## Features

- Incorporates the ARM7TDMI® ARM® Thumb® Processor
  - High-performance 32-bit RISC Architecture
  - High-density 16-bit Instruction Set
  - Leader in MIPS/Watt
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Internal High-speed Flash
  - 128 Kbytes (AT91SAM7L128), Organized in 512 Pages of 256 Bytes Single Plane
  - 64 Kbytes (AT91SAM7L64), Organized In 256 Pages of 256 Bytes Single Plane
  - Single Cycle Access at Up to 15 MHz in Worst Case Conditions
  - 128-bit Read Access
  - Page Programming Time: 4.6 ms, Including Page Auto Erase, Full Erase Time: 10 ms
  - 10,000 Write Cycles, 10-year Data Retention Capability, Sector Lock Capabilities, Flash Security Bit
  - Fast Flash Programming Interface for High Volume Production
- Internal High-speed SRAM, Single-cycle Access at Maximum Speed
  - 6 Kbytes
    - 2 Kbytes Directly on Main Supply That Can Be Used as Backup SRAM
    - 4 Kbytes in the Core
- Memory Controller (MC)
  - Enhanced Embedded Flash Controller, Abort Status and Misalignment Detection
- Enhanced Embedded Flash Controller (EEFC)
  - Interface of the Flash Block with the 32-bit Internal Bus
  - Increases Performance in ARM and Thumb Mode with 128-bit Wide Memory Interface
- Reset Controller (RSTC)
  - Based on Zero-power Power-on Reset and Fully Programmble Brownout Detector
  - Provides External Reset Signal Shaping and Reset Source Status
- Clock Generator (CKGR)
  - Low-power 32 kHz RC Oscillator, 32 kHz On-chip Oscillator, 2 MHz Fast RC Oscillator and one PLL
- Supply Controller (SUPC)
  - Minimizes Device Power Consumption
  - Manages the Different Supplies On Chip
  - Supports Multiple Wake-up Sources
- Power Management Controller (PMC)
  - Software Power Optimization Capabilities, Including Active and Four Low Power Modes:
    - Idle Mode: No Processor Clock
    - Wait Mode: No Processor Clock, Voltage Regulator Output at Minimum
    - Backup Mode: Voltage Regulator and Processor Switched Off
    - Off (Power Down) Mode: Entire Chip Shut Down Except for Force Wake Up Pin (FWUP) that Re-activates the Device. 100 nA Current Consumption.
- In Active Mode, Dynamic Power Consumption <30 mA at 36 MHz
  - Three Programmable External Clock Signals
  - Handles Fast Start Up

# AT91 ARM Thumb-based Microcontroller

# AT91SAM7L128
# AT91SAM7L64

# Preliminary

6257A–ATARM–20-Feb-08

- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Two External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - Two-wire UART and Support for Debug Communication Channel interrupt, Programmable ICE Access Prevention
- Periodic Interval Timer (PIT)
  - 20-bit Programmable Counter plus 12-bit Interval Counter
- Windowed Watchdog (WDT)
  - 12-bit Key-protected Programmable Counter
  - Provides Reset or Interrupt Signals to the System
  - Counter may be Stopped While the Processor is in Debug State or in Idle Mode
- Real-time Clock (RTC)
  - Two Hundred Year Calendar with Alarm
  - Runs Off the Internal RC or Crystal Oscillator
- Three Parallel Input/Output Controllers (PIOA, PIOB, PIOC)
  - Eighty Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up resistor and Synchronous Output
- Eleven Peripheral DMA Controller (PDC) Channels
- One Segment LCD Controller
  - Display Capacity of Forty Segments and Ten Common Terminals
  - Software Selectable LCD Output Voltage (Contrast)
- Two Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation
  - Support for ISO7816 T0/T1 Smart Card, Hardware Handshaking, RS485 Support
  - Manchester Encoder/Decoder
  - Full Modem Line Support on USART1
- One Master/Slave Serial Peripheral Interface (SPI)
  - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
- One Three-channel 16-bit Timer/Counter (TC)
  - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- One Four-channel 16-bit PWM Controller (PWMC)
- One Two-wire Interface (TWI)
  - Master, Multi-Master and Slave Mode Support, All Atmel® Two-wire EEPROMs and $I^2C$ compatible Devices Supported
  - General Call Supported in Slave Mode
- One 4-channel 10-bit Analog-to-Digital Converter, Four Channels Multiplexed with Digital I/Os
- SAM-BA® Boot Assistant
  - Default Boot Program
  - Interface with SAM-BA Graphic User Interface
  - In Application Programming Function (IAP)
- IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins
- Four High-current Drive I/O lines, Up to 4 mA Each
- Power Supplies
  - Embedded 1.8V Regulator, Drawing up to 60 mA for the Core with Programmable Output Voltage
  - Single Supply 1.8V - 3.6V
- Fully Static Operation: Up to 36 MHz at 85°C, Worst Case Conditions
- Available in a 128-lead LQFP Green and a 144-ball LFBGA Green Package

# 1. Description

The AT91SAM7L128/64 are low power members of Atmel's Smart ARM Microcontroller family based on the 32-bit ARM7™ RISC processor and high-speed Flash memory.

- AT91SAM7L128 features a 128 Kbyte high-speed Flash and a total of 6 Kbytes SRAM.
- AT91SAM7L64 features a 64 Kbyte high-speed Flash and a total of 6 Kbytes SRAM.

They also embed a large set of peripherals, including a Segment LCD Controller and a complete set of system functions minimizing the number of external components.

These devices provide an ideal migration path for 8-bit microcontroller users looking for additional performance, extended memory and higher levels of system integration with strong constraints on power consumption.

Featuring innovative power reduction modes and ultra-low-power operation, the AT91SAM7L128/64 is tailored for battery operated applications such as calculators, toys, remote controls, medical devices, mobile phone accessories and wireless sensors.

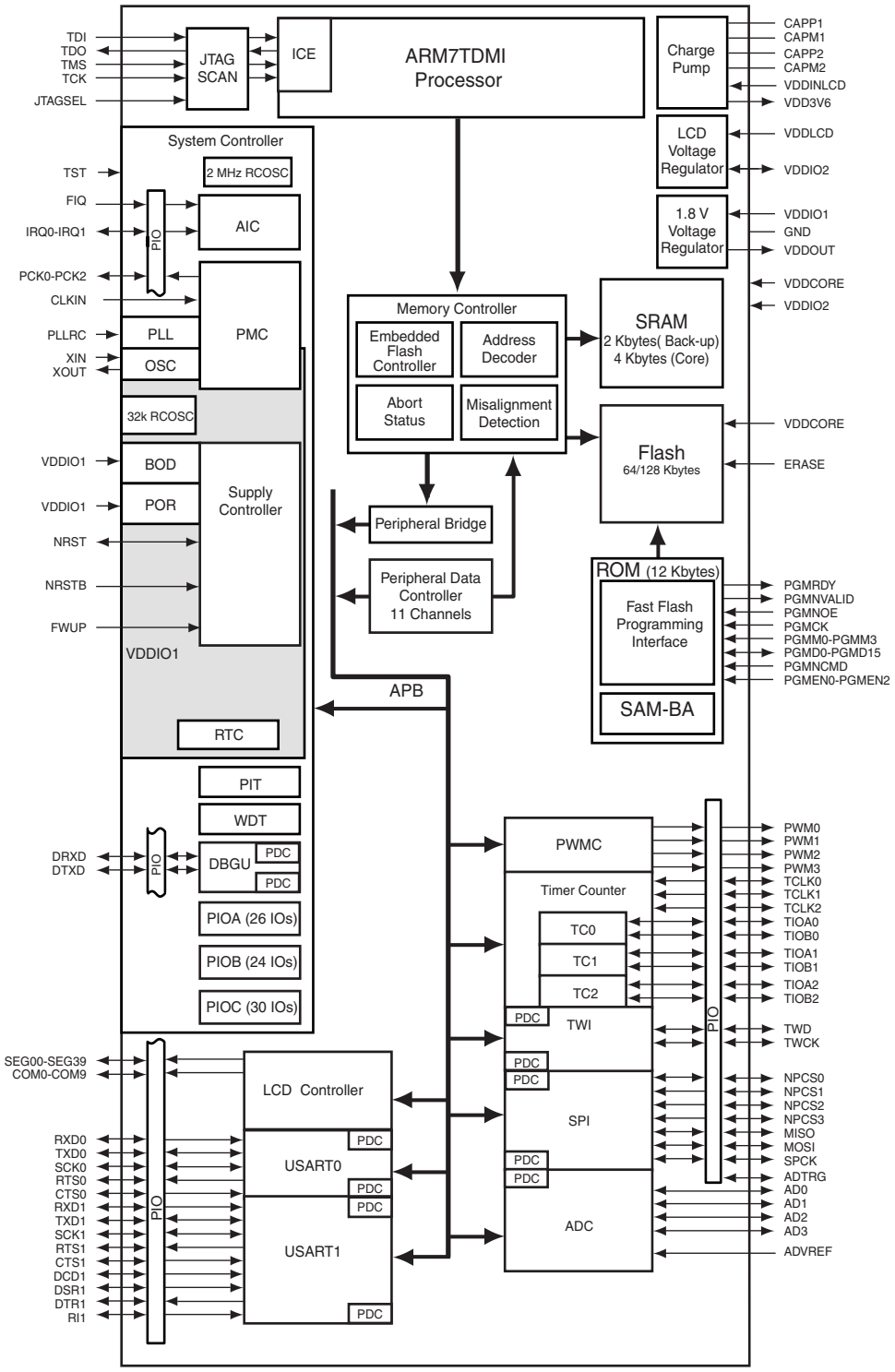
The embedded Flash memory can be programmed in-system via the JTAG-ICE interface or via a parallel interface on a production programmer prior to mounting. Built-in lock bits and a security bit protect the firmware from accidental overwrite and preserve its confidentiality.

The AT91SAM7L128/64 system controller includes a reset controller capable of managing the power-on sequence of the microcontroller and the complete system. Correct device operation can be monitored by a built-in brownout detector and a watchdog running off an integrated oscillator.

By combining the ARM7TDMI processor with on-chip Flash and SRAM, and a wide range of peripheral functions, including USART, SPI, External Bus Timer Counter, RTC and Analog-to-Digital Converters on a monolithic chip, the AT91SAM7L128/64 microcontroller is a powerful device that provides a flexible, cost-effective solution to many embedded control applications.

# 2. Block Diagram

**Figure 2-1.** AT91SAM7L128/64 Block Diagram



# 3. Signal Description

**Table 3-1.** Signal Description List

| Signal Name | Function | Type | Active Level | Voltage Reference | Comments |
|---|---|---|---|---|---|
| **Power** | | | | | |
| VDDIO1 | I/O Lines (PIOC) and Voltage Regulator Power Supply | Power | | | From 1.80V to 3.6V |
| VDDOUT | Voltage Regulator Output | Power | | | |
| VDDCORE | Core Power Supply | Power | | | Connected externally to VDDOUT |
| VDDINLCD | Charge Pump Power Supply | Power | | | From 1.80V to 3.6V |
| VDD3V6 | Charge Pump Output | Power | | | |
| VDDLCD | LCD Voltage Regulator Power Supply | Power | | | |
| VDDIO2 | LCD Voltage Regulator Output and LCD I/O Lines Power Supply (PIOA and PIOB) | Power | | | 1.80V to 3.6V |
| CAPP1 | Charge pump capacitor 1 | Power | | | Capacitor needed between CAPP1 and CAPM1. |
| CAPM1 | Charge pump capacitor 1 | Power | | | |
| CAPP2 | Charge pump capacitor 2 | Power | | | Capacitor needed between CAPP2 and CAPM2. |
| CAPM2 | Charge pump capacitor 2 | Power | | | |
| FWUP | Force Wake-up | Input | Low | VDDIO1 | Needs external Pull-up. |
| WKUP0-15 | Wake-up inputs used in Backup mode and Fast Start-up inputs in Wait mode | Input | | VDDIO1 | |
| GND | Ground | Ground | | | |
| **Clocks, Oscillators and PLLs** | | | | | |
| XIN | 32 kHz Oscillator Input | Input | | VDDIO1 | |
| XOUT | 32 kHz Oscillator Output | Output | | VDDIO1 | |
| CLKIN | Main Clock input | Input | | VDDIO1 | Should be tied low when not used. |
| PCK0 - PCK2 | Programmable Clock Output | Output | | | |
| PLLRC | PLL Filter | Input | | VDDCORE | |
| PLLRCGND | PLL RC Filter Ground | Power | | | Must not be connected to external Ground. |
| **ICE and JTAG** | | | | | |
| TCK | Test Clock | Input | | VDDIO1 | No internal pull-up resistor |
| TDI | Test Data In | Input | | VDDIO1 | No internal pull-up resistor |
| TDO | Test Data Out | Output | | VDDIO1 | |
| TMS | Test Mode Select | Input | | VDDIO1 | No internal pull-up resistor |
| JTAGSEL | JTAG Selection | Input | | VDDIO1 | Internal Pull-down resistor |
| **Flash Memory** | | | | | |
| ERASE | Flash and NVM Configuration Bits Erase Command | Input | High | VDDIO1 | Internal Pull-down (15 kΩ) resistor |

**Table 3-1.** Signal Description List (Continued)

| Signal Name | Function | Type | Active Level | Voltage Reference | Comments |
|---|---|---|---|---|---|
| **Reset/Test** | | | | | |
| NRST | Microcontroller Reset | I/O | Low | VDDIO1 | Internal Pull-up (100 kΩ) resistor |
| TST | Test Mode Select | Input | High | VDDIO1 | Internal Pull-down (15 kΩ) resistor |
| NRSTB | Asynchronous Master Reset | Input | Low | VDDIO1 | Internal Pull-up (15 kΩ) resistor |
| **Debug Unit** | | | | | |
| DRXD | Debug Receive Data | Input | | | |
| DTXD | Debug Transmit Data | Output | | | |
| **AIC** | | | | | |
| IRQ0 - IRQ1 | External Interrupt Inputs | Input | | | |
| FIQ | Fast Interrupt Input | Input | | | |
| **PIO** | | | | | |
| PA0 - PA25 | Parallel IO Controller A | I/O | | VDDIO2 | Pulled-up input at reset |
| PB0 - PB23 | Parallel IO Controller B | I/O | | VDDIO2 | Pulled-up input at reset |
| PC0 - PC29 | Parallel IO Controller C | I/O | | VDDIO1 | Pulled-up input at reset |
| **USART** | | | | | |
| SCK0 - SCK1 | Serial Clock | I/O | | | |
| TXD0 - TXD1 | Transmit Data | I/O | | | |
| RXD0 - RXD1 | Receive Data | Input | | | |
| RTS0 - RTS1 | Request To Send | Output | | | |
| CTS0 - CTS1 | Clear To Send | Input | | | |
| DCD1 | Data Carrier Detect | Input | | | |
| DTR1 | Data Terminal Ready | Output | | | |
| DSR1 | Data Set Ready | Input | | | |
| RI1 | Ring Indicator | Input | | | |
| **Timer/Counter** | | | | | |
| TCLK0 - TCLK2 | External Clock Inputs | Input | | | |
| TIOA0 - TIOA2 | Timer Counter I/O Line A | I/O | | | |
| TIOB0 - TIOB2 | Timer Counter I/O Line B | I/O | | | |
| **PWM Controller** | | | | | |
| PWM0 - PWM3 | PWM Channels | Output | | | |
| **Serial Peripheral Interface** | | | | | |
| MISO | Master In Slave Out | I/O | | | |
| MOSI | Master Out Slave In | I/O | | | |
| SPCK | SPI Serial Clock | I/O | | | |
| NPCS0 | SPI Peripheral Chip Select 0 | I/O | Low | | |
| NPCS1-NPCS3 | SPI Peripheral Chip Select 1 to 3 | Output | Low | | |

**Table 3-1.** Signal Description List (Continued)

| Signal Name | Function | Type | Active Level | Voltage Reference | Comments |
|---|---|---|---|---|---|
| **Two-Wire Interface** | | | | | |
| TWD | Two-wire Serial Data | I/O | | | |
| TWCK | Two-wire Serial Clock | I/O | | | |
| **Analog-to-Digital Converter** | | | | | |
| AD0-AD3 | Analog Inputs | Input | | VDDCORE | |
| ADTRG | ADC Trigger | Input | | | |
| ADVREF | ADC Reference | Analog | | VDDCORE | |
| **Fast Flash Programming Interface** | | | | | |
| PGMEN0-PGMEN2 | Programming Enabling | Input | | VDDIO1 | |
| PGMM0-PGMM3 | Programming Mode | Input | | VDDIO1 | |
| PGMD0-PGMD15 | Programming Data | I/O | | VDDIO1 | |
| PGMRDY | Programming Ready | Output | High | VDDIO1 | |
| PGMNVALID | Data Direction | Output | Low | VDDIO1 | |
| PGMNOE | Programming Read | Input | Low | VDDIO1 | |
| PGMCK | Programming Clock | Input | | VDDIO1 | |
| PGMNCMD | Programming Command | Input | Low | VDDIO1 | |
| **Segmented LCD Controller** | | | | | |
| COM[9:0] | Common Terminals | Output | | VDDIO2 | |
| SEG[39:0] | Segment Terminals | Output | | VDDIO2 | |

# 4. Package and Pinout

The AT91SAM7L128/64 is available in:

- 20 x 14 mm 128-lead LQFP package with a 0.5 mm lead-pitch
- 10 x 10 mm 144-ball LFBGA package with a 0.8 mm pitch.

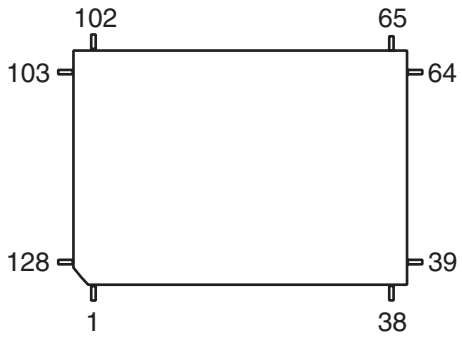
The part is also available in die delivery.

## 4.1 128-lead LQFP Package Outline

Figure 4-1 shows the orientation of the 128-lead LQFP package.

A detailed mechanical description is given in the Mechanical Characteristics section of the product datasheet.

**Figure 4-1.** 128-lead LQFP Package Outline (Top View)



## 4.2 128-lead LQFP Package Pinout

**Table 4-1.** Pinout for 128-lead LQFP Package

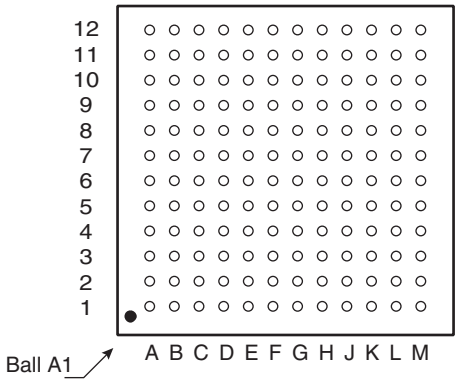
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | TST | 33 | VDDLCD | 65 | PB21 | 97 | PC10/PGMM3 |
| 2 | VDDCORE | 34 | VDD3V6 | 66 | PB22 | 98 | PC11/PGMD0 |
| 3 | PA0 | 35 | CAPM2 | 67 | PB23 | 99 | PC12/PGMD1 |
| 4 | PA1 | 36 | CAPP2 | 68 | GND | 100 | VDDCORE |
| 5 | PA2 | 37 | CAPM1 | 69 | ADVREF | 101 | PC13/PGMD2 |
| 6 | PA3 | 38 | CAPP1 | 70 | AD3 | 102 | PC14/PGMD3 |
| 7 | PA4 | 39 | VDDINLCD | 71 | AD2 | 103 | PC15/PGMD4 |
| 8 | PA5 | 40 | GND | 72 | AD1 | 104 | PC16/PGMD5 |
| 9 | PA6 | 41 | PB0 | 73 | AD0 | 105 | PC17/PGMD6 |
| 10 | PA7 | 42 | PB1 | 74 | VDDOUT | 106 | PC18/PGMD7 |
| 11 | PA8 | 43 | PB2 | 75 | VDDIO1 | 107 | PC19/PGMD8 |
| 12 | PA9 | 44 | PB3 | 76 | GND | 108 | PC20/PGMD9 |
| 13 | PA10 | 45 | PB4 | 77 | PC28 | 109 | PC21/PGMD10 |
| 14 | GND | 46 | PB5 | 78 | PC29 | 110 | PC22/PGMD11 |
| 15 | VDDIO2 | 47 | PB6 | 79 | NRST | 111 | PC23/PGMD12 |
| 16 | PA11 | 48 | PB7 | 80 | ERASE | 112 | PC24/PGMD13 |
| 17 | PA12 | 49 | PB8 | 81 | TCK | 113 | PC25/PGMD14 |
| 18 | PA13 | 50 | PB9 | 82 | TMS | 114 | PC26/PGMD15 |
| 19 | PA14 | 51 | PB10 | 83 | JTAGSEL | 115 | PC27 |
| 20 | PA15 | 52 | PB11 | 84 | VDDCORE | 116 | TDI |
| 21 | PA16 | 53 | PB12 | 85 | VDDIO1 | 117 | TDO |
| 22 | PA17 | 54 | PB13 | 86 | GND | 118 | FWUP |
| 23 | PA18 | 55 | VDDIO2 | 87 | PC0/PGMEN0 | 119 | VDDIO1 |
| 24 | PA19 | 56 | GND | 88 | PC1/PGMEN1 | 120 | GND |
| 25 | PA20 | 57 | PB14 | 89 | PC2/PGMEN2 | 121 | PLLRC |
| 26 | PA21 | 58 | PB15 | 90 | PC3/PGMNCMD | 122 | PLLRCGND |
| 27 | PA22 | 59 | PB16 | 91 | PC4/PGMRDY | 123 | GND |
| 28 | VDDCORE | 60 | PB17 | 92 | PC5/PGMNOE | 124 | VDDCORE |
| 29 | PA23 | 61 | PB18 | 93 | PC6/PGMNVALID | 125 | CLKIN |
| 30 | PA24 | 62 | VDDCORE | 94 | PC7/PGMM0 | 126 | NRSTB |
| 31 | PA25 | 63 | PB19 | 95 | PC8/PGMM1 | 127 | XIN/PGMCK |
| 32 | VDDIO2 | 64 | PB20 | 96 | PC9/PGMM2 | 128 | XOUT |

## 4.3 144-ball LFBGA Package Outline

Figure 4-2 shows the orientation of the 144-ball LFBGA package.

A detailed mechanical description is given in the Mechanical Characteristics section of the product datasheet.

**Figure 4-2.** 144-ball LFBGA Package Outline (Top View)



## 4.4 144-ball LFBGA Pinout

**Table 4-2.** SAM7L128/64 Pinout for 144-ball LFBGA Package

| Pin | Signal Name | Pin | Signal Name | Pin | Signal Name | Pin | Signal Name |
|---|---|---|---|---|---|---|---|
| A1 | XOUT | D1 | PA6 | G1 | VDD3V6 | K1 | CAPM1 |
| A2 | XIN | D2 | PA5 | G2 | PA17 | K2 | VDDIO2 |
| A3 | VDDCORE | D3 | PA7 | G3 | PA16 | K3 | VDDIO2 |
| A4 | GND | D4 | NC | G4 | PA15 | K4 | PA25 |
| A5 | PLLRCGND | D5 | PC26/PGMD15 | G5 | GND | K5 | PB3 |
| A6 | PLLRC | D6 | PC25/PGMD14 | G6 | GND | K6 | PB10 |
| A7 | PC24/PGMD13 | D7 | PC21/PGMD11 | G7 | GND | K7 | PB13 |
| A8 | PC23//PGMD12 | D8 | PC18/PGMD7 | G8 | VDDIO1 | K8 | PB15 |
| A9 | PC17/PGMD6 | D9 | PC6/PGMNVALID | G9 | NRST | K9 | PB20 |
| A10 | NC | D10 | PC7/PGMM0 | G10 | TMS | K10 | VDDCORE |
| A11 | PC14 | D11 | PC4/PGMRDY | G11 | ERASE | K11 | VDDCORE |
| A12 | PC12 | D12 | PC3/PGMNCMD | G12 | VDDOUT | K12 | AD2 |
| B1 | PA1 | E1 | VDDIO2 | H1 | CAPM2 | L1 | CAPP1 |
| B2 | PA0 | E2 | PA10 | H2 | PA22 | L2 | VDDIO2 |
| B3 | NRSTB | E3 | PA9 | H3 | PA19 | L3 | VDDIO2 |
| B4 | TEST | E4 | PA11 | H4 | PA18 | L4 | PB4 |
| B5 | TDO | E5 | PA8 | H5 | GND | L5 | PB5 |
| B6 | PC27 | E6 | VDDIO1 | H6 | GND | L6 | PB11 |
| B7 | GND | E7 | VDDIO1 | H7 | GND | L7 | PB12 |
| B8 | NC | E8 | VDDIO1 | H8 | VDDCORE | L8 | PB17 |
| B9 | PC20/PGMD9 | E9 | PC5/PGMNOE | H9 | PC29 | L9 | PB19 |
| B10 | PC15/PGMD4 | E10 | PC0/PGMEN0 | H10 | VDDCORE | L10 | PB22 |
| B11 | PC13/PGMD2 | E11 | PC2/PGMEN2 | H11 | PC28 | L11 | PB23 |
| B12 | PC11/PGMD0 | E12 | VDDCORE | H12 | AD0 | L12 | AD3 |
| C1 | PA3 | F1 | VDDLCD | J1 | CAPP2 | M1 | VDDINLCD |
| C2 | PA4 | F2 | PA13 | J2 | PA23 | M2 | PB0 |
| C3 | PA2 | F3 | PA14 | J3 | PA24 | M3 | PB1 |
| C4 | CLKIN | F4 | PA12 | J4 | PA21 | M4 | PB2 |
| C5 | FWUP | F5 | GND | J5 | PA20 | M5 | PB6 |
| C6 | TDI | F6 | GND | J6 | PB8 | M6 | PB7 |
| C7 | PC22/PGMD11 | F7 | GND | J7 | PB9 | M7 | VDDIO2 |
| C8 | PC19/PGMD8 | F8 | VDDIO1 | J8 | PB14 | M8 | PB16 |
| C9 | PC16/PGMD5 | F9 | TCK | J9 | VDDCORE | M9 | PB18 |
| C10 | PC9/PGMM2 | F10 | JTAGSEL | J10 | VDDCORE | M10 | PB21 |
| C11 | PC10/PGMM3 | F11 | PC1/PGMEN1 | J11 | VDDCORE | M11 | GND |
| C12 | PC8/PGMM1 | F12 | VDDIO1 | J12 | AD1 | M12 | ADVREF |

# 5. Power Considerations

## 5.1 Power Supplies

The AT91SAM7L128/64 has six types of power supply pins and integrates a voltage regulator, allowing the device to be supplied with only one voltage. The six power supply pin types are:

- VDDOUT pin. It is the output of the voltage regulator. Output voltage can be programmed from 1.55V to 1.80V by steps of 100 mV.
- VDDIO1 pin. It powers the voltage regulator input and all the PIOC IO lines (1.8V-3.6V). VDDIO1 voltage must be above 2.2V to allow the chip to start-up (POR threshold).
- VDDIO2 pin. It powers the PIOA and PIOB I/O lines (1.8V-3.6V). It is also the output of the LCD voltage regulator. The output voltage can be programmed from 2.4V to 3.4V with 16 steps.
- VDDCORE pin. It powers the logic of the device, the PLL, the 2 MHz Fast RC oscillator, the ADC and the Flash memory. It must be connected to the VDDOUT pin with a decoupling capacitor.
- VDDINLCD pin. It powers the charge pump which can be used as LCD Regulator power supply. Voltage ranges from 1.8V to 3.6V.

No separate ground pins are provided for the different power supplies. Only GND pins are provided and should be connected as shortly as possible to the system ground plane.

## 5.2 Low Power Modes

The various low power modes of the AT91SAM7L128/64 are described below.

### 5.2.1 Off (Power Down) Mode

In off (power down) mode, the entire chip is shut down. Only a low level on the FWUP pin can wake up the AT91SAM7L128/64 (by a push-button for example). Internally, except for the FWUP pin through VDDIO1, none of the chip is supplied.

Once the internal main power switch has been activated by FWUP, the 32 kHz RC oscillator and the Supply Controller are supplied, then the core and peripherals are reset and the AT91SAM7L128/64 enters in active mode. Refer to the System Controller Block Diagram, Figure 9-1 on page 30.

At first power-up, if FWUP is tied high, the device enters off mode. The PIOA and PIOB pins' states are undefined. PIOC and NRST pins are initialized as high impedance inputs.

Once the device enters active mode, the core and the parallel input/output controller are reset. Then, if the chip enters off mode, PIOA and PIOB pins are configured as inputs with pull-ups and PIOC pins as high impedance inputs.

Current consumption in this mode is typically 100 nA.

### 5.2.2 Backup Mode

In backup mode, the supply controller, the zero-power power-on reset and the 32 kHz oscillator (software selectable internal RC or external crystal) remain running. The voltage regulator and the core are switched off.

Prior to entering this mode, the RTC, the backup SRAM, the brownout detector, the charge pump, the LCD voltage regulator and the LCD controller can be set on or off separately.

Table 5-1 on page 13 shows an example of backup mode with backup SRAM and RTC running.

When entering this mode, all PIO pins keep their previous states, they are reinitialized as inputs with pull-ups at wake-up.

The AT91SAM7L128/64 can be awakened from this mode through the FWUP pin, an event on WUP0-15 pins, or an RTC alarm or brownout event.

Current consumption is 3.5 µA typical without the LCD controller running.

### 5.2.3 Wait Mode

In wait mode, the voltage regulator must be set in deep mode. Voltage regulator output voltage should be set at a minimum voltage to decrease leakage in the digital core. No clock is running in the core. From this mode, a fast start-up is available (refer to Section 5.4 "Fast Start-Up").

In this mode, all PIO pins keep their previous states.

### 5.2.4 Idle Mode

The processor is in idle mode which means that the processor has no clock but the Master clock (MCK) remains running. The processor can also be wakened by an IRQ or FIQ.

### 5.2.5 Active Mode

The total dynamic power consumption is less than 30 mA at full speed (36 MHz) when running out of the Flash. The power management controller can be used to adapt the frequency and the regulator output voltage can be adjusted to optimize power consumption.

### 5.2.6 Low Power Mode Summary Table

The modes detailed above are the main modes. In off mode, no options are available but once the shutdown controller is set to on, each part can be set to on, or off, separately and more modes can be active. The table below shows a summary of the configurations of the low power modes.

**Table 5-1.** Low Power Mode Configuration Summary

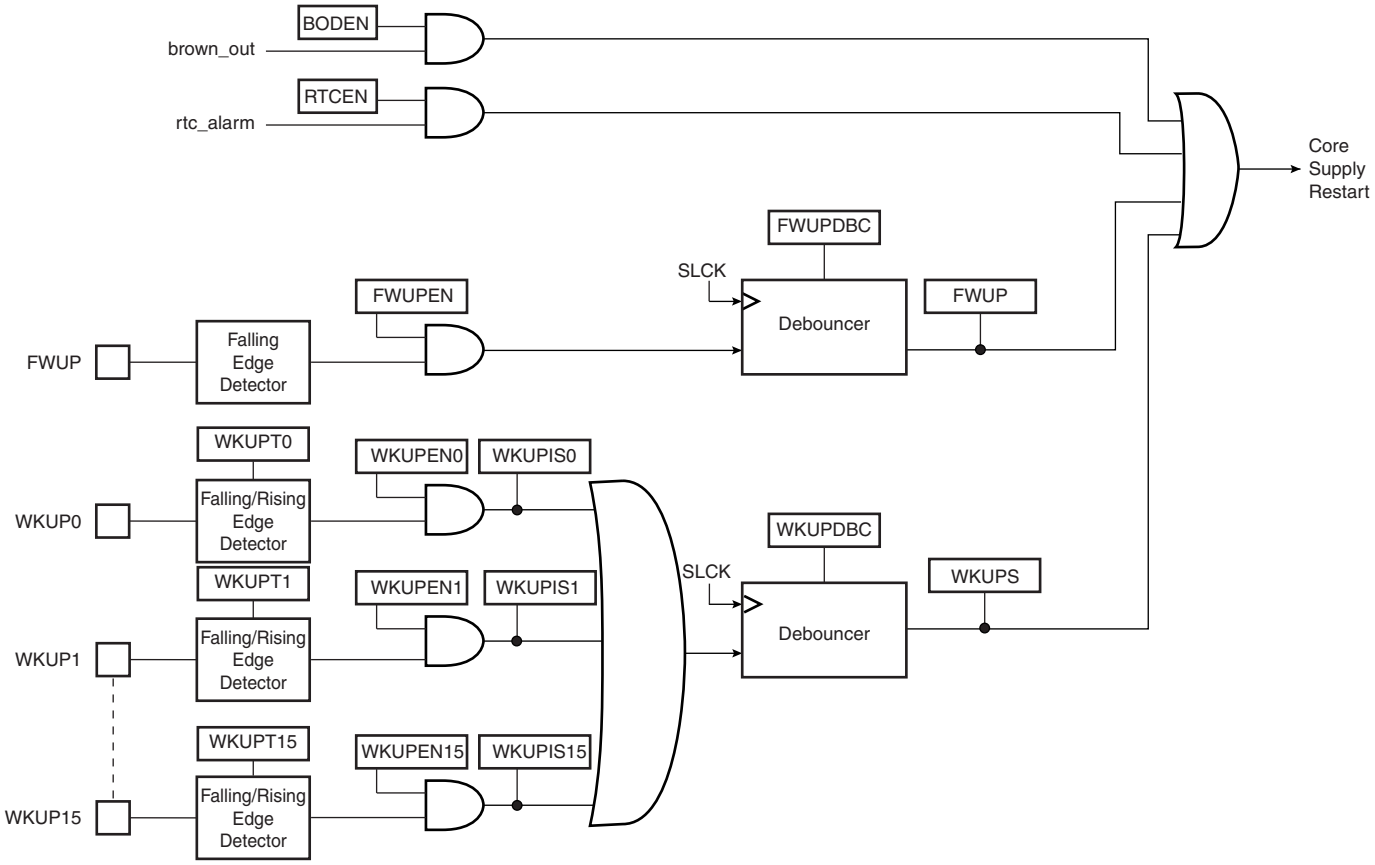
| Mode | FWUP | SUPC, 32 kHz Oscillator, POR | RTC | Backup SRAM | Regulator (Deep Mode) | Core | Potential Wake-up Sources | Consumption[2][3] | Wake-up Time[1] |
|---|---|---|---|---|---|---|---|---|---|
| Off Mode | X | | | | | | FWUP pin | 100 nA typ | < 5 ms |
| Backup Mode (with SRAM and RTC) | X | X | X | X | | | FWUP pin<br>WUP0-15 pins<br>BOD alarm<br>RTC alarm | 3.5 µA typ | < 0.5 ms |
| Wait Mode (with SRAM and RTC) | X | X | X | X | X | X | Fast start-up through WUP0-15 pins | 9 µA typ | < 2 µs (in case of fast start-up) |
| Idle Mode | X | X | | X | X | X | IRQs<br>FIQ | (4) | (4) |

Notes:
1. When considering wake-up time, the time required to start the PLL is not taken into account. Once started, the AT91SAM7128/L64 works with the 2 MHz Fast RC oscillator. The user has to add the PLL start-up time if it is needed in the system. The wake-up time is defined as the time taken for wake up until the first instruction is fetched.
2. The external LCD current consumption and the external loads on PIOs are not taken into account in the calculation.
3. BOD current consumption is not included.
4. Depends on MCK frequency.

## 5.3 Wake-up Sources

The wake-up events allow the device to exit from backup mode. When a wake-up event is detected, the supply controller performs a sequence which automatically reenables the voltage regulator and the backup SRAM power supply, if it is not already enabled.

**Figure 5-1.** Wake Up Sources

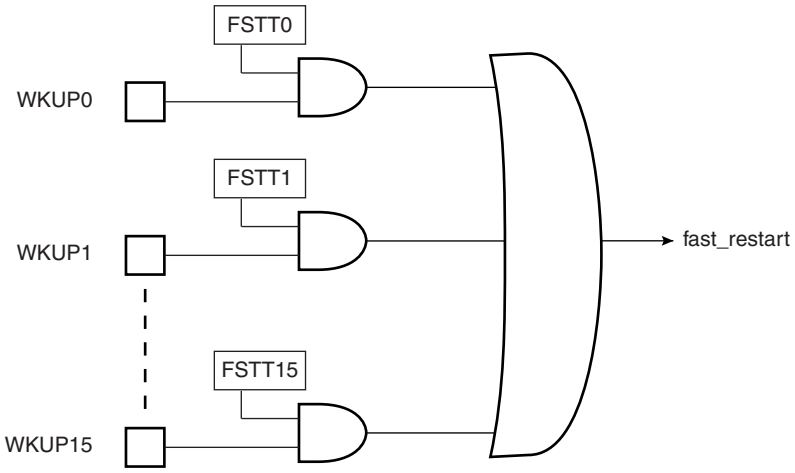



## 5.4 Fast Start-Up

The SAM7L128/64 allows the processor to restart in a few microseconds while the processor is in wait mode. A fast start up can occur upon detection of a low level on one of the 16 wake-up inputs.

The fast restart circuitry, as shown in Figure 5-2, is fully asynchronous and provides a fast start-up signal to the power management controller. As soon as the fast start-up signal is asserted, the PMC automatically restarts the embedded 2 MHz Fast RC oscillator, switches the master clock on this 2 MHz clock and reenables the processor clock, if it is disabled.

**Figure 5-2.** Fast Start-Up Circuitry




## 5.5 Voltage Regulator

The AT91SAM7L128/64 embeds a voltage regulator that is managed by the supply controller. This internal regulator is only intended to supply the internal core of AT91SAM7L128/64. It features three different operating modes:

- In normal mode, the voltage regulator consumes less than 30 µA static current and draws 60 mA of output current.
- In deep mode, the current consumption of the voltage regulator is less than 8.5 µA. It can draw up to 1 mA of output current. The default output voltage is 1.80V and the start-up time to reach normal mode is inferior to 400 µs.
- In shutdown mode, the voltage regulator consumes less than 1 µA while its output is driven internally to GND. The default output voltage is 1.80V and the start-up time to reach normal mode is inferior to 400 µs.

Furthermore, in normal and deep modes, the regulator output voltage can be programmed by software with 4 different steps within the range of 1.55V to 1.80V. The default output voltage is 1.80V in both normal and deep modes. The voltage regulator can regulate 1.80V output voltage as long as the input voltage is above 1.95V. Below 1.95V input voltage, the output voltage remains above 1.65V.

Output voltage adjusting ability allows current consumption reduction on VDDCORE and also enables programming a lower voltage when the input voltage is lower than 1.95V.

At 1.55V, the Flash is still functional but with slower read access time. Programming or erasing the Flash is not possible under these conditions. MCK maximum frequency is 25 MHz with VDDCORE at 1.55V (1.45V minimum).

The regulator has an indicator that can be used by the software to show that the output voltage has the correct value (output voltage has reached at least 80% of the typical voltage). This flag is used by the supply controller. This feature is only possible when the voltage regulator is in normal mode at 1.80V.

Adequate output supply decoupling is mandatory for VDDOUT in order to reduce ripple and avoid oscillations. One external 2.2 µF (or 3.3 µF) X7R capacitor must be connected between VDDOUT and GND.

Adequate input supply decoupling is mandatory for VDDIO1 in order to improve startup stability and reduce source voltage drop. The input decoupling capacitor should be placed close to the chip. For example, two capacitors can be used in parallel, 100 nF NPO and 4.7 µF X7R.

## 5.6 LCD Power Supply

The AT91SAM7L128/64 embeds an on-chip LCD power supply comprising a regulated charge pump and an adjustable voltage regulator.

The regulated charge pump output delivers 3.6V as long as its input is supplied between 1.8V and 3.6V. The regulated charge pump only requires two external flying capacitors and one external tank capacitor to operate.

Adequate input supply decoupling is mandatory for VDDINLCD in order to improve startup stability and reduce source voltage drop. The input decoupling capacitor should be placed close to the chip.

Current consumption of the charge pump and LCD bias when active is 350 µA (max case).

The regulated charge pump can be used to supply the LCD voltage regulator or as a 3.6V voltage reference delivering up to 4 mA.

The LCD voltage regulator output voltage is software selectable from 2.4V to 3.4V with 16 levels. Its input should be supplied in the range of 2.5 to 3.6V. The LCD voltage regulator can be supplied by the regulated charge pump output or by an external supply.

When the LCD voltage regulator is not used, its output must be connected to an external source in order to supply the PIOA and PIOB I/O lines.

Figure 5-3 below shows the typical schematics needed:

**Figure 5-3.** The Charge Pump Supplies the LCD Regulator

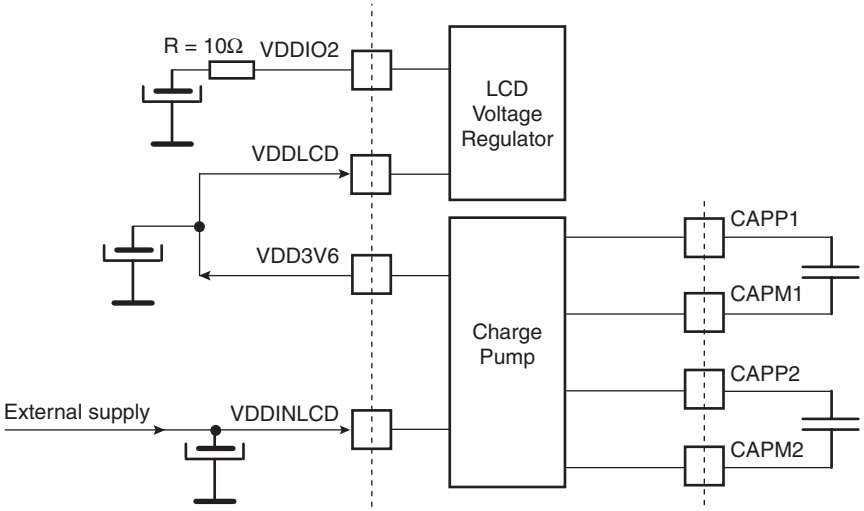
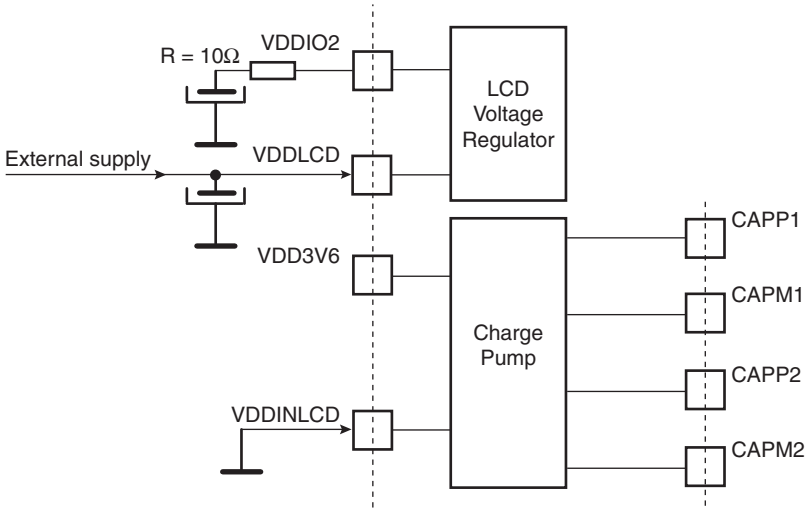
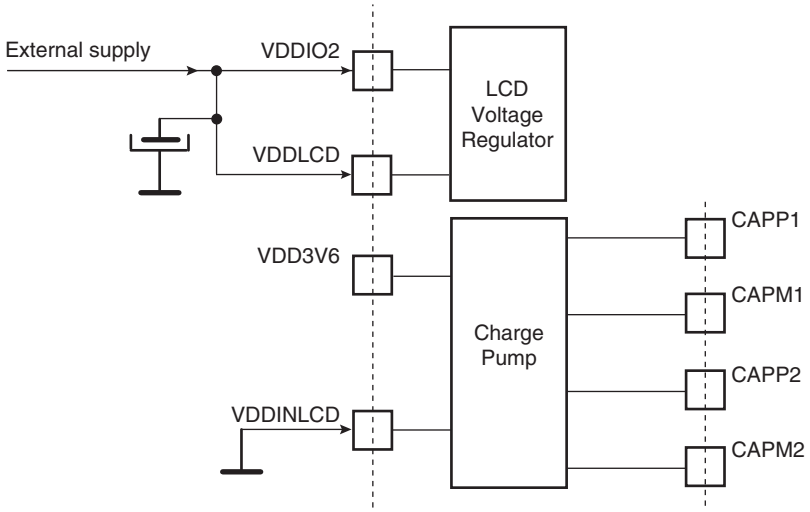


**Figure 5-4.** The LCD Regulator is Externally Supplied




If the charge pump is not needed, the user can apply an external voltage. See Figure 5-5 below:

**Figure 5-5.** The Charge Pump and the LCD Regulator are Not Used

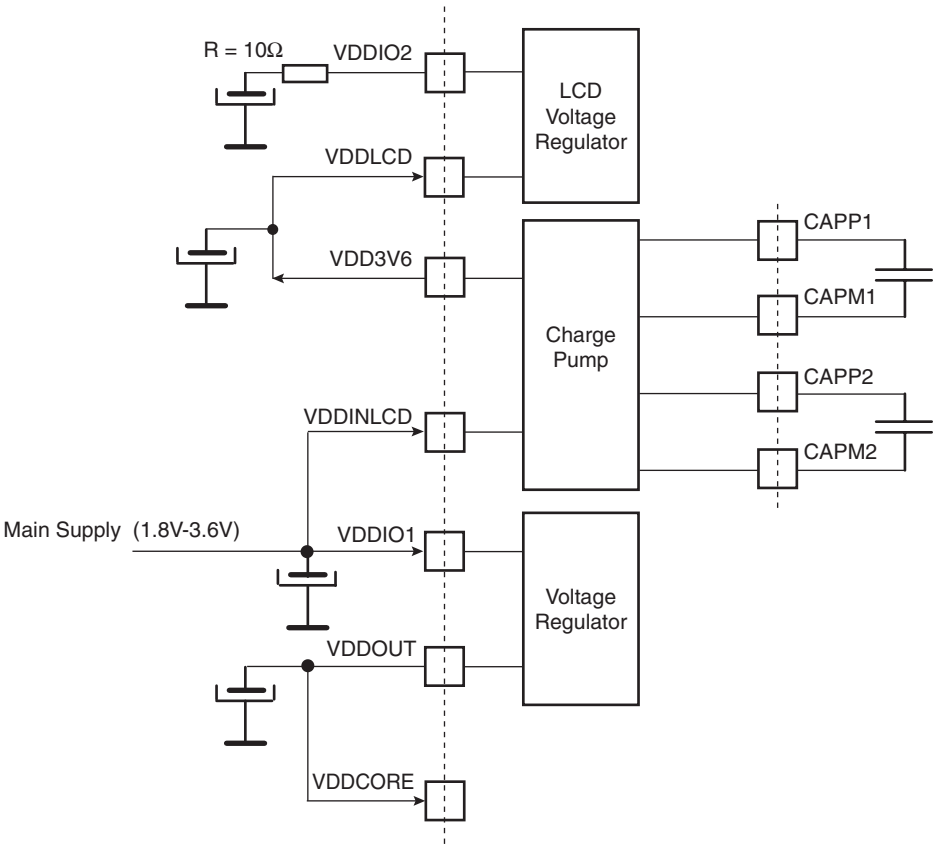



Please note that in this topology, switching time enhancement buffers are not available. (Refer Section 10.13 ”Segment LCD Controller”.)

## 5.7 Typical Powering Schematics

The AT91SAM7L128/64 supports a 1.8V-3.6V single supply mode. The internal regulator input connected to the source and its output feeds VDDCORE. Figure 5-6 shows the power schematics to be used.

**Figure 5-6.** 3.3V System Single Power Supply Schematic



# 6. I/O Line Considerations

## 6.1 JTAG Port Pins

TMS, TDI and TCK are schmitt trigger inputs. TMS, TDI and TCK do not integrate a pull-up resistor.

TDO is an output, driven at up to VDDIO, and has no pull-up resistor.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. The JTAGSEL pin integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations.

## 6.2 Test Pin

The TST pin is used for manufacturing test or fast programming mode of the AT91SAM7L128/64 when asserted high. The TST pin integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations.

To enter fast programming mode, the TST and CLKIN pins must be tied high while FWUP is tied low.

## 6.3 NRST Pin

The NRST pin is bidirectional. It is handled by the on-chip reset controller and can be driven low to provide a reset signal to the external components or asserted low externally to reset the microcontroller. There is no constraint on the length of the reset pulse and the reset controller can guarantee a minimum pulse length.

The NRST pin integrates a permanent pull-up resistor to VDDIO1 of about 100 kΩ.

## 6.4 NRSTB Pin

The NRSTB pin is input only and enables asynchronous reset of the AT91SAM7L128/64 when asserted low. The NRSTB pin integrates a permanent pull-up resistor of about 15 kΩ. This allows connection of a simple push button on the NRBST pin as a system-user reset.

In all modes, this pin will reset the chip. It can be used as an external system reset source.

In harsh environments, it is recommended to add an external capacitor (10 nF) between NRSTB and VDDIO1.

NRSTB pin must not be connected to VDDIO1. There must not be an external pull-up on NRSTB.

## 6.5 ERASE Pin

The ERASE pin is used to reinitialize the Flash content and some of its NVM bits. It integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations.

This pin is debounced by SCLK to improve the glitch tolerance. When the ERASE pin is tied high during less than 100 ms, it is not taken into account. The pin must be tied high during more than 220 ms to perform the reinitialization of the Flash.

## 6.6 PIO Controller Lines

All the I/O lines; PA0 to PA25, PB0 to PB23, PC0 to PC29 integrate a programmable pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the PIO controllers. All I/Os have input schmitt triggers.

Typical pull-up value is 100 kΩ.

Maximum frequency is:

- 36 MHz under 25 pF of load on PIOC
- 36 MHz under 25 pF of load on PIOA and PIOB

## 6.7 I/O Line Current Drawing

The PIO lines PC5 to PC8 are high-drive current capable. Each of these I/O lines can drive up to 4 mA permanently. The remaining I/O lines can draw only 2 mA.

Each I/O is designed to achieve very small leakage. However, the total current drawn by all the I/O lines cannot exceed 150 mA.

# 7. Processor and Architecture

## 7.1 ARM7TDMI Processor

- RISC processor based on ARMv4T Von Neumann Architecture
  - Runs at up to 36 MHz, providing 0.9 MIPS/MHz
- Two instruction sets
  - ARM® high-performance 32-bit instruction set
  - Thumb high code density 16-bit instruction set
- Three-stage pipeline architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

## 7.2 Debug and Test Features

- Integrated embedded in-circuit emulator
  - Two watchpoint units
  - Test access port accessible through a JTAG protocol
  - Debug communication channel
- Debug Unit
  - Two-pin UART
  - Debug communication channel interrupt handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on all digital pins

## 7.3 Memory Controller

- Programmable Bus Arbiter
  - Handles requests from the ARM7TDMI and the Peripheral DMA Controller
- Address decoder provides selection signals for
  - Five internal 1 Mbyte memory areas
  - One 256 Mbyte embedded peripheral area
- Abort Status Registers
  - Source, Type and all parameters of the access leading to an abort are saved
  - Facilitates debug by detection of bad pointers
- Misalignment Detector
  - Alignment checking of all data accesses
  - Abort generation in case of misalignment
- Remap Command
  - Remaps the SRAM in place of the embedded non-volatile memory
  - Allows handling of dynamic exception vectors
  - Peripheral protection against write and/or user access
- Enhanced Embedded Flash Controller

  - Embedded Flash interface, up to three programmable wait states
  - Prefetch buffer, buffering and anticipating the 16-bit requests, reducing the required wait states
  - Key-protected program, erase and lock/unlock sequencer
  - Single command for erasing, programming and locking operations
  - Interrupt generation in case of forbidden operation

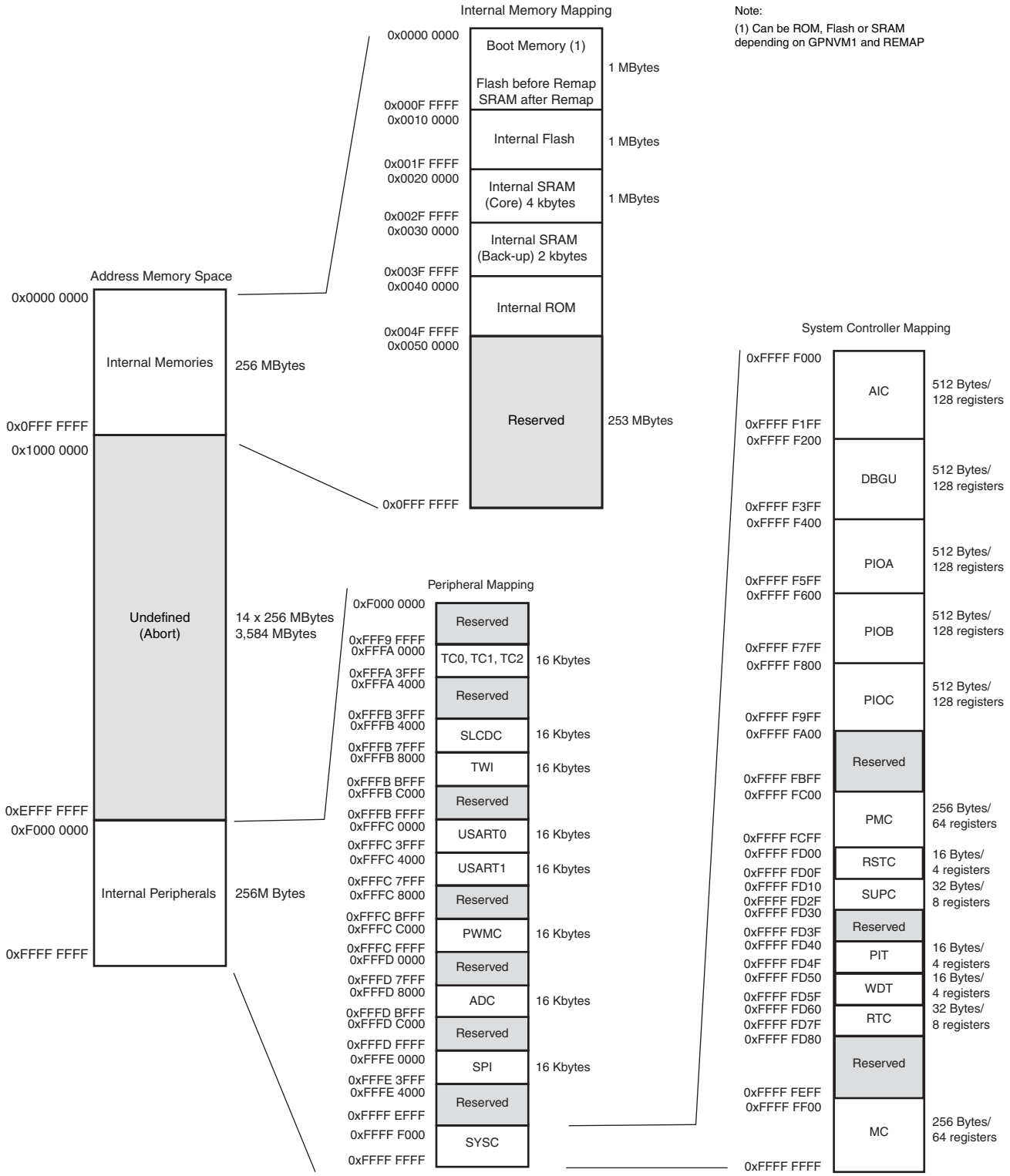
## 7.4 Peripheral DMA Controller

- Handles data transfer between peripherals and memories
- Eleven channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for the Serial Peripheral Interface
  - Two for the Two Wire Interface
  - One for the Analog-to-digital Converter
- Low bus arbitration overhead
  - One Master Clock cycle needed for a transfer from memory to peripheral
  - Two Master Clock cycles needed for a transfer from peripheral to memory
- Next Pointer management for reducing interrupt latency requirements

# 8. Memories

- 128 Kbytes of Flash Memory (AT91SAM7L128)
  - Single plane
  - One bank of 512 pages of 256 bytes
  - Fast access time, 15 MHz single-cycle access in Worst Case conditions
  - Page programming time: 4.6 ms, including page auto-erase
  - Page programming without auto-erase: 2.3 ms
  - Full chip erase time: 10 ms
  - 10,000 write cycles, 10-year data retention capability
  - 16 lock bits, each protecting 16 lock regions of 32 pages
  - Protection Mode to secure contents of the Flash
- 64 Kbytes of Flash Memory (AT91SAM7L64)
  - Single plane
  - One bank of 256 pages of 256 bytes
  - Fast access time, 15 MHz single-cycle access in Worst Case conditions
  - Page programming time: 4.6 ms, including page auto-erase
  - Page programming without auto-erase: 2.3 ms
  - Full chip erase time: 10 ms
  - 10,000 write cycles, 10-year data retention capability
  - 8 lock bits, each protecting 8 lock regions of 32 pages
  - Protection Mode to secure contents of the Flash
- 6 Kbytes of Fast SRAM
  - Single-cycle access at full speed
  - 2 Kbytes of Backup SRAM
  - 4 Kbytes of Core SRAM

**Figure 8-1.** Memory Mapping



## 8.1 Embedded Memories

### 8.1.1 Internal Memories

#### *8.1.1.1 Internal SRAM*

The AT91SAM7L128/64 embeds a high-speed 4-Kbyte SRAM bank and a 2-Kbyte backup SRAM bank. The backup SRAM is directly supplied on 1.8V-3.6V supply domain.

The 4-Kbyte Core SRAM is supplied by VDDCORE which is connected to the output of the voltage regulator.

After reset and until the Remap Command is performed, the 4-Kbyte Core SRAM is only accessible at address 0x0020 0000. The 2-Kbyte Backup SRAM is accessible at address 0x0030 0000.

After remap, the 4-Kbyte Core SRAM also becomes available at address 0x0.

The user can see the 6 Kbytes of SRAM contiguously at address 0x002F F000.

#### *8.1.1.2 Internal ROM*

The AT91SAM7L128/64 embeds an Internal ROM. The ROM is always mapped at address 0x0040 0000. The ROM contains the FFPI and SAM-BA program.

ROM size is 12 Kbytes.

#### *8.1.1.3 Internal Flash*

- The AT91SAM7L128 features one bank of 128 Kbytes of Flash.
- The AT91SAM7L64 features one bank of 64 Kbytes of Flash.

At any time, the Flash is mapped to address 0x0010 0000.

A general purpose NVM (GPNVM1) bit is used to boot either on the ROM (default) or from the Flash.

This GPNVM1 bit can be cleared or set respectively through the commands “Clear General-purpose NVM Bit” and “Set General-purpose NVM Bit” of the EEFC User Interface.

Setting the GPNVM Bit 1 selects the boot from the Flash, clearing it selects the boot from the ROM. Asserting ERASE clears the GPNVM Bit 1 and thus selects the boot from the ROM by default.

**Figure 8-2.** Internal Memory Mapping with GPNVM Bit 1 = 0 (default)

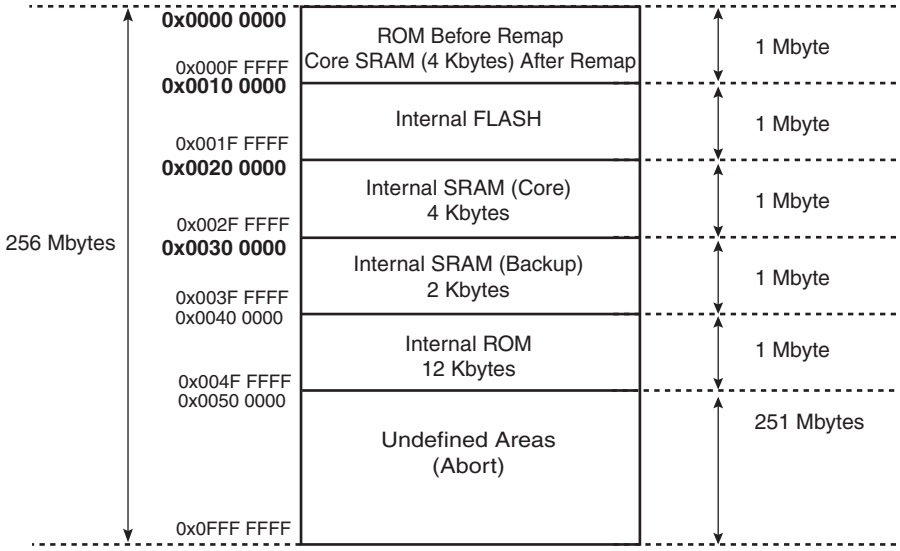
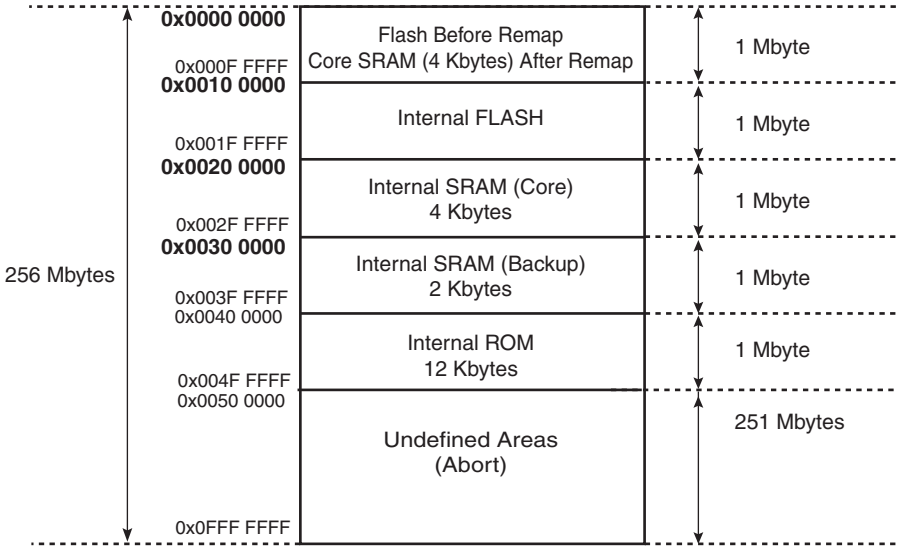



**Figure 8-3.** Internal Memory Mapping with GPNVM Bit 1 = 1




### 8.1.2 Embedded Flash

#### *8.1.2.1 Flash Overview*

- The Flash of the AT91SAM7L128 is organized in 512 pages (single plane) of 256 bytes.
- The Flash of the AT91SAM7L64 is organized in 256 pages (single plane) of 256 bytes.

The Flash contains a 128-byte write buffer, accessible through a 32-bit interface.

#### *8.1.2.2 Flash Power Supply*

The Flash is supplied by VDDCORE through a power switch controlled by the Supply Controller.

#### 8.1.2.3 *Enhanced Embedded Flash Controller*

The Enhanced Embedded Flash Controller (EEFC) manages accesses performed by the masters of the system. It enables reading the Flash and writing the write buffer. It also contains a User Interface, mapped within the Memory Controller on the APB.

The Enhanced Embedded Flash Controller ensures the interface of the Flash block with the 32-bit internal bus. Its 128-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

#### 8.1.2.4 *Lock Regions*

The AT91SAM7L128 Embedded Flash Controller manages 16 lock bits to protect 16 regions of the flash against inadvertent flash erasing or programming commands. The AT91SAM7L128 contains 16 lock regions and each lock region contains 32 pages of 256 bytes. Each lock region has a size of 8 Kbytes.

The AT91SAM7L64 Embedded Flash Controller manages 8 lock bits to protect 8 regions of the flash against inadvertent flash erasing or programming commands. The AT91SAM7L64 contains 8 lock regions and each lock region contains 32 pages of 256 bytes. Each lock region has a size of 8 Kbytes.

If a locked-region's erase or program command occurs, the command is aborted and the EEFC triggers an interrupt.

The 16 NVM bits are software programmable through the EEFC User Interface. The command "Set Lock Bit" enables the protection. The command "Clear Lock Bit" unlocks the lock region.

Asserting the ERASE pin clears the lock bits, thus unlocking the entire Flash.

#### 8.1.2.5 *Security Bit Feature*

The AT91SAM7L128/64 features a security bit, based on a specific General Purpose NVM bit (GPNVM bit 0). When the security is enabled, any access to the Flash, either through the ICE interface or through the Fast Flash Programming Interface, is forbidden. This ensures the confidentiality of the code programmed in the Flash.

This security bit can only be enabled, through the command "Set General Purpose NVM Bit 0" of the EEFC User Interface. Disabling the security bit can only be achieved by asserting the ERASE pin at 1, and after a full Flash erase is performed. When the security bit is deactivated, all accesses to the Flash are permitted.

It is important to note that the assertion of the ERASE pin should always be longer than 200 ms.

As the ERASE pin integrates a permanent pull-down, it can be left unconnected during normal operation. However, it is safer to connect it directly to GND for the final application.

#### 8.1.2.6 *Calibration Bits*

NVM bits are used to calibrate the brownout detector and the voltage regulator. These bits are factory configured and cannot be changed by the user. The ERASE pin has no effect on the calibration bits.

#### 8.1.2.7 *GPNVM Bits*

The AT91SAM7L128/64 features two GPNVM bits that can be cleared or set respectively through the commands “Clear GPNVM Bit” and “Set GPNVM Bit” of the EEFC User Interface..

**Table 8-1.** General-purpose Non-volatile Memory Bits

| GPNVMBit[#] | Function |
|---|---|
| 0 | Security bit |
| 1 | Boot mode selection |

### 8.1.3 Fast Flash Programming Interface

The Fast Flash Programming Interface allows programming the device through either a serial JTAG interface or through a multiplexed fully-handshaked parallel port. It allows gang programming with market-standard industrial programmers.

The FFPI supports read, page program, page erase, full erase, lock, unlock and protect commands.

The Fast Flash Programming Interface is enabled and the Fast Programming Mode is entered when TST and CLKIN are tied high while FWUP is tied low.

- The Flash of the AT91SAM7L128 is organized in 512 pages of 256 bytes (single plane).
- The Flash of the AT91SAM7L64 is organized in 256 pages of 256 bytes (single plane).

The Flash contains a 128-byte write buffer, accessible through a 32-bit interface.

### 8.1.4 SAM-BA Boot

The SAM-BA Boot is a default Boot Program which provides an easy way to program in-situ the on-chip Flash memory.

The SAM-BA Boot Assistant supports serial communication via the DBGU.

The SAM-BA Boot provides an interface with SAM-BA Graphic User Interface (GUI).

The SAM-BA Boot resides in ROM and is mapped at address 0x0 when GPNVM bit 1 is set to 0.

# 9. System Controller

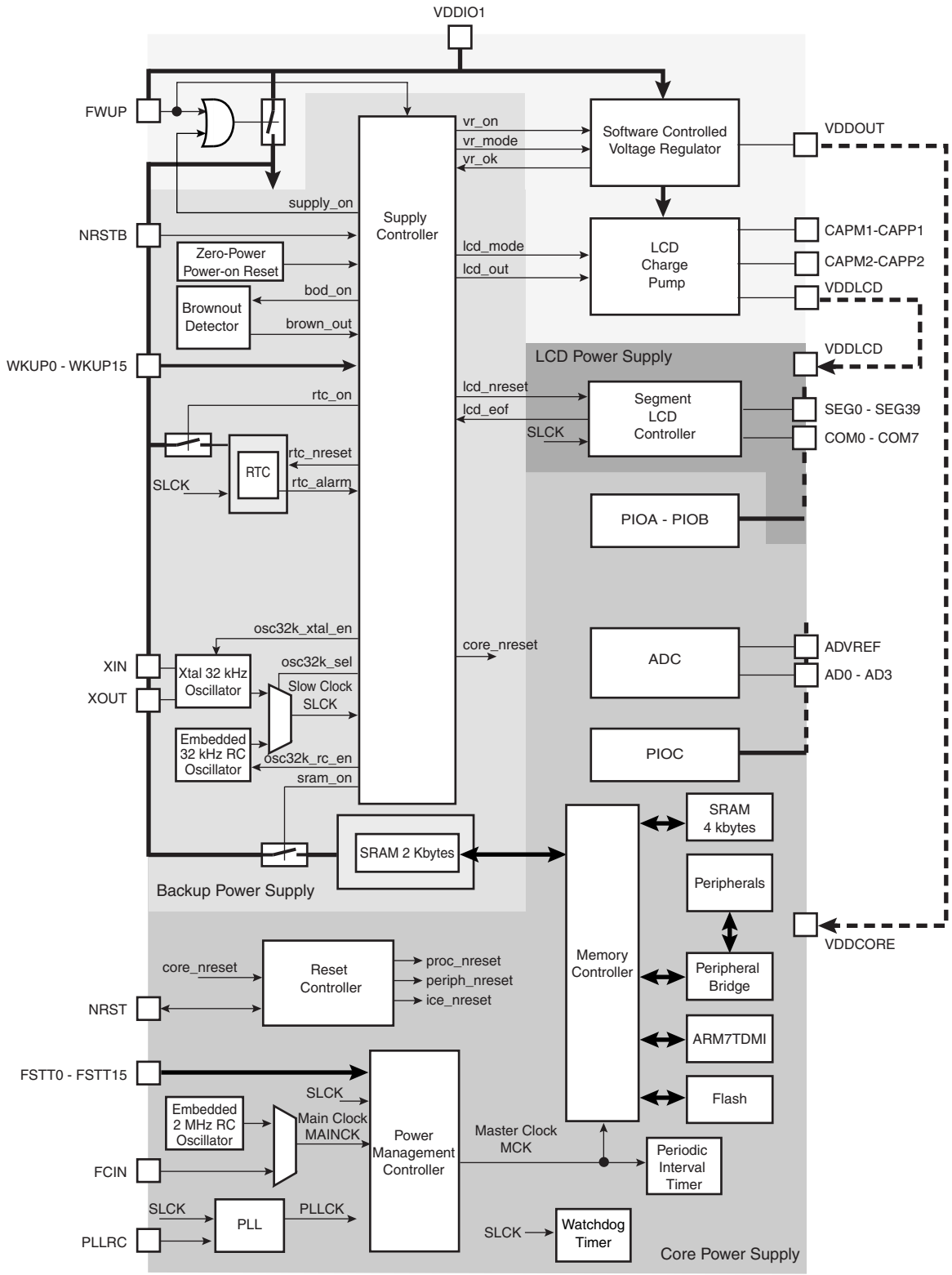
The System Controller manages all vital blocks of the microcontroller, interrupts, clocks, power, time, debug and reset.

The System Controller Block Diagram is shown in Figure 9-1 on page 30.

## 9.1 System Controller Mapping

The System Controller peripherals are all mapped to the highest 4 Kbytes of address space, between addresses 0xFFFF F000 and 0xFFFF FFFF. Figure 8-1 on page 24 shows the mapping of the System Controller. Note that the Memory Controller configuration user interface is also mapped within this address space

**Figure 9-1.** System Controller Block Diagram



## 9.2 Supply Controller (SUPC)

The Supply Controller controls the power supplies of each section of the product:

- the processor and the peripherals
- the Flash memory
- the backup SRAM
- the LCD controller, the charge pump and the LCD voltage regulator
- the Real Time Clock

The Supply Controller has its own reset circuitry and is clocked by the 32 kHz Slow clock generator.

The reset circuitry is based on the NRSTB pin, a zero-power power-on reset cell and a brownout detector cell. The zero-power power-on reset allows the Supply Controller to start properly, while the software-programmable brownout detector allows detection of either a battery discharge or main voltage loss.

The Slow Clock generator is based on a 32 kHz crystal oscillator and an embedded 32 kHz RC oscillator. The Slow Clock defaults to the RC oscillator, but the software can enable the crystal oscillator and select it as the Slow Clock source.

The Supply Controller starts up the device by sequentially enabling the internal power switches and the Voltage Regulator, then it generates the proper reset signals to the core power supply.

It also enables to set the system in different low power modes and to wake it up from a wide range of events.

## 9.3 Reset Controller

- Based on one power-on reset cell and a brownout detector
- Status of the last reset; either power-up reset, software reset, user reset, watchdog reset, brownout reset
- Controls the internal resets and the NRST pin output
- Allows to shape a signal on the NRST line, guaranteeing that the length of the pulse meets any requirement.

### 9.3.1 Brownout Detector (BOD) and Power-on Reset

The AT91SAM7L128/64 embeds one zero-power power-on reset and a brownout detection circuit. Both monitor VDDIO1.

The zero-power power-on reset circuit is always active. It provides an internal reset signal to the AT91SAM7L128/64 for power-on and power-off operations and ensures a proper reset for the Supply Controller.

The brownout detection circuit is disabled by default and can be enabled by software. It monitors VDDIO1.

The brownout detection circuit is factory calibrated.

The threshold is programmable via software. It can be selected from 1.9V to 3.4V with 100 mV steps. It can be programmed to generate either a wake-up alarm or a reset.

It can be used to wake up the chip from backup mode if the supply drops below a selected threshold (to warn the end user about a discharged battery for example) and to reset the chip when the voltage is too low.

BOD current consumption is 25 µA, typically.

To decrease current consumption, the software can disable the brownout detector, especially in low-power mode.

The software can also configure the BOD in “switched” mode. In this mode, an internal state machine switches on and off periodically and stores the output of the BOD.

This decreases the current consumption (inferior to 2 µA) while the detection is still active. This feature is suitable in low-power mode where voltage detection is still needed.

## 9.4 Clock Generator

The clock generator embeds one low-power RC oscillator, one fast RC oscillator, one crystal oscillator and one PLL with the following characteristics:

- RC Oscillator ranges between 22 kHz and 42 kHz
- Fast RC Oscillator ranges between 1.5 MHz and 2.5 MHz
- Crystal Oscillator at 32 kHz (can be bypassed)
- PLL output ranges between 18 MHz and 47 MHz

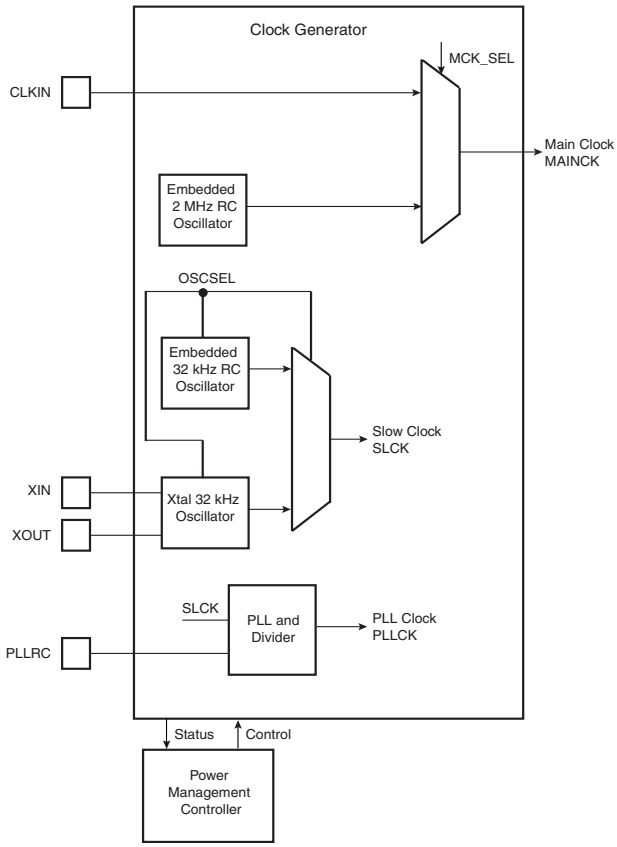
It provides SLCK, MAINCK and PLLCK.

The Supply Controller selects between the internal RC oscillator and the 32 kHz crystal oscillator. The unused oscillator is disabled so that power consumption is optimized.

The 2 MHz Fast RC oscillator is the default selected clock (MAINCK) which is used at start-up . The user can select an external clock (CLKIN) through software.

The PLL needs an external RC filter and starts up in a very short time (inferior to 1 ms).

**Figure 9-2.** Clock Generator Block Diagram




## 9.5 Power Management Controller

The Power Management Controller uses the clock generator outputs to provide:

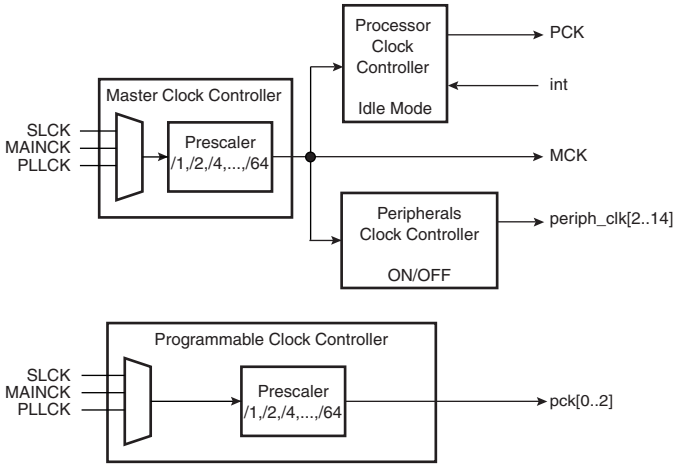
- The Processor Clock PCK
- The Master Clock MCK
- All the peripheral clocks, independently controllable
- Three programmable clock outputs PCKx

The Master Clock (MCK) is programmable from a few hundred Hz to the maximum operating frequency of the device.

The Processor Clock (PCK) switches off when entering processor idle mode, thus allowing reduced power consumption while waiting for an interrupt.

The LCD Controller clock is SCLK.

**Figure 9-3.** Power Management Controller Block Diagram




## 9.6 Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of an ARM Processor
- Individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals (RTC, PIT, EFC, PMC, DBGU, etc.)
  - Other sources control the peripheral interrupts or external interrupts
  - Programmable edge-triggered or level-sensitive internal sources
  - Programmable positive/negative edge-triggered or high/low level-sensitive external sources
- 8-level Priority Controller
  - Drives the normal interrupt nIRQ of the processor
  - Handles priority of the interrupt sources
  - Higher priority interrupts can be served during service of lower priority interrupt
- Vectoring
  - Optimizes interrupt service routine branch and execution
  - One 32-bit vector register per interrupt source
  - Interrupt vector register reads the corresponding current interrupt vector
- Protect Mode
  - Easy debugging by preventing automatic operations
- Fast Forcing
  - Permits redirecting any interrupt source on the fast interrupt
- General Interrupt Mask
  - Provides processor synchronization on events without triggering an interrupt

## 9.7 Debug Unit

- Comprises:
  - One two-pin UART
  - One Interface for the Debug Communication Channel (DCC) support

  - One set of Chip ID Registers
  - One Interface providing ICE Access Prevention
- Two-pin UART
  - USART-compatible User Interface
  - Programmable Baud Rate Generator
  - Parity, Framing and Overrun Error
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
- Debug Communication Channel Support
  - Offers visibility of COMMRX and COMMTX signals from the ARM Processor
- Chip ID Registers
  - Identification of the device revision, sizes of the embedded memories, set of peripherals
  - Chip ID is 0x2733 0740 (VERSION 0) for AT91SAM7L128
  - Chip ID is 0x2733 0540 (VERSION 0) for AT91SAM7L64

## 9.8 Period Interval Timer

- 20-bit programmable counter plus 12-bit interval counter

## 9.9 Watchdog Timer

- 12-bit key-protected Programmable Counter running on prescaled SLCK
- Provides reset or interrupt signals to the system
- Counter may be stopped while the processor is in debug state or in idle mode

## 9.10 Real-time Clock

- Two Hundred Year Calendar
- Programmable Periodic Interrupt
- Time, Date and Alarm 32-bit Parallel Load

## 9.11 PIO Controllers

- Three PIO Controllers.
  - PIO A controls 26 I/O lines
  - PIO B controls 24 I/O lines
  - PIO C controls 30 I/O lines
- Fully programmable through set/clear registers
- Multiplexing of two peripheral functions per I/O line
- For each I/O line (whether assigned to a peripheral or used as general-purpose I/O)
  - Input change interrupt
  - Half a clock period glitch filter
  - Multi-drive option enables driving in open drain
  - Programmable pull-up on each I/O line
  - Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set and Clear of several I/O lines in a single write

# 10. Peripherals

## 10.1 User Interface

The User Peripherals are mapped in the 256 MBytes of the address space between 0xF000 0000 and 0xFFFF EFFF. Each peripheral is allocated 16 Kbytes of address space.

A complete memory map is presented in Figure 8-1 on page 24.

## 10.2 Peripheral Identifiers

The AT91SAM7L128/64 embeds a wide range of peripherals. Table 10-1 defines the Peripheral Identifiers of the AT91SAM7L128/64. Unique peripheral identifiers are defined for both the Advanced Interrupt Controller and the Power Management Controller.

**Table 10-1.** Peripheral Identifiers

| Peripheral ID | Peripheral Mnemonic | Peripheral Name | External Interrupt |
|---|---|---|---|
| 0 | AIC | Advanced Interrupt Controller | FIQ |
| 1 | SYSIRQ(1) | System Interrupt | |
| 2 | PIOA | Parallel I/O Controller A | |
| 3 | PIOB | Parallel I/O Controller B | |
| 4 | PIOC | Parallel I/O Controller C | |
| 5 | SPI | Serial Peripheral Interface | |
| 6 | US0 | USART 0 | |
| 7 | US1 | USART 1 | |
| 8 | Reserved | | |
| 9 | TWI | Two-wire Interface | |
| 10 | PWMC | PWM Controller | |
| 11 | SLCDC | Segmented LCD Controller | |
| 12 | TC0 | Timer/Counter 0 | |
| 13 | TC1 | Timer/Counter 1 | |
| 14 | TC2 | Timer/Counter 2 | |
| 15 | ADC(1) | Analog-to Digital Converter | |
| 16 - 29 | Reserved | | |
| 30 | AIC | Advanced Interrupt Controller | IRQ0 |
| 31 | AIC | Advanced Interrupt Controller | IRQ1 |

Note: 1. Setting SYSIRQ and ADC bits in the clock set/clear registers of the PMC has no effect. The System Controller and ADC are continuously clocked. The ADC clock is automatically started for the first conversion. In Sleep Mode the ADC clock is automatically stopped after each conversion.

## 10.3 Peripheral Multiplexing on PIO Lines

The AT91SAM7L128/64 features three PIO controllers, PIOA, PIOB and PIOC, that multiplex the I/O lines of the peripheral set.

PIO Controller A, B and C control respectively 26, 24 and 30 lines. Each line can be assigned to one of two peripheral functions, A or B.

Table 10-2 on page 38 defines how the I/O lines of the peripherals A, B or the analog inputs are multiplexed on the PIO Controller A, B and C. The two columns “Function” and “Comments” have been inserted for the user’s own comments; they may be used to track how pins are defined in an application.

Note that some peripheral functions that are output only may be duplicated in the table.

At reset, all I/O lines are automatically configured as input with the programmable pull-up enabled, so that the device is maintained in a static state as soon as a reset is detected.

## 10.4 PIO Controller A Multiplexing

**Table 10-2.** Multiplexing on PIO Controller A

| PIO Controller A | | | | Application Usage | |
|---|---|---|---|---|---|
| **I/O Line** | **Peripheral A** | **Peripheral B** | **Extra Function** | **Function** | **Comments** |
| PA0 | | | COM0 | | |
| PA1 | | | COM1 | | |
| PA2 | | | COM2 | | |
| PA3 | | | COM3 | | |
| PA4 | | | COM4 | | |
| PA5 | | | COM5 | | |
| PA6 | | | SEG0 | | |
| PA7 | | | SEG1 | | |
| PA8 | | | SEG2 | | |
| PA9 | | | SEG3 | | |
| PA10 | | | SEG4 | | |
| PA11 | | | SEG5 | | |
| PA12 | | | SEG6 | | |
| PA13 | | | SEG7 | | |
| PA14 | | | SEG8 | | |
| PA15 | | | SEG9 | | |
| PA16 | | | SEG10 | | |
| PA17 | | | SEG11 | | |
| PA18 | | | SEG12 | | |
| PA19 | | | SEG13 | | |
| PA20 | | | SEG14 | | |
| PA21 | | | SEG15 | | |
| PA22 | | | SEG16 | | |
| PA23 | | | SEG17 | | |
| PA24 | | | SEG18 | | |
| PA25 | | | SEG19 | | |

## 10.5 PIO Controller B Multiplexing

**Table 10-3.** Multiplexing on PIO Controller B

| PIO Controller B | | | | Application Usage | |
|---|---|---|---|---|---|
| **I/O Line** | **Peripheral A** | **Peripheral B** | **Extra Function** | **Function** | **Comments** |
| PB0 | | | SEG20 | | |
| PB1 | | | SEG21 | | |
| PB2 | | | SEG22 | | |
| PB3 | | | SEG23 | | |
| PB4 | | | SEG24 | | |
| PB5 | | | SEG25 | | |
| PB6 | | | SEG26 | | |
| PB7 | | | SEG27 | | |
| PB8 | | | SEG28 | | |
| PB9 | | | SEG29 | | |
| PB10 | | | SEG30 | | |
| PB11 | | | SEG31 | | |
| PB12 | NPCS3 | | SEG32 | | |
| PB13 | NPCS2 | | SEG33 | | |
| PB14 | NPCS1 | | SEG34 | | |
| PB15 | RTS1 | | SEG35 | | |
| PB16 | RTS0 | | SEG36 | | |
| PB17 | DTR1 | | SEG37 | | |
| PB18 | PWM0 | | SEG38 | | |
| PB19 | PWM1 | | SEG39 | | |
| PB20 | PWM2 | | COM6 | | |
| PB21 | PWM3 | | COM7 | | |
| PB22 | NPCS1 | PCK1 | COM8 | | |
| PB23 | PCK0 | NPCS3 | COM9 | | |

## 10.6 PIO Controller C Multiplexing

**Table 10-4.** Multiplexing on PIO Controller C

| PIO Controller C | | | | Application Usage | |
|---|---|---|---|---|---|
| **I/O Line** | **Peripheral A** | **Peripheral B** | **Extra Functions** | **Function** | **Comments** |
| PC0 | CTS1 | PWM2 | PGMEN0/WKUP0[1][2] | | |
| PC1 | DCD1 | TIOA2 | PGMEN1/WKUP1[1][2] | | |
| PC2 | DTR1 | TIOB2 | PGMEN2/WKUP2[1][2] | | |
| PC3 | DSR1 | TCLK1 | PGMNCMD/WKUP3[1][2] | | |
| PC4 | RI1 | TCLK2 | PGMRDY/WKUP4[1][2] | | |
| PC5 | IRQ1 | NPCS2 | PGMNOE/WKUP5[1][2] | | |
| PC6 | NPCS1 | PCK2 | PGMNVALID/WKUP6[1][2] | | |
| PC7 | PWM0 | TIOA0 | PGMMO/High drive | | |
| PC8 | PWM1 | TIOB0 | PGMM1/High drive | | |
| PC9 | PWM2 | SCK0 | PGMM2/High drive | | |
| PC10 | TWD | NPCS3 | PGMM3/High drive | | |
| PC11 | TWCK | TCLK0 | PGMD0/WKUP7[1][2] | | |
| PC12 | RXD0 | NPCS3 | PGMD1/WKUP8[1][2] | | |
| PC13 | TXD0 | PCK0 | PGMD2/WKUP9[1][2] | | |
| PC14 | RTS0 | ADTRG | PGMD3/WKUP10[1][2] | | |
| PC15 | CTS0 | PWM3 | PGMD4/WKUP11[1][2] | | |
| PC16 | DRXD | NPCS1 | PGMD5 | | |
| PC17 | DTXD | NPCS2 | PGMD6 | | |
| PC18 | NPCS0 | PWM0 | PGMD7 | | |
| PC19 | MISO | PWM1 | PGMD8 | | |
| PC20 | MOSI | PWM2 | PGMD9 | | |
| PC21 | SPCK | PWM3 | PGMD10 | | |
| PC22 | NPCS3 | TIOA1 | PGMD11 | | |
| PC23 | PCK0 | TIOB1 | PGMD12 | | |
| PC24 | RXD1 | PCK1 | PGMD13 | | |
| PC25 | TXD1 | PCK2 | PGMD14 | | |
| PC26 | RTS0 | FIQ | PGMD15/WKUP12[1][2] | | |
| PC27 | NPCS2 | IRQ0 | WKUP13[1][2] | | |
| PC28 | SCK1 | PWM0 | WKUP14[1][2] | | |
| PC29 | RTS1 | PWM1 | WKUP15[1][2] | | |

Notes: 1. Wake-Up source in Backup mode (managed by the SUPC).
2. Fast Start-Up source in Wait mode (managed by the PMC).

## 10.7 Serial Peripheral Interface

- Supports communication with external serial devices
  - Four chip selects with external decoder allow communication with up to 15 peripherals
  - Serial memories, such as DataFlash® and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface
  - 8- to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delays per chip select, between consecutive transfers and between clock and data
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
  - Maximum frequency at up to Master Clock

## 10.8 Two Wire Interface

- Master, Multi-Master and Slave Mode Operation
- Compatibility with Atmel two-wire interface, serial memory and $I^2C$ compatible devices
- One, two or three bytes for slave address
- Sequential read/write operations
- Bit Rate: Up to 400 kbit/s
- General Call Supported in Slave Mode
- Connecting to PDC channel capabilities optimizes data transfers in Master Mode only
  - One channel for the receiver, one channel for the transmitter
  - Next buffer support

## 10.9 USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode
  - 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB or LSB first
  - Optional break generation and detection
  - By 8 or by 16 over-sampling receiver frequency
  - Hardware handshaking RTS - CTS
  - Modem Signals Management DTR-DSR-DCD-RI on USART1
  - Receiver time-out and transmitter timeguard

  - Multi-drop Mode with address generation and detection
  - Optional Manchester Encoding
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- IrDA modulation and demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo

## 10.10 Timer Counter

- Three 16-bit Timer Counter Channels
  - Three output compare or two input capture
- Wide range of functions including:
  - Frequency measurement
  - Event counting
  - Interval measurement
  - Pulse generation
  - Delay timing
  - Pulse Width Modulation
  - Up/down capabilities
- Each channel is user-configurable and contains:
  - Three external clock inputs
- Five internal clock inputs, as defined in Table 10-5

**Table 10-5.** Timer Counter Clock Assignment

| TC Clock input | Clock |
|---|---|
| TIMER_CLOCK1 | MCK/2 |
| TIMER_CLOCK2 | MCK/8 |
| TIMER_CLOCK3 | MCK/32 |
| TIMER_CLOCK4 | MCK/128 |
| TIMER_CLOCK5 | MCK/1024 |

  - Two multi-purpose input/output signals
  - Two global registers that act on all three TC channels

## 10.11 PWM Controller

- Four channels, one 16-bit counter per channel
- Common clock generator, providing thirteen different clocks
  - One Modulo n counter providing eleven clocks
  - Two independent linear dividers working on modulo n counter outputs

- Independent channel programming
  - Independent enable/disable commands
  - Independent clock selection
  - Independent period and duty cycle, with double buffering
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

## 10.12 Analog-to-Digital Converter

- 4-channel ADC supplied by the internal voltage regulator
- 10-bit 460 Ksamples/sec. or 8-bit 660 Ksamples/sec. Successive Approximation Register ADC
- ±2 LSB Integral Non Linearity, ±1 LSB Differential Non Linearity
- Integrated 4-to-1 multiplexer
- External voltage reference for better accuracy on low voltage inputs
- Individual enable and disable of each channel
- Multiple trigger sources
  - Hardware or software trigger
  - External trigger pin
  - Timer Counter 0 to 2 outputs TIOA0 to TIOA2 trigger
- Sleep Mode and conversion sequencer
  - Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels

## 10.13 Segment LCD Controller

The Segment LCD Controller/driver is intended for monochrome passive liquid crystal display (LCD) with up to 10 common terminals and up to 40 segment terminals.

- 40 segments and 10 common terminals display capacity
- Support static, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9 and 1/10 Duty
- Support static, 1/2, 1/3, 1/4 Bias
- Power-save mode display
- Software-selectable low-power waveform capability
- Flexible frame frequency selection
- Segment and common pins, not needed for driving the display, can be used as ordinary I/O pins
- Switching time enhancement internal buffers

# 11. ARM7TDMI Processor Overview

## 11.1 Overview

The ARM7TDMI core executes both the 32-bit ARM and 16-bit Thumb instruction sets, allowing the user to trade off between high performance and high code density.The ARM7TDMI processor implements Von Neuman architecture, using a three-stage pipeline consisting of Fetch, Decode, and Execute stages.

The main features of the ARM7tDMI processor are:

- ARM7TDMI Based on ARMv4T Architecture
- Two Instruction Sets
  - ARM High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- Three-Stage Pipeline Architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

## 11.2 ARM7TDMI Processor

For further details on ARM7TDMI, refer to the following ARM documents:

ARM Architecture Reference Manual (DDI 0100E)

ARM7TDMI Technical Reference Manual (DDI 0210B)

### 11.2.1 Instruction Type

Instructions are either 32 bits long (in ARM state) or 16 bits long (in THUMB state).

### 11.2.2 Data Type

ARM7TDMI supports byte (8-bit), half-word (16-bit) and word (32-bit) data types. Words must be aligned to four-byte boundaries and half words to two-byte boundaries.

Unaligned data access behavior depends on which instruction is used where.

### 11.2.3 ARM7TDMI Operating Mode

The ARM7TDMI, based on ARM architecture v4T, supports seven processor modes:

**User**: The normal ARM program execution state

**FIQ**: Designed to support high-speed data transfer or channel process

**IRQ**: Used for general-purpose interrupt handling

**Supervisor**: Protected mode for the operating system

**Abort mode**: Implements virtual memory and/or memory protection

**System**: A privileged user mode for the operating system

**Undefined**: Supports software emulation of hardware coprocessors

Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs execute in User mode. The non-user modes, or privileged modes, are entered in order to service interrupts or exceptions, or to access protected resources.

### 11.2.4 ARM7TDMI Registers

The ARM7TDMI processor has a total of 37 registers:

- 31 general-purpose 32-bit registers
- 6 status registers

These registers are not accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

At any one time 16 registers are visible to the user. The remainder are synonyms used to speed up exception processing.

Register 15 is the Program Counter (PC) and can be used in all instructions to reference data relative to the current instruction.

R14 holds the return address after a subroutine call.

R13 is used (by software convention) as a stack pointer.

**Table 11-1.** ARM7TDMI ARM Modes and Registers Layout

| User and System Mode | Supervisor Mode | Abort Mode | Undefined Mode | Interrupt Mode | Fast Interrupt Mode |
|---|---|---|---|---|---|
| R0 | R0 | R0 | R0 | R0 | R0 |
| R1 | R1 | R1 | R1 | R1 | R1 |
| R2 | R2 | R2 | R2 | R2 | R2 |
| R3 | R3 | R3 | R3 | R3 | R3 |
| R4 | R4 | R4 | R4 | R4 | R4 |
| R5 | R5 | R5 | R5 | R5 | R5 |
| R6 | R6 | R6 | R6 | R6 | R6 |
| R7 | R7 | R7 | R7 | R7 | R7 |
| R8 | R8 | R8 | R8 | R8 | R8_FIQ |
| R9 | R9 | R9 | R9 | R9 | R9_FIQ |
| R10 | R10 | R10 | R10 | R10 | R10_FIQ |
| R11 | R11 | R11 | R11 | R11 | R11_FIQ |
| R12 | R12 | R12 | R12 | R12 | R12_FIQ |
| R13 | R13_SVC | R13_ABORT | R13_UNDEF | R13_IRQ | R13_FIQ |
| R14 | R14_SVC | R14_ABORT | R14_UNDEF | R14_IRQ | R14_FIQ |
| PC | PC | PC | PC | PC | PC |
| CPSR | CPSR | CPSR | CPSR | CPSR | CPSR |
| | SPSR_SVC | SPSR_ABORT | SPSR_UNDEF | SPSR_IRQ | SPSR_FIQ |

Mode-specific banked registers

Registers R0 to R7 are unbanked registers. This means that each of them refers to the same 32-bit physical register in all processor modes. They are general-purpose registers, with no special uses managed by the architecture, and can be used wherever an instruction allows a general-purpose register to be specified.

Registers R8 to R14 are banked registers. This means that each of them depends on the current mode of the processor.

#### *11.2.4.1 Modes and Exception Handling*

All exceptions have banked registers for R14 and R13.

After an exception, R14 holds the return address for exception processing. This address is used to return after the exception is processed, as well as to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer.

The fast interrupt mode also banks registers 8 to 12 so that interrupt processing can begin without having to save these registers.

A seventh processing mode, System Mode, does not have any banked registers. It uses the User Mode registers. System Mode runs tasks that require a privileged processor mode and allows them to invoke all classes of exceptions.

#### *11.2.4.2 Status Registers*

All other processor states are held in status registers. The current operating processor status is in the Current Program Status Register (CPSR). The CPSR holds:

- four ALU flags (Negative, Zero, Carry, and Overflow)
- two interrupt disable bits (one for each type of interrupt)
- one bit to indicate ARM or Thumb execution
- five bits to encode the current processor mode

All five exception modes also have a Saved Program Status Register (SPSR) that holds the CPSR of the task immediately preceding the exception.

#### *11.2.4.3 Exception Types*

The ARM7TDMI supports five types of exception and a privileged processing mode for each type. The types of exceptions are:

- fast interrupt (FIQ)
- normal interrupt (IRQ)
- memory aborts (used to implement memory protection or virtual memory)
- attempted execution of an undefined instruction
- software interrupts (SWIs)

Exceptions are generated by internal and external sources.

More than one exception can occur in the same time.

When an exception occurs, the banked version of R14 and the SPSR for the exception mode are used to save state.

To return after handling the exception, the SPSR is moved to the CPSR, and R14 is moved to the PC. This can be done in two ways:

- by using a data-processing instruction with the S-bit set, and the PC as the destination
- by using the Load Multiple with Restore CPSR instruction (LDM)

### 11.2.5 ARM Instruction Set Overview

The ARM instruction set is divided into:

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load and Store instructions
- Coprocessor instructions
- Exception-generating instructions

ARM instructions can be executed conditionally. Every instruction contains a 4-bit condition code field (bit[31:28]).

Table 11-2 gives the ARM instruction mnemonic list.

**Table 11-2.** ARM Instruction Mnemonic List

| Mnemonic | Operation |
|---|---|
| MOV | Move |
| ADD | Add |
| SUB | Subtract |
| RSB | Reverse Subtract |
| CMP | Compare |
| TST | Test |
| AND | Logical AND |
| EOR | Logical Exclusive OR |
| MUL | Multiply |
| SMULL | Sign Long Multiply |
| SMLAL | Signed Long Multiply Accumulate |
| MSR | Move to Status Register |
| B | Branch |
| BX | Branch and Exchange |
| LDR | Load Word |
| LDRSH | Load Signed Halfword |
| LDRSB | Load Signed Byte |
| LDRH | Load Half Word |
| LDRB | Load Byte |
| LDRBT | Load Register Byte with Translation |
| LDRT | Load Register with Translation |
| LDM | Load Multiple |
| SWP | Swap Word |
| MCR | Move To Coprocessor |
| LDC | Load To Coprocessor |

| Mnemonic | Operation |
|---|---|
| CDP | Coprocessor Data Processing |
| MVN | Move Not |
| ADC | Add with Carry |
| SBC | Subtract with Carry |
| RSC | Reverse Subtract with Carry |
| CMN | Compare Negated |
| TEQ | Test Equivalence |
| BIC | Bit Clear |
| ORR | Logical (inclusive) OR |
| MLA | Multiply Accumulate |
| UMULL | Unsigned Long Multiply |
| UMLAL | Unsigned Long Multiply Accumulate |
| MRS | Move From Status Register |
| BL | Branch and Link |
| SWI | Software Interrupt |
| STR | Store Word |
| STRH | Store Half Word |
| STRB | Store Byte |
| STRBT | Store Register Byte with Translation |
| STRT | Store Register with Translation |
| STM | Store Multiple |
| SWPB | Swap Byte |
| MRC | Move From Coprocessor |
| STC | Store From Coprocessor |

### 11.2.6 Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the ARM instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store Multiple instructions
- Exception-generating instruction

In Thumb mode, eight general-purpose registers, R0 to R7, are available that are the same physical registers as R0 to R7 when executing ARM instructions. Some Thumb instructions also

access to the Program Counter (ARM Register 15), the Link Register (ARM Register 14) and the Stack Pointer (ARM Register 13). Further instructions allow limited access to the ARM registers 8 to 15.

Table 11-3 gives the Thumb instruction mnemonic list.

**Table 11-3.** Thumb Instruction Mnemonic List

| Mnemonic | Operation | Mnemonic | Operation |
|---|---|---|---|
| MOV | Move | MVN | Move Not |
| ADD | Add | ADC | Add with Carry |
| SUB | Subtract | SBC | Subtract with Carry |
| CMP | Compare | CMN | Compare Negated |
| TST | Test | NEG | Negate |
| AND | Logical AND | BIC | Bit Clear |
| EOR | Logical Exclusive OR | ORR | Logical (inclusive) OR |
| LSL | Logical Shift Left | LSR | Logical Shift Right |
| ASR | Arithmetic Shift Right | ROR | Rotate Right |
| MUL | Multiply | | |
| B | Branch | BL | Branch and Link |
| BX | Branch and Exchange | SWI | Software Interrupt |
| LDR | Load Word | STR | Store Word |
| LDRH | Load Half Word | STRH | Store Half Word |
| LDRB | Load Byte | STRB | Store Byte |
| LDRSH | Load Signed Halfword | LDRSB | Load Signed Byte |
| LDMIA | Load Multiple | STMIA | Store Multiple |
| PUSH | Push Register to stack | POP | Pop Register from stack |

# 12. Debug and Test Features

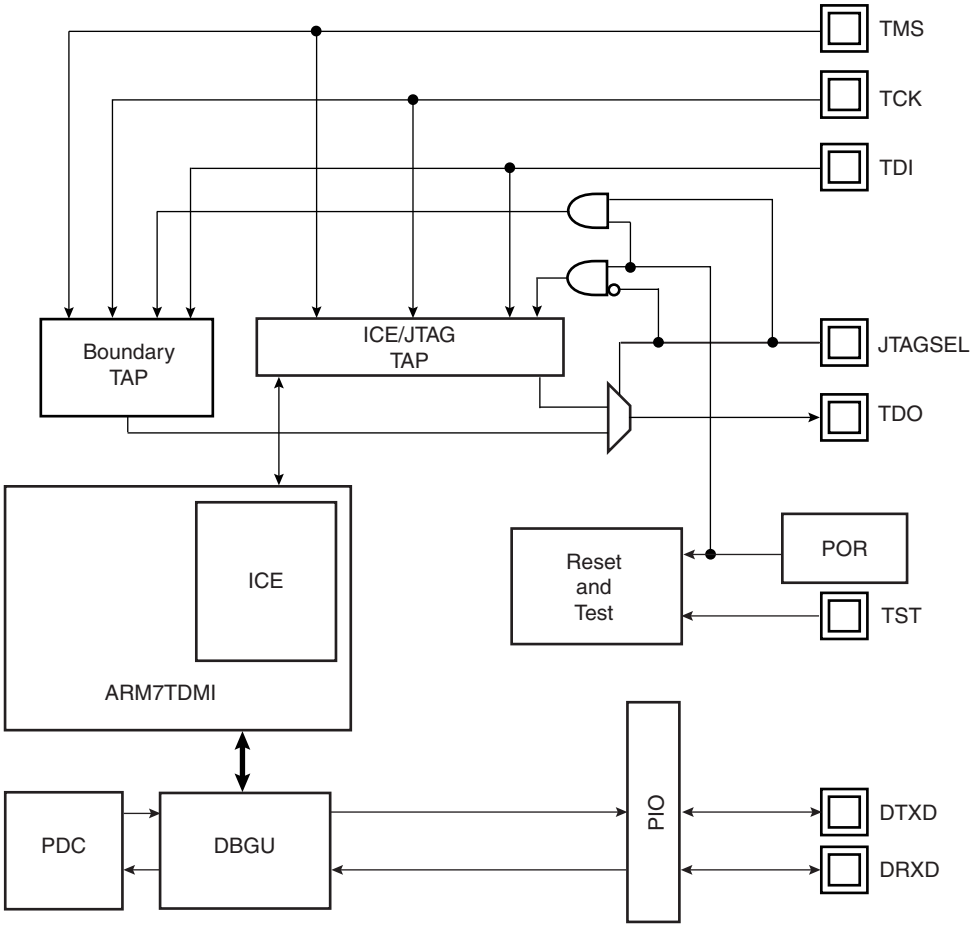
## 12.1 Overview

The AT91SAM7L Series Microcontrollers feature a number of complementary debug and test capabilities. A common JTAG/ICE (Embedded ICE) port is used for standard debugging functions, such as downloading code and single-stepping through programs. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

A set of dedicated debug and test input/output pins gives direct access to these capabilities from a PC-based test environment.

## 12.2 Block Diagram

**Figure 12-1.** Debug and Test Block Diagram

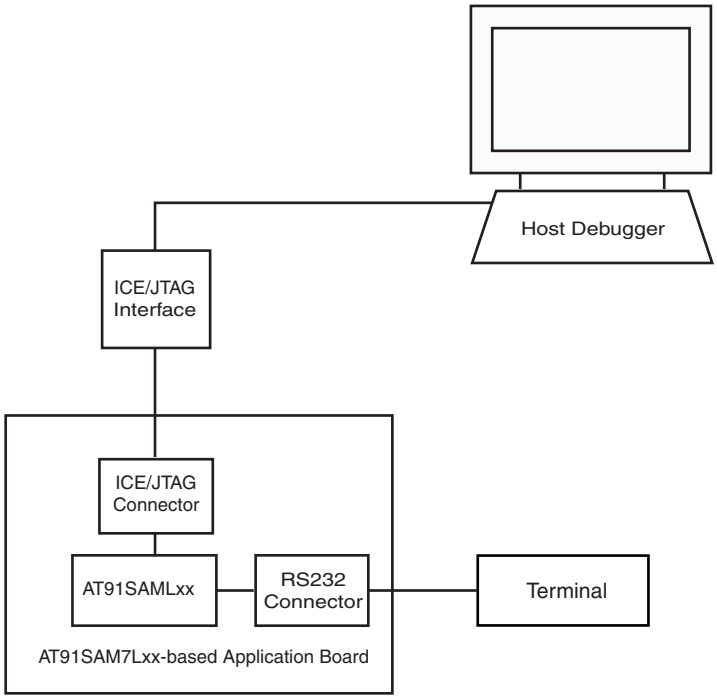


## 12.3 Application Examples

### 12.3.1 Debug Environment

Figure 12-2 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program.

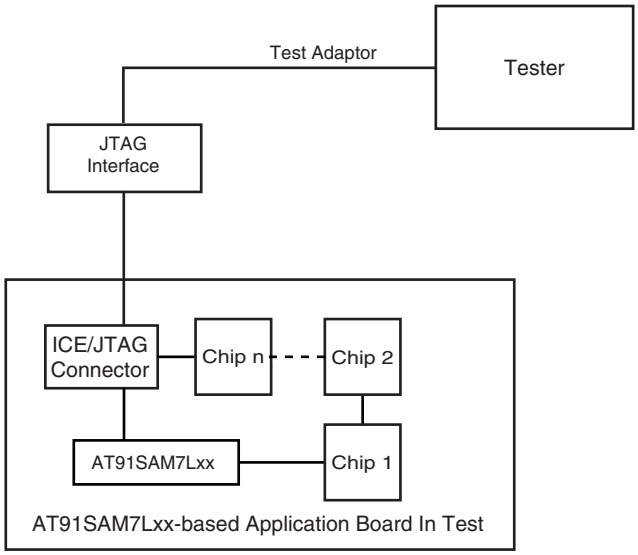
**Figure 12-2.** Application Debug Environment Example



### 12.3.2 Test Environment

Figure 12-3 shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

**Figure 12-3.** Application Test Environment Example




## 12.4 Debug and Test Pin Description

**Table 12-1.** Debug and Test Pin List

| Pin Name | Function | Type | Active Level |
|---|---|---|---|
| **Reset/Test** | | | |
| NRST | Microcontroller Reset | Input/Output | Low |
| TST | Test Mode Select | Input | High |
| **ICE and JTAG** | | | |
| TCK | Test Clock | Input | |
| TDI | Test Data In | Input | |
| TDO | Test Data Out | Output | |
| TMS | Test Mode Select | Input | |
| JTAGSEL | JTAG Selection | Input | |
| **Debug Unit** | | | |
| DRXD | Debug Receive Data | Input | |
| DTXD | Debug Transmit Data | Output | |

## 12.5 Functional Description

### 12.5.1 Test Pin

One dedicated pin, TST, is used to define the device operating mode. The user must make sure that this pin is tied at low level to ensure normal operating conditions. Other values associated with this pin are reserved for manufacturing test.

### 12.5.2 EmbeddedICE™ (Embedded In-circuit Emulator)

The ARM7TDMI EmbeddedICE is supported via the ICE/JTAG port.The internal state of the ARM7TDMI is examined through an ICE/JTAG port.

The ARM7TDMI processor contains hardware extensions for advanced debugging features:

- In halt mode, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM7TDMI registers. This data can be serially shifted out without affecting the rest of the system.
- In monitor mode, the JTAG interface is used to transfer data between the debugger and a simple monitor program running on the ARM7TDMI processor.

There are three scan chains inside the ARM7TDMI processor that support testing, debugging, and programming of the Embedded ICE. The scan chains are controlled by the ICE/JTAG port.

Embedded ICE mode is selected when JTAGSEL is low. It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed after JTAGSEL is changed.

For further details on the Embedded ICE, see the ARM7TDMI (Rev4) Technical Reference Manual (DDI0210B).

### 12.5.3 Debug Unit

The Debug Unit provides a two-pin (DXRD and TXRD) USART that can be used for several debug and trace purposes and offers an ideal means for in-situ programming solutions and debug monitor communication. Moreover, the association with two peripheral data controller channels permits packet handling of these tasks with processor time reduced to a minimum.

The Debug Unit also manages the interrupt handling of the COMMTX and COMMRX signals that come from the ICE and that trace the activity of the Debug Communication Channel.The Debug Unit allows blockage of access to the system through the ICE interface.

A specific register, the Debug Unit Chip ID Register, gives information about the product version and its internal configuration.

**Table 12-2.** AT91SAM7Lxx Chip IDs

| Chip Name | Chip ID |
|---|---|
| AT91SAM7L64 | 0x27330540 |
| AT91SAM7L128 | 0x27330740 |

For further details on the Debug Unit, see the Debug Unit section.

### 12.5.4 IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan allows pin-level access independent of the device packaging technology.

IEEE 1149.1 JTAG Boundary Scan is enabled when TST, JTAGSEL are high and CLKIN, FWUP and RNRSTB are tied low. VDDCORE must be externally supplied between 1.8V and 1.95V. The SAMPLE, EXTEST and BYPASS functions are implemented. In ICE debug mode, the ARM processor responds with a non-JTAG chip ID that identifies the processor to the ICE system. This is not IEEE 1149.1 JTAG-compliant.

It is not possible to switch directly between JTAG and ICE operations. A chip reset must be performed after JTAGSEL is changed.

A Boundary-scan Descriptor Language (BSDL) file is provided to set up test.

#### *12.5.4.1 JTAG Boundary-scan Register*

The Boundary-scan Register (BSR) contains 160 bits that correspond to active pins and associated control signals.

Each AT91SAM7Lxx input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad.

For more information, please refer to BDSL files which are available for the SAM7L Series.

### 12.5.5 ID Code Register

Access: Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VERSION | | | | PART NUMBER | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| PART NUMBER | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PART NUMBER | | | | MANUFACTURER IDENTITY | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MANUFACTURER IDENTITY | | | | | | | 1 |

- **VERSION[31:28]: Product Version Number**

Set to 0x0.

- **PART NUMBER[27:12]: Product Part Number**

| Chip Name | Chip ID |
|---|---|
| AT91SAM7L64 | 0x5B23 |
| AT91SAM7L128 | 0x5B1E |

- **MANUFACTURER IDENTITY[11:1]**

Set to 0x01F.

- **Bit[0] Required by IEEE Std. 1149.1.**

Set to 0x1.

| Chip Name | JTAG ID Code |
|---|---|
| AT91SAM7L64 | 05B2_303F |
| AT91SAM7L128 | 05B1_E03F |

# 13. Reset Controller (RSTC)

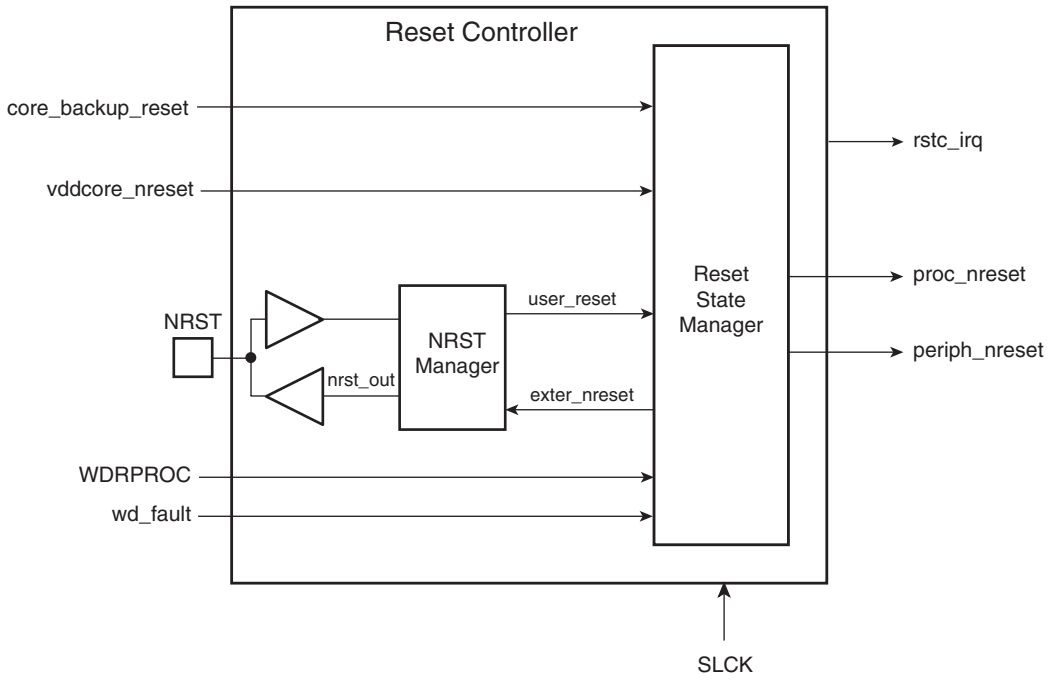
## 13.1 Overview

The Reset Controller (RSTC), based on a zero-power power-on reset cell, handles all the resets of the system without any external components. It reports which reset occurred last.

The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

## 13.2 Block Diagram

**Figure 13-1.** Reset Controller Block Diagram




## 13.3 Functional Description

### 13.3.1 Reset Controller Overview

The Reset Controller is made up of an NRST Manager and a Reset State Manager. It runs at Slow Clock and generates the following reset signals:

- proc_nreset: Processor reset line. It also resets the Watchdog Timer.
- periph_nreset: Affects the whole set of embedded peripherals.
- nrst_out: Drives the NRST pin.

These reset signals are asserted by the Reset Controller, either on external events or on software action. The Reset State Manager controls the generation of reset signals and provides a signal to the NRST Manager when an assertion of the NRST pin is required.

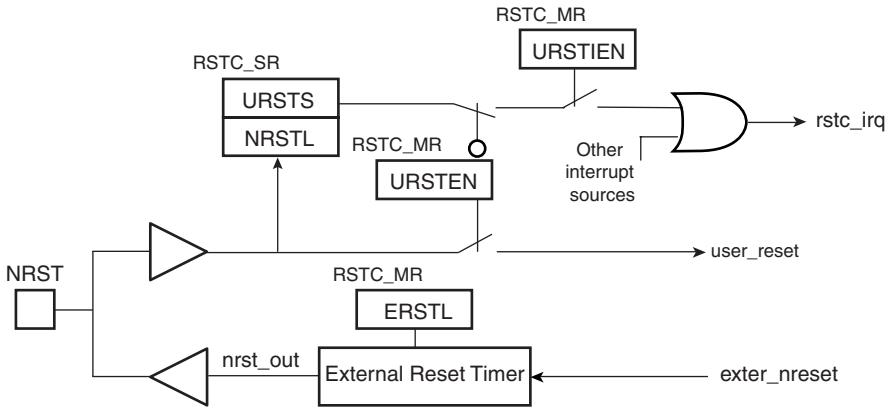
The NRST Manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDIO1, so that its configuration is saved as long as VDDIO1 is on.

### 13.3.2 NRST Manager

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 13-2 shows the block diagram of the NRST Manager.

**Figure 13-2.** NRST Manager




#### 13.3.2.1 *NRST Signal or Interrupt*

The NRST Manager samples the NRST pin at Slow Clock speed. When the line is detected low, a User Reset is reported to the Reset State Manager.

However, the NRST Manager can be programmed to not trigger a reset when an assertion of NRST occurs. Writing the bit URSTEN at 0 in RSTC_MR disables the User Reset trigger.

The level of the pin NRST can be read at any time in the bit NRSTL (NRST level) in RSTC_SR. As soon as the pin NRST is asserted, the bit URSTS in RSTC_SR is set. This bit clears only when RSTC_SR is read.

The Reset Controller can also be programmed to generate an interrupt instead of generating a reset. To do so, the bit URSTIEN in RSTC_MR must be written at 1.

#### 13.3.2.2 *NRST External Reset Control*

The Reset State Manager asserts the signal ext_nreset to assert the NRST pin. When this occurs, the “nrst_out” signal is driven low by the NRST Manager for a time programmed by the field ERSTL in RSTC_MR. This assertion duration, named EXTERNAL_RESET_LENGTH, lasts $2^{(ERSTL+1)}$ Slow Clock cycles. This gives the approximate duration of an assertion between 60 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

As the ERSTL field is within RSTC_MR register, which is backed-up, it can be used to shape the system power-up reset for devices requiring a longer startup time than the Slow Clock Oscillator.

Please note that the NRST output is in high impedance state when the chip is in OFF mode.

### 13.3.3 Brownout Manager

The Brownout manager is embedded within the Supply Controller, please refer to the Supply Controller section for a detailed description.

### 13.3.4 Reset States

The Reset State Manager handles the different reset sources and generates the internal reset signals. It reports the reset status in the field RSTTYP of the Status Register (RSTC_SR). The update of the field RSTTYP is performed when the processor reset is released.

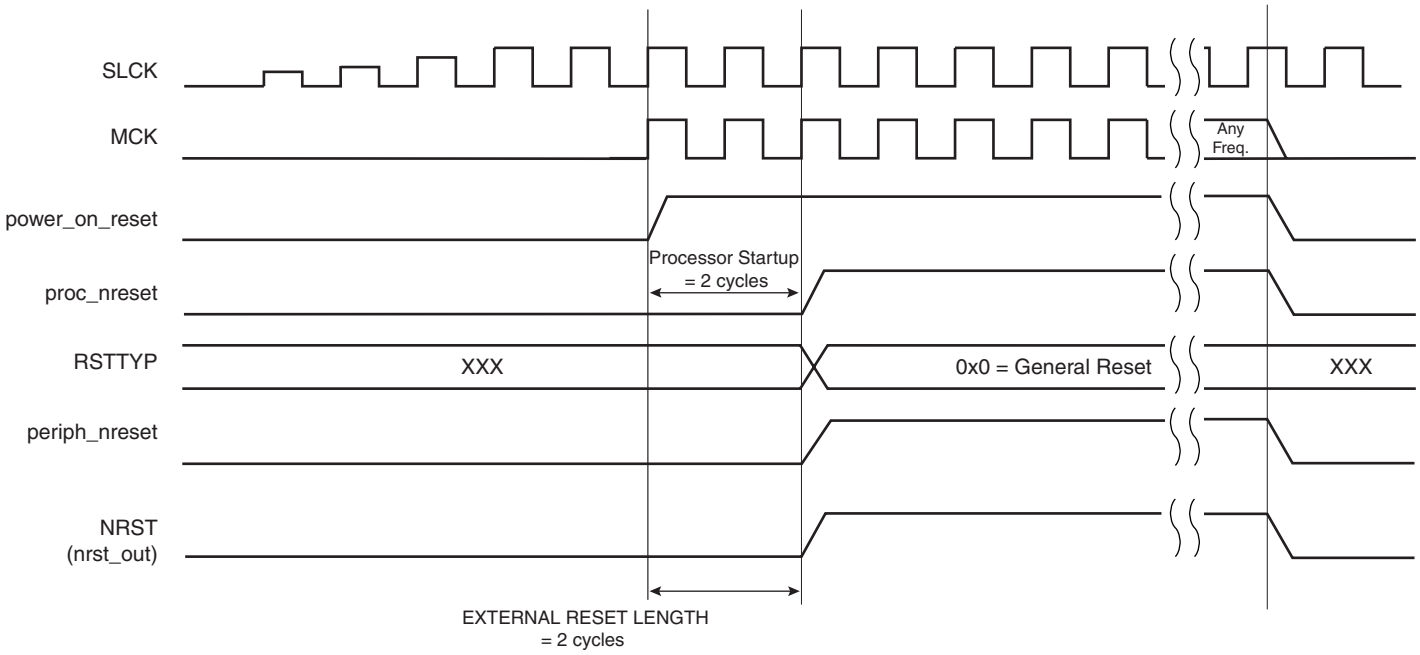
#### *13.3.4.1 General Reset*

A general reset occurs when a Power-on-reset is detected, an Asynchronous Master Reset (NRSTB pin ) is requested, a Brownout or a Voltage regulation loss is detected by the Supply controller. The vddcore_nreset signal is asserted by the Supply Controller when a general reset occurs.

All the reset signals are released and the field RSTTYP in RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises 2 cycles after the vddcore_nreset, as ERSTL defaults at value 0x0.

Figure 13-3 shows how the General Reset affects the reset signals.

**Figure 13-3.** General Reset State



### 13.3.4.2 Backup Reset

A Backup reset occurs when the chip returns from Backup mode. The core_backup_reset signal is asserted by the Supply Controller when a Backup reset occurs.

The field RSTTYP in RSTC_SR is updated to report a Backup Reset.

### 13.3.4.3 User Reset

The User Reset is entered when a low level is detected on the NRST pin and the bit URSTEN in RSTC_MR is at 1. The NRST input signal is resynchronized with SLCK to insure proper behavior of the system.

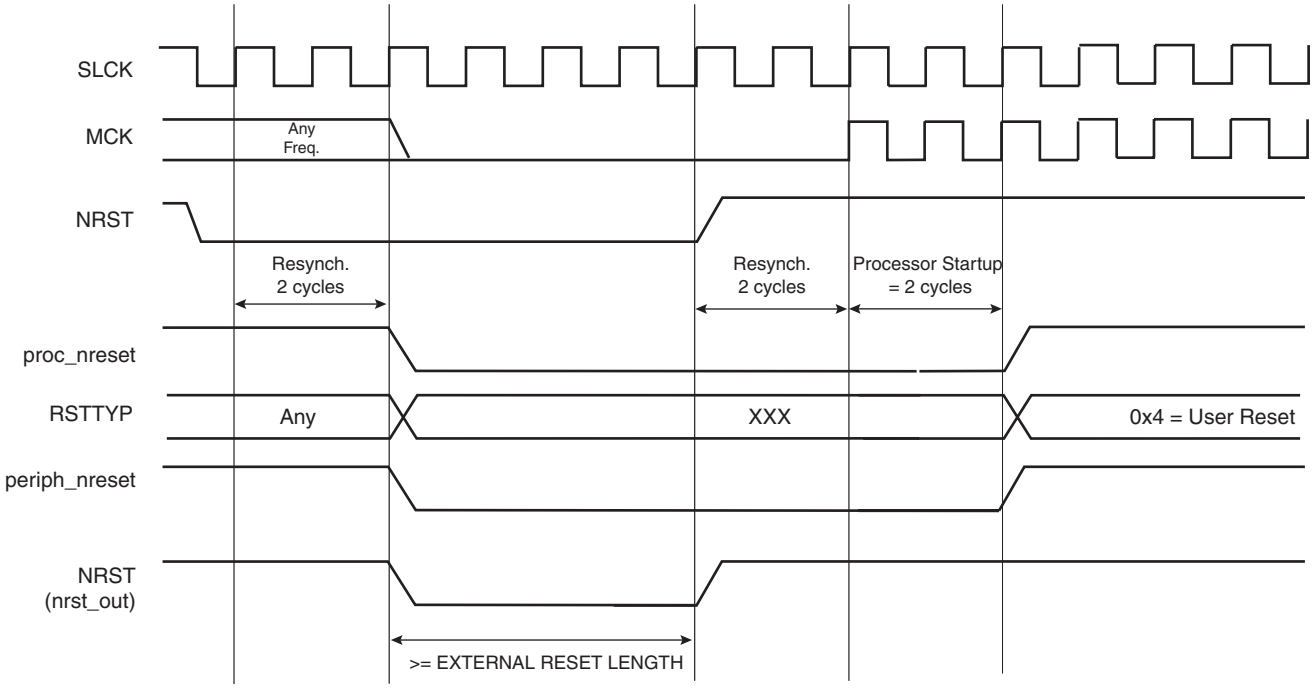
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 3-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

When the processor reset signal is released, the RSTTYP field of the Status Register (RSTC_SR) is loaded with the value 0x4, indicating a User Reset.

The NRST Manager guarantees that the NRST line is asserted for EXTERNAL_RESET_LENGTH Slow Clock cycles, as programmed in the field ERSTL. However, if NRST does not rise after EXTERNAL_RESET_LENGTH because it is driven low externally, the internal reset lines remain asserted until NRST actually rises.

**Figure 13-4.** User Reset State



#### 13.3.4.4 Software Reset

The Reset Controller offers several commands used to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- PROCRST: Writing PROCRST at 1 resets the processor and the watchdog timer.
- PERRST: Writing PERRST at 1 resets all the embedded peripherals, including the memory system, and, in particular, the Remap Command. The Peripheral Reset is generally used for debug purposes.
- EXTRST: Writing EXTRST at 1 asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts 3 Slow Clock cycles.

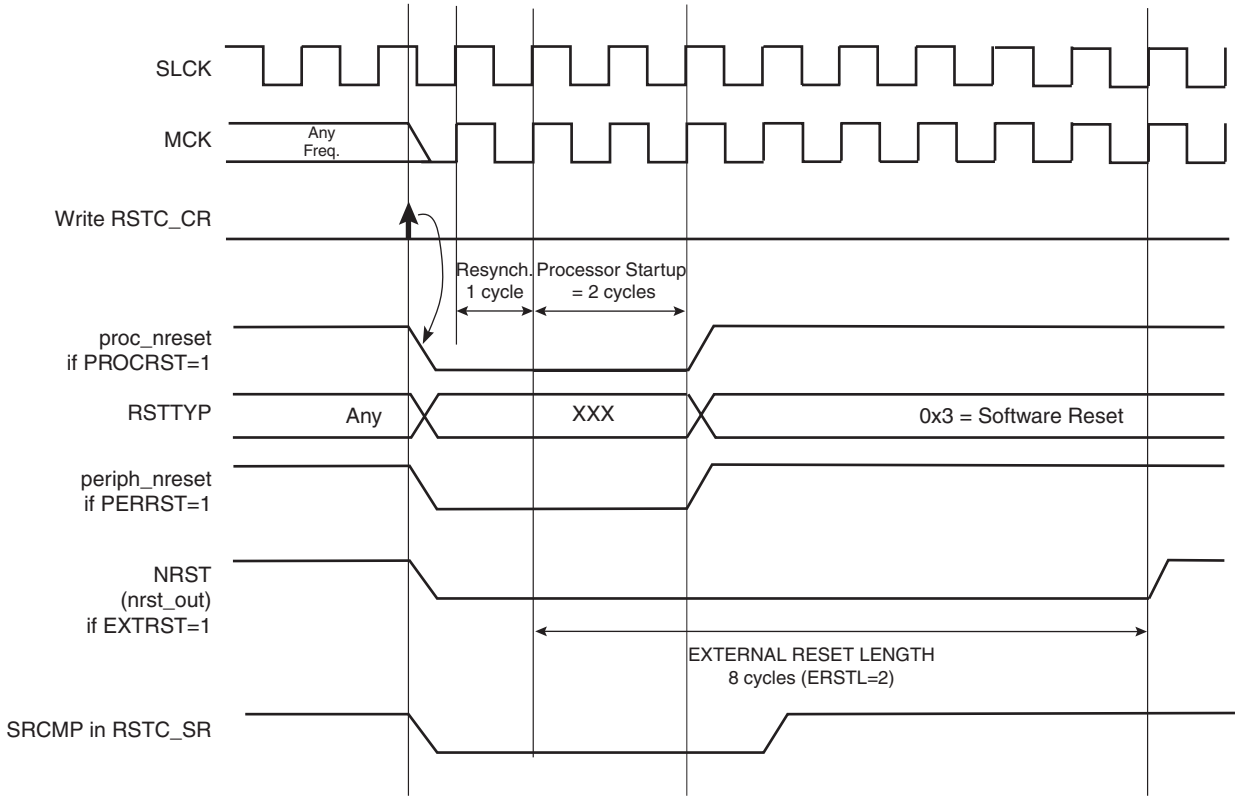
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

If EXTRST is set, the nrst_out signal is asserted depending on the programming of the field ERSTL. However, the resulting falling edge on NRST does not lead to a User Reset.

If and only if the PROCRST bit is set, the Reset Controller reports the software status in the field RSTTYP of the Status Register (RSTC_SR). Other Software Resets are not reported in RSTTYP.

As soon as a software operation is detected, the bit SRCMP (Software Reset Command in Progress) is set in the Status Register (RSTC_SR). It is cleared as soon as the software reset is left. No other software reset can be performed while the SRCMP bit is set, and writing any value in RSTC_CR has no effect.

**Figure 13-5.** Software Reset




### *13.3.4.5 Watchdog Reset*

The Watchdog Reset is entered when a watchdog fault occurs. This state lasts 3 Slow Clock cycles.

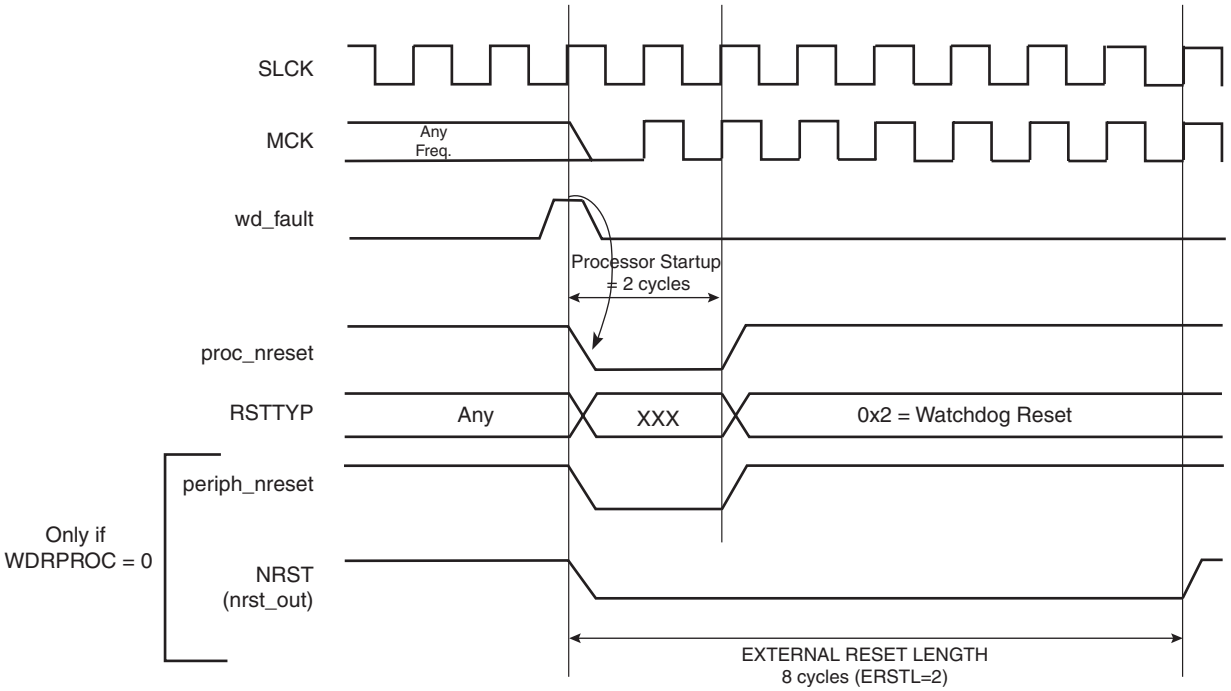
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- If WDRPROC is 0, the Processor Reset and the Peripheral Reset are asserted. The NRST line is also asserted, depending on the programming of the field ERSTL. However, the resulting low level on NRST does not result in a User Reset state.
- If WDRPROC = 1, only the processor reset is asserted.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDRSTEN is set, the Watchdog Timer is always reset after a Watchdog Reset, and the Watchdog is enabled by default and with a period set to a maximum.

When the WDRSTEN in WDT_MR bit is reset, the watchdog fault has no impact on the reset controller.

**Figure 13-6.** Watchdog Reset




### 13.3.5 Reset State Priorities

The Reset State Manager manages the following priorities between the different reset sources, given in descending order:

- General Reset
- Backup Reset
- Watchdog Reset
- Software Reset
- User Reset

Particular cases are listed below:

- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST has no effect.
- When in Watchdog Reset:
  - The processor reset is active and so a Software Reset cannot be programmed.
  - A User Reset cannot be entered.

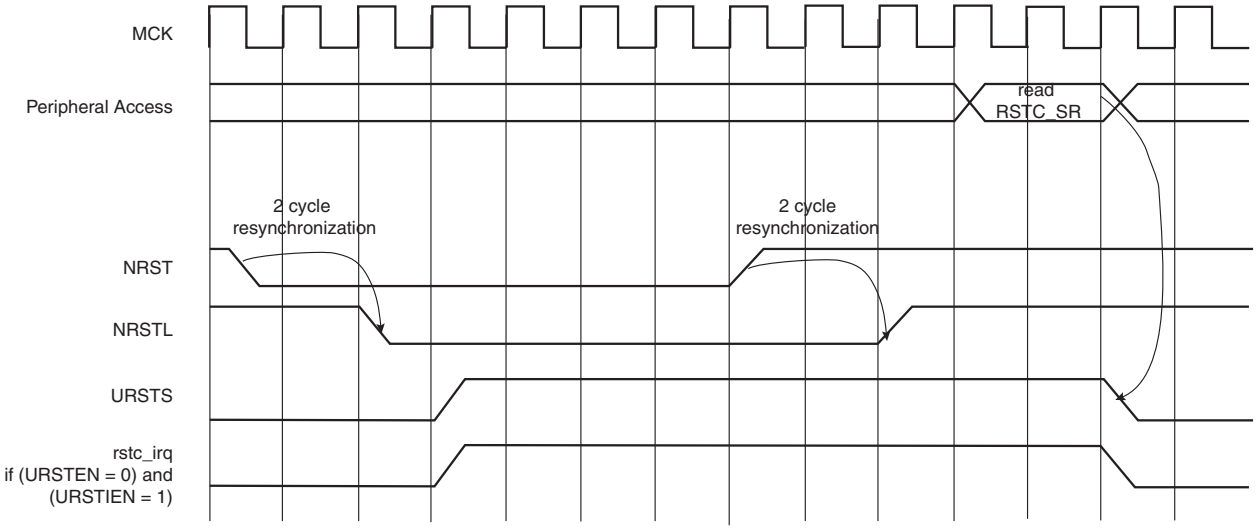
### 13.3.6 Reset Controller Status Register

The Reset Controller status register (RSTC_SR) provides several status fields:

- RSTTYP field: This field gives the type of the last reset, as explained in previous sections.

- SRCMP bit: This field indicates that a Software Reset Command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.
- NRSTL bit: The NRSTL bit of the Status Register gives the level of the NRST pin sampled on each MCK rising edge.
- URSTS bit: A high-to-low transition of the NRST pin sets the URSTS bit of the RSTC_SR register. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 13-7). If the User Reset is disabled (URSTEN = 0) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR register, the URSTS bit triggers an interrupt. Reading the RSTC_SR status register resets the URSTS bit and clears the interrupt.

**Figure 13-7.** Reset Controller Status and Interrupt



## 13.4 Reset Controller (RSTC) User Interface

**Table 13-1.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | RSTC_CR | Write-only | - |
| 0x04 | Status Register | RSTC_SR | Read-only | 0x0000_0000 |
| 0x08 | Mode Register | RSTC_MR | Read-write | 0x0000_0000 |

### 13.4.1 Reset Controller Control Register

**Name:** RSTC_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | EXTRST | PERRST | – | PROCRST |

- **PROCRST: Processor Reset**

0 = No effect.

1 = If KEY is correct, resets the processor.

- **PERRST: Peripheral Reset**

0 = No effect.

1 = If KEY is correct, resets the peripherals.

- **EXTRST: External Reset**

0 = No effect.

1 = If KEY is correct, asserts the NRST pin.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

### 13.4.2 Reset Controller Status Register

**Name:** RSTC_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | SRCMP | NRSTL |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | RSTTYP | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | URSTS |

- **URSTS: User Reset Status**

0 = No high-to-low edge on NRST happened since the last read of RSTC_SR.

1 = At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

Reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

| RSTTYP | | | Reset Type | Comments |
|---|---|---|---|---|
| 0 | 0 | 0 | General Reset | First power-up Reset (Power-on Reset or NRSTB asserted) |
| 0 | 0 | 1 | Backup Reset | Return from Backup mode |
| 0 | 1 | 0 | Watchdog Reset | Watchdog fault occurred |
| 0 | 1 | 1 | Software Reset | Processor reset required by the software |
| 1 | 0 | 0 | User Reset | NRST pin detected low |

- **NRSTL: NRST Pin Level**

Registers the NRST Pin Level at Master Clock (MCK).

- **SRCMP: Software Reset Command in Progress**

0 = No software command is being performed by the reset controller. The reset controller is ready for a software command.

1 = A software reset command is being performed by the reset controller. The reset controller is busy.

### 13.4.3 Reset Controller Mode Register

**Name:** RSTC_MR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | ERSTL | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | | URSTIEN | – | – | – | URSTEN |

- **URSTEN: User Reset Enable**

0 = The detection of a low level on the pin NRST does not generate a User Reset.

1 = The detection of a low level on the pin NRST triggers a User Reset.

- **URSTIEN: User Reset Interrupt Enable**

0 = USRTS bit in RSTC_SR at 1 has no effect on rstc_irq.

1 = USRTS bit in RSTC_SR at 1 asserts rstc_irq if URSTEN = 0.

- **ERSTL: External Reset Length**

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ Slow Clock cycles. This allows assertion duration to be programmed between 60 µs and 2 seconds.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

# 14. Real-time Clock (RTC)

## 14.1 Overview

The Real-time Clock (RTC) peripheral is designed for very low power consumption.

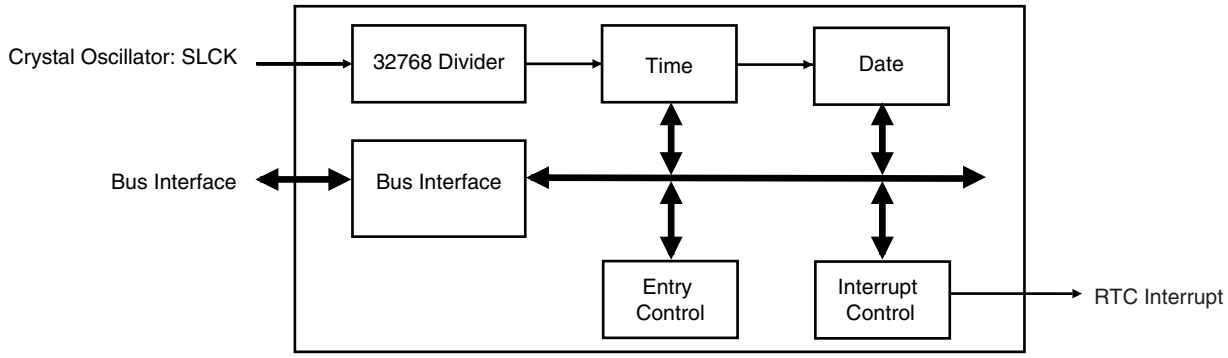
It combines a complete time-of-day clock with alarm and a two-hundred-year Gregorian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

## 14.2 Block Diagram

**Figure 14-1.** RTC Block Diagram




## 14.3 Product Dependencies

### 14.3.1 Power Management

The Real-time Clock is continuously clocked at 32768 Hz. The Power Management Controller has no effect on RTC behavior.

### 14.3.2 Interrupt

The RTC Interrupt is connected to interrupt source 1 (IRQ1) of the advanced interrupt controller. This interrupt line is due to the OR-wiring of the system peripheral interrupt lines (System Timer, Real Time Clock, Power Management Controller, Memory Controller, etc.). When a system interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading the status registers of the above system peripherals successively.

## 14.4 Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds.

The valid year range is 1900 to 2099, a two-hundred-year Gregorian calendar achieving full Y2K compliance.

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years, including year 2000). This is correct up to the year 2099.

After hardware reset, the calendar is initialized to Thursday, January 1, 1998.

### 14.4.1 Reference Clock

The reference clock is Slow Clock (SLCK). It can be driven by the Atmel cell OSC55 or OSC56 (or an equivalent cell) and an external 32.768 kHz crystal.

During low power modes of the processor (idle mode), the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

### 14.4.2 Timing

The RTC is updated in real time at one-second intervals in normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

### 14.4.3 Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

Depending on the combination of fields enabled, a large number of possibilities are available to the user ranging from minutes to 365/366 days.

### 14.4.4 Error Checking

Verification on user interface data is performed when accessing the century, year, month, date, day, hours, minutes, seconds and alarms. A check is performed on illegal BCD entries such as illegal date of the month with regard to the year and century configured.

If one of the time fields is not correct, the data is not loaded into the register/counter and a flag is set in the validity register. The user can not reset this flag. It is reset as soon as an acceptable value is programmed. This avoids any further side effects in the hardware. The same procedure is done for the alarm.

The following checks are performed:

1. Century (check if it is in range 19 - 20)
2. Year (BCD entry check)
3. Date (check range 01 - 31)
4. Month (check if it is in BCD range 01 - 12, check validity regarding “date”)
5. Day (check range 1 - 7)
6. Hour (BCD checks: in 24-hour mode, check range 00 - 23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01 - 12)
7. Minute (check BCD and range 00 - 59)
8. Second (check BCD and range 00 - 59)

Note: If the 12-hour mode is selected by means of the RTC_MODE register, a 12-hour value can be programmed and the returned value on RTC_TIME will be the corresponding 24-hour value. The entry control checks the value of the AM/PM indicator (bit 22 of RTC_TIME register) to determine the range to be checked.

### 14.4.5 Updating Time/Calendar

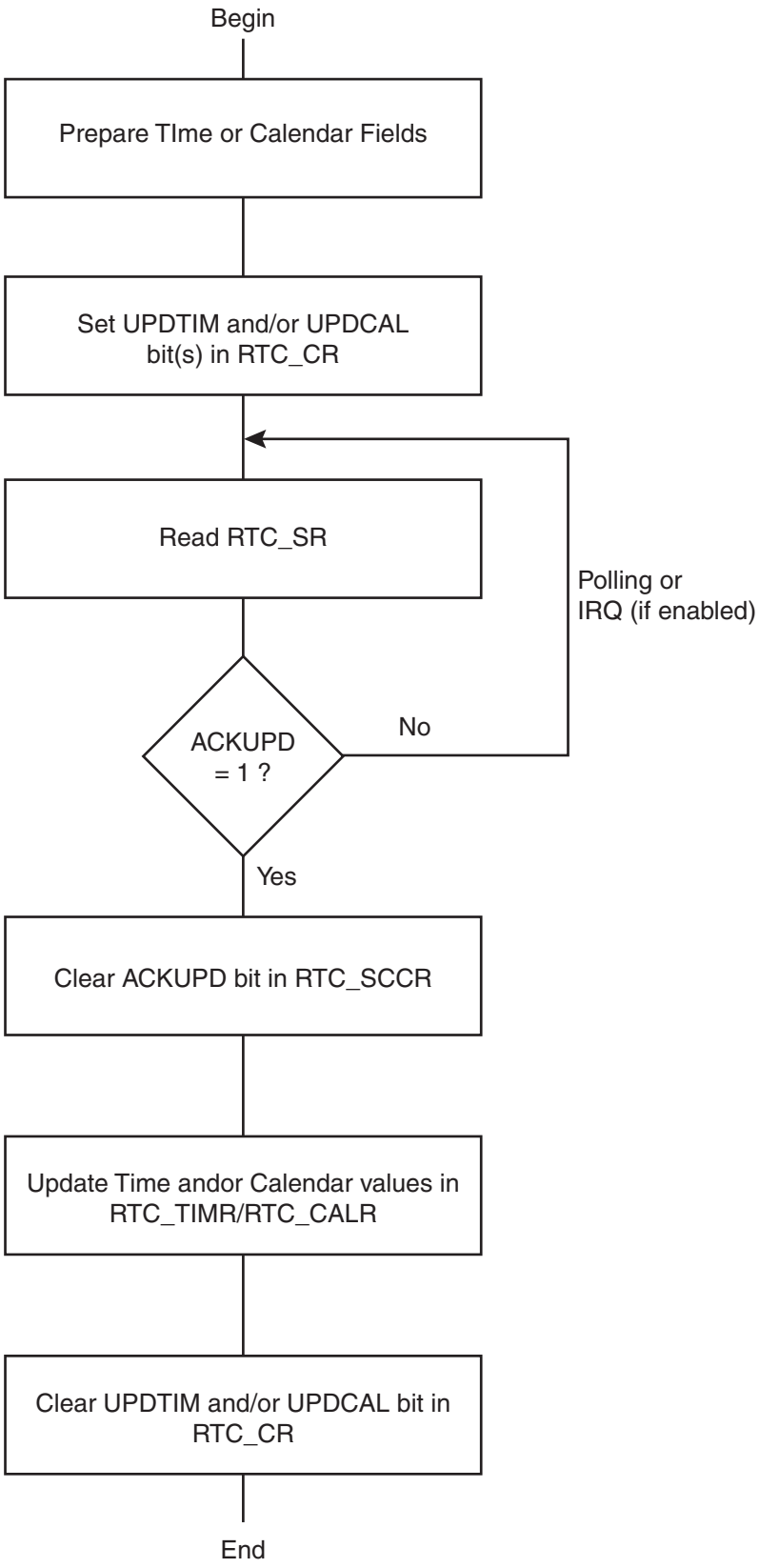
To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register. Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

Then the user must poll or wait for the interrupt (if enabled) of bit ACKUPD in the Status Register. Once the bit reads 1, it is mandatory to clear this flag by writing the corresponding bit in RTC_SCCR. The user can now write to the appropriate Time and Calendar register.

Once the update is finished, the user must reset (0) UPDTIM and/or UPDCAL in the Control

When entering programming mode of the calendar fields, the time fields remain enabled. When entering the programming mode of the time fields, both time and calendar fields are stopped. This is due to the location of the calendar logic circuity (downstream for low-power considerations). It is highly recommended to prepare all the fields to be updated before entering programming mode. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC_CR (Control Register) before setting these bits again. This is done by waiting for the SEC flag in the Status Register before setting UPDTIM/UPDCAL bit. After resetting UPDTIM/UPDCAL, the SEC flag must also be cleared.

**Figure 14-2.** Update Sequence



## 14.5 Real-time Clock (RTC) User Interface

**Table 14-1.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | RTC_CR | Read-write | 0x0 |
| 0x04 | Mode Register | RTC_MR | Read-write | 0x0 |
| 0x08 | Time Register | RTC_TIMR | Read-write | 0x0 |
| 0x0C | Calendar Register | RTC_CALR | Read-write | 0x01819819 |
| 0x10 | Time Alarm Register | RTC_TIMALR | Read-write | 0x0 |
| 0x14 | Calendar Alarm Register | RTC_CALALR | Read-write | 0x01010000 |
| 0x18 | Status Register | RTC_SR | Read-only | 0x0 |
| 0x1C | Status Clear Command Register | RTC_SCCR | Write-only | --- |
| 0x20 | Interrupt Enable Register | RTC_IER | Write-only | --- |
| 0x24 | Interrupt Disable Register | RTC_IDR | Write-only | --- |
| 0x28 | Interrupt Mask Register | RTC_IMR | Read-only | 0x0 |
| 0x2C | Valid Entry Register | RTC_VER | Read-only | 0x0 |
| 0xFC | Reserved Register | – | – | – |

### 14.5.1 RTC Control Register

**Name:** RTC_CR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | CALEVSEL | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | TIMEVSEL | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | UPDCAL | UPDTIM |

- **UPDTIM: Update Request Time Register**

0 = No effect.

1 = Stops the RTC time counting.

Time counting consists of second, minute and hour counters. Time counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the Status Register.

- **UPDCAL: Update Request Calendar Register**

0 = No effect.

1 = Stops the RTC calendar counting.

Calendar counting consists of day, date, month, year and century counters. Calendar counters can be programmed once this bit is set.

- **TIMEVSEL: Time Event Selection**

The event that generates the flag TIMEV in RTC_SR (Status Register) depends on the value of TIMEVSEL.

0 = Minute change.

1 = Hour change.

2 = Every day at midnight.

3 = Every day at noon.

- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL.

0 = Week change (every Monday at time 00:00:00).

1 = Month change (every 01 of each month at time 00:00:00).

2, 3 = Year change (every January 1 at time 00:00:00).

### 14.5.2 RTC Mode Register

**Name:** RTC_MR

**Access Type:** Read-write

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

- **HRMOD: 12-/24-hour Mode**

0 = 24-hour mode is selected.

1 = 12-hour mode is selected.

All non-significant bits read zero.

### 14.5.3 RTC Time Register

**Name:** RTC_TIMR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | AMPM | HOUR | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | MIN | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | SEC | | | | | | |

- **SEC: Current Second**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1 - 12 (BCD) in 12-hour mode or 0 - 23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0 = AM.

1 = PM.

All non-significant bits read zero.

### 14.5.4 RTC Calendar Register

**Name:** RTC_CALR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | DATE | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| DAY | | | MONTH | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| YEAR | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | CENT | | | | | | |

- **CENT: Current Century**

The range that can be set is 19 - 20 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **YEAR: Current Year**

The range that can be set is 00 - 99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MONTH: Current Month**

The range that can be set is 01 - 12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **DAY: Current Day in Current Week**

The range that can be set is 1 - 7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

- **DATE: Current Day in Current Month**

The range that can be set is 01 - 31 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

All non-significant bits read zero.

### 14.5.5 RTC Time Alarm Register

**Name:** RTC_TIMALR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HOUREN | AMPM | HOUR | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MINEN | MIN | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SECEN | SEC | | | | | | |

- **SEC: Second Alarm**

This field is the alarm field corresponding to the BCD-coded second counter.

- **SECEN: Second Alarm Enable**

0 = The second-matching alarm is disabled.

1 = The second-matching alarm is enabled.

- **MIN: Minute Alarm**

This field is the alarm field corresponding to the BCD-coded minute counter.

- **MINEN: Minute Alarm Enable**

0 = The minute-matching alarm is disabled.

1 = The minute-matching alarm is enabled.

- **HOUR: Hour Alarm**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **AMPM: AM/PM Indicator**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **HOUREN: Hour Alarm Enable**

0 = The hour-matching alarm is disabled.

1 = The hour-matching alarm is enabled.

### 14.5.6 RTC Calendar Alarm Register

**Name:** RTC_CALALR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DATEEN | – | DATE | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MTHEN | – | – | MONTH | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | – |

- **MONTH: Month Alarm**

This field is the alarm field corresponding to the BCD-coded month counter.

- **MTHEN: Month Alarm Enable**

0 = The month-matching alarm is disabled.

1 = The month-matching alarm is enabled.

- **DATE: Date Alarm**

This field is the alarm field corresponding to the BCD-coded date counter.

- **DATEEN: Date Alarm Enable**

0 = The date-matching alarm is disabled.

1 = The date-matching alarm is enabled.

### 14.5.7 RTC Status Register

**Name:** RTC_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | CALEV | TIMEV | SEC | ALARM | ACKUPD |

- **ACKUPD: Acknowledge for Update**

0 = Time and calendar registers cannot be updated.

1 = Time and calendar registers can be updated.

- **ALARM: Alarm Flag**

0 = No alarm matching condition occurred.

1 = An alarm matching condition has occurred.

- **SEC: Second Event**

0 = No second event has occurred since the last clear.

1 = At least one second event has occurred since the last clear.

- **TIMEV: Time Event**

0 = No time event has occurred since the last clear.

1 = At least one time event has occurred since the last clear.

The time event is selected in the TIMEVSEL field in RTC_CTRL (Control Register) and can be any one of the following events: minute change, hour change, noon, midnight (day change).

- **CALEV: Calendar Event**

0 = No calendar event has occurred since the last clear.

1 = At least one calendar event has occurred since the last clear.

The calendar event is selected in the CALEVSEL field in RTC_CR and can be any one of the following events: week change, month change and year change.

### 14.5.8 RTC Status Clear Command Register

**Name:** RTC_SCCR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | CALCLR | TIMCLR | SECCLR | ALRCLR | ACKCLR |

- **ACKCLR: Acknowledge Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **ALRCLR: Alarm Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **SECCLR: Second Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **TIMCLR: Time Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

- **CALCLR: Calendar Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC_SR).

### 14.5.9 RTC Interrupt Enable Register

**Name:** RTC_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | CALEN | TIMEN | SECEN | ALREN | ACKEN |

- **ACKEN: Acknowledge Update Interrupt Enable**

0 = No effect.

1 = The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0 = No effect.

1 = The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0 = No effect.

1 = The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0 = No effect.

1 = The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0 = No effect.

- 1 = The selected calendar event interrupt is enabled.

### 14.5.10 RTC Interrupt Disable Register

**Name:** RTC_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | CALDIS | TIMDIS | SECDIS | ALRDIS | ACKDIS |

- **ACKDIS: Acknowledge Update Interrupt Disable**

0 = No effect.

1 = The acknowledge for update interrupt is disabled.

- **ALRDIS: Alarm Interrupt Disable**

0 = No effect.

1 = The alarm interrupt is disabled.

- **SECDIS: Second Event Interrupt Disable**

0 = No effect.

1 = The second periodic interrupt is disabled.

- **TIMDIS: Time Event Interrupt Disable**

0 = No effect.

1 = The selected time event interrupt is disabled.

- **CALDIS: Calendar Event Interrupt Disable**

0 = No effect.

1 = The selected calendar event interrupt is disabled.

### 14.5.11 RTC Interrupt Mask Register

**Name:** RTC_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | CAL | TIM | SEC | ALR | ACK |

- **ACK: Acknowledge Update Interrupt Mask**

0 = The acknowledge for update interrupt is disabled.

1 = The acknowledge for update interrupt is enabled.

- **ALR: Alarm Interrupt Mask**

0 = The alarm interrupt is disabled.

1 = The alarm interrupt is enabled.

- **SEC: Second Event Interrupt Mask**

0 = The second periodic interrupt is disabled.

1 = The second periodic interrupt is enabled.

- **TIM: Time Event Interrupt Mask**

0 = The selected time event interrupt is disabled.

1 = The selected time event interrupt is enabled.

- **CAL: Calendar Event Interrupt Mask**

0 = The selected calendar event interrupt is disabled.

1 = The selected calendar event interrupt is enabled.

## 14.6 RTC Valid Entry Register

**Name:** RTC_VER

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | NVCALALR | NVTIMALR | NVCAL | NVTIM |

- **NVTIM: Non-valid Time**

0 = No invalid data has been detected in RTC_TIMR (Time Register).

1 = RTC_TIMR has contained invalid data since it was last programmed.

- **NVCAL: Non-valid Calendar**

0 = No invalid data has been detected in RTC_CALR (Calendar Register).

1 = RTC_CALR has contained invalid data since it was last programmed.

- **NVTIMALR: Non-valid Time Alarm**

0 = No invalid data has been detected in RTC_TIMALR (Time Alarm Register).

1 = RTC_TIMALR has contained invalid data since it was last programmed.

- **NVCALALR: Non-valid Calendar Alarm**

0 = No invalid data has been detected in RTC_CALALR (Calendar Alarm Register).

1 = RTC_CALALR has contained invalid data since it was last programmed.

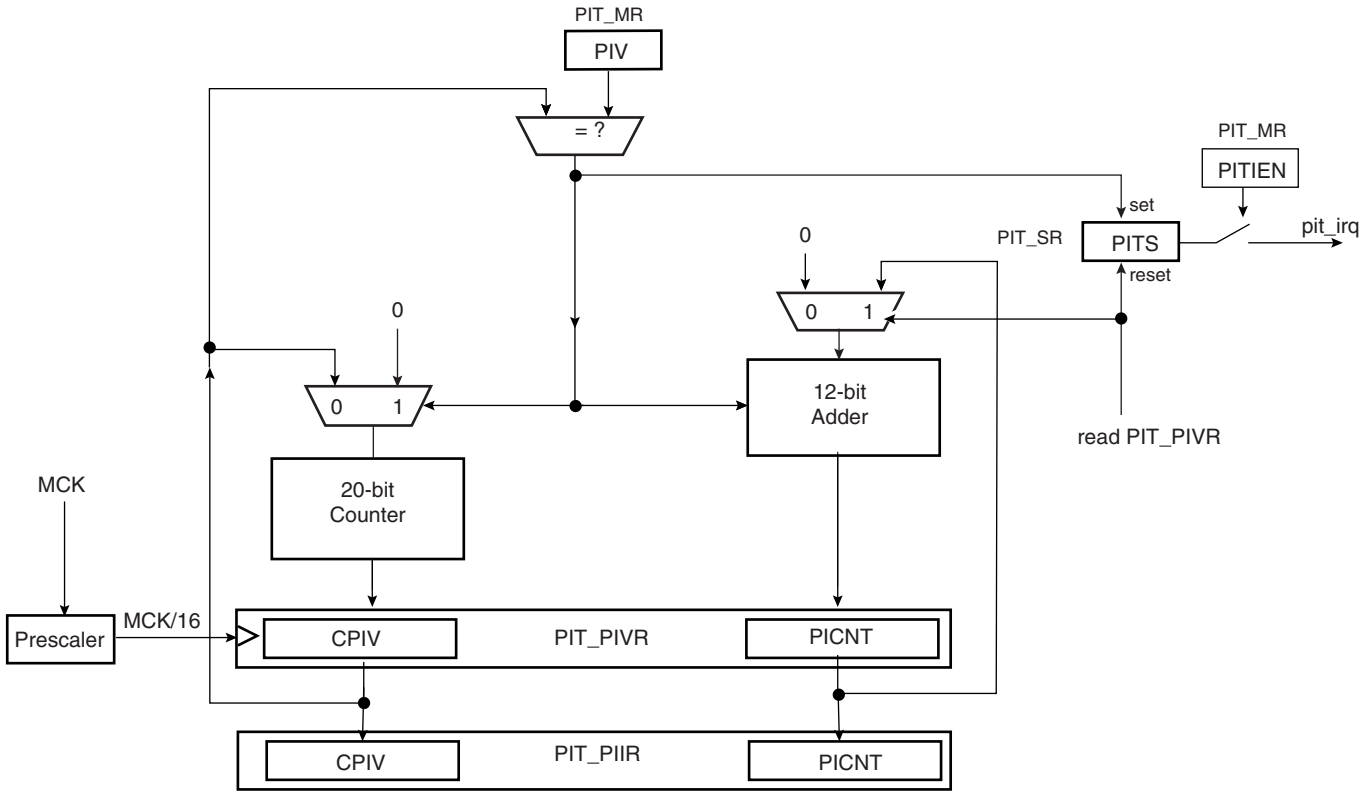
# 15. Periodic Interval Timer (PIT)

## 15.1 Overview

The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

## 15.2 Block Diagram

**Figure 15-1.** Periodic Interval Timer



## 15.3 Functional Description

The Periodic Interval Timer aims at providing periodic interrupts for use by operating systems.

The PIT provides a programmable overflow counter and a reset-on-read feature. It is built around two counters: a 20-bit CPIV counter and a 12-bit PICNT counter. Both counters work at Master Clock /16.

The first 20-bit CPIV counter increments from 0 up to a programmable overflow value set in the field PIV of the Mode Register (PIT_MR). When the counter CPIV reaches this value, it resets to 0 and increments the Periodic Interval Counter, PICNT. The status bit PITS in the Status Register (PIT_SR) rises and triggers an interrupt, provided the interrupt is enabled (PITIEN in PIT_MR).

Writing a new PIV value in PIT_MR does not reset/restart the counters.

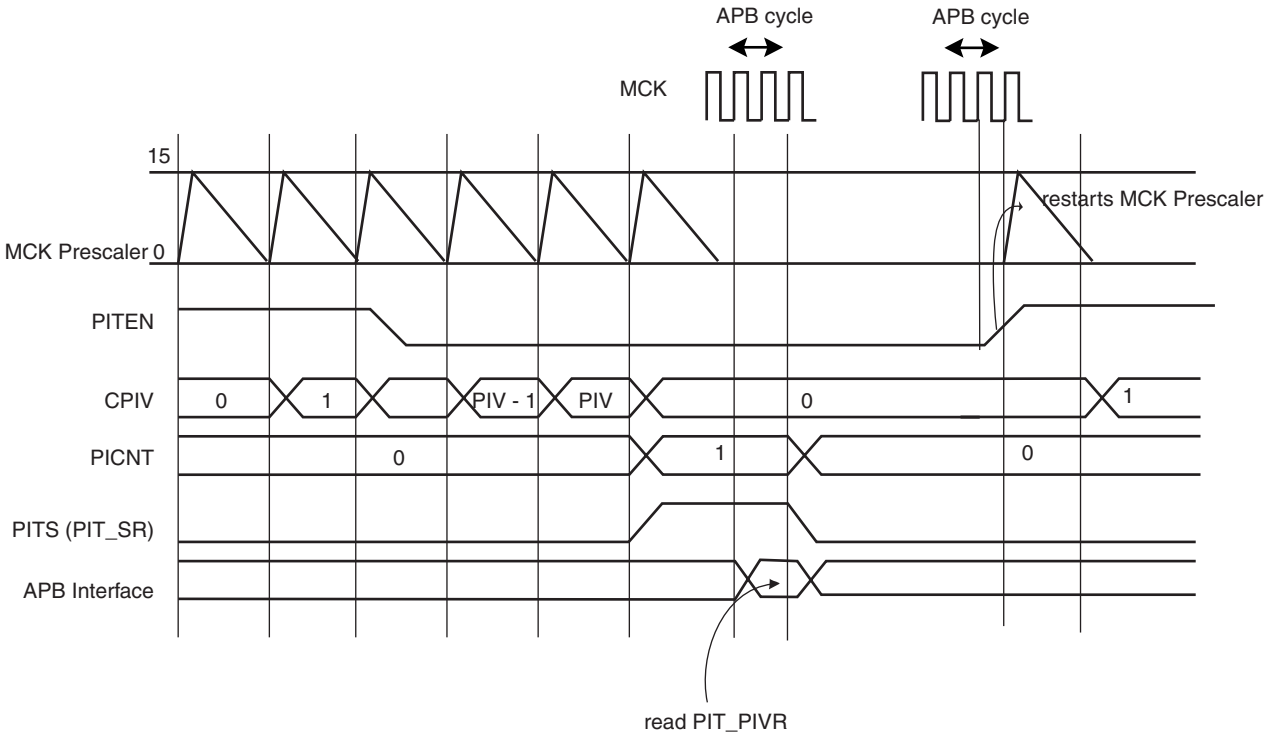
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

When CPIV and PICNT values are obtained by reading the Periodic Interval Image Register (PIT_PIIR), there is no effect on the counters CPIV and PICNT, nor on the bit PITS. For example, a profiler can read PIT_PIIR without clearing any pending interrupt, whereas a timer interrupt clears the interrupt by reading PIT_PIVR.

The PIT may be enabled/disabled using the PITEN bit in the PIT_MR register (disabled on reset). The PITEN bit only becomes effective when the CPIV value is 0. Figure 15-2 illustrates the PIT counting. After the PIT Enable bit is reset (PITEN= 0), the CPIV goes on counting until the PIV value is reached, and is then reset. PIT restarts counting, only if the PITEN is set again.

The PIT is stopped when the core enters debug state.

**Figure 15-2.** Enabling/Disabling PIT with PITEN



## 15.4 Periodic Interval Timer (PIT) User Interface

**Table 15-1.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Mode Register | PIT_MR | Read-write | 0x000F_FFFF |
| 0x04 | Status Register | PIT_SR | Read-only | 0x0000_0000 |
| 0x08 | Periodic Interval Value Register | PIT_PIVR | Read-only | 0x0000_0000 |
| 0x0C | Periodic Interval Image Register | PIT_PIIR | Read-only | 0x0000_0000 |

### 15.4.1 Periodic Interval Timer Mode Register

**Register Name:** PIT_MR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | PITIEN | PITEN |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | PIV | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PIV | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIV | | | | | | | |

- **PIV: Periodic Interval Value**

Defines the value compared with the primary 20-bit counter of the Periodic Interval Timer (CPIV). The period is equal to (PIV + 1).

- **PITEN: Period Interval Timer Enabled**

0 = The Periodic Interval Timer is disabled when the PIV value is reached.

1 = The Periodic Interval Timer is enabled.

- **PITIEN: Periodic Interval Timer Interrupt Enable**

0 = The bit PITS in PIT_SR has no effect on interrupt.

1 = The bit PITS in PIT_SR asserts interrupt.

### 15.4.2 Periodic Interval Timer Status Register

**Register Name:** PIT_SR

**Access Type:** Read-only

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

- **PITS: Periodic Interval Timer Status**

0 = The Periodic Interval timer has not reached PIV since the last read of PIT_PIVR.

1 = The Periodic Interval timer has reached PIV since the last read of PIT_PIVR.

### 15.4.3 Periodic Interval Timer Value Register

**Register Name:** PIT_PIVR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PICNT | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| PICNT | | | | CPIV | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CPIV | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CPIV | | | | | | | |

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

### 15.4.4 Periodic Interval Timer Image Register

**Register Name:** PIT_PIIR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PICNT | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| PICNT | | | | CPIV | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CPIV | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CPIV | | | | | | | |

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

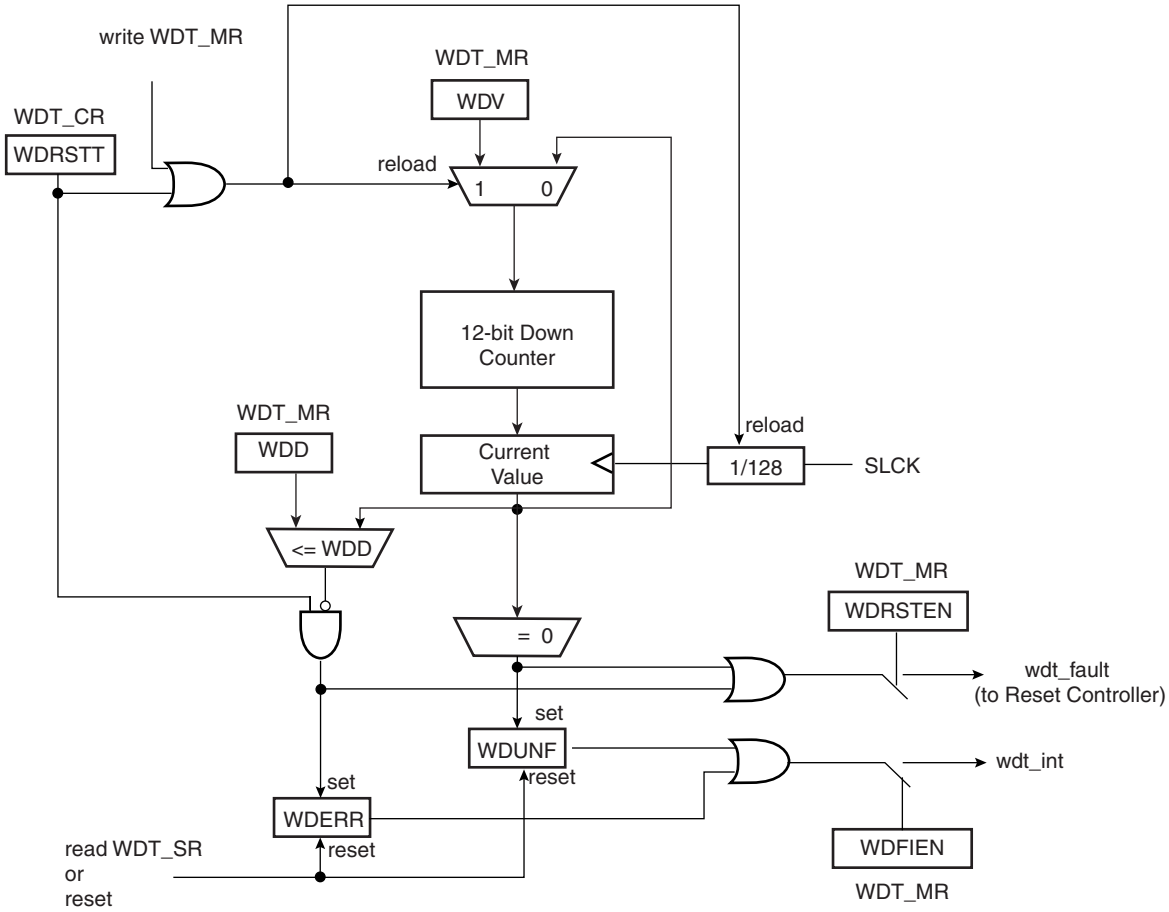
# 16. Watchdog Timer (WDT)

## 16.1 Overview

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock at 32.768 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

## 16.2 Block Diagram

**Figure 16-1.** Watchdog Timer Block Diagram



## 16.3 Functional Description

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It is supplied with VDDCORE. It restarts with initial values on processor reset.

The Watchdog is built around a 12-bit down counter, which is loaded with the value defined in the field WDV of the Mode Register (WDT_MR). The Watchdog Timer uses the Slow Clock divided by 128 to establish the maximum Watchdog period to be 16 seconds (with a typical Slow Clock of 32.768 kHz).

After a Processor Reset, the value of WDV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (field WDRSTEN at 1 after a Backup Reset). This means that a default Watchdog is running at reset, i.e., at power-up. The user must either disable it (by setting the WDDIS bit in WDT_MR) if he does not expect to use it or must reprogram it to meet the maximum Watchdog period the application requires.

The Watchdog Mode Register (WDT_MR) can be written only once. Only a processor reset resets it. Writing the WDT_MR register reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the Watchdog at regular intervals before the timer underflow occurs, by writing the Control Register (WDT_CR) with the bit WDRSTT to 1. The Watchdog counter is then immediately reloaded from WDT_MR and restarted, and the Slow Clock 128 divider is reset and restarted. The WDT_CR register is write-protected. As a result, writing WDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the “wdt_fault” signal to the Reset Controller is asserted if the bit WDRSTEN is set in the Mode Register (WDT_MR). Moreover, the bit WDUNF is set in the Watchdog Status Register (WDT_SR).

To prevent a software deadlock that continuously triggers the Watchdog, the reload of the Watchdog must occur while the Watchdog counter is within a window between 0 and WDD, WDD is defined in the WatchDog Mode Register WDT_MR.

Any attempt to restart the Watchdog while the Watchdog counter is between WDV and WDD results in a Watchdog error, even if the Watchdog is disabled. The bit WDERR is updated in the WDT_SR and the “wdt_fault” signal to the Reset Controller is asserted.

Note that this feature can be disabled by programming a WDD value greater than or equal to the WDV value. In such a configuration, restarting the Watchdog Timer is permitted in the whole range [0; WDV] and does not generate an error. This is the default configuration on reset (the WDD and WDV values are equal).

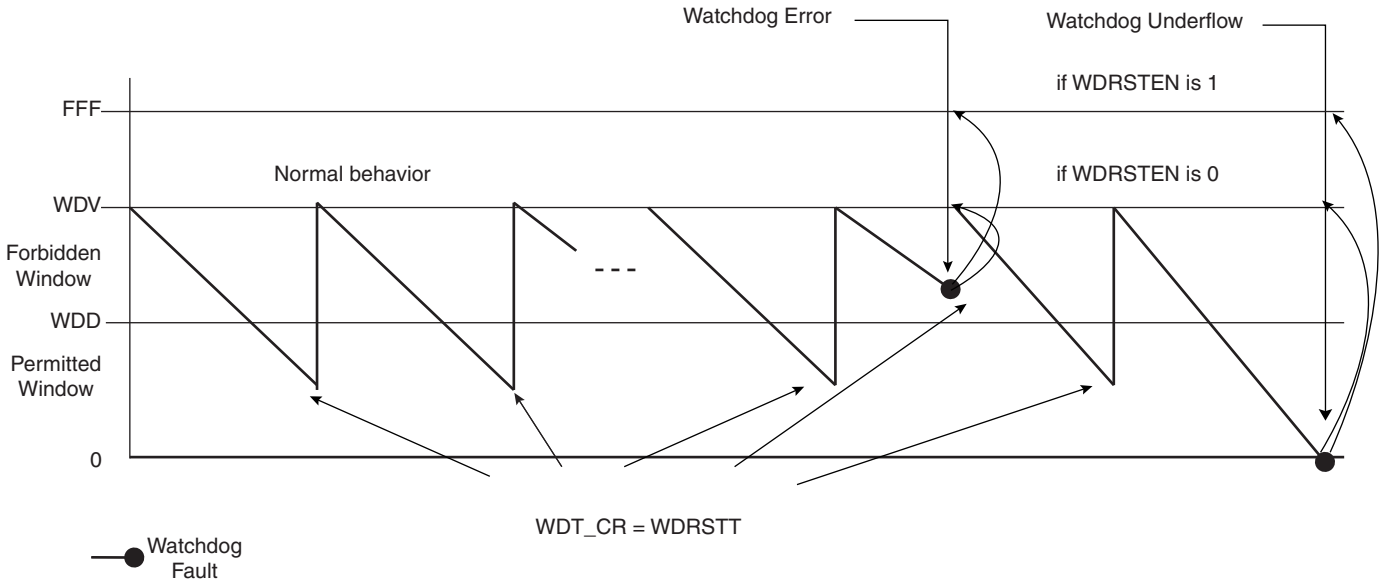
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet. In that case, the processor and the Watchdog Timer are reset, and the WDERR and WDUNF flags are reset.

If a reset is generated or if WDT_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt_fault” signal to the reset controller is deasserted.

Writing the WDT_MR reloads and restarts the down counter.

While the processor is in debug state or in idle mode, the counter may be stopped depending on the value programmed for the bits WDIDLEHLT and WDDBGHLT in the WDT_MR.

**Figure 16-2.** Watchdog Behavior



## 16.4 Watchdog Timer (WDT) User Interface

**Table 16-1.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | WDT_CR | Write-only | - |
| 0x04 | Mode Register | WDT_MR | Read-write Once | 0x3FFF_2FFF |
| 0x08 | Status Register | WDT_SR | Read-only | 0x0000_0000 |

### 16.4.1 Watchdog Timer Control Register

**Register Name:** WDT_CR

**Access Type:** Write-only

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

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

### 16.4.2 Watchdog Timer Mode Register

**Register Name:** WDT_MR

**Access Type:** Read-write Once

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | WDIDLEHLT | WDDBGHLT | WDD | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| WDD | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WDDIS | WDRPROC | WDRSTEN | WDFIEN | WDV | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDV | | | | | | | |

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 12-bit Watchdog Counter.

- **WDFIEN: Watchdog Fault Interrupt Enable**

0: A Watchdog fault (underflow or error) has no effect on interrupt.

1: A Watchdog fault (underflow or error) asserts interrupt.

- **WDRSTEN: Watchdog Reset Enable**

0: A Watchdog fault (underflow or error) has no effect on the resets.

1: A Watchdog fault (underflow or error) triggers a Watchdog reset.

- **WDRPROC: Watchdog Reset Processor**

0: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates all resets.

1: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates the processor reset.

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

If the Watchdog Timer value is less than or equal to WDD, writing WDT_CR with WDRSTT = 1 restarts the timer.

If the Watchdog Timer value is greater than WDD, writing WDT_CR with WDRSTT = 1 causes a Watchdog error.

- **WDDBGHLT: Watchdog Debug Halt**

0: The Watchdog runs when the processor is in debug state.

1: The Watchdog stops when the processor is in debug state.

- **WDIDLEHLT: Watchdog Idle Halt**

0: The Watchdog runs when the system is in idle mode.

1: The Watchdog stops when the system is in idle state.

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

### 16.4.3 Watchdog Timer Status Register

**Register Name:** WDT_SR

**Access Type:** Read-only

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

- **WDUNF: Watchdog Underflow**

0: No Watchdog underflow occurred since the last read of WDT_SR.

1: At least one Watchdog underflow occurred since the last read of WDT_SR.

- **WDERR: Watchdog Error**

0: No Watchdog error occurred since the last read of WDT_SR.

1: At least one Watchdog error occurred since the last read of WDT_SR.

# 17. Supply Controller (SUPC)

## 17.1 Overview

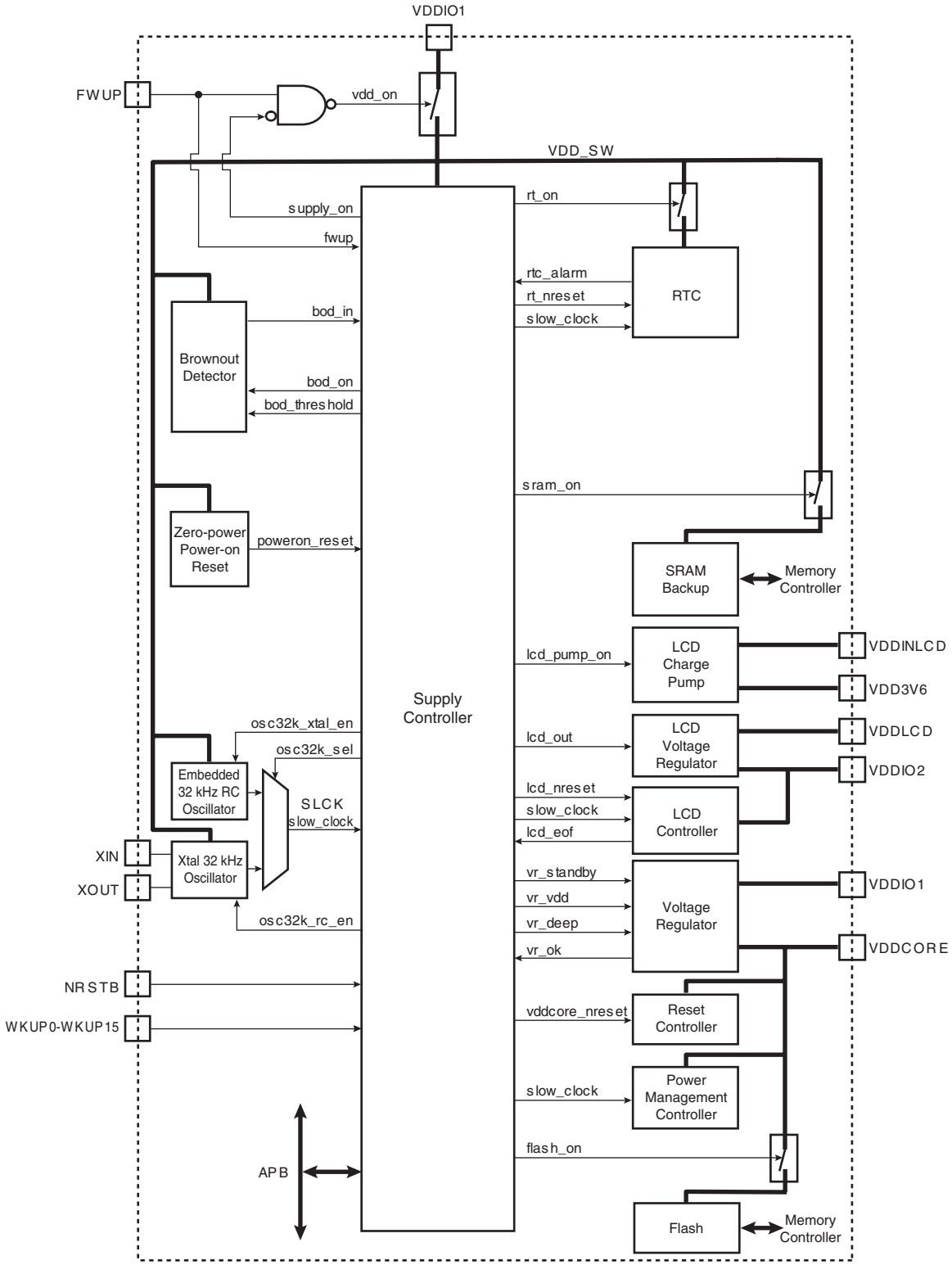
The Supply Controller (SUPC) controls the supply voltages of the system. In a typical application, the Supply Controller allows supply of the device directly from a double NiMH or NiCd battery or from a double CR2031 Lithium battery or from any Lithium rechargeable battery.

The Supply Controller offers a wide range of Low Power Modes, including:

- Off Mode, current consumption reduced to below 1 microamp, exits on the assertion of the Force Wake Up pin (FWUP)
- Backup Mode, current consumption reduced to a few microamps for Clock and SRAM retention, exits on multiple wake-up sources
- Running Mode, reaches a 30-MIPS performance level

## 17.2 Block Diagram

**Figure 17-1.** Supply Controller Block Diagram



## 17.3 Supply Controller Functional Description

### 17.3.1 Supply Controller Overview

The Supply Controller controls the power supplies of each section of the product:

- The Backup, including the Supply Controller, a part of the Reset Controller and the Slow Clock switcher
- The backup SRAM
- The Clock, including the Real Time Clock
- The Flash Memory
- The CORE, including the other part of the Reset Controller, the Processor and the Peripherals
- the LCD controller, the charge pump and the LCD voltage regulator

The Supply Controller has its own reset circuitry and is clocked by the 32 kHz Slow clock generator.

The reset circuitry is based on the NRSTB pin, a zero-power power-on reset cell and a brownout detector cell. The zero-power power-on reset allows the Supply Controller to start properly, while the software-programmable Brownout Detector detects either a battery discharge or main voltage loss.

The Slow Clock generator is based on a 32 kHz crystal oscillator and an embedded 32 kHz RC oscillator. The Slow Clock defaults to the RC oscillator, but the software can enable the crystal oscillator and select it as the Slow Clock source.

The Supply Controller starts up the device by sequentially enabling the internal power switches and the Voltage Regulator, then generates the proper reset signals to the core power supply.

It also sets the system in different low power modes and wakes it up from a wide range of events.

### 17.3.2 Slow Clock Generator

The Supply Controller embeds a slow clock generator that is supplied with the backup power supply. As soon as FWUP is asserted, both the crystal oscillator and the embedded RC oscillator are powered up, but only the embedded RC oscillator is enabled. This allows the slow clock to be valid in a short time (about 100 µs).

The user can select the crystal oscillator to be the source of the slow clock, as it provides a more accurate frequency. The command is made by writing the Supply Controller Control Register, SUPC_CR, with the XTALSEL bit at 1. This results in a sequence which first enables the crystal oscillator, then waits for 32,768 slow clock cycles, then switches the slow clock on the output of the crystal oscillator and then disables the RC oscillator to save power. The switch of the slow clock source is glitch free. The OSCSEL bit of the Supply Controller Status Register, SUPC_SR, allows knowing when the switch sequence is done.

Coming back on the RC oscillator is only possible by shutting down the backup power supply.

If the user does not need the crystal oscillator, the XIN and XOUT pins should be left unconnected.

The user can also set the crystal oscillator in bypass mode instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN. The input characteristics of the XIN pin are given in the product electrical characteristics section. In order to set the

bypass mode, the OSCBYPASS bit of the Supply Controller Mode Register (SUPC_MR) needs to be set at 1.

### 17.3.3 Brownout Detector

The Supply Controller embeds a Brownout Detector.

The Brownout Detector can be used to prevent the processor from falling into an unpredictable state if the power supply drops below a certain level or to detect a battery discharge.

The threshold of the Brownout Detector is programmable. It can be selected from 1.9V to 3.4V by steps of 100 mV. This threshold is programmed in the BODTH field of the Supply Controller Brownout Mode Register, SUPC_BOMR.

The Brownout Detector can also be enabled during one slow clock period of either 32, 256 or 2048 slow clock periods. This can be configured by programming the BODSMPL field in SUPC_BOMR.

Enabling the Brownout Detector for such reduced times allows to divide the typical Brownout Detector power consumption respectively by factors of 32, 256 or 2048, if the user does not need a continuous monitoring of the VDDIO1 Power Supply.

The Brownout Detector can either generate a reset of the Core or a wake up of the Core Power Supply. Generating a Core reset when a brownout occurs is enabled by writing the BODRSTEN bit to 1 in SUPC_BOMR.

Waking up the Core Power Supply when a brownout occurs can be enabled by programming the BODEN bit to 1 in the Supply Controller Wake Up Mode Register, SUPC_WUMR.

### 17.3.4 Backup Power Supply Reset

#### *17.3.4.1 Raising the Backup Power Supply*

Powering VDDIO1 does not power the device since FWUP is not asserted. If the FWUP pin is not used, it shall be connected to GND.

When the FWUP pin is tied to GND, the backup power supply is enabled. The RC oscillator is powered up and the zero-power power-on reset cell maintains its output for a time longer than the startup of the RC oscillator. During this time, the Supply Controller is entirely reset. When this signal is released a counter is started for 30 slow clock cycles. This is the debouncing of the Force Wake Up pin. If the FWUP pin is not maintained low, the backup power supply is powered off.

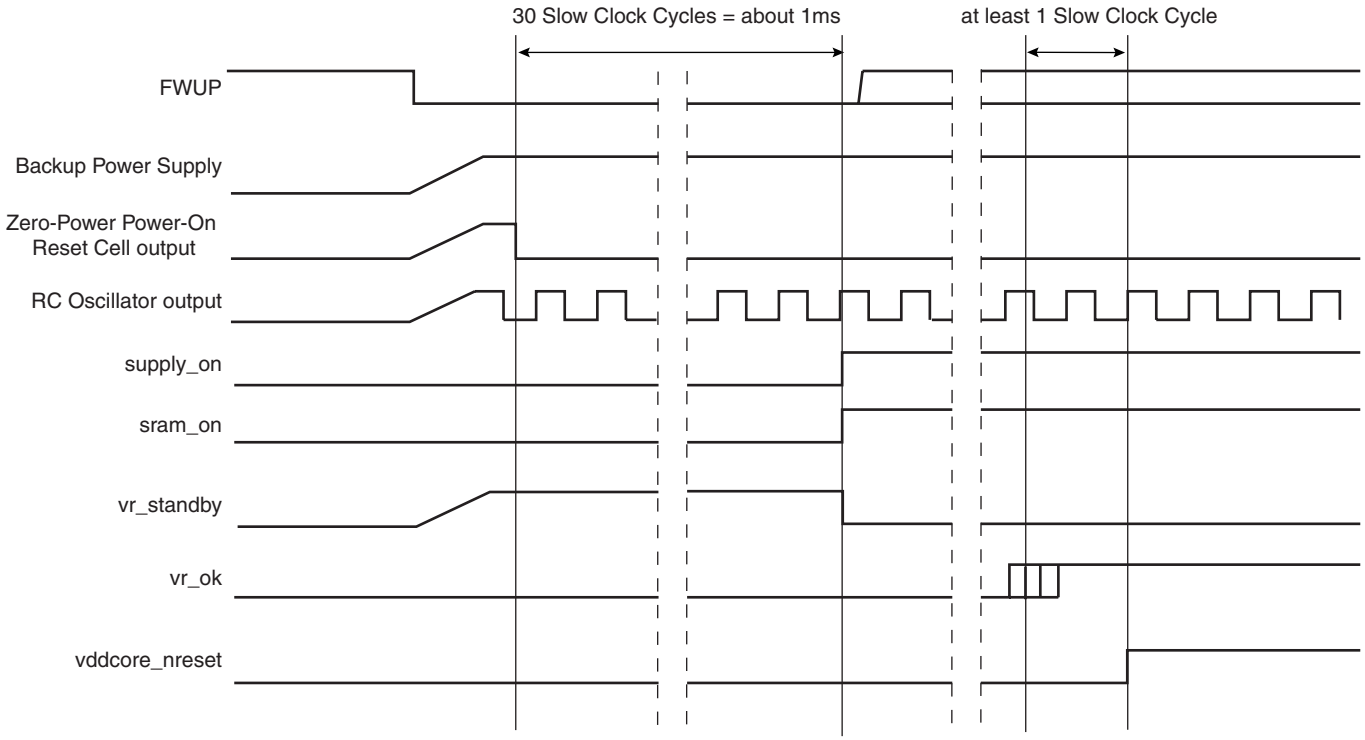
If the FWUP pin is maintained low, the signal, supply_on is asserted, thus auto-maintaining the backup power supply. The FWUP pin becomes a wake up source.

At the same time the supply_on signal is asserted, the voltage regulator, the Flash Memory and the SRAM are powered up according to the User Interface reset state. The voltage regulator starts and provides the vr_ok signal as soon as its output is valid. This results in releasing the vddcore_nreset signal to the Reset Controller after the vr_ok signal has been confirmed as being valid for at least one slow clock cycle.

At the same time the voltage regulator is powered up, the Supply Controller and all of the devices supplied by the backup power supply, are correctly started. The Supply Controller also sets the status bit, FWUPS in the Supply Controller Status Register, SUPC_SR. This status bit is cleared as soon as SUPC_SR is read and indicates the first power up of the backup power supply.

Reading FWUPS to 0 means SUPC_SR has already been read since the power up of the backup power supply and thus, it is not necessary to initialize the Supply Controller.

**Figure 17-2.** Raising the Backup Power Supply




### *17.3.4.2 NRSTB Asynchronous Reset Pin*

The NRSTB pin is an asynchronous reset input, which acts exactly like the zero-power power-on reset cell.

As soon as NRSTB is tied to GND, the supply controller is reset and all the system parts are powered off.

When NRSTB is released, the system can start as described in Section 17.3.4.1 ”Raising the Backup Power Supply”.

**Figure 17-3.** NRSTB Reset when FWUP = 0

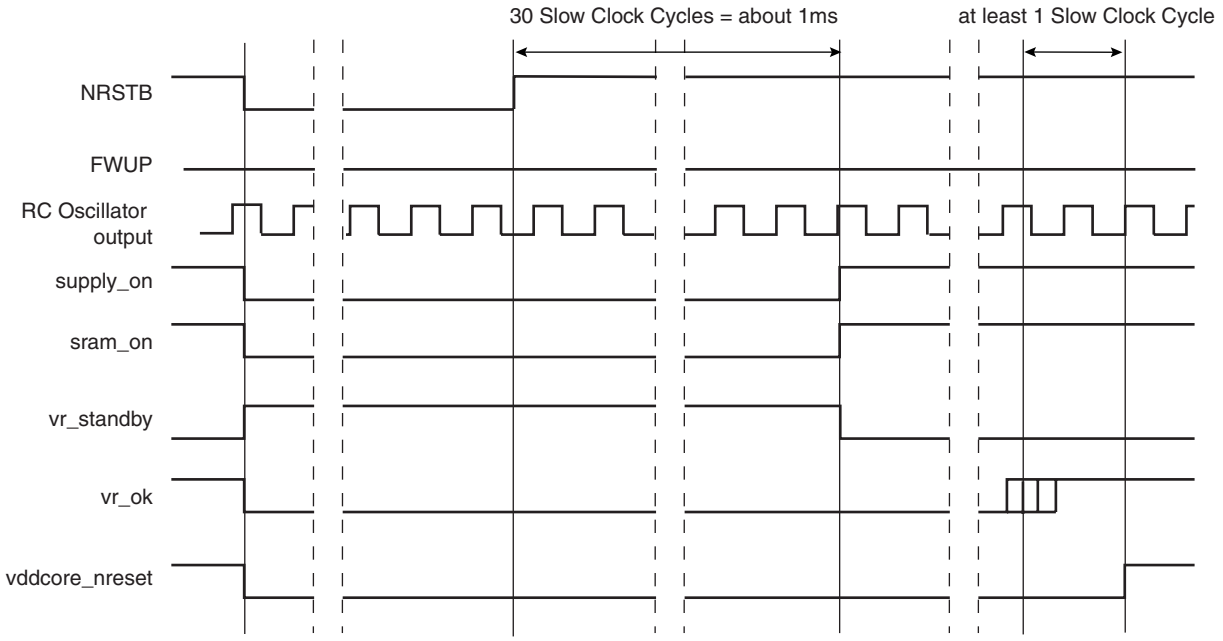



**Figure 17-4.** NRSTB Reset when FWUP = 1 and NRSTB is Released Before FWUP = 0

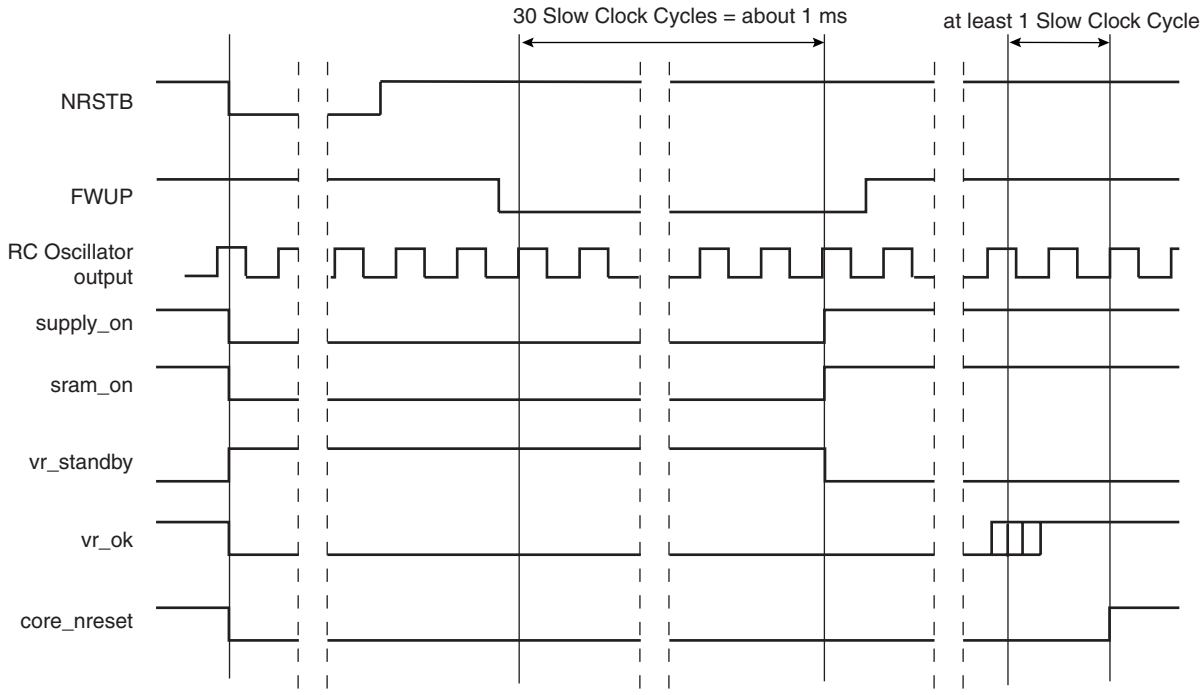
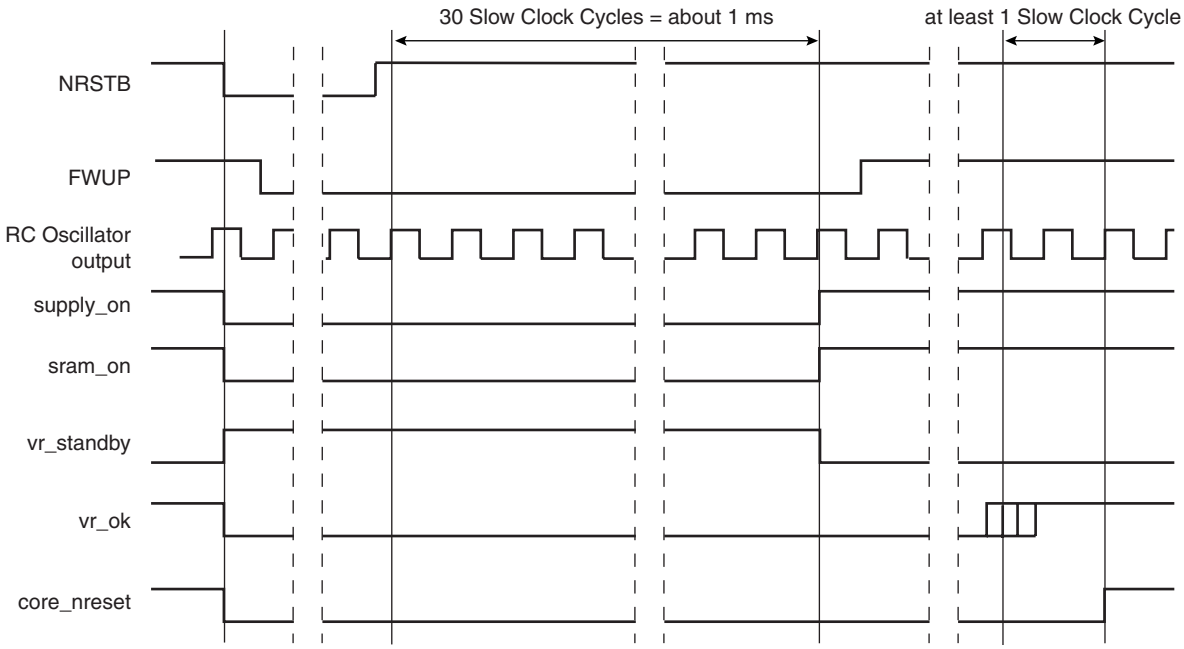


**Figure 17-5.** NRSTB Reset when FWUP = 1 and NRSTB is Released After FWUP = 0




### 17.3.5 Core Reset

The Supply Controller manages the vddcore_nreset signal to the Reset Controller, as described previously in Section 17.3.4 "Backup Power Supply Reset". The vddcore_nreset signal is normally asserted before shutting down the core power supply and released as soon as the core power supply is correctly regulated.

There are two additional sources which can be programmed to activate vddcore_nreset:

- the brownout detector
- voltage regulation loss

#### *17.3.5.1 Brownout Detector Reset*

The Brownout Detector is capable of generating a reset of the system. This can be enabled by setting the BODRSTEN bit in the Supply Controller Mode Register, SUPC_MR.

If BODRSTEN is set and a brownout is detected, the vddcore_nreset signal is immediately activated for a minimum of 2 slow clock cycles.

#### *17.3.5.2 Voltage Regulation Loss Reset*

The voltage regulator provides the vr_ok signal which indicates that the regulation is operating as programmed. If this signal is lost for longer than 1 slow clock period while the voltage regulator is enabled, the Supply Controller can assert vddcore_nreset. This feature is enabled by writing the bit, VRRSTEN (Voltage Regulator Reset Enable) to 1 in the Supply Controller Mode Register, SUPC_MR and if the voltage regulator is set in normal mode (VRMODE is at 0).

When the voltage regulator is in deep mode, this feature is not enabled.

If VRRSTEN is set and the voltage regulation is lost (output voltage of the regulator too low), the vddcore_nreset signal is asserted for a minimum of 2 slow clock periods and then released if vr_ok has been reactivated. The VRRSTS bit is set in the Supply Controller Status Register, SUPC_SR, so that the user can know the source of the last reset.

Until vr_ok is deactivated, the vddcore_nreset signal remains active.

### 17.3.6 Power Supply Control

#### *17.3.6.1 Controlling the Backup Power Supply*

The backup power supply can be controlled by the main power switch. This main power switch can only be enabled by tying the FWUP pin to GND. As soon as the power has risen, the Supply Controller maintains the main power switch closed by asserting the signal, supply_on.

The main power switch can be opened by the software by writing the Supply Controller Control Register SUPC_CR with the shutdown bit, SHDW set at 1.

Writing SUPC_CR with SHDW set at 1 results in the following actions:

- asserts the vddcore_nreset signal then switches off the voltage regulator
- if the LCD charge pump is enabled, asserts the lcd_nreset signal then disables the LCD charge pump
- asserts the Flash Memory reset signal and disables the Flash Memory power supply
- disables the SRAM power supply
- asserts the Clock reset signal, rt_nreset, then disables the Clock power supply
- releases the supply_on signal, thus switching off the backup power supply and enters Off Mode.

The shutdown sequence led by writing SHDW is described in Figure 17-6 on page 111.

It is also possible to wait the current frame of the LCD Controller before shutting down the LCD Controller power supply. This can be done by writing the SHDWEOF bit to 1, instead of the SHDW bit.

If SHDWEOF is set, the sequence is exactly the same, except the end_of_frame signal shall be asserted for at least one slow clock cycle before the lcd_nreset signal is asserted and the charge pump is disabled.

The shutdown sequence led by writing SHDWEOF is described in Figure 17-7 on page 112.

**Figure 17-6.** Shutdown of the Backup Power Supply After Writing SHDW at 1

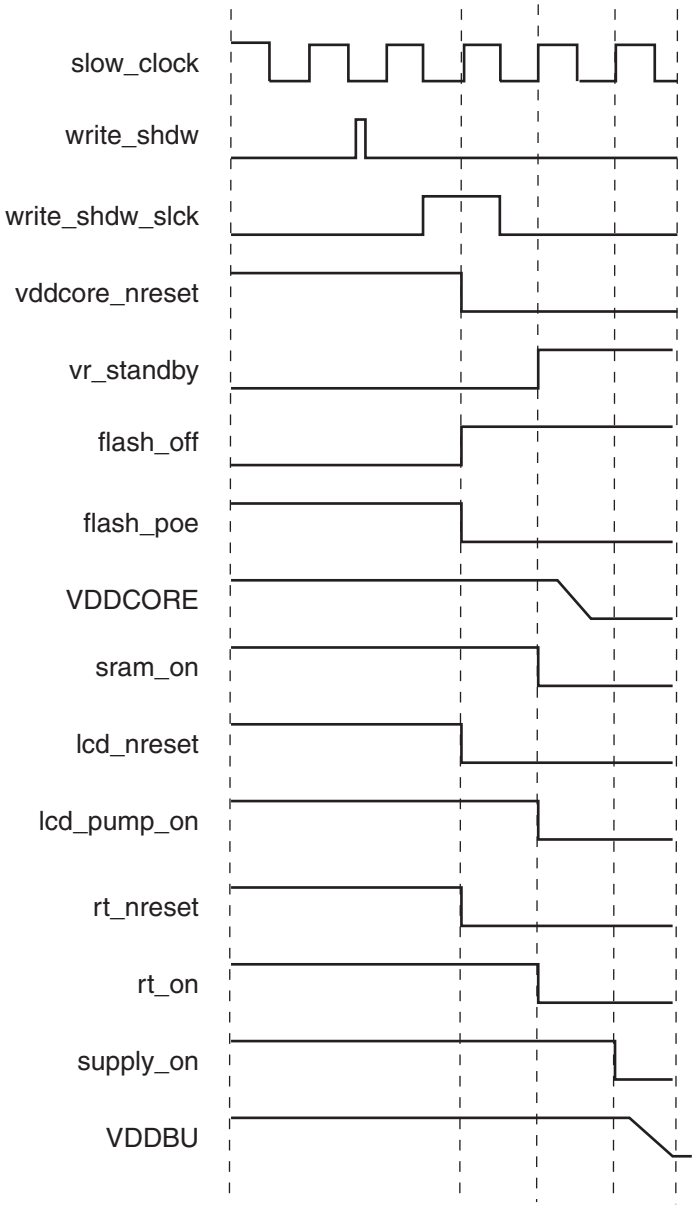
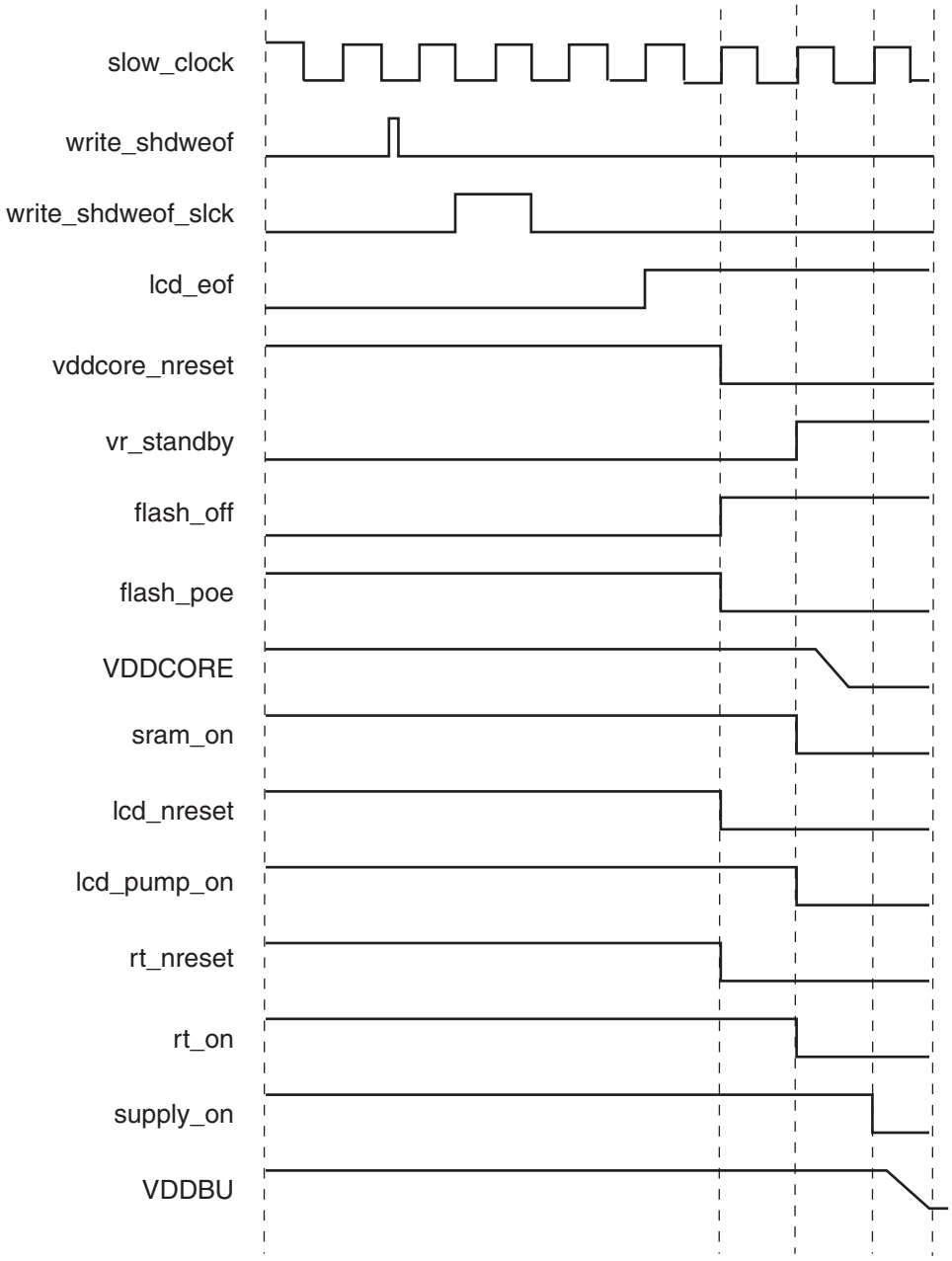


**Figure 17-7.** Shutdown of the Backup Power Supply After Writing SHDWEOF at 1




*17.3.6.2 Controlling the Voltage Regulator*

The Supply Controller can be used to control the embedded 1.8V voltage regulator.

The VRVDD field in the Supply Controller Mode Register, SUPC_MR, allows to select the output voltage between 1.55V and 1.80V, depending on the performance required by the processor.

The VRDEEP field in the Supply Controller Mode Register, SUPC_MR, allows to switch the voltage regulator into deep mode, thus reducing its leakage current to a minimum.

The programmer can switch off the voltage regulator, and thus put the device in Backup Mode, by writing the Supply Controller Control Register, SUPC_CR, with the VROFF bit at 1. This asserts the vddcore_nreset signal after the write resynchronization time which lasts, in the worse

case, 2 slow clock cycles. Once the vddcore_nreset signal is asserted, the processor and the peripherals are stopped 1 slow clock cycle before the core power supply becomes off.

The loss of voltage regulation while the core power supply is enabled can be programmed to generate a reset by writing the VRRSTEN bit to 1 in the Supply Controller Mode Register, SUPC_MR.

#### 17.3.6.3 *Controlling the SRAM Power Supply*

The Supply Controller can be used to switch on or off the power supply of the backup SRAM by opening or closing the SRAM power switch. This power switch is controlled by the SRAMON bit of the Supply Controller Mode Register, SUPC_MR. However, the battery backup SRAM is automatically switched on when the core power supply is enabled, as the processor requires the SRAM as data memory space (Please refer to Figure 17-2 on page 107).

- If SRAMON is written to 1, there is no immediate effect, but the SRAM will be left powered when the Supply Controller enters backup mode, thus retaining its content.
- If SRAMON is written to 0, there is no immediate effect, but the SRAM will be switched off when the Supply Controller enters backup mode. The SRAM is automatically switched on at the exit of the backup mode.

#### 17.3.6.4 *Controlling the Clock Alarm Power Supply*

The Supply Monitor can be used to switch on or off the power supply of the Clock Alarm (Real Time Clock (RTC)) by opening or closing the corresponding power switch. This power switch is controlled by the RTON bit in SUPC_MR. After a backup reset, the Clock is not supplied (RTON = 0).

The status of the Clock Power supply can be seen through the status register (RTS in SUPC_SR).

- If RTON is written to 1 while it is at 0, after the write resynchronization time (about 2 slow clock cycles), the Clock power switch is closed by setting the signal, rt_on at 1, then after one slow clock cycle, the rt_nreset signal is released. This ensures that the Clock is always properly cleared when its power rises.
- If RTON is written to 0 while it is at 1, after the write resynchronization time (about 2 slow clock cycles), the rt_nreset signal is asserted, then after one slow clock cycle, the Clock power switch is opened by resetting the signal, rt_on at 0.

There are several restrictions concerning the write of the RTON field:

- The user must check that the previous power supply switch operation is done before writing RTON again. To do that, the user must check that the RTS flag has the correct value. If RTON is written to 0, the RTS flag is reset at 0. If RTON is written to 1, the RTS flag is set at 1.
- Writing RTON at 1 while it is already at 1 or writing RTON at 0 while it is already at 0 is forbidden and has no effect.

#### 17.3.6.5 *Controlling the LCD Voltage Regulator Power Supply*

The Supply Controller can be used to select the power supply source of the LCD voltage regulator. The LCD voltage regulator can either be supplied through an external power supply or by the embedded charge pump.

This selection is done by the LCDMODE field in the SUPC_MR register. After a backup reset, the LCDMODE field is at 0x0, it means that no power supply source is selected and the LCD Controller reset signal, lcd_nreset is asserted.

The status of the LCD Controller Reset can be seen through the LCDS field in the status register, SUPC_ SR.

- If LCDMODE is written to 0x2 while it is at 0x0 or 0x1, after the write resynchronization time (about 2 slow clock cycles), the external power supply source is selected by setting the output signal, lcd_ext_on at 1, then after one slow clock cycle, the reset signal, lcd_nreset is released.
- If LCDMODE is written to 0x0 while it is at 0x2, after the write resynchronization time (about 2 slow clock cycles), the reset signal, lcd_nreset is asserted, then after one slow clock cycle, the external power supply source is deselected by resetting the output signal, lcd_ext_on at 0.
- If LCDMODE is written to 0x1 while it is at 0x2, after the write resynchronization time (about 2 slow clock cycles), the Supply Controller waits for the End of Frame, then the reset signal, lcd_nreset is asserted, then after one slow clock cycle, the external power supply source is deselected by resetting the output signal, lcd_ext_on at 0.
- If LCDMODE is written to 0x3 while it is at 0x0 or 0x1, after the write resynchronization time (about 2 slow clock cycles), the internal power supply source is selected and the embedded charge pump turned on by setting the output signal, lcd_int_on at 1, then after 15 slow clock cycles, the reset signal, lcd_nreset is released.
- If LCDMODE is written to 0x0 while it is at 0x3, after the write resynchronization time (about 2 slow clock cycles), the reset signal, lcd_nreset is asserted, then after one slow clock cycle, the internal power supply source is deselected and the embedded charge pump turned off by resetting the output signal, lcd_int_on at 0.
- If LCDMODE is written to 0x1 while it is at 0x3, after the write resynchronization time (about 2 slow clock cycles), the Supply Controller waits for the End of Frame, then the reset signal, lcd_nreset is asserted, then after one slow clock cycle, the internal power supply source is deselected and the embedded charge pump turned off by resetting the output signal, lcd_int_on at 0.

There are several restrictions concerning the write of the LCDMODE field:

- The user must check that the previous power supply selection is done before writing LCDMODE again. To do that, the user must check that the LCDS flag has the correct value. If LCDMODE is written to 0x0 or 0x1, the LCDS flag is reset at 0. If LCDMODE is written to 0x0 or 0x1, the LCDS flag is set at 1.
- Writing LCDMODE to 0x2 while it is at 0x3 or writing LCDMODE to 0x3 while it is at 0x2 is forbidden and has no effect.
- Before writing LCDMODE to 0x2, the user must ensure that the external power supply is ready and supplies the VDDLCD pad.
- Before writing LCDMODE to 0x3, the user must ensure that the external power supply doesn't supply the VDDLCD pad.

#### *17.3.6.6 Controlling the Flash Memory Power Supply*

The Supply Controller can be used to switch on or off the power supply of the Flash Memory by opening or closing the Flash Memory power switch (connected to VDDCORE). This power switch is controlled by the FLASHON bit of the Supply Controller Mode Register (SUPC_MR). Before setting FLASHON to 1 or 0, the user needs to program SUPC_FWUT correctly. Based on this counter the Supply Controller will correctly manage the control of the Flash Memory (refer to the wake-up time of the Flash Memory in the Electrical Characteristics section of the product datasheet).

The Flash Memory is automatically switched on when the core power supply is enabled at start up.

The status of the Flash Memory, i.e., ready to use, or not ready, can be seen through the FLASHS field in the status register SUPC_ SR.

- If FLASHON is written to 1 while it is at 0, after one main clock cycle, the Flash Memory power switch is closed by resetting the flash_off signal at 0, then after ninety main clock cycles the FLASHS flag signal is set at 1. This ensures that the Flash Memory is always properly cleared when its power rises.
- If FLASHON is written to 0 while it is at 1, after one main clock cycle, the flag FLASHS is reset to 0, then two main clock cycles after, the Flash Memory power switch is opened by setting the signal, flash_off at 1.

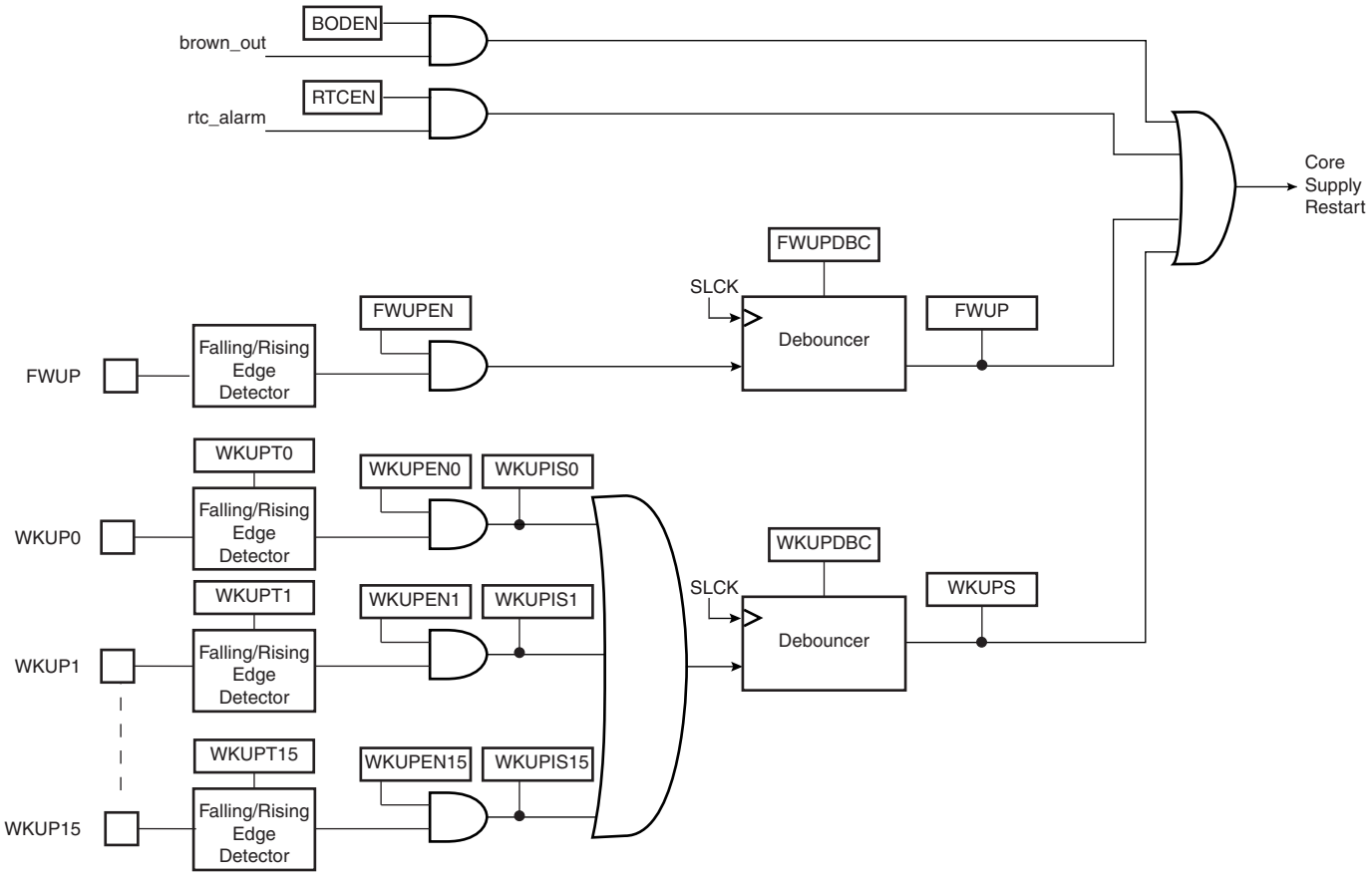
There are several restrictions concerning the write of the FLASHON field:

- The user must check that the previous power supply switch operation is done before writing FLASHON again. To do that, the user must check that the FLASHS flag has the correct value. If FLASHON is written to 0, the FLASHS flag is reset at 0. If FLASHON is written to 1, the FLASHS flag is set at 1.
- Writing FLASHON at 1 while it is already at 1 or writing FLASHON at 0 while it is already at 0 is forbidden and has no effect.

### 17.3.7 Wake Up Sources

The wake up events allow the device to exit backup mode. When a wake up event is detected, the Supply Controller performs a sequence which automatically reenables the core power supply, and the SRAM power supply, if it is not already enabled.

**Figure 17-8.** Wake Up Sources




### *17.3.7.1 Force Wake Up*

The Force Wake Up pin, FWUP is used to start up the backup power supply, as described in the previous paragraphs. Then, when supply_on is asserted by the Supply Controller, the FWUP can be used as a wake up source with a programmable debouncing period.

The FWUP pin is enabled as a wake up source by writing the FWUPEN bit to 1 in the Supply Controller Wake Up Mode Register, SUPC_WUMR. Then, the FWUPDBC field in the same register selects the debouncing period, which can be selected between 3, 32, 512, 4,096 or 32,768 slow clock cycles. This corresponds respectively to about 100 µs, about 1 ms, about 16 ms, about 128 ms and about 1 second (for a typical slow clock frequency of 32 kHz). Programming FWUPDBC to 0x0 selects an immediate wake up, i.e., the FWUP must be low during a minimum of one slow clock period to wake up the core power supply.

If the FWUP pin is asserted for a time longer than the debouncing period, a wake up of the core power supply is started and the FWUP bit in the Supply Controller Status Register, SUPC_SR, is set and remains high until the register is read.

### *17.3.7.2 Wake Up Inputs*

The wake up inputs, WKUP0 to WKUP15, can be programmed to perform a wake up of the core power supply. Each input can be enabled by writing to 1 the corresponding bit, WKUPEN0 to WKUPEN 15, in the Wake Up Inputs Register, SUPC_WUIR. The wake up level can be selected with the corresponding polarity bit, WKUPPL0 to WKUPPL15, also located in SUPC_WUIR.

All the resulting signals are wired-ORed to trigger a debounce counter, which can be programmed with the WKUPDBC field in the Supply Controller Wake Up Mode Register, SUPC_WUMR. The WKUPDBC field can select a debouncing period of 3, 32, 512, 4,096 or 32,768 slow clock cycles. This corresponds respectively to about 100 µs, about 1 ms, about 16 ms, about 128 ms and about 1 second (for a typical slow clock frequency of 32 kHz). Programming WKUPDBC to 0x0 selects an immediate wake up, i.e., an enabled WKUP pin must be active according to its polarity during a minimum of one slow clock period to wake up the core power supply.

If an enabled WKUP pin is asserted for a time longer than the debouncing period, a wake up of the core power supply is started and the signals, WKUP0 to WKUP15 as shown in Figure 17-8, are latched in the Supply Controller Status Register, SUPC_SR. This allows the user to identify the source of the wake up, however, if a new wake up condition occurs, the primary information is lost. No new wake up can be detected since the primary wake up condition has disappeared.

#### *17.3.7.3 Clock Alarms*

The RTC alarm can generate a wake up of the core power supply. This can be enabled by writing, the bit RTCEN to 1 in the Supply Controller Wake Up Mode Register, SUPC_WUMR.

The Supply Controller does not provide any status as the information is available in the User Interface of the Real Time Clock.

#### *17.3.7.4 Brownout Detector*

The brownout detector can generate a wake up of the core power supply. This can be enabled by writing the BODEN bit to 1 in the Supply Controller Mode Register, SUPC_MR.

The Supply Controller provides two status bits in the Supply Controller Status Register for the brownout detector which allow to determine whether the last wake up was due to the brownout detector:

- the BROWNOUT bit provides real time information, which is updated at each measurement cycle or updated at each Slow Clock cycle, if the measurement is continuous
- the BODS bit provides saved information and shows a brownout has occurred since the last read of SUPC_SR

## 17.4 Supply Controller (SUPC) User Interface

The User Interface of the Supply Controller is part of the System Controller User Interface.

### 17.4.1 System Controller (SYSC) User Interface

**Table 17-1.** System Controller Registers

| Offset | System Controller Peripheral | Name |
|---|---|---|
| 0x00-0x0c | Reset Controller | RSTC |
| 0x10-0x2C | Supply Controller | SUPC |
| 0x40-0x4C | Periodic Interval Counter | PIT |
| 0x50-0x5C | Watchdog | WDT |
| 0x60-0x7C | Real Time Clock | RTC |

### 17.4.2 Supply Controller (SUPC) User Interface

**Table 17-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Supply Controller Control Register | SUPC_CR | Write-only | N/A |
| 0x04 | Supply Controller Brownout Mode Register | SUPC_BOMR | Read-write | 0x0000_0000 |
| 0x08 | Supply Controller Mode Register | SUPC_MR | Read-write | 0x0008_0a00 |
| 0x0C | Supply Controller Wake Up Mode Register | SUPC_WUMR | Read-write | 0x0000_0000 |
| 0x10 | Supply Controller Wake Up Inputs Register | SUPC_WUIR | Read-write | 0x0000_0000 |
| 0x14 | Supply Controller Status Register | SUPC_SR | Read-only | 0x0000_0800 |
| 0x18 | Supply Controller Flash Wake-up Timer Register | SUPC_FWUTR | Read-write | 0x0000_005a |
| 0x1C | Reserved | | | |

### 17.4.3 Supply Controller Control Register

**Register Name:** SUPC_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | XTALSEL | VROFF | SHDWEOF | SHDW |

- **SHDW: Shut Down Command**

0 = No effect.

1 = If KEY is correct, enters the device in off mode.

- **SHDWEOF: Shut Down After End of Frame**

0 = No effect.

1 = If KEY is correct, enters the device in off mode at the End of Frame from the LCD Controller.

- **VROFF: Voltage Regulator Off**

0 = No effect.

1 = If KEY is correct, asserts vddcore_nreset and stops the voltage regulator..

- **XTALSEL: Crystal Oscillator Select**

0 = No effect.

1 = If KEY is correct, switches the slow clock on the crystal oscillator output.

- **KEY: Password**

Should be written to value 0xA5. Writing any other value in this field aborts the write operation.

### 17.4.4 Supply Controller Brownout Mode Register

**Register Name:** SUPC_BOMR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | BODRSTEN | – | BODSMPL | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | BODTH | | | |

- **BODTH: Brownout Threshold**

| BODTH | Brownout Threshold |
|---|---|
| 0x0 | 1.9 V |
| 0x1 | 2.0 V |
| 0x2 | 2.1 V |
| 0x3 | 2.2 V |
| 0x4 | 2.3 V |
| 0x5 | 2.4 V |
| 0x6 | 2.5 V |
| 0x7 | 2.6 V |
| 0x8 | 2.7 V |
| 0x9 | 2.8 V |
| 0xA | 2.9 V |
| 0xB | 3.0 V |
| 0xC | 3.1 V |
| 0xD | 3.2 V |
| 0xE | 3.3 V |
| 0xF | 3.4 V |

- **BODSMPL: Brownout Sampling Period**

| BODSMPL | Brownout Sampling Period |
|---|---|
| 0x0 | Brownout Detector disabled |
| 0x1 | Continuous Brownout Detector |
| 0x2 | Brownout Detector enabled one SLCK period every 32 SLCK periods |
| 0x3 | Brownout Detector enabled one SLCK period every 256 SLCK periods |
| 0x4 | Brownout Detector enabled one SLCK period every 2,048 SLCK periods |
| 0x5-0x7 | Reserved |

- **BODRSTEN: Brownout Reset Enable**

0 = The core reset signal, vddcore_nreset is not affected when a brownout occurs.

1 = The core reset signal, vddcore_nreset is asserted when a brownout occurs.

### 17.4.5 Supply Controller Mode Register

**Register Name:** SUPC_MR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | OSCBYPASS | FLASHON | RTON | SRAMON | – |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| – | – | – | VRRSTEN | VRVDD | | | VRDEEP |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | LCDMODE | | LCDOUT | | | |

- **LCDOUT: LCD Charge Pump Output Voltage Selection**

| LCDOUT | LCD Charge Pump Output Voltage |
|---|---|
| 0x0 | 2.400 V |
| 0x1 | 2.467 V |
| 0x2 | 2.533 V |
| 0x3 | 2.600 V |
| 0x4 | 2.667 V |
| 0x5 | 2.733 V |
| 0x6 | 2.800 V |
| 0x7 | 2.867 V |
| 0x8 | 2.933 V |
| 0x9 | 3.000 V |
| 0xA | 3.067 V |
| 0xB | 3.133 V |
| 0xC | 3.200 V |
| 0xD | 3.267 V |
| 0xE | 3.333 V |
| 0xF | 3.400 V |

- **LCDMODE: LCD Power Supply Mode**

| LCDMODE | LCD Controller Power Supply |
|---|---|
| 0x0 | The internal supply source and the external supply source are both deselected and the on-chip charge pump is turned off. |

| LCDMODE | LCD Controller Power Supply |
|---|---|
| 0x1 | At the End of Frame from the LCD Controller, the internal supply source and the external supply source are both deselected and the on-chip charge pump is turned off. |
| 0x2 | The external supply source is selected. |
| 0x3 | The internal supply source is selected and the on-chip charge pump is turned on. |

- **VRDEEP: Voltage Regulator Deep Mode**

0 = Voltage Regulator Deep Mode is disabled.

1 = Voltage Regulator Deep Mode is enabled.

- **VRVDD: Voltage Regulator Output Voltage Selection**

| VRVDD | Voltage Regulator Output Voltage |
|---|---|
| 0x0 | Reserved |
| 0x1 | Reserved |
| 0x2 | 1.55V |
| 0x3 | 1.65V |
| 0x4 | 1.75V |
| 0x5 - 0x7 | 1.80V |

- **VRRSTEN: Voltage Regulation Loss Reset Enable**

0 = Losing the voltage regulation does not affect the core reset signal, vddcore_nreset.

1 = Losing the voltage regulation asserts the core reset signal, vddcore_nreset.

- **SRAMON: SRAM On**

0 = SRAM (Backup) switched off in backup mode.

1 = SRAM (Backup) switched on in backup mode.

- **RTON: Real Time Clock Alarm Power Switch On**

0 = Real Time Clock Alarm switched off.

1 = Real Time Clock Alarm switched on.

- **FLASHON: Flash Memory Power Switch On**

0 = Flash Memory switched off.

1 = Flash Memory switched on.

- **OSCBYPASS: Oscillator Bypass**

0 = No effect. Clock selection depends on XTALSEL value.

1 = The 32-KHz XTAL oscillator is selected and is put in bypass mode.

- **KEY: Password Key**

Should be written to value 0xA5. Writing any other value in this field aborts the write operation.

### 17.4.6 Supply Controller Wake Up Mode Register

**Register Name:** SUPC_WUMR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | WKUPDBC | | | – | FWUPDBC | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | RTCEN | – | BODEN | FWUPEN |

- **FWUPEN: Force Wake Up Enable**

0 = The Force Wake Up pin has no wake up effect.

1 = The Force Wake Up pin low forces the wake up of the core power supply.

- **BODEN: Brownout Wake Up Enable**

0 = The brownout alarm signal has no wake up effect.

1 = The brownout alarm signal forces the wake up of the core power supply.

- **RTCEN: Real Time Clock Wake Up Enable**

0 = The RTC alarm signal has no wake up effect.

1 = The RTC alarm signal forces the wake up of the core power supply.

- **FWUPDBC: Force Wake Up Debouncer**

| FWUPDBC | Force Wake Up Debouncer |
|---|---|
| 0x0 | Immediate, no debouncing, detected active at least on one Slow Clock edge. |
| 0x1 | FWUP shall be low for at least 3 SLCK periods |
| 0x2 | FWUP shall be low for at least 32 SLCK periods |
| 0x3 | FWUP shall be low for at least 512 SLCK periods |
| 0x4 | FWUP shall be low for at least 4,096 SLCK periods |
| 0x5 | FWUP shall be low for at least 32,768 SLCK periods |
| 0x6-0x7 | Reserved |

- **WUPDBC: Wake Up Inputs Debouncer**

| WUPDBC | Wake Up Inputs Debouncer |
|---|---|
| 0x0 | Immediate, no debouncing, detected active at least on one Slow Clock edge. |
| 0x1 | An enabled wake-up input shall be active for at least 3 SLCK periods |
| 0x2 | An enabled wake-up input shall be active for at least 32 SLCK periods |

| WUPDBC | Wake Up Inputs Debouncer |
|---|---|
| 0x3 | An enabled wake-up input shall be active for at least 512 SLCK periods |
| 0x4 | An enabled wake-up input shall be active for at least 4,096 SLCK periods |
| 0x5 | An enabled wake-up input shall be active for at least 32,768 SLCK periods |
| 0x6-0x7 | Reserved |

### 17.4.7 System Controller Wake Up Inputs Register

**Register Name:** SDC_WUIR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| WKUPT15 | WKUPT14 | WKUPT13 | WKUPT12 | WKUPT11 | WKUPT10 | WKUPT9 | WKUPT8 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| WKUPT7 | WKUPT6 | WKUPT5 | WKUPT4 | WKUPT3 | WKUPT2 | WKUPT1 | WKUPT0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WKUPEN15 | WKUPEN14 | WKUPEN13 | WKUPEN12 | WKUPEN11 | WKUPEN10 | WKUPEN9 | WKUPEN8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WKUPEN7 | WKUPEN6 | WKUPEN5 | WKUPEN4 | WKUPEN3 | WKUPEN2 | WKUPEN1 | WKUPEN0 |

- **WKUPEN0 - WKUPEN15: Wake Up Input Enable 0 to 15**

0 = The corresponding wake-up input has no wake up effect.

1 = The corresponding wake-up input forces the wake up of the core power supply.

- **WKUPT0 - WKUPT15: Wake Up Input Transition 0 to 15**

0 = A high to low level transition on the corresponding wake-up input forces the wake up of the core power supply.

1 = A low to high level transition on the corresponding wake-up input forces the wake up of the core power supply.

### 17.4.8 Supply Controller Status Register

**Register Name:** SUPC_SR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| WKUPIS15 | WKUPIS14 | WKUPIS13 | WKUPIS12 | WKUPIS11 | WKUPIS10 | WKUPIS9 | WKUPIS8 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| WKUPIS7 | WKUPIS6 | WKUPIS5 | WKUPIS4 | WKUPIS3 | WKUPIS2 | WKUPIS1 | WKUPIS0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | FWUPIS | FLASHS | RTS | – | LCDS |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSCSEL | BROWNOUT | BODS | BODRSTS | VRRSTS | BODWS | WKUPS | FWUPS |

- **FWUPS: FWUP Wake Up Status**

0 = No wake up due to the assertion of the FWUP pin has occurred since the last read of SUPC_SR.

1 = At least one wake up due to the assertion of the FWUP pin has occurred since the last read of SUPC_SR.

- **WKUPS: WKUP Wake Up Status**

0 = No wake up due to the assertion of the WKUP pins has occurred since the last read of SUPC_SR.

1 = At least one wake up due to the assertion of the WKUP pins has occurred since the last read of SUPC_SR.

- **BODWS: Brownout Detection Wake Up Status**

0 = No wake up due to a brownout detection has occurred since the last read of SUPC_SR.

1 = At least one wake up due to a brownout detection has occurred since the last read of SUPC_SR.

- **VRRSTS: Voltage Regulation Loss Reset Status**

0 = No voltage regulation loss has generated a core reset since the last read of the SUPC_SR.

1 = At least one voltage regulation loss has generated a core reset since the last read of the SUPC_SR.

- **BODRSTS: Brownout Detection Reset Status**

0 = No brownout detection has generated a core reset since the last read of the SUPC_SR.

1 = At least one brownout detection has generated a core reset since the last read of the SUPC_SR.

- **BODS: Brownout Detector Status**

0 = No brownout has been detected since the last read of SUPC_SR.

1 = At least one brownout has been detected since the last read of SUPC_SR.

- **BROWNOUT: Brownout Detector Output Status**

0 = The brownout detector detected VDDIO1 higher than its threshold at its last measurement.

1 = The brownout detector detected VDDIO1 lower than its threshold at its last measurement.

- **OSCSEL: 32-kHz Oscillator Selection Status**

0 = The slow clock, SLCK is generated by the embedded 32-kHz RC oscillator.

1 = The slow clock, SLCK is generated by the 32-kHz crystal oscillator.

- **LCDS: LCD Status**

0 = The LCD Controller is off and cannot be used.

1 = The LCD Controller is on and can be used.

- **RTS: Clock Status**

0 = The Clock is off and cannot be used.

1 = The Clock is on and can be used.

- **FLASHS: Flash Memory Status**

0 = The Flash Memory is off and cannot be used.

1 = The Flash Memory is on and can be used.

- **FWUPIS: FWUP Input Status**

0 = FWUP input is tied low.

1 = FWUP input is tied high.

- **WKUPIS0-WKUPIS15: WKUP Input Status 0 to 15**

0 = The corresponding wake-up input is disabled, or was inactive at the time the debouncer triggered a wake up event.

1 = The corresponding wake-up input was active at the time the debouncer triggered a wake up event.

### 17.4.9 Supply Controller Flash Wake Up Timer Register

**Register Name:** SUPC_FWUTR

**Access Type:** Read-write

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

- **FWUT: Flash Wake Up Timer**

Before waking up the Flash Memory (through the FLASHON bit in SUPC_MR), this field must be correctly set. Refer to the Electrical Characteristics section of the product datasheet to obtain the wake-up time of the Flash Memory.

FWUT = (Maximum wake-up time of the Flash Memory in µs) x (Maximum Master Clock Frequency during the wake-up of the Flash Memory in MHz)/ 2.

This number must be rounded up. The value 0 is not allowed.

For example, for a maximum wake-up time of 60 µs, and a maximum MCK frequency of 3 MHz during the wake up, FWUP is: 60 x 3 / 2 = 90 = 0x5A.

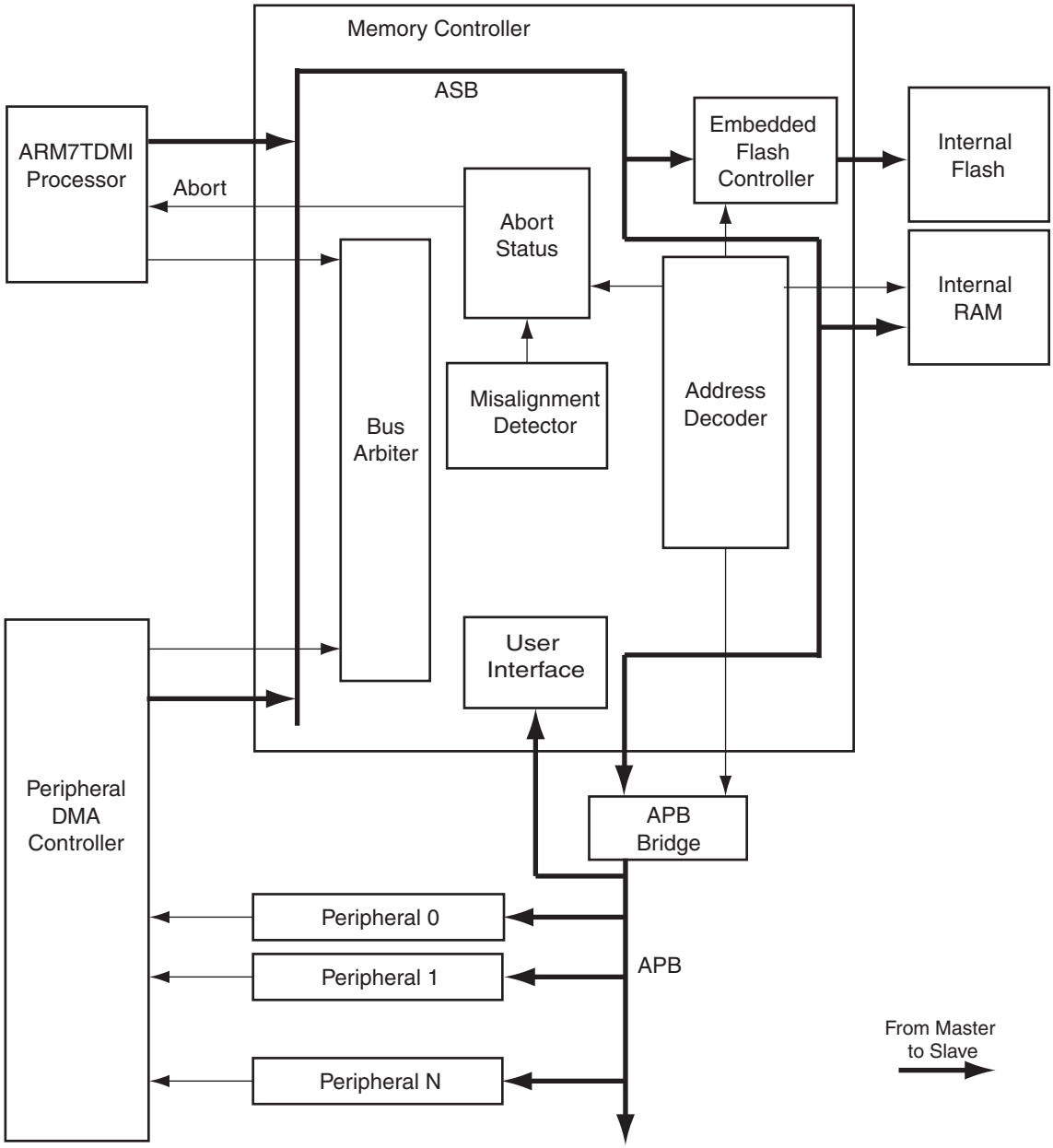
# 18. Memory Controller (MC)

## 18.1 Overview

The Memory Controller (MC) manages the ASB bus and controls accesses requested by the masters, typically the ARM7TDMI processor and the Peripheral DMA Controller. It features a bus arbiter, an address decoder, an abort status, a misalignment detector and an Embedded Flash Controller.

## 18.2 Block Diagram

**Figure 18-1.** Memory Controller Block Diagram



## 18.3 Functional Description

The Memory Controller handles the internal ASB bus and arbitrates the accesses of up to three masters.

It is made up of:

- A bus arbiter
- An address decoder
- An abort status
- A misalignment detector
- An Enhanced Embedded Flash Controller

The MC handles only little-endian mode accesses. The masters work in little-endian mode only.

### 18.3.1 Bus Arbiter

The Memory Controller has a simple, hard-wired priority bus arbiter that gives the control of the bus to one of the two masters. The Peripheral DMA Controller has the highest priority, the ARM processor has the lowest one.

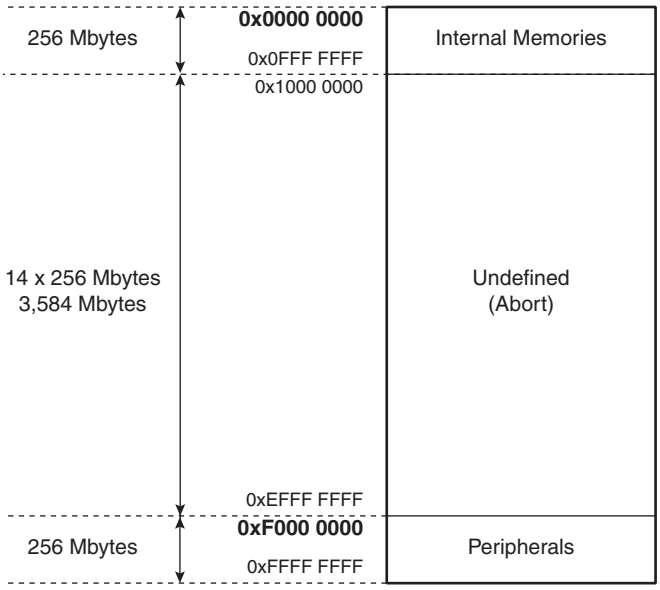
### 18.3.2 Address Decoder

The Memory Controller features an Address Decoder that first decodes the four highest bits of the 32-bit address bus and defines three separate areas:

- One 256-Mbyte address space for the internal memories
- One 256-Mbyte address space reserved for the embedded peripherals
- An undefined address space of 3584 Mbytes representing fourteen 256-Mbyte areas that return an Abort if accessed

Figure 18-2 shows the assignment of the 256-Mbyte memory areas.

**Figure 18-2.** Memory Areas



### 18.3.2.1 *Internal Memory Mapping*

Within the Internal Memory address space, the Address Decoder of the Memory Controller decodes eight more address bits to allocate 1-Mbyte address spaces for the embedded memories.

The allocated memories are accessed all along the 1-Mbyte address space and so are repeated n times within this address space, n equaling 1 Mbyte divided by the size of the memory.

When the address of the access is undefined within the internal memory area, the Address Decoder returns an Abort to the master.

### 18.3.2.2 *Internal Memory Area 0*

The first 32 bytes of Internal Memory Area 0 contain the ARM processor exception vectors, in particular, the Reset Vector at address 0x0.

Before execution of the remap command, the ROM or Flash is mapped into Internal Memory Area 0, depending on the GPNVM Bit 0 state. After the remap command, the 4Kb internal core SRAM at address 0x0020 0000 is mapped into Internal Memory Area 0. The memory mapped into Internal Memory Area 0 is accessible in both its original location and at address 0x0. The user can see the 6 Kbytes contiguously at address 0x002F F000.

Figure 18-3 and Figure 18-4 illustrate the Internal memory mapping in accrodance to the GPNVM Bit 0 state.

**Figure 18-3.** Internal Memory Mapping with GPNVM Bit 0 = 0

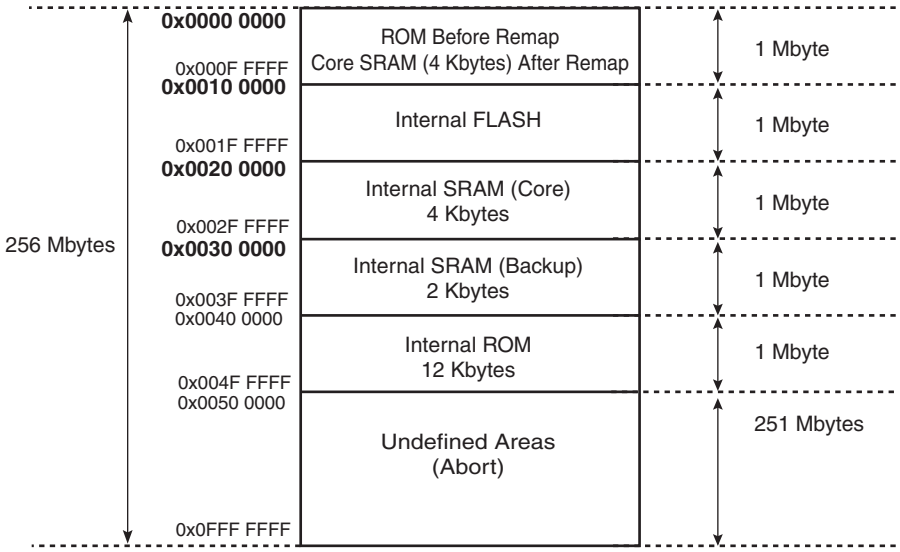
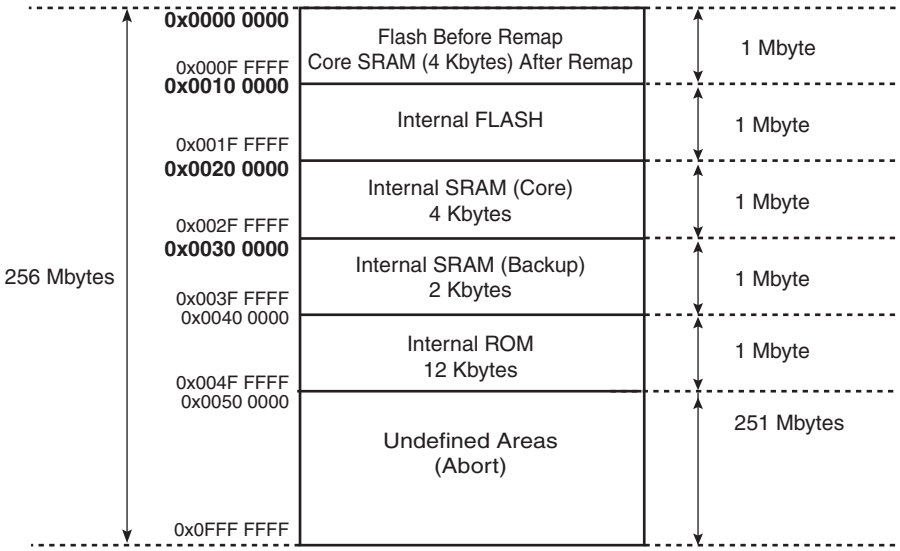


**Figure 18-4.** Internal Memory Mapping with GPNVM Bit 0 = 1




### 18.3.3 Remap Command

After execution, the Remap Command causes the Internal SRAM to be accessed through the Internal Memory Area 0.

As the ARM vectors (Reset, Abort, Data Abort, Prefetch Abort, Undefined Instruction, Interrupt, and Fast Interrupt) are mapped from address 0x0 to address 0x20, the Remap Command allows the user to redefine dynamically these vectors under software control.

The Remap Command is accessible through the Memory Controller User Interface by writing the MC_RCR (Remap Control Register) RCB field to one.

The Remap Command can be cancelled by writing the MC_RCR RCB field to one, which acts as a toggling command. This allows easy debug of the user-defined boot sequence by offering a simple way to put the chip in the same configuration as after a reset.

### 18.3.4 Abort Status

There are two reasons for an abort to occur:

- access to an undefined address
- an access to a misaligned address.

When an abort occurs, a signal is sent back to all the masters, regardless of which one has generated the access. However, only the ARM7TDMI can take an abort signal into account, and only under the condition that it was generating an access. The Peripheral DMA Controller and the EMAC do not handle the abort input signal. Note that the connections are not represented in Figure 18-1.

To facilitate debug or for fault analysis by an operating system, the Memory Controller integrates an Abort Status register set.

The full 32-bit wide abort address is saved in MC_AASR. Parameters of the access are saved in MC_ASR and include:

- the size of the request (field ABTSZ)
- the type of the access, whether it is a data read or write, or a code fetch (field ABTTYP)

- whether the access is due to accessing an undefined address (bit UNDADD) or a misaligned address (bit MISADD)
- the source of the access leading to the last abort (bits MST_EMAC, MST_PDC and MST_ARM)
- whether or not an abort occurred for each master since the last read of the register (bits SVMST_EMAC, SVMST_PDC and SVMST_ARM) unless this information is loaded in MST bits

In the case of a Data Abort from the processor, the address of the data access is stored. This is useful, as searching for which address generated the abort would require disassembling the instructions and full knowledge of the processor context.

In the case of a Prefetch Abort, the address may have changed, as the prefetch abort is pipelined in the ARM processor. The ARM processor takes the prefetch abort into account only if the read instruction is executed and it is probable that several aborts have occurred during this time. Thus, in this case, it is preferable to use the content of the Abort Link register of the ARM processor.

### 18.3.5 Enhanced Embedded Flash Controller

The Enhanced Embedded Flash Controller (EEFC) manages accesses performed by the masters of the system. It enables reading the Flash and writing the write buffer. It also contains a User Interface, mapped within the Memory Controller on the APB.

The Enhanced Embedded Flash Controller ensures the interface of the Flash block with the 32-bit internal bus. Its 128-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs thus making the software generic.

### 18.3.6 Misalignment Detector

The Memory Controller features a Misalignment Detector that checks the consistency of the accesses.

For each access, regardless of the master, the size of the access and the bits 0 and 1 of the address bus are checked. If the type of access is a word (32-bit) and the bits 0 and 1 are not 0, or if the type of the access is a half-word (16-bit) and the bit 0 is not 0, an abort is returned to the master and the access is cancelled. Note that the accesses of the ARM processor when it is fetching instructions are not checked.

The misalignments are generally due to software bugs leading to wrong pointer handling. These bugs are particularly difficult to detect in the debug phase.

As the requested address is saved in the Abort Status Register and the address of the instruction generating the misalignment is saved in the Abort Link Register of the processor, detection and fix of this kind of software bugs is simplified.

## 18.4 Memory Controller (MC) User Interface

**Base Address**: 0xFFFFFF00

**Table 18-1.** Memory Controller (MC) Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | MC Remap Control Register | MC_RCR | Write-only | |
| 0x04 | MC Abort Status Register | MC_ASR | Read-only | 0x0 |
| 0x08 | MC Abort Address Status Register | MC_AASR | Read-only | 0x0 |
| 0x10-0x5C | Reserved | | | |
| 0x60 | EFC0 Configuration Registers | See the Embedded Flash Controller Section | | |

### 18.4.1 MC Remap Control Register

**Register Name**: MC_RCR

**Access Type**: Write-only

**Offset**: 0x0

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

- **RCB: Remap Command Bit**

0: No effect.

1: This Command Bit acts on a toggle basis: writing a 1 alternatively cancels and restores the remapping of the page zero memory devices.

### 18.4.2 MC Abort Status Register

**Register Name**: MC_ASR

**Access Type**: Read-only

**Reset Value**: 0x0

**Offset**: 0x04

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | SVMST_ARM | SVMST_PDC | SVMST_EMAC |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | MST_ARM | MST_PDC | MST_EMAC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | ABTTYP | | ABTSZ | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | MISADD | UNDADD |

- **UNDADD: Undefined Address Abort Status**

0: The last abort was not due to the access of an undefined address in the address space.

1: The last abort was due to the access of an undefined address in the address space.

- **MISADD: Misaligned Address Abort Status**

0: The last aborted access was not due to an address misalignment.

1: The last aborted access was due to an address misalignment.

- **ABTSZ: Abort Size Status**

| ABTSZ | | Abort Size |
|---|---|---|
| 0 | 0 | Byte |
| 0 | 1 | Half-word |
| 1 | 0 | Word |
| 1 | 1 | Reserved |

- **ABTTYP: Abort Type Status**

| ABTTYP | | Abort Type |
|---|---|---|
| 0 | 0 | Data Read |
| 0 | 1 | Data Write |
| 1 | 0 | Code Fetch |
| 1 | 1 | Reserved |

- **MST_EMAC: EMAC Abort Source**

0: The last aborted access was not due to the EMAC.

1: The last aborted access was due to the EMAC.

- **MST_PDC: PDC Abort Source**

0: The last aborted access was not due to the PDC.

1: The last aborted access was due to the PDC.

- **MST_ARM: ARM Abort Source**

0: The last aborted access was not due to the ARM.

1: The last aborted access was due to the ARM.

- **SVMST_EMAC: Saved EMAC Abort Source**

0: No abort due to the EMAC occurred since the last read of MC_ASR or it is notified in the bit MST_EMAC.

1: At least one abort due to the EMAC occurred since the last read of MC_ASR.

- **SVMST_PDC: Saved PDC Abort Source**

0: No abort due to the PDC occurred since the last read of MC_ASR or it is notified in the bit MST_PDC.

1: At least one abort due to the PDC occurred since the last read of MC_ASR.

- **SVMST_ARM: Saved ARM Abort Source**

0: No abort due to the ARM occurred since the last read of MC_ASR or it is notified in the bit MST_ARM.

1: At least one abort due to the ARM occurred since the last read of MC_ASR.

### 18.4.3 MC Abort Address Status Register

**Register Name**: MC_AASR

**Access Type**: Read-only

**Reset Value**: 0x0

**Offset**: 0x08

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

- **ABTADD: Abort Address**

This field contains the address of the last aborted access.

# 19. Enhanced Embedded Flash Controller (EEFC)

## 19.1 Overview

The Enhanced Embedded Flash Controller (EEFC) ensures the interface of the Flash block with the 32-bit internal bus. Its 128-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

## 19.2 Product Dependencies

### 19.2.1 Power Management

The Enhanced Embedded Flash Controller (EEFC) is continuously clocked. The Power Management Controller has no effect on its behavior.

### 19.2.2 Interrupt Sources

The Enhanced Embedded Flash Controller (EEFC) interrupt line is connected to the Memory Controller internal source of the Advanced Interrupt Controller. Using the Enhanced Embedded Flash Controller (EEFC) interrupt requires the AIC to be programmed first. The EEFC interrupt is generated only on FRDY bit rising. To know the Flash status, MC Flash Status Register should be read each time a system interrupt (SYSIRQ, periph ID = 0) occurs.

## 19.3 Functional Description

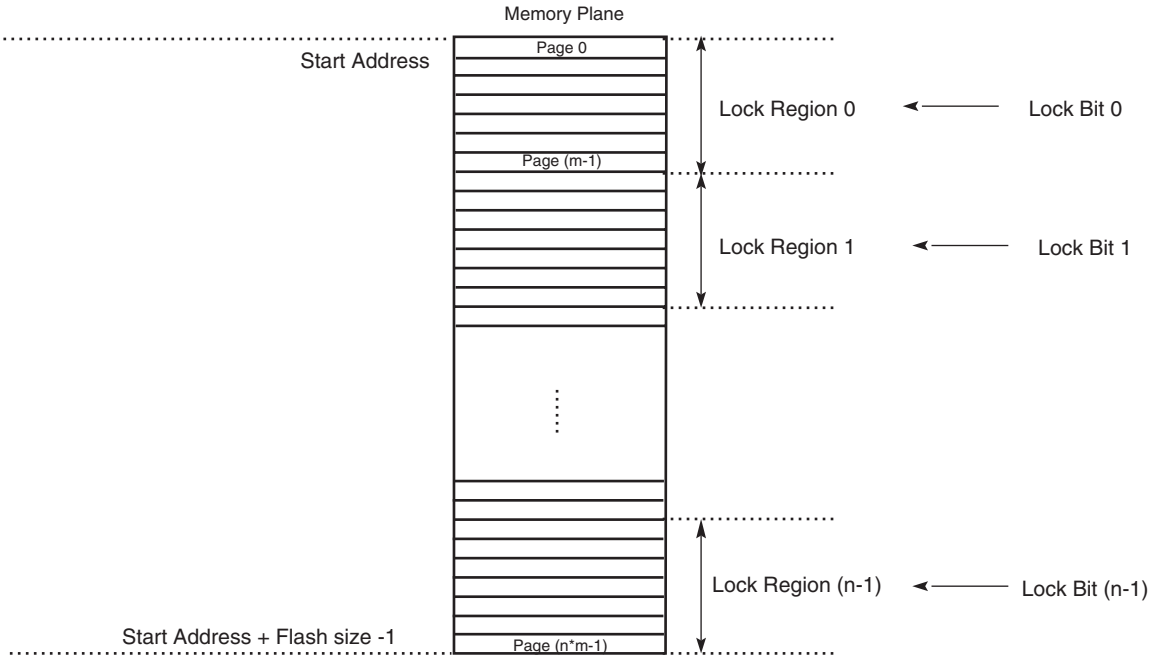
### 19.3.1 Embedded Flash Organization

The embedded Flash interfaces directly with the 32-bit internal bus. The embedded Flash is composed of:

- One memory plane organized in several pages of the same size.
- Two 128-bit read buffers used for code read optimization.
- One 128-bit read buffer used for data read optimization.
- One write buffer that manages page programming. The write buffer size is equal to the page size. This buffer is write-only and accessible all along the 1 MByte address space, so that each word can be written to its final address.
- Several lock bits used to protect write/erase operation on several pages (lock region). A lock bit is associated with a lock region composed of several pages in the memory plane.
- Several bits that may be set and cleared through the Enhanced Embedded Flash Controller (EEFC) interface, called General Purpose Non Volatile Memory bits (GPNVM bits).

The embedded Flash size, the page size, the lock regions organization and GPNVM bits definition are described in the product definition section. The Enhanced Embedded Flash Controller (EEFC) returns a descriptor of the Flash controlled after a get descriptor command issued by the application (see “Getting Embedded Flash Descriptor” on page 147).

**Figure 19-1.** Embedded Flash Organization



### 19.3.2 Read Operations

An optimized controller manages embedded Flash reads, thus increasing performance when the processor is running in ARM and Thumb mode by means of the 128-bit wide memory interface.

The Flash memory is accessible through 8-, 16- and 32-bit reads.

As the Flash block size is smaller than the address space reserved for the internal memory area, the embedded Flash wraps around the address space and appears to be repeated within it.

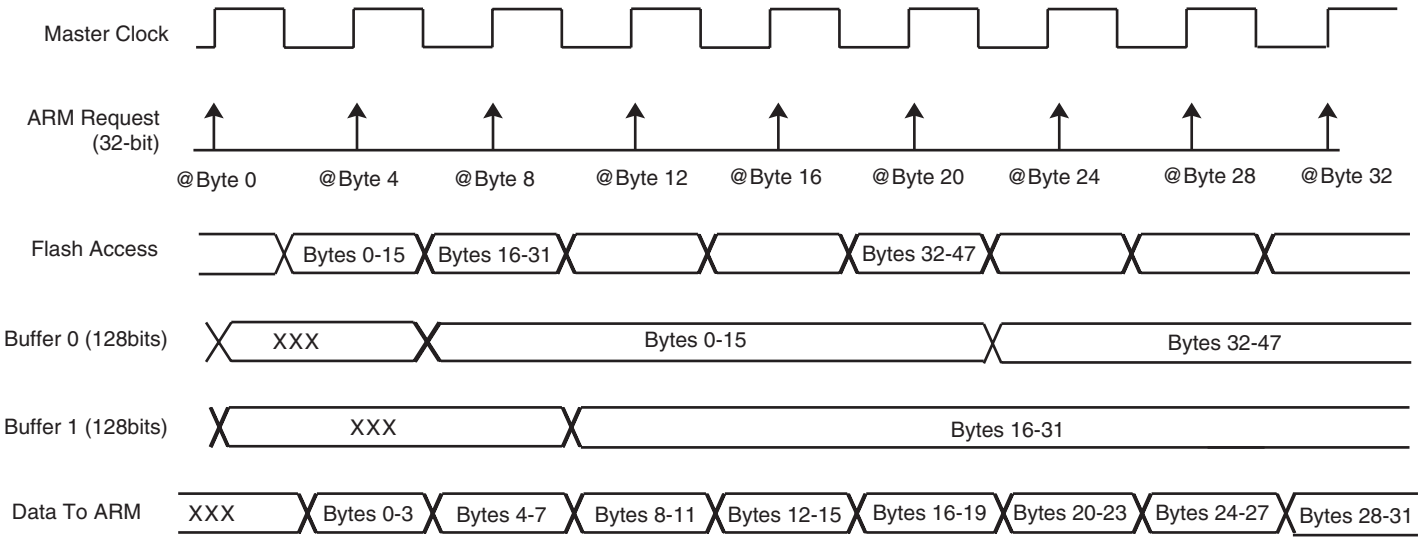
The read operations can be performed with or without wait states. Wait states must be programmed in the field FWS (Flash Read Wait State) in the Flash Mode Register (MC_FMR). Defining FWS to be 0 enables the single-cycle access of the embedded Flash. Refer to the Electrical Characteristics for more details.

#### *19.3.2.1 Code Read Optimization*

A system of 2 x 128-bit buffers is added in order to optimize sequential Code Fetch.

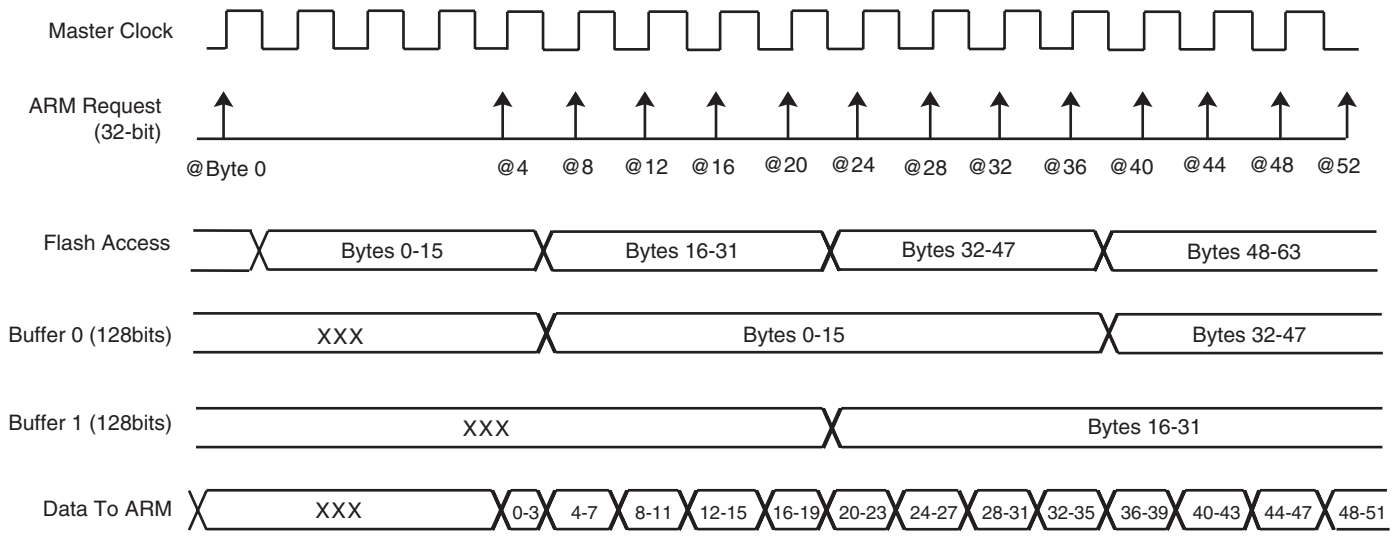
Note: Immediate consecutive code read accesses are not mandatory to benefit from this optimization.

**Figure 19-2.** Code Read Optimization in ARM Mode for FWS = 0



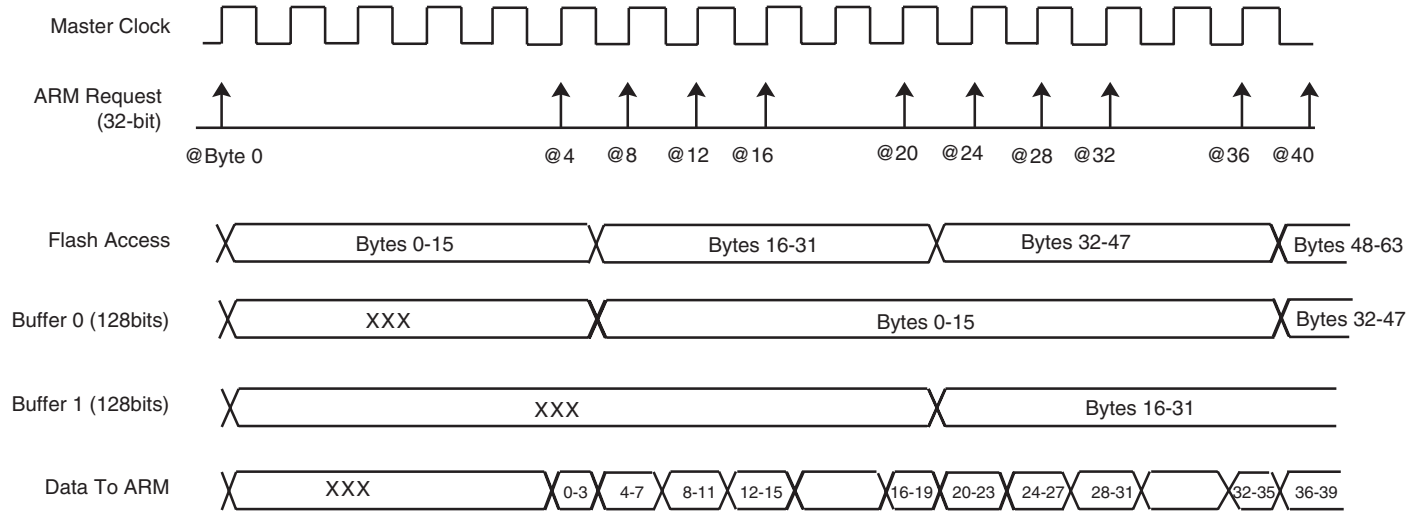

Note: When FWS is equal to 0, all the accesses are performed in a single-cycle access.

**Figure 19-3.** Code Read Optimization in ARM Mode for FWS = 3




Note: When FWS is included between 1 and 3, in case of sequential reads, the first access takes (FWS+1) cycles, the other ones only 1 cycle.

**Figure 19-4.** Code Read Optimization in ARM Mode for FWS = 4



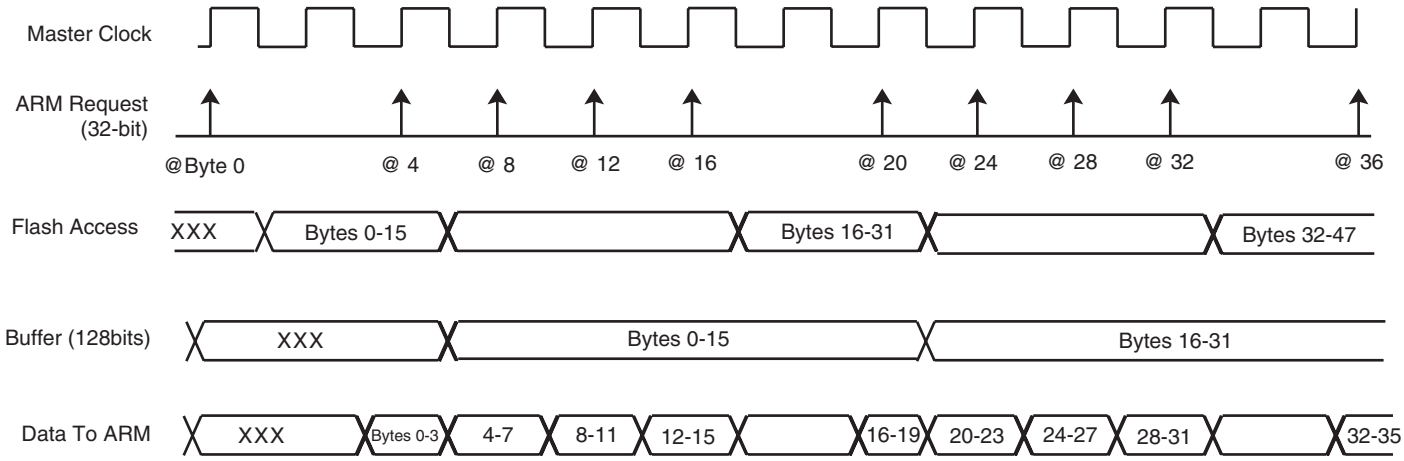

Note: When FWS is included between 4 and 10, in case of sequential reads, the first access takes (FWS+1) cycles, each first access of the 128-bit read (FWS-2) cycles, and the others only 1 cycle.

#### 19.3.2.2 *Data Read Optimization*

The organization of the Flash in 128 bits is associated with two 128-bit prefetch buffers and one 128-bit data read buffer, thus providing maximum system performance. This buffer is added in order to start access at the following data during the second read. This speeds up sequential data reads if, for example, FWS is equal to 1 (see Figure 19-5).

Note: No consecutive data read accesses are mandatory to benefit from this optimization.

**Figure 19-5.** Data Read Optimization in ARM Mode for FWS = 1




### 19.3.3 Flash Commands

The Enhanced Embedded Flash Controller (EEFC) offers a set of commands such as programming the memory Flash, locking and unlocking lock regions, consecutive programming and locking and full Flash erasing, etc.

Commands and read operations can be performed in parallel only on different memory planes. Code can be fetched from one memory plane while a write or an erase operation is performed on another.

**Table 19-1.** Set of Commands

| Command | Value | Mnemonic |
|---|---|---|
| Get Flash Descriptor | 0x0 | GETD |
| Write page | 0x1 | WP |
| Write page and lock | 0x2 | WPL |
| Erase page and write page | 0x3 | EWP |
| Erase page and write page then lock | 0x4 | EWPL |
| Erase all | 0x5 | EA |
| Set Lock Bit | 0x8 | SLB |
| Clear Lock Bit | 0x9 | CLB |
| Get Lock Bit | 0xA | GLB |

**Table 19-1.** Set of Commands (Continued)

| Command | Value | Mnemonic |
|---|---|---|
| Set GPNVM Bit | 0xB | SGPB |
| Clear GPNVM Bit | 0xC | CGPB |
| Get GPNVM Bit | 0xD | GGPB |

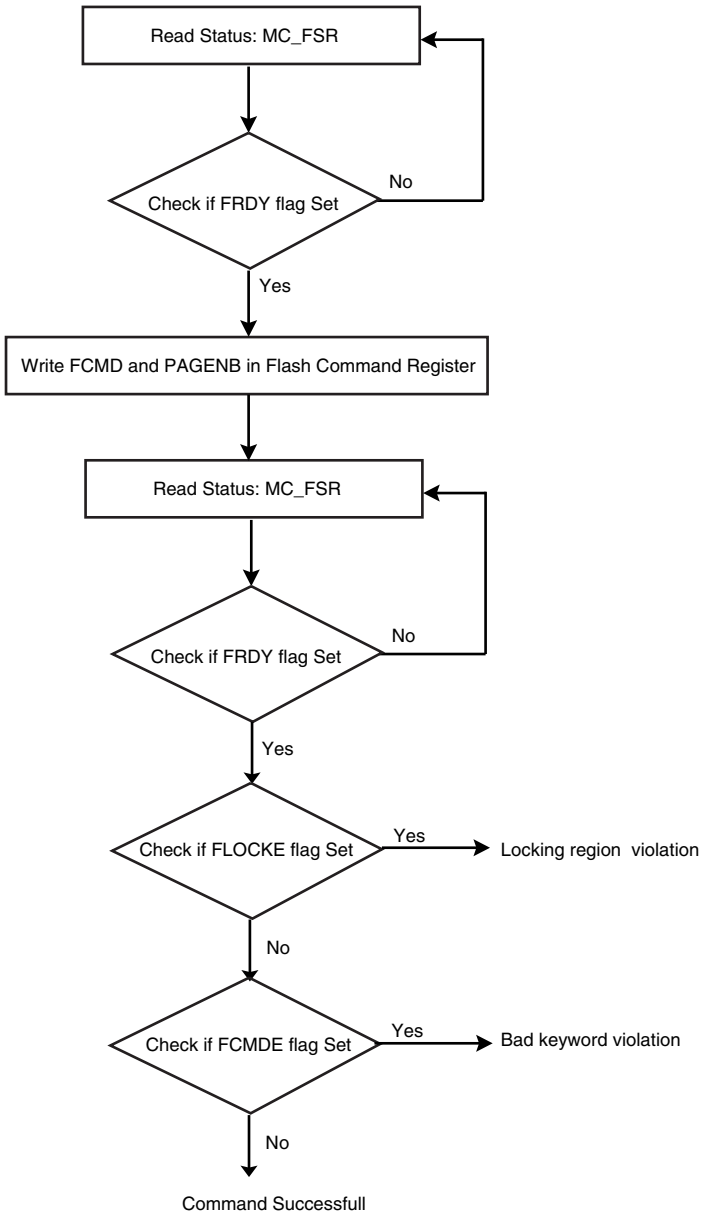
In order to perform one of these commands, the Flash Command Register (MC_FCR) has to be written with the correct command using the field FCMD. As soon as the MC_FCR register is written, **the FRDY flag and the field FVALUE in the MC_FRR register are automatically cleared**. Once the current command is achieved, then the FRDY flag is automatically set. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

All the commands are protected by the same keyword, which has to be written in the 8 highest bits of the MC_FCR register.

Writing MC_FCR with data that does not contain the correct key and/or with an invalid command has no effect on the whole memory plane, but the FCMDE flag is set in the MC_FSR register. This flag is automatically cleared by a read access to the MC_FSR register.

When the current command writes or erases a page in a locked region, the command has no effect on the whole memory plane, but the FLOCKE flag is set in the MC_FSR register. This flag is automatically cleared by a read access to the MC_FSR register.

**Figure 19-6.** Command State Chart




### *19.3.3.1 Getting Embedded Flash Descriptor*

This command allows the system to learn about the Flash organization. The system can take full advantage of this information. For instance, a device could be replaced by one with more Flash capacity, and so the software is able to adapt itself to the new configuration.

To get the embedded Flash descriptor, the application writes the GETD command in the MC_FCR register. The first word of the descriptor can be read by the software application in the MC_FRR register as soon as the FRDY flag in the MC_FSR register rises. The next reads of the MC_FRR register provide the following word of the descriptor. If extra read operations to the MC_FRR register are done after the last word of the descriptor has been returned, then the MC_FRR register value is 0 until the next valid command

.

**Table 19-2.** Flash Descriptor Definition

| Symbol | Word Index | Description |
|---|---|---|
| FL_ID | 0 | Flash Interface Description |
| FL_SIZE | 1 | Flash size in bytes |
| FL_PAGE_SIZE | 2 | Page size in bytes |
| FL_NB_PLANE | 3 | Number of planes. |
| FL_PLANE[0] | 4 | Number of bytes in the first plane. |
| ... | | |
| FL_PLANE[FL_NB_PLANE-1] | 4 + FL_NB_PLANE - 1 | Number of bytes in the last plane. |
| FL_NB_LOCK | 4 + FL_NB_PLANE | Number of lock bits. A bit is associated with a lock region. A lock bit is used to prevent write or erase operations in the lock region. |
| FL_LOCK[0] | 4 + FL_NB_PLANE + 1 | Number of bytes in the first lock region. |
| ... | | |

#### *19.3.3.2 Write Commands*

Several commands can be used to program the Flash.

Flash technology requires that an erase is done before programming. The full memory plane can be erased at the same time, or several pages can be erased at the same time (refer to "Erase Commands" on page 149). Also, a page erase can be automatically done before a page write using EWP or EWPL commands.

After programming, the page (the whole lock region) can be locked to prevent miscellaneous write or erase sequences. The lock bit can be automatically set after page programming using WPL or EWPL commands.

Data to be written are stored in an internal latch buffer. The size of the latch buffer corresponds to the page size. The latch buffer wraps around within the internal memory area address space and is repeated as many times as the number of pages within this address space.

Note: Writing of 8-bit and 16-bit data is not allowed and may lead to unpredictable data corruption.

Write operations are performed in a number of wait states equal to the number of wait states for read operations.

Data are written to the latch buffer before the programming command is written to the Flash Command Register MC_FCR. The sequence is as follows:

- Write the full page, at any page address, within the internal memory area address space.
- Programming starts as soon as the page number and the programming command are written to the Flash Command Register. The FRDY bit in the Flash Programming Status Register (MC_FSR) is automatically cleared.
- When programming is completed, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

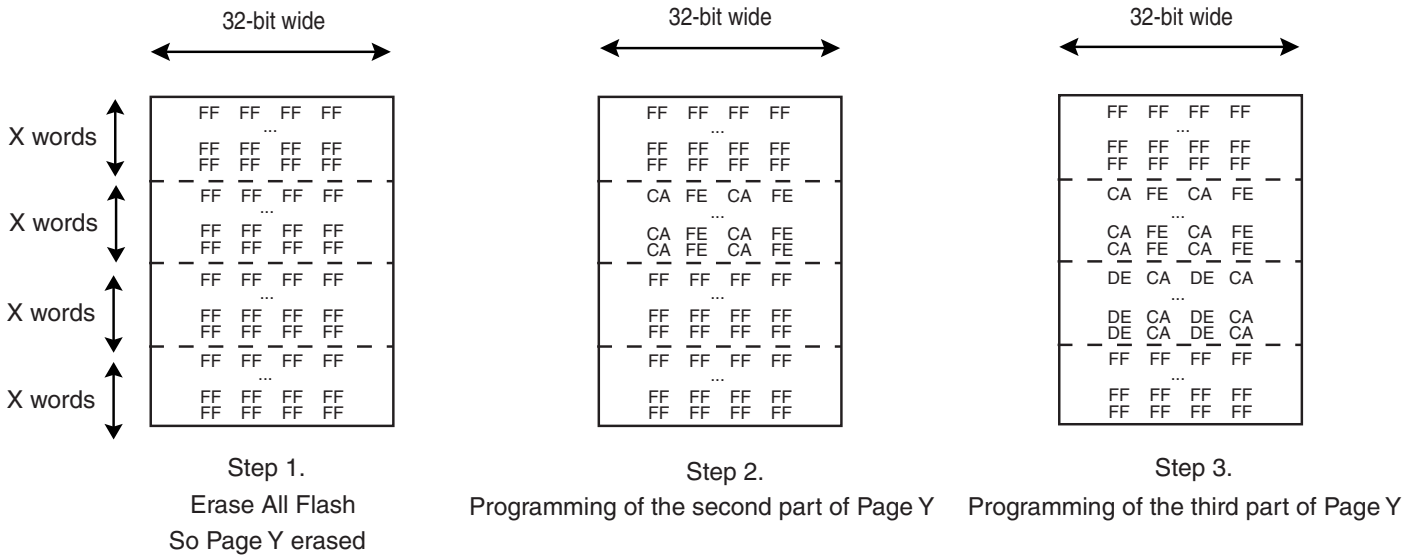
Two errors can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.

- a Lock Error: the page to be programmed belongs to a locked region. A command must be previously run to unlock the corresponding region.

By using the WP command, a page can be programmed in several steps if it has been erased before (see Figure 19-7).

**Figure 19-7.** Example of Partial Page Programming




### 19.3.3.3 *Erase Commands*

Erase commands are allowed only on unlocked regions.

The erase sequence is:

- Erase starts as soon as one of the erase commands and the FARG field are written in the Flash Command Register.
- When the programming completes, the FRDY bit in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.

Two errors can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.
- a Lock Error: at least one page to be erased belongs to a locked region. The erase command has been refused, no page has been erased. A command must be previously run to unlock the corresponding region.

### 19.3.3.4 *Lock Bit Protection*

Lock bits are associated with several pages in the embedded Flash memory plane. This defines lock regions in the embedded Flash memory plane. They prevent writing/erasing protected pages.

The lock sequence is:

- The Set Lock command (SLB) and a page number to be protected are written in the Flash Command Register.

- When the locking completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- If the lock bit number is greater than the total number of lock bits, then the command has no effect. The result of the SLB command can be checked running a GLB (Get Lock Bit) command.

One error can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.

It is possible to clear lock bits previously set. Then the locked region can be erased or programmed. The unlock sequence is:

- The Clear Lock command (CLB) and a page number to be unprotected are written in the Flash Command Register.
- When the unlock completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- If the lock bit number is greater than the total number of lock bits, then the command has no effect.

One error can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.

The status of lock bits can be returned by the Enhanced Embedded Flash Controller (EEFC). The Get Lock Bit status sequence is:

- The Get Lock Bit command (GLB) is written in the Flash Command Register. FARG field is meaningless.
- When the command completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- Lock bits can be read by the software application in the MC_FRR register. The first word read corresponds to the 32 first lock bits, next reads providing the next 32 lock bits as long as it is meaningful. Extra reads to the MC_FRR register return 0.

For example, if the third bit of the first word read in the MC_FRR is set, then the third lock region is locked.

One error can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.

Note: Access to the Flash in read is permitted when a set, clear or get lock bit command is performed.

#### *19.3.3.5 GPNVM Bit*

GPNVM bits do not interfere with the embedded Flash memory plane. Refer to the product definition section for information on the GPNVM Bit Action.

The set GPNVM bit sequence is:

- Start the Set GPNVM Bit command (SGPB) by writing the Flash Command Register with the SGPB command and the number of the GPNVM bit to be set.

- When the GPVNM bit is set, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt was enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- If the GPNVM bit number is greater than the total number of GPNVM bits, then the command has no effect. The result of the SGPB command can be checked by running a GGPB (Get GPNVM Bit) command.

One error can be detected in the MC_FSR register after a programming sequence:

- A Command Error: a bad keyword has been written in the MC_FCR register.

It is possible to clear GPNVM bits previously set. The clear GPNVM bit sequence is:

- Start the Clear GPNVM Bit command (CGPB) by writing the Flash Command Register with CGPB and the number of the GPNVM bit to be cleared.
- When the clear completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- If the GPNVM bit number is greater than the total number of GPNVM bits, then the command has no effect.

One error can be detected in the MC_FSR register after a programming sequence:

- A Command Error: a bad keyword has been written in the MC_FCR register.

The status of GPNVM bits can be returned by the Enhanced Embedded Flash Controller (EEFC). The sequence is:

- Start the Get GPNVM bit command by writing the Flash Command Register with GGPB. The FARG field is meaningless.
- When the command completes, the bit FRDY in the Flash Programming Status Register (MC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in MC_FMR, the interrupt line of the Memory Controller is activated.
- GPNVM bits can be read by the software application in the MC_FRR register. The first word read corresponds to the 32 first GPNVM bits, following reads provide the next 32 GPNVM bits as long as it is meaningful. Extra reads to the MC_FRR register return 0.

For example, if the third bit of the first word read in the MC_FRR is set, then the third GPNVM bit is active.

One error can be detected in the MC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the MC_FCR register.

Note: Access to the Flash in read is permitted when a set, clear or get GPNVM bit command is performed.

#### *19.3.3.6 Security Bit Protection*

When the security is enabled, access to the Flash, either through the ICE interface or through the Fast Flash Programming Interface, is forbidden. This ensures the confidentiality of the code programmed in the Flash.

The security bit is GPNVM0.

Disabling the security bit can only be achieved by asserting the ERASE pin at 1, and after a full Flash erase is performed. When the security bit is deactivated, all accesses to the Flash are permitted.

## 19.4 Enhanced Embedded Flash Controller (EEFC) User Interface

The User Interface of the Enhanced Embedded Flash Controller (EEFC) is integrated within the Memory Controller with base address 0xFFFF FF60.

**Table 19-3.** Register Mapping

| Offset | Register | Name | Access | Reset State |
|---|---|---|---|---|
| 0x00 | MC Flash Mode Register | MC_FMR | Read-write | 0x0 |
| 0x04 | MC Flash Command Register | MC_FCR | Write-only | – |
| 0x08 | MC Flash Status Register | MC_FSR | Read-only | 0x00000001 |
| 0x0C | MC Flash Result Register | MC_FRR | Read-only | 0x0 |
| 0x10 | Reserved | – | – | – |

### 19.4.1 MC Flash Mode Register

**Register Name**: MC_FMR

**Access Type**: Read-write

**Offset**: 0x60

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

- **FRDY: Ready Interrupt Enable**

0: Flash Ready does not generate an interrupt.

1: Flash Ready (to accept a new command) generates an interrupt.

- **FWS: Flash Wait State**

This field defines the number of wait states for read and write operations:

Number of cycles for Read/Write operations = FWS+1

### 19.4.2 MC Flash Command Register

**Register Name**: MC_FCR

**Access Type**: Write-only

**Offset**: 0x64

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

- **FCMD: Flash Command**

This field defines the flash commands. Refer to “Flash Commands” on page 145.

- **FARG: Flash Command Argument**

| | |
|---|---|
| Erase command | For erase all command, this field is meaningless. |
| Programming command | FARG defines the page number to be programmed. |
| Lock command | FARG defines the page number to be locked. |
| GPNVM command | FARG defines the GPNVM number. |
| Get Commands | Field is meaningless. |

- **FKEY: Flash Writing Protection Key**

This field should be written with the value 0x5A to enable the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.

### 19.4.3 MC Flash Status Register

**Register Name**: MC_FSR

**Access Type**: Read-only

**Offset**: 0x68

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | FLOCKE | FCMDE | FRDY |

- **FRDY: Flash Ready Status**

0: The Enhanced Embedded Flash Controller (EEFC) is busy.

1: The Enhanced Embedded Flash Controller (EEFC) is ready to start a new command.

When it is set, this flags triggers an interrupt if the FRDY flag is set in the MC_FMR register.

This flag is automatically cleared when the Enhanced Embedded Flash Controller (EEFC) is busy.

- **FCMDE: Flash Command Error Status**

0: No invalid commands and no bad keywords were written in the Flash Mode Register MC_FMR.

1: An invalid command and/or a bad keyword was/were written in the Flash Mode Register MC_FMR.

This flag is automatically cleared when MC_FSR is read or MC_FCR is written.

- **FLOCKE: Flash Lock Error Status**

0: No programming/erase of at least one locked region has happened since the last read of MC_FSR.

1: Programming/erase of at least one locked region has happened since the last read of MC_FSR.

This flag is automatically cleared when MC_FSR is read or MC_FCR is written.

### 19.4.4 MC Flash Result Register

**Register Name**: MC_FRR

**Access Type**: Read-only

**Offset**: 0x6C

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

- **FVALUE: Flash Result Value**

The result of a Flash command is returned in this register. If the size of the result is greater than 32 bits, then the next resulting value is accessible at the next register read.

# 20. Fast Flash Programming Interface (FFPI)

## 20.1 Overview

The Fast Flash Programming Interface provides two solutions - parallel or serial - for high-volume programming using a standard gang programmer. The parallel interface is fully handshaked and the device is considered to be a standard EEPROM. Additionally, the parallel protocol offers an optimized access to all the embedded Flash functionalities. The serial interface uses the standard IEEE 1149.1 JTAG protocol. It offers an optimized access to all the embedded Flash functionalities.

Although the Fast Flash Programming Mode is a dedicated mode for high volume programming, this mode is not designed for in-situ programming.

## 20.2 Parallel Fast Flash Programming

### 20.2.1 Device Configuration

In Fast Flash Programming Mode, the device is in a specific test mode. Only a certain set of pins is significant. Other pins must be left unconnected.

**Figure 20-1.** Parallel Programming Interface

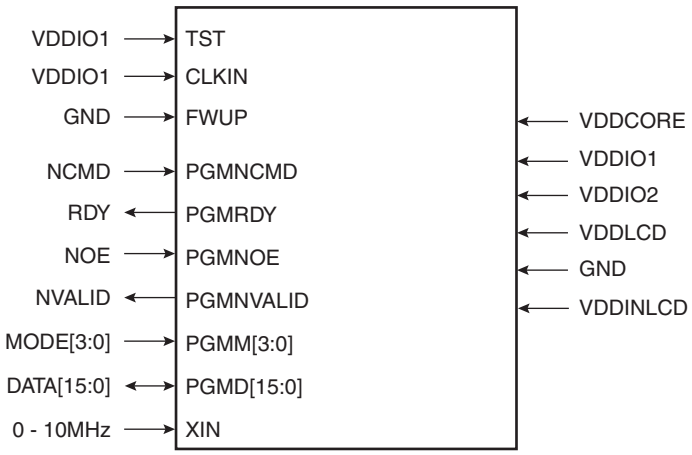
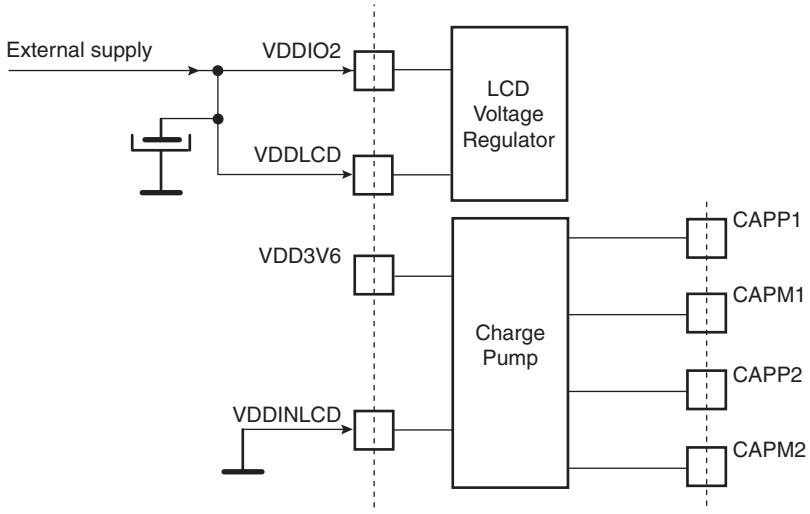


**Table 20-1.** Signal Description List

| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **Power** | | | | |
| VDDIO1 | I/O Lines Power Supply | Power | | Apply externally 2.2V-3.6V (1) |
| VDDIO2 | I/O Lines Power Supply | Power | | Apply externally 2.2V-3.6V (1) |
| VDDCORE | Core Power Supply | Power | | Apply externally 1.80V-1.95V (1) |
| VDDOUT | Voltage Regulator Output | Power | | Connect to VDDCORE. 2.2 µF decoupling capacitor needed |
| VDDINLCD | Charge pump input | Power | | Connect to ground |
| VDD3V6 | Charge pump output | Power | | Left unconnected (1) |
| VDDLCD | LCD voltage input | Power | | Connect to VDDIO2 (1) |
| GND | Ground | Ground | | |
| **Clocks** | | | | |
| XIN | Clock Input | Input | | 0 to 10MHz (0-VDDIO1 square wave) |
| **Test** | | | | |
| TST | Test Mode Select | Input | High | Must be connected to VDDIO1 |
| CLKIN | External clock input used to enter in FFPI mode | Input | High | Must be connected to VDDIO1 |
| FWUP | Wake-up pin | Input | Low | Must be connected to GND |
| **PIO** | | | | |
| PGMNCMD | Valid command available | Input | Low | Pulled-up input at reset |
| PGMRDY | 0: Device is busy<br>1: Device is ready for a new command | Output | High | Pulled-up input at reset |
| PGMNOE | Output Enable (active high) | Input | Low | Pulled-up input at reset |
| PGMNVALID | 0: DATA[15:0] is in input mode<br>1: DATA[15:0] is in output mode | Output | Low | Pulled-up input at reset |
| PGMM[3:0] | Specifies DATA type (See Table 20-2) | Input | | Pulled-up input at reset |
| PGMD[15:0] | Bi-directional data bus | Input/Output | | Pulled-up input at reset |

Note: 1. See Figure 20-2 below.

**Figure 20-2.** The Charge Pump and the LCD Regulator are Not Used




### 20.2.2 Signal Names

Depending on the MODE settings, DATA is latched in different internal registers.

**Table 20-2.** Mode Coding

| MODE[3:0] | Symbol | Data |
|---|---|---|
| 0000 | CMDE | Command Register |
| 0001 | ADDR0 | Address Register LSBs |
| 0010 | ADDR1 | Address Register MSBs |
| 0101 | DATA | Data Register |
| Default | IDLE | No register |

When MODE is equal to CMDE, then a new command (strobed on DATA[15:0] signals) is stored in the command register.

**Table 20-3.** Command Bit Coding

| DATA[15:0] | Symbol | Command Executed |
|---|---|---|
| 0x0011 | READ | Read Flash |
| 0x0012 | WP | Write Page Flash |
| 0x0022 | WPL | Write Page and Lock Flash |
| 0x0032 | EWP | Erase Page and Write Page |
| 0x0042 | EWPL | Erase Page and Write Page then Lock |
| 0x0013 | EA | Erase All |
| 0x0014 | SLB | Set Lock Bit |
| 0x0024 | CLB | Clear Lock Bit |
| 0x0015 | GLB | Get Lock Bit |
| 0x0034 | SGPB | Set General Purpose NVM bit |

**Table 20-3.** Command Bit Coding (Continued)

| DATA[15:0] | Symbol | Command Executed |
|---|---|---|
| 0x0044 | CGPB | Clear General Purpose NVM bit |
| 0x0025 | GGPB | Get General Purpose NVM bit |
| 0x0054 | SSE | Set Security Bit |
| 0x0035 | GSE | Get Security Bit |
| 0x001F | WRAM | Write Memory |
| 0x001E | GVE | Get Version |

### 20.2.3 Entering Programming Mode

The following algorithm puts the device in Parallel Programming Mode:

- Apply GND, TST, CLKIN, FWUP and the supplies as described in table 4.1.
- Apply XIN clock
- Wait for 20 ms
- Start a read or write handshaking.

### 20.2.4 Programmer Handshaking

A handshake is defined for read and write operations. When the device is ready to start a new operation (RDY signal set), the programmer starts the handshake by clearing the NCMD signal. The handshaking is achieved once NCMD signal is high and RDY is high.

#### *20.2.4.1 Write Handshaking*

For details on the write handshaking sequence, refer to Figure 20-3 and Table 20-4.

**Figure 20-3.** Parallel Programming Timing, Write Sequence

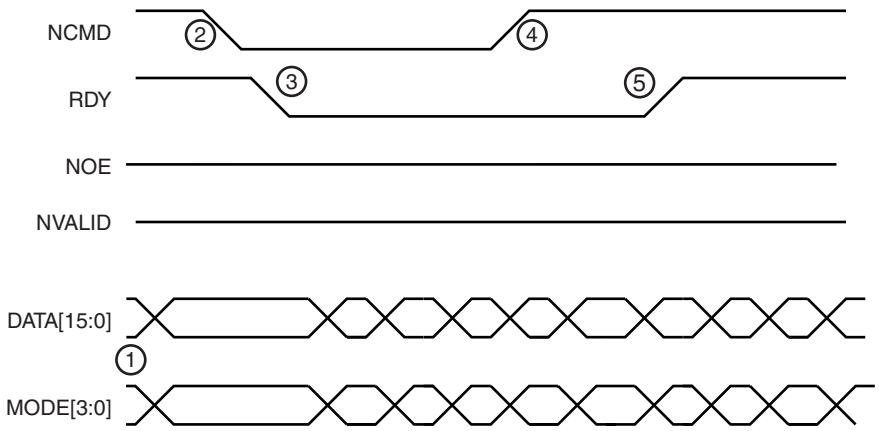


**Table 20-4.** Write Handshake

| Step | Programmer Action | Device Action | Data I/O |
|---|---|---|---|
| 1 | Sets MODE and DATA signals | Waits for NCMD low | Input |
| 2 | Clears NCMD signal | Latches MODE and DATA | Input |
| 3 | Waits for RDY low | Clears RDY signal | Input |
| 4 | Releases MODE and DATA signals | Executes command and polls NCMD high | Input |
| 5 | Sets NCMD signal | Executes command and polls NCMD high | Input |
| 6 | Waits for RDY high | Sets RDY | Input |

*20.2.4.2 Read Handshaking*

For details on the read handshaking sequence, refer to Figure 20-4 and Table 20-5.

**Figure 20-4.** Parallel Programming Timing, Read Sequence

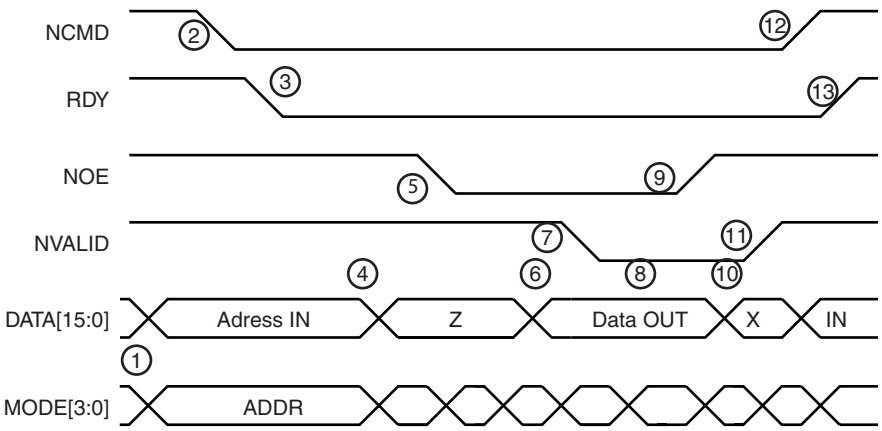



**Table 20-5.** Read Handshake

| Step | Programmer Action | Device Action | DATA I/O |
|---|---|---|---|
| 1 | Sets MODE and DATA signals | Waits for NCMD low | Input |
| 2 | Clears NCMD signal | Latch MODE and DATA | Input |
| 3 | Waits for RDY low | Clears RDY signal | Input |
| 4 | Sets DATA signal in tristate | Waits for NOE Low | Input |
| 5 | Clears NOE signal | | Tristate |
| 6 | Waits for NVALID low | Sets DATA bus in output mode and outputs the flash contents. | Output |
| 7 | | Clears NVALID signal | Output |
| 8 | Reads value on DATA Bus | Waits for NOE high | Output |

**Table 20-5.** Read Handshake (Continued)

| Step | Programmer Action | Device Action | DATA I/O |
|---|---|---|---|
| 9 | Sets NOE signal | | Output |
| 10 | Waits for NVALID high | Sets DATA bus in input mode | X |
| 11 | Sets DATA in output mode | Sets NVALID signal | Input |
| 12 | Sets NCMD signal | Waits for NCMD high | Input |
| 13 | Waits for RDY high | Sets RDY signal | Input |

### 20.2.5 Device Operations

Several commands on the Flash memory are available. These commands are summarized in Table 20-3 on page 159. Each command is driven by the programmer through the parallel interface running several read/write handshaking sequences.

When a new command is executed, the previous one is automatically achieved. Thus, chaining a read command after a write automatically flushes the load buffer in the Flash.

#### *20.2.5.1 Flash Read Command*

This command is used to read the contents of the Flash memory. The read command can start at any valid address in the memory plane and is optimized for consecutive reads. Read handshaking can be chained; an internal address buffer is automatically increased.

**Table 20-6.** Read Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | READ |
| 2 | Write handshaking | ADDR0 | Memory Address LSB |
| 3 | Write handshaking | ADDR1 | Memory Address |
| 4 | Read handshaking | DATA | *Memory Address++ |
| 5 | Read handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |
| n | Write handshaking | ADDR0 | Memory Address LSB |
| n+1 | Write handshaking | ADDR1 | Memory Address |
| n+2 | Read handshaking | DATA | *Memory Address++ |
| n+3 | Read handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |

#### *20.2.5.2 Flash Write Command*

This command is used to write the Flash contents.

The Flash memory plane is organized into several pages. Data to be written are stored in a load buffer that corresponds to a Flash memory page. The load buffer is automatically flushed to the Flash:

- before access to any page other than the current one
- when a new command is validated (MODE = CMDE)

The **Write Page** command **(WP)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

**Table 20-7.** Write Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | WP or WPL or EWP or EWPL |
| 2 | Write handshaking | ADDR0 | Memory Address LSB |
| 3 | Write handshaking | ADDR1 | Memory Address |
| 4 | Write handshaking | DATA | *Memory Address++ |
| 5 | Write handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |
| n | Write handshaking | ADDR0 | Memory Address LSB |
| n+1 | Write handshaking | ADDR1 | Memory Address |
| n+2 | Write handshaking | DATA | *Memory Address++ |
| n+3 | Write handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |

The Flash command **Write Page and Lock (WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

The Flash command **Erase Page and Write (EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

The Flash command **Erase Page and Write the Lock (EWPL)** combines EWP and WPL commands.

### *20.2.5.3 Flash Full Erase Command*

This command is used to erase the Flash memory planes.

All lock regions must be unlocked before the Full Erase command by using the CLB command. Otherwise, the erase command is aborted and no page is erased.

**Table 20-8.** Full Erase Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | EA |
| 2 | Write handshaking | DATA | 0 |

### *20.2.5.4 Flash Lock Commands*

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated. A Bit Mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first lock bit is activated.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits. All the lock bits are also cleared by the EA command.

**Table 20-9.** Set and Clear Lock Bit Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | SLB or CLB |
| 2 | Write handshaking | DATA | Bit Mask |

Lock bits can be read using **Get Lock Bit** command **(GLB)**. The $n^{th}$ lock bit is active when the bit n of the bit mask is set..

**Table 20-10.** Get Lock Bit Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | GLB |
| 2 | Read handshaking | DATA | Lock Bit Mask Status<br>0 = Lock bit is cleared<br>1 = Lock bit is set |

### *20.2.5.5 Flash General-purpose NVM Commands*

General-purpose NVM bits (GP NVM bits) can be set using the **Set GPNVM** command **(SGPB)**. This command also activates GP NVM bits. A bit mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first GP NVM bit is activated.

In the same way, the **Clear GPNVM** command **(CGPB)** is used to clear general-purpose NVM bits. All the general-purpose NVM bits are also cleared by the EA command. The general-purpose NVM bit is deactivated when the corresponding bit in the pattern value is set to 1.

**Table 20-11.** Set/Clear GP NVM Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | SGPB or CGPB |
| 2 | Write handshaking | DATA | GP NVM bit pattern value |

General-purpose NVM bits can be read using the **Get GPNVM Bit** command **(GGPB)**. The $n^{th}$ GP NVM bit is active when bit n of the bit mask is set..

**Table 20-12.** Get GP NVM Bit Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | GGPB |
| 2 | Read handshaking | DATA | GP NVM Bit Mask Status<br>0 = GP NVM bit is cleared<br>1 = GP NVM bit is set |

### *20.2.5.6 Flash Security Bit Command*

A security bit can be set using the **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. An event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

**Table 20-13.** Set Security Bit Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | SSE |
| 2 | Write handshaking | DATA | 0 |

Once the security bit is set, it is not possible to access FFPI. The only way to erase the security bit is to erase the Flash.

In order to erase the Flash, the user must perform the following:

- Power-off the chip
- Power-on the chip with TST = 0 and FWUP =0
- Assert Erase during a period of more than 220 ms
- Power-off the chip

Then it is possible to return to FFPI mode and check that Flash is erased.

### *20.2.5.7 Memory Write Command*

This command is used to perform a write access to any memory location.

The **Memory Write** command **(WRAM)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

**Table 20-14.** Write Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | WRAM |
| 2 | Write handshaking | ADDR0 | Memory Address LSB |
| 3 | Write handshaking | ADDR1 | Memory Address |
| 4 | Write handshaking | DATA | *Memory Address++ |
| 5 | Write handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |
| n | Write handshaking | ADDR0 | Memory Address LSB |
| n+1 | Write handshaking | ADDR1 | Memory Address |
| n+2 | Write handshaking | DATA | *Memory Address++ |
| n+3 | Write handshaking | DATA | *Memory Address++ |
| ... | ... | ... | ... |

### *20.2.5.8 Get Version Command*

The **Get Version** (GVE) command retrieves the version of the FFPI interface.

**Table 20-15.** Get Version Command

| Step | Handshake Sequence | MODE[3:0] | DATA[15:0] |
|---|---|---|---|
| 1 | Write handshaking | CMDE | GVE |
| 2 | Write handshaking | DATA | Version |

## 20.3 Serial Fast Flash Programming

The Serial Fast Flash programming interface is based on IEEE Std. 1149.1 “Standard Test Access Port and Boundary-Scan Architecture”. Refer to this standard for an explanation of terms used in this chapter and for a description of the TAP controller states.

In this mode, data read/written from/to the embedded Flash of the device are transmitted through the JTAG interface of the device.

### 20.3.1 Device Configuration

In Serial Fast Flash Programming Mode, the device is in a specific test mode. Only a distinct set of pins is significant. Other pins must be left unconnected.

**Figure 20-5.** Serial Programming

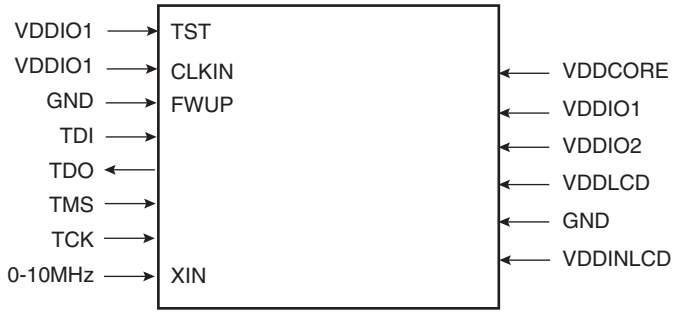



**Table 20-16.** Signal Description List

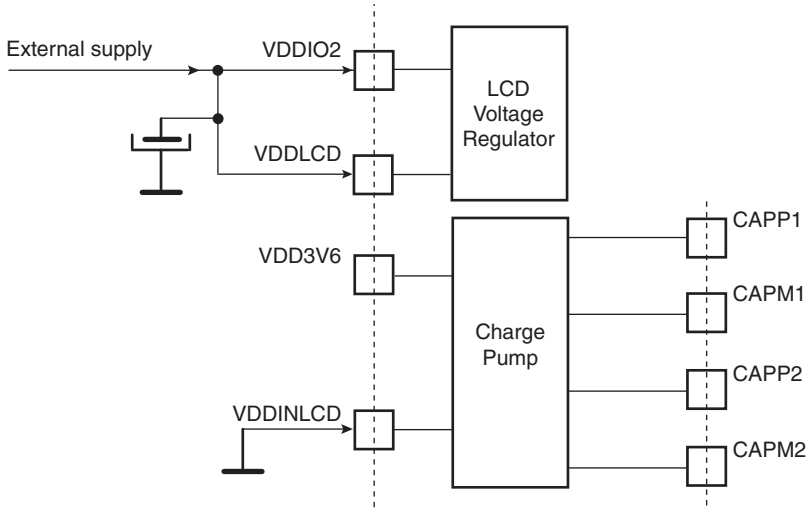
| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **Power** | | | | |
| VDDIO1 | I/O Lines Power Supply | Power | | Apply externally 2.2V-3.6V (1) |
| VDDIO2 | I/O Lines Power Supply | Power | | Apply externally 2.2V-3.6V (1) |
| VDDCORE | Core Power Supply | Power | | Apply externally 1.80V-1.95V (1) |
| VDDOUT | Voltage Regulator Output | Power | | Connect to VDDCORE. 2.2 µF decoupling capacitor needed |
| VDDINLCD | Charge pump input | Power | | Connect to ground |
| VDD3V6 | Charge pump output | Power | | Left unconnected (1) |
| VDDLCD | LCD voltage input | Power | | Connect to VDDIO2 (1) |
| GND | Ground | Ground | | |
| **Clocks** | | | | |
| XIN | Clock Input | Input | | 0 to 10MHz (0-VDDIO1 square wave) |
| **Test** | | | | |
| TST | Test Mode Select | Input | High | Must be connected to VDDIO1 |
| CLKIN | External clock input used to enter in FFPI mode | Input | High | Must be connected to VDDIO1 |
| FWUP | Wake-up pin | Input | Low | Must be connected to GND |

**Table 20-16.** Signal Description List (Continued)

| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **JTAG** | | | | |
| TCK | JTAG TCK | Input | - | Pulled-up input at reset |
| TDI | JTAG Test Data In | Input | - | Pulled-up input at reset |
| TDO | JTAG Test Data Out | Output | - | |
| TMS | JTAG Test Mode Select | Input | - | Pulled-up input at reset |

Note: 1. See Figure 20-6 below.

**Figure 20-6.** The Charge Pump and the LCD Regulator are Not Used




### 20.3.2 Entering Serial Programming Mode

The following algorithm puts the device in Serial Programming Mode:

- Apply GND, TST, CLKIN, FWUP and the supplies as described in Table 20-1, “Signal Description List,” on page 158.
- Apply XIN clock.
- Wait for 10 ms.
- Reset the TAP controller clocking 5 TCK pulses with TMS set.
- Shift 0x2 into the IR register (IR is 4 bits long, LSB first) without going through the Run-Test-Idle state.
- Shift 0x2 into the DR register (DR is 4 bits long, LSB first) without going through the Run-Test-Idle state.
- Shift 0xC into the IR register (IR is 4 bits long, LSB first) without going through the Run-Test-Idle state.

Note: After reset, the device is clocked by the internal RC oscillator. Before clearing RDY signal, if an external clock ( > 32 kHz) is connected to XIN, then the device will switch on the external clock. Else, XIN input is not considered. An higher frequency on XIN speeds up the programmer handshake.

**Table 20-17.** Reset TAP Controller and Go to Select-DR-Scan

| TDI | TMS | TAP Controller State |
|---|---|---|
| X | 1 | |
| X | 1 | |
| X | 1 | |
| X | 1 | |
| X | 1 | Test-Logic Reset |
| X | 0 | Run-Test/Idle |
| Xt | 1 | Select-DR-Scan |

### 20.3.3 Read/Write Handshake

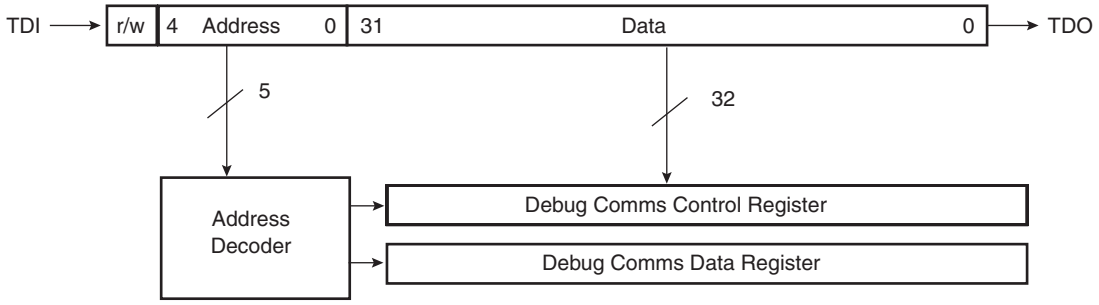
The read/write handshake is done by carrying out read/write operations on two registers of the device that are accessible through the JTAG:

- Debug Comms Control Register: DCCR
- Debug Comms Data Register: DCDR

Access to these registers is done through the TAP 38-bit DR register comprising a 32-bit data field, a 5-bit address field and a read/write bit. The data to be written is scanned into the 32-bit data field with the address of the register to the 5-bit address field and 1 to the read/write bit. A register is read by scanning its address into the address field and 0 into the read/write bit, going through the UPDATE-DR TAP state, then scanning out the data.

Refer to the ARM7TDMI reference manuel for more information on Comm channel operations.

**Figure 20-7.** TAP 8-bit DR Register




A read or write takes place when the TAP controller enters UPDATE-DR state. Refer to the IEEE 1149.1 for more details on JTAG operations.

- The address of the Debug Comms Control Register is 0x04.
- The address of the Debug Comms Data Register is 0x05.
  The Debug Comms Control Register is read-only and allows synchronized handshaking between the processor and the debugger.
  - Bit 1 (W): Denotes whether the programmer can read a data through the Debug Comms Data Register. If the device is busy W = 0, then the programmer must poll until W = 1.

– Bit 0 (R): Denotes whether the programmer can send data from the Debug Comms Data Register. If R = 1, data previously placed there through the scan chain has not been collected by the device and so the programmer must wait.

The write handshake is done by polling the Debug Comms Control Register until the R bit is cleared. Once cleared, data can be written to the Debug Comms Data Register.

The read handshake is done by polling the Debug Comms Control Register until the W bit is set. Once set, data can be read in the Debug Comms Data Register.

### 20.3.4 Device Operations

Several commands on the Flash memory are available. These commands are summarized in Table 20-3 on page 159. Commands are run by the programmer through the serial interface that is reading and writing the Debug Comms Registers.

#### *20.3.4.1 Flash Read Command*

This command is used to read the Flash contents. The memory map is accessible through this command. Memory is seen as an array of words (32-bit wide). The read command can start at any valid address in the memory plane. **This address must be word-aligned**. The address is automatically incremented.

**Table 20-18.** Read Command

| Read/Write | DR Data |
|---|---|
| Write | (Number of Words to Read) << 16 \| READ |
| Write | Address |
| Read | Memory [address] |
| Read | Memory [address+4] |
| ... | ... |
| Read | Memory [address+(Number of Words to Read - 1)* 4] |

#### *20.3.4.2 Flash Write Command*

This command is used to write the Flash contents. The address transmitted must be a valid Flash address in the memory plane.

The Flash memory plane is organized into several pages. Data to be written is stored in a load buffer that corresponds to a Flash memory page. The load buffer is automatically flushed to the Flash:

- before access to any page than the current one
- at the end of the number of words transmitted

The **Write Page** command **(WP)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

**Table 20-19.** Write Command

| Read/Write | DR Data |
|---|---|
| Write | (Number of Words to Write) << 16 \| (WP or WPL or EWP or EWPL) |
| Write | Address |
| Write | Memory [address] |

**Table 20-19.** Write Command (Continued)

| Read/Write | DR Data |
|---|---|
| Write | Memory [address+4] |
| Write | Memory [address+8] |
| Write | Memory [address+(Number of Words to Write - 1)* 4] |

Flash **Write Page and Lock** command **(WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

Flash **Erase Page and Write** command **(EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

Flash **Erase Page and Write the Lock** command **(EWPL)** combines EWP and WPL commands.

#### *20.3.4.3 Flash Full Erase Command*

This command is used to erase the Flash memory planes.

All lock bits must be deactivated before using the **Full Erase** command. This can be done by using the CLB command.

**Table 20-20.** Full Erase Command

| Read/Write | DR Data |
|---|---|
| Write | EA |

#### *20.3.4.4 Flash Lock Commands*

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated at the same time. Bit 0 of Bit Mask corresponds to the first lock bit and so on.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits. All the lock bits can also be cleared by the EA command.

**Table 20-21.** Set and Clear Lock Bit Command

| Read/Write | DR Data |
|---|---|
| Write | SLB or CLB |
| Write | Bit Mask |

Lock bits can be read using **Get Lock Bit** command **(GLB)**. When a bit set in the Bit Mask is returned, then the corresponding lock bit is active.

**Table 20-22.** Get Lock Bit Command

| Read/Write | DR Data |
|---|---|
| Write | GLB |
| Read | Bit Mask |

### 20.3.4.5 Flash General-purpose NVM Commands

General-purpose NVM bits (GP NVM) can be set with the **Set GPNVM** command **(SGPB)**. Using this command, several GP NVM bits can be activated at the same time. Bit 0 of Bit Mask corresponds to the first GPNVM bit and so on.

In the same way, the **Clear GPNVM** command **(CGPB)** is used to clear GP NVM bits. All the general-purpose NVM bits are also cleared by the EA command.

**Table 20-23.** Set and Clear General-purpose NVM Bit Command

| Read/Write | DR Data |
|---|---|
| Write | SGPB or CGPB |
| Write | Bit Mask |

GP NVM bits can be read using **Get GPNVM Bit** command **(GGPB)**. When a bit set in the Bit Mask is returned, then the corresponding GPNVM bit is set.

**Table 20-24.** Get General-purpose NVM Bit Command

| Read/Write | DR Data |
|---|---|
| Write | GGPB |
| Read | Bit Mask |

### 20.3.4.6 Flash Security Bit Command

Security bits can be set using **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. Only an event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

**Table 20-25.** Set Security Bit Command

| Read/Write | DR Data |
|---|---|
| Write | SSE |

Once the security bit is set, it is not possible to access FFPI. The only way to erase the security bit is to erase the Flash.

In order to erase the Flash, the user must perform the following:

- Power-off the chip
- Power-on the chip with TST = 0 and FWUP=0
- Assert Erase during a period of more than 220 ms
- Power-off the chip

Then it is possible to return to FFPI mode and check that Flash is erased.

### 20.3.4.7 Memory Write Command

This command is used to perform a write access to any memory location.

The **Memory Write** command **(WRAM)** is optimized for consecutive writes. An internal address buffer is automatically increased.

**Table 20-26.** Write Command

| Read/Write | DR Data |
|---|---|
| Write | (Number of Words to Write) << 16 \| (WRAM) |
| Write | Address |
| Write | Memory [address] |
| Write | Memory [address+4] |
| Write | Memory [address+8] |
| Write | Memory [address+(Number of Words to Write - 1)* 4] |

#### *20.3.4.8 Get Version Command*

The **Get Version** (GVE) command retrieves the version of the FFPI interface.

**Table 20-27.** Get Version Command

| Read/Write | DR Data |
|---|---|
| Write | GVE |
| Read | Version |

# 21. AT91SAM Boot Program

## 21.1 Overview

The Boot Program integrates different programs permitting download and/or upload into the different memories of the product.

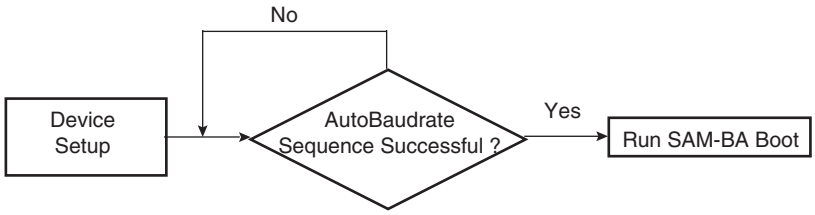
SAM-BA® Boot is executed at power-up only, if the device exits OFF mode and if the GPNVM bit 1 is set to 0.

Once running, SAM-BA™ Boot first initializes the Debug Unit serial port (DBGU) and the PLL frequency, then it waits for transactions on the DBGU serial port.

## 21.2 Flow Diagram

The Boot Program implements the algorithm in Figure 21-1.

**Figure 21-1.** Boot Program Algorithm Flow Diagram




## 21.3 Device Initialization

Initialization follows the steps described below:

1. FIQ initialization
1. Stack setup for ARM supervisor mode
2. Setup the Embedded Flash Controller
3. PLL setup: PLL is initialized to generate a 30 MHz typical frequency
4. Switch Master Clock on PLL Clock divided by 2 (15 MHz MCK frequency)
5. Copy code into SRAM
6. C variable initialization
7. Disable of the Watchdog and enable of the user reset
8. Jump to SAM-BA Boot sequence (see “SAM-BA Boot” )

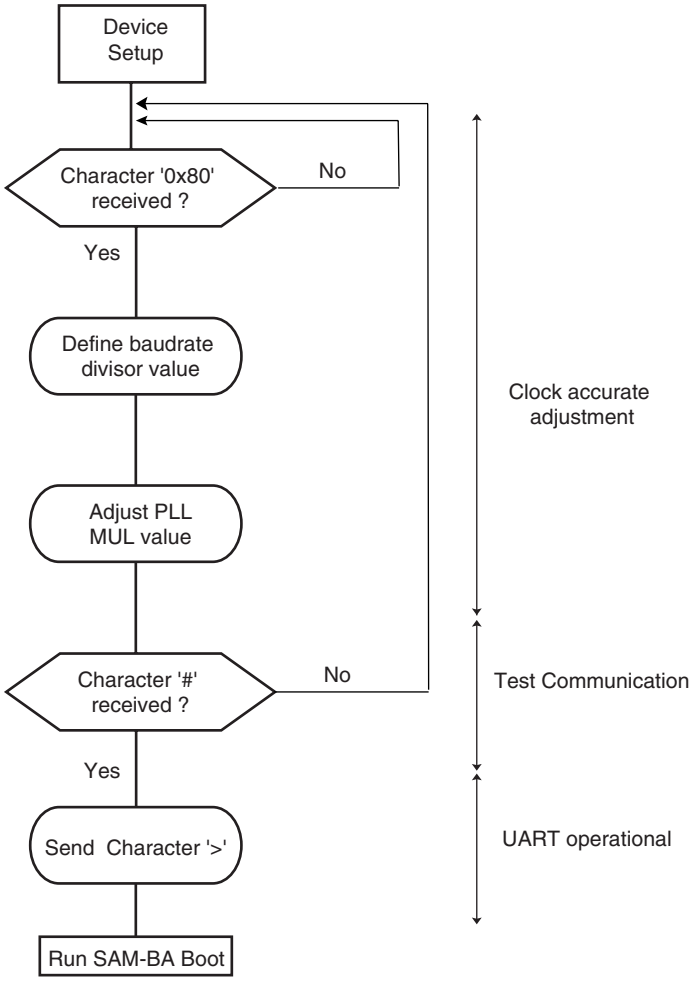
## 21.4 SAM-BA Boot

The SAM-BA boot principle is to:

– Check if the AutoBaudrate sequence has succeeded (see Figure 21-2)

– Check if characters have been received on the DBGU

**Figure 21-2.** AutoBaudrate Flow Diagram



– Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as given in Table 21-1.

**Table 21-1.** Commands Available through the SAM-BA Boot

| Command | Action | Argument(s) | Example |
|---|---|---|---|
| **O** | write a byte | Address, Value# | **O**200001,CA# |
| **o** | read a byte | Address,# | **o**200001,# |
| **H** | write a half word | Address, Value# | **H**200002,CAFE# |
| **h** | read a half word | Address,# | **h**200002,# |
| **W** | write a word | Address, Value# | **W**200000,CAFEDECA# |
| **w** | read a word | Address,# | **w**200000,# |
| **S** | send a file | Address,# | **S**200000,# |
| **R** | receive a file | Address, NbOfBytes# | **R**200000,1234# |
| **G** | go | Address# | **G**200200# |
| **V** | display version | No argument | **V**# |

- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  - *Address*: Address in hexadecimal.
  - *Value*: Byte, halfword or word to write in hexadecimal.
  - *Output*: '>'.
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  - *Address*: Address in hexadecimal
  - *Output*: The byte, halfword or word read in hexadecimal following by '>'
- Send a file (**S**): Send a file to a specified address
  - *Address*: Address in hexadecimal
  - *Output*: '>'.

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal
  - *NbOfBytes*: Number of bytes in hexadecimal to receive
  - *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code
  - *Address*: Address to jump in hexadecimal
  - *Output*: '>'
- Get Version (**V**): Return the SAM-BA boot version
  - *Output*: '>'

### 21.4.1 DBGU Serial Port

Communication is performed through the DBGU serial port initialized to 115200 Baud, 8, n, 1.

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the

binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory to work.

### 21.4.2 Xmodem Protocol

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC-16 to guarantee detection of a maximum bit error.

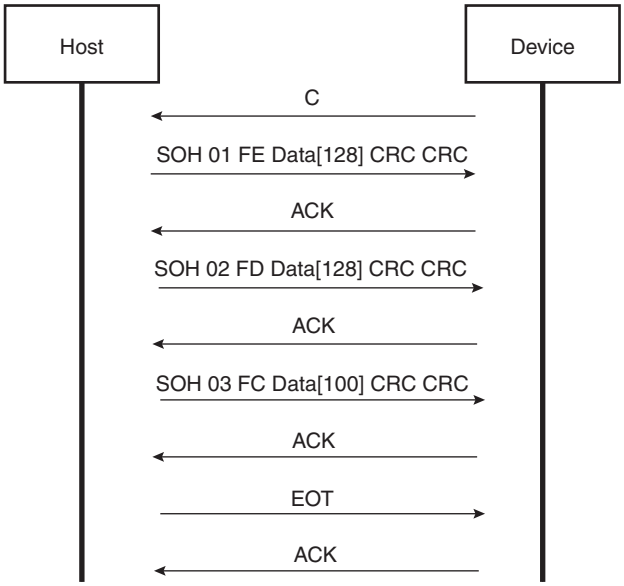
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1's complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 21-3 shows a transmission using this protocol.

**Figure 21-3.** Xmodem Transfer Example



## 21.5 In-Application Programming (IAP) Feature

The IAP feature is a function located in ROM that can be called by any software application.

When called, this function sends the desired FLASH command to the EEFC and waits for the FLASH to be ready (looping while the FRDY bit is not set in the MC_FSR register).

Since this function is executed from ROM, this allows FLASH programming (like sector write) to be done by code running in FLASH.

The IAP function entry point is retrieved by reading the SWI vector in ROM (0x400008).

This funtion takes one argument in parameter: the command to be sent to the EEFC.

This function returns the value of the MC_FSR register.

IAP software code example:

```
(unsigned int) (*IAP_Function)(unsigned long);
void main (void)

{
     unsigned long FlashSectorNum = 200;
     unsigned long flash_cmd = 0;
     unsigned long flash_status = 0;

/* Initialize the function pointer (retrieve function address from SWI
vector) */

     IAP_Function = ((unsigned long) (*)(unsigned long)) 0x400008;

/* Send your data to the sector */

/* build the command to send to EFC */

     flash_cmd =  (0x5A << 24) | (FlashSectorNum << 8) | AT91C_MC_FCMD_EWP;

/* Call the IAP function with appropriate command */

     flash_status = IAP_Function (flash_cmd);

}
```

## 21.6 Hardware and Software Constraints

**Table 21-2.** Pins Driven during Boot Program Execution

| Peripheral | Pin | PIO Line |
|---|---|---|
| DBGU | DRXD | PC16 |
| DBGU | DTXD | PC17 |

Using a 32.768 KHz crystal is not mandatory since SAM-BA boot will automatically use the internal 32Khz RC oscillator. PLL MUL parameter is automatically adapted to provide 115200 baudrate on the DBGU serial port.

# 22. Peripheral DMA Controller (PDC)

## 22.1 Overview

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals such as the UART, USART, SSC, SPI, MCI and the on- and off-chip memories. Using the Peripheral DMA Controller avoids processor intervention and removes the processor interrupt-handling overhead. This significantly reduces the number of clock cycles required for a data transfer and, as a result, improves the performance of the microcontroller and makes it more power efficient.

The PDC channels are implemented in pairs, each pair being dedicated to a particular peripheral. One channel in the pair is dedicated to the receiving channel and one to the transmitting channel of each UART, USART, SSC and SPI.

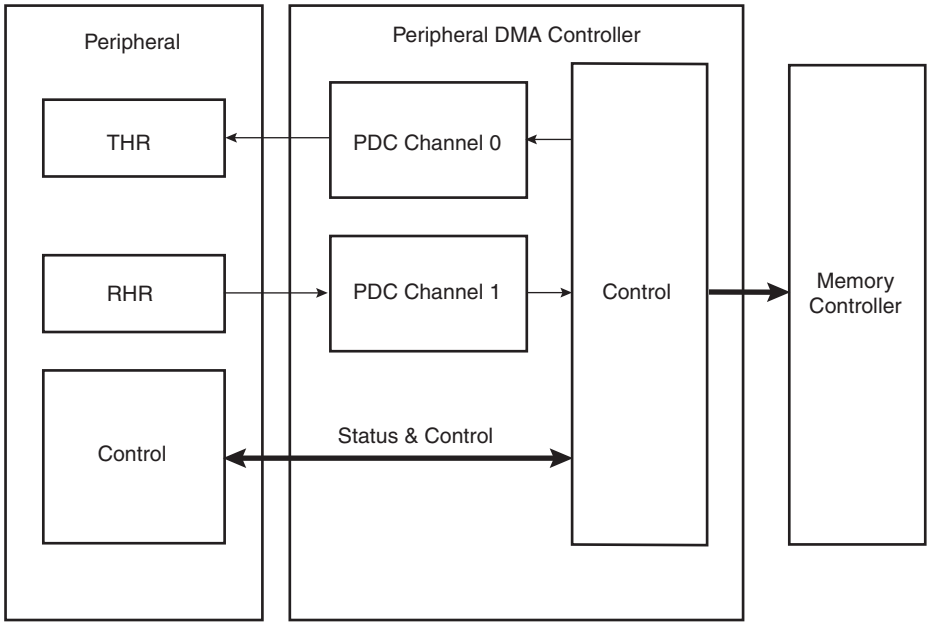
The user interface of a PDC channel is integrated in the memory space of each peripheral. It contains:

- two 32-bit memory pointer registers (send and receive)
- two 16-bit transfer count registers (send and receive)
- two 32-bit registers for next memory pointer (send and receive)
- two 16-bit registesr for next transfer count (send and receive)

The peripheral triggers PDC transfers using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the corresponding peripheral.

## 22.2 Block Diagram

**Figure 22-1.** Block Diagram



## 22.3 Functional Description

### 22.3.1 Configuration

The PDC channels user interface enables the user to configure and control the data transfers for each channel. The user interface of a PDC channel is integrated into the user interface of the peripheral (offset 0x100), which it is related to.

Per peripheral, it contains four 32-bit Pointer Registers (RPR, RNPR, TPR, and TNPR) and four 16-bit Counter Registers (RCR, RNCR, TCR, and TNCR).

The size of the buffer (number of transfers) is configured in an internal 16-bit transfer counter register, and it is possible, at any moment, to read the number of transfers left for each channel.

The memory base address is configured in a 32-bit memory pointer by defining the location of the first address to access in the memory. It is possible, at any moment, to read the location in memory of the next transfer and the number of remaining transfers. The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in PDC Transfer Control Register. These control bits enable reading the pointer and counter registers safely without any risk of their changing between both reads.

The PDC sends status flags to the peripheral visible in its status-register (ENDRX, ENDTX, RXBUFF, and TXBUFE).

ENDRX flag is set when the PERIPH_RCR register reaches zero.

RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.

ENDTX flag is set when the PERIPH_TCR register reaches zero.

TXBUFE flag is set when both PERIPH_TCR and PERIPH_TNCR reach zero.

These status flags are described in the peripheral status register.

### 22.3.2 Memory Pointers

Each peripheral is connected to the PDC by a receiver data channel and a transmitter data channel. Each channel has an internal 32-bit memory pointer. Each memory pointer points to a location anywhere in the memory space (on-chip memory or external bus interface memory).

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented by 1, 2 or 4, respectively for peripheral transfers.

If a memory pointer is reprogrammed while the PDC is in operation, the transfer address is changed, and the PDC performs transfers using the new address.

### 22.3.3 Transfer Counters

There is one internal 16-bit transfer counter for each channel used to count the size of the block already transferred by its associated channel. These counters are decremented after each data transfer. When the counter reaches zero, the transfer is complete and the PDC stops transferring data.

If the Next Counter Register is equal to zero, the PDC disables the trigger while activating the related peripheral end flag.

If the counter is reprogrammed while the PDC is operating, the number of transfers is updated and the PDC counts transfers from the new value.

Programming the Next Counter/Pointer registers chains the buffers. The counters are decremented after each data transfer as stated above, but when the transfer counter reaches zero, the values of the Next Counter/Pointer are loaded into the Counter/Pointer registers in order to re-enable the triggers.

For each channel, two status bits indicate the end of the current buffer (ENDRX, ENDTX) and the end of both current and next buffer (RXBUFF, TXBUFE). These bits are directly mapped to the peripheral status register and can trigger an interrupt request to the AIC.

The peripheral end flag is automatically cleared when one of the counter-registers (Counter or Next Counter Register) is written.

Note: When the Next Counter Register is loaded into the Counter Register, it is set to zero.

### 22.3.4 Data Transfers

The peripheral triggers PDC transfers using transmit (TXRDY) and receive (RXRDY) signals.

When the peripheral receives an external character, it sends a Receive Ready signal to the PDC which then requests access to the system bus. When access is granted, the PDC starts a read of the peripheral Receive Holding Register (RHR) and then triggers a write in the memory.

After each transfer, the relevant PDC memory pointer is incremented and the number of transfers left is decremented. When the memory block size is reached, a signal is sent to the peripheral and the transfer stops.

The same procedure is followed, in reverse, for transmit transfers.

### 22.3.5 Priority of PDC Transfer Requests

The Peripheral DMA Controller handles transfer requests from the channel according to priorities fixed for each product.These priorities are defined in the product datasheet.

If simultaneous requests of the same type (receiver or transmitter) occur on identical peripherals, the priority is determined by the numbering of the peripherals.

If transfer requests are not simultaneous, they are treated in the order they occurred. Requests from the receivers are handled first and then followed by transmitter requests.

## 22.4 Peripheral DMA Controller (PDC) User Interface

**Table 22-1.** Register Mapping

| Offset | Register | Register Name | Access | Reset |
|---|---|---|---|---|
| 0x100 | Receive Pointer Register | PERIPH[(1)]_RPR | Read-write | 0x0 |
| 0x104 | Receive Counter Register | PERIPH_RCR | Read-write | 0x0 |
| 0x108 | Transmit Pointer Register | PERIPH_TPR | Read-write | 0x0 |
| 0x10C | Transmit Counter Register | PERIPH_TCR | Read-write | 0x0 |
| 0x110 | Receive Next Pointer Register | PERIPH_RNPR | Read-write | 0x0 |
| 0x114 | Receive Next Counter Register | PERIPH_RNCR | Read-write | 0x0 |
| 0x118 | Transmit Next Pointer Register | PERIPH_TNPR | Read-write | 0x0 |
| 0x11C | Transmit Next Counter Register | PERIPH_TNCR | Read-write | 0x0 |
| 0x120 | PDC Transfer Control Register | PERIPH_PTCR | Write-only | - |
| 0x124 | PDC Transfer Status Register | PERIPH_PTSR | Read-only | 0x0 |

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI etc).

### 22.4.1 PDC Receive Pointer Register

**Register Name:** PERIPH_RPR

**Access Type:** Read-write

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

- **RXPTR: Receive Pointer Address**

Address of the next receive transfer.

### 22.4.2 PDC Receive Counter Register

**Register Name:** PERIPH_RCR

**Access Type:** Read-write

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

- **RXCTR: Receive Counter Value**

Number of receive transfers to be performed.

### 22.4.3 PDC Transmit Pointer Register

**Register Name:** PERIPH_TPR

**Access Type:** Read-write

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

- **TXPTR: Transmit Pointer Address**

Address of the transmit buffer.

### 22.4.4 PDC Transmit Counter Register

**Register Name:** PERIPH_TCR

**Access Type:** Read-write

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

- **TXCTR: Transmit Counter Value**

TXCTR is the size of the transmit transfer to be performed. At zero, the peripheral data transfer is stopped.

### 22.4.5 PDC Receive Next Pointer Register

**Register Name:** PERIPH_RNPR

**Access Type:** Read-write

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

- **RXNPTR: Receive Next Pointer Address**

RXNPTR is the address of the next buffer to fill with received data when the current buffer is full.

### 22.4.6 PDC Receive Next Counter Register

**Register Name:** PERIPH_RNCR

**Access Type:** Read-write

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

- **RXNCR: Receive Next Counter Value**

RXNCR is the size of the next buffer to receive.

### 22.4.7 PDC Transmit Next Pointer Register

**Register Name:** PERIPH_TNPR

**Access Type:** Read-write

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

- **TXNPTR: Transmit Next Pointer Address**

TXNPTR is the address of the next buffer to transmit when the current buffer is empty.

### 22.4.8 PDC Transmit Next Counter Register

**Register Name:** PERIPH_TNCR

**Access Type:** Read-write

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

- **TXNCR: Transmit Next Counter Value**

TXNCR is the size of the next buffer to transmit.

### 22.4.9 PDC Transfer Control Register

**Register Name:** PERIPH_PTCR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | TXTDIS | TXTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | RXTDIS | RXTEN |

- **RXTEN: Receiver Transfer Enable**

0 = No effect.

1 = Enables the receiver PDC transfer requests if RXTDIS is not set.

- **RXTDIS: Receiver Transfer Disable**

0 = No effect.

1 = Disables the receiver PDC transfer requests.

- **TXTEN: Transmitter Transfer Enable**

0 = No effect.

1 = Enables the transmitter PDC transfer requests.

- **TXTDIS: Transmitter Transfer Disable**

0 = No effect.

1 = Disables the transmitter PDC transfer requests

### 22.4.10 PDC Transfer Status Register

**Register Name:** PERIPH_PTSR

**Access Type:** Read-only

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

- **RXTEN: Receiver Transfer Enable**

0 = Receiver PDC transfer requests are disabled.

1 = Receiver PDC transfer requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0 = Transmitter PDC transfer requests are disabled.

1 = Transmitter PDC transfer requests are enabled.

# 23. Advanced Interrupt Controller (AIC)

## 23.1 Overview

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

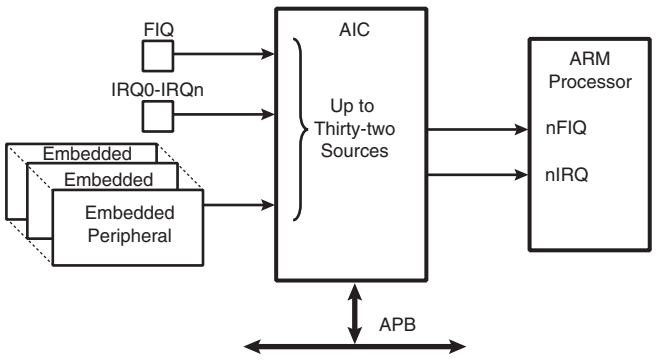
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

Internal interrupt sources can be programmed to be level sensitive or edge triggered. External interrupt sources can be programmed to be positive-edge or negative-edge triggered or high-level or low-level sensitive.

The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

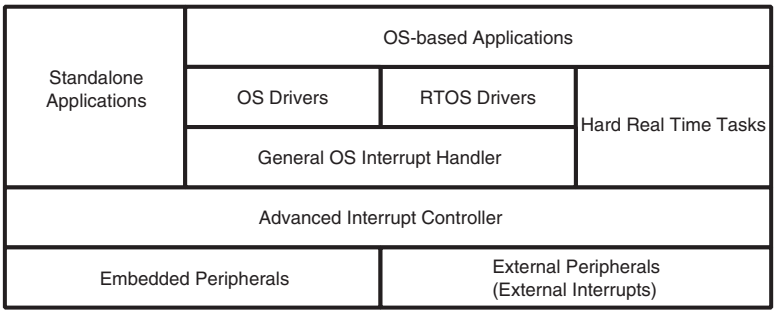
## 23.2 Block Diagram

**Figure 23-1.** Block Diagram



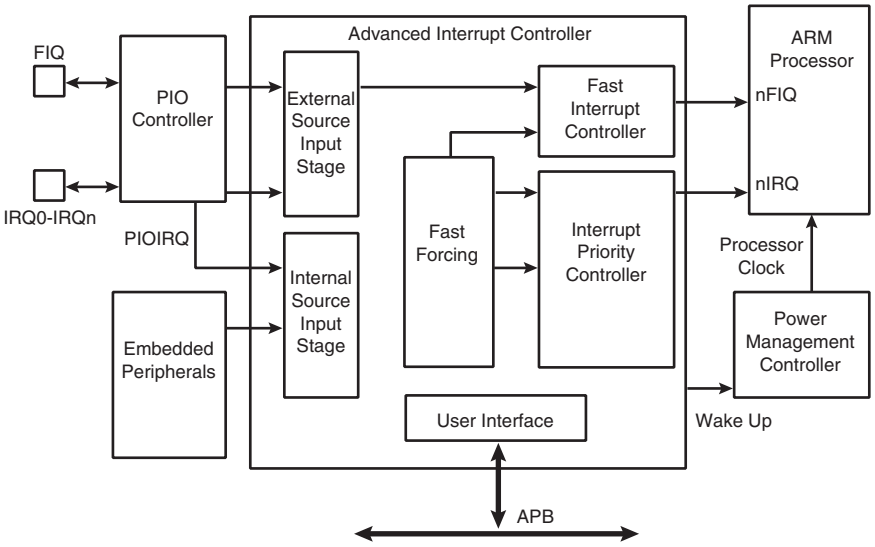

## 23.3 Application Block Diagram

**Figure 23-2.** Description of the Application Block




## 23.4 AIC Detailed Block Diagram

**Figure 23-3.** AIC Detailed Block Diagram



## 23.5 I/O Line Description

**Table 23-1.** I/O Line Description

| Pin Name | Pin Description | Type |
|---|---|---|
| FIQ | Fast Interrupt | Input |
| IRQ0 - IRQn | Interrupt 0 - Interrupt n | Input |

## 23.6 Product Dependencies

### 23.6.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

### 23.6.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

### 23.6.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines, such as the System Timer, the Real Time Clock, the Power Management Controller and the Memory Controller. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

## 23.7 Functional Description

### 23.7.1 Interrupt Source Control

#### *23.7.1.1 Interrupt Source Mode*

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the corresponding AIC_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

#### *23.7.1.2 Interrupt Source Enabling*

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

#### *23.7.1.3 Interrupt Clearing and Setting*

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC_ISCR and AIC_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation. (See “Priority Controller” on page 195.) The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, See “Fast Forcing” on page 199.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

#### *23.7.1.4 Interrupt Status*

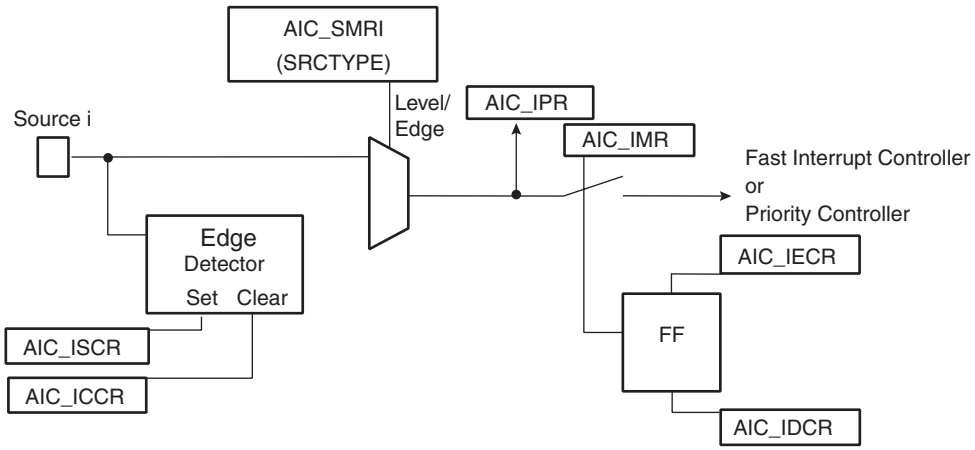
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see “Priority Controller” on page 195) and the register AIC_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

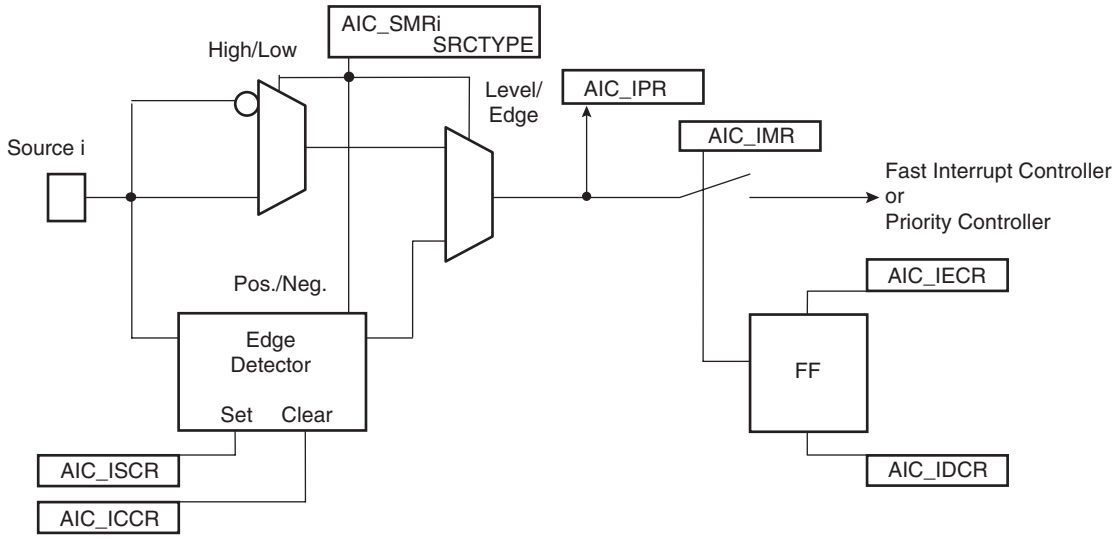
### 23.7.1.5 Internal Interrupt Source Input Stage

**Figure 23-4.** Internal Interrupt Source Input Stage




### 23.7.1.6 External Interrupt Source Input Stage

**Figure 23-5.** External Interrupt Source Input Stage



### 23.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

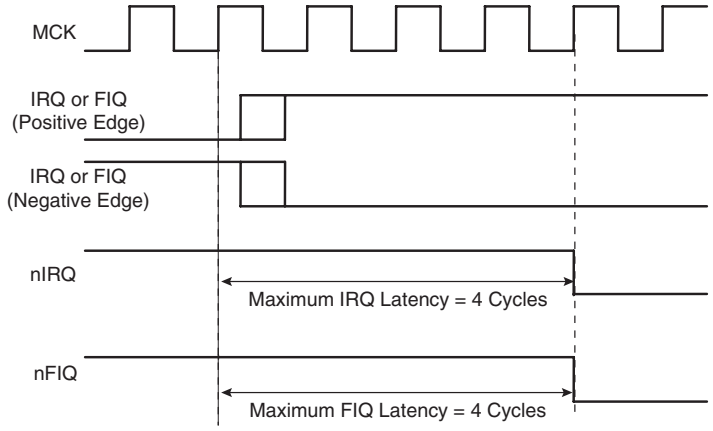
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

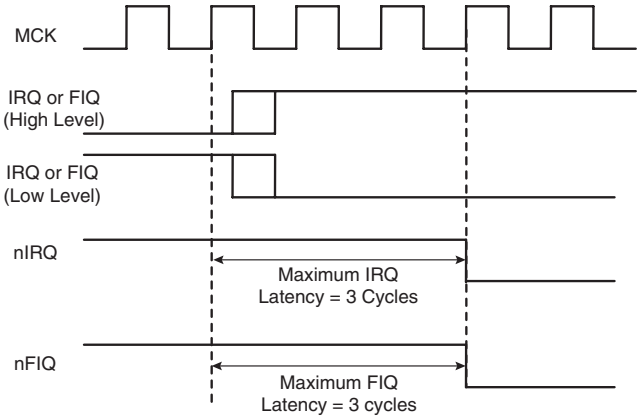
#### *23.7.2.1 External Interrupt Edge Triggered Source*

**Figure 23-6.** External Interrupt Edge Triggered Source



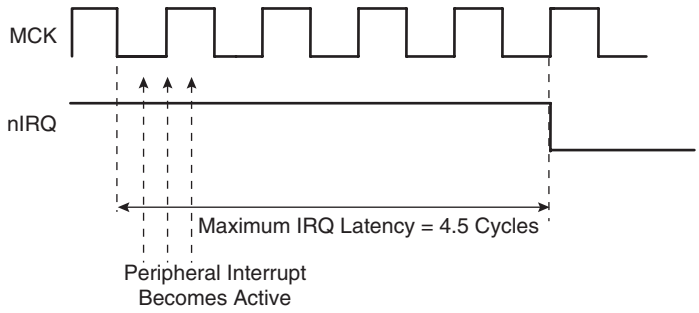

#### *23.7.2.2 External Interrupt Level Sensitive Source*

**Figure 23-7.** External Interrupt Level Sensitive Source



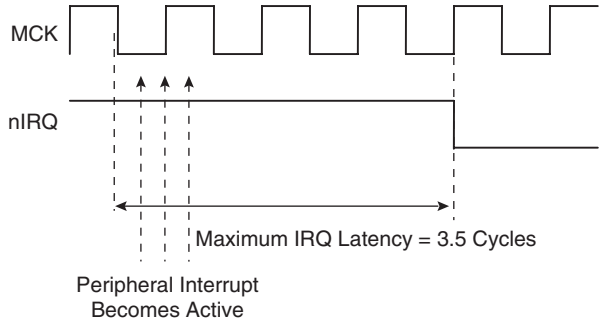
#### 23.7.2.3 *Internal Interrupt Edge Triggered Source*

**Figure 23-8.** Internal Interrupt Edge Triggered Source




#### 23.7.2.4 *Internal Interrupt Level Sensitive Source*

**Figure 23-9.** Internal Interrupt Level Sensitive Source




### 23.7.3 Normal Interrupt

#### 23.7.3.1 *Priority Controller*

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SMR (Source Mode Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC_IVR (Interrupt Vector Register) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC_EOICR (End of Interrupt Command Register). **The write of AIC_EOICR is the exit point of the interrupt handling**.

#### *23.7.3.2 Interrupt Nesting*

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

#### *23.7.3.3 Interrupt Vectoring*

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

```
LDR   PC,[PC,# -&F20]
```

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

#### *23.7.3.4 Interrupt Handlers*

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

```
LDR PC, [PC, # -&F20]
```

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level and the current interrupt number on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the "I" bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The "I" bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the "I" bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

### 23.7.4 Fast Interrupt

#### *23.7.4.1 Fast Interrupt Source*

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

#### *23.7.4.2 Fast Interrupt Control*

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

#### *23.7.4.3 Fast Interrupt Vectoring*

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

```
LDR   PC,[PC,# -&F20]
```

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

#### *23.7.4.4 Fast Interrupt Handlers*

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

```
LDR PC, [PC, # -&F20]
```

3. The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit “F” of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In

the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by four.

2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the program counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt, if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
4. The previous step enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14_fiq and SPSR_fiq if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the interrupt source 0.
6. Finally, the Link Register R14_fiq is restored into the PC after decrementing it by four (with instruction `SUB PC, LR, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The “F” bit in SPSR is significant. If it is set, it indicates that the ARM core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

#### *23.7.4.5 Fast Forcing*

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC_IPR).

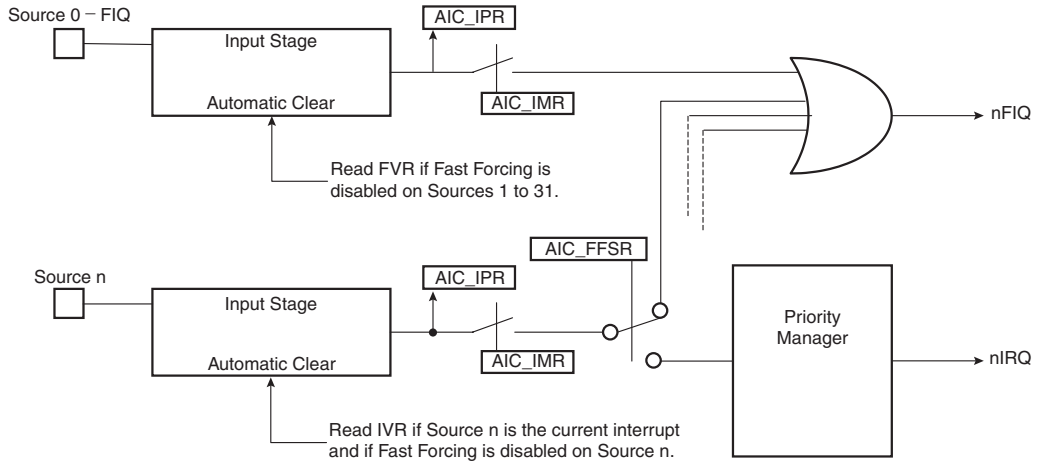
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt Clear Command Register (AIC_ICCR).

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

**Figure 23-10.** Fast Forcing




### 23.7.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

- If an enabled interrupt with a higher priority than the current one is pending, it is stacked.
- If there is no enabled pending interrupt, the spurious vector is returned.

In either case, an End of Interrupt command is necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system as the debug system would become strongly intrusive and cause the application to enter an undesired state.

This is avoided by using the Protect Mode. Writing DBGM in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

### 23.7.6 Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

### 23.7.7 General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

## 23.8 Advanced Interrupt Controller (AIC) User Interface

### 23.8.1 Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor support only a ± 4-Kbyte offset.

**Table 23-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Source Mode Register 0 | AIC_SMR0 | Read-write | 0x0 |
| 0x04 | Source Mode Register 1 | AIC_SMR1 | Read-write | 0x0 |
| --- | --- | --- | --- | --- |
| 0x7C | Source Mode Register 31 | AIC_SMR31 | Read-write | 0x0 |
| 0x80 | Source Vector Register 0 | AIC_SVR0 | Read-write | 0x0 |
| 0x84 | Source Vector Register 1 | AIC_SVR1 | Read-write | 0x0 |
| --- | --- | --- | --- | --- |
| 0xFC | Source Vector Register 31 | AIC_SVR31 | Read-write | 0x0 |
| 0x100 | Interrupt Vector Register | AIC_IVR | Read-only | 0x0 |
| 0x104 | FIQ Interrupt Vector Register | AIC_FVR | Read-only | 0x0 |
| 0x108 | Interrupt Status Register | AIC_ISR | Read-only | 0x0 |
| 0x10C | Interrupt Pending Register[2] | AIC_IPR | Read-only | 0x0[1] |
| 0x110 | Interrupt Mask Register[2] | AIC_IMR | Read-only | 0x0 |
| 0x114 | Core Interrupt Status Register | AIC_CISR | Read-only | 0x0 |
| 0x118 - 0x11C | Reserved | --- | --- | --- |
| 0x120 | Interrupt Enable Command Register[2] | AIC_IECR | Write-only | --- |
| 0x124 | Interrupt Disable Command Register[2] | AIC_IDCR | Write-only | --- |
| 0x128 | Interrupt Clear Command Register[2] | AIC_ICCR | Write-only | --- |
| 0x12C | Interrupt Set Command Register[2] | AIC_ISCR | Write-only | --- |
| 0x130 | End of Interrupt Command Register | AIC_EOICR | Write-only | --- |
| 0x134 | Spurious Interrupt Vector Register | AIC_SPU | Read-write | 0x0 |
| 0x138 | Debug Control Register | AIC_DCR | Read-write | 0x0 |
| 0x13C | Reserved | --- | --- | --- |
| 0x140 | Fast Forcing Enable Register[2] | AIC_FFER | Write-only | --- |
| 0x144 | Fast Forcing Disable Register[2] | AIC_FFDR | Write-only | --- |
| 0x148 | Fast Forcing Status Register[2] | AIC_FFSR | Read-only | 0x0 |
| 0x14C - 0x1E0 | Reserved | --- | --- | --- |
| 0x1EC - 0x1FC | Reserved | | | |

Notes:
1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.
2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.
3. Values in the Version Register vary with the version of the IP block implementation.

### 23.8.2 AIC Source Mode Register

**Register Name:** AIC_SMR0..AIC_SMR31

**Access Type:** Read-write

**Reset Value:** 0x0

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

- **PRIOR: Priority Level**

Programs the priority level for all sources except FIQ source (source 0).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ in the related SMR register AIC_SMRx.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt sources.

| SRCTYPE | | Internal Interrupt Sources | External Interrupt Sources |
|---|---|---|---|
| 0 | 0 | High level Sensitive | Low level Sensitive |
| 0 | 1 | Positive edge triggered | Negative edge triggered |
| 1 | 0 | High level Sensitive | High level Sensitive |
| 1 | 1 | Positive edge triggered | Positive edge triggered |

### 23.8.3 AIC Source Vector Register

**Register Name:** AIC_SVR0..AIC_SVR31

**Access Type:** Read-write

Reset Value: 0x0

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

- **VECTOR: Source Vector**

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

### 23.8.4 AIC Interrupt Vector Register

**Register Name:** AIC_IVR

**Access Type:** Read-only

Reset Value: 0x0

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

- **IRQV: Interrupt Vector Register**

The Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

### 23.8.5 AIC FIQ Vector Register

**Register Name:** AIC_FVR

**Access Type:** Read-only

Reset Value: 0x0

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

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

### 23.8.6 AIC Interrupt Status Register

**Register Name:** AIC_ISR

**Access Type:** Read-only

Reset Value: 0x0

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

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

### 23.8.7 AIC Interrupt Pending Register

**Register Name:** AIC_IPR

**Access Type:** Read-only

Reset Value: 0x0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

### 23.8.8 AIC Interrupt Mask Register

**Register Name:** AIC_IMR

**Access Type:** Read-only

Reset Value: 0x0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

### 23.8.9 AIC Core Interrupt Status Register

**Register Name:** AIC_CISR

**Access Type:** Read-only

Reset Value: 0x0

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

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

### 23.8.10 AIC Interrupt Enable Command Register

**Register Name:** AIC_IECR

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

### 23.8.11 AIC Interrupt Disable Command Register

**Register Name:** AIC_IDCR

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

### 23.8.12 AIC Interrupt Clear Command Register

**Register Name:** AIC_ICCR

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

### 23.8.13 AIC Interrupt Set Command Register

**Register Name:** AIC_ISCR

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | FIQ |

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

### 23.8.14 AIC End of Interrupt Command Register

**Register Name:** AIC_EOICR

Access Type: Write-only

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

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

### 23.8.15 AIC Spurious Interrupt Vector Register

**Register Name:** AIC_SPU

Access Type: Read-write

Reset Value: 0x0

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

- **SIVR: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

### 23.8.16 AIC Debug Control Register

**Register Name:** AIC_DCR

Access Type: Read-write

Reset Value: 0x0

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

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

### 23.8.17 AIC Fast Forcing Enable Register

**Register Name:** AIC_FFER

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | – |

- **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

### 23.8.18 AIC Fast Forcing Disable Register

**Register Name:** AIC_FFDR

Access Type: Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | – |

- **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

### 23.8.19 AIC Fast Forcing Status Register

**Register Name:** AIC_FFSR

Access Type: Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | SYS | – |

- **SYS, PID2-PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enabled on the corresponding interrupt.

# 24. Clock Generator

## 24.1 Overview

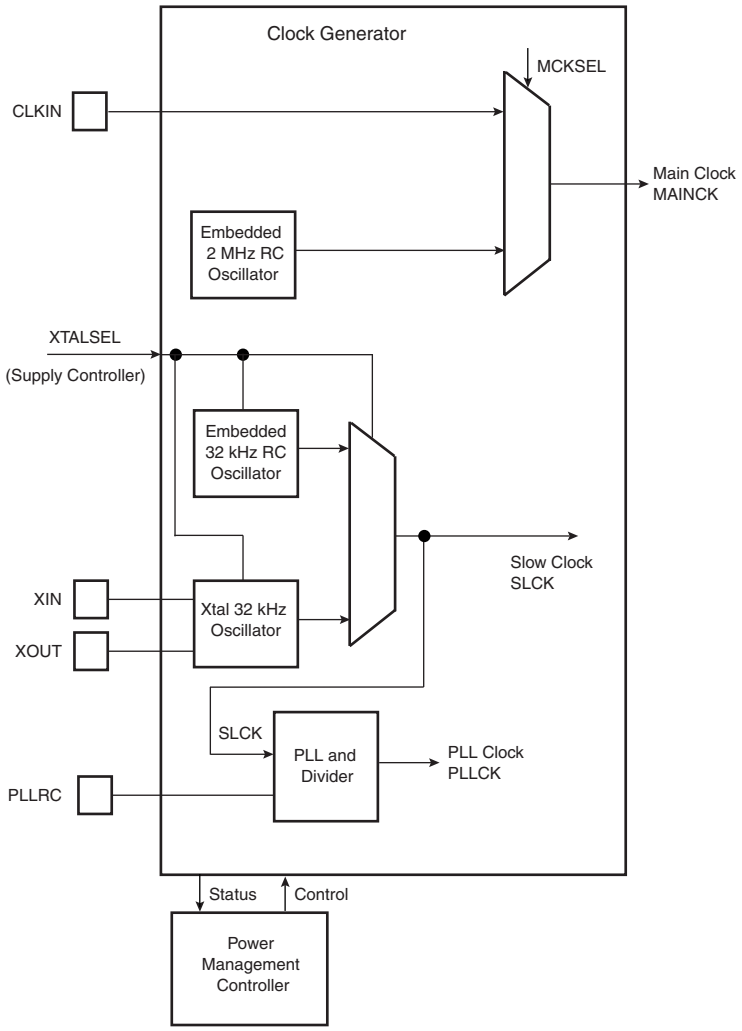
The Clock Generator is made up of one PLL, one fast RC oscillator, one slow RC oscillator and one 32,768 Hz Crystal Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, which is the only permanent clock within the system (except in OFF mode)
- MAINCK is the output of the Main Clock selection: either CLKIN (external clock) or 2 MHz Fast RC Oscillator
- PLLCK is the output of the Divider and PLL block

The Clock Generator User Interface is embedded within the Power Management Controller and is described in Section 25.9 ”Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

**Figure 24-1.** Clock Generator Block Diagram



## 24.2 Slow Clock

The Slow Clock is generated by the Slow Clock Crystal Oscillator or by the Slow Clock RC Oscillator.

The selection is made by writing the XTALSEL bit in the Supply Controller Control Register (SUPC_CR).

By default, the RC Oscillator is selected.

## 24.3 Slow Clock RC Oscillator

By default, the Slow Clock RC Oscillator is enabled and selected. The user has to take into account the possible drifts of the RC Oscillator. More details are given in the section “DC Characteristics” of the product datasheet.

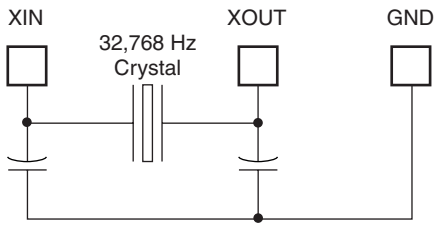
It can be disabled via the XTALSEL bit in the Supply Controller Control Register (SUPC_CR).

## 24.4 Slow Clock Crystal Oscillator

The Clock Generator integrates a 32,768 Hz low-power oscillator.The XIN and XOUT pins must be connected to a 32,768 Hz crystal. Two external capacitors must be wired as shown in Figure 24-2.More details are given in the section “DC Characteristics” of the product datasheet.

Note that the user is not obliged to use the Slow Clock Crystal and can use the RC Oscillator instead. In this case, XIN and XOUT can be left unconnected.

**Figure 24-2.** Typical Slow Clock Crystal Oscillator Connection



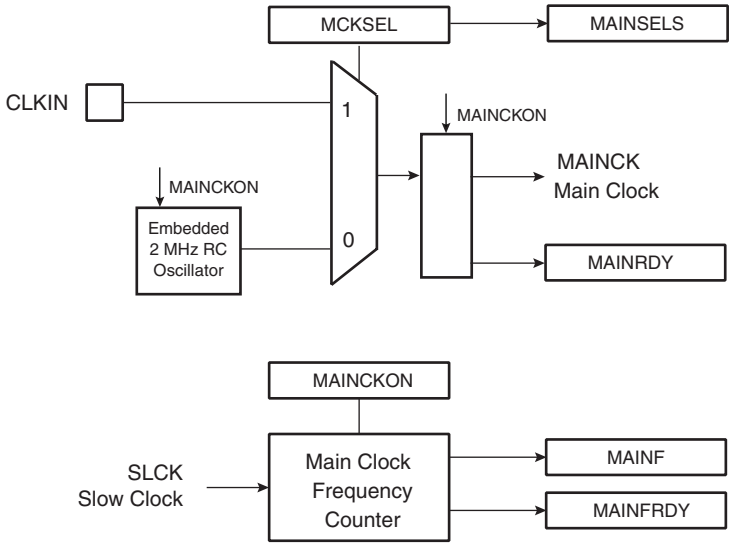

The user can set the Slow Clock Crystal Oscillator in bypass mode instead of connecting a crystal. In this case, the user has to provide the external clock signal on XIN. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section.

The programmer has to be sure to set the OSCBYPASS bit in the Supply Controller Mode Register (SUPC_MR) and XTALSEL bit in the Supply Controller Control Register (SUPC_CR).

## 24.5 Main Clock

Figure 24-3 shows the Main Clock block diagram.

**Figure 24-3.** Main Clock Block Diagram




The Main Clock has two sources:

- 2 MHz Fast RC Oscillator which starts very quickly and is used at startup
- an external clock (CLKIN)

### 24.5.1 2 MHz Fast RC Oscillator

After reset, the 2 MHz Fast RC Oscillator is enabled and selected as MAINCK. MAINCK is the default clock selected to start up the system.

Startup-up time specifications are provided in the “DC Characteristics” section of the product datasheet.

The software can disable or enable the 2 MHz Fast RC Oscillator with the MAINCKON bit in the Clock Generator Main Oscillator Register (CKGR_MOR).

When disabling the Main Clock by clearing the MAINCKON bit in CKGR_MOR, the MAINRDY bit in the Power Management Controller Status Register (PMC_SR) is automatically cleared, indicating the Main Clock is off.

Setting the MAINRDY bit in the Power Management Controller Interrupt Enable Register (PMC_IER) can trigger an interrupt to the processor.

It is recommended to disable the Main Clock as soon as the processor no longer uses it and runs out of SLCK or PLLCK.

Disabling the MAINCKON bit is also used to go into WAIT mode. The user sets the Main Clock as Master clock and disables the Main Clock by clearing the MAINCKON bit.

To wake up from WAIT mode, a fast startup must be done. See Section 25.6 ”The Fast Startup”.

### 24.5.2 Main Clock Frequency Counter

The device features a Main Clock frequency counter that provides the frequency of the Main Clock.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as MAINCKON is set to 1.Then, at the 16th falling edge of Slow Clock, the MAINFRDY bit in the Clock Generator Main Clock Frequency Register (CKGR_MCFR) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the 2 MHz Fast RC Oscillator or CLKIN input signal can be determined.

### 24.5.3 External Clock CLKIN

The user can input a clock on the device. In this case, the user has to provide the external clock signal on the CLKIN pin. The programmer has to be sure to set the MCKSEL bit in the Clock Generator Main Oscillator Register (CKGR_MOR) to 1 for the external clock to operate properly.

The user can check the MAINSELS bit in the Power Management Status Register (PMC_SR) to check that the selection has been completed.

Note that the user must be sure to put MCKSEL bit to 1 only when an external clock is applied on CLKIN. The user does not need to check MAINRDY bit when switching to CLKIN.

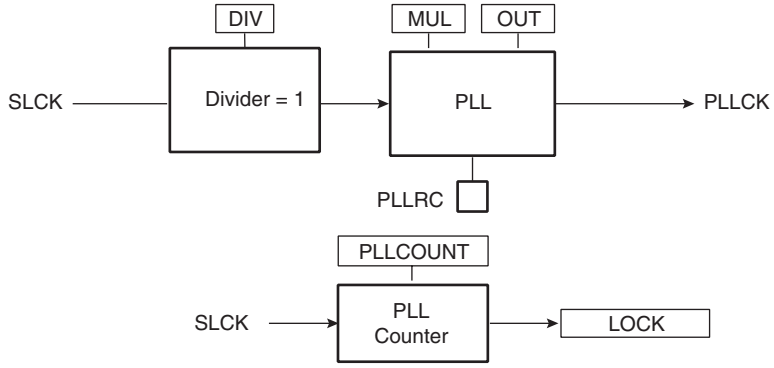
Input characteristics of the CLKIN pin are given in the Electrical Characteristics section.

## 24.6 Divider and PLL Block

The PLL embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLL minimum input frequency when programming the divider.

Figure 24-4 shows the block diagram of the divider and PLL block.

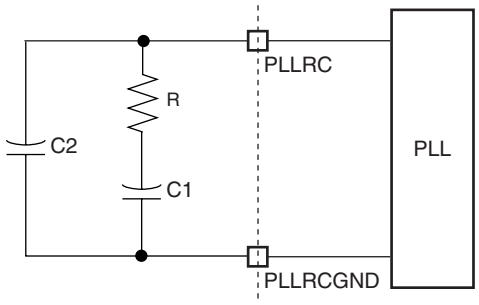
**Figure 24-4.** Divider and PLL Block Diagram




### 24.6.1 PLL Filter

The PLL requires connection to an external second-order filter through the PLLRC pin. Figure 24-5 shows a schematic of these filters.

**Figure 24-5.** PLL Capacitors and Resistors




Values of R, C1 and C2 to be connected to the PLLRC pin must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade off has to be found between output signal overshoot and startup time. See the product electrical PLL characteristics section.

Note that PLLRCGND must never be connected to GND.

### 24.6.2 Divider and Phase Lock Loop Programming

The divider can only be set at 1 when the PLL is activated. The PLL input is SLCK.

When the divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the MUL parameter. The factor applied to the source signal frequency is (MUL + 1). When MUL is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR_PLLR are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

Two PLL startup schemes are available:

- The fast startup scheme allows the PLL to reach at least 70% of its target frequency in less than 60 µs. In this mode the STDMODE field must be set to 0x0 and the PLLCOUNT field can be programed at 0x01 in the CKGR_PLLR register.
- The normal startup procedure of the PLL is performed when the STDMODE field of the CKGR_PLLR register is set to 0x02. In this startup scheme, the PLLCOUNT field must be set with the relevant value function of the programed PLL frequency.

Note that the STMODE field of the CKGR_PLLR register must be set to 0x02 when the PLL is shutdown.

# 25. Power Management Controller (PMC)

## 25.1 Overview

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the ARM Processor.

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), switched off when entering processor in idle mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SPI, TWI, TC, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

## 25.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

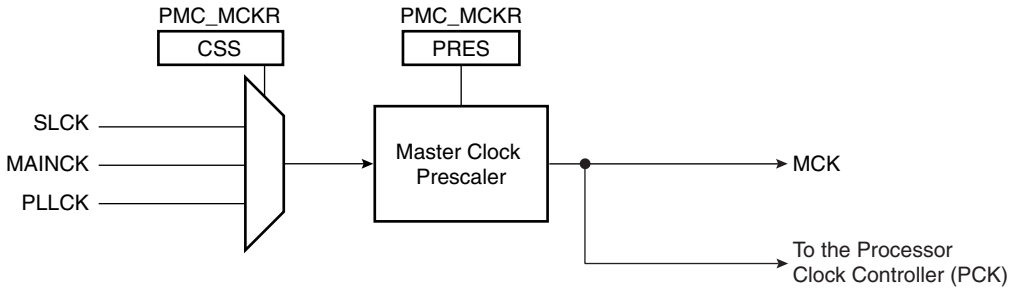
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLL.

The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one, to inform the software when the change is actually done.

**Figure 25-1.** Master Clock Controller



## 25.3 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the processor idle mode. The processor clock can be disabled by writing to 1 the PCK bit in the Power Management Controller System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purposes) can be read in the System Clock Status Register (PMC_SCSR).

The processor clock is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

## 25.4 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by means of the Peripheral Clock Controller. The user can individually enable and disable the Master Clock on the peripherals by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

The bit number within the Peripheral Clock Control registers (PMC_PCER, PMC_PCDR, and PMC_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

## 25.5 Programmable Clock Output Controller

The PMC controls 3 signals to be output on external pins, PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow Clock, the PLL output and the Main Clock by writing the CSS field in PMC_PCKx Register. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status bit in PMC_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

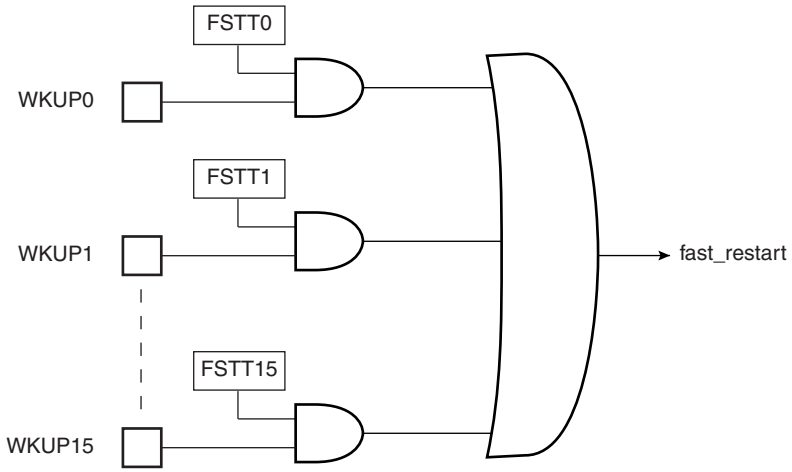
As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

## 25.6 The Fast Startup

The SAM7L device allows the processor to restart in less than six microseconds while the device is in Wait mode. A Fast Startup is enabled upon the detection of a low level on one of the 16 wake-up inputs.

The Fast Restart circuitry, as shown in Figure 25-2, is fully asynchronous and provides a fast startup signal to the Power Management Controller. As soon as the fast startup signal is asserted, this automatically restarts the embedded 2 MHz Fast RC oscillator, switches the Master Clock on the 2 MHz clock and re-enables the processor clock if it is disabled.

**Figure 25-2.** Fast Startup Circuitry




Each wake-up input pin can be enabled to generate a Fast Startup event by writing at 1 the corresponding bit in the Fast Startup Mode Register SUPC_FSMR. Only a low level on the enabled wake-up input pins generates a Fast Startup.

The user interface does not provide any status for Fast Startup, but the user can easily recover this information by reading the PIO Controller.

## 25.7 Programming Sequence

7. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main clock frequency. This measure can be accomplished via the CKGR_MCFR register.

   Once the MAINFRDY field is set in CKGR_MCFR register, the user may read the MAINF field in CKGR_MCFR register. This provides the number of main clock cycles within sixteen slow clock cycles.

8. Setting PLL and divider:

   All parameters needed to configure PLL and the divider are located in the CKGR_PLLR register.

   The DIV field is used to control the divider itself. It must be set to 1 when PLL is used. By default, DIV parameter is set to 0 which means that the divider is turned off.

   The MUL field is the PLL multiplier factor. This parameter can be programmed between 0 and 2047. If MUL is set to 0, PLL will be turned off, otherwise the PLL output frequency is PLL input frequency multiplied by (MUL + 1).

The PLLCOUNT field specifies the number of slow clock cycles before LOCK bit is set in the PMC_SR register after CKGR_PLLR register has been written.

Once the PMC_PLL register has been written, the user must wait for the LOCK bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCK has been enabled in the PMC_IER register. All parameters in CKGR_PLLR can be programmed in a single write operation. If at some stage one of the following parameters, MUL, DIV is modified, LOCK bit will go low to indicate that PLL is not ready yet. When PLL is locked, LOCK will be set again. The user is constrained to wait for LOCK bit to be set before using the PLL output clock.

Code Example:

```
write_register(CKGR_PLLR,0x3209A01)
```

If PLL and divider are enabled, the PLL input clock is the main clock. PLL output clock is PLL input clock multiplied by 801. Once CKGR_PLLR has been written, LOCK bit will be set after eight slow clock cycles.

9. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR register.

The CSS field is used to select the Master Clock divider source. By default, the selected clock source is main clock.

The PRES field is used to control the Master Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Master Clock output is prescaler input divided by PRES parameter. By default, PRES parameter is set to 1 which means that master clock is equal to main clock.

Once the PMC_MCKR register has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting for the interrupt line to be raised if the associated interrupt to MCKRDY has been enabled in the PMC_IER register.

The PMC_MCKR register must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR register is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.

If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR, the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK goes high and MCKRDY is set.
While PLL is unlocked, the Master Clock selection is automatically changed to Slow Clock. For further information, see Section 25.8.2 “Clock Switching Waveforms” on page 224.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

10. Selection of Programmable clocks

Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. 3 Programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

PMC_PCKx registers are used to configure Programmable clocks.

The CSS field is used to select the Programmable clock divider source. Three clock options are available: main clock, slow clock, PLLCK. By default, the clock source selected is slow clock.

The PRES field is used to control the Programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 0 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding Programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER register. All parameters in PMC_PCKx can be programmed in a single write operation.

If the CSS and PRES parameters are to be modified, the corresponding Programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the Programmable clock and wait for the PCKRDYx bit to be set.

Code Example:

```
write_register(PMC_PCK0,0x00000015)
```

Programmable clock 0 is main clock divided by 32.

11. Enabling Peripheral Clocks

Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

15 peripheral clocks can be enabled or disabled. The PMC_PCSR provides a clear view as to which peripheral clock is enabled.

Note: Each enabled peripheral clock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_PCER,0x00000110)
```

Peripheral clocks 4 and 8 are enabled.

```
write_register(PMC_PCDR,0x00000010)
```

Peripheral clock 4 is disabled.

## 25.8 Clock Switching Details

### 25.8.1 Master Clock Switching Timings

Table 25-1 gives the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

**Table 25-1.** Clock Switching Timings (Worst Case)

| From<br>To | Main Clock | SLCK | PLL Clock |
|---|---|---|---|
| Main Clock | – | 4 x SLCK +<br>2.5 x Main Clock | 3 x PLL Clock +<br>4 x SLCK +<br>1 x Main Clock |
| SLCK | 0.5 x Main Clock +<br>4.5 x SLCK | – | 3 x PLL Clock +<br>5 x SLCK |
| PLL Clock | 0.5 x Main Clock +<br>4 x SLCK +<br>PLLCOUNT x SLCK +<br>2.5 x PLLx Clock | 2.5 x PLL Clock +<br>5 x SLCK +<br>PLLCOUNT x SLCK | 2.5 x PLL Clock +<br>4 x SLCK +<br>PLLCOUNT x SLCK |

### 25.8.2 Clock Switching Waveforms

**Figure 25-3.** Switch Master Clock from Slow Clock to PLL Clock

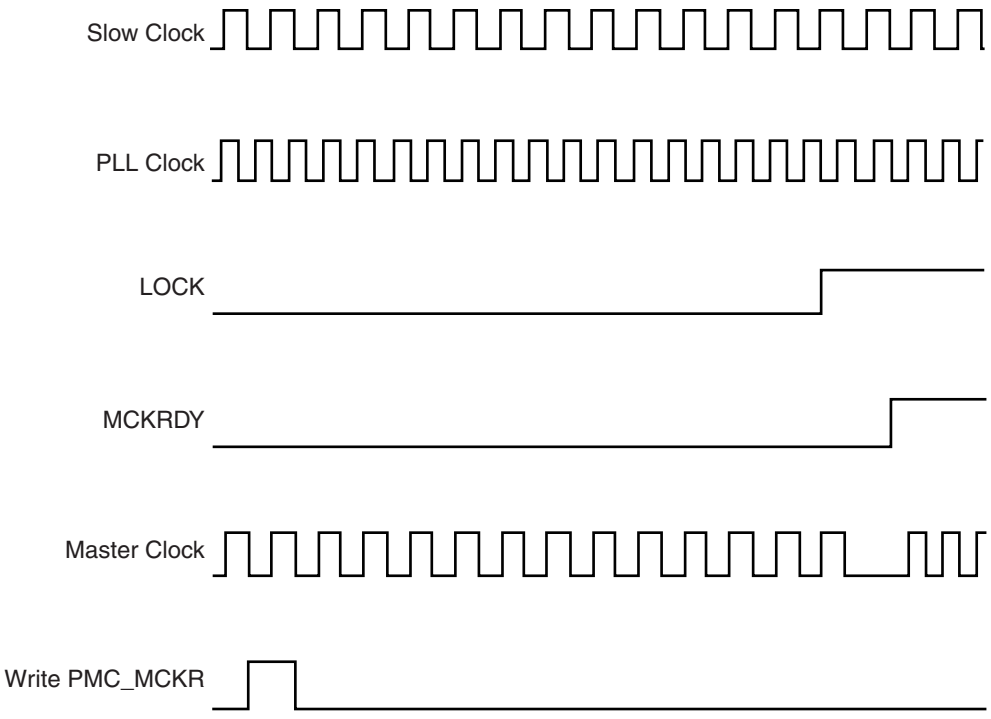


**Figure 25-4.** Switch Master Clock from Main Clock to Slow Clock

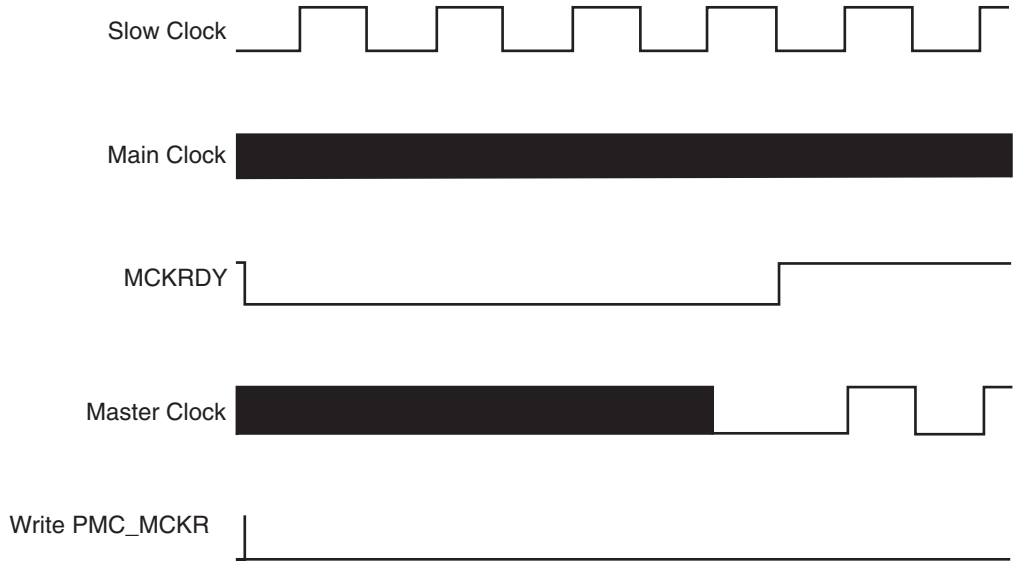



**Figure 25-5.** Change PLL Programming

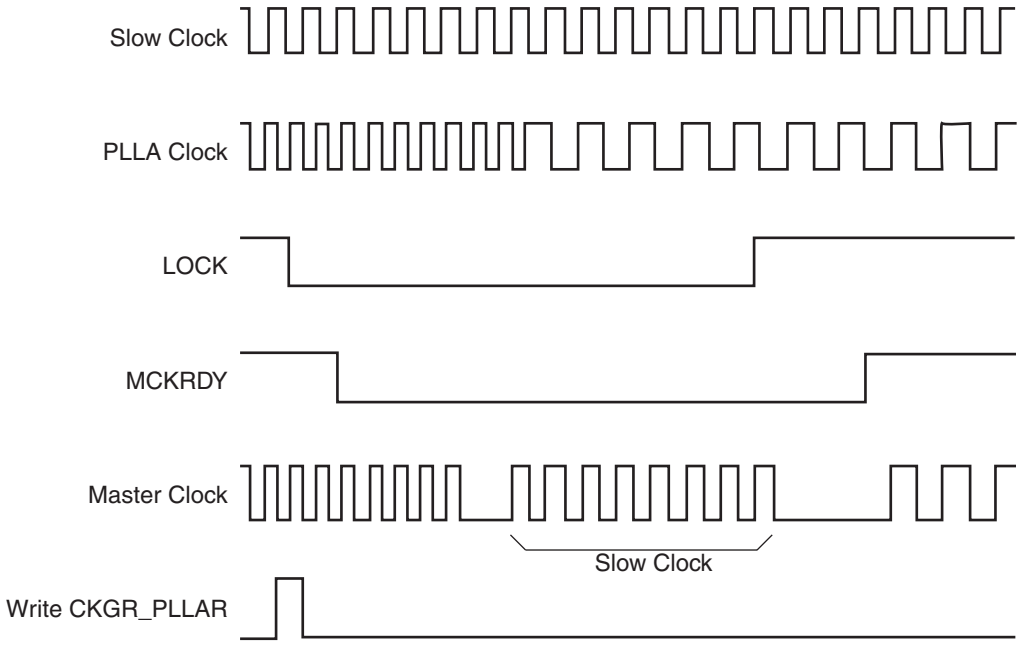
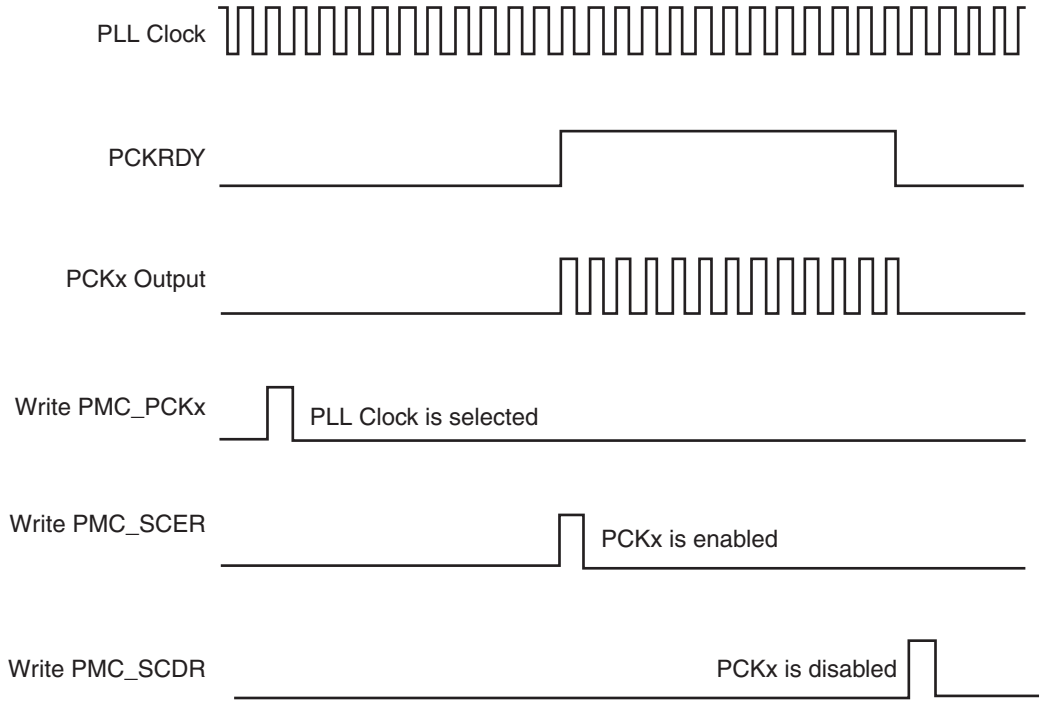


**Figure 25-6.** Programmable Clock Output Programming



## 25.9 Power Management Controller (PMC) User Interface

**Table 25-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | System Clock Enable Register | PMC_SCER | Write-only | – |
| 0x0004 | System Clock Disable Register | PMC_SCDR | Write-only | – |
| 0x0008 | System Clock Status Register | PMC _SCSR | Read-only | 0x0000_0001 |
| 0x000C | Reserved | – | – | – |
| 0x0010 | Peripheral Clock Enable Register | PMC _PCER | Write-only | N.A. |
| 0x0014 | Peripheral Clock Disable Register | PMC_PCDR | Write-only | – |
| 0x0018 | Peripheral Clock Status Register | PMC_PCSR | Read-only | 0x0000_0000 |
| 0x001C | Reserved | – | – | – |
| 0x0020 | Main Oscillator Register | CKGR_MOR | Read-write | 0x0000_0001 |
| 0x0024 | Main Clock Frequency Register | CKGR_MCFR | Read-only | 0x0000_0000 |
| 0x0028 | PLL Register | CKGR_PLLR | Read-write | 0x0000_3F00 |
| 0x0030 | Master Clock Register | PMC_MCKR | Read-write | 0x0000_0001 |
| 0x0038 | Reserved | – | – | – |
| 0x003C | Reserved | – | – | – |
| 0x0040 | Programmable Clock 0 Register | PMC_PCK0 | Read-write | 0x0000_0000 |
| 0x0044 | Programmable Clock 1 Register | PMC_PCK1 | Read-write | 0x0000_0000 |
| ... | ... | ... | ... | ... |
| 0x0060 | Interrupt Enable Register | PMC_IER | Write-only | – |
| 0x0064 | Interrupt Disable Register | PMC_IDR | Write-only | – |
| 0x0068 | Status Register | PMC_SR | Read-only | 0x0001_0008 |
| 0x006C | Interrupt Mask Register | PMC_IMR | Read-only | 0x0000_0000 |
| 0x0070 | Fast Startup Mode Register | PMC_FSMR | Read-write | 0x0000_0000 |
| 0x0074 - 0x007C | Reserved | – | Read-only | – |
| 0x0080- 0x00FC | Reserved | – | – | – |

### 25.9.1 PMC System Clock Enable Register

**Register Name:**PMC_SCER

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCK2 | PCK1 | PCK0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | | - |

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

### 25.9.2 PMC System Clock Disable Register

**Register Name:**PMC_SCDR

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCK2 | PCK1 | PCK0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | PCK |

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

### 25.9.3 PMC System Clock Status Register

**Register Name:**PMC_SCSR

**Access Type:**Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCK2 | PCK1 | PCK0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | PCK |

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

### 25.9.4 PMC Peripheral Clock Enable Register

**Register Name:**PMC_PCER

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | - | - |

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

### 25.9.5 PMC Peripheral Clock Disable Register

**Register Name:**PMC_PCDR

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | - | - |

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

### 25.9.6 PMC Peripheral Clock Status Register

**Register Name:**PMC_PCSR

**Access Type:**Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9 | PID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID7 | PID6 | PID5 | PID4 | PID3 | PID2 | – | – |

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

### 25.9.7 PMC Clock Generator Main Oscillator Register

**Register Name:**CKGR_MOR

**Access Type:**Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | MCKSEL |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | MAINCKON |

- **KEY: Password**

Should be written at value 0x37. Writing any other value in this field aborts the write operation.

- **MAINCKON: 2 MHz RC Oscillator Enable**

At start-up, the 2 MHz Fast RC Oscillator is enabled.

0 = The 2 MHz Fast RC Oscillator is disabled

1 = The 2 MHz Fast RC Oscillator is enabled.

- **MCKSEL: Main Clock Selection**

0 = The 2 MHz Fast RC Oscillator is selected

1 = The CLKIN input is selected.

### 25.9.8 PMC Clock Generator Main Clock Frequency Register

**Register Name:**CKGR_MCFR

**Access Type:**Read-only

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

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINFRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

### 25.9.9 PMC Clock Generator PLL Register

**Register Name:**CKGR_PLLR

**Access Type:**Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | 0 | – | – | MUL | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUL | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STMODE | | PLLCOUNT | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DIV | | | | | | | |

Possible limitations on PLL input frequencies and multiplier factors should be checked before using the PMC.

**Warning:** Bit 29 must always be set to 0 when programming the CKGR_PLLR register.

- **DIV: Divider**

| DIV | Divider Selected |
|---|---|
| 0 | Divider output is 0 |
| 1 | Divider is bypassed (DIV=1) |
| 2 - 255 | Reserved |

- **PLLCOUNT: PLL Counter**

Specifies the number of Slow Clock cycles x8 before the LOCK bit is set in PMC_SR after CKGR_PLLR is written.

- **STMODE: Start Mode**

| STMODE | Start Mode |
|---|---|
| 0 | Fast Startup |
| 1 | Reserved |
| 2 | Normal Startup |
| 3 | Reserved |

STMODE must be set at 2 when the PLL is Off

- **MUL: PLL Multiplier**

0 = The PLL is deactivated.

1 up to 2047 = The PLL Clock frequency is the PLL input frequency multiplied by MUL + 1.

- **0: 0**

0 = Bit 29 must always be programmed to 0 when programming this register.

### 25.9.10 PMC Master Clock Register

**Register Name:**PMC_MCKR

**Access Type:**Read-write

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

- **CSS: Master Clock Selection**

| CSS | | Clock Source Selection |
|---|---|---|
| 0 | 0 | Slow Clock is selected |
| 0 | 1 | Main Clock is selected |
| 1 | 0 | PLL Clock is selected |
| 1 | 1 | Reserved |

- **PRES: Processor Clock Prescaler**

| PRES | | | Processor Clock |
|---|---|---|---|
| 0 | 0 | 0 | Selected clock |
| 0 | 0 | 1 | Selected clock divided by 2 |
| 0 | 1 | 0 | Selected clock divided by 4 |
| 0 | 1 | 1 | Selected clock divided by 8 |
| 1 | 0 | 0 | Selected clock divided by 16 |
| 1 | 0 | 1 | Selected clock divided by 32 |
| 1 | 1 | 0 | Selected clock divided by 64 |
| 1 | 1 | 1 | Reserved |

### 25.9.11 PMC Programmable Clock Register

**Register Name:**PMC_PCKx

**Access Type:**Read-write

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

- **CSS: Master Clock Selection**

| CSS | | Clock Source Selection |
|---|---|---|
| 0 | 0 | Slow Clock is selected |
| 0 | 1 | Main Clock is selected |
| 1 | 0 | PLL Clock is selected |
| 1 | 1 | Reserved |

- **PRES: Programmable Clock Prescaler**

| PRES | | | Programmable Clock |
|---|---|---|---|
| 0 | 0 | 0 | Selected clock |
| 0 | 0 | 1 | Selected clock divided by 2 |
| 0 | 1 | 0 | Selected clock divided by 4 |
| 0 | 1 | 1 | Selected clock divided by 8 |
| 1 | 0 | 0 | Selected clock divided by 16 |
| 1 | 0 | 1 | Selected clock divided by 32 |
| 1 | 1 | 0 | Selected clock divided by 64 |
| 1 | 1 | 1 | Reserved |

### 25.9.12 PMC Interrupt Enable Register

**Register Name:**PMC_IER

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCKRDY2 | PCKRDY1 | PCKRDY0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | MCKRDY | – | LOCK | MAINRDY |

- **MAINRDY: Main Clock Ready Interrupt Enable**
- **LOCK: PLL Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 25.9.13 PMC Interrupt Disable Register

**Register Name:**PMC_IDR

**Access Type:**Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCKRDY2 | PCKRDY1 | PCKRDY0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | MCKRDY | – | LOCK | MAINRDY |

- **MAINRDY: Main Clock Ready Interrupt Disable**
- **LOCK: PLL Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

### 25.9.14 PMC Status Register

**Register Name:**PMC_SR

**Access Type:**Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | MAINSELS |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCKRDY2 | PCKRDY1 | PCKRDY0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | MCKRDY | – | LOCK | MAINRDY |

- **MAINRDY: MAINRDY Flag Status**

0 = Main Clock is not ready

1 = Main Clock is ready

- **LOCK: PLL Lock Status**

0 = PLL is not locked

1 = PLL is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

- **MAINSELS: MAINSELS Main Clock Selection Status**

0 = Selection is in progress (default state)

1 = Selection is done

### 25.9.15 PMC Interrupt Mask Register

**Register Name:**PMC_IMR

**Access Type:**Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCKRDY2 | PCKRDY1 | PCKRDY0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | MCKRDY | – | LOCK | MAINRDY |

- **MAINRDY: Main Clock Ready Interrupt Mask**
- **LOCK: PLL Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

## 25.10 PMC Fast Startup Mode Register

**Register Name:** PMC_FSMR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FSTT15 | FSTT14 | FSTT13 | FSTT12 | FSTT11 | FSTT10 | FSTT9 | FSTT8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FSTT7 | FSTT6 | FSTT5 | FSTT4 | FSTT3 | FSTT2 | FSTT1 | FSTT0 |

- **FSTT0 - FSTT15: Fast Start Input Enable 0 to 15**

0 = The corresponding wake up input has no effect on the Power Management Controller.

1 = The corresponding wake up input enables a fast restart signal to the Power Management Controller.

# 26. Debug Unit (DBGU)

## 26.1 Overview

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

## 26.2 Block Diagram

**Figure 26-1.** Debug Unit Functional Block Diagram

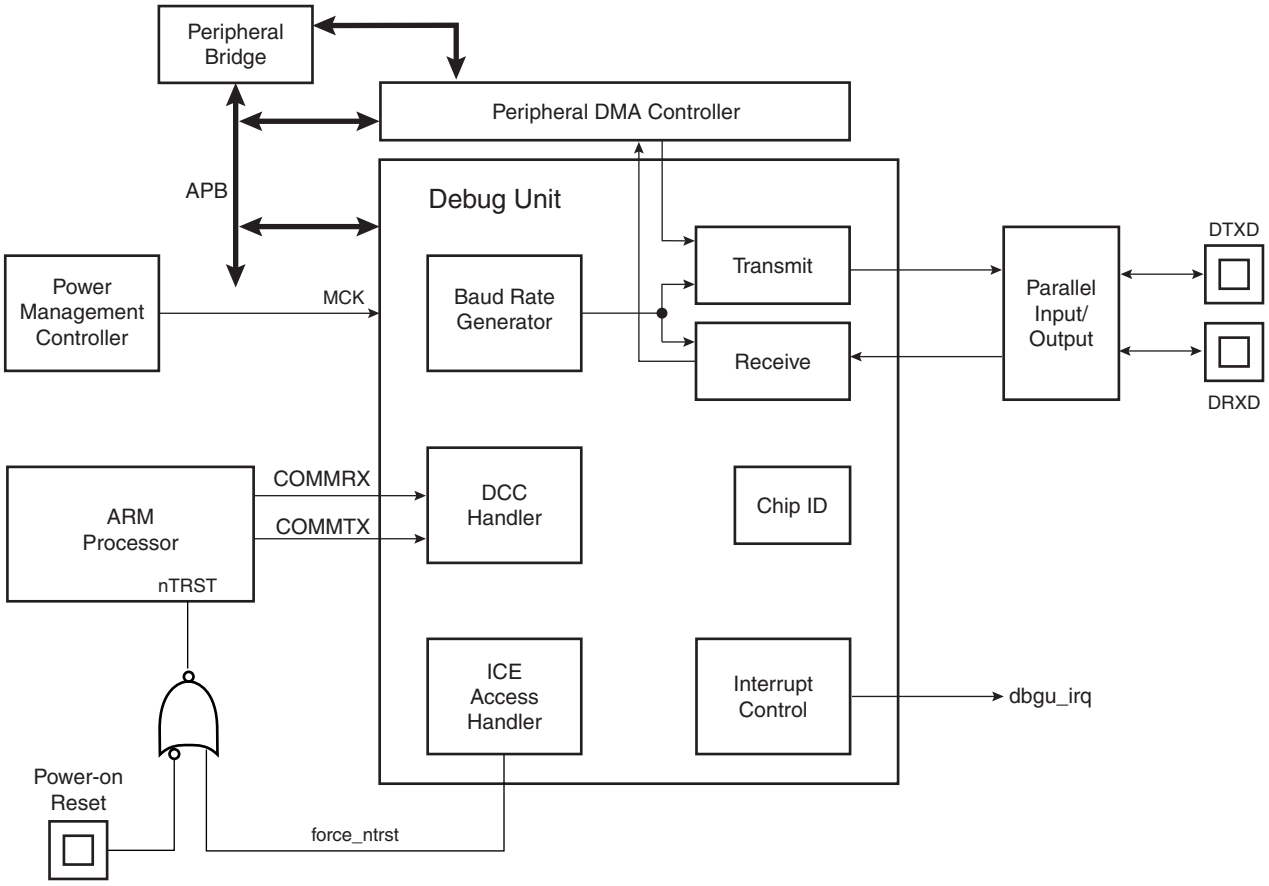
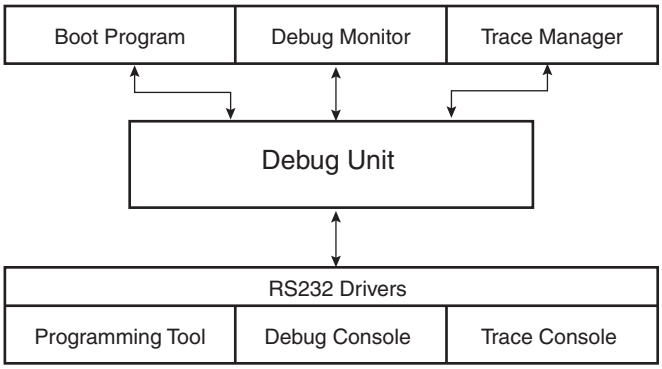



**Table 26-1.** Debug Unit Pin Description

| Pin Name | Description | Type |
|---|---|---|
| DRXD | Debug Receive Data | Input |
| DTXD | Debug Transmit Data | Output |

**Figure 26-2.** Debug Unit Application Example



## 26.3 Product Dependencies

### 26.3.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

### 26.3.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

### 26.3.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 26-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

## 26.4 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

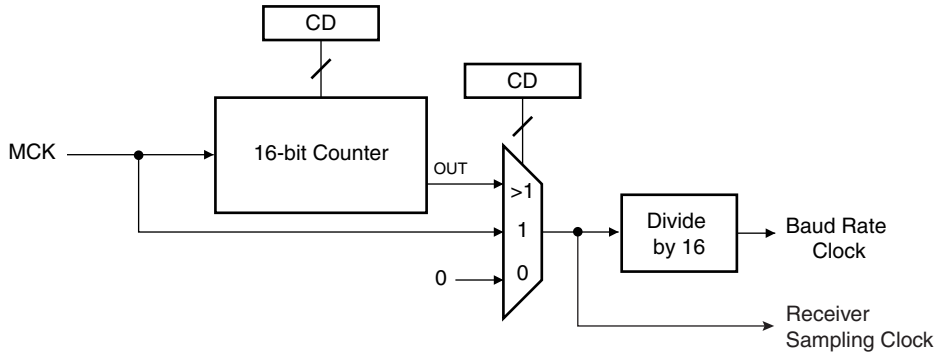
### 26.4.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU_BRGR (Baud Rate Generator Register). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

$$\text{Baud Rate} = \frac{\text{MCK}}{16 \times \text{CD}}$$

**Figure 26-3.** Baud Rate Generator




### 26.4.2 Receiver

#### *26.4.2.1 Receiver Reset, Enable and Disable*

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register DBGU_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing DBGU_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing DBGU_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

#### *26.4.2.2 Start Detection and Data Sampling*

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical mid-point of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

**Figure 26-4.** Start Bit Detection

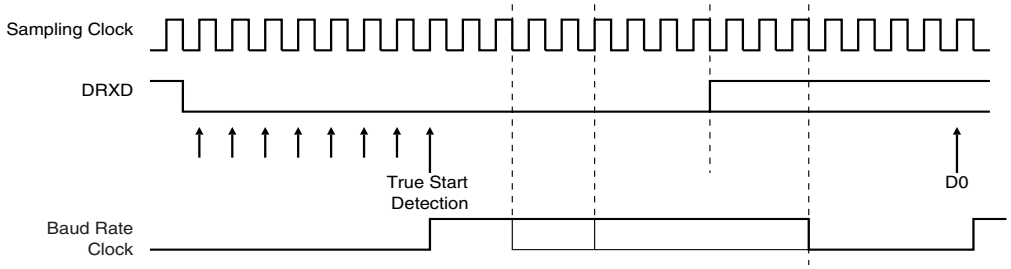
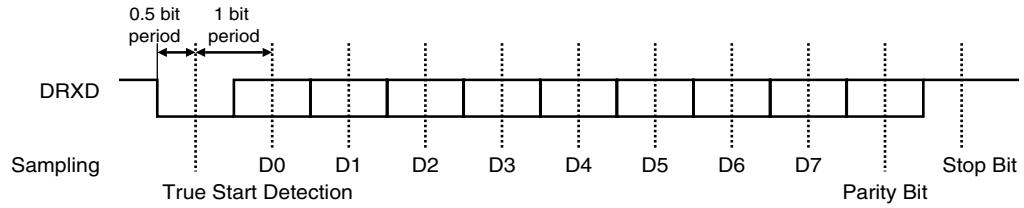



**Figure 26-5.** Character Reception

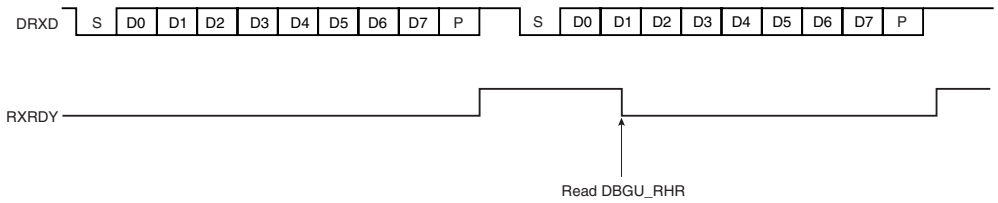
Example: 8-bit, parity enabled 1 stop




### *26.4.2.3 Receiver Ready*

When a complete character is received, it is transferred to the DBGU_RHR and the RXRDY status bit in DBGU_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

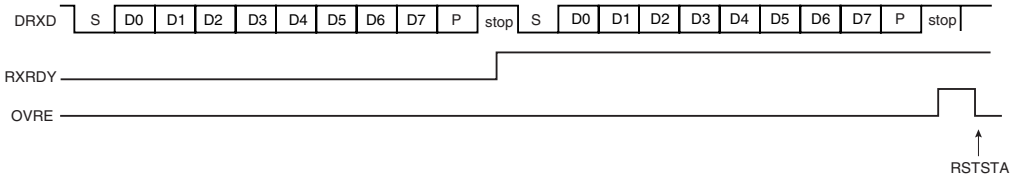
**Figure 26-6.** Receiver Ready




### *26.4.2.4 Receiver Overrun*

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes the control register DBGU_CR with the bit RSTSTA (Reset Status) at 1.

**Figure 26-7.** Receiver Overrun

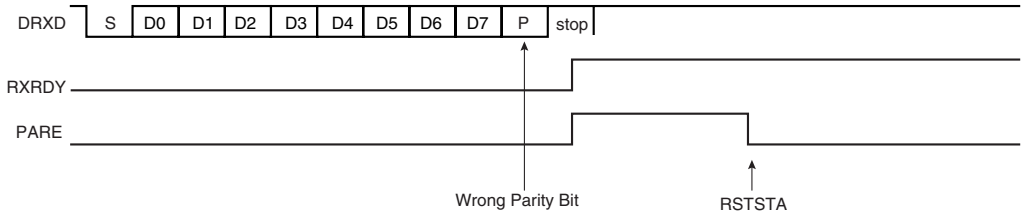



### *26.4.2.5 Parity Error*

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in DBGU_MR. It then compares the result with the received parity

bit. If different, the parity error bit PARE in DBGU_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

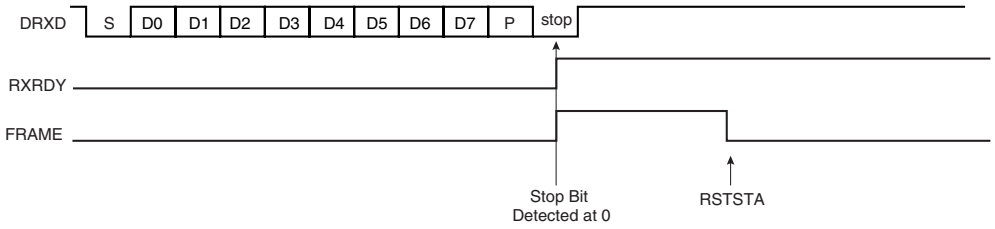
**Figure 26-8.** Parity Error




*26.4.2.6 Receiver Framing Error*

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU_CR is written with the bit RSTSTA at 1.

**Figure 26-9.** Receiver Framing Error




### 26.4.3 Transmitter

*26.4.3.1 Transmitter Reset, Enable and Disable*

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU_THR before actually starting the transmission.

The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

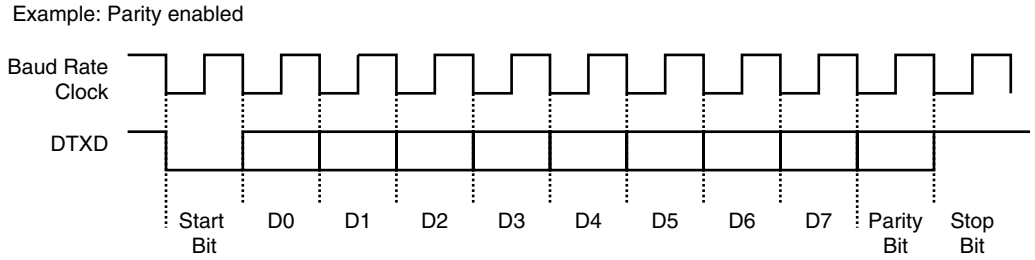
The programmer can also put the transmitter in its reset state by writing the DBGU_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

*26.4.3.2 Transmit Format*

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field

PARE in the mode register DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

**Figure 26-10.** Character Transmission

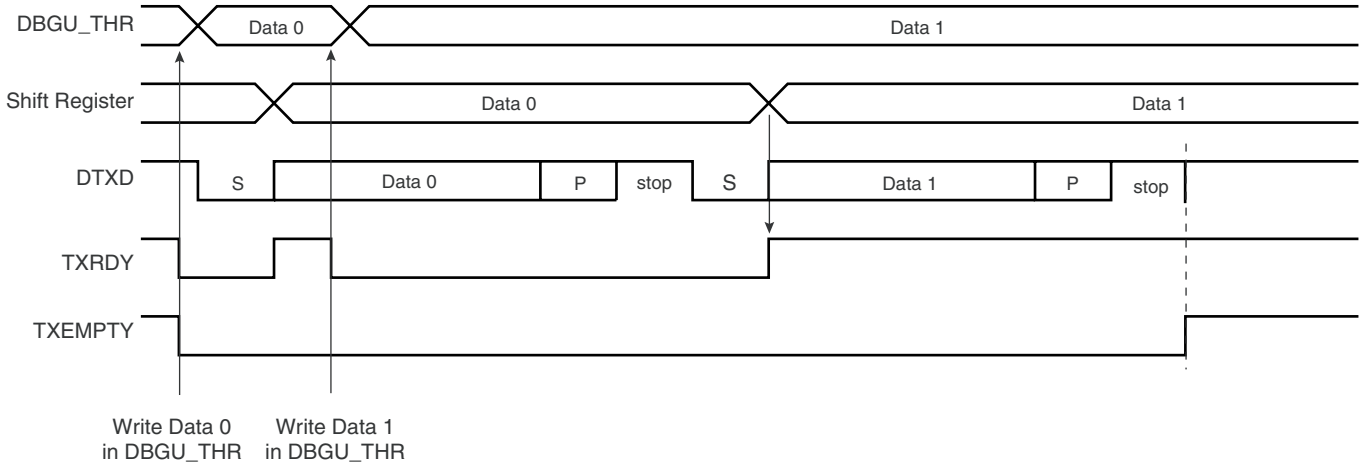



### *26.4.3.3 Transmitter Control*

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

**Figure 26-11.** Transmitter Control




## 26.4.4 Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU_THR.

### 26.4.5 Test Modes

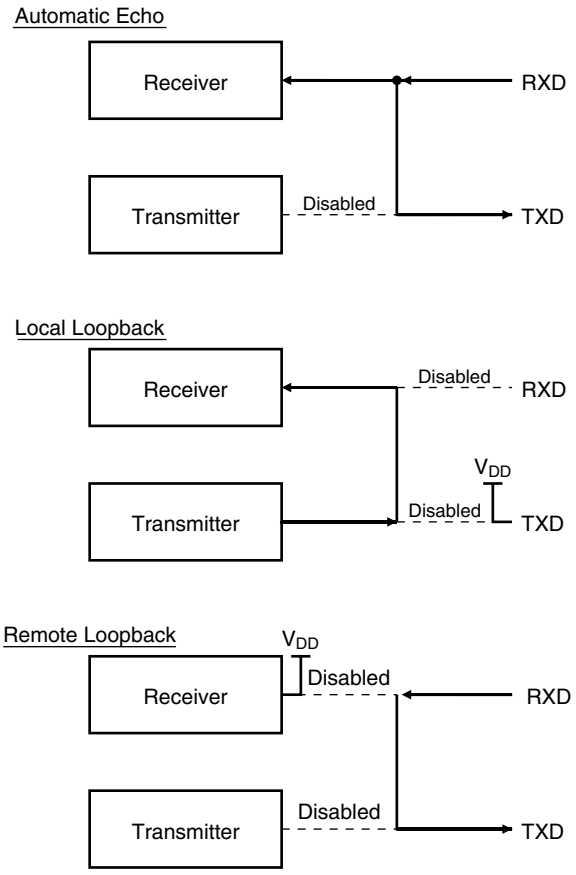
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

The Local Loopback mode allows the transmitted characters to be received. DTXD and DRXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The DRXD pin level has no effect and the DTXD line is held high, as in idle state.

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

**Figure 26-12.** Test Modes




### 26.4.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

### 26.4.7 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripherals
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

### 26.4.8 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

## 26.5 Debug Unit (DBGU) User Interface

**Table 26-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | Control Register | DBGU_CR | Write-only | – |
| 0x0004 | Mode Register | DBGU_MR | Read-write | 0x0 |
| 0x0008 | Interrupt Enable Register | DBGU_IER | Write-only | – |
| 0x000C | Interrupt Disable Register | DBGU_IDR | Write-only | – |
| 0x0010 | Interrupt Mask Register | DBGU_IMR | Read-only | 0x0 |
| 0x0014 | Status Register | DBGU_SR | Read-only | – |
| 0x0018 | Receive Holding Register | DBGU_RHR | Read-only | 0x0 |
| 0x001C | Transmit Holding Register | DBGU_THR | Write-only | – |
| 0x0020 | Baud Rate Generator Register | DBGU_BRGR | Read-write | 0x0 |
| 0x0024 - 0x003C | Reserved | – | – | – |
| 0x0040 | Chip ID Register | DBGU_CIDR | Read-only | – |
| 0x0044 | Chip ID Extension Register | DBGU_EXID | Read-only | – |
| 0x0048 | Force NTRST Register | DBGU_FNR | Read-write | 0x0 |
| 0x0100 - 0x0124 | PDC Area | – | – | – |

### 26.5.1 Debug Unit Control Register

**Name:** DBGU_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | RSTSTA |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDIS | TXEN | RXDIS | RXEN | RSTTX | RSTRX | – | – |

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU_SR.

### 26.5.2 Debug Unit Mode Register

**Name:** DBGU_MR

**Access Type:** Read-write

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

- **PAR: Parity Type**

| PAR | | | Parity Type |
|---|---|---|---|
| 0 | 0 | 0 | Even parity |
| 0 | 0 | 1 | Odd parity |
| 0 | 1 | 0 | Space: parity forced to 0 |
| 0 | 1 | 1 | Mark: parity forced to 1 |
| 1 | x | x | No parity |

- **CHMODE: Channel Mode**

| CHMODE | | Mode Description |
|---|---|---|
| 0 | 0 | Normal Mode |
| 0 | 1 | Automatic Echo |
| 1 | 0 | Local Loopback |
| 1 | 1 | Remote Loopback |

### 26.5.3 Debug Unit Interrupt Enable Register

**Name:** DBGU_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | RXBUFF | TXBUFE | – | TXEMPTY | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | – | TXRDY | RXRDY |

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

### 26.5.4 Debug Unit Interrupt Disable Register

**Name:** DBGU_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | RXBUFF | TXBUFE | – | TXEMPTY | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | – | TXRDY | RXRDY |

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

### 26.5.5 Debug Unit Interrupt Mask Register

**Name:** DBGU_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | RXBUFF | TXBUFE | – | TXEMPTY | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | – | TXRDY | RXRDY |

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

### 26.5.6 Debug Unit Status Register

**Name:** DBGU_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | RXBUFF | TXBUFE | – | TXEMPTY | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | – | TXRDY | RXRDY |

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the DBGU_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the receiver Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the receiver Peripheral Data Controller channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is active.

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

### 26.5.7 Debug Unit Receiver Holding Register

**Name:** DBGU_RHR

**Access Type:** Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

### 26.5.8 Debug Unit Transmit Holding Register

**Name:** DBGU_THR

**Access Type:** Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

### 26.5.9 Debug Unit Baud Rate Generator Register

**Name:** DBGU_BRGR

**Access Type:** Read-write

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

- **CD: Clock Divisor**

| CD | Baud Rate Clock |
|---|---|
| 0 | Disabled |
| 1 | MCK |
| 2 to 65535 | MCK / (CD x 16) |

### 26.5.10 Debug Unit Chip ID Register

**Name:** DBGU_CIDR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| EXT | NVPTYP | | | ARCH | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ARCH | | | | SRAMSIZ | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| NVPSIZ2 | | | | NVPSIZ | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EPROC | | | VERSION | | | | |

- **VERSION: Version of the Device**

Current version of the device.

- **EPROC: Embedded Processor**

| EPROC | | | Processor |
|---|---|---|---|
| 0 | 0 | 1 | ARM946ES |
| 0 | 1 | 0 | ARM7TDMI |
| 1 | 0 | 0 | ARM920T |
| 1 | 0 | 1 | ARM926EJS |

- **NVPSIZ: Nonvolatile Program Memory Size**

| NVPSIZ | | | | Size |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | None |
| 0 | 0 | 0 | 1 | 8K bytes |
| 0 | 0 | 1 | 0 | 16K bytes |
| 0 | 0 | 1 | 1 | 32K bytes |
| 0 | 1 | 0 | 0 | Reserved |
| 0 | 1 | 0 | 1 | 64K bytes |
| 0 | 1 | 1 | 0 | Reserved |
| 0 | 1 | 1 | 1 | 128K bytes |
| 1 | 0 | 0 | 0 | Reserved |
| 1 | 0 | 0 | 1 | 256K bytes |
| 1 | 0 | 1 | 0 | 512K bytes |
| 1 | 0 | 1 | 1 | Reserved |
| 1 | 1 | 0 | 0 | 1024K bytes |

| NVPSIZ | | | | Size |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | Reserved |
| 1 | 1 | 1 | 0 | 2048K bytes |
| 1 | 1 | 1 | 1 | Reserved |

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

| NVPSIZ2 | | | | Size |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | None |
| 0 | 0 | 0 | 1 | 8K bytes |
| 0 | 0 | 1 | 0 | 16K bytes |
| 0 | 0 | 1 | 1 | 32K bytes |
| 0 | 1 | 0 | 0 | Reserved |
| 0 | 1 | 0 | 1 | 64K bytes |
| 0 | 1 | 1 | 0 | Reserved |
| 0 | 1 | 1 | 1 | 128K bytes |
| 1 | 0 | 0 | 0 | Reserved |
| 1 | 0 | 0 | 1 | 256K bytes |
| 1 | 0 | 1 | 0 | 512K bytes |
| 1 | 0 | 1 | 1 | Reserved |
| 1 | 1 | 0 | 0 | 1024K bytes |
| 1 | 1 | 0 | 1 | Reserved |
| 1 | 1 | 1 | 0 | 2048K bytes |
| 1 | 1 | 1 | 1 | Reserved |

- **SRAMSIZ: Internal SRAM Size**

| SRAMSIZ | | | | Size |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Reserved |
| 0 | 0 | 0 | 1 | 1K bytes |
| 0 | 0 | 1 | 0 | 2K bytes |
| 0 | 0 | 1 | 1 | 6K bytes |
| 0 | 1 | 0 | 0 | 112K bytes |
| 0 | 1 | 0 | 1 | 4K bytes |
| 0 | 1 | 1 | 0 | 80K bytes |
| 0 | 1 | 1 | 1 | 160K bytes |
| 1 | 0 | 0 | 0 | 8K bytes |
| 1 | 0 | 0 | 1 | 16K bytes |
| 1 | 0 | 1 | 0 | 32K bytes |
| 1 | 0 | 1 | 1 | 64K bytes |

| SRAMSIZ | | | | Size |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 128K bytes |
| 1 | 1 | 0 | 1 | 256K bytes |
| 1 | 1 | 1 | 0 | 96K bytes |
| 1 | 1 | 1 | 1 | 512K bytes |

- **ARCH: Architecture Identifier**

| ARCH | | |
|---|---|---|
| Hex | Bin | Architecture |
| 0x19 | 0001 1001 | AT91SAM9xx Series |
| 0x29 | 0010 1001 | AT91SAM9XExx Series |
| 0x34 | 0011 0100 | AT91x34 Series |
| 0x37 | 0011 0111 | CAP7 Series |
| 0x39 | 0011 1001 | CAP9 Series |
| 0x3B | 0011 1011 | CAP11 Series |
| 0x40 | 0100 0000 | AT91x40 Series |
| 0x42 | 0100 0010 | AT91x42 Series |
| 0x55 | 0101 0101 | AT91x55 Series |
| 0x60 | 0110 0000 | AT91SAM7Axx Series |
| 0x61 | 0110 0001 | AT91SAM7AQxx Series |
| 0x63 | 0110 0011 | AT91x63 Series |
| 0x70 | 0111 0000 | AT91SAM7Sxx Series |
| 0x71 | 0111 0001 | AT91SAM7XCxx Series |
| 0x72 | 0111 0010 | AT91SAM7SExx Series |
| 0x73 | 0111 0011 | AT91SAM7Lxx Series |
| 0x75 | 0111 0101 | AT91SAM7Xxx Series |
| 0x92 | 1001 0010 | AT91x92 Series |
| 0xF0 | 1111 0000 | AT75Cxx Series |

- **NVPTYP: Nonvolatile Program Memory Type**

| NVPTYP | | | Memory |
|---|---|---|---|
| 0 | 0 | 0 | ROM |
| 0 | 0 | 1 | ROMless or on-chip Flash |
| 1 | 0 | 0 | SRAM emulating ROM |
| 0 | 1 | 0 | Embedded Flash Memory |
| 0 | 1 | 1 | ROM and Embedded Flash Memory<br>NVPSIZ is ROM size<br>NVPSIZ2 is Flash size |

- **EXT: Extension Flag**

0 = Chip ID has a single register definition without extension

1 = An extended Chip ID exists.

### 26.5.11 Debug Unit Chip ID Extension Register

**Name:** DBGU_EXID

**Access Type:** Read-only

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

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

### 26.5.12 Debug Unit Force NTRST Register

**Name:** DBGU_FNR

**Access Type:**Read-write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor's TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor's TAP controller is held low.

# 27. Parallel Input Output Controller (PIO)

## 27.1 Overview

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

## 27.2 Block Diagram

**Figure 27-1.** Block Diagram

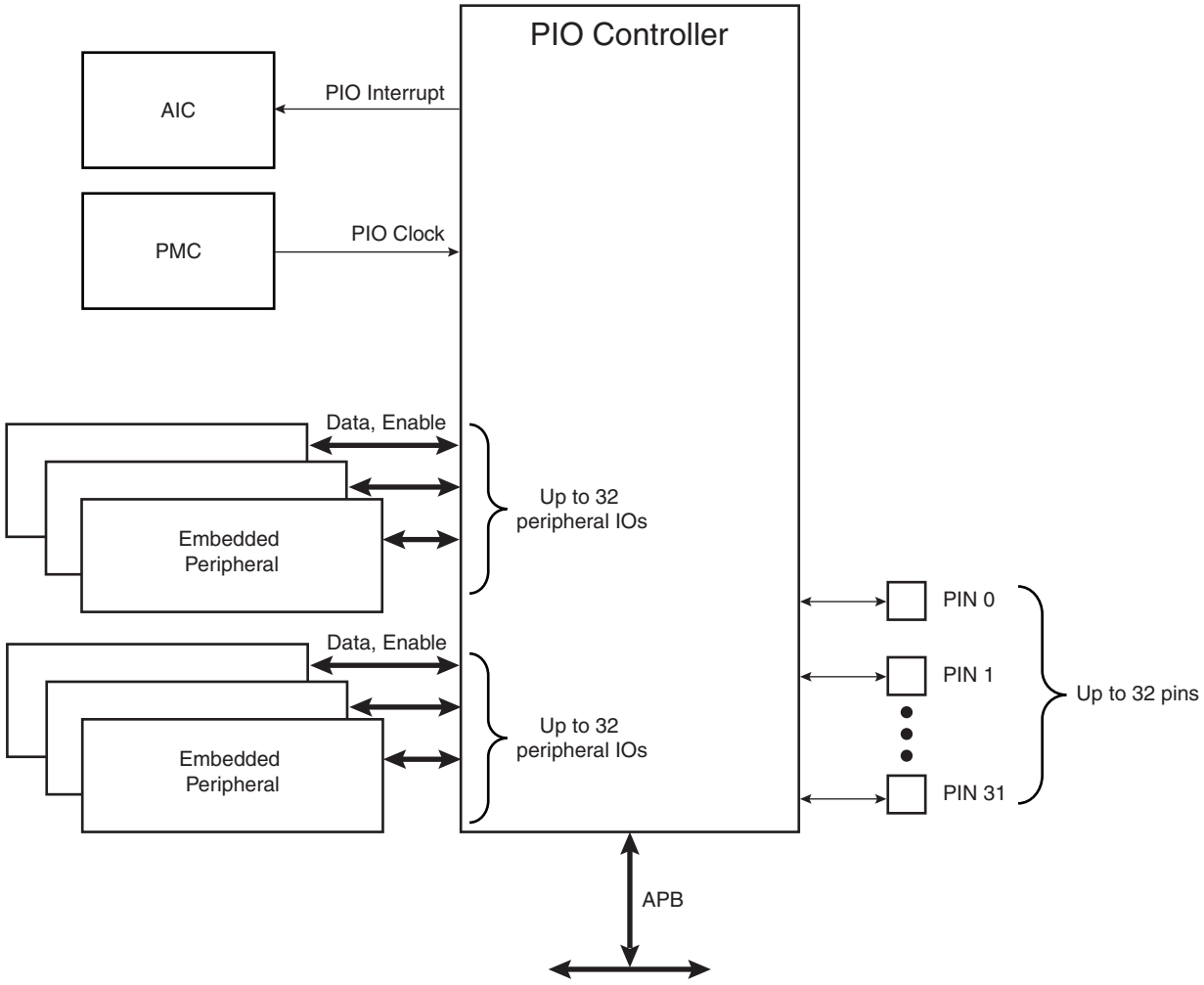
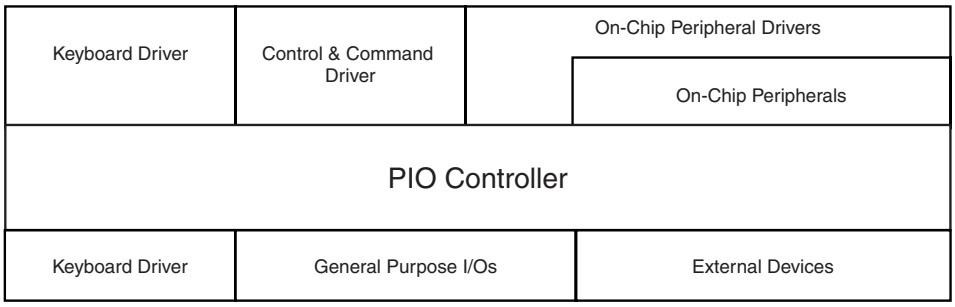



**Figure 27-2.** Application Block Diagram



## 27.3 Product Dependencies

### 27.3.1 Pin Multiplexing

Each pin is configurable, according to product definition as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware-defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO controllers required by their application. When an I/O line is general-purpose only, i.e. not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

### 27.3.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are most generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

### 27.3.3 Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

### 27.3.4 Interrupt Generation

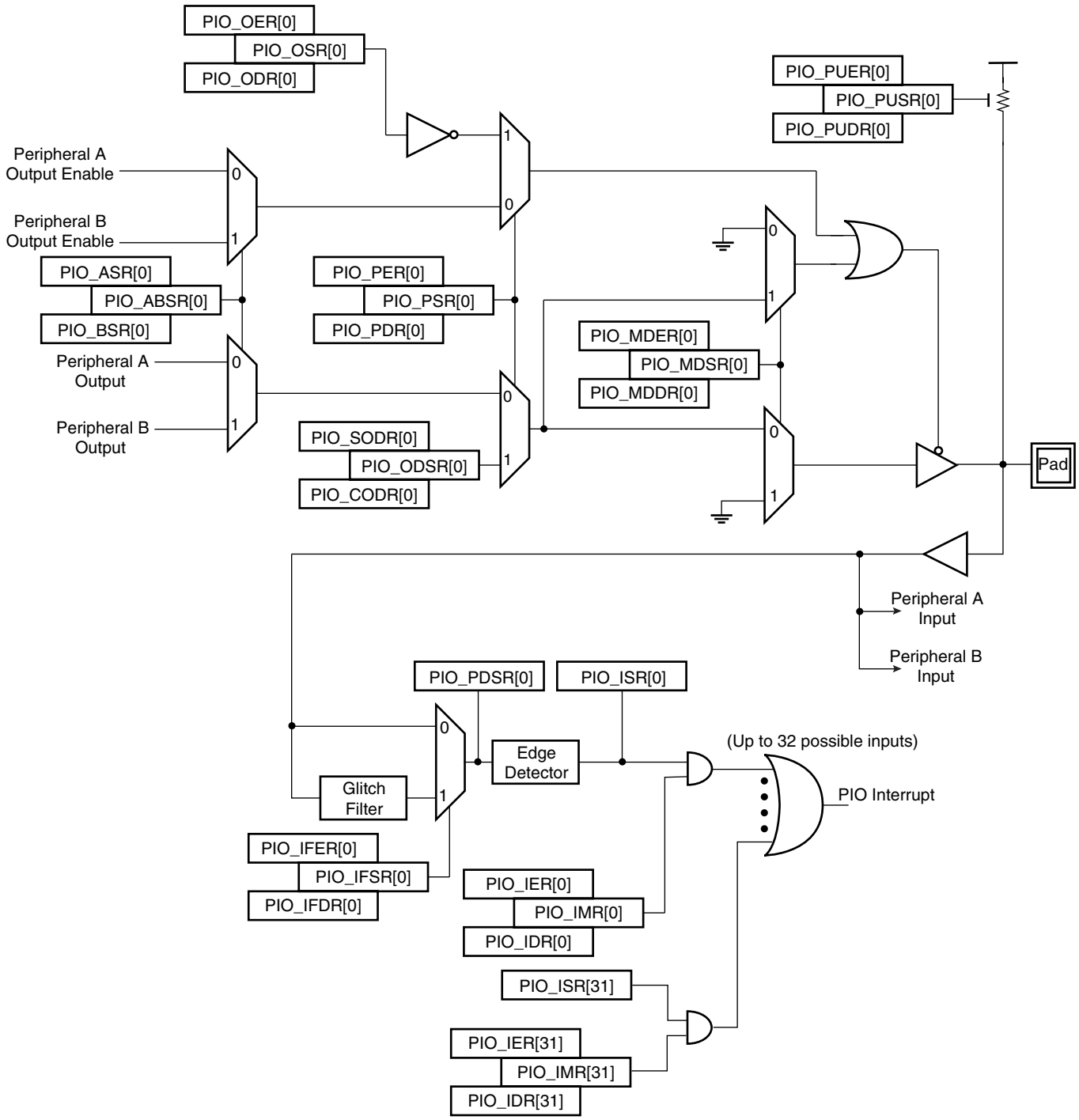
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

## 27.4 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 27-3. In this description each signal shown represents but one of up to 32 possible indexes.

**Figure 27-3.** I/O Line Control Logic



### 27.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by writing respectively PIO_PUER (Pull-up Enable Register) and PIO_PUDR (Pull-up Disable Resistor). Writing in these registers results in setting or clearing the corresponding bit in PIO_PUSR (Pull-up Status Register). Reading a 1 in PIO_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO_PUSR resets at the value 0x0.

### 27.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

### 27.4.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ASR (A Select Register) and PIO_BSR (Select B Register). PIO_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ASR and PIO_BSR manages PIO_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO_ASR or PIO_BSR) in addition to a write in PIO_PDR.

### 27.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register).

The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

### 27.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

### 27.4.6 Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

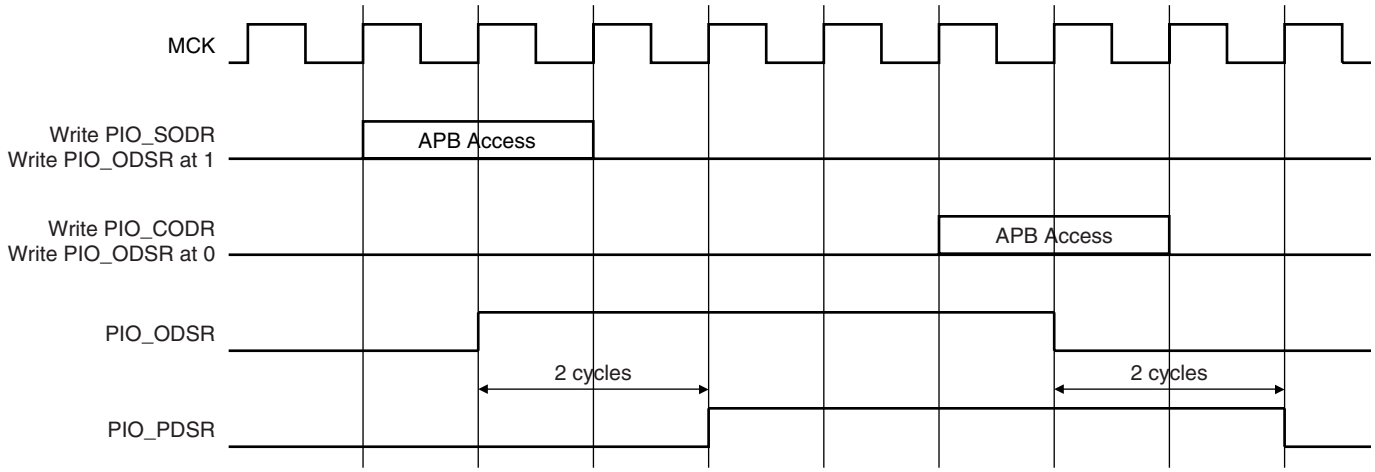
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates the pins that are configured to support external drivers.

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

### 27.4.7 Output Line Timings

Figure 27-4 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 27-4 also shows when the feedback in PIO_PDSR is available.

**Figure 27-4.** Output Line Timings




### 27.4.8 Inputs

The level on each I/O line can be read through PIO_PDSR (Pin Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

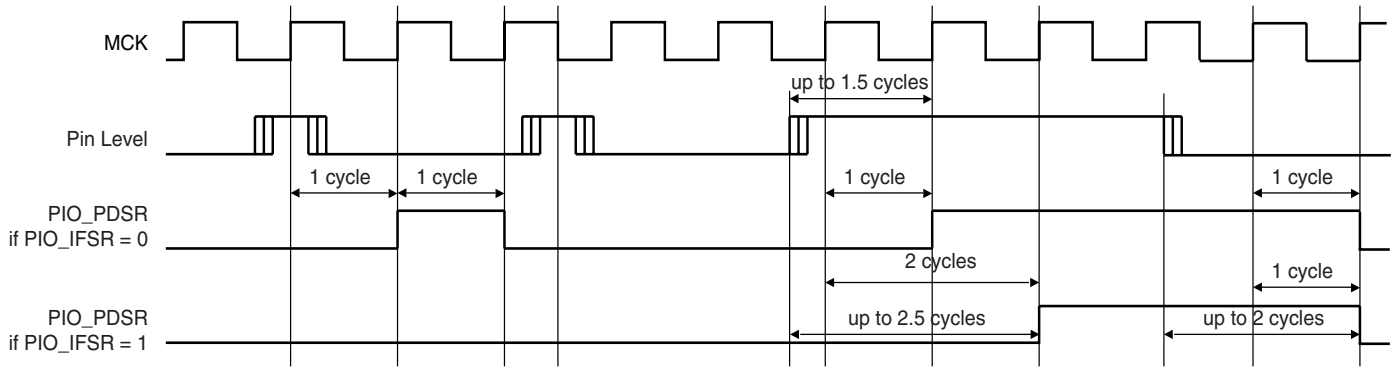
### 27.4.9 Input Glitch Filtering

Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 27-5.

The glitch filters are controlled by the register set; PIO_IFER (Input Filter Enable Register), PIO_IFDR (Input Filter Disable Register) and PIO_IFSR (Input Filter Status Register). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.

**Figure 27-5.** Input Glitch Filter Timing



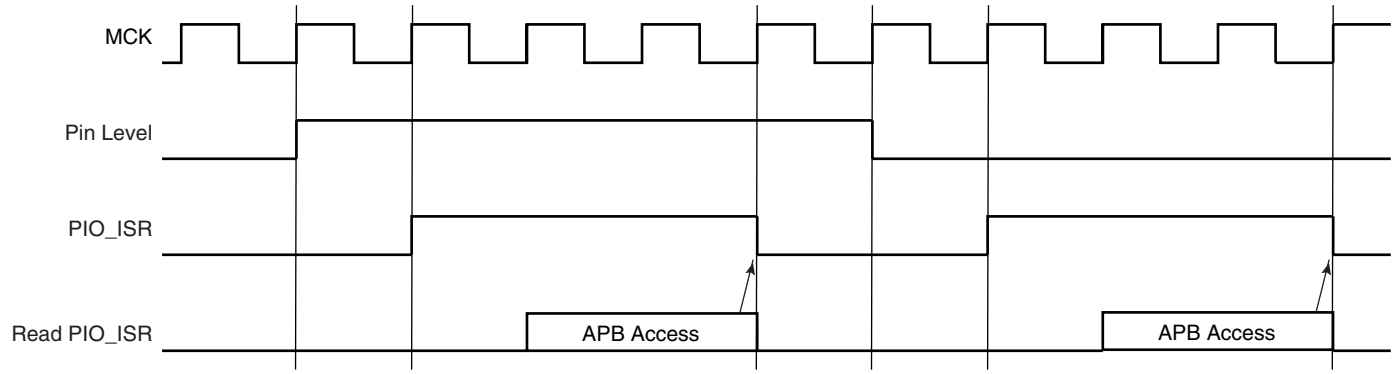

### 27.4.10 Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO_IER (Interrupt Enable Register) and PIO_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled.

**Figure 27-6.** Input Change Interrupt Timings




## 27.5 I/O Lines Programming Example

The programing example as shown in Table 27-1 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor

- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O line 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

**Table 27-1.** Programming Example

| Register | Value to be Written |
|---|---|
| PIO_PER | 0x0000 FFFF |
| PIO_PDR | 0x0FFF 0000 |
| PIO_OER | 0x0000 00FF |
| PIO_ODR | 0x0FFF FF00 |
| PIO_IFER | 0x0000 0F00 |
| PIO_IFDR | 0x0FFF F0FF |
| PIO_SODR | 0x0000 0000 |
| PIO_CODR | 0x0FFF FFFF |
| PIO_IER | 0x0F00 0F00 |
| PIO_IDR | 0x00FF F0FF |
| PIO_MDER | 0x0000 000F |
| PIO_MDDR | 0x0FFF FFF0 |
| PIO_PUDR | 0x00F0 00F0 |
| PIO_PUER | 0x0F0F FF0F |
| PIO_ASR | 0x0F0F 0000 |
| PIO_BSR | 0x00F0 0000 |
| PIO_OWER | 0x0000 000F |
| PIO_OWDR | 0x0FFF FFF0 |

## 27.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

**Table 27-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | PIO Enable Register | PIO_PER | Write-only | – |
| 0x0004 | PIO Disable Register | PIO_PDR | Write-only | – |
| 0x0008 | PIO Status Register | PIO_PSR | Read-only | (1) |
| 0x000C | Reserved | | | |
| 0x0010 | Output Enable Register | PIO_OER | Write-only | – |
| 0x0014 | Output Disable Register | PIO_ODR | Write-only | – |
| 0x0018 | Output Status Register | PIO_OSR | Read-only | 0x0000 0000 |
| 0x001C | Reserved | | | |
| 0x0020 | Glitch Input Filter Enable Register | PIO_IFER | Write-only | – |
| 0x0024 | Glitch Input Filter Disable Register | PIO_IFDR | Write-only | – |
| 0x0028 | Glitch Input Filter Status Register | PIO_IFSR | Read-only | 0x0000 0000 |
| 0x002C | Reserved | | | |
| 0x0030 | Set Output Data Register | PIO_SODR | Write-only | – |
| 0x0034 | Clear Output Data Register | PIO_CODR | Write-only | |
| 0x0038 | Output Data Status Register | PIO_ODSR | Read-only or(2) Read-write | – |
| 0x003C | Pin Data Status Register | PIO_PDSR | Read-only | (3) |
| 0x0040 | Interrupt Enable Register | PIO_IER | Write-only | – |
| 0x0044 | Interrupt Disable Register | PIO_IDR | Write-only | – |
| 0x0048 | Interrupt Mask Register | PIO_IMR | Read-only | 0x00000000 |
| 0x004C | Interrupt Status Register(4) | PIO_ISR | Read-only | 0x00000000 |
| 0x0050 | Multi-driver Enable Register | PIO_MDER | Write-only | – |
| 0x0054 | Multi-driver Disable Register | PIO_MDDR | Write-only | – |
| 0x0058 | Multi-driver Status Register | PIO_MDSR | Read-only | 0x00000000 |
| 0x005C | Reserved | | | |
| 0x0060 | Pull-up Disable Register | PIO_PUDR | Write-only | – |
| 0x0064 | Pull-up Enable Register | PIO_PUER | Write-only | – |
| 0x0068 | Pad Pull-up Status Register | PIO_PUSR | Read-only | 0x00000000 |
| 0x006C | Reserved | | | |

**Table 27-2.** Register Mapping (Continued)

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0070 | Peripheral A Select Register[5] | PIO_ASR | Write-only | – |
| 0x0074 | Peripheral B Select Register[5] | PIO_BSR | Write-only | – |
| 0x0078 | AB Status Register[5] | PIO_ABSR | Read-only | 0x00000000 |
| 0x007C to 0x009C | Reserved | | | |
| 0x00A0 | Output Write Enable | PIO_OWER | Write-only | – |
| 0x00A4 | Output Write Disable | PIO_OWDR | Write-only | – |
| 0x00A8 | Output Write Status Register | PIO_OWSR | Read-only | 0x00000000 |
| 0x00AC | Reserved | | | |

Notes:
1. Reset value of PIO_PSR depends on the product implementation.
2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

### 27.6.1 PIO Controller PIO Enable Register

**Name:** PIO_PER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: PIO Enable**

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

### 27.6.2 PIO Controller PIO Disable Register

**Name:** PIO_PDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: PIO Disable**

0 = No effect.

1 = Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

### 27.6.3 PIO Controller PIO Status Register

**Name:** PIO_PSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: PIO Status**

0 = PIO is inactive on the corresponding I/O line (peripheral is active).

1 = PIO is active on the corresponding I/O line (peripheral is inactive).

### 27.6.4 PIO Controller Output Enable Register

**Name:** PIO_OER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

### 27.6.5 PIO Controller Output Disable Register

**Name:** PIO_ODR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

### 27.6.6 PIO Controller Output Status Register

**Name:** PIO_OSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

### 27.6.7 PIO Controller Input Filter Enable Register

**Name:** PIO_IFER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Filter Enable**

0 = No effect.

1 = Enables the input glitch filter on the I/O line.

### 27.6.8 PIO Controller Input Filter Disable Register

**Name:** PIO_IFDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Filter Disable**

0 = No effect.

1 = Disables the input glitch filter on the I/O line.

### 27.6.9 PIO Controller Input Filter Status Register

**Name:** PIO_IFSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Filer Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

### 27.6.10 PIO Controller Set Output Data Register

**Name:** PIO_SODR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

### 27.6.11 PIO Controller Clear Output Data Register

**Name:** PIO_CODR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

### 27.6.12 PIO Controller Output Data Status Register

**Name:** PIO_ODSR

**Access Type:** Read-only or Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

### 27.6.13 PIO Controller Pin Data Status Register

**Name:** PIO_PDSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

### 27.6.14 PIO Controller Interrupt Enable Register

**Name:** PIO_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Change Interrupt Enable**

0 = No effect.

1 = Enables the Input Change Interrupt on the I/O line.

### 27.6.15 PIO Controller Interrupt Disable Register

**Name:** PIO_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Change Interrupt Disable**

0 = No effect.

1 = Disables the Input Change Interrupt on the I/O line.

### 27.6.16 PIO Controller Interrupt Mask Register

**Name:** PIO_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Change Interrupt Mask**

0 = Input Change Interrupt is disabled on the I/O line.

1 = Input Change Interrupt is enabled on the I/O line.

### 27.6.17 PIO Controller Interrupt Status Register

**Name:** PIO_ISR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Input Change Interrupt Status**

0 = No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1 = At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

### 27.6.18 PIO Multi-driver Enable Register

**Name:** PIO_MDER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Multi Drive Enable.**

0 = No effect.

1 = Enables Multi Drive on the I/O line.

### 27.6.19 PIO Multi-driver Disable Register

**Name:** PIO_MDDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Multi Drive Disable.**

0 = No effect.

1 = Disables Multi Drive on the I/O line.

### 27.6.20 PIO Multi-driver Status Register

**Name:** PIO_MDSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Multi Drive Status.**

0 = The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1 = The Multi Drive is enabled on the I/O line. The pin is driven at low level only.

### 27.6.21 PIO Pull Up Disable Register

**Name:** PIO_PUDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

### 27.6.22 PIO Pull Up Enable Register

**Name:** PIO_PUER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Pull Up Enable.**

0 = No effect.

1 = Enables the pull up resistor on the I/O line.

### 27.6.23 PIO Pull Up Status Register

**Name:** PIO_PUSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Pull Up Status.**

0 = Pull Up resistor is enabled on the I/O line.

1 = Pull Up resistor is disabled on the I/O line.

### 27.6.24 PIO Peripheral A Select Register

**Name:** PIO_ASR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Peripheral A Select.**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

### 27.6.25 PIO Peripheral B Select Register

**Name:** PIO_BSR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Peripheral B Select.**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.

### 27.6.26 PIO Peripheral A B Status Register

**Name:** PIO_ABSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Peripheral A B Status.**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

### 27.6.27 PIO Output Write Enable Register

**Name:** PIO_OWER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Write Enable.**

0 = No effect.

1 = Enables writing PIO_ODSR for the I/O line.

### 27.6.28 PIO Output Write Disable Register

**Name:** PIO_OWDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Write Disable.**

0 = No effect.

1 = Disables writing PIO_ODSR for the I/O line.

### 27.6.29 PIO Output Write Status Register

**Name:** PIO_OWSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0-P31: Output Write Status.**

0 = Writing PIO_ODSR does not affect the I/O line.

1 = Writing PIO_ODSR affects the I/O line.

# 28. Serial Peripheral Interface (SPI)

## 28.1 Overview

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

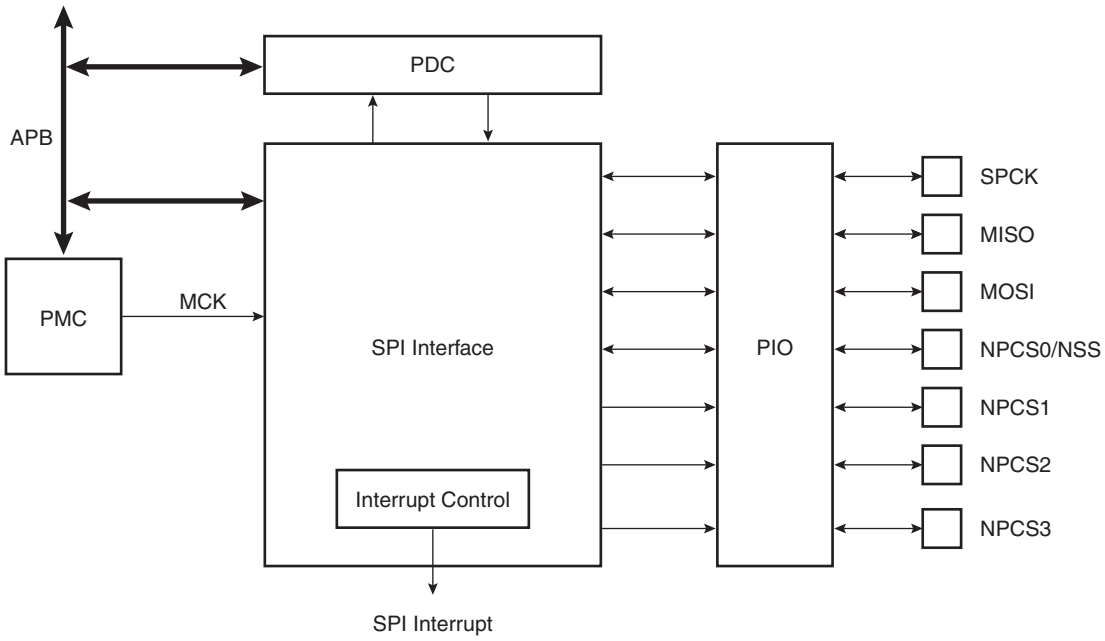
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

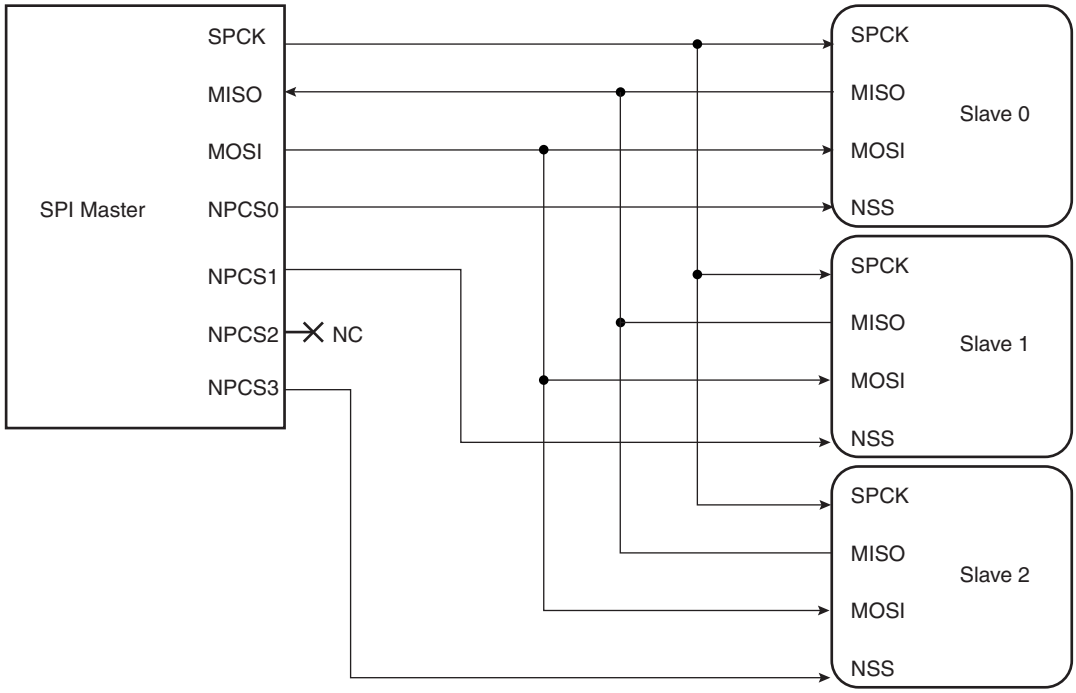
## 28.2 Block Diagram

**Figure 28-1.** Block Diagram




## 28.3 Application Block Diagram

**Figure 28-2.** Application Block Diagram: Single Master/Multiple Slave Implementation



## 28.4 Signal Description

**Table 28-1.** Signal Description

| Pin Name | Pin Description | Type | |
|---|---|---|---|
| | | Master | Slave |
| MISO | Master In Slave Out | Input | Output |
| MOSI | Master Out Slave In | Output | Input |
| SPCK | Serial Clock | Output | Input |
| NPCS1-NPCS3 | Peripheral Chip Selects | Output | Unused |
| NPCS0/NSS | Peripheral Chip Select/Slave Select | Output | Input |

## 28.5 Product Dependencies

### 28.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

### 28.5.2 Power Management

The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

### 28.5.3 Interrupt

The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.

## 28.6 Functional Description

### 28.6.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

### 28.6.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

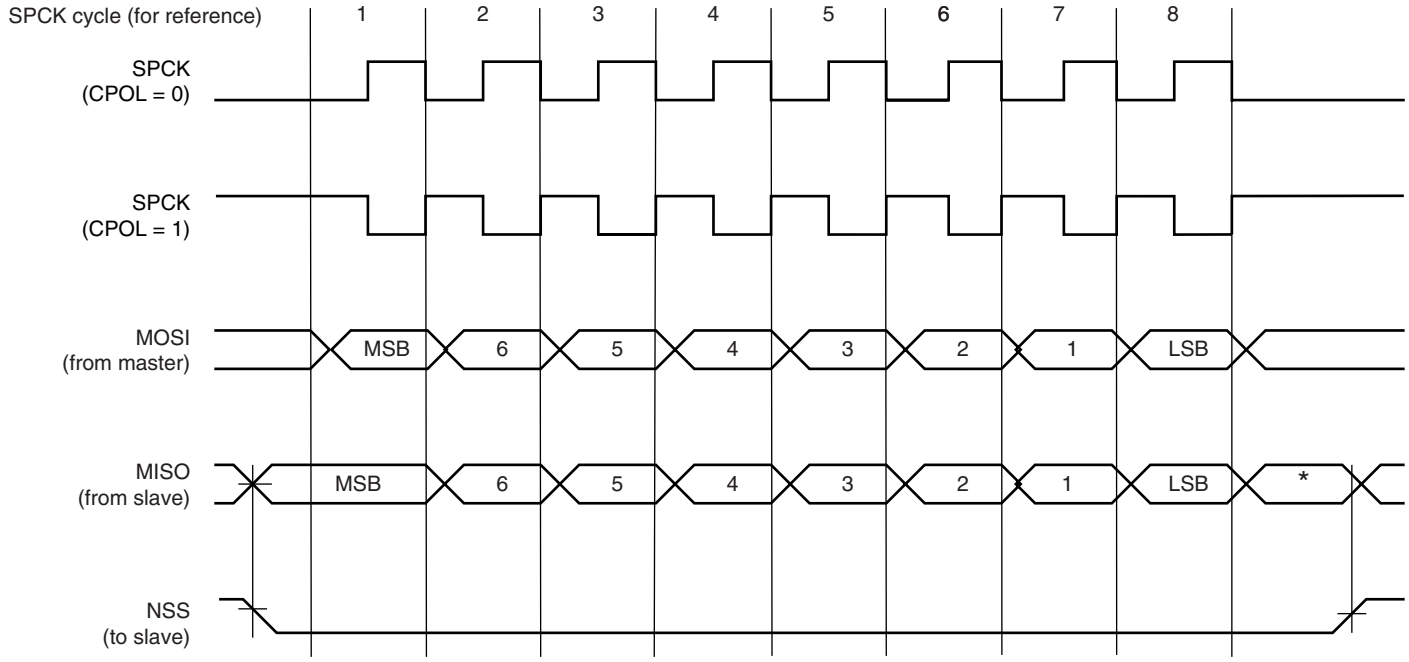
Table 28-2 shows the four modes and corresponding parameter settings.

**Table 28-2.** SPI Bus Protocol Mode

| SPI Mode | CPOL | NCPHA |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |

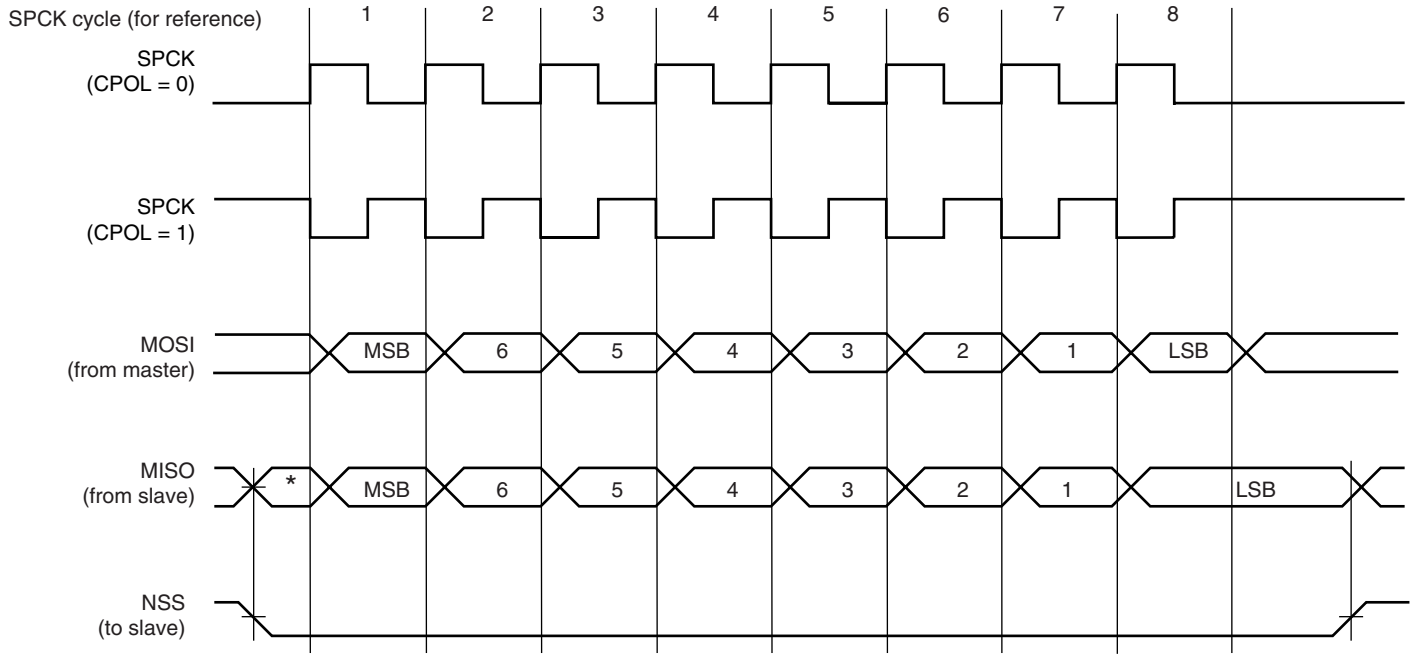
Figure 28-3 and Figure 28-4 show examples of data transfers.

**Figure 28-3.** SPI Transfer Format (NCPHA = 1, 8 bits per transfer)




* Not defined, but normally MSB of previous character received.

**Figure 28-4.** SPI Transfer Format (NCPHA = 0, 8 bits per transfer)




* Not defined but normally LSB of previous character transmitted.

### 28.6.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writting the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

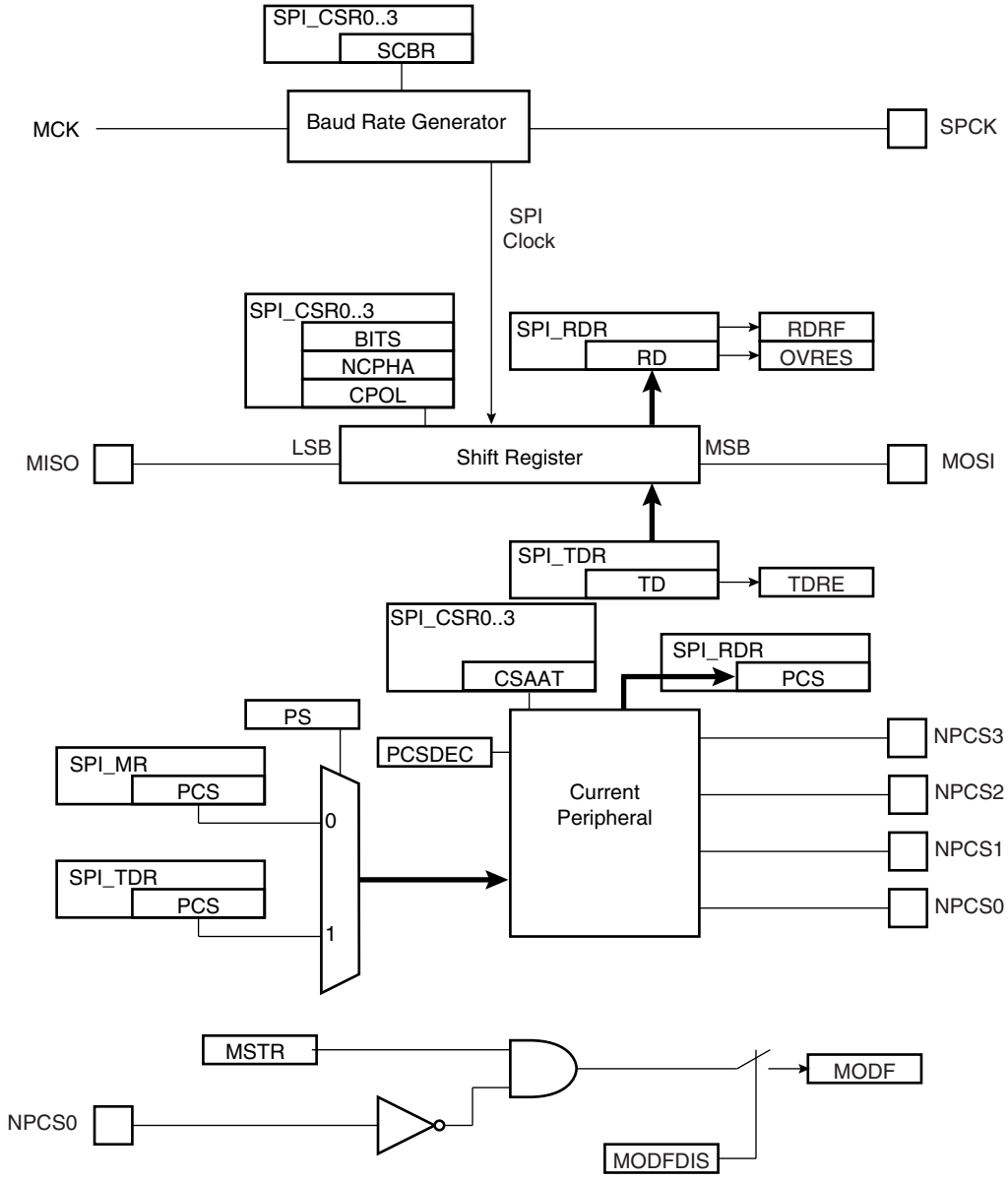
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 28-5 on page 299 shows a block diagram of the SPI when operating in Master Mode. Figure 28-6 on page 300 shows a flow chart describing how transfers are handled.

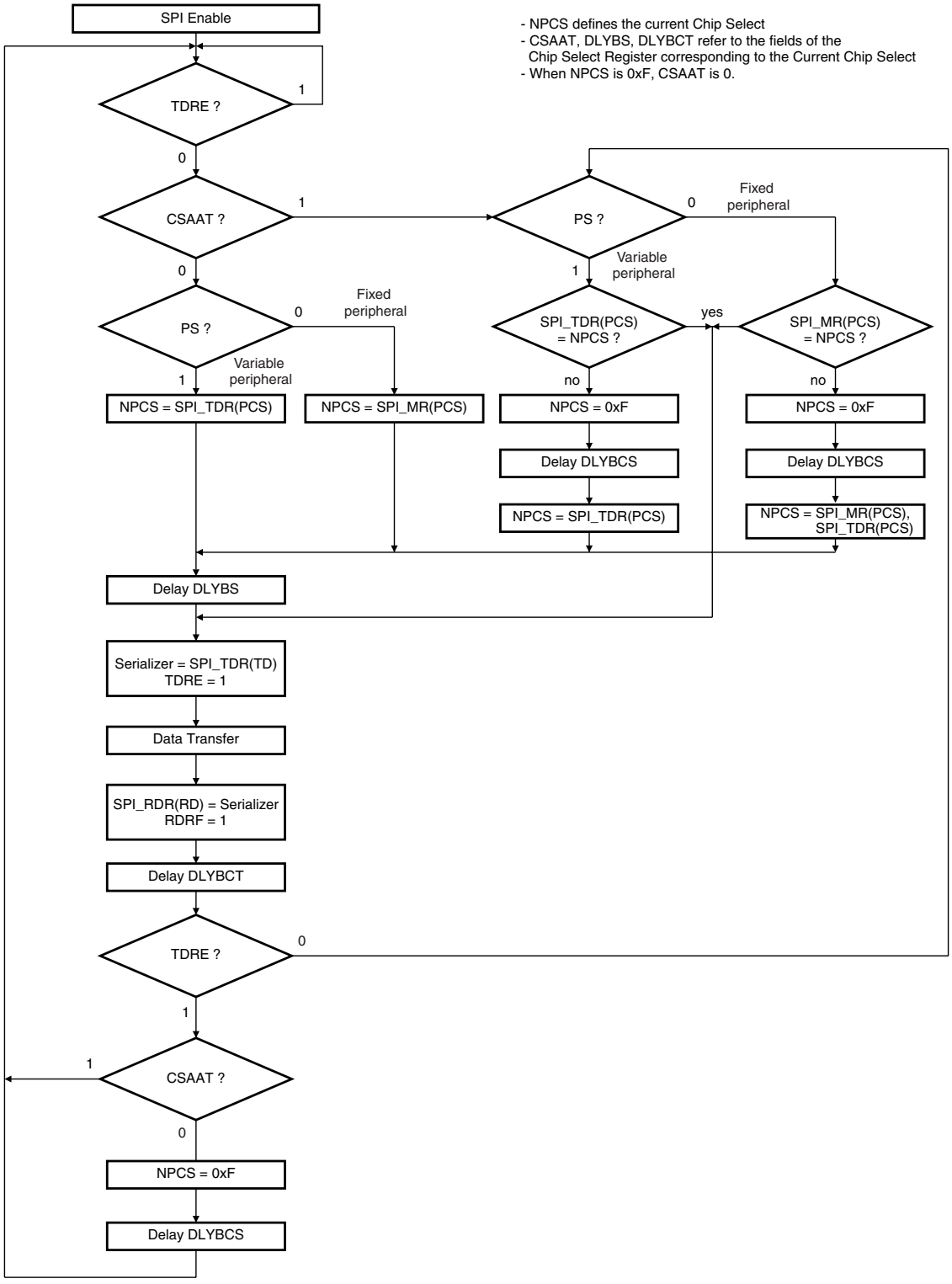
### 28.6.3.1 Master Mode Block Diagram

**Figure 28-5.** Master Mode Block Diagram



### 28.6.3.2 Master Mode Flow Diagram

**Figure 28-6.** Master Mode Flow Diagram



### 28.6.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK) , by a value between 1 and 255.

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

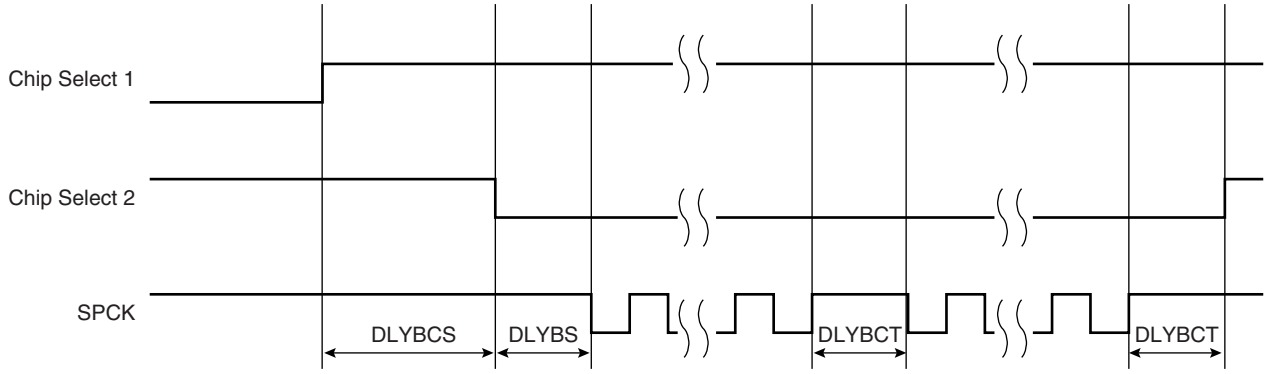
### 28.6.3.4 Transfer Delays

Figure 28-7 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

**Figure 28-7.** Programmable Delays




### 28.6.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral

- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_MR (Mode Register). In this case, the current peripheral is defined by the PCS field in SPI_MR and the PCS field in the SPI_TDR has no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in SPI_TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

#### *28.6.3.6 Peripheral Chip Select Decoding*

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

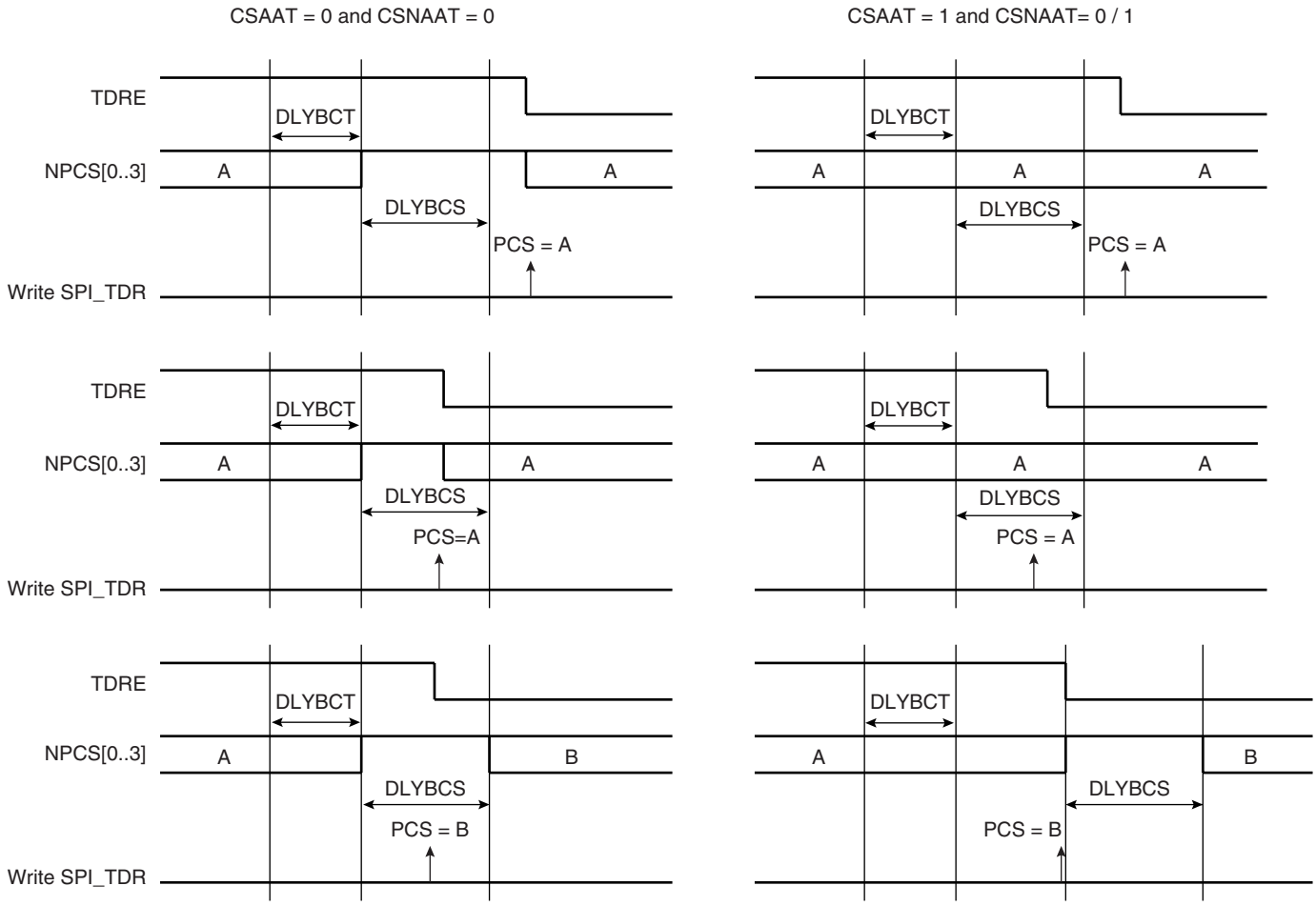
The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

#### *28.6.3.7 Peripheral Deselection*

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

**Figure 28-8.** Peripheral Deselection




*28.6.3.8 Mode Fault Detection*

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal. NPCS0, MOSI, MISO and SPCK must be configured in open drain through the PIO controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the SPI_CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (SPI_MR).

### 28.6.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the BITS field of the Chip Select Register 0 (SPI_CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the

SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

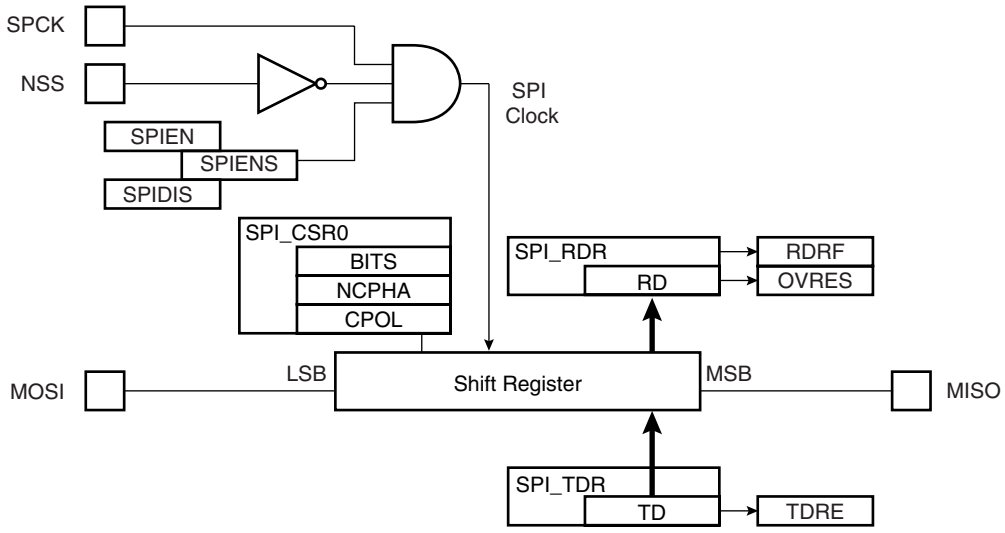
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in SPI_TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in SPI_TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in SPI_TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in SPI_TDR since the last load from SPI_TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 28-9 shows a block diagram of the SPI when operating in Slave Mode.

**Figure 28-9.** Slave Mode Functional Block Diagram



## 28.7 Serial Peripheral Interface (SPI) User Interface

**Table 28-3.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | SPI_CR | Write-only | --- |
| 0x04 | Mode Register | SPI_MR | Read-write | 0x0 |
| 0x08 | Receive Data Register | SPI_RDR | Read-only | 0x0 |
| 0x0C | Transmit Data Register | SPI_TDR | Write-only | --- |
| 0x10 | Status Register | SPI_SR | Read-only | 0x000000F0 |
| 0x14 | Interrupt Enable Register | SPI_IER | Write-only | --- |
| 0x18 | Interrupt Disable Register | SPI_IDR | Write-only | --- |
| 0x1C | Interrupt Mask Register | SPI_IMR | Read-only | 0x0 |
| 0x20 - 0x2C | Reserved | | | |
| 0x30 | Chip Select Register 0 | SPI_CSR0 | Read-write | 0x0 |
| 0x34 | Chip Select Register 1 | SPI_CSR1 | Read-write | 0x0 |
| 0x38 | Chip Select Register 2 | SPI_CSR2 | Read-write | 0x0 |
| 0x3C | Chip Select Register 3 | SPI_CSR3 | Read-write | 0x0 |
| 0x004C - 0x00F8 | Reserved | – | – | – |
| 0x100 - 0x124 | Reserved for the PDC | | | |

### 28.7.1 SPI Control Register

**Name:** SPI_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | LASTXFER |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWRST | – | – | – | – | – | SPIDIS | SPIEN |

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPIDIS is set, SPI finishes its tranfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

### 28.7.2 SPI Mode Register

**Name:** SPI_MR

**Access Type:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DLYBCS | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | PCS | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLB | – | – | MODFDIS | | PCSDEC | PS | MSTR |

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled (

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don't care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is less than or equal to six, six MCK periods will be inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

### 28.7.3 SPI Receive Data Register

**Name:** SPI_RDR

**Access Type:** Read-only

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

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

### 28.7.4 SPI Transmit Data Register

**Name:** SPI_TDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | LASTXFER |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | PCS | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TD | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TD | | | | | | | |

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

This field is only used if Variable Peripheral Select is active (PS = 1).

### 28.7.5 SPI Status Register

**Name:** SPI_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | SPIENS |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | - | TXEMPTY | NSSR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR(1) or SPI_RNCR(1) has a value other than 0.

1 = Both SPI_RCR(1) and SPI_RNCR(1) have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR[1] or SPI_TNCR[1] has a value other than 0.

1 = Both SPI_TCR[1] and SPI_TNCR[1] have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.

### 28.7.6 SPI Interrupt Enable Register

**Name:** SPI_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | - | TXEMPTY | NSSR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **NSSR: NSS Rising Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

- **TXEMPTY: Transmission Registers Empty Enable**

### 28.7.7 SPI Interrupt Disable Register

**Name:** SPI_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | - | TXEMPTY | NSSR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **NSSR: NSS Rising Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

- **TXEMPTY: Transmission Registers Empty Disable**

### 28.7.8 SPI Interrupt Mask Register

**Name:** SPI_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | - | TXEMPTY | NSSR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **NSSR: NSS Rising Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

- **TXEMPTY: Transmission Registers Empty Mask**

### 28.7.9 SPI Chip Select Register

**Name:** SPI_CSR0... SPI_CSR3

**Access Type:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DLYBCT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DLYBS | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SCBR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BITS | | | | CSAAT | – | NCPHA | CPOL |

- **CPOL: Clock Polarity**

0 = The inactive state value of SPCK is logic level zero.

1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSAAT: Chip Select Active After Transfer**

0 = The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1 = The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used.

| BITS | Bits Per Transfer |
|---|---|
| 0000 | 8 |
| 0001 | 9 |
| 0010 | 10 |
| 0011 | 11 |
| 0100 | 12 |
| 0101 | 13 |
| 0110 | 14 |
| 0111 | 15 |
| 1000 | 16 |
| 1001 | Reserved |
| 1010 | Reserved |
| 1011 | Reserved |

| BITS | Bits Per Transfer |
|---|---|
| 1100 | Reserved |
| 1101 | Reserved |
| 1110 | Reserved |
| 1111 | Reserved |

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

# 29. Two Wire Interface (TWI)

## 29.1 Overview

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I²C compatible device such as Real Time Clock (RTC), Dot Matrix/Graphic LCD Controllers and Temperature Sensor, to name but a few. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported. Arbitration of the bus is performed internally and puts the TWI in slave mode automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Below, Table 29-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I2C compatible device.

**Table 29-1.** Atmel TWI compatibility with i2C Standard

| I2C Standard | Atmel TWI |
|---|---|
| Standard Mode Speed (100 KHz) | Supported |
| Fast Mode Speed (400 KHz) | Supported |
| 7 or 10 bits Slave Addressing | Supported |
| START BYTE(1) | Not Supported |
| Repeated Start (Sr) Condition | Supported |
| ACK and NACK Management | Supported |
| Slope control and input filtering (Fast mode) | Not Supported |
| Clock stretching | Supported |

Note: 1. START + b000000001 + Ack + Sr

## 29.2 List of Abbreviations

**Table 29-2.** Abbreviations

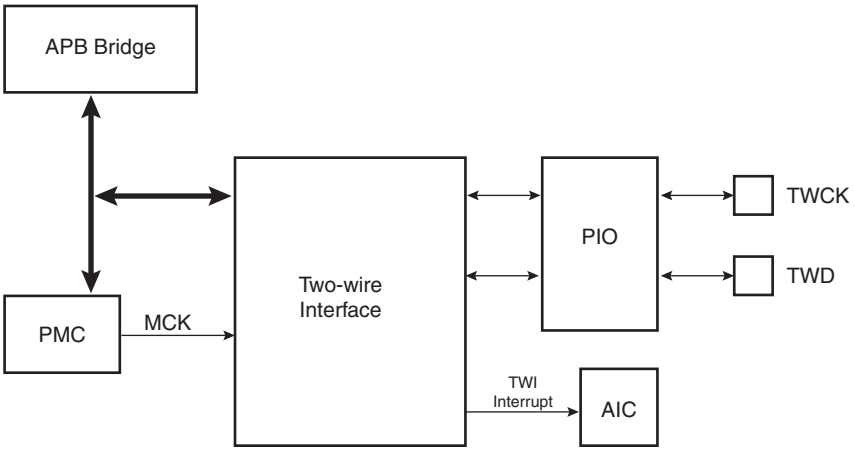
| Abbreviation | Description |
|---|---|
| TWI | Two-wire Interface |
| A | Acknowledge |
| NA | Non Acknowledge |
| P | Stop |
| S | Start |
| Sr | Repeated Start |
| SADR | Slave Address |

**Table 29-2.** Abbreviations

| Abbreviation | Description |
|---|---|
| ADR | Any address except SADR |
| R | Read |
| W | Write |

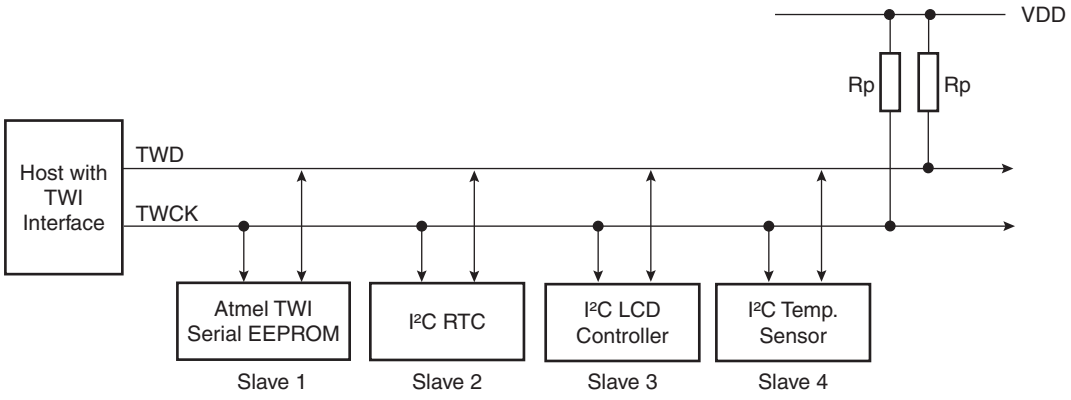
## 29.3 Block Diagram

**Figure 29-1.** Block Diagram




## 29.4 Application Block Diagram

**Figure 29-2.** Application Block Diagram




Rp: Pull up value as given by the I²C Standard

### 29.4.1 I/O Lines Description

**Table 29-3.** I/O Lines Description

| Pin Name | Pin Description | Type |
|---|---|---|
| TWD | Two-wire Serial Data | Input/Output |
| TWCK | Two-wire Serial Clock | Input/Output |

## 29.5 Product Dependencies

### 29.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 29-2 on page 320). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following step:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.

### 29.5.2 Power Management

- Enable the peripheral clock.

The TWI interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TWI clock.

### 29.5.3 Interrupt

The TWI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). In order to handle interrupts, the AIC must be programmed before configuring the TWI.

## 29.6 Functional Description

### 29.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 29-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 29-3).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

**Figure 29-3.** START and STOP Conditions

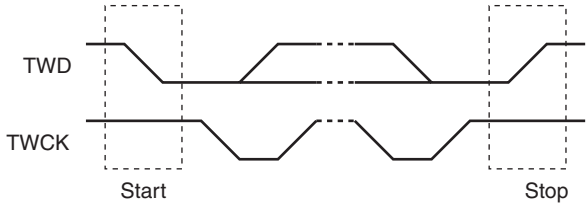
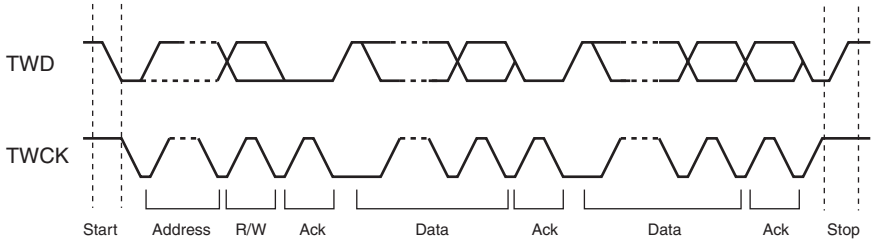



**Figure 29-4.** Transfer Format




### 29.6.2 Modes of Operation

The TWI has six modes of operation:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following chapters.

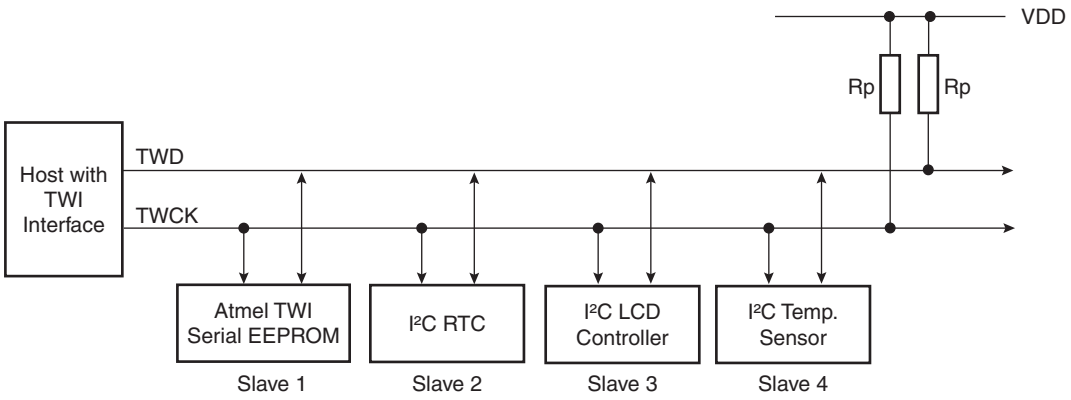
## 29.7 Master Mode

### 29.7.1 Definition

The Master is the device that starts a transfer, generates a clock and stops it.

### 29.7.2 Application Block Diagram

**Figure 29-5.** Master Mode Typical Application Block Diagram




Rp: Pull up value as given by the I²C Standard

### 29.7.3 Programming Master Mode

The following registers have to be programmed before entering Master mode:

1. DADR (+ IADRSZ + IADR if a 10 bit device is addressed): The device address is used to access slave devices in read or write mode.
2. CKDIV + CHDIV + CLDIV: Clock Waveform.
3. SVDIS: Disable the slave mode.
4. MSEN: Enable the master mode.

### 29.7.4 Master Transmitter Mode

After the master initiates a Start condition when writing into the Transmit Holding Register, TWI_THR, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWI_MMR).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the Not Acknowledge bit (**NACK)** in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI_IER). If the slave acknowledges the byte, the data written in the TWI_THR, is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR.

When no more data is written into the TWI_THR, the master generates a stop condition to end the transfer. The end of the complete transfer is marked by the TWI_TXCOMP bit set to one. See Figure 29-6, Figure 29-7, and Figure 29-8.

TXRDY is used as Transmit Ready for the PDC transmit channel.

**Figure 29-6.** Master Write with One Data Byte

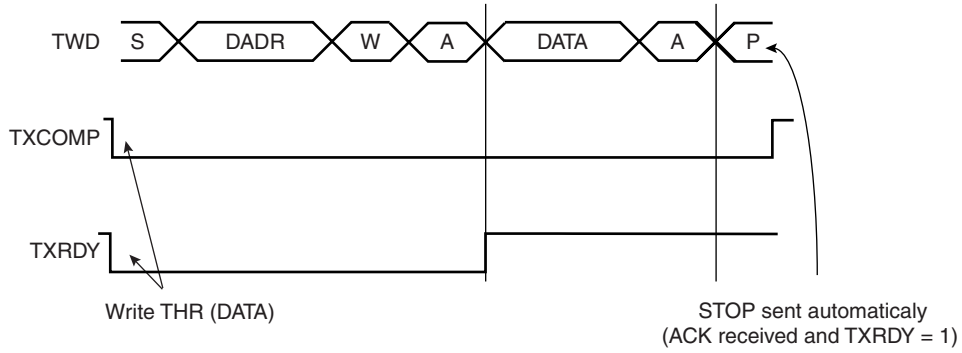



**Figure 29-7.** Master Write with Multiple Data Byte

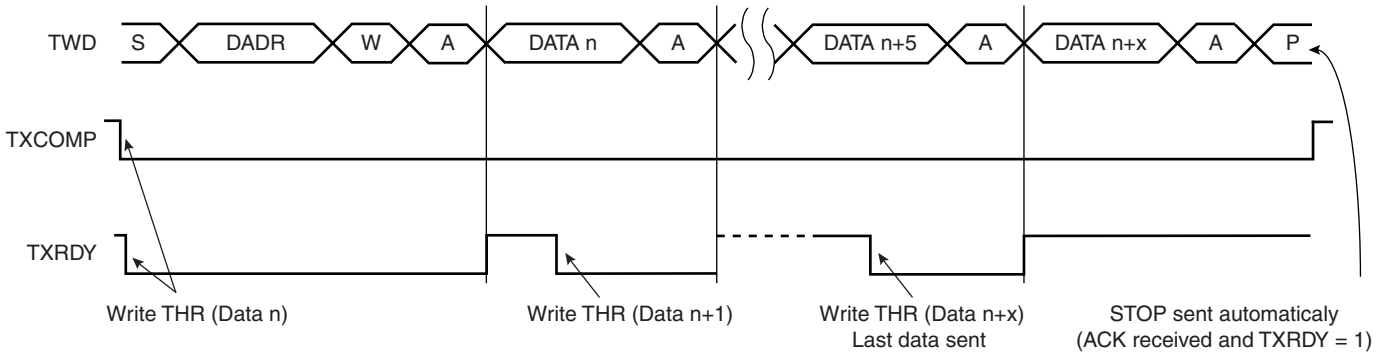
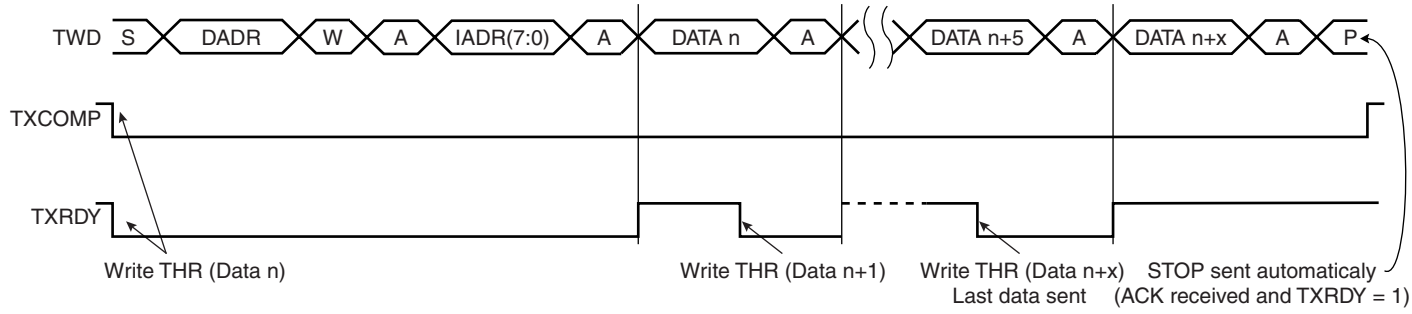



**Figure 29-8.** Master Write with One Byte Internal Address and Multiple Data Bytes




### 29.7.5 Master Receiver Mode

The read sequence begins by setting the START bit. After the start condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWI_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the **NACK** bit in the status register if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data, after the stop condition. See Figure 29-9. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

When a single data byte read is performed, with or without internal address **(IADR)**, the START and STOP bits must be set at the same time. See Figure 29-9. When a multiple data byte read is performed, with or without internal address **(IADR)**, the STOP bit must be set after the next-to-last data received. See Figure 29-10. For Internal Address usage see Section 29.7.6.

**Figure 29-9.** Master Read with One Data Byte

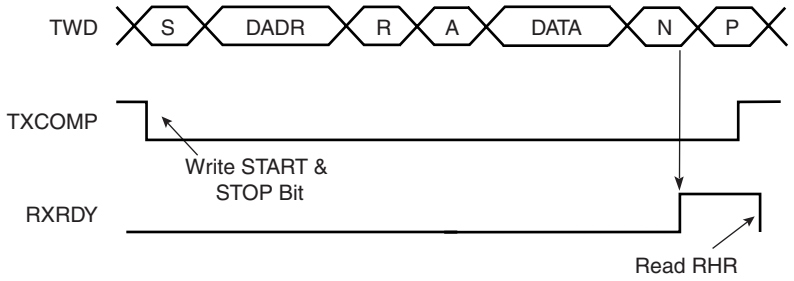
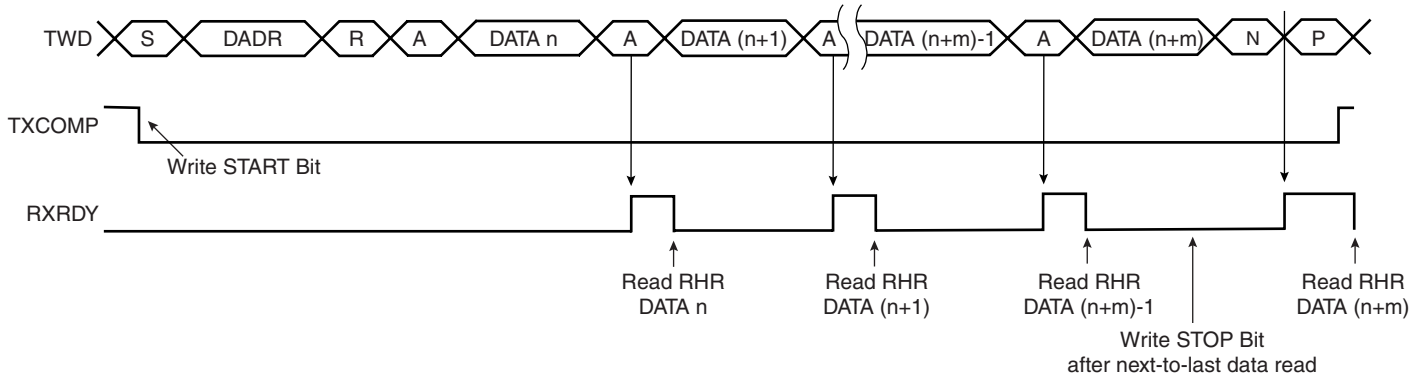



**Figure 29-10.** Master Read with Multiple Data Bytes




RXRDY is used as Receive Ready for the PDC receive channel.

### 29.7.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

#### *29.7.6.1 7-bit Slave Addressing*

When Addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, within a memory page location in a serial memory, for example. When performing read operations with an internal address, the TWI performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second start condition (after sending the IADR) is

sometimes called “repeated start” (Sr) in I2C fully-compatible devices. See Figure 29-12. See Figure 29-11 and Figure 29-13 for Master Write operation with internal address.

The three internal address bytes are configurable through the Master Mode register (TWI_MMR).

If the slave device supports only a 7-bit address, i.e. no internal address, **IADRSZ** must be set to 0.

In the figures below the following abbreviations are used:

- S Start
- Sr Repeated Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

**Figure 29-11.** Master Write with One, Two or Three Bytes Internal Address and One Data Byte

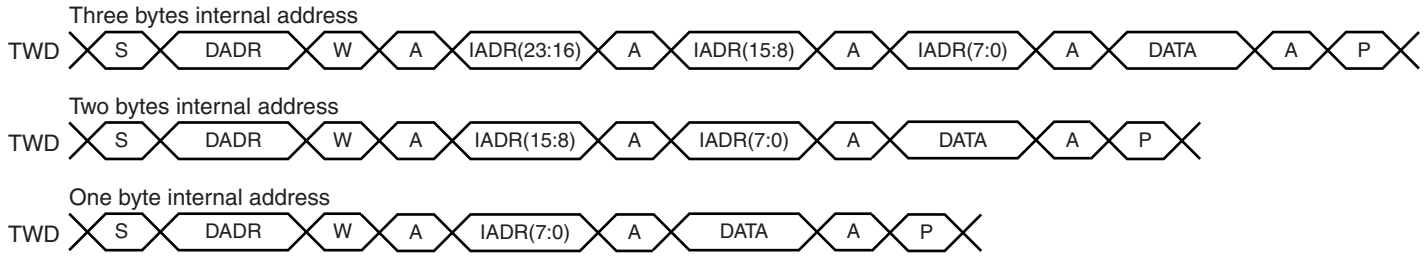
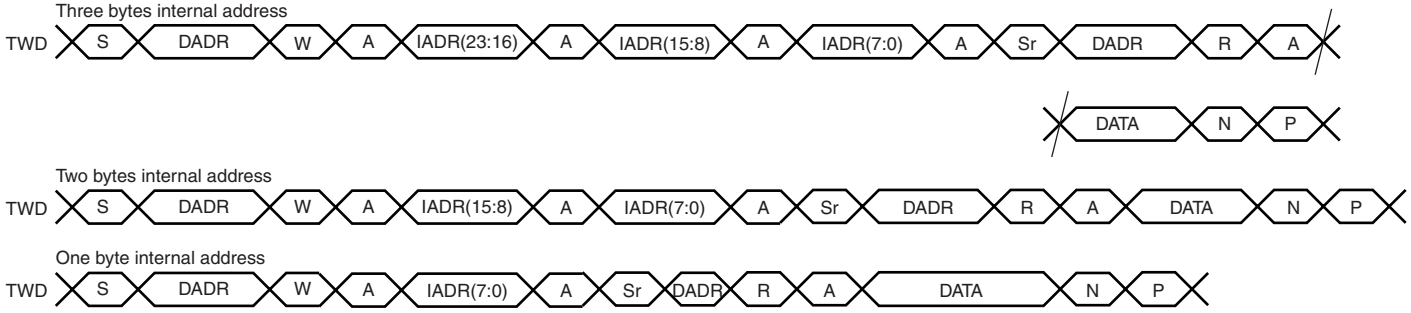



**Figure 29-12.** Master Read with One, Two or Three Bytes Internal Address and One Data Byte




### *29.7.6.2 10-bit Slave Addressing*

For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ)** and set the other slave address bits in the internal address register (TWI_IADR). The two remaining

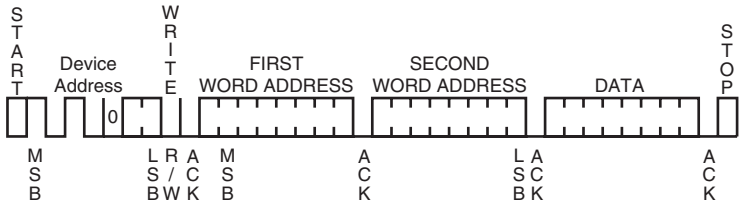
Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

**Example:** Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 29-13 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

**Figure 29-13.** Internal Address Usage



### 29.7.7 Using the Peripheral DMA Controller (PDC)

The use of the PDC significantly reduces the CPU load.

To assure correct implementation, respect the following programming sequences:

#### *29.7.7.1 Data Transmit with the PDC*

1. Initialize the transmit PDC (memory pointers, size, etc.).
2. Configure the master mode (DADR, CKDIV, etc.).
3. Start the transfer by setting the PDC TXTEN bit.
4. Wait for the PDC end TX flag.
5. Disable the PDC by setting the PDC TXDIS bit.

#### *29.7.7.2 Data Receive with the PDC*

1. Initialize the receive PDC (memory pointers, size - 1, etc.).
2. Configure the master mode (DADR, CKDIV, etc.).
3. Start the transfer by setting the PDC RXTEN bit.
4. Wait for the PDC end RX flag.
5. Disable the PDC by setting the PDC RXDIS bit.

### 29.7.8 Read-write Flowcharts

The following flowcharts shown in Figure 29-15 on page 330, Figure 29-16 on page 331, Figure 29-17 on page 332, Figure 29-18 on page 333 and Figure 29-19 on page 334 give examples for read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

**Figure 29-14.** TWI Write Operation with Single Data Byte without Internal Address

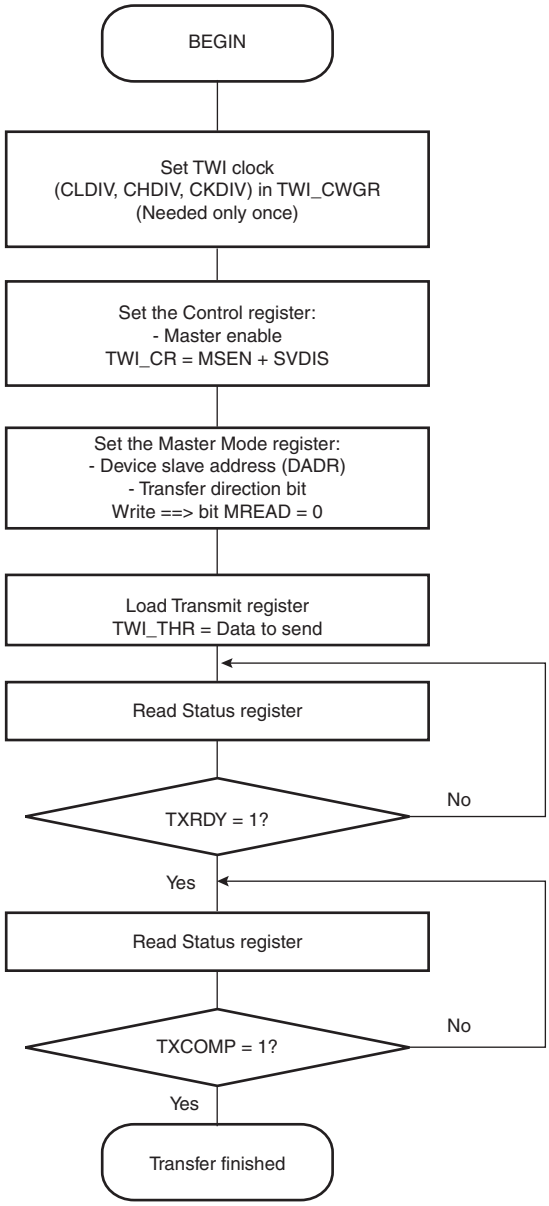


**Figure 29-15.** TWI Write Operation with Single Data Byte and Internal Address

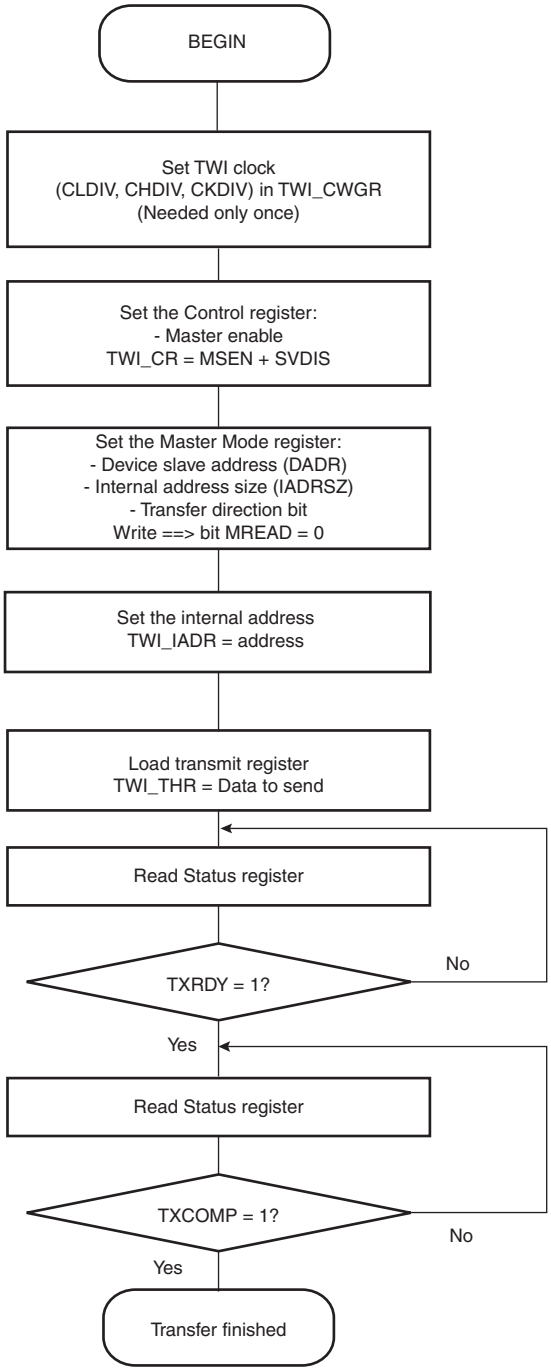


**Figure 29-16.** TWI Write Operation with Multiple Data Bytes with or without Internal Address

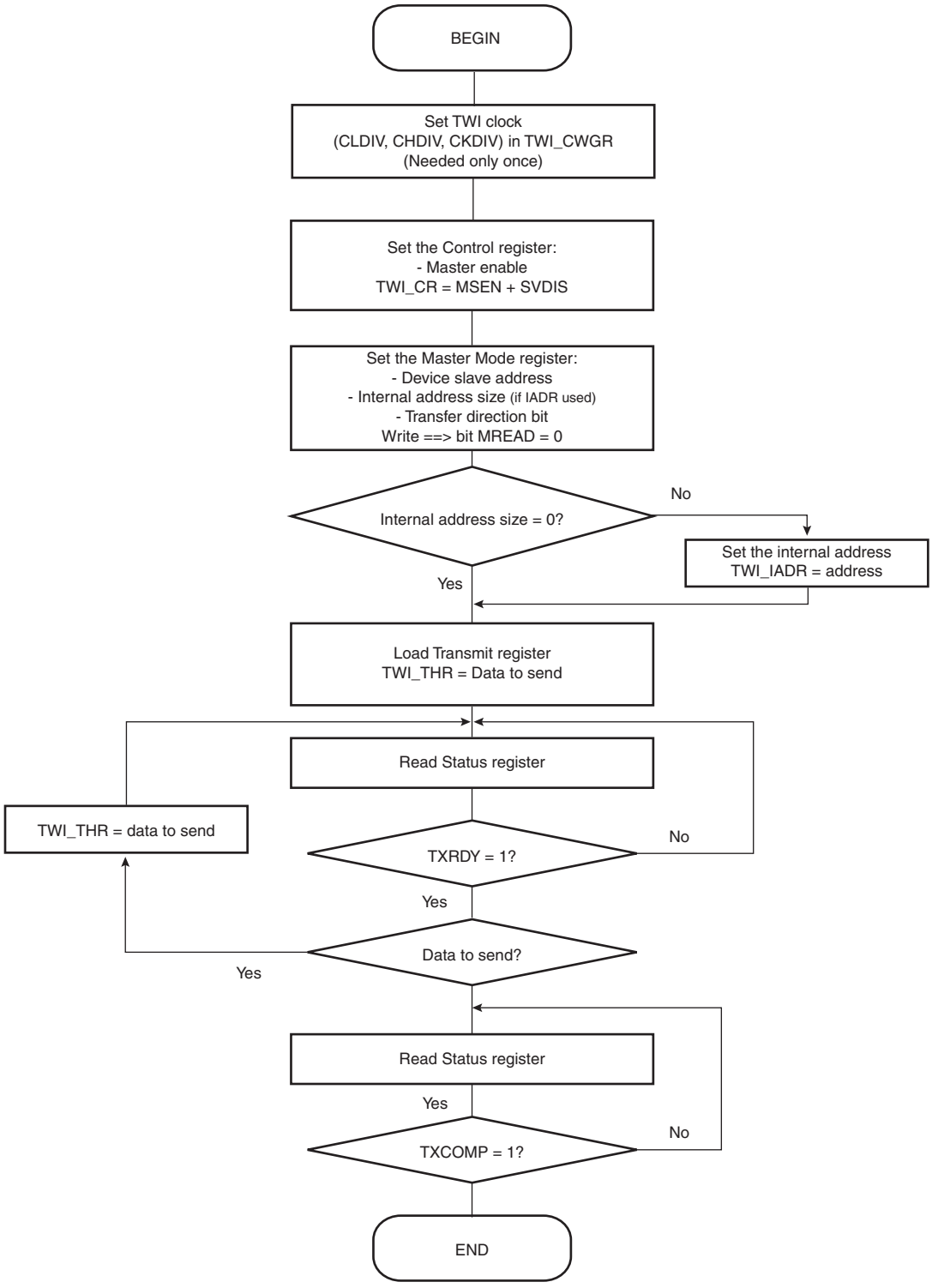


**Figure 29-17.** TWI Read Operation with Single Data Byte without Internal Address

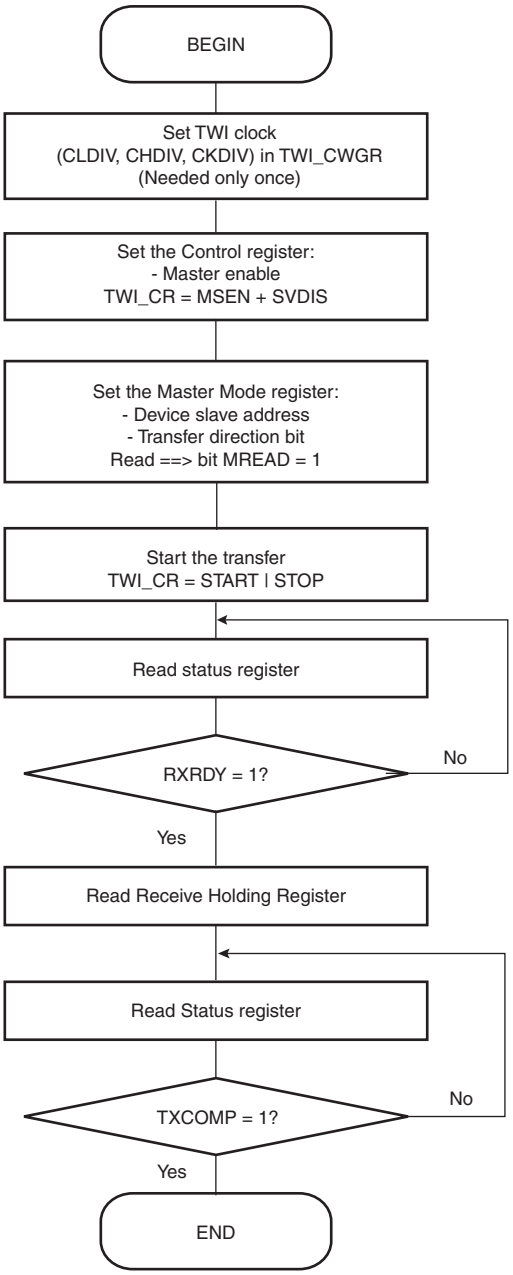


**Figure 29-18.** TWI Read Operation with Single Data Byte and Internal Address

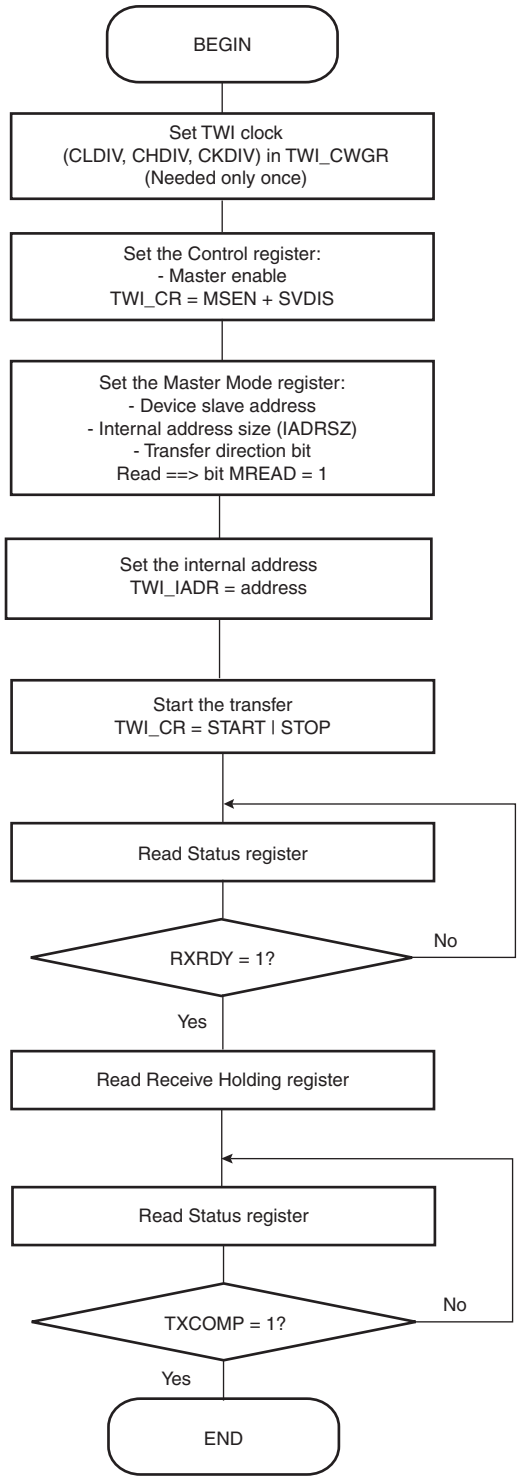
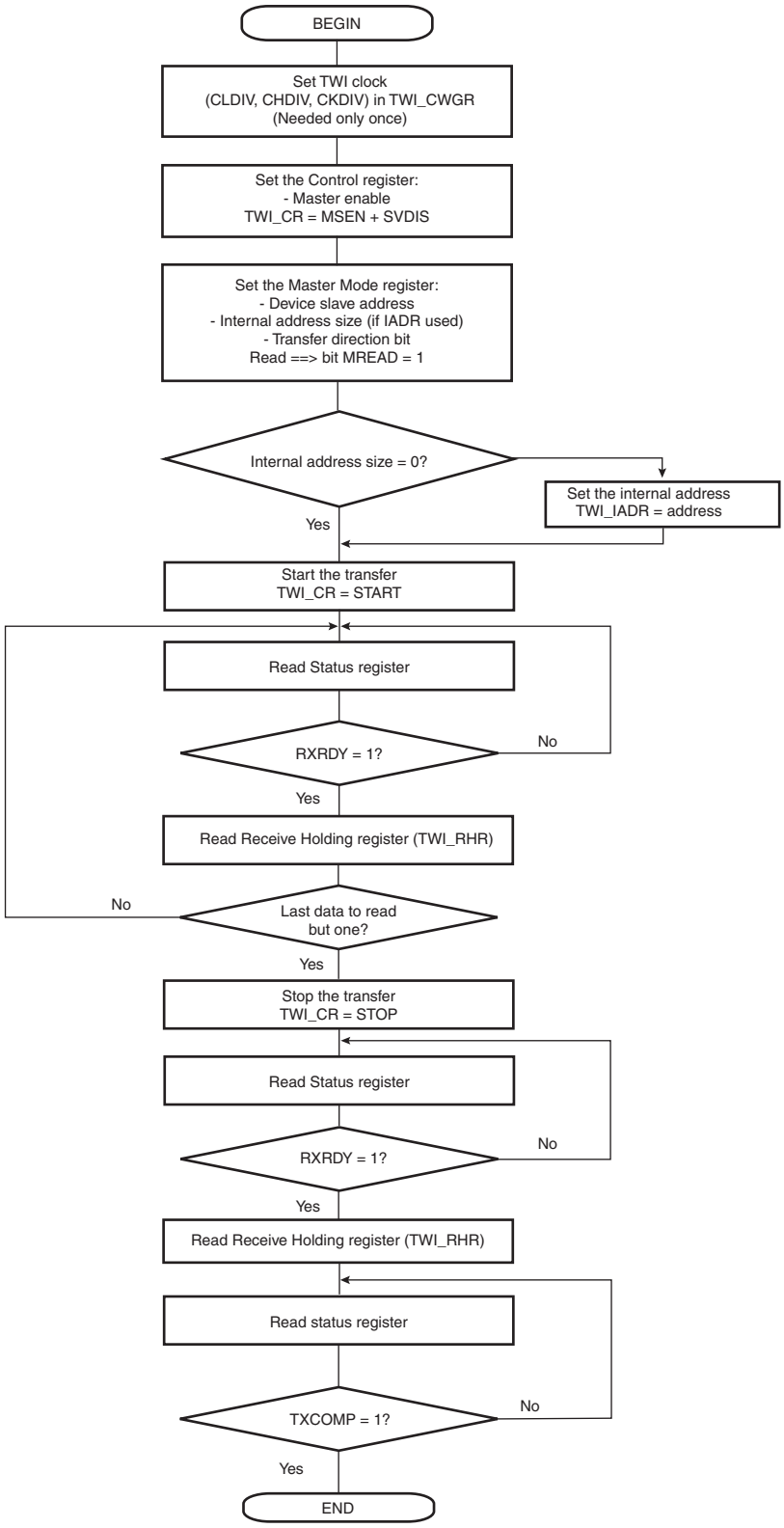


**Figure 29-19.** TWI Read Operation with Multiple Data Bytes with or without Internal Address



## 29.8 Multi-master Mode

### 29.8.1 Definition

More than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master who has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Figure 29-21 on page 336.

### 29.8.2 Different Multi-master Modes

Two multi-master modes may be distinguished:

1. TWI is considered as a Master only and will never be addressed.
2. TWI may be either a Master or a Slave and may be addressed.

Note: In both Multi-master modes arbitration is supported.

#### *29.8.2.1 TWI as Master Only*

In this mode, TWI is considered as a Master only (MSEN is always at one) and must be driven like a Master with the ARBLST (ARBitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the programmer must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 29-20 on page 336).

Note: The state of the bus (busy or free) is not indicated in the user interface.

#### *29.8.2.2 TWI as Master or Slave*

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

Then, in the case where TWI may be either a Master or a Slave, the programmer must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform Slave Access (if TWI is addressed).
2. If TWI has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, TWI scans the bus in order to detect if it is busy or free. When the bus is considered as free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in the case where the Master that won the arbitration wanted to access the TWI.
7. If TWI has to be set in Slave mode, wait until TXCOMP flag is at 1 and then program the Slave mode.

Note: In the case where the arbitration is lost and TWI is addressed, TWI will not acknowledge even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then, the Master must repeat SADR.

**Figure 29-20.** Programmer Sends Data While the Bus is Busy

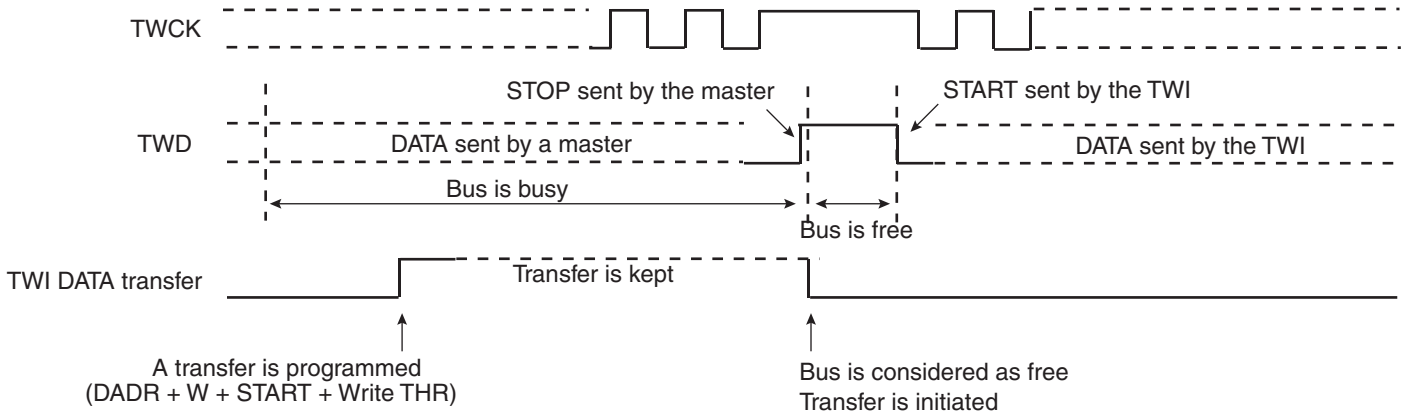
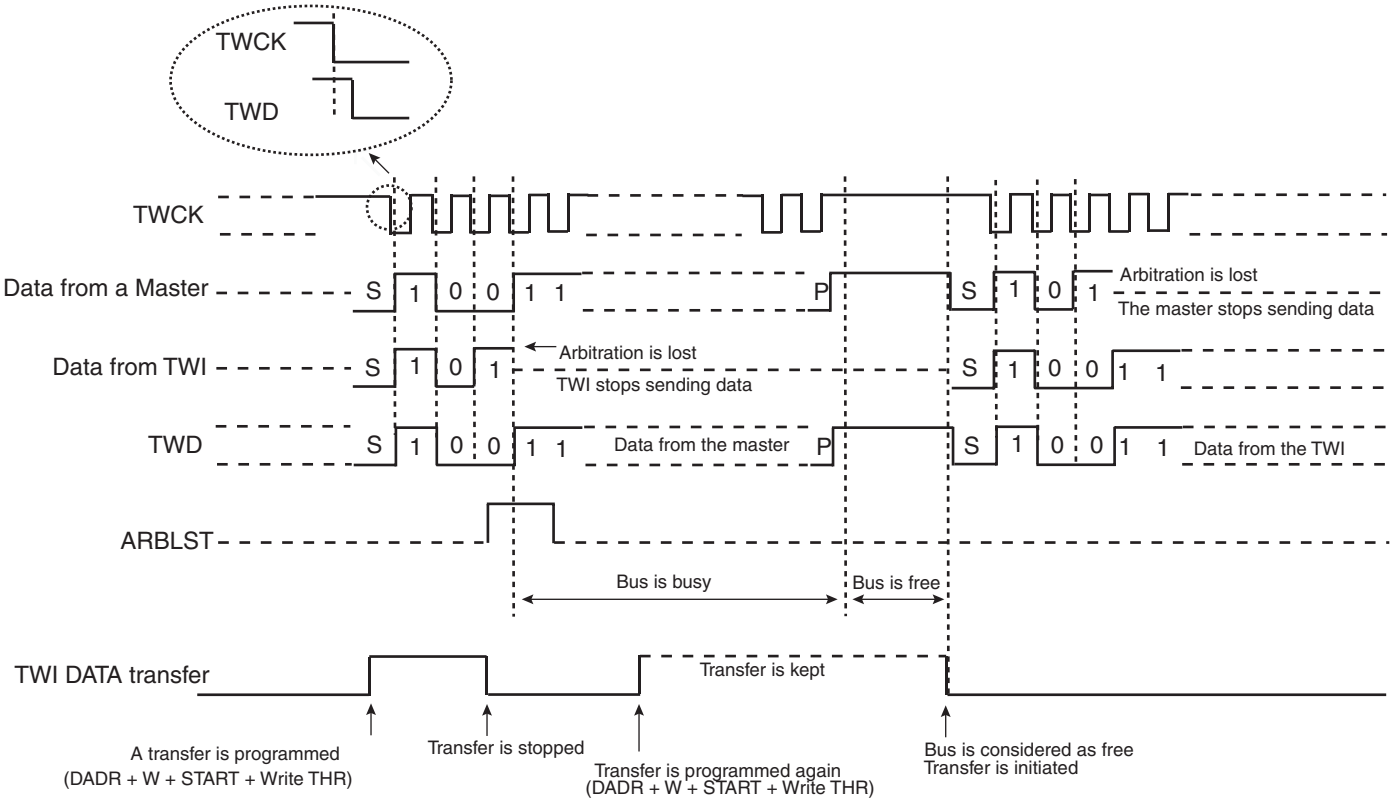
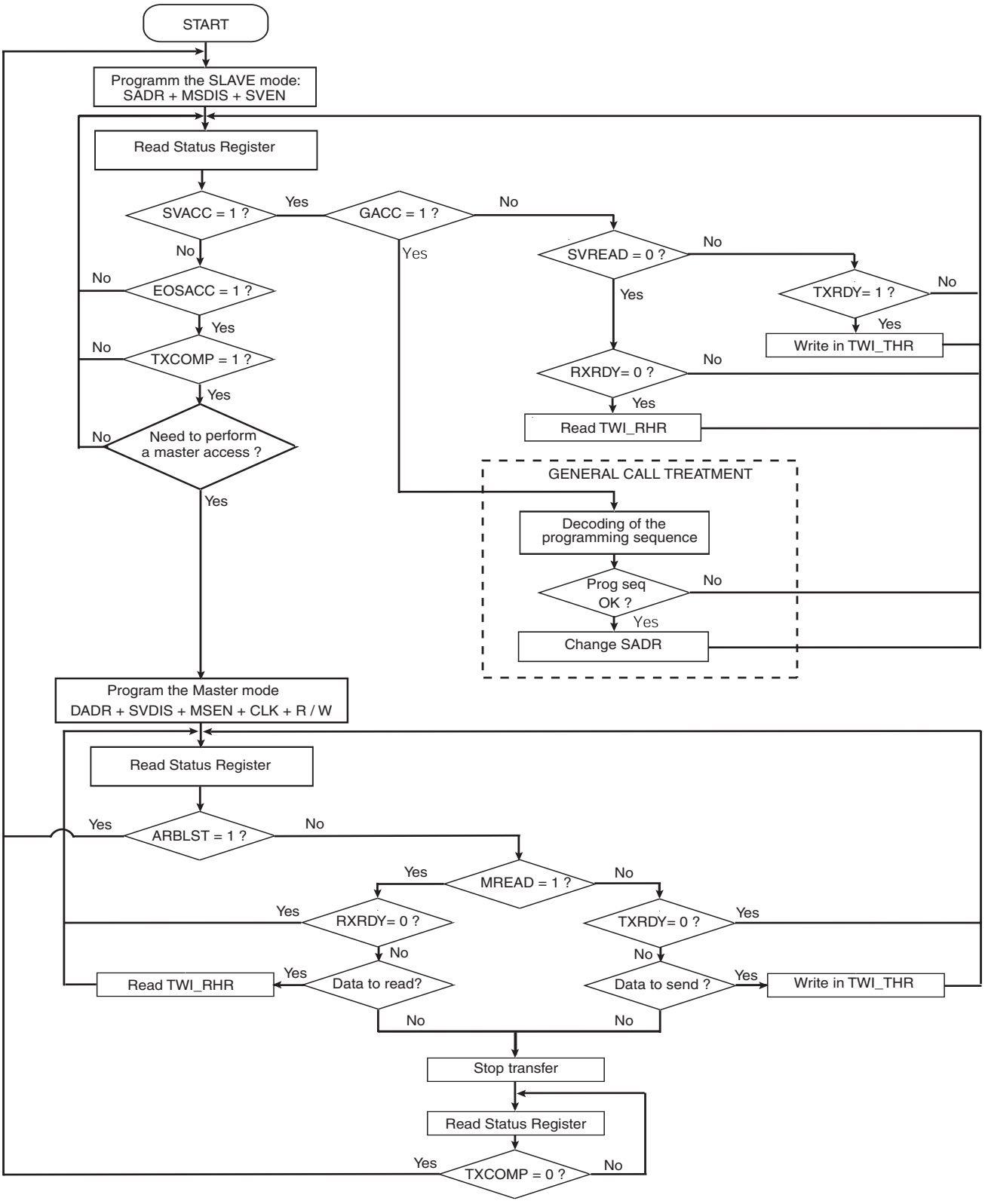



**Figure 29-21.** Arbitration Cases




The flowchart shown in Figure 29-22 on page 337 gives an example of read and write operations in Multi-master mode.

**Figure 29-22.** Multi-master Flowchart



## 29.9 Slave Mode

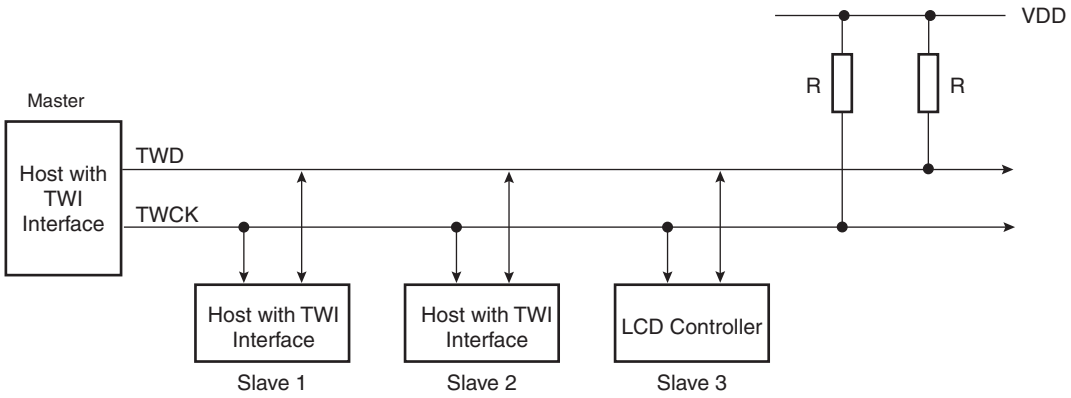
### 29.9.1 Definition

The Slave Mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED_START and STOP conditions are always provided by the master).

### 29.9.2 Application Block Diagram

**Figure 29-23.** Slave Mode Typical Application Block Diagram




### 29.9.3 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. SADR (TWI_SMR): The slave device address is used in order to be accessed by master devices in read or write mode.
2. MSDIS (TWI_CR): Disable the master mode.
3. SVEN (TWI_CR): Enable the slave mode.

As the device receives the clock, values written in TWI_CWGR are not taken into account.

### 29.9.4 Receiving Data

After a Start or Repeated Start condition is detected and if the address sent by the Master matches with the Slave address programmed in the SADR (Slave ADdress) field, SVACC (Slave ACCess) flag is set and SVREAD (Slave READ) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a repeated START is detected. When such a condition is detected, EOSACC (End Of Slave ACCess) flag is set.

#### *29.9.4.1 Read Sequence*

In the case of a Read sequence (SVREAD is high), TWI transfers data written in the TWI_THR (TWI Transmit Holding Register) until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in the TWI_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the shift register is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a repeated START always follows a NACK.

See Figure 29-24 on page 340.

#### 29.9.4.2 *Write Sequence*

In the case of a Write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI_RHR.

TWI continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See Figure 29-25 on page 340.

#### 29.9.4.3 *Clock Synchronization Sequence*

In the case where TWI_THR or TWI_RHR is not written/read in time, TWI performs a clock synchronization.

Clock stretching information is given by the SCLWS (Clock Wait state) bit.

See Figure 29-27 on page 342 and Figure 29-28 on page 343.

#### 29.9.4.4 *General Call*

In the case where a GENERAL CALL is performed, GACC (General Call ACCess) flag is set.

After GACC is set, it is up to the programmer to interpret the meaning of the GENERAL CALL and to decode the new address programming sequence.

See Figure 29-26 on page 341.

#### 29.9.4.5 *PDC*

As it is impossible to know the exact number of data to receive/send, the use of PDC is NOT recommended in SLAVE mode.

### 29.9.5 Data Transfer

#### 29.9.5.1 *Read Operation*

The read mode is defined as a data requirement from the master.

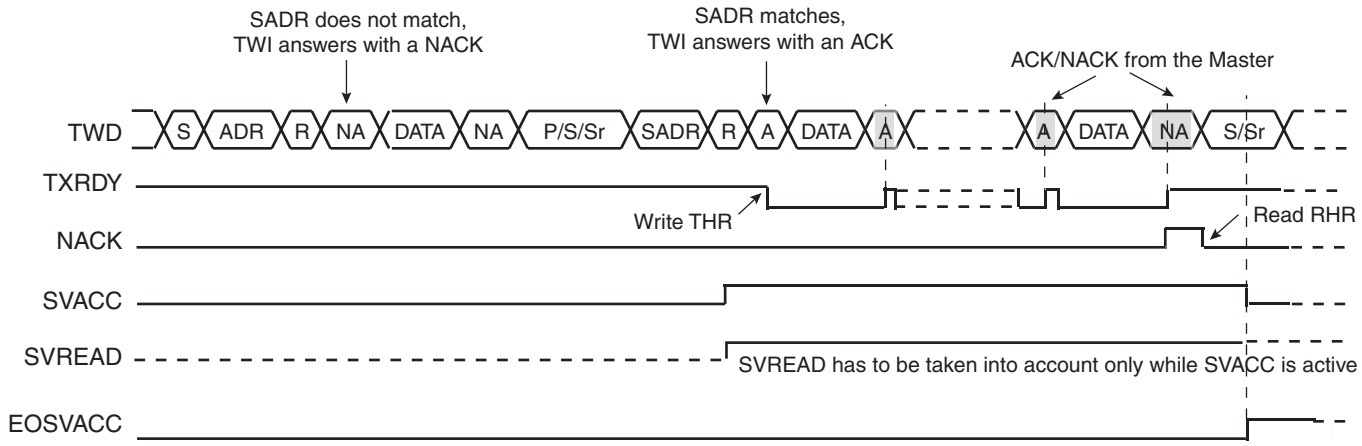
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 29-24 on page 340 describes the write operation.

**Figure 29-24.** Read Access Ordered by a MASTER




Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. TXRDY is reset when data has been transmitted from TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.

### *29.9.5.2 Write Operation*

The write mode is defined as a data transmission from the master.

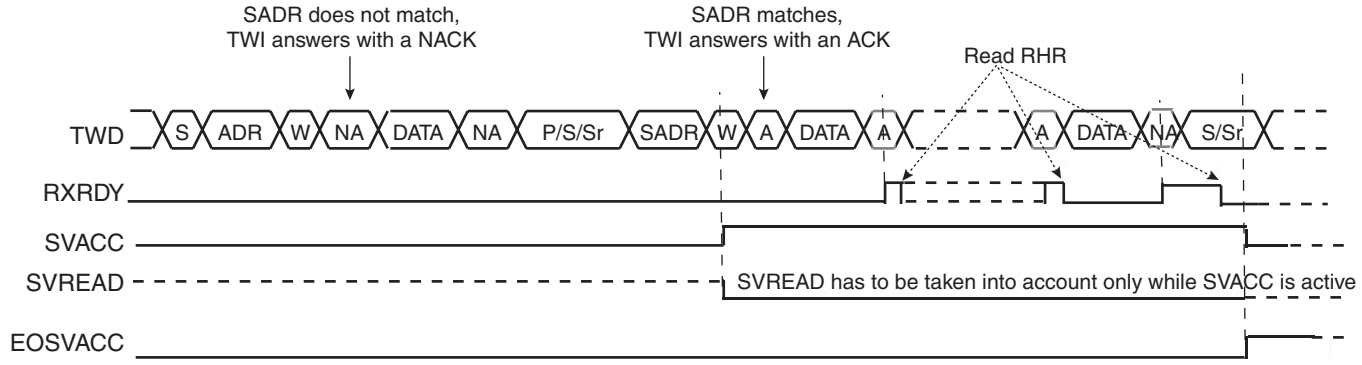
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 29-25 on page 340 describes the Write operation.

**Figure 29-25.** Write Access Ordered by a Master




Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. RXRDY is set when data has been transmitted from the shift register to the TWI_RHR and reset when this data is read.

### 29.9.5.3 General Call

The general call is performed in order to change the address of the slave.

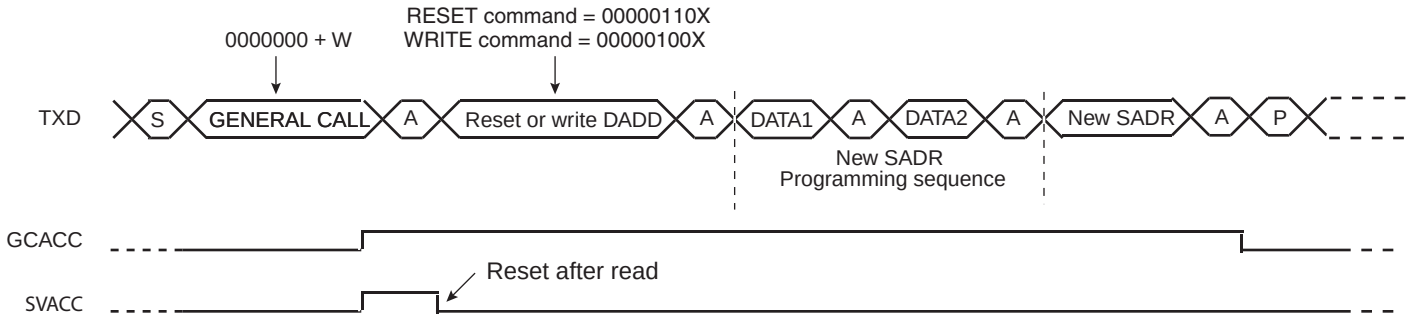
If a GENERAL CALL is detected, GACC is set.

After the detection of General Call, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 29-26 on page 341 describes the General Call access.

**Figure 29-26.** Master Performs a General Call




Note: This method allows the user to create an own programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

### 29.9.5.4 Clock Synchronization

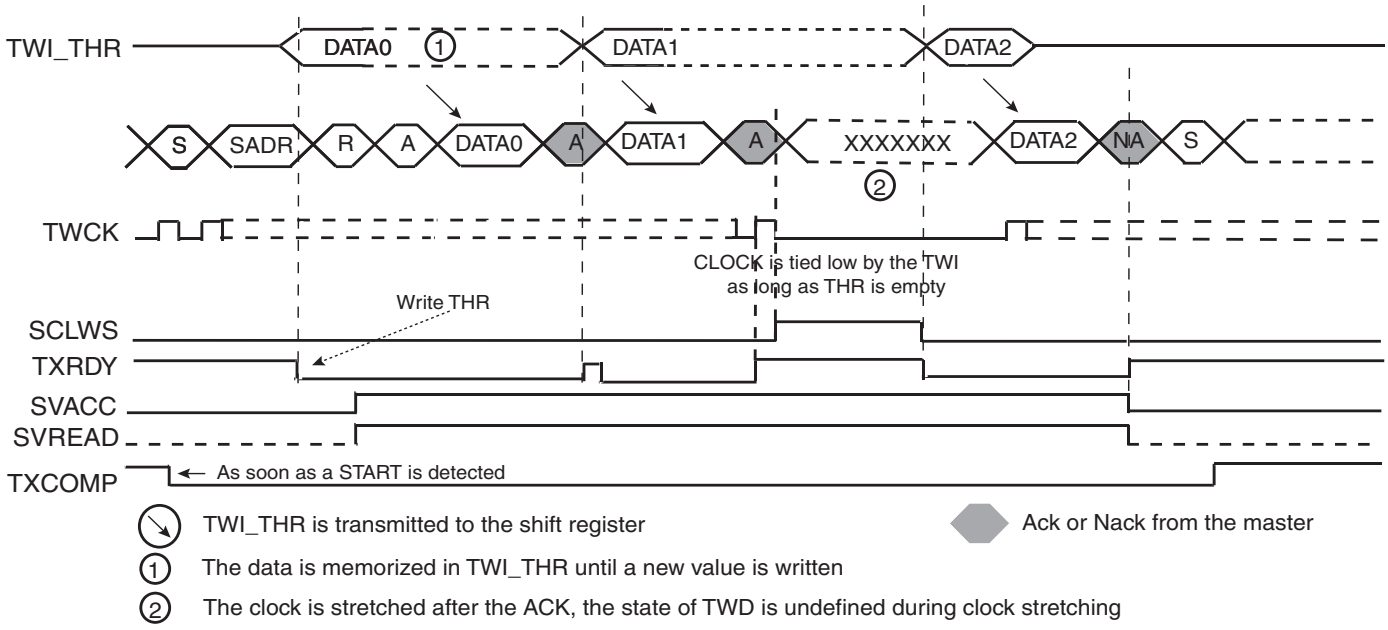
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

### 29.9.5.5 Clock Synchronization in Read Mode

The clock is tied low if the shift register is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the shift register is loaded.

Figure 29-27 on page 342 describes the clock synchronization in Read mode.

**Figure 29-27.** Clock Synchronization in Read Mode



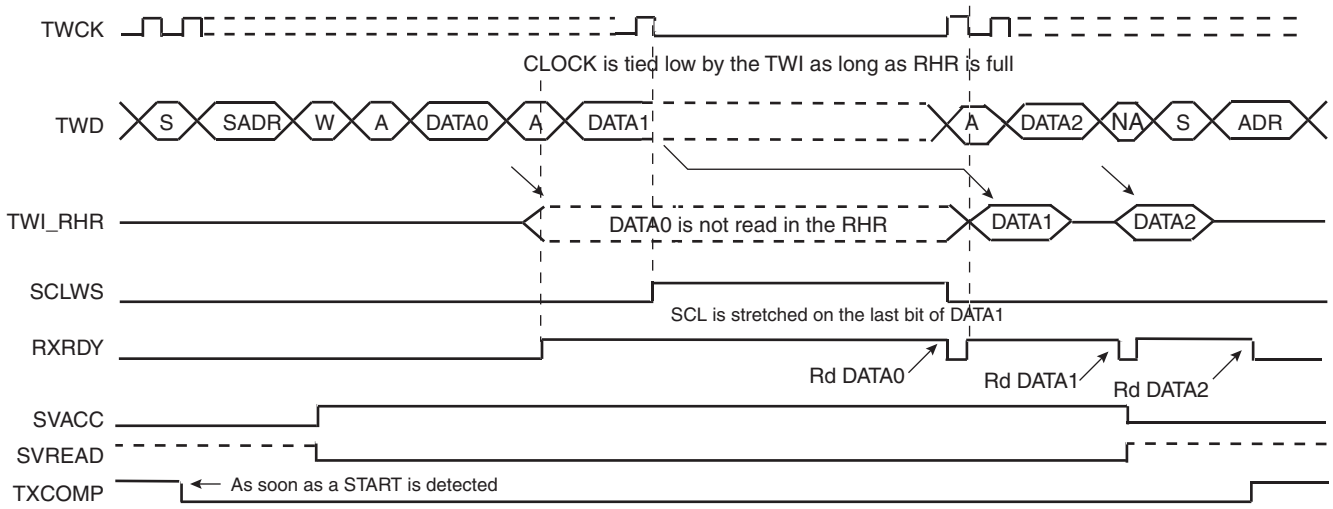

Notes:
1. TXRDY is reset when data has been written in the TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

### 29.9.5.6 Clock Synchronization in Write Mode

The clock is tied low if the shift register and the TWI_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWI_RHR is read.

Figure 29-28 on page 343 describes the clock synchronization in Read mode.

**Figure 29-28.** Clock Synchronization in Write Mode




Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

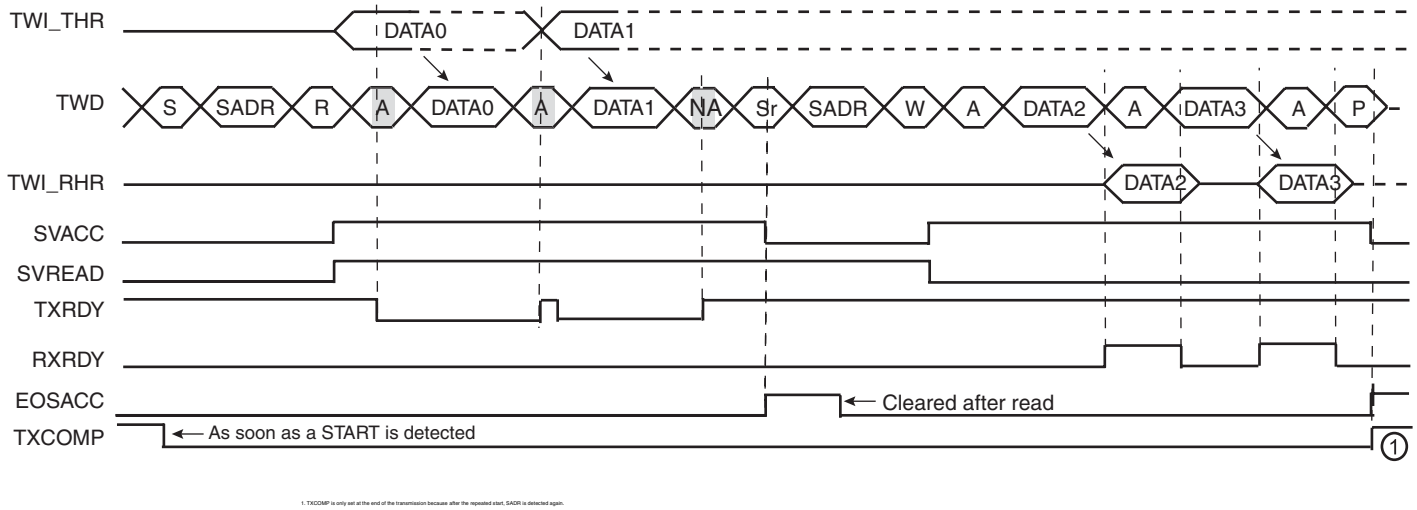
### 29.9.5.7 Reversal after a Repeated Start

### 29.9.5.8 Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 29-29 on page 344 describes the repeated start + reversal from Read to Write mode.

**Figure 29-29.** Repeated Start + Reversal from Read to Write Mode

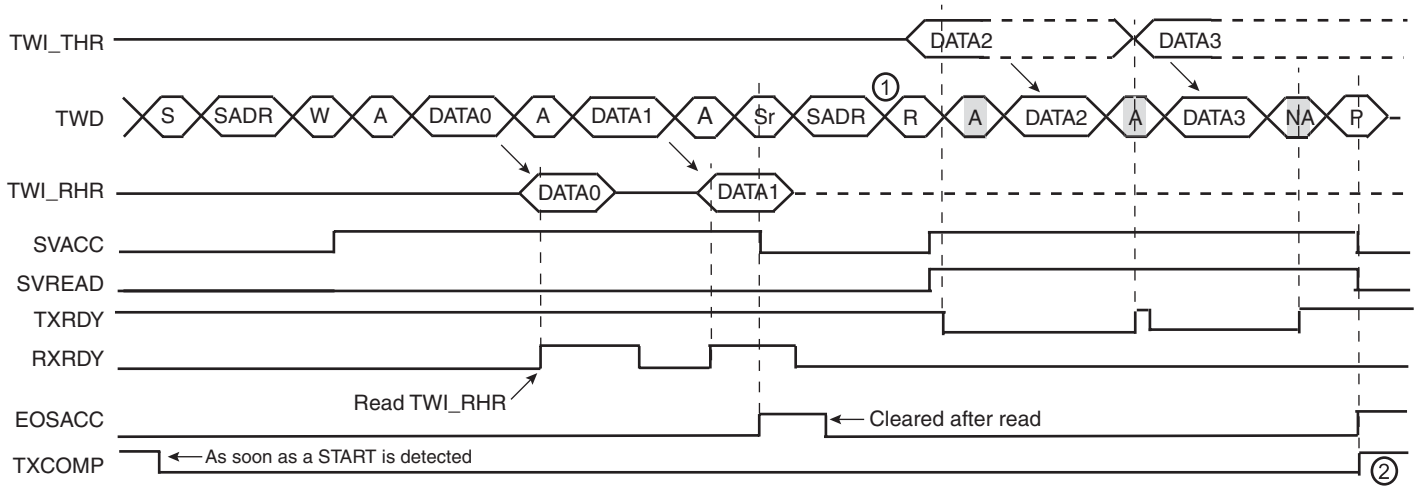



1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

### 29.9.5.9 Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.Figure 29-30 on page 344 describes the repeated start + reversal from Write to Read mode.

**Figure 29-30.** Repeated Start + Reversal from Write to Read Mode



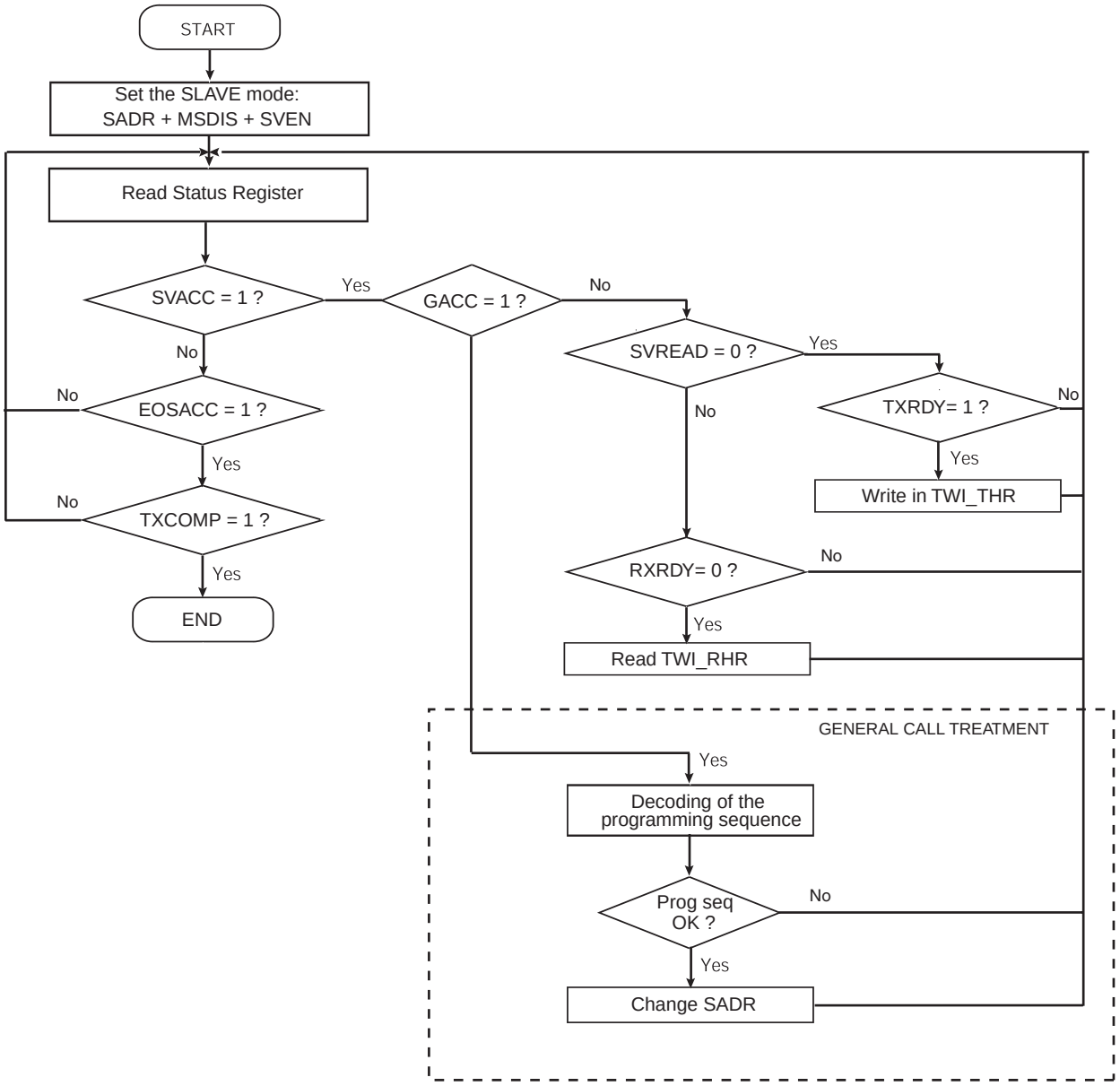

Notes: 1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.

2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

### 29.9.6 Read Write Flowcharts

The flowchart shown in Figure 29-31 on page 345 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

**Figure 29-31.** Read Write Flowchart in Slave Mode



## 29.10 Two-wire Interface (TWI) User Interface

**Table 29-4.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | TWI_CR | Write-only | N / A |
| 0x04 | Master Mode Register | TWI_MMR | Read-write | 0x00000000 |
| 0x08 | Slave Mode Register | TWI_SMR | Read-write | 0x00000000 |
| 0x0C | Internal Address Register | TWI_IADR | Read-write | 0x00000000 |
| 0x10 | Clock Waveform Generator Register | TWI_CWGR | Read-write | 0x00000000 |
| 0x20 | Status Register | TWI_SR | Read-only | 0x0000F009 |
| 0x24 | Interrupt Enable Register | TWI_IER | Write-only | N / A |
| 0x28 | Interrupt Disable Register | TWI_IDR | Write-only | N / A |
| 0x2C | Interrupt Mask Register | TWI_IMR | Read-only | 0x00000000 |
| 0x30 | Receive Holding Register | TWI_RHR | Read-only | 0x00000000 |
| 0x34 | Transmit Holding Register | TWI_THR | Write-only | 0x00000000 |
| 0x38 - 0xFC | Reserved | – | – | – |
| 0x100 - 0x124 | Reserved for the PDC | – | – | – |

### 29.10.1 TWI Control Register

Name: TWI_CR

Access: Write-only

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWRST | – | SVDIS | SVEN | MSDIS | MSEN | STOP | START |

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI_THR).

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read mode.

  – In single data byte master read, the START and STOP must both be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In master read mode, if a NACK bit is received, the STOP is automatically performed.
  – In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

### 29.10.2 TWI Master Mode Register

**Name:** TWI_MMR

Access: Read-write

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | DADR | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | MREAD | – | – | IADRSZ | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | – |

- **IADRSZ: Internal Device Address Size**

| IADRSZ[9:8] | | |
|---|---|---|
| 0 | 0 | No internal device address |
| 0 | 1 | One-byte internal device address |
| 1 | 0 | Two-byte internal device address |
| 1 | 1 | Three-byte internal device address |

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.

### 29.10.3 TWI Slave Mode Register

Name: TWI_SMR

Access: Read-write

**Reset Value:** 0x00000000

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

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

### 29.10.4 TWI Internal Address Register

Name: TWI_IADR

Access: Read-write

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| IADR | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| IADR | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IADR | | | | | | | |

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

### 29.10.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Access: Read-write

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| | | | | | CKDIV | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CHDIV | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLDIV | | | | | | | |

TWI_CWGR is only used in Master mode.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$$T_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

### 29.10.6 TWI Status Register

Name: TWI_SR

Access: Read-only

**Reset Value:** 0x0000F009

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCLWS | ARBLST | NACK |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | OVRE | GACC | SVACC | SVREAD | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed (automatically set / reset)**

TXCOMP used in Master mode:

0 = During the length of the current frame.

1 = When both holding and shifter registers are empty and STOP condition has been sent.

*TXCOMP behavior in Master mode* can be seen in Figure 29-8 on page 324 and in Figure 29-10 on page 325.

TXCOMP used in Slave mode:

0 = As soon as a Start is detected.

1 = After a Stop or a Repeated Start + an address different from SADR is detected.

*TXCOMP behavior in Slave mode* can be seen in Figure 29-27 on page 342, Figure 29-28 on page 343, Figure 29-29 on page 344 and Figure 29-30 on page 344.

- **RXRDY: Receive Holding Register Ready (automatically set / reset)**

0 = No character has been received since the last TWI_RHR read operation.

1 = A byte has been received in the TWI_RHR since the last read.

*RXRDY behavior in Master mode* can be seen in Figure 29-10 on page 325.

*RXRDY behavior in Slave mode* can be seen in Figure 29-25 on page 340, Figure 29-28 on page 343, Figure 29-29 on page 344 and Figure 29-30 on page 344.

- **TXRDY: Transmit Holding Register Ready (automatically set / reset)**

TXRDY used in Master mode:

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI_THR register.

1 = As soon as a data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

*TXRDY behavior in Master mode* can be seen in Figure 29-8 on page 324.

TXRDY used in Slave mode:

0 = As soon as data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1 = It indicates that the TWI_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI_THR to avoid losing it.

*TXRDY behavior in Slave mode* can be seen in Figure 29-24 on page 340, Figure 29-27 on page 342, Figure 29-29 on page 344 and Figure 29-30 on page 344.

- **SVREAD: Slave Read (automatically set / reset)**

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0 = Indicates that a write access is performed by a Master.

1 = Indicates that a read access is performed by a Master.

*SVREAD behavior* can be seen in Figure 29-24 on page 340, Figure 29-25 on page 340, Figure 29-29 on page 344 and Figure 29-30 on page 344.

- **SVACC: Slave Access (automatically set / reset)**

This bit is only used in Slave mode.

0 = TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1 = Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

*SVACC behavior* can be seen in Figure 29-24 on page 340, Figure 29-25 on page 340, Figure 29-29 on page 344 and Figure 29-30 on page 344.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, the programmer decoded the commands that follow and the programming sequence.

*GACC behavior* can be seen in Figure 29-26 on page 341.

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0 = TWI_RHR has not been loaded while RXRDY was set

1 = TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

NACK used in Master mode:

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0 = Each data byte has been correctly received by the Master.

1 = In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data are acknowledged by the TWI.

- **ARBLST: Arbitration Lost (clear on read)**

This bit is only used in Master mode.

0: Arbitration won.

1: Arbitration lost. Another master of the TWI bus has won the multi-master arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State (automatically set / reset)**

This bit is only used in Slave mode.

0 = The clock is not stretched.

1 = The clock is stretched. TWI_THR / TWI_RHR buffer is not filled / emptied before the emission / reception of a new character.

*SCLWS behavior* can be seen in Figure 29-27 on page 342 and Figure 29-28 on page 343.

- **EOSACC: End Of Slave Access (clear on read)**

This bit is only used in Slave mode.

0 = A slave access is being performing.

1 = The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

*EOSACC behavior* can be seen in Figure 29-29 on page 344 and Figure 29-30 on page 344

- **ENDRX: End of RX buffer**

This bit is only used in Master mode.

0 = The Receive Counter Register has not reached 0 since the last write in TWI_RCR or TWI_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in TWI_RCR or TWI_RNCR.

- **ENDTX: End of TX buffer**

This bit is only used in Master mode.

0 = The Transmit Counter Register has not reached 0 since the last write in TWI_TCR or TWI_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in TWI_TCR or TWI_TNCR.

- **RXBUFF: RX Buffer Full**

This bit is only used in Master mode.

0 = TWI_RCR or TWI_RNCR have a value other than 0.

1 = Both TWI_RCR and TWI_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

This bit is only used in Master mode.

0 = TWI_TCR or TWI_TNCR have a value other than 0.

1 = Both TWI_TCR and TWI_TNCR have a value of 0.

### 29.10.7 TWI Interrupt Enable Register

Name: TWI_IER

Access: Write-only

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCL_WS | ARBLST | NACK |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | OVRE | GACC | SVACC | – | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**
- **EOSACC: End Of Slave Access Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 29.10.8 TWI Interrupt Disable Register

Name: TWI_IDR

Access: Write-only

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCL_WS | ARBLST | NACK |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | OVRE | GACC | SVACC | – | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ARBLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

### 29.10.9 TWI Interrupt Mask Register

Name: TWI_IMR

Access: Read-only

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCL_WS | ARBLST | NACK |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | OVRE | GACC | SVACC | – | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

### 29.10.10 TWI Receive Holding Register

Name: TWI_RHR

Access: Read-only

**Reset Value:** 0x00000000

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

- **RXDATA: Master or Slave Receive Holding Data**

### 29.10.11 TWI Transmit Holding Register

Name: TWI_THR

Access: Read-write

**Reset Value:** 0x00000000

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

- **TXDATA: Master or Slave Transmit Holding Data**

# 30. Universal Synchronous Asynchronous Receiver Transceiver (USART)

## 30.1 Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

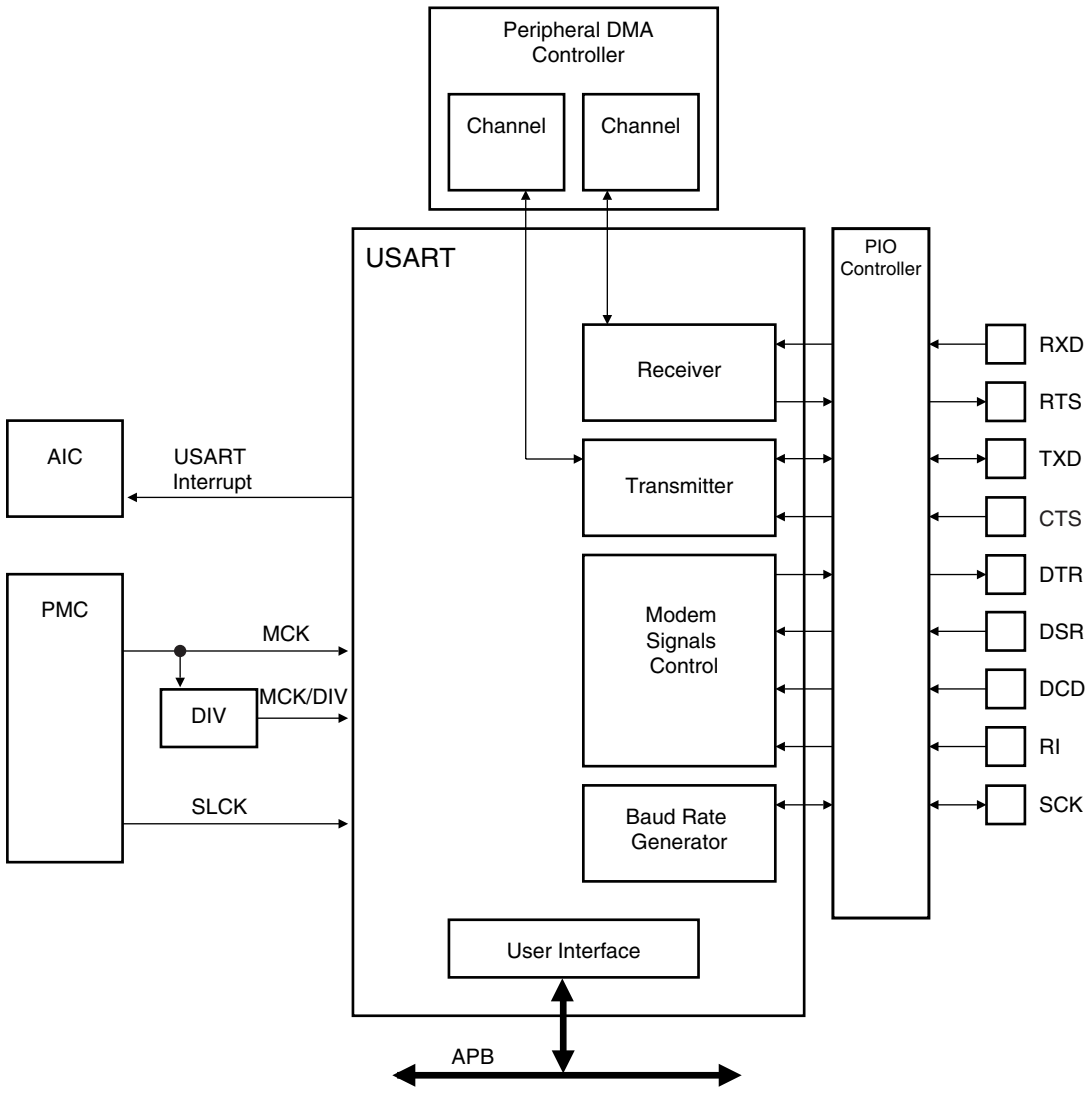
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

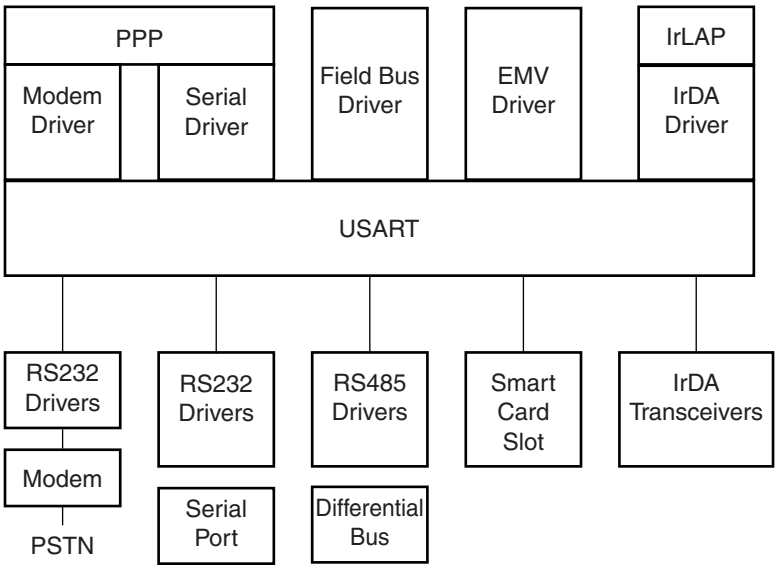
## 30.2 Block Diagram

**Figure 30-1.** USART Block Diagram



## 30.3 Application Block Diagram

**Figure 30-2.** Application Block Diagram



## 30.4 I/O Lines Description

**Table 30-1.** I/O Line Description

| Name | Description | Type | Active Level |
|---|---|---|---|
| SCK | Serial Clock | I/O | |
| TXD | Transmit Serial Data | I/O | |
| RXD | Receive Serial Data | Input | |
| RI | Ring Indicator | Input | Low |
| DSR | Data Set Ready | Input | Low |
| DCD | Data Carrier Detect | Input | Low |
| DTR | Data Terminal Ready | Output | Low |
| CTS | Clear to Send | Input | Low |
| RTS | Request to Send | Output | Low |

## 30.5 Product Dependencies

### 30.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature or Modem mode is used, the internal pull up on TXD must also be enabled.

All the pins of the modems may or may not be implemented on the USART. Only USART1 fully equipped with all the modem signals. On USARTs not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

### 30.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

### 30.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

## 30.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - Remote loopback, local loopback, automatic echo

### 30.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

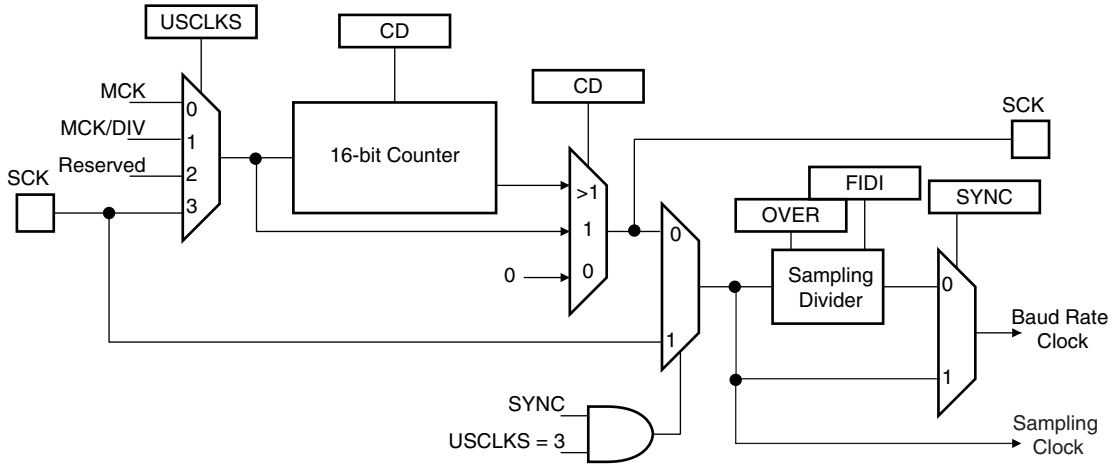
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

**Figure 30-3.** Baud Rate Generator




#### 30.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

### 30.6.1.2 Baud Rate Calculation Example

Table 30-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

**Table 30-2.** Baud Rate Example (OVER = 0)

| Source Clock | Expected Baud Rate | Calculation Result | CD | Actual Baud Rate | Error |
|---|---|---|---|---|---|
| MHz | Bit/s | | | Bit/s | |
| 3 686 400 | 38 400 | 6.00 | 6 | 38 400.00 | 0.00% |
| 4 915 200 | 38 400 | 8.00 | 8 | 38 400.00 | 0.00% |
| 5 000 000 | 38 400 | 8.14 | 8 | 39 062.50 | 1.70% |
| 7 372 800 | 38 400 | 12.00 | 12 | 38 400.00 | 0.00% |
| 8 000 000 | 38 400 | 13.02 | 13 | 38 461.54 | 0.16% |
| 12 000 000 | 38 400 | 19.53 | 20 | 37 500.00 | 2.40% |
| 12 288 000 | 38 400 | 20.00 | 20 | 38 400.00 | 0.00% |
| 14 318 180 | 38 400 | 23.30 | 23 | 38 908.10 | 1.31% |
| 14 745 600 | 38 400 | 24.00 | 24 | 38 400.00 | 0.00% |
| 18 432 000 | 38 400 | 30.00 | 30 | 38 400.00 | 0.00% |
| 24 000 000 | 38 400 | 39.06 | 39 | 38 461.54 | 0.16% |
| 24 576 000 | 38 400 | 40.00 | 40 | 38 400.00 | 0.00% |
| 25 000 000 | 38 400 | 40.69 | 40 | 38 109.76 | 0.76% |
| 32 000 000 | 38 400 | 52.08 | 52 | 38 461.54 | 0.16% |
| 32 768 000 | 38 400 | 53.33 | 53 | 38 641.51 | 0.63% |
| 33 000 000 | 38 400 | 53.71 | 54 | 38 194.44 | 0.54% |
| 40 000 000 | 38 400 | 65.10 | 65 | 38 461.54 | 0.16% |
| 50 000 000 | 38 400 | 81.38 | 81 | 38 580.25 | 0.47% |

The baud rate is calculated with the following formula:

$$BaudRate = MCK/CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

### 30.6.1.3 Fractional Baud Rate in Asynchronous Mode

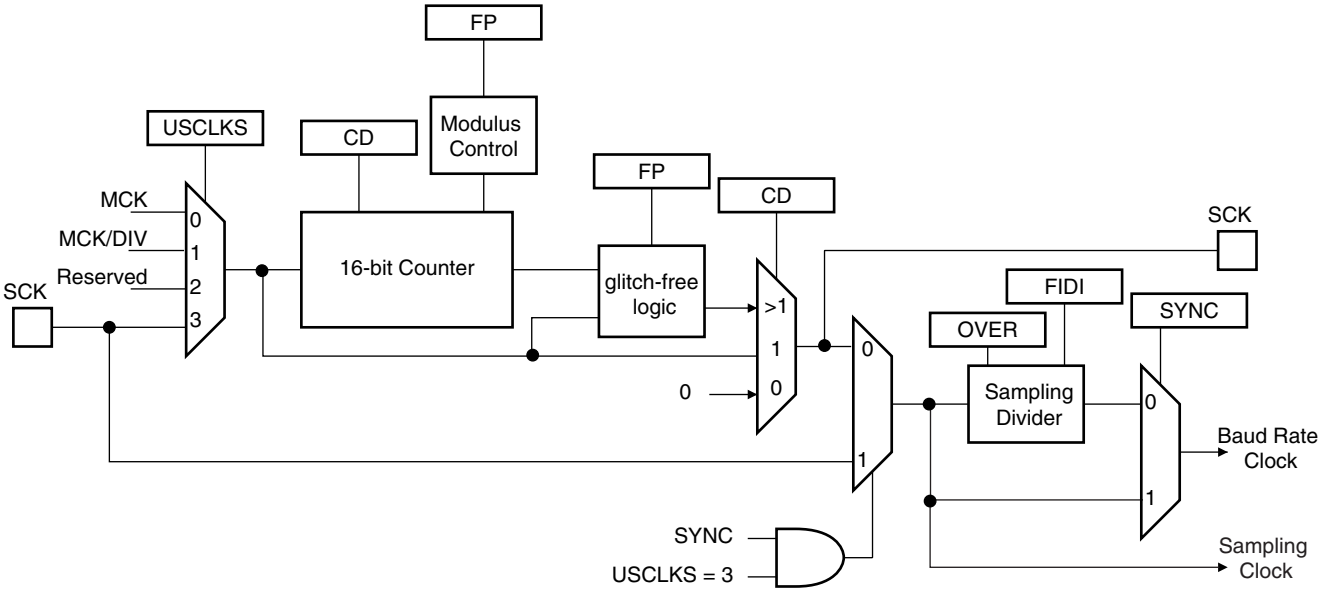
The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register (US_BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the

clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2-Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

**Figure 30-4.** Fractional Baud Rate Generator




### 30.6.1.4 *Baud Rate in Synchronous Mode*

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

### 30.6.1.5 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 30-3.

**Table 30-3.** Binary and Decimal Values for Di

| DI field | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 1000 | 1001 |
|---|---|---|---|---|---|---|---|---|
| Di (decimal) | 1 | 2 | 4 | 8 | 16 | 32 | 12 | 20 |

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 30-4.

**Table 30-4.** Binary and Decimal Values for Fi

| FI field | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 1001 | 1010 | 1011 | 1100 | 1101 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fi (decimal | 372 | 372 | 558 | 744 | 1116 | 1488 | 1860 | 512 | 768 | 1024 | 1536 | 2048 |

Table 30-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

**Table 30-5.** Possible Values for the Fi/Di Ratio

| Fi/Di | 372 | 558 | 774 | 1116 | 1488 | 1806 | 512 | 768 | 1024 | 1536 | 2048 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 372 | 558 | 744 | 1116 | 1488 | 1860 | 512 | 768 | 1024 | 1536 | 2048 |
| 2 | 186 | 279 | 372 | 558 | 744 | 930 | 256 | 384 | 512 | 768 | 1024 |
| 4 | 93 | 139.5 | 186 | 279 | 372 | 465 | 128 | 192 | 256 | 384 | 512 |
| 8 | 46.5 | 69.75 | 93 | 139.5 | 186 | 232.5 | 64 | 96 | 128 | 192 | 256 |
| 16 | 23.25 | 34.87 | 46.5 | 69.75 | 93 | 116.2 | 32 | 48 | 64 | 96 | 128 |
| 32 | 11.62 | 17.43 | 23.25 | 34.87 | 46.5 | 58.13 | 16 | 24 | 32 | 48 | 64 |
| 12 | 31 | 46.5 | 62 | 93 | 124 | 155 | 42.66 | 64 | 85.33 | 128 | 170.6 |
| 20 | 18.6 | 27.9 | 37.2 | 55.8 | 74.4 | 93 | 25.6 | 38.4 | 51.2 | 76.8 | 102.4 |

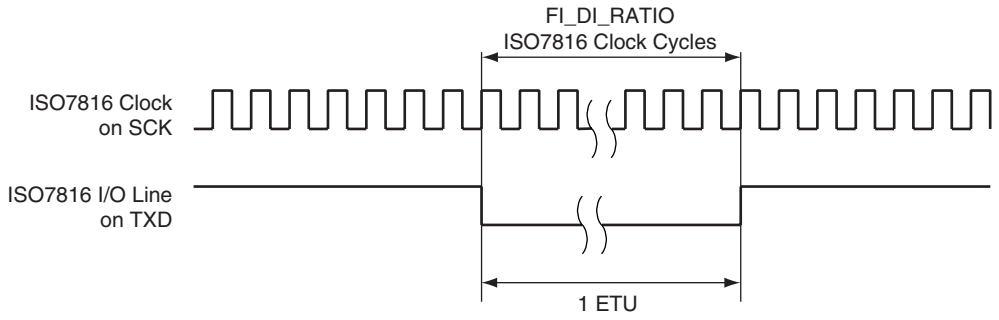
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 30-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

**Figure 30-5.** Elementary Time Unit (ETU)




### 30.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US_CR). The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US_THR). If a timeguard is programmed, it is handled normally.

### 30.6.3 Synchronous and Asynchronous Modes

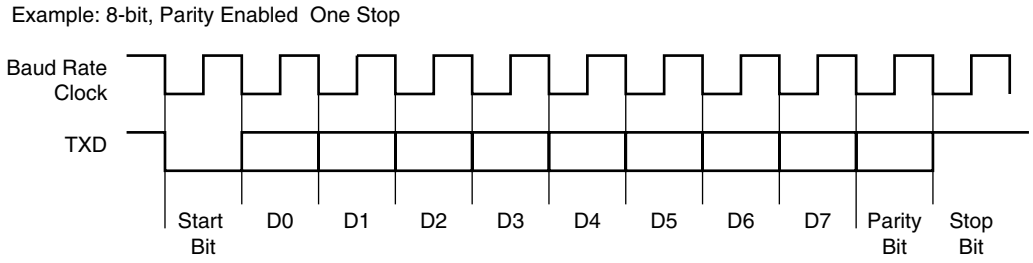
#### *30.6.3.1 Transmitter Operations*

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (US_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The num-

ber of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

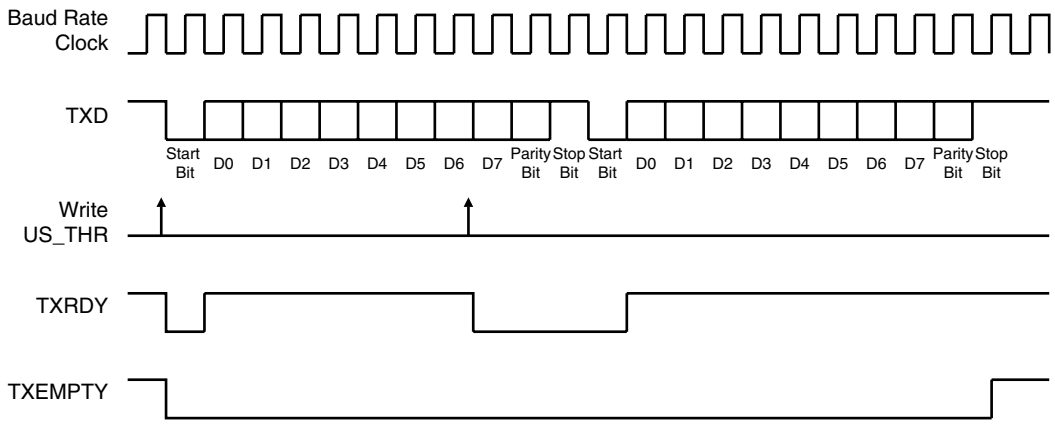
**Figure 30-6.** Character Transmit




The characters are sent by writing in the Transmit Holding Register (US_THR). The transmitter reports two status bits in the Channel Status Register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift Register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

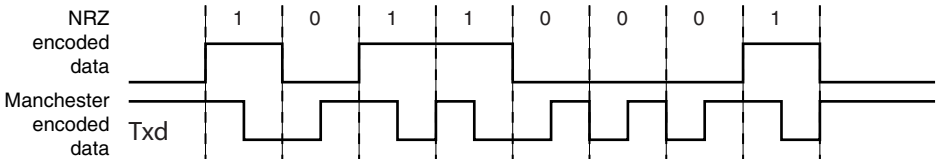
**Figure 30-7.** Transmitter Status




### *30.6.3.2 Manchester Encoder*

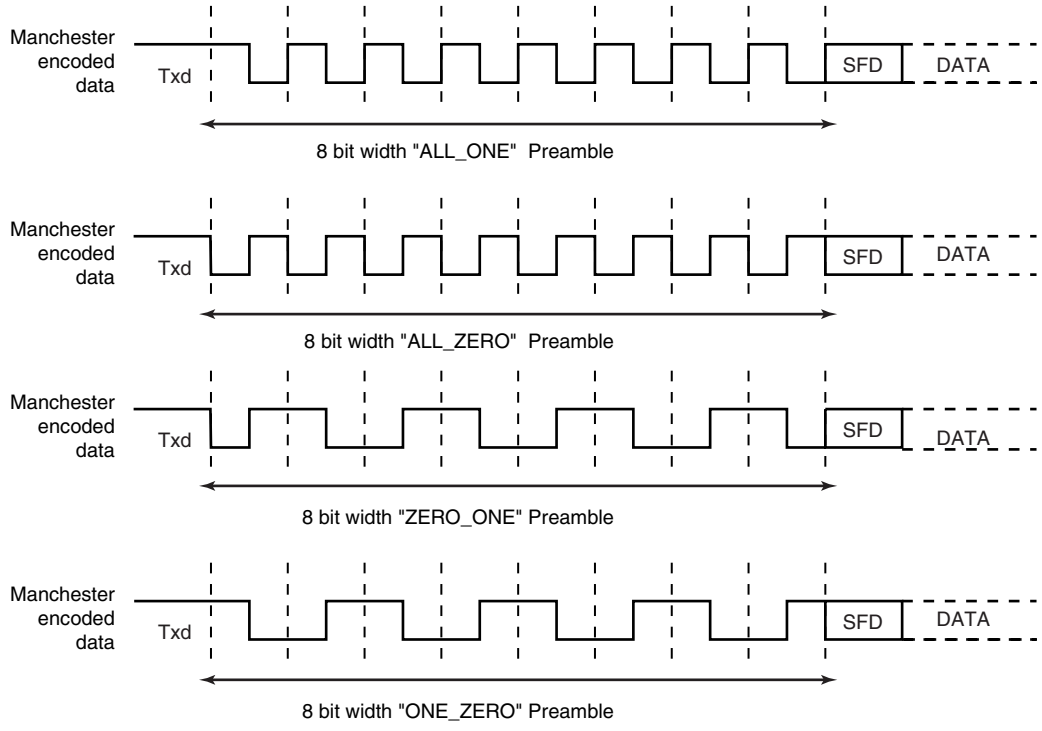
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 30-8 illustrates this coding scheme.

**Figure 30-8.** NRZ to Manchester Encoding




The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 30-9 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

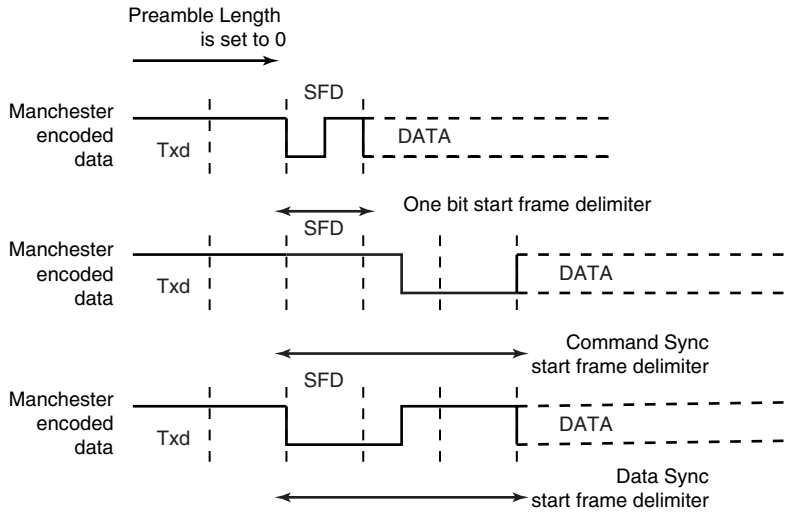
**Figure 30-9.** Preamble Patterns, Default Polarity Assumed




A start frame delimiter is to be configured using the ONEBIT field in the US_MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 30-10 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new

character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the US_MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in US_MR register must be set to 1. In this case, the MODSYNC field in US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR register. The USART character format is modified and includes sync information.

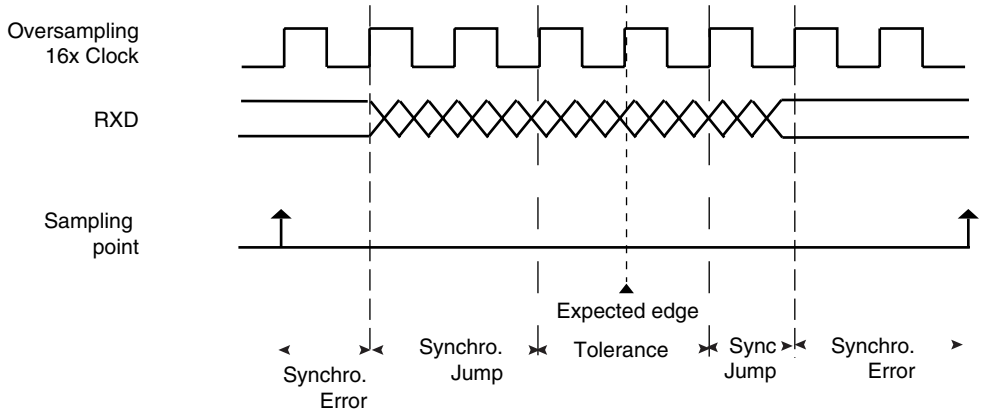
**Figure 30-10.** Start Frame Delimiter




### 30.6.3.3 *Drift Compensation*

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

**Figure 30-11.** Bit Resynchronization




*30.6.3.4 Asynchronous Receiver*

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 30-12 and Figure 30-13 illustrate start detection and character reception when USART operates in asynchronous mode.

**Figure 30-12.** Asynchronous Start Detection

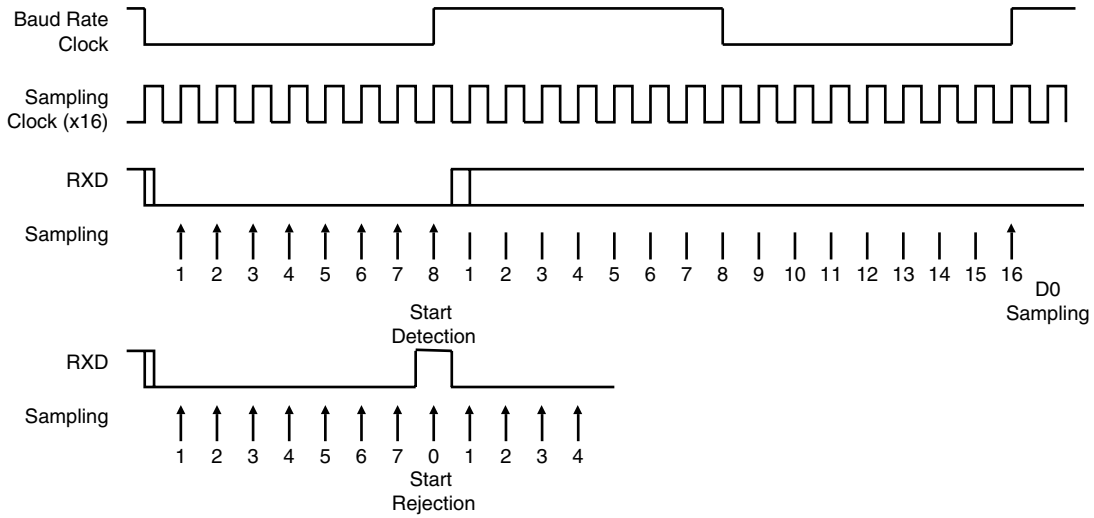
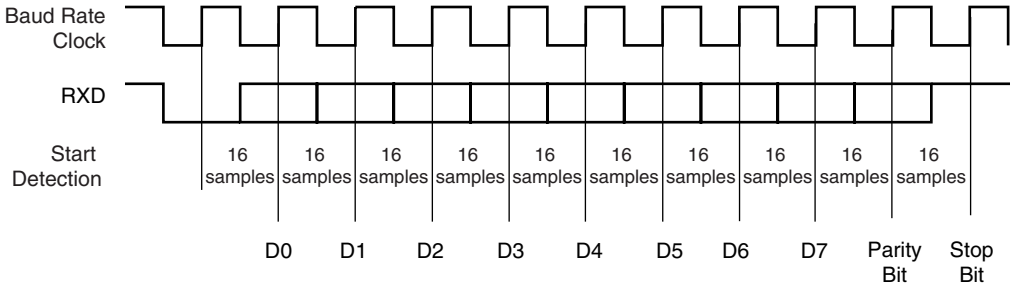



**Figure 30-13.** Asynchronous Character Reception

Example: 8-bit, Parity Enabled



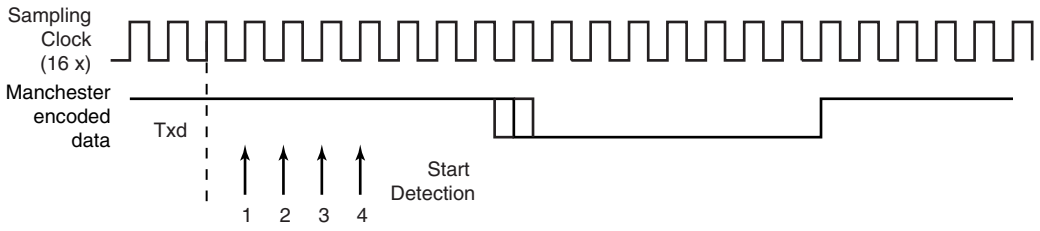

### *30.6.3.5 Manchester Decoder*

When the MAN field in US_MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in US_MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in US_MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in US_MAN. See Figure 30-9 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 30-14.. The sample pulse rejection mechanism applies.

**Figure 30-14.** Asynchronous Start Bit Detection




The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 30-15 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in US_CSR register is raised. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. See Figure 30-16 for an example of Manchester error detection during data phase.

**Figure 30-15.** Preamble Pattern Mismatch

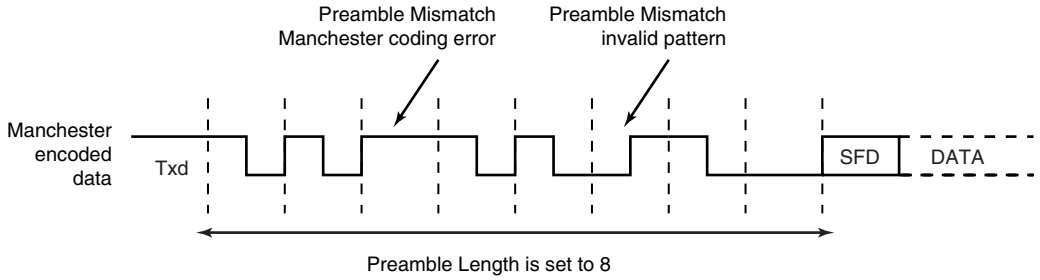
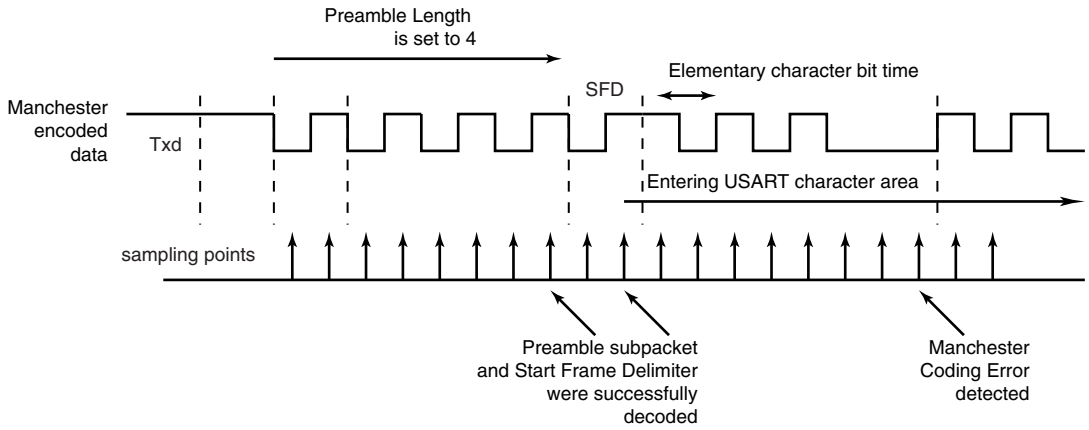



**Figure 30-16.** Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

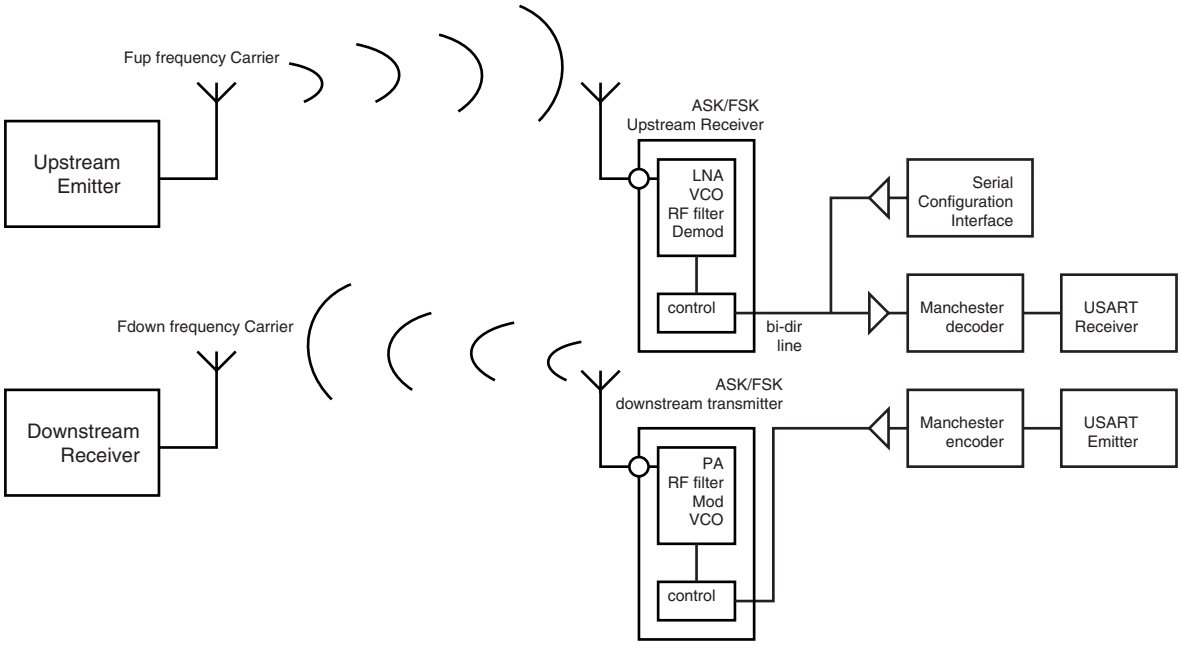
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

### *30.6.3.6 Radio Interface: Manchester Encoded USART Application*

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 30-17.

**Figure 30-17.** Manchester Encoded Characters RF Transmission




The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 30-18 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 30-19.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver

switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

**Figure 30-18.** ASK Modulator Output

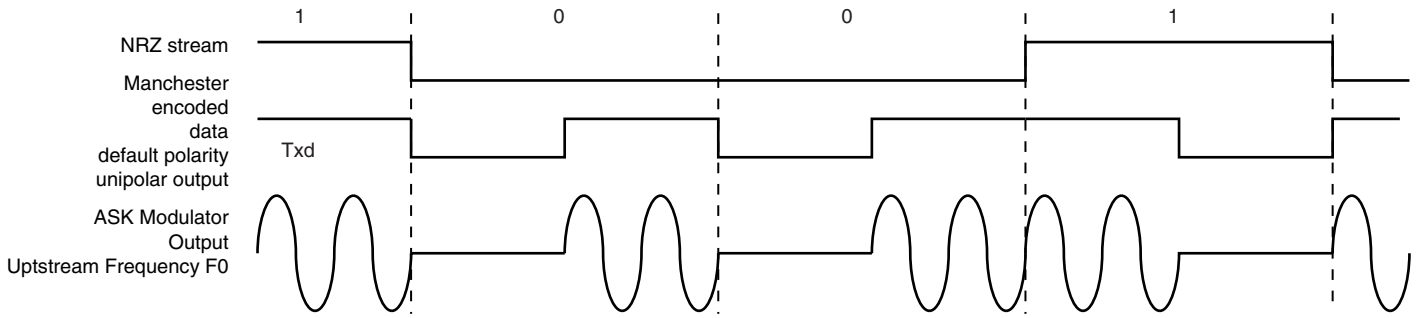
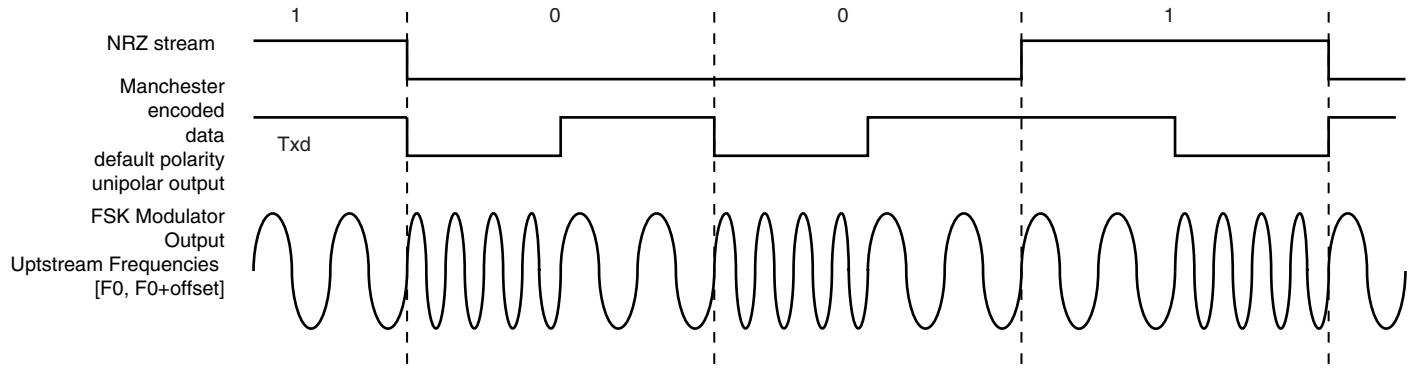



**Figure 30-19.** FSK Modulator Output



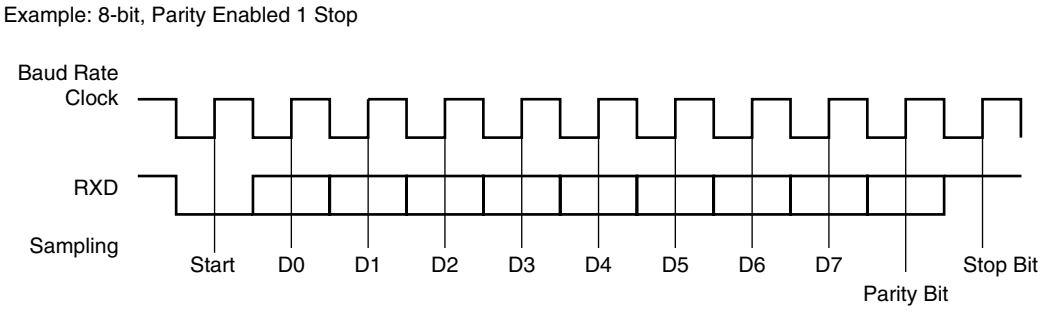

### *30.6.3.7 Synchronous Receiver*

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 30-20 illustrates a character reception in synchronous mode.

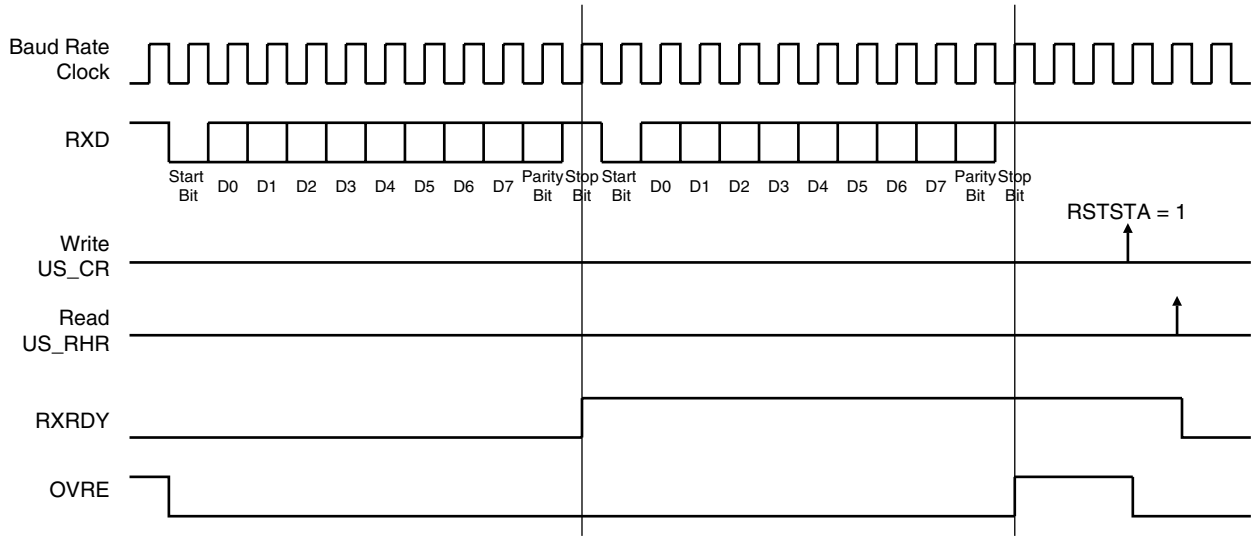
**Figure 30-20.** Synchronous Mode Character Reception



### 30.6.3.8 *Receiver Operations*

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit at 1.

**Figure 30-21.** Receiver Status



*30.6.3.9 Parity*

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see "Multidrop Mode" on page 382. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

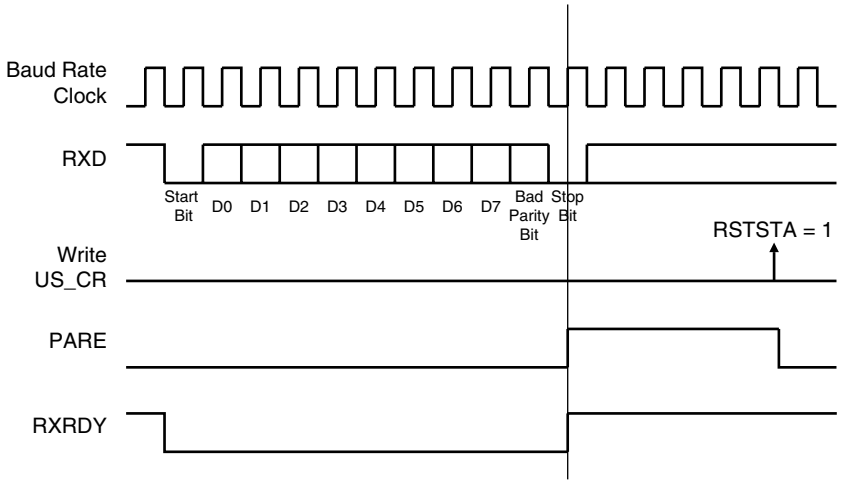
Table 30-6 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

**Table 30-6.** Parity Bit Examples

| Character | Hexa | Binary | Parity Bit | Parity Mode |
|---|---|---|---|---|
| A | 0x41 | 0100 0001 | 1 | Odd |
| A | 0x41 | 0100 0001 | 0 | Even |
| A | 0x41 | 0100 0001 | 1 | Mark |
| A | 0x41 | 0100 0001 | 0 | Space |
| A | 0x41 | 0100 0001 | None | None |

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US_CSR). The PARE bit can be cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. Figure 30-22 illustrates the parity bit status setting and clearing.

**Figure 30-22.** Parity Error




### 30.6.3.10 *Multidrop Mode*

If the PAR field in the Mode Register (US_MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to US_CR. In this case, the next byte written to US_THR is transmitted as an address. Any character written in US_THR without having written the command SENDA is transmitted normally with the parity at 0.

### 30.6.3.11 *Transmitter Timeguard*

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US_TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 30-23, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

**Figure 30-23.** Timeguard Operations

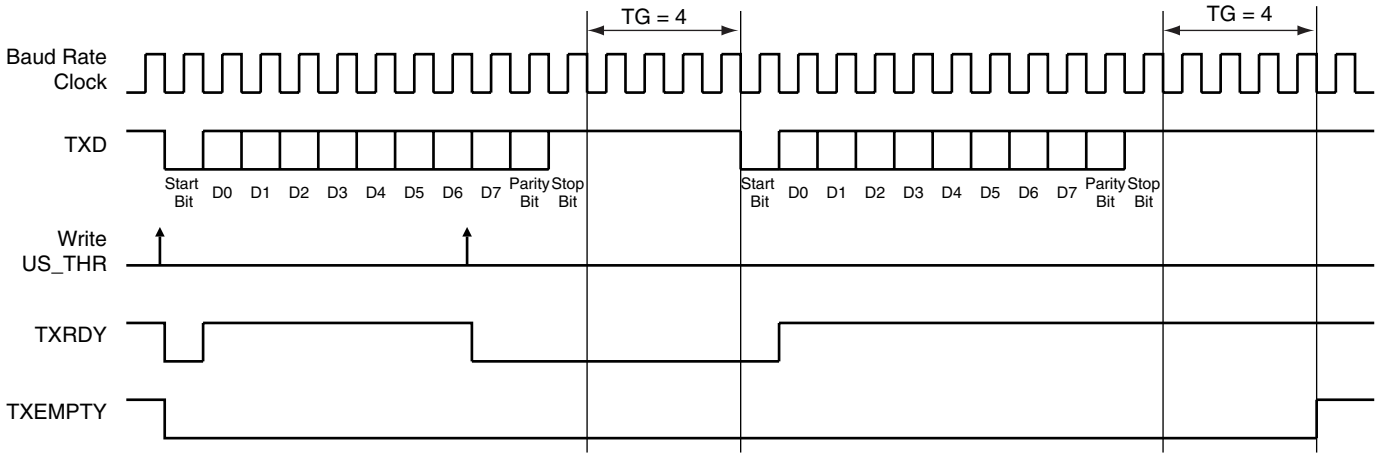



Table 30-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

**Table 30-7.** Maximum Timeguard Length Depending on Baud Rate

| Baud Rate | Bit time | Timeguard |
|---|---|---|
| Bit/sec | µs | ms |
| 1 200 | 833 | 212.50 |
| 9 600 | 104 | 26.56 |
| 14400 | 69.4 | 17.71 |
| 19200 | 52.1 | 13.28 |
| 28800 | 34.7 | 8.85 |
| 33400 | 29.9 | 7.63 |
| 56000 | 17.9 | 4.55 |
| 57600 | 17.4 | 4.43 |
| 115200 | 8.7 | 2.21 |

### *30.6.3.12 Receiver Time-out*

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing the Control Register (US_CR) with the STTTO (Start Time-out) bit at 1. In this case, the idle state

on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.

- Obtain an interrupt while no character is received. This is performed by writing US_CR with the RETTO (Reload and Start Time-out) bit at 1. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 30-24 shows the block diagram of the Receiver Time-out feature.

**Figure 30-24.** Receiver Time-out Block Diagram

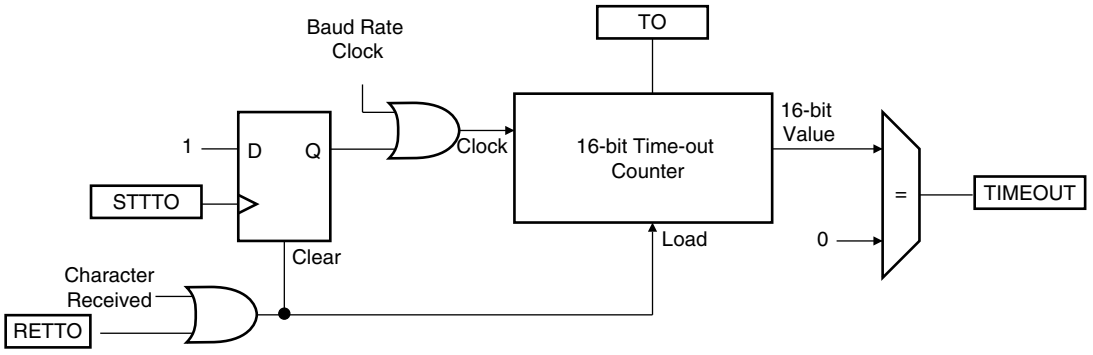



Table 30-8 gives the maximum time-out period for some standard baud rates.

**Table 30-8.** Maximum Time-out Period

| Baud Rate | Bit Time | Time-out |
|---|---|---|
| bit/sec | µs | ms |
| 600 | 1 667 | 109 225 |
| 1 200 | 833 | 54 613 |
| 2 400 | 417 | 27 306 |
| 4 800 | 208 | 13 653 |
| 9 600 | 104 | 6 827 |
| 14400 | 69 | 4 551 |
| 19200 | 52 | 3 413 |
| 28800 | 35 | 2 276 |
| 33400 | 30 | 1 962 |

**Table 30-8.** Maximum Time-out Period (Continued)

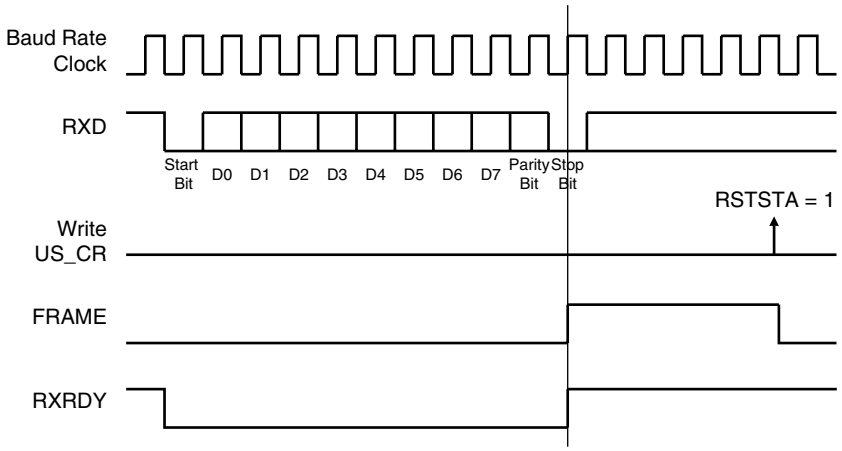
| Baud Rate | Bit Time | Time-out |
|---|---|---|
| 56000 | 18 | 1 170 |
| 57600 | 17 | 1 138 |
| 200000 | 5 | 328 |

### *30.6.3.13 Framing Error*

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1.

**Figure 30-25.** Framing Error Status




### *30.6.3.14 Transmit Break*

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

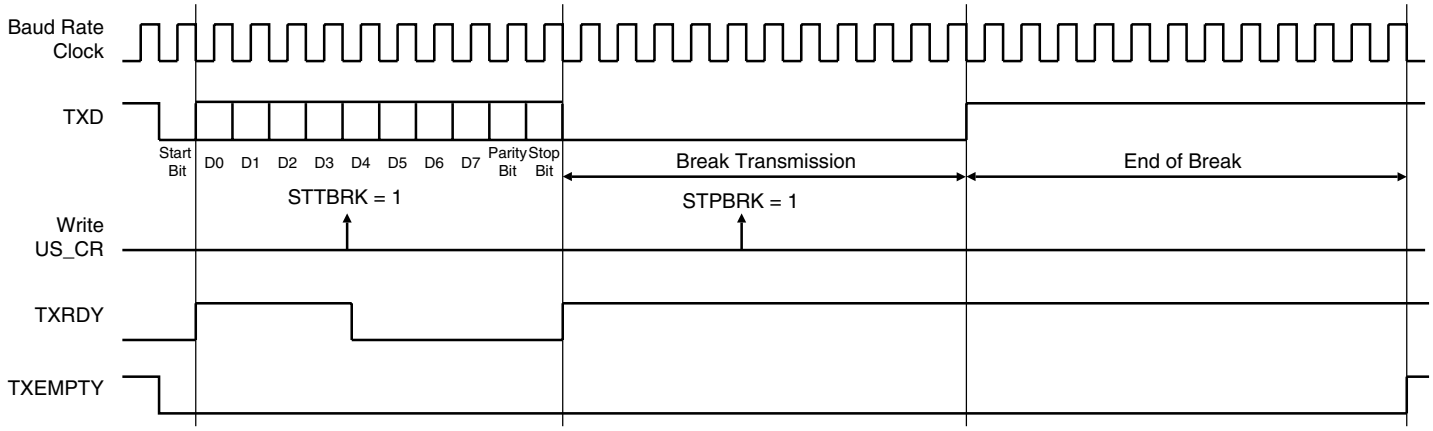
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 30-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

**Figure 30-26.** Break Transmission




### *30.6.3.15 Receive Break*

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

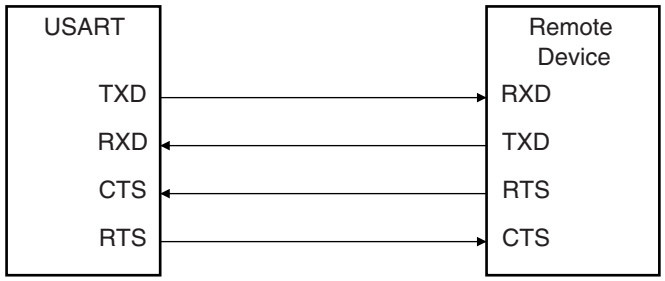
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

### *30.6.3.16 Hardware Handshaking*

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 30-27.

**Figure 30-27.** Connection with a Remote Device for Hardware Handshaking




Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 30-28 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

**Figure 30-28.** Receiver Behavior when Operating with Hardware Handshaking

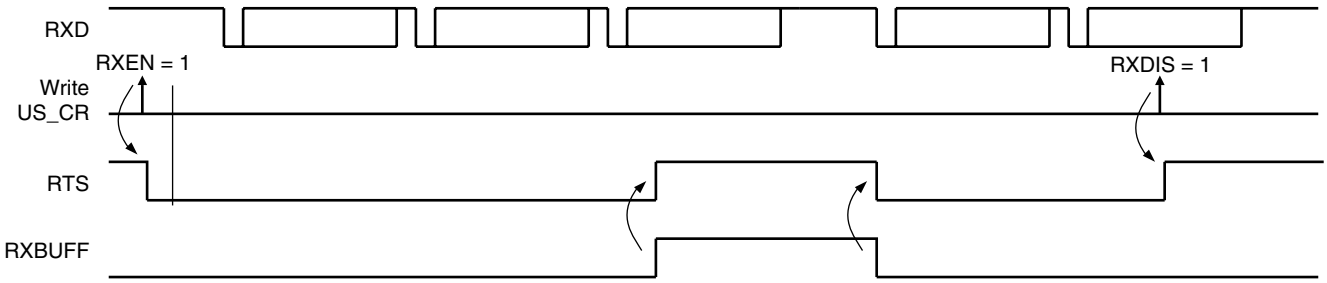
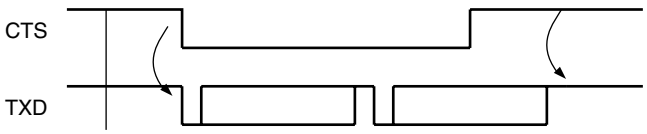



Figure 30-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

**Figure 30-29.** Transmitter Behavior when Operating with Hardware Handshaking



### 30.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

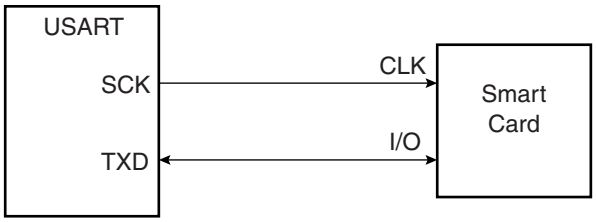
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

#### *30.6.4.1 ISO7816 Mode Overview*

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 367).

The USART connects to a smart card as shown in Figure 30-30. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

**Figure 30-30.** Connection of a Smart Card to the USART




When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "USART Mode Register" on page 400 and "PAR: Parity Type" on page 401.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

#### *30.6.4.2 Protocol T = 0*

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 30-31.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 30-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

**Figure 30-31.** T = 0 Protocol without Parity Error

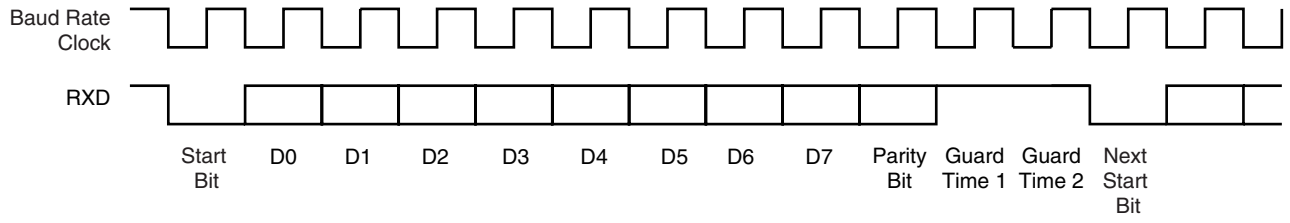
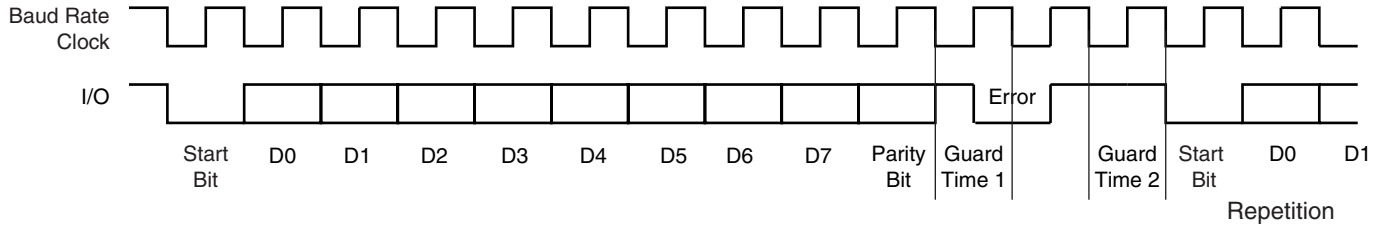



**Figure 30-32.** T = 0 Protocol with Parity Error




### *30.6.4.3 Receive Error Counter*

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

### *30.6.4.4 Receive NACK Inhibit*

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US_SR). The INACK bit can be cleared by writing the Control Register (US_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

### *30.6.4.5 Transmit Character Repetition*

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (US_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (US_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

#### *30.6.4.6 Disable Successive Receive NACK*

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US_MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

#### *30.6.4.7 Protocol T = 1*

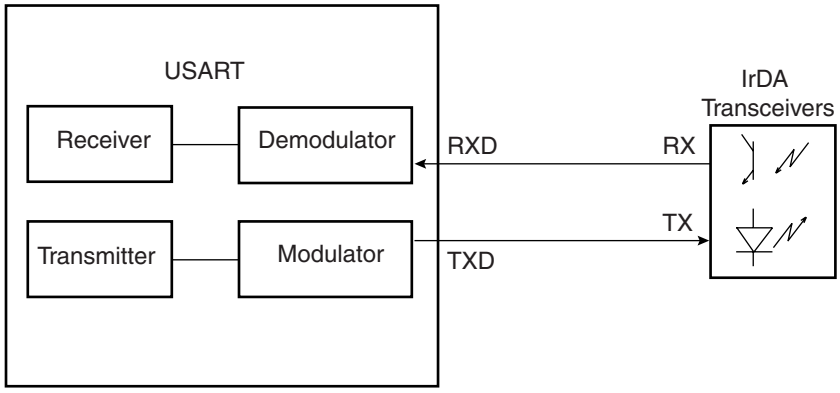
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

### 30.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 30-33. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the USART_MODE field in the Mode Register (US_MR) to the value 0x8. The IrDA Filter Register (US_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

**Figure 30-33.** Connection to IrDA Transceivers




The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX

- Configure the TXD pin as PIO and set it as an output at 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

### 30.6.5.1 IrDA Modulation

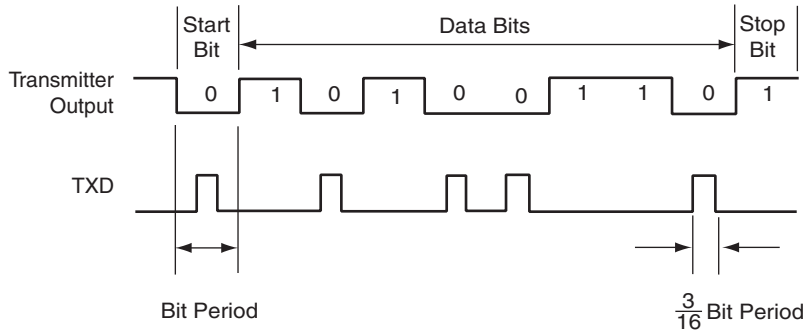
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 30-9.

**Table 30-9.** IrDA Pulse Duration

| Baud Rate | Pulse Duration (3/16) |
|---|---|
| 2.4 Kb/s | 78.13 µs |
| 9.6 Kb/s | 19.53 µs |
| 19.2 Kb/s | 9.77 µs |
| 38.4 Kb/s | 4.88 µs |
| 57.6 Kb/s | 3.26 µs |
| 115.2 Kb/s | 1.63 µs |

Figure 30-34 shows an example of character transmission.

**Figure 30-34.** IrDA Modulation




### 30.6.5.2 IrDA Baud Rate

Table 30-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

**Table 30-10.** IrDA Baud Rate Error

| Peripheral Clock | Baud Rate | CD | Baud Rate Error | Pulse Time |
|---|---|---|---|---|
| 3 686 400 | 115 200 | 2 | 0.00% | 1.63 |
| 20 000 000 | 115 200 | 11 | 1.38% | 1.63 |
| 32 768 000 | 115 200 | 18 | 1.25% | 1.63 |
| 40 000 000 | 115 200 | 22 | 1.38% | 1.63 |
| 3 686 400 | 57 600 | 4 | 0.00% | 3.26 |
| 20 000 000 | 57 600 | 22 | 1.38% | 3.26 |
| 32 768 000 | 57 600 | 36 | 1.25% | 3.26 |

**Table 30-10.** IrDA Baud Rate Error (Continued)

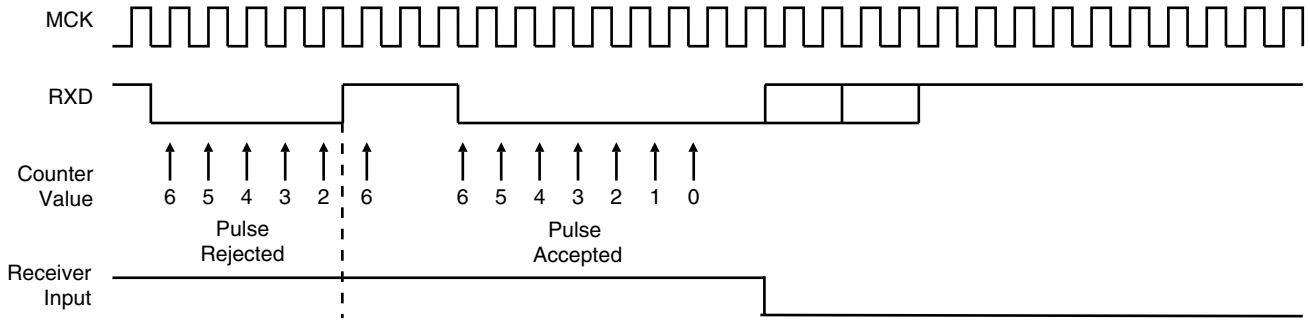
| Peripheral Clock | Baud Rate | CD | Baud Rate Error | Pulse Time |
|---|---|---|---|---|
| 40 000 000 | 57 600 | 43 | 0.93% | 3.26 |
| 3 686 400 | 38 400 | 6 | 0.00% | 4.88 |
| 20 000 000 | 38 400 | 33 | 1.38% | 4.88 |
| 32 768 000 | 38 400 | 53 | 0.63% | 4.88 |
| 40 000 000 | 38 400 | 65 | 0.16% | 4.88 |
| 3 686 400 | 19 200 | 12 | 0.00% | 9.77 |
| 20 000 000 | 19 200 | 65 | 0.16% | 9.77 |
| 32 768 000 | 19 200 | 107 | 0.31% | 9.77 |
| 40 000 000 | 19 200 | 130 | 0.16% | 9.77 |
| 3 686 400 | 9 600 | 24 | 0.00% | 19.53 |
| 20 000 000 | 9 600 | 130 | 0.16% | 19.53 |
| 32 768 000 | 9 600 | 213 | 0.16% | 19.53 |
| 40 000 000 | 9 600 | 260 | 0.16% | 19.53 |
| 3 686 400 | 2 400 | 96 | 0.00% | 78.13 |
| 20 000 000 | 2 400 | 521 | 0.03% | 78.13 |
| 32 768 000 | 2 400 | 853 | 0.04% | 78.13 |

### *30.6.5.3 IrDA Demodulator*

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 30-35 illustrates the operations of the IrDA demodulator.

**Figure 30-35.** IrDA Demodulator Operations

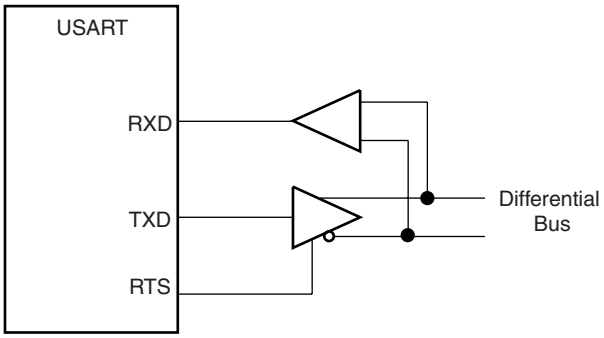



As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

### 30.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 30-36.

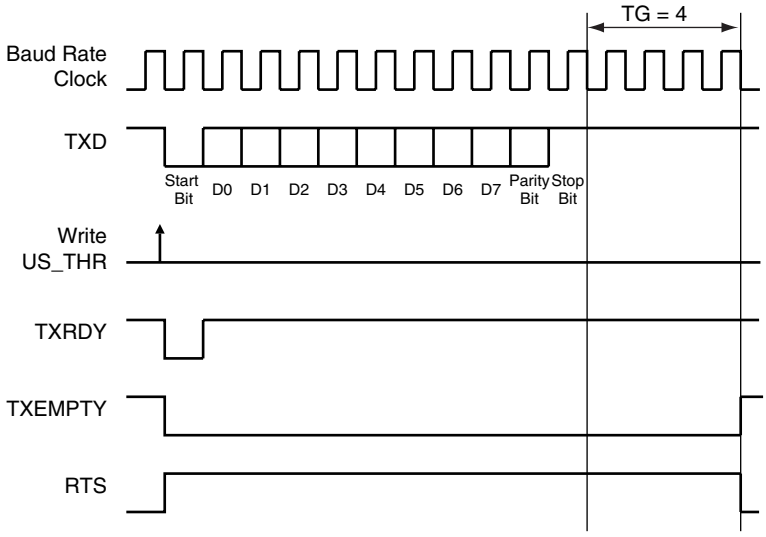
**Figure 30-36.** Typical Connection to a RS485 Bus




The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 30-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

**Figure 30-37.** Example of RTS Drive with Timeguard



### 30.6.7 Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 30-11 gives the correspondence of the USART signals with modem connection standards.

**Table 30-11.** Circuit References

| USART Pin | V24 | CCITT | Direction |
|---|---|---|---|
| TXD | 2 | 103 | From terminal to modem |
| RTS | 4 | 105 | From terminal to modem |
| DTR | 20 | 108.2 | From terminal to modem |
| RXD | 3 | 104 | From modem to terminal |
| CTS | 5 | 106 | From terminal to modem |
| DSR | 6 | 107 | From terminal to modem |
| DCD | 8 | 109 | From terminal to modem |
| RI | 22 | 125 | From terminal to modem |

The control of the DTR output pin is performed by writing the Control Register (US_CR) with the DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable command forces the corresponding pin to its active level, i.e. low. RTS output pin is automatically controlled in this mode

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (US_CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when US_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

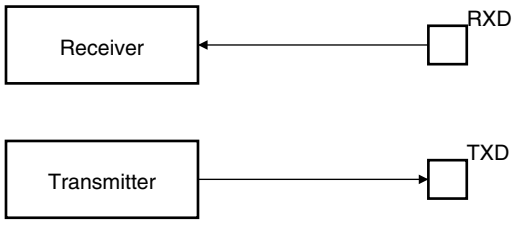
### 30.6.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

#### *30.6.8.1 Normal Mode*

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

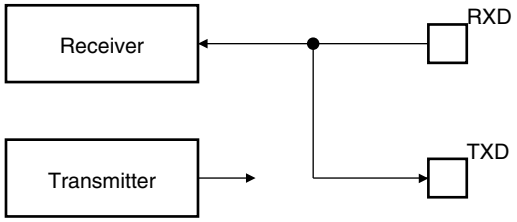
**Figure 30-38.** Normal Mode Configuration




#### *30.6.8.2 Automatic Echo Mode*

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 30-39. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

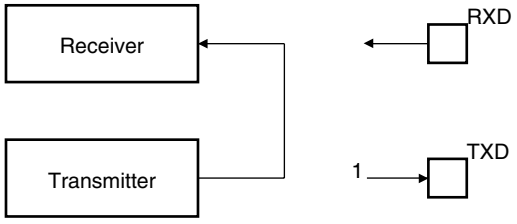
**Figure 30-39.** Automatic Echo Mode Configuration




#### *30.6.8.3 Local Loopback Mode*

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 30-40. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

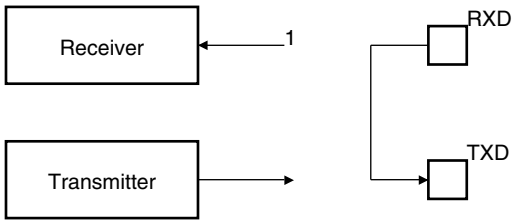
**Figure 30-40.** Local Loopback Mode Configuration



### 30.6.8.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 30-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

**Figure 30-41.** Remote Loopback Mode Configuration



## 30.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

**Table 30-13.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0000 | Control Register | US_CR | Write-only | – |
| 0x0004 | Mode Register | US_MR | Read-write | – |
| 0x0008 | Interrupt Enable Register | US_IER | Write-only | – |
| 0x000C | Interrupt Disable Register | US_IDR | Write-only | – |
| 0x0010 | Interrupt Mask Register | US_IMR | Read-only | 0x0 |
| 0x0014 | Channel Status Register | US_CSR | Read-only | – |
| 0x0018 | Receiver Holding Register | US_RHR | Read-only | 0x0 |
| 0x001C | Transmitter Holding Register | US_THR | Write-only | – |
| 0x0020 | Baud Rate Generator Register | US_BRGR | Read-write | 0x0 |
| 0x0024 | Receiver Time-out Register | US_RTOR | Read-write | 0x0 |
| 0x0028 | Transmitter Timeguard Register | US_TTGR | Read-write | 0x0 |
| 0x2C - 0x3C | Reserved | – | – | – |
| 0x0040 | FI DI Ratio Register | US_FIDI | Read-write | 0x174 |
| 0x0044 | Number of Errors Register | US_NER | Read-only | – |
| 0x0048 | Reserved | – | – | – |
| 0x004C | IrDA Filter Register | US_IF | Read-write | 0x0 |
| 0x0050 | Manchester Encoder Decoder Register | US_MAN | Read-write | 0x30011004 |
| 0x100 - 0x128 | Reserved for PDC Registers | – | – | – |

### 30.7.1 USART Control Register

**Name:** US_CR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RTSDIS | RTSEN | DTRDIS | DTREN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RETTO | RSTNACK | RSTIT | SENDA | STTTO | STPBRK | STTBRK | RSTSTA |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDIS | TXEN | RXDIS | RXEN | RSTTX | RSTRX | – | – |

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR and RXBRK in US_CSR.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect.

1: Starts waiting for a character before clocking the time-out counter. Resets the status bit TIMEOUT in US_CSR.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR at 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

### 30.7.2 USART Mode Register

**Name:** US_MR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ONEBIT | MODSYNC– | MAN | FILTER | – | MAX_ITERATION | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | VAR_SYNC | DSNACK | INACK | OVER | CLKO | MODE9 | MSBF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHMODE | | NBSTOP | | PAR | | | SYNC |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHRL | | USCLKS | | USART_MODE | | | |

- **USART_MODE**

| USART_MODE | | | | Mode of the USART |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal |
| 0 | 0 | 0 | 1 | RS485 |
| 0 | 0 | 1 | 0 | Hardware Handshaking |
| 0 | 0 | 1 | 1 | Modem |
| 0 | 1 | 0 | 0 | IS07816 Protocol: T = 0 |
| 0 | 1 | 1 | 0 | IS07816 Protocol: T = 1 |
| 1 | 0 | 0 | 0 | IrDA |
| Others | | | | Reserved |

- **USCLKS: Clock Selection**

| USCLKS | | Selected Clock |
|---|---|---|
| 0 | 0 | MCK |
| 0 | 1 | MCK/DIV (DIV = 8) |
| 1 | 0 | Reserved |
| 1 | 1 | SCK |

- **CHRL: Character Length.**

| CHRL | | Character Length |
|---|---|---|
| 0 | 0 | 5 bits |
| 0 | 1 | 6 bits |
| 1 | 0 | 7 bits |
| 1 | 1 | 8 bits |

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

| PAR | | | Parity Type |
|---|---|---|---|
| 0 | 0 | 0 | Even parity |
| 0 | 0 | 1 | Odd parity |
| 0 | 1 | 0 | Parity forced to 0 (Space) |
| 0 | 1 | 1 | Parity forced to 1 (Mark) |
| 1 | 0 | x | No parity |
| 1 | 1 | x | Multidrop mode |

- **NBSTOP: Number of Stop Bits**

| NBSTOP | | Asynchronous (SYNC = 0) | Synchronous (SYNC = 1) |
|---|---|---|---|
| 0 | 0 | 1 stop bit | 1 stop bit |
| 0 | 1 | 1.5 stop bits | Reserved |
| 1 | 0 | 2 stop bits | 2 stop bits |
| 1 | 1 | Reserved | Reserved |

- **CHMODE: Channel Mode**

| CHMODE | | Mode Description |
|---|---|---|
| 0 | 0 | Normal Mode |
| 0 | 1 | Automatic Echo. Receiver input is connected to the TXD pin. |
| 1 | 0 | Local Loopback. Transmitter output is connected to the Receiver Input.. |
| 1 | 1 | Remote Loopback. RXD pin is internally connected to the TXD pin. |

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16x Oversampling.

1: 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**

0: User defined configuration of command or data sync field depending on SYNC value.

1: The sync field is updated when a character is written into US_THR register.

- **MAX_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T= 0.

- **FILTER: Infrared Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

- **MAN: Manchester Encoder/Decoder Enable**

0: Manchester Encoder/Decoder are disabled.

1: Manchester Encoder/Decoder are enabled.

- **MODSYNC: Manchester Synchronization Mode**

0:The Manchester Start bit is a 0 to 1 transition

1: The Manchester Start bit is a 1 to 0 transition.

- **ONEBIT: Start Frame Delimiter Selector**

0: Start Frame delimiter is COMMAND or DATA SYNC.

1: Start Frame delimiter is One Bit.

### 30.7.3 USART Interrupt Enable Register

**Name:** US_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | MANE | CTSIC | DCDIC | DSRIC | RIIC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | NACK | RXBUFF | TXBUFE | ITER | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Transfer Interrupt Enable**
- **ENDTX: End of Transmit Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **RIIC: Ring Indicator Input Change Enable**
- **DSRIC: Data Set Ready Input Change Enable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

### 30.7.4 USART Interrupt Disable Register

**Name:** US_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | MANE | CTSIC | DCDIC | DSRIC | RIIC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | NACK | RXBUFF | TXBUFE | ITER | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **RIIC: Ring Indicator Input Change Disable**
- **DSRIC: Data Set Ready Input Change Disable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**
- **MANE: Manchester Error Interrupt Disable**

### 30.7.5 USART Interrupt Mask Register

**Name:** US_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | MANE | CTSIC | DCDIC | DSRIC | RIIC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | NACK | RXBUFF | TXBUFE | ITER | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Transfer Interrupt Mask**
- **ENDTX: End of Transmit Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**
- **DSRIC: Data Set Ready Input Change Mask**
- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**
- **MANE: Manchester Error Interrupt Mask**

### 30.7.6 USART Channel Status Register

**Name:** US_CSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | MANERR |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CTS | DCD | DSR | RI | CTSIC | DCDIC | DSRIC | RIIC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | NACK | RXBUFF | TXBUFE | ITER | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break**

0: No Break received or End of Break detected since the last RSTSTA.

1: Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0: The End of Transfer signal from the Receive PDC channel is inactive.

1: The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The End of Transfer signal from the Transmit PDC channel is inactive.

1: The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

- **TXEMPTY: Transmitter Empty**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **ITER: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSTSTA.

1: Maximum number of repetitions has been reached since the last RSTSTA.

- **TXBUFE: Transmission Buffer Empty**

0: The signal Buffer Empty from the Transmit PDC channel is inactive.

1: The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0: The signal Buffer Full from the Receive PDC channel is inactive.

1: The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0: No Non Acknowledge has not been detected since the last RSTNACK.

1: At least one Non Acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag**

0: No input change has been detected on the RI pin since the last read of US_CSR.

1: At least one input change has been detected on the RI pin since the last read of US_CSR.

- **DSRIC: Data Set Ready Input Change Flag**

0: No input change has been detected on the DSR pin since the last read of US_CSR.

1: At least one input change has been detected on the DSR pin since the last read of US_CSR.

- **DCDIC: Data Carrier Detect Input Change Flag**

0: No input change has been detected on the DCD pin since the last read of US_CSR.

1: At least one input change has been detected on the DCD pin since the last read of US_CSR.

- **CTSIC: Clear to Send Input Change Flag**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **RI: Image of RI Input**

0: RI is at 0.

1: RI is at 1.

- **DSR: Image of DSR Input**

0: DSR is at 0

1: DSR is at 1.

- **DCD: Image of DCD Input**

0: DCD is at 0.

1: DCD is at 1.

- **CTS: Image of CTS Input**

0: CTS is at 0.

1: CTS is at 1.

- **MANERR: Manchester Error**

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

### 30.7.7 USART Receive Holding Register

**Name:** US_RHR

**Access Type:** Read-only

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

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

### 30.7.8 USART Transmit Holding Register

**Name:** US_THR

**Access Type:** Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

### 30.7.9 USART Baud Rate Generator Register

**Name:** US_BRGR

**Access Type:** Read-write

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

- **CD: Clock Divider**

| CD | USART_MODE ≠ ISO7816 | | | USART_MODE = ISO7816 |
|---|---|---|---|---|
| | SYNC = 0 | | SYNC = 1 | |
| | OVER = 0 | OVER = 1 | | |
| 0 | Baud Rate Clock Disabled | | | |
| 1 to 65535 | Baud Rate = Selected Clock/16/CD | Baud Rate = Selected Clock/8/CD | Baud Rate = Selected Clock /CD | Baud Rate = Selected Clock/CD/FI_DI_RATIO |

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

### 30.7.10 USART Receiver Time-out Register

**Name:** US_RTOR

**Access Type:** Read-write

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

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

### 30.7.11 USART Transmitter Timeguard Register

**Name:** US_TTGR

**Access Type:** Read-write

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

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

### 30.7.12 USART FI DI RATIO Register

**Name:** US_FIDI

**Access Type:** Read-write

**Reset Value:** 0x174

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

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.

### 30.7.13 USART Number of Errors Register

**Name:** US_NER

**Access Type:** Read-only

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

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

### 30.7.14 USART IrDA FILTER Register

**Name:** US_IF

**Access Type:** Read-write

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

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

### 30.7.15 USART Manchester Configuration Register

**Name:** US_MAN

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | DRIFT | 1 | RX_MPOL | – | – | RX_PP | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RX_PL | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | TX_MPOL | – | – | TX_PP | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | TX_PL | | | |

- **TX_PL: Transmitter Preamble Length**

0: The Transmitter Preamble pattern generation is disabled

1 - 15: The Preamble Length is TX_PL x Bit Period

- **TX_PP: Transmitter Preamble Pattern**

| TX_PP | | Preamble Pattern default polarity assumed (TX_MPOL field not set) |
|---|---|---|
| 0 | 0 | ALL_ONE |
| 0 | 1 | ALL_ZERO |
| 1 | 0 | ZERO_ONE |
| 1 | 1 | ONE_ZERO |

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1 - 15: The detected preamble length is RX_PL x Bit Period

- **RX_PP: Receiver Preamble Pattern Detected**

| RX_PP | | Preamble Pattern default polarity assumed (RX_MPOL field not set) |
|---|---|---|
| 0 | 0 | ALL_ONE |
| 0 | 1 | ALL_ZERO |
| 1 | 0 | ZERO_ONE |
| 1 | 1 | ONE_ZERO |

- **RX_MPOL: Receiver Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **DRIFT: Drift Compensation**

0: The USART can not recover from an important clock drift

1: The USART can recover from clock drift. The 16X clock mode must be enabled.

# 31. Timer Counter (TC)

## 31.1 Overview

The Timer Counter (TC) includes three identical 16-bit Timer Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The Timer Counter block has two global registers which act upon all three TC channels.

The Block Control Register allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

Table 31-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2

**Table 31-1.** Timer Counter Clock Assignment

| Name | Definition |
|---|---|
| TIMER_CLOCK1 | Timerclock1 |
| TIMER_CLOCK2 | Timercock2 |
| TIMER_CLOCK3 | Timerclock3 |
| TIMER_CLOCK4 | Timerclock4 |
| TIMER_CLOCK5 | Timerclock5 |

## 31.2 Block Diagram

**Figure 31-1.** Timer Counter Block Diagram

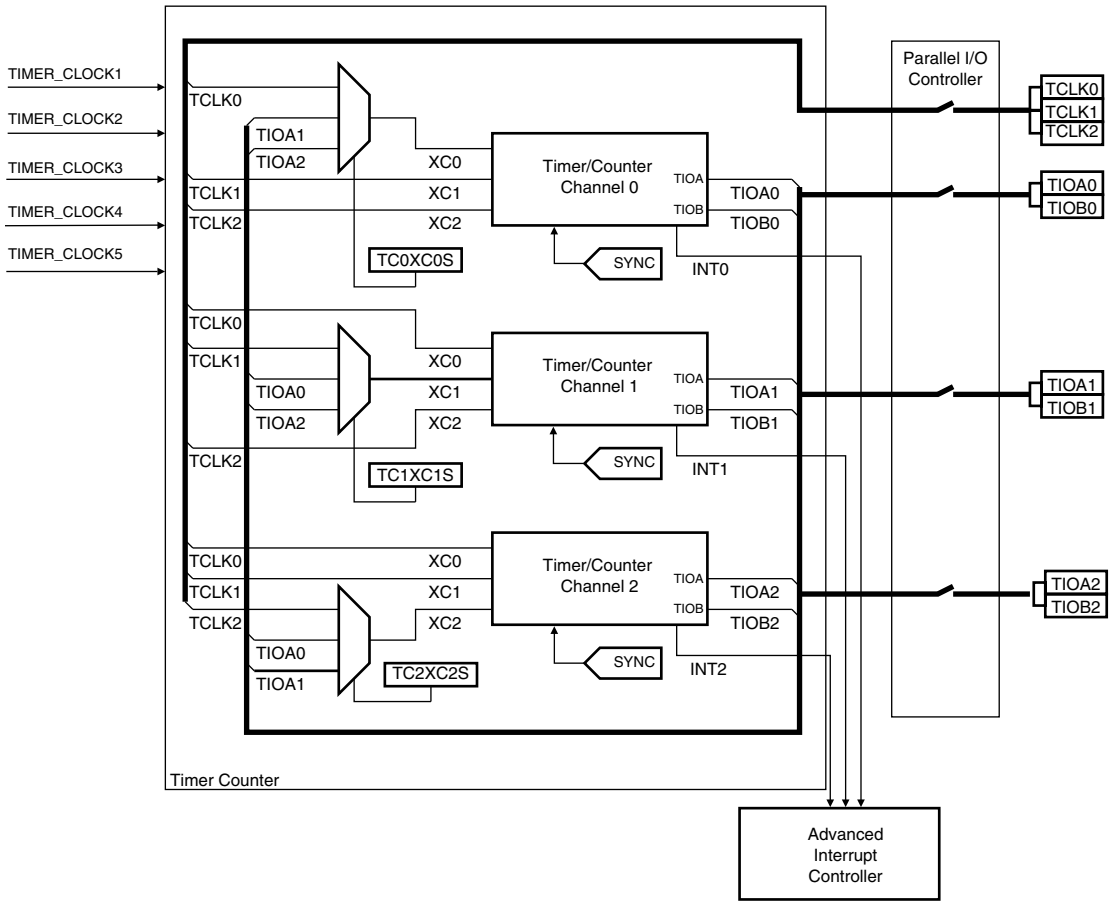



**Table 31-2.** Signal Name Description

| Block/Channel | Signal Name | Description |
|---|---|---|
| Channel Signal | XC0, XC1, XC2 | External Clock Inputs |
| | TIOA | Capture Mode: Timer Counter Input<br>Waveform Mode: Timer Counter Output |
| | TIOB | Capture Mode: Timer Counter Input<br>Waveform Mode: Timer Counter Input/Output |
| | INT | Interrupt Signal Output |
| | SYNC | Synchronization Input Signal |

## 31.3 Pin Name List

**Table 31-3.** TC pin list

| Pin Name | Description | Type |
|---|---|---|
| TCLK0-TCLK2 | External Clock Input | Input |
| TIOA0-TIOA2 | I/O Line A | I/O |
| TIOB0-TIOB2 | I/O Line B | I/O |

## 31.4 Product Dependencies

### 31.4.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

### 31.4.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter clock.

### 31.4.3 Interrupt

The TC has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the TC interrupt requires programming the AIC before configuring the TC.

## 31.5 Functional Description

### 31.5.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 31-4 on page 435.

### 31.5.2 16-bit Counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in TC_SR (Status Register) is set.

The current value of the counter is accessible in real time by reading the Counter Value Register, TC_CV. The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

### 31.5.3 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the internal I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block Mode). See Figure 31-2 on page 423.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5
- External clock signals: XC0, XC1 or XC2

This selection is made by the TCCLKS bits in the TC Channel Mode Register.

The selected clock can be inverted with the CLKI bit in TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2). See Figure 31-3 on page 423

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock

**Figure 31-2.** Clock Chaining Selection

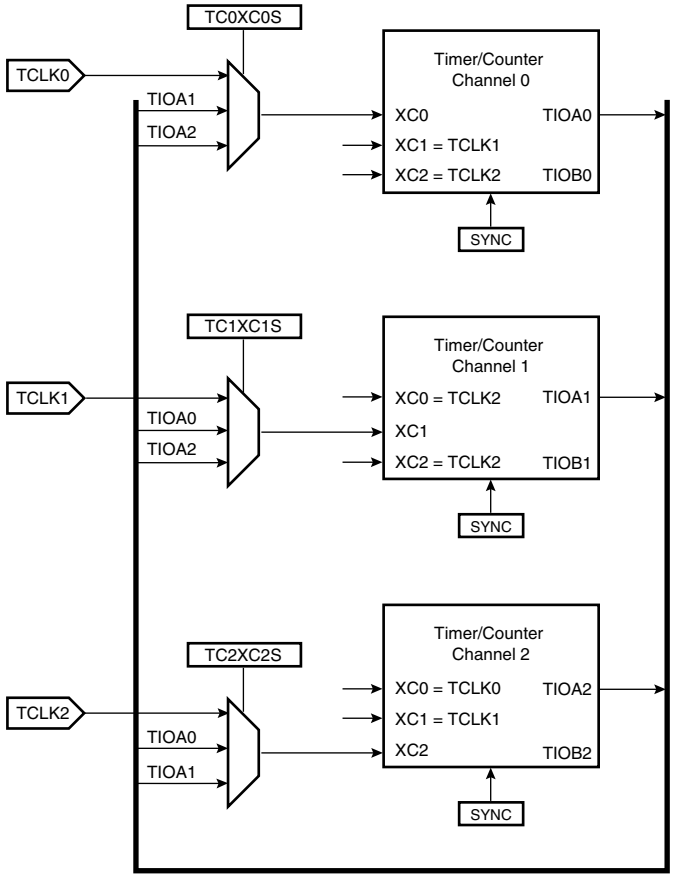
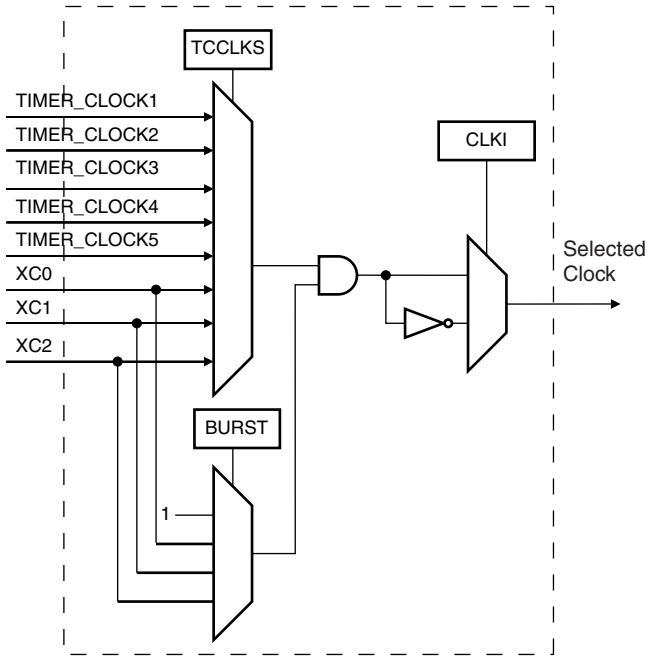



**Figure 31-3.** Clock Selection

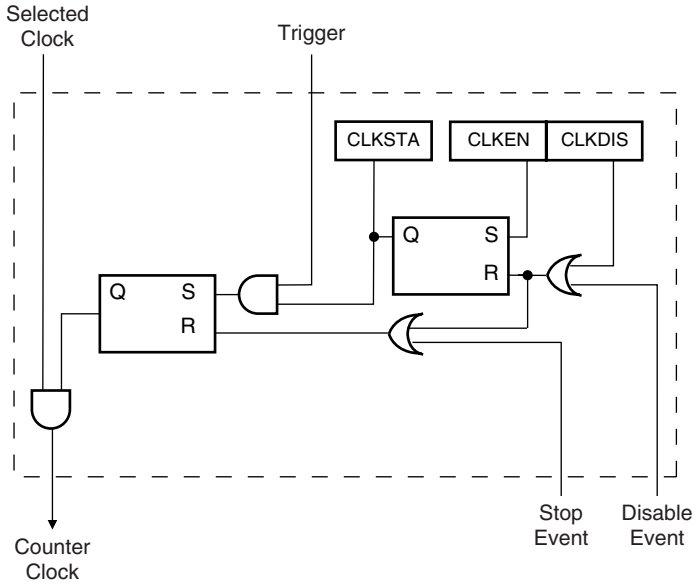


### 31.5.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 31-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in TC_CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode (LDBSTOP = 1 in TC_CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have effect only if the clock is enabled.

**Figure 31-4.** Clock Control




### 31.5.5 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

### 31.5.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in TC_CMR.

The channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRG in TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the master clock period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

### 31.5.7 Capture Operating Mode

This mode is entered by clearing the WAVE parameter in TC_CMR (Channel Mode Register).

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 31-5 shows the configuration of the TC channel when programmed in Capture Mode.

### 31.5.8 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA parameter in TC_CMR defines the TIOA edge for the loading of register A, and the LDRB parameter defines the TIOA edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

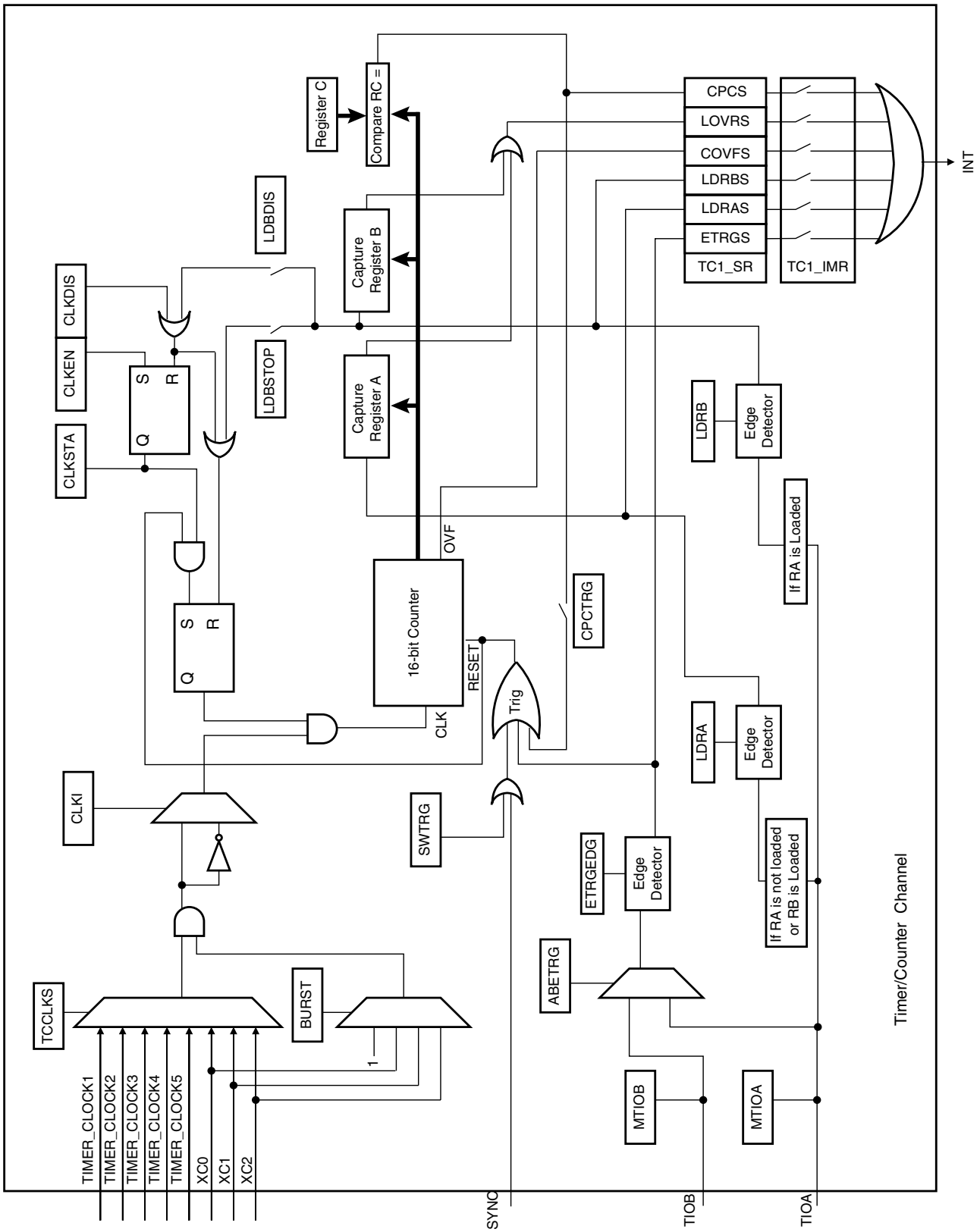
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

### 31.5.9 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in TC_CMR selects TIOA or TIOB input signal as an external trigger. The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

**Figure 31-5.** Capture Mode



### 31.5.10 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 31-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

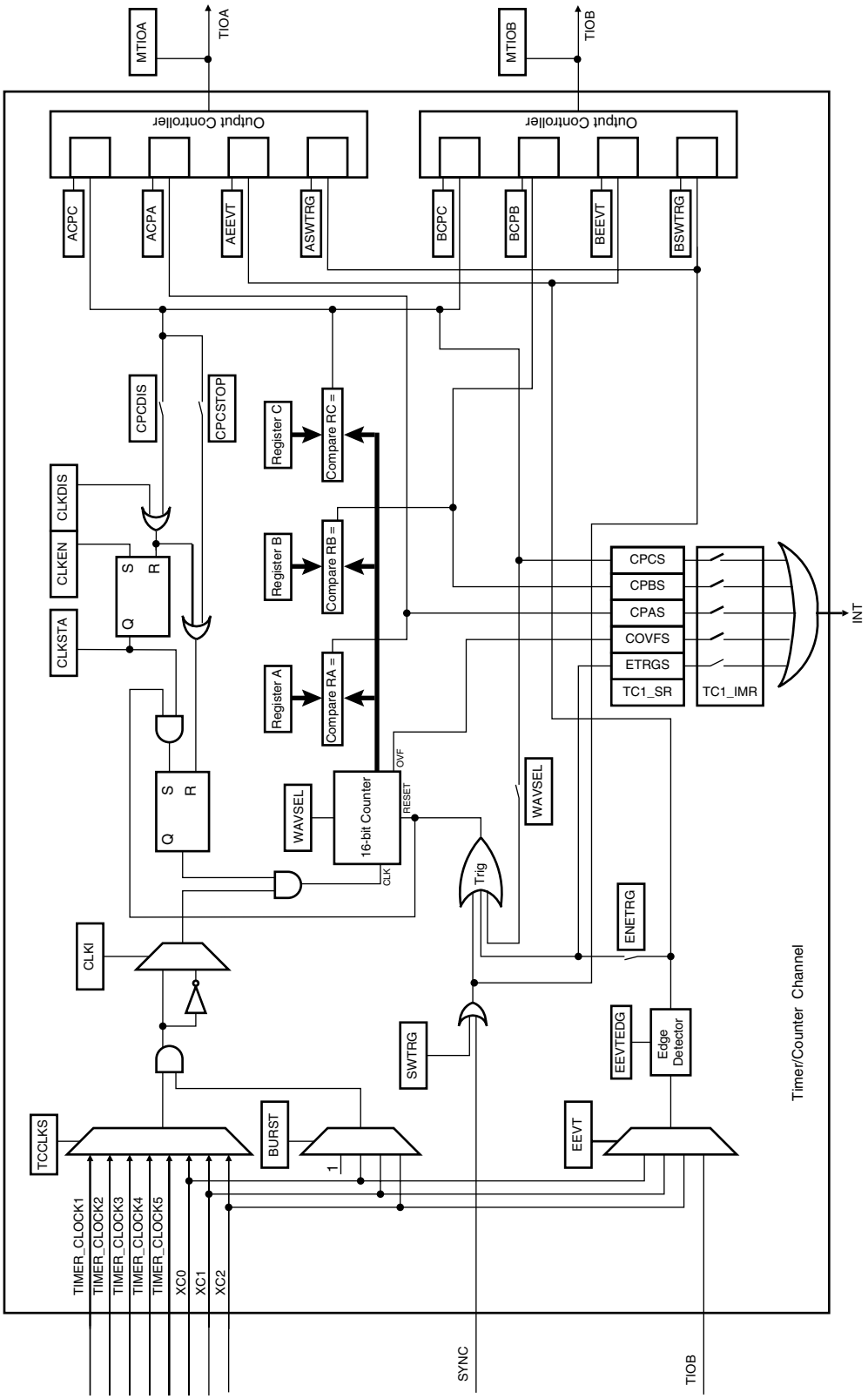
### 31.5.11 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

**Figure 31-6.** Waveform Mode



#### 31.5.11.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 31-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 31-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

**Figure 31-7.** WAVSEL= 00 without trigger

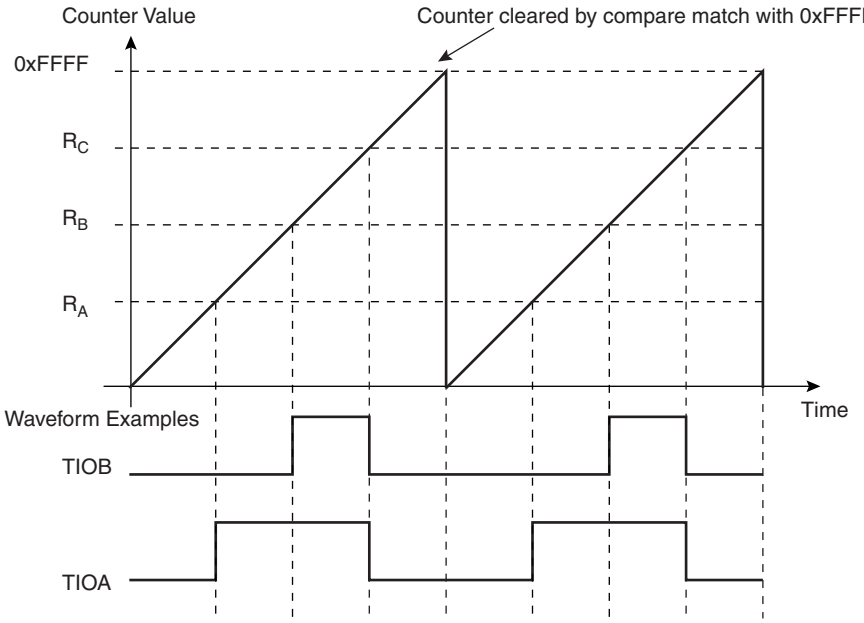
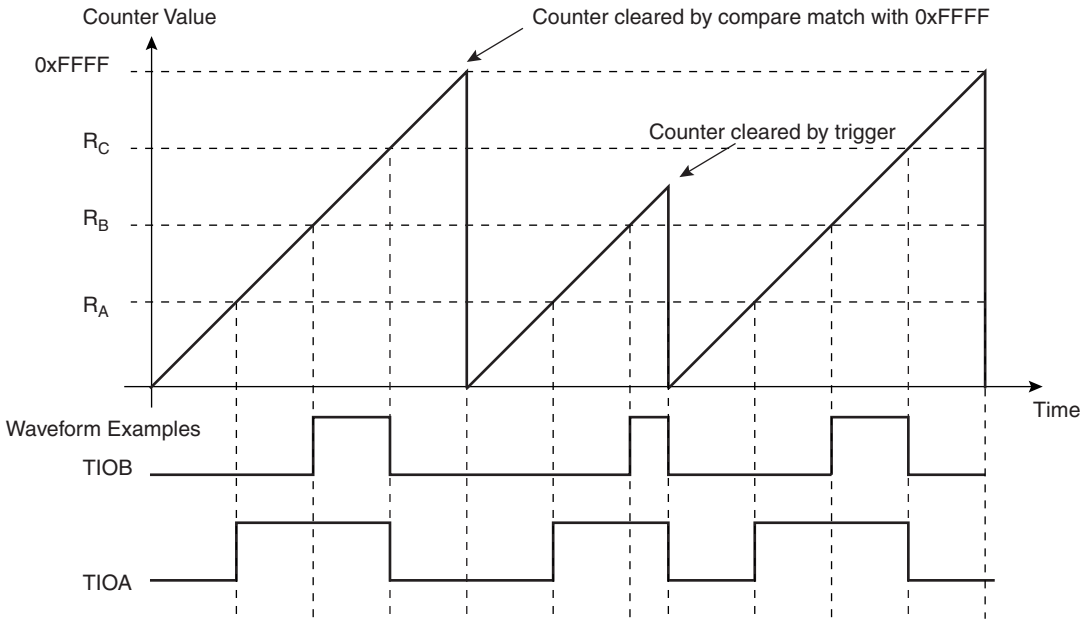


**Figure 31-8.** WAVSEL= 00 with trigger




### 31.5.11.2 *WAVSEL = 10*

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 31-9.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 31-10.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

**Figure 31-9.** WAVSEL = 10 Without Trigger

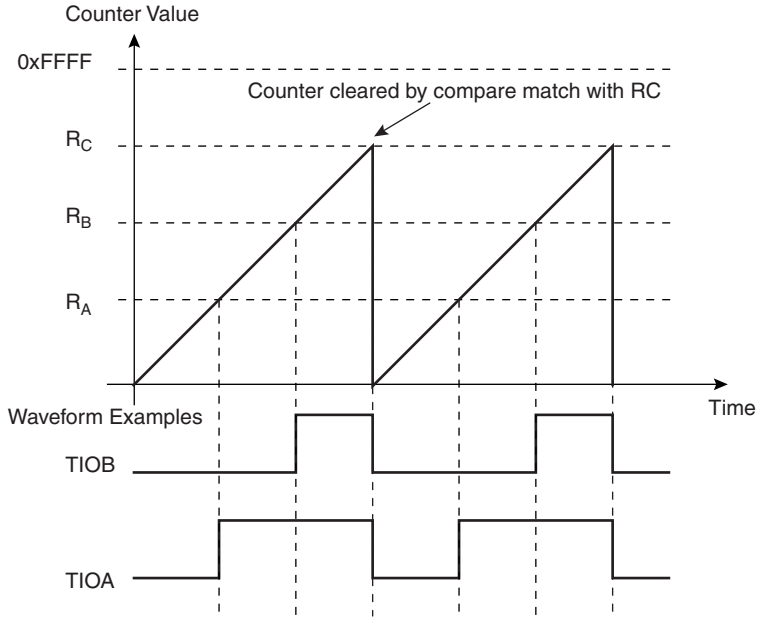
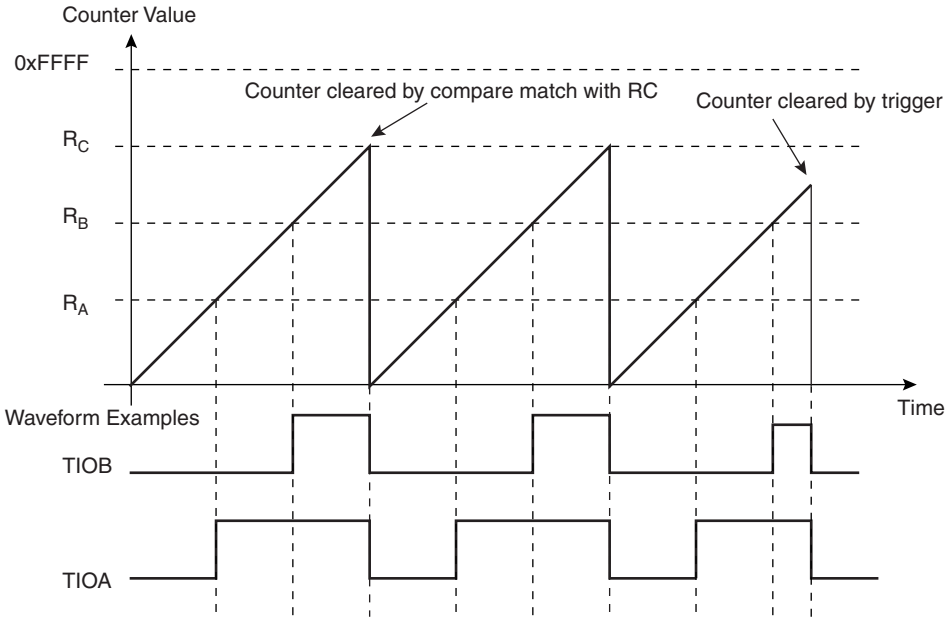


**Figure 31-10.** WAVSEL = 10 With Trigger




*31.5.11.3 WAVSEL = 01*

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 31-11.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 31-12.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

**Figure 31-11.** WAVSEL = 01 Without Trigger

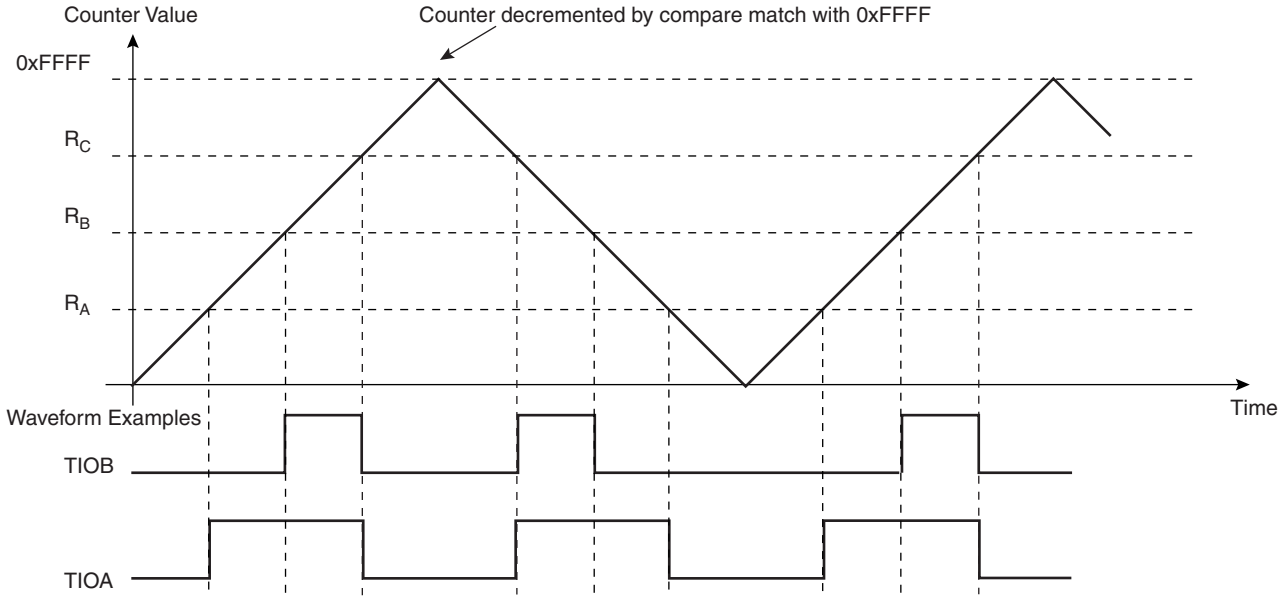
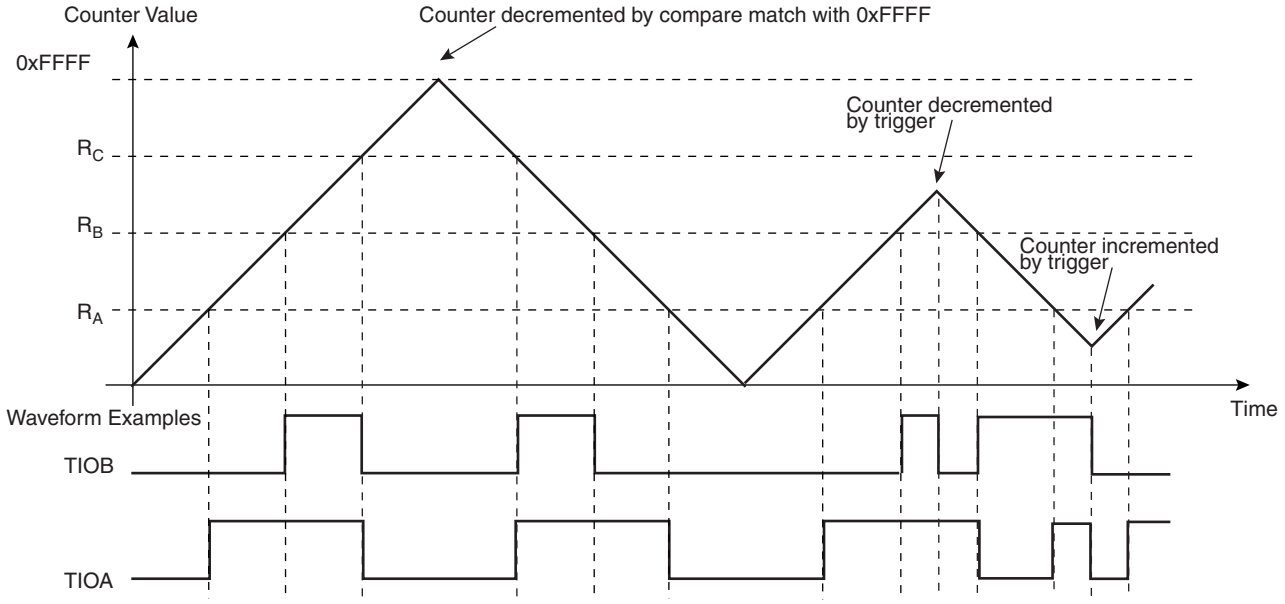



**Figure 31-12.** WAVSEL = 01 With Trigger




*31.5.11.4 WAVSEL = 11*

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 31-13.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 31-14.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

**Figure 31-13.** WAVSEL = 11 Without Trigger

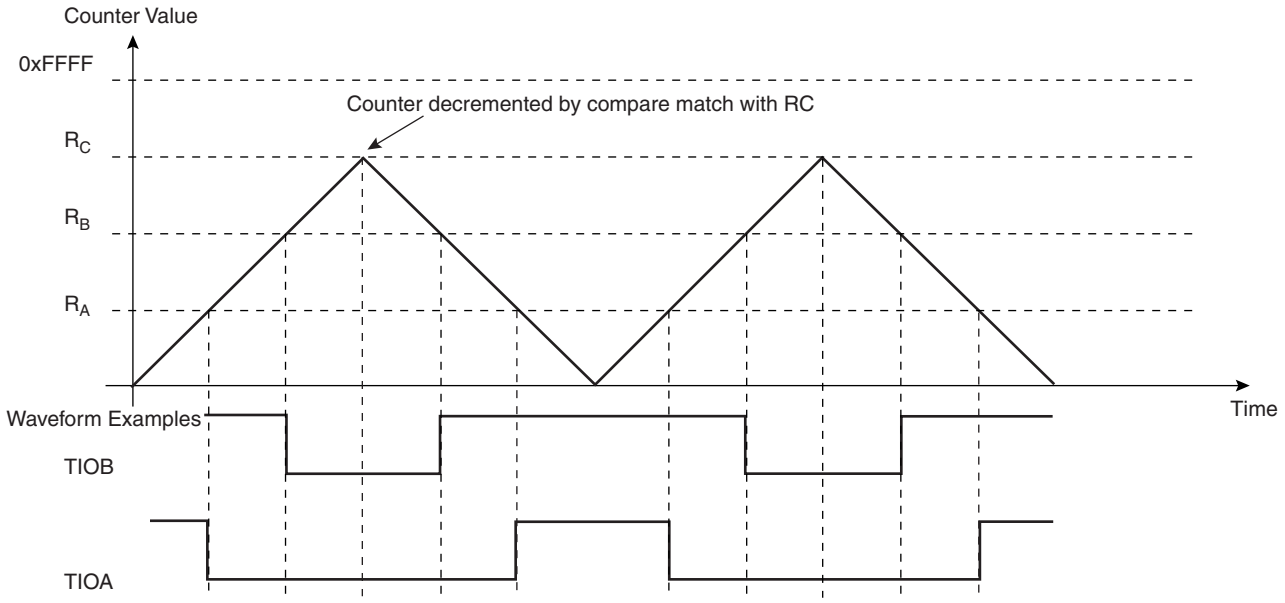
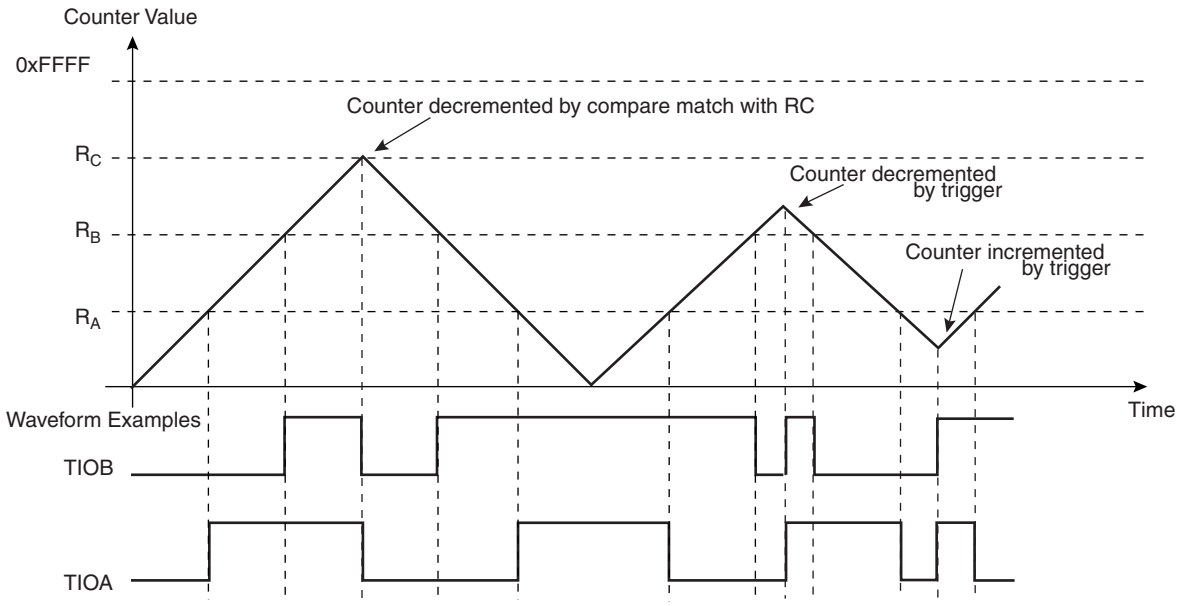



**Figure 31-14.** WAVSEL = 11 With Trigger



### 31.5.12 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in TC_CMR.

As in Capture Mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

### 31.5.13 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

## 31.6 Timer Counter (TC) User Interface

**Table 31-4.** Register Mapping

| Offset[1] | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 + channel * 0x40 + 0x00 | Channel Control Register | TC_CCR | Write-only | – |
| 0x00 + channel * 0x40 + 0x04 | Channel Mode Register | TC_CMR | Read-write | 0 |
| 0x00 + channel * 0x40 + 0x08 | Reserved | | | |
| 0x00 + channel * 0x40 + 0x0C | Reserved | | | |
| 0x00 + channel * 0x40 + 0x10 | Counter Value | TC_CV | Read-only | 0 |
| 0x00 + channel * 0x40 + 0x14 | Register A | TC_RA | Read-write[2] | 0 |
| 0x00 + channel * 0x40 + 0x18 | Register B | TC_RB | Read-write[2] | 0 |
| 0x00 + channel * 0x40 + 0x1C | Register C | TC_RC | Read-write | 0 |
| 0x00 + channel * 0x40 + 0x20 | Status Register | TC_SR | Read-only | 0 |
| 0x00 + channel * 0x40 + 0x24 | Interrupt Enable Register | TC_IER | Write-only | – |
| 0x00 + channel * 0x40 + 0x28 | Interrupt Disable Register | TC_IDR | Write-only | – |
| 0x00 + channel * 0x40 + 0x2C | Interrupt Mask Register | TC_IMR | Read-only | 0 |
| 0xC0 | Block Control Register | TC_BCR | Write-only | – |
| 0xC4 | Block Mode Register | TC_BMR | Read-write | 0 |
| 0xFC | Reserved | – | – | – |

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if WAVE = 0

### 31.6.1 TC Block Control Register

**Register Name:** TC_BCR

**Access Type:** Write-only

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

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

### 31.6.2 TC Block Mode Register

**Register Name:** TC_BMR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | TC2XC2S | | TC1XC1S | | TC0XC0S | |

- **TC0XC0S: External Clock Signal 0 Selection**

| TC0XC0S | | Signal Connected to XC0 |
|---|---|---|
| 0 | 0 | TCLK0 |
| 0 | 1 | none |
| 1 | 0 | TIOA1 |
| 1 | 1 | TIOA2 |

- **TC1XC1S: External Clock Signal 1 Selection**

| TC1XC1S | | Signal Connected to XC1 |
|---|---|---|
| 0 | 0 | TCLK1 |
| 0 | 1 | none |
| 1 | 0 | TIOA0 |
| 1 | 1 | TIOA2 |

- **TC2XC2S: External Clock Signal 2 Selection**

| TC2XC2S | | Signal Connected to XC2 |
|---|---|---|
| 0 | 0 | TCLK2 |
| 0 | 1 | none |
| 1 | 0 | TIOA0 |
| 1 | 1 | TIOA1 |

### 31.6.3 TC Channel Control Register

**Register Name:** TC_CCRx [x=0..2]

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | SWTRG | CLKDIS | CLKEN |

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

### 31.6.4 TC Channel Mode Register: Capture Mode

**Register Name:** TC_CMRx [x=0..2] (WAVE = 0)

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | LDRB | | LDRA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE | CPCTRG | – | – | – | ABETRG | ETRGEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LDBDIS | LDBSTOP | BURST | | CLKI | TCCLKS | | |

- **TCCLKS: Clock Selection**

| TCCLKS | | | Clock Selected |
|---|---|---|---|
| 0 | 0 | 0 | TIMER_CLOCK1 |
| 0 | 0 | 1 | TIMER_CLOCK2 |
| 0 | 1 | 0 | TIMER_CLOCK3 |
| 0 | 1 | 1 | TIMER_CLOCK4 |
| 1 | 0 | 0 | TIMER_CLOCK5 |
| 1 | 0 | 1 | XC0 |
| 1 | 1 | 0 | XC1 |
| 1 | 1 | 1 | XC2 |

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

| BURST | | |
|---|---|---|
| 0 | 0 | The clock is not gated by an external signal. |
| 0 | 1 | XC0 is ANDed with the selected clock. |
| 1 | 0 | XC1 is ANDed with the selected clock. |
| 1 | 1 | XC2 is ANDed with the selected clock. |

- **LDBSTOP: Counter Clock Stopped with RB Loading**

0 = Counter clock is not stopped when RB loading occurs.

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

0 = Counter clock is not disabled when RB loading occurs.

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

| ETRGEDG | | Edge |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | rising edge |
| 1 | 0 | falling edge |
| 1 | 1 | each edge |

- **ABETRG: TIOA or TIOB External Trigger Selection**

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

| LDRA | | Edge |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | rising edge of TIOA |
| 1 | 0 | falling edge of TIOA |
| 1 | 1 | each edge of TIOA |

- **LDRB: RB Loading Selection**

| LDRB | | Edge |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | rising edge of TIOA |
| 1 | 0 | falling edge of TIOA |
| 1 | 1 | each edge of TIOA |

### 31.6.5 TC Channel Mode Register: Waveform Mode

**Register Name:** TC_CMRx [x=0..2] (WAVE = 1)

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BSWTRG | | BEEVT | | BCPC | | BCPB | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ASWTRG | | AEEVT | | ACPC | | ACPA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE | WAVSEL | | ENETRG | EEVT | | EEVTEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPCDIS | CPCSTOP | BURST | | CLKI | TCCLKS | | |

- **TCCLKS: Clock Selection**

| TCCLKS | | | Clock Selected |
|---|---|---|---|
| 0 | 0 | 0 | TIMER_CLOCK1 |
| 0 | 0 | 1 | TIMER_CLOCK2 |
| 0 | 1 | 0 | TIMER_CLOCK3 |
| 0 | 1 | 1 | TIMER_CLOCK4 |
| 1 | 0 | 0 | TIMER_CLOCK5 |
| 1 | 0 | 1 | XC0 |
| 1 | 1 | 0 | XC1 |
| 1 | 1 | 1 | XC2 |

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

| BURST | | |
|---|---|---|
| 0 | 0 | The clock is not gated by an external signal. |
| 0 | 1 | XC0 is ANDed with the selected clock. |
| 1 | 0 | XC1 is ANDed with the selected clock. |
| 1 | 1 | XC2 is ANDed with the selected clock. |

- **CPCSTOP: Counter Clock Stopped with RC Compare**

0 = Counter clock is not stopped when counter reaches RC.

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

| EEVTEDG | | Edge |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | rising edge |
| 1 | 0 | falling edge |
| 1 | 1 | each edge |

- **EEVT: External Event Selection**

| EEVT | | Signal selected as external event | TIOB Direction |
|---|---|---|---|
| 0 | 0 | TIOB | input [1] |
| 0 | 1 | XC0 | output |
| 1 | 0 | XC1 | output |
| 1 | 1 | XC2 | output |

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs.*

- **ENETRG: External Event Trigger Enable**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

- **WAVSEL: Waveform Selection**

| WAVSEL | | Effect |
|---|---|---|
| 0 | 0 | UP mode without automatic trigger on RC Compare |
| 1 | 0 | UP mode with automatic trigger on RC Compare |
| 0 | 1 | UPDOWN mode without automatic trigger on RC Compare |
| 1 | 1 | UPDOWN mode with automatic trigger on RC Compare |

- **WAVE**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

| ACPA | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **ACPC: RC Compare Effect on TIOA**

| ACPC | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **AEEVT: External Event Effect on TIOA**

| AEEVT | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **ASWTRG: Software Trigger Effect on TIOA**

| ASWTRG | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **BCPB: RB Compare Effect on TIOB**

| BCPB | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **BCPC: RC Compare Effect on TIOB**

| BCPC | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **BEEVT: External Event Effect on TIOB**

| BEEVT | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

- **BSWTRG: Software Trigger Effect on TIOB**

| BSWTRG | | Effect |
|---|---|---|
| 0 | 0 | none |
| 0 | 1 | set |
| 1 | 0 | clear |
| 1 | 1 | toggle |

### 31.6.6 TC Counter Value Register

**Register Name:** TC_CVx [x=0..2]

**Access Type:** Read-only

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

- **CV: Counter Value**

CV contains the counter value in real time.

### 31.6.7 TC Register A

**Register Name:** TC_RAx [x=0..2]

**Access Type:** Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RA: Register A**

RA contains the Register A value in real time.

### 31.6.8 TC Register B

**Register Name:** TC_RBx [x=0..2]

**Access Type:** Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RB: Register B**

RB contains the Register B value in real time.

### 31.6.9 TC Register C

**Register Name:** TC_RCx [x=0..2]

**Access Type:** Read-write

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

- **RC: Register C**

RC contains the Register C value in real time.

### 31.6.10 TC Status Register

**Register Name:** TC_SRx [x=0..2]

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | MTIOB | MTIOA | CLKSTA |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow Status**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

### 31.6.11 TC Interrupt Enable Register

**Register Name:** TC_IERx [x=0..2]

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.

### 31.6.12 TC Interrupt Disable Register

**Register Name:** TC_IDRx [x=0..2]

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

### 31.6.13 TC Interrupt Mask Register

**Register Name:** TC_IMRx [x=0..2]

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.

# 32. Pulse Width Modulation Controller (PWM)

## 32.1 Overview

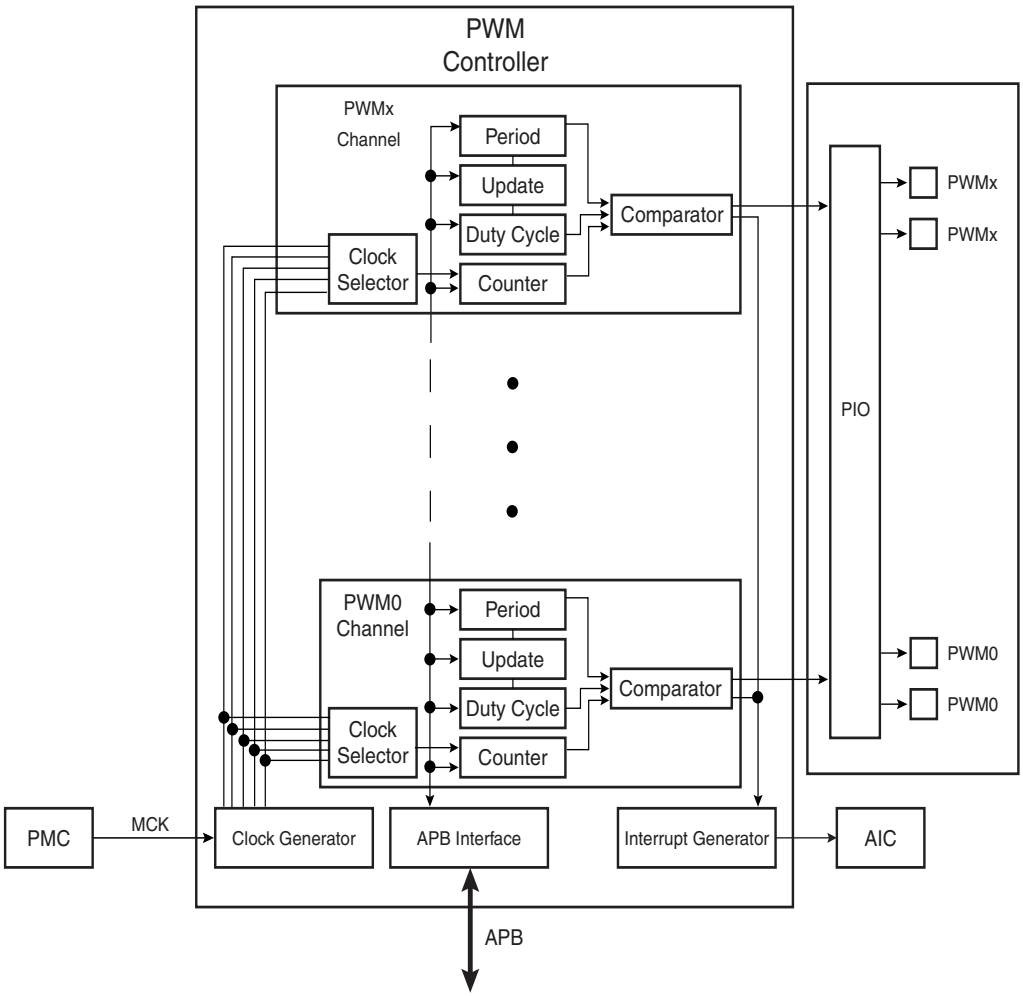
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through APB mapped registers.

Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

## 32.2 Block Diagram

**Figure 32-1.** Pulse Width Modulation Controller Block Diagram



## 32.3 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

**Table 32-1.** I/O Line Description

| Name | Description | Type |
|---|---|---|
| PWMx | PWM Waveform Output for channel x | Output |

## 32.4 Product Dependencies

### 32.4.1 I/O Lines

The pins used for interfacing the PWM may be multiplexed with PIO lines. The programmer must first program the PIO controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the PIO controller.

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

### 32.4.2 Power Management

The PWM is not continuously clocked. The programmer must first enable the PWM clock in the Power Management Controller (PMC) before using the PWM. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

Configuring the PWM does not require the PWM clock to be enabled.

### 32.4.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the PWM interrupt requires the AIC to be programmed first. Note that it is not recommended to use the PWM interrupt line in edge sensitive mode.

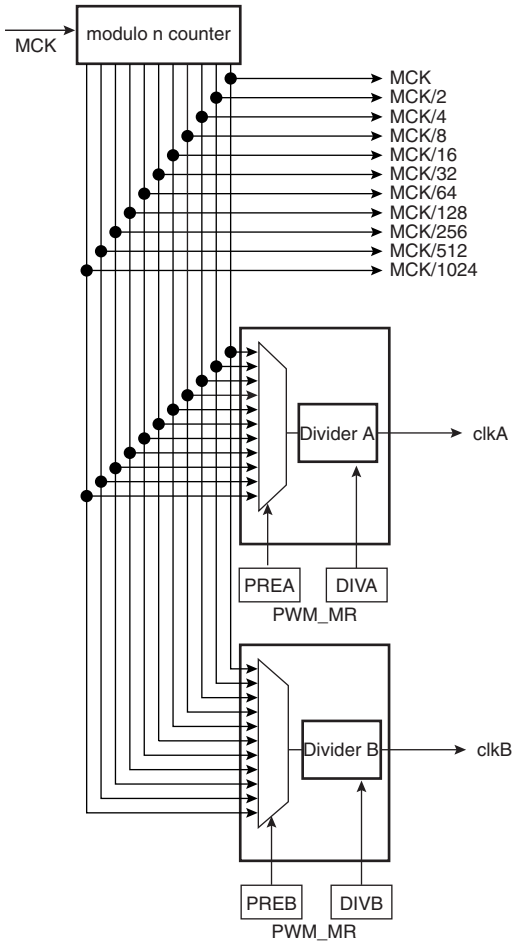
## 32.5 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the system clock, MCK, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

### 32.5.1 PWM Clock Generator

**Figure 32-2.** Functional View of the Clock Generator Block Diagram




**Caution:** Before using the PWM macrocell, the programmer must first enable the PWM clock in the Power Management Controller (PMC).

The PWM macrocell master clock, MCK, is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The clock generator is divided in three blocks:

- a modulo n counter which provides 11 clocks: $F_{MCK}$, $F_{MCK}/2$, $F_{MCK}/4$, $F_{MCK}/8$, $F_{MCK}/16$, $F_{MCK}/32$, $F_{MCK}/64$, $F_{MCK}/128$, $F_{MCK}/256$, $F_{MCK}/512$, $F_{MCK}/1024$
- two linear dividers (1, 1/2, 1/3, ... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Mode register (PWM_MR). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value in the PWM Mode register (PWM_MR).

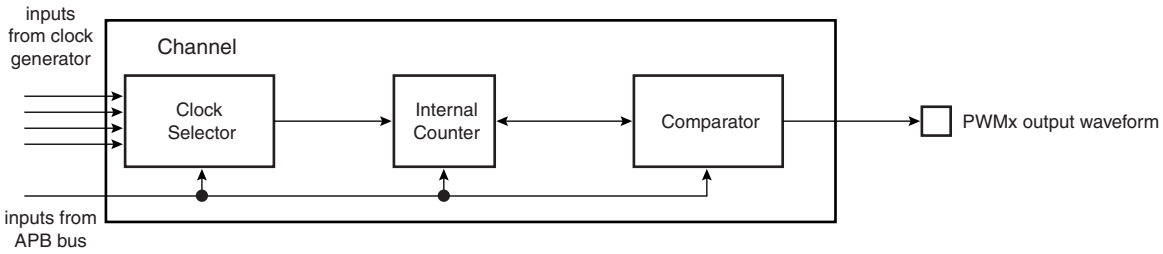
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except clock “clk”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

### 32.5.2 PWM Channel

#### *32.5.2.1 Block Diagram*

**Figure 32-3.** Functional View of the Channel Block Diagram




Each of the 4 channels is composed of three blocks:

- A clock selector which selects one of the clocks provided by the clock generator described in Section 32.5.1 “PWM Clock Generator” on page 455.
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 16 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

#### *32.5.2.2 Waveform Properties*

The different properties of output waveforms are:

- the ***internal clock selection***. The internal channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the PWM_CMRx register. This field is reset at 0.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the PWM_CPRDx register.
  - If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the Master Clock (MCK) divided by an X given prescaler value
  (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

  By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
By using the Master Clock (MCK) divided by an X given prescaler value
(with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- the ***waveform duty cycle***. This channel parameter is defined in the CDTY field of the PWM_CDTYx register.
  If the waveform is left aligned then:

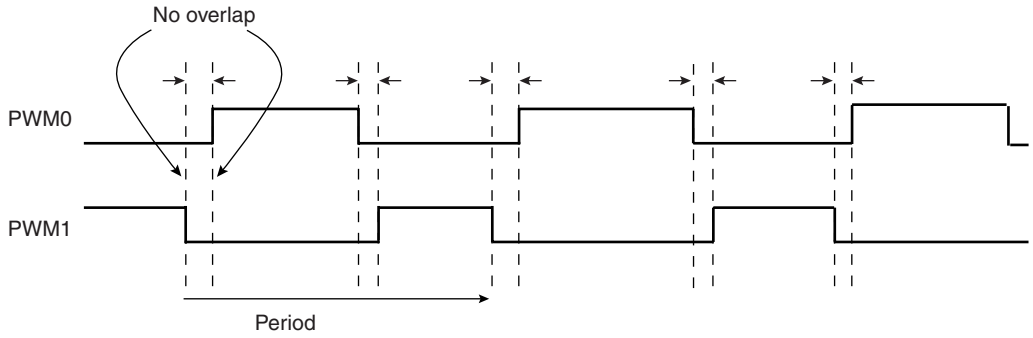
$$\text{duty cycle} = \frac{(period - 1 / \text{fchannel_x_clock} \times CDTY)}{period}$$

  If the waveform is center aligned, then:

$$\text{duty cycle} = \frac{((period/2) - 1 / \text{fchannel_x_clock} \times CDTY))}{(period/2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the PWM_CMRx register. By default the signal starts by a low level.
- the ***waveform alignment***. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the PWM_CMRx register. The default mode is left aligned.

**Figure 32-4.** Non Overlapped Center Aligned Waveforms




Note: 1. See Figure 32-5 on page 459 for a detailed description of center aligned waveforms.

When center aligned, the internal channel counter increases up to CPRD and.decreases down to 0. This ends the period.

When left aligned, the internal channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

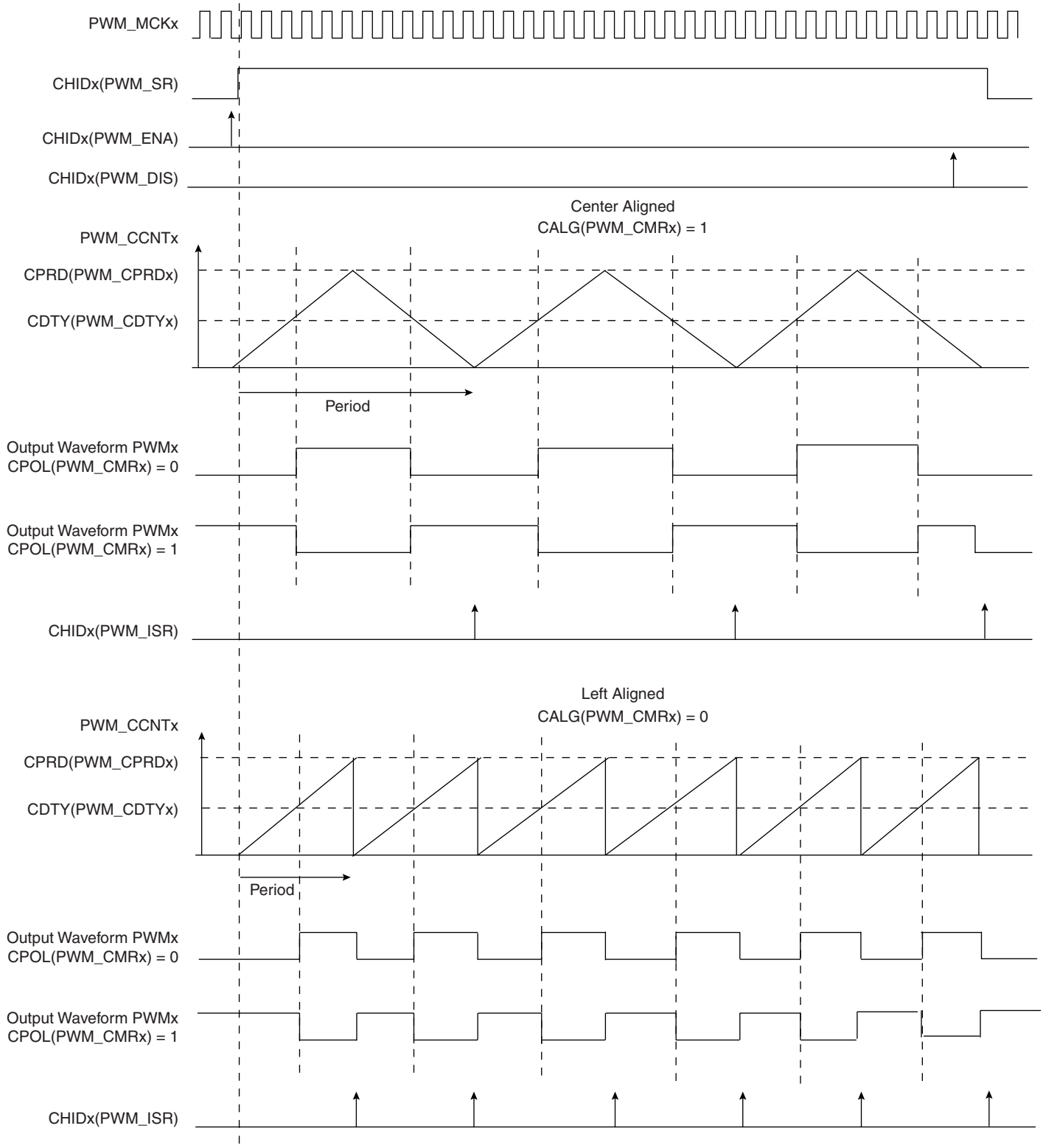
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

**Figure 32-5.** Waveform Properties



### 32.5.3 PWM Controller Operations

#### *32.5.3.1 Initialization*

Before enabling the output channel, this channel must have been configured by the software application:

- Configuration of the clock generator if DIVA and DIVB are required
- Selection of the clock for each channel (CPRE field in the PWM_CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the PWM_CMRx register)
- Configuration of the period for each channel (CPRD in the PWM_CPRDx register). Writing in PWM_CPRDx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CPRDx as explained below.
- Configuration of the duty cycle for each channel (CDTY in the PWM_CDTYx register). Writing in PWM_CDTYx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CUPDx Register to update PWM_CDTYx as explained below.
- Configuration of the output waveform polarity for each channel (CPOL in the PWM_CMRx register)
- Enable Interrupts (Writing CHIDx in the PWM_IER register)
- Enable the PWM channel (Writing CHIDx in the PWM_ENA register)

It is possible to synchronize different channels by enabling them at the same time by means of writing simultaneously several CHIDx bits in the PWM_ENA register.

- In such a situation, all channels may have the same clock selector configuration and the same period specified.

#### *32.5.3.2 Source Clock Selection Criteria*

The large number of source clocks can make selection difficult. The relationship between the value in the Period Register (PWM_CPRDx) and the Duty Cycle Register (PWM_CDTYx) can help the user in choosing. The event number written in the Period Register gives the PWM accuracy. The Duty Cycle quantum cannot be lower than *1/PWM_CPRDx* value. The higher the value of PWM_CPRDx, the greater the PWM accuracy.

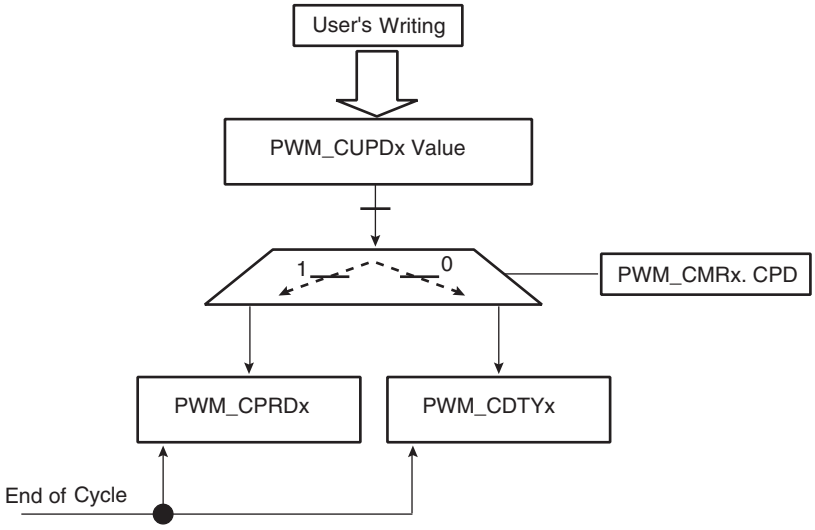
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

#### *32.5.3.3 Changing the Duty Cycle or the Period*

It is possible to modulate the output waveform duty cycle or period.

To prevent unexpected output waveform, the user must use the update register (PWM_CUPDx) to change waveform parameters while the channel is still enabled. The user can write a new period value or duty cycle value in the update register (PWM_CUPDx). This register holds the new value until the end of the current cycle and updates the value for the next cycle. Depending on the CPD field in the PWM_CMRx register, PWM_CUPDx either updates PWM_CPRDx or PWM_CDTYx. Note that even if the update register is used, the period must not be smaller than the duty cycle.

**Figure 32-6.** Synchronized Period or Duty Cycle Update



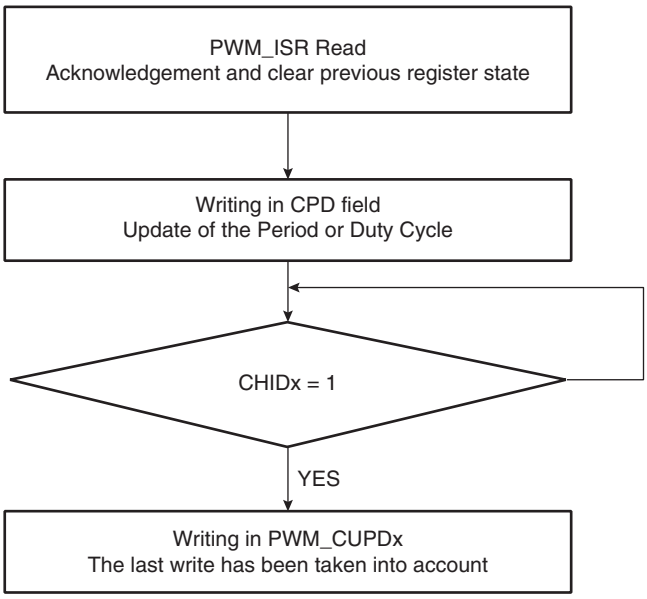

To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 32-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

**Figure 32-7.** Polling Method




Note: Polarity and alignment can be modified only when the channel is disabled.

#### 32.5.3.4 Interrupts

Depending on the interrupt mask in the PWM_IMR register, an interrupt is generated at the end of the corresponding channel period. The interrupt remains active until a read operation in the PWM_ISR register occurs.

A channel interrupt is enabled by setting the corresponding bit in the PWM_IER register. A channel interrupt is disabled by setting the corresponding bit in the PWM_IDR register.

## 32.6 Pulse Width Modulation Controller (PWM) User Interface

**Table 32-2.** Register Mapping[2]

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | PWM Mode Register | PWM_MR | Read-write | 0 |
| 0x04 | PWM Enable Register | PWM_ENA | Write-only | - |
| 0x08 | PWM Disable Register | PWM_DIS | Write-only | - |
| 0x0C | PWM Status Register | PWM_SR | Read-only | 0 |
| 0x10 | PWM Interrupt Enable Register | PWM_IER | Write-only | - |
| 0x14 | PWM Interrupt Disable Register | PWM_IDR | Write-only | - |
| 0x18 | PWM Interrupt Mask Register | PWM_IMR | Read-only | 0 |
| 0x1C | PWM Interrupt Status Register | PWM_ISR | Read-only | 0 |
| 0x100 - 0x1FC | Reserved | | | |
| 0x200 + ch_num * 0x20 + 0x00 | PWM Channel Mode Register | PWM_CMR | Read-write | 0x0 |
| 0x200 + ch_num * 0x20 + 0x04 | PWM Channel Duty Cycle Register | PWM_CDTY | Read-write | 0x0 |
| 0x200 + ch_num * 0x20 + 0x08 | PWM Channel Period Register | PWM_CPRD | Read-write | 0x0 |
| 0x200 + ch_num * 0x20 + 0x0C | PWM Channel Counter Register | PWM_CCNT | Read-only | 0x0 |
| 0x200 + ch_num * 0x20 + 0x10 | PWM Channel Update Register | PWM_CUPD | Write-only | - |

2. Some registers are indexed with “ch_num” index ranging from 0 to 3.

### 32.6.1 PWM Mode Register

**Register Name:** PWM_MR

**Access Type:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | PREB | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DIVB | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | PREA | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DIVA | | | | | | | |

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

| DIVA, DIVB | CLKA, CLKB |
|---|---|
| 0 | CLKA, CLKB clock is turned off |
| 1 | CLKA, CLKB clock is clock selected by PREA, PREB |
| 2-255 | CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor. |

- **PREA, PREB**

| PREA, PREB | | | | Divider Input Clock |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | MCK. |
| 0 | 0 | 0 | 1 | MCK/2 |
| 0 | 0 | 1 | 0 | MCK/4 |
| 0 | 0 | 1 | 1 | MCK/8 |
| 0 | 1 | 0 | 0 | MCK/16 |
| 0 | 1 | 0 | 1 | MCK/32 |
| 0 | 1 | 1 | 0 | MCK/64 |
| 0 | 1 | 1 | 1 | MCK/128 |
| 1 | 0 | 0 | 0 | MCK/256 |
| 1 | 0 | 0 | 1 | MCK/512 |
| 1 | 0 | 1 | 0 | MCK/1024 |
| Other | | | | Reserved |

### 32.6.2 PWM Enable Register

**Register Name:** PWM_ENA

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

### 32.6.3 PWM Disable Register

**Register Name:** PWM_DIS

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.

### 32.6.4 PWM Status Register

**Register Name:** PWM_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

### 32.6.5 PWM Interrupt Enable Register

**Register Name:** PWM_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

### 32.6.6 PWM Interrupt Disable Register

**Register Name:** PWM_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

### 32.6.7 PWM Interrupt Mask Register

**Register Name:** PWM_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

### 32.6.8 PWM Interrupt Status Register

**Register Name:** PWM_ISR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Channel ID**

0 = No new channel period has been achieved since the last read of the PWM_ISR register.

1 = At least one new channel period has been achieved since the last read of the PWM_ISR register.

Note: Reading PWM_ISR automatically clears CHIDx flags.

### 32.6.9 PWM Channel Mode Register

**Register Name:** PWM_CMR[0..3]

**Access Type:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | CPD | CPOL | CALG |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | CPRE | | | |

- **CPRE: Channel Pre-scaler**

| CPRE | | | | Channel Pre-scaler |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | MCK |
| 0 | 0 | 0 | 1 | MCK/2 |
| 0 | 0 | 1 | 0 | MCK/4 |
| 0 | 0 | 1 | 1 | MCK/8 |
| 0 | 1 | 0 | 0 | MCK/16 |
| 0 | 1 | 0 | 1 | MCK/32 |
| 0 | 1 | 1 | 0 | MCK/64 |
| 0 | 1 | 1 | 1 | MCK/128 |
| 1 | 0 | 0 | 0 | MCK/256 |
| 1 | 0 | 0 | 1 | MCK/512 |
| 1 | 0 | 1 | 0 | MCK/1024 |
| 1 | 0 | 1 | 1 | CLKA |
| 1 | 1 | 0 | 0 | CLKB |
| Other | | | | Reserved |

- **CALG: Channel Alignment**

0 = The period is left aligned.

1 = The period is center aligned.

- **CPOL: Channel Polarity**

0 = The output waveform starts at a low level.

1 = The output waveform starts at a high level.

- **CPD: Channel Update Period**

0 = Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the PWM_CUPDx will modify the period at the next period start event.

### 32.6.10 PWM Channel Duty Cycle Register

**Register Name:** PWM_CDTY[0..3]

**Access Type:** Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

### 32.6.11 PWM Channel Period Register

**Register Name:** PWM_CPRD[0..3]

**Access Type:** Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CPRD \times DIVA)}{MCK} \text{ or } \frac{(CPRD \times DIVAB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

# 33. Analog-to-Digital Converter (ADC)

## 33.1 Overview

The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates a 4-to-1 analog multiplexer, making possible the analog-to-digital conversions of 4 analog lines. The conversions extend from 0V to ADVREF.

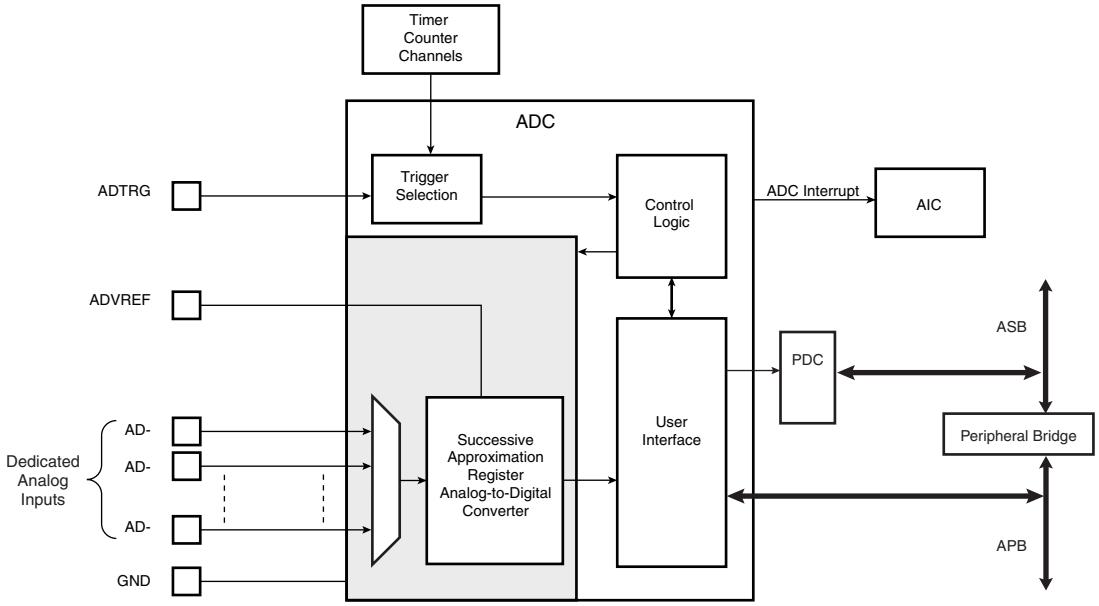
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

## 33.2 Block Diagram

**Figure 33-1.** Analog-to-Digital Converter Block Diagram



## 33.3 Signal Description

**Table 33-1.** ADC Pin Description

| Pin Name | Description |
|---|---|
| ADVREF | Reference voltage |
| AD0 - AD3 | Analog input channels |
| ADTRG | External trigger |

## 33.4 Product Dependencies

### 33.4.1 Power Management

The ADC is automatically clocked after the first conversion in Normal Mode. In Sleep Mode, the ADC clock is automatically stopped after each conversion. As the logic is small and the ADC cell can be put into Sleep Mode, the Power Management Controller has no effect on the ADC behavior.

### 33.4.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the ADC interrupt requires the AIC to be programmed first.

### 33.4.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

### 33.4.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

### 33.4.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be non-connected.

### 33.4.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

## 33.5 Functional Description

### 33.5.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Sample and Hold Clock cycles as defined in the field SHTIM of the “ADC Mode Register” on page 482 and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR).

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/128, if PRESCAL is set to 63 (0x3F). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the Product definition section.

### 33.5.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF.Analog inputs between these voltages convert to values based on a linear conversion.

### 33.5.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the bit LOWRES, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

Moreover, when a PDC channel is connected to the ADC, 10-bit resolution sets the transfer request sizes to 16-bit. Setting the bit LOWRES automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

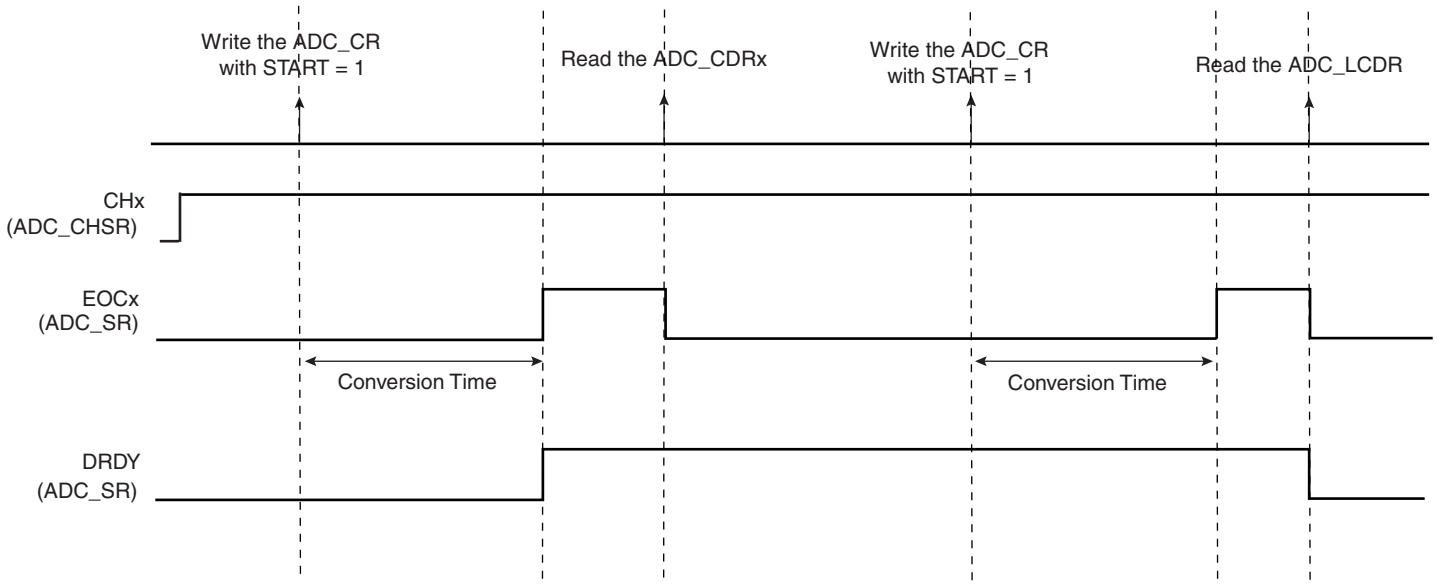
### 33.5.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDR) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR).

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected PDC channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

**Figure 33-2.** EOCx and DRDY Flag Behavior

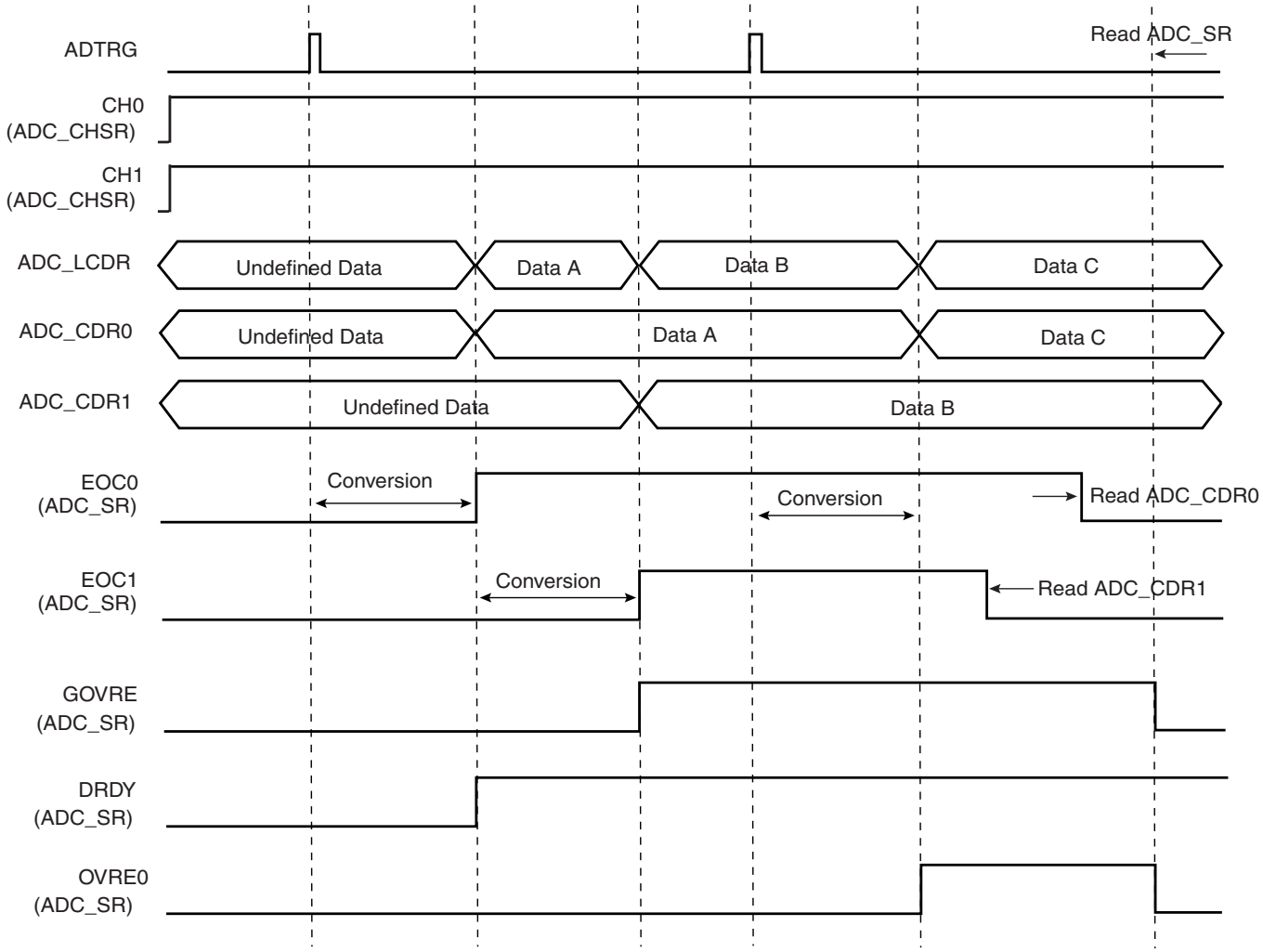



If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in ADC_SR.

The OVRE and GOVRE flags are automatically cleared when ADC_SR is read.

**Figure 33-3.** GOVRE and OVREx Flag Behavior




**Warning:** If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

### 33.5.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the bit START at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the field TRGSEL in the Mode Register (ADC_MR). The selected hardware trigger is enabled with the bit TRGEN in the Mode Register (ADC_MR).

If a hardware trigger is selected, the start of a conversion is detected at each rising edge of the selected signal. If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a PDC, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

**Warning:** Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can be initiated either by the hardware or the software trigger.

### 33.5.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the bit SLEEP in the Mode Register ADC_MR.

The SLEEP mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using a Timer/Counter output. The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the PDC.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

### 33.5.7 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register ADC_MR.

In the same way, a minimal Sample and Hold Time is necessary for the ADC to guarantee the best converted final value between two channels selection. This time has to be programmed through the SHTIM bitfield in the Mode Register ADC_MR.

**Warning:** No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the SHTIM field. See the section ADC Characteristics in the product datasheet.

## 33.6 Analog-to-Digital Converter (ADC) User Interface

**Table 33-2.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | ADC_CR | Write-only | – |
| 0x04 | Mode Register | ADC_MR | Read-write | 0x00000000 |
| 0x08 | Reserved | – | – | – |
| 0x0C | Reserved | – | – | – |
| 0x10 | Channel Enable Register | ADC_CHER | Write-only | – |
| 0x14 | Channel Disable Register | ADC_CHDR | Write-only | – |
| 0x18 | Channel Status Register | ADC_CHSR | Read-only | 0x00000000 |
| 0x1C | Status Register | ADC_SR | Read-only | 0x000C0000 |
| 0x20 | Last Converted Data Register | ADC_LCDR | Read-only | 0x00000000 |
| 0x24 | Interrupt Enable Register | ADC_IER | Write-only | – |
| 0x28 | Interrupt Disable Register | ADC_IDR | Write-only | – |
| 0x2C | Interrupt Mask Register | ADC_IMR | Read-only | 0x00000000 |
| 0x30 | Channel Data Register 0 | ADC_CDR0 | Read-only | 0x00000000 |
| 0x34 | Channel Data Register 1 | ADC_CDR1 | Read-only | 0x00000000 |
| ... | ... | ... | ... | ... |
| 0x4C | Channel Data Register 3 | ADC_CDR3 | Read-only | 0x00000000 |
| 0x50 - 0xFC | Reserved | – | – | – |

### 33.6.1 ADC Control Register

**Register Name:** ADC_CR

**Access Type:** Write-only

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

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

### 33.6.2 ADC Mode Register

**Register Name:** ADC_MR

**Access Type:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | SHTIM | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | STARTUP | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PRESCAL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | SLEEP | LOWRES | TRGSEL | | | TRGEN |

- **TRGEN: Trigger Enable**

| TRGEN | Selected TRGEN |
|---|---|
| 0 | Hardware triggers are disabled. Starting a conversion is only possible by software. |
| 1 | Hardware trigger selected by TRGSEL field is enabled. |

- **TRGSEL: Trigger Selection**

| TRGSEL | | | Selected TRGSEL |
|---|---|---|---|
| 0 | 0 | 0 | TIOA Ouput of the Timer Counter Channel 0 |
| 0 | 0 | 1 | TIOA Ouput of the Timer Counter Channel 1 |
| 0 | 1 | 0 | TIOA Ouput of the Timer Counter Channel 2 |
| 0 | 1 | 1 | Reserved |
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | Reserved |
| 1 | 1 | 0 | External trigger |
| 1 | 1 | 1 | Reserved |

- **LOWRES: Resolution**

| LOWRES | Selected Resolution |
|---|---|
| 0 | 10-bit resolution |
| 1 | 8-bit resolution |

- **SLEEP: Sleep Mode**

| SLEEP | Selected Mode |
|---|---|
| 0 | Normal Mode |
| 1 | Sleep Mode |

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ( (PRESCAL+1) * 2 )

- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock

- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = (SHTIM+1) / ADCClock

### 33.6.3 ADC Channel Enable Register

**Register Name:** ADC_CHER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | CH3 | CH2 | CH1 | CH0 |

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

### 33.6.4 ADC Channel Disable Register

**Register Name:** ADC_CHDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | CH3 | CH2 | CH1 | CH0 |

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

**Warning:** If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

### 33.6.5 ADC Channel Status Register

**Register Name:** ADC_CHSR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | CH3 | CH2 | CH1 | CH0 |

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

### 33.6.6 ADC Status Register

**Register Name:** ADC_SR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RXBUFF | ENDRX | GOVRE | DRDY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| — | — | — | — | OVRE3 | OVRE2 | OVRE1 | OVRE0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | EOC3 | EOC2 | EOC1 | EOC0 |

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished.

1 = Corresponding analog channel is enabled and conversion is complete.

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_SR.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_SR.

1 = At least one General Overrun Error has occurred since the last read of ADC_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.

### 33.6.7 ADC Last Converted Data Register

**Register Name:** ADC_LCDR

**Access Type:** Read-only

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

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

### 33.6.8 ADC Interrupt Enable Register

**Register Name:** ADC_IER

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RXBUFF | ENDRX | GOVRE | DRDY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| — | — | — | — | OVRE3 | OVRE2 | OVRE1 | OVRE0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | EOC3 | EOC2 | EOC1 | EOC0 |

- **EOCx: End of Conversion Interrupt Enable x**
- **OVREx: Overrun Error Interrupt Enable x**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 33.6.9 ADC Interrupt Disable Register

**Register Name:** ADC_IDR

**Access Type:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RXBUFF | ENDRX | GOVRE | DRDY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| — | — | — | — | OVRE3 | OVRE2 | OVRE1 | OVRE0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | EOC3 | EOC2 | EOC1 | EOC0 |

- **EOCx: End of Conversion Interrupt Disable x**
- **OVREx: Overrun Error Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

### 33.6.10 ADC Interrupt Mask Register

**Register Name:** ADC_IMR

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | RXBUFF | ENDRX | GOVRE | DRDY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| — | — | — | — | OVRE3 | OVRE2 | OVRE1 | OVRE0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| — | — | — | — | EOC3 | EOC2 | EOC1 | EOC0 |

- **EOCx: End of Conversion Interrupt Mask x**
- **OVREx: Overrun Error Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

### 33.6.11 ADC Channel Data Register

**Register Name:** ADC_CDRx

**Access Type:** Read-only

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

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

# 34. Segment LCD Controller (SLCDC)

## 34.1 Overview

An LCD consists of several segments (pixels or complete symbols) which can be visible or invisible. A segment has two electrodes with liquid crystal between them. When a voltage above a threshold voltage is applied across the liquid crystal, the segment becomes visible.

The voltage must alternate to avoid an electrophoresis effect in the liquid crystal, which degrades the display. Hence the waveform across a segment must not have a DC component.

The SLCDC controller is intended for monochrome passive liquid crystal display (LCD) with up to 10 common terminals and up to 40 segment terminals.

The SLCDC is programmable to support many different requirements such as:

- Adjusting the driving time of the LCD pads in order to save power and increase the controllability of the DC offset
- Driving smaller LCD (down to 1 common by 1 segment)
- Adjusting the SLCDC frequency in order to obtain the best compromise between frequency and consumption and adapt it to the LCD driver

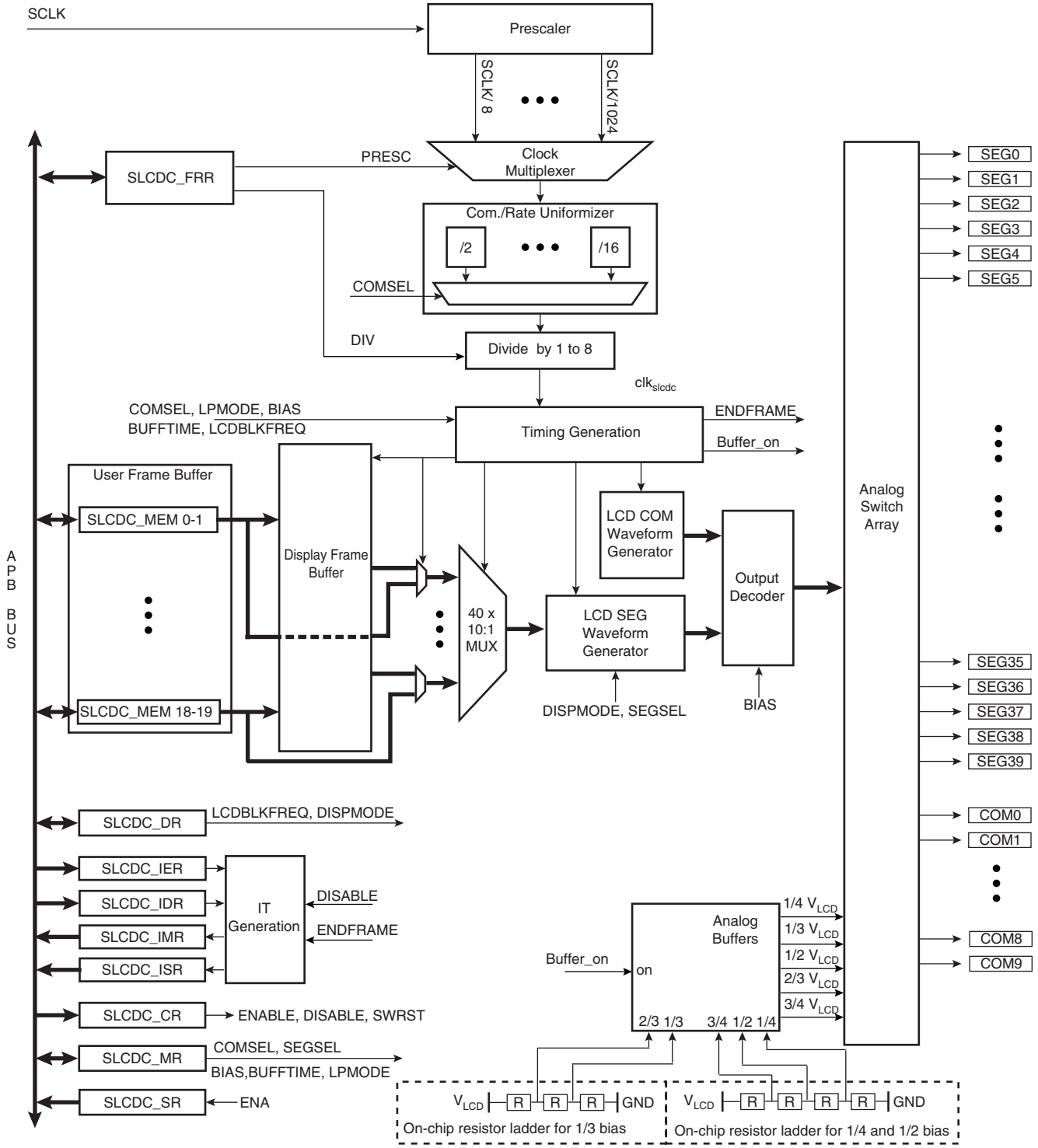
.

**Table 34-1.** List of Terms

| Term | Description |
|---|---|
| LCD | A passive display panel with terminals leading directly to a segment |
| Segment | The least viewing element (pixel) which can be on or off |
| Common(s) | Denotes how many segments are connected to a segment terminal |
| Duty | 1/(Number of common terminals on an actual LCD display) |
| Bias | 1/(Number of voltage levels used driving a LCD display -1) |
| Frame Rate | Number of times the LCD segments are energized per second. |

## 34.2 Block Diagram

**Figure 34-1.** LCD Macrocell Block Diagram



## 34.3 I/O Lines Description

**Table 34-2.** I/O Lines Description

| Name | Description | Type |
|---|---|---|
| SEG [39:0] | Segments control signals | Output |
| COM [9:0] | Commons control signals | Output |

## 34.4 Product Dependencies

### 34.4.1 I/O Lines

The pins used for interfacing the SLCD Controller may be multiplexed with PIO lines. Please refer to product block diagram.

In this case, the assignment of the segment controls and commons are automatically done depending on COMSEL and SEGSEL in SLCDC_MR. If I/O lines of the SLCD Controller are not used by the application, they can be used for other purposes by the PIO Controller.

### 34.4.2 Power Management

The SLCD Controller is clocked by the slow clock (SCLK). All the timings are based upon a typical value of 32 kHz for SCLK. The power management of the SLCD controller is handled by the Shutdown Controller.

The SLCD Controller is supplied by 3V domain.

### 34.4.3 Interrupt Sources

The SLCD Controller interrupt line is connected to one of the internal sources of the Advanced Interrupt Controller. Using the SLCD Controller interrupt requires prior programming of the AIC.

### 34.4.4 Number of Segments and Commons

The product, embeds 40 segments and 10 Commons.

## 34.5 Functional Description

After the initialization sequence the SLCDCC is ready to be enabled in order to enter the display phase (where it is possible to do more than display data written in the SLCDC memory) up to the disable sequence.

- Initialization Sequence:

1. Select the LCD supply source in the shutdown controller
   - Internal: The on chip charge pump is selected,
   - External: the external supply source has to be between 2 and 3.4 V
2. Select the clock division (SLDCD_FRR) to use a proper frame rate
3. Enter the number of common and segments terminals (SLDCD_MR)
4. Select the bias in compliance with the LCD manufacturer data sheet
5. Enter buffer driving time

- During the Display Phase:

1. Data may be written at any time in the SLCDC memory, they are automatically latched and displayed at the next LCD frame
2. It is possible to:
   - Adjust contrast
   - Adjust the frame frequency
   - Adjust buffer driving time
   - Reduce the SLCDC consumption by entering in low-power waveform at any time
   - Use the large set of display features such as blinking, inverted blink, etc.

- Disable Sequence:

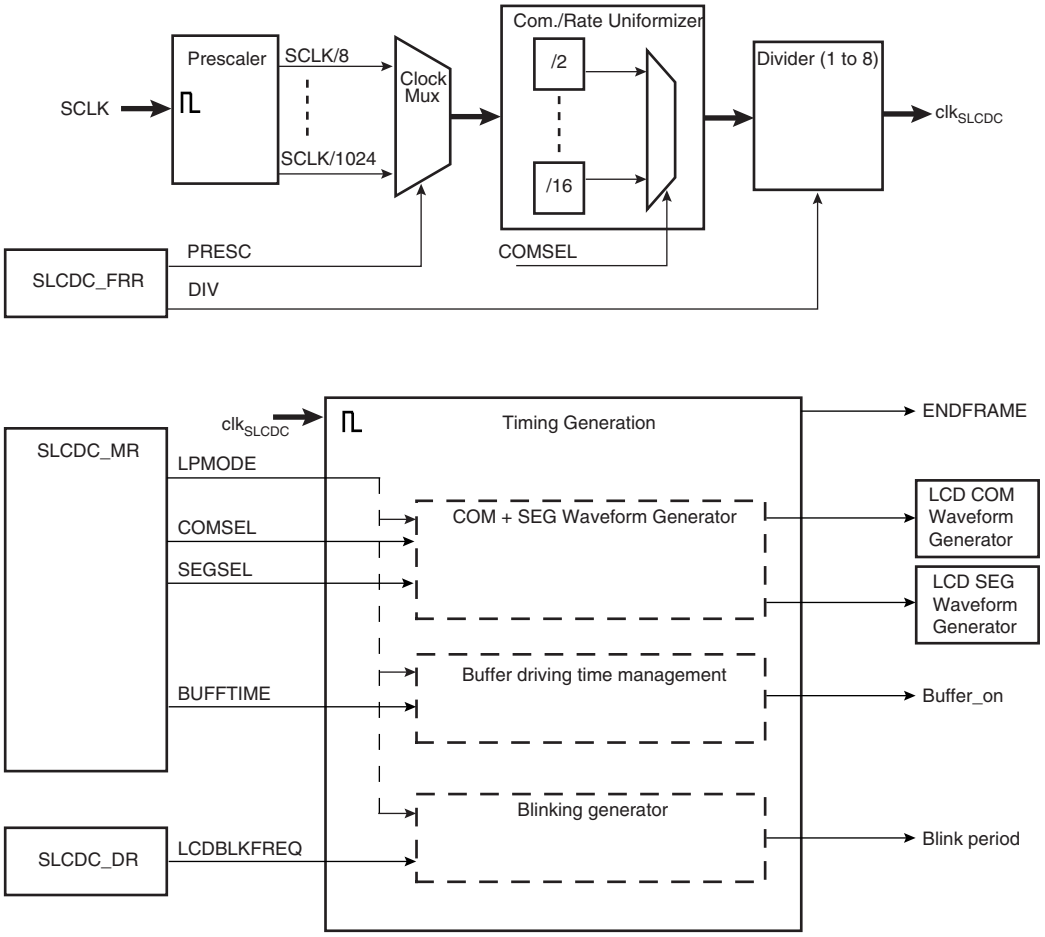
There are two ways to disable the SLCDC

1. By using the LCDDIS (LCD Disable) bit. (In this case, SLCDC configuration and memory content are kept.)
2. Or by using the SWRST (Software Reset) bit.

### 34.5.1 Clock Generation

#### *34.5.1.1 Block Diagram*

**Figure 34-2.** Clock Generation Block Diagram

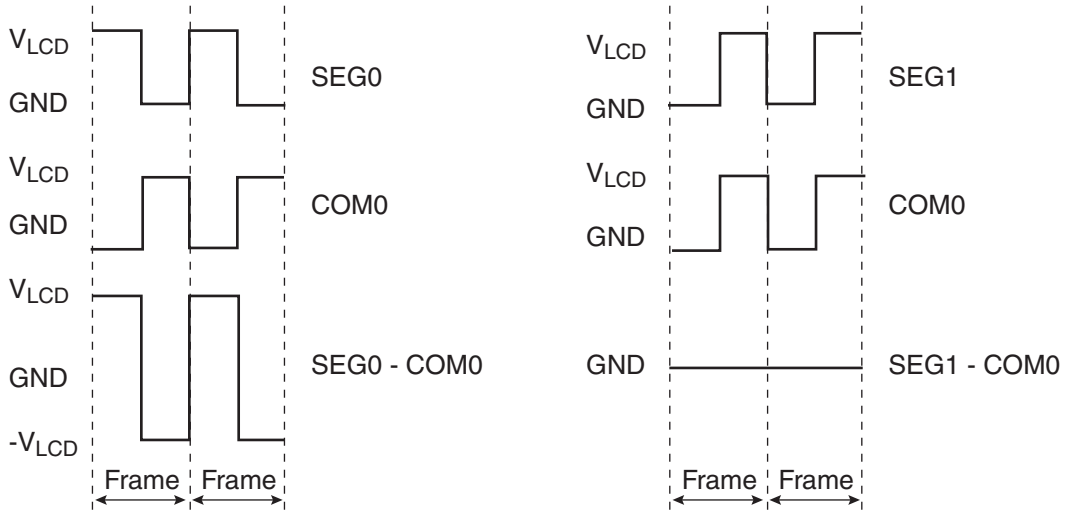


### 34.5.2 Waveform Generation

#### *34.5.2.1 Static Duty and Bias*

This kind of display is driven with the waveform shown in Figure 34-3. SEG0 - COM0 is the voltage across a segment that is on, and SEG1 - COM0 is the voltage across a segment that is off.

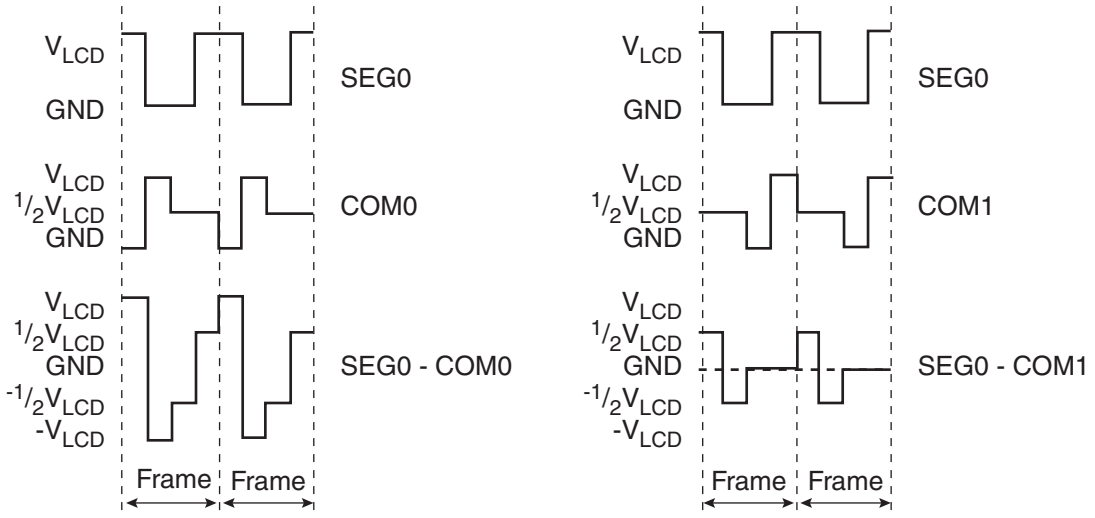
**Figure 34-3.** Driving an LCD with One Common Terminal




#### *34.5.2.2 1/2 Duty and 1/2 Bias*

For an LCD with two common terminals (1/2 duty) a more complex waveform must be used to control segments individually. Although 1/3 bias can be selected, 1/2 bias is most common for these displays. In the waveform shown in Figure 34-4, SEG0 - COM0 is the voltage across a segment that is on, and SEG0 - COM1 is the voltage across a segment that is off.

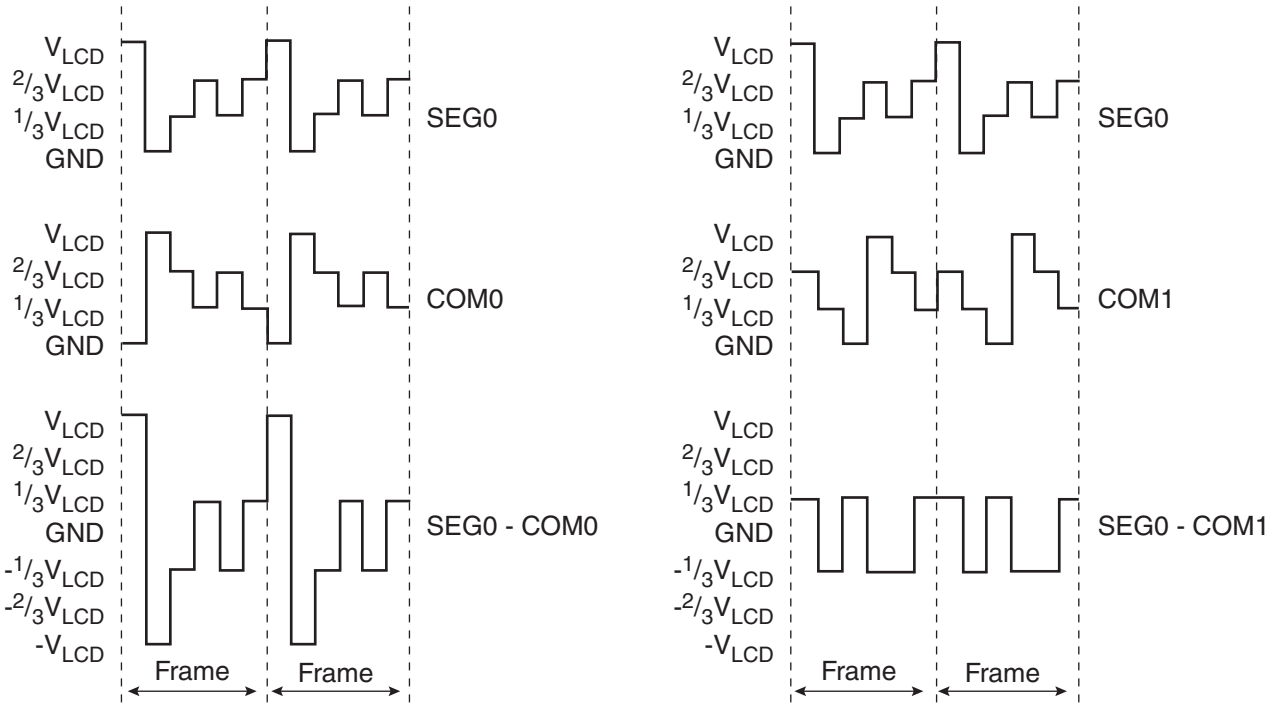
**Figure 34-4.** Driving an LCD with Two Common Terminals



*34.5.2.3 1/3 Duty and 1/3 Bias*

1/3 bias is usually recommended for an LCD with three common terminals (1/3 duty). In the waveform shown in Figure 34-5, SEG0 - COM0 is the voltage across a segment that is on and SEG0-COM1 is the voltage across a segment that is off.

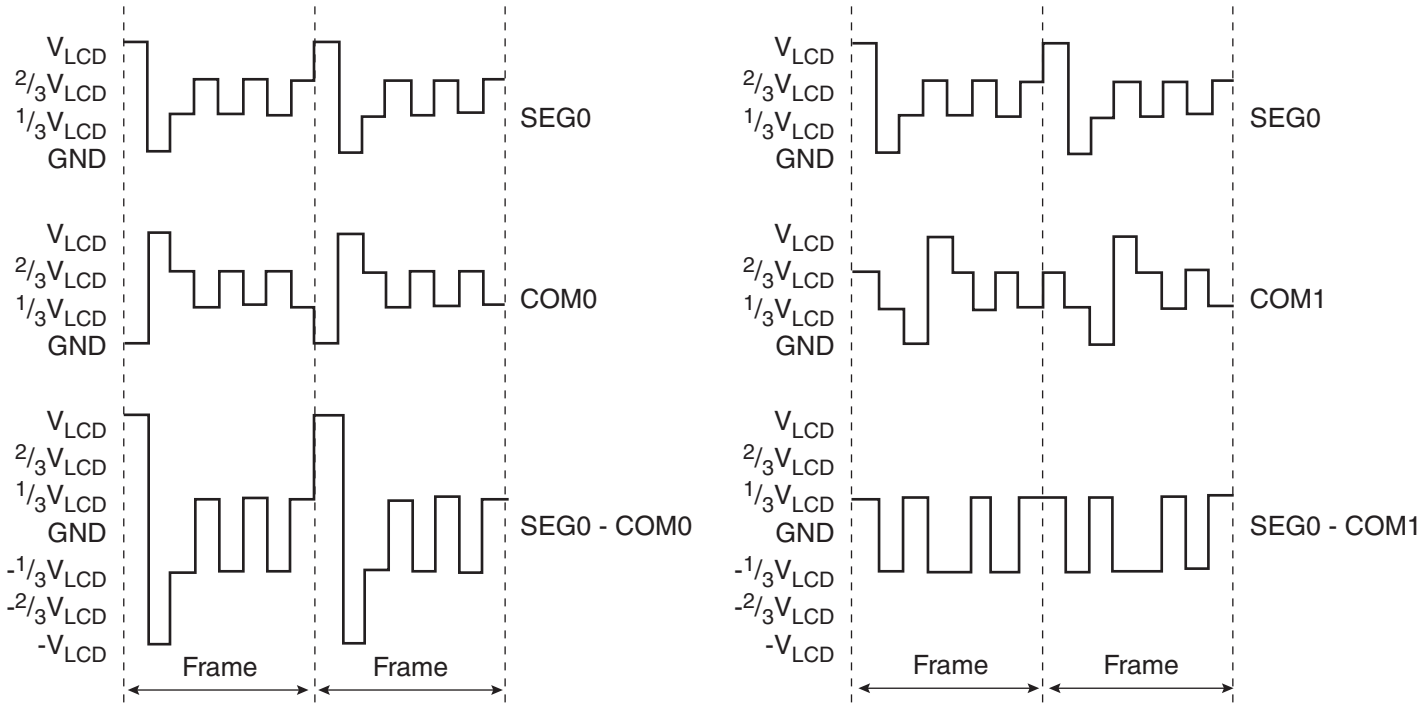
**Figure 34-5.** Driving an LCD with Three Common Terminals



*34.5.2.4 1/4 Duty and 1/3 Bias*

1/3 bias is optimal for LCD displays with four common terminals (1/4 duty). In the waveform shown in Figure 34-6, SEG0 - COM0 is the voltage across a segment that is on and SEG0 - COM1 is the voltage across a segment that is off.

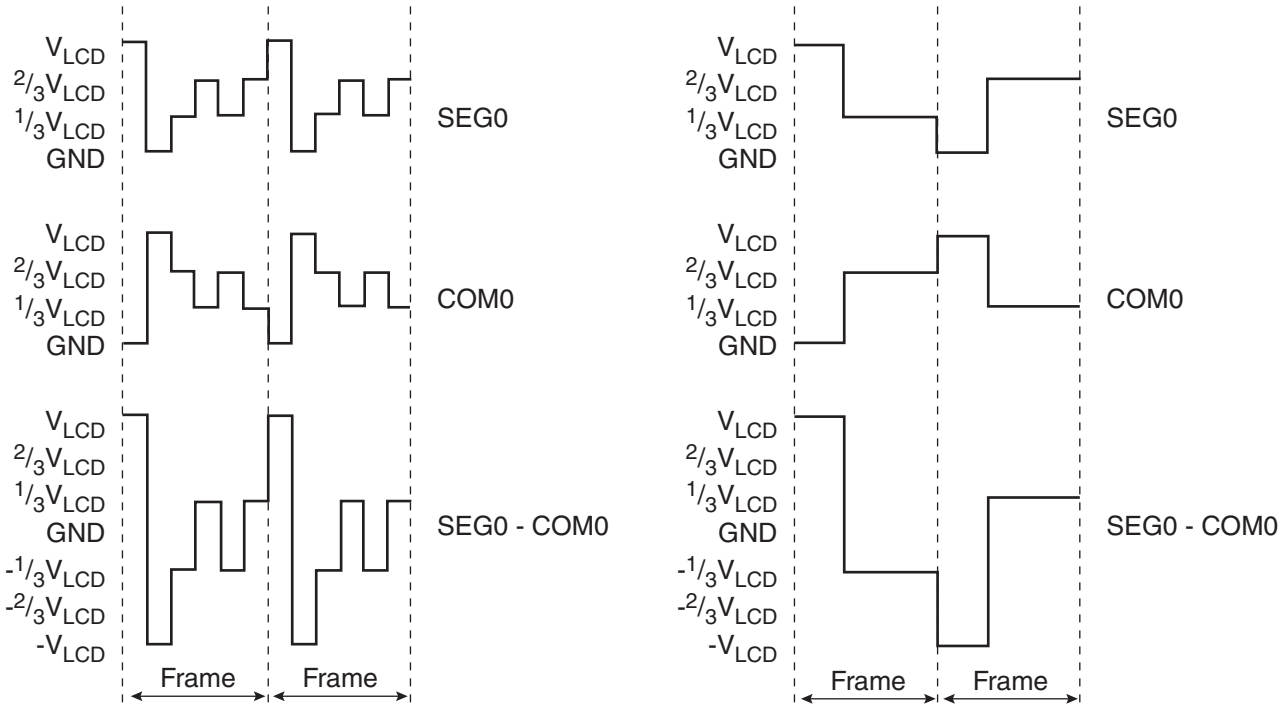
**Figure 34-6.** Driving an LCD with Four Common Terminals



### 34.5.2.5 Low Power Waveform

To reduce toggle activity and hence power consumption, a low power waveform can be selected by writing LPMODE to one. The default and low power waveform is shown in Figure 34-7 for 1/3 duty and 1/3 bias. For other selections of duty and bias, the effect is similar.

**Figure 34-7.** Default and Low Power Waveform




Note: Refer to the LCD specification to verify that low power waveforms are supported.

### 34.5.2.6 Frame Rate

The Frame Rate register (SLCDC_FRR) enables the generation of the frequency used by the SLCD Controller. It is done by a prescaler (division by 8, 16, 32, 64, 128, 256, 512 and 1024) followed by a finer divider (division by 1, 2, 3, 4, 5, 6, 7 or 8).

To calculate the proper frame frequency needed, the equation below must be taken into account:

$$f_{frame} = \frac{fSCLK}{(PRES \cdot DIV \cdot NCOM)}$$

Where:

$f_{SCLK}$ = slow clock frequency

$f_{frame}$ = frame frequency

PRES = prescaler value (8, 16, 32, 64, 128, 256, 512 or 1024)

DIV = divider value (1, 2, 3, 4, 5, 6, 7, or 8)

NCOM = depends of number of commons and is defined in Table 34-3 below:

**Table 34-3.** NCOM

| Number of Commons | NCOM |
|---|---|
| 1 | 16 |
| 2 | 16 |
| 3 | 15 |
| 4 | 16 |
| 5 | 15 |
| 6 | 18 |
| 7 | 14 |
| 8 | 16 |
| 9 | 18 |
| 10 | 20 |

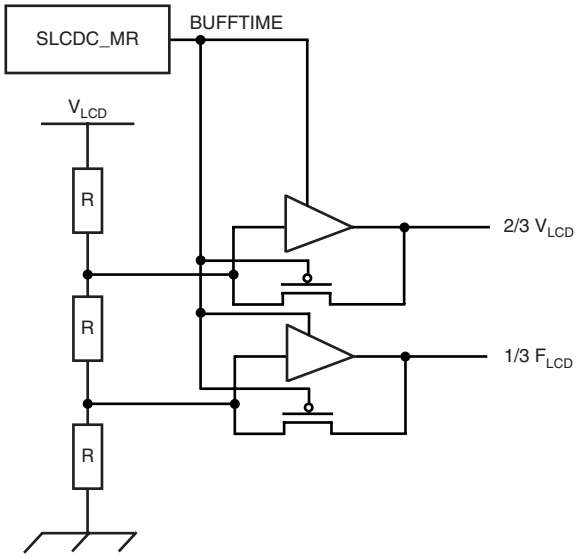
#### *34.5.2.7 Buffer Driving Time*

Intermediate voltage levels are generated from buffer drivers. The buffers are active the amount of time specified by BUFTIME[3:0] in SLCDC_MR, then buffers are bypassed.

Shortening the drive time will reduce power consumption, but displays with high internal resistance or capacitance may need longer drive time to achieve sufficient contrast.

Example for bias = 1/3.

**Figure 34-8.** Buffer Driving

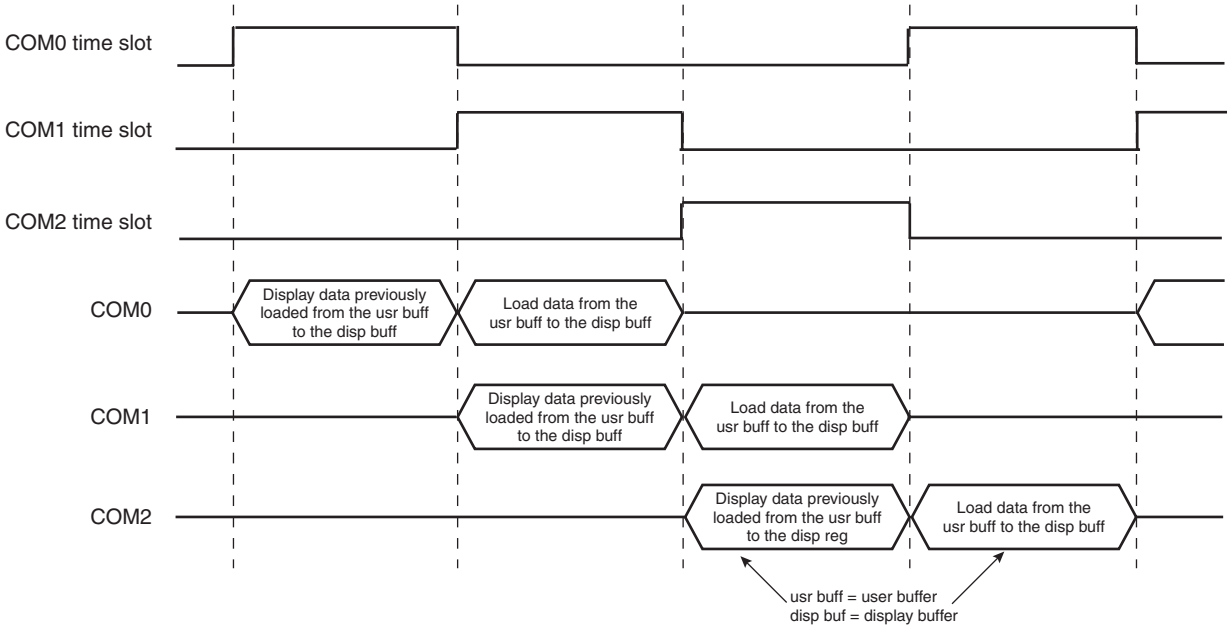



### 34.5.3 Number of Commons, Segments and Bias

It is important to note that the selection of the number of commons, segments and the bias is only taken into account when the SLCDC is disabled.

### 34.5.4 SLCDC memory

**Figure 34-9.** Memory Management




When a bit in the display memory (SLDC_MEM) is written to one, the corresponding segment is energized (on), and non-energized when a bit in the display memory is written to zero.

At the beginning of each common, the display buffer is updated. The value of the previous common is latched in the display memory (it's value is transferred from the user buffer to the frame buffer).

The advantages of this solution are:

- Ability to access the user buffer at any time in the frame, in any display mode and even in low power waveform
- Ability to change only one pixel without reloading the picture

### 34.5.5 Display Features

In order to improve the flexibility of SLCDC the following set of display modes are embedded:

1. Force Mode Off: All pixels are turned off and the memory content is kept.
2. Force Mode On: All pixels are turned on and the memory content is kept.
3. Inverted Mode: All pixels are set in the inverted state as defined in SLCDC memory and the memory content is kept.
4. Two Blinking Modes:
   – Standard Blinking Mode: All pixels are alternately turned off to the predefined **state** in SLCDC memory at LCDBLKFREQ frequency.
   – Inverted Blinking Mode: All pixels are alternately turned off to the predefined **opposite state** in SLCDC memory at LCDBLKFREQ frequency.
5. Buffer Swap Mode: All pixels are alternatively assigned to the state defined in the user buffer then to the state defined in the display buffer.

### 34.5.6 Buffer Swap Mode

This mode allows to assign all pixels to two states alternatively without reloading the user buffer at each change.

The means to alternatively display two states is as follows:

1. Initially, the SLCDC must be in normal mode or in a standard blinking mode.
2. Data corresponding to the first pixel state is written in the user buffer (through the SLCDC_MEM registers).
3. Wait two ENDFRAME events (to be sure that the user buffer is entirely transferred in the display buffer).
4. SLCDC_DR must be programmed with DISPMODE = 6 (User Buffer Only Load Mode). This mode blocks the automatic transfer from the user buffer to the display buffer.
5. Wait ENDFRAME event. (The display mode is internally updated at the beginning of each frame.)
6. Data corresponding to the second pixel state is written in the user buffer (through the SLCDC_MEM registers). So, now the first pixel state is in the display buffer and the second pixel state is in the user buffer.
7. SLCDC_DR must be programmed with DISPMODE = 7 (buffer swap mode) and LCDBLKFREQ must be programmed with the wanted blinking frequency (if not previously done).

Now, each state is alternatively displayed at LCDBLKFREQ frequency.

Except for the phase dealing with the storage of the two display states, the management of the Buffer Swap Mode is the same as the standard blinking mode.

### 34.5.7 Disable Sequence

There are two ways to disable the SLCDC:

1. By using the disable bit. (In this case, register configuration and SLCDC memory are kept.)
2. Or by using the software reset bit that acts like a hardware reset.

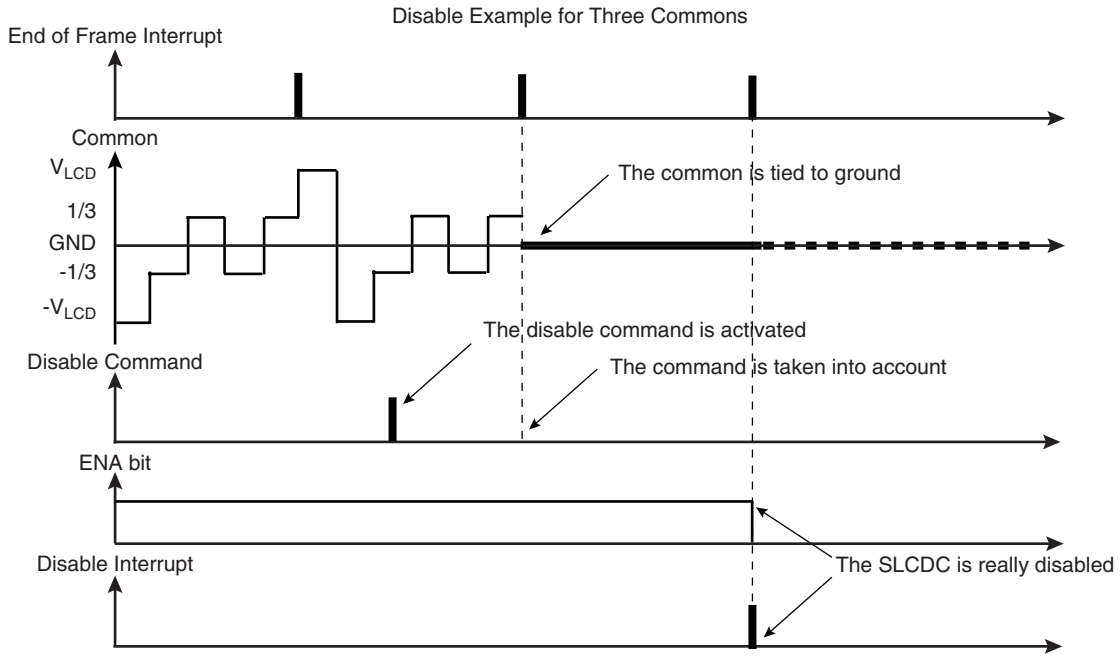
In both cases, no DC voltage should be left across any segment.

#### *34.5.7.1 Disable Bit*

When the LCD Disable Command is activated during a frame, the next frame will be generated in “All Ground” Mode (whereby all commons and segments will be tied to ground).

At the end of this ‘All Ground” frame, the disable bit is reset and the disable interrupt is asserted. This indicates that the SLCDC is really disabled and that the LCD can be switched off.

**Figure 34-10.** Disabling Sequence

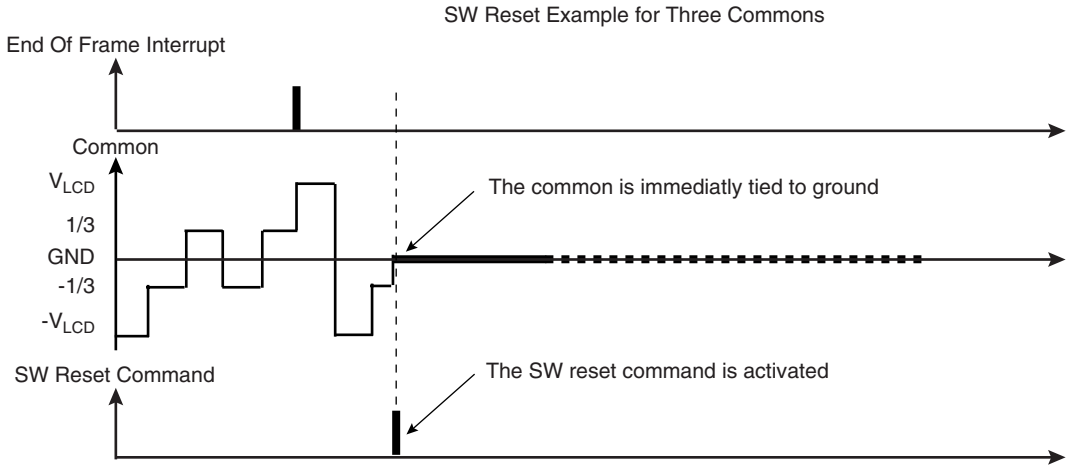



### *34.5.7.2 Software Reset*

When the LCD software reset command is activated during a frame it is immediately taken into account and all commons and segments are tied to ground.

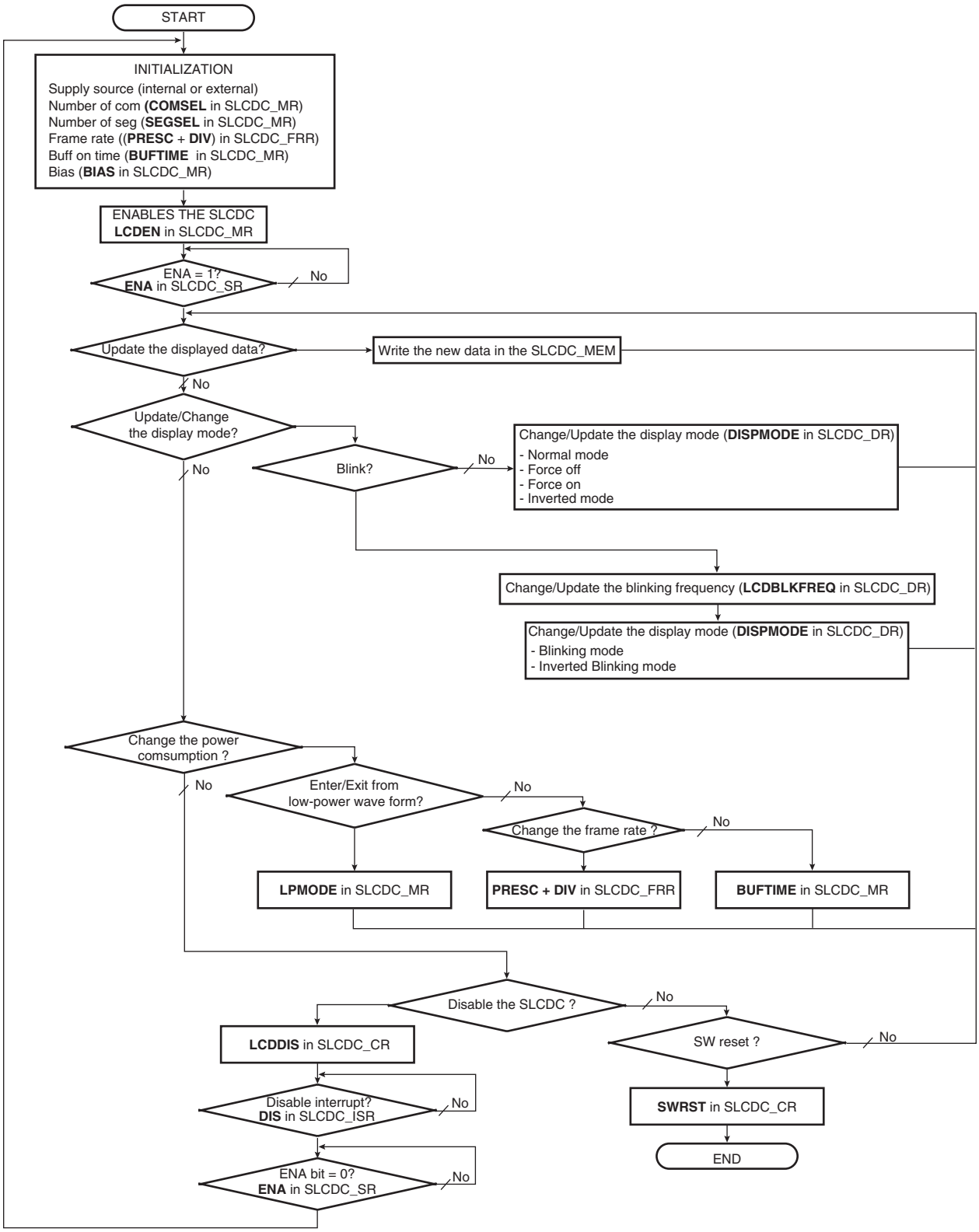
Note that in the case of a software reset, the disable interrupt is not asserted.

**Figure 34-11.** Software Reset



### 34.5.8 Flowchart

**Figure 34-12.** SLCDC Flow Chart



## 34.6 Waveform Specifications

### 34.6.1 DC Characteristics

Refer to the DC Characteristics section of the product datasheet.

### 34.6.2 LCD Contrast

The peak value ($V_{LCD}$) on the output waveform determines the LCD Contrast. VLCD is controlled by software in 16 steps of 62 mV each from 2.4V to 3.4V independent of VDDIN.

This is a function of the supply controller.

## 34.7 Segment LCD Controller (SLCDC) User Interface

**Table 34-4.** Register Mapping

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x0 | SLCDC Control Register | SLCDC_CR | Write-only | - |
| 0x4 | SLCDC Mode Register | SLCDC_MR | Read-write | 0x0 |
| 0x8 | SLCDC Frame Rate Register | SLCDC_FRR | Read-write | 0x0 |
| OxC | SLCDC Display Register | SLCDC_DR | Read-write | 0x0 |
| 0x10 | SLCDC Status Register | SLCDC_SR | Read-only | 0x0 |
| 0x20 | SLCDC Interrupt Enable Register | SLCDC_IER | Write-only | - |
| 0x24 | SLCDC Interrupt Disable Register | SLCDC_IDR | Write-only | - |
| 0x28 | SLCDC Interrupt Mask Register | SLCDC_IMR | Write-only | - |
| 0x2C | SLCDC Interrupt Status Register | SLCDC_ISR | Read-only | 0x0 |
| 0x200 | SLCDC Memory Register | SLCDC_MEM | Read-write | 0x0 |

### 34.7.1 SLCDC Control Register

**Name:** SLCDC_CR
**Access:** Write-only
**Reset Value:** 0x00000000

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

- **LCDEN: Enable the LCDC**

0 = No effect.

1 = The SLCDC is enabled

- **LCDDIS: Disable LCDC**

0 = No effect.

1 = The SLCDC is disabled.

Note: LCDDIS is taken into account at the beginning of the next frame.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a power-up reset. When this command is performed, the SLCDC1 immediately ties all segments end commons lines to values corresponding to a “ground voltage”.

### 34.7.2 SLCDC Mode Register

**Name:** SLCDC_MR
**Access:** Read-write
**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | LPMODE |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | BIAS | | BUFTIME | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | SEGSEL | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | COMSEL | | | |

- **COMSEL: Selection of the Number of Commons**

(Taken into account when the SLCDC is disabled.)

| COMSEL3 | COMSEL2 | COMSEL1 | COMSEL0 | COM Pin | I/O Port Pin |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | COM0 | COM1:9 |
| 0 | 0 | 0 | 1 | COM0:1 | COM2:9 |
| 0 | 0 | 1 | 0 | COM0:2 | COM3:9 |
| 0 | 0 | 1 | 1 | COM0:3 | COM4:9 |
| 0 | 1 | 0 | 0 | COM0:4 | COM5:9 |
| 0 | 1 | 0 | 1 | COM0:5 | COM6:9 |
| 0 | 1 | 1 | 0 | COM0:6 | COM7:9 |
| 0 | 1 | 1 | 1 | COM0:7 | COM8:9 |
| 1 | 0 | 0 | 0 | COM0:8 | COM9 |
| 1 | 0 | 0 | 1 | COM0:9 | None |

- **SEGSEL: Selection of the Number of Segments**

(Taken into account when the SLCDC is disabled.)

| SEGSEL5 | SEGSEL4 | SEGSEL3 | SEGSEL2 | SEGSEL1 | SEGSEL0 | I/O Port in Use as Segment Driver | Maximum Number of Segments |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | SEG0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | SEG0 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | SEG0:1 | 3 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1 | 0 | 0 | 1 | 0 | 1 | SEG0:37 | 38 |
| 1 | 0 | 0 | 1 | 1 | 0 | SEG0:38 | 39 |
| 1 | 0 | 0 | 1 | 1 | 1 | SEG0:39 | 40 |

- **BUFTIME: Buffer On-Time**

(Taken into account from the next begin of frame.)

| BUFTIME3 | BUFTIME2 | BUFTIME1 | BUFTIME0 | Nominal Drive Time |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0% |
| 0 | 0 | 0 | 1 | 2 x $t_{SCLK}$ |
| 0 | 0 | 1 | 0 | 4 x $t_{SCLK}$ |
| 0 | 0 | 1 | 1 | 8 x $t_{SCLK}$ |
| 0 | 1 | 0 | 0 | 16 x $t_{SCLK}$ |
| 0 | 1 | 0 | 1 | 32 x $t_{SCLK}$ |
| 0 | 1 | 1 | 0 | 64 x $t_{SCLK}$ |
| 0 | 1 | 1 | 1 | 128 x $t_{SCLK}$ |
| 1 | 0 | 0 | 0 | 50% |
| 1 | 0 | 0 | 1 | 100% |

- **BIAS: LCD Display Configuration**

(Taken into account when the SLCDC is disabled.)

| BIAS1 | BIAS0 | Ratio |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1/2 |
| 1 | 0 | 1/3 |
| 1 | 1 | 1/4 |

Note: BIAS is only taken into account when the SLCDC is disabled.

- **LPMODE: Low Power Mode** (Taken into account from the next begin of frame.)

0 = Normal Mode.

1 = Low Power Waveform is enabled.

### 34.7.3 SLCDC Frame Rate Register

**Name:** SLCDC_FRR
**Access:** Read-write
**Reset Value:** 0x00000000

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

- **PRES: Clock Prescaler**

(Taken into account from the next begin of frame.)

| PRESC2 | PRESC1 | PRESC0 | Output from Prescaler |
|---|---|---|---|
| 0 | 0 | 0 | SCLK/8 |
| 0 | 0 | 1 | SCLK/16 |
| 0 | 1 | 0 | SCLK/32 |
| 0 | 1 | 1 | SCLK/64 |
| 1 | 0 | 0 | SCLK/128 |
| 1 | 0 | 1 | SCLK/256 |
| 1 | 1 | 0 | SCLK/512 |
| 1 | 1 | 1 | SCLK/1024 |

- **DIV: Clock Division**

(Taken into account from the next begin of frame.)

| DIV2 | DIV1 | DIV0 | Output from Prescaler Divided by: |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 3 |
| 0 | 1 | 1 | 4 |
| 1 | 0 | 0 | 5 |
| 1 | 0 | 1 | 6 |
| 1 | 1 | 0 | 7 |
| 1 | 1 | 1 | 8 |

### 34.7.4 SLCDC Memory Register

**Name:** SLCDC_MEM

**Access:** Read / Write

Write a SLCD memory bit to one and the corresponding segment will be energized (visible). Unused SLCD Memory bits for the actual display can be used freely as storage.

| | | SEG0 | -- | SEG31 | SEG32 | -- | SEG39 | Memory address |
|---|---|---|---|---|---|---|---|---|
| SLCDC_MEM19 | COM9 | | | | X | -- | X | From 0x24C to 0x24F |
| SLCDC_MEM18 | COM9 | X | -- | X | | | | From 0x248 to 0x24B |
| SLCDC_MEM17 | COM8 | | | | X | -- | X | From 0x244 to 0x247 |
| SLCDC_MEM16 | COM8 | X | -- | X | | | | From 0x240 to 0x243 |
| SLCDC_MEM15 | COM7 | | | | X | -- | X | From 0x23C to 0x23F |
| SLCDC_MEM14 | COM7 | X | -- | X | | | | From 0x238 to 0x23B |
| SLCDC_MEM13 | COM6 | | | | X | -- | X | From 0x234 to 0x237 |
| SLCDC_MEM12 | COM6 | X | -- | X | | | | From 0x230 to 0x233 |
| SLCDC_MEM11 | COM5 | | | | X | -- | X | From 0x22C to 0x22F |
| SLCDC_MEM10 | COM5 | X | -- | X | | | | From 0x228 to 0x22B |
| SLCDC_MEM9 | COM4 | | | | X | -- | X | From 0x224to 0x227 |
| SLCDC_MEM8 | COM4 | X | -- | X | | | | From 0x220 to 0x223 |
| SLCDC_MEM7 | COM3 | | | | X | -- | X | From 0x21C to 0x21F |
| SLCDC_MEM6 | COM3 | X | -- | X | | | | From 0x218 to 0x21B |
| SLCDC_MEM5 | COM2 | | | | X | -- | X | From 0x214 to 0x217 |
| SLCDC_MEM4 | COM2 | X | -- | X | | | | From 0x210 to 0x213 |
| SLCDC_MEM3 | COM1 | | | | X | -- | X | From 0x20C to 0x20F |
| SLCDC_MEM2 | COM1 | X | -- | X | | | | From 0x208 to 0x20B |
| SLCDC_MEM1 | COM0 | | | | X | -- | X | From 0x204 to 0x207 |
| SLCDC_MEM0 | COM0 | X | -- | X | | | | From 0x200 to 0x203 |

### 34.7.5 SLCDC Display Register

**Name:** SLCDC_DR
**Access:** Read-write
**Reset Value:** 0x00000000

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

- **DISPMODE: Display Mode Register**

(Taken into account from the next begin of frame.)

| DISPMODE2 | DISPMODE1 | DISPMODE0 | Display Mode |
|---|---|---|---|
| 0 | 0 | 0 | Normal Mode:<br>Latched data are displayed. |
| 0 | 0 | 1 | Force Off Mode**:**<br>All pixels are invisible. (The SLCDC memory is unchanged.) |
| 0 | 1 | 0 | Force On Mode<br>All pixels are visible. (The SLCDC memory is unchanged.) |
| 0 | 1 | 1 | Blinking Mode:<br>All pixels are alternately turned off to the predefined **state** in SLCDC memory at LCDBLKFREQ frequency. (The SLCDC memory is unchanged.) |
| 1 | 0 | 0 | Inverted Mode:<br>All pixels are set in the inverted state as defined in SLCDC memory. (The SLCDC memory is unchanged.) |
| 1 | 0 | 1 | Inverted Blinking Mode:<br>All pixels are alternately turned off to the predefined **opposite state** in SLCDC memory at LCDBLKFREQ frequency. (The SLCDC memory is unchanged.) |
| 1 | 1 | 0 | User Buffer Only Load Mode:<br>Blocks the automatic transfer from User Buffer to Display Buffer. |
| 1 | 1 | 1 | Buffer Swap Mode:<br>All pixels are alternatively assigned to the state defined in the User Buffer, then to the state defined in the Display Buffer at LCDBLKFREQ frequency. |

- **LCDBLKFREQ: LCD Blinking Frequency Selection**

(Taken into account from the next begin of frame.)

Blinking frequency = Frame Frequency/LCDBLKFREQ[7:0].

Note: 0 written in LCDBLKFREQ stops blinking.

### 34.7.6 SLCDC Status Register

**Name:** SLCDC_SR

**Access Type:** Read-only

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

- **ENA: Enable Status** (Automatically Set/Reset)

0 = The SLCDC1 is disabled.

1 = The SLCDC1 is enabled.

### 34.7.7 SLCDC Interrupt Enable Register

**Name:** SLCDC_IER

**Access Type:** Write-only

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

- **ENDFRAME: End of Frame Interrupt Enable**

- **DIS: Disable Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 34.7.8 SLCDC Interrupt Disable Register

**Name:** SLCDC_IDR

**Access Type:** Write-only

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

- **ENDFRAME: End of Frame Interrupt Disable**

- **DIS: Disable Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

### 34.7.9 SLCDC Interrupt Mask Register

**Name:** SLCDC_IMR

**Access Type:** Write-only

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

- **ENDFRAME: End of Frame Interrupt Mask**

- **DIS: Disable Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

### 34.7.10 SLCDC Interrupt Status Register

**Name:** SLCDC_ISR

**Access Type:** Read-only

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

- **ENDFRAME: End of Frame Interrupt Status**

0 = End of Frame Interrupt has not occurred since the last read of the Interrupt Status Register.

1 = End of Frame Interrupt has occurred since the last read of the Interrupt Status Register.

- **DIS: Disable Interrupt Status**

0 = Disable Interrupt has not occurred since the last read of the Interrupt Status Register

1 = Disable Interrupt has occurred since the last read of the Interrupt Status Register.

# 35. AT91SAM7L128/64 Electrical Characteristics

## 35.1 Absolute Maximum Ratings

**Table 35-1.** Absolute Maximum Ratings*

| |
|---|
| Operating Temperature (Industrial) .....-40°C to + 85°C |
| Storage Temperature.....-60°C to + 150°C |
| Voltage on Input Pins<br>with Respect to Ground.....-0.3V to + 5.5V |
| Maximum Operating Voltage<br>(VDDCORE ).....2.0V |
| Maximum Operating Voltage<br>(VDDIO1, VDDIO2 and VDDINLCD).....4.0V |
| Total DC Output Current on all I/O lines<br>128-lead LQFP/144-ball LFBGA.....100 mA |

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## 35.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: $T_A$ = -40°C to 85°C, unless otherwise specified.

**Table 35-2.** DC Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{DDCORE}$ | DC Supply Core | Depends on VDDOUT (externally connected to VDDOUT) | | VDDOUT | | V |
| $V_{VDDIO1}$ | DC Supply I/Os | | 1.8 | | 3.6 | V |
| $V_{VDDIO2}$ | DC Supply I/Os | Adjustable | 1.8 | | 3.6 | V |
| $V_{VDDINLCD}$ | DC Supply Charge Pump | | 1.8 | | 3.6 | V |
| $V_{VDDLCD}$ | DC Supply LCD Regulator | | 2.5 | | 3.6 | V |
| $V_{IL}$ | Input Low-level Voltage | $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC29, NRST, NRSTB, CLKIN | -0.3 | | 0.3 x $V_{VDDIO1}$ | V |
| | | $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23 | -0.3 | | 0.3 x $V_{VDDIO2}$ | V |
| $V_{IH}$ | Input High-level Voltage | $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC29, NRST, NRSTB, CLKIN | 0.7 x $V_{VDDIO1}$ | | $V_{VDDIO1}$ +0.3V | V |
| | | $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23 | 0.7 x $V_{VDDIO2}$ | | $V_{VDDIO2}$ +0.3V | V |
| $V_{Hys}$ | Hysteresis Voltage | $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC29, NRST, NRSTB, CLKIN | 0.25 | | 0.65 | V |
| | | $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23 | 0.25 | | 0.7 | V |
| $V_{OL}$ | Output Low-level Voltage | $I_O$ max, $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC29, NRST | | | 0.4 | V |
| | | $I_O$ max, $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23 | | | 0.4 | |
| $V_{OH}$ | Output High-level Voltage | $I_O$ max, $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC29, NRST | $V_{VDDIO1}$ - 0.4 | | | V |
| | | $I_O$ max, $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23 | $V_{VDDIO2}$ - 0.4 | | | V |
| $I_O$ | Output current | $V_{VDDIO1}$ from 1.8V to 3.6V PC0-PC6, PC11-PC29, NRST $V_{VDDIO2}$ from 1.8V to 3.6V PA0-PA25, PB0-PB23, | | | 2 | mA |
| | | $V_{VDDIO1}$ from 1.8V to 3.6V PC7-PC10 | | | 4 | |

**Table 35-2.** DC Characteristics (Continued)

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $I_{LEAK}$ | Input Leakage Current | Pull-up resistors disabled<br>(Typ: $T_A$ = 25°C, Max: $T_A$ = 85°C)<br>$V_{VDDIO1}$ from 1.8V to 3.6V<br>PC0-PC6, PC11-PC29, NRST<br>$V_{VDDIO2}$ from 1.8V to 3.6V<br>PA0-PA25, PB0-PB23 | | 1 | 20 | nA |
| | | Pull-up resistors disabled<br>(Typ: $T_A$ = 25°C, Max: $T_A$ = 85°C)<br>$V_{VDDIO1}$ from 1.8V to 3.6V<br>PC70-PC10 | | 2 | 40 | nA |
| $R_{PULLUP}$ | Pull-up Resistor | PC0-PC29,NRST<br>$V_{VDDIO1}$ from 1.8V to 3.6V | 75 | 100 | 145 | kΩ |
| | | PA0-PA25, PB0-PB23,<br>$V_{VDDIO2}$ from 1.8V to 3.6V | 40 | 100 | 375 | kΩ |
| $R_{PULLDOWN}$ | Pull-down Resistor | TST, ERASE, JTAGSEL<br>$V_{VDDIO1}$ from 1.8V to 3.6V | 10 | 15 | 25 | kΩ |
| $C_{IN}$ | Input Capacitance | Digital Inputs | | | 4 | pF |

**Table 35-3.** 1.8V Voltage Regulator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDIO1}$ | Supply Voltage | | 1.8 | 2.7 | 3.6 | V |
| $V_{ACCURACY}$ | Output Voltage Accuracy | Normal mode, $I_{Load}$ = 0.1mA to 60 mA | -3 | | 3 | % |
| $D_{DROPOUT}$ | Dropout Voltage | $V_{VDDIN}$ = 1.8V, $I_{Load}$ = 60 mA, Normal mode, 1.8V selected (1) | | | 150 | mV |
| $V_{VDDOUT}$ | Output Voltage | Normal Mode : 100mV step adjustable (2)<br>Deep Mode : 100mV step adjustable (2)<br>Standby mode | 1.55<br>1.55 | <br><br>0 | 1.8<br>1.8 | V |
| $I_{VDDIN}$ | Current consumption | Normal Mode<br>Deep mode | | 20<br>6 | 30<br>8.5 | µA |
| $T_{START}$ | Startup Time | Standby to Normal Mode<br>Deep to Normal Mode<br>1.55V to 1.8V Normal Mode<br>1.55V to 1.8V Deep mode | | 200<br>200<br>200<br>200 | 400<br>400<br>400<br>400 | µS |
| $I_{Load}$ | Maximum DC Output Current | Normal mode<br>Deep Mode | | | 60<br>1 | mA |

Notes: 1. This indicates that the minimum voltage on VDDOUT is: 1.80 - 0.150 = 1.65 V (when VDDIO1 = 1.8V and selected output voltage at 1.80V).

2. Refer to Supply Controller Mode Register, VRVDD field.

**Table 35-4.** Brownout Detector Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDIO1}$ | Supply Voltage | | | $V_{VDDIO1}$ | | V |
| $T_{ACCURACY}$ | Threshold Level Accuracy | 16 selectable steps of 100mV from 1.9V to 3.4V | -1.5 | | 1.5 | % |
| $V_{HYST}$ | Hysteresis | | 10 | 20 | 30 | mV |
| $I_{DD}$ | Current Consumption | Normal mode | | 25 | 48 | µA |
| $T_{START}$ | Startup Time | | | | 140 | µS |

**Table 35-5.** Zero-Power-on Reset Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDIO1}$ | Supply Voltage | | | $V_{VDDIO1}$ | | V |
| $V_{op}$ | Operating voltage rising | At Startup | | | 0.6 | V |
| $V_{th+}$ | Threshold voltage rising | At Startup | 1.8 | 2.0 | 2.2 | V |
| $V_{th-}$ | Threshold voltage falling | | | | 1.8 | V |
| PDELAY | Power-on Reset delay (1) | | 3 | 4 | 6.8 | mS |

Note: 1. Minimum time of a voltage drop for the Power-on reset to react.

**Table 35-6.** DC Flash Characteristics

| Symbol | Parameter | Conditions | Typ | Units |
|---|---|---|---|---|
| $I_{SB}$ | Standby current | @85°C onto VDDCORE = 1.8V<br>@25°C onto VDDCORE = 1.8V | 10<br>1 | µA<br>µA |
| $I_{CC}$ | Active current | Random Read @ 25MHz<br>onto VDDCORE = 1.8V | 12 | mA |
| | | Write<br>onto VDDCORE = 1.8V | 3.5 | mA |

## 35.3 Power Consumption

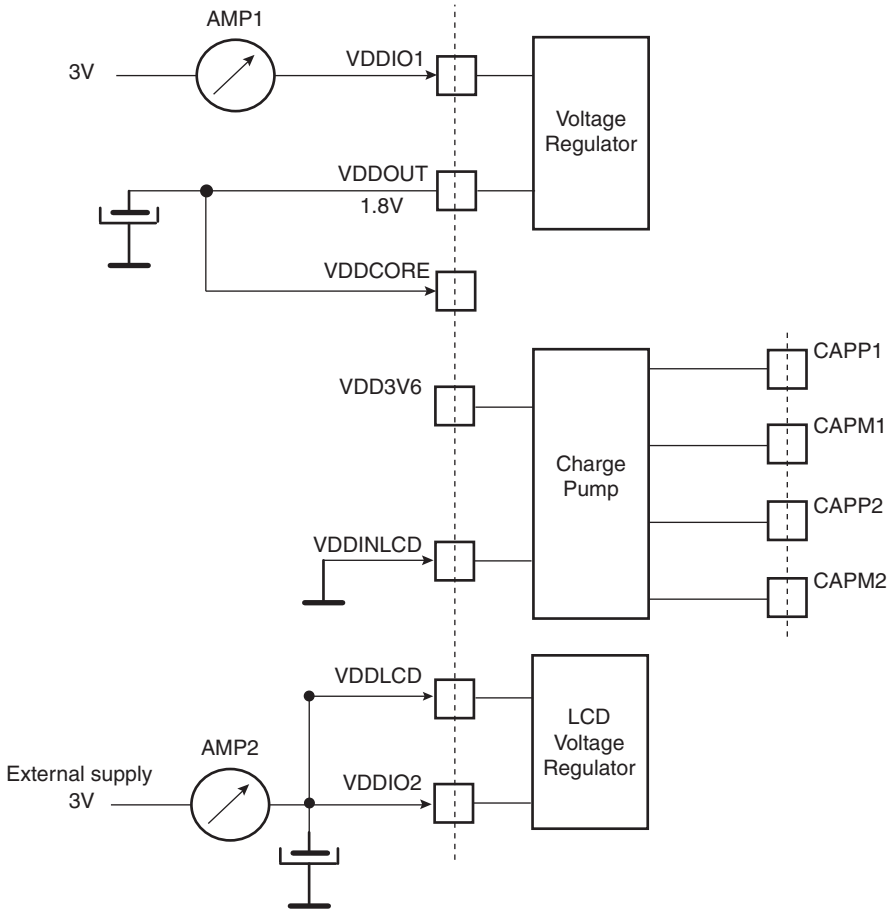
- Typical power consumption of PLLs, Slow Clock and Main Oscillator.
- Power consumption of power supply in different modes: Backup, Wait, Idle, Active and ultra Low-power.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

### 35.3.1 Power Consumption Versus Modes

The values in Table 35-7and Table 35-12 on page 532 are measured values of the power consumption with operating conditions as follows:

- $V_{DDIO1} = V_{DDIO2} = 3V$
- $V_{DDOUTcc}$ set at 1.80V
- $T_A = 25°C$
- There is no consumption on the I/Os of the device

**Figure 35-1.** Measure Schematics



The figures shown below in Table 35-7 represent the power consumption typically measured on the power supplies..

**Table 35-7.** Power Consumption for Low Power Modes (See Figure 35-2 and Figure 35-3)

| Mode | Conditions | VDDIO1 Consumption | Condition | Unit |
|---|---|---|---|---|
| Off Mode (AT91SAM7L128/64) | Only the FWUP pin is supplied | 0.1<br>TBD<br><br>TBD<br>TBD | VDDIO1 = 2.4V @25°C<br>VDDIO1 = 3.0V @25°C<br><br>VDDIO1 = 2.4V @85°C<br>VDDIO1 = 3.0V @85°C | µA |
| Backup Mode (AT91SAM7L128/64) | Voltage Regulator in standby mode<br>RTC OFF<br>Programmable BOD OFF<br>SRAM BACKUP OFF<br>Charge pump OFF<br>LCD Regulator OFF<br>LCD Controller OFF | <br>3.2<br>3.9<br><br>4.31<br>5.34 | <br>VDDIO1 = 2.4V @25°C<br>VDDIO1 = 3.0V @25°C<br><br>VDDIO1 = 2.4V @85°C<br>VDDIO1 = 3.0V @85°C | µA |
| Backup Mode (AT91SAM7L128/64) | Voltage Regulator in standby mode<br>RTC ON<br>Programmable BOD OFF<br>SRAM BACKUP ON<br>Charge pump OFF<br>LCD Regulator OFF<br>LCD Controller OFF | <br>3.3<br>4.14<br><br>5.53<br>6.43 | <br>VDDIO1 = 2.4V @25°C<br>VDDIO1 = 3.0V @25°C<br><br>VDDIO1 = 2.4V @85°C<br>VDDIO1 = 3.0V @85°C | µA |
| Wait Mode (AT91SAM7L128/64) | Voltage Regulator in Deep Mode<br>VDDOUT = 1.55V<br>RTC OFF<br>Programmable BOD OFF<br>FLASH OFF<br>Charge pump OFF<br>LCD Regulator OFF<br>LCD Controller OFF<br>PLL OFF | <br><br>9.57<br>10.04<br><br>17.42<br>18.34 | <br><br>VDDIO1 = 2.4V @25°C<br>VDDIO1 = 3.0V @25°C<br><br>VDDIO1 = 2.4V @85°C<br>VDDIO1 = 3.0V @85°C | µA |
| Idle Mode (AT91SAM7L128/64) | Voltage regulator in Deep mode<br>VDDOUT = 1.55V<br>RTC OFF<br>BOD OFF<br>RC 2MHz OFF<br>Flash is in standby mode.<br>ARM Core in idle mode.<br>MCK @ 500Hz.<br>ADC OFF<br>All peripheral clocks de-activated<br>PLL OFF | <br><br><br>9.76<br>10.6<br><br>19.46<br>20.77 | <br><br><br>VDDIO1 = 2.4V @25°C<br>VDDIO1 = 3.0V @25°C<br><br>VDDIO1 = 2.4V @85°C<br>VDDIO1 = 3.0V @85°C | µA |

**Figure 35-2.** Low-power Modes Consumption @25°C

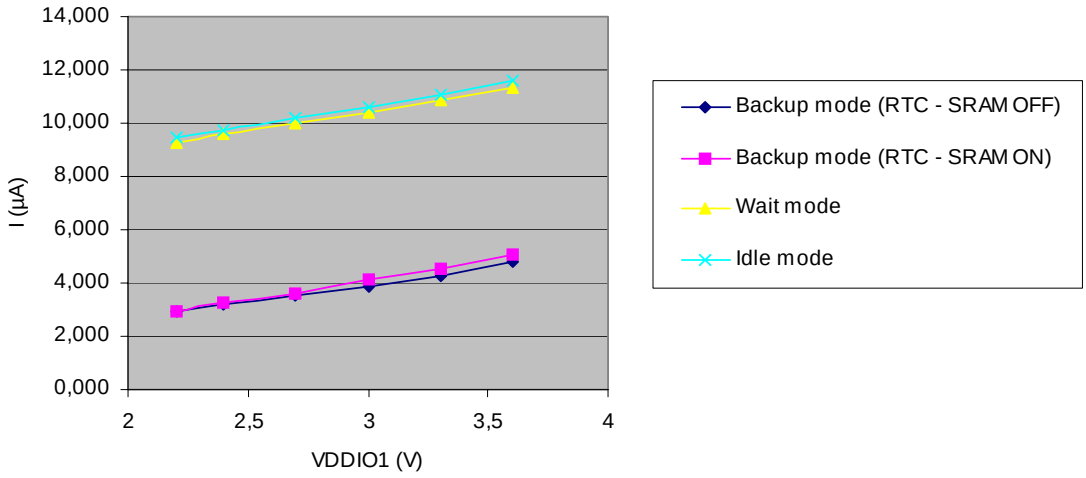
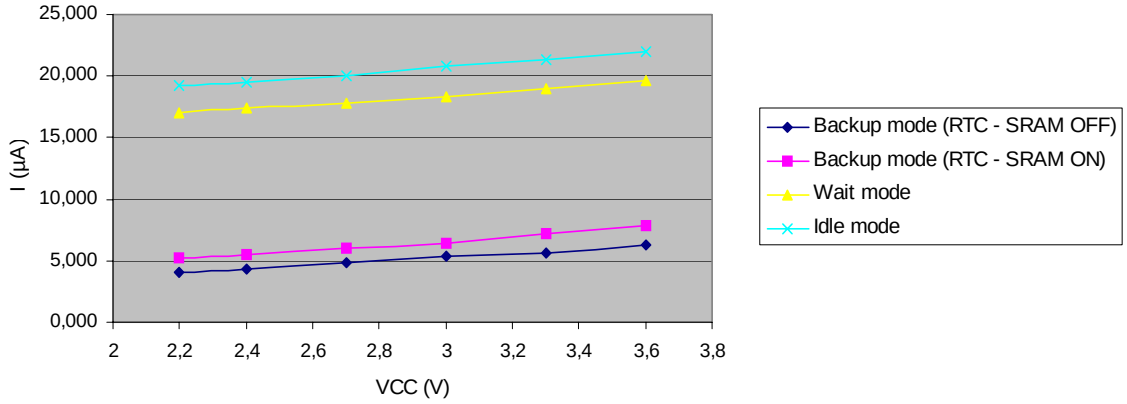



**Figure 35-3.** Low-power Modes Consumption @85°



### 35.3.2 Power Consumption for Active Mode

*35.3.2.1 Low Freaquency*

**Table 35-8.** Low Frequency1

| Mode | Conditions | VDDIO1 Consumption | Condition | Unit |
|---|---|---|---|---|
| Active (AT91SAM7L128/64) (See Figure 35-4) | Voltage regulator in Normal Mode<br>VDDOUT = 1.80V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC ON<br>All peripheral clocks activated<br>RC 2MHz ON<br>PLL OFF | 1.37<br>0.72<br>0.4<br>0.23<br>0.147<br>0.106<br>0.085 | VDDI01= 3V @ 25°C<br>ARM core clock = 2 MHz<br>ARM core clock = 1 MHz<br>ARM core clock = 500 KHz<br>ARM core clock = 250 KHz<br>ARM core clock = 125 KHz<br>ARM core clock = 64 KHz<br>ARM core clock = 32 KHz | mA |
| | | 1.426<br>0.756<br>0.418<br>0.250<br>0.166<br>0.124<br>0.103 | VDDI01= 3V @ 85°C<br>ARM core clock = 2 MHz<br>ARM core clock = 1 MHz<br>ARM core clock = 500 KHz<br>ARM core clock = 250 KHz<br>ARM core clock = 125 KHz<br>ARM core clock = 64 KHz<br>ARM core clock = 32 KHz | |
| Active (AT91SAM7L128/64) (See Figure 35-5) | Voltage regulator in Normal Mode<br>VDDOUT = 1.55V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC ON<br>All peripheral clocks activated<br>RC 2MHz ON<br>PLL OFF | 1.123<br>0.597<br>0.328<br>0.194<br>0.128<br>0.094<br>0.077 | VDDI01= 3V @ 25°C<br>ARM core clock = 2 MHz<br>ARM core clock = 1 MHz<br>ARM core clock = 500 KHz<br>ARM core clock = 250 KHz<br>ARM core clock = 125 KHz<br>ARM core clock = 64 KHz<br>ARM core clock = 32 KHz | mA |
| | | 1.171<br>0.628<br>0.353<br>0.215<br>0.146<br>0.111<br>0.094 | VDDI01= 3V @ 85°C<br>ARM core clock = 2 MHz<br>ARM core clock = 1 MHz<br>ARM core clock = 500 KHz<br>ARM core clock = 250 KHz<br>ARM core clock = 125 KHz<br>ARM core clock = 64 KHz<br>ARM core clock = 32 KHz | |

**Figure 35-4.** Low-range Frequencies, Active Mode Consumption @25°C
(Peripheral Clocks On - PLL Off - RC On - VDD0I1 = 3V)

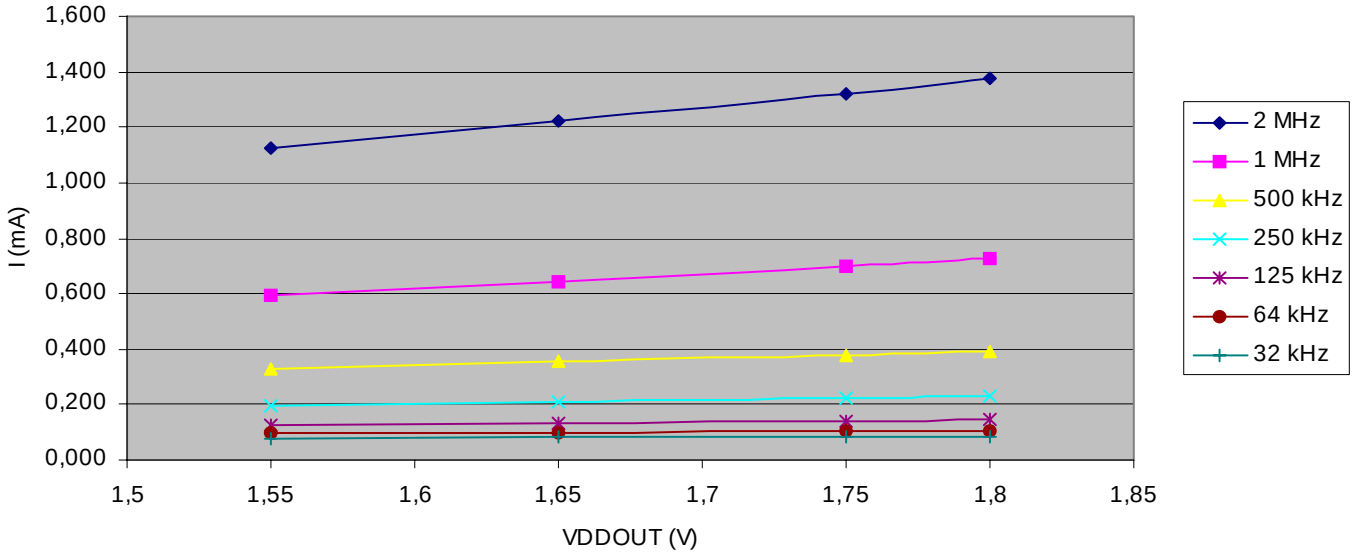



**Figure 35-5.** Low-range Frequencies, Active Mode Consumption @85°C
(Peripheral Clocks On - PLL Off - RC On - VDD0I1 = 3V)

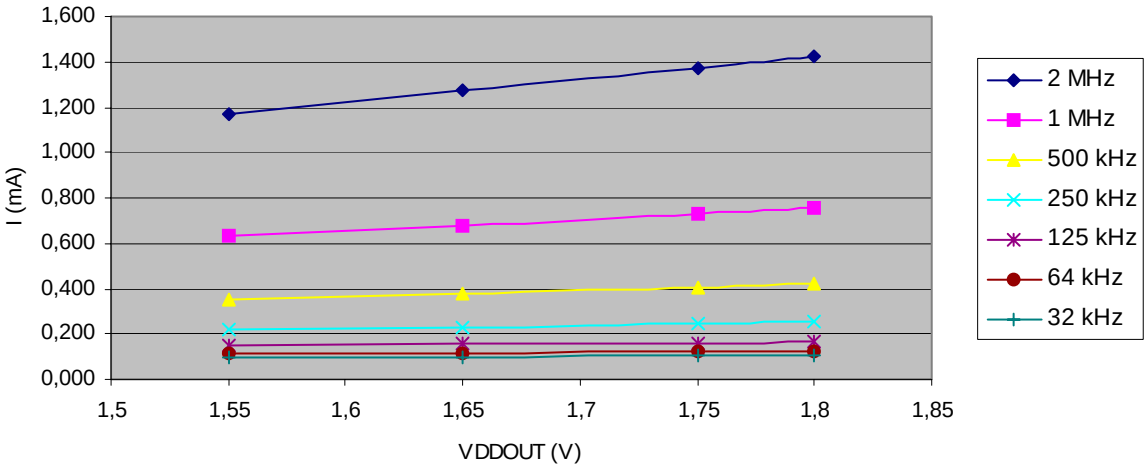


**Table 35-9.** Low Frequency 2 (See Charts that Follow)

| Mode | Conditions | VDDIO1 Consumption | Condition | Unit |
|---|---|---|---|---|
| Active (AT91SAM7L128/64) (See Figure 35-6) | Voltage regulator in Normal Mode VDDOUT = 1.80V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC OFF<br>All peripheral clocks deactivated<br>RC 2MHz ON<br>PLL OFF | | VDDI01= 3V @ 25°C | mA |
| | | 1.192 | ARM core clock = 2 MHz | |
| | | 0.629 | ARM core clock = 1 MHz | |
| | | 0.346 | ARM core clock = 500 KHz | |
| | | 0.206 | ARM core clock = 250 KHz | |
| | | 0.136 | ARM core clock = 125 KHz | |
| | | 0.1 | ARM core clock = 64 KHz | |
| | | 0.082 | ARM core clock = 32 KHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 1.238 | ARM core clock = 2 MHz | |
| | | 0.662 | ARM core clock = 1 MHz | |
| | | 0.372 | ARM core clock = 500 KHz | |
| | | 0.227 | ARM core clock = 250 KHz | |
| | | 0.154 | ARM core clock = 125 KHz | |
| | | 0.118 | ARM core clock = 64 KHz | |
| | | 0.1 | ARM core clock = 32 KHz | |
| Active (AT91SAM7L128/64) (See Figure 35-7) | Voltage regulator in Normal Mode VDDOUT = 1.55V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC OFF<br>All peripheral clocks deactivated<br>RC 2MHz ON<br>PLL OFF | | VDDI01= 3V @ 25°C | mA |
| | | 0.969 | ARM core clock = 2 MHz | |
| | | 0.516 | ARM core clock = 1 MHz | |
| | | 0.287 | ARM core clock = 500 KHz | |
| | | 0.174 | ARM core clock = 250 KHz | |
| | | 0.117 | ARM core clock = 125 KHz | |
| | | 0.089 | ARM core clock = 64 KHz | |
| | | 0.075 | ARM core clock = 32 KHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 1.018 | ARM core clock = 2 MHz | |
| | | 0.545 | ARM core clock = 1 MHz | |
| | | 0.313 | ARM core clock = 500 KHz | |
| | | 0.196 | ARM core clock = 250 KHz | |
| | | 0.136 | ARM core clock = 125 KHz | |
| | | 0.106 | ARM core clock = 64 KHz | |
| | | 0.092 | ARM core clock = 32 KHz | |

**Figure 35-6.** Low-range Frequencies, Active Mode Consumption @25°C
(Peripheral Clocks OFF - PLL Off - RC On - VDD0I1 = 3V)

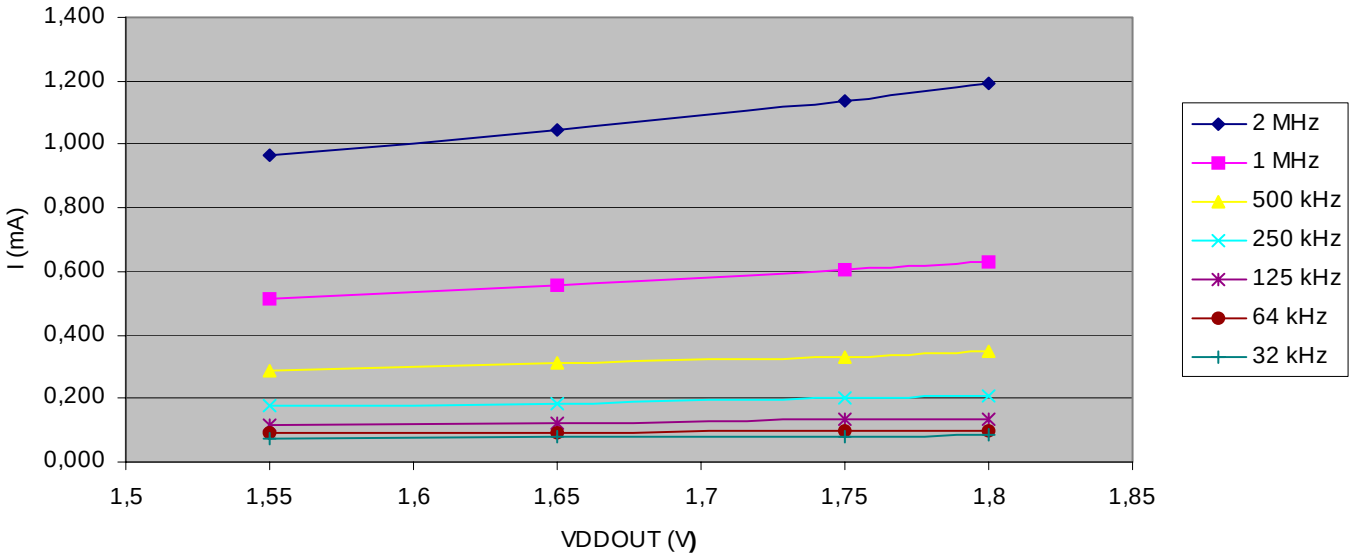
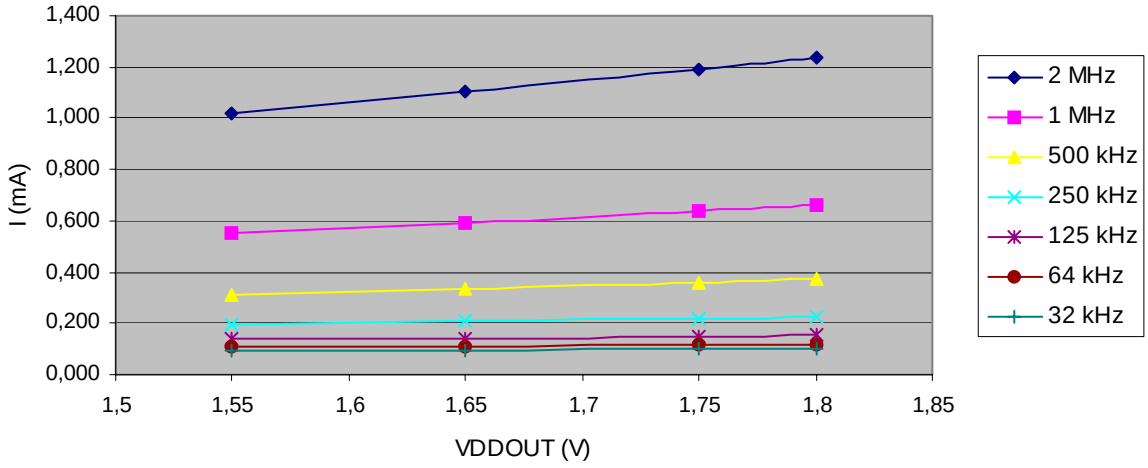



**Figure 35-7.** Low-range Frequencies, Active Mode Consumption @85°C
(Peripheral Clocks Off - PLL OFF - RC On -VDD0I1 = 3V)



### 35.3.2.2 High Frequency

**Table 35-10.** High-range Frequencies, Active Mode (Peripheral Activated)

| Mode | Conditions | VDDIO1 Consumption | Condition | Unit |
|---|---|---|---|---|
| Active<br>(AT91SAM7L128/64)<br>(See Figure 35-8) | Voltage regulator in Normal Mode<br>VDDOUT = 1.80V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC ON<br>All peripheral clocks activated<br>RC 2MHz OFF<br>PLL ON | | VDDI01= 3V @ 25°C | mA |
| | | 3.346 | ARM core clock = 4 MHz | |
| | | 5.37 | ARM core clock = 8 MHz | |
| | | 10.37 | ARM core clock = 16 MHz | |
| | | 11.16 | ARM core clock = 18 MHz | |
| | | 12.13 | ARM core clock = 20 MHz | |
| | | 14.24 | ARM core clock = 24 MHz | |
| | | 19.55 | ARM core clock = 30 MHz | |
| | | 22.63 | ARM core clock = 36 MHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 3.34 | ARM core clock = 4 MHz | |
| | | 5.39 | ARM core clock = 8 MHz | |
| | | 10.39 | ARM core clock = 16 MHz | |
| | | 11.22 | ARM core clock = 18 MHz | |
| | | 12.21 | ARM core clock = 20 MHz | |
| | | 14.35 | ARM core clock = 24 MHz | |
| | | 19.55 | ARM core clock = 30 MHz | |
| | | 22.66 | ARM core clock = 36 MHz | |
| Active<br>(AT91SAM7L128/64)<br>(See Figure 35-9) | Voltage regulator in Normal Mode<br>VDDOUT = 1.55V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC ON<br>All peripheral clocks activated<br>RC 2MHz OFF<br>PLL ON | | VDDI01= 3V @ 25°C | mA |
| | | 2.73 | ARM core clock = 4 MHz | |
| | | 4.47 | ARM core clock = 8 MHz | |
| | | 8.55 | ARM core clock = 16 MHz | |
| | | 9.23 | ARM core clock = 18 MHz | |
| | | 10.1 | ARM core clock = 20 MHz | |
| | | 11.9 | ARM core clock = 24 MHz | |
| | | 16 | ARM core clock = 30 MHz | |
| | | 18.6 | ARM core clock = 36 MHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 2.76 | ARM core clock = 4 MHz | |
| | | 4.52 | ARM core clock = 8 MHz | |
| | | 8.62 | ARM core clock = 16 MHz | |
| | | 9.33 | ARM core clock = 18 MHz | |
| | | 10.21 | ARM core clock = 20 MHz | |
| | | 12.04 | ARM core clock = 24 MHz | |
| | | 16.07 | ARM core clock = 30 MHz | |
| | | 18.75 | ARM core clock = 36 MHz | |

**Figure 35-8.** High-range Frequencies, Active Mode Consumption @25°C
(Peripheral Clocks On - PLL On - RC Off -VDD0I1 = 3V)

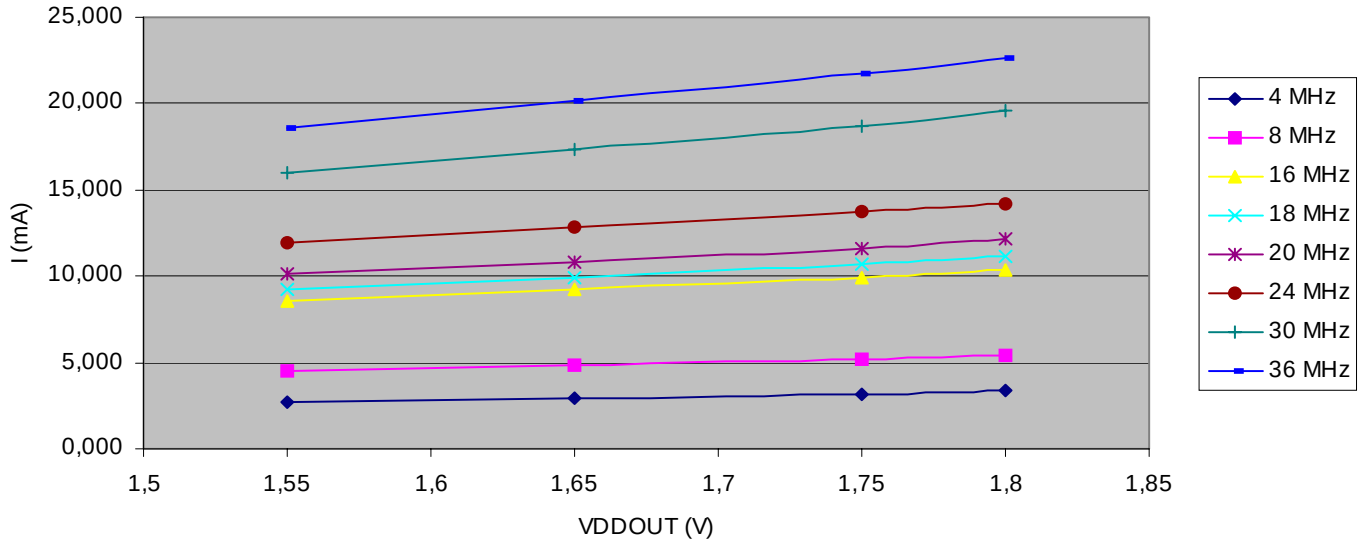



**Figure 35-9.** High-range Frequencies, Active Mode Consumption @85°C
(Peripheral Clocks On - PLL On - RC Off -VDD0I1 = 3V)

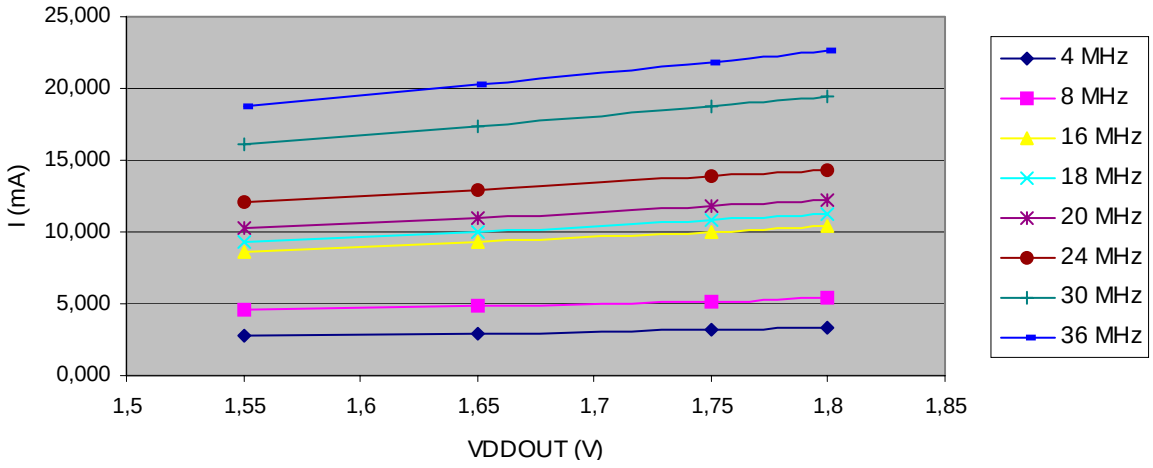


**Table 35-11.** High-range Frequencies, Active Mode

| Mode | Conditions | VDDIO1 Consumption | Condition | Unit |
|---|---|---|---|---|
| Active<br>(AT91SAM7L128/64)<br>(See Figure 35-10) | Voltage regulator in Normal Mode<br>VDDOUT = 1.80V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC OFF<br>All peripheral clocks deactivated<br>RC 2MHz OFF<br>PLL ON | | VDDI01= 3V @ 25°C | mA |
| | | 2.927 | ARM core clock = 4 MHz | |
| | | 4.524 | ARM core clock = 8 MHz | |
| | | 8.693 | ARM core clock = 16 MHz | |
| | | 9.305 | ARM core clock = 18 MHz | |
| | | 10.08 | ARM core clock = 20 MHz | |
| | | 11.75 | ARM core clock = 24 MHz | |
| | | 16.38 | ARM core clock = 30 MHz | |
| | | 18.81 | ARM core clock = 36 MHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 2.923 | ARM core clock = 4 MHz | |
| | | 4.548 | ARM core clock = 8 MHz | |
| | | 8.707 | ARM core clock = 16 MHz | |
| | | 9.336 | ARM core clock = 18 MHz | |
| | | 10.15 | ARM core clock = 20 MHz | |
| | | 11.83 | ARM core clock = 24 MHz | |
| | | 16.31 | ARM core clock = 30 MHz | |
| | | 18.81 | ARM core clock = 36 MHz | |
| Active<br>(AT91SAM7L128/64)<br>(See Figure 35-11) | Voltage regulator in Normal Mode<br>VDDOUT = 1.55V<br>RTC ON<br>Programmable BOD ON (Continuous)<br>Charge pump ON<br>LCD Regulator ON<br>Flash is read<br>ADC OFF<br>All peripheral clocks deactivated<br>RC 2MHz OFF<br>PLL ON | | VDDI01= 3V @ 25°C | mA |
| | | 2.377 | ARM core clock = 4 MHz | |
| | | 3.756 | ARM core clock = 8 MHz | |
| | | 7.120 | ARM core clock = 16 MHz | |
| | | 7.651 | ARM core clock = 18 MHz | |
| | | 8.335 | ARM core clock = 20 MHz | |
| | | 9.763 | ARM core clock = 24 MHz | |
| | | 13.27 | ARM core clock = 30 MHz | |
| | | 15.34 | ARM core clock = 36 MHz | |
| | | | VDDI01= 3V @ 85°C | |
| | | 2.398 | ARM core clock = 4 MHz | |
| | | 3.797 | ARM core clock = 8 MHz | |
| | | 7.182 | ARM core clock = 16 MHz | |
| | | 7.729 | ARM core clock = 18 MHz | |
| | | 8.424 | ARM core clock = 20 MHz | |
| | | 9.882 | ARM core clock = 24 MHz | |
| | | 13.33 | ARM core clock = 30 MHz | |
| | | 15.47 | ARM core clock = 36 MHz | |

**Figure 35-10.** High-range Frequencies, Active Mode Consumption @25°C
(Peripheral Clocks Off - PLL On - RC Off - VDD0I1 = 3V)

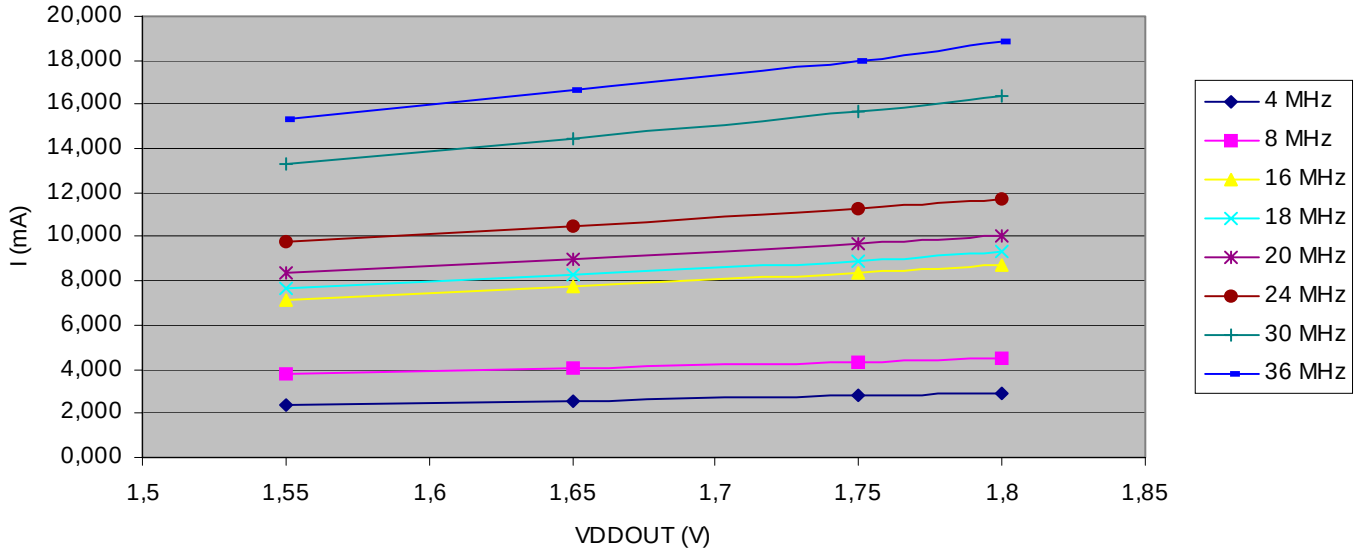
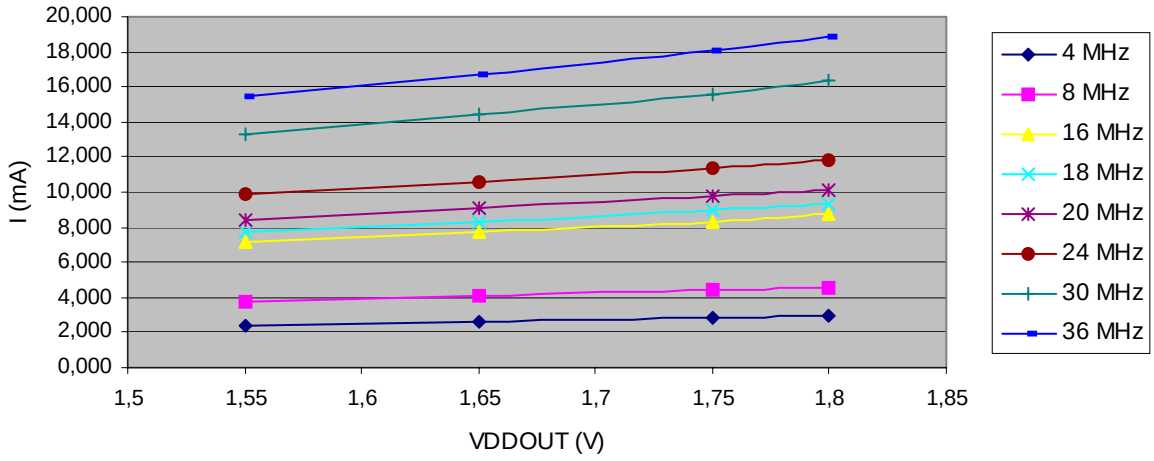



**Figure 35-11.** High-range Frequencies, Active Mode Consumption @85°C
(Peripheral Clocks Off - PLL On - RC Off - VDD0I1 = 3V)



### 35.3.3 Peripheral Power Consumption in Active Mode

**Table 35-12.** Power Consumption on $V_{DDCORE}$[1]

| Peripheral | Consumption (Typ) | Unit |
|---|---|---|
| PIO Controller | 9 | µA/MHz |
| USART | 24 | |
| PWM | 13 | |
| TWI | 17 | |
| SPI | 12 | |
| Timer Counter Channels | 6 | |
| ADC | 8 | |
| ARM7TDMI | 160 | |
| System Peripherals (AT91SAM7L128/64) | 6 | |

Note: 1. Note: VDDIO1= 2.4V, $V_{DDCORE}$ = 1.80V, $T_A$ = 25°C

## 35.4 Crystal Oscillators Characteristics

### 35.4.1 32 kHz RC Oscillator Characteristics

**Table 35-13.** 32 KHz RC Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{VDDIO1}$ | Supply Voltage | VDDIO1 domain | 1.8 | 2.7 | 3.6 | V |
| | RC Oscillator Frequency | | 20 | 32 | 44 | kHz |
| | Frequency Supply Dependency | Typical @ 2.7V | -5 | | 5 | % |
| | Frequency Temperature Dependency | Typical @ 25°C | -7 | | 7 | % |
| | Duty Cycle | | 45 | 50 | 55 | % |
| $t_{ST}$ | Startup Time | $V_{VDDIO1}$ = 1.65V | | | 100 | µs |
| $I_{OSC}$ | Current Consumption | After Startup Time<br>Temp. Range = -40°C to +85°C<br>Typical Consumption at 2.2V supply and Temp = 25°C | | 370 | 780 | nA |
| | Standby Consumption | | | | 0.02 | µA |

### 35.4.2 2 MHz RC Oscillator Characteristics

**Table 35-14.** 2 MHz RC Oscillator Characteristics

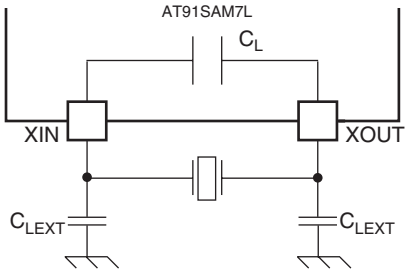
| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| | Supply Voltage | VDDCORE domain | 1.2 | 1.6 | 1.85 | V |
| $1/(t_{CPRC})$ | RC Oscillator Frequency | | 1.35 | 2 | 2.65 | MHz |
| | Frequency Supply Dependency | $VDD_{CORE}$<br>1.2V < 1.6V < 1.95V<br>1.65V < 1.8V < 1.95V<br>1.2V < 1.3V < 1.4V | <br>-3<br>-1.5<br>-1.5 | | <br>3<br>1.5<br>1.5 | % |
| | Frequency Temperature Dependency | Typical @ 25°C | -10 | | +10 | % |
| | Duty Cycle | | 45 | 50 | 55 | % |
| $t_{ST}$ | Startup Time | Frequency > 1MHz | 2 | | 5 | µs |
| $I_{OSC}$ | Current Consumption | After Startup Time | | 18 | 30 | µA |
| | Standby Consumption | | | | 1 | µA |

### 35.4.3 XTAL Oscillator Characteristics

**Table 35-15.** XTAL Oscillator Characteristics

| Symbol | Parameter | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $F_{req}$ | Operating Frequency | Normal mode with crystal | | | | 32.768 | KHz |
| | Supply Voltage | VDDIO1 Domain | | 1.8 | | 3.6 | V |
| | Duty Cycle | | | 40 | 50 | 60 | % |
| | Startup Time | Rs < 50KΩ<br><br>Rs < 100KΩ (1) | CL = 12.5pF<br>CL = 6pF<br>CL = 12.5pF<br>CL = 6pF | | | 900<br>300<br>1200<br>500 | ms |
| | Current consumption | Rs < 50KΩ<br><br>Rs < 100KΩ (1) | CL = 12.5pF<br>CL = 6pF<br>CL = 12.5pF<br>CL = 6pF | | 650<br>450<br>900<br>650 | 1400<br>1200<br>1600<br>1400 | nA |
| $I_{DDST}$ | Standby Current Consumption | Standby mode @ 3.6V | | | | 5 | nA |
| $P_{ON}$ | Drive level | | | | | 0.1 | µW |
| $R_f$ | Internal resistor | between XIN and XOUT | | | 10 | | MΩ |
| $C_{LEXT}$ | Maximum external capacitor on XIN and XOUT | | | | | 20 | pF |
| $C_L$ | Internal Equivalent Load Capacitance | Integrated Load Capacitance (XIN and XOUT in series) | | 2.0 | 2.5 | 3.0 | pF |

Notes: 1. $R_S$ is the series resitor..




### 35.4.4 Crystal Characteristics

**Table 35-16.** Crystal Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| ESR | Equivalent Series Resistor Rs | Crystal @ 32.768KHz | | 50 | 100 | KΩ |
| $C_M$ | Motional capacitance | Crystal @ 32.768KHz | 0.6 | | 3 | fF |
| $C_{SHUNT}$ | Shunt capacitance | Crystal @ 32.768KHz | 0.6 | | 2 | pF |

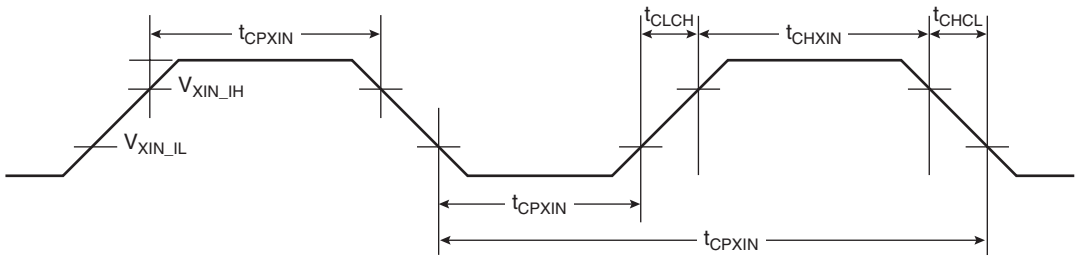
### 35.4.5 XIN Clock Characteristics

**Table 35-17.** XIN Clock Electrical Characteristics (In bypass mode)

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $1/(t_{CPXIN})$ | XIN Clock Frequency | (1) | | 10 | MHz |
| $1/(t_{CPXIN})$ | XIN Clock Frequency | (2) | | 44 | kHz |
| $t_{CPXIN}$ | XIN Clock Period | (1) | 100 | | ns |
| $t_{CPXIN}$ | XIN Clock Period | (2) | 44 | | ns |
| $t_{CHXIN}$ | XIN Clock High Half-period | (1) | 22 | | µs |
| $t_{CHXIN}$ | XIN Clock High Half-period | (2) | 11 | | µs |
| $t_{CLXIN}$ | XIN Clock Low Half-period | (1) | 50 | | ns |
| $t_{CLXIN}$ | XIN Clock Low Half-period | (2) | 11 | | µs |
| $t_{CLCH}$ | Rise Time | | 400 | | ns |
| $t_{CHCL}$ | Fall Time | | 400 | | ns |
| $C_{IN}$ | XIN Input Capacitance | | | 6 | pF |
| $R_{IN}$ | XIN Pull-down Resistor | | 3 | 5 | MΩ |
| $V_{XIN_IL}$ | $V_{XIN}$ Input Low-level Voltage | | -0.3 | 0.2 x $V_{VDDIO1}$ | V |
| $V_{XIN_IH}$ | $V_{XIN}$ Input High-level Voltage | | 0.8 x $V_{VDDIO1}$ | $V_{VDDIO1}$+0.3 | V |

Note: 1. These characteristics apply only in FFPI mode

2. These characteristics apply only when the XTAL Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR register, see the Clock Generator Main Oscillator Register.



### 35.4.6 External Clock CLKIN Characteristics

**Table 35-18.** External Clock CLKIN Characteristics (VDDCORE set at 1.80V)

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/(t_{CPCLKIN})$ | CLKIN Clock Frequency | | 32 | MHz |
| $t_{CPCLKIN}$ | CLKIN Clock Period | 31.0 | | ns |
| $t_{CHCLKIN}$ | CLKIN Clock High Half-period | 14.5 | | ns |
| $t_{CLCLKIN}$ | CLKIN Clock Low Half-period | 14.3 | | ns |

**Table 35-19.** External Clock CLKIN Characteristics (VDDCORE set at 1.75V)

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/(t_{CPCLKIN})$ | CLKIN Clock Frequency | | 30.8 | MHz |
| $t_{CPCLKIN}$ | CLKIN Clock Period | 32.4 | | ns |
| $t_{CHCLKIN}$ | CLKIN Clock High Half-period | 15.2 | | ns |
| $t_{CLCLKIN}$ | CLKIN Clock Low Half-period | 15.0 | | ns |

**Table 35-20.** Externa Clock CLKIN Characteristics (VDDCORE set at 1.65V)

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/(t_{CPCLKIN})$ | CLKIN Clock Frequency | | 28 | MHz |
| $t_{CPCLKIN}$ | CLKIN Clock Period | 35.7 | | ns |
| $t_{CHCLKIN}$ | CLKIN Clock High Half-period | 16.7 | | ns |
| $t_{CLCLKIN}$ | CLKIN Clock Low Half-period | 16.5 | | ns |

**Table 35-21.** External Clock CLKIN Characteristics (VDDCORE set at 1.55V)

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/(t_{CPCLKIN})$ | CLKIN Clock Frequency | | 25 | MHz |
| $t_{CPCLKIN}$ | CLKIN Clock Period | 40.0 | | ns |
| $t_{CHCLKIN}$ | CLKIN Clock High Half-period | 18.7 | | ns |
| $t_{CLCLKIN}$ | CLKIN Clock Low Half-period | 18.5 | | ns |

## 35.5 PLL Characteristics

**Table 35-22.** Phase Lock Loop Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Vdd | Supply Voltage | Supplied by VDDCORE | 1.60 | | | V |
| $F_{OUT}$ | Output Frequency | | 18 | 30 | 47 | MHz |
| $F_{IN}$ | Input Frequency | Connected to SCLK | 20 | 30 | 44 | KHz |
| $I_{PLL}$ | Current Consumption | Active mode @ 20MHz @1.8V<br>Active mode @ 30MHz @1.8V<br>Active mode @ 40MHz @1.8V | | 445<br>490<br>535 | 505<br>555<br>605 | µA |
| | | Standby mode | | 0.005 | 0.5 | µA |

Note: Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

## 35.6 ADC Characteristics

**Table 35-23.** Channel Conversion Time and ADC Clock

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| ADC Clock Frequency | 10-bit resolution mode | | | 6 | MHz |
| ADC Clock Frequency | 8-bit resolution mode | | | 10 | MHz |
| Startup Time | Return from Idle Mode | | | 15 | µs |
| Track and Hold Acquisition Time | | 500 | | | ns |
| Conversion Time | ADC Clock = 6MHz<br>ADC Clock = 10MHz | | | 1.67<br>1 | µs |
| Throughput Rate | ADC Clock = 6MHz<br>ADC Clock = 10MHz | | | 460(1)<br>660(2) | kSPS |

Notes: 1. Corresponds to 13 clock cycles at 6 MHz: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

2. Corresponds to 15 clock cycles at 10 MHz: 5 clock cycles for track and hold acquisition time and 10 clock cycles for conversion

**Table 35-24.** External Voltage Reference Input

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| ADVREF Input Voltage Range | | 1.65 | 1.8 | VDDCORE | V |
| ADVREF Average Current | ADC Clock = 6MHz | | | 250 | µA |
| Current Consumption on VDDCORE | | | | 2.2 | mA |

The user can drive ADC input with impedance up to:

- $Z_{OUT} \leq$ (SHTIM -440) x 20 in 8-bit resolution mode
- $Z_{OUT} \leq$ (SHTIM -550) x 16.6 in 10-bit resolution mode

with SHTIM (Sample and Hold Time register) expressed in ns and $Z_{OUT}$ expressed in ohms.

**Table 35-25.** Analog Inputs

| Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|
| Input Voltage Range | 0 | | $V_{ADVREF}$ | |
| Input Leakage Current | | | ±0.5 | µA |
| Input Capacitance | | 6.5 | 8.5 | pF |

**Table 35-26.** Transfer Characteristics

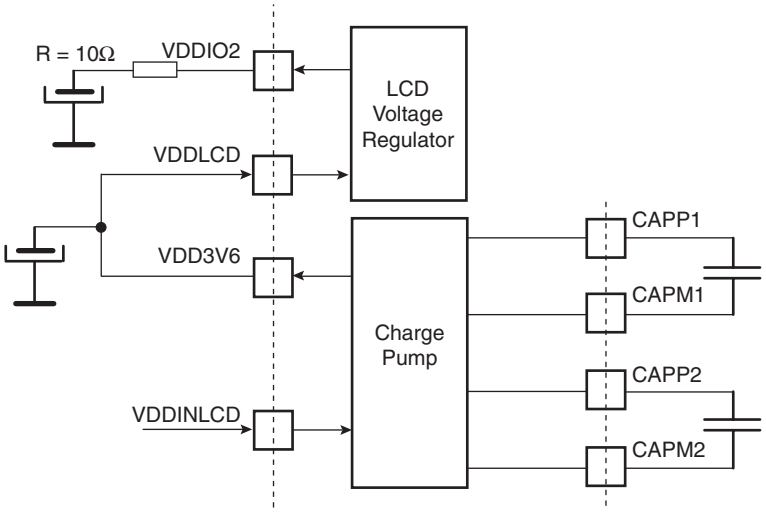
| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Resolution | | | 10 | | Bit |
| Integral Non-linearity | | | | ±2 | LSB |
| Differential Non-linearity | No missing code | | | ±1 | LSB |
| Offset Error | | | | ±3 | LSB |
| Gain Error | | | | ±2 | LSB |
| Absolute accuracy | | | | ±4.2 | LSB |

## 35.7 Regulated Charge Pump Characteristics

**Table 35-27.** Regulated Charge Pump Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDINLCD}$ | Charge Pump Supply Voltage | | 1.8 | 2.7 | 3.6 | V |
| $V_{VDD3V6}$ | Output Voltage | $I_O$ = 4 mA max | | | 3.6 | V |
| $I_{VDDINLCD}$ | Current consumption | Active, No load, with clock, CL=4.7µF,ESR=1Ω | | | | µA |
| | | Onto VDDIO1 =1.8V | | | 50 | |
| | | Onto VDDINLCD = 3.6V | | | 250 | |
| | | Onto VDDIO1 = 1.8V | | | 50 | |
| | | Onto VDDINLCD = 3.6V | | | 65 | |
| $T_{START}$ | Startup Time | | | | 4 | mS |
| | External charge capacitor | Between CAPP1 and CAPM1 (Tolerance +/- 10%) | | 220 | | nF |
| | External charge capacitor | Between CAPP2 and CAPM2 (Tolerance +/- 10%) | | 220 | | nF |
| | External storage capacitor | On VDD3V6 (Tolerance +/- 10%,ESR =<1Ω) | | 4.7 | | µF |

**Figure 35-12.** Charge Pump Schematics




## 35.8 LCD Voltage Characteristic

**Table 35-28.** LCD Voltage Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDLCD}$ | Supply Voltage | | 2.5 | | 3.6 | V |
| | Dropout Voltage | Minimum voltage difference needed between supply voltage and external output voltage selected | 100 | | | mV |
| $V_{VDDIO2}$ | Output Voltage | $I_O$ = 4 mA max | 2.4 | 2.9 | 3.4 | V |
| $I_{VDDLCD}$ | Current consumption | CL=4.7µF,ESR=1Ω<br>No load | | | 30 | µA |
| $T_{START}$ | LCD Startup Time | | | | 1.5 | mS |
| | External storage capacitor | On VDDIO2<br>Tolerance +/- 10%, ESR=1Ω min | | 4.7 | | µF |

## 35.9 LCD Driver Characteristic

**Table 35-29.** LCD Driver Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDIO2}$ | Supply Voltage | | 2.4 | | 3.4 | V |
| $I_{VDDLCD}$ | Current consumption | Resistor Ladder @3.4V<br><br>Each output buffer @3.4V | | | 1.34<br><br>33 | µA |

## 35.10 AC Characteristics

### 35.10.1 Master Clock Characteristics

Master Clock Waveform Parameters

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $1/(t_{CPMCK})$ | Master Clock Frequency | VDDCORE set at 1.55V<br><br>VDDIO1= VDDIO2 = 1.8V<br>VDDIO1= VDDIO2 = 2.5V<br>VDDIO1= VDDIO2 = 3.0V | | <br><br>25<br>30<br>32 | MHz |
| $1/(t_{CPMCK})$ | Master Clock Frequency | VDDCORE set at 1.65V<br><br>VDDIO1= VDDIO2 = 1.8V<br>VDDIO1= VDDIO2 = 2.5V<br>VDDIO1= VDDIO2 = 3.0V | | <br><br>28<br>34<br>36 | MHz |
| $1/(t_{CPMCK})$ | Master Clock Frequency | VDDCORE set at 1.75V<br><br>VDDIO1= VDDIO2 = 1.8V<br>VDDIO1= VDDIO2 = 2.5V<br>VDDIO1= VDDIO2 = 3.0V | | <br><br>30<br>36<br>38 | MHz |
| $1/(t_{CPMCK})$ | Master Clock Frequency | VDDCORE set at 1.80V<br><br>VDDIO1= VDDIO2 = 1.8V<br>VDDIO1= VDDIO2 = 2.5V<br>VDDIO1= VDDIO2 = 3.0V | | <br><br>30.8<br>37.5<br>39.7 | MHz |

### 35.10.2 I/O Characteristics

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (30%-70%)
- minimum output swing: 100mV to **VDDIO1** - 100 mV
- minimum output swing: 100mV to **VDDIO2** - 100 mV
- Addition of rising and falling time inferior to 75% of the period

**Table 35-30.** I/O Characteristics

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| FreqMax1 | Pin Group 1 (1) Maximum output frequency | Load: 25 pF<br>$V_{DDIO1}$ = 1.8V<br>$V_{DDIO1}$ = 2.5V<br>$V_{DDIO1}$ = 3V | | <br>20<br>33<br>37 | MHz |
| $PulseminH_1$ | Pin Group 1 (1) High Level Pulse Width | Load: 25pF<br>$V_{DDIO1}$ = 1.8V<br>$V_{DDIO1}$ = 2.5V<br>$V_{DDIO1}$ = 3V | <br>25<br>15<br>13 | | ns |
| $PulseminL_1$ | Pin Group 1 (1) Low Level Pulse Width | Load: 25 pF<br>$V_{DDIO1}$ = 1.8V<br>$V_{DDIO1}$ = 2.5V<br>$V_{DDIO1}$ = 3V | <br>25<br>15<br>13 | | ns |
| FreqMax2 | Pin Group 2 (2) Maximum output frequency | Load: 25 pF<br>$V_{DDIO2}$ = 1.8V<br>$V_{DDIO2}$ = 2.5V<br>$V_{DDIO2}$ = 3V | | <br>20<br>29<br>36 | MHz |
| $PulseminH_2$ | Pin Group 2 (2) High Level Pulse Width | Load: 25pF<br>$V_{DDIO2}$ = 1.8V<br>$V_{DDIO2}$ = 2.5V<br>$V_{DDIO2}$ = 3V | <br>25<br>17<br>14 | | ns |
| $PulseminL_2$ | Pin Group 2 (2) Low Level Pulse Width | Load: 25pF<br>$V_{DDIO2}$ = 1.8V<br>$V_{DDIO2}$ = 2.5V<br>$V_{DDIO2}$ = 3V | <br>25<br>17<br>14 | | ns |

Notes: 1. Pin Group 1 = PC0-PC29
2. Pin Group 2 = PA0-PA25, PB0-PB23

### 35.10.3 SPI Characteristics

**Figure 35-13.** SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)

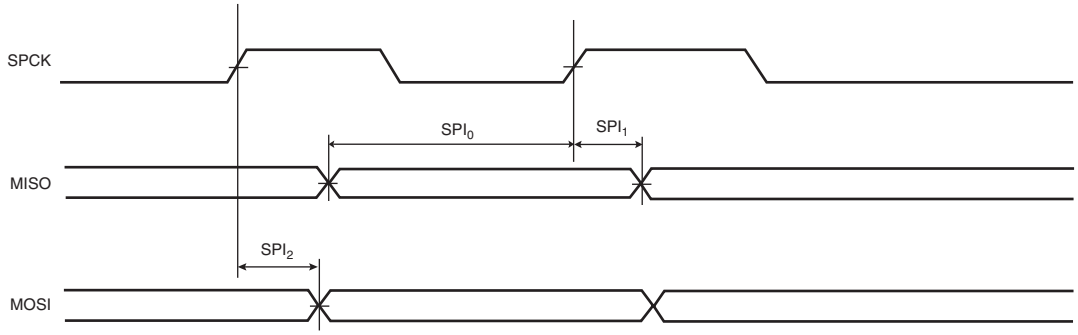



**Figure 35-14.** SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)

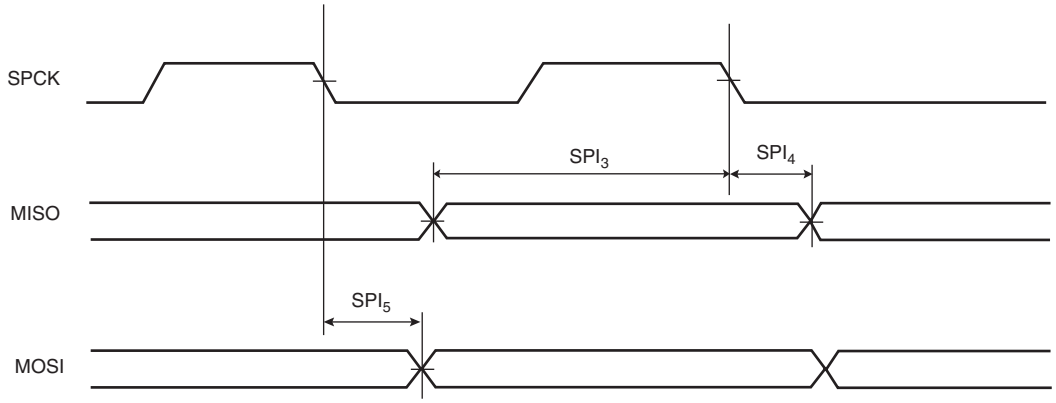



**Figure 35-15.** SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

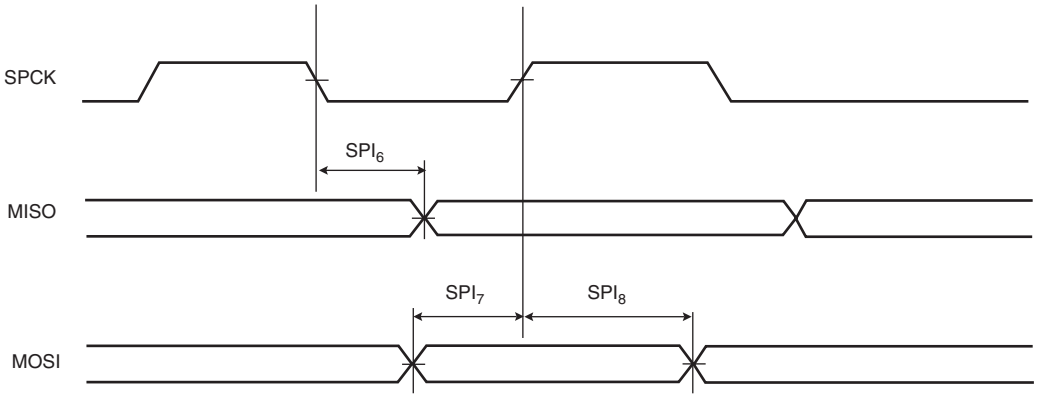


**Figure 35-16.** SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

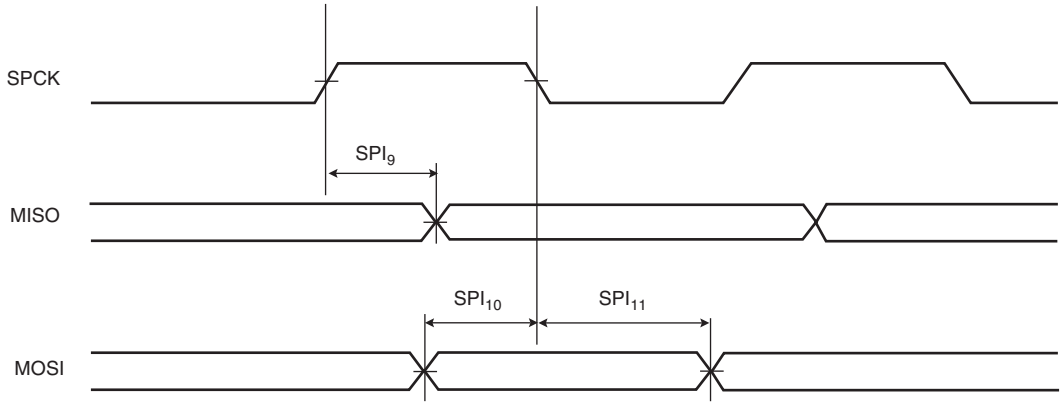



**Table 35-31.** AT91SAM7L128/64 SPI Timings

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $SPI_0$ | MISO Setup time before SPCK rises (master) | 3.3V domain[1] | 26 + ($t_{CPMCK}$)/2[3] | | ns |
| | | 1.8V domain[2] | 34 + ($t_{CPMCK}$)/2[3] | | ns |
| $SPI_1$ | MISO Hold time after SPCK rises (master) | 3.3V domain[1] | 0 | | ns |
| | | 1.8V domain[2] | 0 | | ns |
| $SPI_2$ | SPCK rising to MOSI Delay (master) | 3.3V domain[1] | | 7 | ns |
| | | 1.8V domain[2] | | 10 | ns |
| $SPI_3$ | MISO Setup time before SPCK falls (master) | 3.3V domain[1] | 26 + ($t_{CPMCK}$)/2[3] | | ns |
| | | 1.8V domain[2] | 34 + ($t_{CPMCK}$)/2[3] | | ns |
| $SPI_4$ | MISO Hold time after SPCK falls (master) | 3.3V domain[1] | 0 | | ns |
| | | 1.8V domain[2] | 0 | | ns |
| $SPI_5$ | SPCK falling to MOSI Delay (master) | 3.3V domain[1] | | 7 | ns |
| | | 1.8V domain[2] | | 10 | ns |
| $SPI_6$ | SPCK falling to MISO Delay (slave) | 3.3V domain[1] | | 22.5 | ns |
| | | 1.8V domain[2] | | 30.5 | ns |
| $SPI_7$ | MOSI Setup time before SPCK rises (slave) | 3.3V domain[1] | 1 | | ns |
| | | 1.8V domain[2] | 2.5 | | ns |
| $SPI_8$ | MOSI Hold time after SPCK rises (slave) | 3.3V domain[1] | 2 | | ns |
| | | 1.8V domain[2] | 2 | | ns |
| $SPI_9$ | SPCK rising to MISO Delay (slave) | 3.3V domain[1] | | 23 | ns |
| | | 1.8V domain[2] | | 28 | ns |
| $SPI_{10}$ | MOSI Setup time before SPCK falls (slave) | 3.3V domain[1] | 1 | | ns |
| | | 1.8V domain[2] | 1 | | |
| $SPI_{11}$ | MOSI Hold time after SPCK falls (slave) | 3.3V domain[1] | 2 | | ns |
| | | 1.8V domain[2] | 2 | | ns |

Notes: 1. 3.3V domain: $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 25 pF.
2. 1.8V domain: $V_{VDDIO}$ from 1.65V to 1.95V, maximum external capacitor = 25 pF.
3. $t_{CPMCK}$: Master Clock period in ns.

Note that in SPI master mode the AT91SAM7L128/64 does not sample the data (MISO) on the opposite edge where data clocks out (MOSI) but the same edge is used as shown in Figure 35-13 and Figure 35-14.

### 35.10.4 Embedded Flash Characteristics

The maximum operating frequency is given in tables 35-32, 35-33, 35-34 and 35-35 below but is limited by the Embedded Flash access time when the processor is fetching code out of it. The tables 35-32, 35-33, 35-34 and 35-35 below give the device maximum operating frequency depending on the field FWS of the MC_FMR register. This field defines the number of wait states required to access the Embedded Flash Memory.

**Table 35-32.** Embedded Flash Wait State (VDDCORE set at 1.80V, minimum 1.65V)

| FWS | Read Operations | Maximum Operating Frequency (MHz) |
|---|---|---|
| 0 | 1 cycle | 17.2 |
| 1 | 2 cycles | 30 |
| 2 | 3 cycles | 30 |
| 3 | 4 cycles | 39.7 |

**Table 35-33.** Embedded Flash Wait States (VDDCORE set at 1.75V, minimum 1.70V)

| FWS | Read Operations | Maximum Operating Frequency (MHz) |
|---|---|---|
| 0 | 1 cycle | 16.5 |
| 1 | 2 cycles | 28.6 |
| 2 | 3 cycles | 28.6 |
| 3 | 4 cycles | 38 |

**Table 35-34.** Embedded Flash Wait States (VDDCORE set at 1.65V, minimum 1.60V)

| FWS | Read Operations | Maximum Operating Frequency (MHz) |
|---|---|---|
| 0 | 1 cycle | 15 |
| 1 | 2 cycles | 26 |
| 2 | 3 cycles | 26 |
| 3 | 4 cycles | 36 |

**Table 35-35.** Embedded Flash Wait States (VDDCORE set at 1.55V, minimum 1.50V)

| FWS | Read Operations | Maximum Operating Frequency (MHz) |
|---|---|---|
| 0 | 1 cycle | 13.4 |
| 1 | 2 cycles | 23.2 |
| 2 | 3 cycles | 23.2 |
| 3 | 4 cycles | 32 |

**Table 35-36.** AC Flash Characteristics

| Parameter | Conditions | Typ | Max | Units |
|---|---|---|---|---|
| Program Cycle Time | per page including auto-erase | 4.4 | 4.6 | ms |
| | per page without auto-erase | 2.2 | 2.3 | ms |
| Full Chip Erase | | 10 | | ms |
| Power-up delay | | | 50 | µs |

### 35.10.5 JTAG/ICE Timings

#### *35.10.5.1 ICE Interface Signals*

**Table 35-37.** ICE Interface Timing Specification

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $ICE_0$ | TCK Low Half-period | (1) | 51 | | ns |
| $ICE_1$ | TCK High Half-period | (1) | 51 | | ns |
| $ICE_2$ | TCK Period | (1) | 102 | | ns |
| $ICE_3$ | TDI, TMS, Setup before TCK High | (1) | 0 | | ns |
| $ICE_4$ | TDI, TMS, Hold after TCK High | (1) | 3 | | ns |
| $ICE_5$ | TDO Hold Time | (1) | 13 | | ns |
| $ICE_6$ | TCK Low to TDO Valid | (1) | | 20 | ns |

Note: 1. $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 25pF.

**Figure 35-17.** ICE Interface Signals

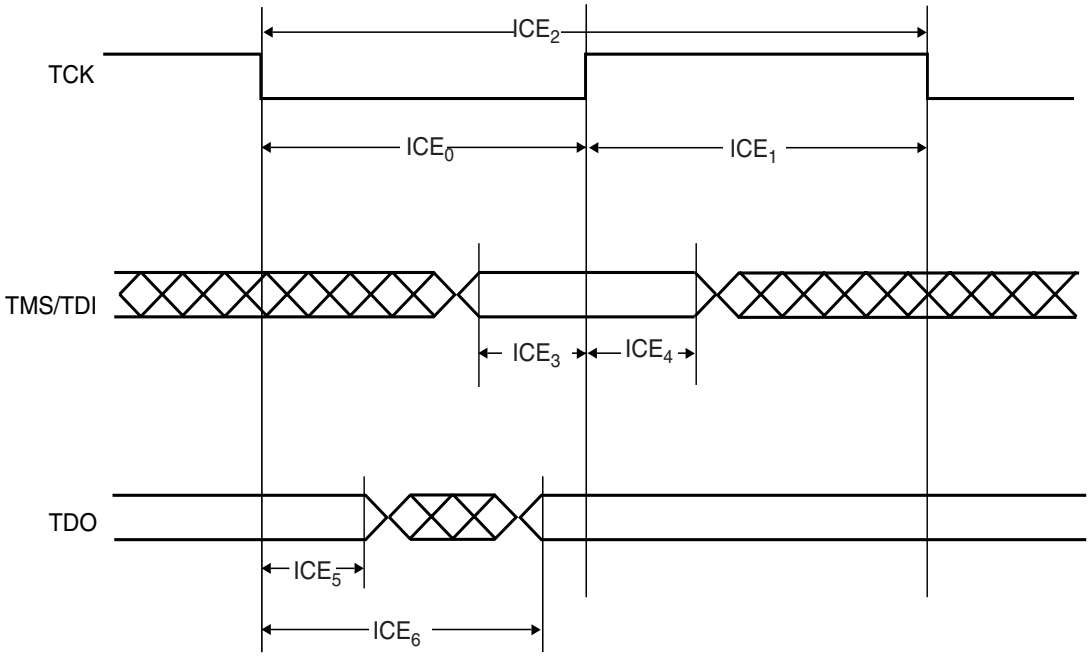
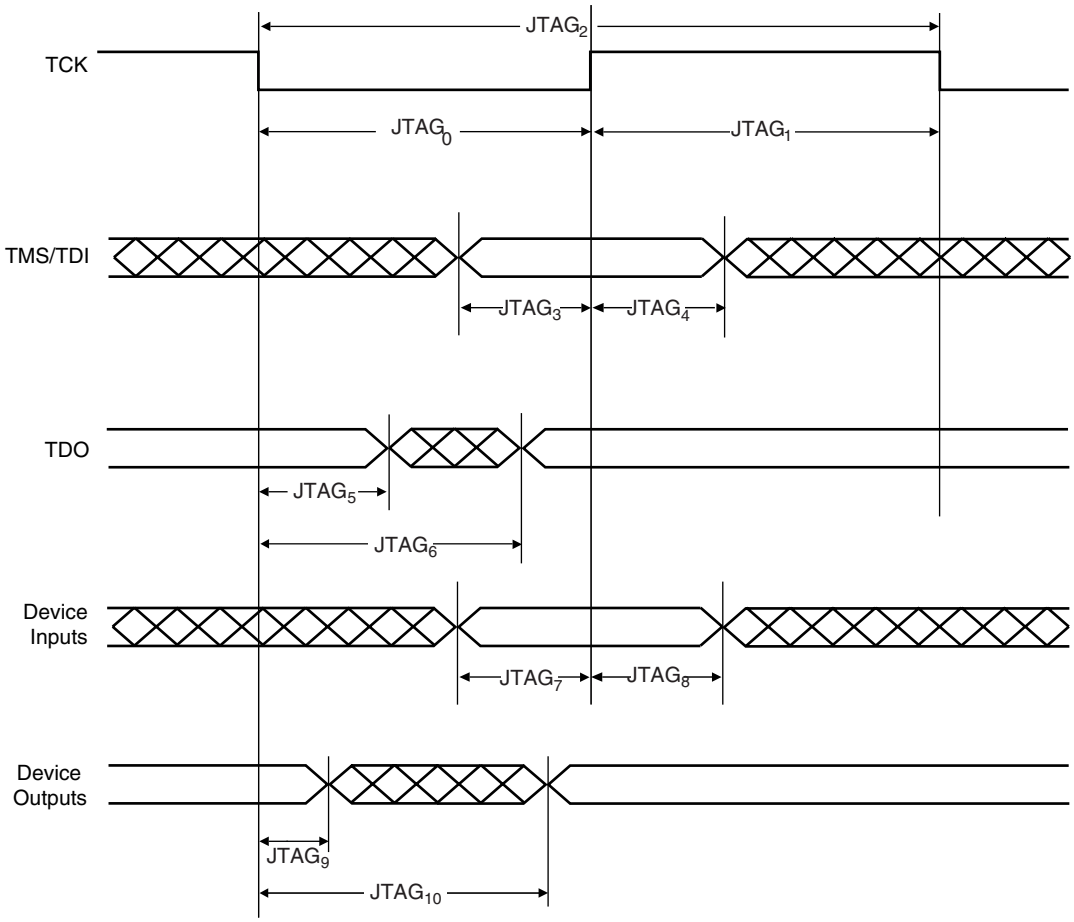


**Table 35-38.** JTAG Interface Timing specification

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $JTAG_0$ | TCK Low Half-period | (1) | 6.5 | | ns |
| $JTAG_1$ | TCK High Half-period | (1) | 5.5 | | ns |
| $JTAG_2$ | TCK Period | (1) | 12 | | ns |
| $JTAG_3$ | TDI, TMS Setup before TCK High | (1) | 2 | | ns |
| $JTAG_4$ | TDI, TMS Hold after TCK High | (1) | 3 | | ns |
| $JTAG_5$ | TDO Hold Time | (1) | 4 | | ns |
| $JTAG_6$ | TCK Low to TDO Valid | (1) | | 16 | ns |
| $JTAG_7$ | Device Inputs Setup Time | (1) | 0 | | ns |
| $JTAG_8$ | Device Inputs Hold Time | (1) | 3 | | ns |
| $JTAG_9$ | Device Outputs Hold Time | (1) | 6 | | ns |
| $JTAG_{10}$ | TCK to Device Outputs Valid | (1) | | 18 | ns |

Note: 1. $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF.

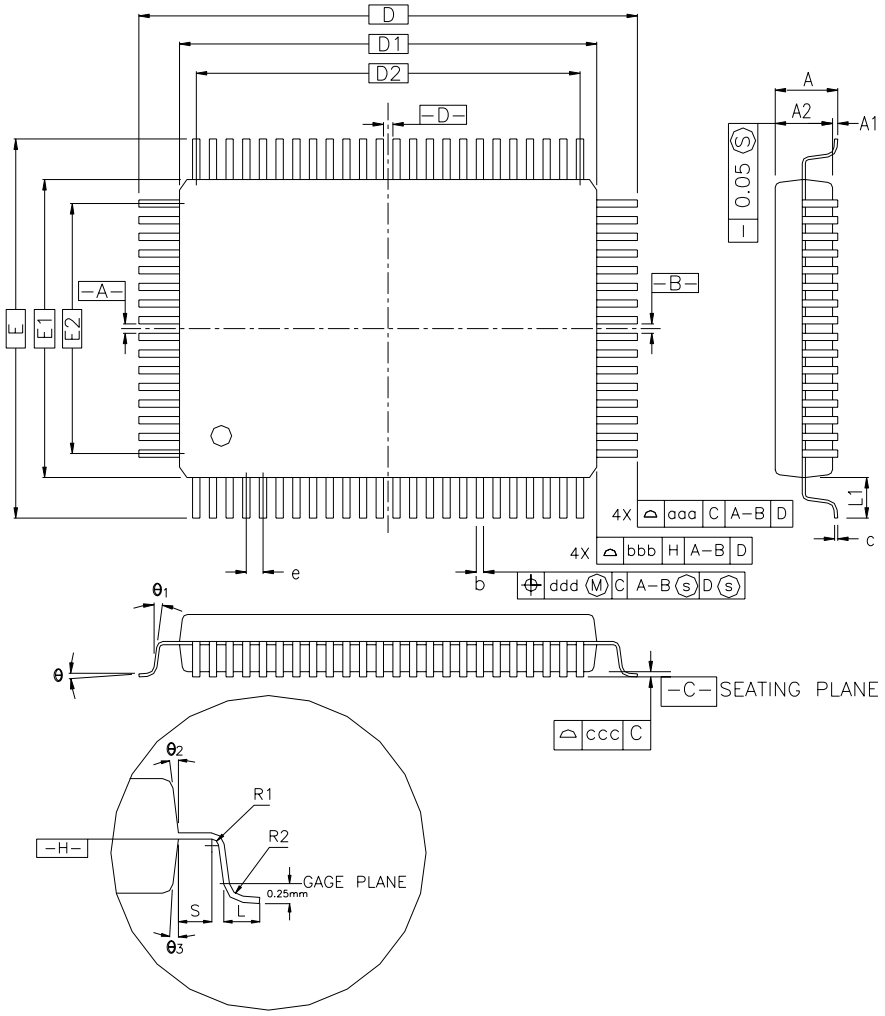
**Figure 35-18.** JTAG Interface Signals



# 36. AT91SAM7L128/64 Mechanical Characteristics

## 36.1 Package Drawings

**Figure 36-1.** LQFP128 Package Drawing




COTROL DIMENSIONS ARE IN MILLIMETERS.

| SYMBOL | MILLIMETER | | | INCH | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | — | — | 1.60 | — | — | 0.063 |
| A1 | 0.05 | — | 0.15 | 0.002 | — | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| D | 22.00 BSC. | | | 0.866 BSC. | | |
| D1 | 20.00 BSC. | | | 0.787 BSC. | | |
| E | 16.00 BSC. | | | 0.630 BSC. | | |
| E1 | 14.00 BSC. | | | 0.551 BSC. | | |
| R2 | 0.08 | — | 0.20 | 0.003 | — | 0.008 |
| R1 | 0.08 | — | — | 0.003 | — | — |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| θ1 | 0° | — | — | 0° | — | — |
| θ2 | 11° | 12° | 13° | 11° | 12° | 13° |
| θ3 | 11° | 12° | 13° | 11° | 12° | 13° |
| c | 0.09 | — | 0.20 | 0.004 | — | 0.008 |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | 1.00 REF | | | 0.039 REF | | |
| S | 0.20 | — | — | 0.008 | — | — |
| b | 0.17 | 0.20 | 0.27 | 0.007 | 0.008 | 0.011 |
| e | 0.50 BSC. | | | 0.020 BSC. | | |
| D2 | 18.50 | | | 0.728 | | |
| E2 | 12.50 | | | 0.492 | | |
| TOLERANCES OF FORM AND POSITION | | | | | | |
| aaa | 0.20 | | | 0.008 | | |
| bbb | 0.20 | | | 0.008 | | |
| ccc | 0.08 | | | 0.003 | | |
| ddd | 0.08 | | | 0.003 | | |

**Table 36-1.** Device and LQFP Package Maximum Weight

| | | |
|---|---|---|
| AT91SAM7L128/64 | 800 | mg |

**Table 36-2.** Package Reference

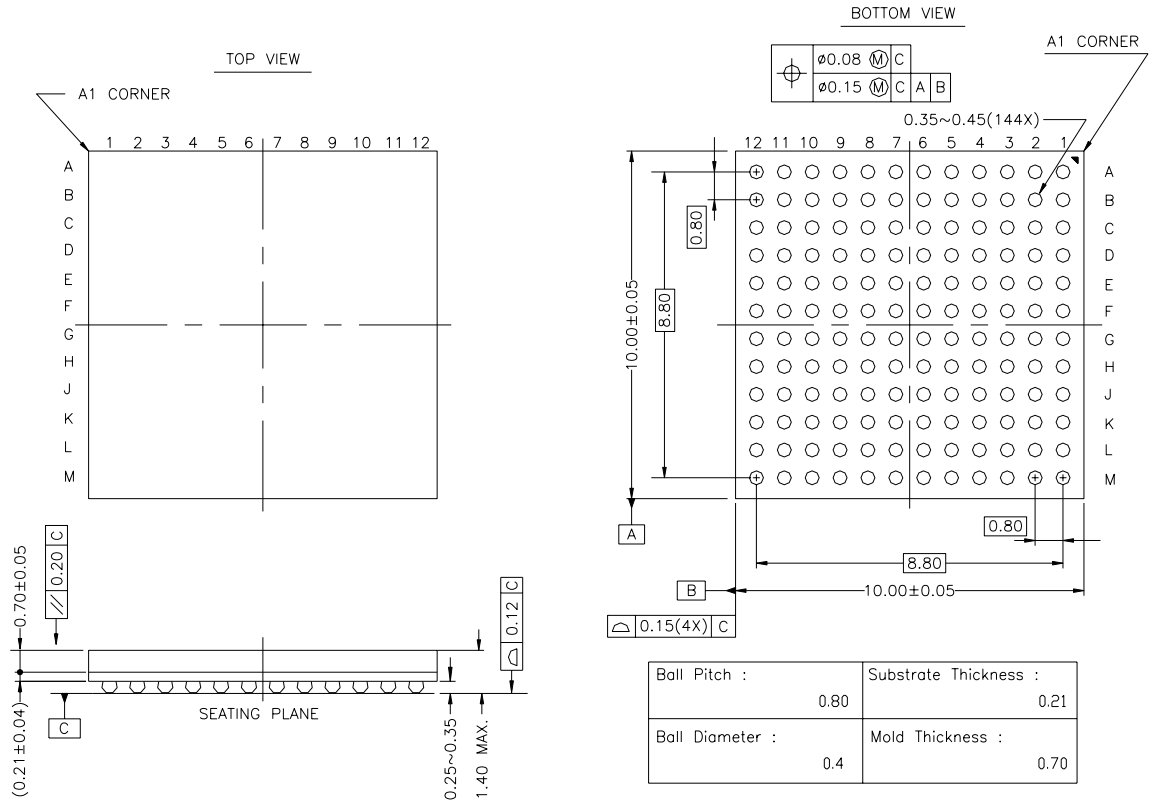
| | |
|---|---|
| JEDEC Drawing Reference | MS-026 |
| JESD97 Classification | e2 |

**Table 36-3.** LQFP Package Characteristics

| | |
|---|---|
| Moisture Sensitivity Level | 3 |

This package respects the recommendations of the NEMI User Group.

**Figure 36-2.** 144-ball LFBGA Package Drawing




All dimensions are in mm

**Table 36-4.** Device and LFBGA Package Maximum Weight

| | | |
|---|---|---|
| AT91SAM7L128/64 | | mg |

**Table 36-5.** Package Reference

| | |
|---|---|
| JEDEC Drawing Reference | MS-026 |
| JESD97 Classification | e1 |

**Table 36-6.** LFBGA Package Characteristics

| | |
|---|---|
| Moisture Sensitivity Level | 3 |

This package respects the recommendations of the NEMI User Group.

## 36.2 Soldering Profile

Table 36-7 gives the recommended soldering profile from J-STD-020C.

**Table 36-7.** Soldering Profile

| Profile Feature | Green Package |
|---|---|
| Average Ramp-up Rate (217°C to Peak) | 3°C/sec. max. |
| Preheat Temperature 175°C ±25°C | 180 sec. max. |
| Temperature Maintained Above 217°C | 60 sec. to 150 sec. |
| Time within 5°C of Actual Peak Temperature | 20 sec. to 40 sec. |
| Peak Temperature Range | 260°C |
| Ramp-down Rate | 6°C/sec. max. |
| Time 25°C to Peak Temperature | 8 min. max. |

Note: The package is certified to be backward compatible with Pb/Sn soldering profile.

A maximum of three reflow passes is allowed per component.

# 37. AT91SAM7L128/64 Ordering Information

**Table 37-1.** Ordering Information

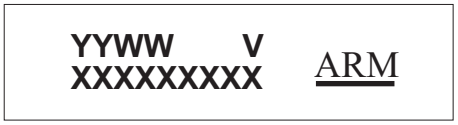
| Ordering Code | Package | Package Type | Temperature Operating Range |
|---|---|---|---|
| AT91SAM7L128-AU | LQFP128 | Green | Industrial (-40°C to 85°C) |
| AT91SAM7L64-AU | LQFP128 | Green | Industrial (-40°C to 85°C) |
| AT91SAM7L128-CU | LFBGA144 | Green | Industrial (-40°C to 85°C) |
| AT91SAM7L64-CU | LFBGA144 | Green | Industrial (-40°C to 85°C) |

# 38. AT91SAM7L128/64 Errata

## 38.1 Marking

All devices are marked with the Atmel logo and the ordering code.

Additional marking has the following format:




where

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision

“XXXXXXXXX”: lot number

## 38.2 AT91SAM7L128/64

Refer to Section 38.1 “Marking” on page 551.

### 38.2.1 Analog-to-Digital Converter (ADC)

#### *38.2.1.1 ADC: Sleep Mode*

If Sleep mode is activated while there is no activity (no conversion is being performed), it will take effect only after a conversion occurs.

**Problem Fix/Workaround**

To activate sleep mode as soon as possible, it is recommended to write successively, ADC Mode Register (SLEEP) then ADC Control Register (START bit field), in order to start an analog-todigital conversion and then put ADC into sleep mode at the end of this conversion.

### 38.2.2 Pulse Width Modulation Controller (PWM)

#### *38.2.2.1 PWM: Counter Start Value*

In left aligned mode, the first start value of the counter is 0. For the other periods, the counter starts at 1.

**Problem Fix/Workaround**

None.

### 38.2.3 Serial Peripheral Interface (SPI)

#### *38.2.3.1 SPI: Baudrate Set to 1*

When the Baudrate is set at 1 (so, the serial clock frequency equals the master clock), and when the BITS field (number of bits to be transmitted) in SPI_CSRx equals an odd value (in this case 9, 11, 13 or 15), an additional pulse will be generated on SPCK.

It does not occur when the BITS field is equal to 8, 10, 12, 14 or 16 and the Baudrate is equal to 1.

**Problem Fix/Workaround**

None.

#### *38.2.3.2 SPI: Bad Serial Clock Generation on 2nd Chip Select*

Bad Serial clock generation on the 2nd chip select when SCBR = 1, CPOL = 1 and NCPHA = 0.

This occurs using SPI with the following conditions:

- Master Mode
- CPOL = 1 and NCPHA = 0
- Multiple chip selects are used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency) and the other transfers set with SCBR are not equal to 1
- Transmitting with the slowest chip select and then with the fastest one, then an additional pulse is generated on output SPCK during the second transfer.

**Problem Fix/Workaround**

Do not use a multiple Chip Select configuration where at least one SCRx register is configured with SCBR = 1 and the others differ from 1 if NCPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

### 38.2.4 Two Wire Interface (TWI)

#### *38.2.4.1 TWI: Switching from Slave to Master Mode*

When the TWI is set in slave mode and if a master write access is performed, the start event is correctly generated but the SCL line is stuck at 1, so no transfer is possible.

**Problem Fix/Workaround**

Two software workarounds are possible:

1. Perform a software reset before going to master mode (TWI must be reconfigured).
2. Perform a slave read access before switching to master mode.

#### *38.2.4.2 TWI: Switching from Slave to Master Mode*

The RXRDY Flag is not reset when a Software reset is performed.

**Problem Fix/Workaround**

After a Software Reset, the Register TWI_RHR must be read.

### 38.2.5 Universal Synchronous Asynchronous Receiver Transmitter (USART)

#### *38.2.5.1 USART: DCD is Active High Instead of Low*

DCD signal is active at “High” level in USART block (Modem Mode).

DCD should be active at “Low” level.

**Problem Fix/Workaround**

Add an inverter.

## Revision History

| Doc. Rev | Comments | Change Request Ref. |
|---|---|---|
| 6257A | First issue | |

## Table of Contents

*Features* ..... *1*

*1 Description* ..... *3*

*2 Block Diagram* ..... *4*

*3 Signal Description* ..... *5*

*4 Package and Pinout* ..... *8*

4.1 128-lead LQFP Package Outline ..... 8

4.2 128-lead LQFP Package Pinout ..... 9

4.3 144-ball LFBGA Package Outline ..... 10

4.4 144-ball LFBGA Pinout ..... 11

*5 Power Considerations* ..... *12*

5.1 Power Supplies ..... 12

5.2 Low Power Modes ..... 12

5.3 Wake-up Sources ..... 14

5.4 Fast Start-Up ..... 14

5.5 Voltage Regulator ..... 15

5.6 LCD Power Supply ..... 16

5.7 Typical Powering Schematics ..... 18

*6 I/O Line Considerations* ..... *19*

6.1 JTAG Port Pins ..... 19

6.2 Test Pin ..... 19

6.3 NRST Pin ..... 19

6.4 NRSTB Pin ..... 19

6.5 ERASE Pin ..... 19

6.6 PIO Controller Lines ..... 20

6.7 I/O Line Current Drawing ..... 20

*7 Processor and Architecture* ..... *21*

7.1 ARM7TDMI Processor ..... 21

7.2 Debug and Test Features ..... 21

7.3 Memory Controller ..... 21

7.4 Peripheral DMA Controller ..... 22

*8 Memories* ..... *23*

8.1 Embedded Memories ..... 25

*9 System Controller ..... 29*

9.1 System Controller Mapping ..... 29
9.2 Supply Controller (SUPC) ..... 31
9.3 Reset Controller ..... 31
9.4 Clock Generator ..... 32
9.5 Power Management Controller ..... 33
9.6 Advanced Interrupt Controller ..... 34
9.7 Debug Unit ..... 34
9.8 Period Interval Timer ..... 35
9.9 Watchdog Timer ..... 35
9.10 Real-time Clock ..... 35
9.11 PIO Controllers ..... 35

*10 Peripherals ..... 36*

10.1 User Interface ..... 36
10.2 Peripheral Identifiers ..... 36
10.3 Peripheral Multiplexing on PIO Lines ..... 37
10.4 PIO Controller A Multiplexing ..... 38
10.5 PIO Controller B Multiplexing ..... 39
10.6 PIO Controller C Multiplexing ..... 40
10.7 Serial Peripheral Interface ..... 41
10.8 Two Wire Interface ..... 41
10.9 USART ..... 41
10.10 Timer Counter ..... 42
10.11 PWM Controller ..... 42
10.12 Analog-to-Digital Converter ..... 43
10.13 Segment LCD Controller ..... 43

*11 ARM7TDMI Processor Overview ..... 45*

11.1 Overview ..... 45
11.2 ARM7TDMI Processor ..... 46

*12 Debug and Test Features ..... 51*

12.1 Overview ..... 51
12.2 Block Diagram ..... 51
12.3 Application Examples ..... 52
12.4 Debug and Test Pin Description ..... 53
12.5 Functional Description ..... 54

**13 Reset Controller (RSTC) ..... 57**

13.1 Overview ..... 57
13.2 Block Diagram ..... 57
13.3 Functional Description ..... 57
13.4 Reset Controller (RSTC) User Interface ..... 65

**14 Real-time Clock (RTC) ..... 69**

14.1 Overview ..... 69
14.2 Block Diagram ..... 69
14.3 Product Dependencies ..... 69
14.4 Functional Description ..... 70
14.5 Real-time Clock (RTC) User Interface ..... 73
14.6 RTC Valid Entry Register ..... 85

**15 Periodic Interval Timer (PIT) ..... 87**

15.1 Overview ..... 87
15.2 Block Diagram ..... 87
15.3 Functional Description ..... 88
15.4 Periodic Interval Timer (PIT) User Interface ..... 90

**16 Watchdog Timer (WDT) ..... 95**

16.1 Overview ..... 95
16.2 Block Diagram ..... 95
16.3 Functional Description ..... 96
16.4 Watchdog Timer (WDT) User Interface ..... 98

**17 Supply Controller (SUPC) ..... 103**

17.1 Overview ..... 103
17.2 Block Diagram ..... 104
17.3 Supply Controller Functional Description ..... 105
17.4 Supply Controller (SUPC) User Interface ..... 118

**18 Memory Controller (MC) ..... 131**

18.1 Overview ..... 131
18.2 Block Diagram ..... 131
18.3 Functional Description ..... 132
18.4 Memory Controller (MC) User Interface ..... 136

**19 Enhanced Embedded Flash Controller (EEFC) ..... 141**

19.1 Overview ..... 141

19.2 Product Dependencies ..... 141
19.3 Functional Description ..... 141
19.4 Enhanced Embedded Flash Controller (EEFC) User Interface ..... 152

***20 Fast Flash Programming Interface (FFPI) ..... 157***

20.1 Overview ..... 157
20.2 Parallel Fast Flash Programming ..... 157
20.3 Serial Fast Flash Programming ..... 166

***21 AT91SAM Boot Program ..... 173***

21.1 Overview ..... 173
21.2 Flow Diagram ..... 173
21.3 Device Initialization ..... 173
21.4 SAM-BA Boot ..... 174
21.5 In-Application Programming (IAP) Feature ..... 177
21.6 Hardware and Software Constraints ..... 178

***22 Peripheral DMA Controller (PDC) ..... 179***

22.1 Overview ..... 179
22.2 Block Diagram ..... 179
22.3 Functional Description ..... 180
22.4 Peripheral DMA Controller (PDC) User Interface ..... 182

***23 Advanced Interrupt Controller (AIC) ..... 189***

23.1 Overview ..... 189
23.2 Block Diagram ..... 190
23.3 Application Block Diagram ..... 190
23.4 AIC Detailed Block Diagram ..... 190
23.5 I/O Line Description ..... 191
23.6 Product Dependencies ..... 191
23.7 Functional Description ..... 192
23.8 Advanced Interrupt Controller (AIC) User Interface ..... 202

***24 Clock Generator ..... 213***

24.1 Overview ..... 213
24.2 Slow Clock ..... 214
24.3 Slow Clock RC Oscillator ..... 214
24.4 Slow Clock Crystal Oscillator ..... 214
24.5 Main Clock ..... 215

24.6 Divider and PLL Block ..... 216

***25 Power Management Controller (PMC) ..... 218***

25.1 Overview ..... 218
25.2 Master Clock Controller ..... 218
25.3 Processor Clock Controller ..... 219
25.4 Peripheral Clock Controller ..... 219
25.5 Programmable Clock Output Controller ..... 219
25.6 The Fast Startup ..... 220
25.7 Programming Sequence ..... 220
25.8 Clock Switching Details ..... 224
25.9 Power Management Controller (PMC) User Interface ..... 227
25.10 PMC Fast Startup Mode Register ..... 242

***26 Debug Unit (DBGU) ..... 243***

26.1 Overview ..... 243
26.2 Block Diagram ..... 244
26.3 Product Dependencies ..... 245
26.4 UART Operations ..... 245
26.5 Debug Unit (DBGU) User Interface ..... 252

***27 Parallel Input Output Controller (PIO) ..... 267***

27.1 Overview ..... 267
27.2 Block Diagram ..... 268
27.3 Product Dependencies ..... 269
27.4 Functional Description ..... 270
27.5 I/O Lines Programming Example ..... 274
27.6 Parallel Input/Output Controller (PIO) User Interface ..... 276

***28 Serial Peripheral Interface (SPI) ..... 293***

28.1 Overview ..... 293
28.2 Block Diagram ..... 294
28.3 Application Block Diagram ..... 294
28.4 Signal Description ..... 295
28.5 Product Dependencies ..... 295
28.6 Functional Description ..... 296
28.7 Serial Peripheral Interface (SPI) User Interface ..... 305

***29 Two Wire Interface (TWI) ..... 319***

29.1 Overview ..... 319
29.2 List of Abbreviations ..... 319
29.3 Block Diagram ..... 320
29.4 Application Block Diagram ..... 320
29.5 Product Dependencies ..... 321
29.6 Functional Description ..... 321
29.7 Master Mode ..... 323
29.8 Multi-master Mode ..... 335
29.9 Slave Mode ..... 338
29.10 Two-wire Interface (TWI) User Interface ..... 346

***30 Universal Synchronous Asynchronous Receiver Transceiver (USART) ..... 361***

30.1 Overview ..... 361
30.2 Block Diagram ..... 362
30.3 Application Block Diagram ..... 363
30.4 I/O Lines Description ..... 364
30.5 Product Dependencies ..... 365
30.6 Functional Description ..... 366
30.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface ..... 397

***31 Timer Counter (TC) ..... 419***

31.1 Overview ..... 419
31.2 Block Diagram ..... 420
31.3 Pin Name List ..... 421
31.4 Product Dependencies ..... 421
31.5 Functional Description ..... 422
31.6 Timer Counter (TC) User Interface ..... 435

***32 Pulse Width Modulation Controller (PWM) ..... 453***

32.1 Overview ..... 453
32.2 Block Diagram ..... 453
32.3 I/O Lines Description ..... 454
32.4 Product Dependencies ..... 454
32.5 Functional Description ..... 454
32.6 Pulse Width Modulation Controller (PWM) User Interface ..... 463

***33 Analog-to-Digital Converter (ADC) ..... 473***

33.1 Overview ..... 473
33.2 Block Diagram ..... 473
33.3 Signal Description ..... 474
33.4 Product Dependencies ..... 474
33.5 Functional Description ..... 475
33.6 Analog-to-Digital Converter (ADC) User Interface ..... 480

***34 Segment LCD Controller (SLCDC) ..... 491***

34.1 Overview ..... 491
34.2 Block Diagram ..... 492
34.3 I/O Lines Description ..... 493
34.4 Product Dependencies ..... 493
34.5 Functional Description ..... 494
34.6 Waveform Specifications ..... 505
34.7 Segment LCD Controller (SLCDC) User Interface ..... 506

***35 AT91SAM7L128/64 Electrical Characteristics ..... 517***

35.1 Absolute Maximum Ratings ..... 517
35.2 DC Characteristics ..... 518
35.3 Power Consumption ..... 521
35.4 Crystal Oscillators Characteristics ..... 533
35.5 PLL Characteristics ..... 537
35.6 ADC Characteristics ..... 537
35.7 Regulated Charge Pump Characteristics ..... 538
35.8 LCD Voltage Characteristic ..... 539
35.9 LCD Driver Characteristic ..... 539
35.10 AC Characteristics ..... 540

***36 AT91SAM7L128/64 Mechanical Characteristics ..... 547***

36.1 Package Drawings ..... 547
36.2 Soldering Profile ..... 549

***37 AT91SAM7L128/64 Ordering Information ..... 550***

***38 AT91SAM7L128/64 Errata ..... 551***

38.1 Marking ..... 551
38.2 AT91SAM7L128/64 ..... 552

***Revision History ..... 555***

***Table of Contents ..... i***

## Headquarters


***Atmel Corporation***
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600


## International


***Atmel Asia***
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

***Atmel Europe***
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

***Atmel Japan***
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581


## Product Contact

***Web Site***
www.atmel.com
www.atmel.com/AT91SAM

***Technical Support***
AT91SAM Support
Atmel techincal support

***Sales Contacts***
www.atmel.com/contacts/

***Literature Requests***
www.atmel.com/literature


**Disclaimer:** The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.



**© 2008 Atmel Corporation. All rights reserved.** Atmel®, logo and combinations thereof, DataFlash®, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM® the ARM® Powered logo and others are registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.