocol, 9-8
        burst mechanism, 9-16
        burst transfer, 9-15
        bus exception control cycles, 9-43
        single beat transfer
            single beat read flow, 9-8

single beat write flow, 9-8, 9-11
single beat transfer, 9-8
storage reservation, 9-40
termination signals, 9-38
bus operations, 9-7
bus transfer signals, 9-1
features, 9-1
signal descriptions, 9-3
bus request (BR), 9-31
bus signals (illustration), 9-3
BUSY 16-5, 16-15
BYP 13-13, 13-47
Bypass mode 13-13
BYTES field 3-18

–C–

C bit 3-14
CA bit 3-18
cache control instructions, 3-42
CAN2.0B
system 16-3
CANCTRL0 16-25
CANCTRL1 16-26
CANCTRL2 16-28
CANICR 16-24
Carry 3-18
CCL 17-13
CCW 13-1, 13-46
CF1 13-41
CF2 13-41
CFSR 17-15
CGBMSK 21-50
CH 17-15, 17-18, 17-19
CHAN 13-47
CHANNEL 17-16
Channel
assignments
multiplexed 13-48
nonmultiplexed 13-48
conditions latch (CCL) 17-13
interrupt
enable
/disable field (CH) 17-15
request level (CIRL) 17-15
status (CH) 17-19
invalid 13-47
number (CHAN) 13-47
orthogonality 17-3
priority registers 17-18
register breakpoint flag (CHBK) 17-14
reserved 13-47
CHBK 17-14
CHBMSK 21-50
checkstop state, 3-45
CHSTP bit 21-54
CHSTPE 21-55
CIE1 13-37
CIE2 13-39
CIER 17-15, 17-19
CIRL 17-15
CISR 17-8, 17-19
class, instruction, 3-38
CLKOUT to TA, BI assertion (when driven by the Memory Controller) G-18
CLKOUT, 9-7
CLKS 17-13
Clock
block diagram 13-26
frequency 13-26
generation 13-24
phase (CPHA) 14-17
polarity (CPOL) 14-17
CMPA–CMPD 21-45
CMPE–CMPF 21-46
CMPG–CMPH 21-46
CNRX/TX pins 16-2
CNTC 21-52
CNTV 21-52
Code 16-5
Coherency 13-6, 13-22, 17-4
COMM D-18
Command
RAM 14-22
word pointer (CWP) 13-42
Comparator 13-14
Comparator A–D value registers 21-45
Comparator E–F value registers 21-46
Comparator G–H value registers 21-46
Compare instructions 3-17
Compare size 21-50
Compare type 21-48, 21-50
Completed queue pointer (CPTQP) 14-21
Condition register 3-15, 3-17
CONT 14-23
contention, 9-36
Continue (CONT) 14-23
Continuous transfer mode 14-15
Control registers
1 (QACR1) 13-36
2 (QACR2) 13-38
controlling termination of a bus cycle for a bus error, 9-43
Conversion
command word table (CCW) 13-1, 13-14
cycle times 13-12
stages 13-45
Count register 3-19
COUNTA 21-52
COUNTB 21-53
CPHA 14-17, 14-34
CPOL 14-17, 14-34
CPR 17-18
CPTQP 14-21, 14-24
CR 3-5, 3-15, 3-19
and compare instructions 3-17
CR, 9-5
CR0 field 3-16
CR1 field 3-16
CRCERR 16-31
CRWE 21-50
CRWF 21-50

CSG 21-50
CSH 21-50
CTA 21-48
CTB 21-48
CTC 21-48
CTD 21-48
CTE 21-50
CTF 21-50
CTG 21-50
CTH 21-50
CTR 3-5
CWP 13-42
Cyclic redundancy check error (CRCERR) 16-31

–D–

D(0
    31), 9-6
D0 20-3
DAC 13-1
DAE/source instruction service register 3-22
DAR 3-22, 3-46, 3-51, 3-52
DAR, 3-46, 3-51
Data
    field for RX/TX frames (TOUCAN) 16-5
    frame 14-50
Data address register 3-22
Data space only 20-3
data storage interrupt, 3-45
DCNR 17-19
DDRQA 13-33, 13-35
DDRQS 14-9, 14-33, 14-37
Debug enable register 21-55
debug mode disable, 3-45
DEC 3-23
DECE 21-54
DECEE 21-56
Decrementer
    register 3-23
Delay
    after transfer (DT) 14-23, 14-35
    before SCK (DSCKL) 14-18
DER 21-55
Development Port
    trap enable selection 21-48
Digital
    control section
        contents 13-1, 13-14–13-29
    input
        /output port (PQA) 13-3
        port (PQB) 13-4
    to analog converter (DAC) 13-14
DIO D-38
DIS 20-3
Disable TPU2 pins field (DTPU) 17-20
Disabled mode 13-18
Discrete input/output (DIO) D-38
DIV2 17-20
DIV8 clock 17-7
Divide by two control field (DIV2) 17-20
DIW0EN 21-48
DIW1EN 21-48
DIW2EN 21-48
DIW3EN 21-48
DLW0EN 21-52
DLW1EN 21-52
Double
    -buffered 14-52, 14-56
DPI 21-55
DPTRAM 18-4
DSCK 14-23
DSCKL 14-18
DSCR 17-12
DSISR 3-22, 3-46, 3-51, 3-52
DSSR 17-14
DT 14-23
DTL 14-18
DTPU 17-20

–E–

EA 3-33
EBRK 21-55
ECR 21-53
EE bit 3-21, 3-26
Effective address 3-33
EID 3-26
EIE 3-26
eieio, 3-43
ELE bit 3-21
EMPTY 16-5
EMU 17-4, 17-11
Emulation
    control (EMU) 17-11
    support 17-4
Encoded
    one of three channel priority levels (CH) 17-18
    time function for each channel (CHANNEL) 17-16
    type of host service (CH) 17-18
Ending queue pointer (ENDQP) 14-19
End-of-
    frame (EOF) 16-16
End-of-queue condition 13-45
ENDQP 14-19, 14-24
Entry
    table bank select field (ETBANK) 17-20
EOF 16-16
EOQ 13-17
EP bit 3-21
ERRINT 16-32
Error
    conditions 14-56
    counters 16-9
    interrupt (ERRINT) 16-32
ESTAT 16-30
ETBANK 17-20
ETRIG 13-4
Event timing 17-3
Exception cause register 21-53
Exception prefix 3-21
Exceptions 3-34
    classes 3-34

little endian mode 3-21
ordered 3-34
precise 3-35
unordered 3-34
vector table 3-35, 3-36
Execution units 3-4
Extended message format 16-1
frames 16-5
External
digital supply pin 13-5
interrupt
disable 3-26
enable 3-26
multiplexing 13-9
trigger pins 13-4
trigger single-scan mode 13-20
External interrupt
enable 3-21, 3-26
Externally
multiplexed mode (MUX) 13-36
EXTEST 22-5
EXTI 21-54
EXTIE 21-56

–F–

Fast quadrature decode TPU function (FQD) D-29
Fault confinement state (FCS) 16-10, 16-31
FCS 16-10, 16-31
FE 14-49, 14-56
FE bits 3-21, 3-22
FE flag 3-14
features
bus interface, 9-1
Fetch serialized 21-1
FEX bit 3-14
FG bit 3-14
FI bit 3-14
Final sample time 13-12
FL bit 3-14
Floating-point
available 3-21
condition code 3-14
enabled exception summary 3-14
equal or zero 3-14
exception mode 3-21, 3-22
exception summary 3-14
fraction inexact 3-14
fraction rounded 3-14
greater than or positive 3-14
inexact exception 3-14
enable 3-15
invalid operation exception
enable 3-15
for ×*0 3-14
for ×/× 3-14
for ×-× 3-14
for 0/0 3-14
for invalid compare 3-14
for invalid integer convert 3-15
for invalid square root 3-15
for SNaN 3-14
for software request 3-15
summary 3-14
less than or negative 3-14
overflow exception 3-14
enable 3-15
registers 3-12
result class descriptor 3-14
result flags 3-14
rounding control 3-15
status and control register 3-12
underflow exception 3-14
unit 3-5
unordered or NaN 3-14
zero divide exception 3-14
enable 3-15
floating-point unavailable interrupt, 3-47
FORMERR 16-31
FP bit 3-21
FPCC bit 3-14
FPRF field 3-14
FPRs 3-12
FPSCK 17-20
FPSCR 3-12
FPU 3-5
FPUVE 21-54
FPUVEE 21-56
F_{QCLK} 13-25
FQD D-29
FQM D-11
FR 3-14
Frame 14-50
size 14-57
Frames
overload 16-16
remote 16-15
Framing error (FE) flag 14-49, 14-56
FREEZ ACK 16-17
FREEZE
assertion response (FRZ)
QADC 13-6, 13-33
QSM 14-5
TPU 17-13
FREEZE (internal signal) 13-46
Frequency measurement (FQM) D-11
FRZ 13-7, 13-33, 16-11, 17-13
FRZACK 16-11
FU bit 3-14
FULL 16-5
Function
library for TPU 17-4
FX bit 3-14

–G–

General SPRs 3-25
General-purpose registers (GPRs) 3-12
Global registers 13-32

–H–

Hall effect decode (HALLD) D-20
HALLD D-20
HALT 14-20, 16-11
Halt
    acknowledge flag (HALTA) 14-21
    QSPI (HALT) 14-20
HALTA 14-21
HALTA and MODF Interrupt Enable (HMIE) 14-41
HALTA/MODF interrupt enable (HMIE) bit 14-20
Hang on T4 (HOT4) 17-13
HMIE 14-20
HOT4 17-13
HSQR 17-16
HSSR 17-17

–I–

I/O port operation 13-7
IBRK 21-55
I-bus
    watchpoint programming 21-48
I-bus support
    control register 21-47
ICTRL 21-47
ID
    Extended (IDE) field 16-6
    HIGH field 16-6
    LOW field 16-6
IDE 16-6
Identifier (ID) 16-1
    bit field 16-6
IDLE 14-48, 14-57, 16-31
Idle
    CAN status (IDLE) 16-31
    frame 14-50
    -line
        detect type (ILT) 14-46
        detected (IDLE) 14-48, 14-57
        detection process 14-57
        interrupt enable (ILIE) 14-46, 14-57
        type (ILT) bit 14-57
IEEE 1149.1-1990 standard. See JTAG
IFLAG 16-33
Ignore first match 21-48
IIFM 21-48
ILIE 14-46, 14-57
illegal and reserved instructions, 3-38
ILSCI 14-8, 14-9
ILT 14-46, 14-57
IMASK 16-32
IMB 13-1, 13-24
implementation dependent software emulation interrupt, 3-48
implementation specific data TLB error interrupt, 3-50
implementation specific debug interrupt, 3-51
implementation specific instruction TLB error interrupt, 3-49
IMUL–IDIV 3-5
Information processing time (IPT) 16-9
Initial sample time 13-12
Input
    sample time (IST) 13-28, 13-47
Instruction
    pipeline 3-36
    sequencer 3-3
    set summary 3-29
    timing 3-36
Instruction fetch
    show cycle control 21-1
instruction storage interrupt, 3-46
instructions
    cache control, 3-42
    storage control, 3-44
instructions, partially executed, 3-52
Integer exception register 3-17
Integer unit 3-5
Interchannel communication 17-4
Intermission 16-16
Intermodule bus (IMB) 13-1
Interrupt
    sources 13-30
Interrupt Level of SCI (ILSCI) 14-8, 14-9
Interrupts
    QADC 13-29
    TOUCAN 16-19
    TPU 17-5
interrupts, 3-44
Inter-transfer delay 14-14
invalid and preferred instructions, 3-39
Invalid channel number 13-47
IPT 16-9
IRQ 17-5
ISCTL 21-1
IST 13-28, 13-47
isync, 3-42
IU 3-5
IW 21-48

–J–

Joint test action group. See JTAG
JTAG 22-1
    instruction register 22-4
    non-IEEE 1149.1-1990 operation 22-7
    signals 22-1, 22-2

–K–

KR/RETRY, 9-5

–L–

LBRK 21-55
LBUF 16-27
L-bus support
    control register 1 21-49
    control register 2 21-50
LCK 20-3
LCTRL1 21-49
LCTRL2 21-50

LE bit 3-22
Least significant bit (LSB) 13-14
Left justified
signed result word table (LJSRR) 13-49
unsigned result word table (LJURR) 13-49
Length of delay after transfer (DTL) 14-18
Link register 3-18
Little endian mode 3-22
LJSRR 13-49
LJURR 13-49
Load/store unit 3-5, 3-6
Lock
/release/busy mechanism 16-15
Loop
mode
(LOOPS) 14-46
LOOPQ 14-20
LOOPS 14-46
Low power stop (LPSTOP)
QADC 13-6
QSM 14-5
Lowest buffer transmitted first (LBUF) 16-27
Low-power
stop mode enable (STOP)
QADC 13-33
TPU 17-11
LR 3-5, 3-18, 17-19
LSB 13-14
LSU 3-5, 3-6
LW0EN 21-51
LW0IA 21-51
LW0IADC 21-51
LW0LA 21-51
LW0LADC 21-51
LW0LD 21-51
LW0LDDC 21-51
LW1EN 21-51
LW1IA 21-51
LW1IADC 21-51
LW1LA 21-51
LW1LADC 21-51
LW1LD 21-51
LW1LDDC 21-51

–M–

M 14-46, 14-51
Machine
check enable 3-21
state register 3-20
status save/restore register 0 3-24
status save/restore register 1 3-24
machine check interrupt, 3-45
Machine status save/restore register 1 3-24
Mask
examples for normal/extended messages 16-8
registers (RX) 16-7
Master
/slave mode select (MSTR) 14-17
master
external
arbitration phase, 9-30
MCE 21-54
MCEE 21-55
MCPWM D-22
ME bit 3-21
Message
buffer
address map 16-22
code for RX/TX buffers 16-5
deactivation 16-13
structure 16-3
format error (FORMERR) 16-31
Mid-analog supply voltage 13-14
MISO 14-33, 14-37
Mode
fault flag (MODF) 14-21, 14-26
select (M) 14-46
Mode Fault Flag (MODF) 14-41
Modes
disabled 13-18
reserved 13-18
scan. See Scan modes
MODF 14-21, 14-26, 14-41
Modulus
counter 14-51
MOSI 14-33, 14-37
Most significant bit (MSB) 13-14
MQ1 13-37
MQ2 13-39
MSB 13-14
MSR 3-20, 3-45, 3-47, 3-48, 3-49, 3-50, 3-51, 3-52
MSR, 3-44
MSTR 14-17
Multichannel pulse-width modulation (MCPWM) D-22
parameters
slave channel A
non-inverted center aligned mode D-25, D-41
slave edge-aligned mode D-24
Multimaster operation 14-26
Multiphase motor commutation (COMM) D-18
parameters D-19
Multiple end-of-queue 13-17
Multiplexed analog inputs 13-5
MUX 13-8, 13-36

–N–

New
queue pointer value (NEWQP) 14-19
New input capture/transistion counter (NITC) D-16
parameters D-17
NEWQP 14-19, 14-24
NF 14-49, 14-56
NI bit 3-15
NITC D-16
Noise
error flag (NF) 14-49
errors 14-56
flag (NF) 14-56
Non-IEEE 1149.1-1990 operation 22-7

Non-IEEE floating-point operation 3-15
nonoptional instructions, 3-38
Non-recoverable interrupt 3-26
NOT ACTIVE 16-5
Not ready (NOTRDY) 16-20
NOTRDY 16-16, 16-20
NRI 3-26

–O–

OC D-34
OE bit 3-15
OP0, 9-28
OP1, 9-28
OP2, 9-28
OP3, 9-28
operand placement (effects), 3-42
operand representation (illustration), 9-28
optional instructions, 3-38
OR 14-48
Ordered exceptions 3-34
Output compare (OC) D-34
OV (overflow) bit 3-18
Overload frames 16-16
OVERRUN 16-5
Overrun error (OR) 14-48
OX bit 3-14

–P–

P 13-47
Parity
    (PF) flag 14-56
    checking 14-52
    enable (PE) 14-46
    error (PF) bit 14-49
    errors 14-56
    type (PT) 14-46
    type (PT) bit 14-52
Pause (P) 13-15, 13-47
PCBK 17-14
PCS 14-23
    to SCK delay (DSCK) 14-23
PCS0/SS 14-38
PCS3-PCS0/SS 14-41
PE 14-46
Period
    /pulse-width accumulator (PPWA) D-32
Periodic
    /interval timer 13-29
Periodic interrupt
    timer 6-14
Peripheral
    chip-selects (PCS) 14-23, 14-36
Peripheral Chip-Select 3-0/Slave Select (PCS3-PCSO/SS) 14-41
PF 14-49, 14-56
PF1 13-41
PF2 13-41
Phase buffer segment 1/2 (PSEG1/2) bit field 16-28
phase-lock loop, 9-7
PIE1 13-37
PIE2 13-39
PIT 6-14
PLL, 9-7
Pointer 14-15
port size device interfaces (illustration), 9-29
port width, 9-1
PORTQA 13-33, 13-34
PORTQB 13-33, 13-34
PORTQS 14-10
PowerPC standards
    PowerPC Operating Environment Architecture (Book 3)
        branch processor, 3-43
        fixed-point processor
            special purpose registers, 3-43
        fixed-point processor, 3-43
        interrupts, 3-44
        optional facilities and instructions, 3-53
        storage control instructions, 3-44
        timer facilities, 3-53
    PowerPC Operating Environment Architecture (Book 3), 3-43
    PowerPC User Instruction Set Architecture (Book 1)
        branch instructions, 3-39
        branch processor, 3-39
        computation modes, 3-38
        exceptions, 3-39
        fixed point-processor, 3-39
        instruction classes, 3-38
        load/store processor, 3-40
        reserved fields, 3-38
    PowerPC User Instruction Set Architecture (Book 1), 3-38
    PowerPC Virtual Environment Architecture (Book 2)
        operand placement effects, 3-42
        storage control instructions, 3-42
        timebase, 3-43
    PowerPC Virtual Environment Architecture (Book 2), 3-42
PowerPC User Instruction Set Architecture
    Book 1
        instruction fetching, 3-39
PPWA D-32
PQA 13-8
PQB 13-4, 13-8
PQSPAR 14-9, 14-33, 14-37
PR bit 3-7, 3-21
PRE 21-54
Precise exceptions 3-35
PREE 21-56
Prescaler 13-26
    clock
        (PSCK) 17-11
        high time (PSH) 13-36
        low time (PSL) 13-36
    clock high time (PSH) 13-26
    control
        for TCR1 17-5
        for TCR2 17-7

divide
factor field 16-28
register (PRESDIV) 16-8, 16-27
PRESDIV (bit field) 16-28
PRESDIV (register) 16-8, 16-9, 16-27
Privilege level 3-7, 3-21
Processor version register 3-25
Programmable
channel service priority 17-4
transfer length 14-14
Programmable time accumulator (PTA) D-4
parameters D-5, D-44, D-46
Propagation segment time (PROPSEG) 16-27
PROPSEG 16-11, 16-27
PSCK 17-11
PSEG1 16-28
PSEG2 16-9, 16-11, 16-28
PSEGS1 16-11
PSH 13-26, 13-36
PSL 13-36
PT 14-46, 14-52
PTA D-4
PTR, 9-4, 9-37
Pulse-width modulation (PWM) D-36
parameters D-37
PVR 3-25
PWM D-36

–Q–

QACR0 13-35
QACR1 13-36
QACR2 13-38, 13-39
QADC
features 13-2
pin functions diagram 13-2
registers
control register 0 (QACR0) 13-35
control register 1 (QACR1) 13-36
control register 2 (QACR2) 13-39
conversion command word table (CCW) 13-46
interrupt register (QADCINT) 13-32, 13-33
module configuration register (QADCMCR) 13-6, 13-32, 13-33
port
A data register (PORTQA) 13-33
B data register (PORTQB) 13-33
data direction register (DDRQA) 13-33
QA data direction register (DDRQA) 13-35
QA data register (PORTQA) 13-34
QB data register (PORTQB) 13-34
status register (QASR) 13-41, 13-42
test register (QADCTEST) 13-32, 13-33
QADCINT 13-32, 13-33
QADCMCR 13-6, 13-32, 13-33
QADCTEST 13-32, 13-33
QASR 13-40
QASR0 13-41
QASR1 13-42
QCLK 13-21, 13-24
frequency 13-25
QDDR 14-12, 14-41
QILR 14-8
QOM D-6
QPAR 14-11
QPDR 14-10, 14-41
QS 13-42
QSM
pin function 14-10
QSPI 14-13
operating modes 14-26
operation 14-24
RAM 14-21
registers
pin control registers 14-9
port QS
data direction register (DDRQS) 14-9
data register (PORTQS) 14-10
QSPI
control register 0 (SPCR0) 14-16
control register 1 (SPCR1) 14-18
control register 2 (SPCR2) 14-18
control register 3 (SPCR3) 14-19
status register (SPSR) 14-19
SCI
control register 0 (SCCR0) 14-45
control register 1 (SCCR1) 14-45
data register (SCDR) 14-49
status register (SCSR) 14-47
SCI 14-41
operation 14-50
pins 14-50
registers 14-44
QSM Data Direction Register (QDDR) 14-12, 14-41
QSM Interrupt Level Register (QILR) 14-8
QSM Pin Assignment Register (QPAR) 14-11
QSM Port Data Register (QPDR) 14-10, 14-41
QSPI 14-13
block diagram 14-14
enable (SPE) 14-18
finished flag (SPIF) 14-21
initialization operation 14-27
loop mode (LOOPQ) 14-20
master operation flow 14-28
operating modes 14-26
master mode 14-26, 14-33
wraparound mode 14-37
slave mode 14-26, 14-37
operation 14-24
peripheral chip-selects 14-36
RAM 14-21, 14-22
command RAM 14-22
receive RAM 14-22
transmit RAM 14-22
QSPI Enable (SPE) 14-41
QSPI Status Register (SPSR) 14-41
Queue 13-14
pointers
completed queue pointer (CPTQP) 14-24
end queue pointer (ENDQP) 14-24
new queue pointer (NEWQP) 14-24

status (QS) 13-42
Queue 1
completion
flag (CF1) 13-41
interrupt enable (CIE1) 13-37
operating mode (MQ1) 13-37
pause
flag (PF1) 13-41
interrupt enable (PIE1) 13-37
single-scan enable (SSE1) 13-37
trigger overrun (TOR1) 13-41
Queue 2
completion
flag (CF2) 13-41
interrupt enable (CIE2) 13-39
operating mode (MQ2) 13-39
pause
flag (PF2) 13-41
interrupt enable (PIE2) 13-39
resume (RES) 13-39
single-scan enable bit (SSE2) 13-39
trigger overrun (TOR2) 13-41
Queued
analog-to-digital converter. See QADC 13-1
serial
peripheral interface (QSPI) 14-13
Queued output match TPU function (QOM) D-6

–R–

R0 20-3
RAF 14-48
RD/WR, 9-4
RDRF 14-48, 14-56
RE 14-44, 14-46, 14-56
RE bit 3-22, 3-26
read cycle, data bus requirements, 9-30
Read only, SRAM 20-3
read/write (RD/WR), 9-36
Receive
data
register full (RDRF) 14-48
error status flag (RXWARN) 16-31
RAM 14-22
time sample clock (RT) 14-52, 14-56
Receiver
active (RAF) 14-48
data register (RDRF) flag 14-56
enable (RE) 14-46, 14-56
interrupt enable (RIE) 14-46
wakeup (RWU) 14-47, 14-58
Receiver Enable (RE) 14-44
Reception of transmitted frames 16-13
Recoverable exception 3-22, 3-26
Registers
CMPA–CMPD 21-45
CMPE–CMPF 21-46
CMPG–CMPH 21-46
COUNTA 21-52
COUNTB 21-53
DER 21-55
ECR 21-53
ICTRL 21-47
LCTRL1 21-49
LCTRL2 21-50
MI_RA 0-3 region attribute register 4-21
supervisor level 3-20
test (RAMTST) 18-4
user level 3-11
registers
special purpose
added registers, 3-44
unsupported registers, 3-43
special purpose, 3-43
Remote
frames 16-15
transmission request (RTR) 16-5, 16-6
RES 13-39
reservation protocol for a multi-level (local) bus, 9-41
Reserved
channel number 13-47
mode 13-18
Reset
status register 7-5
Resistor-divider chain 13-14
Resolution time 13-12
Result word table 13-1, 13-15, 13-49
Resynchronization jump width (RJW) bit field 16-28
RETRY, 9-43
RIE 14-46
Right justified, unsigned result word table (RJURR) 13-49
RJURR 13-49
RJW 16-11, 16-28
RN field 3-15
RSR 7-5
RSV, 9-37
RT 14-56
RTR 16-5, 16-6, 16-15
RWU 14-47, 14-58
RX
Length 16-5
RX14MSKHI 16-30
RX14MSKLO 16-30
RX15MSKHI 16-30
RX15MSKLO 16-30
RXECTR 16-33
RXGMSKHI 16-29
RXGMSKLO 16-29
RXWARN 16-31

–S–

S0 14-8, 20-3
SAMP 16-27
Sample amplifier bypass (BYP) 13-47
Sampling mode (SAMP) 16-27
SAR 13-14
SBK 14-47, 14-53
Scan modes
single-scan modes
external trigger 13-20
SCBR 14-45

SCCR0 14-45
SCCR1 14-45
SCDR 14-49
SCI 14-33, 14-41
baud
clock 14-51
rate (SCBR) 14-45
equation 14-45
idle-line detection 14-57
internal loop 14-58
operation 14-50
parity checking 14-52
pins 14-50
receiver
block diagram 14-43
operation 14-56
wakeup 14-58
registers 14-44
SCCR0 14-44
SCCR1 14-44
SCI Baud Rates 14-52
SCI SUBMODULE 14-12
SCSR 14-44
transmitter
block diagram 14-42
operation 14-52
SCI Control Register 0 (SCCR0) 14-44
SCI Control Register 1 (SCCR1) 14-44
SCI Status Register (SCSR) 14-44
SCK 14-11, 14-32, 14-37
actual delay before SCK (equation) 14-35
baud rate (equation) 14-34
S-clock 16-9
SCSR 14-47
SE bit 3-21
SEE 21-54
Send break (SBK) 14-47, 14-53
Sequencer, instruction 3-3
Serial
clock baud rate (SPBR) 14-17
communication interface (SCI) 14-41
formats 14-51
mode (M) bit 14-51
shifter 14-52
Serial Clock (SCK) 14-11
Serialization
fetch 21-1
Service
request breakpoint flag (SRBK) 17-14
SGLR 17-19
Simplified mnemonics 3-33
Single-step trace enable 3-21
SIU
module configuration register 6-18
SIU signals, 9-4
SIUMCR 6-18
SIW0EN 21-48
SIW1EN 21-48
SIW2EN 21-48
SIW3EN 21-48
Slave Select (SS) 14-41
Slave select signal (SS) 14-37, 14-38
SLW0EN 21-52
SLW1EN 21-52
snooping external bus activity, 3-42
SO bit 3-18
SOF 16-9
Soft reset control field (SOFT_RST) 17-20
SOFT_RST 17-20
SOFTRST 16-11
Software trap enable selection 21-48
SPBR 14-17
SPCR0 14-16
SPCR1 14-18
SPCR2 14-18
SPCR3 14-19
SPE 14-18, 14-41
Special-purpose registers, general 3-25
SPI
finished interrupt enable (SPIFIE) 14-19
TSBD 14-8
SPI Test Scan Path Select (TSBD) 14-8
SPIF 14-21
SPIFIE 14-19
SPRG0–SPRG3 3-25
SPRGs 3-25
SPRs
general 3-25
SPSR 14-19, 14-41
SPWM D-40
SRAM
data space only 20-3
disabling 20-3
locking 20-3
read only 20-3
registers 20-2
supervisor space only 14-8, 20-3
two-cycle mode 20-3
SRAMMCR 20-3
SRBK 17-14
SRR 16-6
SRR0 3-24, 3-44, 3-45, 3-47, 3-48, 3-49, 3-50, 3-51, 3-52
SRR1 3-24, 3-44, 3-45, 3-47, 3-48, 3-49, 3-50, 3-51, 3-52
SS 14-41
SS 14-37, 14-38
SSE1 13-37
SSE2 13-39
Standard
message format 16-1
frames 16-5
Start
bit (beginning of data frame) 14-50
-of-frame (SOF) symbol 16-9
State machine 13-25, 14-56
Status register (QASR) 13-40
STF 17-11
STOP 13-33, 16-17, 17-11
Stop
clocks to TCRs (CLKS) 17-13
enable (STOP) bit

QADC 13-6
QSM 14-5
TOUCAN 16-17
TPU 17-11
flag (STF) 17-11
SCI end of data frame bit 14-50
storage control instructions, 3-44
storage reservation, 9-40
STS, 9-5
STUFFERR 16-31
Subqueue 13-15
Substitute remote request (SRR) 16-6
Successive approximation register (SAR) 13-14
Summary overflow 3-18
Supervisor
/unrestricted data space (SUPV)
QADC 13-33
TPU 17-11
Supervisor mode 3-20
and SRAM 14-8, 20-3
SUPV 13-7, 13-33
SUSG 21-50
SUSH 21-50
Synchronized pulse-width modulation (SPWM) D-40
SYSE 21-54
SYSEE 21-56
system clock output, 9-7
system reset interrupt, 3-44

–T–

T2CFILTER 17-20
T2CG 17-7, 17-11
T2CLK pin filter control (T2CFILTER) 17-20
T2CSL 17-11
TA, 9-6
Table stepper motor (TSM) D-8
TAP controller 22-3
TB 3-19
TBL 3-19, 3-23
TBU 3-19, 3-23
TC 14-48, 14-53
TCIE 14-46, 14-54
TCK 22-3
TCNMCR 16-22
TCR 17-12
TCR1P 17-11
TCR2 clock/gate control (T2CG) 17-11
TDI 22-3
TDO 22-3
TDRE 14-48
TE 14-44, 14-46
TEA, 9-6
termination signals, 9-38
Test
access port controller. See TAP controller
clock 22-3
data input 22-3
data output 22-3
mode select 22-3
reset 22-3
test register (RAMTST) 18-4
TICR 17-14, 17-21
TIE 14-46, 14-54
Time
quanta clock 16-9
stamp 16-5, 16-10
Time base 3-19
timebase, 3-43
TIMER 16-29
Timer
count register
1 prescaler control (TCR1P) 17-11
synchronize mode (TSYNC) 16-27
Timing, instruction 3-36
TMS 22-3
TOR1 13-41
TOR2 13-41
TOUCAN
address
map 16-20
bit timing configuration 16-8
operation 16-9
external pins 16-2
initialization sequence 16-11
interrupts 16-19
message buffer address map 16-22
operation 16-3
receive process 16-13
registers
control register 0 (CANCTRL0) 16-25
control register 1 (CTRL1) 16-8
control register 1(CANCTRL1) 16-26
control register 2 (CANCTRL2) 16-28
control register 2 (CTRL2) 16-8
error and status register (ESTAT) 16-30
free running timer register (TIMER) 16-29
interrupt
configuration register (CANICR) 16-24
flag register (IFLAG) 16-33
mask register (IMASK) 16-32
module configuration register (TCNMCR) 16-22
receive
buffer 14 mask registers (RX14MSKHI/LO) 16-30
buffer 15 mask registers (RX15MSKHI/LO) 16-30
global mask registers (RXGMSKLO/HI 16-29
RX/TX error counter registers (RXECTR/TXECTR) 16-33
test configuration register (CANTCR) 16-24
special operating modes 16-16
auto power save mode 16-18
debug mode 16-16
low-power stop mode 16-17
transmit process 16-12
TPU
address map 17-8
components 17-2
FREEZE flag (TPUF) 17-14

function library 17-4
host interface 17-2
interrupts 17-5
microengine 17-2
operation 17-3
coherency 17-4
emulation support 17-4
event timing 17-3
interchannel communication 17-4
programmable channel service priority 17-4
parameter RAM 17-2, 17-22
address map 17-22
registers
channel
function select registers (CFSR) 17-15
interrupt
enable register (CIER) 17-5, 17-15
status register (CISR) 17-5, 17-19
priority registers (CPR) 17-18
decoded channel number register (DCNR) 17-19
development
support control register (DSCR) 17-12
support status register (DSSR) 17-14
host
sequence registers (HSQR) 17-16
service request registers (HSSR) 17-17
link register (LR) 17-19
module configuration register (TPUMCR) 17-10
service grant latch register (SGLR) 17-19
test configuration register (TCR) 17-12
TPU interrupt configuration register (TICR) 17-14, 17-21
scheduler 17-2
time
bases 17-2
timer channels 17-2
TPU Reference Manual 17-3, 17-17
TPU2
module configuration register 2 (TPUMCR2) 17-20
TPUF 17-14
TPUMCR 17-10
TPUMCR2 17-20
TR 21-54
trace interrupt, 3-47
transaction (bus), 9-8
Transfer
length options 14-36
transfer acknowledge (TA), 9-38
transfer error acknowledge (TEA), 9-39
transfer size (TSIZ), 9-37
transfer start (TS), 9-36
transfers, alignment and packaging, 9-28
transfers, burst-inhibited, 9-16
transfers, termination signals, 9-39
Transmission
complete
(TC) flag 14-53
interrupt enable (TCIE) 14-54
Transmit
/receive status (TX/RX) 16-31
bit error (BITERR) 16-31
complete
bit (TC) 14-48
interrupt enable (TCIE) 14-46
data
register empty (TDRE) flag 14-48
error status flag (TXWARN) 16-31
interrupt enable (TIE) 14-46, 14-54
pin configuration control (TXMODE) 16-25
RAM 14-22
Transmitter Enable (TE) 14-44
Transmitter enable (TE) 14-46, 14-52
TRE 21-56
Trigger
event 13-44
TRST 22-3
TS, 9-5
TSIZ(0
1), 9-4
TSIZ0, 9-1
TSIZ1, 9-1
TSM D-8
T_{SR} 13-6
TSYNC 16-27
Two-cycle mode, SRAM 20-3
TX
Length 16-5
TX/RX 16-31
TXECTR 16-33
TXMODE 16-25
TXWARN 16-31

–U–

UART D-13
UISA register set 3-11
Universal asynchronous receiver/transmitter (UART) D-13
parameters
receiver parameters D-15
transmitter parameters D-14
Unordered exceptions 3-34
User level registers 3-11
Using the TPU Function Library and TPU Emulation Mode 17-4
UX bit 3-14

–V–

V_{DD} 13-5
V_{DDA} 13-5
$V_{DDA/2}$ 13-14
VE bit 3-15
Vector table, exception 3-36
Vector table, exceptions 3-35
V_{IH} 13-8
V_{IL} 13-8
Voltage
reference pins 13-5
V_{RH} 13-5, 13-14, 13-47
V_{RL} 13-5, 13-14, 13-47

V_{SS} 13-5
V_{SSA} 13-5
VX bit 3-14
VXCVI bit 3-15
VXIDI 3-14
VXIMZ bit 3-14
VXISI 3-14
VXSNAN 3-14
VXSOFT bit 3-15
VXSQRT bit 3-15
VXVC bit 3-14
VXZDZ bit 3-14

–W–

WAKE 14-46, 14-58
Wake interrupt (WAKEINT) 16-32
WAKEINT 16-17, 16-32
WAKEMSK 16-17
Wakeup
    address mark (WAKE) 14-46, 14-58
Wired-OR
    mode
        for QSPI pins (WOMQ) 14-17
        for SCI pins (WOMS) 14-46, 14-53
WOMQ 14-17
WOMS 14-46, 14-53
Wrap
    enable (WREN) 14-19
    to (WRTO) 14-19
Wraparound mode 14-15
    master 14-37
WREN 14-19
write cycle data bus contents, 9-30
WRTO 14-19

–X–

XE bit 3-15
XER 3-17
XX bit 3-14

–Z–

ZE bit 3-15
ZX bit 3-14

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters can and do vary in different applications. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola is a registered trademark of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

To obtain additional product information:

USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

MFAX: RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609

INTERNET: http://www.mot.com

MOTOROLA