National Semiconductor

December 2001

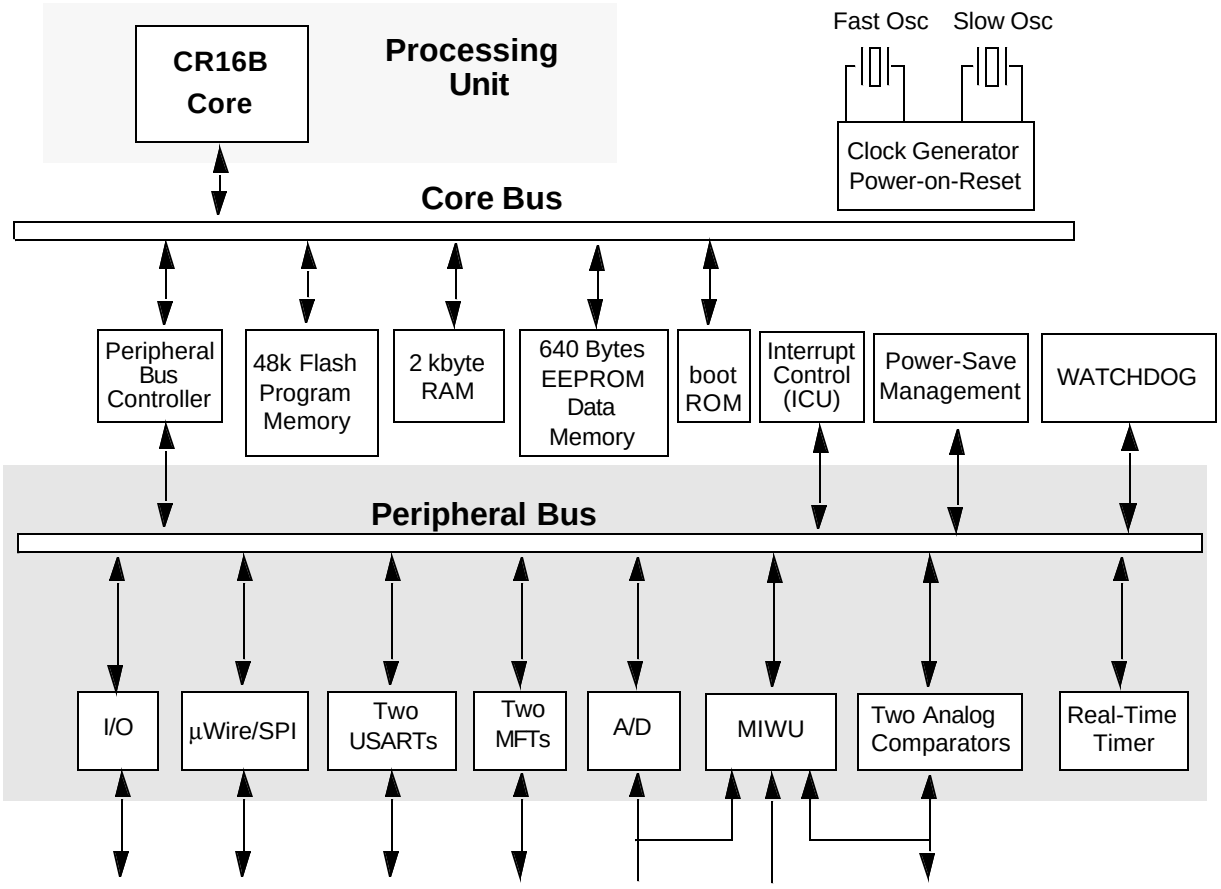
CR16MES5/CR16MES9/CR16MFS5/CR16MFS9/ CR16MHS5/CR16MHS9/CR16MNS5/CR16MNS9/ CR16M9S5/CR16MUS5/CR16MUS9/ Family of CompactRISC 16-Bit Microcontrollers

1.0 General Description

The family of CompactRISC™ microcontrollers are general-purpose 16-bit microcontrollers based on a Reduced Instruction Set Computer (RISC) architecture. The device operates as a complete microcomputer with all system timing, interrupt logic, flash program memory or ROM memory, RAM, EEPROM data memory, and I/O ports included on-chip. It is ideally suited to a wide range of embedded controller applications because of its high performance, on-chip integrated features and low power consumption, resulting in decreased system cost.

The family of CompactRISC 16-bit microcontrollers offer the high performance of a RISC architecture while retaining the advantages of a traditional Complex Instruction Set Computer (CISC): compact code, on-chip memory and I/O, and reduced cost. The CPU uses a three-stage instruction pipeline that allows execution of up to one instruction per clock cycle, or up to 20 million instructions per second (MIPS) at a clock rate of 20 MHz.

Block Diagram



Please note that not all family members contain same peripheral modules and features.

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

©2001 National Semiconductor Corporation

Table of Contents

1.0 General Description 1
2.0 Features 3
3.0 Device Overview 5
3.1 CR16B CPU Core 5
3.2 Memory 5
3.3 Input/Output Ports 5
3.4 Bus Interface Unit 5
3.5 Interrupts 5
3.6 Multi-Input Wake-up 6
3.7 Dual Clock and Reset 6
3.8 Power Management 6
3.9 Multi-Function Timer 6
3.10 Real-Time TIMER and Watchdog 6
3.11 USART 6
3.12 MICROWIRE/SPI 6
3.13 A/D Converter 6
3.14 Analog Comparators 7
3.15 Development Support 7
3.16 Pin Description 10
4.0 System Configuration 12
4.1 $\overline{\text{ENV0}}$ and $\overline{\text{ENV1}}$ Pins 12
4.2 Module Configuration (MCFG) Register 12
4.3 Module Status (MSTAT) Register 12
5.0 Input/Output Ports 13
5.1 Port Registers 13
5.2 Open-Drain Operation 14
6.0 CPU and Core Registers 15
6.1 General-Purpose Registers 15
6.2 Dedicated Address Registers 15
6.3 Processor Status Register 15
6.4 Configuration Register 16
6.5 Addressing Modes 16
6.6 Stacks 16
6.7 Instruction Set 16
7.0 Bus Interface Unit 18
7.1 Bus Cycles 18
7.2 BIU Control Registers 18
7.3 Wait and Hold States Used 19
8.0 Memory 21
8.1 Flash Program Memory 21
8.2 RAM Memory 24
8.3 EEPROM Data Memory 24
8.4 ISP Memory 24
9.0 Interrupts 27
9.1 Interrupt Operation 27
9.2 Non-Maskable Interrupt 28
9.3 Maskable Interrupts 29
9.4 Interrupt Registers 29
9.5 Interrupt Programming Procedures 31
10.0 Power Management 33
10.1 Active Mode 33
10.2 Power Save Mode 33
10.3 Idle Mode 33
10.4 Halt Mode 33
10.5 Switching Between Power Modes 33
11.0 Dual Clock and Reset 36
11.1 External Crystal Network 36
11.2 Main System Clock 37
11.3 Slow System Clock 37
11.4 Power-On Reset 38
11.5 External Reset 38
11.6 Dual Clock and Reset Registers 38
11.7 Slow Clock Prescaler Register (PRSSC) 38
12.0 Multi-Input Wake-Up 39
12.1 Wake-Up Edge Detection Register (WKEDG) 39
12.2 Wake-Up Enable Register (WKENA) 39
12.3 Wake-Up Source Select Register (WKCTRL) 40
12.4 Wake-Up Pending Register (WKPND) 40
12.5 Wake-Up Pending Clear Register (WKPCL) 40
12.6 Programming Procedures 40
13.0 Real-Time Timer and WATCHDOG 41
13.1 TWM Structure 41
13.2 Timer T0 Operation 41
13.3 WATCHDOG Operation 42
13.4 TWM Registers 42
13.5 WATCHDOG Programming Procedure 43
14.0 Multi-Function Timer 45
14.1 Timer Structure 45
14.2 Timer Operating Modes 47
14.3 Timer Interrupts 50
14.4 Timer I/O Functions 50
14.5 Timer Registers 51
15.0 MICROWIRE/SPI 54
15.1 MICROWIRE Operation 54
15.2 Master Mode 55
15.3 Slave Mode 56
15.4 Interrupt Generation 57
15.5 MICROWIRE Interface Registers 58
16.0 USART 61
16.1 Functional Overview 61
16.2 USART Operation 61
16.3 USART Registers 65
16.4 Baud Rate Calculations 67
17.0 Analog Comparators 68
17.1 Analog Comparator Control/Status Register (CMPCTRL)68
17.2 Analog Comparator Usage 68
18.0 A/D Converter 69
18.1 Operating Modes 69
18.2 A/D Converter Registers 70
18.3 A/D Converter Programming 72
19.0 Memory Map 73
20.0 Register Layouts 77
20.1 Register layout 77
21.0 ELECTRICAL CHARACTERISTICS 81
Comparator AC and DC Characteristics 83
Output Signal Levels 85
22.0 Appendix 95
22.1 8-bit MICROWIRE/SPI (MWSPI) 95
22.2 Timing and watchdog module 95
23.0 Physical Dimension 97

1.0 General Description (Continued)

In the following text, device is alsays refered to the family of CompactRISC 16-bit microcontrollers. For the exact feature set, check individual datasheets.

The device is available in a variety of package sizes and types. All devices have 48 kbytes of reprogrammable flash program memory, 1.5 kbytes of ISP memory, 2 kbytes of static RAM, and 640 bytes of non-volatile EEPROM data memory. The 80-pin device has two USARTs, two 16-bit multi-function timers, one SPI/MICROWIRE-PLUS™ serial interface, an 8-channel A/D converter, two analog comparators, WATCHDOG™ protection mechanism, and up to 48 general-purpose I/O pins. The 44-pin devices offer the same basic features as the 80-pin device, but with fewer I/O ports and peripheral modules due to the smaller number of available pins.

All devices operate with a high-frequency crystal as the main clock source. Some packages allow the device to operate with either the main clock source or with a slow (32.768 KHz) oscillator in Power Save mode. The device supports several Power Save modes which are combined with multi-source interrupt and wake-up capabilities.

Powerful cross-development tools are available from National Semiconductor and third party suppliers to support the development and debugging of application software for the device. These tools let you program the application software in C and are designed to take full advantage of the CompactRISC architecture.

2.0 Features

- CPU Features
  - — Fully static core, capable of operating at any rate from 0 to 20 MHz (4 MHz minimum in active mode)
  - — 50 ns instruction cycle time with a 20 MHz external clock frequency
  - — Multi-source vectored interrupts (internal, external, and on-chip peripheral)
  - — On-chip power-on reset
- On-Chip Memory
  - — 48 kbytes of flash program memory or ROM memory (100K cycle)
  - — 1.5 kbytes of ISP memory (100K cycle)
  - — 2 kbytes of static RAM data memory
  - — 640 bytes of non-volatile EEPROM data memory, word-programmable (100K cycle)
- On-Chip Peripherals
  - — Up to two Universal Synchronous/Asynchronous Receiver/Transmitter (USART) devices
  - — Programmable Idle Timer and real-time clock (T0)
  - — Up to two dual 16-bit multi-function timers (MFT1 and MFT2)
  - — SPI/MICROWIRE-PLUS serial interface
  - — 8-channel, 8-bit Analog-to-Digital (A/D) converter with external voltage reference, programmable sample-and-hold delay, and programmable conversion frequency
  - — Up to two analog comparators
  - — Integrated WATCHDOG logic
- I/O Features
  - — Up to 48 general-purpose I/O pins (shared with on-chip peripheral I/O pins)
  - — Programmable I/O pin characteristics: TRI-STATE output, push-pull output, weak pull-up input, high-impedance input
  - — Software-configurable Schmitt triggers on inputs
- Power Supply
  - — 4.5V to 5.5V single-supply operation
- Temperature Range
  - — 0°C to +70°C
  - — –40°C to +85°C
  - — –40°C to +125°C
- Development Support
  - — Real-time emulation and full program debug capabilities available
  - — CompactRISC tools provide C programming and debugging support

CR16 CompactRISC microcontroller Family Selection Guide

Programmable devices

NSID	Speed (MHz)	Flash/ ROM (kByte)	EEPROM Data Memory (Bytes)	SRAM (kBytes)	USART	Timer	I/Os	Temp. Range	Peripherals	Package Type
CR16MHS9VJEx	20	48	640	2	2	2	48	E, I	ADC, Comparators	80PQFP
CR16MFS944Vx	20	48	640	2	2	1	33	E, I	ADC	44PLCC
CR16MES944Vx	20	48	640	2	1	2	33	E, I	ADC	44PLCC
CR16MNS944Vx	20	48	None	2	1	2	33	C, I	None	44PLCC
CR16MUS944Vx	8	48	None	2	1	2	33	C	None	44PLCC

ROM devices

NSID	Speed (MHz)	Flash/ ROM (kByte)	EEPROM Data Memory (Bytes)	SRAM (kBytes)	USART	Timer	I/Os	Temp. Range	Peripherals	Package Type
CR16MHS5VJExy	20	48	640	2	2	2	48	E, I	ADC, Comparators	80PQFP
CR16MFS544Vxy	20	48	640	2	2	1	33	E, I	ADC	44PLCC
CR16MES544Vxy	20	48	640	2	1	2	33	E, I	ADC	44PLCC
CR16MPS544Vxy	20	48	None	2	1	2	33	C, I	ADC	44PLCC
CR16MNS544Vxy	20	48	None	2	1	2	33	C, I	None	44PLCC
CR16MUS544Vxy	8	48	None	2	1	2	33	C	None	44PLCC

Note:

- Suffix x in the NSID is defined below:

Temperature Ranges:

E = Extended	-40°C to +125°C is represented when x is 7
I = Industrial	-40°C to +85°C is represented when x is 8
C = Commercial	0°C to +70°C is represented when x is 9

- Suffix y in the NSID defines the ROM code.

Note: All devices contains Clock and Reset, MICROWIRE/ API, Multi-Input Wake-Up (MIWU), Power Management (PMM), and the Real-Time Timer and Watchdog (TWM) modules.

44-Pin PLCC versus 80-Pin PQFP

For 44PLCC packages, MICROWIRE/SPI slave mode, the first 4 MIWU channels and the Vref pin are not available. 80-pin PQFP packages provide the MICROWIRE/SPI master and slave modes, 8 MIWU channels, Vref pin, and two USARTs and two MFTs.

3.0 Device Overview

The family of CompactRISC 16-bit microcontrollers are complete microcomputers with all system timing, interrupt logic, program memory, data memory, and I/O ports included on-chip, making it well-suited to a wide range of embedded controller applications.

3.1 CR16B CPU CORE

The device uses the CR16B CPU core module. This is the same core used in other CompactRISC family members.

The high performance of the CPU core results from the implementation of a pipelined architecture with a two-bytes-per-cycle pipelined system bus. As a result, the CPU can support a peak execution rate of one instruction per clock cycle.

Compared with conventional RISC processors, the device differs in the following ways:

- The CPU core uses on-chip rather than external memory. This eliminates the need for large and complex bus interface units.
- Most instructions are 16 bits, so all basic instructions are just two bytes long. (Additional bytes are sometimes required for immediate values, so instructions can be two or four bytes long.)
- Non-aligned word access is allowed. Each instruction can operate on 8-bit or 16-bit.
- The device is designed to operate with a clock rate in the 10 to 25 MHz range rather than 100 MHz or more. Most embedded systems face EMI and noise constraints that limit clock speed to these lower ranges. A lower clock speed means a simpler, less costly silicon implementation.
- The instruction pipeline uses three stages. A smaller pipeline eliminates the need for costly branch prediction mechanisms and bypass registers, while maintaining adequate performance for typical embedded controller applications.

3.2 MEMORY

The CompactRISC architecture supports a uniform linear address space of 2 megabytes. The device implementation of this architecture uses only the lowest 64 kbytes of address space. Four types of on-chip memory occupy specific intervals within this address space: 48 kbytes of flash program memory, 1.5 kbytes of ISP memory, 2 kbytes of static RAM, and 640 bytes of EEPROM data memory.

The 48 kbytes of flash program memory are used to store the application program. It has security features to prevent unintentional programming and to prevent unauthorized access to the program code. This memory can be programmed either with the device plugged into an EPROM programmer unit (external programming) or with the device installed in the application system (in-system programming).

The 2 kbytes of static RAM are used for temporary storage of data and for the program stack and interrupt stack. Read and write operations can be byte-wide or word-wide, depending on the instruction executed by the CPU. Each memory access requires one clock cycle; no wait cycles or hold cycles are required.

The 640 bytes of EEPROM data memory are used for non-volatile storage of data, such as configuration settings entered by the end-user. The CPU reads or writes this memory by using ordinary byte-wide or word-wide memory access commands. After the CPU performs a write to this memory, the on-chip hardware completes the EEPROM programming in the background. A register status bit indicates the status of the EEPROM programming operation.

There is a factory programmed boot memory used to store In-System-Programming (ISP) code. (this code allows programming of the program memory via one of the USART interfaces in the final application.)

For the flash program memory, the device internally generates the necessary voltages for programming. No additional power supply is required.

3.3 INPUT/OUTPUT PORTS

Each device has 48 software-configurable I/O pins, organized into six 8-pin ports called Port B, Port C, Port F, Port G, Port L, and Port I. Each pin can be configured to operate as a general-purpose input or general-purpose output. In addition, many I/O pins can be configured to operate as a designated input or output for an on-chip peripheral module such as the USART, timer, A/D converter, or MICROWIRE/SPI interface.

The I/O pin characteristics are fully programmable. Each pin can be configured to operate as a TRI-STATE output, push-pull output, weak pull-up input, or high-impedance input. Input pins can be software-configured to use Schmitt triggers for noise resistance.

Each 44-pin device has a subset of the pins available in the 80-pin device. This results in the loss of some features that are available in the larger-package device:

- One of the two USARTs or one of the two multi-function timers (depending on package selection)
- Synchronous mode in the remaining USART(s)
- Slave mode operation for the MICROWIRE/SPI interface
- Separate external V_{REF} for the A/D converter
- Comparators
- Four of the eight Multi-Input Wakeup pins
- $\overline{\text{NMI}}$ interrupt input pin

3.4 BUS INTERFACE UNIT

The Bus Interface Unit (BIU) controls the interface between the on-chip modules to the internal core bus. It determines the configured parameters for bus access (such as the number of wait states for memory access) and issues the appropriate bus signals for each requested access.

The BIU uses a set of control registers to determine how many wait states and hold states are to be used when accessing EEPROM memory. Upon start-up of the device, these registers must be programmed with appropriate values so that the minimum allowable number states is used. This number varies with the clock frequency and the type of on-chip device being accessed.

3.5 INTERRUPTS

The Interrupt Control Unit (ICU) receives interrupt requests from internal and external sources and generates interrupts to the CPU. An interrupt is an event that temporarily stops the normal flow of program execution and causes a separate interrupt service routine to be executed. After the interrupt is serviced, CPU execution continues with the next instruction in the program following the point of interruption.

Interrupts from the timers, USARTs, MICROWIRE/SPI interface, multi-input wake-up, and A/D converter are all maskable interrupts; they can be enabled or disabled by the software. There are 16 of these maskable interrupts, organized into 16 predetermined levels of priority.

The highest-priority interrupt is the Non-Maskable Interrupt ($\overline{\text{NMI}}$), which is generated by a signal received on the $\overline{\text{NMI}}$ input pin. This interrupt is not available in the 44-pin packages.

3.6 MULTI-INPUT WAKE-UP

The Multi-Input Wake-up (MIWU) module can be used for either of two purposes: to provide inputs for waking up (exiting) from the HALT, IDLE, or Power Save mode; or to provide general-purpose edge-triggered maskable interrupts from external sources. This eight-channel module generates one combined interrupt to the CPU based on the signals received on its eight input channels. Channels can be individually enabled or disabled, and programmed to respond to positive or negative edges.

3.7 DUAL CLOCK AND RESET

The Dual Clock and Reset (CLK2RES) module generates a high-speed main system clock from an external crystal network. It also provides the main system reset signal and a power-on reset function.

In the 80-pin package, the module also generates a slow system clock (32.768 KHz) from another external crystal network. The slow clock is used for operating the device in power-save mode. For the 44-pin devices and for devices not using a secondary crystal network, the slow clock can be generated by dividing the high-speed main clock by a prescaler factor.

3.8 POWER MANAGEMENT

The Power Management Module (PMM) improves the efficiency of the device by changing the operating mode (and therefore the power consumption) according to the current level of activity.

The device can operate in any of four power modes:

- Active: The device operates at full speed using the high-frequency clock. All device functions are fully operational.
- Power Save: The device operates at reduced speed using the slow clock. The CPU and some modules can continue to operate at this low speed.
- IDLE: The device is inactive except for the Power Management Module and Timing and Watchdog Module, which continue to operate using the slow clock.
- HALT: The device is inactive but still retains its internal state (RAM and register contents).

3.9 MULTI-FUNCTION TIMER

The Multi-Function Timer (MFT16) module contains two independent timer/counter units called MFT1 and MFT2, each containing a pair of 16-bit timer/counter registers. Each timer/counter unit can be configured to operate in any of the following modes:

- Processor-Independent Pulse Width Modulation (PWM) mode, which generates pulses of a specified width and duty cycle, and which also provides a general-purpose timer/counter
- Dual Input Capture mode, which measures the elapsed time between occurrences of external events, and which also provides a general-purpose timer/counter
- Dual Independent Timer mode, which generates system timing signals or counts occurrences of external events
- Single Input Capture and Single Timer mode, which provides one external event counter and one system timer

3.10 REAL-TIME TIMER AND WATCHDOG

The Timing and Watchdog Module (TWM) generates the clocks and interrupts used for timing periodic functions in the system. It also provides Watchdog protection against software errors. The module operates on the slow (32.768 KHz) clock.

The real-time timer generates a periodic interrupt to the CPU at a software-programmed interval. This can be used for real-time functions such as a time-of-day clock.

The Watchdog is designed to detect program execution errors such as an infinite loop or a “runaway” program. Once Watchdog operation is initiated, the application program must periodically write a specific value to a Watchdog register, within specific time intervals. If the software fails to do so, a Watchdog error is triggered, which resets the device.

3.11 USART

The USART is a Universal Synchronous/Asynchronous Receiver-Transmitter, a device used for serial communications. It supports a wide range of programmable baud rates and data formats, and handles parity generation and several error detection schemes. The baud rate is generated on-chip, under software control.

The synchronous mode of operation is not available in the 44-pin devices.

3.12 MICROWIRE/SPI

The MICROWIRE/SPI (MWSPI) interface module supports asynchronous serial communications with other devices that conform to MICROWIRE or Serial Peripheral Interface (SPI) specifications.

The MICROWIRE interface allows several devices to communicate over a single system consisting of three wires: serial in, serial out, and shift clock. At any given time, one device on the MICROWIRE interface operates as the master, while all other devices operate as slaves. An 80-pin device supports the full set of slave select and Ready lines for multi-slave implementation, while a 44-pin device has only the basic Data-in/Data-out/Clock lines, limiting its implementation to master mode.

3.13 A/D CONVERTER

The A/D Converter (ADC) module is an 8-channel multiplexed-input analog-to-digital converter. The A/D Converter receives an analog voltage signal on an input pin and converts the analog signal into an 8-bit digital value using successive approximation. The CPU can then read the result from a memory-mapped register. The module supports four automated operating modes, providing single-channel or scanned 4-channel operation in single or continuous mode.

The 80-pin device has a separate pin, Vref, for the A/D reference voltage. The 44-pin devices use the AVCC (analog VCC) power supply pin as the reference voltage.

3.14 ANALOG COMPARATORS

The Dual Analog Comparator (ACMP2) module contains two independent analog comparators with all necessary control logic. Each comparator unit compares the analog input voltages applied to two input pins and determines which voltage is higher. The CPU uses a memory-mapped register to control the comparator and to obtain the comparison results. The comparison result can also be applied to comparator output pins.

3.15 DEVELOPMENT SUPPORT

A powerful cross-development tool set is available from National Semiconductor and third parties to support the development and debugging of application software for the device. The tool set lets you program the application software in C and is designed to take full advantage of the CompactRISC architecture.

There are In-System Emulation (ISE) devices available for the device from iSYSTEM™, as well as lower-cost evaluation boards. See your National Semiconductor sales representative for current information on availability of various features of emulation equipment and evaluation boards.

Table 1 Package Pin Assignments

Pin Name	Alternate Function(s)	44-pin PLCC Package Pin Number	80-pin PQFP Package Pin Number	Type
GND		44	13	PWR
Vcc		1	14	PWR
GND		N/A	15	PWR
$\overline{\text{ENV0}}$-44/SLCLK[a]		2	N/A	I/O
PC0		3	17	I/O
PC1		4	18	I/O
PC2		5	19	I/O
PC3		6	20	I/O
PC4		7	21	I/O
PC5		8	22	I/O
PC6		9	23	I/O
PC7		10	24	I/O
$\overline{\text{ENV0}}$-80/SLCLK[a]		N/A	26	I/O
CLKOUT2[b]		11	27	I/O
$\overline{\text{ENV1}}$/CLK[a]		11	28	I/O
PG7	CKX1	N/A	29	I/O
PG6	TDX1	12	30	I/O
PG5	RDX1	13	31	I/O
PG4	$\overline{\text{MRDY}}$	N/A	32	I/O
PG3	$\overline{\text{MCS}}$	N/A	33	I/O
PG2	MSK	14	34	I/O
PG1	MDODI	15	35	I/O
PG0	MDIDO	16	36	I/O
PF7		N/A	38	I/O
PF6	CKX2	N/A	39	I/O
PF5	T2B	17	40	I/O
PF4	T2A	18	41	I/O
PF3[c]	TDX2	19 or N/A *	42	I/O
PF2[c]	RDX2	20 or N/A *	43	I/O
PF1[c]	T1B	N/A or 19 *	44	I/O
PF0[c]	T1A	N/A or 20 *	45	I/O
NMI		N/A	46	I
X1CKO		21	48	O
X1CKI		22	49	I
GND		N/A	50	PWR
Vcc		23	51	PWR

Table 1 Package Pin Assignments

Pin Name	Alternate Function(s)	44-pin PLCC Package Pin Number	80-pin PQFP Package Pin Number	Type
GND		24	52	PWR
X2CKO		N/A	53	O
X2CKI		N/A	54	I
PI0	ACH0[e,f]	25	57	I/O
PI1	ACH1[e,f]	26	58	I/O
PI2	ACH2[e,f]	27	59	I/O
PI3	ACH3[e,f]	28	60	I/O
PI4	ACH4[e,f], MIWU4	29	61	I/O
PI5	ACH5[e,f], MIWU5	30	62	I/O
PI6	ACH6[e,f], MIWU6	31	63	I/O
PI7	ACH7[e,f], MIWU7	32	64	I/O
Vref		N/A	66	PWR
AVcc		33[f]	67	PWR
AGND		34[f]	68	PWR
GND		N/A	69	PWR
Vcc		N/A	70	PWR
GND		N/A	71	PWR
$\overline{\text{RESET}}$[d]		35	76	I
PB0		36	77	I/O
PB1		37	78	I/O
PB2		38	79	I/O
PB3		39	80	I/O
PB4		40	1	I/O
PB5		41	2	I/O
PB6		42	3	I/O
PB7		43	4	I/O
PL0	COMP1N[e]	N/A	73	I/O
PL1	COMP1P[e]	N/A	74	I/O
PL2	COMP1O	N/A	5	I/O
PL3		N/A	6	I/O
PL4	COMP2N[e], MIWU0	N/A	8	I/O
PL5	COMP2P[e], MIWU1	N/A	9	I/O

Table 1 Package Pin Assignments

Pin Name	Alternate Function(s)	44-pin PLCC Package Pin Number	80-pin PQFP Package Pin Number	Type
PL6	COMP2O, MIWU2	N/A	11	I/O
PL7	MIWU3	N/A	12	I/O

Notes:
a. The $\overline{ENV0}$ and $\overline{ENV1}$ pins each have a weak pullup to keep the input from floating.
b. The CLKOUT2 function is shared with the $\overline{ENV1}$/CLK pin in the 44-pin device.
c. In the 44-pin CR16MES, CR16MNS, CR16MPS, and CR16MUS packages, PF1 and PF0 are available; PF3 and PF2 are not available.
In the 44-pin CR16MFS, CR16MOS, CR16MQS, and CR16MVS packages, PF3 and PF2 are available; PF1 and PF0 are not available.
d. The $\overline{RESET}$ input has a weak pulldown.
e. These functions are always enabled due to the direct low-impedance path to these pins.
f. These functions may not be available on some 44-pin devices.

3.16 PIN DESCRIPTION

The following is a brief description of all CR16MHR6 pins.

Some pins have alternate functions which may be enabled. These pins can be individually configured as general purpose pins, even when the module they belong to is enabled.

Table 2 Input Pins

Signal	Type	Active	Pin (* for a shared pin)	Function
X1CKI	OSC	High		Main oscillator clock input.
X2CKI	OSC	High		32kHz oscillator clock input.
$\overline{\text{RESET}}$	CMOS	Low		Chip general reset pin. Schmitt trigger input, asynchronous.
$\overline{\text{ISE}}$	CMOS	Low		Interrupt input for development system.
T1B	CMOS	Prog.	*	Timer 1 input B. Shares pin with I/O port pin PF1.
T2B	CMOS	Prog.	*	Timer 2 input B. Shares pin with I/O port pin PF5.
RDX1	CMOS	High	*	USART 1 receive data input. Shares pin with I/O port pin PG5.
RDX2	CMOS	High	*	USART 2 receive data input. Shares pin with I/O port pin PF4.
ACH0	Analog		*	A2D converter channel 0. Shares pin with I/O port pin PI0
ACH1	Analog		*	A2D converter channel 1. Shares pin with I/O port pin PI1
ACH2	Analog		*	A2D converter channel 2. Shares pin with I/O port pin PI2
ACH3	Analog		*	A2D converter channel 3. Shares pin with I/O port pin PI3
ACH4	Analog		*	A2D converter channel 4. Shares pin with I/O port pin PI4
ACH5	Analog		*	A2D converter channel 5. Shares pin with I/O port pin PI5
ACH6	Analog		*	A2D converter channel 6. Shares pin with I/O port pin PI6
ACH7	Analog		*	A2D converter channel 7. Shares pin with I/O port pin PI7
$\overline{\text{MCS}}$	CMOS	Low	*	SPI/MICROWIRE slave select. Shares pin with I/O port pin PG3.
$\overline{\text{NMI}}$	CMOS	Low		External non-maskable interrupt.
$\overline{\text{ENV0}}$-44	CMOS	Low	*	Strap pin on 44-pin package to select operating environment.
$\overline{\text{ENV0}}$-80	CMOS	Low	*	Strap pin on 80-pin package to select operating environment.
$\overline{\text{ENV1}}$	CMOS	Low	*	Strap pin to select operating environment.
$\overline{\text{ENV2}}$	CMOS	Low		Strap pin to select operating environment.

Table 3 Output Pins

Signal	Type	Active	Pin (* for a shared pin)	Function
X1CKO	OSC	High		Main oscillator clock output.
X2CKO	OSC	High		32 kHz oscillator clock output.
CLK	CMOS	High		External reference clock for development environment.
TDX1	CMOS	High	*	USART 1 transmit data output. Shares pin with I/O port pin PG6.
TDX2	CMOS	High	*	USART 2 transmit data output. Shares pin with I/O port pin PF5.
$\overline{\text{MRDY}}$	CMOS	Low	*	SPI/MICROWIRE slave ready output. Shares pin with I/O port pin PG4
CLKOUT2	OSC	High		System clock divided-by-2 output (shared with the $\overline{\text{ENV1}}$ pin in the 44-pin device)

Table 4 Input/Output Pins

Signal	Type	Active	Pin (* for a shared pin)	Function
PF[0:7]	CMOS	High	*	Generic I/O port. Shared with T1A, T1B, T2A, T2B, RDX2, TDX2, CKX2.
PG[0:7]	CMOS	High	*	Generic I/O port. Shared with MDIDO, MDODI, MSK, $\overline{\text{MCS}}$, $\overline{\text{MRDY}}$, RDX1, TDX1, CKX1.
PB[0:7]	CMOS	High	*	Generic I/O port.
PC[0:7]	CMOS	High	*	Generic I/O port.
PL[0:7]	CMOS	High	*	Generic I/O port. Shared with six comparator pins and four MIWU pins.
PI[0:7]	CMOS	High	*	Generic I/O port. Shared with ADC input channels 0-7 and four MIWU pins.
T1A	CMOS	Prog	*	Timer 1 input A. Shared with I/O port pin PF0.
T2A	CMOS	Prog	*	Timer 2 input A. Shared with I/O port pin PF4.
MDIDO	CMOS	High	*	Master In/Slave Out port: SPI/Microwire. Shared with I/O pin PG0.
MDODI	CMOS	High	*	Master Out/Slave In port: SPI/Microwire. Shared with I/O pin PG1.
MSK	CMOS	High	*	SPI/Microwire clock. Shared with I/O pin PG2.
CKX1	CMOS	High	*	USART 1 clock signal. Shared with I/O pin PG7.
CKK2	CMOS	High	*	USART 2 clock signal. Shared with I/O pin PF6.

Table 5 Power Supply

Signal	Function
Vcc	Main digital power supply.
Vref	Voltage reference supply for analog to digital converter.
AVcc	Analog power supply for analog/digital converter.
AGND	Analog reference ground supply.
GND	Main digital reference ground.

4.0 System Configuration

The CR16MHS9 has two input pins, $\overline{\text{ENV0}}$ and $\overline{\text{ENV1}}$, which are used to specify the operating environment of the device upon reset. There are also two system configuration registers, called the Module Configuration (MCFG) register and the Module Status (MSTAT) register.

4.1 $\overline{\text{ENV0}}$ AND $\overline{\text{ENV1}}$ PINS

Upon reset, the operating mode of the device is determined by the state of the $\overline{\text{ENV0}}$ and $\overline{\text{ENV1}}$ input pins, as indicated in Table6.

Table 6 Operating Environment Selection

$\overline{\text{ENV1}}$	$\overline{\text{ENV0}}$	Operating Environment
0	0	Test Mode
0	1	Test Mode
1	0	In-System Programming mode
1	1	Internal ROM enabled Mode (IRE), if program memory is not empty; or ISP-Mode, if program memory is empty

In the case where the $\overline{\text{ENV1}}$ and $\overline{\text{ENV0}}$ pins are both high, the reset algorithm looks at the FLCTRL2.EMPTY bit to determine whether the program memory is empty, and sets the operating mode accordingly.

The $\overline{\text{ENV0}}$ and $\overline{\text{ENV1}}$ pins have on-chip pull-up devices that are enabled during reset while the pins are being sampled. Therefore, if they are left unconnected, the inputs are considered high and the normal operating mode (IRE-Mode) is selected and the CPU starts to execute code at address 0. To enter any other operating mode, the external hardware must drive the appropriate input low.

In the case where the ISP-Mode is selected, the chip starts executing the ISP code residing in the on-chip boot ROM area.

The Test Modes are reserved for factory testing and for external programming of the flash program memory; they should not be invoked otherwise.

4.2 MODULE CONFIGURATION (MCFG) REGISTER

The MCFG register is a byte-wide, read/write register that sets the general programmable features of the device.

Upon reset, the non-reserved bits of this register are cleared to zero. The start-up software must write a specific value to this register in order to configure the CLK output pin function.

When the software writes to this register, it must write a zero to each reserved bit for the device to operate properly. The register should be written in active mode only, not in power save, HALT, or IDLE mode. However, the register contents are preserved during all power modes.

The MCFG register format is shown below.

7 6	5	4	3	2	1	0
Reserved	CLK2OE	SLCOE2	FEEDM	SLCLKOE	CLKOE	Reserved

CLKOE CPU Clock Output Enable. When this bit is cleared (0), the CLK pin remains in the high-impedance state. When this bit is set (1) in normal operating mode, the CLK pin operates as a CPU clock output.

SLCLKOE Slow Clock Output Enable. When cleared (0), the SLCLK pin of the 44-pin package ($\overline{\text{ENV0}}$-44 pin) remains in the high-impedance state. When set (1), this pin produces the slow clock as an output.

FEEDM Fast EEPROM Data Memory Access. This bit is set (1) for zero-wait-state access to the EEPROM data memory, or cleared (0) for one-wait-state access to the data ROM. For information on the required number of wait states, see Table8.

SLCOE2 Slow Clock Output Enable. When cleared (0), the SLCLK pin of the 80-pin package (ENV0-80 pin) remains in the high-impedance state. When set (1), this pin produces the slow clock as an output.

CLK2OE CPU Clock Divide-by-2 Output Enable. When this bit is cleared (0), the CLKOUT2 pin remains in the high-impedance state. When this bit is set (1) and the CLKOE bit is cleared, the CLKOUT2 pin operates as clock output, with a frequency of one-half that of the CPU clock.

4.3 MODULE STATUS (MSTAT) REGISTER

The MSTAT register is a byte-wide, read-only register that indicates the general status of the device.

The MCFG register format is shown below.

7 4	3	2	1	0
Reserved	PGMBUSY	Reserved	OENV1	OENV0

OENV(1:0) Operating Environment. These two bits contain the values applied to the $\overline{\text{ENV1}}$ and $\overline{\text{ENV0}}$ pins upon reset. These bit values are controlled by the external hardware upon reset and are held constant in the register until the next reset.

PGMBUSY Flash Programming Busy. This bit is automatically set to 1 when either the program memory or the data memory is busy being programmed. It is cleared to 0 when neither of the two flash memories are busy being programmed. When this bit is set, the software should not attempt to access either of these two memories.

5.0 Input/Output Ports

Each device has up to 48 software-configurable I/O pins, organized into six ports of up to eight pins per port. The exact number of port pins varies with the package type. The ports are named Port B, Port C, Port F, Port G, Port L, and Port I.

Each pin can be configured to operate as a general-purpose input or general-purpose output. In addition, many I/O pins can be configured to operate as a designated input or output for an on-chip peripheral module such as the USART. This is called the pin's “alternate function.” The alternate functions of all I/O pins are shown in the pinout diagrams in Table1.

The I/O pin characteristics are fully programmable. Each pin can be configured to operate as a TRI-STATE output, push-pull output, weak pull-up input, or high-impedance input. Different pins within the same port can be individually configured to operate in different modes.

Figure1 is a diagram showing the functional features of an I/O port pin. The register bits, multiplexers, and buffers allow the port pin to be configured into the various operating modes.The output buffer is a TRI-STATE buffer with weak pull-up capability. The weak pull-up, if used, prevents the port pin from going to an undefined state when it operates as an input.

The input buffer is disabled when it is not needed to prevent leakage current. When enabled, it buffers the input signal and sends the pin's logic level to the appropriate on-chip module where it is latched. A Schmitt trigger, when enabled, minimizes the effects of electrical noise.

The electrical characteristics and drive capabilities of the input and output buffers are described in Section21.0.

For some pins, a direct low-impedance path is provided between the pin and an internal analog function. These are the input pins to the A/D converter and the analog comparators

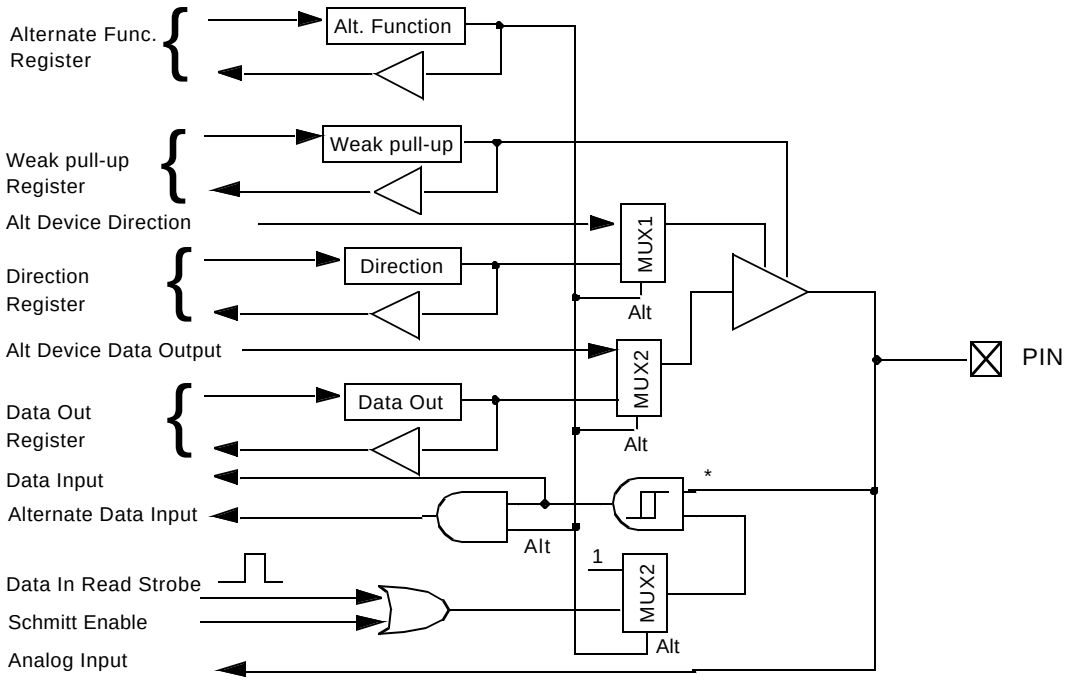
5.1 PORT REGISTERS

Each port has an associated set of memory-mapped registers used for controlling the port and for holding the port data. In general, there are six such registers:

- PxALT: Port alternate function register
- PxDIR: Port direction register
- PxDIN: Port data input register
- PxDOUT: Port data output register
- PxWKPU: Port weak pull-up register
- PxSCHEN: Port Schmitt trigger enable register

In the descriptions of the ports and port registers, the lowercase letter “x” represents the port designation, either B, C, F, G, L, or I. For example, “PxDIR register” means any one of the port direction registers: PBDIR, PCDIR, PFDIR, and so on.

All of the port registers are byte-wide read/write registers, except for the port data input registers, which are read-only registers. Each register bit controls the function of the corresponding port pin. For example, PFDIR.2 (bit 2 of the PFDIR register) controls the operation of port pin PF2.



* Schmitt trigger is used at input when enabled or when the pin’s alternate function is selected.

Figure 1. I/O Pin Functional Diagram

5.1.1 Port Alternate Function Register

Each port that supports an alternate function (any port other than Port B or Port C) has an alternate function register (PxALT). This register determines whether the port pins are used for general-purpose I/O or for the predetermined alternate function. Each port pin can be controlled independently.

A bit cleared to 0 in the alternate function register causes the corresponding pin to be used for general-purpose I/O. In this configuration, the output buffer is controlled by the direction register and the data output register. The input buffer is routed to the data input register. The input buffer is blocked except when the buffer is actually being read.

A bit set to 1 in the alternate function register causes the corresponding pin to be used for its predetermined peripheral I/O function. The output buffer data and TRI-STATE configuration are controlled by signals coming from the on-chip peripheral device. The input buffer is enabled continuously in this case. To minimize power consumption, the input signal should be held within 0.2 volts of the VCC or GND voltage.

A reset operation clears the port alternate function registers to 0, which programs the pins to operate as general-purpose I/O ports. This register must be enabled before the corresponding alternate function is enabled.

5.1.2 Port Direction Register

The port direction register (PxDIR) determines whether each port pin is used for input or for output. A bit cleared to 0 causes the pin to operate as an input, which puts the output buffer in the high-impedance state. A bit set to 1 causes the pin to operate as an output, which enables the output buffer.

A reset operation clears the port direction registers to 0, which programs the pins to operate as inputs.

5.1.3 Port Data Input Register

The data input register (PxDIN) is a read-only register that returns the current state of each port pin. The CPU can read this register at any time even when the pin is configured as an output.

5.1.4 Port Data Output Register

The data output register (PxDOUT) holds the data to be driven onto each port pin configured to operate as a general-purpose output. In this configuration, writing to the register changes the output value. Reading the register returns the last value written to the register.

A reset operation leaves the register contents unchanged. Upon power-up, the registers contain unknown values.

5.1.5 Port Weak Pull-Up Register

The weak pull-up register (PxWKPU) determines whether each port pin uses a weak pull-up on the output buffer. A bit set to 1 causes the weak pull-up to be used, while a bit cleared to 0 causes the weak pull-up not to be used.

The pull-up device, if enabled by the register bit, operates in the general-purpose I/O mode whenever the port output buffer is in the TRI-STATE mode. In the alternate function mode, the pull-ups are always disabled.

A reset operation clears the port weak pull-up registers to 0, which disables all pull-ups.

5.1.6 Port Schmitt Input Enable Register

PxSCHEN registers are byte-wide read/write registers. They enable the Schmitt trigger characteristics on Px input buffers when these pins are used as general purpose I/O ports. When cleared (0) with the pins used as general purpose I/O ports, each bit in PxSCHEN enables the corresponding input buffer only when the port is read, and therefore the input buffer may not exhibit any Schmitt trigger behavior. When set (1), the input buffer is enabled, and will exhibit Schmitt trigger behavior. PxSCHEN registers are cleared upon reset.

5.2 OPEN-DRAIN OPERATION

A port pin can be configured to operate as an inverting open-drain output buffer. To do this, the CPU should clear the bit in the data output register (PxDOUT) and then use the port direction register (PxDIR) to set the value of the port pin. With the direction register bit set to 1 (direction=out), the value zero is forced on the pin. With the direction register bit cleared to 0 (direction=in), the pin is placed in the TRI-STATE mode. If desired, the internal weak pull-up can be enabled to pull the signal high when the output buffer is in the TRI-STATE mode.

6.0 CPU and Core Registers

The device uses the same CR16B CPU core as other CompactRISC family members. The core's Reduced Instruction Set Computer (RISC) architecture allows a processing rate of up to one instruction per clock cycle.

The CPU core uses the following set of internal registers:

- General-purpose registers (R0-R13, RA, and SP)
- Dedicated address registers (PC, ISP, and INTBASE)
- Processor Status Register (PSR)
- Configuration Register (CFG)

All of these registers are 16 bits wide except for the three address registers, which are 21 bits wide.

Some register bits are designated as “reserved.” The CPU must write a zero to each of these bit locations when it writes to the register. Read operations from reserved bit locations return undefined values.

6.1 GENERAL-PURPOSE REGISTERS

There are 16 general-purpose registers, designated R0 through R13, RA, and SP. Registers R0 through R13 can be used for any purpose such as holding variables, addresses, or index values. The RA register is usually used to store the return address upon entry into a subroutine. The SP register is usually used as the pointer to the program run-time stack.

If a general-purpose register is used for a byte-wide operation, only the low-order byte is referenced or modified. The high-order byte is not used or affected by a byte-wide operation.

6.2 DEDICATED ADDRESS REGISTERS

There are three dedicated address registers: the Program Counter (PC), the Interrupt Stack Pointer (ISP), and the Interrupt Base Register (INTBASE). Each of these registers is 21 bits wide.

6.2.1 Program Counter

The PC register contains the address of the first byte of the instruction currently being executed. It is automatically incremented or changed by the appropriate amount each time an instruction is executed.

The five most significant and the least significant bit of this register are always zero. The least significant bit of the PC is always zero, thus instruction must always be aligned to even addresses in the range of 0000 to FFFE hex.

Upon reset, the PC register is initialized to zero and program execution starts at that address. When a reset signal is received, bits 1 through 16 of the PC register (prior to initialization) are stored in register R0. This allows the software to determine the point in the program at which the reset occurred.

6.2.2 Interrupt Stack Pointer

The ISP register points to the lowest address of the last item stored on the interrupt stack. This stack is used by the hardware when an interrupt or trap service procedure is invoked.

The five most significant bits and the least significant bit of this register are always zero. The last item stored on the interrupt stack must be at an even address in the range of E000-E7FF hex, which is the range of the RAM memory in which the stack resides.

6.2.3 Interrupt Base Register

The INTBASE register holds the address of the Dispatch Table for interrupts and traps. The five most significant bits and the least significant bit of this register are always zero.

6.3 PROCESSOR STATUS REGISTER

The Processor Status Register (PSR) holds status information and selects the operating modes for the CPU core. The format of the register is shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I	P	E	0	N	Z	F	0	0	L	T	C

C bit — The Carry (C) bit indicates whether a carry or borrow has occurred after addition or subtraction. It is set to 1 if a carry or borrow has occurred, or cleared to 0 otherwise.

T bit — The Trace (T) bit, when set, causes a Trace (TRC) trap to be executed after every instruction. This bit is automatically cleared to 0 when a trap or interrupt occurs.

L bit — The Low (L) bit is set by comparison operations. In a comparison of unsigned integers, the bit is set to 1 if the second operand (Rdest) is less than the first operand (Rsrc). Otherwise, it is cleared to 0.

F bit — The Flag (F) bit is a general condition flag that is set by various instructions. It may be used to signal exception conditions or to distinguish the results of an instruction. For example, integer arithmetic instructions use this bit to indicate an overflow condition after an addition or subtraction operation.

Z bit — The Zero (Z) bit is set by comparison operations. In a comparison of integers, the bit is set to 1 if the two operands are equal. Otherwise, it is cleared to 0.

N bit — The Negative (N) bit is set by comparison operations. In a comparison of signed integers, the bit is set to 1 if the second operand (Rdest) is less than the first operand (Rsrc). Otherwise, it is cleared to 0.

E bit — The Local Maskable Interrupt Enable (E) bit is used to enable or disable maskable interrupts. If this bit and the Global Maskable Interrupt Enable (I) bit are both set to 1, all maskable interrupts are accepted. Otherwise, only nonmaskable interrupts are accepted. The E bit is set to 1 by the Enable Interrupts (EI) instruction and cleared to 0 by the Disable Interrupts (DI) instruction.

P bit — The Trace Trap Pending (P) bit is used together with the Trace (T) bit to prevent a Trace (TRC) trap from occurring more than once for any instruction. The P bit may be cleared to 0 (no TRC trap pending) or set to 1 (TRC trap pending).

I bit — The Global Maskable Interrupt Enable (I) bit is used to enable or disable maskable interrupts. If this bit and the Local Maskable Interrupt Enable (E) bit are both set to 1, all maskable interrupts are accepted. Otherwise, only the non-maskable interrupts are accepted. This bit is automatically cleared to 0 when an interrupt occurs and automatically set to 1 upon completion of an interrupt service routine.

Upon reset, all non-reserved bits of the register are cleared to 0 except for the E bit (bit 9), which is set to 1. When a device reset occurs, the PSR contents prior to the reset are stored into register R1, allowing the initialization software to determine the state of the device prior to the reset operation.

6.4 CONFIGURATION REGISTER

The Configuration (CFG) register is a 16-bit core register that determines the size of the INTBASE register. For the device, the CFG register should always be left in its default state (cleared to zero), resulting in a 16-bit INTBASE register.

6.5 ADDRESSING MODES

Each instruction operates on one or more operands. An operand can be a register or a memory location.

Most instructions use one, two, or three device registers as operands. The instruction opcode specifies the registers to be operated on. Some instructions may use an immediate value (a value provided in the instruction itself) instead of a register.

Memory locations are accessed only by the Load and Store commands. The memory location to use for a particular instruction can be specified as an absolute, relative, or far-relative address.

The instruction set supports the following addressing modes:

Register Mode — The operand is a general-purpose register: R0 through R13, RA, or SP. For example:

ADDB R1, R2

Immediate Mode — A constant operand value is specified within the instruction. In a branch instruction, the immediate operand is a displacement from the program counter (PC). In the assembly language syntax, a dollar sign indicates an immediate value. For example:

MULW $4, R4

Relative Mode — The operand is located in memory. Its address is obtained by adding the contents of a general purpose register to the constant value encoded into the displacement field of the instruction. For example:

LOADW 12(R5), R6

Far-Relative Mode — The operand is located in memory. Its address is obtained by concatenating a pair of adjacent general-purpose registers to form a 21-bit value, and adding this value to the constant value encoded into the displacement field of the instruction. The device implementation only uses the first 64K of the address space, so the relative mode is sufficient to access the entire usable address space; the far-relative mode is not used.

Absolute Mode — The operand is located in memory. Its address is specified within the instruction. For example:

LOADB 4000, R6

For additional information on the instruction set and instruction encoding, see the CompactRISC CR16B Programmer's Reference manual.

6.6 STACKS

A stack is a one-dimensional data buffer in which values are entered and removed one at a time. The last value entered is the first one removed. A register called the stack pointer contains the current address of the last item entered on the stack. In the device, when an item is entered or “pushed” onto the stack, the stack expands downward in memory (the stack pointer is decremented). When an item is removed or “popped” from the stack, the stack shrinks upward in memory (the stack pointer is incremented).

The device uses two type of stacks: the program stack and the interrupt stack.

The program stack is used by the software to save and restore register values upon entry into and exit from a subroutine. The software can also use the program stack to store local and temporary variables. The stack pointer for this stack is the SP register.

The interrupt stack is used to save and restore the program state when an exception occurs (an interrupt or software trap). The on-chip hardware automatically pushes the program state information onto the stack before the exception service procedure is executed. Upon exit from the exception service procedure, the hardware pops this information from the stack and restores the program state. The stack pointer for this stack is the ISP, or Interrupt Stack Pointer.

6.7 INSTRUCTION SET

Table7 is a summary list of all instructions in the device instruction set. For each instruction, the table shows the mnemonic with a brief description of the operation performed.

In the Mnemonic column, the lower-case letter “i” is used to indicate the type of integer that the instruction operates on, either “B” for byte or “W” for word. For example, the notation ADDi for the “add” instruction means that there are two forms of this instruction, ADDB and ADDW, which operate on bytes and words, respectively.

Similarly, the lower-case string “cond” is used to indicate the type of condition tested by the instruction. For example, the notation Jcond represents a class of conditional jump instruc-

tions: JEQ for Jump on Equal, JNE for Jump on Not Equal, and so on.

For detailed information on all instructions, see the CompactRISC CR16B Programmer's Reference manual.

Table 7 Device Instruction Set Summary

Mnemonic	Description
ADDi	Add Integer
ADDUi	Add Unsigned Integer
ADDCi	Add Integer with Carry
ANDi	Bitwise Logical AND
ASHUi	Arithmetic Shift Unsigned
Bcond	Conditional Branch
Bcond0i	Compare Register to 0 and Branch
Bcond1i	Compare Register to 1and Branch
BAL	Branch and Link
BR	Unconditional Branch
CBITi	Clear Bit in Integer
CMPi	Compare Integer
DI	Disable Maskable Interrupts
EI	Enable Maskable Interrupts
EIWAIT	Enable Interrupts and Wait for Interrupt
EXCP	Exception
Jcond	Conditional Jump
JAL	Jump and Link
JUMP	Jump
LOADi	Load Integer
LOADM	Load Multiple Registers
LPR	Load Processor Register
LSHi	Logical Shift Integer
MOVi	Move Integer
MOVXB	Move with Sign-Extension
MOVZB	Move with Zero-Extension
MULi	Multiply Integer
MULSi	Multiply Signed
MULUW	Multiply Unsigned
NOP	No Operation
ORi	Bitwise Logical OR
POPrt	Pop Registers from Stack
PUSH	Push Registers on Stack
RETX	Return from Exception
Scond	Save Condition as Boolean
MULi	Multiply Integer
SBITi	Set Bit in Integer
STORi	Store Integer
STORM	Store Registers to Memory
SUBi	Subtract Integer

Table 7 Device Instruction Set Summary

Mnemonic	Description
SUBCi	Subtract Integer with Carry
TBIT	Test Bit
WAIT	Wait for Interrupt
XORi	Bitwise Logical Exclusive OR

7.0 Bus Interface Unit

The Bus Interface Unit (BIU) controls the interface between the on-chip modules and the internal core bus. It determines the configured parameters for bus access (such as the number of wait states for memory access) and issues the appropriate bus signals for the requested access.

7.1 BUS CYCLES

There are four types of data transfer bus cycles:

- Normal read
- Fast read
- Early write
- Late write

Note: For write operations, this device utilizes the Late write operation.

The type of data cycle used in a particular transaction depends on the type of CPU operation (a write or a read), the type of memory or I/O being accessed, and the access type programmed into the BIU control registers (early/late write or normal/fast read).

For read operations, a basic normal read takes two clock cycles, whereas a fast read bus cycle takes one clock cycle. Upon reset of the device, normal read bus cycles are enabled by default.

For write operations, a basic late write bus cycle takes two clock cycles, whereas a basic early write bus cycle takes three clock cycles. Upon reset of the device, early write bus cycles are enabled by default. However, late write bus cycles are needed for ordinary write operations, so this configuration should be changed by the application software (see Section7.2.1).

In certain cases, one or more additional clock cycles are added to a bus access cycle. There are two types of additional clock cycles for ordinary memory accesses, called internal wait cycles (TIW) and hold (T_{hold}) cycles.

A wait cycle is inserted in a bus cycle just after the memory address has been placed on the address bus. This gives the accessed memory more time to respond to the transaction request. A hold cycle is inserted at the end of a bus cycle. This holds the data on the data bus for an extended number of clock cycles.

7.2 BIU CONTROL REGISTERS

The BIU has a set of control registers that determine how many wait cycles and hold cycles are to be used for accessing memory. Upon start-up of the device, these registers should be programmed with appropriate values so that the minimum allowable number of cycles is used. This number varies with the clock frequency used.

There are two applicable BIU registers: the BIU Configuration (BCFG) register and the Static Zone 0 Configuration (SZCFG0) register. These registers control the bus cycle configuration used for accessing the flash program memory.

Note: A system configuration register called the Module Configuration (MCFG) register controls the number of wait cycles used for accessing the EEPROM data memory. This register is described in Section4.2.

7.2.1 BIU Configuration (BCFG) Register

The BIU Configuration (BCFG) Register is a byte-wide, read/write register that selects either early write or late write bus cycles. The register address is F900 hex. Upon reset, the register is initialized to 07 hex. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved					Note 1	Note 1	EWR

EWR — Early Write. This bit is cleared to 0 for late write operation (two clock cycles to write) or set to 1 for early write operation.

Note 1: This bit (bit 1 or bit 2) controls the configuration of the 224-pin device used in emulation equipment. The CPU should set this bit to 1 when it writes to the register.

Upon reset, the BCFG register is initialized to 07 hex, which selects early write operation. However, late write operation is required for normal device operation, so the software should change the register value to 06 hex.

7.2.2 I/O Zone Configuration (IOCFG) Register

The I/O Zone Configuration (IOCFG) register is a word-wide, read/write register that sets the timing and bus characteristics of I/O Zone memory accesses. In the device implementation, the registers associated to Port B and Port C reside in the I/O memory array. (These ports are used as a 16-bit data port, if the device operates in development mode.)

The IOCFG register address is F902 hex. Upon reset, the register is initialized to 069F hex. The register format is shown below.

15	14	13	12	11	10	9	8
Reserved					1	IPST	Reserved

7	6	5	4	3	2	1	0
BW	Reserved		HOLD		WAIT		

WAIT — Memory Wait cycles
This field specifies the number of TIW (internal wait state) clock cycles added for each memory access, ranging from 000 binary for no additional TIW wait cycles to 111 binary for seven additional TIW wait cycles. These bits are ignored if the SZCFG0.FRE bit is set to 1.

HOLD — Memory Hold cycles
This field specifies the number of T_{hold} clock cycles used for each memory access, ranging from 00 binary for no T_{hold} cycles to 11 binary for three T_{hold} clock cycles. These bits are ignored if the SZCFG0.FRE bit is set to 1.

BW — Bus Width
BW defines the external bus width for the I/O zone. Bus width is initialized during reset to its default value, which is 16 bits. If this bit is cleared to 0 the external bus is 8 bits wide. To set to external bus width to 16 bits, this bit has to be set to 1. For the 80 pin package the Bus width has to be set to 1. When using the device in 224 pin package in emulator equipment the

bus width may have to be set to 8 bits depending on the requirements of the external hardware.

IPST Post Idle
An idle cycle follows the current bus cycle, when the next bus cycle is in a different zone. For the 80-pin package where all zones are on-chip, this bit can be cleared to 0; as this additional idle-cycle is not required.

7.2.3 Static Zone 0 Configuration (SZCFG0) Register

The Static Zone 0 Configuration (SZCFG0) register is a word-wide, read/write register that sets the timing and bus characteristics of Zone 0 memory accesses. In the CR16MHS9 implementation of the CompactRISC architecture, Zone 0 is occupied by the flash EEPROM program memory.

The SZCFG0 register address is F904 hex. Upon reset, the register is initialized to 069F hex. The register format is shown below.

15	14	13	12	11	10	9	8
Reserved				FRE	IPRE	IPST	Reserved

7	6	5	4	3	2	1	0
BW	Note 1		HOLD		WAIT		

WAIT Memory Wait cycles
This field specifies the number of TIW (internal wait state) clock cycles added for each memory access, ranging from 000 binary for no additional TIW wait cycles to 111 binary for seven additional TIW wait cycles. These bits are ignored if the SZCFG0.FRE bit is set to 1.

HOLD Memory Hold cycles
This field specifies the number of T_{hold} clock cycles used for each memory access, ranging from 00 binary for no T_{hold} cycles to 11 binary for three T_{hold} clock cycles. These bits are ignored if the SZCFG0.FRE bit is set to 1.

BW BW defines the external bus width for the zone. Bus width is initialized during reset to its default value, which is 16 bits. If this bit is cleared to 0 the external bus is 8 bits wide. To set to external bus width to 16 bits, this bit has to be set to 1. For the 80 pin package the Bus width has to be set to 1.

IPST Post Idle
An idle cycle follows the current bus cycle, when the next bus cycle is in a different zone. For the 80-pin package where all zones are on-chip, this bit can be cleared to 0 as this additional idle-cycle is not required.

IPRE Preliminary Idle
An idle cycle is inserted prior to the current bus cycle, when this bus cycle is in a new zone. For the 80-pin package where all zones are on-chip, this bit can be cleared to 0 as this additional idle-cycle is not required.

FRE Fast Read Enable
This bit enables (1) or disables (0) fast read bus cycles. A fast read operation takes one clock cycle. A normal read operation takes at least two clock cycles.

Note 1: These bits (bit 5 and 6) control the configuration to the 224-pin device used in emulation equipment. The CPU should clear these bits to 0 when it writes to the register.

7.3 WAIT AND HOLD STATES USED

The number of wait cycles and hold cycles inserted into a bus cycle depends on whether it is a read or write operation, the type of memory or I/O being accessed, and the control register settings.

7.3.1 Flash Program Memory

When the CPU accesses the flash program memory (address 0000-BFFF hex), the number of added wait and hold cycles depends on type of accesses and the BIU register settings.

For a read operation in fast read mode (SZCFG0.FRE=1), no wait cycles or hold cycles are used.

For a read operation in normal read mode (SZCFG0.FRE=0), the number of inserted wait cycles is one plus the value written to the SZCFG0.WAIT field. The number in this field can range from zero to seven, so the total number of wait cycles can range from one to eight. The number of inserted hold cycles is equal to the value written to the SCCFG0.HOLD field, which can range from zero to three.

For a write operation in fast read mode (SZCFG0.FRE=1), the number of inserted wait cycles is one. No hold cycles are used.

For a write operation normal read mode (SZCFG0.FRE=0), the number of wait cycles is equal to the value written to the SZCFG0. WAIT field plus one (in the late write mode) or two (in the early write mode). The number of inserted hold cycles is equal to the value written to the SCCFG0.HOLD field, which can range from zero to three.

Writing to the flash program memory is a ROM programming operation that requires some additional steps, as explained in Section8.3.2.

7.3.2 RAM Memory

When the CPU accesses RAM memory (address E000-E7FF hex), no wait cycles or hold cycles are used.

7.3.3 EEPROM Data Memory

There is either no wait state or one wait state used when the CPU accesses the EEPROM data memory (address F000-F27F hex). The number of required wait states (zero or one) depends on the CPU clock frequency and operating mode, and is controlled by programming of the FEEDM bit in the MCFG register, as explained in Section8.3. No hold cycles are used.

7.3.4 Core Register and Peripheral Accesses

When the CPU accesses core registers and on-chip peripherals in the range of F000-FFFF, one wait cycle is used. No hold cycles are used.

For the FB00-FBFF (Ports B and C) the IOCFG register determines the timing.

7.3.5 Wait/Hold Summary Table

Table8 is a summary showing the number of wait cycles and hold cycles used for various address ranges.

Table 8 Access Timing Table

Address Range (hex)	Memory or I/O Type	Wait Cycles Added	Hold Cycles Added
0000-BFFF	Flash Program Memory	For read operation: 0 if SZCFG0.FRE=1, or 1+SZCFG0.WAIT if SZCFG0.FRE=0; For write operation: 1+BCFG.EWR if SZCFG0.FRE=1, or 1+BCFG.EWR+SZCFG0.WAIT if SZCFG0.FRE=0	0 if SZCFG0.FRE=1, or SZCFG0.HOLD if SZCFG0.FRE=0
E000-E7FF	Static RAM Memory	0	0
F000-F27F	EEPROM Data Memory	For read operation: 0 if MCFG.FEEDM=1, or 1 if MCFG.FEEDM=0	0
F900-FFFF F800-F9FF FC00-FFFF	Core Registers And On-Chip Peripherals	1	0
FB00-FBFF	Ports B and C	For read operation: 1+IOCFG.WAIT For write: 1+BCFG.EWR+IOCFG.WAIT	IOCFG.HOLD

7.3.6 Recommended Register Settings

Table9 shows the recommended register settings for various clock rates. Different clock rates require different register settings because the flash program memories have specific set-up and hold requirements that can be met only by using enough wait cycles and hold cycles.

Table 9 Recommended Register Settings

Clock Rate	SZCFG0	IOCFG
< 12.5 MHz, 0 wait state	0E80 hex	0288 hex
12.5 to 16 MHz, 0 wait state	0E80 hex	0288 hex
12.5 to 16 MHz, 1 wait state	0680 hex	0288 hex
> 16 MHz, 1 wait state	0680 hex	0288 hex

8.0 Memory

The CompactRISC architecture supports a uniform linear address space of 2 megabytes, addressed by 21 bits. The device implementation of this architecture uses only the lowest 64 kbytes of address space, addressed by 16 bits. Each memory location contains a byte consisting of eight bits.

Four types of on-chip memory occupy specific intervals within the address space: 48 kbytes of flash program memory, 1.5 kbytes of ISP memory, 2 kbytes of static RAM, and 640 bytes of EEPROM data memory. All of these memories are 16 bits wide, but their contents can be accessed either as bytes (eight bits wide) or words (16 bits wide).

The CPU core uses the Load and Store instructions to access memory. These instructions can operate on bytes or words. For a byte access, the CPU operates on a single byte occupying a specified memory address. For a word access, the CPU operates on two consecutive bytes. In that case, the specified address refers to the least significant byte of the data value; the most significant byte is located at the next higher address. Thus, the ordering of bytes in memory is from least to most significant byte. For more efficient data access operations, 16-bit variables should be stored starting at word boundaries (at even address).

8.1 FLASH PROGRAM MEMORY

The flash program memory is used to store the application program. The 48 kbytes of this memory reside in the address range of 0000-BFFF hex. A normal CPU write operation to this memory has no effect.

The program memory has the following features:

- 48 kbytes arranged as 24K by 16 bits
- Page size of 64 words, divided into two rows of 32 words
- 30 µs programming pulse per word
- Page mode erase with a 1 ms pulse
- Programming high voltage and timing generated on-chip
- Boot-ROM controlled in-system programming capability
- User-coded programming capability
- Security features to limit read/write access

8.1.1 Reading

Program memory read accesses can operate without wait cycles with a CPU clock rate of up to 12.5 MHz in the normal mode. At higher clock rates, memory read accesses can operate with one wait state.

The programmed number of wait cycles used (either zero or one) is controlled by the BIU Configuration (BCFG) register and the Static Zone 0 Configuration (SZCFG0) register. These registers are described in Section7.2.1 and 7.2.3.

8.1.2 Conventional Programming Modes

The flash program memory can be programmed either with device plugged into an EPROM programmer unit (external programming) or with the device installed in the application system (in-system programming). The device internally generates the necessary voltages for programming. No additional power supply is required.

Programming the memory requires placing the device in a special programming mode. Programming commands issued while the device is in its normal operating mode are ignored.

To enter one of the programming modes, the device must be reset with the $\overline{\text{ENV1}}$ and $\overline{\text{ENV0}}$ pins configured to select the desired mode: Test Mode for external programming or In-System Programming mode for in-system programming. Once the device enters the programming mode, the programming software performs the task of downloading the program code through a serial port and writing the code to the EEPROM memory. The software used for programming the EEPROM resides outside of the device for external programming or in the on-chip boot ROM for conventional in-system programming.

8.1.3 User-Coded Programming Routines

Instead of using an EPROM programmer unit or the conventional in-system programming mode, you can write your own processor code to program and erase the flash program memory. Writing your own code is more flexible than using the other programming methods. Like the conventional in-system programming mode, you can program the device while it is installed in the system, but no PC is required to download the program data. It is not necessary to reset the device or use the $\overline{\text{ENV0}}$/$\overline{\text{ENV1}}$ pins to configure the device.

If you write your own flash EEPROM programming code, the code must reside in the 2 kbytes of RAM memory, in the address range of E000-E7FF. This is because the entire flash program memory becomes unavailable when you program or erase any part of that memory.

Also, for programming the flash program memory, you need to design a protocol for the device to obtain the programming data. For example, you can use the on-chip USART to obtain the programming data from an external device through a serial interface.

The flash program memory is divided into 384 pages, each page containing 64 words (each 16 bits wide). Each page is further divided into two rows of 32 words. Erasure is carried out one page at a time, whereas programming is carried out one row (or one partial row) at a time.

Once an erase or programming operation is started, the PGMBUSY bit in the MSTAT register is automatically set, and then cleared when the operation is complete. All high-voltage pulses and timing needed for programming and erasing are provided internally. The program memory cannot be accessed while the PGMBUSY bit is set.

8.1.4 Flash EEPROM Programming and Verify

The flash EEPROM program memory programming and erase can be performed using different methods. It can be done through user code that is stored in system RAM, or through In-System-Programming mode, but should not be programmed through the flash EEPROM program memory itself as no instruction or data can be fetched from it while it is being programmed. All program and erase operations must be preceded immediately by writing the proper key to the program memory key register PGMKEY.

The flash EEPROM program memory is divided into 384 pages, each page containing 64 words (each 16 bits wide). Each page is further divided into two adjacent rows. A page erase will erase one page. Programming is done by writing to all the words within a row, one word following another sequen-

tially within one single high voltage pulse. This is supported through a double-buffered write-data buffer scheme. Byte programming is not supported. Programming should be done on erased rows.

A page erase requires the following code sequence (assuming that this sequence will not be interrupted to do another flash erase or programming):

1. Check for MSTAT.PGMBUSY not set.
2. Set FLCSR.ERASE = 1.
3. If interrupt was enabled, disable interrupt.
4. Write proper key value to PGMKEY.
5. Write to any valid location within the page to be erased.
6. If interrupt was disabled in step 3, re-enable interrupt.
7. Set FLCSR.ERASE = 0.

When programming, the data to be written into the flash EEPROM program memory is first written into a double-buffered write-data buffer. When a piece of data is written to the page while the flash EEPROM program memory is idle, the write cycle will start. Due to the double-buffered nature of the write-data buffer, a second word can be written to the flash EEPROM program memory. This will then set FLCSR.PMLFULL flag indicating the buffer is now full. When the first write is done, the memory address would be incremented, and the second word would be written to that address while keeping the high voltage pulse active; the FLCSR.PMLFULL flag is cleared. Another word can then be written to the buffer, and this programming will repeat until there are no more words to be programmed. This allows pipelined writes to different words on the same row within the same high voltage pulse. If the programming sequence exceeds a row, the flash programming interface will automatically initiate a programming pulse for the next row. The FLCSR.PMLFULL bit is also cleared. when programming of the last word of the current row is completed, e.g. programming of the entire row is completed and MSTAT.PGMBUSY is cleared. This means, the separation of the program memory into rows is transparent to the user, as the transition is handled by the flash program memory interface. Figure2 shows a flowchart for a programming sequence.

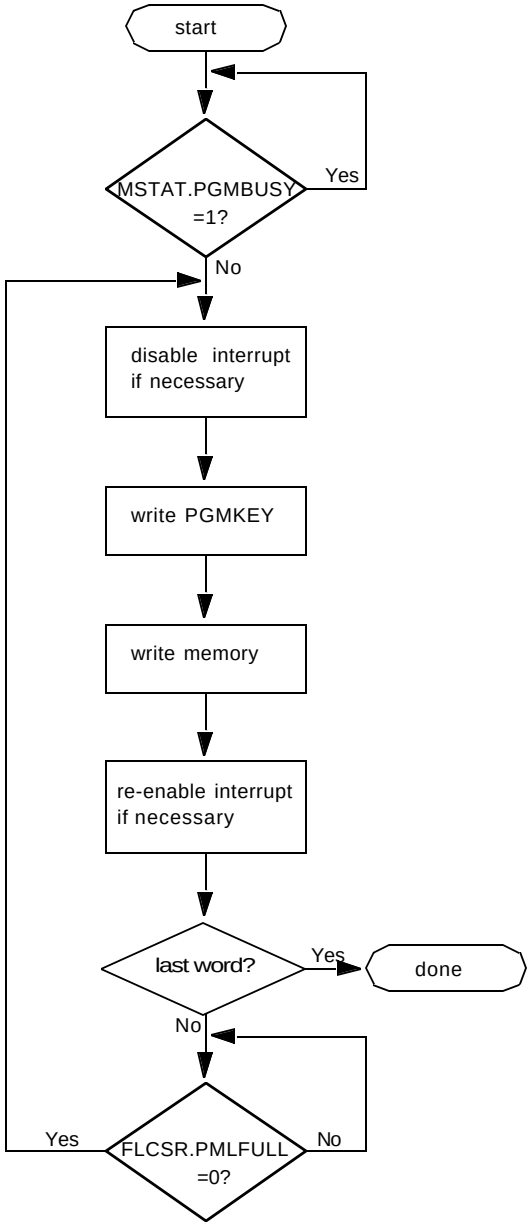


Figure 2. Programming Sequence for the Program Memory

Erase Procedure

Erasing a page requires the following code sequence:

1. Verify that the MSTAT.PGMBUSY bit is cleared.
2. Disable any enabled interrupts.
3. Set the FLCSR.ERASE bit to 1.
4. Write the proper key value to the PGMKEY register.
5. Write to any valid location in the page to be erased.
6. Clear the FLCSR.ERASE bit to 0.
7. Re-enable any interrupts disabled in Step 2.

Erased bits read back as 1.

Programming Procedure

Programming is done by writing to all the words within a row, one word following another sequentially, within a single high-voltage pulse for the entire row. It is not possible to program single bytes. Programming should be done only on erased rows.

Programming a row requires the following code sequence:

1. Verify that the MSTAT.PGMBUSY bit is cleared.
2. Disable any enabled interrupts.
3. Write the proper key value to the PGMKEY register.
4. Write the desired word to the lowest address of the row.
5. Wait until the FLCSR.PMLFULL bit is cleared to 0.
6. Write the proper key value to the PGMKEY register.
7. Write the desired word to the next address of the row.
8. Repeat Steps 5, 6, and 7 until the whole row is written.
9. Re-enable any interrupts disabled in Step 2.

The programmed values can be verified by normal read operations from the applicable memory locations.

Each time you write a word to the flash program memory using the procedure just described, the data word is stored in a two-level write-data buffer. The first word written while the program memory is idle starts the write cycle. Due to double-buffering of the write path, a second word can be written to the program memory while the first word is still being programmed. Writing the second word sets the FLCSR.PMLFULL flag, thus indicating that the write-data buffer is full.

When programming of the first word is done, the memory address is incremented, programming of the second word starts, and the PLCSR.PMFULL flag is cleared. The next word can then be written to the write-data buffer. This process is repeated until there are no more words in the buffer or the entire row has been programmed. All of this happens within the same high-voltage programming pulse.

If a reset occurs in the middle of an erase or programming operation, the operation is terminated. The reset is extended until the flash memory returns to the idle state.

8.1.5 Erase and Programming Timing

The internal hardware of the device handles the timing of erase and programming operations. To drive the timing control circuits, the device divides the system clock by a programmable prescaler factor. You should select a prescaler value to produce a program/erase clock of 200 kHz (or as close as possible to 200 kHz without exceeding 200 kHz). For the timing control circuit to operate correctly, you must program the prescaler value in advance and leave it unchanged while a program or erase operation is in progress. A similar (but separate) prescaler factor is applied to the EEPROM data memory. See Section8.1.7 and Section8.3.4 for details.

8.1.6 Flash Program Memory Control and Status Register (FLCSR)

The Flash Program Memory Control and Status (FLCSR) register is a byte-wide, read/write register that contains several status and control bits related to the program memory. Upon reset, this register is cleared to zero when the flash memory on the chip is in the idle state.

The register format is shown below.

7	6	5	4	3	2	1	0
Reserved			IENPROG	PMLFULL	PMBUSY	ERASE	Reserved

ERASE Erase Flash Program Memory Page. When set (1), a write to the program memory erases the whole page containing the addressed memory location. When cleared (0), a write to the program memory either has no effect on programming the memory word (if properly set up for programming). This bit should be changed only when the flash memory is not busy being programmed or erased.

PMBUSY Program Memory Busy. This bit is automatically set to 1 when the flash program memory is busy being programmed, and cleared to 0 at all other times. (The MSTAT.PGMBUSY is also set to 1 whenever the PMBUSY bit is set to 1.)

PMLFULL Program Memory Write-Latch Buffer Full. When set (1), the double-buffered data register for program memory write operations is full. When cleared (0), the double-buffered data register is not full.

IENPROG Interrupt Enable for Program Memory Write. When set (1), the flash program memory write mechanism generates an interrupt whenever the PMLFULL bit changes from 1 to 0. This is a signal to the CPU to write the next word for programming. When IENPROG is cleared, no such interrupt is generated.

8.1.7 Program Memory Timing Prescaler Register (FLPSLR)

The FLPSLR register is a byte-wide, read/write register that selects the prescaler divider ratio for the flash program memory programming clock. Before you program or erase the program memory for the first time, you should program the FLPSLR register with the proper prescaler value, an 8-bit value called FTDIV. The device divides the system clock by (FTDIV+1) to produce the program memory programming clock.

You should choose a value of FTDIV to produce a clock of the highest possible frequency that is equal to or just less than 200 kHz. For example, if the system clock frequency is 12.5 MHz, use the value 3E hex (62 decimal) for FTDIV, because 12.5 MHz / (62+1) = 198.4 kHz. Do not modify this register while a flash program or erase operation is in progress.

Upon reset, this register is programmed by default with the value 33 hex (51 decimal), which is an appropriate setting for a 10.4 MHz system clock.

8.1.8 Program Memory Write Key Register (PGMKEY)

The PGMKEY register is a byte-wide, write-only register that must be written with a key value (A3 hex) immediately prior to each write to the flash program memory. Otherwise, the write operation to the program memory will fail. This feature is intended to prevent unintentional programming of the program memory.

Reading this register always returns FF hex.

8.2 RAM MEMORY

The static RAM memory is used for temporary storage of data and for the program and interrupt stacks. The 2 kbytes of this memory reside in the address range of E000-E7FF hex. Each memory access requires one clock cycle, for a byte or word access. For non-aligned word access, each memory access requires multiple clock cycles. No wait cycles or hold cycles are required.

8.3 EEPROM DATA MEMORY

The EEPROM data memory is used for non-volatile storage of data. The 640 bytes of this memory reside in the address range of F000-F27F hex. The CPU reads or writes this memory by using ordinary byte-wide or word-wide memory access commands.

8.3.1 Reading

EEPROM data memory read accesses can operate without wait cycles with a CPU clock rate of up to 12.5 MHz in the normal mode. At higher clock rates, read accesses can operate with one wait state.

The programmed number of wait cycles used (either zero or one) is controlled by a bit in the Module Configuration register (MCFG.FEEDM). This register is described in Section4.2.

8.3.2 Programming

Before you begin programming the EEPROM data memory, you should set the value in the EEPROM Data Memory Prescaler register. This register sets the prescaler used to generate the data memory programming clock from the system clock, as described in Section8.3.4.

After the CPU performs a write to the EEPROM data memory, the on-chip hardware completes the EEPROM programming in the background. When programming begins, the on-chip hardware sets the DMCSR.DMBUSY bit to 1, and also sets the MSTAT.PGMBUSY bit to 1. When programming is completed, it resets these status bits back to 0. Once the software writes to the EEPROM data memory, it should not attempt to access the EEPROM data memory again until programming is completed and the status bit is reset to 0.

The device hardware internally generates the voltages and timing signals necessary for programming. No additional power supply is required, nor any software required except to check the status bit for completion of programming. The minimum time required to erase and reprogram a byte or word is 1.16 ms. The programmed values can be verified by using normal memory read operations.

If a reset occurs during a programming or erase operation, the operation is terminated. The reset is extended until the flash memory returns to the idle state.

The EEPROM data memory does not have permanent read-protection or write-protection features like those available for the EEPROM program memory. However, the Data Memory Write Key Register provides a way to “lock” the data written to the data memory.

8.3.3 Data Memory Control and Status Register (DMCSR)

The DMCSR register is a byte-wide, read/write register used with the EEPROM data memory. There are two status/control bits, as shown in the register format below.

7	6	5	4	3	2	1	0
Reserved					DMBUSY	ERASE	Reserved

ERASE — Erase ISP Flash Program Memory Page. When set (1) a valid write to the ISP flash EEPROM program memory will erase the entire ISP flash EEPROM program memory page pointed to by the write address rather than performing a write to the addressed memory location. This bit should be cleared to 0 and remain cleared after the write operation.

DMBUSY — Data Memory Busy. This bit is automatically set to 1 when the EEPROM data memory is busy being programmed, and cleared to 0 at all other times. (The MSTAT.PGMBUSY is also set to 1 whenever the DMBUSY bit is set to 1.)

Upon reset, the DMCSR register is cleared to zero when the flash memory on the chip is in the idle state.

8.3.4 Data Memory Prescaler Register (DMPSLR)

The DMPSLR register is a byte-wide, read/write register that selects the prescaler divider ratio for the EEPROM data memory programming clock. Before you write to the data memory for the first time, you should program the DMPSLR register with the proper prescaler value, an 8-bit value called FTDIV. The device divides the system clock by (FTDIV+1) to produce the data memory programming clock.

You should choose a value of FTDIV to produce a clock of the highest possible frequency that is equal to or just less than 200 kHz. Upon reset, this register is programmed by default with the value 33 hex (51 decimal), which is an appropriate setting for a 10.4 MHz system clock.

8.3.5 Data Memory Write Key Register (DMKEY)

The DMKEY register is a byte-wide, read/write register that provides a way to “lock” the data contained in the EEPROM data memory. Upon reset, the register is automatically set to C9 hex, which is the key value. Writing to the EEPROM data memory is allowed as long as the DMKEY register contains this value. When the register contains any value other than C9 hex, writing the EEPROM data memory is disallowed.

To “lock” the current data stored in the data memory, write another value (such as 00 hex) to the DMKEY register. To “unlock” the data memory, write the value C9 hex to the DMKEY register.

Note: Operation of this register is different from the PGMKEY register used with the program memory. It is not necessary to write the key value to DMKEY every time you write to the data memory.

8.4 ISP MEMORY

The In-System Program memory is part of the flash memory array that contains the flash EEPROM data memory. It is not possible to access the ISP memory while programming the

flash EEPROM data memory or access the flash EEPROM data memory while programming the ISP memory. The 1.5 kbytes of ISP memory resides in the address range of DA00-DFF7 and is used for storing a boot ROM. The ROM contains the code that performs in-system programming, and is programmed at the factory. In ISP mode, code execution starts at address DA00.

The ISP program memory and flash EEPROM data memory share the same memory array, which makes it impossible to access one type of memory while the other is being programmed.

The ISP memory has the following features:

— 1.5 kbytes ISP flash EEPROM program memory
— Page size of 4 words, divided into two rows of 2 words each
— Odd and even bytes within a page can be erased separately
— The lowest 384 rows are the ISP memory, the others are data memory; A1 selects the columns and A[11:2] select the rows
— 30µs programming pulse width per word
— Page mode erase with 1ms pulse
— All erased memory bits read 1
— Fast read access time
— Requires valid key for program and erase to proceed
— Provide memory protection and security features for flash EEPROM program memory
— Security features may limit accesses to ISP memory
— Disable memory when address is out of range to prevent accessing data memory
— Provide busy status during programming and erase
— Read/write accesses disabled during programming/erase
— Programming high voltage and timing generated on-chip
— Support flash memory test mode with PADX for security overrides

8.4.1 Reading

The ISP flash EEPROM program memory read accesses can operate without wait cycles with a CPU clock rate of up to 20MHz in the normal mode. At higher clock rates, read accesses can operate with one wait state.

The programmed number of wait cycles used (either zero or one) is controlled by BIU Configuration (BCFG) register and the Static Zone 0 Configuration (SZOFG0) register. These registers are described in Section7.2.1 and 7.2.3.

8.4.2 User-Coded Programming Routines

All program and erase operations must be preceded by writing the proper key to the program memory key register ISPKEY. The programming code can reside in the in-system RAM, but cannot reside in the ISP flash EEPROM program memory or flash EEPROM data memory as accesses within these ranges are not permitted while ISP flash EEPROM program memory is being programmed.

The ISP flash memory is divided into 192 pages, each page containing 4 words (each 16 bits wide). Each page is further divided into two rows. Erase is carried out one page at a time, whereas programming is carried out one row (or one partial row) at a time.

Once an erase or programming operation is started, the PGMBUSY bit in the MSTAT register is automatically set, and then cleared when the operation is complete. All high-voltage pulses and timing needed for programming and erasing are provided internally. The program memory cannot be accessed while the PGMBUSY bit is set.

Erase Procedure

Erasing a page requires the following code sequence:

1. Verify that the MSTAT.PGMBUSY bit is cleared.
2. Set the DMCSR.ERASE bit to 1.
3. Locally disable interrupts.
4. Write proper key value to the ISPKEY register.
5. Write to any valid page to be erased.
6. Re-enable interrupts disabled in Step 3.
7. Set the DMCSR.ERASE bit to 0.

8.4.3 Programming Procedure

Programming is done by writing one byte or word at a time and should be done on already erased memory.

Programming the ISP flash EEPROM program memory requires the following code sequence:

1. Verify that the MSTAT.PGMBUSY bit is cleared.
2. Locally disable interrupts.
3. Write proper key value to the ISPKEY register.
4. Write a byte or word to the addressed location.
5. Re-enable interrupts disabled in Step 2.

Programmed values can be verified through normal read operations.

If a reset occurs in the middle of an erase or programming operation, the operation is terminated. The rest is extended until the flash EEPROM memory returns to the idle state.

8.4.4 Erase and Programming Timing

The program and erase timing are controlled by the flash EEPROM data memory logic.

8.4.5 Protection Features

The last byte of the ISP memory are reserved for special functions and provide memory protection and security for the flash EEPROM program memory. Read and write protection is provided.

8.4.6 Program Memory Security Features

The program memory has security features to prevent unauthorized access to the program code and unintentional programming. These features are invoked by programming a non-volatile control register, the Flash EEPROM Security (FLSEC) register. This register contains a read-access control bit and a write-access control bit.

Both control bits are initially set to 1. Programming the read-access bit to 0 prevents outside access to the program code; any attempt to access the code from outside returns all zeros. Programming the write-access bit to 0 prevents any further programming of the flash program memory, thus protecting the program code from overwriting.

Programming either bit to 0 (or both bits to 0) also prevents any further changes to the FLSEC register itself. Thus, invoking read protection and/or write protection is permanent.

The Flash Memory Security (FLSEC) register format is shown below.

7	6	5	4	3	2	1	0
Reserved					FROMWR	Reserved	FROMRD

FROMRD When this bit is set to 1, the flash program memory can be read from outside the device. When this bit is cleared to 0, the flash program memory cannot be read from outside the device, and the FLSEC register itself is write-protected.

FROMWR When this bit is set to 1, the flash program memory can be erased and overwritten by subsequent programming operations. When this bit is cleared to 0, the program memory and the FLSEC register itself are both write-protected.

9.0 Interrupts

The Interrupt Control Unit (ICU) receives interrupt requests from internal and external sources and generates interrupts to the CPU. Interrupts from the timers, USARTs, MICROWIRE/SPI interface, Multi-Input Wake-Up, and A/D converter are all maskable interrupts. The highest-priority interrupt is the Non-Maskable Interrupt (NMI), which is triggered by a falling edge received on the $\overline{\text{NMI}}$ input pin. The $\overline{\text{NMI}}$ pin is not available on the 44-pin packages.

9.1 INTERRUPT OPERATION

An *exception* is an event that temporarily stops the normal flow of program execution and causes execution of a separate service routine. Upon completion of the service routine, execution of the interrupted program continues from the point at which it was stopped.

There are two kinds of exceptions, called *traps* and *interrupts*. A trap is the result of some action or condition in the program itself, such as execution of an Exception (EXCP) instruction. An interrupt is a CPU-external event, such as a signal received on a Multi-Input Wake-Up input or a request from an on-chip peripheral module for service.

The operation of traps is beyond the scope of this data sheet. For information on traps, and for additional detailed information on interrupts not provided in this data sheet, please refer to the CompactRISC CR16B Programmer's Reference Manual.

9.1.1 Interrupt Operation Summary

When an interrupt occurs, the on-chip hardware performs the following steps:

1. Decrements the Interrupt Stack Point (ISP) by four.
2. Saves the contents of the Program Counter (PC) and Processor Status Register (PSR) on the interrupt stack.
3. Clears the I, P, and T bits in the Processor Status Register (PSR). These are the Global Maskable Interrupt Enable bit, Trace Trap Pending bit, and Trace bit, respectively.
4. Reads the interrupt vector from the Interrupt Vector Register (IVCT).
5. Combines the interrupt vector with the value in the Interrupt Base (INTBASE) register to obtain an address in the Interrupt Dispatch Table, and loads the dispatch table entry into the Program Counter (PC).

From this point onward, the CPU executes the interrupt service routine. The service routine ends with a Return from Exception (RETX) instruction. This returns the CPU to the interrupted program. The CPU restores the contents of the PC and PSR registers from the stack and increments the Interrupt Stack Pointer by four.

9.1.2 Service Routine Addresses

When an interrupt or trap occurs, the CPU executes a service routine. There are different service routines for different interrupts and traps. Each service routine may reside anywhere in program memory. The starting addresses of the service routines are contained in a table called the Dispatch Table. Entries in the table are organized in the order shown in Table10.

Table 10 Dispatch Table Entries

0: Reserved
1: NMI
2: Reserved
3: Reserved
4: Reserved
5: SVC (Supervisor Call Trap)
6: DVC (Divided by Zero Trap)
7: FLG (Flag Trap)
8: BPT (Breakpoint Trap)
9: TRC (Trace Trap)
10: UND (Undefined Instruction Trap)
11: Reserved
12: Reserved
13: Reserved
14: Reserved
15: Reserved
16: INT0 (Reserved)
17: INT1 (A/D Converter)
18: INT2 (Multi-Input Wake-Up)
19: INT3 (Reserved)
20: INT4 (USART2 Tx)
21: INT5 (USART1 Tx)
22: INT6 (MICROWIRE/SPI Rx/Tx)
23: INT7 (Reserved)
24: INT8 (USART2 Rx)
25: INT9 (USART1 Rx)
26: INT10 (Timer 2 Input B)
27: INT11 (Timer 2 Input A)
28: INT12 (Timer 1 Input B)
29: INT13 (Timer 1 Input A)
30: INT14 (Timer 0)
31: INT15 (Reserved)

Each entry in the Dispatch Table consists of two bytes that provide bits 1 through 16 of the starting address of the corresponding service routine. The full 21-bit address of a service routine is reconstructed by adding a leading 0 and a trailing 0 to the 16-bit table entry. Because the program memory of the device only occupies the range of 0000-BFFF hex, entries in the table are restricted to this range.

The INTBASE register is a pointer to the Dispatch Table. Upon reset, the initialization software must write the starting address of the Dispatch Table to the INTBASE register, a 21-bit register with the five most significant bits and the least significant bit always equal to 0. It is typically kept in the flash EEPROM program memory. The Dispatch Table is 32 words long.

Each interrupt or trap source has an associated vector number ranging from 0 to 31, as indicated in Table10. When an interrupt occurs, the hardware multiplies the vector by 2, adds the result to the contents of the INTBASE register, and uses the resulting address to obtain the service routine starting address from the corresponding entry in the Dispatch Table. This address is placed in the Program Counter so that the CPU begins executing the interrupt service routine.

Figure3 summarizes the method used by the device to generate the starting address of a service routine.

9.1.3 Stack Usage

When an interrupt occurs, the CPU automatically preserves the contents of the Program Counter (PC) and Processor Status Register (PSR) by pushing them on the interrupt stack and decrementing the Interrupt Stack Pointer by four. The service routine ends with a Return from Exception (RETX) instruction, which returns control to the interrupted program by restoring the PC and PSR values and incrementing the Interrupt Stack Pointer (ISP) by four.

Prior to using any interrupts, the Interrupt Stack Pointer (ISP) must be initialized so that it points to a space in RAM where the interrupt stack will be kept. The stack grows downward in memory (toward address zero) when an interrupt occurs and items are pushed onto the stack. The stack shrinks upward in memory when an interrupt service routine ends and items are popped from the stack.

Many routines need to use the general-purpose registers R0 through R13. To preserve the existing register contents, a routine can save register contents on the program stack upon start of the routine and restore the register contents prior to completion of the routine. The software can also use the program stack to transfer data parameters from one routine to another when the parameters are too large to easily fit into the scratch registers (large structures, large arrays, or a set of more than four parameters in a single routine). A high-level language typically allocates the local (non-static) variables on the stack.

The stack pointer for the program stack is the SP register, which must be initialized prior to any register save/restore operations or data transfer operations. Using the program stack, an interrupt routine needs to initially save the contests of all registers that it uses, and restore those register contents before returning to the interrupted program.

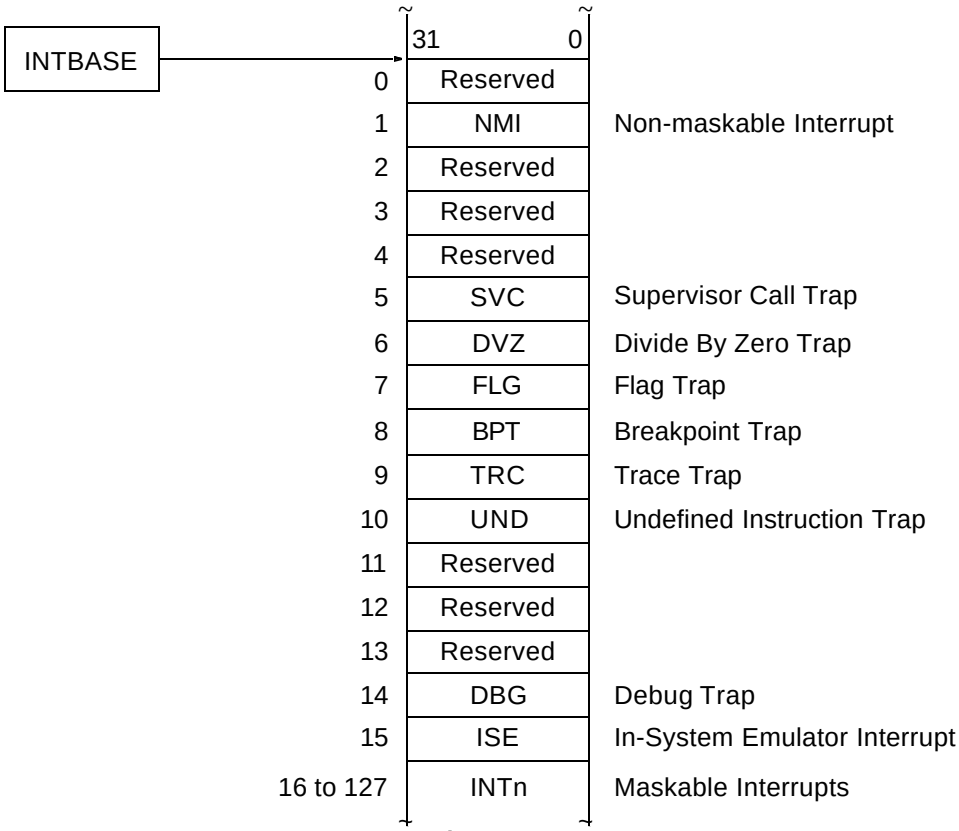


Figure 3.

9.2 NON-MASKABLE INTERRUPT

A non-maskable interrupt is triggered by a falling edge on the $\overline{\text{NMI}}$ input pin, which generates a software trap. The $\overline{\text{NMI}}$ pin is an asynchronous input with Schmitt trigger characteristics and an internal synchronization circuit. Therefore, no external synchronizing is needed.

Upon reset, the non-maskable interrupt is disabled and should remain disabled until the software initializes the interrupt table, interrupt base, and interrupt stack pointer. It can

be enabled by setting either of two control bits in the External NMI Control/Status (EXNMI) register. The two bits are called the EN (Enable) bit and the ENLCK (Enable and Lock) bit.

The EN bit enables the NMI trap until an NMI trap event or a reset occurs. An NMI trap automatically resets the EN bit. Using this bit to enable the NMI trap is intended for applications where the $\overline{\text{NMI}}$ pin is toggled frequently but nested NMI traps are not needed. The trap service routine should re-enable the NMI trap by setting the EN bit before returning to the main program.

The ENLCK bit enables the NMI trap and locks it in the enabled state. In other words, it leaves the NMI trap enabled even after the trap occurs. It can be cleared only by a reset operation. After the bit is set, an NMI trap is triggered by each falling edge on the $\overline{\text{NMI}}$ pin, allowing nested NMI traps.

To use the EN bit, the ENLCK must remain cleared to 0. Otherwise, the EN bit is ignored.

9.3 MASKABLE INTERRUPTS

Maskable interrupts can be enabled or disabled under software control. There are 16 maskable interrupt sources (including some reserved for future expansion), organized into levels of priority. If more than one interrupt event occurs at any given time, the interrupt source with the highest priority is serviced first. The others must wait until the highest-priority interrupt is serviced and is no longer pending.

Figure11 lists the maskable interrupt sources of the device in order of priority, from the highest-priority interrupt (IRQ15) to the lowest (IRQ0).

Table 11 Maskable Interrupt Priority List

Interrupt Request	Source
IRQ15	Reserved (highest priority)
IRQ14	Timer 0
IRQ13	Timer 1 Input A
IRQ12	Timer 1 Input B
IRQ11	Timer 2 Input A
IRQ10	Timer 2 Input B
IRQ9	USART1 Rx
IRQ8	USART2 Rx
IRQ7	Reserved
IRQ6	MICROWIRE/SPI Rx/Tx
IRQ5	USART1 Tx
IRQ4	USART2 Tx
IRQ3	Reserved
IRQ2	Multi-Input Wake-Up
IRQ1	A/D Converter
IRQ0	Reserved (lowest priority)

To enable a maskable interrupt, the enable bit must be set in the applicable peripheral module and also in the appropriate Interrupt and Enable Mask register, IENAM0 or IENAM1. In addition, both the Global Maskable Interrupt Enable bit (I) and the Local Maskable Interrupt Enable bit (E) must be set to 1 in the PSR register. If either one of these bits is 0, then all maskable interrupts are disabled.

Both the E bit and I bit can be controlled with the Load Processor Register (LPR) instruction. In addition, the E bit is easily changed by executing the Enable Interrupts (EI) or Disable Interrupts (DI) instruction. Using the EI and DI instructions avoids the possibility of an interrupt occurring within a read-modify-write operation on the PSR register.

For more information on the PSR bits, see Section6.3.

9.4 INTERRUPT REGISTERS

The Interrupt Control Unit uses the following interrupt control and status registers:

- Non-Maskable Interrupt Status Register (NMISTAT)
- External NMI Control/Status Register (EXNMI)
- Interrupt Enable and Mask Register 0 (IENAM0)
- Interrupt Enable and Mask Register 1 (IENAM1)
- Interrupt Vector Register (IVCT)
- Interrupt Status Register 0 (ISTAT0)
- Interrupt Status Register 1 (ISTAT1)

The following CPU core registers are also used in processing interrupts:

- Interrupt Stack Pointer (ISP)
- Interrupt Base Register (INTBASE)
- Processor Status Register (PSR)

9.4.1 Non-Maskable Interrupt Status Register (NMISTAT)

The NMISTAT register is a byte-wide, read-only register that holds the current pending status of the Non-Maskable Interrupt (NMI). This register is cleared upon reset. It is also cleared each time it is read. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved							EXT

EXT — External Non-Maskable Interrupt Request. When set to 1 by the hardware, it indicates an external Non-Maskable Interrupt request has occurred. See the description of the EXNMI register below for more information.

9.4.2 External NMI Control/Status Register (EXNMI)

The EXNMI register is a byte-wide, read/write register that shows the current state of the $\overline{\text{NMI}}$ pin and also allows the NMI trap to be enabled by setting either the EN bit or the ENLCK bit. Both of these bits are cleared upon reset. When the software writes to this register, it must write 0 to all reserved bit positions for the device to function properly. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved					ENLCK	PIN	EN

EN — Enable NMI Trap. When set to 1, $\overline{\text{NMI}}$ traps are enabled and falling edge on the $\overline{\text{NMI}}$ pin generates a NMI trap. Each occurrence of an NMI trap automatically clears the EN bit. The trap service routine should set the EN bit to 1 before returning control to the interrupted program. When EN is cleared to 0, NMI traps are disabled unless they are enabled with the ENLCK bit. When the ENLCK bit is set to 1, the EN bit is ignored.

PIN — $\overline{\text{NMI}}$ Pin. This bit shows the current state of the $\overline{\text{NMI}}$ input pin (without logical inversion). A 1 indicates a high level and a 0 indicates a low level on the pin. This is a read-only bit. In a write operation, the value written to this bit position is ignored.

ENLCK — Enable and Lock NMI Trap. When set to 1, NMI traps are enabled and locked in the enabled state. Each falling edge on the $\overline{\text{NMI}}$ pin generates a NMI trap, even if a previous NMI trap has occurred and is still being processed. When ENLCK is cleared to 0, NMI traps are disabled unless they are enabled with the EN bit.

9.4.3 Interrupt Vector Register (IVCT)

The IVCT register is a byte-wide, read-only register that contains the encoded value of the enabled and pending maskable interrupt with the highest priority. The on-chip hardware automatically updates this field whenever there is a change in the highest-priority enabled and pending maskable interrupt. The CPU reads this register during an interrupt acknowledge core bus cycle to determine where to begin executing the interrupt service routine. The register contents are guaranteed to be valid at that time. The register is not guaranteed to contain valid data during a hardware update operation. The register format is shown below.

7	6	5	4	3	2	1	0
0	0	0	INTVECT				

INTVECT — Interrupt Vector. This 5-bit field contains the encoded value of the enabled and pending maskable interrupt with the highest priority. For example, if interrupts IRQ1 and IRQ6 are both enabled and pending, the higher-priority interrupt is IRQ6. As a result, the 5-bit interrupt vector is 10110.

9.4.4 Interrupt Enable and Mask Register 0 (IENAM0)

The IENAM0 register is a byte-wide, read/write register that enables or disables the individual interrupts IRQ0 through IRQ7. The register format is shown below.

7	6	5	4	3	2	1	0
IENA(7:0)							

A bit set to 1 enables the corresponding interrupt. A bit cleared to 0 disables the corresponding interrupt. Upon reset, this register is initialized to FF hex.

9.4.5 Interrupt Enable and Mask Register 1 (IENAM1)

The IENAM0 register is a byte-wide, read/write register that enables or disables the individual interrupts IRQ8 through IRQ15. The register format is shown below.

7	6	5	4	3	2	1	0
IENA(15:8)							

A bit set to 1 enables the corresponding interrupt. A bit cleared to 0 disables the corresponding interrupt. Upon reset, this register is initialized to FF hex.

9.4.6 Interrupt Status Register 0 (ISTAT0)

The ISTAT0 register is a byte-wide, read-only register that indicates which maskable interrupt inputs to the ICU (IRQ0 through IRQ7) are currently active. The register format is shown below.

7	6	5	4	3	2	1	0
IST7	IST6	IST5	IST4	IST3	IST2	IST1	IST0

IST(0-7) — Interrupt Status bits. Each bit indicates the current status of an interrupt input to the ICU, corresponding to interrupts IRQ0 through IRQ7. A bit set to 1 indicates an active interrupt input, even when the interrupt is masked out by the IENAM0 register. A bit cleared to 0 indicates an inactive interrupt input.

9.4.7 Interrupt Status Register 1 (ISTAT1)

The ISTAT1 register is a byte-wide, read-only register that indicates which maskable interrupt inputs to the ICU (IRQ8 through IRQ15) are currently active. The register format is shown below.

7	6	5	4	3	2	1	0
IST15	IST14	IST13	IST12	IST11	IST10	IST9	IST8

IST(8-15) Interrupt Status bits. Each bit indicates the current status of an interrupt input to the ICU, corresponding to interrupts IRQ8 through IRQ15. A bit set to 1 indicates an active interrupt input, even when the interrupt is masked out by the IENAM0 register. A bit cleared to 0 indicates an inactive interrupt input.

9.4.8 Core Registers (PSR, INTBASE, and ISP)

Some of the CPU core registers are used for controlling interrupts: the Processor Status Register (PSR), the Interrupt Base Register (INTBASE), and the Interrupt Stack Pointer (ISP) register.

PSR Register

The Processor Status Register (PSR) is a 16-bit register that holds status information and selects the operating modes for the CPU core. Bit 9 and Bit 11 are the Local Maskable Interrupt Enable (E) bit and the Global Maskable Interrupt Enable (I) bit, respectively, as shown in Figure4. These bits are used to enable or disable maskable interrupts.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I	P	E	0	N	Z	F	0	0	L	T	C

Figure 4. Processor Status Register (PSR) Format

Both the E bit and I bit can be controlled with the Load Processor Register (LPR) instruction. The E bit can also be controlled by the Enable Interrupts (EI) and Disable Interrupts (DI) instructions. If the E and I bits are both set to 1, all maskable interrupts are accepted. Otherwise, only the nonmaskable interrupt is accepted.

Upon reset, the E bit is set to 1 and the I bit is cleared to 0.

The processor uses the I bit to block maskable interrupts while executing an interrupt handler. When an interrupt occurs, the processor saves the existing I bit (as part of the PSR) on the interrupt stack and then clears the current I bit to prevent further interrupts. When RETX is executed, the former I bit it restored (with the rest of the PSR), again enabling masked interrupts.

The E bit is intended to be used in a localized manner, allowing a process to operate without interruption for a short period while it accesses and modifies system variables and semaphores. The E bit can be set to 1 by executing the Enable Interrupt (EI) or cleared to 0 by executing the Disable Interrupts (DI) instruction. Using these two instructions avoids the possibility of an interrupt occurring within a read-modify-write operation on the PSR register.

INTBASE Register

The Interrupt Base Register (INTBASE) is a 21-bit register that holds the address of the Dispatch Table for interrupts and traps. The five most significant bits and the least significant bit of this register are always zero. The Dispatch Table is 32 words long, and limited to the flash program memory space. Thus, the Dispatch Table must start at an even address in the range of 0004 to BFC0 hex.

ISP Register

The Interrupt Stack Pointer (ISP) is a 21-bit register that points to the lowest address of the last item stored on the interrupt stack. The on-chip hardware modifies the stack contents and stack pointer when an interrupt or trap event occurs and upon completion of an interrupt or trap service routine.

The five most significant bits and the least significant bit of this register are always zero. Because the stack must reside in RAM and the device RAM occupies the address range of E000-E7FF hex, the interrupt stack is restricted to this range.

9.5 INTERRUPT PROGRAMMING PROCEDURES

The following subsections provide information on initializing the device for interrupts, clearing interrupts, and nesting interrupts.

9.5.1 Initialization

Upon reset, all interrupts are disabled. To program the device for interrupt operation and to enable interrupts, use the following procedure in the application software:

1. Set the interrupt stack pointer (ISP).
2. Load the INTBASE register so that it points to the base of the interrupt Dispatch Table in ROM.
3. Perform any required preparation steps for the interrupt service routines.
4. Initialize the peripheral devices that can generate interrupts and set their respective interrupt enable bits.
5. Use the Load Processor Register (LPR) instruction to set I bit in the PSR register.
6. When the device is ready to execute interrupts, set the E bit in the PSR register by executing the Enable Interrupts (EI) instruction.

Once maskable interrupts are enabled by setting the E and I bits, you can disable and re-enable all maskable interrupts by using the Enable Interrupts (EI) and Disable Interrupts (DI) instructions, which set and clear the E bit.

9.5.2 Clearing Interrupts

Clearing an interrupt request before it is serviced may cause a spurious interrupt because the CPU may detect an interrupt not reflected in the Interrupt Vector (IVCT) register. To ensure reliable operation, clear interrupt requests only while interrupts are disabled.

Changing the polarity of an interrupt input (for example, in the Multi-Input Wake-Up module) can cause a spurious interrupt, and therefore should be done only while interrupts are disabled.

For the same reason, clearing an enable bit in a peripheral module should be carried out only while the interrupt is disabled.

9.5.3 Nesting Interrupts

Interrupts may be nested, or in other words, an interrupt service routine can itself be interrupted by a different interrupt source. There is no hardware limitation on the number of in-

terrupt nesting levels. However, the interrupt stack must not be allowed to overflow its allocated memory space.

Unless specifically enabled by the software, nested interrupts will not occur. When the CPU acknowledges an interrupt, the I bit in the PSR register is automatically cleared to 0 for the duration of the service routine, disabling any further maskable interrupts.

To allow nested interrupts, an interrupt service routine should first set or clear the respective interrupt enable bits to specify which peripherals will be allowed to interrupt the current service routine. The present interrupt routine should be disabled (or interrupt pending bit cleared). The service routine should then set the PSR.I bit to 1, thus enabling maskable interrupts. This bit can be controlled with the Store Processor Register (SPR) and Load Processor Register (LPR) instructions.

Note: Clearing the pending bit of the current interrupt should not be immediately followed by enabling further interrupts by setting the I bit in the PSR register. Wait states must be inserted into the software after clearing the interrupt pending bit and before another interrupt. Placing a NOP instruction will perform this instruction. This is because the instruction which resets the pending bit may not yet be finished when the interrupts are already enabled again by setting the I bit in the PSR register. To avoid this situation the user has to make sure that prior to enabling the interrupt an additional instruction is inserted. This could look like the example below:

```
SBITi   $0, T1ICRL      # clear pending bit
NOP                     # NOP instruction
MOVW    $0x0a00, r0     # enable further interrupts
LPR     r0, psr
```

A CBITi or SBITi instruction may be used to clear the interrupt pending bit. In such cases, a spurious interrupt may occur.

10.0 Power Management

The Power Management Module (PMM) improves the efficiency of the device by changing the operating mode (and therefore the power consumption) according to the current level of device activity.

The device can operate in any of four power modes:

- Active
- Power Save
- Idle
- Halt

Table12 summarizes the main properties of the four operating modes: the state of the high-frequency oscillator (on or off), the type of clock used by most modules, and the clock used by the Timing and Watchdog Module (TWM).

Table 12 Power Mode Operating Summary

Mode	High-Frequency Oscillator	Clock Used	TWM Clock
Active	On	Main Clock	Slow Clock
Power Save	On or Off	Slow Clock	Slow Clock
Idle	On or Off	None	Slow Clock
Halt	Off	None	None

The low-frequency oscillator continues to operate in all four modes and power must be provided continuously to the device power supply pins. In the Halt mode, however, the internal SLCLK does not toggle, and as a result, the TWM timer and Watchdog Module do not operate. For the Power Save and Idle modes, the high-frequency oscillator can be turned on or off under software control, as long as the low-frequency oscillator exists in the applicable device package.

10.1 ACTIVE MODE

In the Active mode, all device modules are fully operational. This is the operating mode upon reset. Most device modules use the clock generated by the high-frequency clock oscillator. The clock rate is determined by the external crystal network.

Power consumption in the Active mode can be reduced by selectively disabling inactive modules and/or by executing the WAIT instruction. When WAIT is executed, the core stops executing new instructions and waits for an interrupt.

10.2 POWER SAVE MODE

In the Power Save mode, all device modules operate off the low-frequency clock. If the low-frequency clock is generated from an external crystal network, the high-frequency clock oscillator can be turned off to further reduce power consumption.

All on-chip modules continue to operate in the Power Save mode, with the SLCLK acting as their system clock. If this mode is entered by using the WAIT command, the CPU is inactive and waits for an interrupt to wake up. Otherwise, CPU continues to function normally at the lower frequency of the slow clock.

The low frequency of the clock in Power Save mode limits the operation of modules such as the USARTs, MICROWIRE interface, A/D Converter, and timers because they are driven by the slow clock rather than the normal high-speed clock. In order to work properly in Power Save mode, modules that perform real-time operations (such as a USART baud rate generator) must be reprogrammed to use the slower clock.

To reduce power consumption as much as possible, the program should execute a WAIT instruction during periods of CPU inactivity.

10.3 IDLE MODE

In the Idle mode, the clock is stopped for most of the device. Only the Power Management Module and Timing and Watchdog Module continue to operate. Both of these modules use the slow clock in this mode.

10.4 HALT MODE

In the Halt mode, all device clocks are disabled and the high-frequency oscillator is shut off. In this mode, the device consumes the least possible power while maintaining the device memory and register contents. The low-frequency oscillator continues to operate in this mode, but with very low power consumption due to its power-optimized design.

10.5 SWITCHING BETWEEN POWER MODES

Switching from a higher to a lower power consumption mode is accomplished by writing an appropriate value to the Power Management Control/Status Register (PMCSR). Switching from a lower power consumption mode to the Active mode is usually triggered by a hardware interrupt. Figure5 shows the four power consumption modes and the events that trigger a transition from one mode to another.

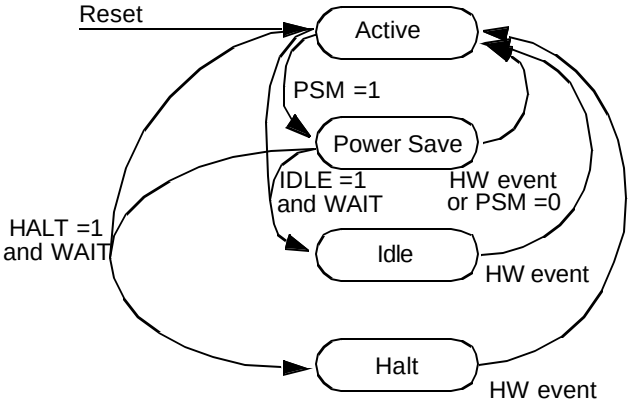


Figure 5. Power Modes and Transitions

Some of the power-up transitions are based on the occurrence of a wake-up event. An event of this type can be either a maskable interrupt or a non-maskable interrupt (NMI). All of the maskable hardware wake-up events are gathered and processed by the Multi-Input Wake-Up Module, which is active in all modes. Once a wake-up event is detected, it is latched until an interrupt acknowledge cycle occurs or a reset is applied.

A wake-up event causes a transition to the Active mode and restores normal clock operation, but does not start execution

of the program. It is the interrupt service routine associated with the wake-up source (MIWU or NMI) that causes actual program execution to resume.

10.5.1 Power Management Control/Status Register (PMCSR)

The Power Management Control/Status Register (PMCSR) is a byte-wide, read/write register that controls the operating power mode (Active, Power Save, Idle, or Halt) and enables or disables the high-frequency oscillator in the Power Save and Idle modes. The two most significant bits, OLFC and OHFC, are read-only status bits controlled by the hardware. Upon reset, the non-reserved bits of this register are cleared. The format of the register is shown below.

7	6	5	4	3	2	1	0
OLFC	OHFC	WBPSM	Reserved	HALT	IDLE	DHF	PSM

PSM — Power Save Mode. When this bit is 0, the device operates in the Active mode. Writing a 1 to this bit position puts the device into the Power Save mode, either immediately or upon execution of the next WAIT instruction, depending on the WBPSM bit.
The PSM bit can be set and cleared by the software. It is also cleared by the hardware when a hardware wake-up event is detected.

DHF — Disable High-Frequency Oscillator. This bit enables (0) or disables (1) the high-frequency oscillator in the Power Save or Idle mode. (The high-frequency oscillator is always enabled in Active mode and always disabled in Halt mode, regardless of this bit settings.) The DHF bit is cleared automatically when a hardware wake-up event is detected.

IDLE — Idle Mode. When this bit is set, the device enters the Idle mode upon execution of a WAIT instruction. In order to enter the Idle mode from the Active mode, the WBPSM bit must be set before the WAIT instruction is executed.
The IDLE bit can be set and cleared by the software. When a hardware wake-up event is detected, this bit is cleared automatically and the device returns to the Active mode.

HALT — Halt Mode. When this bit is set, the device enters the Halt mode upon execution of a WAIT instruction. In order to enter the Halt mode from the Active mode, the WBPSM bit must be set before the WAIT instruction is executed.
The Halt bit can be set and cleared by the software. When a hardware wake-up event is detected, this bit is cleared automatically and the device returns to the Active mode.

WBPSM — Wait Before Entering Power Save Mode. When the CPU writes a 1 to the PSM bit, the WBPSM determines when the transition from Active to Power Save mode is done. If the WBPSM bit is 0, the switch to Power Save mode is initiated immediately; the PSM bit in the register is set to 1 upon completion of the switch to Power Save mode. If the WBPSM bit is 1, the device continues to operate in Active mode until the next WAIT instruction, and then enters the Power Save mode. In this case, the PSM bit is set to 1 immediately, even if a WAIT instruction has not yet been executed.
In the Active mode, the WBPSM bit must be set in order to enter the Idle or Halt mode.

OHFC — Oscillating High-Frequency Clock. This read-only bit indicates the status of the high-frequency clock. If this bit is 1, the high-frequency clock is available and stable. If this bit is 0, the high-frequency clock is either disabled, not available to the Power Management Module, or operating but not yet stable. The device can switch to the Active mode only when this bit is 1.

OLFC — Oscillating Low-Frequency Clock. This read-only bit indicates the status of the low-frequency (slow) clock. If this bit is 1, it indicates that the slow clock is running and stable. The slow clock can be either the prescaled fast clock (the default) or the external oscillator (if selected). The Dual Clock module will not allow a transition to the slow crystal mode unless the slow crystal is operating, so this bit should be 1 under normal circumstances.
The OLFC bit is the multiplexed and sampled output of the “Good Main Clk” and “Good Low Speed Clk” indicator signals. These values are determined at reset or power-up, and then selected according to the programmed slow clock source.
The device can switch from the Active mode to the Power Save or Idle mode only if the OLFC bit is 1. (There is no such restriction on switching to the Halt mode.)

10.5.2 Active to Power Save Mode

A transition from the Active mode to the Power Save mode is accomplished by writing a 1 to the PMCSR.PSM bit. The transition to Power Save mode is either initiated immediately or upon execution of the next WAIT instruction, depending on the PMCSR.WBPSM bit.

For an immediate transition to Power Save mode (PMCSR.WBPSM=0), the CPU continues to operate using the low-frequency clock. The PMCSR.PSM bit is set to 1 when the transition to the Power Save mode is completed.

For a transition upon the next WAIT instruction (PMCSR.WBPSM=1), the CPU continues to operate in the Active mode until it executes a WAIT instruction. Upon execution of the WAIT instruction, the device enters the Power Save mode and the CPU waits for the next interrupt event. In this case, the PMCSR.PSM bit is set to 1 when it is written, even before the WAIT instruction is executed.

10.5.3 Entering the Idle Mode

Entry into the Idle mode is accomplished by writing a 1 to the PMCSR.IDLE bit and then executing a WAIT instruction.

The Idle mode can be entered only from the Active or Power Save mode. For entry from the Active mode, the PMCSR.WBPSM bit must be set before the WAIT instruction is executed.

10.5.4 Disabling the High-Frequency Clock

In systems where the low-frequency crystal is available and is used to generate the Slow Clock (SLCLK), power consumption can be reduced further in the Power Save or Idle mode by disabling the high-frequency clock. This is accomplished by writing a 1 to the PMCSR.DHF bit before executing the WAIT instruction that puts the device in the Power Save or Idle mode. The high-frequency clock is turned off only after the device enters the Power Save or Idle mode.

The CPU operates on the low-frequency clock in Power Save mode. It can turn off the high-frequency clock at any time by writing a 1 to the PMCSR.DHF bit.

The high-frequency oscillator is always enabled in Active mode and always disabled in Halt mode, regardless of the PMCSR.DHF bit setting.

Immediately following power-up and entry into the Active mode, the software must wait for the low-frequency clock to become stable before it can put the device in the Power Save mode. It should monitor the PMCSR.OLFC bit for this purpose. Once this bit is set to 1, the slow clock is stable and the Power Save mode can be entered.

10.5.5 Entering the Halt Mode

Entry into the Halt mode is accomplished by writing a 1 to the PMCSR.HALT bit and then executing a WAIT instruction.

The Halt mode can be entered only from the Active or Power Save mode. For entry from the Active mode, the PMCSR.WBPSM bit must be set before the WAIT instruction is executed.

10.5.6 Software-Controlled Transition to Active Mode

A transition from the Power Save mode to the Active mode can be accomplished by either a software command or a hardware wake-up event. The software method is to write a 0 to the PMCSR.PSM bit. The value of the register bit changes only after the transition to the Active mode is completed.

If the high-frequency oscillator is disabled for Power Save operation, the oscillator must be enabled and allowed to stabilize before the transition to Active mode. To enable the high-frequency oscillator, the software writes a 0 to the PMCSR.DHF bit. Before writing a 0 to the PMCSR.PSM bit, the software should first monitor the PMCSR.OHFC bit to determine whether the oscillator has stabilized.

10.5.7 Wake-Up Transition to Active Mode

A hardware wake-up event switches the device directly from Power Save, Idle, or Halt mode to the Active mode. When a wake-up event occurs, the on-chip hardware performs the following steps:

1. Clears the PMCSR.DHF bit, thus enabling the high-frequency clock (if it was disabled).
2. Waits for the PMCSR.OHFC bit to be set, which indicates that the high-frequency clock is operating and is stable.
3. Switches the device into the Active mode.

10.5.8 Power Mode Switching Protection

The Power Management Module has several mechanisms to protect the device from malfunctions caused by missing or unstable clock signals.

The PMCSR.OHFC and PMCSR.OLFC bits indicate the current status of the high-frequency and low-frequency clock oscillators, respectively. The software can check the appropriate bit before it changes to an operating mode that requires the clock. A status bit set to 1 indicates an operating, stable clock. A status bit cleared to 0 indicates a clock that is disabled, not available, or not yet stable.

During a power mode transition, if there is a request to switch to a mode that uses clock with the status bit cleared to 0, the switch is delayed until that bit is set to 1 by the hardware.

When the system is built without an external crystal network for the low-frequency clock, the high-frequency clock is divided by a prescaler factor to produce the low-frequency clock. In this situation, the high-frequency clock is disabled only in the Halt mode, and cannot be disabled for the Power Save or Idle mode, regardless of the software command issued.

Without an external crystal network for the low-frequency clock, the device comes out of the Halt or Idle mode and enters the Active mode with the high-speed oscillator used as the clock. The device can still enter the Power Save from the Active mode by using the high-frequency-clock divider to generate the slow clock (PMCSR.DHF=0).

Note: For correct operation in the absence of a low-frequency crystal, the X2CKI pin must be tied low (not left floating) so the hardware can detect the absence of the crystal.

11.0 Dual Clock and Reset

The Dual Clock and Reset module (CLK2RES) generates a high-speed main system clock from an external crystal network and a slow clock (32.768 kHz or other rate) for operating the device in Power Save mode. It also provides the main system reset signal and a power-on reset function.

Figure6 is block diagram of the Dual Clock and Reset module.

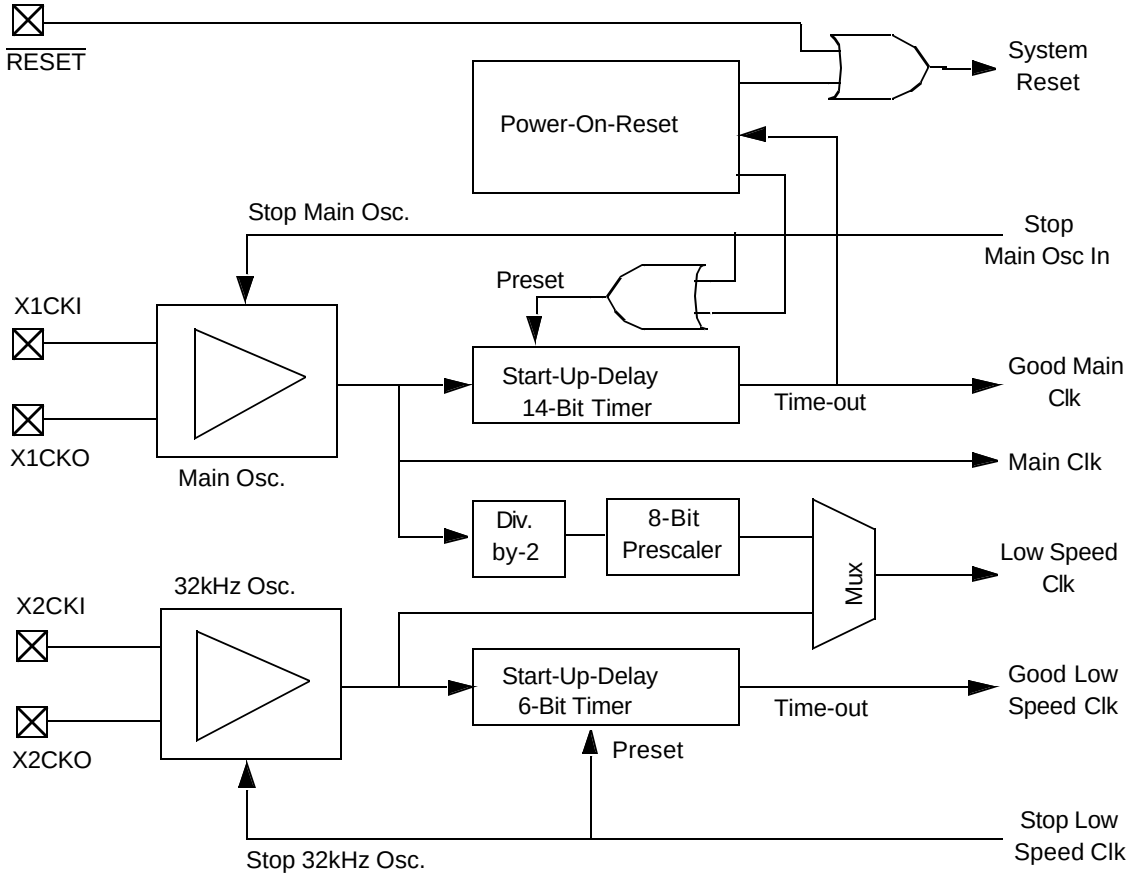


Figure 6. Dual Clock and Reset Module Block Diagram

11.1 EXTERNAL CRYSTAL NETWORK

An external crystal network is required at pins X1CKI and X1CKO for the main clock. A similar external crystal network may be used at pins X2CKI and X2CKO for the slow clock in packages that have these pins. If an external crystal network is not used for the slow clock, the clock is generated by dividing the fast main clock.

Figure7 shows the required crystal network at X1CKI/X1CKO and optional crystal network at X2CKI/X2CKO. Table13 shows the component specifications for the main crystal network and Table14 shows the component specifications for the 32.768 kHz crystal network.

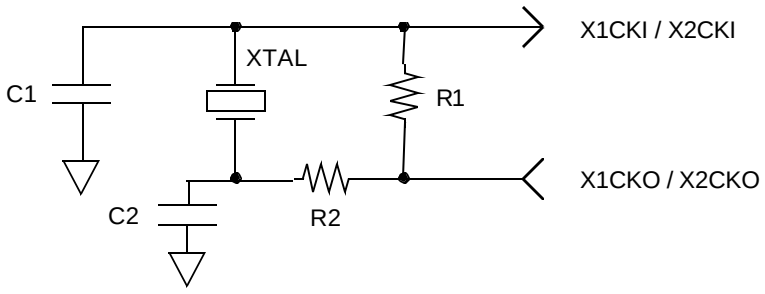


Figure 7. External Crystal Network

Table 13 Component Values of the High Frequency Crystal Circuit

Component	Parameters	Values	Values	Values	Values	Tolerance
Oscillator	Resonance Frequency Type Max. Serial Resistance Max. Shunt Capacitance Load Capacitance	4 MHz AT-Cut 75 Ω 4 pF 12 pF	12 MHz AT-Cut 35 Ω 4 pF 15 pF	16 MHz AT-Cut 35 Ω 4 pF 15 pF	20 MHz AT-Cut 35 Ω 4 pF 20 pF	N/A
Crystal	Resistor R1	1 MΩ	1 MΩ	1 MΩ	1 MΩ	5%
	Resistor R2	0 Ω	0 Ω	0 Ω	0 Ω	5%
	Capacitor C1, C2	22 pF	20 pF	20 pF	20 pF	20%

Table 14 Component Values of the Low Frequency Crystal Circuit

Component	Parameters	Values	Tolerance
Oscillator	Resonance Frequency Type Maximum Serial Resistance Maximum Shunt Capacitance Load Capacitance	32.768kHz Parallel N-Cut or XY-bar 40 kΩ 2 pF 9-13 pF	N/A
Crystal	Resistor R1	10-20 MΩ	5%
	Resistor R2	4.7 kΩ	5%
	Capacitor C1, C2	20 pF	20%

The crystal oscillator you choose may require external components different from the ones specified above. In that case, consult with National Semiconductor for the component specifications.

The crystals and other oscillator components should be placed close to the X1CKI/X1CKO and X2CKI/X2CKO device input pins to keep the printed trace lengths to an absolute minimum.

Choose capacitor component values in the tables to obtain the specified load capacitance for the crystal when combined with the parasitic capacitance of the trace, socket, and package (which can vary from 0 to 8 pF). As a guideline, the load capacitance is:

$C_L = (C_1 * C_2)/(C_1+C_2) + C_{parasitic}$

$C_2 > C_1$

C_1 can be trimmed to obtain the desired load capacitance.

The start-up time of the 32.768 kHz oscillator can vary from one to six seconds. The long start-up time is due to the high "Q" value and high serial resistance of the crystal necessary to minimize power consumption in Power Save mode.

11.2 MAIN SYSTEM CLOCK

The main system clock is generated by the main oscillator. It can be stopped by the Power Management Module to reduce power consumption during periods of reduced activity. When the main clock is restarted, a 14-bit timer generates a "Good Main Clk" signal after a start-up delay of 32,768 clock cycles. This signal is an indicator that the main clock oscillator is stable.

The "Stop Main Osc" signal from the Power Management Module stops and starts the main oscillator. When this signal is asserted, it presets the 14-bit timer to 3FFF hex and stops the main oscillator. When the signal goes inactive, the main oscillator starts and the 14-bit timer counts down from its preset value. When the timer reaches zero, it stops counting and asserts the "Good Main Clk" signal.

11.3 SLOW SYSTEM CLOCK

The slow (32.768 kHz) clock is necessary for operating the device in Power Save modes and to provide a clock source for modules such as the Timing and Watchdog Module.

The slow clock operates in a manner similar to the main clock. The "Stop Slow Osc" signal from the Power Management Module stops and starts the slow oscillator. When this signal is asserted, it presets a 6-bit timer to 3F hex and disables the slow oscillator. When the signal goes inactive, the slow oscillator starts and the 6-bit timer counts down from its preset value. When the timer reaches zero, it stops counting and asserts the "Good Low Speed Clk" signal, thus indicating that the slow clock is stable.

For systems that do not require a reduced power consumption mode, the external crystal network may be omitted for the slow clock. In that case, the slow clock can be created by dividing the main clock by a prescaler factor. The prescaler circuit consists of a fixed divide-by-2 counter and a programmable 8-bit prescaler register. This allows a choice of clock divisors ranging from 2 to 512. The resulting slow clock frequency must not exceed 100 KHz.

A software-programmable multiplexer selects either the prescaled main clock or the 32.768 kHz oscillator as the slow clock. Upon reset, the prescaled main clock is selected, ensuring that the slow clock is always present initially. Selection of the 32.768 kHz oscillator as the slow clock disables the clock prescaler, which allows the CLK1 oscillator to be turned

off during power-save operation, thus reducing power consumption and radiated emissions. This can be done only if the module detects a toggling low-speed oscillator. If the low-speed oscillator is not operating, the prescaler remains available as the slow clock source.

11.4 POWER-ON RESET

The Power-On Reset circuit generates a system reset signal upon power-up and holds the signal active for a period of time to allow the crystal oscillator to stabilize. The circuit detects a power turn-on condition, which presets the 14-bit timer to 3FFF hex. Once oscillation starts and the clock becomes active, the timer starts counting down. When the count reaches zero, the 14-bit timer stops counting and the internal reset signal is deactivated (unless the $\overline{\text{RESET}}$ pin is held low).

The circuit sets a power-on reset flag bit upon detection of a power-on condition. The CPU can read this flag to determine whether a reset was caused by a power-up or by the $\overline{\text{RESET}}$ input.

Note: Power-On Reset circuit cannot be used to detect a drop in the supply voltage.

11.5 EXTERNAL RESET

An active-low reset input pin called $\overline{\text{RESET}}$ allows the device to be reset at any time. When the signal goes low, it generates an internal system reset signal that remains active until the $\overline{\text{RESET}}$ signal goes high again.

11.6 DUAL CLOCK AND RESET REGISTERS

The Dual Clock and Reset module (CLK2RES) contains two registers: the Clock and Reset Control register (CRCTRL) and the Slow Clock Prescaler register (PRSSC).

11.6.1 Clock and Reset Control Register (CRCTRL)

Clock and Reset Control Register (CRCTRL) is a byte-wide read/write register that contains the power-on reset flag and selects the type of slow clock. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved						POR	SCLK

SCLK — Slow Clock Select. When this bit is set to 1, the 32.728 kHz oscillator is used for the slow clock. When this bit is cleared to 0, the prescaled main clock is used for the slow clock. Upon reset, this bit is cleared to 0.

POR — Power-On Reset. This bit is set to 1 by the hardware when a power-on condition is detected, allowing the CPU to determine whether a power-up has occurred. The CPU can clear this bit to 0 but cannot set it to 1. Any attempt by the CPU to set this bit is ignored.

11.7 SLOW CLOCK PRESCALER REGISTER (PRSSC)

The Slow Clock Prescaler (PRSSC) register is a byte-wide read/write register that holds the clock divisor used to generate the slow clock from the main clock. The format of the register is shown below.

7	6	5	4	3	2	1	0
SCDIV							

SCDIV — Slow Clock Divisor. If the clock divider is enabled (CRCTRL.SCLK=0), the main clock is divided by (SCDIV+1)*2 to produce the slow system clock. Upon reset, PRSSC register is set to FF hex.

12.0 Multi-Input Wake-Up

The Multi-Input Wake-Up (MIWU) module monitors its eight input channels for a software-selectable trigger condition. Upon detection of a trigger condition, the module generates either a wake-up request or an interrupt request. A wake-up request can be used by the power management unit to exit the Halt, Idle, or Power Save mode and return to the active mode. An interrupt request generates an interrupt to the CPU (interrupt IRQ2), allowing interrupt processing in response to external events.

The wake-up event only activates the clocks and CPU, but does not by itself initiate execution of any code. It is the interrupt request associated with the MIWU that gets the CPU to start executing code by jumping to the proper interrupt routine. Therefore, setting up the MIWU interrupt handler is essential for any wake-up operation.

Figure8 is a block diagram showing the internal operation of the Multi-Input Wake-Up module.

The input pins for the Multi-Input Wake-Up channels are named WUI0 through WUI7. These pins are alternate functions of I/O pins in Port I and Port L. The first Multi-Input Wake-Up channel is software-selectable between the WUI0 input pin and the T0OUT signal from the Timing and Watchdog (TWM) module, which can be used to wake up the device after a programmed time interval. The WKCTL register controls this selection. The remaining seven channels always use input pins WUI1 through WUI7.

Each input can be configured to trigger on rising or falling edges, as determined by the setting in the WKEDG register. Each trigger event is latched into the WKPND register. If a trigger event is enabled by its respective bit in the WNENA register, an active wake-up/interrupt signal is generated. The software can determine which channel has generated the active signal by reading the WKPND register.

The Multi-Input Wake-Up module is active at all times, including the Halt mode. All device clocks are stopped in this mode. Therefore, detecting an external trigger condition and the subsequent setting of the pending flag are not synchronous to the system clock.

12.1 WAKE-UP EDGE DETECTION REGISTER (WKEDG)

The Wake-Up Edge Detection (WKEDG) register is a byte-wide read/write register that controls the edge sensitivity of the Multi-Input Wake-Up pins. Register bits 0 through 7 control input pins WUI0 through WUI7, respectively. A bit cleared to 0 configures the corresponding input to trigger on a rising edge (a low-to-high transition). A bit set to 1 configures the corresponding input to trigger on a falling edge (a high-to-low transition).

This register is cleared upon reset, which configures all eight inputs to be triggered on rising edges.

The register format is shown below.

7	6	5	4	3	2	1	0
WKED7	WKED6	WKED5	WKED4	WKED3	WKED2	WKED1	WKED0

12.2 WAKE-UP ENABLE REGISTER (WKENA)

The Wake-Up Enable (WKENA) register is a byte-wide read/write register that enables or disables each of the Multi-Input Wake-Up channels. Register bits 0 through 7 control channels WUI0 through WUI7, respectively. A bit cleared to 0 disables the wake-up/interrupt function and a bit set to 1 enables the function.

This register is cleared upon reset, which disables all eight wake-up/interrupt channels.

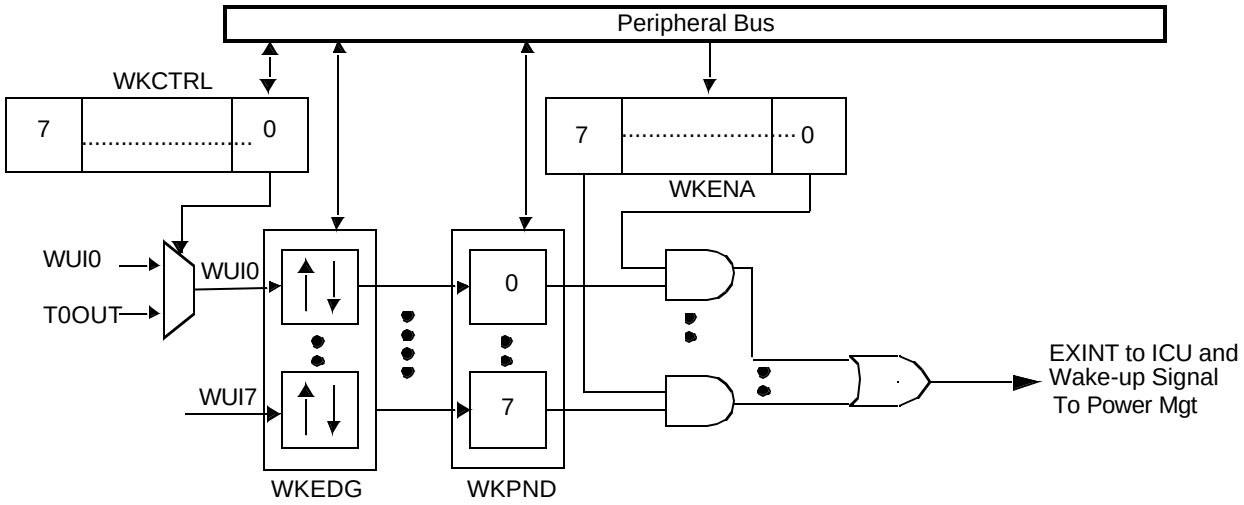


Figure 8. Multi-Input Wake-Up Module Block Diagram

The register format is shown below.

7	6	5	4	3	2	1	0
WKEN7	WKEN6	WKEN5	WKEN4	WKEN3	WKEN2	WKEN1	WKEN0

12.3 WAKE-UP SOURCE SELECT REGISTER (WKCTRL)

The Wake-Up Source Select (WKCTRL) register is a byte-wide read/write register that selects the trigger source for the first of the eight channels. Register bit 0 controls this function; the seven higher-order bits are reserved. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved							WKSEL0

WKSEL0 Wake-Up Select 0. This bit cleared to 0 selects the WUI0 pin as the trigger source. This bit set to 1 selects the T0OUT signal from the Timing and Watchdog (TWM) module as the trigger source, which can be used to wake up the device after a programmed time interval. This bit is cleared upon reset. All reserved bits must be written with 0 for this module to function properly.

12.4 WAKE-UP PENDING REGISTER (WKPND)

The Wake-Up Pending (WKPND) register is a byte-wide read/write register in which the Multi-Input Wake-Up module latches any detected trigger conditions. Register bits 0 through 7 serve as latches for channels WUI0 through WUI7, respectively. A bit cleared to 0 indicates that no trigger condition has occurred. A bit set to 1 indicates that a trigger condition has occurred and is pending on the corresponding channel. This register is cleared upon reset.

The CPU can only write a 1 to any bit position in this register. If the CPU attempts to write a 0, it has no effect on that bit. To clear a bit in this register, the CPU must use the WKPCL register (described below). This implementation prevents a potential hardware-software conflict during a read-modify-write operation on the WKPND register.

The register format is shown below.

7	6	5	4	3	2	1	0
WKPD7	WKPD6	WKPD5	WKPD4	WKPD3	WKPD2	WKPD1	WKPD0

12.5 WAKE-UP PENDING CLEAR REGISTER (WKPCL)

The Wake-Up Pending Clear (WKPCL) register is a byte-wide write-only register that lets the CPU clear bits in the WKPND register. Writing a 1 to a bit position in the WKPCL register clears the corresponding bit in the WKPND register. Writing a 0 leaves the corresponding bit in the WKPND register unchanged.

Reading this register location returns unknown data. Therefore, do not use a read-modify-write sequence to set the individual bits. In other words, do not attempt to read the register and do a logical OR with the register value. Instead, just write the mask directly to the register address.

The register format is shown below.

7	6	5	4	3	2	1	0
WKCL7	WKCL6	WKCL5	WKCL4	WKCL3	WKCL2	WKCL1	WKCL0

12.6 PROGRAMMING PROCEDURES

To set up and use the Multi-Input Wake-Up function, use the following procedure. Performing the steps in the order shown will prevent false triggering of a wake-up condition. This same procedure should be used following a reset because the wake-up inputs are left floating, resulting in unknown data on the input pins.

1. Clear the WKENA register to disable the wake-up channels.
2. If the input originates from an I/O port (the usual case), set the corresponding bit in the port direction register to configure the I/O pin to operate as an input.
3. Write the WKEDG register to select the desired type of edge sensitivity (clear to 0 for rising edge, set to 1 for falling edge).
4. Set all bits in the WKPCL register to clear any pending bits in the WKPND register.
5. Set the bits in the WKENA register corresponding to the wake-up channels to be activated.

To change the edge sensitivity of a wake-up channel, use the following procedure. Performing the steps in the order shown will prevent false triggering of a wake-up/interrupt condition.

1. Clear the WKENA bit associated with the input to be reprogrammed.
2. Write the new value to the corresponding bit position in the WKEDG register to reprogram the edge sensitivity of the input.
3. Set the corresponding bit in the WKPCL register to clear the pending bit in the WKPND register.
4. Set the same WKENA bit to re-enable the wake-up function.

13.0 Real-Time Timer and WATCHDOG

The Timing and WATCHDOG Module (TWM) generates the clocks and interrupts used for timing periodic functions in the system, and also provides Watchdog protection against software errors. The module operates off the slow clock either generated by the external 32kHz oscillator or from the prescaled high speed system clock. The maximum operating clock frequency is 100kHz.

The WATCHDOG is designed to detect program execution errors. Once WATCHDOG operation is initiated, the software must periodically write a specific value to a WATCHDOG register. If the software fails to do so, a WATCHDOG error is triggered, which resets the device.

The TWM is flexible in allowing selection of a variety of clock ratios and clock sources for the WATCHDOG circuit. Once the software configures the TWM, it can lock the configuration for a higher level of protection against erroneous software action. Once locked, the TWM can be released only by a device reset.

13.1 TWM STRUCTURE

Figure9 is a block diagram showing the internal structure of the Timing and WATCHDOG module. There are two main sections: the Real-Time Timer (T0) section at the top and the WATCHDOG section on the bottom.

All counting activities of the module are based on the slow clock (SLCLK). A prescaler counter divides this clock to make a slower clock. The prescaler factor is defined by a 3-bit field in the Timer and WATCHDOG Prescaler register, which selects either 1, 2, 4, 8, 16, or 32 and the divide-by factor. Thus, the prescaled clock period can be set to 1, 2, 4, 8, 16, or 32 times the slow clock period. The prescaled clock signal is called T0IN.

13.2 TIMER T0 OPERATION

Timer T0 is a programmable 16-bit down counter that can be used as the time base for real-time operations such as a periodic audible tick. It can also be used to drive the WATCHDOG circuit.

The timer starts counting from the value loaded into the TWMT0 register and counts down on each rising edge of T0IN. When the timer reaches zero, it is automatically reloaded from the TWMT0 register and continues counting down from that value. Thus, the frequency of the timer is:

f_{SLCLK} / [(TWMT0+1) * prescaler]

When an external crystal oscillator is used as the SLCLK source or when the fast clock is divided accordingly, f_{SLCLK} is 32.768 kHz.

The value stored in TWMT0 can range from 0001 hex to FFFF hex.

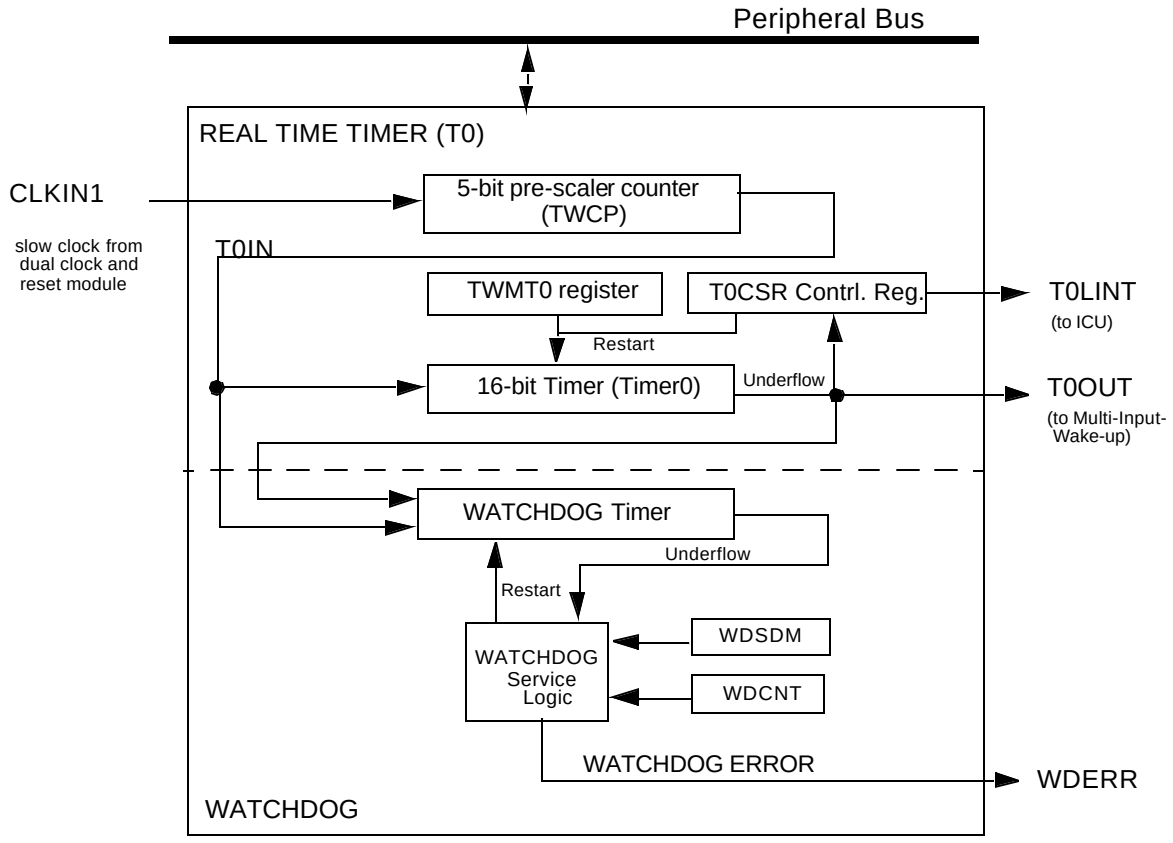


Figure 9. Timing and WATCHDOG Module Block Diagram

When the counter reaches zero, an internal timer signal called T0OUT is set to 1 for one T0IN clock cycle. This signal sets the TC bit in the TWMT0 Control and Status Register (T0CSR). It also generates an interrupt called RTI (IRQ14) if the interrupt is enabled by the T0CSR.T0INTE bit.

If the software loads TWMT0 with a new value, the timer uses that value the next time that it reloads the 16-bit timer register (in other words, after reaching zero). The software can restart the timer at any time (on the very next edge of the T0IN clock) by setting the Restart (RST) bit in the T0CSR register. The T0CSR.RST bit is cleared automatically upon restart of the 16-bit timer.

Note: If the user wishes to switch to power save or idle mode after setting T0CSR.RST, the user must wait for reset operation to complete before doing the switch.

13.3 WATCHDOG OPERATION

The WATCHDOG is an 8-bit down counter that operates on the rising edge of a specified clock source. Upon reset, the WATCHDOG is disabled; it does not count and no WATCHDOG signal is generated. A write to either the WATCHDOG Count (WDCNT) register or the WATCHDOG Service Data Match (WDSDM) register starts the counter. The WATCHDOG counter counts down from the value programmed in to the WDCNT register. Once started, only a reset can stop the WATCHDOG from operating.

The WATCHDOG can be programmed to use either T0OUT or T0IN as its clock source (the output and input of Timer T0, respectively). The TWCFG.WDCT0I bit controls this clock selection.

The software must periodically “service” the WATCHDOG. There are two ways to service the WATCHDOG, the choice depending on the programmed value of the WDSDME bit in the Timer and WATCHDOG Configuration (TWCFG) register.

If TWCFG.WDSDME bit is cleared to 0, the WATCHDOG is serviced by writing a value to the WDCNT register. The value written to the register is reloaded into the WATCHDOG counter. The counter then continues counting down from that value.

If TWCFG.WDSDME bit is set to 1, the WATCHDOG is serviced by writing the value 5C hex to the WATCHDOG Service Data Match (WDSDM) register. This reloads the WATCHDOG counter with the value previously programmed into the WDCNT register. The counter then continues counting down from that value.

A WATCHDOG error signal is generated by any of the following events:

- — The WATCHDOG serviced too late .
- — The WATCHDOG serviced too often.
- — The WDSDM register is written with a value other than 5C hex when WDSDM type servicing is enabled (TWCFG.WDSDME=1).

A WATCHDOG error condition resets the device.

13.3.1 Register Locking

The Timer and WATCHDOG Configuration (TWCFG) register is used to set the WATCHDOG configuration. It controls the WATCHDOG clock source (T0IN or T0OUT), the type of WATCHDOG servicing (using WDCNT or WDSDM), and the locking state of the TWCFG, TWCPR, TIMER0, T0CSR, and WDCNT registers. A register that is locked cannot be read or written. A write operation is ignored and a read operation returns unpredictable results.

If the TWCFG register is itself locked, it remains locked until the device is reset. Any other locked registers also remain locked until the device is reset. This feature prevents a runaway program from tampering with the programmed WATCHDOG function.

13.3.2 Power Save Mode Operation

The Timer and WATCHDOG Module is active in both the Power Save and Idle modes. The clocks and counters continue to operate normally in these modes. The WDSDM register is accessible in the Power Save and Idle modes, but the other TWM registers are accessible only in the Active mode. Therefore, WATCHDOG servicing must be carried out using the WDSDM register in the Power Save or Idle mode.

In the Halt mode, the entire device is frozen, including the Timer and WATCHDOG Module. Upon return to the Active mode, operation of the module resumes at the point at which it was stopped.

Note: After a restart or WATCHDOG service through WDCNT, do not enter Power Save mode for a period equivalent to 5 slow clock cycles.

13.4 TWM REGISTERS

The TWM registers controls the operation of the Timing and WATCHDOG Module. There are six such registers:

- Timer and WATCHDOG Configuration Register (TWCFG)
- Timer and WATCHDOG Clock Prescaler Register (TWCP)
- TWM Timer 0 Register (TWMT0)
- TWMT0 Control and Status Register (T0CSR)
- WATCHDOG Count Register (WDCNT)
- WATCHDOG Service Data Match Register (WDSDM)

The WDSDM register is accessible in both Active and Power Save mode. The other TWM registers are accessible only in Active mode.

13.4.1 Timer and WATCHDOG Configuration Register (TWCFG)

The TWCFG register is a byte-wide, read/write register that selects the WATCHDOG clock input and service method, and also allows the WATCHDOG registers to be selectively locked. Once a bit is set, that bit cannot be cleared until the device resets. Upon reset, the non-reserved bits of the register are all cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		WDSDME	WDCT0I	LWDCNT	LTWMT0	LTWCP	LTWCFG

LTWCFG Lock TWCFG Register. When cleared to 0, access to the TWCFG register is allowed. When set to 1, the TWCFG register is locked. A

locked register cannot be read or written; a read operation returns unpredictable values and a write operation is ignored. Locking the TWCFG register remains in effect until the device is reset.

LTWCP — Lock TWCP Register. When cleared to 0, access to the TWCP register is allowed. When set to 1, the TWCP register is locked.

LTWMT0 — Lock TWMT0 Register. When cleared to 0, access to the TWMT0 and T0CSR registers are allowed. When set to 1, the TWMT0 and T0CSR registers are locked.

LWDCNT — Lock LDWCNT Register. When cleared to 0, access to the LDWCNT register is allowed. When set to 1, the LDWCNT register is locked.

WDCT0I — WATCHDOG Clock from T0IN. When cleared to 0, the T0OUT signal (the output of Timer T0) is used as the WATCHDOG clock. When set to 1, the T0IN signal (the prescaled slow clock) is used as the WATCHDOG clock.

WDSDME — WATCHDOG Service Data Match Enable. When cleared to 0, WATCHDOG servicing is accomplished by writing a count value to the WDCNT register; write operations to the WATCHDOG Service Data Match (WDSDM) register are ignored. When set to 1, WATCHDOG servicing is accomplished by writing the value 5C hex to the WDSDM register.

13.4.2 Timer and WATCHDOG Clock Prescaler Register (TWCP)

The TWCP register is a byte-wide, read/write register that defines the prescaler value used for dividing the low frequency clock to generate the T0IN clock. Upon reset, the non-reserved bits of the register are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved					MDIV		

MDIV — Main Clock Divide. This 3-bit field defines the prescaler factor used for dividing the low speed device clock to create the T0IN clock. The allowed 3-bit values and the corresponding clock divisors and clock rates are listed below.

MDIV	Clock Divisor	TOIN Frequency (f_{SCLK}=32.768 kHz)
000	1	32.768 kHz
001	2	16.384 kHz
010	4	8.192 kHz
011	8	4.096 kHz
100	16	2.048 kHz
101	32	1.024 kHz
other	Reserved	N/A

13.4.3 TWM Timer 0 Register (TWMT0)

The TWMT0 register is a word-wide, read/write register that defines the T0OUT interrupt rate. Upon reset, TWMT0 register is initialized to FFFF hex. The register format is shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESET															

PRESET — Timer T0 Preset. Timer T0 is reloaded with this value on each underflow. Thus, the frequency of the Timer T0 interrupt is the frequency of T0IN divided by (PRESET+1). The allowed values of PRESET are 0001 hex through FFFF hex.

13.4.4 TWMT0 Control and Status Register (T0CSR)

The T0CSR register is a byte-wide, read/write register that controls Timer T0 and shows its current status. Upon reset, the non-reserved bits of the register are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved					T0INTE	TC	RST

RST — Restart. When this bit is set to 1, it forces the timer to reload the value in the TWMT0 register on the next rising edge of the selected input clock. The RST bit is reset automatically by the hardware on the same rising edge of the selected input clock. Writing a 0 to this bit position has no effect. Upon reset, the non-reserved bits of the register are cleared to 0.

TC — Terminal Count. This bit is set to 1 by the hardware when the Timer T0 count reaches zero and is cleared to 0 when the software reads the T0CSR register. It is a read-only bit. Any data written to this bit position is ignored.

T0INTE — Timer T0 Interrupt Enable. When this bit is set to 1, it enables an interrupt to the CPU each time the Timer T0 count reaches zero. When this bit is cleared to 0, Timer T0 interrupts are disabled.

13.4.5 WATCHDOG Count Register (WDCNT)

The WDCNT register is a byte-wide, write-only register that holds the value that is loaded into the WATCHDOG counter each time the WATCHDOG is serviced. The WATCHDOG is started by the first write to this register. Each successive write to this register restarts the WATCHDOG count with the written value. Upon reset, this register is initialized to 0F hex.

13.4.6 WATCHDOG Service Data Match Register (WDSDM)

The WSDSM register is a byte-wide, write-only register used for servicing the WATCHDOG. When this type of servicing is enabled (TWCFG.WDSDME=1), the WATCHDOG is serviced by writing the value 5C hex to the WSDSM register. Each such servicing reloads the WATCHDOG counter with the value previously written to the WDCNT register. Writing any data other than 5C hex triggers a WATCHDOG error. Writing to the register more than once in one WATCHDOG clock cycle also triggers a WATCHDOG error signal. If this type of servicing is disabled (TWCFG.WDSDME=0), any write to the WSDSM register is ignored.

13.5 WATCHDOG PROGRAMMING PROCEDURE

The highest level of protection against software errors is achieved by programming and then locking the WATCHDOG registers and using the WDSDM register for servicing. This is the procedure:

1. Write the desired values into the TWM Clock Prescaler register (TWCP) and the TWM Timer 0 register (TWMT0) to control the T0IN and T0OUT clock rates. The frequency of T0IN can be programmed to any of six frequencies ranging from $1/32*f_{SLCLK}$ to f_{SLCLK} (1.024 kHz to 32.768 kHz if SLCLK is 32.768 kHz). The frequency of T0OUT is equal to the frequency of T0IN divided by (1+PRESET), where PRESET is the value written to the TWMT0 register.
2. Configure the WATCHDOG clock to use either T0IN or T0OUT by setting or clearing the TWCFG.WDCT0I bit.
3. Write the initial value into the WDCNT register. This starts operation of the WATCHDOG and specifies the maximum allowed number of WATCHDOG clock cycles between service operations.
4. Lock the WATCHDOG registers and enable the WATCHDOG Service Data Match Enable function by setting bits 0, 1, 2, 3, and 5 in the TWCFG register.
5. Service the WATCHDOG by periodically writing the value 5C hex to the WDSDM register at an appropriate rate. Servicing must occur at least once per period programmed into the WDCNT register, but no more than once in a single WATCHDOG input clock cycle.

14.0 Multi-Function Timer

The Multi-Function Timer (MFT16) module contains two independent timer/counter units called MFT1 and MFT2, each containing a pair of 16-bit timer/counters. Each timer/counter unit offers a choice of clock sources for operation and can be configured to operate in any of the following modes:

- Processor-Independent Pulse Width Modulation (PWM) mode, which generates pulses of a specified width and duty cycle, and which also provides a general-purpose timer/counter
- Dual Input Capture mode, which measures the elapsed time between occurrences of external events, and which also provides a general-purpose timer/counter
- Dual Independent Timer mode, which generates system timing signals or counts occurrences of external events
- Single Input Capture and Single Timer mode, which provides one external event counter and one system timer

The two timer units, MFT1 and MFT2, are identical in operation and separately programmable. Each timer unit uses two I/O pins, called T1A and T1B (for Timer MFT1) or T2A and T2B (for Timer MFT2). The timer I/O pins are alternate functions of the Port F I/O pins.

In the description of the timers, the lower-case letter “n” represents the timer number, either 1 or 2. For example, “TnA” means I/O pin T1A or T2A.

14.1 TIMER STRUCTURE

Figure10 is a block diagram showing the internal structure of each timer. There are two main functional blocks: a Timer/Counter and Action block and a Clock Source block. The Timer/Counter and Action block contains two separate timer/counter units, called Timer/Counter I and Timer/Counter II (a total of four timer/counter unit in both MFT1 and MFT2).

14.1.1 Timer/Counter Block

The Timer/Counter block contains the following functional blocks:

— two 16-bit counters, Timer/Counter I (TnCNT1) and Timer/Counter II (TnCNT2)
— two 16-bit reload/capture registers, TnCRA and TnCRB
— control logic necessary to configure the timer to operate in any of the four operating modes
— interrupt control and I/O control logic

In a power-saving mode that uses the low-frequency (32.768 kHz) clock as the system clock, the synchronization circuit requires that the slow clock operate at no more than one-fourth the speed of the 32.768 kHz system clock.

14.1.2 Clock Source Block

The Clock Source block generates the signals used to clock the two timer/counter registers. The internal structure of the Clock Source block is shown in Figure11.

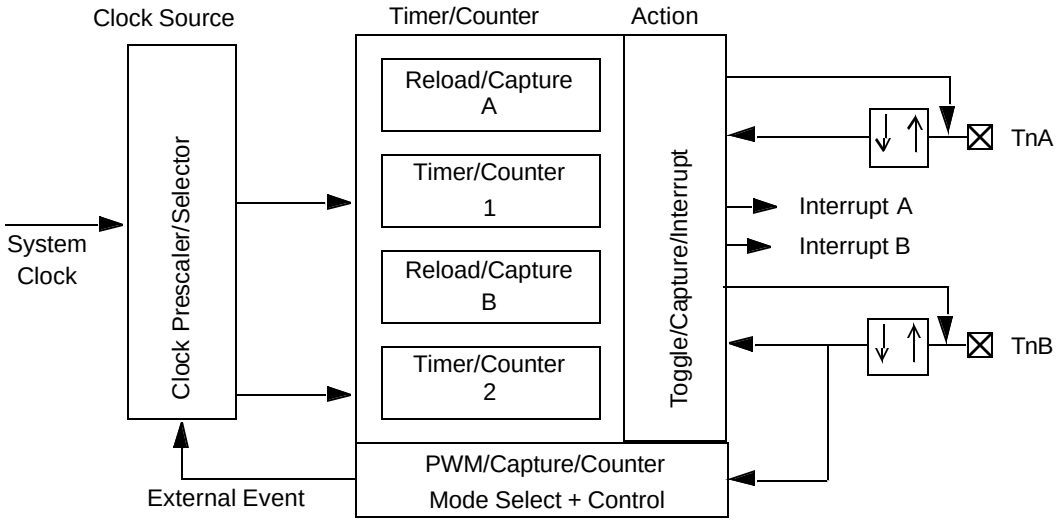


Figure 10. Multi-Function Timer Block Diagram

Counter Clock Source Select

There are two clock source selectors that allow the software to independently select the clock source for each of the two 16-bit counters from any one of the following sources:

— no clock (which stops the counter)
— prescaled system clock
— external event count based on TnB
— pulse accumulate mode based on TnB
— slow clock (derived from the low-frequency oscillator or divided from the high-speed oscillator)

Prescaler

The 5-bit clock prescaler allows the software to run the timer with a prescaled clock signal. The prescaler consists of a 5-bit read/write prescaler register (TnPRSC) and a 5-bit down counter. The system clock is divided by the value contained in the prescaler register plus 1. Thus, the timer clock period

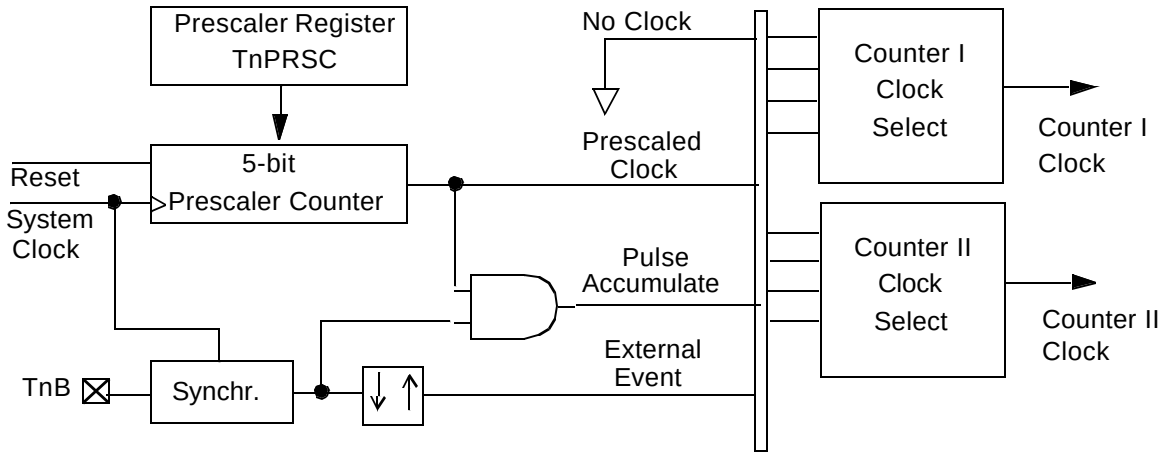


Figure 11. Clock Source Block Diagram

can be set to any value from 1 to 32 divisions of the system clock period. The prescaler register and down counter are both cleared upon reset.

External Event Clock

The TnB I/O pin can be configured to operate as an external event input clock for either of the two 16-bit counters. This input can be programmed to detect either rising or falling edges. The minimum pulse width of the external signal is one system clock cycle. This means that the maximum frequency at which the counter can run in this mode is one-half of the system clock frequency. This clock source is not available in the capture modes (modes 2 and 4) because the TnB pin is used as one of the two capture inputs.

Pulse Accumulate Mode

The counter can also be configured to count prescaler output clock pulses when the TnB is high and not count when TnB is low, as illustrated in Figure12. The resulting count is an indicator of the cumulative time that TnB is high. This is called the “pulse accumulate” mode. In this mode, an AND gate generates a clock signal for the counter whenever a prescaler clock pulse is generated and TnB input is high. (The polarity of the TnB signal is programmable, so the counter can count when TnB is low rather than high.) The pulse accumulate mode is not available in the capture modes (modes 2 and 4) because the TnB pin is used as one of the two capture inputs.

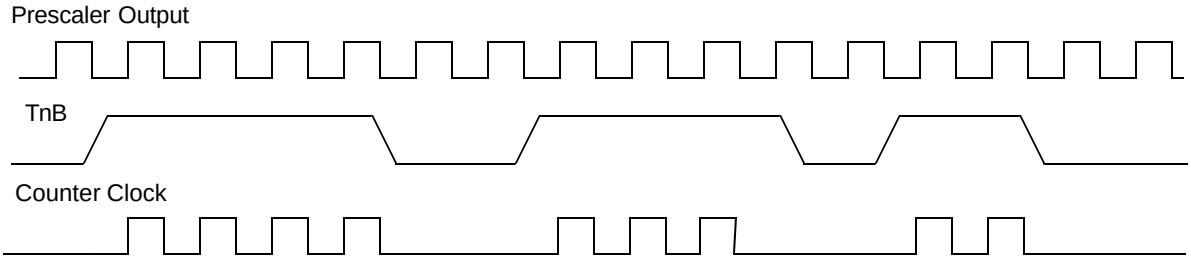


Figure 12. Pulse Accumulate Mode Operation

Slow Clock

The slow clock is generated by the Dual Clock and Reset (CLK2RES) module. The clock source is either the divided fast clock or the external 32.768 kHz clock crystal (if available and selected). The slow clock can be used as the clock source for the two 16-bit counters. Because the slow clock can be asynchronous to the system clock, a circuit is provided to synchronize the clock signal to the high-frequency system clock before it is used for clocking the counters. The synchronization circuit requires that the slow clock operate at no more than one-fourth the speed of the system clock.

Limitations in Low-Power Modes

The Power Save mode uses the low-frequency clock as the system clock. In this mode, the slow clock cannot be used as a clock source for the timers because both CLK and SLCLK are driven then at the same frequency, and the 2:1 system-clock to input clock ratio needed for the synchronization cannot be maintained. However, the External Event Clock and Pulse Accumulate Mode will still work, as long as the external event pulses are at least the size of the whole slow-clock period. Using the prescaled system clock will also work, but at a much slower rate than the original system clock.

Some Power Save modes stops the system clock (the high-frequency and/or low-frequency clock) completely. If the system clock is stopped, the timer stops counting until the system clock resumes operation.

In the Idle or Halt mode, the system clock stops completely, which stops the operation of the timers. In that case, the timers stop counting until the system clock resumes operation.

14.2 TIMER OPERATING MODES

Each timer/counter unit can be configured to operate in any of the following modes:

- Processor-Independent Pulse Width Modulation (PWM) mode
- Dual Input Capture mode
- Dual Independent Timer mode
- Single Input Capture and Single Timer mode

Upon reset, the timers are disabled. To configure and start the timers, the software must write a set of values to the registers that control the timers. The registers are described in Section14.5.

14.2.1 Mode 1: Processor-Independent PWM

Mode 1 is the Processor-Independent Pulse Width Modulation (PWM) mode, which generates pulses of a specified width and duty cycle, and which also provides a separate general-purpose timer/counter.

Figure13 is a block diagram of the Multi-Function Timer configured to operate in Mode 1. Timer/Counter I (TnCNT1) functions as the time base for the PWM timer. It counts down at the clock rate selected for the counter. When an underflow occurs, the timer register is reloaded alternately from the TnCRA and TnCRB register, and counting proceeds downward from the loaded value.

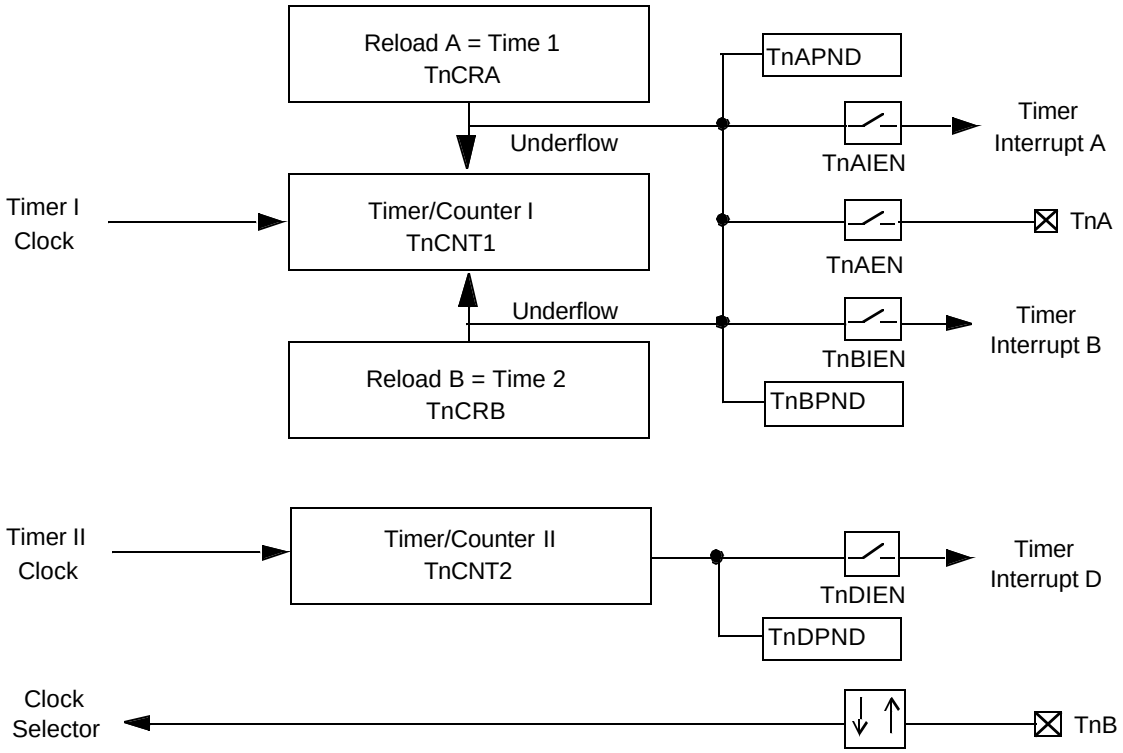


Figure 13. Mode 1: Processor-Independent PWM Block Diagram

On the first underflow, the timer is loaded from TnCRA, then from TnCRB on the next underflow, then from TnCRA again on the next underflow, and so on. Every time the counter is stopped and restarted, it always obtains its first reload value from TnCRA. This is true whether the timer is restarted upon reset, after entering Mode 1 from another mode, or after stopping and restarting the clock with the Timer/Counter I clock selector.

The timer can be configured to toggle the TnA output bit upon each underflow. This generates a clock signal on TnA with the width and duty cycle determined by the values stored in the TnCRA and TnCRB registers. This is a “processor-independent” PWM clock because once the timer is set up, no more action is required from the CPU to generate a continuous PWM signal.

The timer can be configured to generate separate interrupts upon reload from TnCRA and TnCRB. The interrupts can be enabled or disabled under software control. The CPU can determine the cause of each interrupt by looking at the TnAPND and TnBPND flags, which are set by the hardware upon each occurrence of a timer reload.

In Mode 1, Timer/Counter II (TnCNT2) can be used either as a simple system timer, an external event counter, or a pulse accumulate counter. The clock counts down using the clock selected with the Timer/Counter II clock selector. It generates an interrupt upon each underflow if the interrupt is enabled with the TnDIEN bit.

14.2.2 Mode 2: Dual Input Capture

Mode 2 is the Dual Input Capture mode, which measures the elapsed time between occurrences of external events, and which also provides a separate general-purpose timer/counter.

Figure14 is a block diagram of the Multi-Function Timer configured to operate in Mode 2. The time base of the capture timer depends on Timer/Counter I, which counts down using the clock selected with the Timer/Counter I clock selector. The TnA and TnB pins function as capture inputs. A transition received on the TnA pin transfers the timer contents to the TnCRA register. Similarly, a transition received on the TnB pin transfers the timer contents to the TnCRB register. Each input pin can be configured to sense either rising or falling edges.

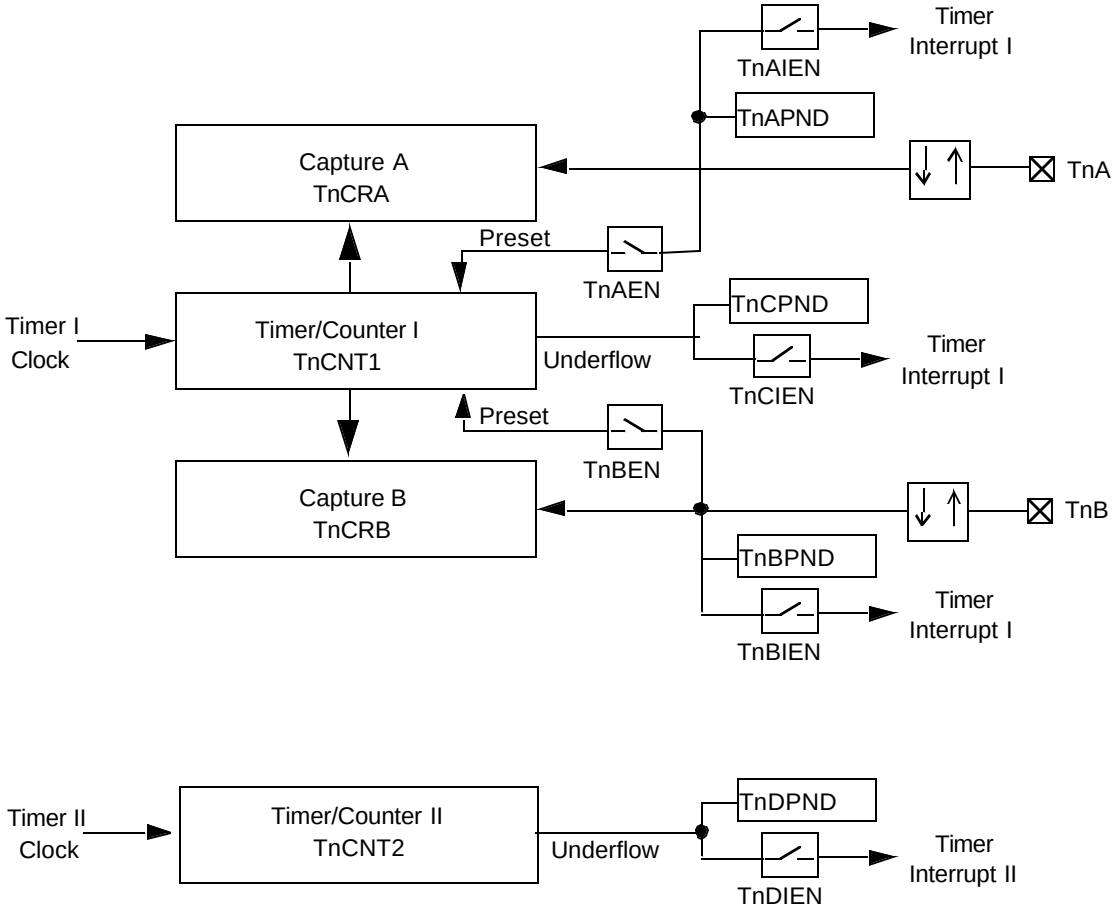


Figure 14. Mode 2: Dual Input Capture Block Diagram

The TnA and TnB inputs can be configured to preset the counter to FFFF hex upon reception of a valid capture event. In this case, the current value of the counter is transferred to the corresponding capture register and then the counter is preset to FFFF hex. Using this approach allows the software to determine the on-time and off-time and period of an external signal with a minimum of CPU overhead.

The values captured in the TnCRA register at different times reflect the elapsed time between transitions on the TnA pin. The same is true for the TnCRB register and the TnB pin. The input signal on TnA or TnB must have a pulse width equal to or greater than one system clock cycle.

There are three separate interrupts associated with the capture timer, each with its own enable bit and pending flag. The three interrupt events are reception of a transition on TnA, reception of a transition on TnB, and underflow of the TnCNT1 counter. The enable bits for these events are TnAIEN, TnBIEN, and TnCIEN, respectively.

In Mode 2, Timer/Counter II (TnCNT2) can be used as a simple system timer. The clock counts down using the clock selected with the Timer/Counter II clock selector. It generates an interrupt upon each underflow if the interrupt is enabled with the TnDIEN bit.

Neither Timer/Counter I (TnCNT1) nor Timer/Counter II (TnCNT2) can be configured to operate as an external event counter or to operate in the pulse accumulate mode because the TnB input is used as a capture input. Attempting to select one of these configurations will cause one or both counters to stop.

14.2.3 Mode 3: Dual Independent Timer/Counter

Mode 3 is the Dual Independent Timer mode, which generates system timing signals or counts occurrences of external events.

Figure15 is a block diagram of the Multi-Function Timer configured to operate in Mode 3. The timer is configured to operate as a dual independent system timer or dual external event counter. In addition, Timer/Counter I can generate a 50% duty cycle PWM signal on the TnA pin. The TnB pin can be used as an external event input or pulse accumulate input and can be used as the clock source for either Timer/Counter I or Timer/Counter II. Both counters can also be clocked by the prescaled system clock.

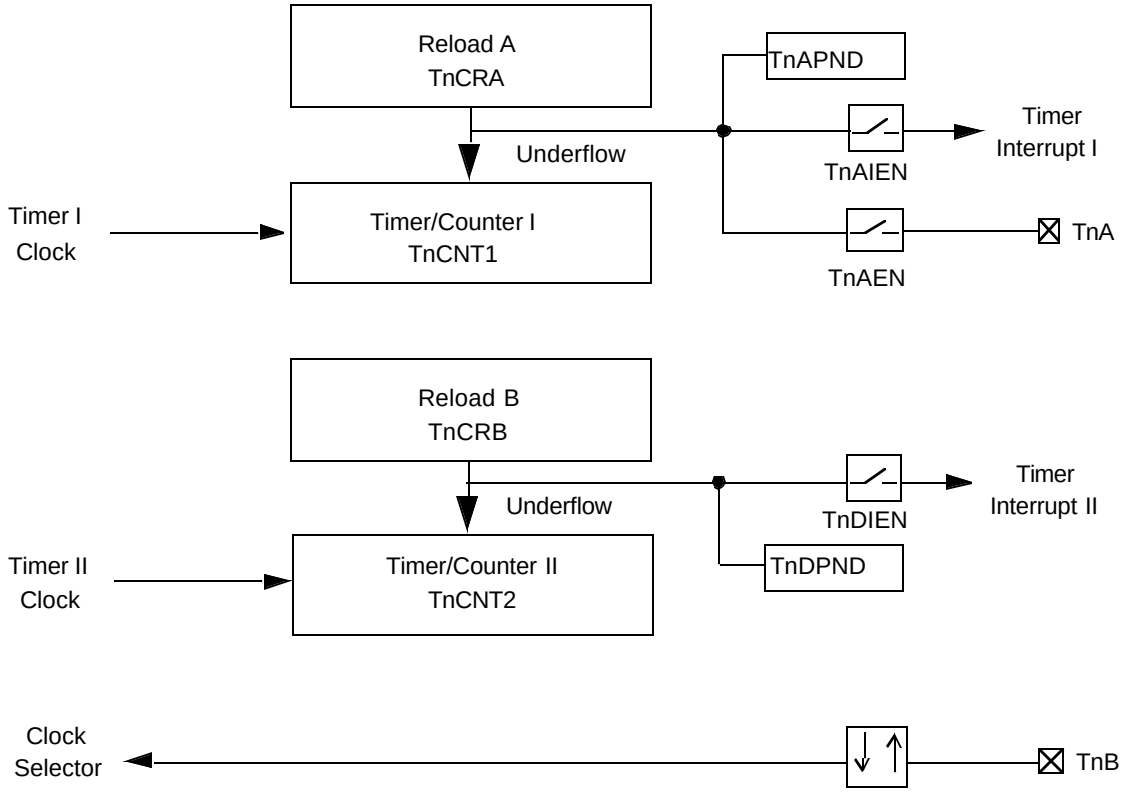


Figure 15. Mode 3: Dual Independent Timer/Counter Block Diagram

Timer/Counter I (TnCNT1) counts down at the rate of the selected clock. Upon underflow, it is reloaded from the TnCRA register and counting proceeds down from the reloaded value. In addition, the TnA pin is toggled on each underflow if this function is enabled by the TnAEN bit. The initial state of the TnA pin is software-programmable. When the TnA pin is toggled from low to high, it sets the TnCPND interrupt pending flag and also generates an interrupt if the interrupt is enabled by the TnAIEN bit.

Because TnA toggles on every underflow, a 50% duty cycle PWM signal can be generated on TnA without any further action from the CPU once the pulse train is initiated.

Timer/Counter II (TnCNT2) counts down at the rate of the selected clock. Upon underflow, it is reloaded from the TnCRB register and counting proceeds down from the reloaded value. In addition, each underflow sets the TnDPND interrupt pending flag and generates an interrupt if the interrupt is enabled by the TnDIEN bit.

14.2.4 Mode 4: Input Capture Plus Timer

Mode 4 is the Single Input Capture and Single Timer mode, which provides one external event counter and one system timer.

Figure16 is a block diagram of the Multi-Function Timer configured to operate in Mode 4. This mode offers a combination of Mode 3 and Mode 2 functions. Timer/Counter I is used as a system timer as in Mode 3 and Timer/Counter II is used as a capture timer as in Mode 2, but with a single input rather than two inputs.

Timer/Counter I (TnCNT1) operates the same as in Mode 3. It counts down at the rate of the selected clock. Upon underflow, it is reloaded from the TnCRA register and counting proceeds down from the reloaded value. The TnA pin is toggled on each underflow if this function is enabled by the TnAEN bit. When the TnA pin is toggled from low to high, it sets the TnCPND interrupt pending flag and also generates an interrupt if the interrupt is enabled by the TnAIEN bit. A 50% duty cycle PWM signal can be generated on TnA without any further action from the CPU once the pulse train is initiated.

Timer/Counter II (TnCNT1) counts down at the rate of the selected clock. The TnB pin functions as the capture input. A transition received on TnB transfers the timer contents to the TnCRB register. The input pin can be configured to sense either rising or falling edges.

The TnB input can be configured to preset the counter to FFFF hex upon reception of a valid capture event. In this case, the current value of the counter is transferred to the capture register and then the counter is preset to FFFF hex.

The values captured in the TnCRB register at different times reflect the elapsed time between transitions on the TnA pin. The input signal on TnB must have a pulse width equal to or greater than one system clock cycle.

There are two separate interrupts associated with the capture timer, each with its own enable bit and pending flag. The two interrupt events are reception of a transition on TnB and underflow of the TnCNT2 counter. The enable bits for these events are TnBIEN and TnDIEN, respectively.

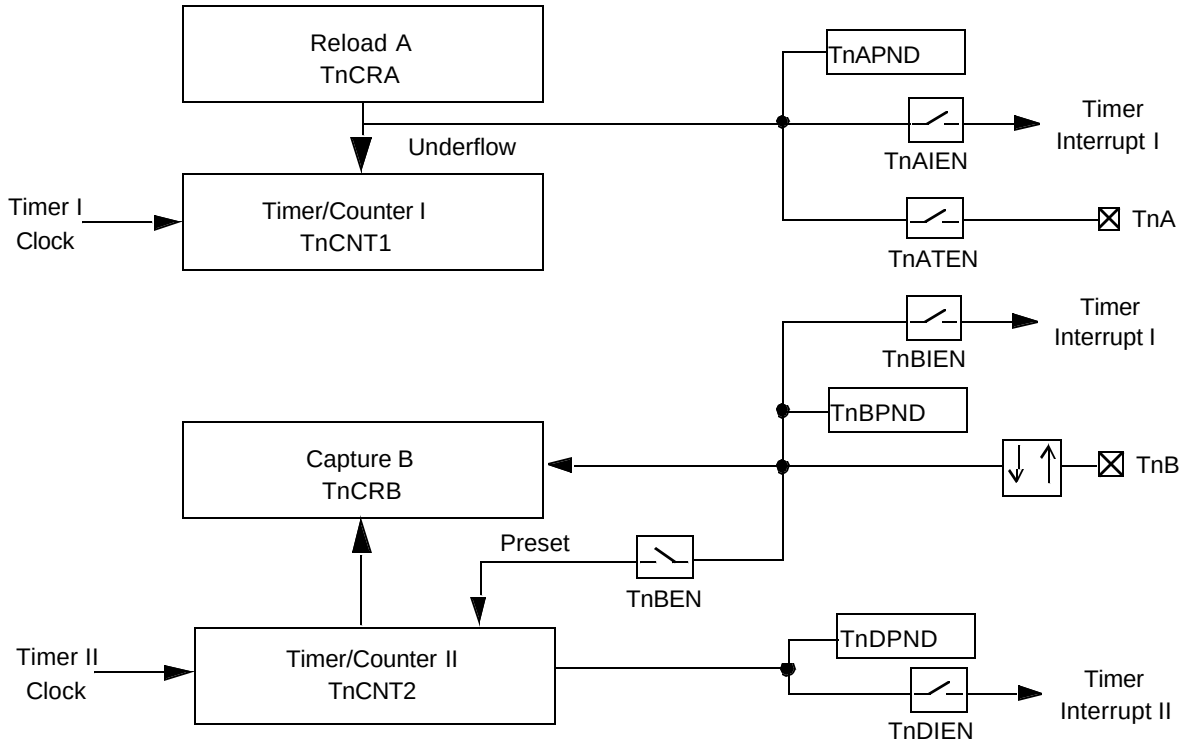


Figure 16. Mode 4: Input Capture Plus Timer Block Diagram

Neither Timer/Counter I (TnCNT1) nor Timer/Counter II (TnCNT2) can be configured to operate as an external event counter or to operate in the pulse accumulate mode because the TnB input is used as a capture input. Attempting to select one of these configurations will cause one or both counters to stop. In this mode, Timer/Counter II must be enabled at all times.

14.3 TIMER INTERRUPTS

Each Multi-Function Timer unit has four interrupt sources, designated A, B, C, and D. Interrupt sources A, B, and C are mapped into a single system interrupt called Timer Interrupt I, while interrupt source D is mapped into a system interrupt called Timer Interrupt II. Each of the four interrupt sources has its own enable bit and pending flag. The enable flags are named TnAIEN, TnBIEN, TnCIEN, and TnDIEN. The pending flags are named TnAPND, TnBPND, TnCPND, and TnDPND.

For Multi-Function Timer unit MFT1, Timer Interrupts I and II are system interrupts T1A and T1B (IRQ13 and IRQ12), respectively. For Multi-Function Timer unit MFT2, Timer Interrupts I and II are system interrupts T2A and T2B (IRQ11 and IRQ10), respectively.

Table15 shows the events that trigger interrupts A, B, C, and D in each of the four operating modes. Note that some interrupt sources are not used in some operating modes, as indicated by the notation “N/A” (Not Applicable) in the table.

14.4 TIMER I/O FUNCTIONS

Each Multi-Function Timer unit uses two I/O pins, called T1A and T1B (for Timer MFT1) or T2A and T2B (for Timer MFT2). The function of each pin depends on the timer operating mode and the TnAEN and TnBEN enable bits. Table16 shows the functions of the pins in each operating mode, and for each combination of enable bit settings.

When pin TnA is configured to operate as a PWM output (TnAEN = 1), the state of the pin is toggled on each under-

flow of the TnCNT1 counter. In this case, the initial value on the pin is determined by the TnAOUT bit. For example, to start with TnA high, the software should set the TnAOUT bit to 1 prior to enabling the timer clock. This option is available only when the timer is configured to operate in Mode 1, 3, or 4 (in other words, when TnCRA is not used in Capture mode).

Table 15 Timer Interrupts Overview

Sys. Int.	Interrupt pending flag	Mode 1	Mode 2	Mode 3	Mode 4
		PWM + Counter	Dual Input Capture + counter	Dual Counter	Single Capture + counter
Timer Int. I (TnA)	TnAPND	TnCNT1 reload from TnCRA	Input capture on TnA transition	TnCNT1 reload from TnCRA	TnCNT1 reload from TnCRA
	TnBPND	TnCNT1 reload from TnCRB	Input Capture on TnB transition	N/A	Input Capture on TnB transition
	TnCPND	N/A	TnCNT1 underflow	N/A	N/A
Timer Int. II (TnB)	TnDPND	TnCNT2 underflow	TnCNT2 underflow	TnCNT2 reload from TnCRB	TnCNT2 underflow

Table 16 Timer I/O Functions

I/O	TnAEN TnBEN	Mode 1	Mode 2	Mode 3	Mode 4
		PWM + Counter	Dual Input Capture + counter	Dual Counter	Single Capture + counter
TnA	TnAEN=0 TnBEN=X	No Output	Capture TnCNT1 into TnCRA	No Output toggle	No Output toggle
	TnAEN=1 TnBEN=X	Toggle Output on underflow of TnCNT1	Capture TnCNT1 into TnCRA and preset TnCNT1	Toggle Output on underflow of TnCNT1	Toggle Output on underflow of TnCNT1
TnB	TnAEN=X TnBEN=0	Ext. Event or Pulse Accumulate Input	Capture TnCNT1 into TnCRB	Ext. Event or Pulse Accumulate Input	Capture TnCNT2 into TnCRB
	TnAEN=X TnBEN=1	Ext. Event or Pulse Accumulate Input	Capture TnCNT1 into TnCRB and preset TnCNT1	Ext. Event or Pulse Accumulate Input	Capture TnCNT2 into TnCRB and preset TnCNT2

14.5 TIMER REGISTERS

The following CPU-accessible registers are used to control the Multi-Function Timers:

- Clock Prescaler Register (TnPRSC)
- Clock Unit Control Register (TnCKC)
- Timer/Counter I Register (TnCNT1)
- Timer/Counter II Register (TnCNT2)
- Reload/Capture A Register (TnCRA)
- Reload/Capture B Register (TnCRB)
- Timer Mode Control Register (TnCTRL)
- Timer Interrupt Control Register (TnICTL)
- Timer Interrupt Clear Register (TnICLR)

14.5.1 Clock Prescaler Register (TnPRSC)

The Clock Prescaler (TnPRSC) register is a byte-wide, read/write register that holds the current value of the 5-bit clock prescaler (CLKPS). This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved			CLKPS				

CLKPS Clock Prescaler. When the timer is configured to use the prescaled clock, the system clock is divided by CLKPS+1 to produce the timer clock. Thus, the system clock divide-by factor can range from 1 to 32.

14.5.2 Clock Unit Control Register (TnCKC)

The Clock Unit Control (TnCKC) register is a byte-wide, read/write register that selects the clock source for each timer/counter. This register is cleared upon reset, which disables the timer/counters. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		C2CSEL			C1CSEL		

C1CSEL Counter I Clock Select. This 3-bit field defines the clock mode for Timer/Counter I as follows:

000 = no clock (timer/counter I stopped)
001 = prescaled system clock
010 = external event on TnB (modes 1 and 3 only)

011 = pulse accumulate mode based on TnB (modes 1 and 3 only)
100 = slow clock *
other values = undefined

C2CSEL Counter II Clock Select. This 3-bit field defines the clock mode for Timer/Counter II as follows:

000 = no clock (Timer/Counter II stopped modes 1, 2, and 3 only)
001 = prescaled system clock
010 = external event on TnB (modes 1 and 3 only)
011 = pulse accumulate mode based on TnB (modes 1 and 3 only)
100 = slow clock *
other values = undefined

* Operation of the slow clock is determined by the CRCTRL.SCLK control bit, as described in Section11.6.1.

14.5.3 Timer/Counter I Register (TnCNT1)

The Timer/Counter I (TnCNT1) register is a word-wide, read/write register that holds the current count value for Timer/Counter I. The register contents are not affected by a reset and are unknown upon power-up.

14.5.4 Timer/Counter II Register (TnCNT2)

The Timer/Counter II (TnCNT2) register is a word-wide, read/write register that holds the current count value for Timer/Counter II. The register contents are not affected by a reset and are unknown upon power-up.

14.5.5 Reload/Capture A Register (TnCRA)

The Reload/Capture A (TnCRA) register is a word-wide, read/write register that holds the reload or capture value for Timer/Counter I. The register contents are not affected by a reset and are unknown upon power-up.

14.5.6 Reload/Capture B Register (TnCRB)

The Reload/Capture B (TnCRB) register is a word-wide, read/write register that holds the reload or capture value for Timer/Counter II. The register contents are not affected by a reset and are unknown upon power-up.

14.5.7 Timer Mode Control Register (TnCTRL)

The Timer Mode Control (TnCTRL) register is a byte-wide, read/write register that sets the operating mode of the timer/counter and the TnA and TnB pins. This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved	TnAOUT	TnBEN	TnAEN	TnBEDG	TnAEDG	MDSEL	

MDSEL Mode Select. This 2-bit field sets the operating mode of the timer/counter as follows:

00 = Mode 1: PWM plus system timer
01 = Mode 2: Dual Input Capture plus system timer
10 = Mode 3: Dual Timer/Counter
11 = Mode 4: Single Input Capture and Single Timer

TnAEDG TnA Edge Polarity. When cleared (0), input pin TnA is sensitive to falling edges (high to low transitions). When set (1), input pin TnA is sensitive to rising edges (low to high transitions).

TnBEDG TnB Edge Polarity. When cleared (0), input pin TnB is sensitive to falling edges (high to low transitions). When set (1), input pin TnB is sensitive to rising edges (low to high transitions). In pulse accumulate mode, when this bit is set (1), the counter is enabled only when TnB is high; when this bit is cleared (0), the counter is enabled only when TnB is low.

TnAEN TnA Enable. When set (1), the TnA pin is enabled to operate as a preset input or as a PWM output, depending on the timer operating mode. In Mode 2 (Dual Input Capture), a transition on the TnA pin presets the TnCNT1 counter to FFFF hex. In the other modes, TnA functions as a PWM output. When this bit is cleared (0), operation of the pin for the timer/counter is disabled.

TnBEN TnB Enable. When set (1), the TnB pin in enabled to operate in Mode 2 (Dual Input Capture) or Mode 4 (Single Input Capture and Single Timer). A transition on the TnB pin presets the corresponding timer/counter to FFFF hex (TnCNT1 in Mode 2 or TnCNT2 in Mode 4). When this bit is cleared (0), operation of the pin for the timer/counter is disabled. This bit setting has no effect in Mode 1 or Mode 3.

TnAOUT TnA Output Data. This is a status bit that indicates the current state of the TnA pin when the pin is used as a PWM output. When set (1), the TnA pin is high; when cleared (0), the TnA pin is low. The hardware sets and clears this bit, but the software can also read or write this bit at any time and thus control the state of the output pin. In case of conflict, a software write has precedence over a hardware update. This bit setting has no effect when TnA is used as an input.

14.5.8 Timer Interrupt Control Register (TnICTL)

The Timer Interrupt Control (TnICTL) register is a byte-wide, read/write register that contains the interrupt enable bits and interrupt pending bits for the four timer interrupt sources, designated A, B, C, and D. The condition that causes each type of interrupt depends on the operating mode, as shown in Table15.

This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
TnDIEN	TnCIEN	TnBIEN	TnAIEN	TnDPND	TnCPND	TnBPND	TnAPND

TnAPND Timer Interrupt Source A Pending. When this bit is set (1), it indicates that timer interrupt condition “A” has occurred. When this bit is cleared (0), it indicates that the interrupt condition has not occurred. For an explanation of interrupt conditions A, B, C, and D, see Table15

This bit can be set by the hardware or by the software. To clear this bit, the software must use the Timer Interrupt Clear Register (TnI-

CLR). Any attempt by the software to directly write a 0 to this bit is ignored.

TnBPND Timer Interrupt Source B Pending. See the description of TnAPND.

TnCPND Timer Interrupt Source C Pending. See the description of TnAPND.

TnDPND Timer Interrupt Source D Pending. See the description of TnAPND.

TnAIEN Timer Interrupt A Enable. When set (1), this bit enables an interrupt on each occurrence of interrupt condition “A.” When cleared (0), an occurrence of interrupt condition “A” does not generate an interrupt to the CPU, but still sets the associated pending flag (TnAPND). For an explanation of interrupt conditions A, B, C, and D, see Table15.

TnBIEN Timer Interrupt B Enable. See the description of TnAIEN.

TnCIEN Timer Interrupt C Enable. See the description of TnAIEN.

TnDIEN Timer Interrupt D Enable. See the description of TnAIEN.

14.5.9 Timer Interrupt Clear Register (TnICLR)

The Timer Interrupt Clear (TnICLR) register is a byte-wide, write-only register that allows the software to clear the TnAPND, TnBPND, TnCPND, and TnDPND bits in the Timer Interrupt Control (TnICTRL) register. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved				TnDCLR	TnCCLR	TnBCLR	TnACLR

TnACLR Timer Pending A Clear. When written with a 1, the Timer Interrupt Source A Pending bit (TnAPND) is cleared in the Timer Interrupt Control register (TnICTL). Writing a 0 to the TnACLR bit has no effect.

TnBCLR Timer Pending B Clear. See the description of TnACLR.

TnCCLR Timer Pending C Clear. See the description of TnACLR.

TnDCLR Timer Pending D Clear. See the description of TnACLR.

15.0 MICROWIRE/SPI

MICROWIRE/PLUS is a synchronous serial communications protocol, originally implemented in National Semiconductor's COPS™ and HPC™ families of microcontrollers to minimize the number of connections, and therefore the cost, of communicating with peripherals.

The device has an enhanced MICROWIRE interface module (MWSPI) that can communicate with all peripherals that conform to MICROWIRE or Serial Peripheral Interface (SPI) specifications. This enhanced MICROWIRE interface is capable of operating as either a master or slave. Figure17 shows a typical enhanced MICROWIRE interface application.

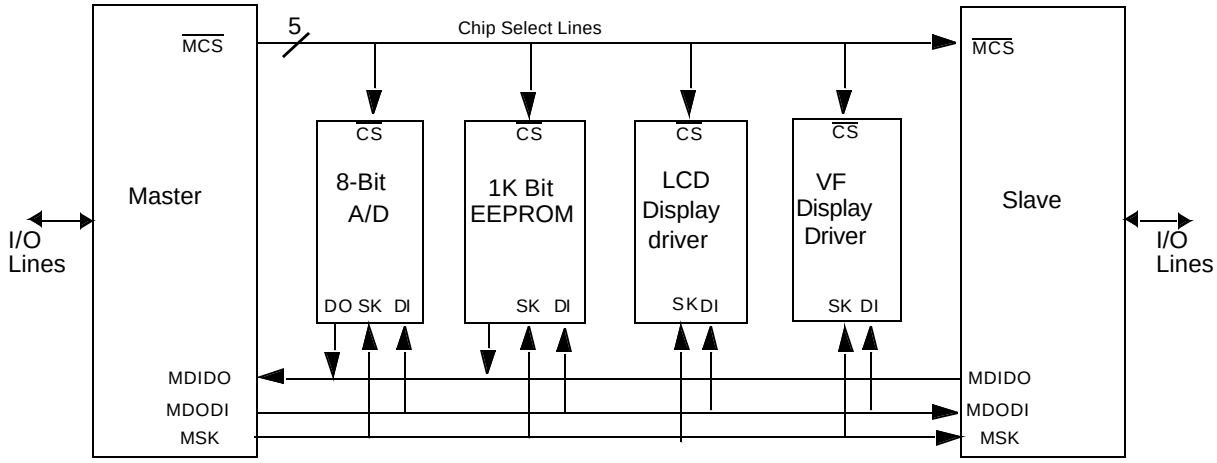


Figure 17. MICROWIRE Interface

The enhanced MICROWIRE interface module includes the following features:

- — Programmable operation as a Master or Slave
- — Programmable shift-clock frequency (master only)
- — 8-bit serial I/O data shift register
- — Two modes of clocking data
- — Serial clock can be low or high when idle
- — Double read buffer (master mode only)
- — Busy flag, Read Buffer Full flag, and Overrun flag for polling and as interrupt sources
- — Supports multiple masters
- — Maximum bit rate of 4M bits/second (master and slave)
- — Supports very low-end slaves with the Slave Ready output
- — Echo back enable/disable (Slave only)

15.1 MICROWIRE OPERATION

The MICROWIRE interface allows several devices to be connected on one three-wire system. At any given time, one of these devices operates as the master while all other devices operate as slaves.

The master device supplies the synchronous clock (MSK) for the serial interface and initiates the data transfer. The slave devices respond by sending (or receiving) the requested data. Each slave device uses the master's clock for serially shifting data out (or in), while the master shifts the data in (or out).

The three-wire system includes: the serial data in signal (MDIDO for master mode, MDODI for slave mode), the serial data out signal (MDODI for master mode, MDIDO for slave mode) and the serial clock (MSK).

In slave mode, an optional fourth signal ($\overline{MCS}$) may be used to enable the slave transmit. At any given time, only one slave can respond to the master. Each slave device has its own chip select signal ($\overline{MCS}$) for this purpose.

The MICROWIRE interface allows the device to operate either as a master or slave. This is configured via the MMNS bit.

Figure18 shows a block diagram of the enhanced MICROWIRE serial interface in the device.

15.1.1 Shifting

The MICROWIRE interface is a full duplex transmitter/receiver. An 8-bit shifter is used for both transmitting and receiving. The transmitted data is shifted out through MDODI pin (master mode) or MDIDO pin (slave mode), starting with the most significant bit. At the same time, the received data is shifted in through MDIDO pin (master mode) or MDODI pin (slave mode), also starting with the most significant bit first.

The shift in and shift out are controlled by the MSK clock. In each clock cycle of MSK, one bit of data is transmitted/received. The 8-bit shifter is accessible via the MWDAT register. Reading the MWDAT register returns the value in the read buffer. Writing to the MWDAT register updates the 8-bit shifter.

15.1.2 Reading

The enhanced MICROWIRE interface implements a double buffer on read. As illustrated in Figure18, the double read buffer consists of the 8-bit shifter and a buffer, called the read buffer.

The 8-bit shifter loads the read buffer with new data when the data transfer sequence is completed and previous data in the

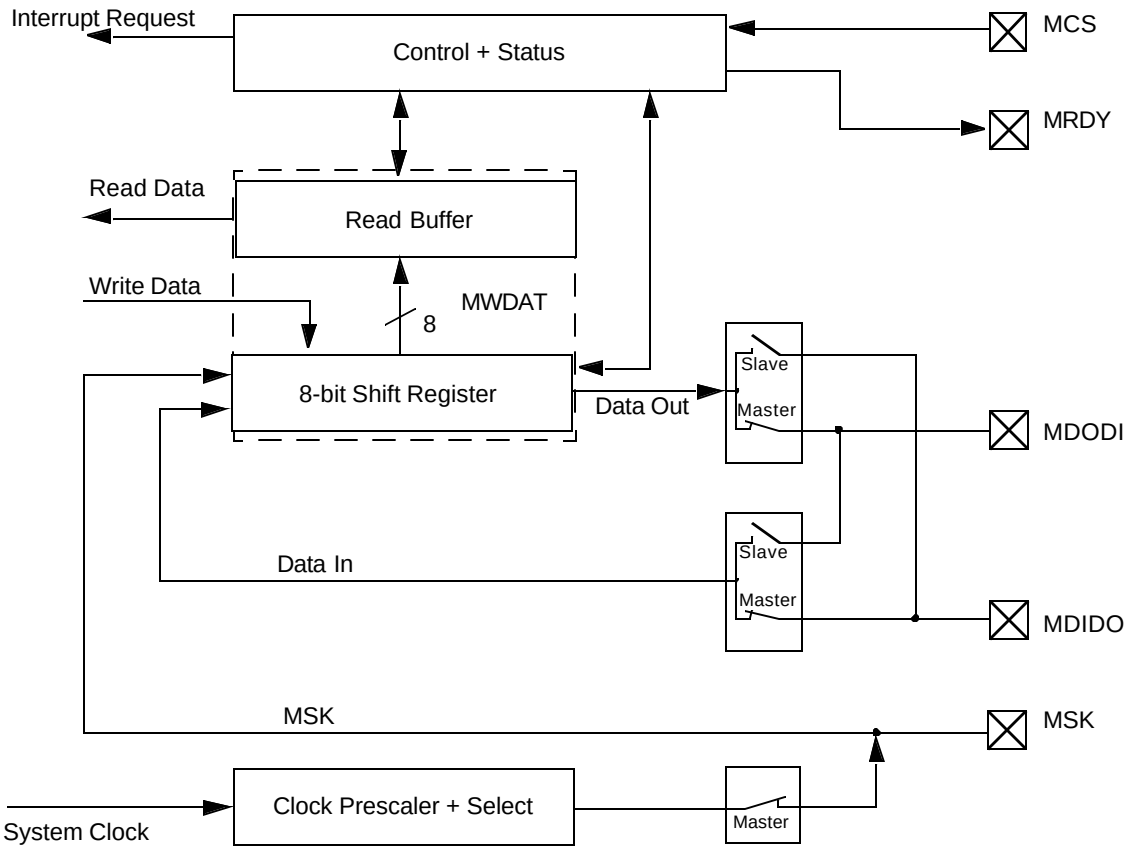


Figure 18. MICROWIRE Block Diagram

read buffer has been read. In master mode, an Overrun error occurs when the read buffer is full, the 8-bit shifter is full and a new data transfer sequence starts.

The “Receive Buffer Full” (MRBF) bit indicates if the MWDAT register holds valid data. The MOVR bit indicates that an overrun condition has occurred.

15.1.3 Writing

The “MICROWIRE Busy” (MBSY) bit indicates whether the MWDAT register can be written. All write operations to the MWDAT register update the shifter while the data contained in the read buffer is not affected. Undefined results will occur if the MWDAT register is written while the MBSY bit is set to 1.

15.1.4 Clocking Modes

Two clocking modes are supported: the normal mode and the alternate mode.

In the normal mode, the output data is shifted out on the rising edge of MSK on the MDODI pin (master mode) or MDIDO (slave mode). The input data, which is received via MDIDO pin (master mode) or the MDODI pin (slave mode), is sampled on the fallowing edge of MSK.

In the alternate mode, the output data is shifted out on the rising edge of MSK on the MDODI pin (master mode) or MDIDO pin (slave mode). The input data, which is received via MDIDO pin (master mode) or MDODI pin (slave mode), is sampled on the falling edge of MSK.

The clocking modes are selected with the MSKM bit. The MIDL bit allows selection of the value of MSK when it is idle (when there is no data being transferred). Various MSK clock frequencies can be programmed via the MCDV bits. Figures 19, 20, 21, and 22 show the data transfer timing for the normal and the alternate modes with the MIDL bit equal to 0 and equal to 1.

Note that when data is shifted out on MDODI (master mode) or MDIDO (slave mode) on the leading edge of the MSK clock, bit 6 is shifted out on the second leading edge of the MSK clock. When data are shifted out on MDODI (master mode) or MDIDO (slave mode) on the trailing edge of MSK, bit 6 is shifted out on the first trailing edge of MSK.

15.2 MASTER MODE

In Master mode, the MSK pin is an output for the shift clock, MSK. When data is written to the 8-bit shifter (MWDAT register), eight clocks are generated to shift the eight bits of data and then MSK goes idle again. The MSK idle state can be either high or low, depending on the MIDL bit.

If MDIDO is sampled on the leading edge of MSK, a sequence of eight clock is generated after a delay that may range from half a period of MSK to one and a half periods of MSK. If MDIDO is sampled on the trailing edge of MSK, a se-

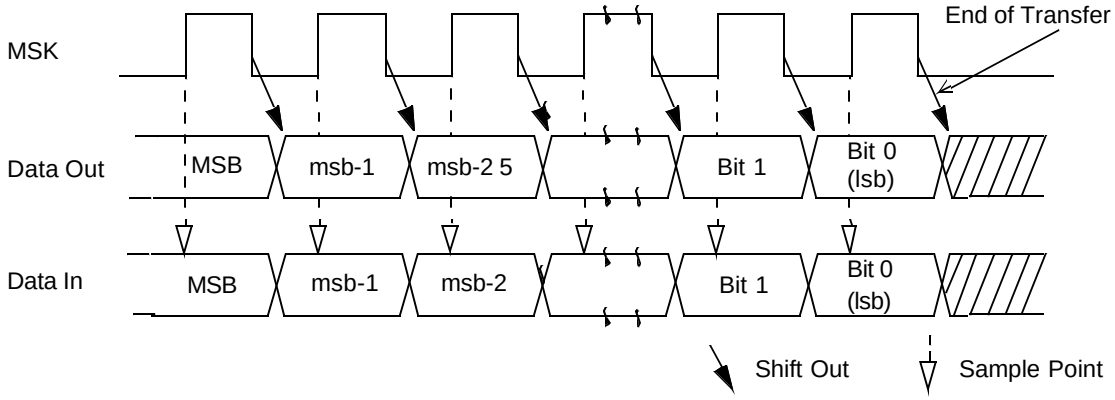


Figure 19. Normal Mode, MIDL Bit = 0

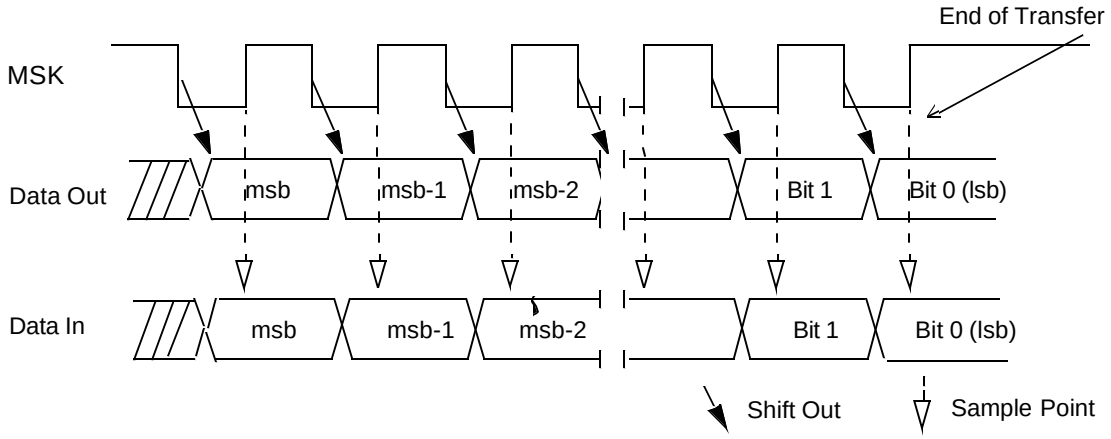


Figure 20. Normal Mode, MIDL Bit = 1

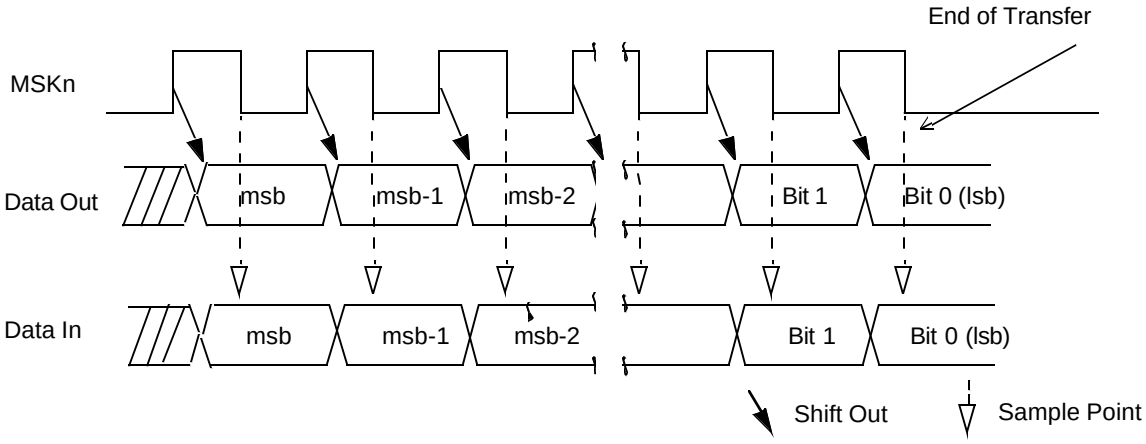


Figure 21. Alternate Mode, MIDL Bit = 0

quence of eight clocks is generated after a delay that may range from zero to one period of MSK.

15.3 SLAVE MODE

In Slave mode, the MSK pin is an input for the shift clock MSK. MDIDO and $\overline{\text{MRDY}}$ (an optional signal) are placed in TRI-STATE mode when $\overline{\text{MCS}}$ is inactive. Data transfer is enabled when $\overline{\text{MCS}}$ is active. MWCTL3.MBFL must be cleared to 0.

The slave starts driving MDIDO when $\overline{\text{MCS}}$ is activated. The most significant bit (MSB) of the 8-bit shifter is output onto the MDIDO pin first. After eight clocks, the data transfer is completed.

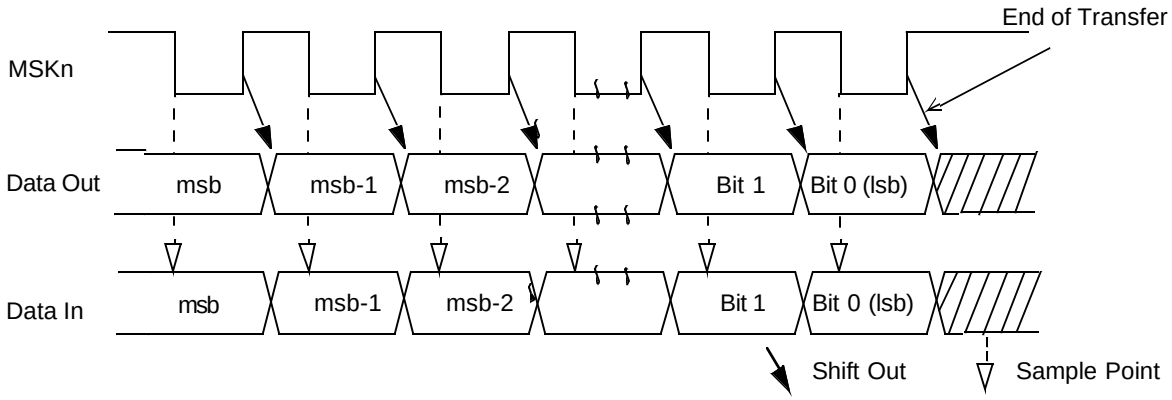


Figure 22. Alternate Mode, MIDL Bit = 1

If a new shift process starts before MWDAT was written, i.e., while MWDAT does not contain any valid data, and the "Echo Enable" (MECH) bit is set to 1, the data received from MDO-DI is transmitted on MDIDO in addition to being shifted to MWDAT. If the MECH bit is cleared to 0, the data transmitted on MDIDO is the data held in the MWDAT register, regardless of its validity. The master may negate the $\overline{MCS}$ signal to synchronize the bit count between the master and the slave. In the case that the slave is the only slave in the system, $\overline{MCS}$ can be tied to V_{SS}.

An additional output signal, $\overline{MRDY}$, may be used. This signal supports very low-end slaves by indicating to the master the current status of the MICROWIRE channel.

Upon reset, $\overline{MRDY}$ is inactive. Before transfer starts, $\overline{MRDY}$ should be asserted by setting the MWCTL3.MRDY bit. $\overline{MRDY}$ is de-asserted when the data transfer of one data byte has been completed.

$\overline{MRDY}$ is asserted again only by writing 1 to the MRDY bit. This should be done after the MWDAT register is read.

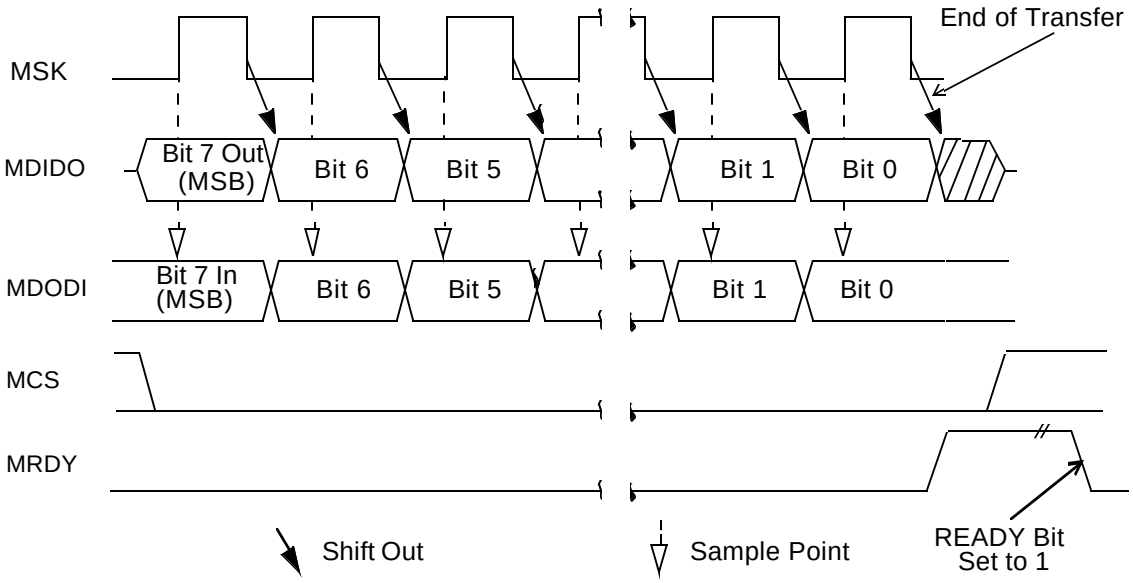


Figure 23. Slave, Normal Mode, MIDL Bit = 0, MRBF Bit = 1

15.4 INTERRUPT GENERATION

An interrupt is generated in any of the following cases:

- When the read buffer is full (MRBF=1) and the "Enable Interrupt for Read" bit is set (MEIR=1).
- Whenever the shifter is not busy, i.e. the MBSY bit is cleared (MBSY=0) and the "Enable Interrupt for Write" bit is set (MEIW=1).
- When an overrun condition occurs (MOVR is set to 1) and the "Enable Interrupt on Overrun" bit is set (MEIO=1). This usage is restricted to master mode.

Figure24 illustrates the various interrupt capabilities of this module.

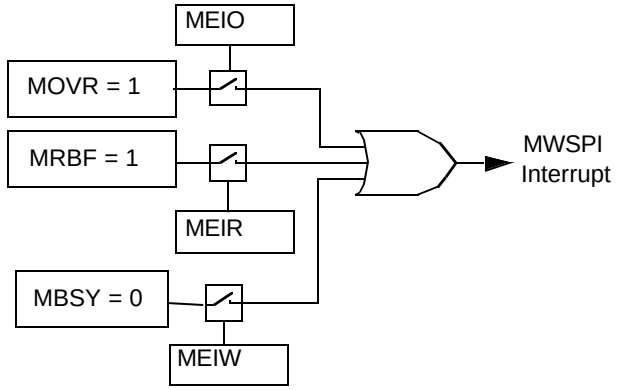


Figure 24. MWSPI Interrupts

15.5 MICROWIRE INTERFACE REGISTERS

The software interacts with the MICROWIRE interface by accessing the MICROWIRE registers. There are five such registers:

- MICROWIRE Data Register (MWDAT)
- MICROWIRE Control 1 Register (MWCTL1)
- MICROWIRE Control 2 Register (MWCTL2)
- MICROWIRE Control 3 Register (MWCTL3)
- MICROWIRE Status Register (MWSTAT)

15.5.1 MICROWIRE Data Register (MWDAT)

The MWDAT register is a byte-wide, read/write register used to transmit and receive data through the MDODI and MDIDO pins. Figure25 shows the hardware structure of the register.

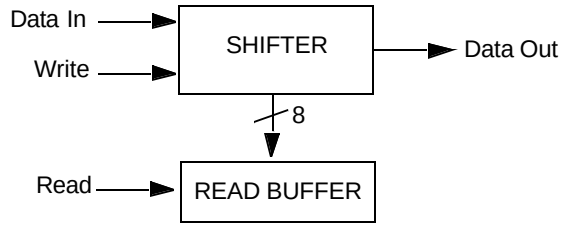


Figure 25. MWDAT Register Structure

Reading the MWDAT register returns the data received through the MDIDO pin in master mode or the MDODI pin in slave mode. This data is read from a buffer. After the CPU reads the buffer, if new data is ready in the shifter, the hardware immediately transfers the new value from the shifter to the buffer.

Writing the MWDAT register loads the shifter directly without buffering, thus overwriting any existing data in the shifter. The data byte is shifted out through the MDODI pin in master mode or through the MDIDO pin in slave mode, most significant bit first.

The MWDAT register should be accessed only after a MICROWIRE interrupt. Before the CPU reads the register, the Read Buffer Full status bit should be set (MWSTAT.MRBF=1). Before the CPU writes the register, the MICROWIRE Busy status bit should be cleared (MWSTAT.MBSY=0).

For normal operation, the CPU should first read the received data, thus allowing pending received data to be transferred from the shifter into the MWDAT register, before it writes the next transmit data into the MWDAT address, which loads the shifter directly. (MWSTAT.MBSY=0 is an indicator that the shifter is empty and ready to receive the next transmit data.) Otherwise, a second received data byte pending in the shifter may get overwritten by the transmit byte.

The MWDAT register contains unknown data following a reset operation.

15.5.2 MICROWIRE Control 1 Register (MWCTL1)

The MWCTL1 register is a byte-wide, read/write register that controls the operating mode of the MICROWIRE interface module. Upon reset, all non-reserved bits are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved				MEN	MIDL	MSKM	MMNS

MMNS MICROWIRE Master/Slave Select. When cleared to 0, the device operates as a slave. When set to 1, the device operates as the master.

MSKM MICROWIRE Clocking Mode. When cleared to 0, the device uses the normal clocking mode. When set to 1, the device uses the alternate clocking mode. In the normal mode, the output data is clocked out on the falling edge of MSK and the input data is sampled on the rising edge of MSK. In the alternate mode, the output data is clocked out on the rising edge of MSK and the input data is sampled on the falling edge of MSK.

MIDL MICROWIRE Idle. This bit sets the value of the MSK output when the MICROWIRE interface is idle: 0 for low or 1 for high. This bit should be changed only when the MICROWIRE interface module is disabled (MEN=0) or when no bus transaction is in progress (MWSTAT.MBSY=0).

MEN MICROWIRE Enable. This bit enables (1) or disables (0) the MICROWIRE interface module. Clearing this bit disables the module, clears the status bits in the MICROWIRE status register (the MBSY, MRBF, and MOVR flags in MWSTAT), and places the MICROWIRE interface pins in the states described in Table17.

Table 17 Pin Values with MICROWIRE Disabled

MSK	Master: MnIDL Bit Slave: input
$\overline{\text{MCS}}$	Input
MDIDO	Master: input Slave: TRI-STATE
MDODI	Master: known Value Slave: input
$\overline{\text{MRDY}}$	TRI-STATE

15.5.3 MICROWIRE Control 2 Register (MWCTL2)

The MWCTL2 register is a byte-wide, read/write register that controls the clock divider used to generate the MSK shift clock from the system clock. The MICROWIRE interface module generates the MSK clock only in master mode, so this register is ignored in slave mode. Upon reset, all non-reserved bits are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
MDV1	MCDV						

MCDV — MICROWIRE Clock Divider Value. This 7-bit field specifies the divide-by factor used for generating the MSK shift clock from the system clock. The divide-by factor is 2*(MCDV+1). This allows selection of a divide-by ratio from 2 to 256. This field is ignored in slave mode (MWCTL1.MMNS=0) or if the MDV1 bit is set.

MDV1 — MICROWIRE Clock Divide-by-1. When cleared to 0, the MSK shift clock rate is determined by the MCDV field. When set to 1, the divide-by factor is 1 and the MSK clock operates at the same rate as the system clock. This bit is ignored in slave mode (MWCTL1.MMNS=0).

15.5.4 MICROWIRE Control 3 Register (MWCTL3)

The MWCTL3 register is a byte-wide, read/write register that controls the $\overline{\text{MRDY}}$ pin and enables or disables the MICROWIRE interrupts and echo back function. Upon reset, all non-reserved bits are cleared to 0. When the software writes to this register, the reserved bits must be written with 0 for the MICROWIRE interface to function properly. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		MEIW	MEIR	MEIO	MECH	MRDY	MBFL

MBFL — MICROWIRE Buffer Length. This bit sets the data buffer length in slave mode to either one or two bytes.

When MBFL is cleared to 0, the buffer length is one byte. The $\overline{\text{MRDY}}$ pin goes high when transmission or reception of one data byte is completed. This is the proper setting for slave mode.

When MBFL is set to 1, the buffer length is two bytes. The $\overline{\text{MRDY}}$ pin goes high when the read buffer is full and a subsequent transmission or reception of a data byte is completed. This setting is only intended for master mode and should not be used for slave mode.

MRDY — MICROWIRE Ready. This write-only bit allows the software to control the $\overline{\text{MRDY}}$ pin when the device operates in slave mode. Writing a 1 to this bit position asserts the $\overline{\text{MRDY}}$ pin (makes it go low). This should be done only when the MICROWIRE interface is idle (MWSTAT.MBSY=0).

The hardware changes the $\overline{\text{MRDY}}$ pin from low to high at the end of a data transfer, as defined by the MBFL bit.

Writing a 0 to MRDY has no effect. The MRDY bit is ignored entirely in master mode (MWCTL1.MMNS=1).

Reading this bit returns an unknown value.

MECH — MICROWIRE Echo Back. This bit enables (1) or disables (0) the echo back function in slave mode. This bit should be written only when the MICROWIRE interface is idle (MWSTAT.MBSY=0). The MECH bit is ignored in master mode.

In the echo back mode, MDODI is transmitted (echoed back) on MDIDO if MWDAT does not contain any valid data. With the echo back function disabled, the data held in the MWDAT register is transmitted on MDIDO, whether or not the data is valid.

MEIO — MICROWIRE Enable Interrupt on Overrun. This bit enables or disables the overrun error interrupt. When set to 1, an interrupt is generated when the Receive Overrun Error flag (MWSTAT.MOVR) is set. Otherwise, no interrupt is generated when an overrun error occurs. This bit should only be enabled in master mode.

MEIR — MICROWIRE Enable Interrupt for Read. When set to 1, an interrupt is generated when the Read Buffer Full flag (MWSTAT.MRBF) is set. Otherwise, no interrupt is generated when the read buffer is full.

MEIW — MICROWIRE Enable Interrupt for Write. When set to 1, an interrupt is generated when the Busy bit (MWSTAT.MBSY) is cleared, which indicates that a data transfer sequence has been completed and the read buffer is ready to receive the new data. Otherwise, no interrupt is generated when the Busy bit is cleared.

15.5.5 MICROWIRE Status Register (MWSTAT)

The MICROWIRE Status Register is a byte-wide, read-only register that shows the current status of the MICROWIRE interface module. Upon reset, all non-reserved bits are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved				MOVR	Reserved	MRBF	MBSY

MBSY — MICROWIRE Busy. This bit, when set to 1, indicates that the MICROWIRE shifter is busy.

In master mode, MBSY is set to 1 when the MWDAT register is written. In slave mode, this bit is set to 1 on the first leading edge of MSK when $\overline{\text{MCS}}$ is asserted or when the MWDAT register is written, whatever occurs first.

In both master and slave modes, this bit is cleared to 0 when the MICROWIRE data transfer sequence is completed and the read buffer is ready to receive the new data; in other words, when the previous data held in the read buffer has already been read.

If the previous data in the read buffer has not been read and a new data has been received into the shift register, the MBSY will not be cleared, as the transfer could not be completed. This is because the contents of the shift

register could not be copied into the read buffer.

MRBF — MICROWIRE Read Buffer Full. This bit, when set to 1, indicates that the MICROWIRE read buffer is full and ready to be read by the software. It is set to 1 when the shifter loads the read buffer, which occurs upon completion of a transfer sequence if the read buffer is empty.

The MRBF bit is updated when the MWDAT register is read. At that time, the MRBF bit is cleared to 0 if the shifter does not contain any new data (in other words, the shifter is not receiving data or has not yet received a full byte of data). The MRBF bit remains set to 1 if the shifter already holds new data at the time that MWDAT is read. In that case, MWDAT is immediately reloaded with the new data and is ready to be read by the software.

MOVR — MICROWIRE Receive Overrun Error. This bit, when set to 1 in master mode, indicates that a receive overrun error has occurred. This error occurs when the read buffer is full, the 8-bit shifter is full, and a new data transfer sequence starts. This bit is undefined in slave mode.
The MOVR bit, once set, remains set until cleared by the software. The software clears this bit by writing a 1 to its bit position. Writing a 0 to this bit position has no effect. No other bits in the MWSTAT register are affected by a write operation to the register.

16.0 USART

The USART module is a full-duplex Universal Synchronous/ Asynchronous Receiver/Transmitter that supports a wide range of software-programmable baud rates and data formats. It handles automatic parity generation and several error detection schemes. There are one or two independent USART modules in each device, depending on the package type.

Each USART module offers the following features:

- — Full-duplex double-buffered receiver/transmitter
- — Synchronous or asynchronous operation
- — Programmable baud rate from SYS_CLK/ [2*(1+2^11)*16] up to SYSCLK/2 for USART configured to run in synchronous mode
- — Programmable baud rate from SYS_CLK/ [16*(1+2^11)*16] up to SYSCLK/16 for USART configured to run in asynchronous mode
- — Programmable framing formats: seven, eight, or nine data bits; one or two stop bits; and odd, even, mark, space, or no parity
- — Hardware parity generation for data transmission and parity check for data reception
- — Interrupts on “transmit ready” and “receive ready” conditions, separately enabled
- — Software-controlled break transmission and detection
- — Internal diagnostic capability
- — Automatic detection of parity, framing, and overrun errors

16.1 FUNCTIONAL OVERVIEW

Figure26 is a block diagram of the USART module showing the basic functional units in the USART:

- — Transmitter
- — Receiver
- — Baud Rate Generator
- — Control and Error Detection

Note: In the description of the USART, the lower-case letter “n” represents the USART number. For example, TDXn means TDX1 or TDX2.

The Transmitter block consists of an 8-bit transmit shift register and an 8-bit transmit buffer. Data bytes are loaded in parallel from the buffer into the shift register and then shifted out serially on the TDXn pin.

The Receiver block consists of an 8-bit receive shift register and an 8-bit receive buffer. Data is received serially on the RDXn pin and shifted into the shift register. Once eight bits have been received, the contents of the shift register are transferred in parallel to the receive buffer.

The Transmitter and Receiver blocks both contain extensions for 9-bit data transfers, as required by the 9-bit and loopback operating modes.

The Baud Rate Generator generates the clock for the synchronous and asynchronous operating modes. It consists of two registers and a two-stage counter. The registers are used to specify a prescaler value and a baud rate divisor. The first stage of the counter divides the USART clock based on the value of the programmed prescaler to create a slower clock. The second stage of the counter divides the output of the first stage based on the programmed baud rate divisor to create the baud rate clock.

The Control and Error Detection block contains the USART control registers, control logic, error detection circuit, parity generator/checker, and interrupt generation logic. The control registers and control logic determine the data format, mode of operation, clock source, and type of parity used. The error detection circuit generates parity bits and checks for parity, framing, and overrun errors.

16.2 USART OPERATION

The USART has two basic modes of operation: synchronous and asynchronous. In addition, there are two special-purpose synchronous and asynchronous modes, called attention and diagnostic. This section describes the operating modes of the USART.

16.2.1 Asynchronous Mode

The asynchronous mode of the USART enables the device to communicate with other devices using just two communication signals: transmit and receive.

In the asynchronous mode, the transmit shift register (TSFT) and the transmit buffer (UnTBUF) double-buffer the data for transmission. To transmit a character, a data byte is loaded in the UnTBUF register. The data is then transferred to the TSFT register. While the TSFT is shifting out the current character (LSB first) on the TDXn pin, the UnTBUF register is loaded by software with the next byte to be transmitted. When TSFT finishes transmission of the last stop bit of the current frame, the contents of UnTBUF are transferred to the TSFT register and the Transmit Buffer Empty flag (UnTBE) is set. The UnTBE flag is automatically reset by the USART when the software loads a new character into the UnTBUF register. During transmission, the UnXMIP bit is set high by the USART. This bit is reset only after the USART has sent the last stop bit of the current character and the UnTBUF register is empty. The UnTBUF register is a read/write register. The TSFT register is not user accessible.

In asynchronous mode, the input frequency to the USART is 16 times the baud rate. In other words, there are 16 clock cycles per bit time. In asynchronous mode the baud rate generator is always the USART clock source.

The receive shift register (RSFT) and the receive buffer (UnRBUF) double buffer the data being received. The USART receiver continuously monitors the signal on the RDXn pin for a low level to detect the beginning of a start bit. Upon sensing this low level, the USART waits for seven input clock cycles and samples again three times. If all three samples still indicate a valid low, then the receiver considers this to be a valid start bit, and the remaining bits in the character frame are each sampled three times, around the mid-bit position. For any bit following the start bit, the logic value is found by majority voting, i.e. the two samples with the same value define the value of the data bit. Figure27 illustrates the process of start bit detection and bit sampling.

Serial data input on the RDXn pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the UnRBUF register and the Receive Buffer Full flag (UnRBF) is set. The UnRBF

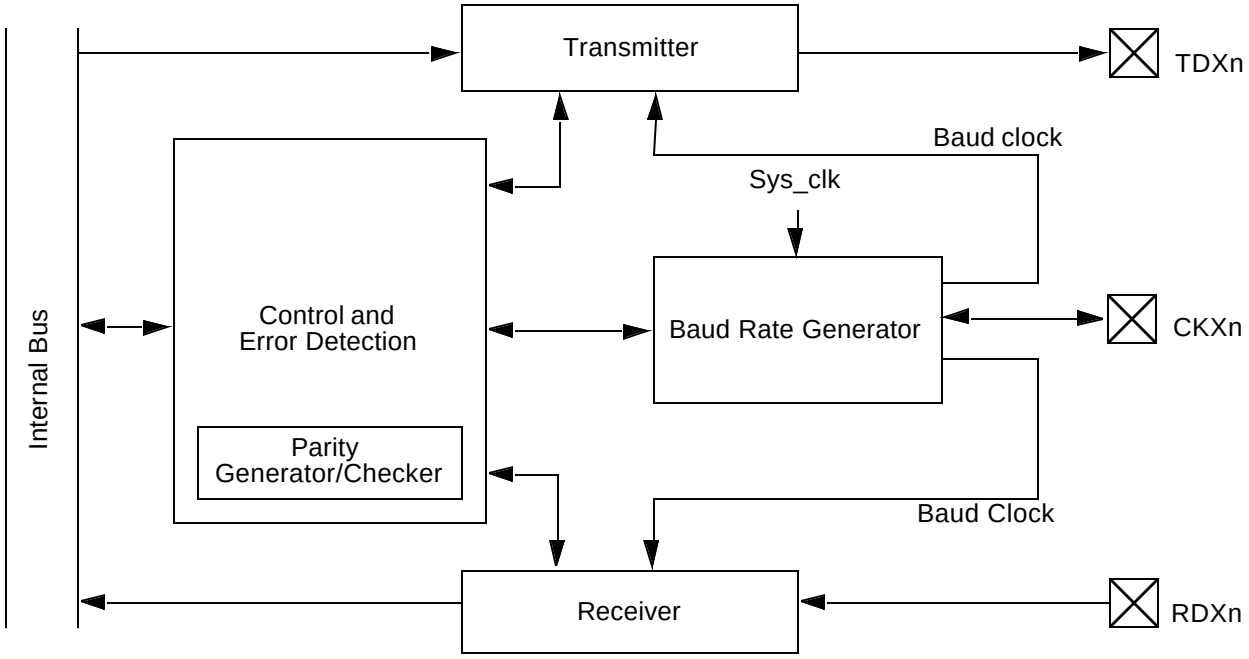


Figure 26. USART Block Diagram

flag is automatically reset when software reads the character from the UnRBUF register. The RSFT register is not user accessible.

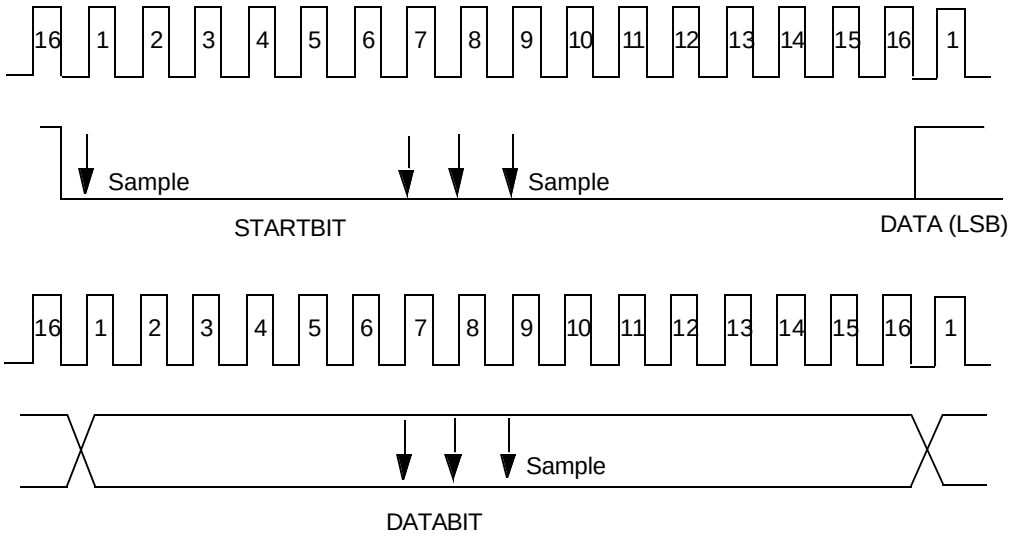


Figure 27. USART Asynchronous Communication

16.2.2 Synchronous Mode

The synchronous mode of the USART enables the device to communicate with other devices using three communication signals: transmit, receive, and clock. In this mode, data bits are transferred synchronously with the USART clock signal. Data bits are transmitted on the rising edges and received on the falling edges of the clock signal, as shown in Figure28. Data bytes are transmitted and received least significant bit (LSB) first.

In the synchronous mode, the transmit shift register (TSFT) and the transmit buffer (UnTBUF) double-buffer the data for transmission. To transmit a character, a data byte is loaded in the UnTBUF register. The data is then transferred to the TSFT register. The TSFT register shifts out one bit of the current character, LSB first, on each rising edge of the clock. While the TSFT is shifting out the current character on the TDXn pin, the UnTBUF register may be loaded by the software with the next byte to be transmitted. When the TSFT finishes transmission of the last stop bit within the current frame, the contents of UnTBUF are transferred to the TSFT

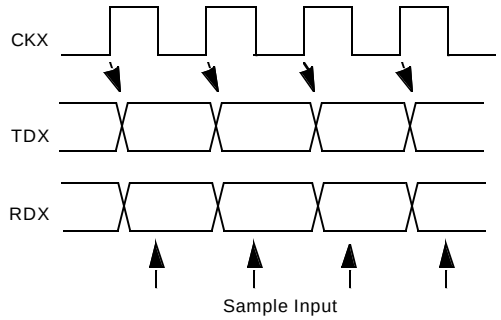


Figure 28. USART Synchronous Communication

register and the Transmit Buffer Empty flag (UnTBE) is set. The UnTBE flag is automatically reset by the USART when the software loads a new character into the UnTBUF register. During transmission, the UnXMIP bit is set high by the USART. This bit is reset only after the USART has sent the last frame bit of the current character and the UnTBUF register is empty.

The receive shift register (RSFT) and the receive buffer (UnRBUF) double-buffer the data being received. Serial data received on the RDXn pin is shifted into the RSFT register at the first falling edge of the clock. Each subsequent falling edge of the clock causes an additional bit to be shifted into the RSFT register. The USART assumes a complete character has been received after the correct number of rising edges on CKXn (based on the selected frame format) have been detected. Upon receiving a complete character, the contents of the RSFT register are copied into the UnRBUF register and the Receive Buffer Full flag (UnRBF) is set. The UnRBF flag is automatically reset when the software reads the character from the UnRBUF register.

The transmitter and receiver may be clocked from either an external source provided to the CKXn pin or by the internal baud rate generator. In the latter case, the clock signal is placed on the CKXn pin as an output.

16.2.3 Attention Mode

The Attention mode is available for networking this device with other processors. This mode requires the 9-bit data format with no parity. The number of start bits and number of stop bits are programmable. In this mode, two types of 9-bit characters are sent on the network: address characters consisting of 8 address bits and a 1 in the ninth bit position and data characters consisting of 8 data bits and a 0 in the ninth bit position.

While in Attention mode, the USART receiver monitors the communication flow but ignores all characters until an address character is received. Upon the receipt of an address character, the contents of the receive shift register are copied to the receive buffer. The UnRBF flag is set and an interrupt (if enabled) is generated. The UnATN bit is automatically reset to zero, and the USART begins receiving all subsequent characters. The software must examine the contents of the UnRBUF register and respond by accepting the subsequent characters (by leaving the UnATN bit reset) or waiting for the next address character (by setting the UnATN bit again).

The operation of the USART transmitter is not affected by the selection of this mode. The value of the ninth bit to be transmitted is programmed by setting or clearing a bit called UnXB9 in the USART Frame Select Register. The value of the ninth bit received is read from another register bit called UnRB9 in the USART Status Register.

16.2.4 Diagnostic Mode

The Diagnostic mode is available for testing of the USART. In this mode, the TDXn and RDXn pins are internally connected together, and data that is shifted out of the transmit shift register is immediately transferred to the receive shift register. This mode supports only the 9-bit data format with no parity. The number of start and stop bits is programmable.

16.2.5 Frame Format Selection

The format shown in Figure29 consists of a start bit, seven data bits (excluding parity), and one or two stop bits. If parity bit generation is enabled by setting the UnPEN bit, a parity bit is generated and transmitted following the seven data bits.

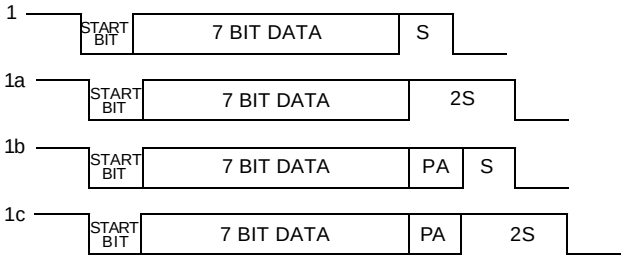


Figure 29. Seven Data Bit Frame Options

The format shown in Figure30 consists of one start bit, eight data bits (excluding parity), and one or two stop bits. If parity bit generation is enabled by setting the UnPEN bit, a parity bit is generated and transmitted following the eight data bits.

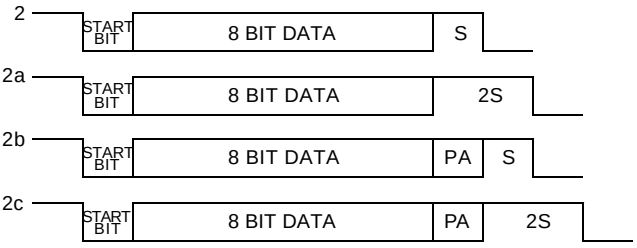


Figure 30. Eight Data Bit Frame Options

The format shown in Figure31 consists of one start bit, nine data bits, and one or two stop bits. This format also supports the USART attention feature. When operating in this format, all eight bits of UnTBUF and UnRBUF are used for data. The ninth data bit is transmitted and received using two bits in the control registers, called UnXB9 and UnRB9. Parity is not generated or verified in this mode.

16.2.6 Baud Rate Generator

The Baud Rate Generator creates the basic baud clock from the system clock. The system clock is passed through a two-

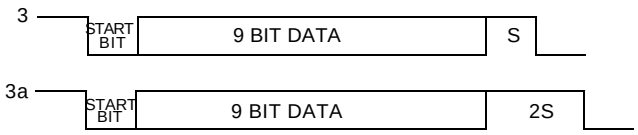


Figure 31. Nine Data Bit Frame Options

stage divider chain consisting of a 5-bit baud rate prescaler (UnPSC) and an 11-bit baud rate divisor (UnDIV).

The relationship between the 5-bit prescaler select (UnPSC) setting and the prescaler factors is shown in Table18.

Table 18 Prescaler Factors

Prescaler Select	Prescaler Factor	Prescaler Select	Prescaler Factor
00000	1	10000	8.5
00001	1	10001	9
00010	1.5	10010	9.5
00011	2	10011	10
00100	2.5	10100	10.5
00101	3	10101	11
00110	3.5	10110	11.5
00111	4	10111	12
01000	4.5	11000	12.5
01001	5	11001	13
01010	5.5	11010	13.5
01011	6	11011	14
01100	6.5	11100	14.5
01101	7	11101	15
01110	7.5	11110	15.5

Table 18 Prescaler Factors

Prescaler Select	Prescaler Factor	Prescaler Select	Prescaler Factor
01111	8	11111	16

A prescaler factor of zero corresponds to “no clock.” The “no clock” condition is the USART power down mode, in which the USART clock is turned off to reduce power consumption. The application program should select the “no clock” condition before entering a new baud rate. Otherwise, it could cause incorrect data to be received or transmitted. The UnPSR register must contain a value other than zero when an external clock is used at CKXn.

In asynchronous mode, the baud rate is calculated by:

$$BR = \frac{SYS_CLK}{(16 \times N \times P)}$$

where BR is the baud rate, SYS_CLK is the system clock, N is the value of the baud rate divisor + 1, and P is the prescaler divide factor selected by the value in the UnPSR register.

The divide by 16 is performed because in the asynchronous mode, the input frequency to the USART is 16 times the baud rate. In synchronous mode, the input clock to the USART is equal to the baud rate.

16.2.7 Interrupts

The USART is capable of generating interrupts on:

- Receive Buffer Full
- Receive Error
- Transmit Buffer Empty

Figure32 shows a diagram of the interrupt sources and associated enable bits.

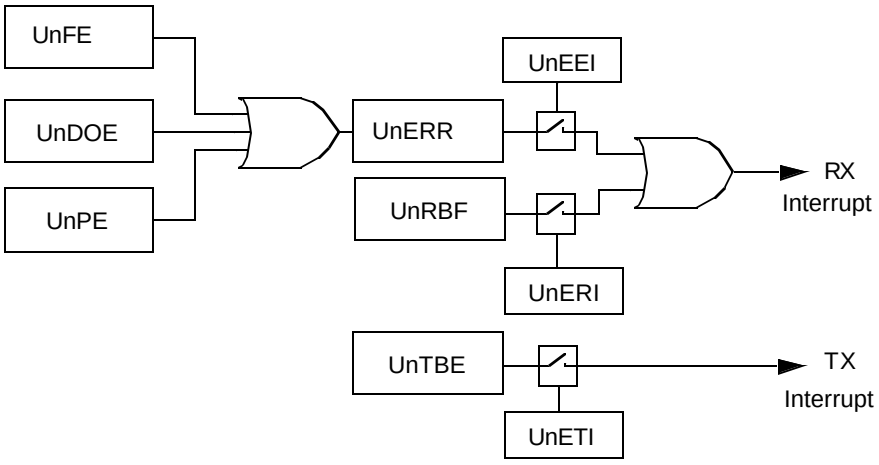


Figure 32. USART Interrupts

The interrupts can be individually enabled or disabled using the Enable Transmit Interrupt (UnETI), Enable Receive Interrupt (UnERI) and Enable Receive Error Interrupt (UnEER) bits in the UnICTRL register.

A transmit interrupt is generated when both the UnTBE and UnETI bits are set. To remove this interrupt, software must either disable the interrupt by clearing the UnETI bit or write to the UnTBUF register (thus clearing the UnTBE bit).

A receive interrupt is generated on two conditions:

1. Both the UnRBF and UnERI bits are set. To remove this interrupt, software must either disable the interrupt by clearing the UnERI bit or read from the UnRBUF register (thus clearing the UnRBF bit).
2. Both the UnERR and the UnEEI bits are set. To remove this interrupt the software must either disable it by clearing the UnEEI bit or read the UnSTAT register (thus clearing the UnERR bit).

16.2.8 Break Generation and Detection

A line break is generated when the BRK bit is set in the UnMDSL register. The TDXn line remains low until the program resets the BRK bit.

A line break is detected if RDXn remains low for 10 bit times or longer after a missing stop bit is detected.

16.2.9 Parity Generation and Detection

Parity is only generated or checked with the 7-bit and 8-bit data formats. It is not generated or checked in the diagnostic loopback mode, the attention mode, or in the normal mode with the 9-bit data format. Parity generation and checking are enabled and disabled via the PEN bit in the UnFRS register. The UnPSEL bits in the UnFRS register are used to select odd, even, mark, or space parity.

16.3 USART REGISTERS

The software interacts with the USART by accessing the USART registers. There are eight such registers:

- USART Receive Data Buffer (UnRBUF)
- USART Transmit Data Buffer (UnTBUF)
- USART Baud Rate Prescaler Register (UnPSR)
- USART Baud Rate Divisor Register (UnBAUD)
- USART Frame Select Register (UnFRS)
- USART Mode Select Register (UnMDSL)
- USART Status Register (UnSTAT)
- USART Interrupt Control Register (UnICTRL)

16.3.1 USART Receive Data Buffer (UnRBUF)

The USART Receive Data Buffer is a byte-wide, read/write register used to receive each data byte.

16.3.2 USART Transmit Data Buffer (UnTBUF)

The USART Transmit Data Buffer is a byte-wide, read/write register used to transmit each data byte.

16.3.3 USART Baud Rate Prescaler (UnPSR)

The USART Baud Rate Prescaler Register is a byte-wide, read/write register that contains the 5-bit clock prescaler and the upper three bits of the baud rate divisor. This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
UnPSC					UnDIV10	UnDIV9	UnDIV8

UnPSC — Prescaler. This 5-bit field specifies the prescaler value used for dividing the system clock in the first stage of the two-stage divider chain. For the prescaler factors corresponding to each 5-bit value, see Table18.

UnDIV[10:8] — Baud Rate Divisor (bits 10-8). This field contains the three highest-order bits (bits 10, 9, and 8) of the USART baud rate divisor used in the second stage of the two-stage divider chain. The remaining bits of the baud rate divisor are contained in the UnBAUD register.

16.3.4 USART Baud Rate Divisor (UnBAUD)

The USART Baud Rate Divisor Register is a byte-wide, read/write register that contains the lower eight bits of the baud rate divisor. This register contents are unknown upon power-up and are left unchanged by a reset operation. The register format is shown below.

7	6	5	4	3	2	1	0
UnDIV7	UnDIV6	UnDIV5	UnDIV4	UnDIV3	UnDIV2	UnDIV1	UnDIV0

UnDIV[7:0] — Baud Rate Divisor (bits 7-0). This field contains the eight lowest-order bits of the USART baud rate divisor used in the second stage of the two-stage divider chain. The three highest-order bits are contained in the UnPSR register. The divisor value used is the 11-bit UnDIV value plus 1.

16.3.5 USART Frame Select Register (UnFRS)

The USART Frame Select Register is a byte-wide, read/write register that controls the frame format, including the number of data bits, number of stop bits, and parity type. This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved	UnPEN	UnPSEL		UnXB9	UnSTP	UnCHAR	

UnCHAR — Character Frame Format. This 2-bit field selects the number of data bits per frame, not including the parity bit, as follows:

00 = eight data bits per frame
01 = seven data bits per frame
10 = nine data bits per frame
11 = loopback mode; nine data bits per frame

UnSTP — Number of Stop Bits. This bit sets the number of stop bits transmitted in each frame. If this bit is 0, one stop bit is transmitted. If this bit is 1, two stop bits are transmitted.

UnXB9 — Transmit 9th Data Bit. This bit is the value of the ninth data bit, either 0 or 1, transmitted when the USART is configured to transmit nine data bits per frame. It has no effect when the USART is configured to transmit seven or eight data bits per frame.

UnPSEL — Parity Select. This 2-bit field selects parity type as follows:

00 = odd parity
01 = even parity
10 = mark (0)
11 = space (1)

When the USART is configured to transmit nine

data bits per frame, the parity bit is omitted and the UnPSEL field is ignored.

UnPEN — Parity Enable. This bit enables (1) or disables (0) parity bit generation and parity checking. When the USART is configured to transmit nine data bits per frame, there is no parity bit and the UnPEN bit is ignored.

16.3.6 USART Mode Select Register (UnMDSL)

The USART Mode Select Register is a byte-wide, read/write register that selects the clock source, synchronization mode, attention mode, and line break generation. This register is cleared upon reset. When the software writes to this register, the reserved bits must be cleared to 0 for proper operation. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved				UnCKS	UnBRK	UnATN	UnMOD

UnMOD — Mode of Operation. Set to 0 for asynchronous operation or 1 for synchronous operation.

UnATN — Attention Mode. When set to 1, this bit selects the attention mode of operation for the USART. When cleared to 0, the attention mode is disabled. The hardware clears this bit after an address frame is received. An address frame is a 9-bit character with a 1 in the ninth bit position.

UnBRK — Force Transmission Break. Setting this bit to 1 causes the TDXn pin to go low. TDXn remains low until the UnBRK bit is cleared to 0 by the software.

UnCKS — Synchronous Clock Source. This bit controls the clock source when the USART operates in the synchronous mode (UnMOD=1). If the UnCKS bit is set to 1, the USART operates from an external clock provided on the CKXn pin. If the UnCKS bit is cleared to 0, the USART operates from the baud rate clock produced by the USART on the CKXn pin. This bit is ignored when the USART operates in the asynchronous mode.

16.3.7 USART Status Register (UnSTAT)

The USART Status Register is a byte-wide, read-only register that contains the receive and transmit status bits. This register is cleared upon reset. Any attempt by the software to write to this register is ignored. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved	UnXMIP	UnRB9	UnBKD	UnERR	UnDOE	UnFE	UnPE

UnPE — Parity Error. This bit is set to 1 when a parity error is detected within a received character. This bit is automatically cleared to 0 by the hardware when the UnSTAT register is read.

UnFE — Framing Error. This bit is set to 1 when the USART fails to receive a valid stop bit at the end of a frame. This bit is automatically cleared to 0 by the hardware when the UnSTAT register is read.

UnDOE — Data Overrun Error. This bit is set to 1 when a new character is received and transferred to the UnBUF register before the software has read the previous character from UnBUF. This bit is automatically cleared to 0 by the hardware when the UnSTAT register is read.

UnERR — Error Status Flag. This bit is set when a parity, framing, or overrun error occurs (any time that the UnPE, UnFE, or UnDOE bit is set). It is automatically cleared to 0 by the hardware when the UnPE, UnFE, and UnDOE bits are all 0.

UnBKD — Break Detect. This bit is set to 1 when a line break condition occurs. This condition is detected if RDXn remains low for at least ten bit times after a missing stop bit has been detected at the end of a frame.

The hardware automatically clears the UnBKD bit upon read of the UnSTAT register, but only if the break condition on RXDn no longer exists. If reading the UnSTAT register does not clear the UnBKD bit because the break is still actively driven on the line, the hardware clears the bit as soon as the break condition no longer exists (when RXDn returns to a high level).

UnRB9 — Received 9th Data Bit. With the USART configured to operate in the 9-bit data format, this is equal to the ninth data bit of the last frame received.

UnXMIP — Transmit In Progress. The hardware sets this bit to 1 when the USART is transmitting data and clears it to 0 at the end of the last frame bit.

16.3.8 USART Interrupt Control Register (UnICTRL)

The USART Interrupt Control Register is a byte-wide register that contains the receive and transmit interrupt status flags (read-only bits) and the interrupt enable bits (read/write bits). The register is set to 01 hex upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
UnEEI	UnERI	UnETI	Reserved			UnRBF	UnTBE

UnTBE — Transmit Buffer Empty. This read-only bit is set to 1 by the hardware when the USART transfers data from the UnTBUF register to the transmit shift register for transmission. It is automatically cleared to 0 by the hardware on the next write to the UnTBUF register.

UnRBF — Receive Buffer Full. This read-only bit is set by the hardware when the USART has received a complete data frame and has transferred the data from the receive shift register to the UnRBUF register. It is automatically cleared to 0 by the hardware when the UnRBUF register is read.

UnETI — Enable Transmitter Interrupt. This read/write bit, when set to 1, enables generation of an interrupt when the hardware sets the UnTBE bit.

UnERI — Enable Receiver Interrupt. This read/write bit, when set to 1, enables generation of an interrupt when the hardware sets the UnRBF bit.

UnEEI — Enable Receive Error Interrupt. This read/write bit, when set to 1, enables generation of an interrupt when the hardware sets the UnERR bit in the UnSTAT register.

16.4 BAUD RATE CALCULATIONS

The USART baud rate is determined by the system clock frequency and the values programmed into the UnPSR and UnBAUD registers. Unless the system clock frequency is an exact multiple of the desired baud rate, there will be a small amount of error in the resulting baud rate clock.

The method of baud rate calculation depends on whether the USART is configured to operate in the asynchronous or synchronous mode.

16.4.1 Baud Rate in Asynchronous Mode

The equation for calculating the baud rate in asynchronous mode is:

$$BR = \frac{SYS_CLK}{(16 \times N \times P)}$$

where BR is the baud rate, SYS_CLK is the system clock, N is the value of the baud rate divisor + 1, and P is the prescaler divide factor selected by the value in the UnPSR register.

Assuming a system clock of 5 MHz and a desired baud rate of 9600, the NxP term according to the equation above is:

$$N \times P = \frac{(5 \times 10^6)}{(16 \times 9600)} = 32.552$$

The NxP term is then divided by each Prescaler Factor from Table18 to obtain a value closest to an integer. The factor for this example is 6.5.

$$N = \frac{32.552}{6.5} = 5.008 \quad (\mathrm{N} = 5)$$

The baud rate register is programmed with a baud rate divisor of 4 (N = baud rate divisor +1). This produces a baud clock of:

$$BR = \frac{(5 \times 10^6)}{(16 \times 5 \times 6.5)} = 9615.385$$

$$\%error = \frac{(9615.385 - 9600)}{9600} = 0.16$$

Note that the percent error is much lower than would be possible without the non-integer prescaler factor. Refer to the table below for more examples.

System Clock	Desired Baud Rate	N	P	Actual Baud Rate	Percent Error
4 MHz	9600	2	13	9615.385	0.16
5 MHz	9600	5	6.5	9615.385	0.16
10 MHz	19200	5	6.5	19230.769	0.16
20 MHz	19200	5	13	19230.769	0.16

16.4.2 Baud Rate in Synchronous Mode

The equation for calculating the baud rate in synchronous mode is:

$$BR = \frac{SYS_CLK}{(2 \times N \times P)}$$

where BR is the baud rate, SYS_CLK is the system clock, N is the value of the baud rate divisor + 1, and P is the prescaler divide factor selected by the value in the UnPSR register.

Use the same procedure to determine the values of N and P as in the asynchronous mode. In this case, however, only integer prescaler values are allowed.

17.0 Analog Comparators

The Dual Analog Comparator (ACMP2) module contains two independent analog comparators with all necessary control logic. Each comparator unit compares the analog input voltages applied to two input pins and determines which voltage is higher. The comparison results can be placed on two output pins and/or read by the software from a register.

Figure33 is a block diagram of the Dual Analog Comparator module.

The two comparators are designated Comparator 1 (CMP1) and Comparator 2 (CMP2). Each comparator has a positive and a negative input, called CMP1P and CMP1N for Comparator 1 and CMP2P and CMP2N for Comparator 2. An optional output, CMP1O for Comparator 1 or CMP2O for Comparator 2, allows the external hardware to read the comparison results. If the positive input is greater than the negative input, the result is a logic 1. Otherwise, the result is a logic 0. These same results are available to the software by reading the CMPCTRL register. CMP1OP and CMP2OP are the direct outputs of the analog comparator. These signals are connected to the channels of the Multi-Wake-Up Module.

17.1 ANALOG COMPARATOR CONTROL/ STATUS REGISTER (CMPCTRL)

The CMPCTRL register is a byte-wide, read/write register that controls the comparator module and contains the comparison results. The control bits are read/write bits and the result bits are read-only bits. This register is cleared upon reset. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		CMP2OE	CMP1OE	CMP2EN	CMP1EN	CMP2RD	CMP1RD

CMP1RD Comparator 1 Read. This read-only bit contains the output of Comparator 1 when the comparator is enabled (CMP1EN=1). CMP1RD is set to 1 when the voltage on CMP1P is greater than the voltage on CMP1N. This bit is always 0 when Comparator 1 is disabled.

CMP2RD Comparator 2 Read. This read-only bit contains the output of Comparator 2 when the comparator is enabled (CMP2EN=1). CMP2RD is set to 1 when the voltage on CMP2P is greater than the voltage on CMP2N. This bit is always 0 when Comparator 2 is disabled.

CMP1EN Comparator 1 Enable. This read/write bit enables (1) or disables (0) Comparator 1.

CMP2EN Comparator 2 Enable. This read/write bit enables (1) or disables (0) Comparator 2.

CMP1OE Comparator 1 Output Enable. This read/write bit, when set to 1, enables the use of the CMP1O pin as the output of Comparator 1 when Comparator 1 is enabled (CMP1EN=1). If Comparator 1 is disabled (CMP1EN=0), setting the CMP1OE bit results in a logic 0 on the CMP1O output pin.

CMP2OE Comparator 2 Output Enable. This read/write bit, when set to 1, enables the use of the CMP2O pin as the output of Comparator 2 when Comparator 2 is enabled (CMP2EN=1). If Comparator 2 is disabled (CMP2EN=0), setting the CMP2OE bit results in a logic 0 on the CMP2O output pin.

17.2 ANALOG COMPARATOR USAGE

The comparator I/O pins are alternate functions of the Port L pins. In order for a comparator to operate, its two input pins must be configured to operate as inputs in the alternate function mode.

Using a comparator's output pin is optional. If it is to be used, it must be configured to operate as an output in the alternate function mode. The comparison result bits in the CMPCTRL register are available to the CPU whether or not the output pin is enabled.

The comparators uses DC current whenever they are enabled. Therefore, in order to reduce power consumption, it is recommended that the comparators be disabled when they are not needed, especially before entering any of the Power Save modes.

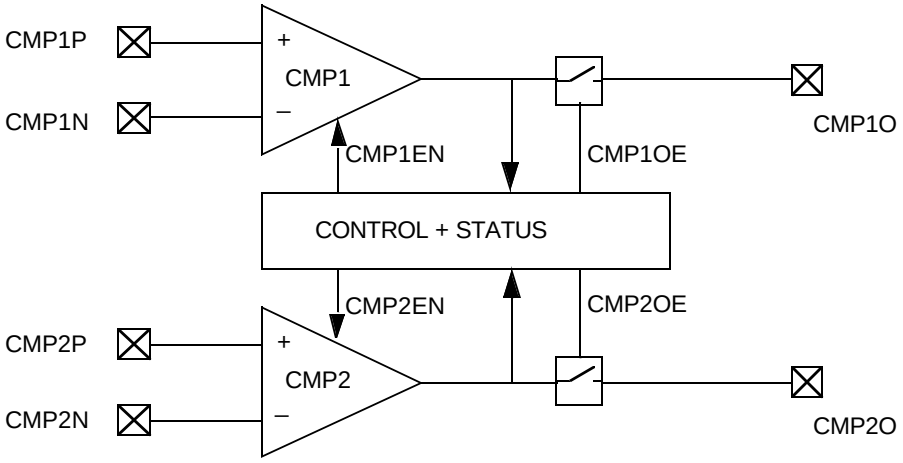


Figure 33. Dual Analog Comparator Block Diagram

18.0 A/D Converter

The A/D Converter (ADC) module is an 8-channel, multiplexed-input, analog-to-digital converter. The A/D Converter receives an analog voltage on an input pin and converts that voltage into an 8-bit digital value using successive approximation. The CPU can then read the result from a memory-mapped register. The module supports four automated operating modes, providing single-channel or 4-channel scanned operation in single-conversion or continuous mode.

Figure34 is a block diagram of the A/D Converter module.

The analog input signal is selected from eight analog inputs using an 8-channel analog multiplexer. The input pins are alternate functions of Port I.

A sample-and-hold circuit samples the analog voltage prior to conversion and holds it stable and throughout the conversion process. A programmable initial delay period allows the sampled voltage to stabilize before the conversion process begins.

A capacitor should be connected between the V_{REF} and the AV_{CC} pin in order to minimize noise. The recommended value for this capacitor is about 0.47 µF

The input voltage range is from 0V to V_{REF} (the A/D reference voltage). The 80-pin device have a separate pin, V_{REF}, for the reference voltage. The 44-pin devices use the AV_{CC} (analog V_{CC}) power supply pin as the reference voltage.

The internal analog-to-digital converter block is based on a successive approximation algorithm, which compares the sampled voltage against an internally generated sequence of analog voltages. The result is a linear conversion of the analog voltage to an unsigned 8-bit value ranging from 00 hex for 0.0 volts to FF hex for V_{REF}.

The clock used by the converter block is generated by a clock divider that scales down the system clock by a programmable factor. The conversion algorithm requires ten A/D Converter clock cycles, or 10 microseconds at the maximum allowed A/D Converter clock rate of 1 MHz.

Conversion can start after the power supply is stable and ADCEN set for 100 µs.

The conversion results are stored in a 4-level data buffer. Depending on the operating mode, the buffer can hold the results of four successive conversions from a single channel or four conversions from adjacent channels scanned in sequence.

18.1 OPERATING MODES

The A/D Converter can be configured to operate in any one of four modes:

- Single channel, single conversion
- Single channel, continuous conversion
- 4-channel scan, single conversion
- 4-channel scan, continuous conversion

The configuration is set by the SCAN and CONT fields in the ADC Control 2 Register (ADCCNT2), as indicated in Table19. The A/D converter must be disabled when switching to a different mode.

Table 19 ADC Operation Modes

SCAN	CONT	Mode
00	0	Single Channel, Single Conversion
00	1	Single Channel, Continuous Conversion
01	0	4 Channels Scan, Single Conversion
01	1	4 Channel Scan, Continuous Conversion

18.1.1 Single Channel, Single Conversion Mode

In the single channel, single conversion mode, the A/D Converter performs a single conversion using a specified channel.

The software starts a conversion by setting the START bit in the ADCCNT2 register. Upon completion of the conversion, the A/D Converter places the result in register ADDATA0, clears the START bit, and sets the EOC (end of conversion) bit in the ADCST register. If the A/D Converter interrupt is enabled, an interrupt to the CPU is generated at this time.

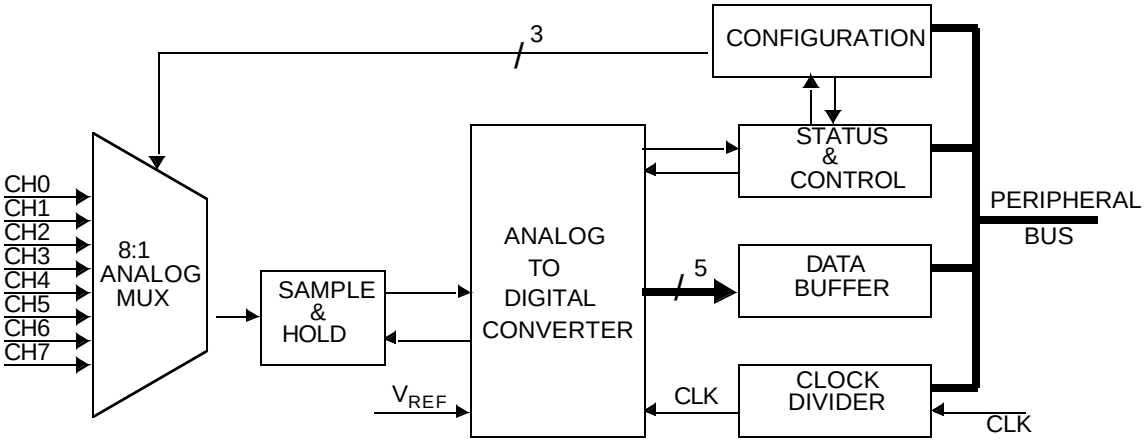


Figure 34. A/D Converter Block Diagram

18.1.2 Single Channel, Continuous Conversion Mode

In the single channel, continuous conversion mode, the A/D Converter performs conversions repeatedly using the same specified channel.

The software starts a conversion sequence by setting the START bit. The A/D Converter performs four A/D conversions in sequence using the same channel, pausing only for the programmable sampling delay time used in all conversion operations. It loads the four results into the A/D data registers in sequence, starting with ADDATA0 and ending with ADDATA3. After it loads all four registers, it sets the EOC (end of conversion) bit. If the A/D Converter interrupt is enabled, an interrupt to the CPU is generated at this time.

The START bit remains set until cleared by the software. If the software does not clear the START bit, the A/D Converter continues performing conversions using the same input channel, storing the results in ADDATA0 following ADDATA3. To prevent an overrun error, the software must read the results from the data registers before the A/D Converter writes the next result into ADDATA0 following ADDATA3.

When the software clears the START bit, the A/D Converter first completes the conversion currently in progress, then stops and sets the EOC bit. A 2-bit buffer pointer in the ADCST register points to the register containing the final result.

18.1.3 4-Channel Scan, Single Conversion Mode

In the 4-channel scan, single conversion mode, the A/D Converter performs four conversions using four adjacent input channels.

The software starts the conversion sequence by setting the START bit. The A/D Converter performs four A/D conversions in sequence using four adjacent channels, starting with the specified channel and pausing only for the programmable sampling delay time. It loads the four results into the A/D data registers in sequence, starting with ADDATA0 and ending with ADDATA3. After it loads all four registers, it clears the START bit and sets the EOC (end of conversion) bit. If the A/D Converter interrupt is enabled, an interrupt to the CPU is generated at this time.

18.1.4 Channel Scan, Continuous Conversion Mode

In the 4-channel scan, continuous conversion mode, the A/D Converter performs conversions repeatedly using four adjacent input channels.

The software starts conversion operations by setting the START bit. The A/D Converter performs four A/D conversions in sequence using four adjacent channels, starting with the specified channel and pausing only for the programmable sampling delay time. It loads the four results into the A/D data registers in sequence, starting with ADDATA0 and ending with ADDATA3. After it loads all four registers, it sets the EOC (end of conversion) bit. If the A/D Converter interrupt is enabled, an interrupt to the CPU is generated at this time.

The START bit remains set until cleared by the software. If the software does not clear the START bit, the A/D Converter continues performing conversions, repeating the same sequence using the same four input channels and the same sequence of data registers. To prevent an overrun error, the software must read the results from the data registers before the A/D Converter writes the next result into ADDATA0.

When the software clears the START bit, the A/D Converter first completes the 4-channel conversion sequence currently in progress, then stops and sets the EOC bit.

18.2 A/D CONVERTER REGISTERS

The software controls the A/D Converter and reads the A/D results by accessing the ADC registers. There are eight such registers:

- ADC Status Register (ADCST)
- ADC Control 1 Register (ADCCNT1)
- ADC Control 2 Register (ADCCNT2)
- ADC Control 3 Register (ADCCNT3)
- ADC Data Registers (ADDATA0 through ADDATA3)

18.2.1 ADC Status Register (ADCST)

The ADCST register is a byte-wide register that indicates the current status of the A/D Converter. One bit in this register, the OVF flag bit, is cleared by writing a 1 to its bit position. The other bits are read-only bits, so the values written to them are ignored. Upon reset, the register is set to 30 hex. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		BUFPTR		Reserved	OVF	BUSY	EOC

EOC — End of Conversion. This read-only bit reports the status of the most recent A/D Converter operation. When cleared to 0, it indicates that the conversion is not complete. When set to 1, it indicates that the conversion is complete. The hardware sets this bit when it places the conversion results in the buffer and clears it when any of the data registers are read.

BUSY — ADC Busy. This read-only bit is set to 1 when the A/D Converter is busy converting data and is cleared to 0 when the A/D Converter is idle or disabled.

OVF — Overflow. The hardware sets this bit to 1 when the A/D Converter finishes a conversion and attempts to store the results in one of the data registers (ADDATA0-ADDATA3) while the register is full. When this happens, the A/D Converter overwrites the data in the data register, sets the OVF flag, and continues operating.
The OVF flag remains set until cleared by the software. The software clears the flag by writing a 1 to it. Writing a 0 to this bit has no effect.

BUFPTR — Buffer Pointer. This 2-bit, read-only field identifies the data register that was most recently written with new data:
00 = ADDATA0
01 = ADDATA1
10 = ADDATA2
11 = ADDATA3
This register is initialized to 11 when a new conversion is started (when ADCCNT2.START is changed from 0 to 1) and is automatically incremented every time a result is written to buffers ADDATA0-ADDATA3. The result is a four-entry

cyclic FIFO buffer, with BUFPTR pointing to the last entry written by the A/D Converter.

18.2.2 ADC Control 1 Register (ADCCNT1)

The ADCCNT1 register is a byte-wide, read/write register used to enable the A/D Converter and its interrupts, and also to control the reference voltage source. When writing to this register, all reserved bits must be written with 0 for the A/D Converter to function properly. Changing any bits other than ADCEN (bit 0) is not allowed while the A/D Converter is active (ADCST.BUSY or ADCCNT2.START set). Upon reset, all non-reserved bits are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved					INTE	Reserved	ADCEN

ADCEN A/D Converter Enable. Setting this bit enables the A/D Converter and allows a conversion to be started by setting the start bit (ADCCNT2.START). Clearing the ADCEN bit disables the A/D Converter, terminates any conversion in progress, and clears the ADC status flags (ADCST.EOC, ADCST.BUSY, ADCST.OVF, and ADCCNT2.START).

INTE Interrupt Enable. This bit enables (1) or disables (0) A/D Converter interrupts. If enabled, and interrupt occurs at the end of a conversion sequence or when the ADC data buffer is full, depending on the operating mode.

All reserved bits must be written with 0 for ADC to operate properly.

18.2.3 ADC Control 2 Register (ADCCNT2)

The ADCCNT2 register is a byte-wide, read/write register used to specify the A/D Converter operating mode and to start conversion operations. All register fields other than the START bit should be changed only while the A/D Converter is inactive (START=0). Data written to the SCAN and CONT fields is ignored if the START bit is already set. Upon reset, the non-reserved bits of this register are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
START	SCAN		CONT	CHANNEL			

CHANNEL Channel Select. This 4-bit field selects one of the eight analog input channels as follows:
0000 = ACH0
0001 = ACH1
0010 = ACH2
0011 = ACH3
0100 = ACH4
0101 = ACH5
0110 = ACH6
0111 = ACH7
1XXX = reserved

CONT Continuous Conversion. When cleared to 0, the A/D Converter stops operating upon completion of the programmed conversion cycle (a single conversion or a sequence of four conversions on four channels). When set to 1, the A/D Converter operates continuously by repeating the programmed conversion cycle.

SCAN Scan Mode. This 2-bit field selects the single-conversion mode or 4-channel scan mode as follows:
00 = single-conversion mode
01 = 4-channel scan mode
1X = reserved

START Start Conversion. The software sets this bit to 1 to start a conversion or a 4-channel conversion cycle. In the “continuous” mode, this bit remains set until cleared by the software. In the “single” (non-continuous) mode, the hardware clears this bit upon completion of the programmed conversion cycle. The software should not attempt to set this bit while the A/D Converter is busy (ADCST.BUSY=1).

18.2.4 ADC Control 3 Register (ADCCNT3)

The ADCCNT3 register is a byte-wide, read/write register used to specify the analog sampling time delay and the divide-by factor for generating the ADC clock. This register should be written only when the A/D Converter is disabled (ADCCNT1.ADCEN=0). Upon reset, the non-reserved bits of the ADCCNT3 register are cleared to 0. The register format is shown below.

7	6	5	4	3	2	1	0
Reserved		DELAY			CDIV		

CDIV Clock Divide. This 3-bit field sets the divide-by factor for generating the A/D Converter clock from the system clock. The frequency of the A/D Converter clock is equal to the system clock divided by the programmed factor. The resulting A/D Converter clock frequency must be less than or equal to 1 MHz. The divide-by factor is defined as follows:
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 = reserved
111 = reserved

DELAY Sampling Time Delay. This 3-bit field defines the number of A/D Converter clock cycles of delay from the time that the input channel is selected until the analog voltage is sampled. The programmed delay should be sufficient, dependent on the source impedance, to allow the sampled signal to reach its final level before the conversion begins. The delay is defined as follows:

000 = 1 A/D Converter clock cycle
001 = 2 A/D Converter clock cycles
010 = 4 A/D Converter clock cycles
011 = 8 A/D Converter clock cycles
100 = 16 A/D Converter clock cycles
101 = 32 A/D Converter clock cycles

110 = 64 A/D Converter clock cycles
111 = reserved

18.2.5 ADC Data Registers (ADDATA0-ADDATA3)

The four ADC Data Registers (ADDATA0 through ADDATA3) are byte-wide, read/write registers that hold the conversion results, which are stored sequentially starting with ADDATA0 and ending with ADDATA3. The results held in these registers are valid only after the ADCST.EOC flag is set. Upon reset, the contents of these registers are undefined.

The value read from a data register is a linear mapping of the analog input voltage to an 8-bit value. The value 00 hex represents 0.0 volts and the value FF hex represents the reference voltage, V_{REF}.

18.3 A/D CONVERTER PROGRAMMING

The software should set the A/D Converter configuration before it enables the A/D Converter module. The configuration consists of the following settings:

- ADC clock rate: ADCCNT3.CDIV
- Sampling delay: ADCCNT3.DELAY
- Interrupt enable (if required): ADCCNT1.INTE

The ADC clock is created by scaling down the system clock. The fastest allowable clock for the A/D Converter is 1 MHz. Therefore, for the fastest possible operation of the A/D Converter, use the smallest available divide-by factor that results in a clock frequency of 1 MHz or lower. The available divide-by factors are 1, 2, 4, 8, 16, and 32.

For example, if the system clock is 10 MHz, use a divide-by factor of 16. In that case, the A/D Converter clock frequency is 625 kHz, the clock period is 1.6 microseconds, and the A/D conversion time is 16 microseconds (ten clock A/D Converter clock cycles).

The programmable sampling time delay should be made small for faster operation, but large enough to allow the input voltage to settle. The internal resistance and capacitance of the A/D Converter, together with the source resistance of the device that drives the A/D input determine the charge-up time required for the voltage to settle. Figure35 shows a schematic of the charge-up circuit. For the values of R_{AIN} and C_{AIN}, see Section21.0.

Interrupts or polling can be used to read the A/D Converter results. For interrupts, the A/D Converter interrupt must be enabled by setting the ADCCNT1.INTE bit. The interrupt is cleared automatically when any one of the data registers (ADDATA0-ADDATA3) is read. For polling, the software reads the ADCST.EOC bit to determine whether the conversion sequence is completed.

Once the A/D Converter configuration has been set up, the software can use the following procedure to perform an A/D conversion sequence:

1. Enable the A/D Converter by setting the ADCCNT1.ADCEN bit and wait 100 µs before performing any conversion.
2. Select the operating mode and channel by writing to the SCAN, CONT, and CHANNEL fields of the ADCCNT2 register. At the same time, start the conversion by setting the START bit in the same register.
3. Wait until the conversion is finished, either by polling or using the A/D Converter interrupt.
4. Read the conversion results from the data registers, ADDATA0 through ADDATA3 (or just ADDATA0 in the single-channel, single-conversion mode).
5. In the continuous conversion modes, repeat Step 3 and Step 4 for as long as samples are needed. Then stop the A/D Converter by clearing either the START bit (ADCCNT2.START) or the A/D Converter enable bit (ADCCNT1.ADCEN).

To minimize power consumption, the A/D Converter should be disabled when it is not needed, especially before entering a Power Save mode.

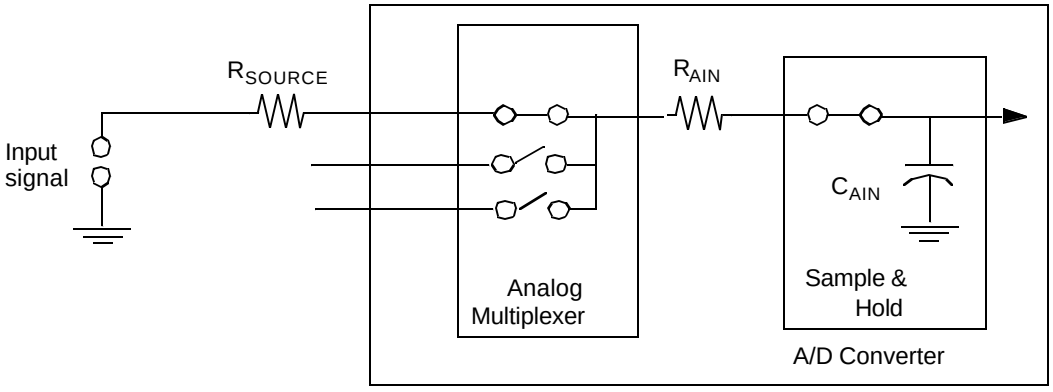


Figure 35. Sample-and-Hold Charge-Up Schematic

19.0 Memory Map

Please refer to individual datasheets for its own set of memory maps. The CompactRISC architecture supports a uniform linear address space of 2 megabytes. The device implementation of this architecture uses only the lowest 64 kbytes of address space, ranging from 0000 to FFFF hex. Table20 is a memory map showing the types of memory and peripherals that occupy this memory space. Address ranges not listed in the table are reserved and should not be read or written.

Table 20 Device Memory Map

Address Range (hex)	Description
0000-BFFF	Flash Program Memory (48 kbytes)
DA00-DFF7	ISP Memory (1.5 kbytes)
DFF8-DFFF	Program ROM control/status
E000-E7FF	Static RAM (2 kbytes)
F000-F27F	EEPROM Data Memory (640 bytes)
F900-F930	Device configuration registers
FB00-FB06	Port B registers
FB10-FB16	Port C registers
FC40-FC88	Clock, Power Management, and Wake-up registers
FCA0-FCA8	Port G registers
FD20-FD28	Port F registers
FE00-FE0C	Interrupt registers
FE40-FE4E	USART 1 registers
FE60-FE68	MICROWIRE registers
FE80-FE8E	USART 2 registers
FEE0-FEE8	Port I registers
FF00-FF08	Port L registers
FF20-FF2A	Timer and WATCHDOG registers
FF40-FF50	MFT1 Timer registers
FF60-FF70	MFT2 Timer registers
FFC0-FFD0	A/D Converter registers[a]
FFE0-FFE0	Analog Comparator register[a]

[a]. 44 pin devices may not include this module

Table21 is a detailed memory map showing the specific memory address of the memory, I/O ports, and registers. The table shows the starting address, the size, and a brief description of each memory block and register. For detailed information on using these memory locations, see the applicable sections in the data sheet.

All addresses not listed in the table are reserved and should not be read or written. An attempt to access an unlisted address will have unpredictable results.

Each byte-wide register occupies a single address and can be accessed only in a byte-wide transaction. Each word-wide register occupies two consecutive memory addresses and can be accessed only in a word-wide transaction. Both the byte-wide and word-wide registers reside at word boundaries (even address). Thus, each byte-wide register uses only the lowest eight bits of the internal data bus.

Most device registers are read/write registers. However, some registers are read-only or write-only, as indicated in the table. An attempt to read a write-only register or to write a read-only register will have unpredictable results.

When the software writes to a register in which one or more bits are reserved, it must write a zero to each reserved bit unless indicated otherwise in the description of the register. Reading a reserved bit returns an undefined value.

Table 21 Device Detailed Memory Map

Register Name	Size	Register Address (hex)	Access Type	Contents
	48K	0000	Read/Write	Flash Program Memory
FLCTRL	byte	DFFE	Read/Write	Flash Program Memory Control Register (EMPTY bit)
FLSEC	byte	DFFF	Read/Write	Flash Program Memory Security Register
	2K	E000	Read/Write	Static RAM
	640	F000	Read/Write	EEPROM Data Memory
BCFG	byte	F900	Read/Write	BIU Configuration Register
SZCFG0	word	F904	Read/Write	Static Zone 0 Configuration Register
MCFG	byte	F910	Read/Write	Module Configuration Register
MSTAT	byte	F914	Read Only	Module Status Register
DMCSR	byte	F940	Read/Write	EEPROM Data Memory Control and Status Register
DMPSLR	byte	F942	Read/Write	EEPROM Data Memory Prescaler Register
DMKEY	byte	F954	Read/Write	EEPROM Data Memory Write Key Register
FLCSR	byte	F960	Read/Write	Flash Program Memory Control and Status Register
FLPSLR	byte	F962	Read/Write	Flash Program Memory Prescaler Register
PGMKEY	byte	F974	Read/Write	Flash Program Memory Write Key Register
PBDIR	byte	FB00	Read/Write	Port B Direction Register
PBDIN	byte	FB02	Read Only	Port B Data Input Register
PBDOUT	byte	FB04	Read/Write	Port B Data Output Register
PBWKPU	byte	FB06	Read/Write	Port B Weak Pull-Up Register
PCDIR	byte	FB10	Read/Write	Port C Direction Register
PCDIN	byte	FB12	Read/Write	Port C Data Input Register (read-only)
PCDOUT	byte	FB14	Read/Write	Port C Data Output Register
PCWKPU	byte	FB16	Read/Write	Port C Weak Pull-Up Register
CRCTRL	byte	FC40	Read/Write	Clock and Reset Control Register
PRSSC	byte	FC42	Read/Write	Slow Clock Prescaler Register
PMCSR	byte	FC60	Read/Write	Power Management Control/Status Register
WKEDG	byte	FC80	Read/Write	Wake-Up Edge Detection Register
WKENA	byte	FC82	Read/Write	Wake-Up Enable Register
WKCTRL	byte	FC84	Read/Write	Wake-Up Source Select Register
WKPND	byte	FC86	Read Set	Wake-Up Pending Register
WKPCL	byte	FC88	Write Only	Wake-Up Pending Clear Register
PGALT	byte	FCA0	Read/Write	Port G Alternate Function Register
PGDIR	byte	FCA2	Read/Write	Port G Direction Register
PGDIN	byte	FCA4	Read Only	Port G Data Input Register
PGDOUT	byte	FCA6	Read/Write	Port G Data Output Register
PGWKPU	byte	FCA8	Read/Write	Port G Weak Pull-Up Register
PGSCHEN	byte	FCAA	Read/Write	Port G Schmitt Trigger Enable Register
PFALT	byte	FD20	Read/Write	Port F Alternate Function Register
PFDIR	byte	FD22	Read/Write	Port F Direction Register
PFDIN	byte	FD24	Read Only	Port F Data Input Register
PFDOUT	byte	FD26	Read/Write	Port F Data Output Register

Table 21 Device Detailed Memory Map

Register Name	Size	Register Address (hex)	Access Type	Contents
PFWKPU	byte	FD28	Read/Write	Port F Weak Pull-Up Register
PFSCHEN	byte	FD2A	Read/Write	Port F Schmitt Trigger Enable Register
IVCT	byte	FE00	Read Only	Interrupt Vector Register
NMISTAT	byte	FE02	Read Only	NMI Status Register
EXNMI	byte	FE04	Read/Write	External NMI Control/Status Register
ISTAT0	byte	FE0A	Read Only	Interrupt Status Register 0
ISTAT1	byte	FE0C	Read Only	Interrupt Status Register 1
IENAM	byte	FE0E	Read/Write	Interrupt and Enable Mask Register 0
IENAM1	byte	FE10	Read/Write	Interrupt and Enable Mask Register 1
U1TBUF	byte	FE40	Read/Write	USART 1 Transmit Data Buffer
U1RBUF	byte	FE42	Read Only	USART 1 Receive Data Buffer
U1ICTRL	byte	FE44	Read/Write	USART 1 Interrupt Control Register
U1STAT	byte	FE46	Read Only	USART 1 Status Register
U1FRS	byte	FE48	Read/Write	USART 1 Frame Select Register
U1MDSL	byte	FE4A	Read/Write	USART 1 Mode Select Register
U1BAUD	byte	FE4C	Read/Write	USART 1 Baud Rate Divisor Register
U1PSR	byte	FE4E	Read/Write	USART 1 Baud Rate Prescaler
MWDAT	byte	FE60	Read/Write	MICROWIRE Data Register
MWCTL1	byte	FE62	Read/Write	MICROWIRE Control 1 Register
MWCTL2	byte	FE64	Read/Write	MICROWIRE Control 2 Register
MWCTL3	byte	FE66	Read/Write	MICROWIRE Control 3 Register
MWSTAT	byte	FE68	Read Only	MICROWIRE Status Register
U2TBUF	byte	FE80	Read/Write	USART 2 Transmit Data Buffer
U2RBUF	byte	FE82	Read Only	USART 2 Receive Data Buffer
U2ICTRL	byte	FE84	Read/Write	USART 2 Interrupt Control Register
U2STAT	byte	FE86	Read Only	USART 2 Status Register
U2FRS	byte	FE88	Read/Write	USART 2 Frame Select Register
U2MDSL	byte	FE8A	Read/Write	USART 2 Mode Select Register
U2BAUD	byte	FE8C	Read/Write	USART 2 Baud Rate Divisor Register
U2PSR	byte	FE8E	Read/Write	USART 2 Baud Rate Prescaler
PIALT	byte	FEE0	Read/Write	Port I Alternate Function Register
PIDIR	byte	FEE2	Read/Write	Port I Direction Register
PIDIN	byte	FEE4	Read Only	Port I Data Input Register
PIDOUT	byte	FEE6	Read/Write	Port I Data Output Register
PIWKPU	byte	FEE8	Read/Write	Port I Weak Pull-Up Register
PISCHEN	byte	FEEA	Read/Write	Port I Schmitt Trigger Enable Register
PLALT	byte	FF00	Read/Write	Port L Alternate Function Register
PLDIR	byte	FF02	Read/Write	Port L Direction Register
PLDIN	byte	FF04	Read Only	Port L Data Input Register (read-only)
PLDOUT	byte	FF06	Read/Write	Port L Data Output Register
PLWKPU	byte	FF08	Read/Write	Port L Weak Pull-Up Register
PLSCHEN	byte	FF0A	Read/Write	Port L Schmitt Trigger Enable Register

Table 21 Device Detailed Memory Map

Register Name	Size	Register Address (hex)	Access Type	Contents
TWCFG	byte	FF20	Read/Write	Timer and WATCHDOG Configuration Register
TWCP	byte	FF22	Read/Write	Timer and WATCHDOG Clock Prescaler Register
TWMT0	word	FF24	Read/Write	TWM Timer 0 Register
T0CSR	byte	FF26	Read/Write	TWMT0 Control and Status Register
WDCNT	byte	FF28	Write Only	WATCHDOG Count Register
WDSDM	byte	FF2A	Write Only	WATCHDOG Service Data Match Register
T1CNT1	word	FF40	Read/Write	T1 Timer/Counter I Register
T1CRA	word	FF42	Read/Write	T1 Reload/Capture A Register
T1CRB	word	FF44	Read/Write	T1 Reload/Capture B Register
T1CNT2	word	FF46	Read/Write	T1 Timer/Counter II Register
T1PRSC	byte	FF48	Read/Write	T1 Clock Prescaler Register
T1CKC	byte	FF4A	Read/Write	T1 Clock Unit Control Register
T1CTRL	byte	FF4C	Read/Write	T1 Timer Mode Control Register
T1ICTL	byte	FF4E	Read/Write	T1 Timer Interrupt Control Register
T1ICLR	byte	FF50	Read/Write	T1 Timer Interrupt Clear Register
T2CNT2	word	FF60	Read/Write	T2 Timer/Counter I Register
T2CRA	word	FF62	Read/Write	T2 Reload/Capture A Register
T2CRB	word	FF64	Read/Write	T2 Reload/Capture B Register
T2CNT2	word	FF66	Read/Write	T2 Timer/Counter II Register
T2PRSC	byte	FF68	Read/Write	T2 Clock Prescaler Register
T2CKC	byte	FF6A	Read/Write	T2 Clock Unit Control Register
T2CTRL	byte	FF6C	Read/Write	T2 Timer Mode Control Register
T2ICTL	byte	FF6E	Read/Write	T2 Timer Interrupt Control Register
T2ICLR	byte	FF70	Read/Write	T2 Timer Interrupt Clear Register
ADCST	byte	FFC0	Read/Write	A/D Converter Status Register
ADCCNT1	byte	FFC2	Read/Write	A/D Converter Control 1 Register
ADCCNT2	byte	FFC4	Read/Write	A/D Converter Control 2 Register
ADCCNT3	byte	FFC6	Read/Write	A/D Converter Control 3 Register
ADDATA0	byte	FFCA	Read/Write	A/D Converter Data 0 Register
ADDATA1	byte	FFCC	Read Only	A/D Converter Data 1 Register
ADDATA2	byte	FFCE	Read Only	A/D Converter Data 2 Register
ADDATA3	byte	FFD0	Read Only	A/D Converter Data 3 Register
CMPCTRL	byte	FFE0	Read/Write	Analog Comparator Control/Status Register

20.0 Register Layouts

The following tables show the functions of the bit fields of the device registers. For more information on using these registers, see the detailed description of the applicable function elsewhere in the data sheet.

20.1 REGISTER LAYOUT

System Configuration Registers	7	6	5	4	3	2	1	0
MCFG	Reserved		CLK2OE	SLCOE2	FEEDM	SLCLKOE	CLKOE	Reserved
MSTAT	Reserved				PGMBUSY	OENV2	OENV1	OENV0

BIU Registers	15 12	11	10	9	8	7	6	5	4	3	2	1	0
BCFG						Reserved							EWR
IOCFG	Reserved		1	IPST	Res	BW	Reserved		HOLD		WAIT		
SZCFG0	Reserved	FRE	IPRE	IPST	Res	BW	Reserved		HOLD		WAIT		

EEPROM Data Memory Registers	7	6	5	4	3	2	1	0
DMCSR	Reserved					DMBUSY	ERASE	Reserved
DMPSLR	FTDIV							
DMKEY	DMKEYVAL							

Flash Program Memory Registers	7	6	5	4	3	2	1	0
FLCSR	Reserved			IENPROG	PMLFULL	PMBUSY	ERASE	Reserved
FLCSR	FTDIV							
PGMKEY	PMKEYVAL							
FLSEC	Reserved					FROMWR	ERASE	FROMRD

GPIO Registers	7	6	5	4	3	2	1	0
PxALT	Px Pins Alternate Function Enable							
PxDIR	Px Port Direction							
PxDIN	Px Port Output Data							
PxDOUT	Px Port Input Data							
PxWPU	Px Port Weak Pull-up Enable							
PxSCHEN	Px Port Schmitt Trigger Enable							

ICU Registers	7	6	5	4	3	2	1	0
IVCT	0	0	0	1	INTVECT			
NMISTAT	Reserved							EXT
NMIMNTR	Reserved				WD	ERR	UND	EXT
ISTAT0	IST(7:0)							
ISTAT1	IST(15:8)							
IENAM0	IENA(7:0)							
IENAM1	IENA(15:8)							
EXNMI	Reserved					ENCLK	PIN	EN

MIWU Registers	7	6	5	4	3	2	1	0
WKEDG	WKED7	WKED6	WKED5	WKED4	WKED3	WKED2	WKED1	WKED0
WKENA	WKEN7	WKEN6	WKEN5	WKEN4	WKEN3	WKEN2	WKEN1	WKEN0
WKCTRL	Reserved							WKSEL0
WKPND	WKPD7	WKPD6	WKPD5	WKPD4	WKPD3	WKPD2	WKPD1	WKPD0
WKPCL	WKCL7	WKCL6	WKCL5	WKCL4	WKCL3	WKCL2	WKCL1	WKCL0

Dual Clock + Reset Registers	7	6	5	4	3	2	1	0
CRCTRL	Reserved						POR	SCLK
PRSSC	SCDIV							

Power Management Register	7	6	5	4	3	2	1	0
PMCSR	OLFC	OHFC	WBPSM	Reserved	HALT	IDLE	DHF	PSM

USART Registers	7	6	5	4	3	2	1	0
UnTBUF	UnTBUF							
UnRBUF	UnRBUF							
UnICTRL	UnEEI	UnERI	UnETI	Reserved			UnRBF	UnTBE
UnSTAT	Reserved	UnXMIP	UnRB9	UnBKD	UnERR	UnDOE	UnFE	UnPE
UnFRS	Reserved	UnPEN	UnPSEL		UnXB9	UnSTP	UnCHAR	
UnMDSL	Reserved				UnCKS	UnBRK	UnATN	UnMOD
UnBAUD	UnDIV[7]: UnDIV[0]							
UnPSR	UnPSC					UnDIV[10]: UnDIV[8]		

MWSPI Registers	7	6	5	4	3	2	1	0
MWDAT	MWnDAT							
MWCTL1	Reserved				MEN	MIDL	MCKM	MMNS
MWCTL2	MDV1	MCDV						
MWCTL3	Reserved		MEIW	MEIR	MEIO	MECH	MRDY	MBFL
MWSTAT	Reserved				MOVR	Reserved	MRBF	MBSY

TIMER Registers	15 8	7	6	5	4	3	2	1	0
TnCNT1	TnCNT1								
TnCRA	TnCRA								
TnCRB	TnCRB								
TnCNT2	TnCNT2								
TnPRSC		Reserved			CLKPS				
TnCKC		Reserved		C2CSEL			C1CSEL		
TnCTRL		Reserved	TnAOUT	TnBEN	TnAEN	TnBEDG	TnAEDG	MDSEL	
TnICTL		TnDIEN	TnCIEN	TnBIEN	TnAIEN	TnDPND	TnCPND	TnBPND	TnAPND
TnICLR		Reserved				TnDCLR	TnCCLR	TnBCLR	TnACLR

TWM Registers	15 8	7	6	5	4	3	2	1	0
TWCFG		Reserved		WDSDME	WDCT0I	LWDCNT	LTWMT0	LTWCP	LTWCFG
TWCP		Reserved					MDIV		
TWMTO	PRESET								
T0CSR		Reserved					T0INTE	TC	RST
WDCNT		PRESET							
WDSDM		RSTDATA							

A/D Registers	7	6	5	4	3	2	1	0
ADCST	Reserved		BUFPTR		Reserved	OVF	BUSY	EOC
ADCCNT1	Reserved					INTE	Reserved	ADCEN
ADCCNT2	START	SCAN		CONT	CHANNEL			
ADCCNT3	Reserved		DELAY		CDIV			
ADDATA0	RESULT 1 DATA							
ADDATA1	RESULT 2 DATA							
ADDATA2	RESULT 3 DATA							
ADDATA3	RESULT 4 DATA							

Analog Comp. Registers	7 6	5	4	3	2	1	0
CMPCTRL	Reserved	CMP2OE	CMP1OE	CMP2EN	CMP1EN	CMP2RD	CMP1RD

21.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	–0.6V to V_{CC} +0.6V
ESD Protection Level	6 kV (Human Body Model)
Total Current into V_{CC} Pin (Source)	80 mA
Total Current out of GND Pin (Sink)	100 mA
Storage Temperature Range	–65°C to +140°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

Thermal Characteristics

Characteristics	Symbol	Value	Unit
Average junction temperature	T_J	$T_A + (P_D \times \Theta_{JA})$	°C
Ambient temperature	T_A	User-determined	°C
Package thermal resistance (junction-to-ambient) 80-pin quad flat pack (QFP) 44-pin package (PLCC)	Θ_{JA}	 49.8 56	°C/W
Total power dissipation[1]	P_D	$P_{INT} + P_{I/O}$ or $\frac{K}{T_J + 273°C}$	W
Device internal power dissipation	P_{INT}	$I_{DD} \times V_{DD}$	W
I/O pin power dissipation[2]	$P_{I/O}$	User-determined	W
A constant[3]	K	$P_D \times (T_A + 273°C) + \Theta_{JA} \times P_D^2$	W, °C

1. This is an approximate value, neglecting $P_{I/O}$.

2. For most applications $P_{I/O} << P_{INT}$ and can be neglected.

3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iterntively for any value of T_A.

DC Electrical Characteristics: –40°C ≤ TA ≤ +85°C (also supports -40°C to 125°C) depends on package type

Symbol	Parameter	Conditions	Min	Max	Units
	Operating Voltage		4.5	5.5	V
V_{IH}	Logical 1 CMOS Hysteresis Input Voltage		0.8Vcc	Vcc + 0.5	V
V_{IL}	Logical 0 CMOS Input Voltage		-0.5	0.2Vcc	V
Vxl	Low Level Input Voltage OSC	External X1 clock	0 Vcc	0.2Vcc	V
Vxh	High Level Input Voltage OSC	External X1 clock	0.5Vcc	Vcc	V
Vxl2	X2CKI Logical 0 Input Voltage	External X2 clock		0.3	V
Vxh2	X2CKI Logical 1 Input Voltage	External X2clock	1.2		V
V_{hys}	Hysteresis Loop Width [a b]		0.1Vcc		V
I_{OH}	Logical 1 CMOS Output Current	V_{OH} = 3.8V, Vcc=4.5V	-1.6		mA
I_{OL}	Other Pins Logical 0 CMOS Output Current	V_{OL} = 0.45V, Vcc=4.5V	1.6		mA
I_{OHW}	Weak Pull-up Current	V_{OH} = 3.8V, Vcc=4.5V	-10		µA
I_L	Input Leakage Current	0V ≤ Vin ≤ Vcc	- 2.0	2.0[j]	µA
I_O(Off)	Output Leakage Current (I/O pins in input mode)	0V ≤ Vout ≤ Vcc	- 2.0	2.0[j]	µA

Symbol	Parameter	Conditions	Min	Max	Units
Icca1	Digital Supply Current Active Mode, normal [c]	TC = 20 MHz, Vcc= 5.5V		38	mA
Iccprog	Digital Supply Current Active Mode, flash mem.[d]	TC = 20 MHz, Vcc = 5.5V		46	mA
Icca2	Digital Supply Current Active Mode, WAIT [e]	TC = 20 MHz, Vcc = 5.5V		23.2	mA
Icca1	Digital Supply Current Active Mode, normal [c]	TC = 8 MHz, Vcc= 5.5V		38	mA
Iccprog	Digital Supply Current Active Mode, flash mem.[d]	TC = 8 MHz, Vcc = 5.5V		46	mA
Icca2	Digital Supply Current Active Mode, WAIT [e]	TC = 8 MHz, Vcc = 5.5V		23.2	mA
Iccps	Digital Supply Current Power Save Mode [f]	Vcc= 5.5V		9	mA
Iccid	Digital Supply Current Idle Mode [g]	Vcc = 5.5V		60	µA
Iccq	Digital Supply Current Halt Mode [h]	Vcc = 5.5V		15[k]	µA
Iacc	Analog Supply Current Active Mode [i]	Vcc = 5.5V		3	mA

a. Guaranteed by design.
b. Hysteresis applies only when the Schmitt trigger is enabled or when alternate function is selected.
c. Running from internal memory, Iout = 0 mA, XCKI1 = 20 and 8 MHz, not programming flash memory.
d. Same conditions as c. (Icca1), but while programming or erasing one of the flash memory arrays.
e. CPU executing an WAIT instruction, Iout = 0 mA, XCKI1 = 20 and 8 MHz, peripherals not active.
f. Running from internal memory, Iout = 0mA, Iout = 0mA, XCKI1 = 20 MHz, XCKI2 = 32.768 kHz.
g. Iout = 0mA, XCKI1 = Vcc, X2CKI = 32.768 kHz
h. Same conditions as g. (Iccps), but in Halt mode instead of Idle mode
i. A/D converter and analog comparators enabled.
j. I_L and I_O are 2.0 µA at 85°C and 5.0 µA at 125°C
k. I_{acq} is 20 µA at 85°C and 50 µA at 125°Ci.

A/D Converter Characteristics

Symbol	Parameter	Conditions[a]	Min	Typ	Max	Units
N_{IL}	Integral Error[b]	$V_{REF} = V_{CC}$			1	LSB
N_{DL}	Differential Error[c]	$V_{REF} = V_{CC}$			1	LSB
V_{IN}	Input Voltage Range	$V_{REF} < V_{CC}$ - 0.1	0		V_{REF}	V
V_{REFEX}	External Reference Voltage		3.0		V_{DD}	V
V_{REFI}	Internal Reference Voltage		2.375	2.5	2.625	V
I_{VREF}	V_{REF} input current	V_{REF} = 5 V			1.0	mA
I_{AL}	Analog input leakage current	$V_{REF} = V_{CC}$			±1	µA
R_{AIN}	Analog input resistance[d]				200	Ω
C_{AIN}	Analog input capacitance[e]				5	pF
tADCCLK	Conversion Clock period				1	µs
C_{REFEX}	External Vref bypass capacitance		0.47			µF
t_{ACT}	First A/D conversion after Vcc stable		100			µsec

a. All parameters specified for f_{OSC} = 1 MHz,
V_{DD} = 5.0V ± 10%, V_{DD} = 5.0V ± 10% unless otherwise noted.
b. Integral (non-linearity) error; the maximum difference between the best-fit straight line reference and the actual conversion curve.
c. Differential (non-linearity) error; the maximum difference between the best-fit stair-step reference with 1-LSB resolution and the actual conversion curve.
d. The resistance between the device input and the internal analog input capacitance.
e. The input signal is measured across the internal capacitance.

Comparator AC and DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{OS}	Input Offset Voltage	Vcc = 5V, $0.4V \le V_{IN} \le V_{CC} - 1.5V$			±25	mV
V_{CM}	Input Common Mode Voltage Range		0.4		V_{CC} -1.5	V
I_{CS}	DC Supply Current per Comparator (When Enabled)	V_{CC}=5.5V			250	µA
	Response Time	1V Step / 100mV Overdrive			1	µs

Flash EEPROM Program Memory Programming

Symbol	Parameter	Conditions	Min	Max	Units
t_{PWP}	Programming pulse width [a]		30	40	µs
t_{EWP}	Erase pulse width[b]		1	-	ms
t_{SDP}	Charge pump power-up delay[c]		10	-	µs
t_{TTP}	Program/erase transition time[d]		5	-	µs
t_{PAH}	Programming address hold, new address setup time		2	-	clock cycles
t_{PEP}	Charge pump enable hold time		1	-	clock cycles
t_{EDP}	Charge pump power hold time[e]		5		µs
t_{CHVP}	Cumulative program high voltage period for each row after erase.[f]		-	25	ms
	Data retention		100	-	years
			-	100K	cycles

a. The programming pulse width is determined by the following equation:
$t_{PWP} = T_{clk}$ x (FTDIV+1) x (FTPROG+1), where T_{clk} is the system clock period, FTDIV is the contents of the FLPSLR register and FTPROG is the contents of the FLPROG register.

b. The erase pulse width is determined by the following equation:
$t_{EWP} = T_{clk}$ x (FTDIV+1) x 4 x (FTER+1), where T_{clk} is the system clock period, FTDIV is the contents of the FLPSLR register and FTER is the contents of the FLERASE register.

c. The program/erase start delay time is determined by the following equation:
$t_{SDP} = T_{clk}$ x (FTDIV+1) x (FTSTART+1), where T_{clk} is the system clock period, FTDIV is the contents of the FLPSLR register and FTSTART is the contents of the FLSTART register.

d. The program/erase transition time is determined by the following equation:
$t_{TTP} = T_{clk}$ x (FTDIV+1) x (FTTRAN+1), where T_{clk} is the system clock period, FTDIV is the contents of the FLPSLR register and FTTRAN is the contents of the FLTRAN register.

e. The program/erase end delay time is determined by the following equation:
$t_{EDP} = T_{clk}$ x (FTDIV+1) x (FTEND+1), where T_{clk} is the system clock period, FTDIV is the contents of the FLPSLR register and FTEND is the contents of the FLEND register.

f. Cumulative program high voltage period for each row after erase t_{CHVP} is the accumulated duration a flash cell is exposed to the programming voltage after the last erase cycle. It is the sum of all t_{HV} after the last erase.

Flash EEPROM Data Programming

Symbol	Parameter	Conditions	Min	Max	Units
	re-programming time[a]		1.32	-	ms
t_{PWD}	Programming pulse width [b]		30	40	µs
t_{EWD}	Erase pulse width[c]		1	-	ms
t_{SDD}	Charge pump power-up time[d]		10	-	µs
t_{TTD}	Program/erase transition time[e]		5	-	µs
t_{PED}	Charge pump enable hold time		1	-	clock cycles
t_{EDD}	Charge pump power hold time[f]		5	-	µs
	Write/erase endurance (high endurance)		100,000	-	cycles
	Data retention		100	-	years

a. One re-programming cycle involves one erase pulse followed by programming of four bytes.

b. The programming pulse width is determined by the following equation:
$t_{PWD} = T_{clk}$ x (FTDIV+1) x (FTPROG+1), where T_{clk} is the system clock period, FTDIV is the contents of the DMPSLR register and FTPROG is the contents of the DMPROG register.

c. The erase pulse width is determined by the following equation:
$t_{EWD} = T_{clk}$ x (FTDIV+1) x 4 x (FTER+1), where T_{clk} is the system clock period, FTDIV is the contents of the DMPSLR register and FTER is the contents of the DMERASE register.

d. The program/erase start delay time is determined by the following equation:
$t_{SDD} = T_{clk}$ x (FTDIV+1) x (FTSTART+1), where T_{clk} is the system clock period, FTDIV is the contents of the DMPSLR register and FTSTART is the contents of the DMSTART register.

e. The program/erase transition time is determined by the following equation:
$t_{TTD} = T_{clk}$ x (FTDIV+1) x (FTTRAN+1), where T_{clk} is the system clock period, FTDIV is the contents of the DMPSLR register and FTTRAN is the contents of the DMTRAN register.

f. The program/erase end delay time is determined by the following equation:
$t_{EDD} = T_{clk}$ x (FTDIV+1) x (FTEND+1), where T_{clk} is the system clock period, FTDIV is the contents of the DMPSLR register and FTEND is the contents of the DMEND register.

Flash EEPROM ISP-Memory Programming

Symbol	Parameter	Conditions	Min	Max	Units
t_{PWI}	Programming pulse with[a]		30	40	µs
t_{EWI}	Erase pulse width[b]		1	-	ms
	Data retention		100	-	years

a. Programming timing is controlled by the flash EEPROM data memory interface
b. Erase timing is controlled by the flash EEPROM data memory interface

Output Signal Levels

All output signals are powered by the digital supply (V_{CC}).

Table22 summarizes the states of the output signals during the reset state (when V_{CC} power exists in the reset state) and during the Power Save mode.

The $\overline{\text{RESET}}$ and $\overline{\text{NMI}}$ input pins are active during the Power Save mode. In order to guarantee that the Power Save current not exceed 1mA, these inputs must be driven to a voltage lower than 0.5V or higher than V_{CC}-0.5V. An input voltage between 0.5V and (V_{CC}-0.5V) may result in power consumption exceeding 1 mA.

Table 22 Output Pins During Reset and Power-Save

Signals on a pin	Reset state (with Vcc)	Power Save mode	Comments
PF[0:7]	TRI-STATE	Previous state	I/O ports will maintain their values when entering power-save mode
PG[0:7]	TRI-STATE	Previous state	
PI[0:7]	TRI-STATE	Previous state	
PL[0:7]	TRI-STATE	Previous state	
PB[0:7]	TRI-STATE	Previous state	
PC[0:7]	TRI-STATE	Previous state	

21.0.1 Timing Waveforms

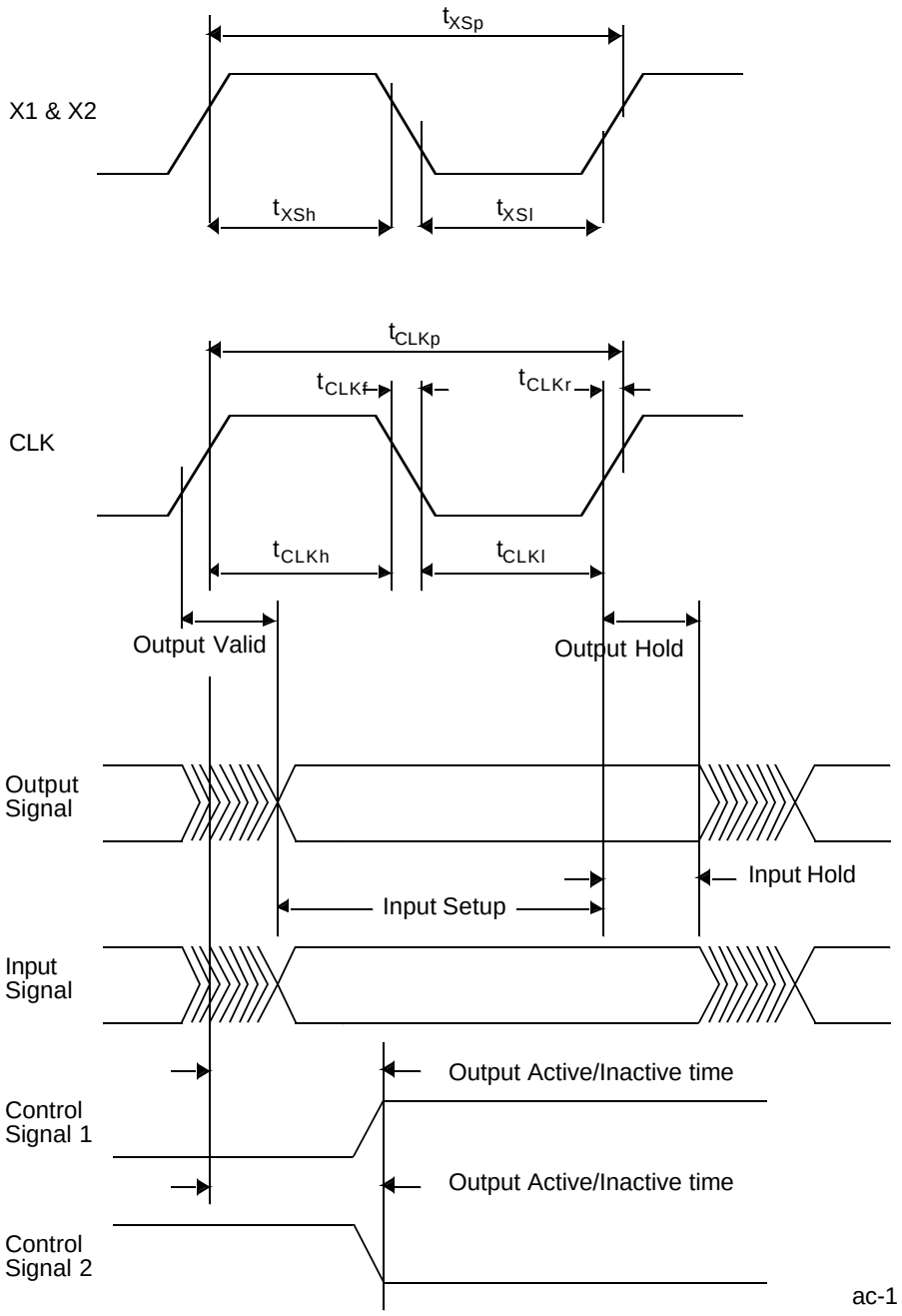


Figure 36. Clock Waveforms

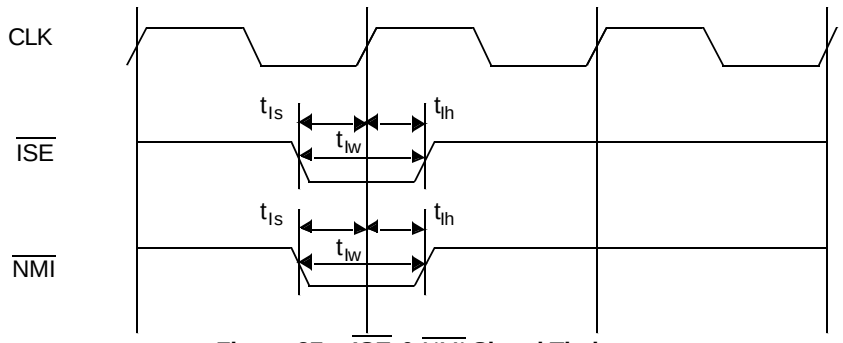


Figure 37. $\overline{ISE}$ & $\overline{NMI}$ Signal Timing

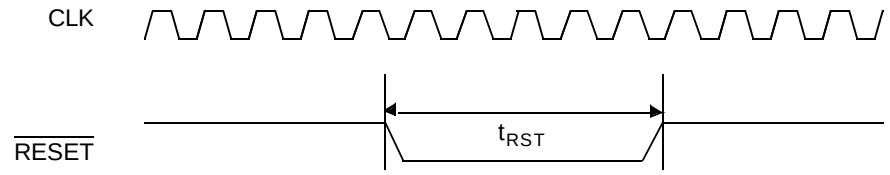


Figure 38. Non-Power-On Reset

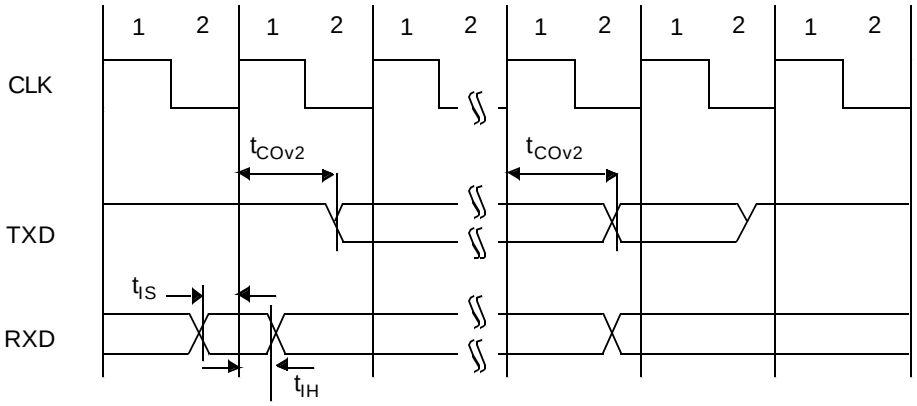


Figure 39. USART Asynchronous Mode Timing

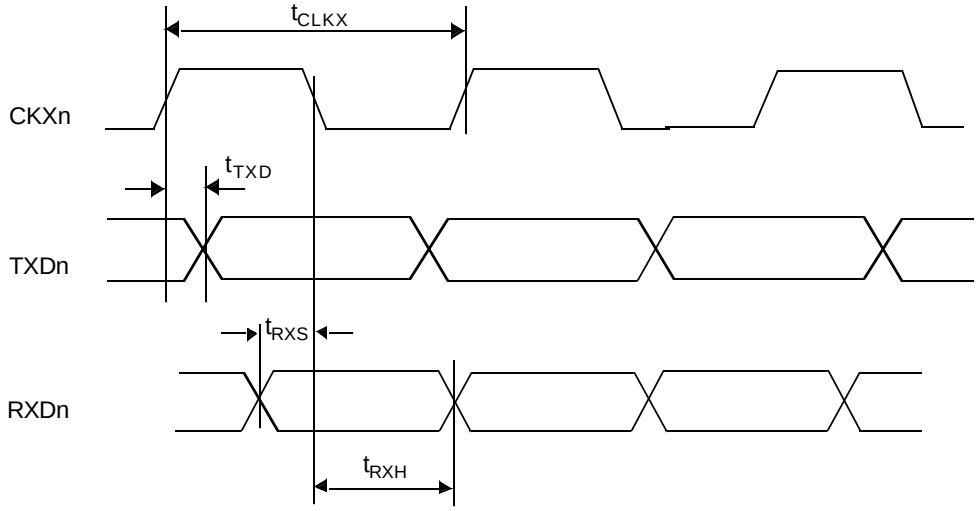


Figure 40. USART Synchronous Mode Timing

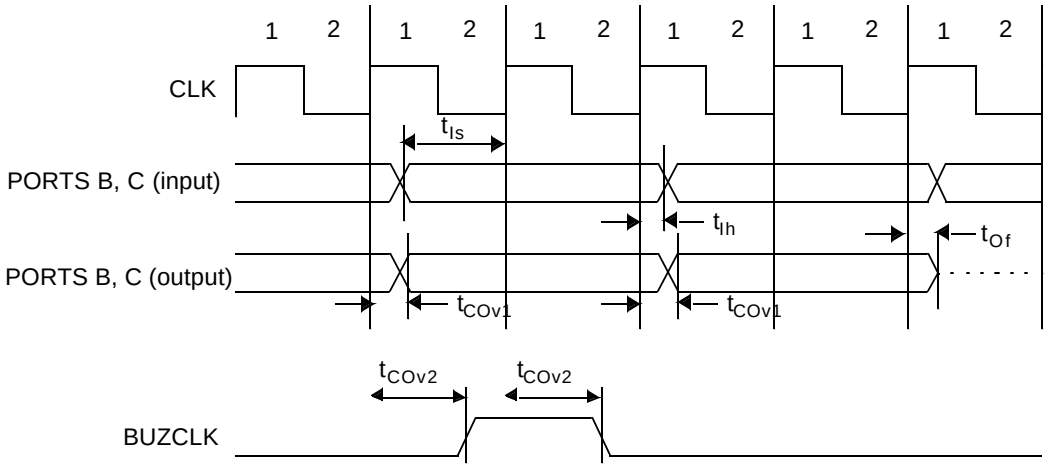


Figure 41. Ports Signals Timing

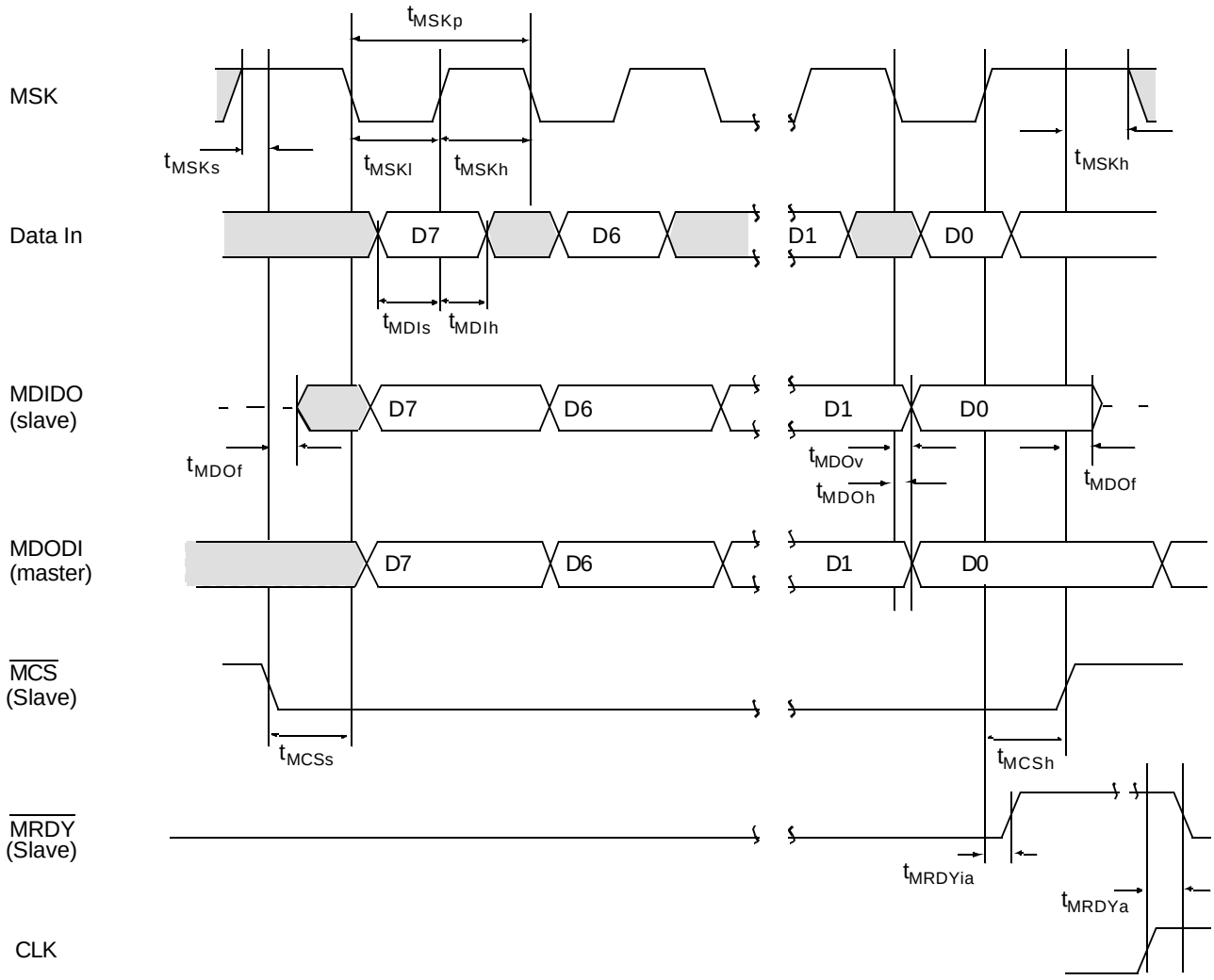


Figure 42. MICROWIRE Transaction Timing, Normal Mode, MIDL Bit = 1

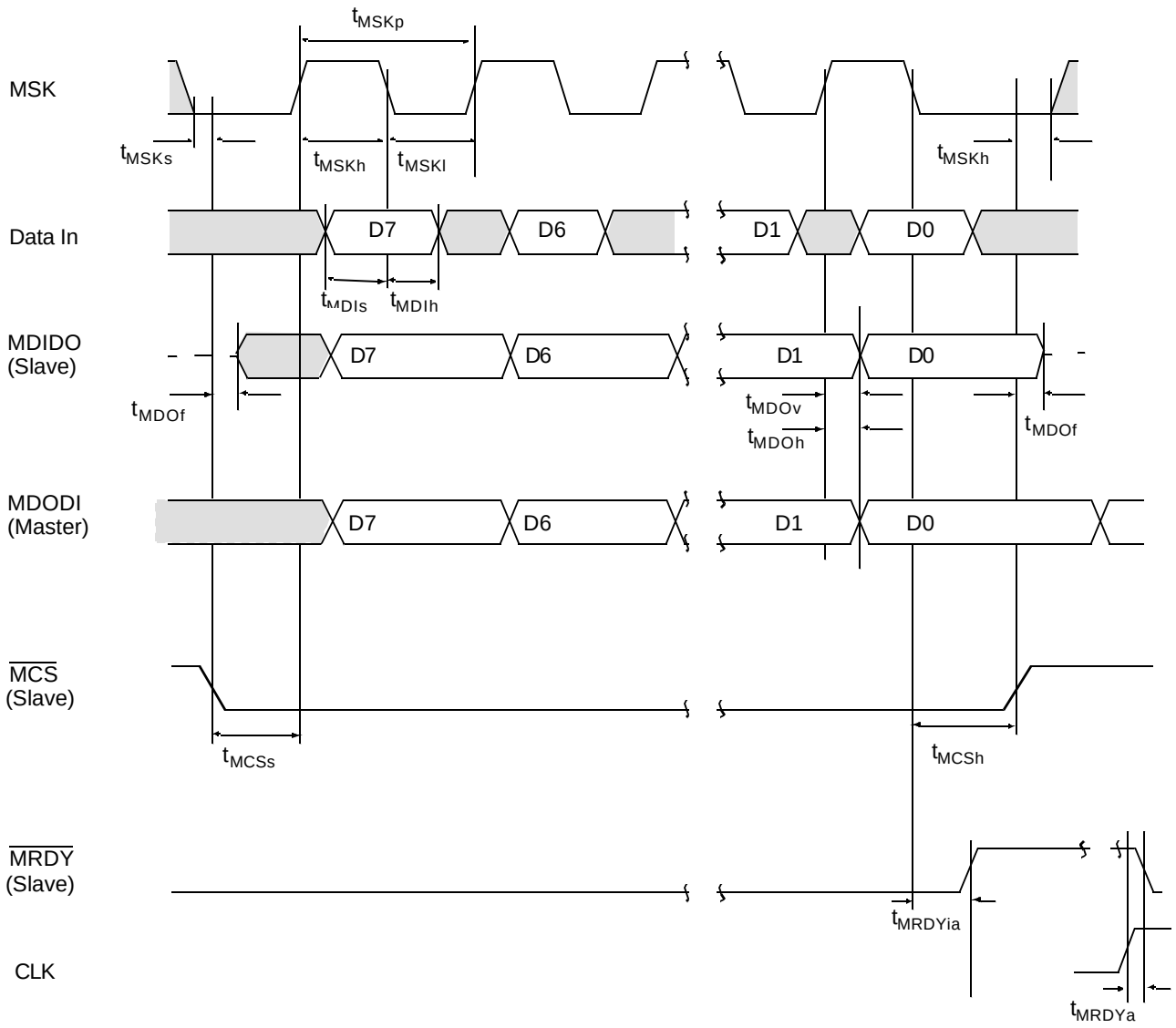


Figure 43. MICROWIRE Transaction Timing, Alternate Mode, MIDL bit = 0

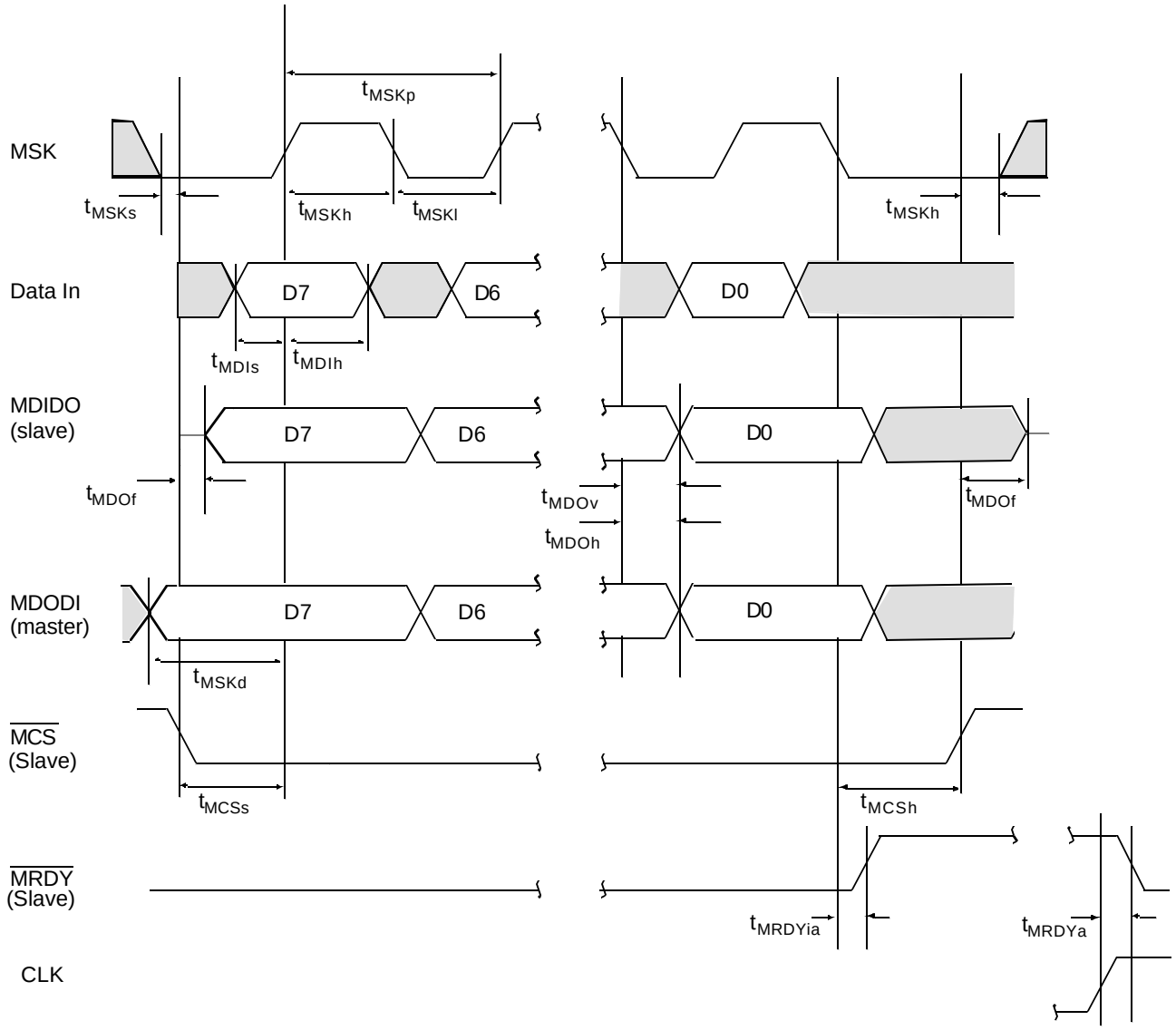


Figure 44. MICROWIRE Transaction Timing, Normal Mode, MIDL bit = 0

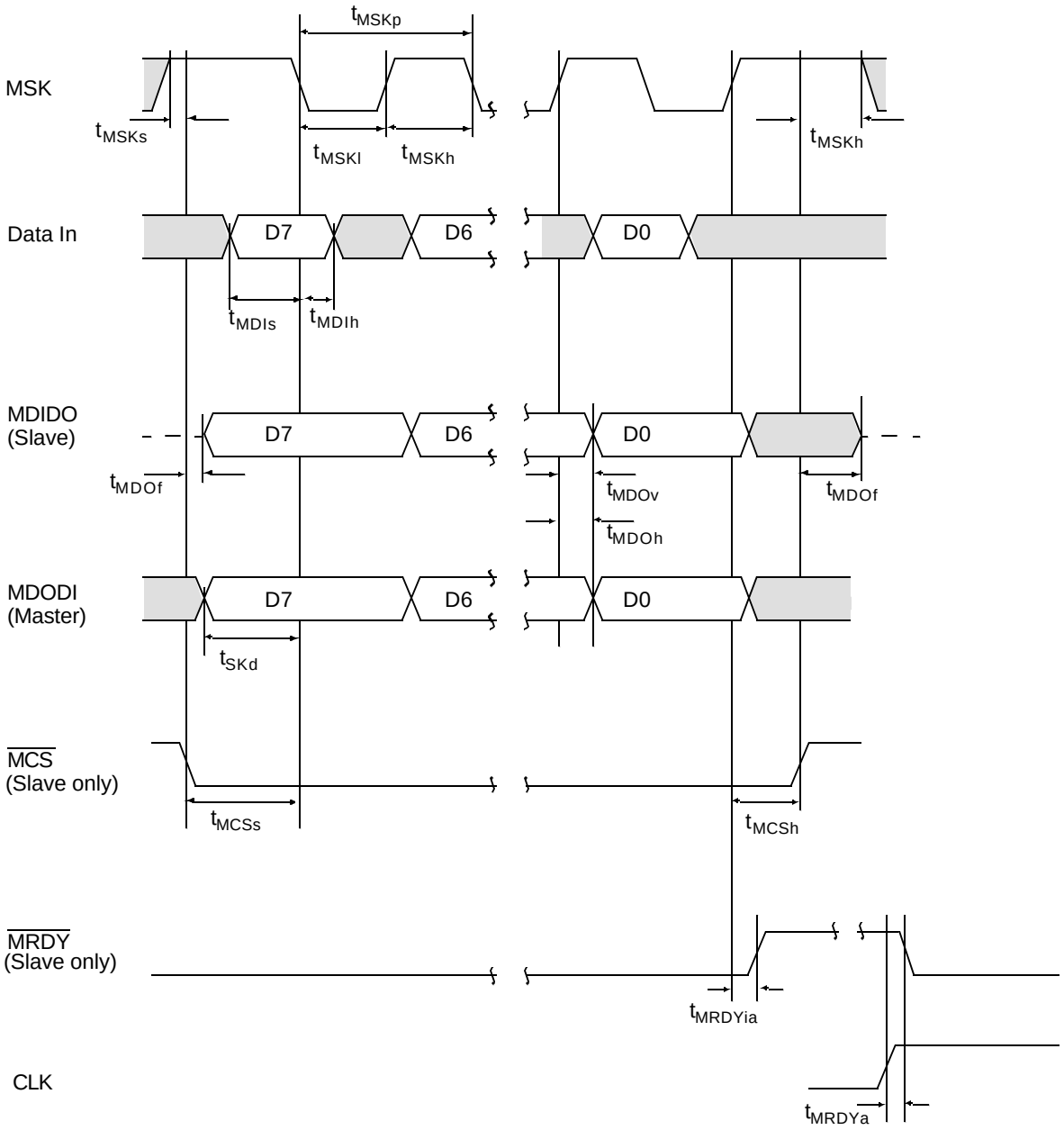


Figure 45. MICROWIRE Transaction Timing, Alternate Mode, MIDL bit = 1

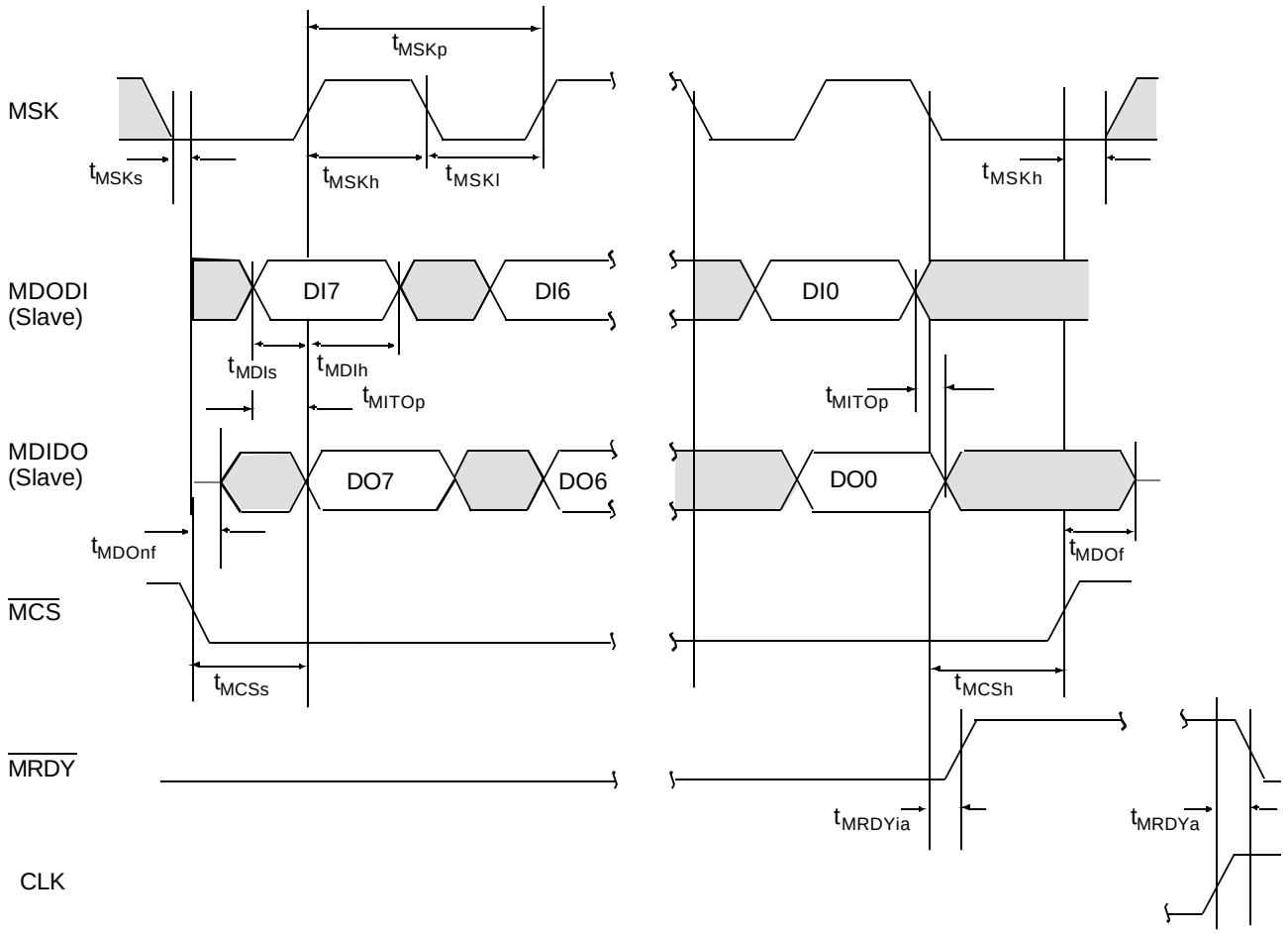


Figure 46. MICROWIRE Transaction Timing, Data Echoed to Output, Normal Mode, MIDLBit= 0, MECHBit= 1, Slave

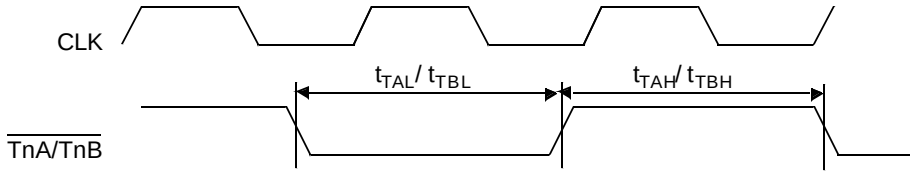


Figure 47. Multi-Function-Timer (MFT16) Input Timing

21.0.2 Timing Tables

Table 23 Output Signals

Symbol	Figure	Description	Reference	Min (ns)	Max (ns)
Tclk [a]	36	CLK clock period	R.E. CLK to next R.E. CLK	50	64000[a]
t_{CLKh}	36	CLK high time	At 2.0V (Both Edges)	17.3	
t_{CLKl}	36	CLK low time	At 0.8V (Both Edges)	17.3	
t_{CLKr}	36	CLK rise time on R.E. CLK	0.8V to 2.0V		3
t_{CLKf}	36	CLK fall time on F.E. CLK	2.0V to 0.8V		3
t_{COv1}		CMOS output valid All signals with prop. delay from CLK R.E.	After R.E. CLK		35
USART Output Signals					
t_{TXD}	45	TXDn output valid	After R.E. CLKXn		35
MICROWIRE / SPI Output Signals					
t_{MSKh}	42	MICROWIRE Clock High	At 2.0V (both edges)	80	
t_{MSKl}	42	MICROWIRE Clock Low	At 0.8V (both edges)	80	
t_{MSKp}	42	MICROWIRE Clock Period	MnIDL bit = 0: R.E. MSK to next R.E. MSKn	200	
	43		MnIDL bit = 1: F.E. MSK to next F.E. MSKn		
t_{MSKd}	42	MSK Leading Edge Delayed (master only)	Data Out Bit #7 Valid	0.5 t_{MSK}	1.5 t_{MSK}
t_{MDOf}	42	MICROWIRE Data Float [b] (slave only)	After R.E. $\overline{MCSn}$		56
t_{MDOh}	42	MICROWIRE Data Out Hold	Normal Mode: After F.E. MSK	0.0	
			Alternate Mode: After R.E. MSK		
t_{MDOnf}	42	MICROWIRE Data No Float (slave only)	After F.E. $\overline{MWCS}$	0	56
t_{MDOv}	42	MICROWIRE Data Out Valid	Normal Mode: After F.E. MSK		56
			Alternate Mode: After R.E. MSK		
t_{MITOp}	46	MDODI to MDIDO (slave only)	Propagation Time Value is the same in all clocking modes of the MICROWIRE		56
t_{MRDYa}	42	$\overline{MRDY}$ Active (slave only)	After R.E. of CLK	0	28
t_{MRDYia}	42	$\overline{MRDY}$ Inactive (slave only)	MIDL bit = 0: After F.E. MSK	0	56
			MIDL bit = 1: After R.E. MSK		

a. Tclk is the actual clock period of the CPU clock used in the system.
The value of Tclk is system dependent.
The maximum cycle time of 64000ns is for Power Save mode; in active mode, the maximum cycle time is limited to 250ns by the high frequency oscillator.
b. Guaranteed by design, but not fully tested.
c. Hold time is 0 ns (min) for all outputs, unless specified otherwise.

Table 24 Input Signal Requirements

Symbol	Figure	Description	Reference	Min (ns)	Max (ns)
t_{XSp}	36	X1 period	R.E. X1 to next R.E. X1	50	250
t_{XSh}	36	X1 high time, external clock	At 2V level (Both Edges)	0.5 Tclk - 4	
t_{XSl}	36	X1 low time, external clock	At 0.8V level (Both Edges)	0.5 Tclk - 4	
t_{X2p}	36	X2 period [a]	R.E. X2 to next R.E. X2	10,000	
t_{X2h}	36	X2 high time, external clock	At 2V level (both edges)	0.5 Tclk - 500	

Table 24 Input Signal Requirements

Symbol	Figure	Description	Reference	Min (ns)	Max (ns)
t_{X2l}	36	X2 low time, external clock	At 0.8V level (both edges)	0.5 Tclk - 500	
t_{Is}	42	Input setup time ISE	Before R.E. CLK	12	
t_{Ih}	42	Input hold time $\overline{ISE}$, $\overline{NMI}$, RXD1, RXD2	After R.E. CLK	0	
t_{RST}	43	Reset time	Reset active to reset end	4Tclk	
Input Signals					
		Input Pulse Width		1*Tclk+13	
USART Input Signals					
t_{Is}	39	Input setup time RXDn (asynchronous mode)	Before R.E. CLK	12	
t_{Ih}	39	Input hold time RXDn (asynchronous mode)	After R.E. CLK	0	
t_{CLKX}	40	CKXn input period (synchronous mode)		200	
t_{RXS}	40	RDXn setup time (synchronous mode)	Before F.E. CKX in synchronous mode	4	
t_{RXH}	40	RDXn hold time (synchronous mode)	After F.E. CKX in synchronous mode	2	
MICROWIRE / SPI Input Signals					
t_{MSKh}	42	MICROWIRE Clock High	At 2.0V (both edges)	80	
t_{MSKl}	42	MICROWIRE Clock Low	At 0.8V (both edges)	80	
t_{MSKp}	42	MICROWIRE Clock Period	MnIDL bit = 0; R.E. MSK to next R.E. MSK	200	
	43		MIDL bit = 1; F.E. MSK to next F.E. MSK		
t_{MSKh}	42	MSK Hold (slave only)	After $\overline{MCS}$ becomes inactive	40	
t_{MSKs}	42	MSK Setup (slave only)	Before $\overline{MCS}$ becomes active	80	
t_{MCSh}	42	$\overline{MCS}$ Hold (slave only)	MIDL bit = 0: After F.E. MSK	40	
	43		MIDL bit = 1: After R.E. MSK		
t_{MCSs}	42	$\overline{MCS}$ Setup (slave only)	MIDL bit = 0: Before R.E. MSK	80	
	43		MIDL bit = 1: Before F.E. MSK		
t_{MDIh}	42	MICROWIRE Data In Hold (master)	Normal Mode: After R.E. MSK	0	
	44		Alternate Mode: After F.E. MSK		
	42	MICROWIRE Data In Hold (slave)	Normal Mode: After R.E. MSK	40	
	44		Alternate Mode: After F.E. MSK		
t_{MDIs}	42	MICROWIRE Data In Setup	Normal Mode: Before R.E. MSK	80	
	44		Alternate Mode: Before F.E. MSK		
Multi-Function Timer Input Signals					
tTAH	47	TnA High Time	R.E. CLK	T_{CLK}+5	
tTAL	47	TnA Low Time	R.E. CLK	T_{CLK}+5	
tTBH	47	TnB High Time	R.E. CLK	T_{CLK}+5	
tTBL	47	TnB Low Time	R.E. CLK	T_{CLK}+5	

a. Only when operating with an external square wave on X2CKI; otherwise a 32 kHz crystal network must be used between X2CKI and X2CKO. If the slow clock is internally generated from the fast clock, it may not exceed this given limit.

22.0 Appendix

22.1 8-BIT MICROWIRE/SPI (MWSPI)

22.1.1 MWSPI Problem Description

According to the specification, the MSKn clock output in master mode should have the value of the MnIDL bit of the MUnCTL1 register, even when the module is disabled. However, the MSKn pin will always be at low level, when the alternate function of the MSKn pin is enabled and the module is disabled. Thus, even if the MnIDL bit is set, the MSKn clock will change to a low level as soon as the module is disabled. If any slave is selected at this time, it will interpret this unwanted transition as a shift clock.

22.1.2 MWSPI Problem Cause

Even if the module is disabled and the alternate function of the MSKn pin is enabled, the module can still influence the MSKn pin and drives the default value ‘0’.

22.1.3 MWSPI Problem Solutions

When the MSKn idle level of ‘1’ is to be used, the following procedure should be followed when the module is disabled:

1. Set the MSKn pin to high level in the corresponding port data output register.
2. Configure the MSKn pin to an output in the corresponding port direction register.
3. Disable the alternate function of the MSKn pin in the corresponding port alternate function register.
4. Disable the MWSPI module.

22.2 TIMING AND WATCHDOG MODULE

22.2.1 Timing and WATCHDOG Module Problem Description

The available window for a valid WATCHDOG service varies with the TWM configuration and the operating mode of the R16MHS9. Therefore it is not possible to generally provide the limits for the maximum service window. However, the limits for the minimum service window is guaranteed and should be used.

22.2.2 Timing and WATCHDOG Module Problem Cause

The timing and WATCHDOG module uses two different clock signals for its operation, the slow system clock as well as the fast system clock.

The slow system clock can either be generated by an external 32 kHz quartz or it can be derived from the fast system clock by means of a prescaler counter in the CLK2RES modules. The TWM can operate off a maximum slow system clock of 100 kHz. The WATCHDOG counter (down-counter) is either clocked directly by the slow system (T0IN) or it is decremented every time the counter T0 underflows (T0OUT).

The fast system clock is used for accesses to TWM registers, which build the user interface of the TWM. These user interface registers include all memory-mapped registers of the TWM.

Every time the user (CR16B core) writes to a TWM configuration register or to the WATCHDOG Service Data Match register, this “high speed operation” must be synchronized to the internal TWM logic running at the slow clock rate. This synchronization process takes a variable number of low speed clock cycles, depending on the ratio between the low-speed and the high-speed system clock and the phase shift between the two clock signals. The more the two frequencies differ from each other, the longer it takes the synchronization process.

In other words, write operations to the TWM registers take a certain number of low-speed clock cycles to show the desired effects to the TWM logic.

This fact is especially critical for the write operation for the WATCHDOG service, as it affects the allowed window for a valid WATCHDOG service.

If the device runs in active mode, the synchronization process can take up to four WATCHDOG counter clock cycles. This limits the available WATCHDOG service to the window shown in Figure48:

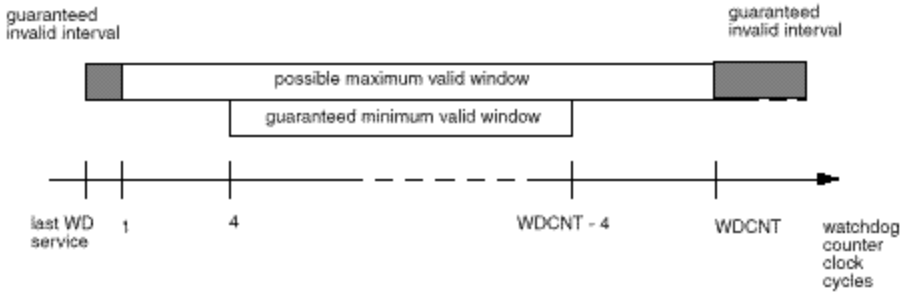


Figure 48. WATCHDOG Services Windows in Active Mode

If the device runs in power save mode, the synchronization process can take up to eight WATCHDOG counter clock cycles. This limits the available WATCHDOG service to the window shown in Figure49:

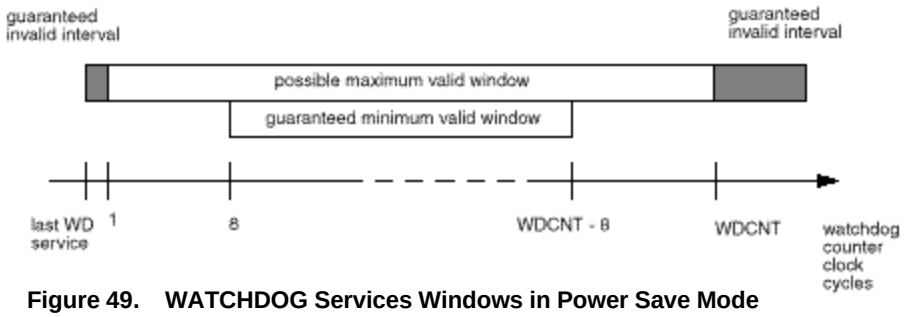
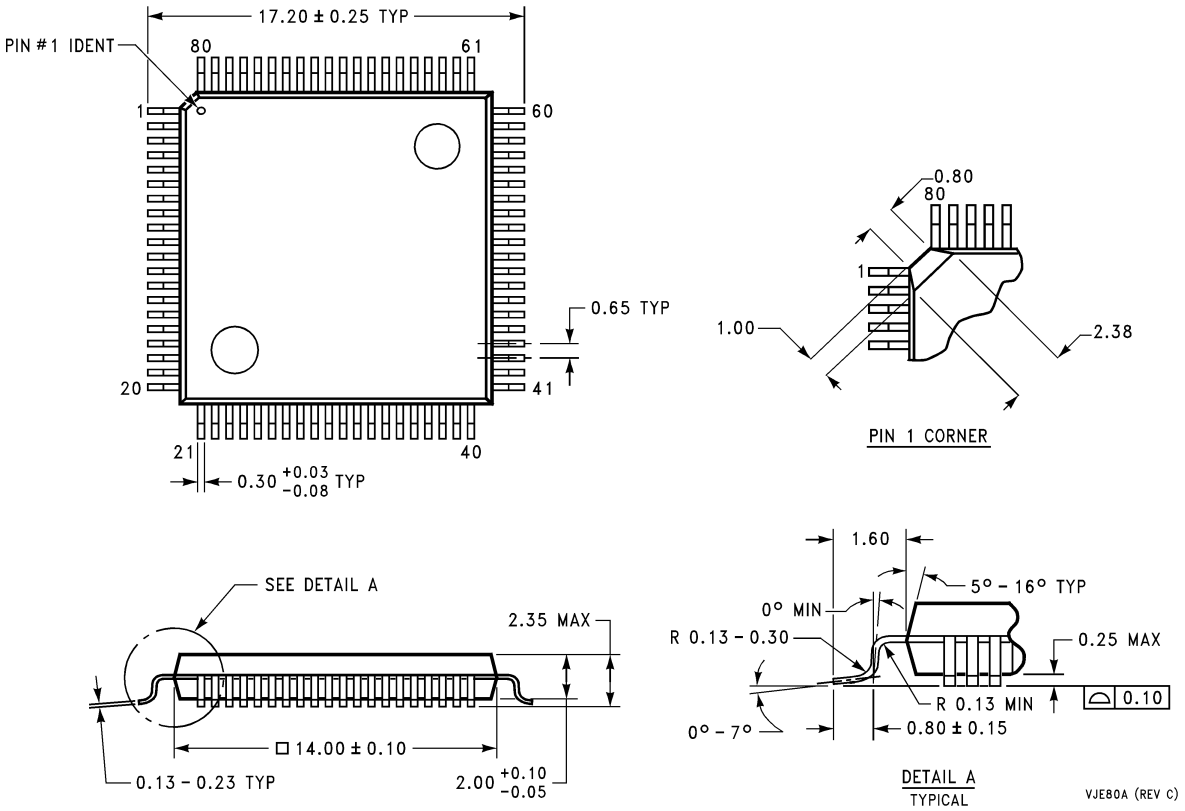


Figure 49. WATCHDOG Services Windows in Power Save Mode

22.2.3 Timing and WATCHDOG Module Problem Solutions

In order to guarantee a valid WATCHDOG service under all circumstances, the WATCHDOG should only be serviced within the guaranteed minimum valid window, as illustrated in figure 48 and figure 49 in the previous section.

23.0 Physical Dimension inches (millimeters) unless otherwise noted



80 Lead Molded Plastic Quad Flat Package
See NS Package Number VJE80A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Francais Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-254-4466
Fax: 65-250-4466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

www.national.com

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied, and National reserves the right, at any time without notice, to change said circuitry or specifications.