32-BIT ARM-BASED MICROPROCESSORS

SAM9G20

Description

The SAM9G20 embedded microprocessor unit is based on the integration of an Arm926EJ-S™ processor with fast ROM and RAM memories and a wide range of peripherals.

The SAM9G20 embeds an Ethernet MAC, one USB Device Port, and a dual port USB Host controller with on-chip USB transceivers. It also integrates several standard peripherals, such as the USART, SPI, TWI, Timer Counters, Synchronous Serial Controller, ADC and MultiMedia Card Interface.

The SAM9G20 is architectured on a 6-layer matrix, allowing a maximum internal bandwidth of six 32-bit buses. It also features an External Bus Interface capable of interfacing with a wide range of memory devices.

The SAM9G20 is an enhancement of the SAM9260 with the same peripheral features. It is pin-to-pin compatible with the exception of power supply pins. Speed is increased to reach 400 MHz on the Arm core and 133 MHz on the system bus and EBI.

Features

- Incorporates the Arm926EJ-S Arm® Thumb® Processor
  - DSP Instruction Extensions, Arm Jazelle® Technology for Java® Acceleration
  - 32-Kbyte Data Cache, 32-Kbyte Instruction Cache, Write Buffer
  - CPU Frequency 400 MHz
  - Memory Management Unit
  - EmbeddedICE, Debug Communication Channel Support
- Additional Embedded Memories
  - One 64-Kbyte Internal ROM, Single-cycle Access at Maximum Matrix Speed
  - Two 16-Kbyte Internal SRAM, Single-cycle Access at Maximum Matrix Speed
- External Bus Interface (EBI)
  - Supports SDRAM, Static Memory, ECC-enabled SLC NAND Flash and CompactFlash®
- USB 2.0 Full Speed (12 Mbit/s) Device Port
  - On-chip Transceiver, 2,432-byte Configurable Integrated DPRAM
- USB 2.0 Full Speed (12 Mbit/s) Host and Dual Port
  - Single or Dual On-chip Transceivers
  - Integrated FIFOs and Dedicated DMA Channels
- Ethernet MAC 10/100 Base T
  - Media Independent Interface or Reduced Media Independent Interface
  - 128-byte FIFOs and Dedicated DMA Channels for Receive and Transmit
- Image Sensor Interface
  - ITU-R BT. 601/656 External Interface, Programmable Frame Capture Rate
  - 12-bit Data Interface for Support of High Sensibility Sensors
  - SAV and EAV Synchronization, Preview Path with Scaler, YCbCr Format
- Bus Matrix
  - Six 32-bit-layer Matrix
  - Boot Mode Select Option, Remap Command
- Fully-featured System Controller, including
  - Reset Controller, Shutdown Controller
  - 128-bit (4 x 32-bit) General Purpose Backup Registers
  - Clock Generator and Power Management Controller
  - Advanced Interrupt Controller and Debug Unit

© 2017 Microchip Technology Inc.

  - Periodic Interval Timer, Watchdog Timer and Real-time Timer
- Reset Controller (RSTC)
  - Based on a Power-on Reset Cell, Reset Source Identification and Reset Output Control
- Clock Generator (CKGR)
  - Selectable 32768 Hz Low-power Oscillator or Internal Low Power RC Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - 3 to 20 MHz On-chip Oscillator, One up to 800 MHz PLL and One up to 100 MHz PLL
- Power Management Controller (PMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Two Programmable External Clock Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Three External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - 2-wire UART and Support for Debug Communication Channel, Programmable ICE Access Prevention
  - Mode for General Purpose 2-wire UART Serial Communication
- Periodic Interval Timer (PIT)
  - 20-bit Interval Timer plus 12-bit Interval Counter
- Watchdog Timer (WDT)
  - Key-protected, Programmable Only Once, Windowed 16-bit Counter Running at Slow Clock
- Real-time Timer (RTT)
  - 32-bit Free-running Backup Counter Running at Slow Clock with 16-bit Prescaler
- One 4-channel 10-bit Analog-to-Digital Converter
- Three 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC)
  - 96 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
  - All I/O Lines are Schmitt Trigger Inputs
- Peripheral DMA Controller Channels (PDC)
- One Two-slot MultiMedia Card Interface (MCI)
  - SDCard/SDIO and MultiMediaCard™ Compliant
  - Automatic Protocol Control and Fast Automatic Data Transfers with PDC
- One Synchronous Serial Controller (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I²S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- Four Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation, Manchester Encoding/Decoding
  - Support for ISO7816 T0/T1 Smart Card, Hardware Handshaking, RS485 Support
  - Full Modem Signal Control on USART0
- Two 2-wire UARTs
- Two Master/Slave Serial Peripheral Interfaces (SPI)
  - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
  - Synchronous Communications
- Two Three-channel 16-bit Timer/Counters (TC)
  - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
  - High-Drive Capability on Outputs TIOA0, TIOA1, TIOA2
- One Two-wire Interface (TWI)
  - Compatible with Standard Two-wire Serial Memories
  - One, Two or Three Bytes for Slave Address
  - Sequential Read/Write Operations
  - Master, Multi-master and Slave Mode Operation
  - Bit Rate: Up to 400 Kbit/s
  - General Call Supported in Slave Mode

© 2017 Microchip Technology Inc.

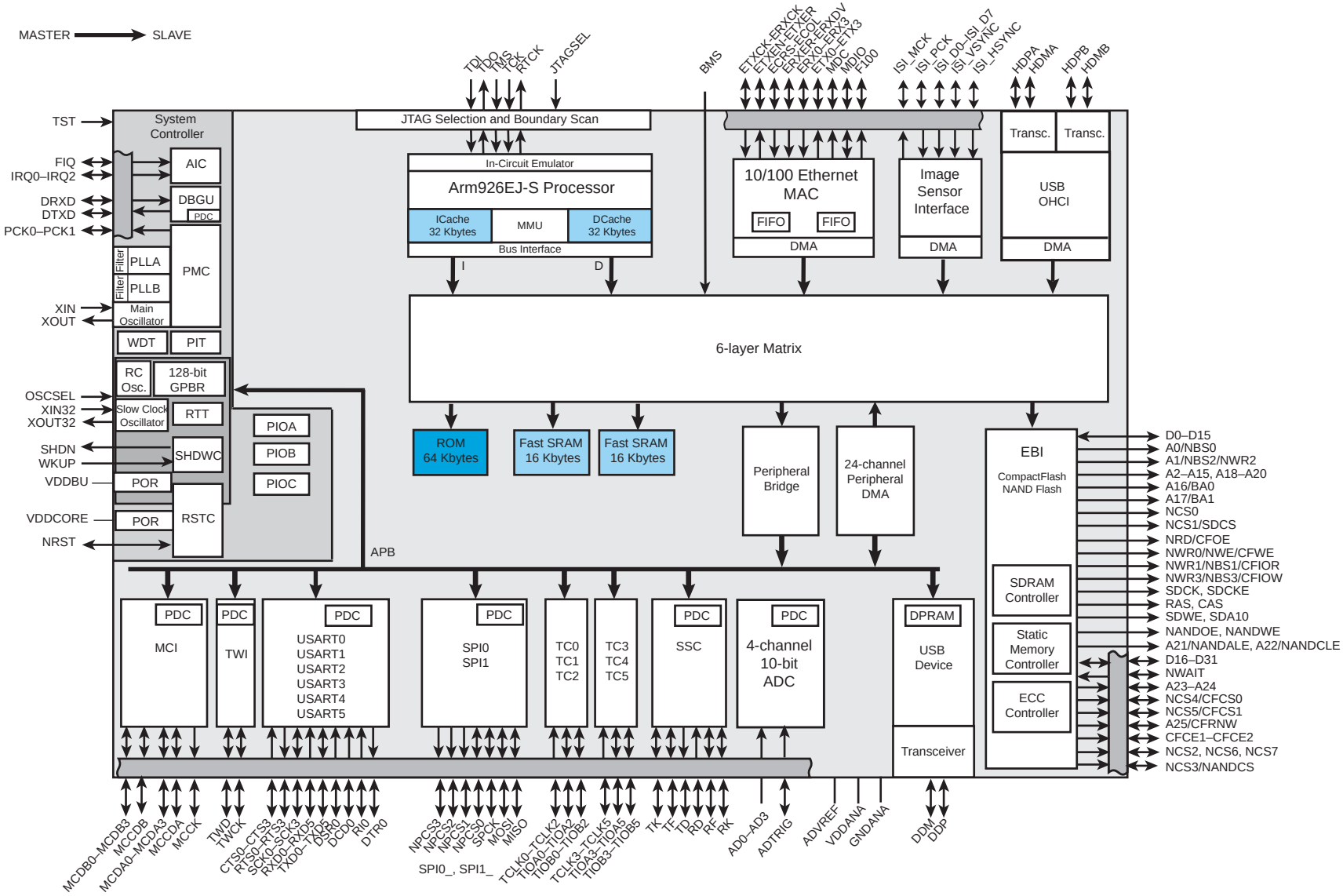
  - Connection to Peripheral DMA Controller (PDC) Channel Capabilities Optimizes Data Transfers in Master Mode
- IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins
- Required Power Supplies
  - 0.9V to 1.1V for VDDBU, VDDCORE, VDDPLL
  - 1.65 to 3.6V for VDDOSC
  - 1.65V to 3.6V for VDDIOP (Peripheral I/Os)
  - 3.0V to 3.6V for VDDUSB
  - 3.0V to 3.6V VDDANA (Analog-to-Digital Converter)
  - Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOM (Memory I/Os)
- Available in 217-ball LFBGA and 247-ball TFBGA Green-compliant Packages

© 2017 Microchip Technology Inc.

© 2017 Microchip Technology Inc.

1. Block Diagram

Figure 1-1: SAM9G20 Block Diagram



2. Signal Description

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power Supplies				
VDDIOM	EBI I/O Lines Power Supply	Power	–	1.65–1.95 V or 3.0–3.6 V
VDDIOP	Peripherals I/O Lines Power Supply	Power	–	1.65–3.6 V
VDDBU	Backup I/O Lines Power Supply	Power	–	0.9–1.1 V
VDDANA	Analog Power Supply	Power	–	3.0–3.6 V
VDDPLL	PLL Power Supply	Power	–	0.9–1.1 V
VDDOSC	Oscillator Power Supply	Power	–	1.65–3.6 V
VDDCORE	Core Chip Power Supply	Power	–	0.9–1.1 V
VDDUSB	USB Power Supply	Power	–	1.65–3.6 V
GND	Ground	Ground	–	
GNDANA	Analog Ground	Ground	–	
GNDBU	Backup Ground	Ground	–	
GNDUSB	USB Ground	Ground	–	
GNDPLL	PLL Ground	Ground	–	
Clocks, Oscillators and PLLs				
XIN	Main Oscillator Input	Input	–	
XOUT	Main Oscillator Output	Output	–	
XIN32	Slow Clock Oscillator Input	Input	–	
XOUT32	Slow Clock Oscillator Output	Output	–	
OSCSEL	Slow Clock Oscillator Selection	Input	–	Accepts between 0V and VDDBU
PCK0–PCK1	Programmable Clock Output	Output	–	
Shutdown, Wakeup Logic				
SHDN	Shutdown Control	Output	–	
WKUP	Wake-up Input	Input	–	Accepts between 0V and VDDBU
ICE and JTAG				
NTRST	Test Reset Signal	Input	Low	Pull-up resistor
TCK	Test Clock	Input	–	No pull-up resistor
TDI	Test Data In	Input	–	No pull-up resistor
TDO	Test Data Out	Output	–	
TMS	Test Mode Select	Input	–	No pull-up resistor
JTAGSEL	JTAG Selection	Input	–	Pull-down resistor. Accepts between 0V and VDDBU.
RTCK	Return Test Clock	Output	–	

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level	Comments
Reset/Test				
NRST	Microprocessor Reset	I/O	Low	Pull-up resistor
TST	Test Mode Select	Input	–	Pull-down resistor. Accepts between 0V and VDDBU.
BMS	Boot Mode Select	Input	–	No pull-up resistor BMS = 0 when tied to GND BMS = 1 when tied to VDDIOP
Debug Unit - DBGU				
DRXD	Debug Receive Data	Input	–	
DTXD	Debug Transmit Data	Output	–	
Advanced Interrupt Controller - AIC				
IRQ0–IRQ2	External Interrupt Inputs	Input	–	
FIQ	Fast Interrupt Input	Input	–	
PIO Controller - PIOA / PIOB / PIOC				
PA0–PA31	Parallel IO Controller A	I/O	–	Pulled-up input at reset
PB0–PB31	Parallel IO Controller B	I/O	–	Pulled-up input at reset
PC0–PC31	Parallel IO Controller C	I/O	–	Pulled-up input at reset
External Bus Interface - EBI				
D0–D31	Data Bus	I/O	–	Pulled-up input at reset
A0–A25	Address Bus	Output	–	0 at reset
NWAIT	External Wait Signal	Input	Low	
Static Memory Controller - SMC				
NCS0–NCS7	Chip Select Lines	Output	Low	
NWR0–NWR3	Write Signal	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NBS0–NBS3	Byte Mask Signal	Output	Low	
CompactFlash Support				
CFCE1–CFCE2	CompactFlash Chip Enable	Output	Low	
CFOE	CompactFlash Output Enable	Output	Low	
CFWE	CompactFlash Write Enable	Output	Low	
CFIOR	CompactFlash IO Read	Output	Low	
CFIOW	CompactFlash IO Write	Output	Low	
CFRNW	CompactFlash Read Not Write	Output	–	
CFCS0–CFCS1	CompactFlash Chip Select Lines	Output	Low	

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level	Comments
NAND Flash Support				
NANDCS	NAND Flash Chip Select	Output	Low	
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NANDALE	NAND Flash Address Latch Enable	Output	Low	
NANDCLE	NAND Flash Command Latch Enable	Output	Low	
SDRAM Controller - SDRAMC				
SDCK	SDRAM Clock	Output	–	
SDCKE	SDRAM Clock Enable	Output	High	
SDCS	SDRAM Controller Chip Select	Output	Low	
BA0–BA1	Bank Select	Output	–	
SDWE	SDRAM Write Enable	Output	Low	
RAS - CAS	Row and Column Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output	–	
Multimedia Card Interface - MCI				
MCCK	Multimedia Card Clock	Output	–	
MCCDA	Multimedia Card Slot A Command	I/O	–	
MCDA0–MCDA3	Multimedia Card Slot A Data	I/O	–	
MCCDB	Multimedia Card Slot B Command	I/O	–	
MCDB0–MCDB3	Multimedia Card Slot B Data	I/O	–	
Universal Synchronous Asynchronous Receiver Transmitter - USARTx				
SCKx	USARTx Serial Clock	I/O	–	
TXDx	USARTx Transmit Data	I/O	–	
RXDx	USARTx Receive Data	Input	–	
RTSx	USARTx Request To Send	Output	–	
CTSx	USARTx Clear To Send	Input	–	
DTR0	USART0 Data Terminal Ready	Output	–	
DSR0	USART0 Data Set Ready	Input	–	
DCD0	USART0 Data Carrier Detect	Input	–	
RI0	USART0 Ring Indicator	Input	–	
Synchronous Serial Controller - SSC				
TD	SSC Transmit Data	Output	–	
RD	SSC Receive Data	Input	–	
TK	SSC Transmit Clock	I/O	–	
RK	SSC Receive Clock	I/O	–	
TF	SSC Transmit Frame Sync	I/O	–	

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level	Comments
RF	SSC Receive Frame Sync	I/O	–	
Timer/Counter - TCx				
TCLKx	TC Channel x External Clock Input	Input	–	
TIOAx	TC Channel x I/O Line A	I/O	–	
TIOBx	TC Channel x I/O Line B	I/O	–	
Serial Peripheral Interface - SPIx				
SPIx_MISO	Master In Slave Out	I/O	–	
SPIx_MOSI	Master Out Slave In	I/O	–	
SPIx_SPCK	SPI Serial Clock	I/O	–	
SPIx_NPCS0	SPI Peripheral Chip Select 0	I/O	Low	
SPIx_NPCS1–SPIx_NPCS3	SPI Peripheral Chip Select	Output	Low	
Two-Wire Interface - TWI				
TWD	Two-wire Serial Data	I/O	–	
TWCK	Two-wire Serial Clock	I/O	–	
USB Host Port - UHP				
HDPA	USB Host Port A Data +	Analog	–	
HDMA	USB Host Port A Data -	Analog	–	
HDPB	USB Host Port B Data +	Analog	–	
HDMB	USB Host Port B Data -	Analog	–	
USB Device Port - UDP				
DDM	USB Device Port Data -	Analog	–	
DDP	USB Device Port Data +	Analog	–	
Ethernet MAC 10/100 - EMAC				
ETXCK	Transmit Clock or Reference Clock	Input	–	MII only, REFCK in RMII
ERXCK	Receive Clock	Input	–	MII only
ETXEN	Transmit Enable	Output	–	
ETX0–ETX3	Transmit Data	Output	–	ETX0–ETX1 only in RMII
ETXER	Transmit Coding Error	Output	–	MII only
ERXDV	Receive Data Valid	Input	–	RXDV in MII, CRSDV in RMII
ERX0–ERX3	Receive Data	Input	–	ERX0–ERX1 only in RMII
ERXER	Receive Error	Input	–	
ECRS	Carrier Sense and Data Valid	Input	–	MII only
ECOL	Collision Detect	Input	–	MII only
EMDC	Management Data Clock	Output	–	
EMDIO	Management Data Input/Output	I/O	–	

© 2017 Microchip Technology Inc.

Table 2-1: Signal Description List

Signal Name	Function	Type	Active Level	Comments
Image Sensor Interface - ISI				
ISI_D0-ISI_D11	Image Sensor Data	Input	–	
ISI_MCK	Image Sensor Reference Clock	Output	–	
ISI_HSYNC	Image Sensor Horizontal Synchro	Input	–	
ISI_VSYNC	Image Sensor Vertical Synchro	Input	–	
ISI_PCK	Image Sensor Data clock	Input	–	
Analog-to-Digital Converter - ADC				
AD0-AD3	Analog Inputs	Analog	–	Digital pulled-up inputs at reset
ADVREF	Analog Positive Reference	Analog	–	
ADTRG	ADC Trigger	Input	–	

3. Package and Pinout

The SAM9G20 is available in the following Green-compliant packages:

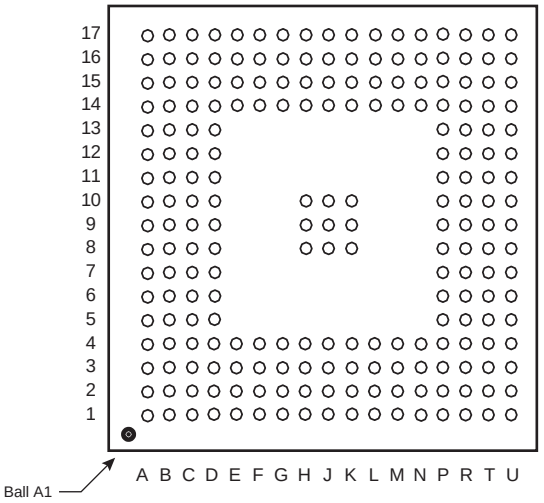
- 217-ball LFBGA, 15 x 15 mm x 1.4 mm (0.8 mm pitch)
- 247-ball TFBGA, 10 x 10 x 1.1 mm (0.5 mm pitch)

3.1 217-ball LFBGA Package Outline

Figure 3-1 shows the orientation of the 217-ball LFBGA package.

A detailed mechanical description is given in Section 41.1 “217-ball LFBGA Package Drawing”.

Figure 3-1: 217-ball LFBGA Package (Top View)



3.2 217-ball LFBGA Pinout

Table 3-1: Pinout for 217-ball LFBGA Package

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	CFIOW/NBS3/NWR3	D5	A5	J14	TDO	P17	PB5
A2	NBS0/A0	D6	GND	J15	PB19	R1	NC
A3	NWR2/NBS2/A1	D7	A10	J16	TDI	R2	GNDANA
A4	A6	D8	GND	J17	PB16	R3	PC29
A5	A8	D9	VDDCORE	K1	PC24	R4	VDDANA
A6	A11	D10	GNDUSB	K2	PC20	R5	PB12
A7	A13	D11	VDDIOM	K3	D15	R6	PB23
A8	BA0/A16	D12	GNDUSB	K4	PC21	R7	GND
A9	A18	D13	DDM	K8	GND	R8	PB26
A10	A21	D14	HDPB	K9	GND	R9	PB28
A11	A22	D15	NC	K10	GND	R10	PA0
A12	CFWE/NWE/NWR0	D16	VDDBU	K14	PB4	R11	PA4
A13	CFOE/NRD	D17	XIN32	K15	PB17	R12	PA5
A14	NCS0	E1	D10	K16	GND	R13	PA10

© 2017 Microchip Technology Inc.

Table 3-1: Pinout for 217-ball LFBGA Package (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A15	PC5	E2	D5	K17	PB15	R14	PA21
A16	PC6	E3	D3	L1	GND	R15	PA23
A17	PC4	E4	D4	L2	PC26	R16	PA24
B1	SDCK	E14	HDPA	L3	PC25	R17	PA29
B2	CFIOR/NBS1/NWR1	E15	HDMA	L4	VDDOSC	T1	NC
B3	SDCS/NCS1	E16	GNDBU	L14	PA28	T2	GNDPLL
B4	SDA10	E17	XOUT32	L15	PB9	T3	PC0
B5	A3	F1	D13	L16	PB8	T4	PC1
B6	A7	F2	SDWE	L17	PB14	T5	PB10
B7	A12	F3	D6	M1	VDDCORE	T6	PB22
B8	A15	F4	GND	M2	PC31	T7	GND
B9	A20	F14	OSCSEL	M3	GND	T8	PB29
B10	NANDWE	F15	BMS	M4	PC22	T9	PA2
B11	PC7	F16	JTAGSEL	M14	PB1	T10	PA6
B12	PC10	F17	TST	M15	PB2	T11	PA8
B13	PC13	G1	PC15	M16	PB3	T12	PA11
B14	PC11	G2	D7	M17	PB7	T13	VDDCORE
B15	PC14	G3	SDCKE	N1	XIN	T14	PA20
B16	PC8	G4	VDDIOM	N2	VDDPLL	T15	GND
B17	WKUP	G14	GND	N3	PC23	T16	PA22
C1	D8	G15	NRST	N4	PC27	T17	PA27
C2	D1	G16	RTCK	N14	PA31	U1	GNDPLL
C3	CAS	G17	TMS	N15	PA30	U2	ADVREF
C4	A2	H1	PC18	N16	PB0	U3	PC2
C5	A4	H2	D14	N17	PB6	U4	PC3
C6	A9	H3	D12	P1	XOUT	U5	PB20
C7	A14	H4	D11	P2	VDDPLL	U6	PB21
C8	BA1/A17	H8	GND	P3	PC30	U7	PB25
C9	A19	H9	GND	P4	PC28	U8	PB27
C10	NANDOE	H10	GND	P5	PB11	U9	PA12
C11	PC9	H14	VDDCORE	P6	PB13	U10	PA13
C12	PC12	H15	TCK	P7	PB24	U11	PA14
C13	DDP	H16	NTRST	P8	VDDIOP	U12	PA15
C14	HDMB	H17	PB18	P9	PB30	U13	PA19
C15	NC	J1	PC19	P10	PB31	U14	PA17
C16	VDDUSB	J2	PC17	P11	PA1	U15	PA16

Table 3-1: Pinout for 217-ball LFBGA Package (Continued)

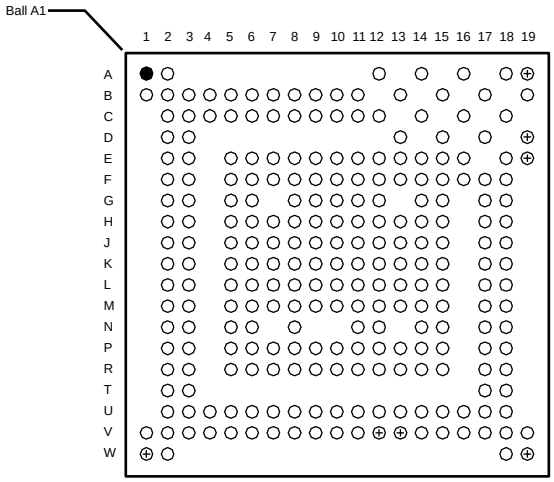
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
C17	SHDN	J3	VDDIOM	P12	PA3	U16	PA18
D1	D9	J4	PC16	P13	PA7	U17	VDDIOP
D2	D2	J8	GND	P14	PA9		
D3	RAS	J9	GND	P15	PA26		
D4	D0	J10	GND	P16	PA25		

3.3 247-ball TFBGA Package Outline

Figure 3-2 shows the orientation of the 247-ball TFBGA package.

A detailed mechanical description is given in Section 41.2 “247-ball TFBGA Package Drawing”.

Figure 3-2: 247-ball TFBGA Package (Top View)



3.4 247-ball TFBGA Package Pinout

Table 3-2: Pinout for 247-ball TFBGA Package

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	D13	F7	CFIOR/NBS1/NWR1	K10	GND	P17	RTCK
A2	D12	F8	SDA10	K11	VDDIOM	P18	PB16
A12	A9	F9	NBS0/A0	K12	GND	R2	GND
A14	A13	F10	A6	K13	GND	R3	PB29
A16	A20	F11	A12	K14	XOUT32	R5	PB26
A18	A22	F12	A15	K15	XIN32	R6	PB27
A19	NANDOE	F13	BA1/A17	K17	HDPA	R7	PA5
B1	D15	F14	PC10	K18	HDMA	R8	GND
B2	D14	F15	PC14	L2	NC	R9	PA12
B3	D10	F16	VDDUSB	L3	NC	R10	GND

© 2017 Microchip Technology Inc.

Table 3-2: Pinout for 247-ball TFBGA Package (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
B4	D9	F17	PC9	L5	ADVREF	R11	PA19
B5	D7	F18	PC12	L6	PC2	R12	PA26
B6	D3	G2	PC26	L7	GND	R13	PB1
B7	D2	G3	PC25	L8	GND	R14	GND
B8	RAS	G5	PC24	L9	GND	R15	PB7
B9	CAS	G6	PC21	L10	GND	R17	PB14
B10	NWR2/NBS2/A1	G8	VDDCORE	L11	VDDCORE	R18	PB9
B11	A3	G9	A5	L12	GND	T2	PA1
B13	A10	G10	VDDCORE	L13	OSCSEL	T3	PB10
B15	A18	G11	VDDCORE	L14	GNDBU	T17	PB19
B17	A21	G12	VDDCORE	L15	GND	T18	PB17
B19	VDDUSB	G14	PC13	L17	NRST	U2	GNDANA
C2	PC15	G15	GND	L18	TCK	U3	PB21
C3	D11	G17	GNDUSB	M2	PC0	U4	PB28
C4	D8	G18	PC11	M3	PC1	U5	PB31
C5	SDCKE	H2	PC31	M5	PC3	U6	PA4
C6	SDWE	H3	PC30	M6	NTRST	U7	PA3
C7	SDCK	H5	PC28	M7	GND	U8	PA9
C8	D1	H6	PC27	M8	GND	U9	GND
C9	SDCS/NCS1	H7	PC29	M9	GND	U10	PA15
C10	A2	H8	GND	M10	PA16	U11	PA21
C11	A7	H9	GND	M11	VDDCORE	U12	PA25
C12	A11	H10	VDDIOM	M12	GND	U13	PA29
C14	A19	H11	VDDIOM	M13	VDDIOP	U14	PA27
C16	GNDUSB	H12	GND	M14	TST	U15	PA31
C18	CFWE/NWE/NWR0	H13	VDDCORE	M15	JTAGSEL	U16	GND
D2	PC17	H14	SHDW	M17	PB18	U17	PB2
D3	PC16	H15	VDDBU	M18	TMS	U18	GND
D13	A14	H17	HDPB	N2	PB20	V1	PB12
D15	NANDWE	H18	HDMB	N3	PB13	V2	PB23
D17	CFOE/NRD	J2	VDDOSC	N5	PB11	V3	PB30
D19	NCS0	J3	VDDPLL	N6	BMS	V4	PA2
E2	PC18	J5	XOUT	N8	GND	V5	PA8
E3	PC19	J6	XIN	N11	PA17	V6	PA10
E5	D6	J7	VDDPLL	N12	PA23	V7	PA13
E6	D5	J8	GND	N14	GND	V8	VDDIOP

© 2017 Microchip Technology Inc.

Table 3-2: Pinout for 247-ball TFBGA Package (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
E7	D0	J9	VDDIOM	N15	VDDIOP	V9	PA14
E8	CFIOW/NBS3/NWR3	J10	VDDIOM	N17	TDO	V10	VDDIOP
E9	GND	J11	VDDIOM	N18	TDI	V11	PA20
E10	A4	J12	GND	P2	PB24	V12	PA22
E11	A8	J13	GND	P3	PB22	V13	VDDIOP
E12	VDDIOM	J14	WKUP	P5	GND	V14	PA30
E13	BA0/A16	J15	DDP	P6	GND	V15	PB0
E14	PC8	J17	DDM	P7	PA6	V16	GND
E15	PC4	J18	VDDIOP	P8	PA7	V17	PB4
E16	PC5	K2	GNDPLL	P9	PA11	V18	GND
E18	PC7	K3	GND	P10	GND	V19	PB6
E19	PC6	K5	NC	P11	PA18	W1	PB25
F2	PC22	K6	GNDPLL	P12	PA24	W2	PA0
F3	PC23	K7	VDDANA	P13	PA28	W18	PB8
F5	PC20	K8	GND	P14	PB3	W19	PB15
F6	D4	K9	GND	P15	PB5		

© 2017 Microchip Technology Inc.

4. Power Considerations

4.1 Power Supplies

The SAM9G20 has several types of power supply pins. Some supply pins share common ground (GND) pins whereas others have separate grounds. See Table 4-1.

Table 4-1: SAM9G20 Power Supply Pins

Pin(s)	Item(s) powered	Range	Nominal	Ground
VDDCORE	Core, including the processor Embedded memories Peripherals	0.9–1.1 V	1.0V	GND
VDDIOM	External Bus Interface I/O lines	1.65–1.95 V	1.8V	
		3.0–3.6 V	3.3V	
VDDOSC	Main Oscillator cells	1.65–3.6 V	–	
VDDIOP	Peripherals I/O lines	1.65–1.95 V	1.8V	
		3.0–3.6 V	3.3V	
VDDBU	Slow Clock oscillator Internal RC oscillator Part of the System Controller	0.9–1.1 V	1.0V	GNDBU
VDDPLL	PLL cells	0.9–1.1 V	1.0V	GNDPLL
VDDUSB	USB transceiver	3.0–3.6 V	3.3V	GNDUSB
VDDANA	Analog-to-Digital Converter	3.0–3.6 V	3.3V	GNDANA

4.2 Programmable I/O Lines

The power supply pins VDDIOM accept two voltage ranges. This allows the device to reach its maximum speed either out of 1.8V or 3.3V external memories.

The maximum speed is 133 MHz on the pin SDCK (SDRAM Clock) loaded with 10 pF. The other signals (control, address and data signals) do not go over 66 MHz, loaded with 30 pF for power supply at 1.8V and 50 pF for power supply at 3.3V.

The EBI I/Os accept two slew rate modes, Fast and Slow. This allows to adapt the rising and falling time on SDRAM clock, control and data to the bus load.

The voltage ranges and the slew rates are determined by programming VDDIOMSEL and IOSR bits in the Chip Configuration registers located in the Matrix User Interface.

At reset, the selected voltage defaults to 3.3V nominal and power supply pins can accept either 1.8V or 3.3V. The user must make sure to program the EBI voltage range before getting the device out of its Slow Clock mode.

At reset, the selected slew rates defaults are Fast.

4.3 Power Sequence Requirements

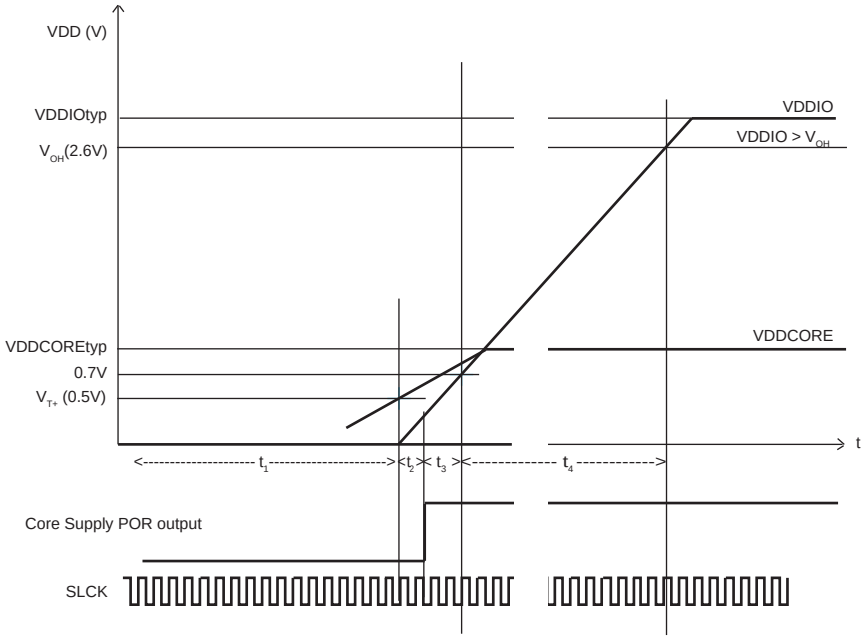
The SAM9G20 board design must comply with the power-up and power-down sequence guidelines described in the following sections to guarantee reliable operation of the device. Any deviation from these sequences may lead to the following situations:

- Excessive current consumption during the power-up phase which, in the worst case, can result in irreversible damage to the device.
- Prevent the device from booting.

4.3.1 Power-up Sequence

The power sequence described below is applicable to all the SAM9G20 revisions. However, the power sequence can be simplified for the revision B device. In this revision, the over consumption during the power-up phase has been limited to less than 200 mA. This current can not damage the device and if it is acceptable for the final application, the power sequence becomes VDDIO followed by VDDCORE. VDDIO must be established first (> 0.7V) to ensure a correct sampling of the BMS signal and also to guarantee the correct voltage level when accessing an external memory.

Figure 4-1: VDDCORE and VDDIO Constraints at Startup



VDDCORE and VDDBU are controlled by Power-on-Reset (POR) to guarantee that these power sources reach their target values prior to the release of POR. (See Figure 4-1.)

- VDDIOM and VDDIOP must NOT be powered until VDDCORE has reached a level superior or equal to V_{T+} (0.5V).
- VDDIOP must be ≥ V_{IH} (refer to Table 40-2 ”DC Characteristics” for more details), (t_{RST} + t_1) at the latest, after VDDCORE has reached V_{T+}.
- VDDIOM must reach V_{OH} (refer to Table 40-2 ”DC Characteristics” for more details), (t_{RST} + t_1 + t_2) at the latest, after VDDCORE has reached V_{T+}.
- t_2 = t_{RST} = 30 µs
- t_3 = 3 × t_{SLCK}
- t_4 = 14 × t_{SLCK}

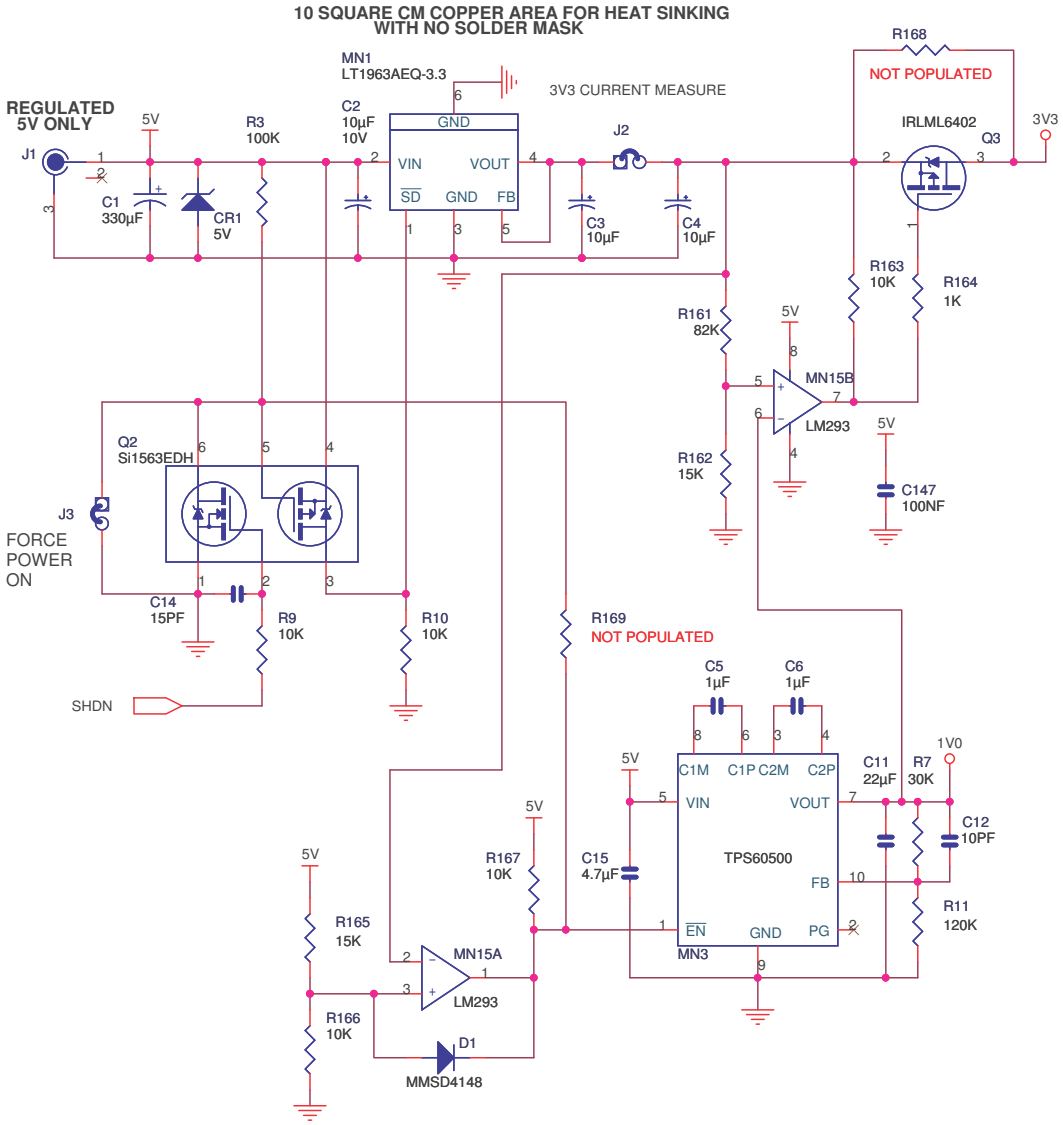
The t_{SLCK} min (22 µs) is obtained for the maximum frequency of the internal RC oscillator (44 kHz). This gives:

- t_2 = t_{RST} = 30 µs
- t_3 = 66 µs
- t_4 = 308 µs

Figure 4-2 shows an example of the implementation on the evaluation kit AT91SAM9G20-EK Rev.C.

© 2017 Microchip Technology Inc.

Figure 4-2: Power-up Sequence Implementation Example on AT91SAM9G20-EK Rev.C



4.3.2 Power-down Sequence

Switch off the VDDIOM and VDDIOP power supplies prior to, or at the same time, as VDDCORE. No power-up or power-down restrictions apply to VDDBU, VDDPL, VDDANA and VDDUSB.

5. I/O Line Considerations

5.1 JTAG Port Pins

TMS, TDI and TCK are schmitt trigger inputs and have no pull-up resistors.

TDO and RTCK are outputs, driven at up to VDDIOP, and have no pull-up resistor.

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

The NTRST signal is described in Section 5.3 “Reset Pins”.

All the JTAG signals are supplied with VDDIOP.

5.2 Test Pin

The TST pin is used for manufacturing test purposes when asserted high. It integrates a permanent pull-down resistor of about 15 kΩ to GNDBU, so that it can be left unconnected for normal operations. Driving this line at a high level leads to unpredictable results.

This pin is supplied with VDDBU.

5.3 Reset Pins

NRST is an open-drain output integrating a non-programmable pull-up resistor. It can be driven with voltage at up to VDDIOP.

NTRST is an input which allows reset of the JTAG Test Access port. It has no action on the processor.

As the product integrates power-on reset cells, which manages the processor and the JTAG reset, the NRST and NTRST pins can be left unconnected.

The NRST and NTRST pins both integrate a permanent pull-up resistor of 100 kΩ minimum to VDDIOP.

The NRST signal is inserted in the Boundary Scan.

5.4 PIO Controllers

All the I/O lines are Schmitt trigger inputs and all the lines managed by the PIO Controllers integrate a programmable pull-up resistor of 75 kΩ typical with the exception of P4–P31. For details, refer to Section 40. “Electrical Characteristics”. Programming of this pull-up resistor is performed independently for each I/O line through the PIO Controllers.

5.5 I/O Line Drive Levels

The PIO lines drive current capability is described in Section 40.2 “DC Characteristics”.

5.6 Shutdown Logic Pins

The SHDN pin is a tri-state output only pin, which is driven by the Shutdown Controller. There is no internal pull-up. An external pull-up to VDDBU is needed and its value must be higher than 1 MΩ.. The resistor value is calculated according to the regulator enable implementation and the SHDN level.

The pin WKUP is an input-only. It can accept voltages only between 0V and VDDBU.

© 2017 Microchip Technology Inc.

6. Processor and Architecture

6.1 Arm926EJ-S Processor

- RISC Processor Based on Arm v5TEJ Architecture with Jazelle technology for Java acceleration
- Two Instruction Sets
  - Arm High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- DSP Instruction Extensions
- 5-Stage Pipeline Architecture:
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)
  - Data Memory (M)
  - Register Write (W)
- 32-Kbyte Data Cache, 32-Kbyte Instruction Cache
  - Virtually-addressed 4-way Associative Cache
  - Eight words per line
  - Write-through and Write-back Operation
  - Pseudo-random or Round-robin Replacement
- Write Buffer
  - Main Write Buffer with 16-word Data Buffer and 4-address Buffer
  - DCache Write-back Buffer with 8-word Entries and a Single Address Entry
  - Software Control Drain
- Standard Arm v4 and v5 Memory Management Unit (MMU)
  - Access Permission for Sections
  - Access Permission for large pages and small pages can be specified separately for each quarter of the page
  - 16 embedded domains
- Bus Interface Unit (BIU)
  - Arbitrates and Schedules AHB Requests
  - Separate Masters for both instruction and data access providing complete Matrix system flexibility
  - Separate Address and Data Buses for both the 32-bit instruction interface and the 32-bit data interface
  - On Address and Data Buses, data can be 8-bit (Bytes), 16-bit (Half-words) or 32-bit (Words)

6.2 Bus Matrix

- 6-layer Matrix, handling requests from 6 masters
- Programmable Arbitration strategy
  - Fixed-priority Arbitration
  - Round-Robin Arbitration, either with no default master, last accessed default master or fixed default master
- Burst Management
  - Breaking with Slot Cycle Limit Support
  - Undefined Burst Length Support
- One Address Decoder provided per Master
  - Three different slaves may be assigned to each decoded memory area: one for internal boot, one for external boot, one after remap
- Boot Mode Select
  - Non-volatile Boot Memory can be internal or external
  - Selection is made by BMS pin sampled at reset
- Remap Command
  - Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory
- Allows Handling of Dynamic Exception Vectors

© 2017 Microchip Technology Inc.

6.2.1 Matrix Masters

The Bus Matrix of the SAM9G20 manages six Masters, which means that each master can perform an access concurrently with others, according the slave it accesses is available.

Each Master has its own decoder that can be defined specifically for each master. In order to simplify the addressing, all the masters have the same decodings.

Table 6-1: List of Bus Matrix Masters

Master 0	Arm926™ Instruction
Master 1	Arm926 Data
Master 2	PDC
Master 3	ISI Controller
Master 4	Ethernet MAC
Master 5	USB Host DMA

6.2.2 Matrix Slaves

Each Slave has its own arbiter, thus allowing to program a different arbitration per Slave.

Table 6-2: List of Bus Matrix Slaves

Slave 0	Internal SRAM0 16 Kbytes
Slave 1	Internal SRAM1 16 Kbytes
Slave 2	Internal ROM
	USB Host User Interface
Slave 3	External Bus Interface
Slave 4	Internal Peripherals

6.2.3 Masters to Slaves Access

All the Masters can normally access all the Slaves. However, some paths do not make sense, like as example allowing access from the Ethernet MAC to the Internal Peripherals. Thus, these paths are forbidden or simply not wired, and shown “–” in Table 6-3.

Table 6-3: SAM9G20 Masters to Slaves Access

Master		0 & 1	2	3	4	5
Slave		Arm926 Instruction & Data	Peripheral DMA Controller	ISI Controller	Ethernet MAC	USB Host Controller
0	Internal SRAM 16 Kbytes	X	X	X	X	X
1	Internal SRAM 16 Kbytes	X	X	X	X	X
2	Internal ROM	X	X	–	–	–
	UHP User Interface	X	X	–	–	–
3	External Bus Interface	X	X	X	X	X
4	Internal Peripherals	X	X	–	–	–

© 2017 Microchip Technology Inc.

6.3 Peripheral DMA Controller

- Acting as one Matrix Master
- Allows data transfers from/to peripheral to/from any memory space without any intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- Twenty-four channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for the Serial Synchronous Controller
  - Two for each Serial Peripheral Interface
  - One for Multimedia Card Interface
  - One for Analog-to-Digital Converter
  - Two for the Two-wire Interface

The Peripheral DMA Controller handles transfer requests from the channel according to the following priorities (Low to High priorities):

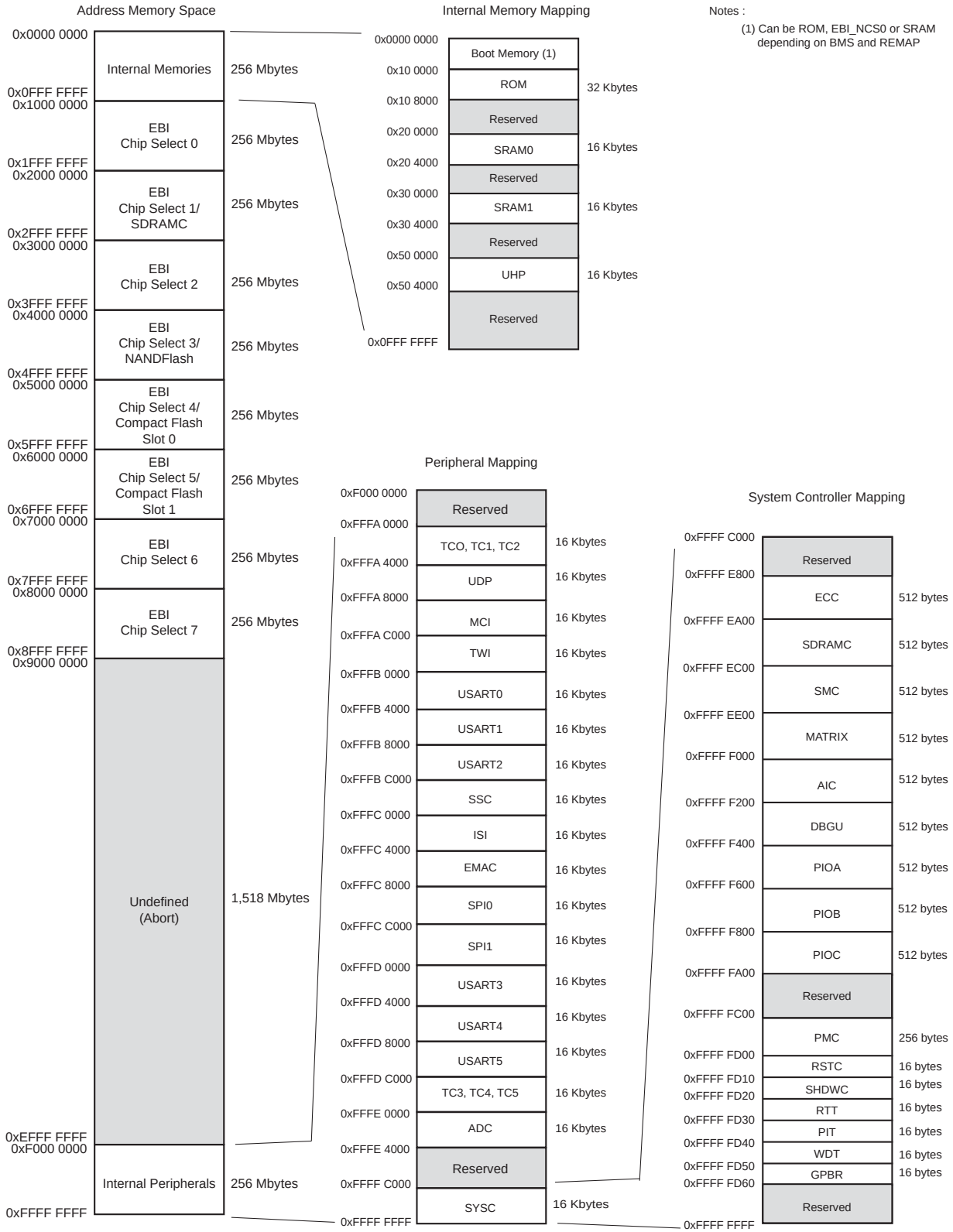
- TWI Transmit Channel
- DBGU Transmit Channel
- USART5 Transmit Channel
- USART4 Transmit Channel
- USART3 Transmit Channel
- USART2 Transmit Channel
- USART1 Transmit Channel
- USART0 Transmit Channel
- SPI1 Transmit Channel
- SPI0 Transmit Channel
- SSC Transmit Channel
- TWI Receive Channel
- DBGU Receive Channel
- USART5 Receive Channel
- USART4 Receive Channel
- USART3 Receive Channel
- USART2 Receive Channel
- USART1 Receive Channel
- USART0 Receive Channel
- ADC Receive Channel
- SPI1 Receive Channel
- SPI0 Receive Channel
- SSC Receive Channel
- MCI Transmit/Receive Channel

6.4 Debug and Test Features

- Arm926 Real-time In-circuit Emulator
  - Two real-time Watchpoint Units
  - Two Independent Registers: Debug Control Register and Debug Status Register
  - Test Access Port Accessible through JTAG Protocol
  - Debug Communications Channel
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

7. Memories

Figure 7-1: SAM9G20 Memory Mapping



© 2017 Microchip Technology Inc.

A first level of address decoding is performed by the Bus Matrix, i.e., the implementation of the Advanced High Performance Bus (AHB) for its Master and Slave interfaces with additional features.

Decoding breaks up the 4 Gbytes of address space into 16 banks of 256 Mbytes. Banks 1 to 7 are directed to the EBI that associates these banks to the external chip selects EBI_NCS0 to EBI_NCS7. Bank 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1 Mbyte of internal memory area. Bank 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

Other areas are unused and performing an access within them provides an abort to the master requesting such an access.

Each Master has its own bus and its own decoder, thus allowing a different memory mapping per Master. However, in order to simplify the mappings, all the masters have a similar address decoding.

Regarding Master 0 and Master 1 (Arm926 Instruction and Data), three different Slaves are assigned to the memory space decoded at address 0x0: one for internal boot, one for external boot, one after remap. Refer to Table 7-1 ”Internal Memory Mapping” for details.

7.1 Embedded Memories

- 64 Kbyte ROM
  - Single Cycle Access at full matrix speed
- Two 16 Kbyte Fast SRAM
  - Single Cycle Access at full matrix speed

7.1.1 Boot Strategies

Table 7-1 summarizes the Internal Memory Mapping for each Master, depending on the Remap status and the BMS state at reset.

Table 7-1: Internal Memory Mapping

Address	REMAP = 0		REMAP = 1
	BMS = 1	BMS = 0	
0x0000 0000	ROM	EBI_NCS0	SRAM0 16 KB
0x0010 0000	ROM		
0x0020 0000	SRAM0 16 KB		
0x0030 0000	SRAM1 16 KB		
0x0050 0000	USB Host User Interface		

The system always boots at address 0x0. To ensure a maximum number of possibilities for boot, the memory layout can be configured with two parameters.

REMAP allows the user to lay out the first internal SRAM bank to 0x0 to ease development. This is done by software once the system has booted. When REMAP = 1, BMS is ignored. Refer to Section 18. “SAM9G20 Bus Matrix” for more details.

When REMAP = 0, BMS allows the user to lay out to 0x0, at his convenience, the ROM or an external memory. This is done via hardware at reset.

Note: Memory blocks not affected by these parameters can always be seen at their specified base addresses. See the complete memory map presented in Figure 7-1.

The SAM9G20 matrix manages a boot memory that depends on the level on the BMS pin at reset. The internal memory area mapped between address 0x0 and 0x000F FFFF is reserved for this purpose.

If BMS is detected at 1, the boot memory is the embedded ROM.

If BMS is detected at 0, the boot memory is the memory connected on the Chip Select 0 of the External Bus Interface.

7.1.1.1 BMS = 1, Boot on Embedded ROM

The system boots using the Boot Program.

- Boot on slow clock (On-chip RC or 32768 Hz)
- Auto baudrate detection
- Downloads and runs an application from external storage media into internal SRAM
- Downloaded code size depends on embedded SRAM size

© 2017 Microchip Technology Inc.

- Automatic detection of valid application
- Bootloader on a non-volatile memory
  - SDCard (boot ROM does not support high capacity SDCards.)
  - NAND Flash
  - SPI DataFlash and Serial Flash connected on NPCS0 and NPCS1 of the SPI0
  - EEPROM on TWI
- SAM-BA® Boot in case no valid program is detected in external NVM, supporting
  - Serial communication on a DBGU
  - USB Device HS Port

7.1.1.2 BMS = 0, Boot on External Memory

- Boot on slow clock (On-chip RC or 32768 Hz)
- Boot with the default configuration for the Static Memory Controller, byte select mode, 16-bit data bus, Read/Write controlled by Chip Select, allows boot on 16-bit non-volatile memory.

The customer-programmed software must perform a complete configuration.

To speed up the boot sequence when booting at 32 kHz EBI CS0 (BMS = 0), the user must take the following steps:

1. Program the PMC (main oscillator enable or bypass mode).
2. Program and start the PLL.
3. Reprogram the SMC setup, cycle, hold, mode timings registers for CS0 to adapt them to the new clock.
4. Switch the main clock to the new value.

7.2 External Memories

The external memories are accessed through the External Bus Interface. Each Chip Select line has a 256-Mbyte memory area assigned.

Refer to the memory map in Figure 7-1.

7.2.1 External Bus Interface

- Integrates three External Memory Controllers
  - Static Memory Controller
  - SDRAM Controller
  - ECC Controller
- Additional logic for NAND Flash
- Full 32-bit External Data Bus
- Up to 26-bit Address Bus (up to 64 Mbytes linear)
- Up to 8 chip selects, Configurable Assignment:
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2
  - Static Memory Controller on NCS3, Optional NAND Flash support
  - Static Memory Controller on NCS4–NCS5, Optional CompactFlash support
  - Static Memory Controller on NCS6–NCS7

7.2.2 Static Memory Controller

- 8-, 16- or 32-bit Data Bus
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Asynchronous read in Page Mode supported (4- up to 32-byte page size)
- Multiple device adaptability
  - Compliant with LCD Module
  - Control signals programmable setup, pulse and hold time for each Memory Bank

© 2017 Microchip Technology Inc.

- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time
- Slow Clock mode supported

7.2.3 SDRAM Controller

- Supported devices
  - Standard and Low-power SDRAM (Mobile SDRAM)
- Numerous configurations supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with two or four Internal Banks
  - SDRAM with 16- or 32-bit Datapath
- Programming facilities
  - Word, half-word, byte access
  - Automatic page break when Memory Boundary has been reached
  - Multibank Ping-pong Access
  - Timing parameters specified by software
  - Automatic refresh operation, refresh rate is programmable
- Energy-saving capabilities
  - Self-refresh, power down and deep power down modes supported
- Error detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by software
- CAS Latency of 1, 2 and 3 supported
- Auto Precharge Command not used

7.2.4 Error Correction Code Controller

- Hardware Error Correction Code (ECC) Generation
  - Detection and Correction by Software
- Supports NAND Flash and SmartMedia™ Devices with 8- or 16-bit Data Path.
- Supports NAND Flash/SmartMedia with Page Sizes of 528, 1056, 2112 and 4224 Bytes, Specified by Software
- Supports 1 bit correction for a page of 512,1024,2048 and 4096 Bytes with 8- or 16-bit Data Path
- Supports 1 bit correction per 512 bytes of data for a page size of 512, 2048 and 4096 Bytes with 8-bit Data Path
- Supports 1 bit correction per 256 bytes of data for a page size of 512, 2048 and 4096 Bytes with 8-bit Data Path

© 2017 Microchip Technology Inc.

8. System Controller

The System Controller is a set of peripherals, which allow handling of key elements of the system, such as power, resets, clocks, time, interrupts, watchdog, etc.

The System Controller User Interface embeds also the registers allowing to configure the Matrix and a set of registers for the chip configuration. The chip configuration registers allows configuring:

- EBI chip select assignment and Voltage range for external memories

The System Controller's peripherals are all mapped within the highest 16 Kbytes of address space, between addresses 0xFFFF E800 and 0xFFFF FFFF.

However, all the registers of System Controller are mapped on the top of the address space. All the registers of the System Controller can be addressed from a single pointer by using the standard Arm instruction set, as the Load/Store instruction has an indexing mode of ±4 Kbytes.

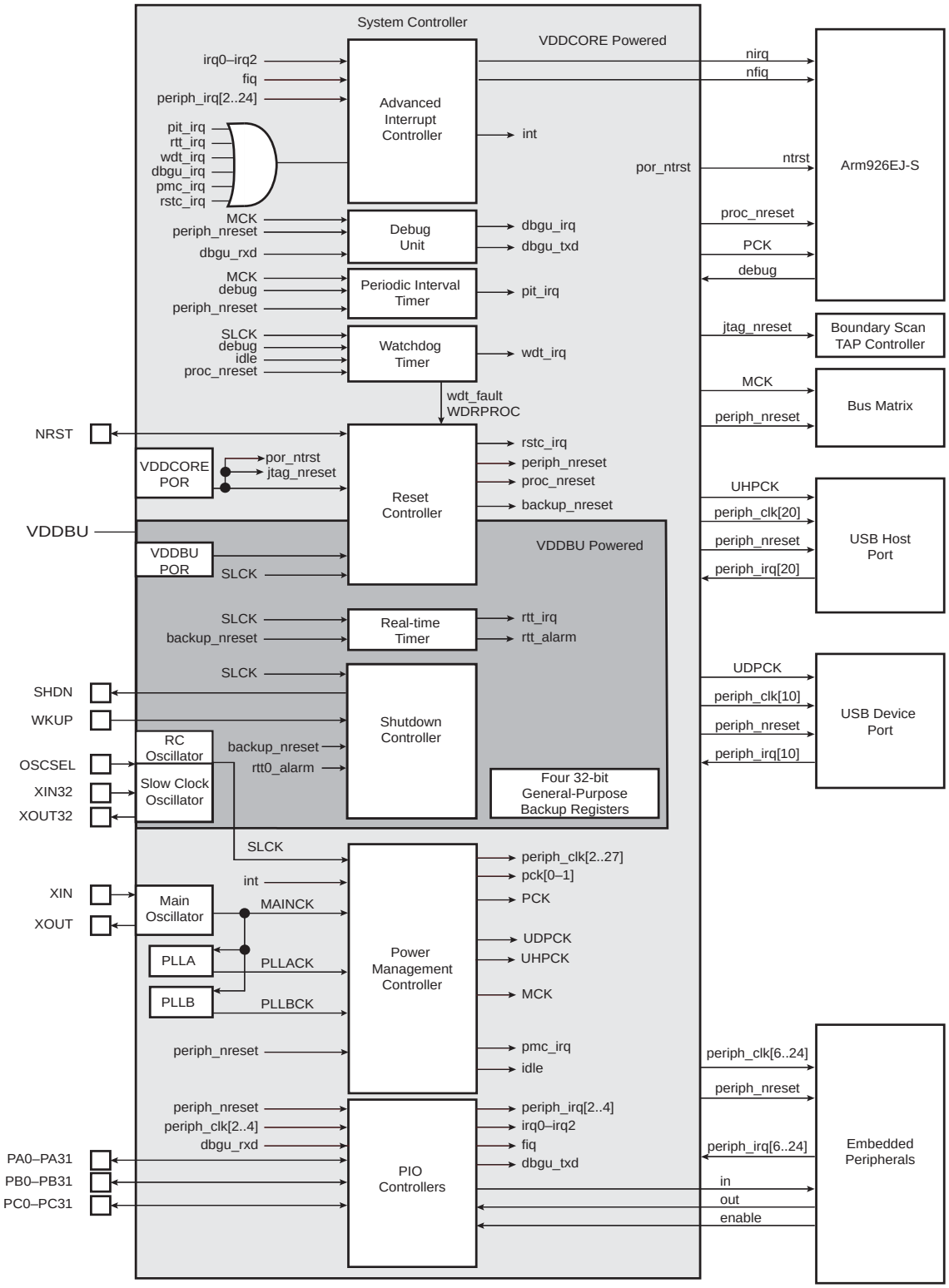
Figure 8-1 shows the System Controller block diagram.

Figure 7-1 shows the mapping of the User Interfaces of the System Controller peripherals.

© 2017 Microchip Technology Inc.

8.1 System Controller Block Diagram

Figure 8-1: SAM9G20 System Controller Block Diagram



8.2 Reset Controller

- Based on two Power-on-Reset cells
  - one on VDDBU and one on VDDCORE
- Status of the last reset
  - Either general reset (VDDBU rising), wake-up reset (VDDCORE rising), software reset, user reset or watchdog reset
- Controls the internal resets and the NRST pin output
  - Allows shaping a reset signal for the external devices

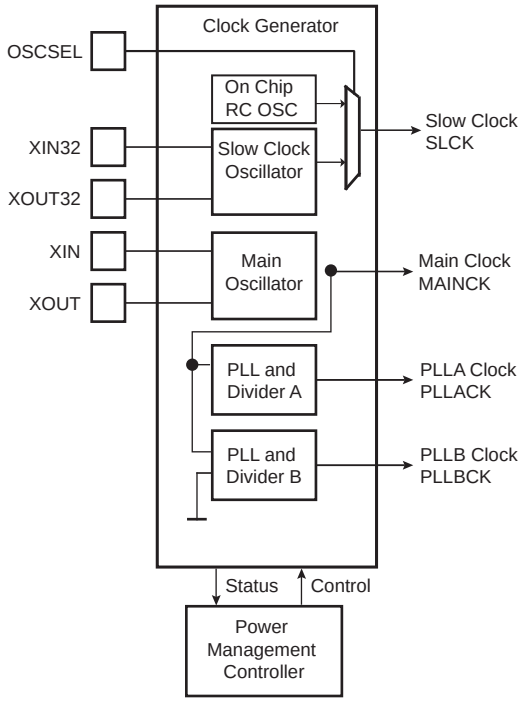
8.3 Shutdown Controller

- Shutdown and Wake-Up logic
  - Software programmable assertion of the SHDN pin
  - Deassertion Programmable on a WKUP pin level change or on alarm

8.4 Clock Generator

- Embeds a Low power 32768 Hz Slow Clock Oscillator and a Low power RC oscillator selectable with OSCSEL signal
  - Provides the permanent Slow Clock SLCK to the system
- Embeds the Main Oscillator
  - Oscillator bypass feature
  - Supports 3 to 20 MHz crystals
- Embeds two PLLs
  - The PLL A outputs 400–800 MHz clock
  - The PLL B outputs 100 MHz clock
  - Both integrate an input divider to increase output accuracy
  - PLL A and PLL B embed their own filters

Figure 8-2: Clock Generator Block Diagram

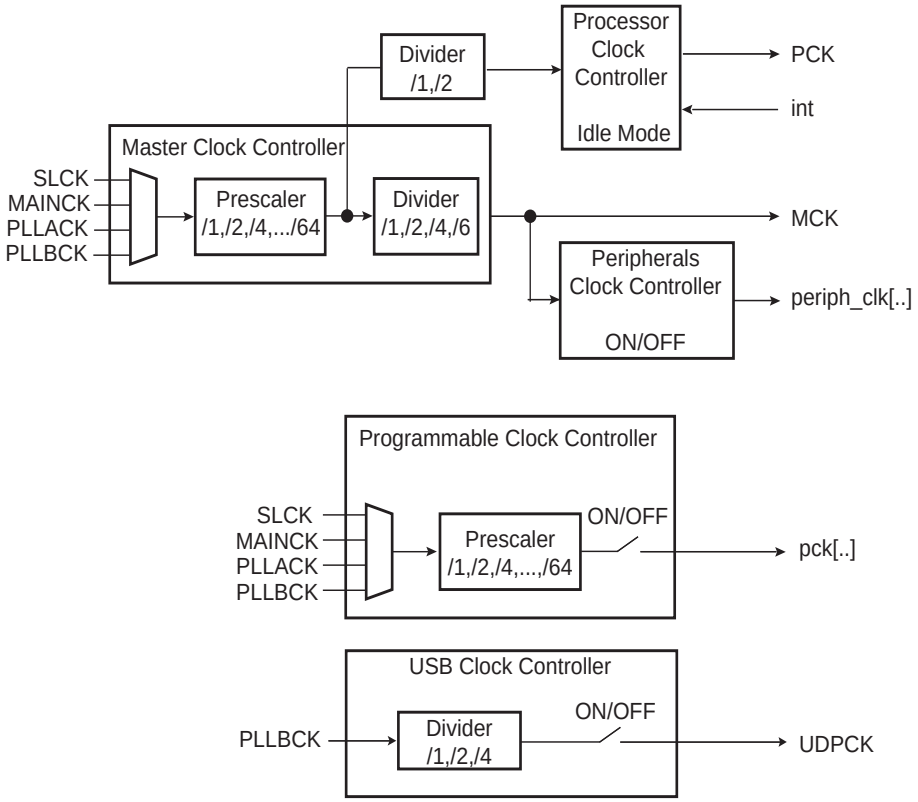


© 2017 Microchip Technology Inc.

8.5 Power Management Controller

- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK, in particular to the Matrix and the memory interfaces.The MCK divider can be 1,2,4,6
  - the USB Device Clock UDPCK
  - independent peripheral clocks, typically at the frequency of MCK
  - 2 programmable clock outputs: PCK0, PCK1
- Five flexible operating modes:
  - Normal Mode, processor and peripherals running at a programmable frequency
  - Idle Mode, processor stopped waiting for an interrupt
  - Slow Clock Mode, processor and peripherals running at low frequency
  - Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
  - Backup Mode, Main Power Supplies off, VDDBU powered by a battery

Figure 8-3: SAM9G20 Power Management Controller Block Diagram



8.6 Periodic Interval Timer

- Includes a 20-bit Periodic Counter, with less than 1 µs accuracy
- Includes a 12-bit Interval Overlay Counter
- Real Time OS or Linux®/Windows CE® compliant tick generator

8.7 Watchdog Timer

- 16-bit key-protected only-once-Programmable Counter
- Windowed, prevents the processor being in a dead-lock on the watchdog access

8.8 Real-time Timer

- Real-time Timer 32-bit free-running back-up Counter
- Integrates a 16-bit programmable prescaler running on slow clock
- Alarm Register capable of generating a wake-up of the system through the Shutdown Controller

8.9 General-purpose Backup Registers

- Four 32-bit general-purpose backup registers

8.10 Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of the Arm Processor
- Thirty-two individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive
- Three External Sources plus the Fast Interrupt signal
- 8-level Priority Controller
  - Drives the Normal Interrupt of the processor
  - Handles priority of the interrupt sources 1 to 31
  - 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 when protect models are enabled
- Fast Forcing
  - Permits redirecting any normal interrupt source on the Fast Interrupt of the processor

8.11 Debug Unit

- Composed of two functions:
  - Two-pin UART
  - Debug Communication Channel (DCC) support
- Two-pin UART
  - Implemented features are 100% compatible with the standard Microchip USART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Support for two PDC channels with connection to receiver and transmitter
- Debug Communication Channel Support
  - Offers visibility of and interrupt trigger from COMMRX and COMMTX signals from the Arm Processor’s ICE Interface

8.12 Chip Identification

- Chip ID:0x019905A1
- JTAG ID: 0x05B2403F
- Arm926 TAP ID:0x0792603F

© 2017 Microchip Technology Inc.

9. Peripherals

9.1 User Interface

The peripherals are mapped in the upper 256 Mbytes of the address space between the addresses 0xFFFA 0000 and 0xFFFC FFFF. Each User Peripheral is allocated 16 Kbytes of address space. A complete memory map is presented in Figure 7-1.

9.2 Peripheral Identifiers

Table 9-1 defines the Peripheral Identifiers of the SAM9G20. A peripheral identifier is required for the control of the peripheral interrupt with the Advanced Interrupt Controller and for the control of the peripheral clock with the Power Management Controller.

Table 9-1: Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC	System Controller Interrupt	–
2	PIOA	Parallel I/O Controller A	–
3	PIOB	Parallel I/O Controller B	–
4	PIOC	Parallel I/O Controller C	–
5	ADC	Analog to Digital Converter	–
6	US0	USART 0	–
7	US1	USART 1	–
8	US2	USART 2	–
9	MCI	Multimedia Card Interface	–
10	UDP	USB Device Port	–
11	TWI	Two-wire Interface	–
12	SPI0	Serial Peripheral Interface 0	–
13	SPI1	Serial Peripheral Interface 1	–
14	SSC	Synchronous Serial Controller	–
15	–	Reserved	–
16	–	Reserved	–
17	TC0	Timer/Counter 0	–
18	TC1	Timer/Counter 1	–
19	TC2	Timer/Counter 2	–
20	UHP	USB Host Port	–
21	EMAC	Ethernet MAC	–
22	ISI	Image Sensor Interface	–
23	US3	USART 3	–
24	US4	USART 4	–
25	US5	USART 5	–
26	TC3	Timer/Counter 3	–
27	TC4	Timer/Counter 4	–
28	TC5	Timer/Counter 5	–

© 2017 Microchip Technology Inc.

Table 9-1: Peripheral Identifiers (Continued)

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
29	AIC	Advanced Interrupt Controller	IRQ0
30	AIC	Advanced Interrupt Controller	IRQ1
31	AIC	Advanced Interrupt Controller	IRQ2

Note: Setting AIC, SYSC, UHP, ADC and IRQ0-2 bits in the clock set/clear registers of the PMC has no effect. The ADC clock is automatically started for the first conversion. In Sleep Mode the ADC clock is automatically stopped after each conversion.

9.2.1 Peripheral Interrupts and Clock Control

9.2.1.1 System Interrupt

The System Interrupt in Source 1 is the wired-OR of the interrupt signals coming from:

- the SDRAM Controller
- the Debug Unit
- the Periodic Interval Timer
- the Real-time Timer
- the Watchdog Timer
- the Reset Controller
- the Power Management Controller

The clock of these peripherals cannot be deactivated and Peripheral ID 1 can only be used within the Advanced Interrupt Controller.

9.2.1.2 External Interrupts

All external interrupt signals, i.e., the Fast Interrupt signal FIQ or the Interrupt signals IRQ0 to IRQ2, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

9.3 Peripheral Signal Multiplexing on I/O Lines

The SAM9G20 features three PIO controllers (PIOA, PIOB, PIOC) that multiplex the I/O lines of the peripheral set.

Each PIO Controller controls up to 32 lines. Each line can be assigned to one of two peripheral functions, A or B. Table 9-2, Table 9-3 and Table 9-4 define how the I/O lines of the peripherals A and B are multiplexed on the PIO controllers. The two columns “Function” and “Comments” have been inserted in this table for the user’s own comments; they may be used to track how pins are defined in an application.

Note that some peripheral functions which are output only might be duplicated within both tables.

The column “Reset State” indicates whether the PIO Line resets in I/O mode or in peripheral mode. If I/O appears, the PIO Line resets in input with the pull-up enabled, so that the device is maintained in a static state as soon as the reset is released. As a result, the bit corresponding to the PIO Line in the PIO Controller PIO Status Register (PIO_PSR) resets low.

If a signal name appears in the “Reset State” column, the PIO Line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released. Note that the pull-up resistor is also enabled in this case.

© 2017 Microchip Technology Inc.

9.3.1 PIO Controller A Multiplexing

Table 9-2: Multiplexing on PIO Controller A

PIO Controller A					Application Usage		
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	Comments
PA0	SPI0_MISO	MCDB0		I/O	VDDIOP	–	
PA1	SPI0_MOSI	MCCDB		I/O	VDDIOP	–	
PA2	SPI0_SPCK	–		I/O	VDDIOP	–	
PA3	SPI0_NPCS0	MCDB3		I/O	VDDIOP	–	
PA4	RTS2	MCDB2		I/O	VDDIOP	–	
PA5	CTS2	MCDB1		I/O	VDDIOP	–	
PA6	MCDA0	–		I/O	VDDIOP	–	
PA7	MCCDA	–		I/O	VDDIOP	–	
PA8	MCCK	–		I/O	VDDIOP	–	
PA9	MCDA1	–		I/O	VDDIOP	–	
PA10	MCDA2	ETX2		I/O	VDDIOP	–	
PA11	MCDA3	ETX3		I/O	VDDIOP	–	
PA12	ETX0	–		I/O	VDDIOP	–	
PA13	ETX1	–		I/O	VDDIOP	–	
PA14	ERX0	–		I/O	VDDIOP	–	
PA15	ERX1	–		I/O	VDDIOP	–	
PA16	ETXEN	–		I/O	VDDIOP	–	
PA17	ERXDV	–		I/O	VDDIOP	–	
PA18	ERXER	–		I/O	VDDIOP	–	
PA19	ETXCK	–		I/O	VDDIOP	–	
PA20	EMDC	–		I/O	VDDIOP	–	
PA21	EMDIO	–		I/O	VDDIOP	–	
PA22	ADTRG	ETXER		I/O	VDDIOP	–	
PA23	TWD	ETX2		I/O	VDDIOP	–	
PA24	TWCK	ETX3		I/O	VDDIOP	–	
PA25	TCLK0	ERX2		I/O	VDDIOP	–	
PA26	TIOA0	ERX3		I/O	VDDIOP	–	
PA27	TIOA1	ERXCK		I/O	VDDIOP	–	
PA28	TIOA2	ECRS		I/O	VDDIOP	–	
PA29	SCK1	ECOL		I/O	VDDIOP	–	
PA30	SCK2	RXD4		I/O	VDDIOP	–	
PA31	SCK0	TXD4		I/O	VDDIOP	–	

9.3.2 PIO Controller B Multiplexing

Table 9-3: Multiplexing on PIO Controller B

PIO Controller B					Application Usage		
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	Comments
PB0	SPI1_MISO	TIOA3		I/O	VDDIOP	–	
PB1	SPI1_MOSI	TIOB3		I/O	VDDIOP	–	
PB2	SPI1_SPCK	TIOA4		I/O	VDDIOP	–	
PB3	SPI1_NPCS0	TIOA5		I/O	VDDIOP	–	
PB4	TXD0	–		I/O	VDDIOP	–	
PB5	RXD0	–		I/O	VDDIOP	–	
PB6	TXD1	TCLK1		I/O	VDDIOP	–	
PB7	RXD1	TCLK2		I/O	VDDIOP	–	
PB8	TXD2	–		I/O	VDDIOP	–	
PB9	RXD2	–		I/O	VDDIOP	–	
PB10	TXD3	ISI_D8		I/O	VDDIOP	–	
PB11	RXD3	ISI_D9		I/O	VDDIOP	–	
PB12	TXD5	ISI_D10		I/O	VDDIOP	–	
PB13	RXD5	ISI_D11		I/O	VDDIOP	–	
PB14	DRXD	–		I/O	VDDIOP	–	
PB15	DTXD	–		I/O	VDDIOP	–	
PB16	TK0	TCLK3		I/O	VDDIOP	–	
PB17	TF0	TCLK4		I/O	VDDIOP	–	
PB18	TD0	TIOB4		I/O	VDDIOP	–	
PB19	RD0	TIOB5		I/O	VDDIOP	–	
PB20	RK0	ISI_D0		I/O	VDDIOP	–	
PB21	RF0	ISI_D1		I/O	VDDIOP	–	
PB22	DSR0	ISI_D2		I/O	VDDIOP	–	
PB23	DCD0	ISI_D3		I/O	VDDIOP	–	
PB24	DTR0	ISI_D4		I/O	VDDIOP	–	
PB25	RI0	ISI_D5		I/O	VDDIOP	–	
PB26	RTS0	ISI_D6		I/O	VDDIOP	–	
PB27	CTS0	ISI_D7		I/O	VDDIOP	–	
PB28	RTS1	ISI_PCK		I/O	VDDIOP	–	
PB29	CTS1	ISI_VSYNC		I/O	VDDIOP	–	
PB30	PCK0	ISI_HSYNC		I/O	VDDIOP	–	
PB31	PCK1	ISI_MCK		I/O	VDDIOP	–	

© 2017 Microchip Technology Inc.

9.3.3 PIO Controller C Multiplexing

Table 9-4: Multiplexing on PIO Controller C

PIO Controller C					Application Usage		
I/O Line	**Peripheral A**	**Peripheral B**	**Comments**	**Reset State**	**Power Supply**	**Function**	**Comments**
PC0	–	SCK3	AD0	I/O	VDDANA	–	
PC1	–	PCK0	AD1	I/O	VDDANA	–	
PC2	–	PCK1	AD2	I/O	VDDANA	–	
PC3	–	SPI1_NPCS3	AD3	I/O	VDDANA	–	
PC4	A23	SPI1_NPCS2		A23	VDDIOM	–	
PC5	A24	SPI1_NPCS1		A24	VDDIOM	–	
PC6	TIOB2	CFCE1		I/O	VDDIOM	–	
PC7	TIOB1	CFCE2		I/O	VDDIOM	–	
PC8	NCS4/CFCS0	RTS3		I/O	VDDIOM	–	
PC9	NCS5/CFCS1	TIOB0		I/O	VDDIOM	–	
PC10	A25/CFRNW	CTS3		A25	VDDIOM	–	
PC11	NCS2	SPI0_NPCS1		I/O	VDDIOM	–	
PC12	IRQ0	NCS7		I/O	VDDIOM	–	
PC13	FIQ	NCS6		I/O	VDDIOM	–	
PC14	NCS3/NANDCS	IRQ2		I/O	VDDIOM	–	
PC15	NWAIT	IRQ1		I/O	VDDIOM	–	
PC16	D16	SPI0_NPCS2		I/O	VDDIOM	–	
PC17	D17	SPI0_NPCS3		I/O	VDDIOM	–	
PC18	D18	SPI1_NPCS1		I/O	VDDIOM	–	
PC19	D19	SPI1_NPCS2		I/O	VDDIOM	–	
PC20	D20	SPI1_NPCS3		I/O	VDDIOM	–	
PC21	D21	–		I/O	VDDIOM	–	
PC22	D22	TCLK5		I/O	VDDIOM	–	
PC23	D23	–		I/O	VDDIOM	–	
PC24	D24	–		I/O	VDDIOM	–	
PC25	D25	–		I/O	VDDIOM	–	
PC26	D26	–		I/O	VDDIOM	–	
PC27	D27	–		I/O	VDDIOM	–	
PC28	D28	–		I/O	VDDIOM	–	
PC29	D29	–		I/O	VDDIOM	–	
PC30	D30	–		I/O	VDDIOM	–	
PC31	D31	–		I/O	VDDIOM	–	

9.4 Embedded Peripherals

9.4.1 Serial Peripheral Interface

- Supports communication with serial external devices
  - Four chip selects with external decoder support 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 between consecutive transfers and between clock and data per chip select
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
- Very fast transfers supported
  - Transfers with baud rates up to MCK
  - The chip select line may be left active to speed up transfers on the same device

9.4.2 Two-wire Interface

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations
- Supports either master or slave modes
- Compatible with standard two-wire serial memories
- Master, multi-master and slave mode operation
- Bit rate: up to 400 Kbits
- General Call supported in slave mode
- Connection to Peripheral DMA Controller (PDC) capabilities optimizes data transfers in master mode only
  - One channel for the receiver, one channel for the transmitter
  - Next buffer support

9.4.3 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 or 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
  - Optional modem signal management DTR-DSR-DCD-RI
  - Receiver time-out and transmitter timeguard
  - Optional 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

The USART contains features allowing management of the Modem Signals DTR, DSR, DCD and RI. In the SAM9G20, only the USART0 implements these signals, named DTR0, DSR0, DCD0 and RI0.

© 2017 Microchip Technology Inc.

The USART1 and USART2 do not implement all the modem signals. Only RTS and CTS (RTS1 and CTS1, RTS2 and CTS2, respectively) are implemented in these USARTs for other features.

Thus, programming the USART1, USART2 or the USART3 in Modem Mode may lead to unpredictable results. In these USARTs, the commands relating to the Modem Mode have no effect and the status bits relating the status of the modem signals are never activated.

9.4.4 Serial Synchronous Controller

- Provides serial synchronous communication links used in audio and telecom applications (with CODECs in Master or Slave Modes, I^2S, TDM Buses, Magnetic Card Reader, etc.)
- Contains an independent receiver and transmitter and a common clock divider
- Offers a configurable frame sync and data length
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal

9.4.5 Timer Counter

- Two blocks of three 16-bit Timer Counter channels
- Each channel can be individually programmed to perform a 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
  - Two multi-purpose input/output signals
- Each block contains two global registers that act on all three TC Channels

Note: TC Block 0 (TC0, TC1, TC2) and TC Block 1 (TC3, TC4, TC5) have identical user interfaces. See Figure 7-1 for TC Block 0 and TC Block 1 base addresses.

9.4.6 Multimedia Card Interface

- One double-channel MultiMedia Card Interface
- Compatibility with MultiMedia Card Specification Version 3.11
- Compatibility with SD Memory Card Specification Version 1.1
- Compatibility with SDIO Specification Version V1.0.
- Card clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used
- MCI has two slots, each supporting
  - One slot for one MultiMediaCard bus (up to 30 cards) or
  - One SD Memory Card
- Support for stream, block and multi-block data read and write

9.4.7 USB Host Port

- Compliance with Open HCI Rev 1.0 Specification
- Compliance with USB V2.0 Full-speed and Low-speed Specification
- Supports both Low-Speed 1.5 Mbps and Full-speed 12 Mbps devices
- Root hub integrated with two downstream USB ports in the 217-LFBGA package
- Two embedded USB transceivers
- Supports power management
- Operates as a master on the Matrix

9.4.8 USB Device Port

- USB V2.0 full-speed compliant, 12 MBits per second
- Embedded USB V2.0 full-speed transceiver
- Embedded 2,432-byte dual-port RAM for endpoints
- Suspend/Resume logic
- Ping-pong mode (two memory banks) for isochronous and bulk endpoints
- Six general-purpose endpoints
  - Endpoint 0 and 3: 64 bytes, no ping-pong mode
  - Endpoint 1 and 2: 64 bytes, ping-pong mode
  - Endpoint 4 and 5: 512 bytes, ping-pong mode
- Embedded pad pull-up

9.4.9 Ethernet MAC 10/100

- Compatibility with IEEE Standard 802.3
- 10 and 100 MBits per second data throughput capability
- Full- and half-duplex operations
- MII or RMII interface to the physical layer
- Register Interface to address, data, status and control registers
- DMA Interface, operating as a master on the Memory Controller
- Interrupt generation to signal receive and transmit completion
- 28-byte transmit and 28-byte receive FIFOs
- Automatic pad and CRC generation on transmitted frames
- Address checking logic to recognize four 48-bit addresses
- Supports promiscuous mode where all valid frames are copied to memory
- Supports physical layer management through MDIO interface

9.4.10 Image Sensor Interface

- ITU-R BT. 601/656 8-bit mode external interface support
- Support for ITU-R BT.656-4 SAV and EAV synchronization
- Vertical and horizontal resolutions up to 2048 x 2048
- Preview Path up to 640 x 480 in RGMB mode, 2048 x2048 in grayscale mode
- Support for packed data formatting for YCbCr 4:2:2 formats
- Preview scaler to generate smaller size image
- Programmable frame capture rate

9.4.11 Analog-to-Digital Converter

- 4-channel ADC
- 10-bit 312K samples/sec. Successive Approximation Register ADC
- -2/+2 LSB Integral Non Linearity, -1/+1 LSB Differential Non Linearity
- Individual enable and disable of each channel
- External voltage reference for better accuracy on low voltage inputs
- Multiple trigger source – 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
- Four analog inputs shared with digital signals

© 2017 Microchip Technology Inc.

10. Arm926EJ-S Processor Overview

10.1 Overview

The Arm926EJ-S processor is a member of the Arm9™ family of general-purpose microprocessors. The Arm926EJ-S implements Arm architecture version 5TEJ and is targeted at multi-tasking applications where full memory management, high performance, low die size and low power are all important features.

The Arm926EJ-S processor supports the 32-bit Arm and 16-bit THUMB instruction sets, enabling the user to trade off between high performance and high code density. It also supports 8-bit Java instruction set and includes features for efficient execution of Java bytecode, providing a Java performance similar to a JIT (Just-In-Time compilers), for the next generation of Java-powered wireless and embedded devices. It includes an enhanced multiplier design for improved DSP performance.

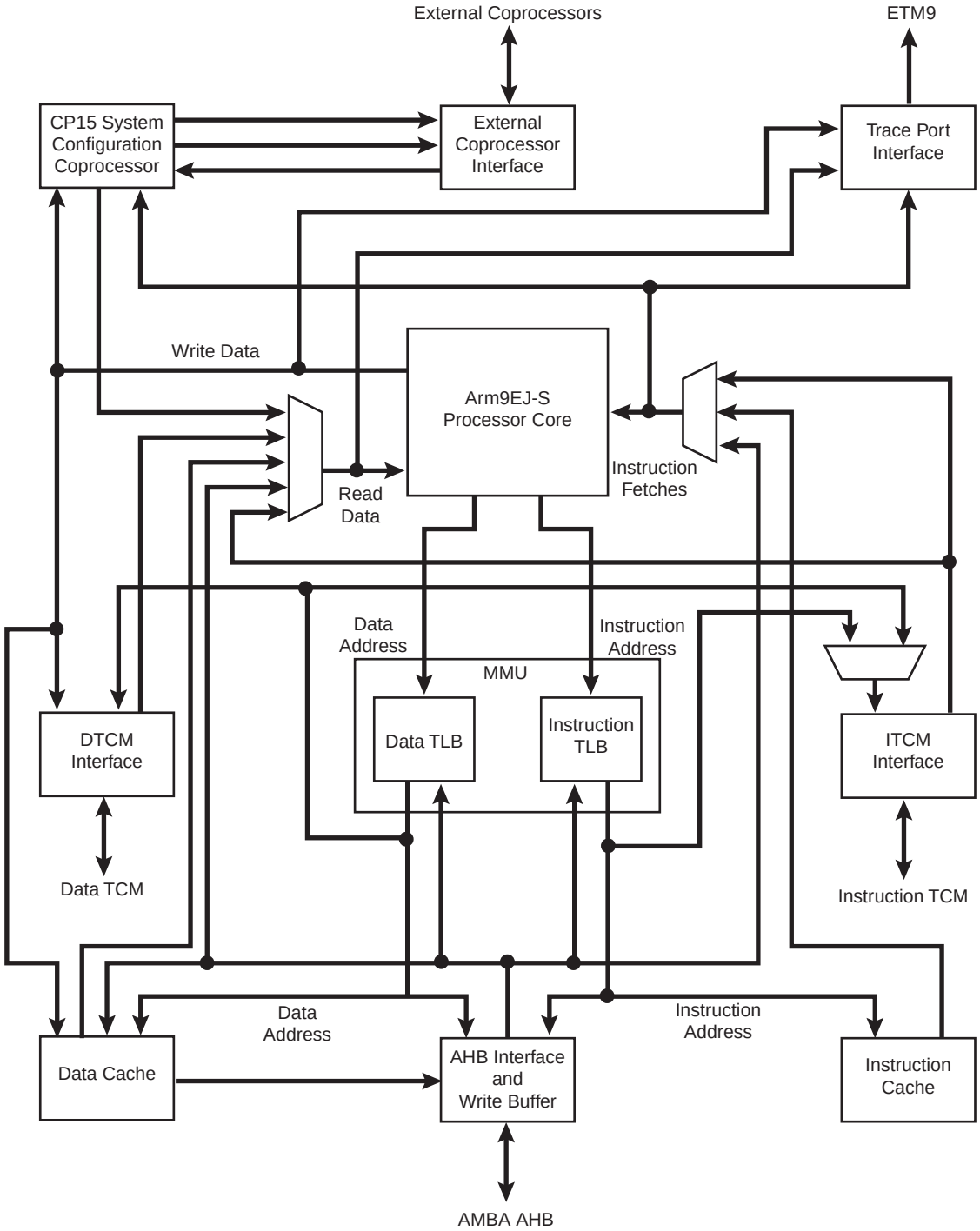
The Arm926EJ-S processor supports the Arm debug architecture and includes logic to assist in both hardware and software debug.

The Arm926EJ-S provides a complete high performance processor subsystem, including:

- an Arm9EJ-S integer core
- a Memory Management Unit (MMU)
- separate instruction and data AMBA AHB bus interfaces
- separate instruction and data TCM interfaces

10.2 Block Diagram

Figure 10-1: Arm926EJ-S Internal Functional Block Diagram



© 2017 Microchip Technology Inc.

10.3 Arm9EJ-S Processor

10.3.1 Arm9EJ-S Operating States

The Arm9EJ-S processor can operate in three different states, each with a specific instruction set:

- Arm state: 32-bit, word-aligned Arm instructions.
- THUMB state: 16-bit, halfword-aligned Thumb instructions.
- Jazelle state: variable length, byte-aligned Jazelle instructions.

In Jazelle state, all instruction Fetches are in words.

10.3.2 Switching State

The operating state of the Arm9EJ-S core can be switched between:

- Arm state and THUMB state using the BX and BLX instructions, and loads to the PC
- Arm state and Jazelle state using the BXJ instruction

All exceptions are entered, handled and exited in Arm state. If an exception occurs in Thumb or Jazelle states, the processor reverts to Arm state. The transition back to Thumb or Jazelle states occurs automatically on return from the exception handler.

10.3.3 Instruction Pipelines

The Arm9EJ-S core uses two kinds of pipelines to increase the speed of the flow of instructions to the processor.

A five-stage (five clock cycles) pipeline is used for Arm and Thumb states. It consists of Fetch, Decode, Execute, Memory and Writeback stages.

A six-stage (six clock cycles) pipeline is used for Jazelle state It consists of Fetch, Jazelle/Decode (two clock cycles), Execute, Memory and Writeback stages.

10.3.4 Memory Access

The Arm9EJ-S core supports byte (8-bit), half-word (16-bit) and word (32-bit) access. Words must be aligned to four-byte boundaries, half-words must be aligned to two-byte boundaries and bytes can be placed on any byte boundary.

Because of the nature of the pipelines, it is possible for a value to be required for use before it has been placed in the register bank by the actions of an earlier instruction. The Arm9EJ-S control logic automatically detects these cases and stalls the core or forward data.

10.3.5 Jazelle Technology

The Jazelle technology enables direct and efficient execution of Java byte codes on Arm processors, providing high performance for the next generation of Java-powered wireless and embedded devices.

The new Java feature of Arm9EJ-S can be described as a hardware emulation of a JVM (Java Virtual Machine). Java mode will appear as another state: instead of executing Arm or Thumb instructions, it executes Java byte codes. The Java byte code decoder logic implemented in Arm9EJ-S decodes 95% of executed byte codes and turns them into Arm instructions without any overhead, while less frequently used byte codes are broken down into optimized sequences of Arm instructions. The hardware/software split is invisible to the programmer, invisible to the application and invisible to the operating system. All existing Arm registers are re-used in Jazelle state and all registers then have particular functions in this mode.

Minimum interrupt latency is maintained across both Arm state and Java state. Since byte codes execution can be restarted, an interrupt automatically triggers the core to switch from Java state to Arm state for the execution of the interrupt handler. This means that no special provision has to be made for handling interrupts while executing byte codes, whether in hardware or in software.

10.3.6 Arm9EJ-S Operating Modes

In all states, there are seven operation modes:

- User mode is the usual Arm program execution state. It is used for executing most application programs
- Fast Interrupt (FIQ) mode is used for handling fast interrupts. It is suitable for high-speed data transfer or channel process
- Interrupt (IRQ) mode is used for general-purpose interrupt handling
- Supervisor mode is a protected mode for the operating system
- Abort mode is entered after a data or instruction prefetch abort
- System mode is a privileged user mode for the operating system
- Undefined mode is entered when an undefined instruction exception occurs

© 2017 Microchip Technology Inc.

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, known as privileged modes, are entered in order to service interrupts or exceptions or to access protected resources.

10.3.7 Arm9EJ-S Registers

The Arm9EJ-S core has a total of 37 registers:

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

Table 10-1 shows all the registers in all modes.

Table 10-1: Arm9TDMI 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

The Arm state register set contains 16 directly-accessible registers, r0 to r15, and an additional register, the Current Program Status Register (CPSR). Registers r0 to r13 are general-purpose registers used to hold either data or address values. Register r14 is used as a Link register that holds a value (return address) of r15 when BL or BLX is executed. Register r15 is used as a program counter (PC), whereas the Current Program Status Register (CPSR) contains condition code flags and the current mode bits.

In privileged modes (FIQ, Supervisor, Abort, IRQ, Undefined), mode-specific banked registers (r8 to r14 in FIQ mode or r13 to r14 in the other modes) become available. The corresponding banked registers r14_fiq, r14_svc, r14_abt, r14_irq, r14_und are similarly used to hold the values (return address for each mode) of r15 (PC) when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines. There is another register called Saved Program Status Register (SPSR) that becomes available in privileged modes instead of CPSR. This register contains condition code flags and the current mode bits saved as a result of the exception that caused entry to the current (privileged) mode.

© 2017 Microchip Technology Inc.

In all modes and due to a software agreement, register r13 is used as stack pointer.

The use and the function of all the registers described above should obey Arm Procedure Call Standard (APCS) which defines:

- constraints on the use of registers
- stack conventions
- argument passing and result return

For more details, refer to Arm Software Development Kit.

The Thumb state register set is a subset of the Arm state set. The programmer has direct access to:

- Eight general-purpose registers r0–r7
- Stack pointer, SP
- Link register, LR (Arm r14)
- PC
- CPSR

There are banked registers SPs, LRs and SPSRs for each privileged mode (for more details see the Arm9EJ-S Technical Reference Manual, revision r1p2 page 2-12).

10.3.7.1 Status Registers

The Arm9EJ-S core contains one CPSR, and five SPSRs for exception handlers to use. The program status registers:

- hold information about the most recently performed ALU operation
- control the enabling and disabling of interrupts
- set the processor operation mode

Figure 10-2: Status Register Format

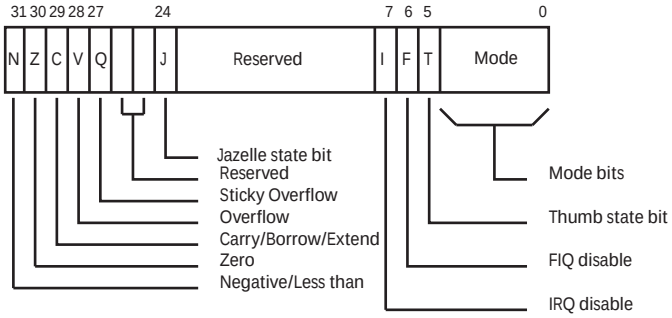


Figure 10-2 shows the status register format, where:

- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- The Sticky Overflow (Q) flag can be set by certain multiply and fractional arithmetic instructions like QADD, QDADD, QSUB, QDSUB, SMLAxy, and SMLAWy needed to achieve DSP operations.
  The Q flag is sticky in that, when set by an instruction, it remains set until explicitly cleared by an MSR instruction writing to the CPSR. Instructions cannot execute conditionally on the status of the Q flag.
- The J bit in the CPSR indicates when the Arm9EJ-S core is in Jazelle state, where:
  - J = 0: The processor is in Arm or Thumb state, depending on the T bit
  - J = 1: The processor is in Jazelle state.
- Mode: five bits to encode the current processor mode

10.3.7.2 Exceptions

Exception Types and Priorities

The Arm9EJ-S supports five types of exceptions. Each type drives the Arm9EJ-S in a privileged mode. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Data and Prefetched aborts (Abort)
- Undefined instruction (Undefined)
- Software interrupt and Reset (Supervisor)

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

More than one exception can happen at a time, therefore the Arm9EJ-S takes the arisen exceptions according to the following priority order:

- Reset (highest priority)
- Data Abort
- FIQ
- IRQ
- Prefetch Abort
- BKPT, Undefined instruction, and Software Interrupt (SWI) (Lowest priority)

The BKPT, or Undefined instruction, and SWI exceptions are mutually exclusive.

Note that there is one exception in the priority scheme: when FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the Arm9EJ-S core enters the Data Abort handler, and proceeds immediately to FIQ vector. A normal return from the FIQ causes the Data Abort handler to resume execution. Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection.

Exception Modes and Handling

Exceptions arise whenever the normal flow of a program must be halted temporarily, for example, to service an interrupt from a peripheral.

When handling an Arm exception, the Arm9EJ-S core performs the following operations:

1. Preserves the address of the next instruction in the appropriate Link Register that corresponds to the new mode that has been entered. When the exception entry is from:
   - Arm and Jazelle states, the Arm9EJ-S copies the address of the next instruction into LR (current PC(r15) + 4 or PC + 8 depending on the exception).
   - THUMB state, the Arm9EJ-S writes the value of the PC into LR, offset by a value (current PC + 2, PC + 4 or PC + 8 depending on the exception) that causes the program to resume from the correct place on return.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value that depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The register r13 is also banked across exception modes to provide each exception handler with private stack pointer.

The Arm9EJ-S can also set the interrupt disable flags to prevent otherwise unmanageable nesting of exceptions.

When an exception has completed, the exception handler must move both the return value in the banked LR minus an offset to the PC and the SPSR to the CPSR. The offset value varies according to the type of exception. This action restores both PC and the CPSR.

The fast interrupt mode has seven private registers r8 to r14 (banked registers) to reduce or remove the requirement for register saving which minimizes the overhead of context switching.

The Prefetch Abort is one of the aborts that indicates that the current memory access cannot be completed. When a Prefetch Abort occurs, the Arm9EJ-S marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the Execute stage in the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

The breakpoint (BKPT) instruction is a new feature of Arm9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.
A breakpoint instruction does not cause the Arm9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

10.3.8 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 (bits[31:28]).

For further details, see the Arm Technical Reference Manual, Arm ref. DDI0198B.

© 2017 Microchip Technology Inc.

Table 10-2 gives the Arm instruction mnemonic list.

Table 10-2: Arm Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMULL	Sign Long Multiply	UMULL	Unsigned Long Multiply
SMLAL	Signed Long Multiply Accumulate	UMLAL	Unsigned Long Multiply Accumulate
MSR	Move to Status Register	MRS	Move From Status Register
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRSH	Load Signed Halfword		
LDRSB	Load Signed Byte		
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRBT	Load Register Byte with Translation	STRBT	Store Register Byte with Translation
LDRT	Load Register with Translation	STRT	Store Register with Translation
LDM	Load Multiple	STM	Store Multiple
SWP	Swap Word	SWPB	Swap Byte
MCR	Move To Coprocessor	MRC	Move From Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor
CDP	Coprocessor Data Processing		

10.3.9 New Arm Instruction Set

Table 10-3: New Arm Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
BXJ	Branch and exchange to Java	MRRC	Move double from coprocessor
BLX (1)	Branch, Link and exchange	MCR2	Alternative move of Arm reg to coprocessor
SMLAxy	Signed Multiply Accumulate 16 * 16 bit	MCRR	Move double to coprocessor
SMLAL	Signed Multiply Accumulate Long	CDP2	Alternative Coprocessor Data Processing
SMLAWy	Signed Multiply Accumulate 32 * 16 bit	BKPT	Breakpoint
SMULxy	Signed Multiply 16 * 16 bit	PLD	Soft Preload, Memory prepare to load from address
SMULWy	Signed Multiply 32 * 16 bit	STRD	Store Double
QADD	Saturated Add	STC2	Alternative Store from Coprocessor
QDADD	Saturated Add with Double	LDRD	Load Double
QSUB	Saturated subtract	LDC2	Alternative Load to Coprocessor
QDSUB	Saturated Subtract with double	CLZ	Count Leading Zeroes

Note 1: A Thumb BLX contains two consecutive Thumb instructions, and takes four cycles.

10.3.10 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

For further details, see the Arm Technical Reference Manual, Arm ref. DDI0198B.

Table 10-4 gives the Thumb instruction mnemonic list.

Table 10-4: 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	BLX	Branch, Link, and Exchange
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt

© 2017 Microchip Technology Inc.

Table 10-4: Thumb Instruction Mnemonic List (Continued)

Mnemonic	Operation	Mnemonic	Operation
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
BCC	Conditional Branch	BKPT	Breakpoint

10.4 CP15 Coprocessor

Coprocessor 15, or System Control Coprocessor CP15, is used to configure and control all the items in the list below:

- Arm9EJ-S
- Caches (ICache, DCache and write buffer)
- TCM
- MMU
- Other system options

To control these features, CP15 provides 16 additional registers. See Table 10-5.

Table 10-5: CP15 Registers

Register	Name	Read/Write
0	ID Code(1)	Read/Unpredictable
0	Cache type(1)	Read/Unpredictable
0	TCM status(1)	Read/Unpredictable
1	Control	Read/Write
2	Translation Table Base	Read/Write
3	Domain Access Control	Read/Write
4	Reserved	None
5	Data fault Status(1)	Read/Write
5	Instruction fault status(1)	Read/Write
6	Fault Address	Read/Write
7	Cache Operations	Read/Write
8	TLB operations	Unpredictable/Write
9	cache lockdown(2)	Read/Write
9	TCM region	Read/Write
10	TLB lockdown	Read/Write
11	Reserved	None
12	Reserved	None
13	FCSE PID(1)	Read/Write

Table 10-5: CP15 Registers (Continued)

Register	Name	Read/Write
13	Context ID(1)	Read/Write
14	Reserved	None
15	Test configuration	Read/Write

Note 1: Register locations 0,5, and 13 each provide access to more than one register. The register accessed depends on the value of the opcode_2 field.

2: Register location 9 provides access to more than one register. The register accessed depends on the value of the CRm field.

10.4.1 CP15 Registers Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from Arm Register) instruction is used to write an Arm register to CP15.
- MRC (Move to Arm Register from Coprocessor) instruction is used to read the value of CP15 to an Arm register.

Other instructions like CDP, LDC, STC can cause an undefined instruction exception.

The assembler code for these instructions is:

```
MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2.
```

The MCR, MRC instructions bit pattern is shown below:

31	30	29	28	27	26	25	24
cond				1	1	1	0

23	22	21	20	19	18	17	16
opcode_1			L	CRn			

15	14	13	12	11	10	9	8
Rd				1	1	1	1

7	6	5	4	3	2	1	0
opcode_2			1	CRm			

CRm[3:0]: Specified Coprocessor Action

Determines specific coprocessor action. Its value is dependent on the CP15 register used. For details, refer to CP15 specific register behavior.

opcode_2[7:5]

Determines specific coprocessor operation code. By default, set to 0.

Rd[15:12]: Arm Register

Defines the Arm register whose value is transferred to the coprocessor. If R15 is chosen, the result is unpredictable.

CRn[19:16]: Coprocessor Register

Determines the destination coprocessor register.

L: Instruction Bit

0: MCR instruction

1: MRC instruction

opcode_1[23:20]: Coprocessor Code

Defines the coprocessor specific code. Value is c15 for CP15.

cond [31:28]: Condition

For more details, see Chapter 2 in Arm926EJ-S TRM.

© 2017 Microchip Technology Inc.

10.5 Memory Management Unit (MMU)

The Arm926EJ-S processor implements an enhanced Arm architecture v5 MMU to provide virtual memory features required by operating systems like Symbian OS®, Windows CE, and Linux. These virtual memory features are memory access permission controls and virtual to physical address translations.

The Virtual Address generated by the CPU core is converted to a Modified Virtual Address (MVA) by the FCSE (Fast Context Switch Extension) using the value in CP15 register13. The MMU translates modified virtual addresses to physical addresses by using a single, two-level page table set stored in physical memory. Each entry in the set contains the access permissions and the physical address that correspond to the virtual address.

The first level translation tables contain 4096 entries indexed by bits [31:20] of the MVA. These entries contain a pointer to either a 1 MB section of physical memory along with attribute information (access permissions, domain, etc.) or an entry in the second level translation tables; coarse table and fine table.

The second level translation tables contain two subtables, coarse table and fine table. An entry in the coarse table contains a pointer to both large pages and small pages along with access permissions. An entry in the fine table contains a pointer to large, small and tiny pages.

Table 7 shows the different attributes of each page in the physical memory.

Table 10-6: Mapping Details

Mapping Name	Mapping Size	Access Permission By	Subpage Size
Section	1 Mbyte	Section	–
Large Page	64 Kbytes	4 separated subpages	16 Kbytes
Small Page	4 Kbytes	4 separated subpages	1 Kbyte
Tiny Page	1 Kbyte	Tiny Page	–

The MMU consists of:

- Access control logic
- Translation Look-aside Buffer (TLB)
- Translation table walk hardware

10.5.1 Access Control Logic

The access control logic controls access information for every entry in the translation table. The access control logic checks two pieces of access information: domain and access permissions. The domain is the primary access control mechanism for a memory region; there are 16 of them. It defines the conditions necessary for an access to proceed. The domain determines whether the access permissions are used to qualify the access or whether they should be ignored.

The second access control mechanism is access permissions that are defined for sections and for large, small and tiny pages. Sections and tiny pages have a single set of access permissions whereas large and small pages can be associated with 4 sets of access permissions, one for each subpage (quarter of a page).

10.5.2 Translation Look-aside Buffer (TLB)

The Translation Look-aside Buffer (TLB) caches translated entries and thus avoids going through the translation process every time. When the TLB contains an entry for the MVA (Modified Virtual Address), the access control logic determines if the access is permitted and outputs the appropriate physical address corresponding to the MVA. If access is not permitted, the MMU signals the CPU core to abort.

If the TLB does not contain an entry for the MVA, the translation table walk hardware is invoked to retrieve the translation information from the translation table in physical memory.

10.5.3 Translation Table Walk Hardware

The translation table walk hardware is a logic that traverses the translation tables located in physical memory, gets the physical address and access permissions and updates the TLB.

The number of stages in the hardware table walking is one or two depending whether the address is marked as a section-mapped access or a page-mapped access.

© 2017 Microchip Technology Inc.

There are three sizes of page-mapped accesses and one size of section-mapped access. Page-mapped accesses are for large pages, small pages and tiny pages. The translation process always begins with a level one fetch. A section-mapped access requires only a level one fetch, but a page-mapped access requires an additional level two fetch. For further details on the MMU, refer to chapter 3 in Arm926EJ-S Technical Reference Manual.

10.5.4 MMU Faults

The MMU generates an abort on the following types of faults:

- Alignment faults (for data accesses only)
- Translation faults
- Domain faults
- Permission faults

The access control mechanism of the MMU detects the conditions that produce these faults. If the fault is a result of memory access, the MMU aborts the access and signals the fault to the CPU core.The MMU retains status and address information about faults generated by the data accesses in the data fault status register and fault address register. It also retains the status of faults generated by instruction fetches in the instruction fault status register.

The fault status register (register 5 in CP15) indicates the cause of a data or prefetch abort, and the domain number of the aborted access when it happens. The fault address register (register 6 in CP15) holds the MVA associated with the access that caused the Data Abort. For further details on MMU faults, refer to chapter 3 in Arm926EJ-S Technical Reference Manual.

10.6 Caches and Write Buffer

The Arm926EJ-S contains a 32 KB Instruction Cache (ICache), a 32 KB Data Cache (DCache), and a write buffer. Although the ICache and DCache share common features, each still has some specific mechanisms.

The caches (ICache and DCache) are four-way set associative, addressed, indexed and tagged using the Modified Virtual Address (MVA), with a cache line length of eight words with two dirty bits for the DCache. The ICache and DCache provide mechanisms for cache lock-down, cache pollution control, and line replacement.

A new feature is now supported by Arm926EJ-S caches called allocate on read-miss commonly known as wrapping. This feature enables the caches to perform critical word first cache refilling. This means that when a request for a word causes a read-miss, the cache performs an AHB access. Instead of loading the whole line (eight words), the cache loads the critical word first, so the processor can reach it quickly, and then the remaining words, no matter where the word is located in the line.

The caches and the write buffer are controlled by the CP15 register 1 (Control), CP15 register 7 (cache operations) and CP15 register 9 (cache lockdown).

10.6.1 Instruction Cache (ICache)

The ICache caches fetched instructions to be executed by the processor. The ICache can be enabled by writing 1 to I bit of the CP15 Register 1 and disabled by writing 0 to this same bit.

When the MMU is enabled, all instruction fetches are subject to translation and permission checks. If the MMU is disabled, all instructions fetches are cachable, no protection checks are made and the physical address is flat-mapped to the modified virtual address. With the MVA use disabled, context switching incurs ICache cleaning and/or invalidating.

When the ICache is disabled, all instruction fetches appear on external memory (AHB) (see Tables 4-1 and 4-2 in page 4-4 in Arm926EJ-S TRM).

On reset, the ICache entries are invalidated and the ICache is disabled. For best performance, ICache should be enabled as soon as possible after reset.

10.6.2 Data Cache (DCache) and Write Buffer

Arm926EJ-S includes a DCache and a write buffer to reduce the effect of main memory bandwidth and latency on data access performance. The operations of DCache and write buffer are closely connected.

10.6.2.1 DCache

The DCache needs the MMU to be enabled. All data accesses are subject to MMU permission and translation checks. Data accesses that are aborted by the MMU do not cause linefills or data accesses to appear on the AMBA ASB interface. If the MMU is disabled, all data accesses are noncachable, nonbufferable, with no protection checks, and appear on the AHB bus. All addresses are flat-mapped, VA = MVA = PA, which incurs DCache cleaning and/or invalidating every time a context switch occurs.

The DCache stores the Physical Address Tag (PA Tag) from which every line was loaded and uses it when writing modified lines back to external memory. This means that the MMU is not involved in write-back operations.

© 2017 Microchip Technology Inc.

Each line (8 words) in the DCache has two dirty bits, one for the first four words and the other one for the second four words. These bits, if set, mark the associated half-lines as dirty. If the cache line is replaced due to a linefill or a cache clean operation, the dirty bits are used to decide whether all, half or none is written back to memory.

DCache can be enabled or disabled by writing either 1 or 0 to bit C in register 1 of CP15 (see Tables 4-3 and 4-4 on page 4-5 in Arm926EJ-S TRM).

The DCache supports write-through and write-back cache operations, selected by memory region using the C and B bits in the MMU translation tables.

The DCache contains an eight data word entry, single address entry write-back buffer used to hold write-back data for cache line eviction or cleaning of dirty cache lines.

The Write Buffer can hold up to 16 words of data and four separate addresses. DCache and Write Buffer operations are closely connected as their configuration is set in each section by the page descriptor in the MMU translation table.

10.6.2.2 Write Buffer

The Arm926EJ-S contains a write buffer that has a 16-word data buffer and a four- address buffer. The write buffer is used for all writes to a bufferable region, write-through region and write-back region. It also allows to avoid stalling the processor when writes to external memory are performed. When a store occurs, data is written to the write buffer at core speed (high speed). The write buffer then completes the store to external memory at bus speed (typically slower than the core speed). During this time, the Arm9EJ-S processor can preform other tasks.

DCache and Write Buffer support write-back and write-through memory regions, controlled by C and B bits in each section and page descriptor within the MMU translation tables.

10.6.2.3 Write-though Operation

When a cache write hit occurs, the DCache line is updated. The updated data is then written to the write buffer which transfers it to external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

10.6.2.4 Write-back Operation

When a cache write hit occurs, the cache line or half line is marked as dirty, meaning that its contents are not up-to-date with those in the external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

10.7 Bus Interface Unit

The Arm926EJ-S features a Bus Interface Unit (BIU) that arbitrates and schedules AHB requests. The BIU implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

The multi-master bus architecture has a number of benefits:

- It allows the development of multi-master systems with an increased bus bandwidth and a flexible architecture.
- Each AHB layer becomes simple because it only has one master, so no arbitration or master-to-slave muxing is required. AHB layers, implementing AHB-Lite protocol, do not have to support request and grant, nor do they have to support retry and split transactions.
- The arbitration becomes effective when more than one master wants to access the same slave simultaneously.

10.7.1 Supported Transfers

The Arm926EJ-S processor performs all AHB accesses as single word, bursts of four words, or bursts of eight words. Any Arm9EJ-S core request that is not 1, 4, 8 words in size is split into packets of these sizes. Note that the Microchip bus is AHB-Lite protocol compliant, hence it does not support split and retry requests.

The following table gives an overview of the supported transfers and different kinds of transactions they are used for.

Table 10-7: Supported Transfers

HBurst[2:0]	Description	
SINGLE	Single transfer	Single transfer of word, half word, or byte: • data write (NCNB, NCB, WT, or WB that has missed in DCache) • data read (NCNB or NCB) • NC instruction fetch (prefetched and non-prefetched) • page table walk read
INCR4	Four-word incrementing burst	Half-line cache write-back, Instruction prefetch, if enabled. Four-word burst NCNB, NCB, WT, or WB write.
INCR8	Eight-word incrementing burst	Full-line cache write-back, eight-word burst NCNB, NCB, WT, or WB write.
WRAP8	Eight-word wrapping burst	Cache linefill

10.7.2 Thumb Instruction Fetches

All instructions fetches, regardless of the state of Arm9EJ-S core, are made as 32-bit accesses on the AHB. If the Arm9EJ-S is in Thumb state, then two instructions can be fetched at a time.

10.7.3 Address Alignment

The Arm926EJ-S BIU performs address alignment checking and aligns AHB addresses to the necessary boundary. 16-bit accesses are aligned to halfword boundaries, and 32-bit accesses are aligned to word boundaries.

© 2017 Microchip Technology Inc.

11. Debug and Test

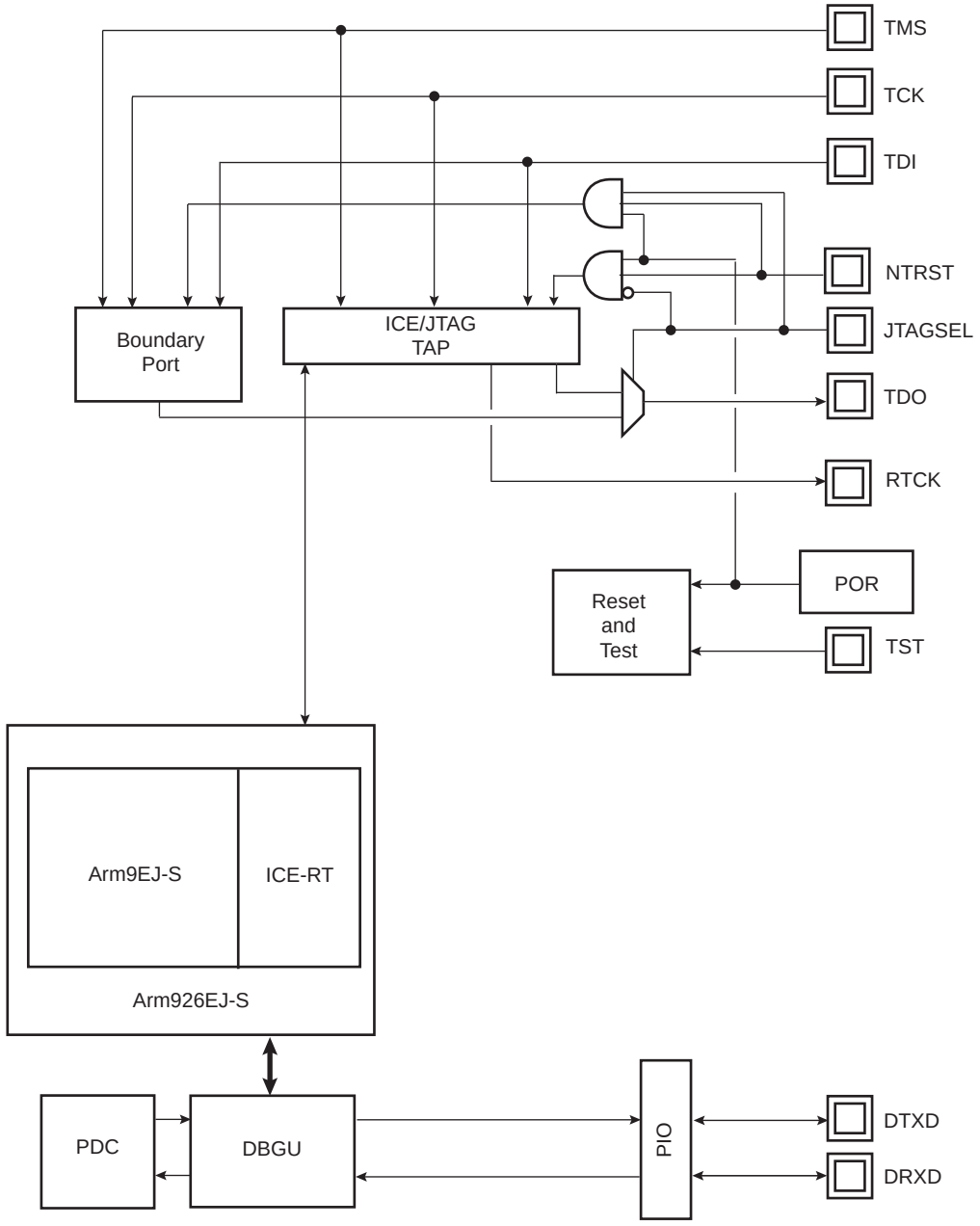
11.1 Overview

The SAM9G20 features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) 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.

11.2 Block Diagram

Figure 11-1: Debug and Test Block Diagram



TAP: Test Access Port

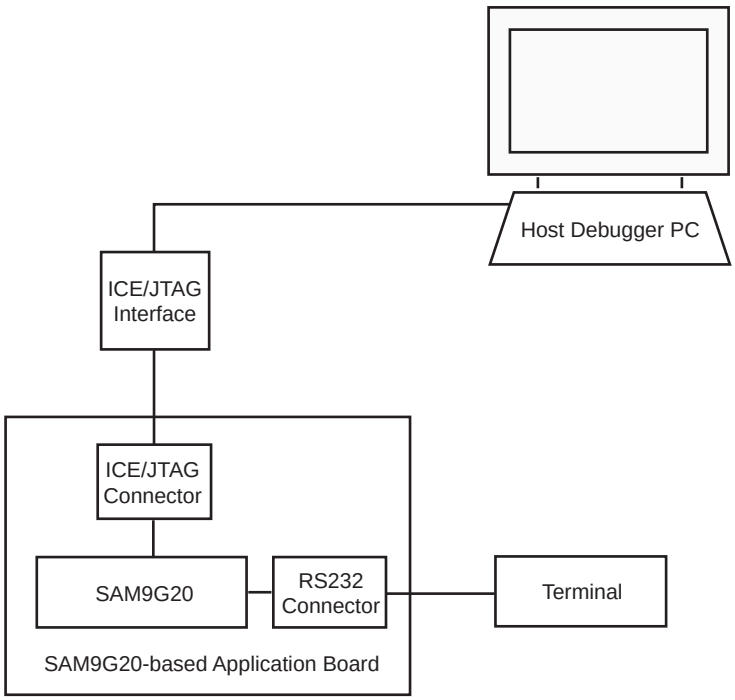
© 2017 Microchip Technology Inc.

11.3 Application Examples

11.3.1 Debug Environment

Figure 11-2 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. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

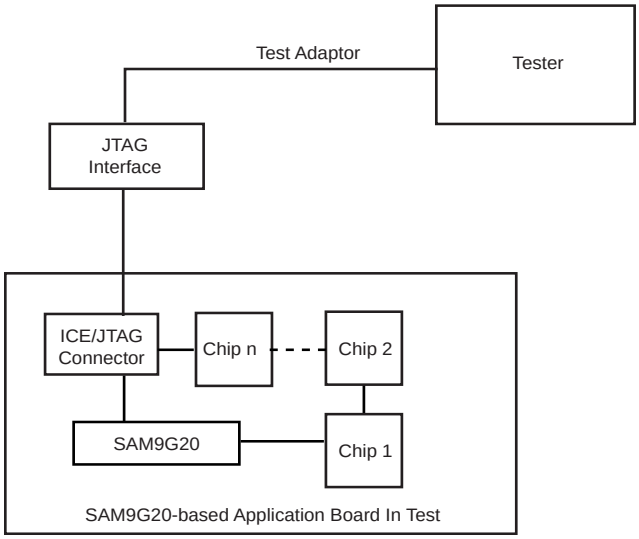
Figure 11-2: Application Debug and Trace Environment Example



11.3.2 Test Environment

The figure below 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 11-3: Application Test Environment Example



11.4 Debug and Test Pin Description

Table 11-1: Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microprocessor Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
NTRST	Test Reset Signal	Input	Low
TCK	Test Clock	Input	–
TDI	Test Data In	Input	–
TDO	Test Data Out	Output	–
TMS	Test Mode Select	Input	–
RTCK	Returned Test Clock	Output	–
JTAGSEL	JTAG Selection	Input	–
Debug Unit			
DRXD	Debug Receive Data	Input	–
DTXD	Debug Transmit Data	Output	–

11.5 Functional Description

11.5.1 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.

11.5.2 EmbeddedICE

The Arm9EJ-S EmbeddedICE-RT™ is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an Arm9EJ-S core embedded within the Arm926EJ-S. The internal state of the Arm926EJ-S is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the Arm9EJ-S registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the Arm9EJ-S processor which support testing, debugging, and programming of the EmbeddedICE-RT. The scan chains are controlled by the ICE/JTAG port.

EmbeddedICE 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 EmbeddedICE-RT, see the Arm document:

Arm9EJ-S Technical Reference Manual (DDI 0222A).

11.5.3 JTAG Signal Description

TMS is the Test Mode Select input which controls the transitions of the test interface state machine.

TDI is the Test Data Input line which supplies the data to the JTAG registers (Boundary Scan Register, Instruction Register, or other data registers).

TDO is the Test Data Output line which is used to serially output the data from the JTAG registers to the equipment controlling the test. It carries the sampled values from the boundary scan chain (or other JTAG registers) and propagates them to the next chip in the serial test circuit.

© 2017 Microchip Technology Inc.

NTRST (optional in IEEE Standard 1149.1) is a Test-Reset input which is mandatory in Arm cores and used to reset the debug logic. On Microchip Arm926EJ-S-based cores, NTRST is a Power On Reset output. It is asserted on power on. If necessary, the user can also reset the debug logic with the NTRST pin assertion during 2.5 MCK periods.

TCK is the Test ClocK input which enables the test interface. TCK is pulsed by the equipment controlling the test and not by the tested device. It can be pulsed at any frequency. Note the maximum JTAG clock rate on Arm926EJ-S cores is 1/6th the clock of the CPU. This gives 5.45 kHz maximum initial JTAG clock rate for an Arm9E™ running from the 32.768 kHz slow clock.

RTCK is the Return Test Clock. Not an IEEE Standard 1149.1 signal added for a better clock handling by emulators. From some ICE Interface probes, this return signal can be used to synchronize the TCK clock and take not care about the given ratio between the ICE Interface clock and system clock equal to 1/6th. This signal is only available in JTAG ICE Mode and not in boundary scan mode.

11.5.4 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.

The SAM9G20 Debug Unit Chip ID value is 0x0199 05A1 on 32-bit width.

For further details on the Debug Unit, see Section 27. “Debug Unit (DBGU)”.

11.5.5 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 JTAGSEL is high. 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.

11.5.5.1 JTAG Boundary-scan Register

The Boundary-scan Register (BSR) contains 308 bits that correspond to active pins and associated control signals.

Each SAM9G20 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.

Table 11-2: SAM9G20 JTAG Boundary Scan Register

Bit Number	Pin Name	Pin Type	Associated BSR Cells
307	A0	IN/OUT	CONTROL
306			INPUT/OUTPUT
305	A1	IN/OUT	CONTROL
304			INPUT/OUTPUT
303	A10	IN/OUT	CONTROL
302			INPUT/OUTPUT
301	A11	IN/OUT	CONTROL
300			INPUT/OUTPUT
299	A12	IN/OUT	CONTROL
298			INPUT/OUTPUT
297	A13	IN/OUT	CONTROL
296			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
295	A14	IN/OUT	CONTROL
294			INPUT/OUTPUT
293	A15	IN/OUT	CONTROL
292			INPUT/OUTPUT
291	A16	IN/OUT	CONTROL
290			INPUT/OUTPUT
289	A17	IN/OUT	CONTROL
288			INPUT/OUTPUT
287	A18	IN/OUT	CONTROL
286			INPUT/OUTPUT
285	A19	IN/OUT	CONTROL
284			INPUT/OUTPUT
283	A2	IN/OUT	CONTROL
282			INPUT/OUTPUT
281	A20	IN/OUT	CONTROL
280			INPUT/OUTPUT
279	A21	IN/OUT	CONTROL
278			INPUT/OUTPUT
277	A22	IN/OUT	CONTROL
276			INPUT/OUTPUT
275	A3	IN/OUT	CONTROL
274			INPUT/OUTPUT
273	A4	IN/OUT	CONTROL
272			INPUT/OUTPUT
271	A5	IN/OUT	CONTROL
270			INPUT/OUTPUT
269	A6	IN/OUT	CONTROL
268			INPUT/OUTPUT
267	A7	IN/OUT	CONTROL
266			INPUT/OUTPUT
265	A8	IN/OUT	CONTROL
264			INPUT/OUTPUT
263	A9	IN/OUT	CONTROL
262			INPUT/OUTPUT
261	BMS	INPUT	INPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
260	CAS	IN/OUT	CONTROL
259			INPUT/OUTPUT
258	D0	IN/OUT	CONTROL
257			INPUT/OUTPUT
256	D1	IN/OUT	CONTROL
255			INPUT/OUTPUT
254	D10	IN/OUT	CONTROL
253			INPUT/OUTPUT
252	D11	IN/OUT	CONTROL
251			INPUT/OUTPUT
250	D12	IN/OUT	CONTROL
249			INPUT/OUTPUT
248	D13	IN/OUT	CONTROL
247			INPUT/OUTPUT
246	D14	IN/OUT	CONTROL
245			INPUT/OUTPUT
244	D15	IN/OUT	CONTROL
243			INPUT/OUTPUT
242	D2	IN/OUT	CONTROL
241			INPUT/OUTPUT
240	D3	IN/OUT	CONTROL
239			INPUT/OUTPUT
238	D4	IN/OUT	CONTROL
237			INPUT/OUTPUT
236	D5	IN/OUT	CONTROL
235			INPUT/OUTPUT
234	D6	IN/OUT	CONTROL
233			INPUT/OUTPUT
232	D7	IN/OUT	CONTROL
231			INPUT/OUTPUT
230	D8	IN/OUT	CONTROL
229			INPUT/OUTPUT
228	D9	IN/OUT	CONTROL
227			INPUT/OUTPUT
226	NANDOE	IN/OUT	CONTROL
225			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
224	NANDWE	IN/OUT	CONTROL
223			INPUT/OUTPUT
222	NCS0	IN/OUT	CONTROL
221			INPUT/OUTPUT
220	NCS1	IN/OUT	CONTROL
219			INPUT/OUTPUT
218	NRD	IN/OUT	CONTROL
217			INPUT/OUTPUT
216	NRST	IN/OUT	CONTROL
215			INPUT/OUTPUT
214	NWR0	IN/OUT	CONTROL
213			INPUT/OUTPUT
212	NWR1	IN/OUT	CONTROL
211			INPUT/OUTPUT
210	NWR3	IN/OUT	CONTROL
209			INPUT/OUTPUT
208	OSCSEL	INPUT	INPUT
207	PA0	IN/OUT	CONTROL
206			INPUT/OUTPUT
205	PA1	IN/OUT	CONTROL
204			INPUT/OUTPUT
203	PA10	IN/OUT	CONTROL
202			INPUT/OUTPUT
201	PA11	IN/OUT	CONTROL
200			INPUT/OUTPUT
199	PA12	IN/OUT	CONTROL
198			INPUT/OUTPUT
197	PA13	IN/OUT	CONTROL
196			INPUT/OUTPUT
195	PA14	IN/OUT	CONTROL
194			INPUT/OUTPUT
193	PA15	IN/OUT	CONTROL
192			INPUT/OUTPUT
191	PA16	IN/OUT	CONTROL
190			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
189	PA17	IN/OUT	CONTROL
188			INPUT/OUTPUT
187	PA18	IN/OUT	CONTROL
186			INPUT/OUTPUT
185	PA19	IN/OUT	CONTROL
184			INPUT/OUTPUT
183	PA2	IN/OUT	CONTROL
182			INPUT/OUTPUT
181	PA20	IN/OUT	CONTROL
180			INPUT/OUTPUT
179	PA21	IN/OUT	CONTROL
178			INPUT/OUTPUT
177	PA22	IN/OUT	CONTROL
176			INPUT/OUTPUT
175	PA23	IN/OUT	CONTROL
174			INPUT/OUTPUT
173	PA24	IN/OUT	CONTROL
172			INPUT/OUTPUT
171	PA25	IN/OUT	CONTROL
170			INPUT/OUTPUT
169	PA26	IN/OUT	CONTROL
168			INPUT/OUTPUT
167	PA27	IN/OUT	CONTROL
166			INPUT/OUTPUT
165	PA28	IN/OUT	CONTROL
164			INPUT/OUTPUT
163	PA29	IN/OUT	CONTROL
162			INPUT/OUTPUT
161	PA3	IN/OUT	CONTROL
160			INPUT/OUTPUT
159	–	internal	–
158	–	internal	–
157	–	internal	–
156	–	internal	–
155	PA4	IN/OUT	CONTROL
154			INPUT/OUTPUT

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
153	PA5	IN/OUT	CONTROL
152			INPUT/OUTPUT
151	PA6	IN/OUT	CONTROL
150			INPUT/OUTPUT
149	PA7	IN/OUT	CONTROL
148			INPUT/OUTPUT
147	PA8	IN/OUT	CONTROL
146			INPUT/OUTPUT
145	PA9	IN/OUT	CONTROL
144			INPUT/OUTPUT
143	PB0	IN/OUT	CONTROL
142			INPUT/OUTPUT
141	PB1	IN/OUT	CONTROL
140			INPUT/OUTPUT
139	PB10	IN/OUT	CONTROL
138			INPUT/OUTPUT
137	PB11	IN/OUT	CONTROL
136			INPUT/OUTPUT
135	–	internal	–
134	–	internal	–
133	–	internal	–
132	–	internal	–
131	PB14	IN/OUT	CONTROL
130			INPUT/OUTPUT
129	PB15	IN/OUT	CONTROL
128			INPUT/OUTPUT
127	PB16	IN/OUT	CONTROL
126			INPUT/OUTPUT
125	PB17	IN/OUT	CONTROL
124			INPUT/OUTPUT
123	PB18	IN/OUT	CONTROL
122			INPUT/OUTPUT
121	PB19	IN/OUT	CONTROL
120			INPUT/OUTPUT
119	PB2	IN/OUT	CONTROL
118			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
117	PB20	IN/OUT	CONTROL
116			INPUT/OUTPUT
115	PB21	IN/OUT	CONTROL
114			INPUT/OUTPUT
113	PB22	IN/OUT	CONTROL
112			INPUT/OUTPUT
111	PB23	IN/OUT	CONTROL
110			INPUT/OUTPUT
109	PB24	IN/OUT	CONTROL
108			INPUT/OUTPUT
107	PB25	IN/OUT	CONTROL
106			INPUT/OUTPUT
105	PB26	IN/OUT	CONTROL
104			INPUT/OUTPUT
103	PB27	IN/OUT	CONTROL
102			INPUT/OUTPUT
101	PB28	IN/OUT	CONTROL
100			INPUT/OUTPUT
99	PB29	IN/OUT	CONTROL
98			INPUT/OUTPUT
97	PB3	IN/OUT	CONTROL
96			INPUT/OUTPUT
95	PB30	IN/OUT	CONTROL
94			INPUT/OUTPUT
93	PB31	IN/OUT	CONTROL
92			INPUT/OUTPUT
91	PB4	IN/OUT	CONTROL
90			INPUT/OUTPUT
89	PB5	IN/OUT	CONTROL
88			INPUT/OUTPUT
87	PB6	IN/OUT	CONTROL
86			INPUT/OUTPUT
85	PB7	IN/OUT	CONTROL
84			INPUT/OUTPUT
83	PB8	IN/OUT	CONTROL
82			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
81	PB9	IN/OUT	CONTROL
80			INPUT/OUTPUT
79	PC0	IN/OUT	CONTROL
78			INPUT/OUTPUT
77	PC1	IN/OUT	CONTROL
76			INPUT/OUTPUT
75	PC10	IN/OUT	CONTROL
74			INPUT/OUTPUT
73	PC11	IN/OUT	CONTROL
72			INPUT/OUTPUT
71	–	internal	–
70	–	internal	–
69	PC13	IN/OUT	CONTROL
68			INPUT/OUTPUT
67	PC14	IN/OUT	CONTROL
66			INPUT/OUTPUT
65	PC15	IN/OUT	CONTROL
64			INPUT/OUTPUT
63	PC16	IN/OUT	CONTROL
62			INPUT/OUTPUT
61	PC17	IN/OUT	CONTROL
60			INPUT/OUTPUT
59	PC18	IN/OUT	CONTROL
58			INPUT/OUTPUT
57	PC19	IN/OUT	CONTROL
56			INPUT/OUTPUT
55	–	internal	–
54	–	internal	–
53	PC20	IN/OUT	CONTROL
52			INPUT/OUTPUT
51	PC21	IN/OUT	CONTROL
50			INPUT/OUTPUT
49	PC22	IN/OUT	CONTROL
48			INPUT/OUTPUT
47	PC23	IN/OUT	CONTROL
46			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
45	PC24	IN/OUT	CONTROL
44			INPUT/OUTPUT
43	PC25	IN/OUT	CONTROL
42			INPUT/OUTPUT
41	PC26	IN/OUT	CONTROL
40			INPUT/OUTPUT
39	PC27	IN/OUT	CONTROL
38			INPUT/OUTPUT
37	PC28	IN/OUT	CONTROL
36			INPUT/OUTPUT
35	PC29	IN/OUT	CONTROL
34			INPUT/OUTPUT
33	–	internal	–
32	–	internal	–
31	PC30	IN/OUT	CONTROL
30			INPUT/OUTPUT
29	PC31	IN/OUT	CONTROL
28			INPUT/OUTPUT
27	PC4	IN/OUT	CONTROL
26			INPUT/OUTPUT
25	PC5	IN/OUT	CONTROL
24			INPUT/OUTPUT
23	PC6	IN/OUT	CONTROL
22			INPUT/OUTPUT
21	PC7	IN/OUT	CONTROL
20			INPUT/OUTPUT
19	PC8	IN/OUT	CONTROL
18			INPUT/OUTPUT
17	PC9	IN/OUT	CONTROL
16			INPUT/OUTPUT
15	RAS	IN/OUT	CONTROL
14			INPUT/OUTPUT
13	RTCK	OUT	CONTROL
12			OUTPUT
11	SDA10	IN/OUT	CONTROL
10			INPUT/OUTPUT

© 2017 Microchip Technology Inc.

Table 11-2: SAM9G20 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
09	SDCK	IN/OUT	CONTROL
08			INPUT/OUTPUT
07	SDCKE	IN/OUT	CONTROL
06			INPUT/OUTPUT
05	SDWE	IN/OUT	CONTROL
04			INPUT/OUTPUT
03	SHDN	OUT	CONTROL
02			OUTPUT
01	TST	INPUT	INPUT
00	WKUP	INPUT	INPUT

© 2017 Microchip Technology Inc.

11.5.6 JID 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

Product part Number is 0x5B24

MANUFACTURER IDENTITY[11:1]

Set to 0x01F.

Bit[0] required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B2_403F.

12. Boot Program

12.1 Overview

The Boot Program integrates different programs that manage download and/or upload into the different memories of the product.

First, it initializes the Debug Unit serial port (DBGU) and the USB High Speed Device Port.

The Boot program tries to detect SPI flash memories. The Serial Flash Boot program and DataFlash Boot program are executed. It looks for a sequence of seven valid Arm exception vectors in a Serial Flash or DataFlash connected to the SPI. All these vectors must be B-branch or LDR load register instructions except for the sixth vector. This vector is used to store the size of the image to download.

If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no valid Arm vector sequence is found, NAND Flash Boot program is then executed. The NAND Flash Boot program looks for a sequence of seven valid Arm exception vectors. If such a sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no valid Arm exception vector is found, the SDCard Boot program is then executed. It looks for a boot.bin file in the root directory of a FAT12/16/32 formatted SDCard on Slot A. If such a file is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If the SDCard is not formatted or if boot.bin file is not found, TWI Boot program is then executed. The TWI Boot program searches for a valid application in an EEPROM memory. If such a file is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

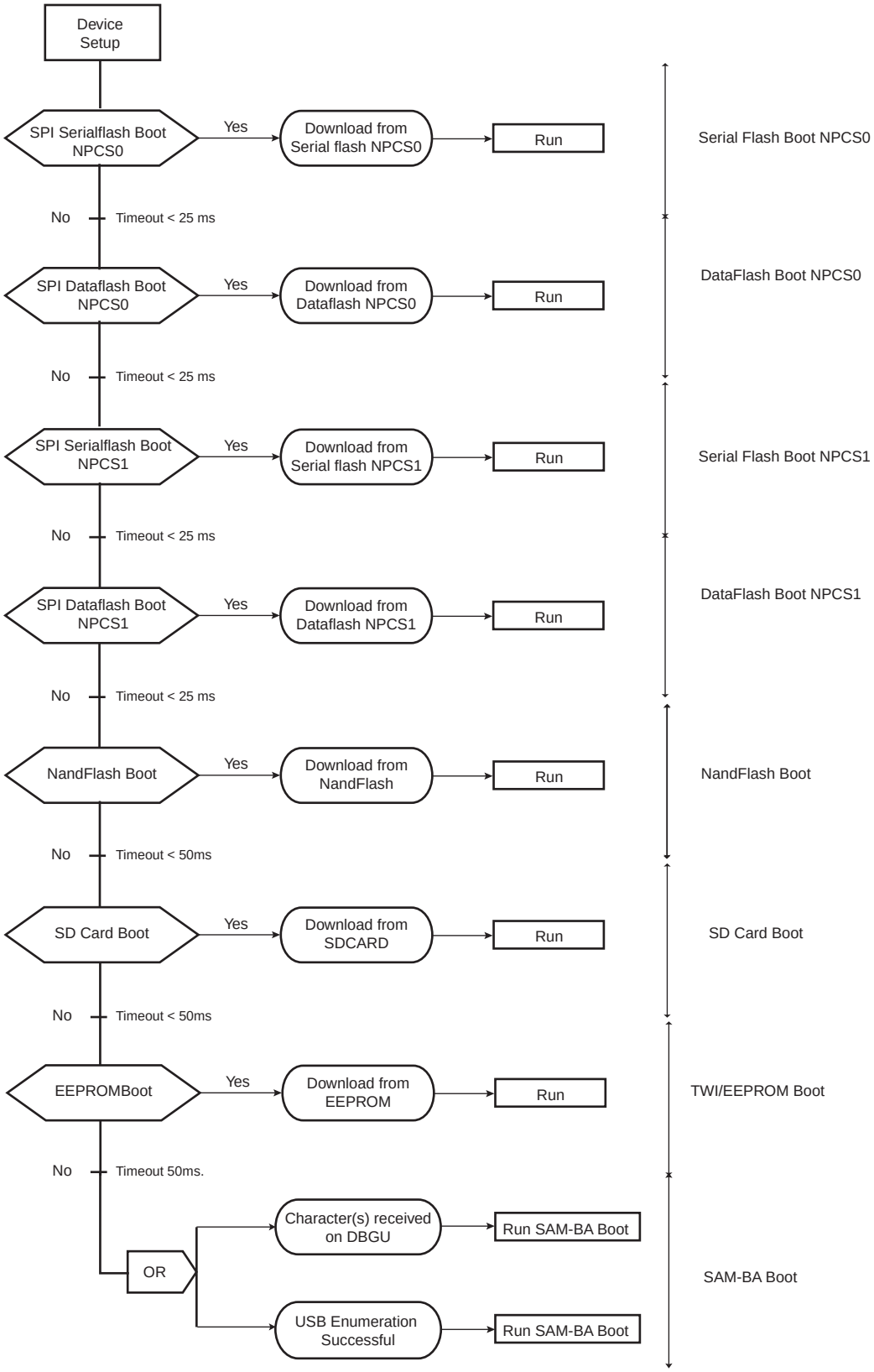
If no valid application is found, SAM-BA Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

12.2 Flow Diagram

The Boot Program implements the algorithm in Figure 12-1.

© 2017 Microchip Technology Inc.

Figure 12-1: Boot Program Algorithm Flow Diagram



12.3 Device Initialization

Initialization follows the steps described below:

1. Stack setup for Arm supervisor mode
2. Main Oscillator Frequency Detection
3. C variable initialization
4. PLL setup: PLLB is initialized to generate a 48 MHz clock necessary to use the USB Device. A register located in the Power Management Controller (PMC) determines the frequency of the main oscillator and thus the correct factor for the PLLB.

- If internal RC Oscillator is used (OSCSEL = 0) and Main Oscillator is active, TTable 12-1 defines the crystals supported by the Boot Program when using the internal RC oscillator. also supported by the Boot Program.

Table 12-1: Crystals Supported by Software Auto-Detection (MHz)

	3.0	8.0	18.432	Other
Boot in DBGU	Yes	Yes	Yes	Yes
Boot on USB	Yes	Yes	Yes	No

Note: Any other crystal can be used but it prevents using the USB.

- If internal RC Oscillator is used (OSCSEL = 0) and Main Oscillator is bypassed, Table 12-2 defines the frequencies supported by the Boot Program when bypassing the main oscillator.

Table 12-2: Crystals Supported by Software Auto-Detection (MHz)

	3.0	8.0	20	50	Other
Boot in DBGU	Yes	Yes	Yes	Yes	Yes
Boot on USB	Yes	Yes	Yes	Yes	No

Note: Any other crystal can be used but it prevents using the USB.

- If an external 32768 Hz Oscillator is used (OSCSEL = 1), defines the crystals supported by the Boot Program. Table 12-3 defines the crystals supported by the Boot Program.

Table 12-3: Crystals Supported by Software Auto-Detection (MHz)

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.608	4.9152	5.0	5.24288
6.0	6.144	6.4	6.5536	7.159090
7.3728	7.864320	8.0	9.8304	10.0
11.05920	12.0	12.288	13.56	14.31818
14.7456	16.0	17.734470	18.432	20.0

Note: Booting either on USB or on DBGU is possible with any of these input frequencies.

- If an external 32768 Hz Oscillator is used (OSCSEL = 1) and Main Oscillator is bypassed. Table 12-4 defines the crystals supported by the Boot Program.

Table 12-4: Input Frequencies Supported (OSCEL = 1)

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.608	4.9152	5.0	5.24288
6.0	6.144	6.4	6.5536	7.159090
7.3728	7.864320	8.0	9.8304	10.0

© 2017 Microchip Technology Inc.

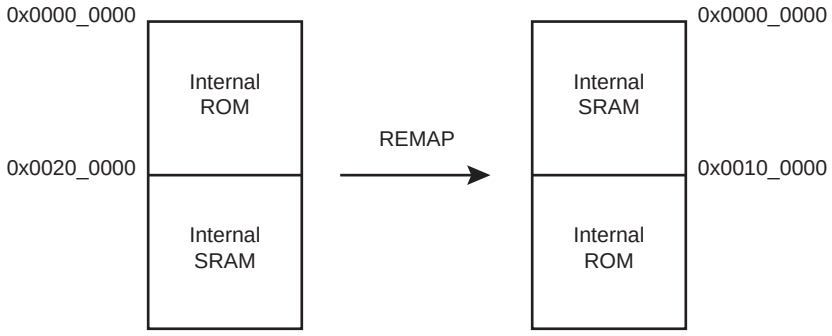
Table 12-4: Input Frequencies Supported (OSCEL = 1)

11.05920	12.0	12.288	13.56	14.31818
14.7456	16.0	17.734470	18.432	20.0
24.0	24.576	25.0	28.224	32.0
33.0	40.0	48.0	50	

Note: Booting either on USB or on DBGU is possible with any of these input frequencies.

5. Initialization of the DBGU serial port (115200 bauds, 8, N, 1)
6. Jump to Serial Flash Boot sequence through NPCS0. If Serial Flash Boot succeeds, perform a remap and jump to 0x0.
7. Jump to DataFlash Boot sequence through NPCS0. If DataFlash Boot succeeds, perform a remap and jump to 0x0.
8. Jump to Serial Flash Boot sequence through NPCS1. If Serial Flash Boot succeeds, perform a remap and jump to 0x0.
9. Jump to DataFlash Boot sequence through NPCS1. If DataFlash Boot succeeds, perform a remap and jump to 0x0.
10. Jump to NAND Flash Boot sequence. If NAND Flash Boot succeeds, perform a remap and jump to 0x0.
11. Jump to SDCard Boot sequence. If SDCard Boot succeeds, perform a remap and jump to 0x0.
12. Jump to EEPROM Boot sequence. If EEPROM Boot succeeds, perform a remap and jump to 0x0.
13. Activation of the Instruction Cache
14. Jump to SAM-BA Boot sequence
15. Disable the WatchDog
16. Initialization of the USB Device Port

Figure 12-2: Remap Action after Download Completion



12.4 Valid Image Detection

The DataFlash Boot software looks for a valid application by analyzing the first 28 bytes corresponding to the Arm exception vectors. These bytes must implement Arm instructions for either branch or load PC with PC relative addressing.

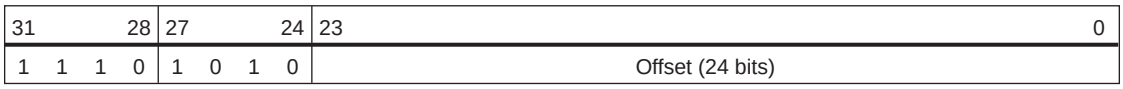
The sixth vector, at offset 0x14, contains the size of the image to download. The user must replace this vector with his/her own vector (see “Structure of Arm Vector 6”).

12.4.1 Valid Arm exception vectors

Figure 12-3: LDR Opcode

31 28	27 24	23 20	19 16	15 12	11 0
1 1 1 0	0 1 I P	U 0 W 1	Rn	Rd	

Figure 12-4: B Opcode



Unconditional instruction: 0xE for bits 31 to 28

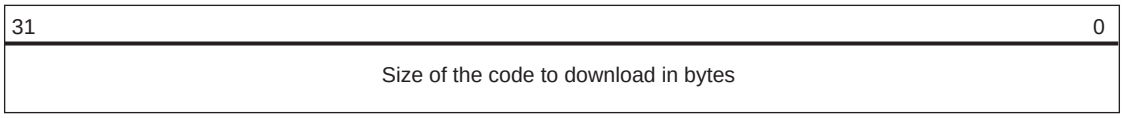
Load PC with PC relative addressing instruction:

- Rn = Rd = PC = 0xF
- I==0
- P==1
- U offset added (U==1) or subtracted (U==0)
- W==1

12.4.2 Structure of Arm Vector 6

The Arm exception vector 6 is used to store information needed by the DataFlash boot program. This information is described below.

Figure 12-5: Structure of the Arm Vector 6



12.4.2.1 Example

An example of valid vectors follows:

```
00          ea000006                B               0x20
04          eafffffe                B               0x04
08          ea00002f                B               _main
0c          eafffffe                B               0x0c
10          eafffffe                B               0x10
14          00001234                B               0x14            <- Code size = 4660 bytes
18          eafffffe                B               0x18
```

The size of the image to load into SRAM is contained in the location of the sixth Arm vector. Thus the user must replace this vector by the correct size of his/her application.

12.5 Serial Flash Boot

The Serial Flash boot looks for a valid application in the SPI Serial Flash memory.

SPI0 is configured in master mode to generate a SPCK at 1 MHz. Serial Flash shall be connected to NPCS0 or NPCS1.

The Serial Flash boot reads the Serial Flash status register (Instruction code 0x05). The Serial Flash is considered as ready if bit 0 of the returned status register is cleared.

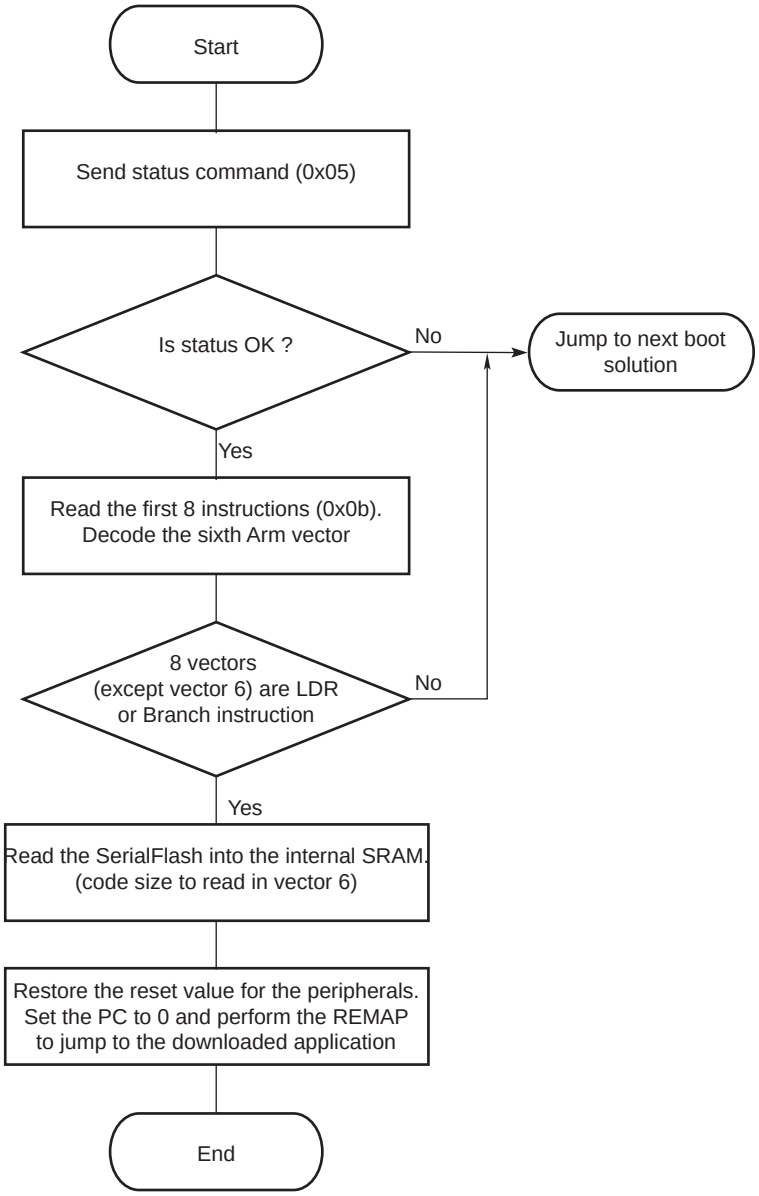
If no Serial Flash is connected or if it does not answer, Serial Flash boot exits after 1000 attempts.

If the Serial Flash is ready, Serial Flash boot reads the first 8 words into SRAM (Instruction code “Continuos read array” 0x0b) and checks if it corresponds to valid exception vectors according to the Valid Image detection algorithm.

If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

© 2017 Microchip Technology Inc.

Figure 12-6: Serial Flash Download



12.6 DataFlash Boot Sequence

The DataFlash boot looks for a valid application in the SPI DataFlash memory.

SPI0 is configured in master mode to generate a SPCK at 1 MHz. DataFlash shall be connected to NPCS0 or NPCS1.

The DataFlash boot reads the DataFlash flash status register (Instruction code 0xD7). The DataFlash is considered as ready if bit 7 of the returned status register is set.

If no DataFlash is connected or if it does not answer, DataFlash boot exits after 1000 attempts.

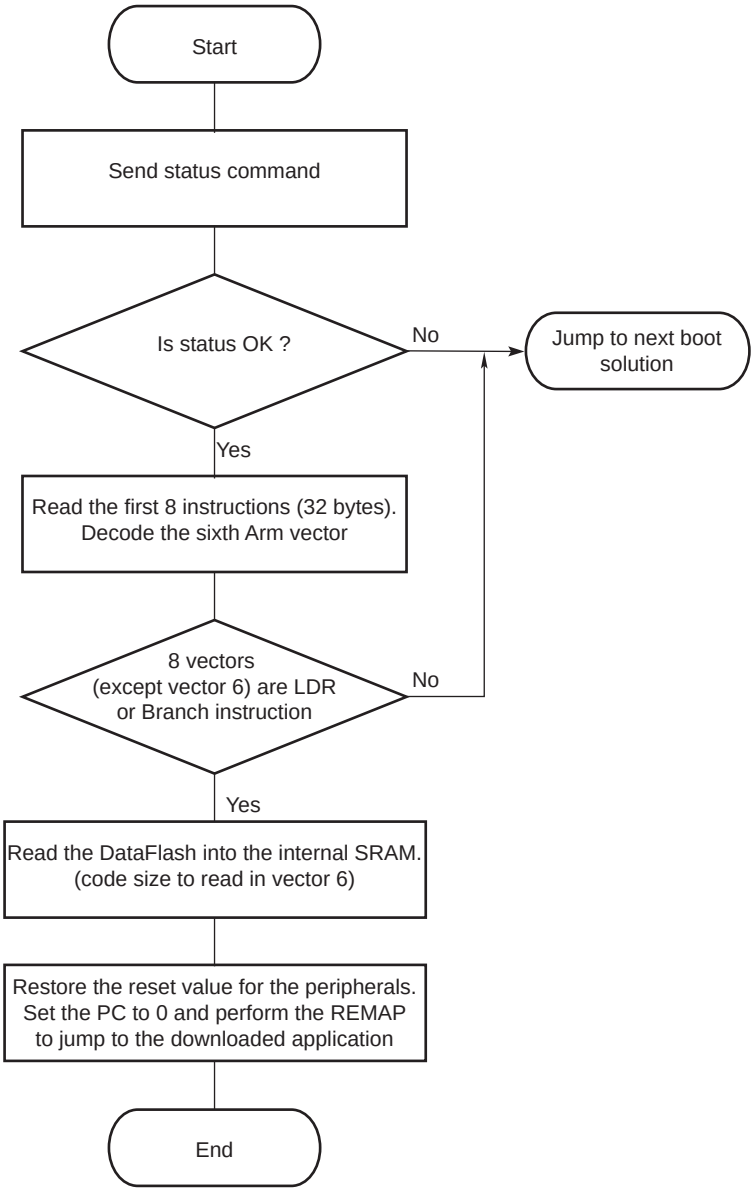
If the DataFlash is ready, DataFlash boot reads the first 8 words into SRAM (Instruction code “Continuous Read Array” 0x0b) and checks if it corresponds to valid exception vectors according to the Valid Image detection algorithm.

If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

The DataFlash boot is configured to be compatible with the future design of the DataFlash.

© 2017 Microchip Technology Inc.

Figure 12-7: Serial DataFlash Download



12.7 NAND Flash Boot

The NAND Flash Boot program searches for a valid application in the NAND Flash memory. The first block must be guaranteed by the manufacturer. There is no ECC check.

If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. See “Valid Image Detection” for more information on Valid Image Detection.

12.7.1 Supported NAND Flash Devices

Any 8- or 16-bit NAND Flash Device.

© 2017 Microchip Technology Inc.

12.8 SDCard Boot

The SDCard Boot program searches for a valid application in the SD Card memory.

(Boot ROM does not support high capacity SDCards.)

It looks for a boot.bin file in the root directory of a FAT12/16/32 formatted SDCard. If a valid file is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

Note: The bootable SDCard slot is Slot A.

12.9 EEPROM Boot

The EEPROM Boot program searches for a valid application in an EEPROM connected to the TWI address: 0x0050_0000.

If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. See "Valid Image Detection" for more information on Valid Image Detection.

12.10 SAM-BA Boot

If no valid DataFlash device has been found during the DataFlash boot sequence, the SAM-BA boot program is performed.

The SAM-BA boot principle is to:

- Wait for USB Device enumeration.
- In parallel, wait for character(s) received on the DBGU.
- Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as in Table 12-5.

Table 12-5: 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*: '>'

© 2017 Microchip Technology Inc.

- 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*: '>'

12.10.1 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.

12.10.2 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.

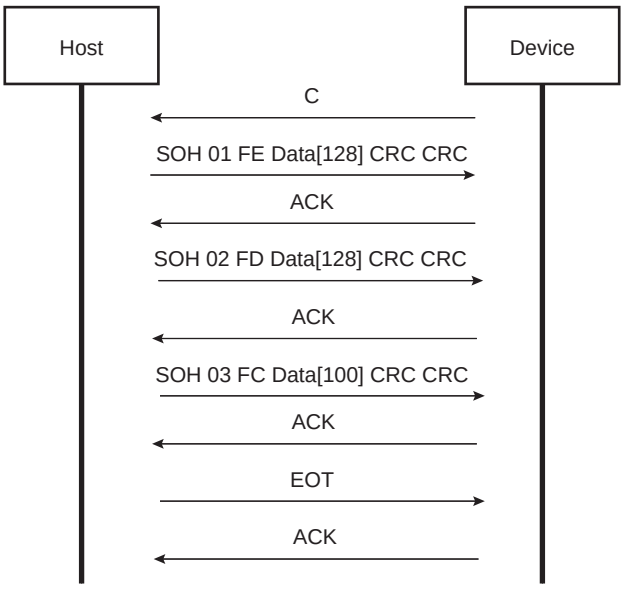
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<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 12-8 shows a transmission using this protocol.

Figure 12-8: Xmodem Transfer Example



12.10.3 USB Device Port

A 48 MHz USB clock is necessary to use the USB Device port. It has been programmed earlier in the device initialization procedure with PLLB configuration.

The device uses the USB communication device class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, beginning with Windows 98SE. The CDC document, available at www.usb.org, describes a way to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID is Microchip's vendor ID 0x03EB. The product ID is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

© 2017 Microchip Technology Inc.

Microchip provides an INF example to see the device as a new serial port and also provides another custom driver used by the SAM-BA application: atm6124.sys. Refer to the document “USB Basic Application”, literature number 6123, for more details.

12.10.3.1 Enumeration Process

The USB protocol is a master/slave protocol. This is the host that starts the enumeration sending requests to the device through the control endpoint. The device handles standard requests as defined in the USB Specification.

Table 12-6: Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value
SET_ADDRESS	Sets the device address for all future device access
SET_CONFIGURATION	Sets the device configuration
GET_CONFIGURATION	Returns the current device configuration value
GET_STATUS	Returns status for the specified recipient
SET_FEATURE	Used to set or enable a specific feature
CLEAR_FEATURE	Used to clear or disable a specific feature

The device also handles some class requests defined in the CDC class.

Table 12-7: Handled Class Requests

Request	Definition
SET_LINE_CODING	Configures DTE rate, stop bits, parity and number of character bits
GET_LINE_CODING	Requests current DTE rate, stop bits, parity and number of character bits
SET_CONTROL_LINE_STATE	RS-232 signal used to tell the DCE device the DTE device is now present

Unhandled requests are STALLed.

12.10.3.2 Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through the endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

If the command requires a response, the host can send IN transactions to pick up the response.

12.11 Hardware and Software Constraints

- The DataFlash, Serial Flash, NAND Flash, SDCard(1), and EEPROM downloaded code size must be inferior to 16K bytes.
- The code is always downloaded from the device address 0x0000_0000 to the address 0x0000_0000 of the internal SRAM (after remap).
- The downloaded code must be position-independent or linked at address 0x0000_0000.
- The DataFlash must be connected to NPCS0 of the SPI.

Note 1: Boot ROM does not support high capacity SDCards.

The SPI and NAND Flash drivers use several PIOs in alternate functions to communicate with devices. Care must be taken when these PIOs are used by the application. The devices connected could be unintentionally driven at boot time, and electrical conflicts between SPI output pins and the connected devices may appear.

To assure correct functionality, it is recommended to plug in critical devices to other pins.

Table 12-8 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 second if no correct boot program is found.

Before performing the jump to the application in internal SRAM, all the PIOs and peripherals used in the boot program are set to their reset state.

Table 12-8: Pins Driven during Boot Program Execution

Peripheral	Pin	PIO Line
SPI0	MOSI	PIOA1
SPI0	MISO	PIOA0
SPI0	SPCK	PIOA2
SPI0	NPCS0	PIOA3
SPI0	NPCS1	PIOC11
PIOC	NANDCS	PIOC14
Address Bus	NAND CLE	A22
Address Bus	NAND ALE	A21
MCI0	MCDA0	PIOA6
MCI0	MCCDA	PIOA7
MCI0	MCCK	PIOA8
MCI0	MCDA1	PIOA9
MCI0	MCDA2	PIOA10
MCI0	MCDA3	PIOA11
TWI	TWCK	PIOA24
TWI	TWD	PIOA23
DBGU	DRXD	PIOB14
DBGU	DTXD	PIOB15

© 2017 Microchip Technology Inc.

13. Reset Controller (RSTC)

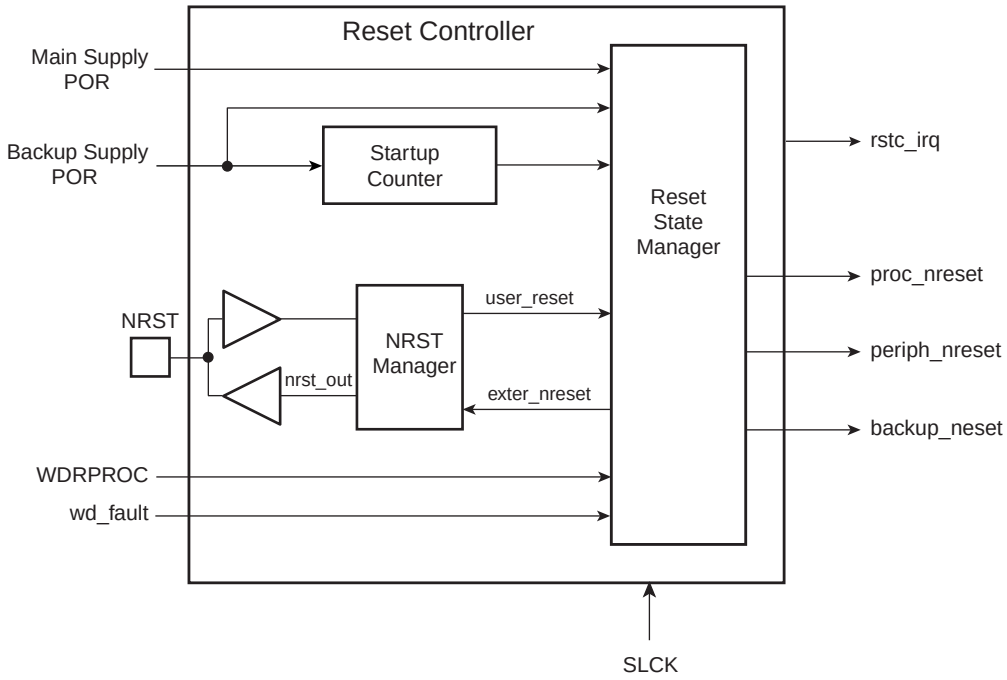
13.1 Overview

The Reset Controller (RSTC), based on power-on reset cells, 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, a Startup Counter 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.
- backup_nreset: Affects all the peripherals powered by VDDBU.
- 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 startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in Section 40.5 Crystal Oscillator Characteristics.

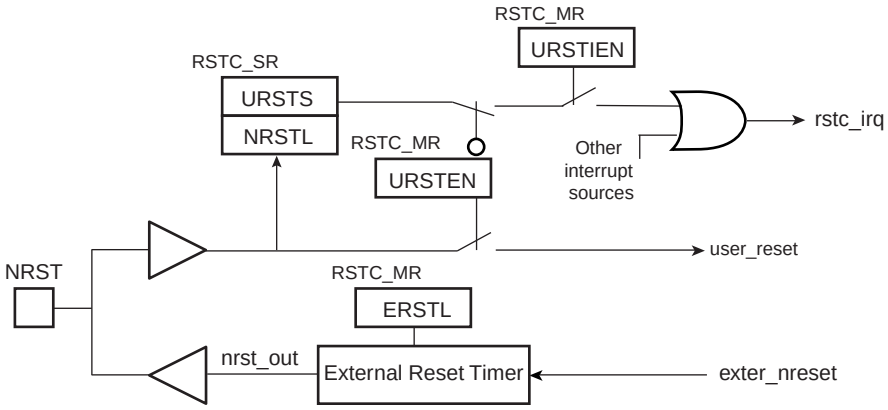
The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

© 2017 Microchip Technology Inc.

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.

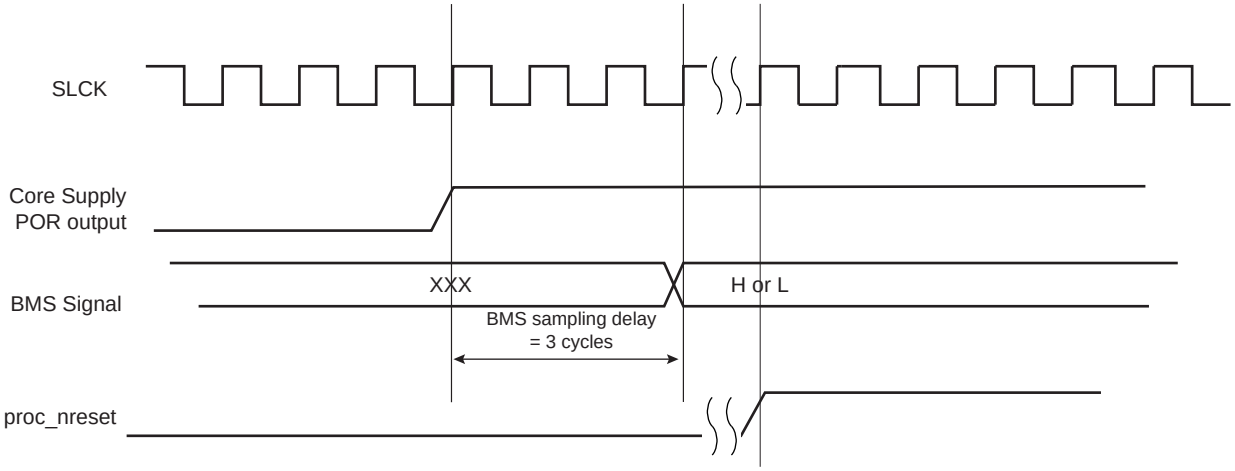
As the field is within RSTC_MR, which is backed-up, this field can be used to shape the system power-up reset for devices requiring a longer startup time than the Slow Clock Oscillator.

13.3.3 BMS Sampling

The product matrix manages a boot memory that depends on the level on the BMS pin at reset. The BMS signal is sampled three slow clock cycles after the Core Power-On-Reset output rising edge.

© 2017 Microchip Technology Inc.

Figure 13-3: BMS Sampling



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 VDDBU and VDDCORE are powered on. The backup supply POR cell output rises and is filtered with a Startup Counter, which operates at Slow Clock. The purpose of this counter is to make sure the Slow Clock oscillator is stable before starting up the device. The length of startup time is hardcoded to comply with the Slow Clock Oscillator startup time.

After this time, the processor clock is released at Slow Clock and all the other signals remain valid for 3 cycles for proper processor and logic reset. Then, 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 backup_nreset, as ERSTL defaults at value 0x0.

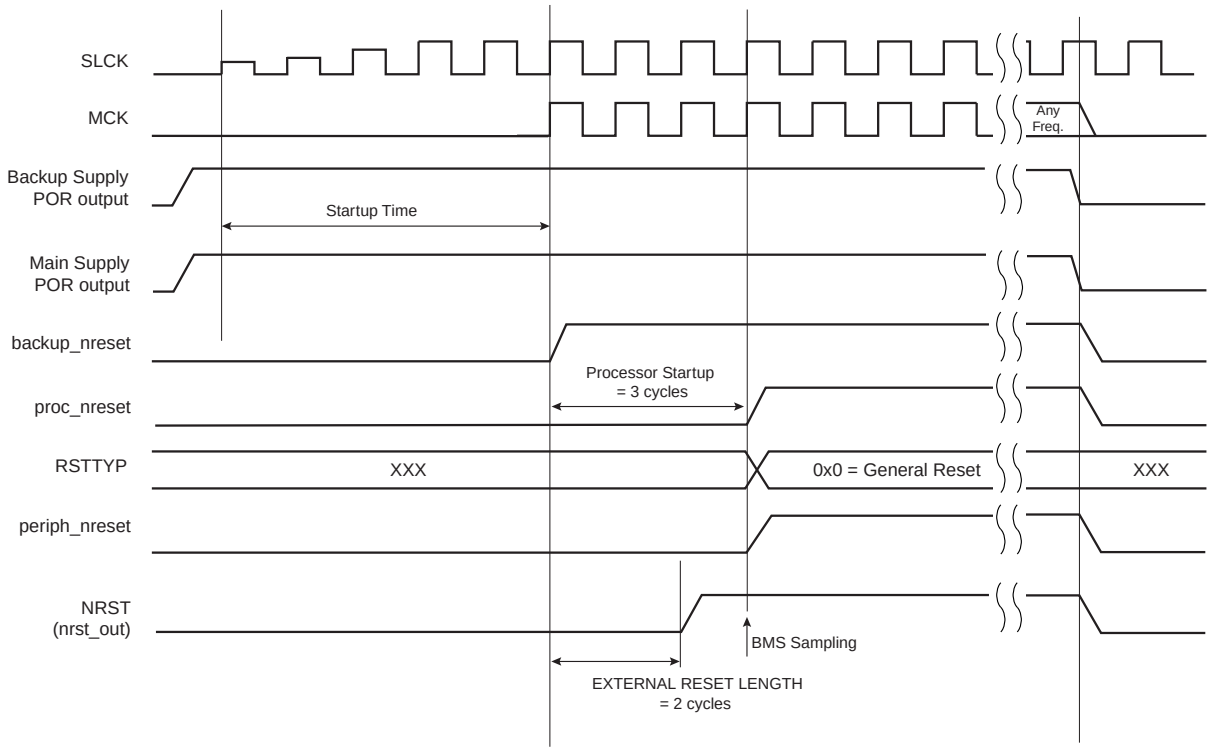
When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shutdown.

VDDBU only activates the backup_nreset signal.

The backup_nreset must be released so that any other reset can be generated by VDDCORE (Main Supply POR output).

Figure 13-4 shows how the General Reset affects the reset signals.

Figure 13-4: General Reset State



13.3.4.2 Wake-up Reset

The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles, depending on the requirements of the Arm processor.

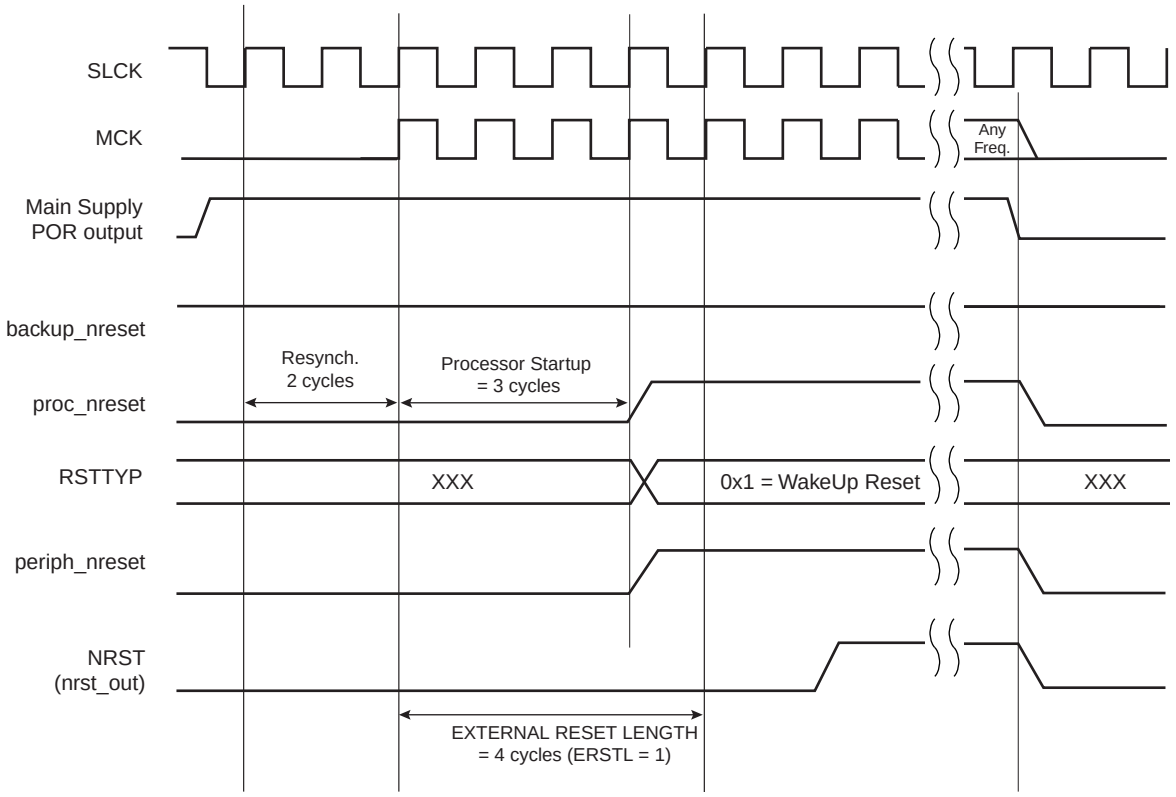
At the end of this delay, the processor and other reset signals rise. The field RSTTYP in RSTC_SR is updated to report a Wake-up Reset.

The “nrst_out” remains asserted for EXTERNAL_RESET_LENGTH cycles. As RSTC_MR is backed-up, the programmed number of cycles is applicable.

When the Main Supply is detected falling, the reset signals are immediately asserted. This transition is synchronous with the output of the Main Supply POR.

© 2017 Microchip Technology Inc.

Figure 13-5: Wake-up State



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.

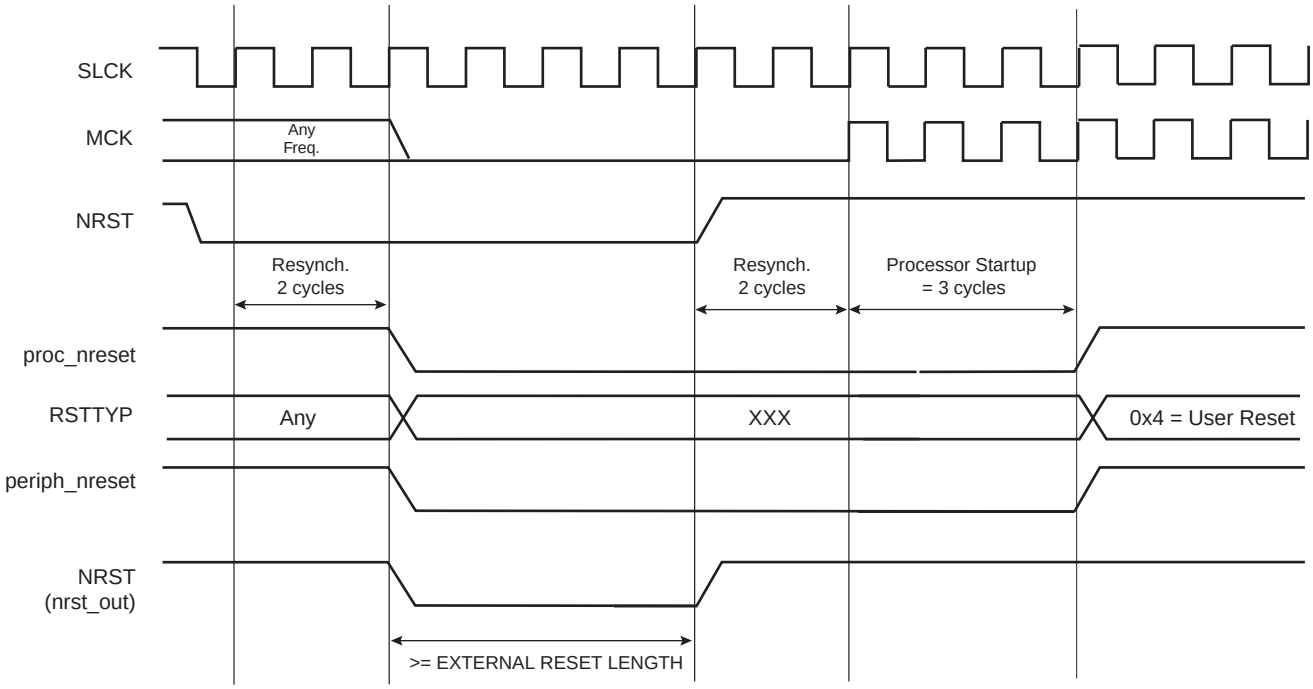
The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 3-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

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.

© 2017 Microchip Technology Inc.

Figure 13-6: 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.

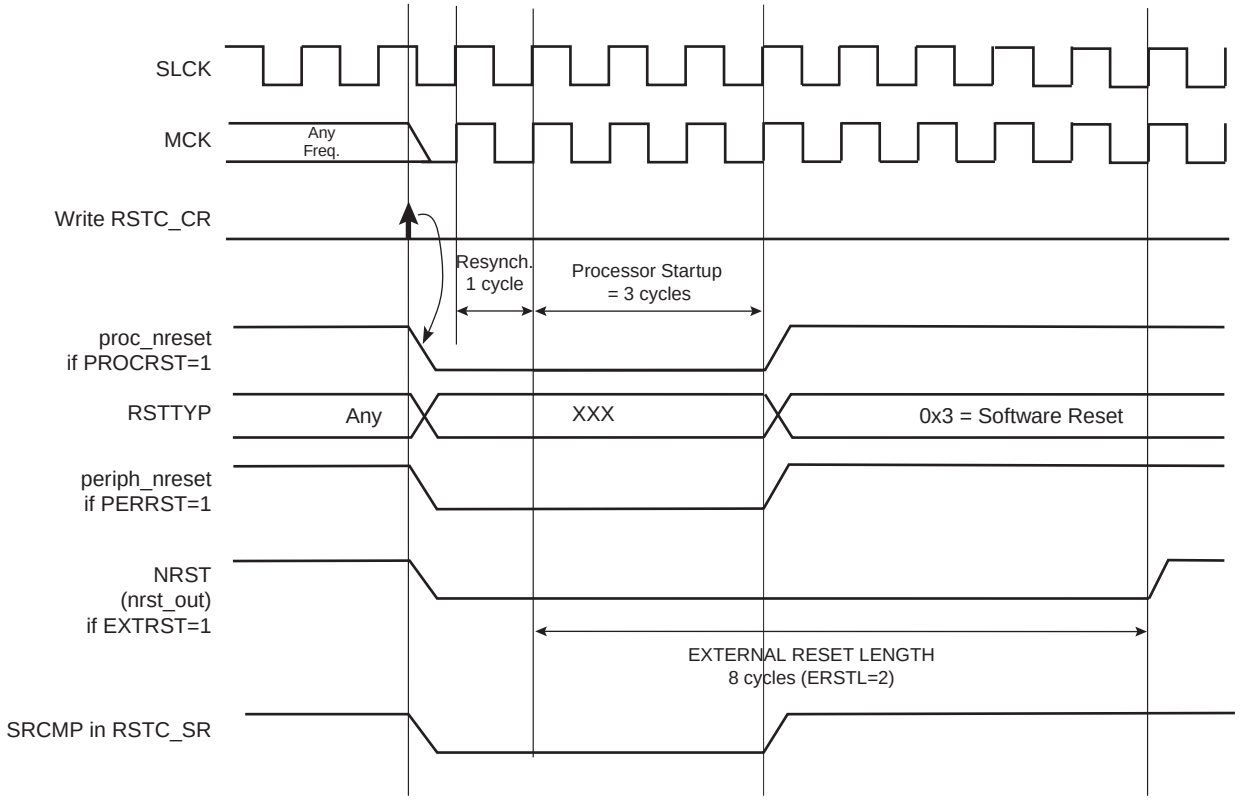
If EXTRST is set, the nrst_out signal is asserted depending on the programming of the field ERSTL. However, the resulting falling edge on NRST does not lead to a User Reset.

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.

© 2017 Microchip Technology Inc.

Figure 13-7: 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.

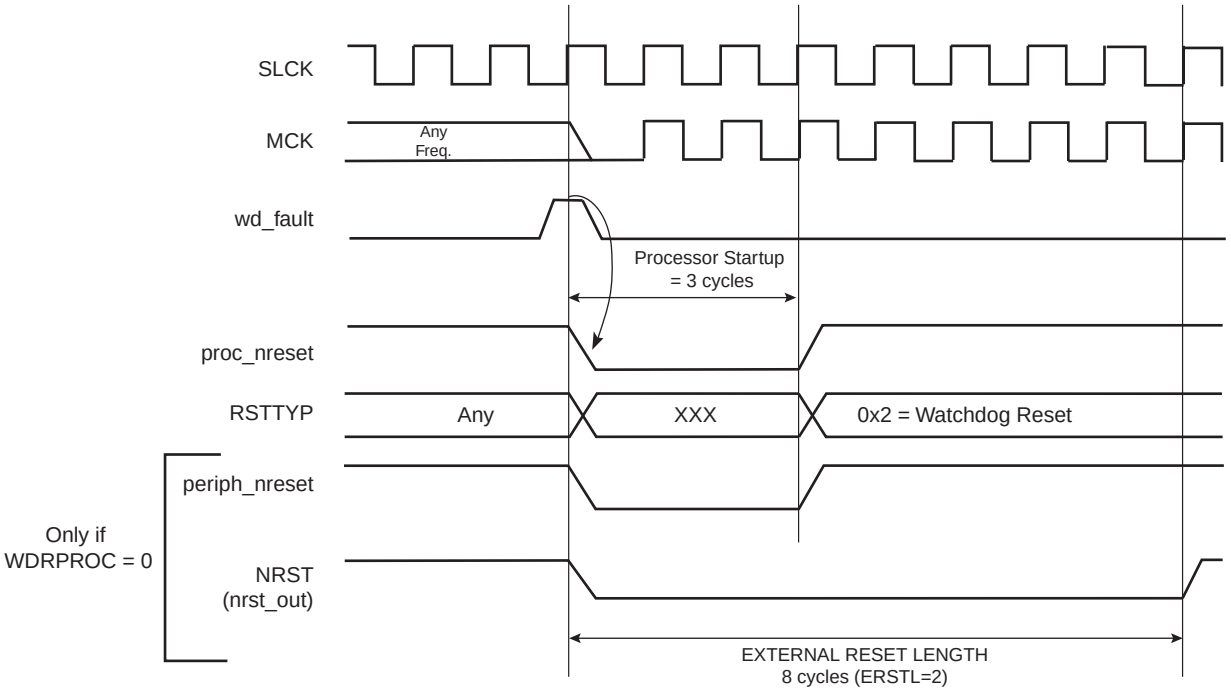
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- 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 13-8: 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:

- Backup Reset
- Wake-up 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.

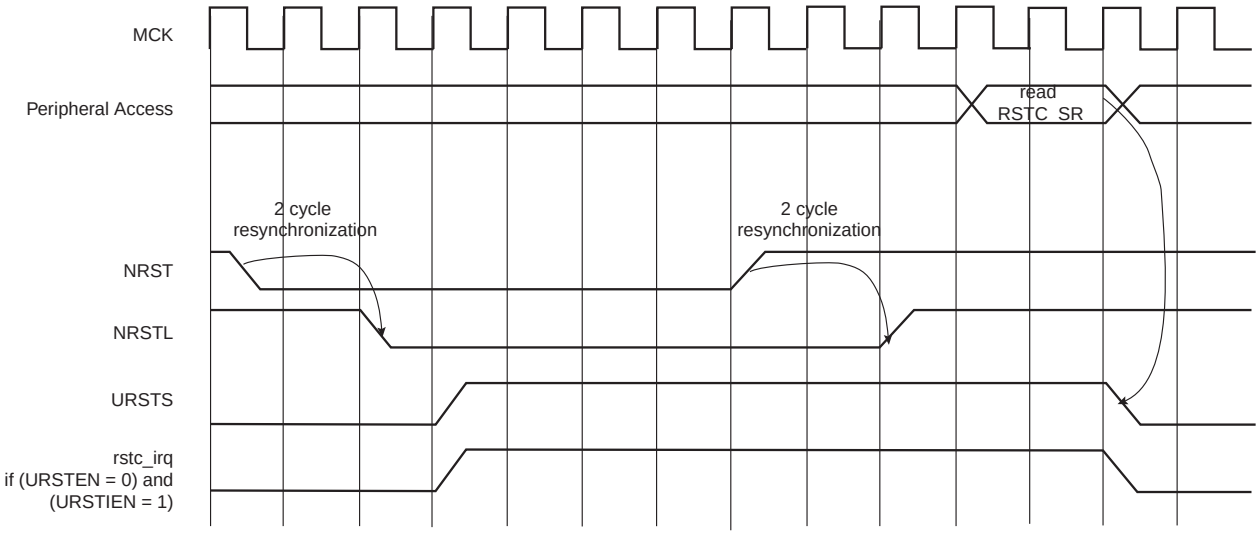
13.3.6 Reset Controller Status Register

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. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 13-9). If the User Reset is disabled (URSTEN = 0) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR, the URSTS bit triggers an interrupt. Reading the RSTC_SR status register resets the URSTS bit and clears the interrupt.

© 2017 Microchip Technology Inc.

Figure 13-9: Reset Controller Status and Interrupt



13.4 Reset Controller (RSTC) User Interface

Table 13-1: Register Mapping

Offset	Register	Name	Access	Reset Value	Backup Reset Value
0x00	Control Register	RSTC_CR	Write-only	–	–
0x04	Status Register	RSTC_SR	Read-only	0x0000_0001	0x0000_0000
0x08	Mode Register	RSTC_MR	Read/Write	–	0x0000_0000

Note 1: The reset value of RSTC_SR either reports a General Reset or a Wake-up Reset depending on last rising power supply.

© 2017 Microchip Technology Inc.

13.4.1 Reset Controller Control Register

Name:RSTC_CR

Access: 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: 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	Both VDDCORE and VDDBU rising
0	0	1	Wake Up Reset	VDDCORE rising
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.

© 2017 Microchip Technology Inc.

13.4.3 Reset Controller Mode Register

Name:RSTC_MR

Access: 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.

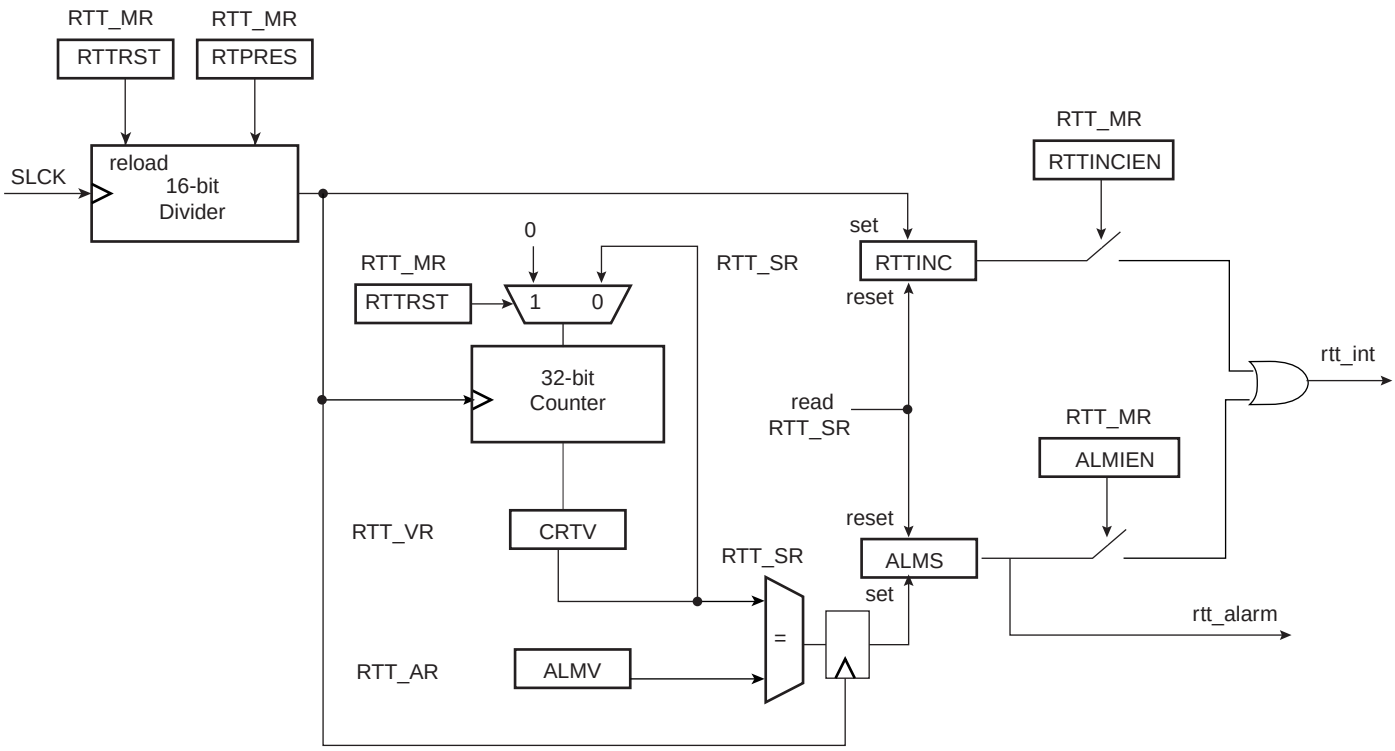
14. Real-time Timer (RTT)

14.1 Overview

The Real-time Timer is built around a 32-bit counter and used to count elapsed seconds. It generates a periodic interrupt and/or triggers an alarm on a programmed value.

14.2 Block Diagram

Figure 14-1: Real-time Timer



14.3 Functional Description

The Real-time Timer is used to count elapsed seconds. It is built around a 32-bit counter fed by Slow Clock divided by a programmable 16-bit value. The value can be programmed in the field RTPRES of the Real-time Mode Register (RTT_MR).

Programming RTPRES at 0x00008000 corresponds to feeding the real-time counter with a 1 Hz signal (if the Slow Clock is 32768 Hz). The 32-bit counter can count up to 2^{32} seconds, corresponding to more than 136 years, then roll over to 0.

The Real-time Timer can also be used as a free-running timer with a lower time-base. The best accuracy is achieved by writing RTPRES to 3. Programming RTPRES to 1 or 2 is possible, but may result in losing status events because the status register is cleared two Slow Clock cycles after read. Thus if the RTT is configured to trigger an interrupt, the interrupt occurs during 2 Slow Clock cycles after reading RTT_SR. To prevent several executions of the interrupt handler, the interrupt must be disabled in the interrupt handler and re-enabled when the status register is clear.

The Real-time Timer value (CRTV) can be read at any time in the register RTT_VR (Real-time Value Register). As this value can be updated asynchronously from the Master Clock, it is advisable to read this register twice at the same value to improve accuracy of the returned value.

The current value of the counter is compared with the value written in the alarm register RTT_AR (Real-time Alarm Register). If the counter value matches the alarm, the bit ALMS in RTT_SR is set. The alarm register is set to its maximum value, corresponding to 0xFFFF_FFFF, after a reset.

The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32768 Hz.

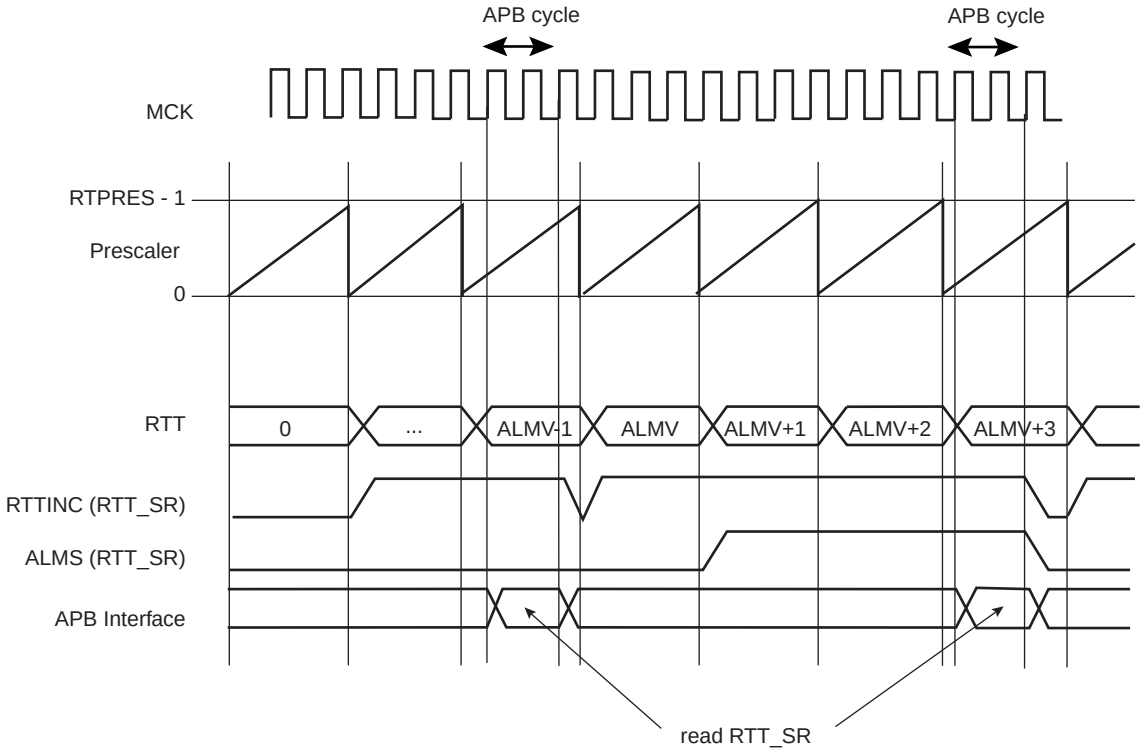
Reading the RTT_SR status register resets the RTTINC and ALMS fields.

© 2017 Microchip Technology Inc.

Writing the bit RTTRST in RTT_MR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

Note: Because of the asynchronism between the Slow Clock (SCLK) and the System Clock (MCK):
1) The restart of the counter and the reset of the RTT_VR current value register is effective only 2 slow clock cycles after the write of the RTTRST bit in the RTT_MR.
2) The status register flags reset is taken into account only 2 slow clock cycles after the read of the RTT_SR (Status Register).

Figure 14-2: RTT Counting



14.4 Real-time Timer (RTT) User Interface

Table 14-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Mode Register	RTT_MR	Read/Write	0x0000_8000
0x04	Alarm Register	RTT_AR	Read/Write	0xFFFF_FFFF
0x08	Value Register	RTT_VR	Read-only	0x0000_0000
0x0C	Status Register	RTT_SR	Read-only	0x0000_0000

© 2017 Microchip Technology Inc.

14.4.1 Real-time Timer Mode Register

Name:RTT_MR

Access: Read/Write

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

RTPRES: Real-time Timer Prescaler Value

Defines the number of SLCK periods required to increment the Real-time timer. RTPRES is defined as follows:

RTPRES = 0: The prescaler period is equal to 2^{16}.

RTPRES ≠ 0: The prescaler period is equal to RTPRES.

ALMIEN: Alarm Interrupt Enable

0: The bit ALMS in RTT_SR has no effect on interrupt.

1: The bit ALMS in RTT_SR asserts interrupt.

RTTINCIEN: Real-time Timer Increment Interrupt Enable

0: The bit RTTINC in RTT_SR has no effect on interrupt.

1: The bit RTTINC in RTT_SR asserts interrupt.

RTTRST: Real-time Timer Restart

1: Reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

© 2017 Microchip Technology Inc.

14.4.2 Real-time Timer Alarm Register

Name:RTT_AR

Access: Read/Write

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

ALMV: Alarm Value

Defines the alarm value (ALMV+1) compared with the Real-time Timer.

© 2017 Microchip Technology Inc.

14.4.3 Real-time Timer Value Register

Name:RTT_VR

Access: Read-only

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

CRTV: Current Real-time Value

Returns the current value of the Real-time Timer.

14.4.4 Real-time Timer Status Register

Name:RTT_SR

Access: 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
–	–	–	–	–	–	RTTINC	ALMS

ALMS: Real-time Alarm Status

0: The Real-time Alarm has not occurred since the last read of RTT_SR.

1: The Real-time Alarm occurred since the last read of RTT_SR.

RTTINC: Real-time Timer Increment

0: The Real-time Timer has not been incremented since the last read of the RTT_SR.

1: The Real-time Timer has been incremented since the last read of the RTT_SR.

© 2017 Microchip Technology Inc.

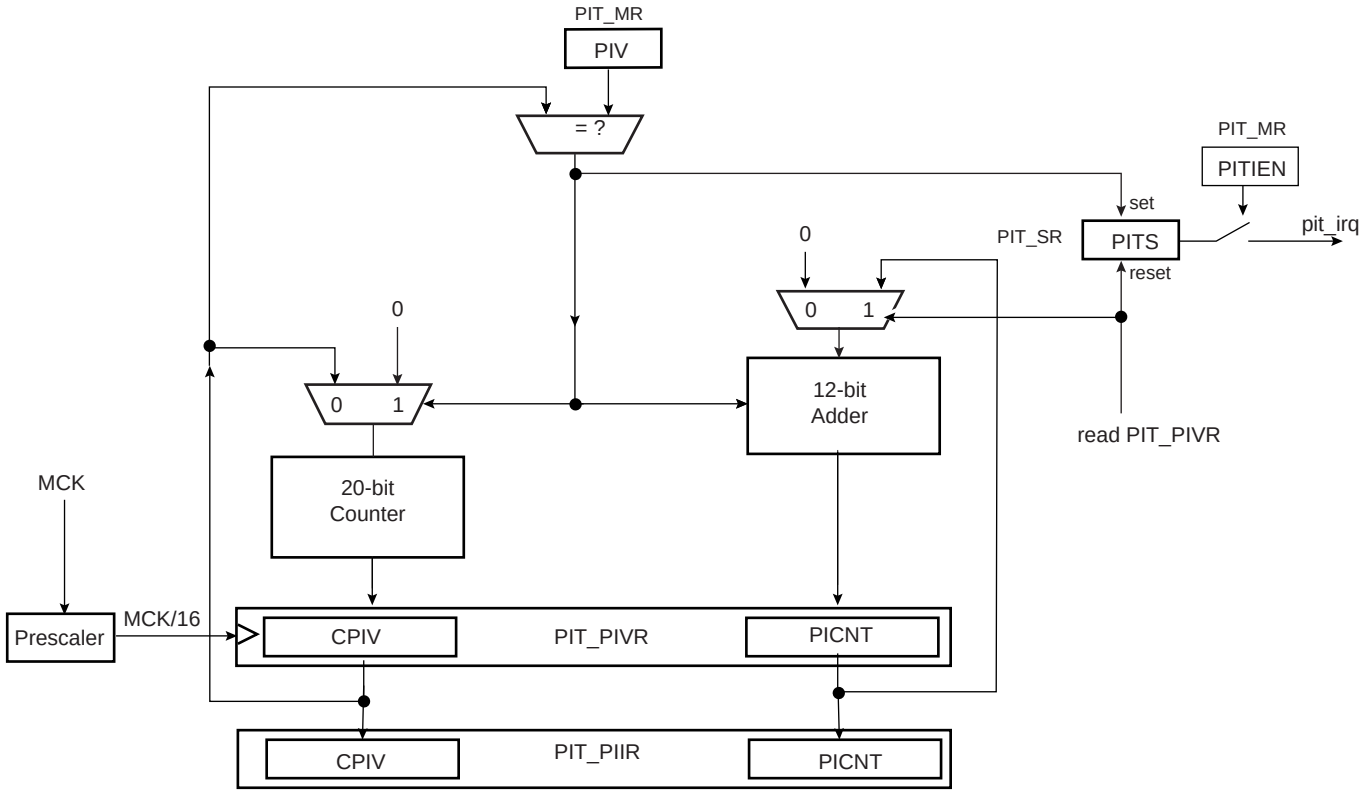
15. Periodic Interval Timer (PIT)

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.

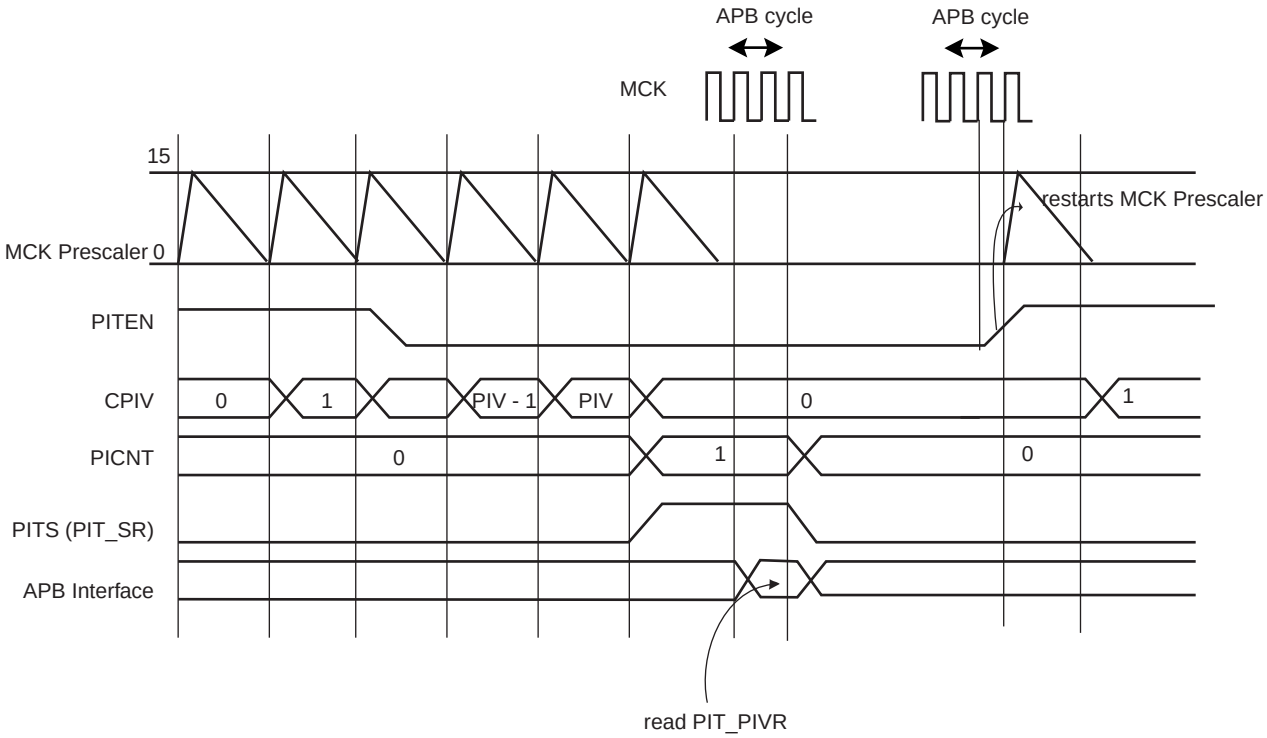
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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 (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



© 2017 Microchip Technology Inc.

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

Name:PIT_MR

Access: 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.

© 2017 Microchip Technology Inc.

15.4.2 Periodic Interval Timer Status Register

Name:PIT_SR

Access: 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

Name:PIT_PIVR

Access: 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.

© 2017 Microchip Technology Inc.

15.4.4 Periodic Interval Timer Image Register

Name:PIT_PIIR

Access: 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.

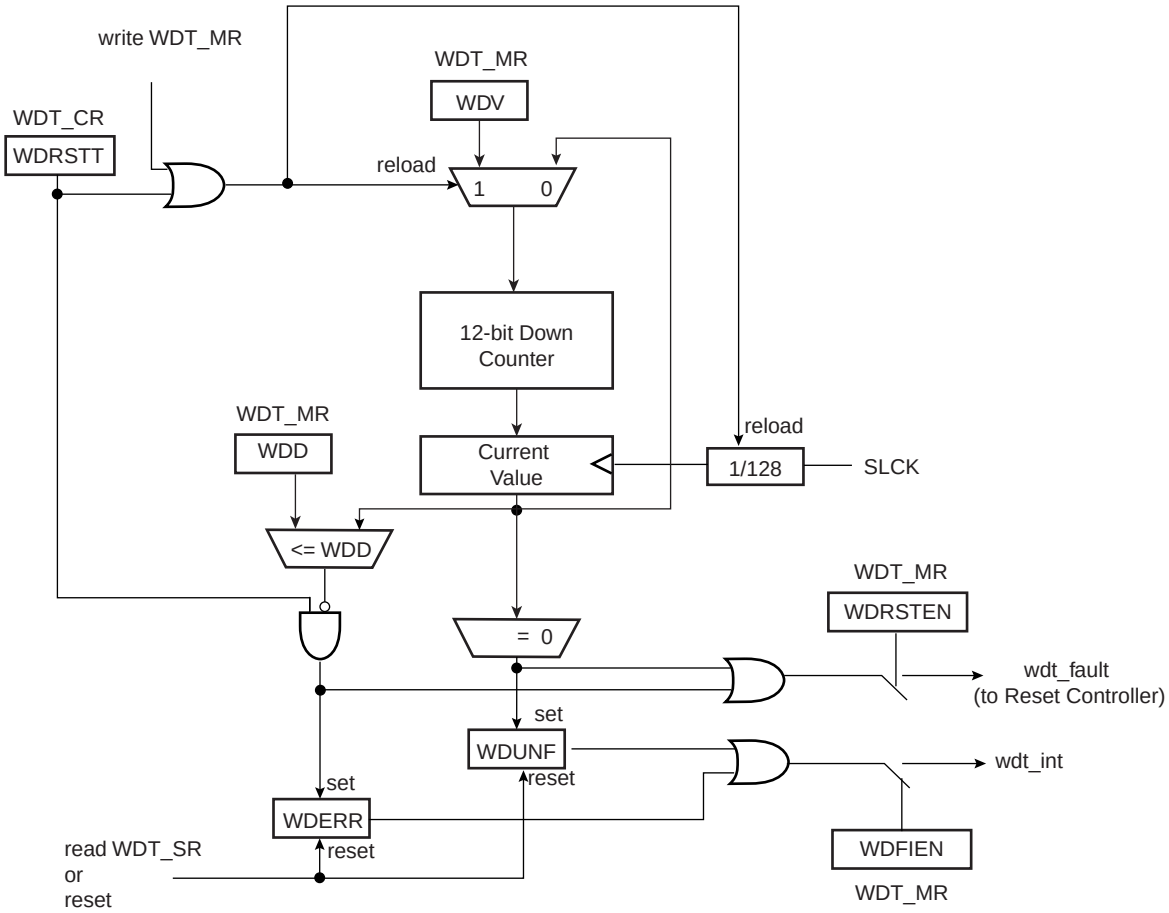
16. Watchdog Timer (WDT)

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 reloads the timer with the newly programmed mode parameters.

© 2017 Microchip Technology Inc.

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 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).

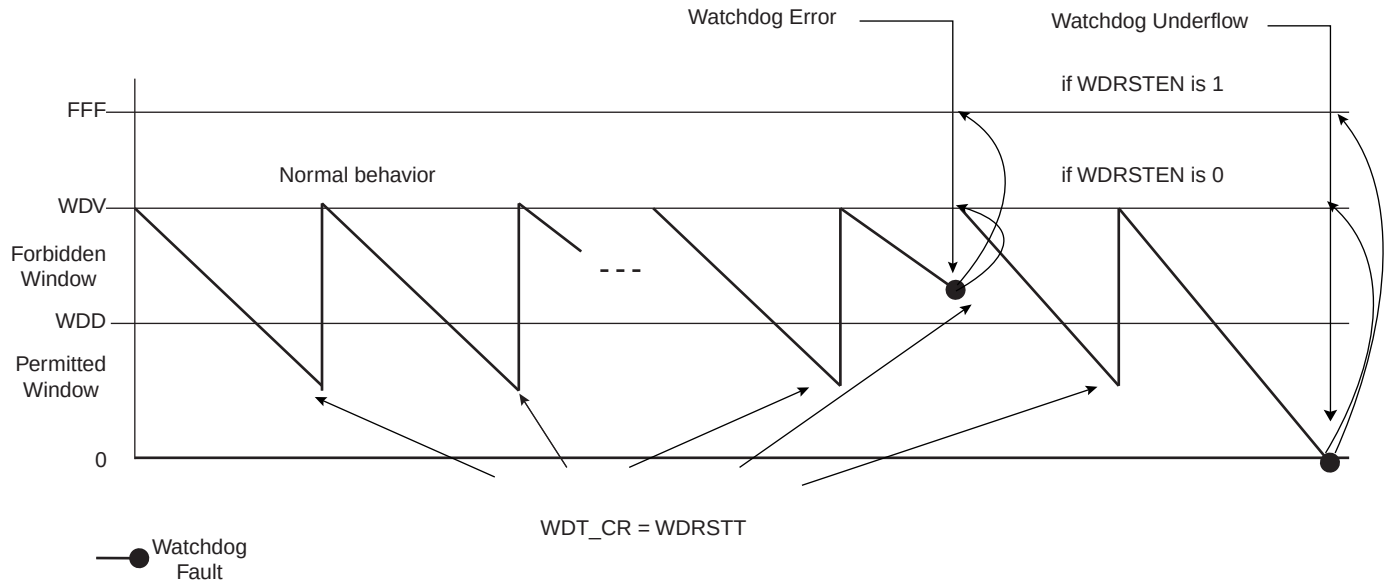
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer datasheet.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

© 2017 Microchip Technology Inc.

16.4.1 Watchdog Timer Control Register

Name:WDT_CR

Access: 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

Name:WDT_MR

Access: 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.

© 2017 Microchip Technology Inc.

16.4.3 Watchdog Timer Status Register

Name:WDT_SR

Access: 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.

© 2017 Microchip Technology Inc.

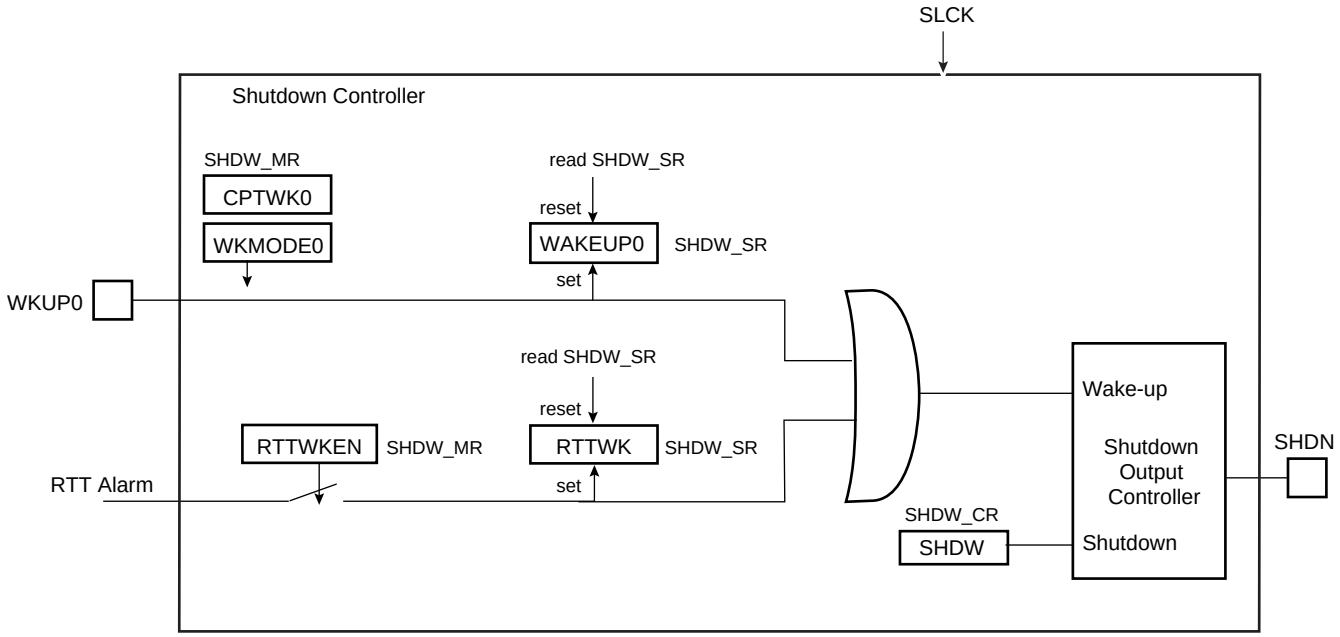
17. Shutdown Controller (SHDWC)

17.1 Overview

The Shutdown Controller controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

17.2 Block Diagram

Figure 17-1: Shutdown Controller Block Diagram



17.3 I/O Lines Description

Table 17-1: I/O Lines Description

Name	Description	Type
WKUP0	Wake-up 0 input	Input
SHDN	Shutdown output	Output

17.4 Product Dependencies

17.4.1 Power Management

The Shutdown Controller is continuously clocked by Slow Clock. The Power Management Controller has no effect on the behavior of the Shutdown Controller.

© 2017 Microchip Technology Inc.

17.5 Functional Description

The Shutdown Controller manages the main power supply. To do so, it is supplied with VDDBU and manages wake-up input pins and one output pin, SHDN.

A typical application connects the pin SHDN to the shutdown input of the DC/DC Converter providing the main power supplies of the system, and especially VDDCORE and/or VDDIO. The wake-up inputs (WKUP0) connect to any push-buttons or signal that wake up the system.

The software is able to control the pin SHDN by writing the Shutdown Control Register (SHDW_CR) with the bit SHDW at 1. The shutdown is taken into account only 2 slow clock cycles after the write of SHDW_CR. This register is password-protected and so the value written should contain the correct key for the command to be taken into account. As a result, the system should be powered down.

A level change on WKUP0 is used as wake-up. Wake-up is configured in the Shutdown Mode Register (SHDW_MR). The transition detector can be programmed to detect either a positive or negative transition or any level change on WKUP0. The detection can also be disabled. Programming is performed by defining WKMODE0.

Moreover, a debouncing circuit can be programmed for WKUP0. The debouncing circuit filters pulses on WKUP0 shorter than the programmed number of 16 SLCK cycles in CPTWK0 of the SHDW_MR. If the programmed level change is detected on a pin, a counter starts. When the counter reaches the value programmed in the corresponding field, CPTWK0, the SHDN pin is released. If a new input change is detected before the counter reaches the corresponding value, the counter is stopped and cleared. WAKEUP0 of the Status Register (SHDW_SR) reports the detection of the programmed events on WKUP0 with a reset after the read of SHDW_SR.

The Shutdown Controller can be programmed so as to activate the wake-up using the RTT alarm (the detection of the rising edge of the RTT alarm is synchronized with SLCK). This is done by writing the SHDW_MR using the RTTWKEN fields. When enabled, the detection of the RTT alarm is reported in the RTTWK bit of the SHDW_SR Status register. It is reset after the read of SHDW_SR. When using the RTT alarm to wake up the system, the user must ensure that the RTT alarm status flag is cleared before shutting down the system. Otherwise, no rising edge of the status flag may be detected and the wake-up fails.

© 2017 Microchip Technology Inc.

17.6 Shutdown Controller (SHDWC) User Interface

Table 17-2: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Shutdown Control Register	SHDW_CR	Write-only	–
0x04	Shutdown Mode Register	SHDW_MR	Read/Write	0x0000_0003
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

© 2017 Microchip Technology Inc.

17.6.1 Shutdown Control Register

Name:SHDW_CR

Access: 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
–	–	–	–	–	–	–	SHDW

SHDW: Shutdown Command

0: No effect.

1: If KEY is correct, asserts the SHDN pin.

KEY: Password

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

17.6.2 Shutdown Mode Register

Name:SHDW_MR

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	RTTWKEN

15	14	13	12	11	10	9	8
CPTWK1				–	–	–	

7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

WKMODE0: Wake-up Mode 0

WKMODE[1:0]		Wake-up Input Transition Selection
0	0	None. No detection is performed on the wake-up input
0	1	Low to high level
1	0	High to low level
1	1	Both levels change

CPTWK0: Counter on Wake-up 0

Defines the number of 16 Slow Clock cycles, the level detection on the corresponding input pin shall last before the wake-up event occurs. Because of the internal synchronization of WKUP0, the SHDN pin is released
(CPTWK x 16 + 1) Slow Clock cycles after the event on WKUP.

RTTWKEN: Real-time Timer Wake-up Enable

0: The RTT Alarm signal has no effect on the Shutdown Controller.

1: The RTT Alarm signal forces the de-assertion of the SHDN pin.

© 2017 Microchip Technology Inc.

17.6.3 Shutdown Status Register

Name:SHDW_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	RTTWK
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

WAKEUP0: Wake-up 0 Status

0: No wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

1: At least one wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

RTTWK: Real-time Timer Wake-up

0: No wake-up alarm from the RTT occurred since the last read of SHDW_SR.

1: At least one wake-up alarm from the RTT occurred since the last read of SHDW_SR.

© 2017 Microchip Technology Inc.

18. SAM9G20 Bus Matrix

18.1 Overview

The Bus Matrix implements a multi-layer AHB based on the AHB-Lite protocol that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects 6 AHB Masters to 5 AHB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency).

The Bus Matrix user interface is compliant with the Arm Advanced High-performance Bus and provides a Chip Configuration User Interface with registers that allow the Bus Matrix to support application specific features.

18.2 Memory Mapping

The Bus Matrix provides one decoder for every AHB Master Interface. The decoder offers each AHB Master several memory mappings. In fact, depending on the product, each memory area may be assigned to several slaves. Booting at the same address while using different AHB slaves (i.e., external RAM, internal ROM or internal Flash, etc.) becomes possible.

The Bus Matrix user interface provides Master Remap Control Register (MATRIX_MRCR) that performs remap action for every master independently.

18.3 Special Bus Granting Techniques

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first accesses of a burst or single transfer. The bus granting mechanism sets a default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master and fixed default master.

18.3.1 No Default Master

At the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits Low Power mode.

18.3.2 Last Access Master

At the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

18.3.3 Fixed Default Master

At the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike last access master, the fixed master doesn't change unless the user modifies it by a software action (field FIXED_DEFMSTR of the related MATRIX_SCFG).

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields, DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field is used to select the default master type (no default, last access master, fixed default master) whereas the 4-bit FIXED_DEFMSTR field is used to select a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Refer to Section 18.5 "Bus Matrix (MATRIX) User Interface".

18.4 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflicting cases occur, in particular when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user the possibility to choose between 2 arbitration types for each slave:

1. Round-Robin Arbitration (the default)
2. Fixed Priority Arbitration

This choice is made through the field ARBT of the Slave Configuration Registers (MATRIX_SCFG).

Each algorithm may be complemented by selecting a default master configuration for each slave.

When a re-arbitration has to be done, it is realized only under specific conditions described in Section 18.4.1 "Arbitration Rules".

© 2017 Microchip Technology Inc.

18.4.1 Arbitration Rules

Each arbiter has the ability to arbitrate between two or more different master's requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: when a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: when a slave is currently doing a single access.
3. End of Burst Cycles: when the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst (see Section 18.4.1.1 "Undefined Length Burst Arbitration").
4. Slot Cycle Limit: when the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken (see Section 18.4.1.2 "Slot Cycle Limit Arbitration").

18.4.1.1 Undefined Length Burst Arbitration

In order to avoid too long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer.

A predicted end of burst is used as for defined length burst transfer, which is selected between the following:

1. Infinite: no predicted end of burst is generated and therefore INCR burst transfer is never broken.
2. Four beat bursts: predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
3. Eight beat bursts: predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
4. Sixteen beat bursts: predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the field ULBT of the Master Configuration Registers (MATRIX_MCFG).

18.4.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break too long accesses such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (MATRIX_SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, half word or word transfer.

18.4.2 Round-Robin Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a round-robin manner. If two or more master's requests arise at the same time, the master with the lowest number is first serviced then the others are serviced in a round-robin manner.

There are three round-robin algorithms implemented:

- Round-Robin arbitration without default master
- Round-Robin arbitration with last access master
- Round-Robin arbitration with fixed default master

18.4.2.1 Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

18.4.2.2 Round-Robin Arbitration with Last Access Master

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. At the end of the current transfer, if no other master request is pending, the slave remains connected to the last master that performs the access. Other non privileged masters still get one latency cycle if they want to access the same slave. This technique can be used for masters that mainly perform single accesses.

18.4.2.3 Round-Robin Arbitration with Fixed Default Master

This is another biased round-robin algorithm, it allows the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Requests attempted by this fixed default master do not cause any latency whereas other non privileged masters get one latency cycle. This technique can be used for masters that mainly perform single accesses.

© 2017 Microchip Technology Inc.

18.4.3 Fixed Priority Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If two or more master's requests are active at the same time, the master with the highest priority number is serviced first. If two or more master's requests with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority Registers for Slaves (MATRIX_PRAS and MATRIX_PRBS).

© 2017 Microchip Technology Inc.

18.5 Bus Matrix (MATRIX) User Interface

Table 18-1: Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000000
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read/Write	0x00000002
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read/Write	0x00000002
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read/Write	0x00000002
0x0010	Master Configuration Register 4	MATRIX_MCFG4	Read/Write	0x00000002
0x0014	Master Configuration Register 5	MATRIX_MCFG5	Write-only	0x00000002
0x0018–0x003C	Reserved	–	–	–
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x00000010
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x00000010
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x00000010
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x00000010
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x00000010
0x0054–0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read/Write	0x00000000
0x0084	Reserved	–	–	–
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read/Write	0x00000000
0x008C	Reserved	–	–	–
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read/Write	0x00000000
0x0094	Reserved	–	–	–
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read/Write	0x00000000
0x009C	Reserved	–	–	–
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read/Write	0x00000000
0x00A8–0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104–0x010C	Reserved	–	–	–

© 2017 Microchip Technology Inc.

18.5.1 Bus Matrix Master Configuration Registers

Name:MATRIX_MCFG0...MATRIX_MCFG5

Access: Read/Write

(MATRIX_MCFG5 is 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
–	–	–	–	–	ULBT		

ULBT: Undefined Length Burst Type

0: Infinite Length Burst

No predicted end of burst is generated and therefore INCR bursts coming from this master cannot be broken.

1: Single Access

The undefined length burst is treated as a succession of single accesses allowing rearbitration at each beat of the INCR burst.

2: Four Beat Burst

The undefined length burst is split into 4-beat bursts allowing rearbitration at each 4-beat burst end.

3: Eight Beat Burst

The undefined length burst is split into 8-beat bursts allowing rearbitration at each 8-beat burst end.

4: Sixteen Beat Burst

The undefined length burst is split into 16-beat bursts allowing rearbitration at each 16-beat burst end.

© 2017 Microchip Technology Inc.

18.5.2 Bus Matrix Slave Configuration Registers

Name:MATRIX_SCFG0...MATRIX_SCFG4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	ARBT	

23	22	21	20	19	18	17	16
–			FIXED_DEFMSTR			DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst

When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.

This limit has been placed to avoid locking very slow slaves when very long bursts are used.

This limit should not be very small though. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

DEFMSTR_TYPE: Default Master Type

0: No Default Master

At the end of current slave access, if no other master request is pending, the slave is disconnected from all masters.

This results in one cycle latency for the first access of a burst transfer or for a single access.

1: Last Default Master

At the end of current slave access, if no other master request is pending, the slave stays connected with the last master having accessed it.

This results in not having the one cycle latency when the last master tries to access the slave again.

2: Fixed Default Master

At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master whose

number has been written in the FIXED_DEFMSTR field.

This results in not having the one cycle latency when the fixed master tries to access the slave again.

FIXED_DEFMSTR: Fixed Default Master

This is the number of the Default Master for this slave. Only used if DEFMASTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMASTR_TYPE to 0.

ARBT: Arbitration Type

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

2: Reserved

3: Reserved

18.5.3 Bus Matrix Priority Registers For Slaves

Name:MATRIX_PRAS0...MATRIX_PRAS4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	M5PR		–	–	M4PR	

15	14	13	12	11	10	9	8
–	–	M3PR		–	–	M2PR	

7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

MxPR: Master x Priority

Fixed priority of Master x for access to the selected slave. The higher the number, the higher the priority.

© 2017 Microchip Technology Inc.

18.5.4 Bus Matrix Master Remap Control Register

Name:MATRIX_MRCR

Access: 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
–	–	–	–	–	–	RCB1	RCB0

RCBx: Remap Command Bit for AHB Master x

0: Disable remapped address decoding for the selected Master

1: Enable remapped address decoding for the selected Master

18.6 Chip Configuration User Interface

Table 18-2: Register Mapping

Offset	Register	Name	Access	Reset
0x0110–0x0118	Reserved	–	–	–
0x011C	EBI Chip Select Assignment Register	EBI_CSA	Read/Write	0x00010000
0x0130–0x01FC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

18.6.1 EBI Chip Select Assignment Register

Name:EBI_CSA

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	IOSR	VDDIOMSEL

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	EBI_DBPUC

7	6	5	4	3	2	1	0
–	–	EBI_CS5A	EBI_CS4A	EBI_CS3A	–	EBI_CS1A	–

EBI_CS1A: EBI Chip Select 1 Assignment

0: EBI Chip Select 1 is assigned to the Static Memory Controller.

1: EBI Chip Select 1 is assigned to the SDRAM Controller.

EBI_CS3A: EBI Chip Select 3 Assignment

0: EBI Chip Select 3 is only assigned to the Static Memory Controller and EBI_NCS3 behaves as defined by the SMC.

1: EBI Chip Select 3 is assigned to the Static Memory Controller and the SmartMedia Logic is activated.

EBI_CS4A: EBI Chip Select 4 Assignment

0: EBI Chip Select 4 is only assigned to the Static Memory Controller and EBI_NCS4 behaves as defined by the SMC.

1: EBI Chip Select 4 is assigned to the Static Memory Controller and the CompactFlash Logic (first slot) is activated.

EBI_CS5A: EBI Chip Select 5 Assignment

0: EBI Chip Select 5 is only assigned to the Static Memory Controller and EBI_NCS5 behaves as defined by the SMC.

1: EBI Chip Select 5 is assigned to the Static Memory Controller and the CompactFlash Logic (second slot) is activated.

EBI_DBPUC: EBI Data Bus Pull-Up Configuration

0: EBI D0–D15 Data Bus bits are internally pulled-up to the VDDIOM power supply.

1: EBI D0–D15 Data Bus bits are not internally pulled-up.

VDDIOMSEL: Memory voltage selection

0: Memories are 1.8V powered.

1: Memories are 3.3V powered.

IOSR: I/O Slew Rate Selection

Refer to Section 40.4.4 “I/O Slew Rates”.

© 2017 Microchip Technology Inc.

19. SAM9G20 External Bus Interface

19.1 Overview

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an Arm-based device. The Static Memory, SDRAM and ECC Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, and SDRAM.

The EBI also supports the CompactFlash and the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to eight chip select lines (NCS[7:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

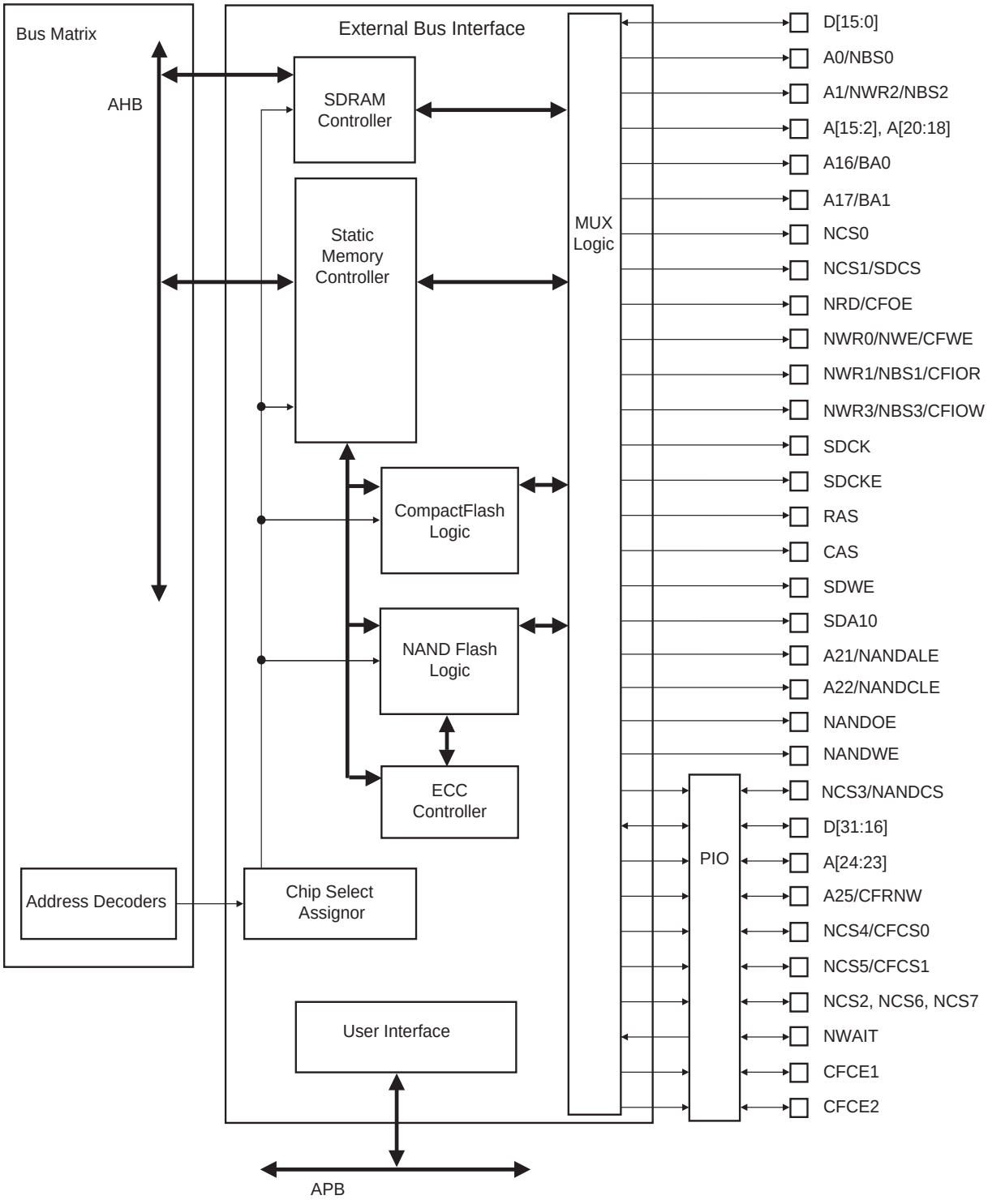
19.2 Block Diagram

19.2.1 External Bus Interface

Figure 19-1 shows the organization of the External Bus Interface.

© 2017 Microchip Technology Inc.

Figure 19-1: Organization of the External Bus Interface



19.3 I/O Lines Description

Table 19-1: EBI I/O Lines Description

Name	Function	Type	Active Level
EBI			
EBI_D0–EBI_D31	Data Bus	I/O	
EBI_A0–EBI_A25	Address Bus	Output	
EBI_NWAIT	External Wait Signal	Input	Low
SMC			
EBI_NCS0–EBI_NCS7	Chip Select Lines	Output	Low
EBI_NWR0–EBI_NWR3	Write Signals	Output	Low
EBI_NRD	Read Signal	Output	Low
EBI_NWE	Write Enable	Output	Low
EBI_NBS0–EBI_NBS3	Byte Mask Signals	Output	Low
EBI for CompactFlash Support			
EBI_CFCE1–EBI_CFCE2	CompactFlash Chip Enable	Output	Low
EBI_CFOE	CompactFlash Output Enable	Output	Low
EBI_CFWE	CompactFlash Write Enable	Output	Low
EBI_CFIOR	CompactFlash I/O Read Signal	Output	Low
EBI_CFIOW	CompactFlash I/O Write Signal	Output	Low
EBI_CFRNW	CompactFlash Read Not Write Signal	Output	
EBI_CFCS0–EBI_CFCS1	CompactFlash Chip Select Lines	Output	Low
EBI for NAND Flash Support			
EBI_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI_NANDOE	NAND Flash Output Enable	Output	Low
EBI_NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
EBI_SDCK	SDRAM Clock	Output	
EBI_SDCKE	SDRAM Clock Enable	Output	High
EBI_SDCS	SDRAM Controller Chip Select Line	Output	Low
EBI_BA0–EBI_BA1	Bank Select	Output	
EBI_SDWE	SDRAM Write Enable	Output	Low
EBI_RAS–EBI_CAS	Row and Column Signal	Output	Low
EBI_NWR0–EBI_NWR3	Write Signals	Output	Low
EBI_NBS0–EBI_NBS3	Byte Mask Signals	Output	Low
EBI_SDA10	SDRAM Address 10 Line	Output	

The connection of some signals through the MUX logic is not direct and depends on the Memory Controller currently being used.

© 2017 Microchip Technology Inc.

Table 19-2 details the connections between the two memory controllers and the EBI pins.

Table 19-2: EBI Pins and Memory Controllers I/O Lines Connections

EBIx Pins	SDRAMC I/O Lines	SMC I/O Lines
EBI_NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
EBI_A0/NBS0	Not Supported	SMC_A0/NLB
EBI_A1/NBS2/NWR2	Not Supported	SMC_A1
EBI_A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
EBI_SDA10	SDRAMC_A10	Not Supported
EBI_A12	Not Supported	SMC_A12
EBI_A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
EBI_A[22:15]	Not Supported	SMC_A[22:15]
EBI_A[25:23]	Not Supported	SMC_A[25:23]
EBI_D[31:0]	D[31:0]	D[31:0]

19.4 Application Example

19.4.1 Hardware Interface

Table 19-3 details the connections to be applied between the EBI pins and the external devices for each memory controller.

Table 19-3: EBI Pins and External Static Devices Connections

	Pins of the SMC Interfaced Device					
Signals: EBI_	**8-bit Static Device**	**2 x 8-bit Static Devices**	**16-bit Static Device**	**4 x 8-bit Static Devices**	**2 x 16-bit Static Devices**	**32-bit Static Device**
D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7	D0–D7
D8–D15	–	D8–D15	D8–D15	D8–D15	D8–15	D8–15
D16–D23	–	–	–	D16–D23	D16–D23	D16–D23
D24–D31	–	–	–	D24–D31	D24–D31	D24–D31
A0/NBS0	A0	–	NLB	–	NLB(3)	BE0(5)
A1/NWR2/NBS2	A1	A0	A0	WE(2)	NLB(4)	BE2(5)
A2–A22	A[2:22]	A[1:21]	A[1:21]	A[0:20]	A[0:20]	A[0:20]
A23–A25	A[23:25]	A[22:24]	A[22:24]	A[21:23]	A[21:23]	A[21:23]
NCS0	CS	CS	CS	CS	CS	CS
NCS1/SDCS	CS	CS	CS	CS	CS	CS
NCS2	CS	CS	CS	CS	CS	CS
NCS3/NANDCS	CS	CS	CS	CS	CS	CS
NCS4/CFCS0	CS	CS	CS	CS	CS	CS
NCS5/CFCS1	CS	CS	CS	CS	CS	CS
NCS6	CS	CS	CS	CS	CS	CS
NCS7	CS	CS	CS	CS	CS	CS
NRD/CFOE	OE	OE	OE	OE	OE	OE

Table 19-3: EBI Pins and External Static Devices Connections (Continued)

Signals: EBI_	Pins of the SMC Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
NWR0/NWE	WE	WE[1]	WE	WE[2]	WE	WE
NWR1/NBS1	–	WE[1]	NUB	WE[2]	NUB[3]	BE1[5]
NWR3/NBS3	–	–	–	WE[2]	NUB[4]	BE3[5]

Note 1: NWR1 enables upper byte writes. NWR0 enables lower byte writes.

2: NWRx enables corresponding byte x writes (x = 0, 1, 2 or 3).

3: NBS0 and NBS1 enable respectively lower and upper bytes of the lower 16-bit word.

4: NBS2 and NBS3 enable respectively lower and upper bytes of the upper 16-bit word.

5: BEx: Byte x Enable (x = 0, 1, 2 or 3).

Table 19-4: EBI Pins and External Device Connections

Signals: EBI_	Pins of the Interfaced Device				
	SDRAM Controller	Static Memory Controller			
	SDRAM	CompactFlash (EBI only)	CompactFlash True IDE Mode (EBI only)	NAND Flash	
D0–D7	D0–D7	D0–D7	D0–D7	I/O0–I/O7	
D8–D15	D8–D15	D8–15	D8–15	I/O8–I/O15	
D16–D31	D16–D31	–	–	–	
A0/NBS0	DQM0	A0	A0	–	
A1/NWR2/NBS2	DQM2	A1	A1	–	
A2–A10	A[0:8]	A[2:10]	A[2:10]	–	
A11	A9	–	–	–	
SDA10	A10	–	–	–	
A12	–	–	–	–	
A13–A14	A[11:12]	–	–	–	
A15	–	–	–	–	
A16/BA0	BA0	–	–	–	
A17/BA1	BA1	–	–	–	
A18–A20	–	–	–	–	
A21/NANDALE	–	–	–	ALE	
A22/NANDCLE	–	REG	REG	CLE	
A23–A24	–	–	–	–	
A25	–	CFRNW[1]	CFRNW[1]	–	
NCS0	–	–	–	–	
NCS1/SDCS	CS	–	–	–	
NCS2	–	–	–	–	

© 2017 Microchip Technology Inc.

Table 19-4: EBI Pins and External Device Connections (Continued)

	Pins of the Interfaced Device			
	SDRAM Controller	Static Memory Controller		
Signals: EBI_	SDRAM	CompactFlash (EBI only)	CompactFlash True IDE Mode (EBI only)	NAND Flash
NCS3/NANDCS	–	–	–	CE(3)
NCS4/CFCS0	–	CFCS0(1)	CFCS0(1)	–
NCS5/CFCS1	–	CFCS1(1)	CFCS1(1)	–
NCS6	–	–	–	–
NCS7	–	–	–	–
NANDOE	–	–	–	RE
NANDWE	–	–	–	WE
NRD/CFOE	–	OE	–	–
NWR0/NWE/CFWE	–	WE	WE	–
NWR1/NBS1/CFIOR	DQM1	IOR	IOR	–
NWR3/NBS3/CFIOW	DQM3	IOW	IOW	–
CFCE1	–	CE1	CS0	–
CFCE2	–	CE2	CS1	–
SDCK	CLK	–	–	–
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT	–	WAIT	WAIT	–
Pxx(2)	–	CD1 or CD2	CD1 or CD2	–
Pxx(2)	–	–	–	CE(3)
Pxx(2)	–	–	–	RDY

Note 1: Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.

2: Any PIO line.

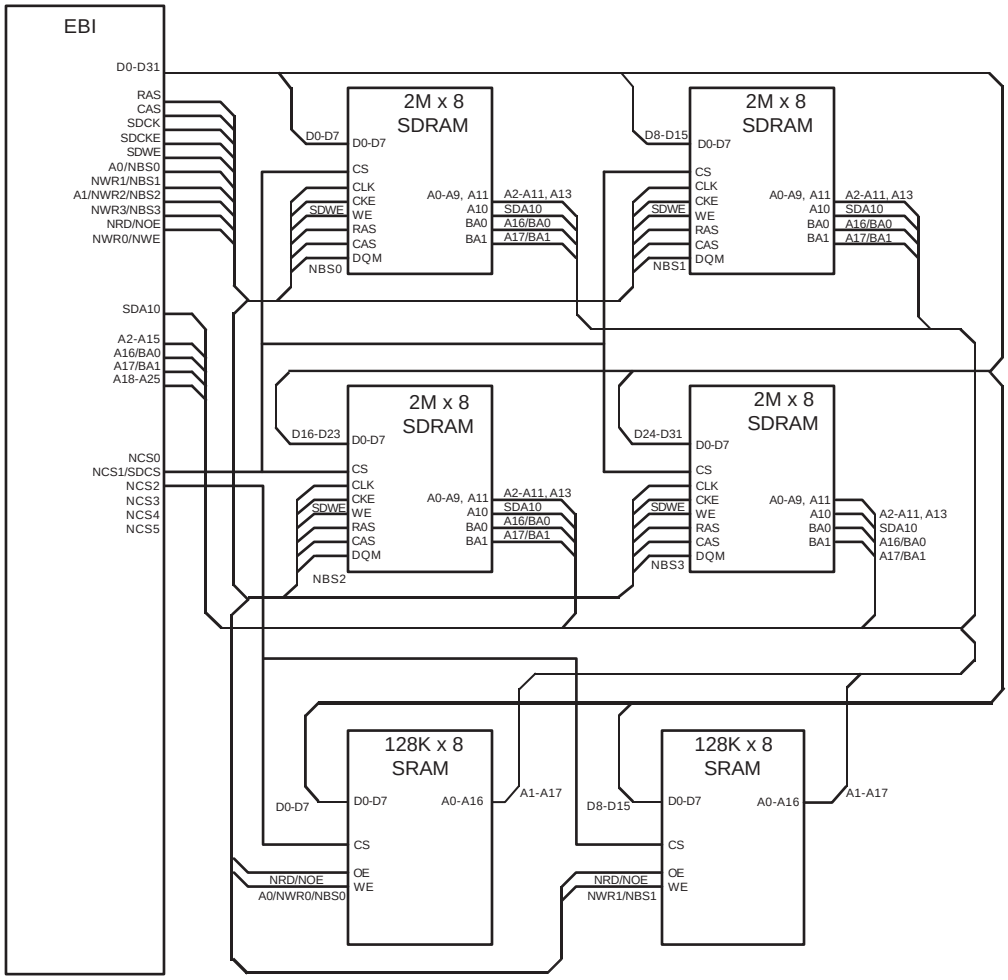
3: CE connection depends on the NAND Flash. For standard NAND Flash devices, it must be connected to any free PIO line. For “CE don’t care” NAND Flash devices, it can be either connected to NCS3/NANDCS or to any free PIO line.

© 2017 Microchip Technology Inc.

19.4.2 Connection Examples

Figure 19-2 shows an example of connections between the EBI and external devices.

Figure 19-2: EBI Connections to Memory Devices



19.5 Product Dependencies

19.5.1 I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

19.6 Functional Description

The EBI transfers data between the internal AHB Bus (handled by the Bus Matrix) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control buses and is composed of the following elements:

- the Static Memory Controller (SMC)
- the SDRAM Controller (SDRAMC)
- the ECC Controller (ECC)
- a chip select assignment feature that assigns an AHB address space to the external devices
- a multiplex controller circuit that shares the pins between the different Memory Controllers
- programmable CompactFlash support logic
- programmable NAND Flash support logic

© 2017 Microchip Technology Inc.

19.6.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 26 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDRAM are executed independently by the SDRAM Controller without delaying the other external Memory Controller accesses.

19.6.2 Pull-up Control

The EBI Chip Select Assignment Register (EBI_CSA) in Section 18.6 “Chip Configuration User Interface” permits enabling of on-chip pull-up resistors on the data bus lines not multiplexed with the PIO Controller lines. The pull-up resistors are enabled after reset. Setting the EBI_CSA.EBI_DBPUC bit disables the pull-up resistors on the lines D0–D15. Enabling the pull-up resistor on the lines D16–D31 can be performed by programming the appropriate PIO controller.

19.6.3 Static Memory Controller

For information on the Static Memory Controller, refer to Section 20. “Static Memory Controller (SMC)”.

19.6.4 SDRAM Controller

For information on the SDRAM Controller, refer to Section 21. “SDRAM Controller (SDRAMC)”.

19.6.5 ECC Controller

For information on the ECC Controller, refer to Section 22. “Error Correction Code Controller (ECC)”.

19.6.6 CompactFlash Support

The External Bus Interface integrates circuitry that interfaces to CompactFlash devices.

The CompactFlash logic is driven by the Static Memory Controller (SMC) on the NCS4 and/or NCS5 address space. Programming the EBI_CS4A and/or EBI_CS5A bit in the EBI_CSA register to the appropriate value enables this logic. For details on this register, refer to Section 18. “SAM9G20 Bus Matrix”. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS4 and/or NCS5 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x6000 0000 and 0x6FFF FFFF for NCS5).

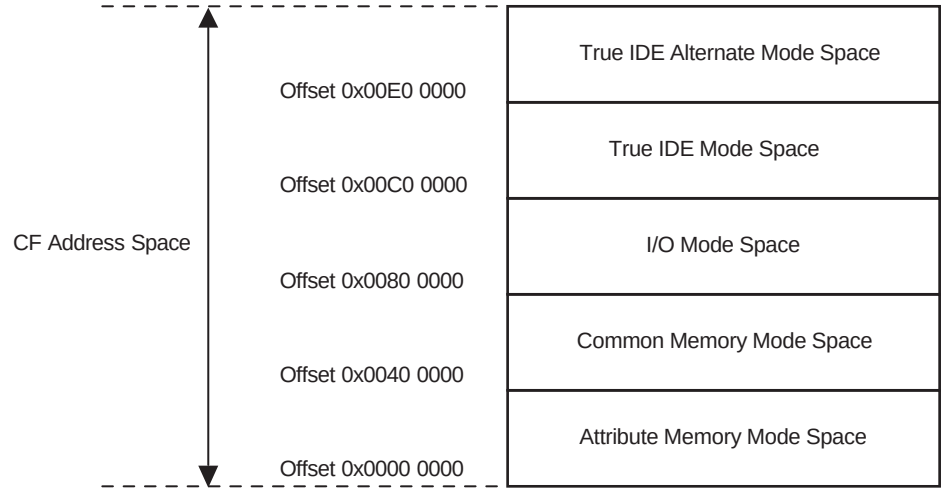
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

19.6.6.1 I/O Mode, Common Memory Mode, Attribute Memory Mode and True IDE Mode

Within the NCS4 and/or NCS5 address space, the current transfer address is used to distinguish I/O mode, common memory mode, attribute memory mode and True IDE mode.

The different modes are accessed through a specific memory mapping as illustrated on Figure 19-3. A[23:21] bits of the transfer address are used to select the desired mode as described in Table 19-5.

Figure 19-3: CompactFlash Memory Mapping



Note: The A22 pin is used to drive the REG signal of the CompactFlash Device (except in True IDE mode).

Table 19-5: CompactFlash Mode Selection

A[23:21]	Mode Base Address
000	Attribute Memory
010	Common Memory
100	I/O Mode
110	True IDE Mode
111	Alternate True IDE Mode

19.6.6.2 CFCE1 and CFCE2 Signals

To cover all types of access, the SMC must be alternatively set to drive 8-bit data bus or 16-bit data bus. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS4 or NCS5). The DBW field in the SMC MODE Register corresponding to the NCS4 and/or NCS5 address space must be configured as shown in Table 19-6 to enable the required access type.

NBS1 and NBS0 are the byte selection signals from SMC and are available when the SMC is set in Byte Select mode on the corresponding Chip Select.

The CFCE1 and CFCE2 waveforms are identical to the corresponding NCSx waveform. For details on these waveforms and timings, refer to Section 20. “Static Memory Controller (SMC)”.

Table 19-6: CFCE1 and CFCE2 Truth Table

Mode	CFCE2	CFCE1	DBW	Comment	SMC Access Mode
Attribute Memory	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0]	Byte Select
Common Memory	NBS1	NBS0	16bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	
I/O Mode	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	

© 2017 Microchip Technology Inc.

Table 19-6: CFCE1 and CFCE2 Truth Table (Continued)

Mode	CFCE2	CFCE1	DBW	Comment	SMC Access Mode
True IDE Mode					
Task File	1	0	8 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[7:0]	
Data Register	1	0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
Alternate True IDE Mode					
Control Register Alternate Status Read	0	1	Don't Care	Access to Even Byte on D[7:0]	Don't Care
Drive Address	0	1	8 bits	Access to Odd Byte on D[7:0]	
Standby Mode or Address Space is not assigned to CF	1	1	–	–	–

19.6.6.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 19-4 illustrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 (and/or NCS5) chip select to the appropriate values.

Figure 19-4: CompactFlash Read/Write Control Signals

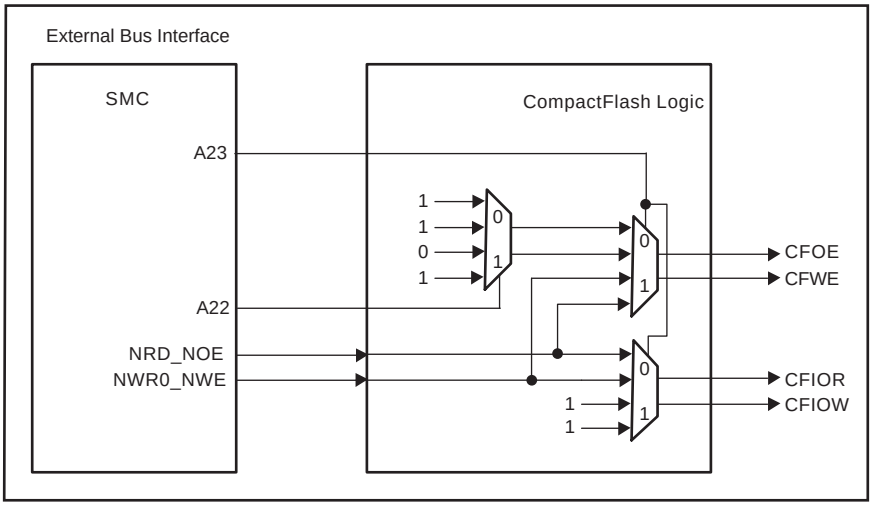


Table 19-7: CompactFlash Mode Selection

Mode Base Address	CFOE	CFWE	CFIOR	CFIOW
Attribute Memory Common Memory	NRD	NWR0_NWE	1	1
I/O Mode	1	1	NRD	NWR0_NWE
True IDE Mode	0	1	NRD	NWR0_NWE

19.6.6.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 19-8 and Table 19-9 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 19-8 are strictly dedicated to the CompactFlash interface as soon as the EBI_CS4A and/or EBI_CS5A bit(s) in the EBI_CSA register is/are set. These pins must not be used to drive any other memory devices.

The EBI pins in Table 19-9 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (EBI_CS4A = 1 and/or EBI_CS5A = 1).

Table 19-8: Dedicated CompactFlash Interface Multiplexing

	CompactFlash Signals		EBI Signals	
Pins	**CS4A = 1**	**CS5A = 1**	**CS4A = 0**	**CS5A = 0**
NCS4/CFCS0	CFCS0		NCS4	
NCS5/CFCS1		CFCS1		NCS5

Table 19-9: Shared CompactFlash Interface Multiplexing

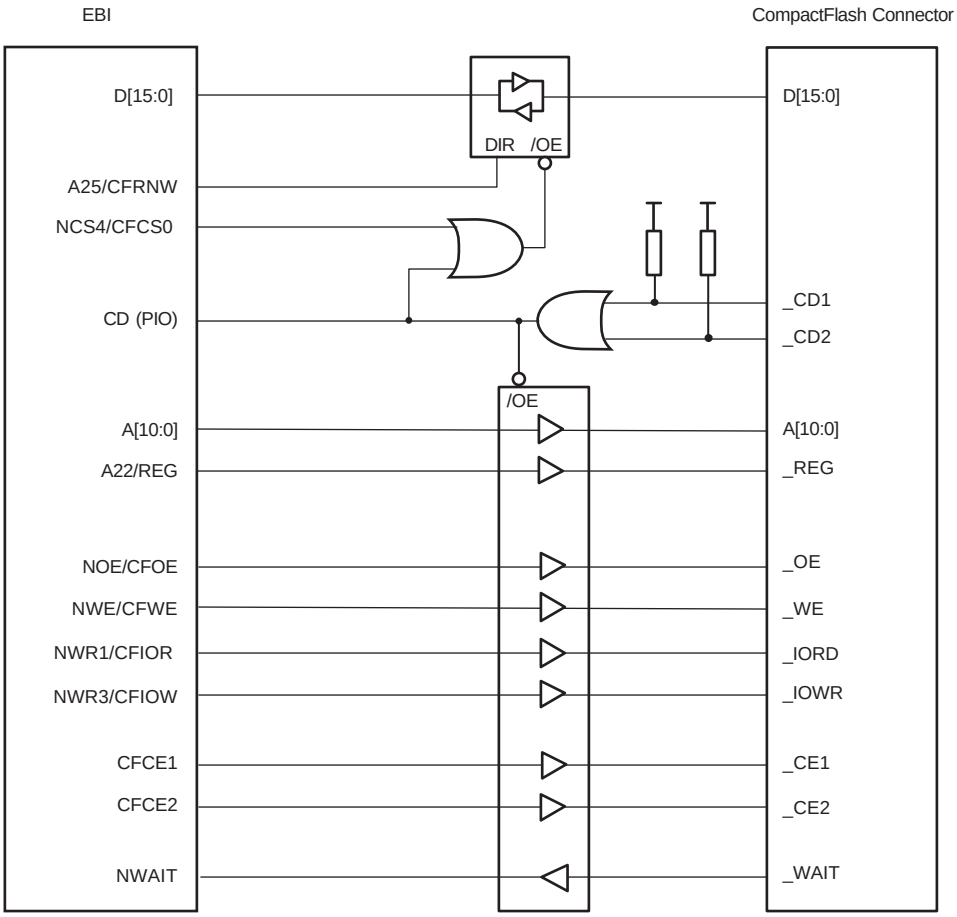
	Access to CompactFlash Device	Access to Other EBI Devices
Pins	**CompactFlash Signals**	**EBI Signals**
NRD/CFOE	CFOE	NRD
NWR0/NWE/CFWE	CFWE	NWR0/NWE
NWR1/NBS1/CFIOR	CFIOR	NWR1/NBS1
NWR3/NBS3/CFIOW	CFIOW	NWR3/NBS3
A25/CFRNW	CFRNW	A25

19.6.6.5 Application Example

Figure 19-5 illustrates an example of a CompactFlash application. CFCS0 and CFRNW signals are not directly connected to the CompactFlash slot 0, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS0 signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash _WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to Section 20. “Static Memory Controller (SMC)”.

© 2017 Microchip Technology Inc.

Figure 19-5: CompactFlash Application Example



19.6.7 NAND Flash Support

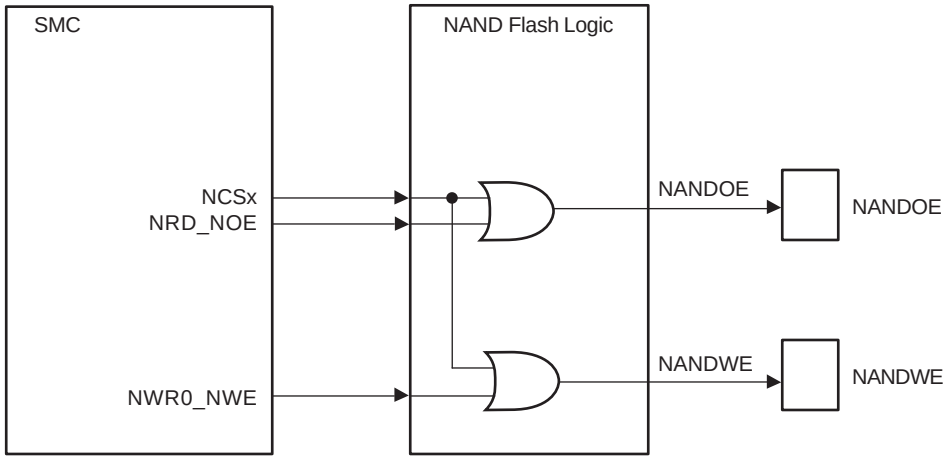
The External Bus Interface integrates circuitry that interfaces to NAND Flash devices.

19.6.7.1 External Bus Interface

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the EBI_CS3A field in the EBI_CSA register to the appropriate value enables the NAND Flash logic. For details on this register, refer to Section 18. “SAM9G20 Bus Matrix”. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. See Figure 19-6 for more information. For details on the waveforms, refer to Section 20. “Static Memory Controller (SMC)”.

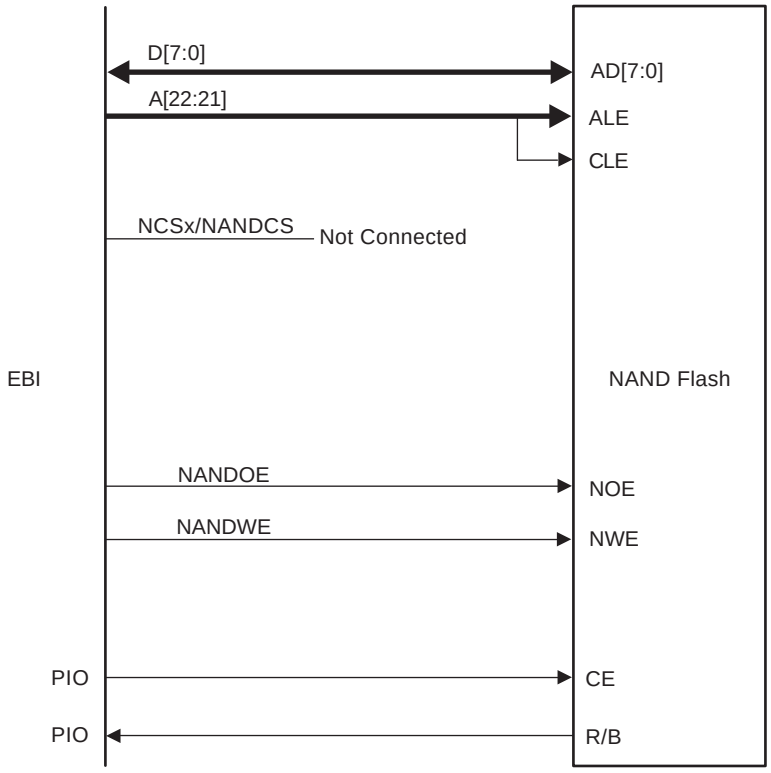
Figure 19-6: NAND Flash Signal Multiplexing on EBI Pins



19.6.7.2 NAND Flash Signals

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCSx is not selected, preventing the device from returning to standby mode.

Figure 19-7: NAND Flash Application Example



Note: The External Bus Interface is also able to support 16-bit devices.

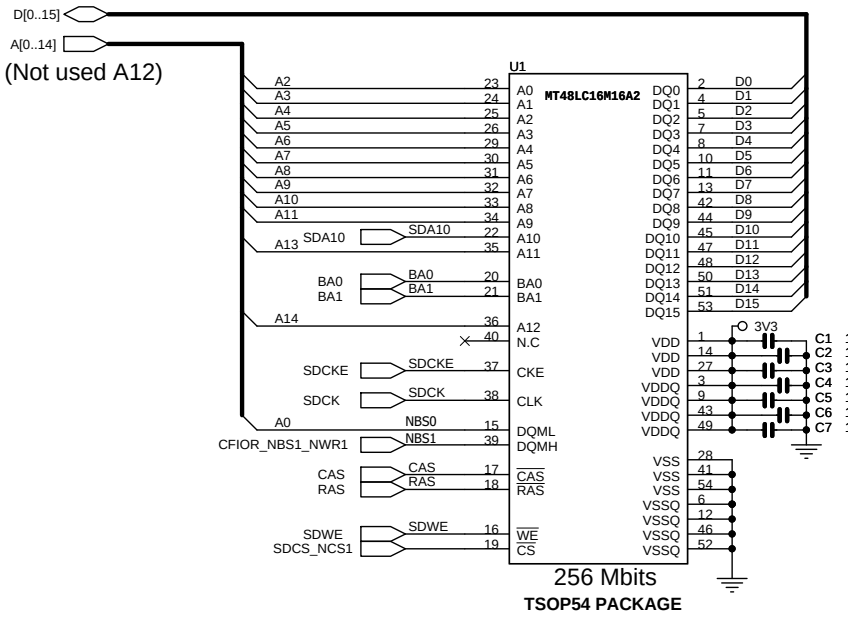
© 2017 Microchip Technology Inc.

19.7 Implementation Examples

The following hardware configurations are given for illustration only. The user should refer to the memory manufacturer website to check current device availability.

19.7.1 16-bit SDRAM

Figure 19-8: Hardware Configuration - 16-bit SDRAM



19.7.1.1 Software Configuration - 16-bit SDRAM

The following configuration has to be performed:

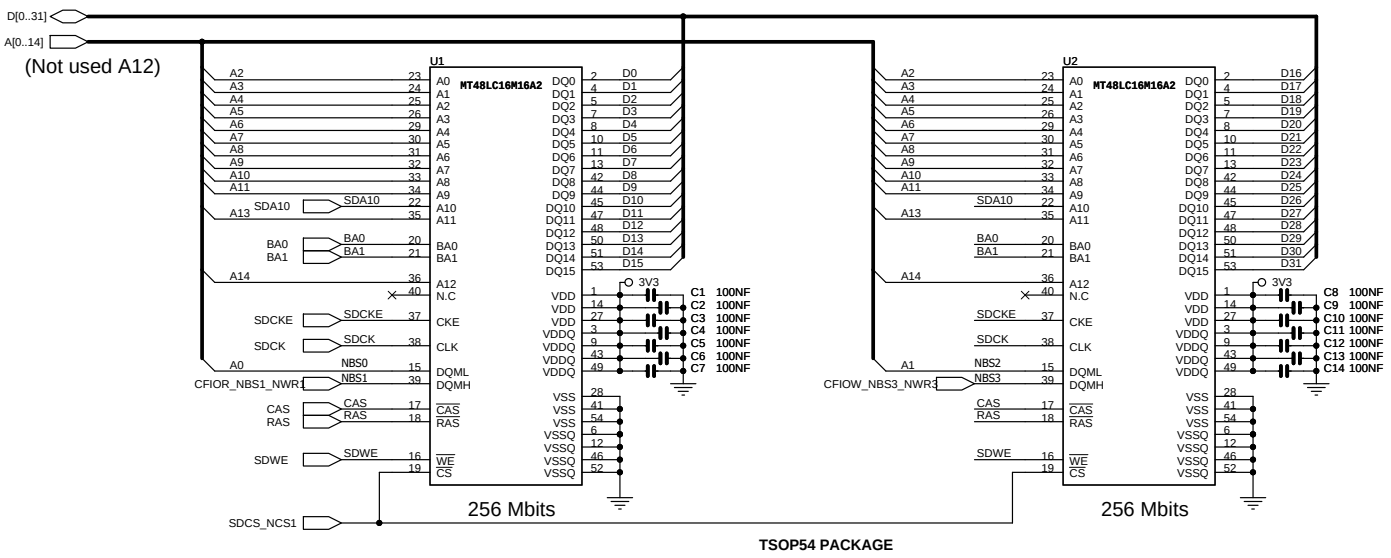
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The Data Bus Width is to be programmed to 16 bits.

The SDRAM initialization sequence is described in Section 21.4.1 “SDRAM Device Initialization”.

19.7.2 32-bit SDRAM

19.7.2.1 Hardware Configuration - 32-bit SDRAM



19.7.2.2 Software Configuration - 32-bit SDRAM

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

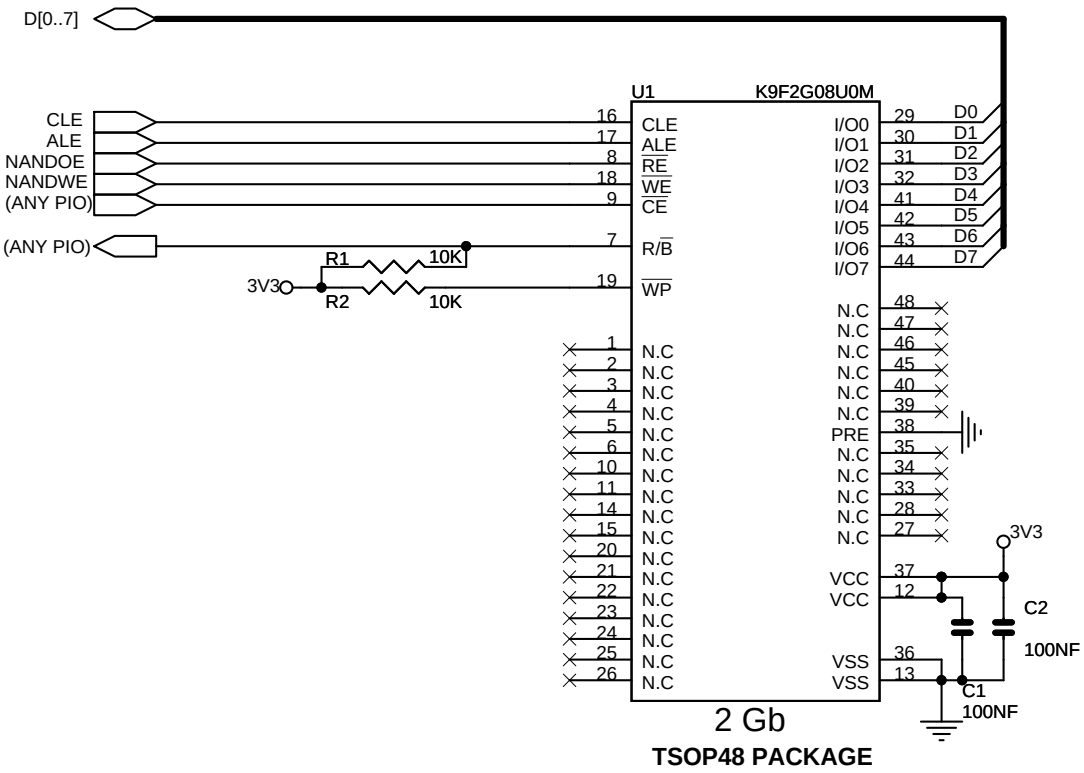
The Data Bus Width is to be programmed to 32 bits. The data lines D[16..31] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

The SDRAM initialization sequence is described in Section 21.4.1 “SDRAM Device Initialization”.

© 2017 Microchip Technology Inc.

19.7.3 8-bit NAND Flash

Figure 19-9: Hardware Configuration - 8-bit NAND Flash



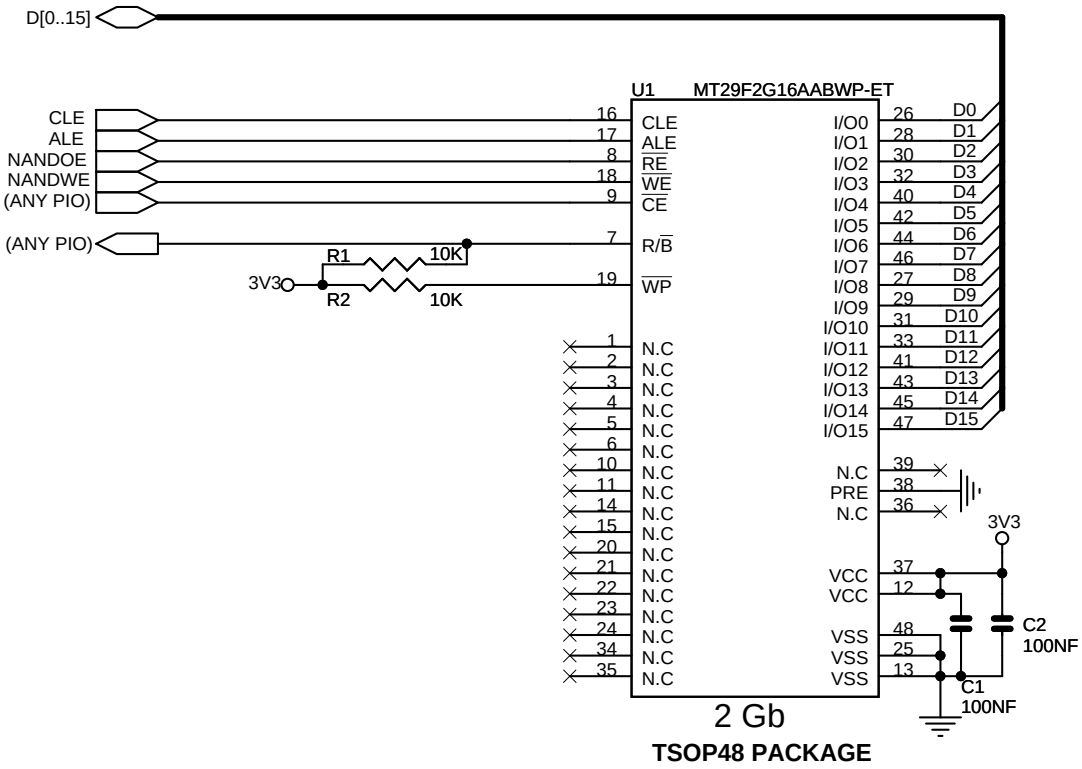
19.7.3.1 Software Configuration - 8-bit NAND Flash

The following configuration has to be performed:

- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register located in the bus matrix memory space
- Reserve A21 / A22 for ALE / CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bit A21 and A22 during accesses.
- Configure a PIO line as an input to manage the Ready/Busy signal.
- Configure Static Memory Controller CS3 Setup, Pulse, Cycle and Mode accordingly to NAND Flash timings, the data bus width and the system bus frequency.

19.7.4 16-bit NAND Flash

Figure 19-10: Hardware Configuration - 16-bit NAND Flash



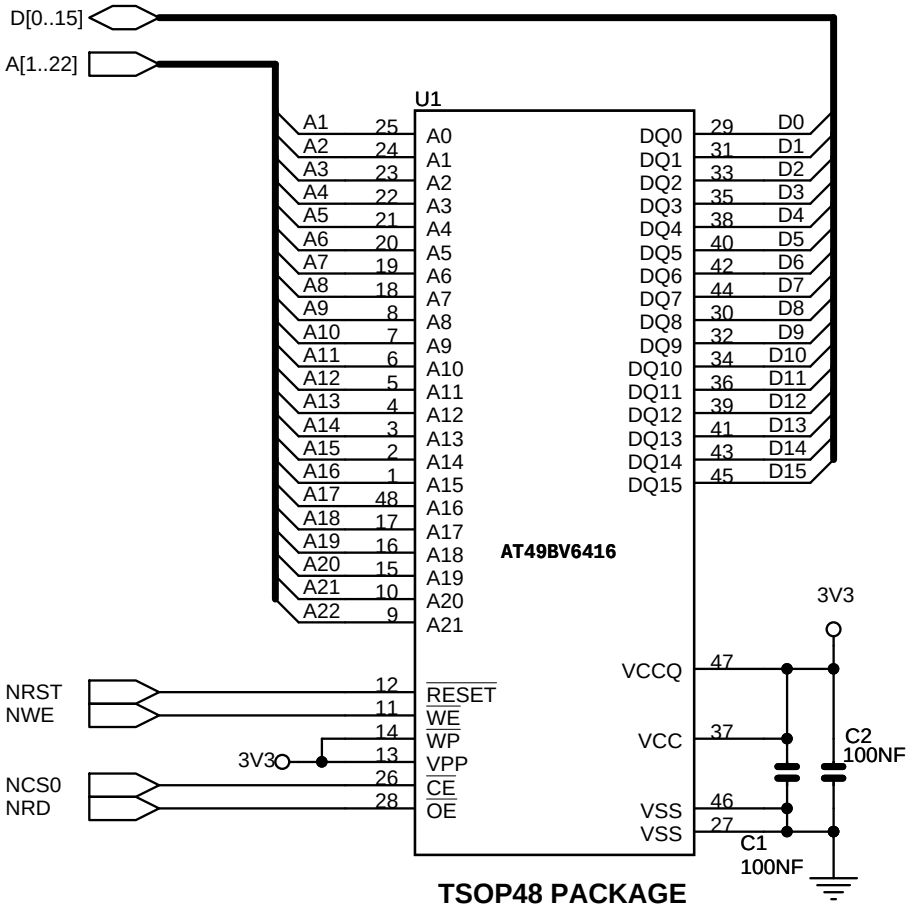
19.7.4.1 Software Configuration - 16-bit NAND Flash

The software configuration is the same as for an 8-bit NAND Flash except the data bus width programmed in the SMC MODE Register.

© 2017 Microchip Technology Inc.

19.7.5 NOR Flash on NCS0

Figure 19-11: Hardware Configuration - NOR Flash on NCS0



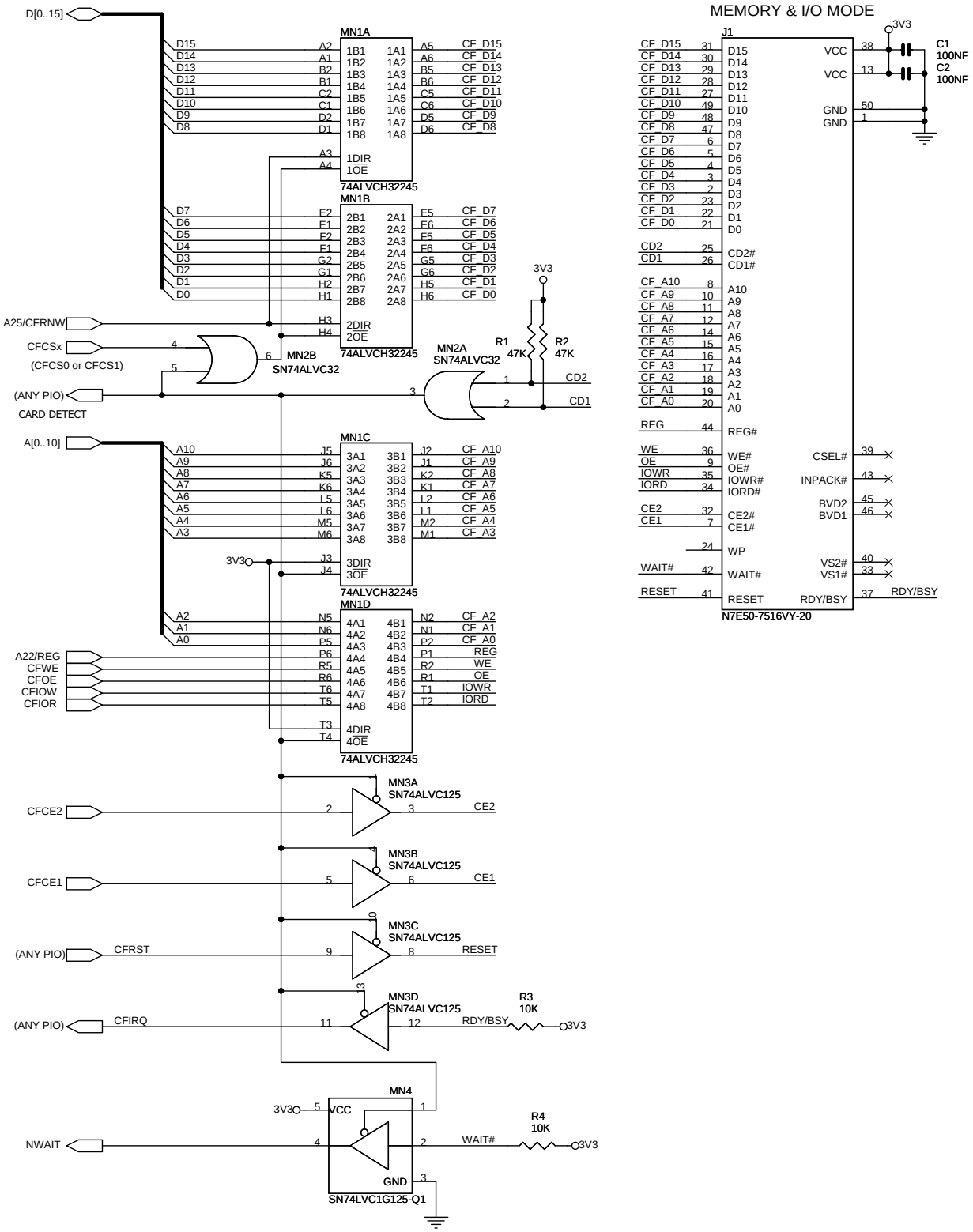
19.7.5.1 Software Configuration - NOR Flash on NCS0

The default configuration for the Static Memory Controller, byte select mode, 16-bit data bus, Read/Write controlled by Chip Select, allows boot on 16-bit non-volatile memory at slow clock.

For another configuration, configure the Static Memory Controller CS0 Setup, Pulse, Cycle and Mode depending on Flash timings and system bus frequency.

19.7.6 Compact Flash

19.7.6.1 Hardware Configuration - Compact Flash



© 2017 Microchip Technology Inc.

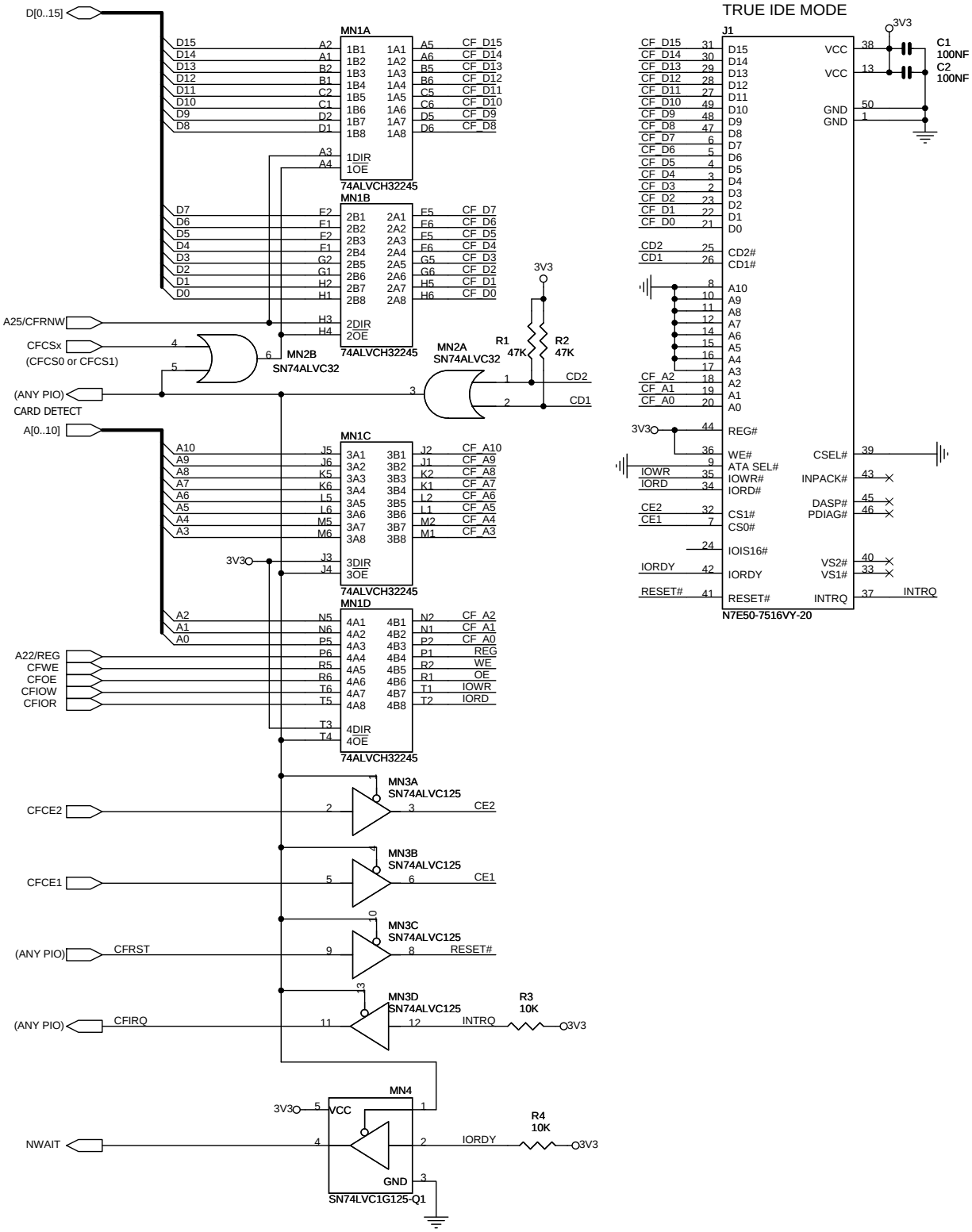
19.7.6.2 Software Configuration - Compact Flash

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI_CS4A or/and EBI_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A23 is to select I/O (A23 = 1) or Memory mode (A23 = 0) and the address line A22 for REG function.
- A23, CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.

19.7.7 Compact Flash True IDE

19.7.7.1 Hardware Configuration - Compact Flash True IDE



© 2017 Microchip Technology Inc.

19.7.7.2 Software Configuration - Compact Flash True IDE

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI_CS4A or/and EBI_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A21 is to select Alternate True IDE (A21 = 1) or True IDE (A21 = 0) modes.
- CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.

20. Static Memory Controller (SMC)

20.1 Overview

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 8 Chip Selects and a 26-bit address bus. The 32-bit data bus can be configured to interface with 8-, 16-, or 32-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

20.2 I/O Lines Description

Table 20-1: I/O Line Description

Name	Description	Type	Active Level
NCS[7:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A1/NWR2/NBS2	Address Bit 1/Write 2/Byte 2 Select Signal	Output	Low
NWR3/NBS3	Write 3/Byte 3 Select Signal	Output	Low
A[25:2]	Address Bus	Output	
D[31:0]	Data Bus	I/O	
NWAIT	External Wait Signal	Input	Low

20.3 Multiplexed Signals

Table 20-2: Static Memory Controller (SMC) Multiplexed Signals

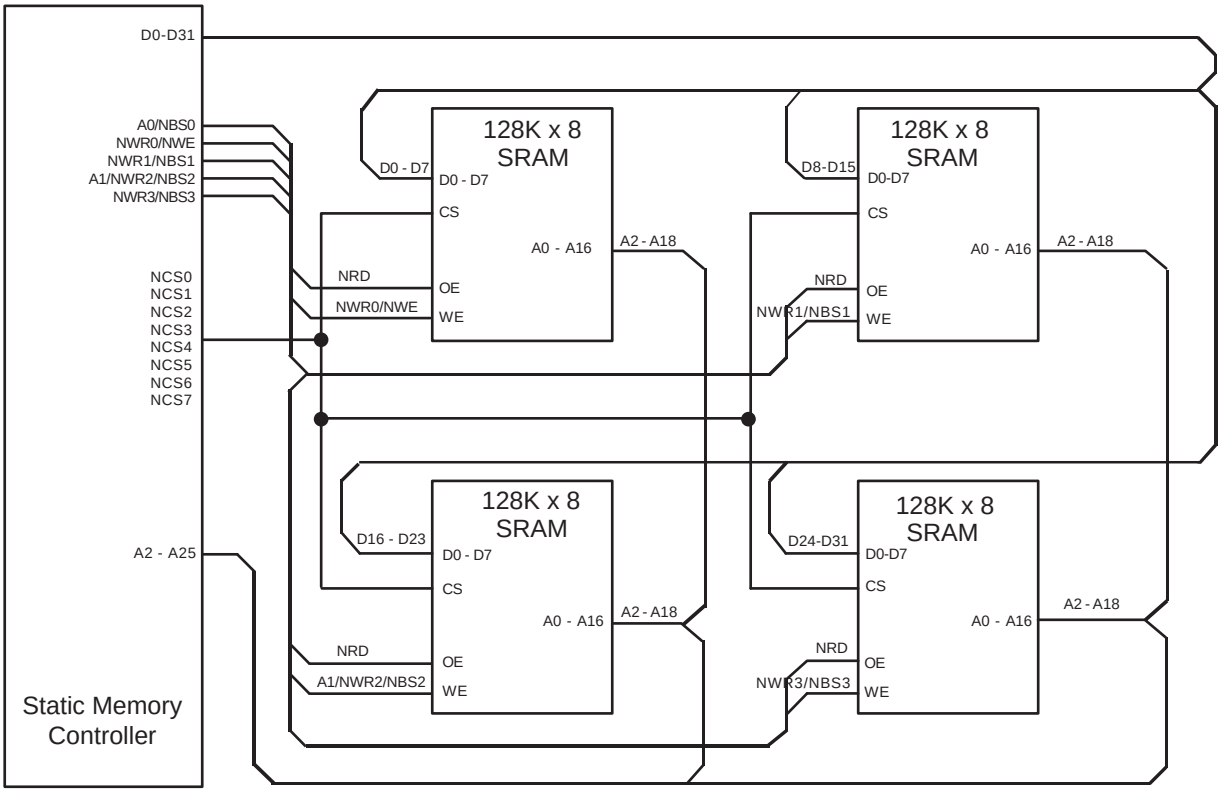
Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see “Byte Write or Byte Select Access”.
A0	NBS0		8-bit or 16-/32-bit data bus, see “Data Bus Width”.
NWR1	NBS1		Byte-write or byte-select access see “Byte Write or Byte Select Access”.
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see “Data Bus Width”. Byte-write or byte-select access, see “Byte Write or Byte Select Access”.
NWR3	NBS3		Byte-write or byte-select access see “Byte Write or Byte Select Access”.

© 2017 Microchip Technology Inc.

20.4 Application Example

20.4.1 Hardware Interface

Figure 20-1: SMC Connections to Static Memory Devices



20.5 Product Dependencies

20.5.1 I/O Lines

The pins used for interfacing the Static Memory Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Static Memory Controller pins to their peripheral function. If I/O Lines of the SMC are not used by the application, they can be used for other purposes by the PIO Controller.

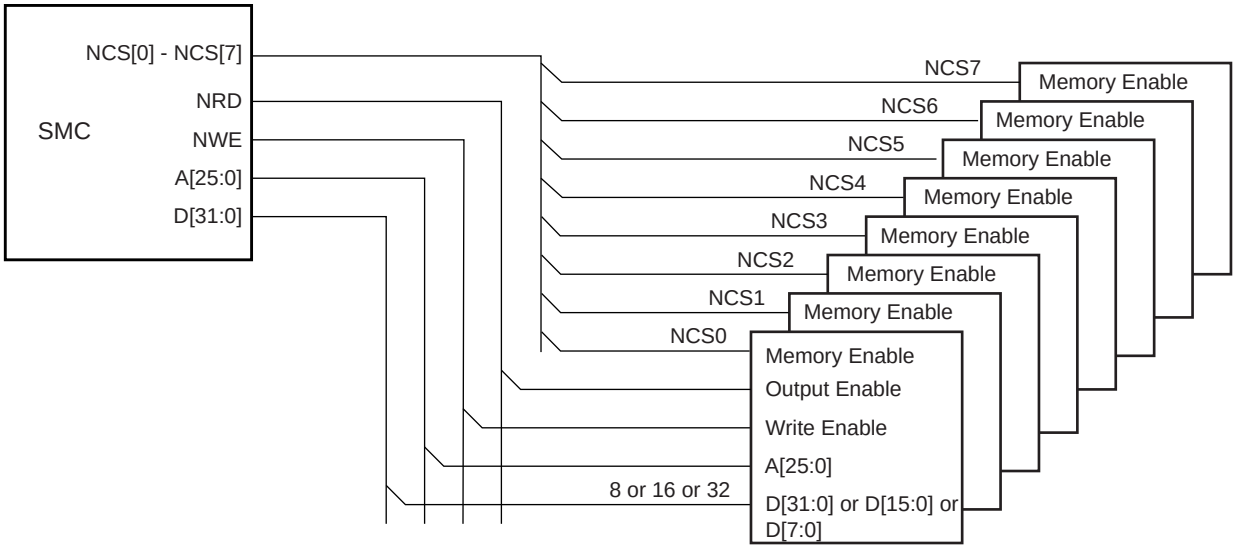
20.6 External Memory Mapping

The SMC provides up to 26 address lines, A[25:0]. This allows each chip select line to address up to 64 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 64 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 20-2).

A[25:0] is only significant for 8-bit memory, A[25:1] is used for 16-bit memory, A[25:2] is used for 32-bit memory.

Figure 20-2: Memory Connections for Eight External Devices



20.7 Connection to External Devices

20.7.1 Data Bus Width

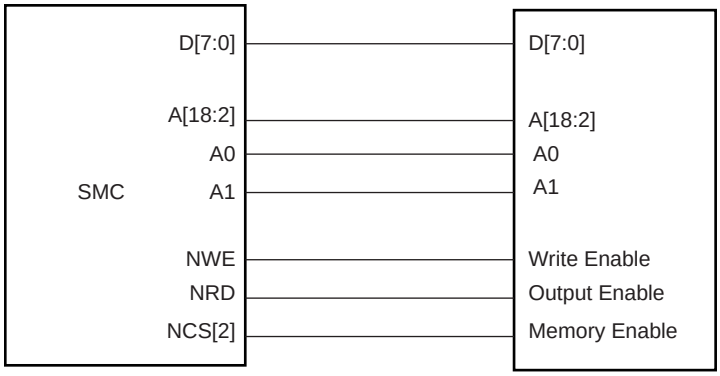
A data bus width of 8, 16, or 32 bits can be selected for each chip select. This option is controlled by the field DBW in SMC_MODE (Mode Register) for the corresponding chip select.

Figure 20-3 shows how to connect a 512K x 8-bit memory on NCS2. Figure 20-4 shows how to connect a 512K x 16-bit memory on NCS2. Figure 20-5 shows two 16-bit memories connected as a single 32-bit memory

20.7.2 Byte Write or Byte Select Access

Each chip select with a 16-bit or 32-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the BAT field of the SMC_MODE register for the corresponding chip select.

Figure 20-3: Memory Connection for an 8-bit Data Bus



© 2017 Microchip Technology Inc.

Figure 20-4: Memory Connection for a 16-bit Data Bus

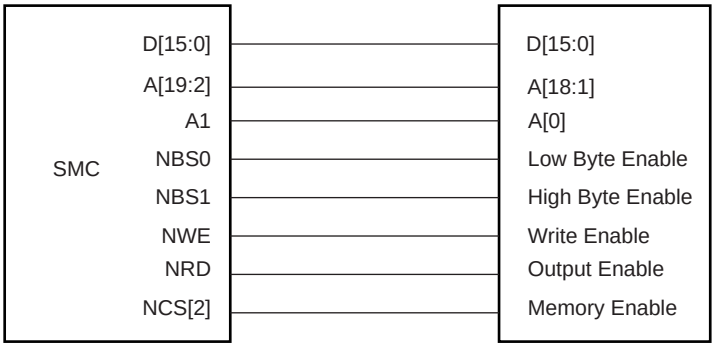
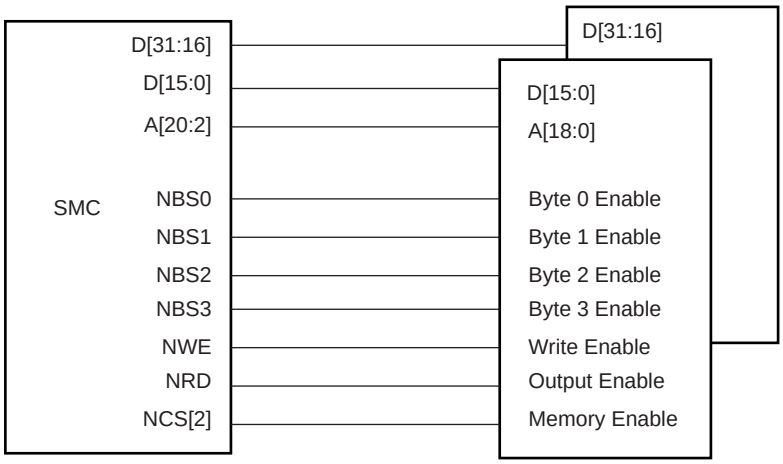


Figure 20-5: Memory Connection for a 32-bit Data Bus



20.7.2.1 Byte Write Access

Byte write access supports one byte write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in Byte Write Access mode.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 2 x 8-bit devices as a 16-bit memory.

- For 32-bit devices: NWR0, NWR1, NWR2 and NWR3, are the write signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 4 x 8-bit devices as a 32-bit memory.

  Byte Write option is illustrated on Figure 20-6.

20.7.2.2 Byte Select Access

In this mode, read/write operations can be enabled/disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

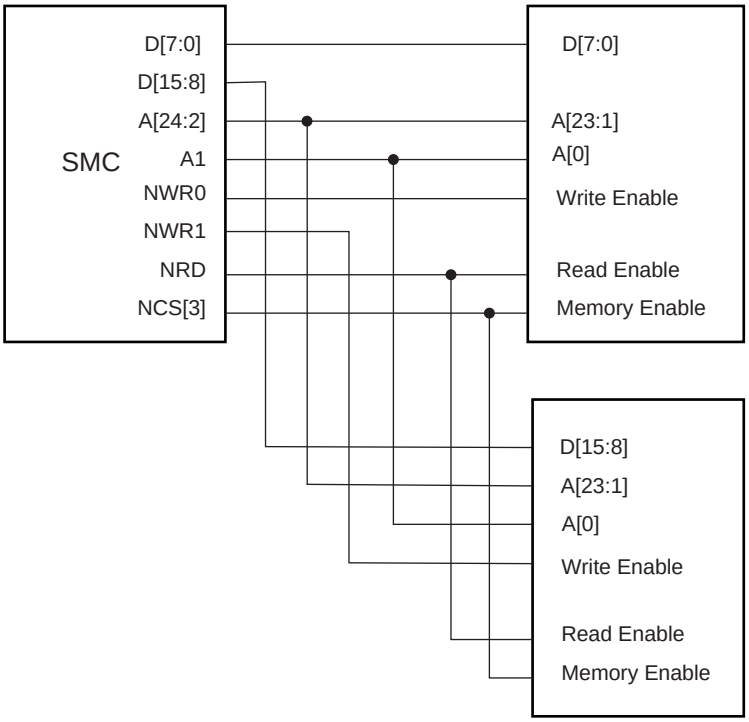
  Byte Select Access is used to connect one 16-bit device.

- For 32-bit devices: NBS0, NBS1, NBS2 and NBS3, are the selection signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. Byte Select Access is used to connect two 16-bit devices.

Figure 20-7 shows how to connect two 16-bit devices on a 32-bit data bus in Byte Select Access mode, on NCS3 (BAT = Byte Select Access).

© 2017 Microchip Technology Inc.

Figure 20-6: Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



20.7.2.3 Signal Multiplexing

Depending on the BAT, only the write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 20-3 shows signal multiplexing depending on the data bus width and the byte access type.

For 32-bit devices, bits A0 and A1 are unused. For 16-bit devices, bit A0 of address is unused. When Byte Select Option is selected, NWR1 to NWR3 are unused. When Byte Write option is selected, NBS0 to NBS3 are unused.

© 2017 Microchip Technology Inc.

Figure 20-7: Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

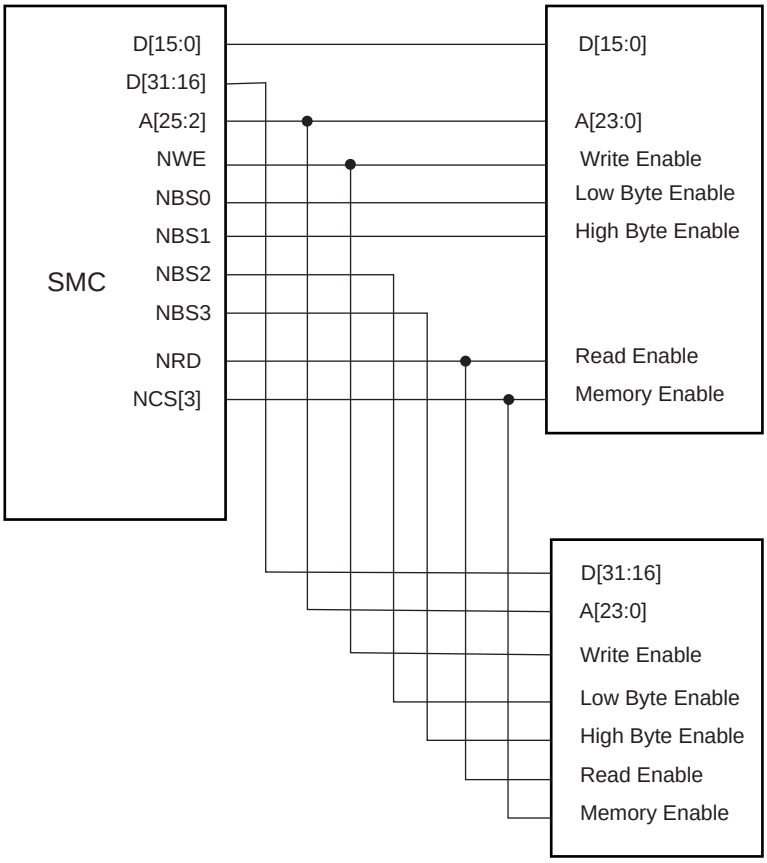


Table 20-3: SMC Multiplexed Signal Translation

Signal Name	32-bit Bus			16-bit Bus		8-bit Bus
Device Type	**1x32-bit**	**2x16-bit**	**4 x 8-bit**	**1x16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Select	Byte Write	Byte Select	Byte Write	
NBS0_A0	NBS0	NBS0		NBS0		A0
NWE_NWR0	NWE	NWE	NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NBS1	NWR1	NBS1	NWR1	
NBS2_NWR2_A1	NBS2	NBS2	NWR2	A1	A1	A1
NBS3_NWR3	NBS3	NBS3	NWR3			

20.8 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS3) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR3) in byte write access type. NWR0 to NWR3 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..7] chip select lines.

20.8.1 Read Waveforms

The read cycle is shown on Figure 20-8.

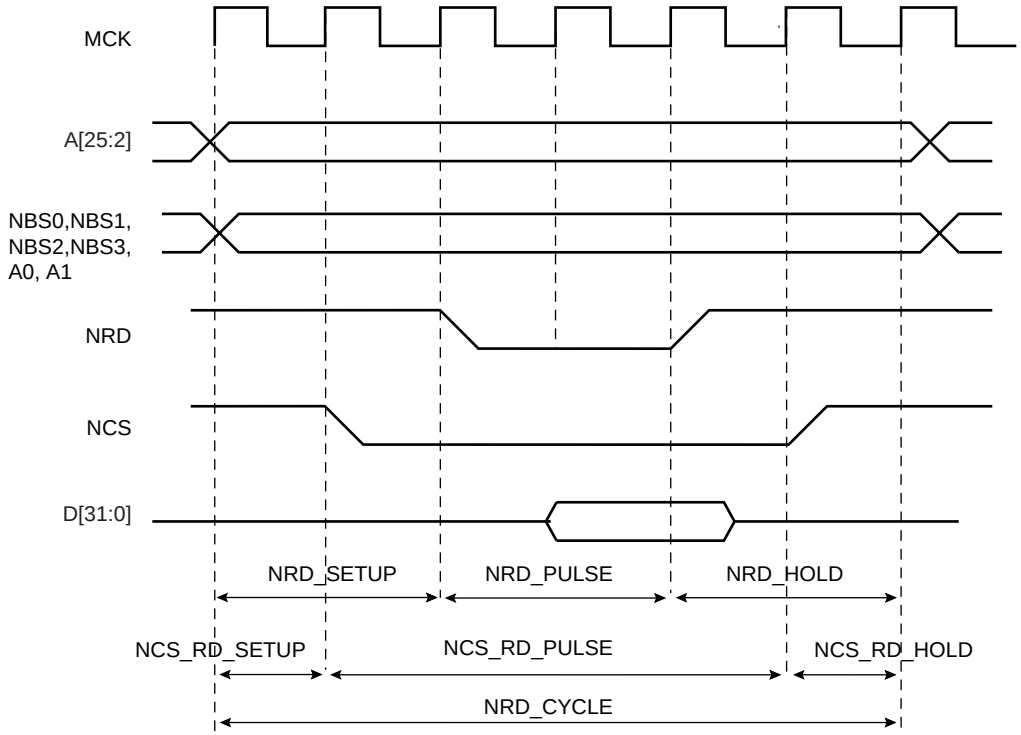
The read cycle starts with the address setting on the memory address bus, i.e.:

{A[25:2], A1, A0} for 8-bit devices

{A[25:2], A1} for 16-bit devices

A[25:2] for 32-bit devices.

Figure 20-8: Standard Read Cycle



20.8.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing.

1. NRD_SETUP: the NRD setup time is defined as the setup of address before the NRD falling edge;
2. NRD_PULSE: the NRD pulse length is the time between NRD falling edge and NRD rising edge;
3. NRD_HOLD: the NRD hold time is defined as the hold time of address after the NRD rising edge.

20.8.1.2 NCS Waveform

Similarly, the NCS signal can be divided into a setup time, pulse length and hold time:

1. NCS_RD_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCS_RD_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCS_RD_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

20.8.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is equal to:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD

= NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

© 2017 Microchip Technology Inc.

All NRD and NCS timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NRD and NCS timings are coherent, user must define the total read cycle instead of the hold timing. NRD_CYCLE implicitly defines the NRD hold time and NCS hold time as:

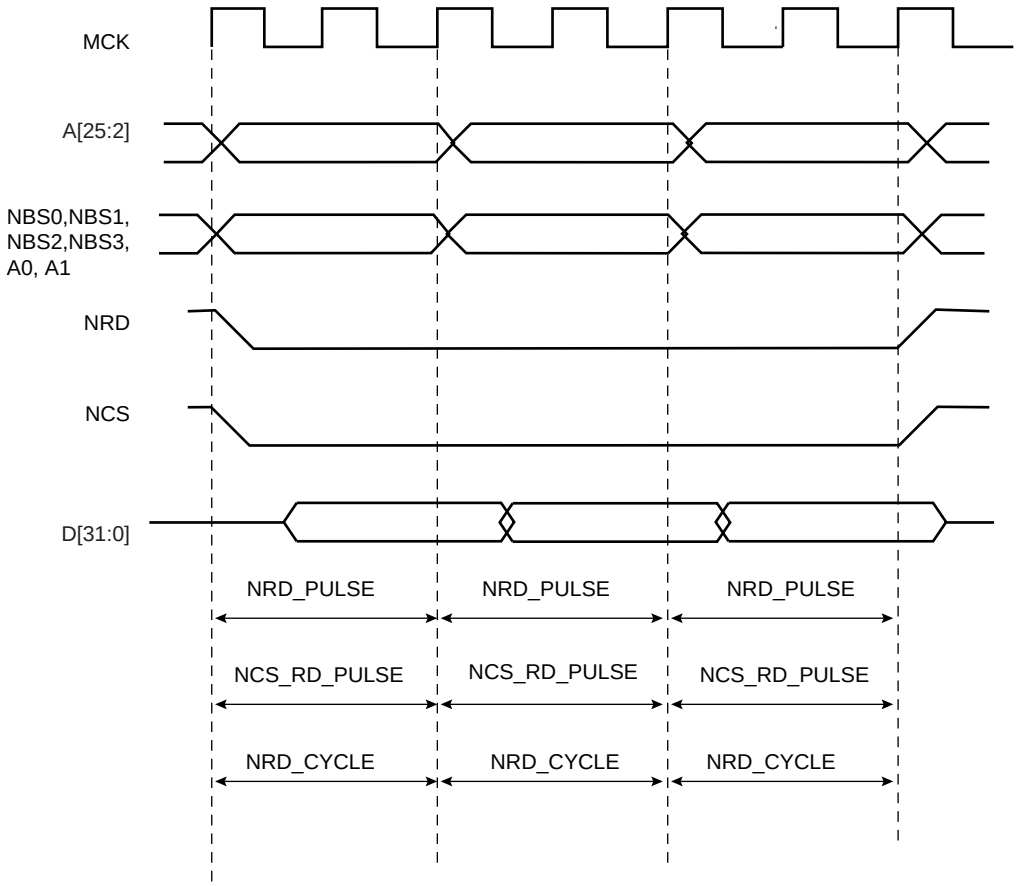
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

20.8.1.4 Null Delay Setup and Hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 20-9).

Figure 20-9: No Setup, No Hold On NRD and NCS Read Signals



20.8.1.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

20.8.2 Read Mode

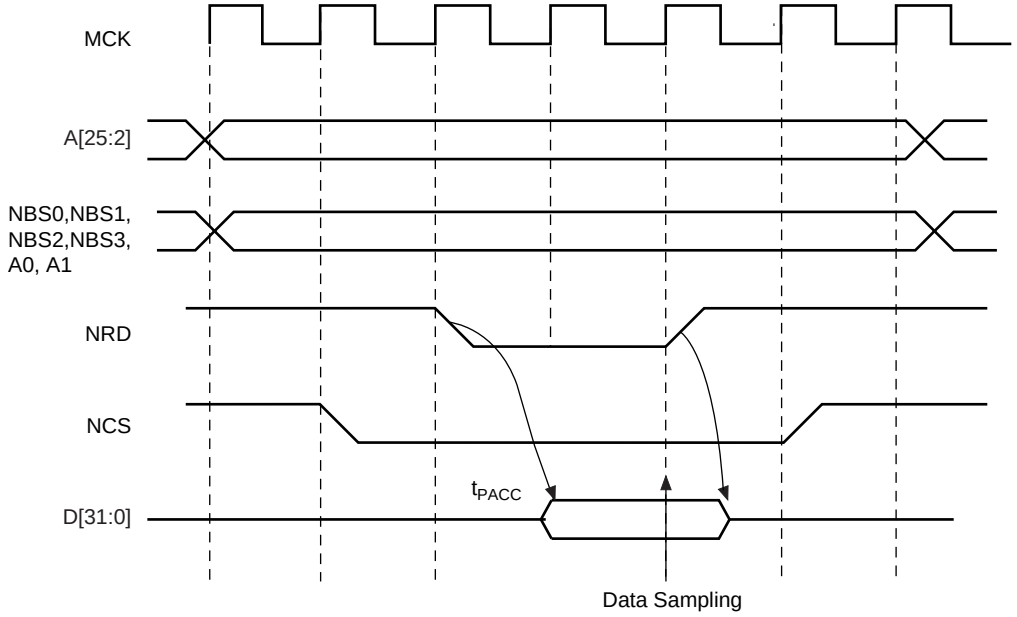
As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

20.8.2.1 Read is Controlled by NRD (READ_MODE = 1):

Figure 20-10 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to ‘Z’ after the rising edge of NRD. In this case, the READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

© 2017 Microchip Technology Inc.

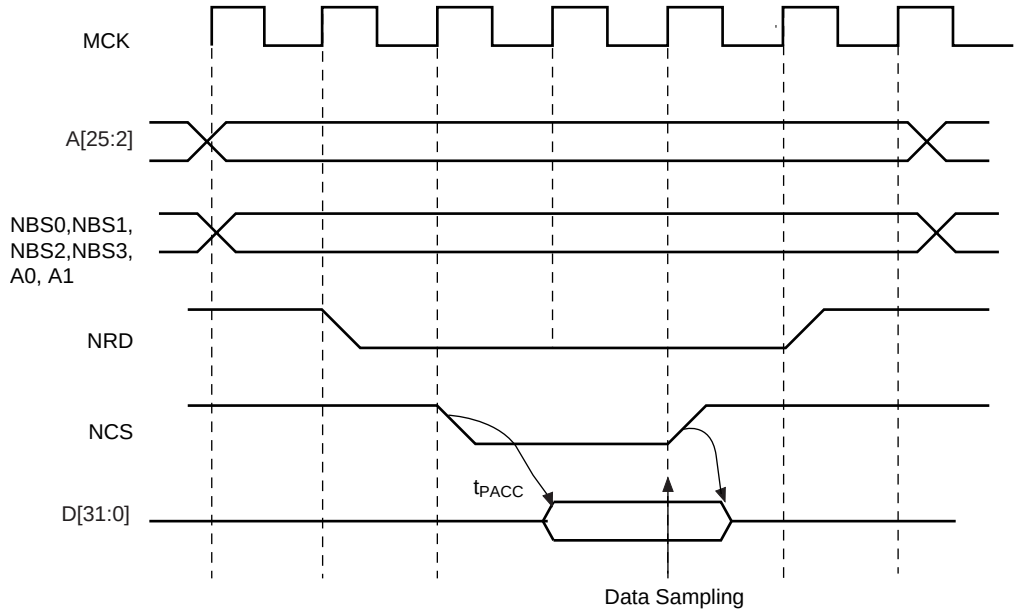
Figure 20-10: READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD



20.8.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 20-11 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the READ_MODE must be set to 0 (read is controlled by NCS): the SMC internally samples the data on the rising edge of Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 20-11: READ_MODE = 0: Data is sampled by SMC before the rising edge of NCS



© 2017 Microchip Technology Inc.

20.8.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 20-12. The write cycle starts with the address setting on the memory address bus.

20.8.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

1. NWE_SETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge;
2. NWE_PULSE: The NWE pulse length is the time between NWE falling edge and NWE rising edge;
3. NWE_HOLD: The NWE hold time is defined as the hold time of address and data after the NWE rising edge.

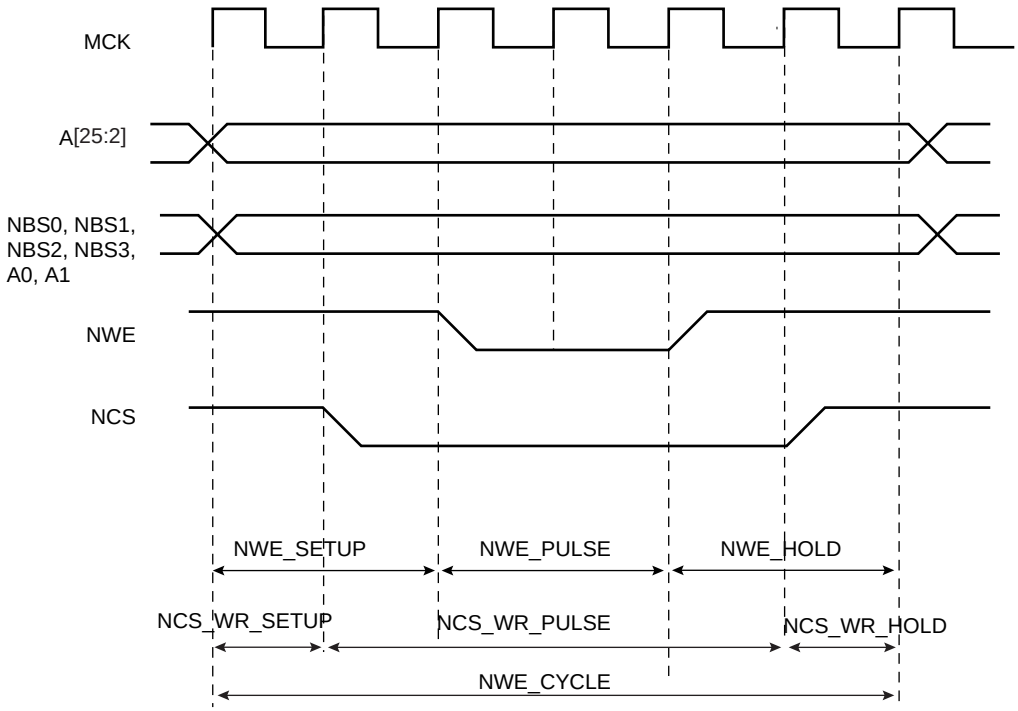
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

20.8.3.2 NCS Waveforms

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

1. NCS_WR_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCS_WR_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCS_WR_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 20-12: Write Cycle



20.8.3.3 Write Cycle

The write-cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is equal to:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD

= NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NWE and NCS timings are coherent, the user must define the total write cycle instead of the hold timing. This implicitly defines the NWE hold time and NCS (write) hold times as:

NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

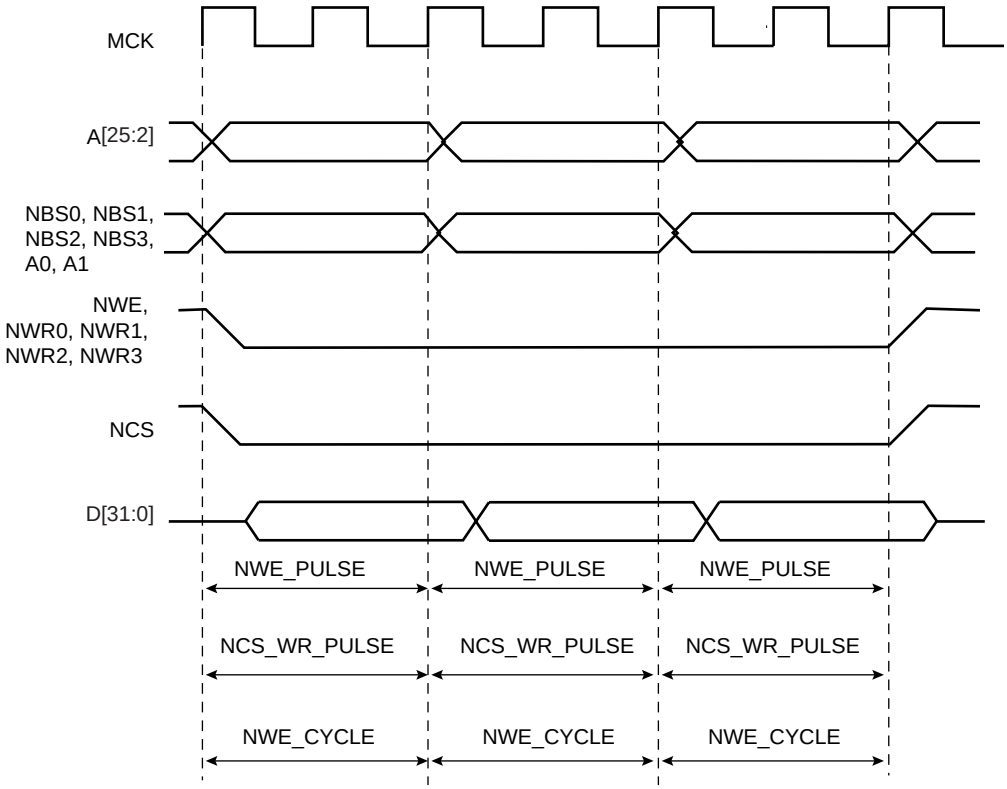
NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

© 2017 Microchip Technology Inc.

20.8.3.4 Null Delay Setup and Hold

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 20-13). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 20-13: Null Setup and Hold Values of NCS and NWE in Write Cycle



20.8.3.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

20.8.4 Write Mode

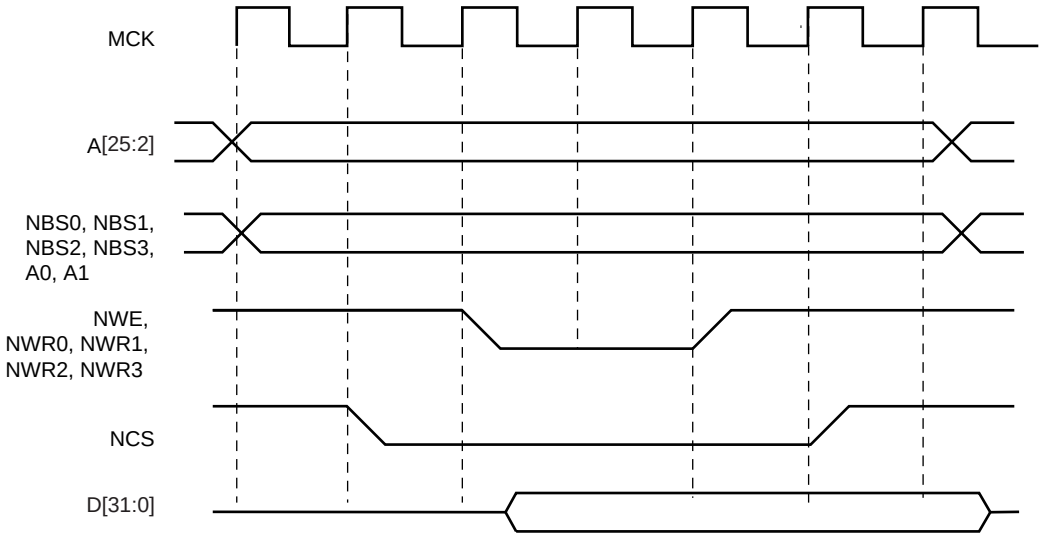
The WRITE_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal controls the write operation.

20.8.4.1 Write is Controlled by NWE (WRITE_MODE = 1):

Figure 20-14 shows the waveforms of a write operation with WRITE_MODE set to 1. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are turned out after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

© 2017 Microchip Technology Inc.

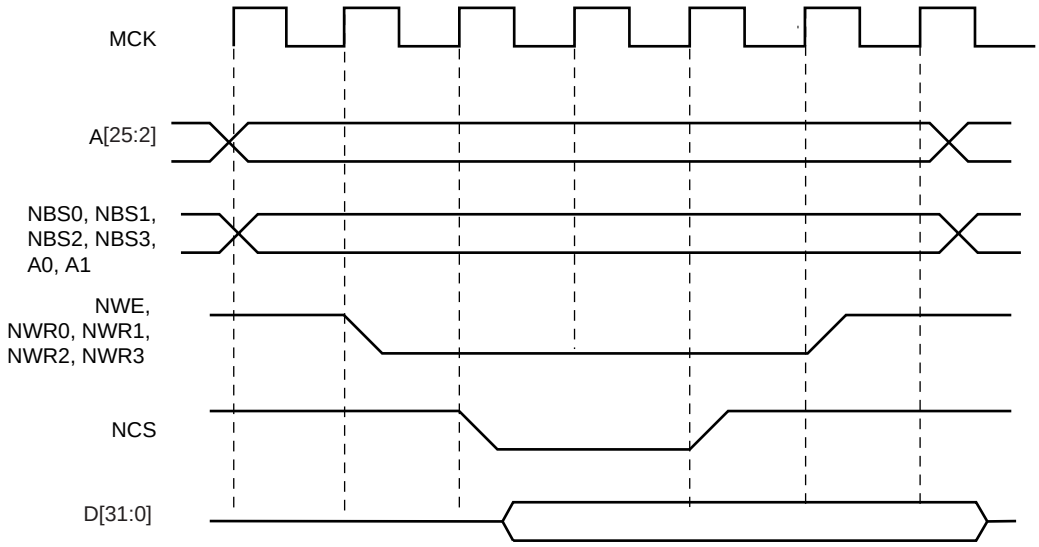
Figure 20-14: WRITE_MODE = 1. The write operation is controlled by NWE



20.8.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 20-15 shows the waveforms of a write operation with WRITE_MODE set to 0. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are turned out after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 20-15: WRITE_MODE = 0. The write operation is controlled by NCS



20.8.5 Coding Timing Parameters

All timing parameters are defined for one chip select and are grouped together in one SMC_REGISTER according to their type.

The SMC_SETUP register groups the definition of all setup parameters:

- NRD_SETUP, NCS_RD_SETUP, NWE_SETUP, NCS_WR_SETUP

The SMC_PULSE register groups the definition of all pulse parameters:

- NRD_PULSE, NCS_RD_PULSE, NWE_PULSE, NCS_WR_PULSE

The SMC_CYCLE register groups the definition of all cycle parameters:

- NRD_CYCLE, NWE_CYCLE

Table 20-4 shows how the timing parameters are coded and their permitted range.

Table 20-4: Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤ 31	128 ≤ 128 + 31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤ 63	256 ≤ 256 + 63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤ 127	256 ≤ 256 + 127 512 ≤ 512 + 127 768 ≤ 768 + 127

20.8.6 Reset Values of Timing Parameters

Table 20-8 “Register Mapping” gives the default value of timing parameters at reset.

20.8.7 Usage Restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

For read operations:

Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true for WRITE_MODE = 1 only. See Section 20.9.2 “Early Read Wait State”.

For read and write operations: a null value for pulse parameters is forbidden and may lead to unpredictable behavior.

In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

20.9 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

20.9.1 Chip Select Wait States

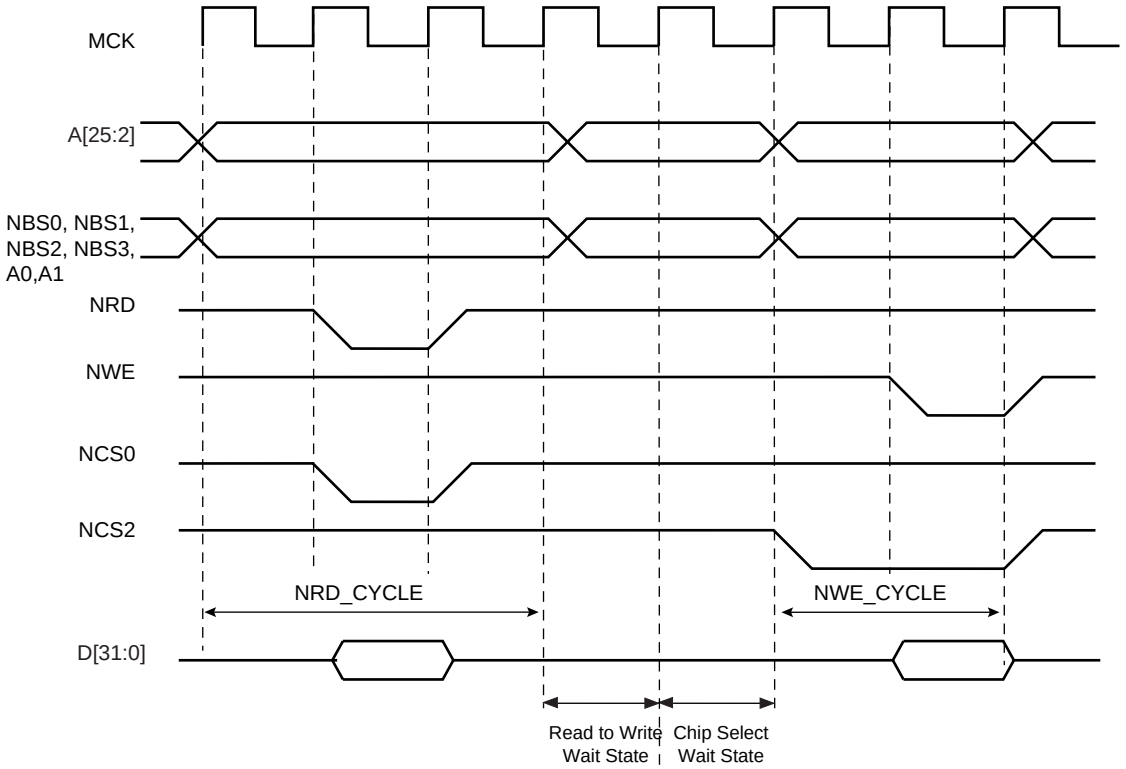
The SMC always inserts an idle cycle between 2 transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..7], NRD lines are all set to 1.

Figure 20-16 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

© 2017 Microchip Technology Inc.

Figure 20-16: Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



20.9.2 Early Read Wait State

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 20-17).
- in NCS write controlled mode (WRITE_MODE = 0), if there is no hold timing on the NCS signal and the NCS_RD_SETUP parameter is set to 0, regardless of the read mode (Figure 20-18). The write operation must end with a NCS rising edge. Without an Early Read Wait State, the write operation could not complete properly.
- in NWE controlled mode (WRITE_MODE = 1) and if there is no hold timing (NWE_HOLD = 0), the feedback of the write control signal is used to control address, data, chip select and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an Early Read Wait State is inserted and address, data and control signals are maintained one more cycle. See Figure 20-19.

Figure 20-17: Early Read Wait State: Write with No Hold Followed by Read with No Setup

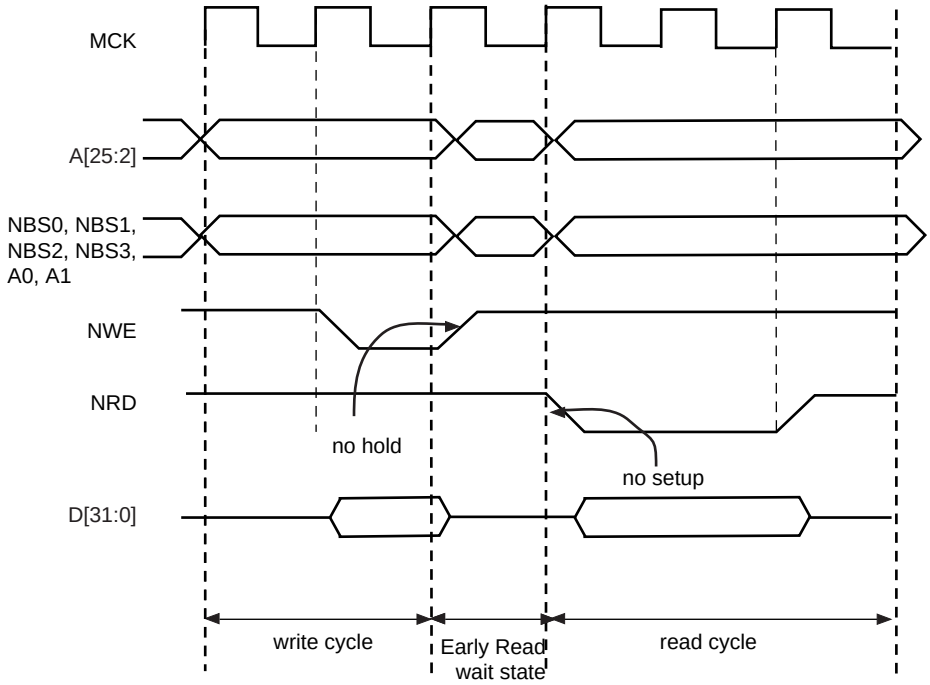
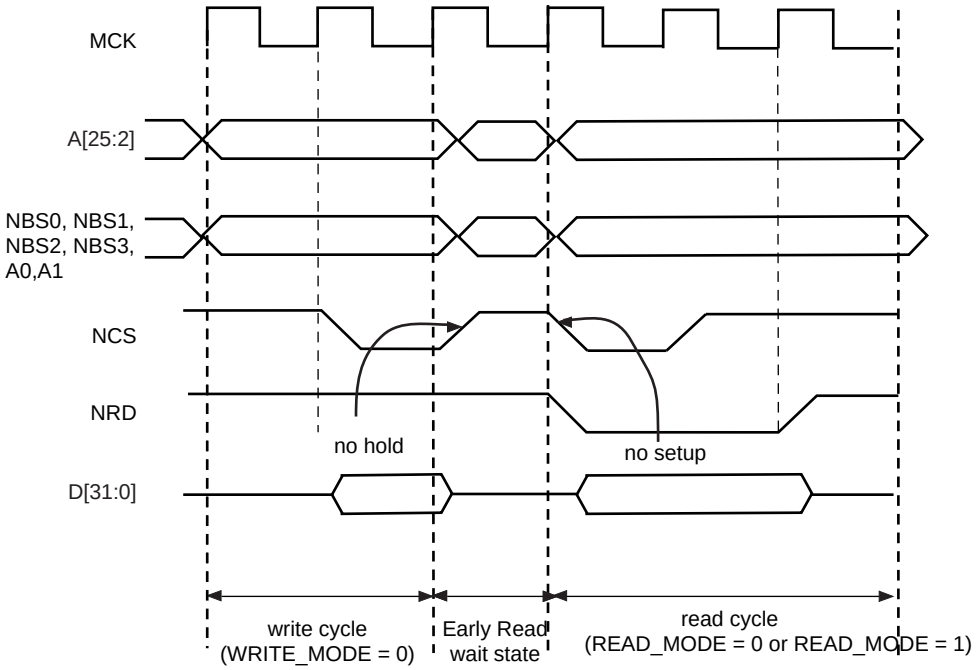
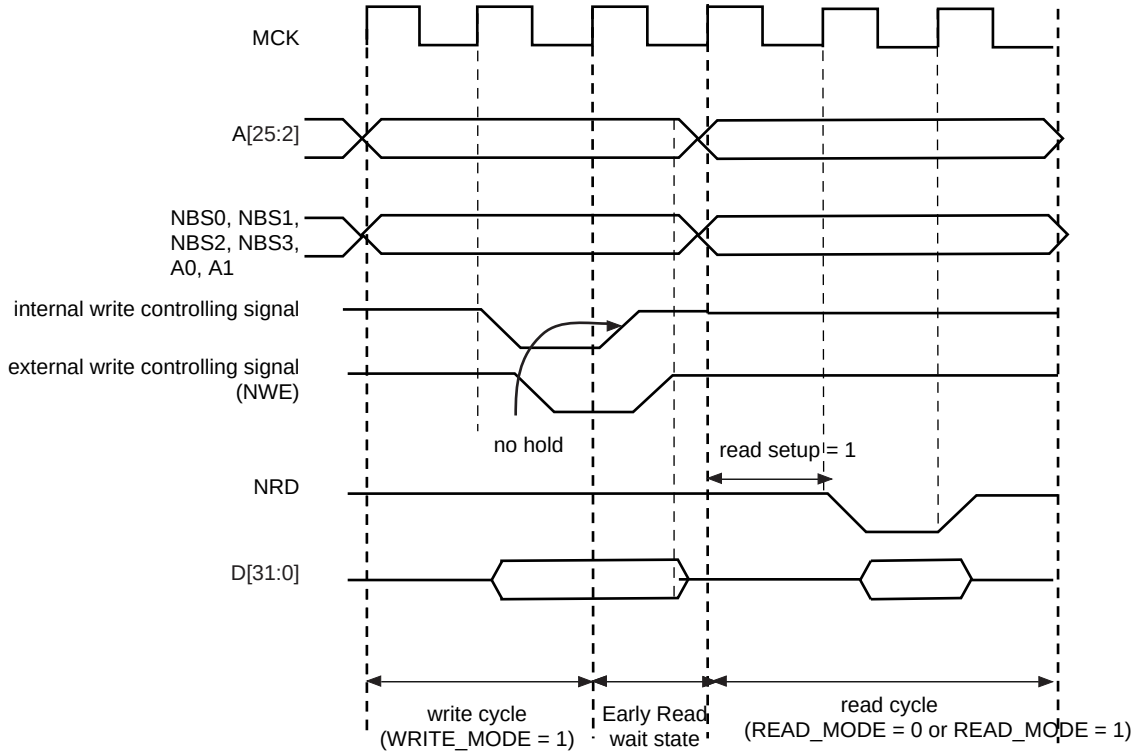


Figure 20-18: Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read, No NCS Setup



© 2017 Microchip Technology Inc.

Figure 20-19: Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read, 1 Set-up Cycle



20.9.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called “Reload User Configuration Wait State” is used by the SMC to load the new set of parameters to apply to next accesses.

The Reload Configuration Wait State is not applied in addition to the Chip Select Wait State. If accesses before and after re-programming the user interface are made to different devices (Chip Selects), then one single Chip Select Wait State is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a Reload Configuration Wait State is inserted, even if the change does not concern the current Chip Select.

20.9.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

20.9.3.2 Slow Clock Mode Transition

A Reload Configuration Wait State is also inserted when the Slow Clock Mode is entered or exited, after the end of the current transfer (see Section 20.12 “Slow Clock Mode”).

20.9.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 20-16.

20.10 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF_CYCLES field of the SMC_MODE register for the corresponding chip select. The value of TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on the READ_MODE and the TDF_MODE fields of the SMC_MODE register for the corresponding chip select.

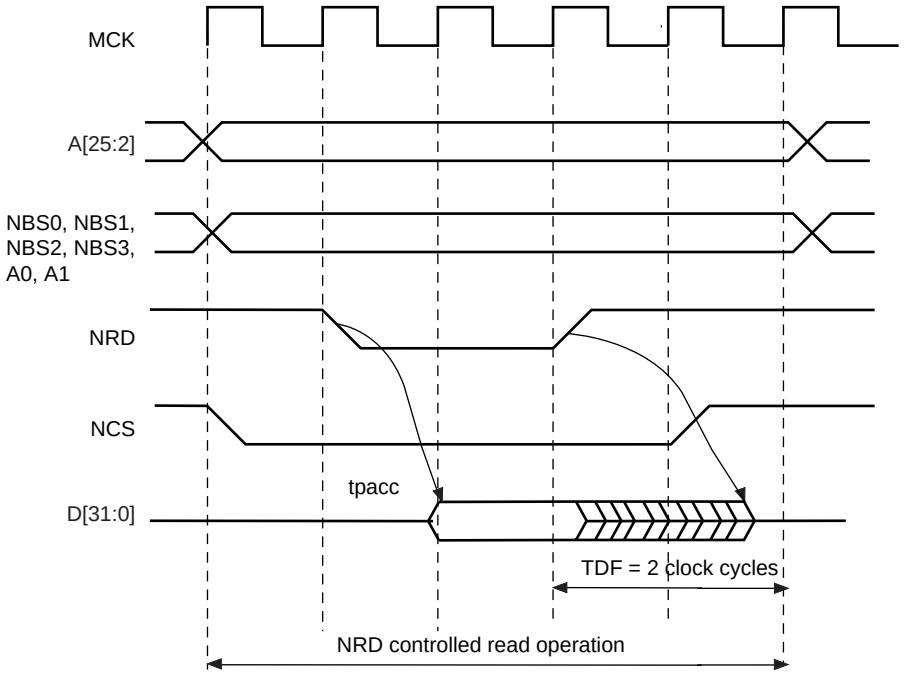
20.10.1 READ_MODE

Setting the READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

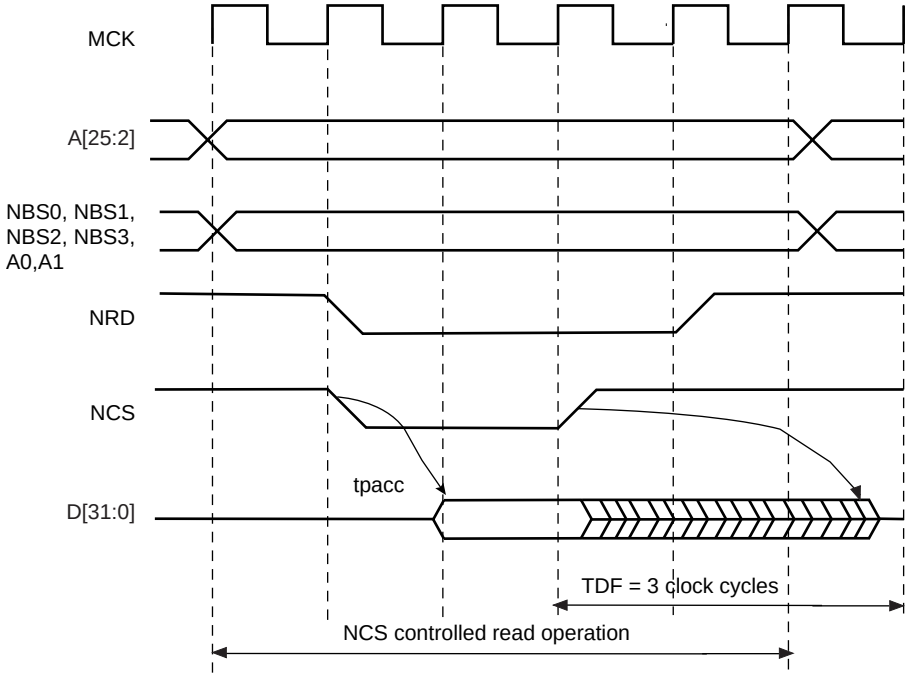
Figure 20-20 illustrates the Data Float Period in NRD-controlled mode (READ_MODE = 1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 20-21 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 20-20: TDF Period in NRD Controlled Read Access (TDF = 2)



© 2017 Microchip Technology Inc.

Figure 20-21: TDF Period in NCS Controlled Read Operation (TDF = 3)



20.10.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the SMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

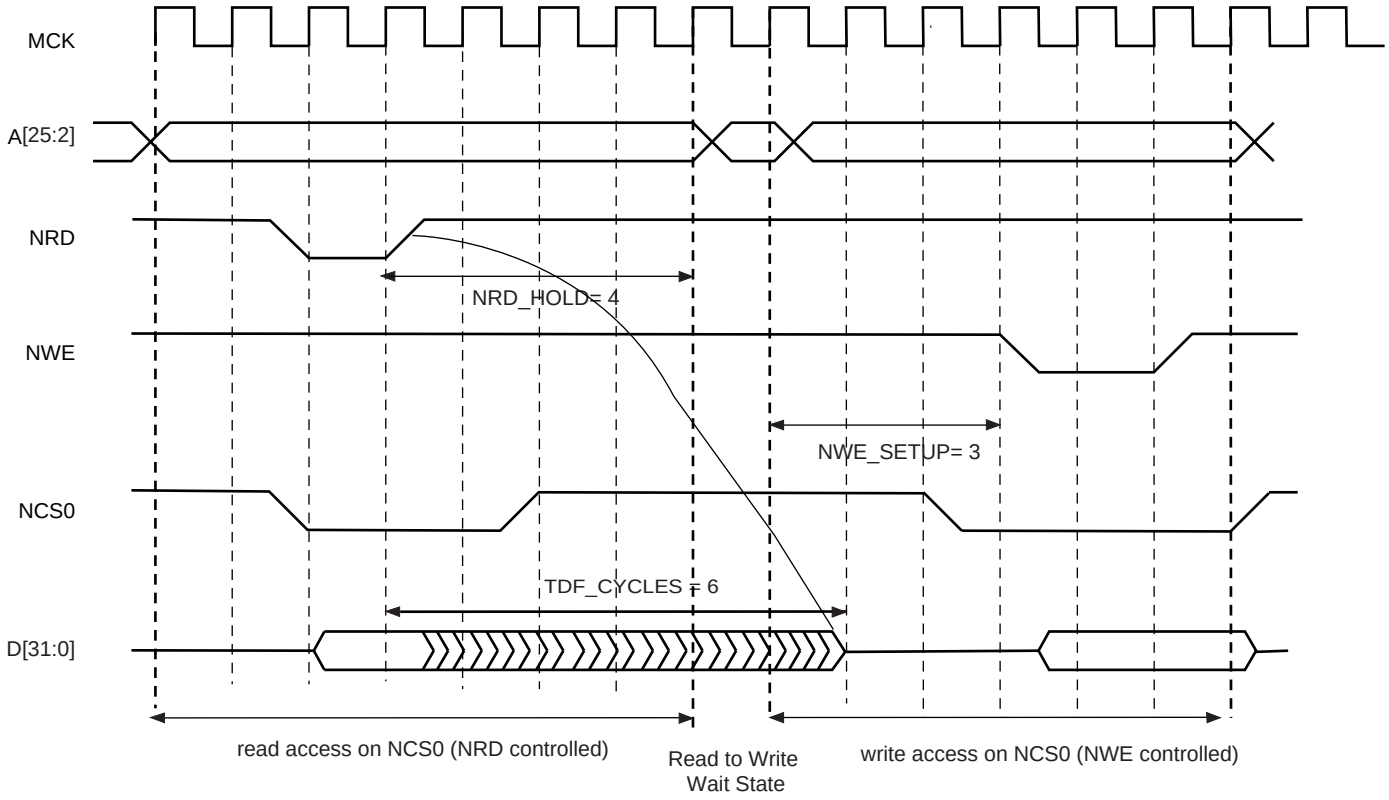
Figure 20-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

Figure 20-22: TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



20.10.3 TDF Optimization Disabled (TDF_MODE = 0)

When optimization is disabled, tdf wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional tdf wait states will be inserted.

Figure 20-23, Figure 20-24 and Figure 20-25 illustrate the cases:

- read access followed by a read access on another chip select,
- read access followed by a write access on another chip select,
- read access followed by a write access on the same chip select,

with no TDF optimization.

© 2017 Microchip Technology Inc.

Figure 20-23: TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

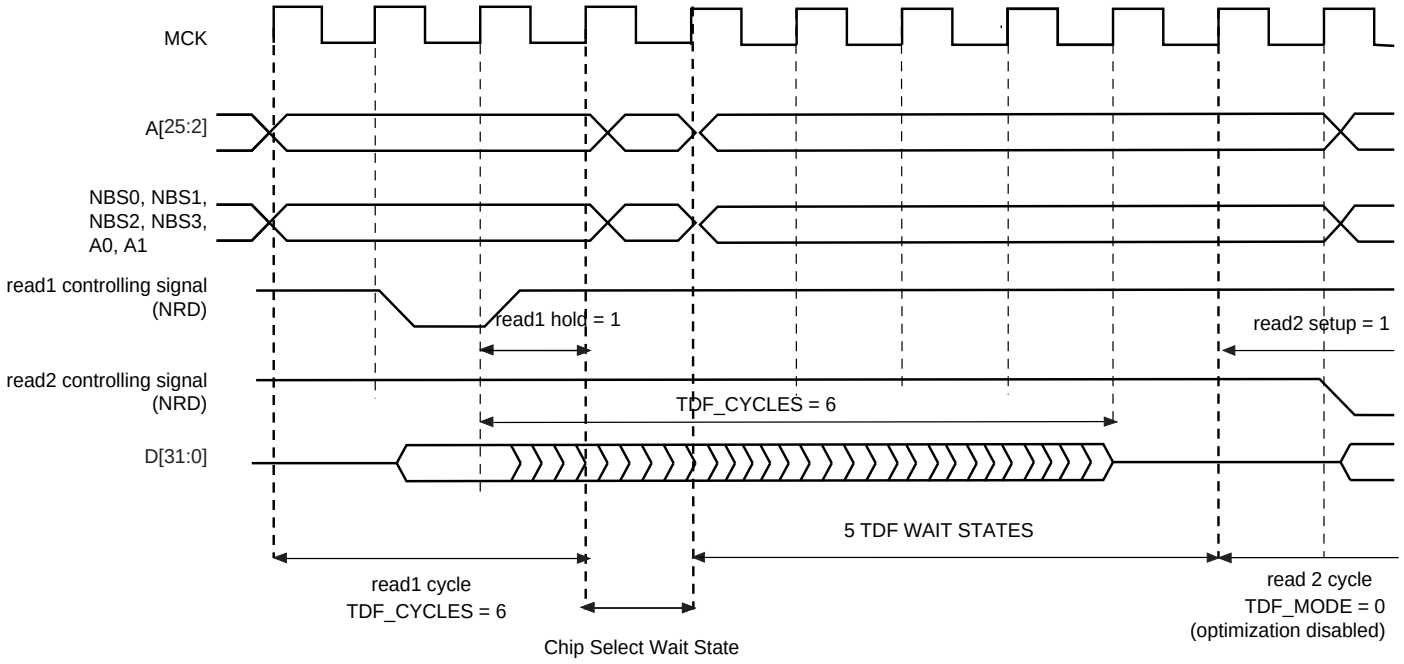


Figure 20-24: TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

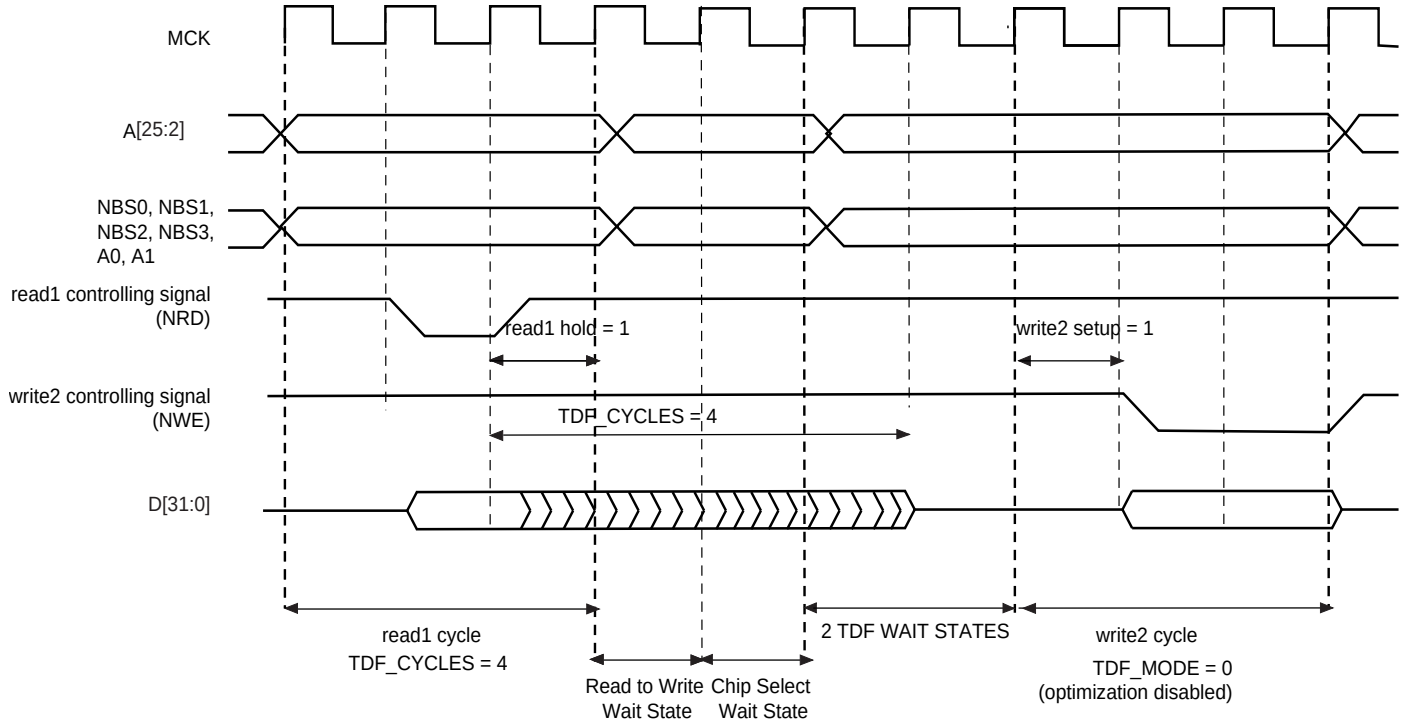
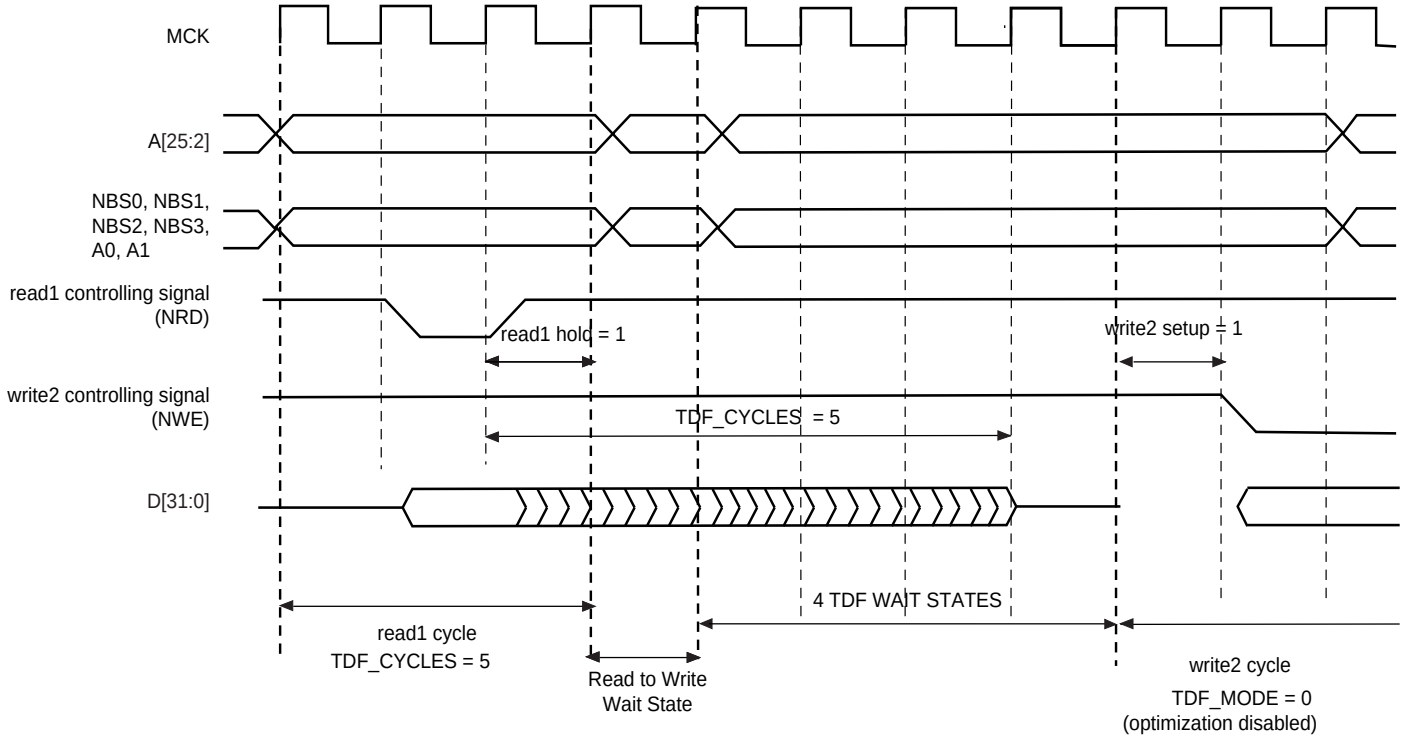


Figure 20-25: TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



20.11 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The EXNW_MODE field of the SMC_MODE register on the corresponding chip select must be set to either to “10” (frozen mode) or “11” (ready mode). When the EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

20.11.1 Restriction

When one of the EXNW_MODE is enabled, it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (“Asynchronous Page Mode”), or in Slow Clock Mode (“Slow Clock Mode”).

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

20.11.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 20-26. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 20-27.

© 2017 Microchip Technology Inc.

Figure 20-26: Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

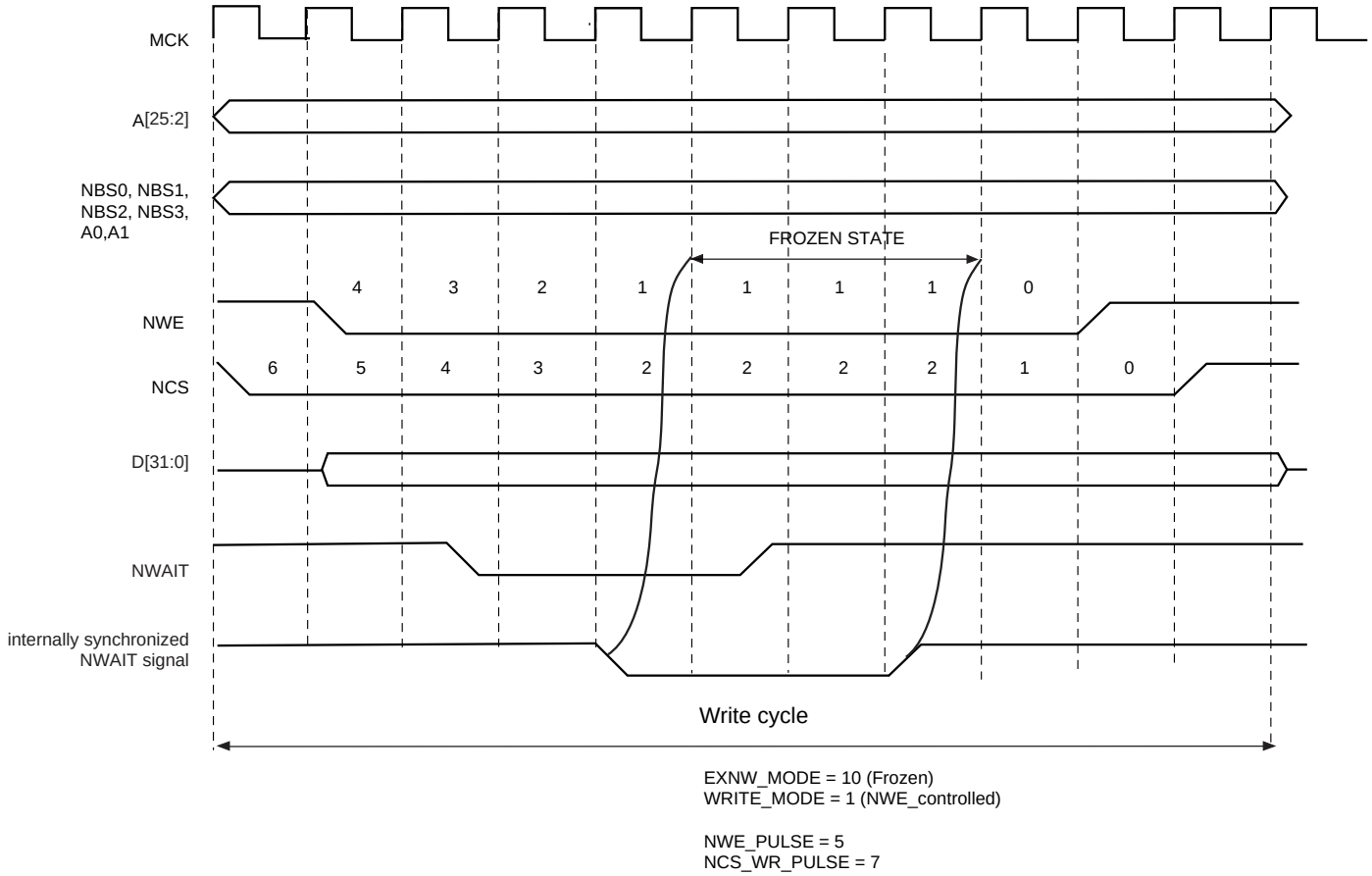
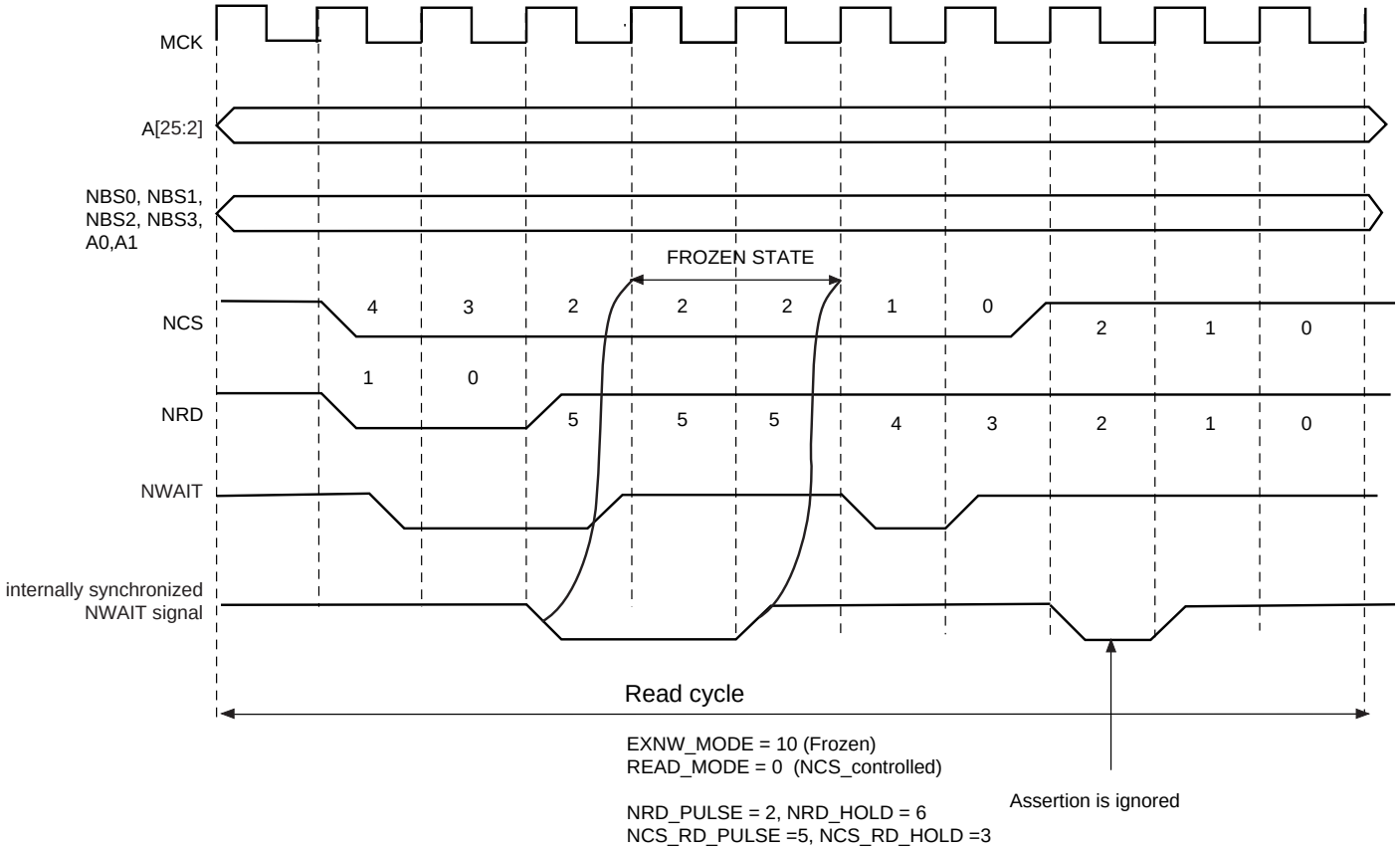


Figure 20-27: Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



© 2017 Microchip Technology Inc.

20.11.3 Ready Mode

In Ready mode (EXNW_MODE = 11), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 20-28 and Figure 20-29. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 20-29.

Figure 20-28: NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

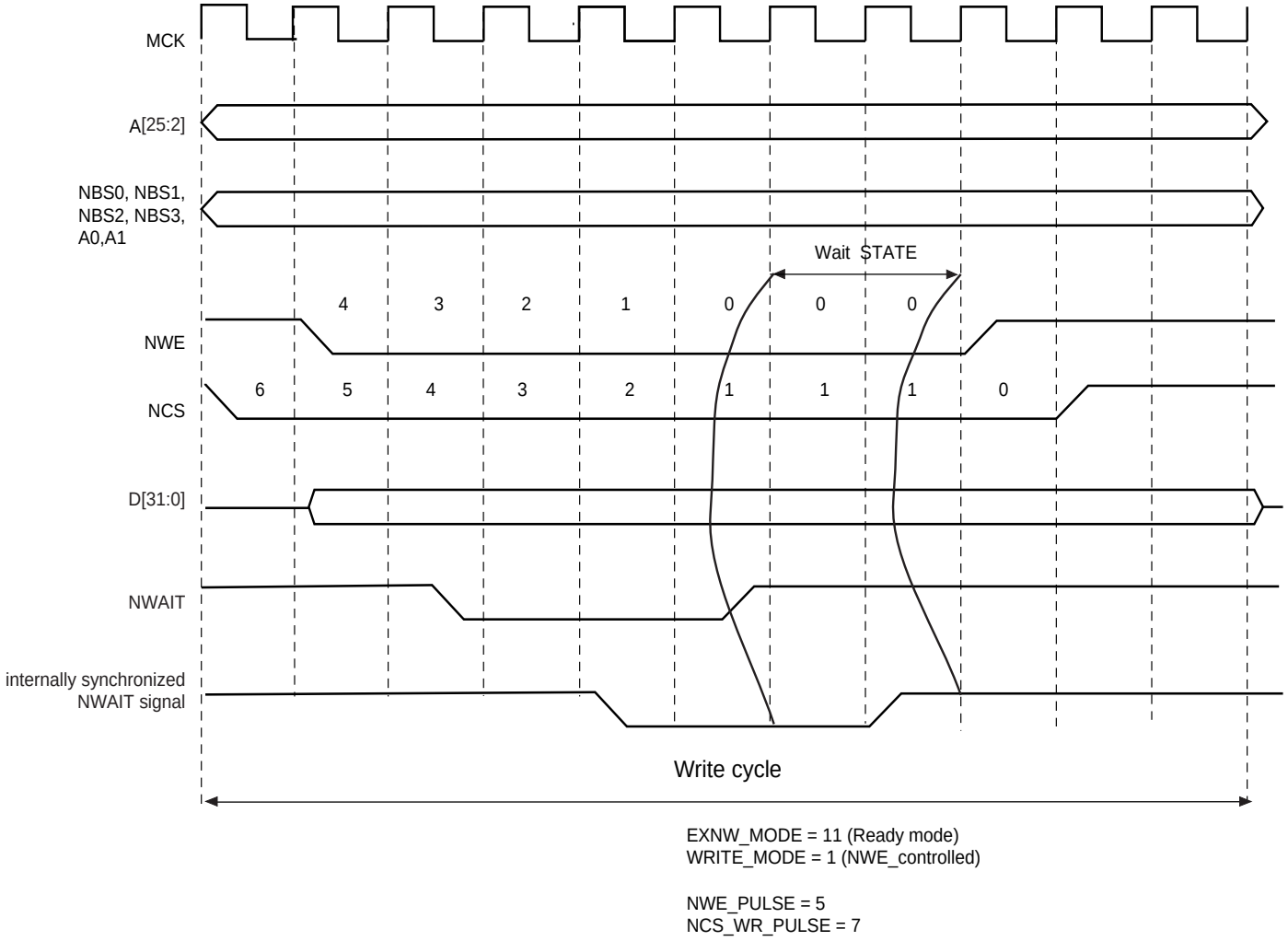
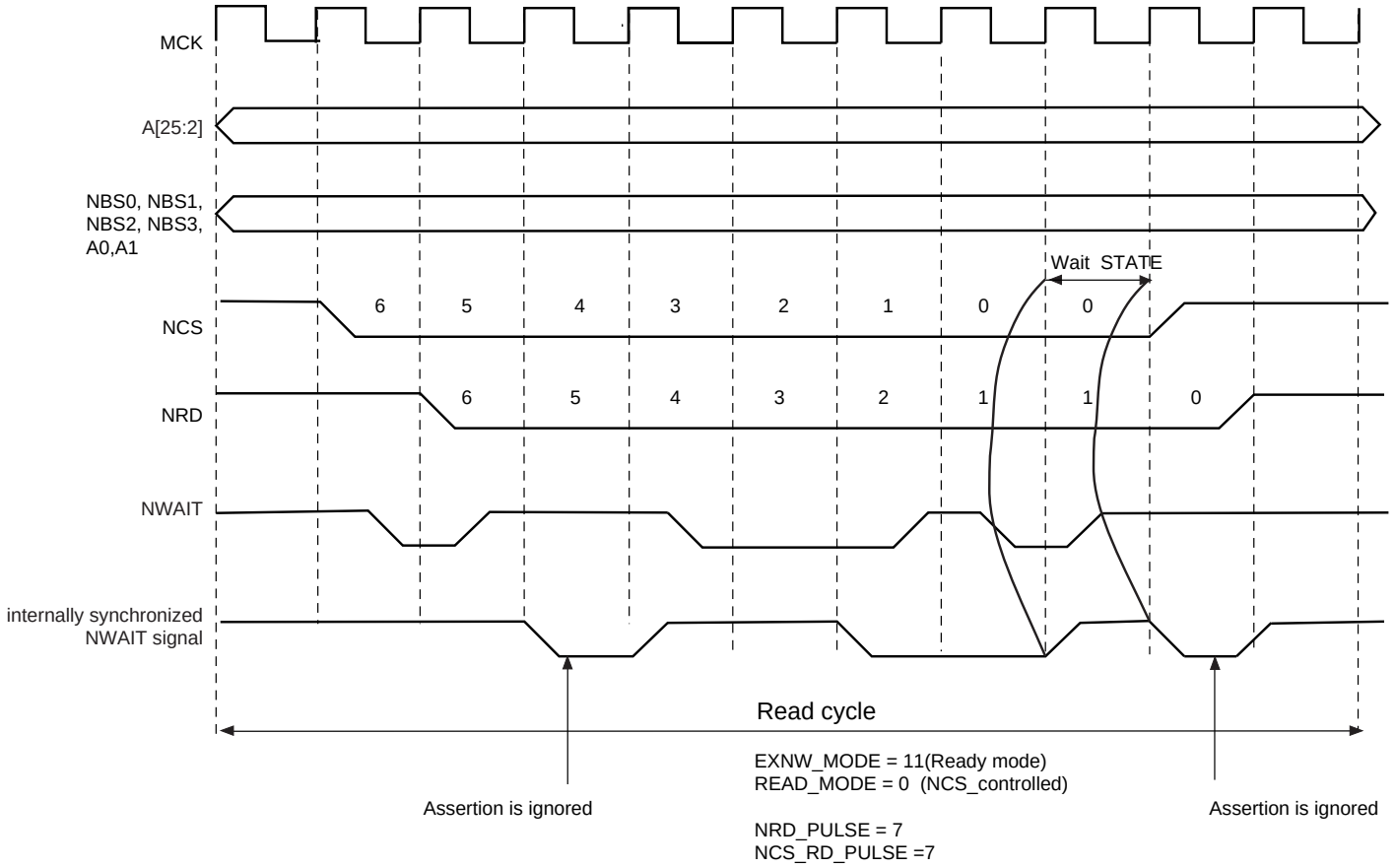


Figure 20-29: NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



© 2017 Microchip Technology Inc.

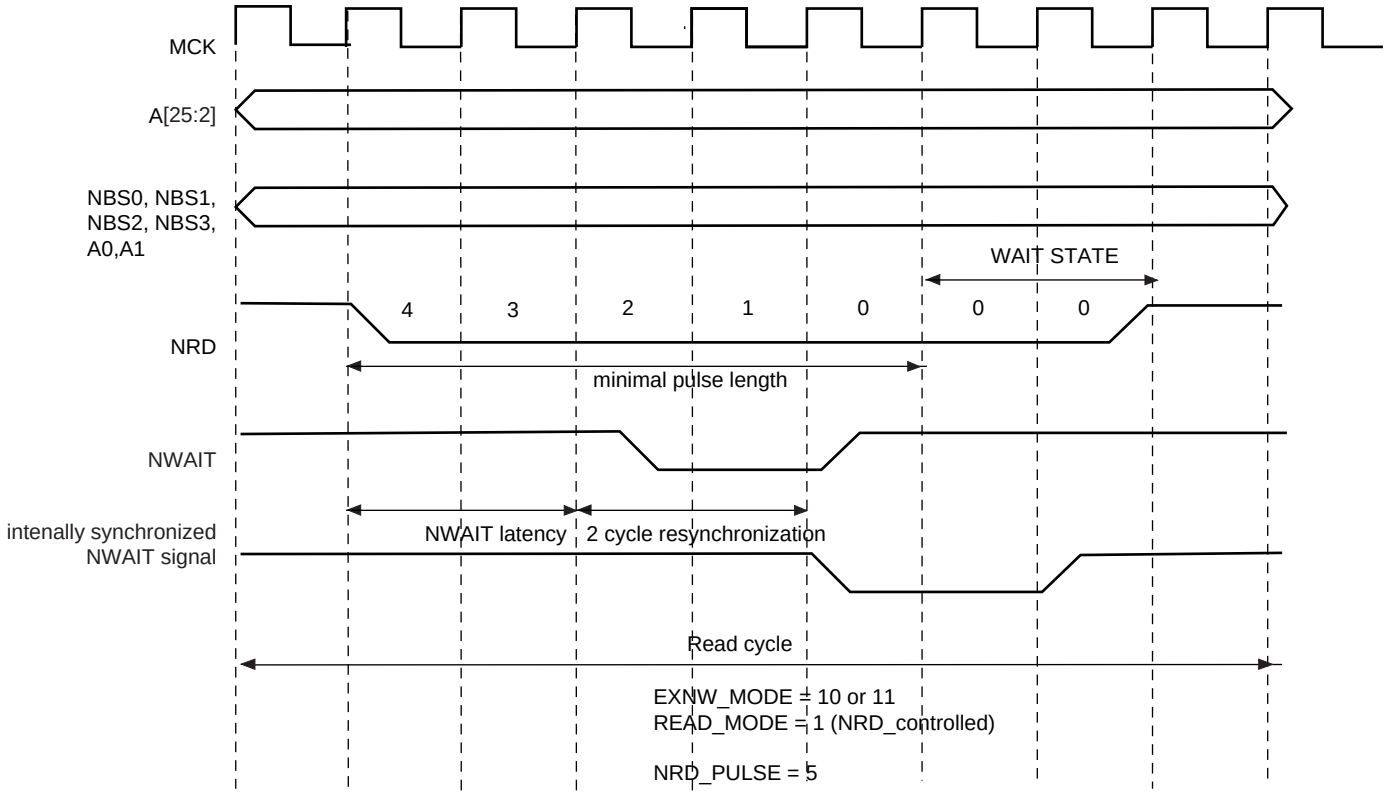
20.11.4 NWAIT Latency and Read/Write Timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the 2 cycles of resynchronization + 1 cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 20-30.

When EXNW_MODE is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

minimal pulse length = NWAIT latency + 2 resynchronization cycles + 1 cycle

Figure 20-30: NWAIT Latency



20.12 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32 kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

20.12.1 Slow Clock Mode Waveforms

Figure 20-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 20-5 indicates the value of read and write parameters in slow clock mode.

Figure 20-31: Read/Write Cycles in Slow Clock Mode

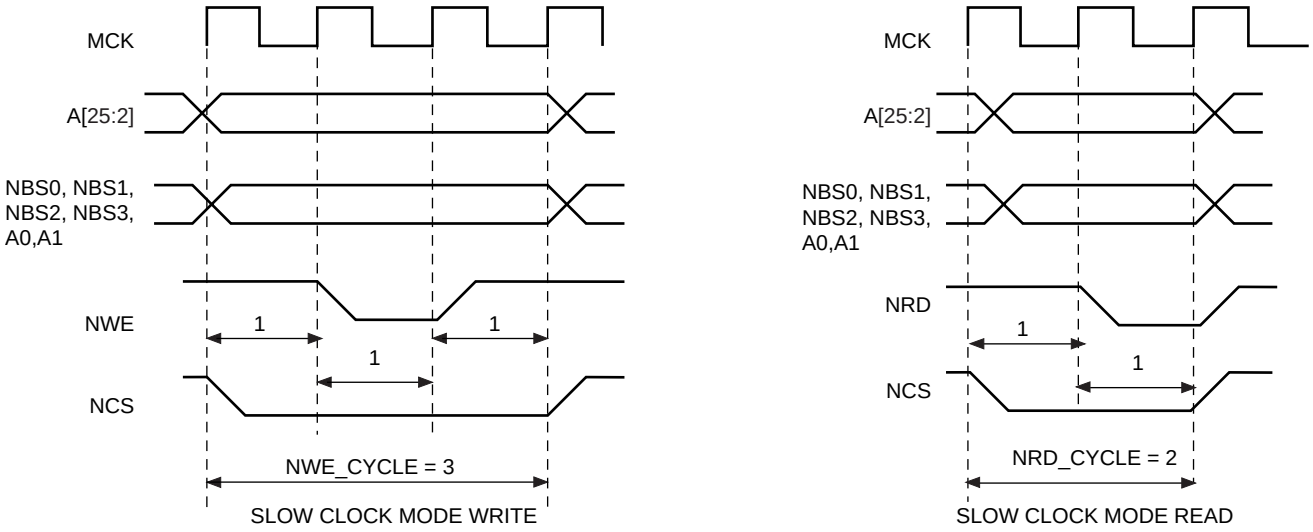


Table 20-5: Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

20.12.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters. See Figure 20-32. The external device may not be fast enough to support such timings.

Figure 20-33 illustrates the recommended procedure to properly switch from one mode to the other.

© 2017 Microchip Technology Inc.

Figure 20-32: Clock Rate Transition Occurs while the SMC is Performing a Write Operation

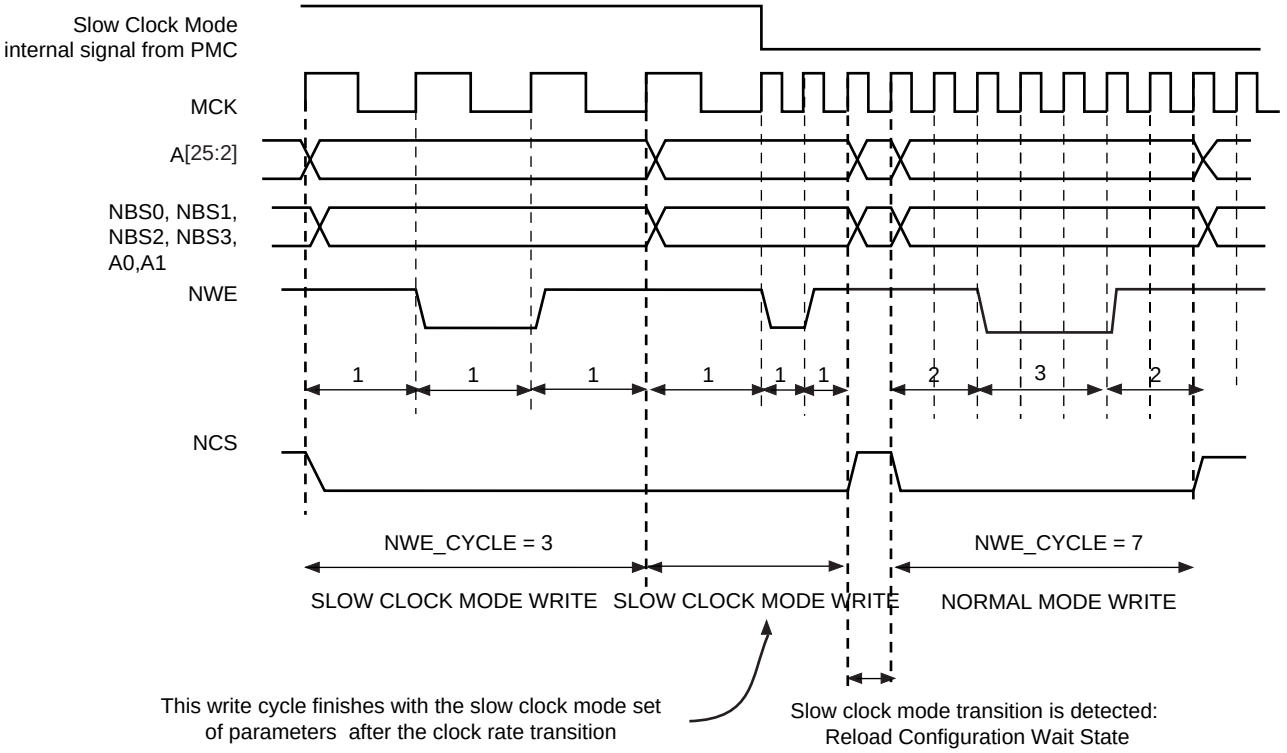
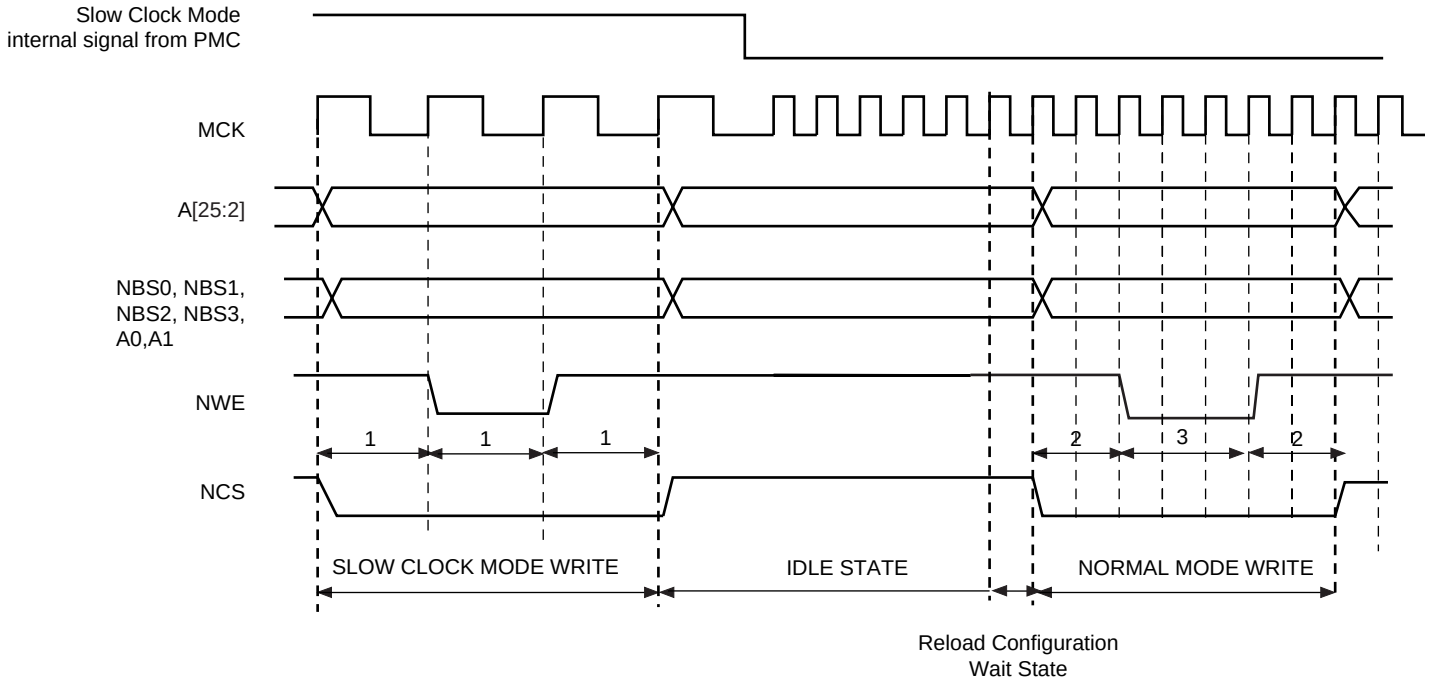


Figure 20-33: Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



20.13 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the page mode is enabled in the SMC_MODE register (PMEN field). The page size must be configured in the SMC_MODE register (PS field) to 4, 8, 16 or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 20-6.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 20-34. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 20-6: Page Address and Data Address within a Page

Page Size	Page Address(1)	Data Address in the Page(2)
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

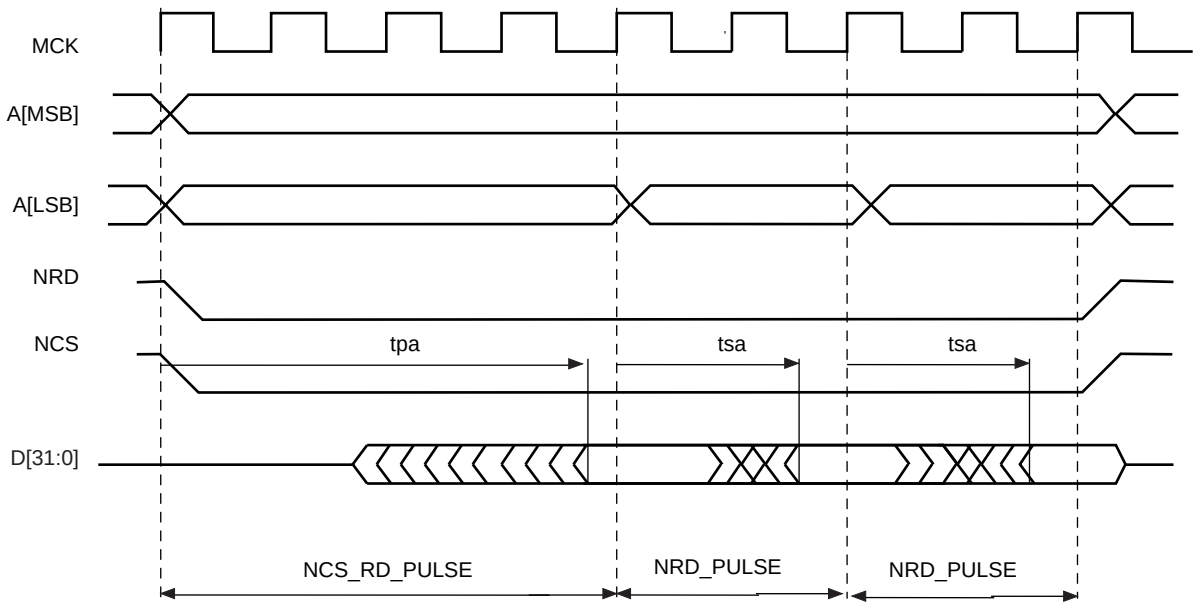
Note 1: A denotes the address bus of the memory device

2: For 16-bit devices, the bit 0 of address is ignored. For 32-bit devices, bits [1:0] are ignored.

20.13.1 Protocol and Timings in Page Mode

Figure 20-34 shows the NRD and NCS timings in page mode access.

Figure 20-34: Page Mode Read Protocol (Address MSB and LSB are defined in Table 20-6)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the NCS_RD_PULSE field of the SMC_PULSE register. The pulse length of subsequent accesses within the page are defined using the NRD_PULSE parameter.

In page mode, the programming of the read timings is described in Table 20-7:

© 2017 Microchip Technology Inc.

Table 20-7: Programming of Read Timings in Page Mode

Parameter	Value	Definition
READ_MODE	'x'	No impact
NCS_RD_SETUP	'x'	No impact
NCS_RD_PULSE	t_{pa}	Access time of first access to the page
NRD_SETUP	'x'	No impact
NRD_PULSE	t_{sa}	Access time of subsequent accesses in the page
NRD_CYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCS_RD_PULSE timings as page access timing (t_{pa}) and the NRD_PULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

20.13.2 Byte Access Type in Page Mode

The Byte Access Type configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, configure the BAT field of the SMC_REGISTER to 0 (byte select access type).

20.13.3 Page Mode Restriction

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

20.13.4 Sequential and Non-sequential Accesses

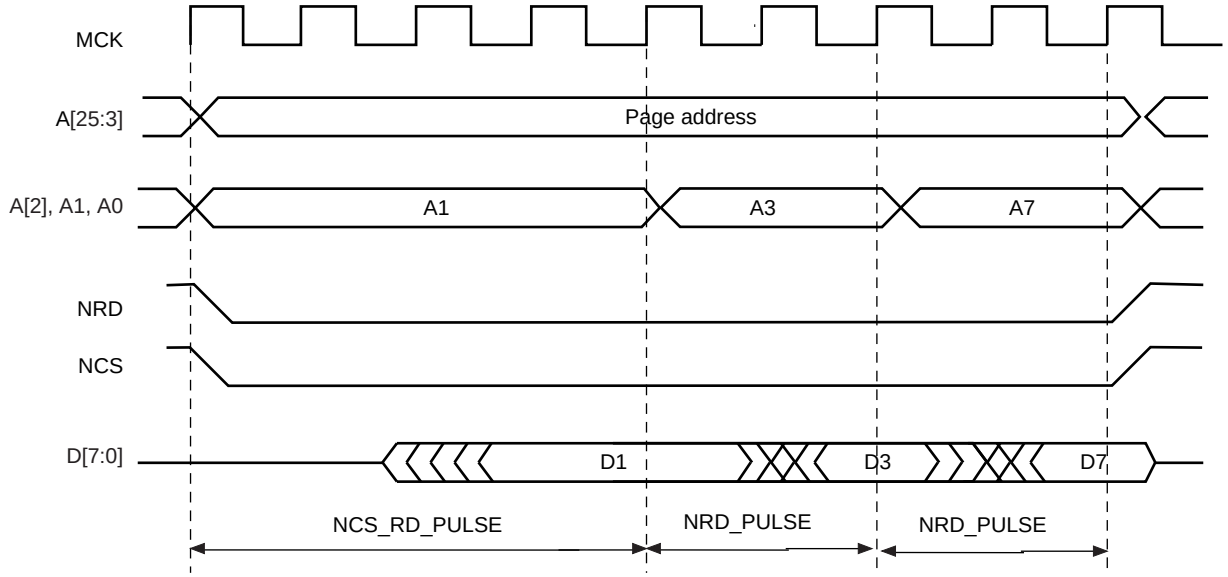
If the chip select and the MSB of addresses as defined in Table 20-6 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 20-35 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

© 2017 Microchip Technology Inc.

Figure 20-35: Access to Non-sequential Data within the Same Page



© 2017 Microchip Technology Inc.

20.14 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 20-8. For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In Table 20-8, “CS_number” denotes the chip select number. 16 bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing any one of the SMC_MODE registers.

Table 20-8: Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x00000000
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/Write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00030003
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	0x10001000

© 2017 Microchip Technology Inc.

20.14.1 SMC Setup Register

Name:SMC_SETUP[0 ..7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

NWE_SETUP: NWE Setup Length

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

NCS_WR_SETUP: NCS Setup Length in WRITE Access

In write access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

NRD_SETUP: NRD Setup Length

The NRD signal setup length is defined in clock cycles as:

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

NCS_RD_SETUP: NCS Setup Length in READ Access

In read access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

© 2017 Microchip Technology Inc.

20.14.2 SMC Pulse Register

Name:SMC_PULSE[0..7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

NWE_PULSE: NWE Pulse Length

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

NCS_WR_PULSE: NCS Pulse Length in WRITE Access

In write access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

NRD_PULSE: NRD Pulse Length

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

NCS_RD_PULSE: NCS Pulse Length in READ Access

In standard read access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

© 2017 Microchip Technology Inc.

20.14.3 SMC Cycle Register

Name:SMC_CYCLE[0..7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRD_CYCLE

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

NWE_CYCLE: Total Write Cycle Length

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

NRD_CYCLE: Total Read Cycle Length

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

© 2017 Microchip Technology Inc.

20.14.4 SMC MODE Register

Name:SMC_MODE[0..7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN

23	22	21	20	19	18	17	16
–	–	–	TDF_MODE	TDF_CYCLES			

15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT

7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

READ_MODE:

1: The read operation is controlled by the NRD signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.

– If TDF optimization is enabled (TDF_MODE = 1), TDF wait states are inserted after the setup of NRD.

0: The read operation is controlled by the NCS signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.

– If TDF optimization is enabled (TDF_MODE = 1), TDF wait states are inserted after the setup of NCS.

WRITE_MODE

1: The write operation is controlled by the NWE signal.

– If TDF optimization is enabled (TDF_MODE = 1), TDF wait states will be inserted after the setup of NWE.

0: The write operation is controlled by the NCS signal.

– If TDF optimization is enabled (TDF_MODE = 1), TDF wait states will be inserted after the setup of NCS.

EXNW_MODE: NWAIT Mode

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNW_MODE		NWAIT Mode
0	0	Disabled
0	1	Reserved
1	0	Frozen Mode
1	1	Ready Mode

- Disabled Mode: The NWAIT input signal is ignored on the corresponding Chip Select.
- Frozen Mode: If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.
- Ready Mode: The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

© 2017 Microchip Technology Inc.

BAT: Byte Access Type

This field is used only if DBW defines a 16- or 32-bit data bus.

- 1: Byte write access type:
  - Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3.
  - Read operation is controlled using NCS and NRD.
- 0: Byte select access type:
  - Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3
  - Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3

DBW: Data Bus Width

DBW		Data Bus Width
0	0	8-bit bus
0	1	16-bit bus
1	0	32-bit bus
1	1	Reserved

TDF_CYCLES: Data Float Time

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

TDF_MODE: TDF Optimization

1: TDF optimization is enabled.

- The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization is disabled.

- The number of TDF wait states is inserted before the next access begins.

PMEN: Page Mode Enabled

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

PS: Page Size

If page mode is enabled, this field indicates the size of the page in bytes.

PS		Page Size
0	0	4-byte page
0	1	8-byte page
1	0	16-byte page
1	1	32-byte page

© 2017 Microchip Technology Inc.

21. SDRAM Controller (SDRAMC)

21.1 Overview

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The SDRAM Controller supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAM controller supports a CAS latency of 1, 2 or 3 and optimizes the read access depending on the frequency.

The different modes available—self-refresh, power-down and deep power-down modes—minimize power consumption on the SDRAM device.

21.2 I/O Lines Description

Table 21-1: I/O Line Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[3:0]	Data Mask Enable Signals	Output	Low
SDRAMC_A[12:0]	Address Bus	Output	
D[31:0]	Data Bus	I/O	

21.3 Application Example

21.3.1 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAM controller allows mapping different memory types according to the values set in the SDRAMC configuration register.

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 21-2 to Table 21-7 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

21.3.1.1 32-bit Memory Data Bus Width

Table 21-2: SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
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**
					Bk[1:0]		Row[10:0]											Column[7:0]									M[1:0]	
				Bk[1:0]			Row[10:0]											Column[8:0]									M[1:0]	
			Bk[1:0]				Row[10:0]											Column[9:0]									M[1:0]	
		Bk[1:0]					Row[10:0]											Column[10:0]									M[1:0]	

Table 21-3: SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
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**
				Bk[1:0]			Row[11:0]											Column[7:0]									M[1:0]	
			Bk[1:0]				Row[11:0]											Column[8:0]									M[1:0]	
		Bk[1:0]					Row[11:0]											Column[9:0]									M[1:0]	
	Bk[1:0]						Row[11:0]											Column[10:0]									M[1:0]	

Table 21-4: SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
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**
			Bk[1:0]				Row[12:0]											Column[7:0]									M[1:0]	
		Bk[1:0]					Row[12:0]											Column[8:0]									M[1:0]	
	Bk[1:0]						Row[12:0]											Column[9:0]									M[1:0]	
Bk[1:0]							Row[12:0]											Column[10:0]									M[1:0]	

Note 1: M[1:0] is the byte address inside a 32-bit word.

2: Bk[1] = BA1, Bk[0] = BA0.

© 2017 Microchip Technology Inc.

21.3.1.2 16-bit Memory Data Bus Width

Table 21-5: SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
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**
						Bk[1:0]		Row[10:0]											Column[7:0]								M0
					Bk[1:0]			Row[10:0]											Column[8:0]								M0
				Bk[1:0]				Row[10:0]											Column[9:0]								M0
			Bk[1:0]					Row[10:0]											Column[10:0]								M0

Table 21-6: SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
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**
					Bk[1:0]			Row[11:0]											Column[7:0]								M0
				Bk[1:0]				Row[11:0]											Column[8:0]								M0
			Bk[1:0]					Row[11:0]											Column[9:0]								M0
		Bk[1:0]						Row[11:0]											Column[10:0]								M0

Table 21-7: SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
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**
				Bk[1:0]				Row[12:0]											Column[7:0]								M0
			Bk[1:0]					Row[12:0]											Column[8:0]								M0
		Bk[1:0]						Row[12:0]											Column[9:0]								M0
	Bk[1:0]							Row[12:0]											Column[10:0]								M0

Note 1: M0 is the byte address inside a 16-bit half-word.

2: Bk[1] = BA1, Bk[0] = BA0.

21.4 Product Dependencies

21.4.1 SDRAM Device Initialization

The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

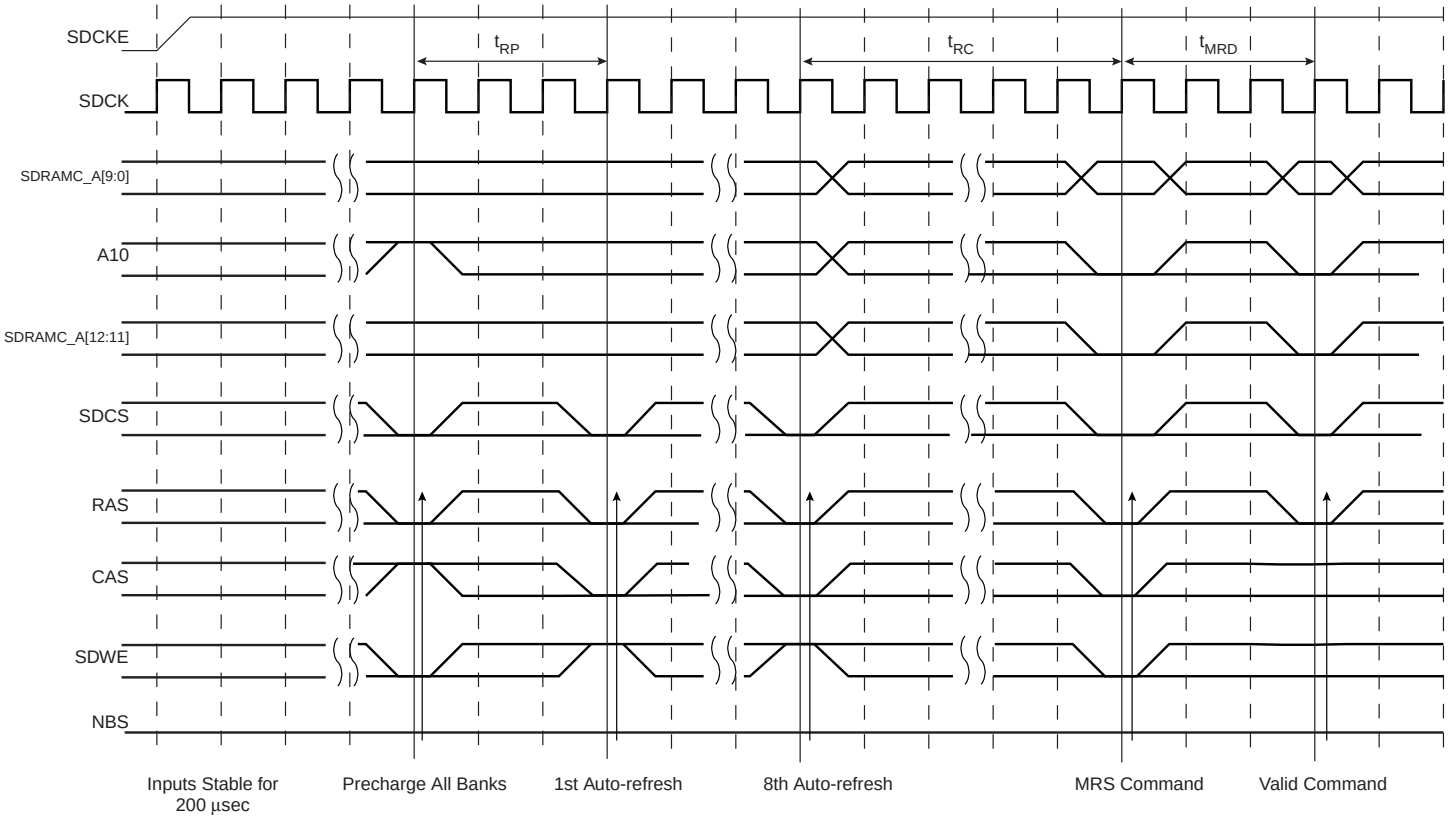
1. SDRAM features must be set in the configuration register: asynchronous timings (TRC, TRAS, etc.), number of columns, rows, CAS latency, and the data bus width.
2. For mobile SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low Power Register.
3. The SDRAM memory type must be set in the Memory Device Register.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. (1)A NOP command is issued to the SDRAM devices. The application must set Mode to 1 in the Mode Register and perform a write access to any SDRAM address.
6. An All Banks Precharge command is issued to the SDRAM devices. The application must set Mode to 2 in the Mode Register and perform a write access to any SDRAM address.
7. Eight auto-refresh (CBR) cycles are provided. The application must set the Mode to 4 in the Mode Register and perform a write access to any SDRAM location eight times.
8. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
9. For mobile SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are set to 1. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address the SDRAM write access should be done at the address 0x20800000 or 0x20400000.
10. The application must go into Normal Mode, setting Mode to 0 in the Mode Register and performing a write access at any location in the SDRAM.
11. Write the refresh rate into the count field in the SDRAMC Refresh Timer register. (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652 µs x 100 MHz) or 781(7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Note 1: It is strongly recommended to respect the instructions stated in Step 5 of the initialization process in order to be certain that the subsequent commands issued by the SDRAMC will be taken into account.

© 2017 Microchip Technology Inc.

Figure 21-1: SDRAM Device Initialization Sequence



21.4.2 I/O Lines

The pins used for interfacing the SDRAM Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SDRAM Controller pins to their peripheral function. If I/O lines of the SDRAM Controller are not used by the application, they can be used for other purposes by the PIO Controller.

21.4.3 Interrupt

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

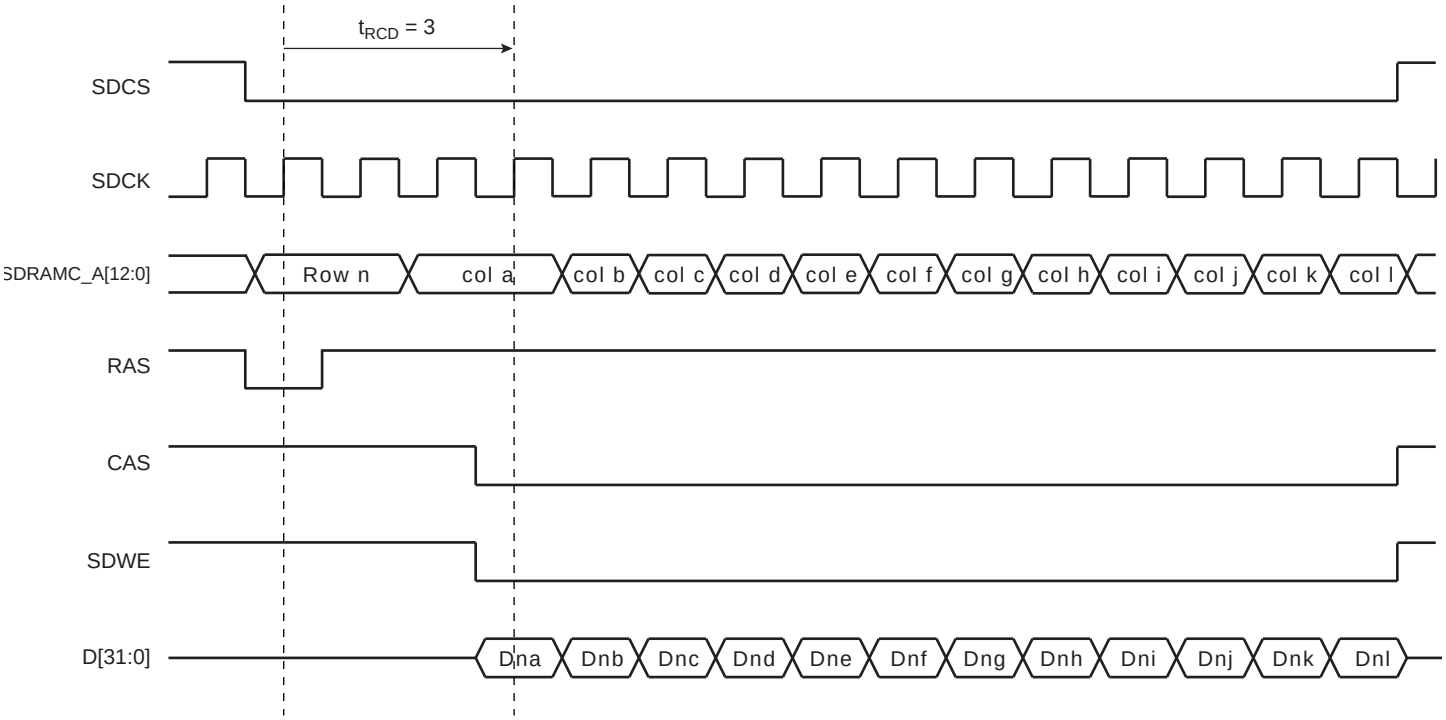
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

21.5 Functional Description

21.5.1 SDRAM Controller Write Cycle

The SDRAM Controller allows burst access or single access. In both cases, the SDRAM controller keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands. For definition of these timing parameters, refer to the "SDRAMC Configuration Register" . This is described in Figure 21-2 below.

Figure 21-2: Write Burst, 32-bit SDRAM Access



21.5.2 SDRAM Controller Read Cycle

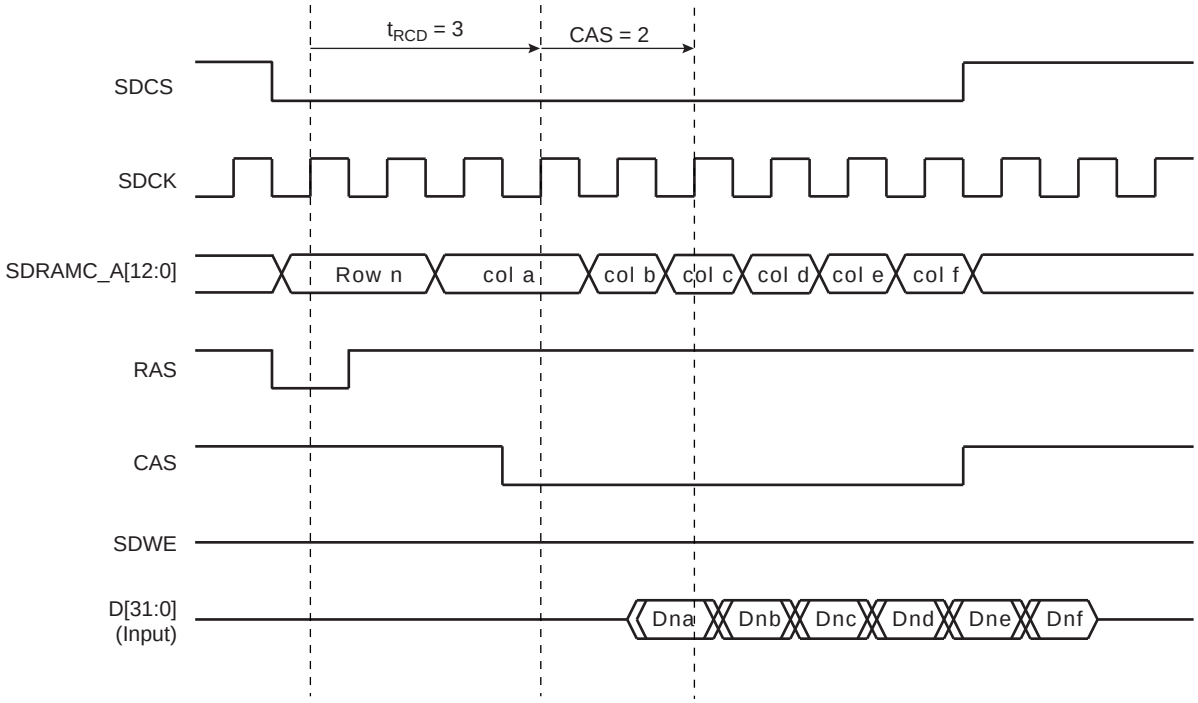
The SDRAM Controller allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAM Controller keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAM controller automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}) and between active and read command (t_{RCD}). These two parameters are set in the configuration register of the SDRAM Controller. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the configuration register).

For a single access or an incremented burst of unspecified length, the SDRAM Controller anticipates the next access. While the last value of the column is returned by the SDRAM Controller on the bus, the SDRAM Controller anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

© 2017 Microchip Technology Inc.

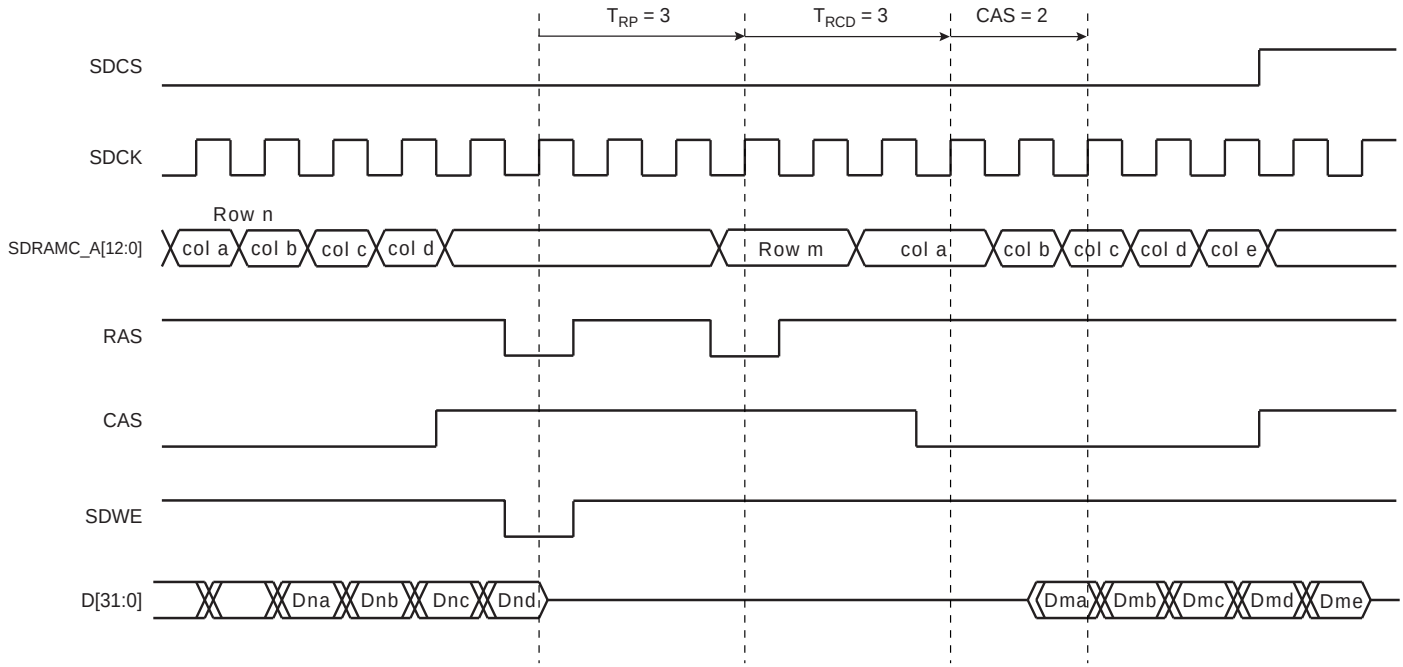
Figure 21-3: Read Burst, 32-bit SDRAM Access



21.5.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command. This is described in Figure 21-4 below.

Figure 21-4: Read Burst with Boundary Row Access



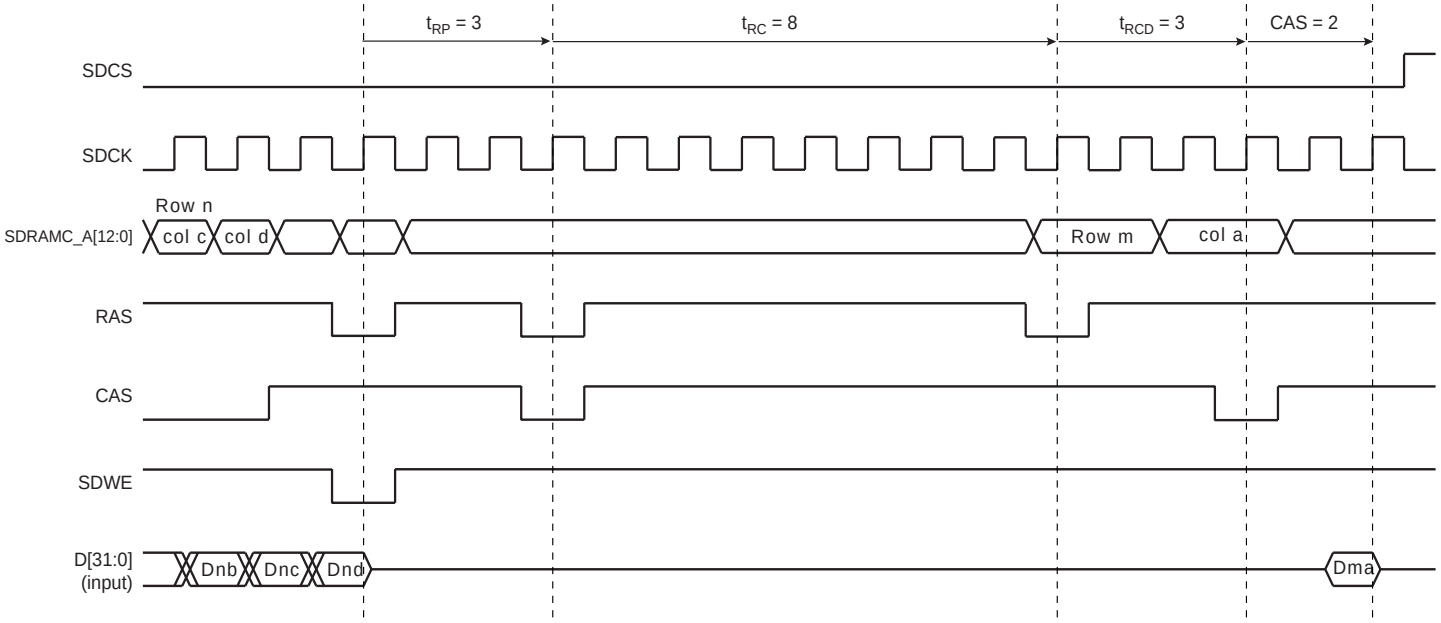
21.5.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. An internal timer is loaded with the value in the register SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It is acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 21-5.

Figure 21-5: Refresh Cycle Followed by a Read Access



21.5.5 Power Management

Three low-power modes are available:

- Self-refresh Mode: The SDRAM executes its own Auto-refresh cycle without control of the SDRAM Controller. Current drained by the SDRAM is very low.
- Power-down Mode: Auto-refresh cycles are controlled by the SDRAM Controller. Between auto-refresh cycles, the SDRAM is in power-down. Current drained in Power-down mode is higher than in Self-refresh Mode.
- Deep Power-down Mode: (Only available with Mobile SDRAM) The SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAM Controller activates one low-power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self-refresh and power-down mode after the last access by programming a timeout value in the Low Power Register.

21.5.5.1 Self-refresh Mode

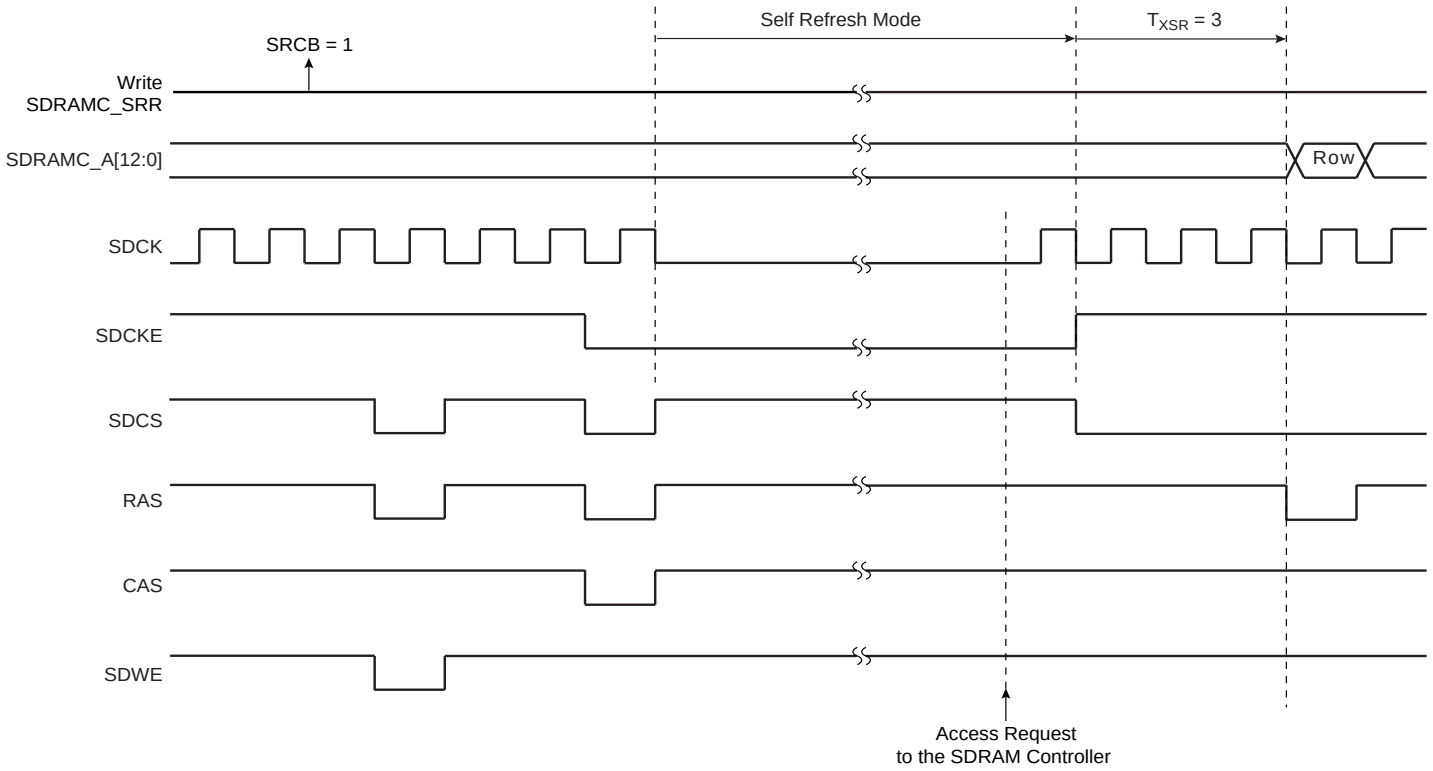
This mode is selected by programming the LPCB field to 1 in the SDRAMC Low Power Register. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode.

Some low-power SDRAMs (e.g., mobile SDRAM) can refresh only one quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self-refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR) and Drive Strength (DS) parameters must be set in the Low Power Register and transmitted to the low-power SDRAM during initialization.

The SDRAM device must remain in self-refresh mode for a minimum period of t_{RAS} and may remain in self-refresh mode for an indefinite period. This is described in Figure 21-6.

© 2017 Microchip Technology Inc.

Figure 21-6: Self-refresh Mode Behavior

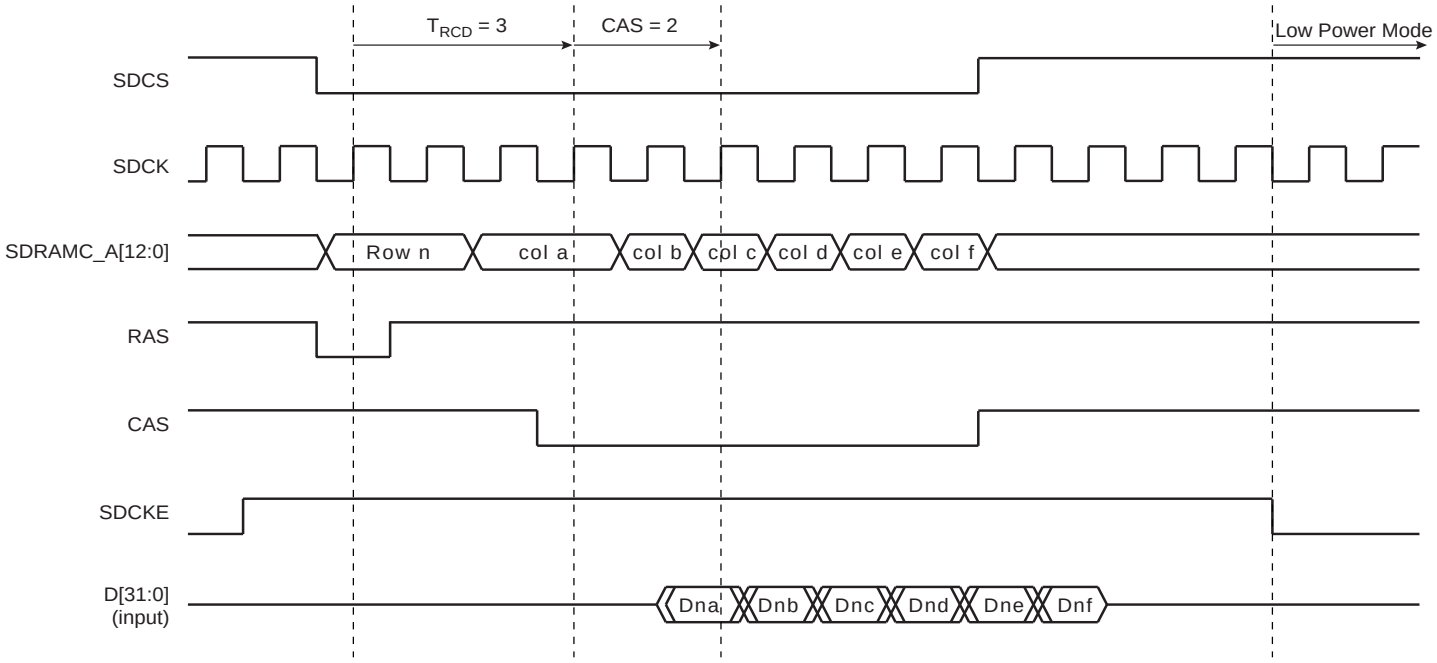


21.5.5.2 Low-power Mode

This mode is selected by programming the LPCB field to 2 in the SDRAMC Low Power Register. Power consumption is greater than in self-refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed by the SDRAM itself, the SDRAM Controller carries out the refresh operation. The exit procedure is faster than in self-refresh mode.

This is described in Figure 21-7.

Figure 21-7: Low-power Mode Behavior



21.5.5.3 Deep Power-down Mode

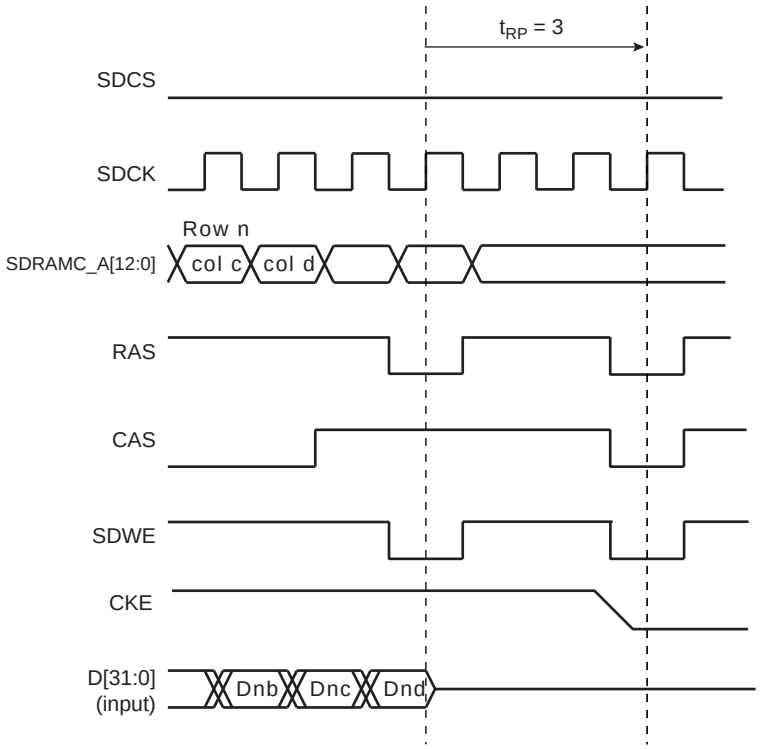
This mode is selected by programming the LPCB field to 3 in the SDRAMC Low Power Register. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the application must not access to the SDRAM until a new initialization sequence is done (See Section 21.4.1 SDRAM Device Initialization).

This is described in the following figure.

© 2017 Microchip Technology Inc.

Figure 21-8: Deep Power-down Mode Behavior



21.6 SDRAM Controller (SDRAMC) User Interface

Table 21-8: Register Mapping

Offset	Register	Name	Access	Reset
0x00	SDRAMC Mode Register	SDRAMC_MR	Read/Write	0x00000000
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read/Write	0x00000000
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read/Write	0x852372C0
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read/Write	0x0
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x0
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x0
0x24	SDRAMC Memory Device Register	SDRAMC_MDR	Read/Write	0x0
0x28–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

21.6.1 SDRAMC Mode Register

Name:SDRAMC_MR

Access: 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
–	–	–	–	–	MODE		

MODE: SDRAMC Command Mode

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

MODE			Description
0	0	0	Normal mode. Any access to the SDRAM is decoded normally. To activate this mode, command must be followed by a write to the SDRAM.
0	0	1	The SDRAM Controller issues a NOP command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
0	1	0	The SDRAM Controller issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
0	1	1	The SDRAM Controller issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
1	0	0	The SDRAM Controller issues an “Auto-Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued. To activate this mode, command must be followed by a write to the SDRAM.
1	0	1	The SDRAM Controller issues an “Extended Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the “Extended Load Mode Register” command must be followed by a write to the SDRAM. The write in the SDRAM must be done in the appropriate bank; most low-power SDRAM devices use the bank 1.
1	1	0	Deep power-down mode. Enters deep power-down mode.

© 2017 Microchip Technology Inc.

21.6.2 SDRAMC Refresh Timer Register

Name:SDRAMC_TR

Access: 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
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

COUNT: SDRAMC Refresh Timer Count

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 µs per row is a typical value for a burst of length one.

To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

© 2017 Microchip Technology Inc.

21.6.3 SDRAMC Configuration Register

Name:SDRAMC_CR

Access: Read/Write

31	30	29	28	27	26	25	24
TXSR				TRAS			
23	22	21	20	19	18	17	16
TRCD				TRP			
15	14	13	12	11	10	9	8
TRC				TWR			
7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

NC: Number of Column Bits

Reset value is 8 column bits.

NC		Column Bits
0	0	8
0	1	9
1	0	10
1	1	11

NR: Number of Row Bits

Reset value is 11 row bits.

NR		Row Bits
0	0	11
0	1	12
1	0	13
1	1	Reserved

NB: Number of Banks

Reset value is two banks.

NB	Number of Banks
0	2
1	4

CAS: CAS Latency

Reset value is two cycles.

In the SDRAMC, only a CAS latency of one, two and three cycles are managed.

CAS		CAS Latency (Cycles)
0	0	Reserved
0	1	1
1	0	2
1	1	3

DBW: Data Bus Width

Reset value is 16 bits

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

TWR: Write Recovery Delay

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.

TRC: Row Cycle Delay

Reset value is seven cycles.

This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

TRP: Row Precharge Delay

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 0 and 15.

TRCD: Row to Column Delay

Reset value is two cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

TRAS: Active to Precharge Delay

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

TXSR: Exit Self Refresh to Active Delay

Reset value is eight cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

© 2017 Microchip Technology Inc.

21.6.4 SDRAMC Low Power Register

Name:SDRAMC_LPR

Access: 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
–	–	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
–	PASR			–	–	LPCB	

LPCB: Low-power Configuration Bits

00	Low Power Feature is inhibited: no Power-down, Self-refresh or Deep Power-down command is issued to the SDRAM device.
01	The SDRAM Controller issues a Self-refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the Self Refresh Mode when accessed and enters it after the access.
10	The SDRAM Controller issues a Power-down Command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the Power-down Mode when accessed and enters it after the access.
11	The SDRAM Controller issues a Deep Power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

PASR: Partial Array Self-refresh (only for low-power SDRAM)

PASR parameter is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self-refresh mode. This parameter must be set according to the SDRAM device specification.

TCSR: Temperature Compensated Self-Refresh (only for low-power SDRAM)

TCSR parameter is transmitted to the SDRAM during initialization to set the refresh interval during self-refresh mode depending on the temperature of the low-power SDRAM. This parameter must be set according to the SDRAM device specification.

DS: Drive Strength (only for low-power SDRAM)

DS parameter is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.

TIMEOUT: Time to define when low-power mode is enabled

00	The SDRAM controller activates the SDRAM low-power mode immediately after the end of the last transfer.
01	The SDRAM controller activates the SDRAM low-power mode 64 clock cycles after the end of the last transfer.
10	The SDRAM controller activates the SDRAM low-power mode 128 clock cycles after the end of the last transfer.
11	Reserved.

21.6.5 SDRAMC Interrupt Enable Register

Name:SDRAMC_IER

Access: 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
–	–	–	–	–	–	–	RES

RES: Refresh Error Status

0: No effect.

1: Enables the refresh error interrupt.

© 2017 Microchip Technology Inc.

21.6.6 SDRAMC Interrupt Disable Register

Name:SDRAMC_IDR

Access: 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
–	–	–	–	–	–	–	RES

RES: Refresh Error Status

0: No effect.

1: Disables the refresh error interrupt.

21.6.7 SDRAMC Interrupt Mask Register

Name:SDRAMC_IMR

Access: 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
–	–	–	–	–	–	–	RES

RES: Refresh Error Status

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

© 2017 Microchip Technology Inc.

21.6.8 SDRAMC Interrupt Status Register

Name:SDRAMC_ISR

Access: 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
–	–	–	–	–	–	–	RES

RES: Refresh Error Status

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.

21.6.9 SDRAMC Memory Device Register

Name:SDRAMC_MDR

Access: 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
–	–	–	–	–	–	MD	

MD: Memory Device Type

00	SDRAM
01	Low-power SDRAM
10	Reserved
11	Reserved.

© 2017 Microchip Technology Inc.

22. Error Correction Code Controller (ECC)

22.1 Overview

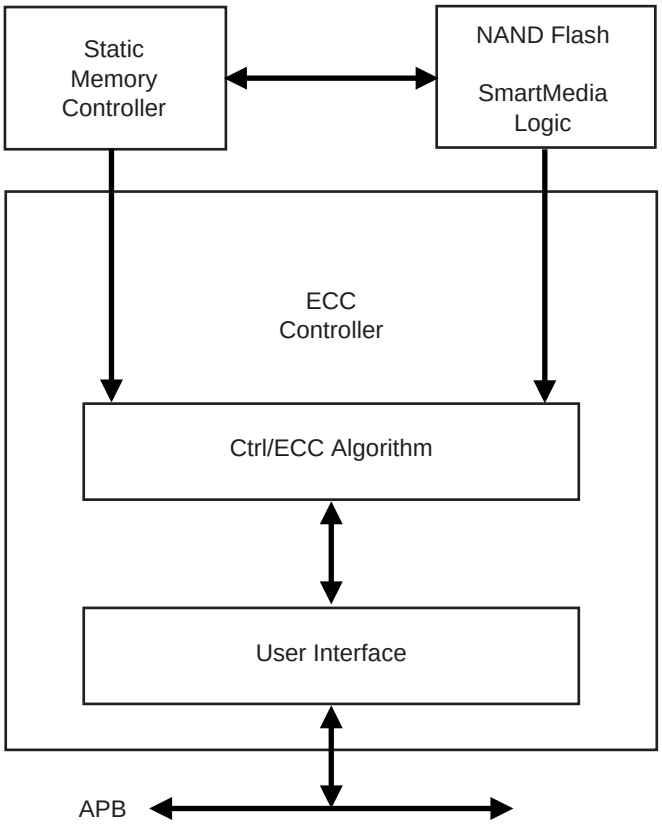
NAND Flash/SmartMedia devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash/SmartMedia lifetime, additional invalid blocks may occur which can be detected/corrected by ECC code.

The ECC Controller is a mechanism that encodes data in a manner that makes possible the identification and correction of certain errors in data. The ECC controller is capable of single bit error correction and 2-bit random detection. When NAND Flash/SmartMedia have more than 2 bits of errors, the data cannot be corrected.

The ECC user interface is compliant with the Arm Advanced Peripheral Bus (APB rev2).

22.2 Block Diagram

Figure 22-1: Block Diagram



22.3 Functional Description

A page in NAND Flash and SmartMedia memories contains an area for main data and an additional area used for redundancy (ECC). The page is organized in 8-bit or 16-bit words. The page size corresponds to the number of words in the main area plus the number of words in the extra area used for redundancy.

Over time, some memory locations may fail to program or erase properly. In order to ensure that data is stored properly over the life of the NAND Flash device, NAND Flash providers recommend to utilize either 1 ECC per 256 bytes of data, 1 ECC per 512 bytes of data or 1 ECC for all of the page.

The only configurations required for ECC are the NAND Flash or the SmartMedia page size (528/2112/4224) and the type of correction wanted (1 ECC for all the page/1 ECC per 256 bytes of data /1 ECC per 512 bytes of data). Page size is configured setting the PAGESIZE field in the ECC Mode Register (ECC_MR). Type of correction is configured setting the TYPCORRECT field in the ECC Mode Register (ECC_MR).

© 2017 Microchip Technology Inc.

ECC is automatically computed as soon as a read (00h)/write (80h) command to the NAND Flash or the SmartMedia is detected. Read and write access must start at a page boundary.

ECC results are available as soon as the counter reaches the end of the main area. Values in the ECC Parity Registers (ECC_PR0 to ECC_PR15) are then valid and locked until a new start condition occurs (read/write command followed by address cycles).

22.3.1 Write Access

Once the Flash memory page is written, the computed ECC codes are available in the ECC Parity (ECC_PR0 to ECC_PR15) registers. The ECC code values must be written by the software application in the extra area used for redundancy. The number of write accesses in the extra area is a function of the value of the type of correction field. For example, for 1 ECC per 256 bytes of data for a page of 512 bytes, only the values of ECC_PR0 and ECC_PR1 must be written by the software application. Other registers are meaningless.

22.3.2 Read Access

After reading the whole data in the main area, the application must perform read accesses to the extra area where ECC code has been previously stored. Error detection is automatically performed by the ECC controller. Note that it is mandatory to read consecutively the entire main area and the locations where Parity and NParity values have been previously stored to let the ECC controller perform error detection.

The application can check the ECC Status Registers (ECC_SR1/ECC_SR2) for any detected errors. It is up to the application to correct any detected error. ECC computation can detect four different circumstances:

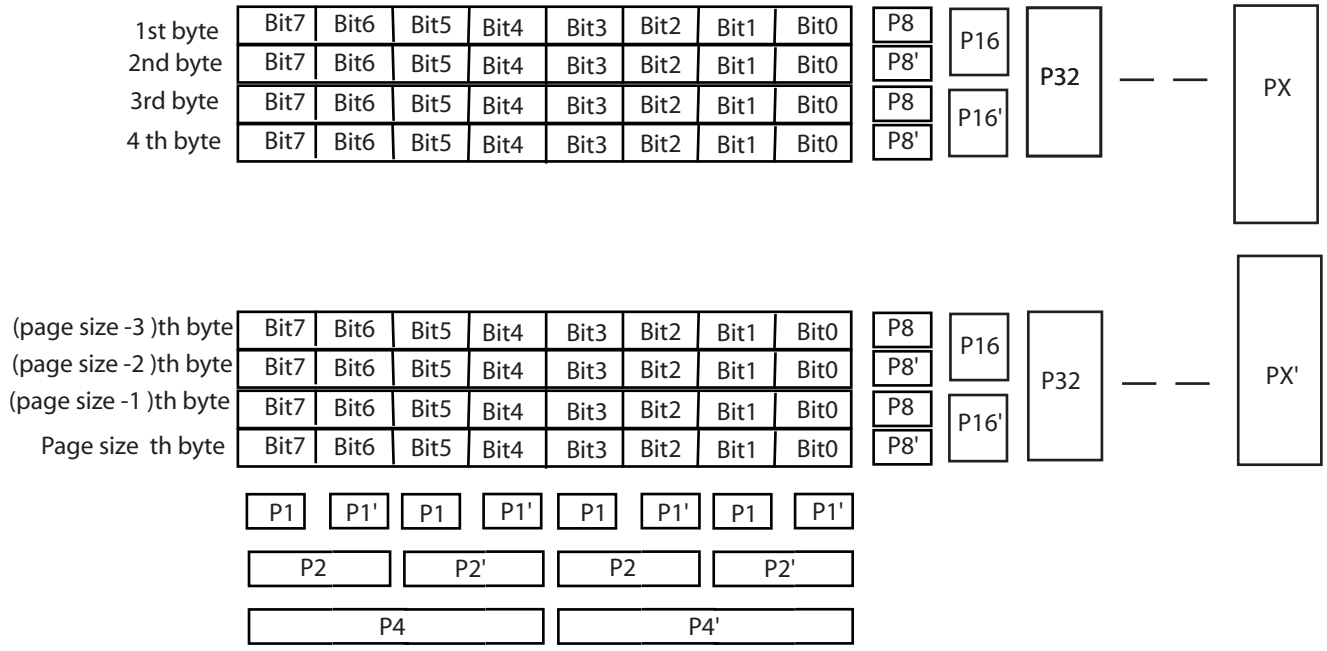
- No error: XOR between the ECC computation and the ECC code stored at the end of the NAND Flash or SmartMedia page is equal to 0. No error flags in the ECC Status Registers (ECC_SR1/ECC_SR2).
- Recoverable error: Only the RECERR flags in the ECC Status registers (ECC_SR1/ECC_SR2) are set. The corrupted word offset in the read page is defined by the WORDADDR field in the ECC Parity Registers (ECC_PR0 to ECC_PR15). The corrupted bit position in the concerned word is defined in the BITADDR field in the ECC Parity Registers (ECC_PR0 to ECC_PR15).
- ECC error: The ECCERR flag in the ECC Status Registers (ECC_SR1/ECC_SR2) are set. An error has been detected in the ECC code stored in the Flash memory. The position of the corrupted bit can be found by the application performing an XOR between the Parity and the NParity contained in the ECC code stored in the Flash memory.
- Non correctable error: The MULERR flag in the ECC Status Registers (ECC_SR1/ECC_SR2) are set. Several unrecoverable errors have been detected in the Flash memory page.

ECC Status Registers, ECC Parity Registers are cleared when a read/write command is detected or a software reset is performed.

For Single-bit Error Correction and Double-bit Error Detection (SEC-DED) hsiao code is used. 24-bit ECC is generated in order to perform one bit correction per 256 or 512 bytes for pages of 512/2048/4096 8-bit words. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-bit words.They are generated according to the schemes shown in Figure 22-2 and Figure 22-3.

© 2017 Microchip Technology Inc.

Figure 22-2: Parity Generation for 512/1024/2048/4096 8-bit Words



Page size = 512 Px = 2048
Page size = 1024 Px = 4096
Page size = 2048 Px = 8192
Page size = 4096 Px = 16384

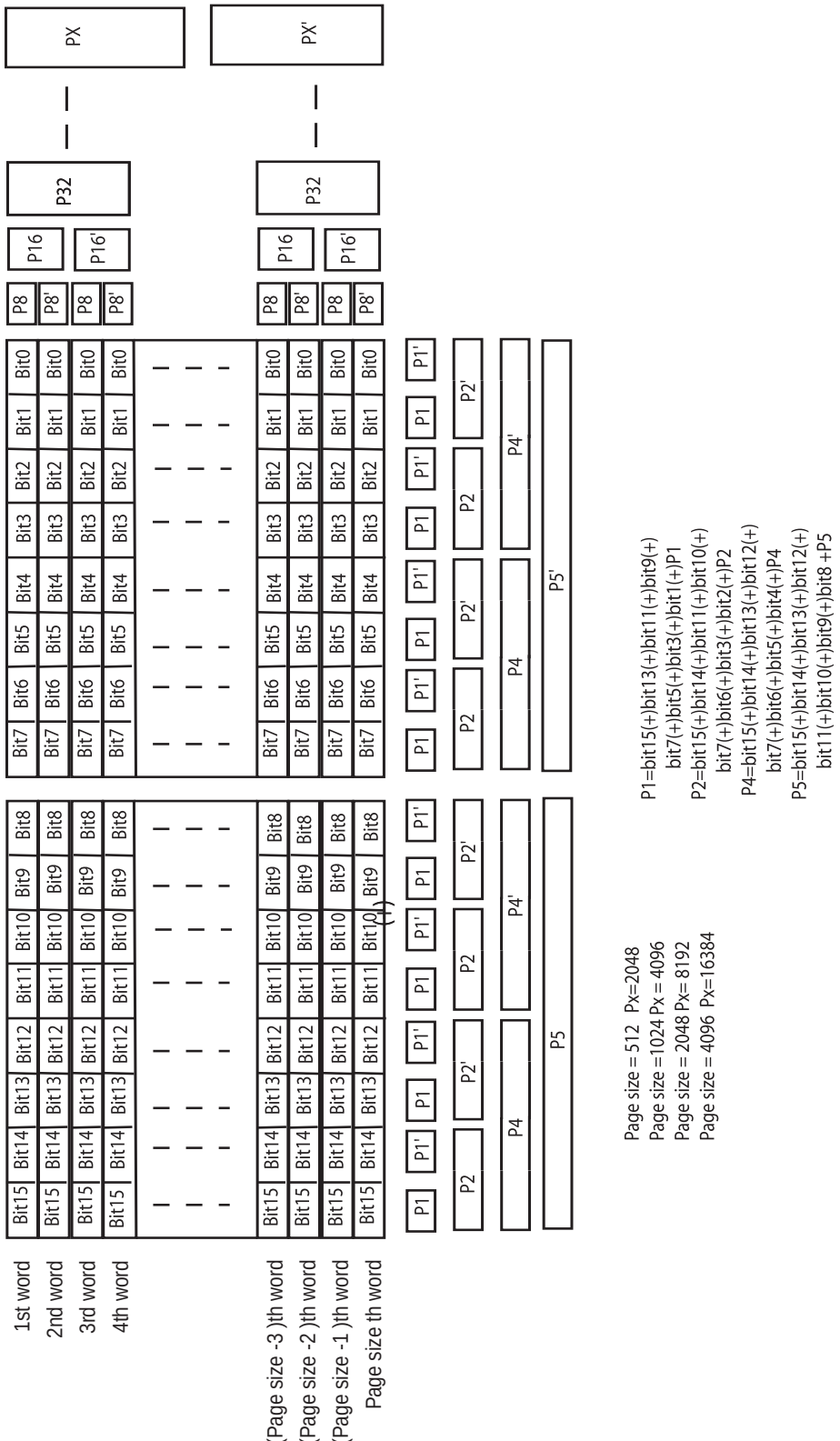
P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
P1'=bit6(+)bit4(+)bit2(+)bit0(+)P1'
P2'=bit5(+)bit4(+)bit1(+)bit0(+)P2'
P4'=bit7(+)bit6(+)bit5(+)bit4(+)P4'

To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

Page size = 2^n

```
for i =0 to n
begin
 for (j = 0 to page_size_byte)
 begin
  if(j[i] ==1)
   P[2^(i+3)]=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
              bit2(+)bit1(+)bit0(+)P[2^(i+3)]
  else
   P[2^(i+3)]'=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
               bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
  end
end
```

Figure 22-3: Parity Generation for 512/1024/2048/4096 16-bit Words



To calculate P8’ to PX’ and P8 to PX, apply the algorithm that follows.

© 2017 Microchip Technology Inc.

```
Page size = 2^n

   for i =0 to n
   begin
    for (j = 0 to page_size_word)
    begin
     if(j[i] ==1)
      P[2^(i+3)]= bit15(+)bit14(+)bit13(+)bit12(+)
                  bit11(+)bit10(+)bit9(+)bit8(+)
                  bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                  bit2(+)bit1(+)bit0(+)P[2^(n+3)]
     else
      P[2^(i+3)]'=bit15(+)bit14(+)bit13(+)bit12(+)
                  bit11(+)bit10(+)bit9(+)bit8(+)
                  bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                  bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
     end
   end
```

22.4 Error Correction Code Controller (ECC) User Interface

Table 22-1: Register Mapping

Offset	Register	Name	Access	Reset
0x00	ECC Control Register	ECC_CR	Write-only	–
0x04	ECC Mode Register	ECC_MD	Read/Write	0x0
0x08	ECC Status1 Register	ECC_SR1	Read-only	0x0
0x0C	ECC Parity Register 0	ECC_PR0	Read-only	0x0
0x10	ECC Parity Register 1	ECC_PR1	Read-only	0x0
0x14	ECC Status2 Register	ECC_SR2	Read-only	0x0
0x18	ECC Parity 2	ECC_PR2	Read-only	0x0
0x1C	ECC Parity 3	ECC_PR3	Read-only	0x0
0x20	ECC Parity 4	ECC_PR4	Read-only	0x0
0x24	ECC Parity 5	ECC_PR5	Read-only	0x0
0x28	ECC Parity 6	ECC_PR6	Read-only	0x0
0x2C	ECC Parity 7	ECC_PR7	Read-only	0x0
0x30	ECC Parity 8	ECC_PR8	Read-only	0x0
0x34	ECC Parity 9	ECC_PR9	Read-only	0x0
0x38	ECC Parity 10	ECC_PR10	Read-only	0x0
0x3C	ECC Parity 11	ECC_PR11	Read-only	0x0
0x40	ECC Parity 12	ECC_PR12	Read-only	0x0
0x44	ECC Parity 13	ECC_PR13	Read-only	0x0
0x48	ECC Parity 14	ECC_PR14	Read-only	0x0
0x4C	ECC Parity 15	ECC_PR15	Read-only	0x0
0x14–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

22.4.1 ECC Control Register

Name:ECC_CR

Access: 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
–	–	–	–	–	–	–	RST

RST: RESET Parity

Provides reset to current ECC by software.

1: Reset ECC Parity registers

0: No effect

22.4.2 ECC Mode Register

Name:ECC_MR

Access: 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
–	–	TYPCORREC		–	–	PAGESIZE	

PAGESIZE: Page Size

This field defines the page size of the NAND Flash device.

Page Size	Description
00	528 words
01	1056 words
10	2112 words
11	4224 words

A word has a value of 8 bits or 16 bits, depending on the NAND Flash or SmartMedia memory organization.

TYPECORREC: Type of Correction

00: 1 bit correction for a page size of 512/1024/2048/4096 bytes.

01: 1 bit correction for 256 bytes of data for a page size of 512/2048/4096 bytes.

10: 1 bit correction for 512 bytes of data for a page size of 512/2048/4096 bytes.

© 2017 Microchip Technology Inc.

22.4.3 ECC Status Register 1

Name:ECC_SR1

Access: Read-only

31	30	29	28	27	26	25	24
–	ECCERR7	ECCERR7	RECERR7	–	ECCERR6	ECCERR6	RECERR6
23	22	21	20	19	18	17	16
–	ECCERR5	ECCERR5	RECERR5	–	ECCERR4	ECCERR4	RECERR4
15	14	13	12	11	10	9	8
–	MULERR3	ECCERR3	RECERR3	–	MULERR2	ECCERR2	RECERR2
7	6	5	4	3	2	1	0
–	MULERR1	ECCERR1	RECERR1	–	ECCERR0	ECCERR0	RECERR0

RECERR0: Recoverable Error

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR0: ECC Error

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

If TYPECORRECT = 0, read both ECC Parity 0 and ECC Parity 1 registers, the error occurred at the location which contains a 1 in the least significant 16 bits; else read ECC Parity 0 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR0: Multiple Error

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR1: Recoverable Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR1: ECC Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes

Fixed to 0 if TYPECORREC = 0

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 1 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR1: Multiple Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR2: Recoverable Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

ECCERR2: ECC Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 2 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR2: Multiple Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR3: Recoverable Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR3: ECC Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 3 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR3: Multiple Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR4: Recoverable Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR4: ECC Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 4 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR4: Multiple Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR5: Recoverable Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

© 2017 Microchip Technology Inc.

ECCERR5: ECC Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 5 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR5: Multiple Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR6: Recoverable Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR6: ECC Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 6 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR6: Multiple Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR7: Recoverable Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

ECCERR7: ECC Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 7 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR7: Multiple Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

22.4.4 ECC Status Register 2

Name:ECC_SR2

Access: Read-only

31	30	29	28	27	26	25	24
–	ECCERR15	ECCERR15	RECERR15	–	ECCERR14	ECCERR14	–RECERR14
23	22	21	20	19	18	17	16
–	ECCERR13	ECCERR13	RECERR13	–	ECCERR12	ECCERR12	RECERR12
15	14	13	12	11	10	9	8
–	MULERR11	ECCERR11	RECERR11	–	MULERR10	ECCERR10	RECERR10
7	6	5	4	3	2	1	0
–	MULERR9	ECCERR9	RECERR9	–	ECCERR8	ECCERR8	RECERR8

RECERR8: Recoverable Error in the page between the 2048th and the 2303rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected

ECCERR8: ECC Error in the page between the 2048th and the 2303rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 8 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR8: Multiple Error in the page between the 2048th and the 2303rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR9: Recoverable Error in the page between the 2304th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR9: ECC Error in the page between the 2304th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 9 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR9: Multiple Error in the page between the 2304th and the 2559th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR10: Recoverable Error in the page between the 2560th and the 2815th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

© 2017 Microchip Technology Inc.

ECCERR10: ECC Error in the page between the 2560th and the 2815th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 10 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR10: Multiple Error in the page between the 2560th and the 2815th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR11: Recoverable Error in the page between the 2816th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected.. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected

ECCERR11: ECC Error in the page between the 2816th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 11 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR11: Multiple Error in the page between the 2816th and the 3071st bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR12: Recoverable Error in the page between the 3072nd and the 3327th bytes

Fixed to 0 if TYPECORREC = 0

0: No Errors Detected

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected

ECCERR12: ECC Error in the page between the 3072nd and the 3327th bytes

Fixed to 0 if TYPECORREC = 0

0: No Errors Detected

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 12 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR12: Multiple Error in the page between the 3072nd and the 3327th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR13: Recoverable Error in the page between the 3328th and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

ECCERR13: ECC Error in the page between the 3328th and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 13 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR13: Multiple Error in the page between the 3328th and the 3583rd bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR14: Recoverable Error in the page between the 3584th and the 3839th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

ECCERR14: ECC Error in the page between the 3584th and the 3839th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 14 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR14: Multiple Error in the page between the 3584th and the 3839th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

RECERR15: Recoverable Error in the page between the 3840th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Errors Detected.

1: Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected

ECCERR15: ECC Error in the page between the 3840th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0

0: No Errors Detected.

1: A single bit error occurred in the ECC bytes.

Read ECC Parity 15 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

MULERR15: Multiple Error in the page between the 3840th and the 4095th bytes

Fixed to 0 if TYPECORREC = 0.

0: No Multiple Errors Detected.

1: Multiple Errors Detected.

© 2017 Microchip Technology Inc.

22.5 Registers for 1 ECC for a page of 512/1024/2048/4096 bytes

22.5.1 ECC Parity Register 0

Name:ECC_PR0

Access: 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
WORDADDR							
7	6	5	4	3	2	1	0
WORDADDR				BITADDR			

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR: Bit Address

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR: Word Address

During a page read, this value contains the word address (8-bit or 16-bit word depending on the memory plane organization) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

© 2017 Microchip Technology Inc.

22.5.2 ECC Parity Register 1

Name:ECC_PR1

Access: 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
NPARITY							
7	6	5	4	3	2	1	0
NPARITY							

NPARITY

Parity N

© 2017 Microchip Technology Inc.

22.6 Registers for 1 ECC per 512 bytes for a page of 512/2048/4096 bytes, 8-bit word

22.6.1 ECC Parity Register 0

Name:ECC_PR0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY0							
15	14	13	12	11	10	9	8
NPARITY0				WORDADD0			
7	6	5	4	3	2	1	0
WORDADDR0					BITADDR0		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR0: corrupted Bit Address in the page between the first byte and the 511th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR0: corrupted Word Address in the page between the first byte and the 511th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY0

Parity N

22.6.2 ECC Parity Register 1

Name:ECC_PR1

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY1							
15	14	13	12	11	10	9	8
NPARITY1				WORDADD1			
7	6	5	4	3	2	1	0
WORDADDR1					BITADDR1		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR1: corrupted Bit Address in the page between the 512th and the 1023rd bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR1: corrupted Word Address in the page between the 512th and the 1023rd bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY1

Parity N

© 2017 Microchip Technology Inc.

22.6.3 ECC Parity Register 2

Name:ECC_PR2

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY2							
15	14	13	12	11	10	9	8
NPARITY2				WORDADD2			
7	6	5	4	3	2	1	0
WORDADDR2					BITADDR2		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR2: corrupted Bit Address in the page between the 1023rd and the 1535th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR2: corrupted Word Address in the page in the page between the 1023rd and the 1535th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY2

Parity N

22.6.4 ECC Parity Register 3

Name:ECC_PR3

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY3							
15	14	13	12	11	10	9	8
NPARITY3				WORDADD3			
7	6	5	4	3	2	1	0
WORDADDR3					BITADDR3		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR3: corrupted Bit Address in the page between the1536th and the 2047th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR3 corrupted Word Address in the page between the 1536th and the 2047th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY3

Parity N

© 2017 Microchip Technology Inc.

22.6.5 ECC Parity Register 4

Name:ECC_PR4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY4							
15	14	13	12	11	10	9	8
NPARITY4				WORDADD4			
7	6	5	4	3	2	1	0
WORDADDR4					BITADDR4		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR4: corrupted Bit Address in the page between the 2048th and the 2559th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR4: corrupted Word Address in the page between the 2048th and the 2559th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY4

Parity N

22.6.6 ECC Parity Register 5

Name:ECC_PR5

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY5							
15	14	13	12	11	10	9	8
NPARITY5				WORDADD5			
7	6	5	4	3	2	1	0
WORDADDR5					BITADDR5		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR5: corrupted Bit Address in the page between the 2560th and the 3071st bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR5: corrupted Word Address in the page between the 2560th and the 3071st bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY5

Parity N

© 2017 Microchip Technology Inc.

22.6.7 ECC Parity Register 6

Name:ECC_PR6

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY6							
15	14	13	12	11	10	9	8
NPARITY6				WORDADD6			
7	6	5	4	3	2	1	0
WORDADDR6					BITADDR6		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR6: corrupted Bit Address in the page between the 3072nd and the 3583rd bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR6: corrupted Word Address in the page between the 3072nd and the 3583rd bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY6

Parity N

22.6.8 ECC Parity Register 7

Name:ECC_PR7

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY7							
15	14	13	12	11	10	9	8
NPARITY7				WORDADD7			
7	6	5	4	3	2	1	0
WORDADDR7					BITADDR7		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR7: corrupted Bit Address in the page between the 3584h and the 4095th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR7: corrupted Word Address in the page between the 3584th and the 4095th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY7

Parity N

© 2017 Microchip Technology Inc.

22.7 Registers for 1 ECC per 256 bytes for a page of 512/2048/4096 bytes, 8-bit word

22.7.1 ECC Parity Register 0

Name:ECC_PR0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY0						
15	14	13	12	11	10	9	8
NPARITY0				0	WORDADD0		
7	6	5	4	3	2	1	0
WORDADDR0					BITADDR0		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR0: corrupted Bit Address in the page between the first byte and the 255th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR0: corrupted Word Address in the page between the first byte and the 255th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY0

Parity N

22.7.2 ECC Parity Register 1

Name:ECC_PR1

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY1						
15	14	13	12	11	10	9	8
NPARITY1				0	WORDADD1		
7	6	5	4	3	2	1	0
WORDADDR1					BITADDR1		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area

BITADDR1: corrupted Bit Address in the page between the 256th and the 511th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR1: corrupted Word Address in the page between the 256th and the 511th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY1

Parity N

© 2017 Microchip Technology Inc.

22.7.3 ECC Parity Register 2

Name:ECC_PR2

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY2						
15	14	13	12	11	10	9	8
NPARITY2				0	WORDADD2		
7	6	5	4	3	2	1	0
WORDADDR2					BITADDR2		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR2: corrupted Bit Address in the page between the 512th and the 767th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR2: corrupted Word Address in the page between the 512th and the 767th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY2

Parity N

22.7.4 ECC Parity Register 3

Name:ECC_PR3

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY3						
15	14	13	12	11	10	9	8
NPARITY3				0	WORDADD3		
7	6	5	4	3	2	1	0
WORDADDR3					BITADDR3		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR3: corrupted Bit Address in the page between the 768th and the 1023rd bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR3: corrupted Word Address in the page between the 768th and the 1023rd bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless

NPARITY3

Parity N

© 2017 Microchip Technology Inc.

22.7.5 ECC Parity Register 4

Name:ECC_PR4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY4						
15	14	13	12	11	10	9	8
NPARITY4				0	WORDADD4		
7	6	5	4	3	2	1	0
WORDADDR4					BITADDR4		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area

BITADDR4: corrupted bit address in the page between the 1024th and the 1279th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR4: corrupted word address in the page between the 1024th and the 1279th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY4

Parity N

22.7.6 ECC Parity Register 5

Name:ECC_PR5

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY5						
15	14	13	12	11	10	9	8
NPARITY5				0	WORDADD5		
7	6	5	4	3	2	1	0
WORDADDR5					BITADDR5		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR5: corrupted Bit Address in the page between the 1280th and the 1535th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR5: corrupted Word Address in the page between the 1280th and the 1535th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY5

Parity N

© 2017 Microchip Technology Inc.

22.7.7 ECC Parity Register 6

Name:ECC_PR6

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY6						
15	14	13	12	11	10	9	8
NPARITY6				0	WORDADDR6		
7	6	5	4	3	2	1	0
WORDADDR6					BITADDR6		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR6: corrupted bit address in the page between the 1536th and the 1791st bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR6: corrupted word address in the page between the 1536th and the 1791st bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY6

Parity N

22.7.8 ECC Parity Register 7

Name:ECC_PR7

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY7						
15	14	13	12	11	10	9	8
NPARITY7				0	WORDADDR7		
7	6	5	4	3	2	1	0
WORDADDR7					BITADDR7		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR7: corrupted Bit Address in the page between the 1792nd and the 2047th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR7: corrupted Word Address in the page between the 1792nd and the 2047th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY7

Parity N

© 2017 Microchip Technology Inc.

22.7.9 ECC Parity Register 8

Name:ECC_PR8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY8						
15	14	13	12	11	10	9	8
NPARITY8				0	WORDADDR8		
7	6	5	4	3	2	1	0
WORDADDR8					BITADDR8		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR8: corrupted Bit Address in the page between the 2048th and the2303rd bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR8: corrupted Word Address in the page between the 2048th and the 2303rd bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY8

Parity N.

22.7.10 ECC Parity Register 9

Name:ECC_PR9

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY9						
15	14	13	12	11	10	9	8
NPARITY9				0	WORDADDR9		
7	6	5	4	3	2	1	0
WORDADDR9					BITADDR9		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area

BITADDR9: corrupted bit address in the page between the 2304th and the 2559th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR9: corrupted word address in the page between the 2304th and the 2559th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless

NPARITY9

Parity N

© 2017 Microchip Technology Inc.

22.7.11 ECC Parity Register 10

Name:ECC_PR10

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY10						
15	14	13	12	11	10	9	8
NPARITY10				0	WORDADDR10		
7	6	5	4	3	2	1	0
WORDADDR10					BITADDR10		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR10: corrupted Bit Address in the page between the 2560th and the 2815th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR10: corrupted Word Address in the page between the 2560th and the 2815th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY10

Parity N

22.7.12 ECC Parity Register 11

Name:ECC_PR11

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY11						
15	14	13	12	11	10	9	8
NPARITY11				0	WORDADDR11		
7	6	5	4	3	2	1	0
WORDADDR11					BITADDR11		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR11: corrupted Bit Address in the page between the 2816th and the 3071st bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR11: corrupted Word Address in the page between the 2816th and the 3071st bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY11

Parity N

© 2017 Microchip Technology Inc.

22.7.13 ECC Parity Register 12

Name:ECC_PR12

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY12						
15	14	13	12	11	10	9	8
NPARITY12				0	WORDADDR12		
7	6	5	4	3	2	1	0
WORDADDR12					BITADDR12		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR12; corrupted Bit Address in the page between the 3072nd and the 3327th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR12: corrupted Word Address in the page between the 3072nd and the 3327th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY12

Parity N

22.7.14 ECC Parity Register 13

Name:ECC_PR13

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY13						
15	14	13	12	11	10	9	8
NPARITY13				0	WORDADDR13		
7	6	5	4	3	2	1	0
WORDADDR13					BITADDR13		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR13: corrupted Bit Address in the page between the 3328th and the 3583rd bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR13: corrupted Word Address in the page between the 3328th and the 3583rd bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY13

Parity N

© 2017 Microchip Technology Inc.

22.7.15 ECC Parity Register 14

Name:ECC_PR14

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY14						
15	14	13	12	11	10	9	8
NPARITY14				0	WORDADDR14		
7	6	5	4	3	2	1	0
WORDADDR14					BITADDR14		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

BITADDR14: corrupted Bit Address in the page between the 3584th and the 3839th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR14: corrupted Word Address in the page between the 3584th and the 3839th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY14

Parity N

22.7.16 ECC Parity Register 15

Name:ECC_PR15

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY15						
15	14	13	12	11	10	9	8
NPARITY15				0	WORDADDR15		
7	6	5	4	3	2	1	0
WORDADDR15					BITADDR15		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area

BITADDR15: corrupted Bit Address in the page between the 3840th and the 4095th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR15: corrupted Word Address in the page between the 3840th and the 4095th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

NPARITY15

Parity N

© 2017 Microchip Technology Inc.

23. Peripheral DMA Controller (PDC)

23.1 Overview

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals and the on- and/or off-chip memories. The link between the PDC and a serial peripheral is operated by the AHB to ABP bridge.

The PDC contains 24 channels. The full-duplex peripherals feature 21 mono directional channels used in pairs (transmit only or receive only). The half-duplex peripherals feature 1 bi-directional channel.

The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one set (pointer, counter) for next transfer. The bi-directional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by current transmit, next transmit, current receive and next receive.

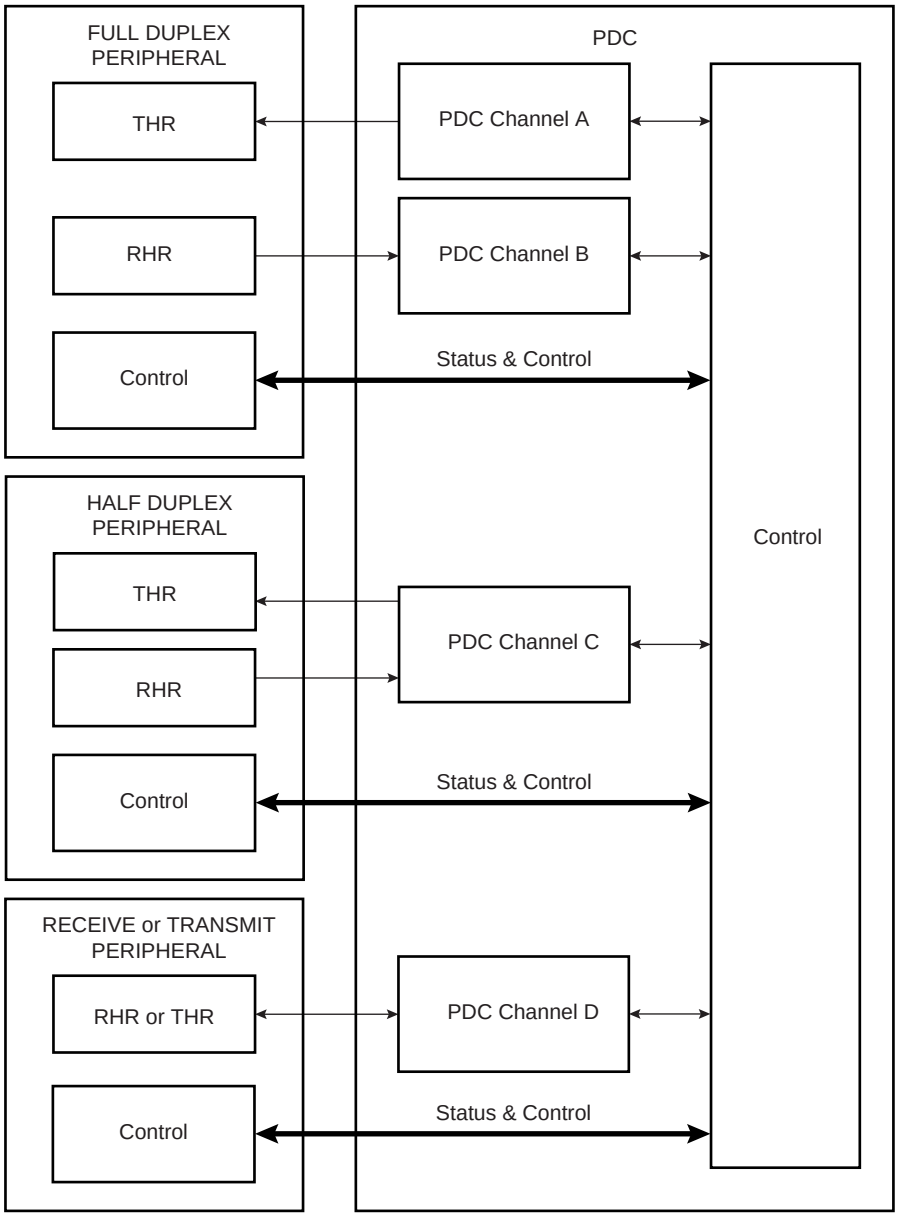
Using the PDC removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.

© 2017 Microchip Technology Inc.

23.2 Block Diagram

Figure 23-1: Block Diagram



23.3 Functional Description

23.3.1 Configuration

The PDC channel user interface enables the user to configure and control data transfers for each channel. The user interface of each PDC channel is integrated into the associated peripheral user interface.

The user interface of a serial peripheral, whether it is full or half duplex, contains four 32-bit pointers (RPR, RNPR, TPR, TNPR) and four 16-bit counter registers (RCR, RNCR, TCR, TNCR). However, the transmit and receive parts of each type are programmed differently: the transmit and receive parts of a full duplex peripheral can be programmed at the same time, whereas only one part (transmit or receive) of a half duplex peripheral can be programmed at a time.

32-bit pointers define the access location in memory for current and next transfer, whether it is for read (transmit) or write (receive). 16-bit counters define the size of current and next transfers. It is possible, at any moment, to read the number of transfers left for each channel.

© 2017 Microchip Technology Inc.

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 associated peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in the peripheral's Transfer Control Register.

At the end of a transfer, the PDC channel sends status flags to its associated peripheral. These flags are visible in the peripheral status register (ENDRX, ENDTX, RXBUFF, and TXBUFE). Refer to Section 23.3.3 Transfer Counters and to the associated peripheral user interface.

23.3.2 Memory Pointers

Each full duplex peripheral is connected to the PDC by a receive channel and a transmit channel. Both channels have 32-bit memory pointers that point respectively to a receive area and to a transmit area in on- and/or off-chip memory.

Each half duplex peripheral is connected to the PDC by a bidirectional channel. This channel has two 32-bit memory pointers, one for current transfer and the other for next transfer. These pointers point to transmit or receive data depending on the operating mode of the peripheral.

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented respectively by 1, 2 or 4 bytes.

If a memory pointer address changes in the middle of a transfer, the PDC channel continues operating using the new address.

23.3.3 Transfer Counters

Each channel has two 16-bit counters, one for current transfer and the other one for next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of next counter is zero, the channel stops transferring data and sets the appropriate flag. But if the next counter value is greater then zero, the values of the next pointer/next counter are copied into the current pointer/current counter and the channel resumes the transfer whereas next pointer/next counter get zero/zero as values. At the end of this transfer the PDC channel sets the appropriate flags in the Peripheral Status Register.

The following list gives an overview of how status register flags behave depending on the counters' values:

- ENDRX flag is set when the PERIPH_RCR reaches zero.
- RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.
- ENDTX flag is set when the PERIPH_TCR 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.

23.3.4 Data Transfers

The serial peripheral triggers its associated PDC channels' transfers using transmit enable (TXEN) and receive enable (RXEN) flags in the transfer control register integrated in the peripheral's user interface.

When the peripheral receives an external data, it sends a Receive Ready signal to its PDC receive channel which then requests access to the Matrix. When access is granted, the PDC receive channel starts reading the peripheral Receive Holding Register (RHR). The read data are stored in an internal buffer and then written to memory.

When the peripheral is about to send data, it sends a Transmit Ready to its PDC transmit channel which then requests access to the Matrix. When access is granted, the PDC transmit channel reads data from memory and puts them to Transmit Holding Register (THR) of its associated peripheral. The same peripheral sends data according to its mechanism.

23.3.5 PDC Flags and Peripheral Status Register

Each peripheral connected to the PDC sends out receive ready and transmit ready flags and the PDC sends back flags to the peripheral. All these flags are only visible in the Peripheral Status Register.

Depending on the type of peripheral, half or full duplex, the flags belong to either one single channel or two different channels.

23.3.5.1 Receive Transfer End

This flag is set when PERIPH_RCR reaches zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_RCR or PERIPH_RNCR.

23.3.5.2 Transmit Transfer End

This flag is set when PERIPH_TCR reaches zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

© 2017 Microchip Technology Inc.

23.3.5.3 Receive Buffer Full

This flag is set when PERIPH_RCR reaches zero with PERIPH_RNCR also set to zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.3.5.4 Transmit Buffer Empty

This flag is set when PERIPH_TCR reaches zero with PERIPH_TNCR also set to zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

© 2017 Microchip Technology Inc.

23.4 Peripheral DMA Controller (PDC) User Interface

Table 23-1: Register Mapping

Offset	Register	Name[1]	Access	Reset
0x100	Receive Pointer Register	PERIPH_RPR	Read/Write	0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0
0x120	Transfer Control Register	PERIPH_PTCR	Write-only	–
0x124	Transfer Status Register	PERIPH_PTSR	Read-only	0

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.)

23.4.1 Receive Pointer Register

Name:PERIPH_RPR

Access: 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 Register

RXPTR must be set to receive buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

© 2017 Microchip Technology Inc.

23.4.2 Receive Counter Register

Name:PERIPH_RCR

Access: 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 Register

RXCTR must be set to receive buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0: Stops peripheral data transfer to the receiver

1–65535 = Starts peripheral data transfer if corresponding channel is active

23.4.3 Transmit Pointer Register

Name:PERIPH_TPR

Access: 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 Counter Register

TXPTR must be set to transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

© 2017 Microchip Technology Inc.

23.4.4 Transmit Counter Register

Name:PERIPH_TCR

Access: 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 Register

TXCTR must be set to transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0: Stops peripheral data transfer to the transmitter

1–65535 = Starts peripheral data transfer if corresponding channel is active

23.4.5 Receive Next Pointer Register

Name:PERIPH_RNPR

Access: 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

RXNPTR contains next receive buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

© 2017 Microchip Technology Inc.

23.4.6 Receive Next Counter Register

Name:PERIPH_RNCR

Access: 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
RXNCTR							
7	6	5	4	3	2	1	0
RXNCTR							

RXNCTR: Receive Next Counter

RXNCTR contains next receive buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

23.4.7 Transmit Next Pointer Register

Name:PERIPH_TNPR

Access: 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

TXNPTR contains next transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

© 2017 Microchip Technology Inc.

23.4.8 Transmit Next Counter Register

Name:PERIPH_TNCR

Access: 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
TXNCTR							
7	6	5	4	3	2	1	0
TXNCTR							

TXNCTR: Transmit Counter Next

TXNCTR contains next transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

23.4.9 Transfer Control Register

Name:PERIPH_PTCR

Access: 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 PDC receiver channel requests if RXTDIS is not set.

When a half duplex peripheral is connected to the PDC, enabling the receiver channel requests automatically disables the transmitter channel requests. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

RXTDIS: Receiver Transfer Disable

0: No effect.

1: Disables the PDC receiver channel requests.

When a half duplex peripheral is connected to the PDC, disabling the receiver channel requests also disables the transmitter channel requests.

TXTEN: Transmitter Transfer Enable

0: No effect.

1: Enables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, it enables the transmitter channel requests only if RXTEN is not set. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

TXTDIS: Transmitter Transfer Disable

0: No effect.

1: Disables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, disabling the transmitter channel requests disables the receiver channel requests.

© 2017 Microchip Technology Inc.

23.4.10 Transfer Status Register

Name:PERIPH_PTSR

Access: 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: PDC Receiver channel requests are disabled.

1: PDC Receiver channel requests are enabled.

TXTEN: Transmitter Transfer Enable

0: PDC Transmitter channel requests are disabled.

1: PDC Transmitter channel requests are enabled.

24. Clock Generator

24.1 Overview

The Clock Generator is made up of 2 PLLs, a Main Oscillator, as well as an RC Oscillator and a 32768 Hz low-power Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the Main Oscillator

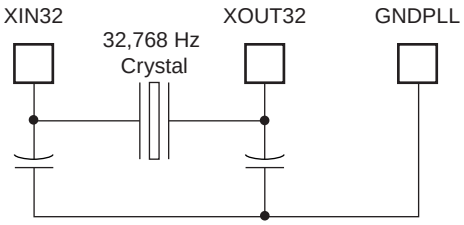
The Clock Generator User Interface is embedded within the Power Management Controller one and is described in Section 25.9 “Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

- PLLACK is the output of the Divider and PLL A block
- PLLBCK is the output of the Divider and PLL B block

24.2 Slow Clock Crystal Oscillator

The Clock Generator integrates a 32768 Hz low-power oscillator. The XIN32 and XOUT32 pins must be connected to a 32768 Hz crystal. Two external capacitors must be wired as shown in Figure 24-1.

Figure 24-1: Typical Slow Clock Crystal Oscillator Connection



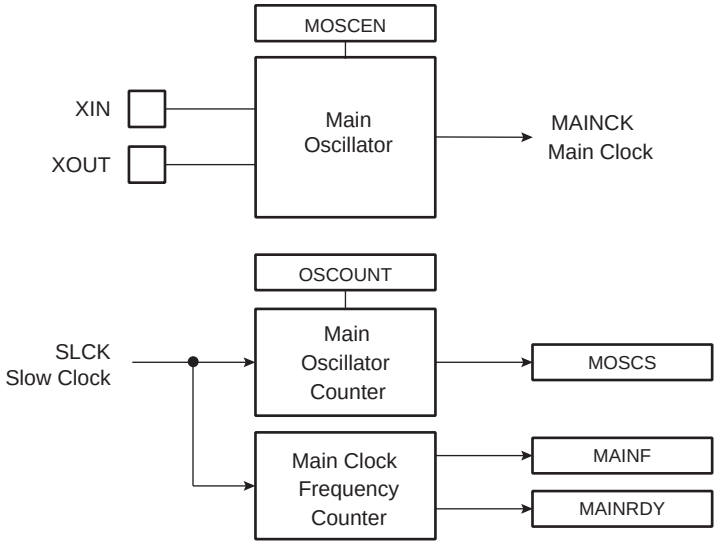
24.3 Slow Clock RC Oscillator

The user has to take into account the possible drifts of the RC Oscillator. More details are given in Section 40.2 “DC Characteristics”.

24.4 Main Oscillator

Figure 24-2 shows the Main Oscillator block diagram.

Figure 24-2: Main Oscillator Block Diagram

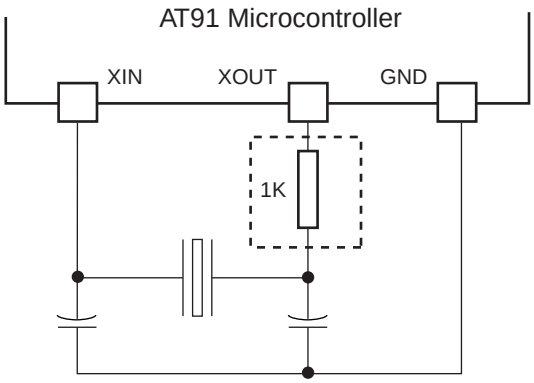


© 2017 Microchip Technology Inc.

24.4.1 Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 3 to 20 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 24-3. For further details on the electrical characteristics of the Main Oscillator, see Section 40.2 “DC Characteristics”.

Figure 24-3: Typical Crystal Connection



24.4.2 Main Oscillator Startup Time

The startup time of the Main Oscillator is given in Section 40.2 “DC Characteristics”. The startup time depends on the crystal frequency and decreases when the frequency rises.

24.4.3 Main Oscillator Control

To minimize the power required to start up the system, the main oscillator is disabled after reset and slow clock is selected.

The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCEN bit in CKGR_MOR, the MOSCS bit in PMC_SR is automatically cleared, indicating the main clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator.

When the MOSCEN bit and the OSCOUNT are written in CKGR_MOR to enable the main oscillator, the MOSCS bit in PMC_SR (Status Register) is cleared and the counter starts counting down on the slow clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCS bit is set, indicating that the main clock is valid. Setting the MOSCS bit in PMC_IMR can trigger an interrupt to the processor.

24.4.4 Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the MAINRDY bit in CKGR_MCFR (Main Clock Frequency Register) 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 crystal connected on the Main Oscillator can be determined.

24.4.5 Main Oscillator Bypass

The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. 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 to 1 and the MOSCEN bit to 0 in the Main OSC register (CKGR_MOR) for the external clock to operate properly.

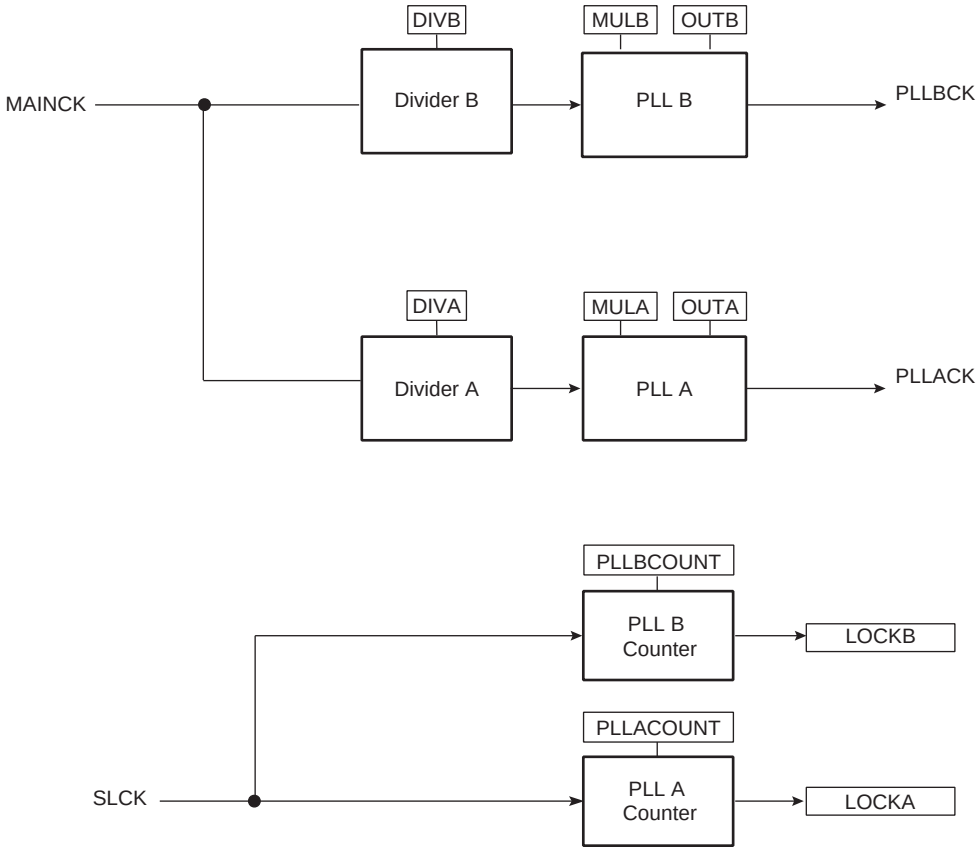
© 2017 Microchip Technology Inc.

24.5 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



24.5.1 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 1. 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 parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. 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 (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), 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. Refer to the PLL Characteristics sub section of the Product Electrical Characteristics.

© 2017 Microchip Technology Inc.

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, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in this datasheet.
- UHP Clock (UHPCK), required by USB Host Port operations.
- 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.

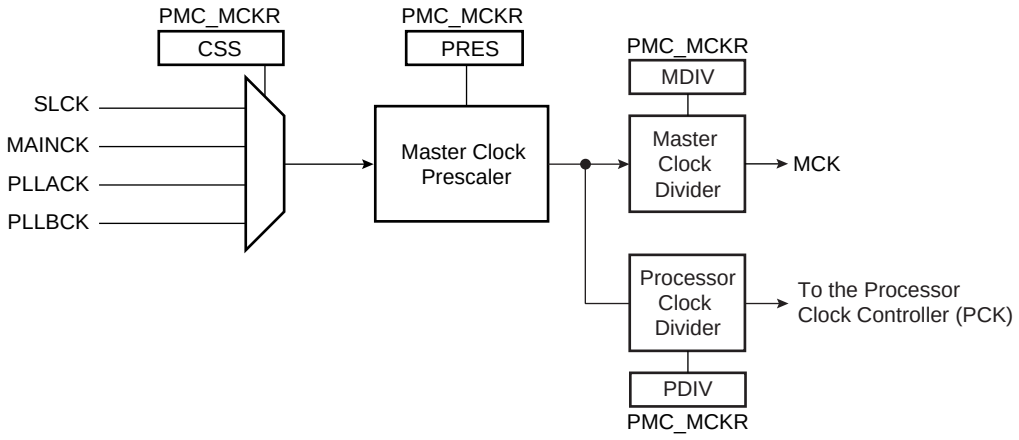
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

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. The Master Clock divider can be programmed through the MDIV field in PMC_MCKR.

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 the 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 PCK 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 and entering Wait for Interrupt Mode. The Processor Clock is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

© 2017 Microchip Technology Inc.

Note: The Arm Wait for Interrupt mode is entered by a CP15 coprocessor operation. Refer to the Microchip application note, Optimizing Power Consumption of AT91SAM9261-based Systems, lit. number 6217.

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.

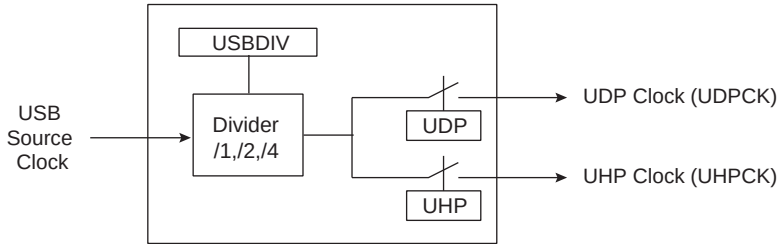
25.4 USB Clock Controller

The USB Source Clock is always generated from the PLL B output. If using the USB, the user must program the PLL to generate a 48 MHz or a 96 MHz signal with an accuracy of ± 0.25% depending on the USBDIV bit in CKGR_PLLBR (see Figure 25-2).

When the PLL B output is stable, i.e., the LOCKB is set:

- The USB host clock can be enabled by setting the UHP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the activity of this clock. The USB host port require both the 12/48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 25-2: USB Clock Controller



25.5 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way 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.

25.6 Programmable Clock Output Controller

The PMC controls 2 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 A output, the PLL B output and the main clock by writing the CSS field in PMC_PCKx. 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.

© 2017 Microchip Technology Inc.

25.7 Programming Sequence

1. Enabling the Main Oscillator:

   The main oscillator is enabled by setting the MOSCEN field in the CKGR_MOR. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the OSCOUNT field in the CKGR_MOR.

   Once this register has been correctly configured, the user must wait for MOSCS field in the PMC_SR to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCS has been enabled in the PMC_IER.

   Code Example:

   ```
   write_register(CKGR_MOR,0x00000701)
   ```

   Start Up Time = 8 * OSCOUNT / SLCK = 56 Slow Clock Cycles.

   So, the main oscillator will be enabled (MOSCS bit set) after 56 Slow Clock Cycles.

2. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main oscillator frequency. This measure can be accomplished via the CKGR_MCFR.

   Once the MAINRDY field is set in CKGR_MCFR, the user may read the MAINF field in CKGR_MCFR. This provides the number of main clock cycles within sixteen slow clock cycles.

3. Setting PLL A and divider A:

   All parameters necessary to configure PLL A and divider A are located in the CKGR_PLLAR.

   It is important to note that Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

   The DIVA field is used to control the divider A itself. The user can program a value between 0 and 255. Divider A output is divider A input divided by DIVA. By default, DIVA parameter is set to 0 which means that divider A is turned off.

   The OUTA field is used to select the PLL A output frequency range.

   The MULA field is the PLL A multiplier factor. This parameter can be programmed between 0 and 254. If MULA is set to 0, PLL A will be turned off. Otherwise PLL A output frequency is PLL A input frequency multiplied by (MULA + 1).

   The PLLACOUNT field specifies the number of slow clock cycles before LOCKA bit is set in the PMC_SR after CKGR_PLLAR has been written.

   Once CKGR_PLLAR has been written, the user is obliged to wait for the LOCKA bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKA has been enabled in the PMC_IER.

   All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage one of the following parameters, MULA, DIVA is modified, LOCKA bit will go low to indicate that PLL A is not ready yet. When PLL A is locked, LOCKA will be set again. User has to wait for LOCKA bit to be set before using the PLL A output clock.

   Code Example:

   ```
   write_register(CKGR_PLLAR,0x20030602)
   ```

   PLL A input clock is main clock divided by 2. PLL An output clock is PLL A input clock multiplied by 4. Once CKGR_PLLAR has been written, LOCKA bit will be set after six slow clock cycles.

4. Setting PLL B and divider B:

   All parameters needed to configure PLL B and divider B are located in the CKGR_PLLBR.

   The DIVB field is used to control divider B itself. A value between 0 and 255 can be programmed. Divider B output is divider B input divided by DIVB parameter. By default DIVB parameter is set to 0 which means that divider B is turned off.

   The OUTB field is used to select the PLL B output frequency range.

   The MULB field is the PLL B multiplier factor. This parameter can be programmed between 0 and 62. If MULB is set to 0, PLL B will be turned off, otherwise the PLL B output frequency is PLL B input frequency multiplied by (MULB + 1).

   The PLLBCOUNT field specifies the number of slow clock cycles before LOCKB bit is set in the PMC_SR after CKGR_PLLBR has been written.

© 2017 Microchip Technology Inc.

Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s).

Code Example:

```
write_register(CKGR_PLLBR,0x20030602)
```

PLL B input clock is main clock divided by 2. PLL B output clock is PLL B input clock multiplied by 4. Once CKGR_PLLBR has been written, LOCKB bit will be set after six slow clock cycles.

5. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR.

The CSS field is used to select the clock source of the Master Clock and Processor Clock dividers. By default, the selected clock source is slow clock.

The PRES field is used to control the Master/Processor Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Prescaler output is the selected clock source divided by PRES parameter. By default, PRES parameter is set to 1 which means that the input clock of the Master Clock and Processor Clock dividers is equal to slow clock.

The MDIV field is used to control the Master Clock divider. It is possible to choose between different values (0, 1, 2, 3). The Master Clock output is Master/Processor Clock Prescaler output divided by 1, 2, 4 or 6, depending on the value programmed in MDIV.

The PDIV field is used to control the Processor Clock divider. It is possible to choose between different values (0, 1). The Processor Clock output is Master/Processor Clock Prescaler output divided by 1 or 2, depending on the value programmed in PDIV.

By default, MDIV and PDIV are set to 0, which indicates that Processor Clock is equal to the Master Clock.

Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. 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.

The PMC_MCKR must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.

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 (CKGR_PLLAR or CKGR_PLLBR), the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK (LOCKA or LOCKB) goes high and MCKRDY is set. While PLLA is unlocked, the Master Clock selection is automatically changed to Slow Clock. While PLLB is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 25.8.2 “Clock Switching Waveforms”.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
        wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
```


© 2017 Microchip Technology Inc.

```
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

6. 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. Depending on the system used, 2 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. Four clock options are available: main clock, slow clock PLLACK, PLLBCK. 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 1 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. 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. 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.

7. 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.

   Depending on the system used, 13 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 25-1and Table 25-2 give 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 / To	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Note 1: PLL designates either the PLL A or the PLL B Clock.

2: PLLCOUNT designates either PLLACOUNT or PLLBCOUNT.

Table 25-2: Clock Switching Timings (Worst Case)

From / To	PLLA Clock	PLLB Clock
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
PLLB Clock	3 x PLLB Clock + 4 x SLCK + 1.5 x PLLB Clock	2.5 x PLLB Clock + 4 x SLCK + PLLBCOUNT x SLCK

© 2017 Microchip Technology Inc.

25.8.2 Clock Switching Waveforms

Figure 25-3: Switch Master Clock from Slow Clock to PLL Clock

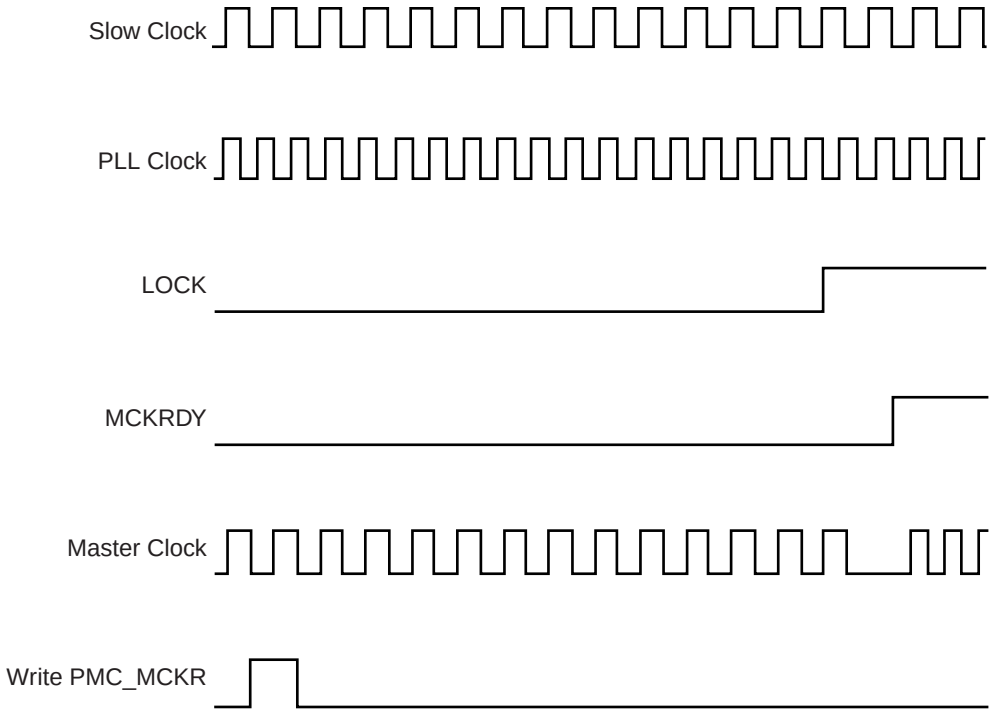


Figure 25-4: Switch Master Clock from Main Clock to Slow Clock

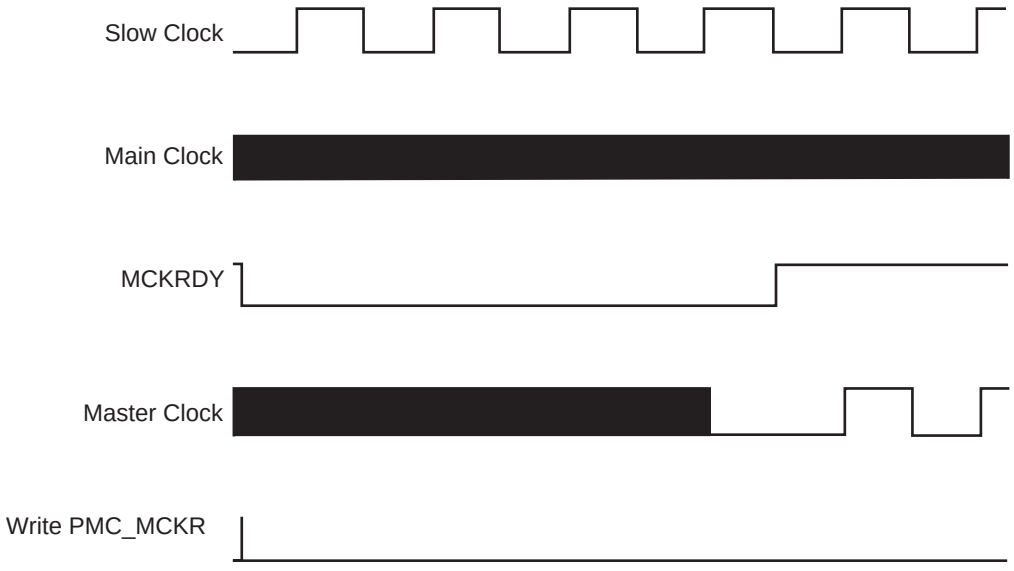


Figure 25-5: Change PLLA Programming

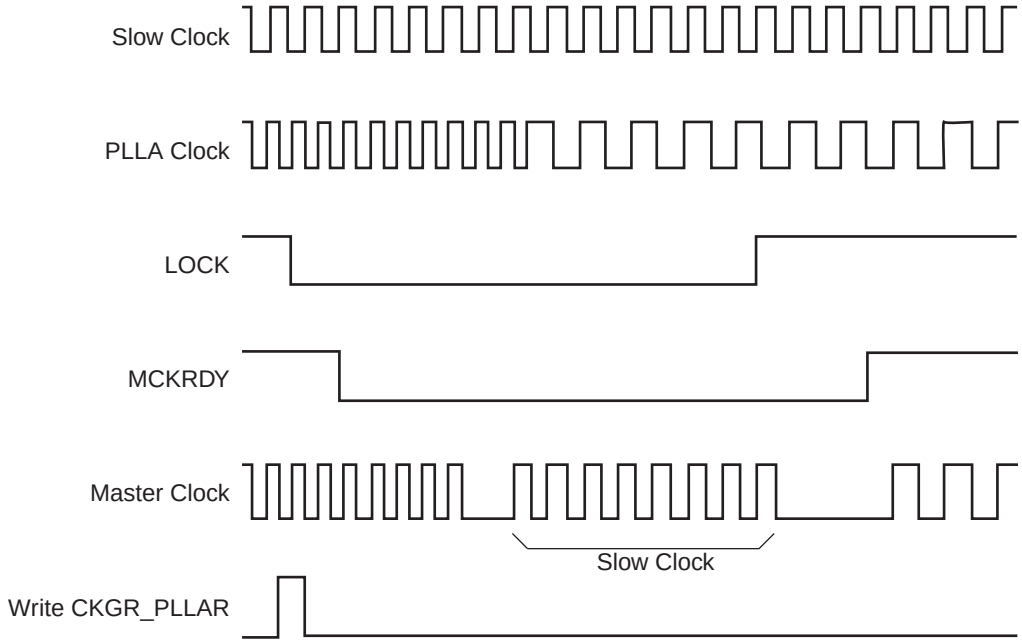
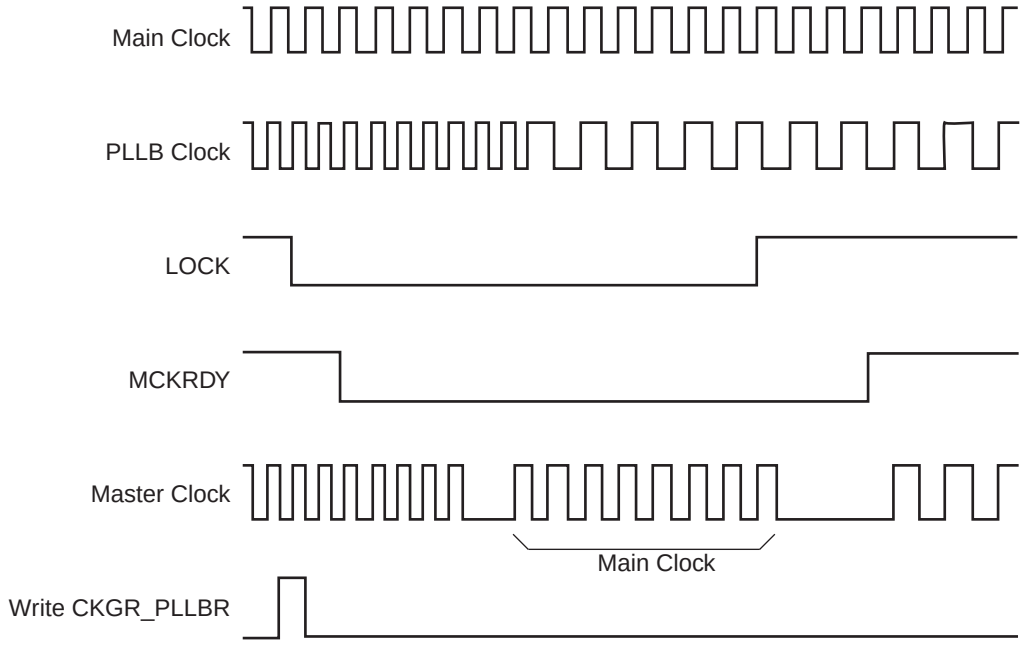
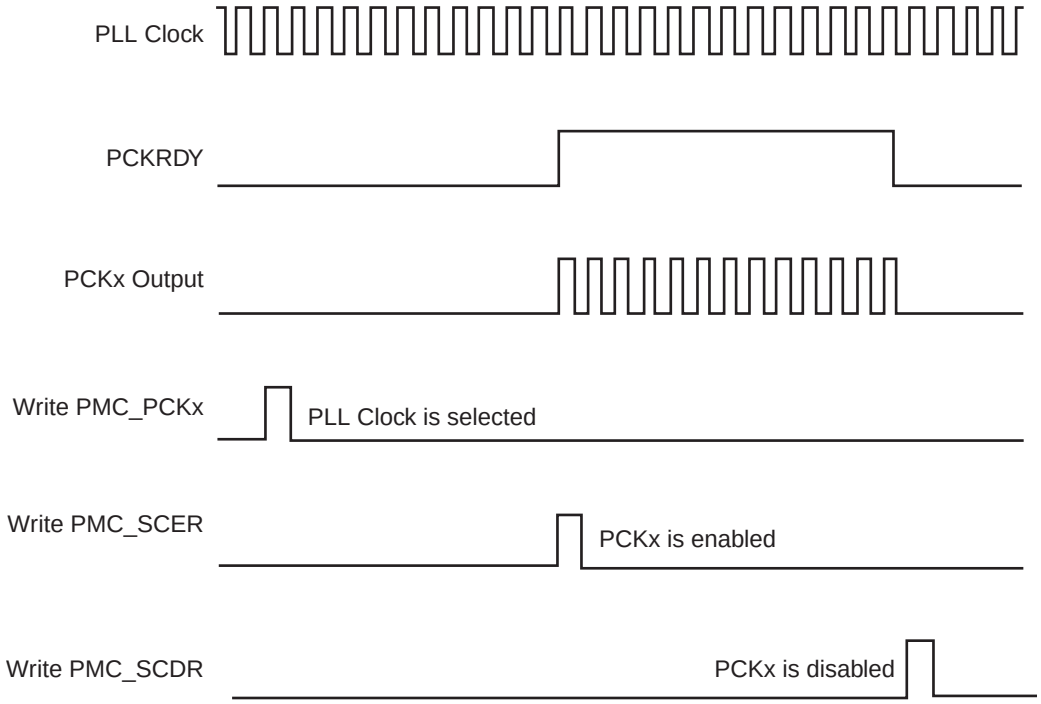


Figure 25-6: Change PLLB Programming



© 2017 Microchip Technology Inc.

Figure 25-7: Programmable Clock Output Programming



25.9 Power Management Controller (PMC) User Interface

Table 25-3: 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	0x03
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL A Register	CKGR_PLLAR	Read/Write	0x3F00
0x002C	PLL B Register	CKGR_PLLBR	Read/Write	0x3F00
0x0030	Master Clock Register	PMC_MCKR	Read/Write	0x0
0x0038	Reserved	–	–	–
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read/Write	0x0
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read/Write	0x0
0x0048–0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	–
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	–
0x0068	Status Register	PMC_SR	Read-only	0x08
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070–0x007C	Reserved	–	–	–
0x0080	PLL Charge Pump Current Register	PMC_PLLICPR	Read/Write	0x0

© 2017 Microchip Technology Inc.

25.9.1 PMC System Clock Enable Register

Name:PMC_SCER

Access: 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	–

UHP: USB Host Port Clock Enable

0: No effect.

1: Enables the 12 and 48 MHz clock of the USB Host Port.

UDP: USB Device Port Clock Enable

0: No effect.

1: Enables the 48 MHz clock of the USB Device Port.

PCKx: Programmable Clock x Output Enable

0: No effect.

1: Enables the corresponding Programmable Clock output.

25.9.2 PMC System Clock Disable Register

Name:PMC_SCDR

Access: 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

PCK: Processor Clock Disable

0: No effect.

1: Disables the Processor clock. This is used to enter the processor in Idle Mode.

UHP: USB Host Port Clock Disable

0: No effect.

1: Disables the 12 and 48 MHz clock of the USB Host Port.

UDP: USB Device Port Clock Disable

0: No effect.

1: Disables the 48 MHz clock of the USB Device Port.

PCKx: Programmable Clock x Output Disable

0: No effect.

1: Disables the corresponding Programmable Clock output.

© 2017 Microchip Technology Inc.

25.9.3 PMC System Clock Status Register

Name:PMC_SCSR

Access: 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

PCK: Processor Clock Status

0: The Processor clock is disabled.

1: The Processor clock is enabled.

UHP: USB Host Port Clock Status

0: The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1: The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

UDP: USB Device Port Clock Status

0: The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1: The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

PCKx: Programmable Clock x Output Status

0: The corresponding Programmable Clock output is disabled.

1: The corresponding Programmable Clock output is enabled.

© 2017 Microchip Technology Inc.

25.9.4 PMC Peripheral Clock Enable Register

Name:PMC_PCER

Access: 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 Section 9.2 “Peripheral Identifiers”.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

© 2017 Microchip Technology Inc.

25.9.5 PMC Peripheral Clock Disable Register

Name:PMC_PCDR

Access: 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 Section 9.2 “Peripheral Identifiers”.

25.9.6 PMC Peripheral Clock Status Register

Name:PMC_PCSR

Access: 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 Section 9.2 “Peripheral Identifiers”.

© 2017 Microchip Technology Inc.

25.9.7 PMC Clock Generator Main Oscillator Register

Name:CKGR_MOR

Access: 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
OSCOUNT							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	OSCBYPASS	MOSCEN

MOSCEN: Main Oscillator Enable

A crystal must be connected between XIN and XOUT.

0: The Main Oscillator is disabled.

1: The Main Oscillator is enabled. OSCBYPASS must be set to 0.

When MOSCEN is set, the MOSCS flag is set once the Main Oscillator startup time is achieved.

OSCBYPASS: Oscillator Bypass

0: No effect.

1: The Main Oscillator is bypassed. MOSCEN must be set to 0. An external clock must be connected on XIN.

When OSCBYPASS is set, the MOSCS flag in PMC_SR is automatically set.

Clearing MOSCEN and OSCBYPASS bits allows resetting the MOSCS flag.

OSCOUNT: Main Oscillator Start-up Time

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Oscillator start-up time.

25.9.8 PMC Clock Generator Main Clock Frequency Register

Name:CKGR_MCFR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	MAINRDY
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.

MAINRDY: 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.

© 2017 Microchip Technology Inc.

25.9.9 PMC Clock Generator PLL A Register

Name:CKGR_PLLAR

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	1	–	–	–	–	–

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

DIVA: Divider A

DIVA	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2–255	Divider output is the Main Clock divided by DIVA.

PLLACOUNT: PLL A Counter

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

OUTA: PLL A Clock Frequency Range

To optimize clock performance, this field must be programmed as specified in Section 40.5.7 “PLL Characteristics”.

MULA: PLL A Multiplier

0: The PLL A is deactivated.

1– 254: The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

© 2017 Microchip Technology Inc.

25.9.10 PMC Clock Generator PLL B Register

Name:CKGR_PLLBR

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	USBDIV		–	–	–	–

23	22	21	20	19	18	17	16
–	–	MULB					

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

Possible limitations on PLL B input frequencies and multiplier factors should be checked before using the PMC.

DIVB: Divider B

DIVB	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2–255	Divider output is the selected clock divided by DIVB.

PLLBCOUNT: PLL B Counter

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC_SR after CKGR_PLLBR is written.

OUTB: PLL B Clock Frequency Range

To optimize clock performance, this field must be programmed as specified in Section 40.5.7 “PLL Characteristics”.

MULB: PLL B Multiplier

0: The PLL B is deactivated.

1–62: The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

USBDIV: Divider for USB Clock.

USBDIV		Divider Selected
0	0	Divider output is PLL B clock output
0	1	Divider output is PLL B clock output divided by 2
1	0	Divider output is PLL B clock output divided by 4
1	1	Reserved

© 2017 Microchip Technology Inc.

25.9.11 PMC Master Clock Register

Name:PMC_MCKR

Access: 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
–	–	–	PDIV	–	–	MDIV	
7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

CSS: Master/Processor Clock Source Selection

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLLA Clock is selected
1	1	PLLB Clock is selected

PRES: Master/Processor Clock Prescaler

PRES			Master/Processor Clock Dividers Input 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

MDIV: Master Clock Division

MDIV		Master Clock Division
0	0	Master Clock is Prescaler Output Clock divided by 1.
0	1	Master Clock is Prescaler Output Clock divided by 2.
1	0	Master Clock is Prescaler Output Clock divided by 4.
1	1	Master Clock is Prescaler Output Clock divided by 6.

© 2017 Microchip Technology Inc.

PDIV: Processor Clock Division

PDIV	Processor Clock Division
0	Processor Clock is Prescaler Output Clock divided by 1.
1	Processor Clock is Prescaler Output Clock divided by 2.

Warning: If MDIV is written to 0, the write in PDIV is not taken in account and PDIV is forced to 0 (The Master Clock frequency cannot be superior to the Processor Clock frequency).

© 2017 Microchip Technology Inc.

25.9.12 PMC Programmable Clock Register

Name:PMC_PCKx

Access: 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 A Clock is selected
1	1	PLL B Clock is selected

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

© 2017 Microchip Technology Inc.

25.9.13 PMC Interrupt Enable Register

Name:PMC_IER

Access: 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

MOSCS: Main Oscillator Status Interrupt Enable

LOCKA: PLL A Lock Interrupt Enable

LOCKB: PLL B Lock Interrupt Enable

MCKRDY: Master Clock Ready Interrupt Enable

PCKRDYx: Programmable Clock Ready x Interrupt Enable

0: No effect.

1: Enables the corresponding interrupt.

© 2017 Microchip Technology Inc.

25.9.14 PMC Interrupt Disable Register

Name:PMC_IDR

Access: 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

MOSCS: Main Oscillator Status Interrupt Disable

LOCKA: PLL A Lock Interrupt Disable

LOCKB: PLL B Lock Interrupt Disable

MCKRDY: Master Clock Ready Interrupt Disable

PCKRDYx: Programmable Clock Ready x Interrupt Disable

0: No effect.

1: Disables the corresponding interrupt.

25.9.15 PMC Status Register

Name:PMC_SR

Access: 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
OSC_SEL	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

MOSCS: Main Oscillator Status

0: Main oscillator is not stabilized.

1: Main oscillator is stabilized.

LOCKA: PLL A Lock Status

0: PLL A is not locked

1: PLL A is locked.

LOCKB: PLL B Lock Status

0: PLL B is not locked.

1: PLL B is locked.

MCKRDY: Master Clock Status

0: Master Clock is not ready.

1: Master Clock is ready.

OSC_SEL: Slow Clock Oscillator Selection

0: Internal slow clock RC oscillator.

1: External slow clock 32 kHz oscillator.

PCKRDYx: Programmable Clock Ready Status

0: Programmable Clock x is not ready.

1: Programmable Clock x is ready.

© 2017 Microchip Technology Inc.

25.9.16 PMC Interrupt Mask Register

Name:PMC_IMR

Access: 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

MOSCS: Main Oscillator Status Interrupt Mask

LOCKA: PLL A Lock Interrupt Mask

LOCKB: PLL B Lock Interrupt Mask

MCKRDY: Master Clock Ready Interrupt Mask

PCKRDYx: Programmable Clock Ready Interrupt Mask

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

25.9.17 PLL Charge Pump Current Register

Name:PMC_PLLICPR

Access: Read/Write

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

ICPLLA: Charge Pump Current

To optimize clock performance, this field must be programmed as specified in Section 40.5.7 “PLL Characteristics”.

ICPLLB: Charge Pump Current

Must be set to 0.

© 2017 Microchip Technology Inc.

26. Advanced Interrupt Controller (AIC)

26.1 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.

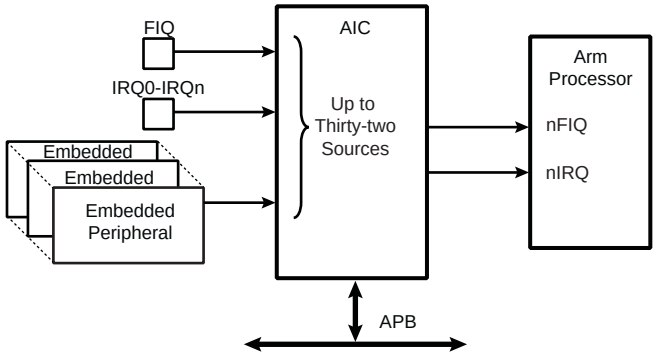
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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.

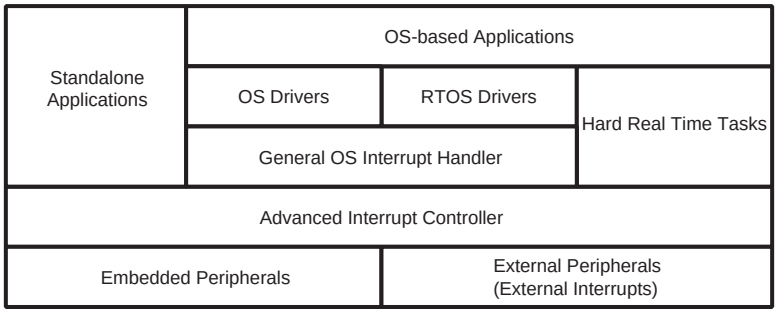
26.2 Block Diagram

Figure 26-1: Block Diagram



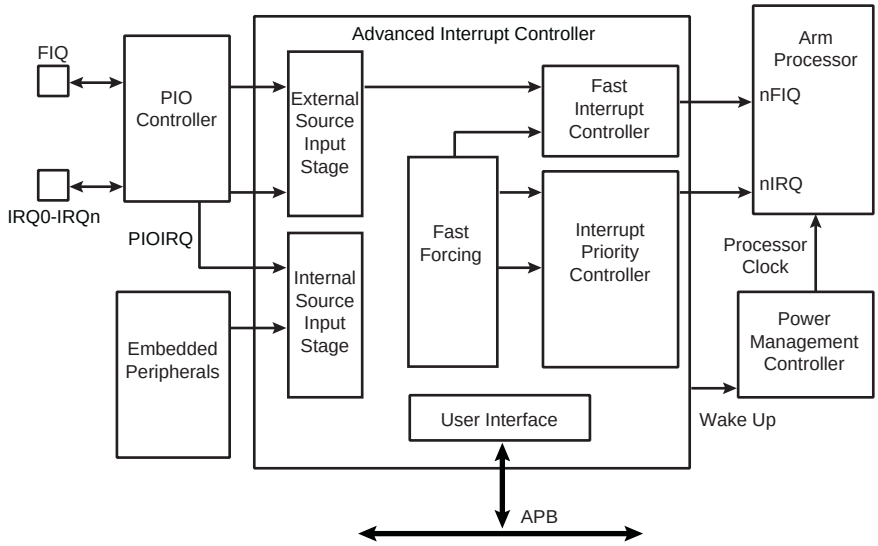
26.3 Application Block Diagram

Figure 26-2: Description of the Application Block



26.4 AIC Detailed Block Diagram

Figure 26-3: AIC Detailed Block Diagram



26.5 I/O Line Description

Table 26-1: I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0–IRQn	Interrupt 0–Interrupt n	Input

26.6 Product Dependencies

26.6.1 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.

26.6.2 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.

26.6.3 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. 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.

© 2017 Microchip Technology Inc.

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.

26.7 Functional Description

26.7.1 Interrupt Source Control

26.7.1.1 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.

26.7.1.2 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. A disabled interrupt does not affect servicing of other interrupts.

26.7.1.3 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. (Section 26.7.3.1 Priority Controller) 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 Section 26.7.4.5 Fast Forcing)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

26.7.1.4 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.

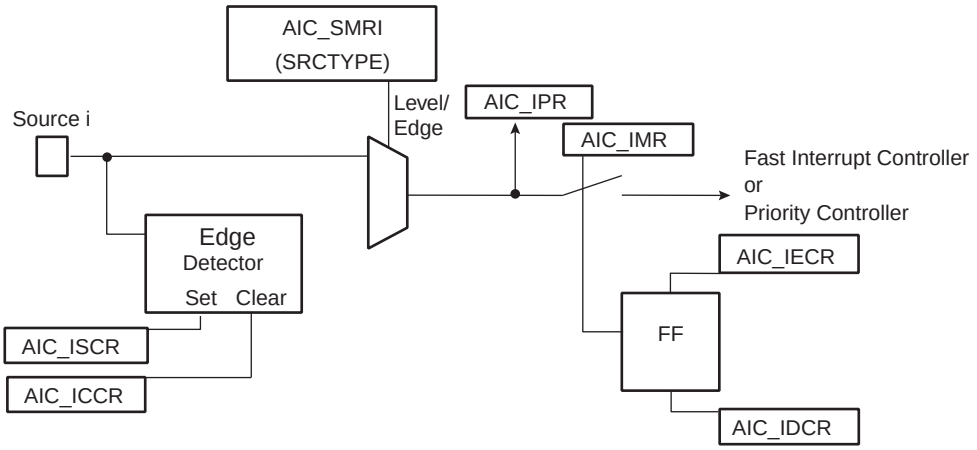
The AIC_ISR reads the number of the current interrupt (see Section 26.7.3.1 Priority Controller) and the register AIC_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

© 2017 Microchip Technology Inc.

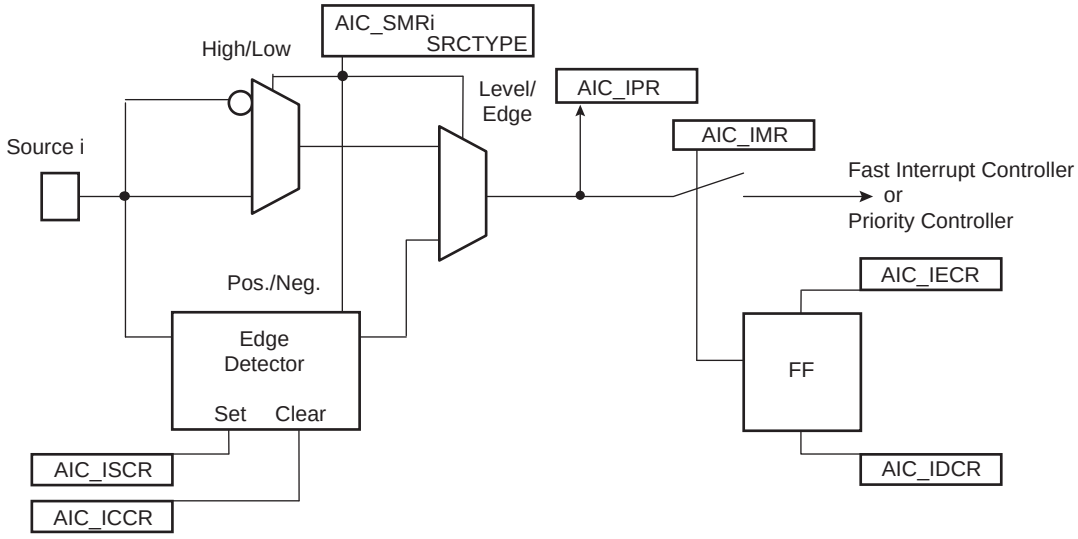
26.7.1.5 Internal Interrupt Source Input Stage

Figure 26-4: Internal Interrupt Source Input Stage



26.7.1.6 External Interrupt Source Input Stage

Figure 26-5: External Interrupt Source Input Stage



26.7.2 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.

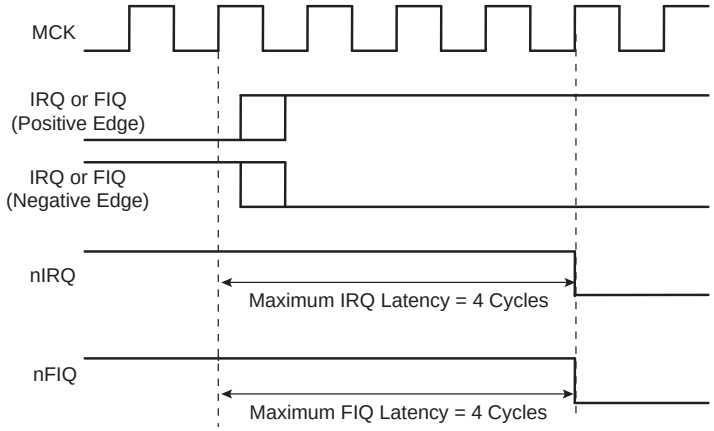
This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The 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.

© 2017 Microchip Technology Inc.

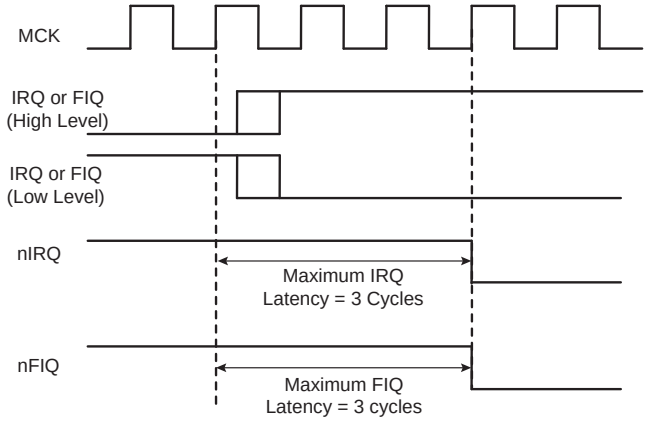
26.7.2.1 External Interrupt Edge Triggered Source

Figure 26-6: External Interrupt Edge Triggered Source



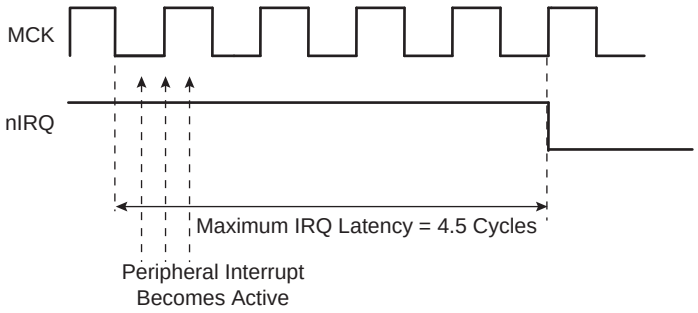
26.7.2.2 External Interrupt Level Sensitive Source

Figure 26-7: External Interrupt Level Sensitive Source



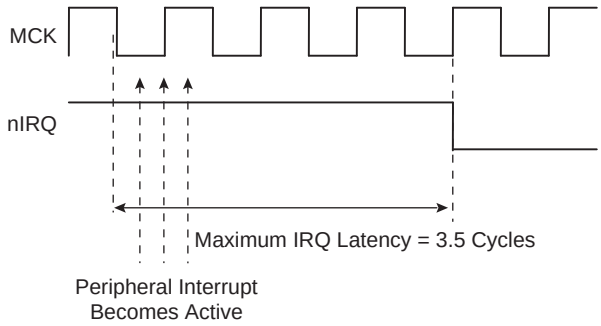
26.7.2.3 Internal Interrupt Edge Triggered Source

Figure 26-8: Internal Interrupt Edge Triggered Source



26.7.2.4 Internal Interrupt Level Sensitive Source

Figure 26-9: Internal Interrupt Level Sensitive Source



26.7.3 Normal Interrupt

26.7.3.1 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**.

26.7.3.2 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.

26.7.3.3 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.

© 2017 Microchip Technology Inc.

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.

26.7.3.4 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).

© 2017 Microchip Technology Inc.

26.7.4 Fast Interrupt

26.7.4.1 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.

26.7.4.2 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.

26.7.4.3 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.

26.7.4.4 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.

© 2017 Microchip Technology Inc.

26.7.4.5 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).

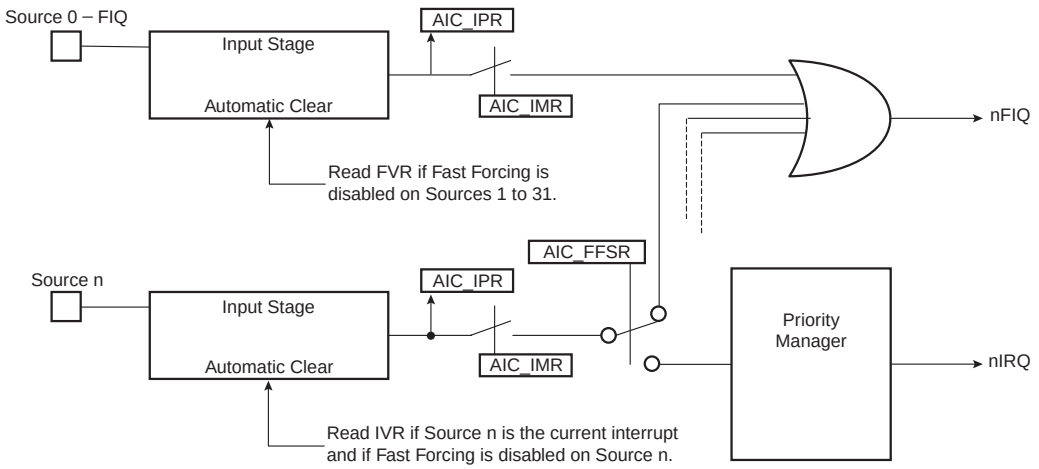
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt 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 26-10: Fast Forcing



26.7.5 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 PROT in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

© 2017 Microchip Technology Inc.

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.

26.7.6 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.

26.7.7 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.

© 2017 Microchip Technology Inc.

26.8 Advanced Interrupt Controller (AIC) User Interface

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 26-2: 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	---	---	---

Note 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 Section 9.2 “Peripheral Identifiers”.

3: Values in the Version Register vary with the version of the IP block implementation.

26.8.1 AIC Source Mode Register

Name:AIC_SMR0..AIC_SMR31

Access: 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
–	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

© 2017 Microchip Technology Inc.

26.8.2 AIC Source Vector Register

Name:AIC_SVR0..AIC_SVR31

Access: Read/Write

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.

© 2017 Microchip Technology Inc.

26.8.3 AIC Interrupt Vector Register

Name: AIC_IVR

Access: Read-only

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.

© 2017 Microchip Technology Inc.

26.8.4 AIC FIQ Vector Register

Name: AIC_FVR

Access: Read-only

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.

26.8.5 AIC Interrupt Status Register

Name: AIC_ISR

Access: 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
–	–	–	IRQID				

IRQID: Current Interrupt Identifier

The Interrupt Status Register returns the current interrupt source number.

© 2017 Microchip Technology Inc.

26.8.6 AIC Interrupt Pending Register

Name: AIC_IPR

Access: 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	FIQ

FIQ, SYS, PID2–PID31: Interrupt Pending

0: Corresponding interrupt is not pending.

1: Corresponding interrupt is pending.

26.8.7 AIC Interrupt Mask Register

Name:AIC_IMR

Access: 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	FIQ

FIQ, SYS, PID2–PID31: Interrupt Mask

0: Corresponding interrupt is disabled.

1: Corresponding interrupt is enabled.

© 2017 Microchip Technology Inc.

26.8.8 AIC Core Interrupt Status Register

Name: AIC_CISR

Access: 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
–	–	–	–	–	–	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.

26.8.9 AIC Interrupt Enable Command Register

Name: AIC_IECR

Access: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.

© 2017 Microchip Technology Inc.

26.8.10 AIC Interrupt Disable Command Register

Name: AIC_IDCR

Access: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.

26.8.11 AIC Interrupt Clear Command Register

Name:AIC_ICCR

Access: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.

© 2017 Microchip Technology Inc.

26.8.12 AIC Interrupt Set Command Register

Name: AIC_ISCR

Access: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.

26.8.13 AIC End of Interrupt Command Register

Name:AIC_EOICR

Access: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.

© 2017 Microchip Technology Inc.

26.8.14 AIC Spurious Interrupt Vector Register

Name:AIC_SPU

Access:Read/Write

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.

26.8.15 AIC Debug Control Register

Name:AIC_DCR

Access: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
–	–	–	–	–	–	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.

© 2017 Microchip Technology Inc.

26.8.16 AIC Fast Forcing Enable Register

Name:AIC_FFER

Access: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.

26.8.17 AIC Fast Forcing Disable Register

Name:AIC_FFDR

Access: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.

© 2017 Microchip Technology Inc.

26.8.18 AIC Fast Forcing Status Register

Name:AIC_FFSR

Access: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.

27. Debug Unit (DBGU)

27.1 Overview

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Microchip'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. The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication. 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 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.

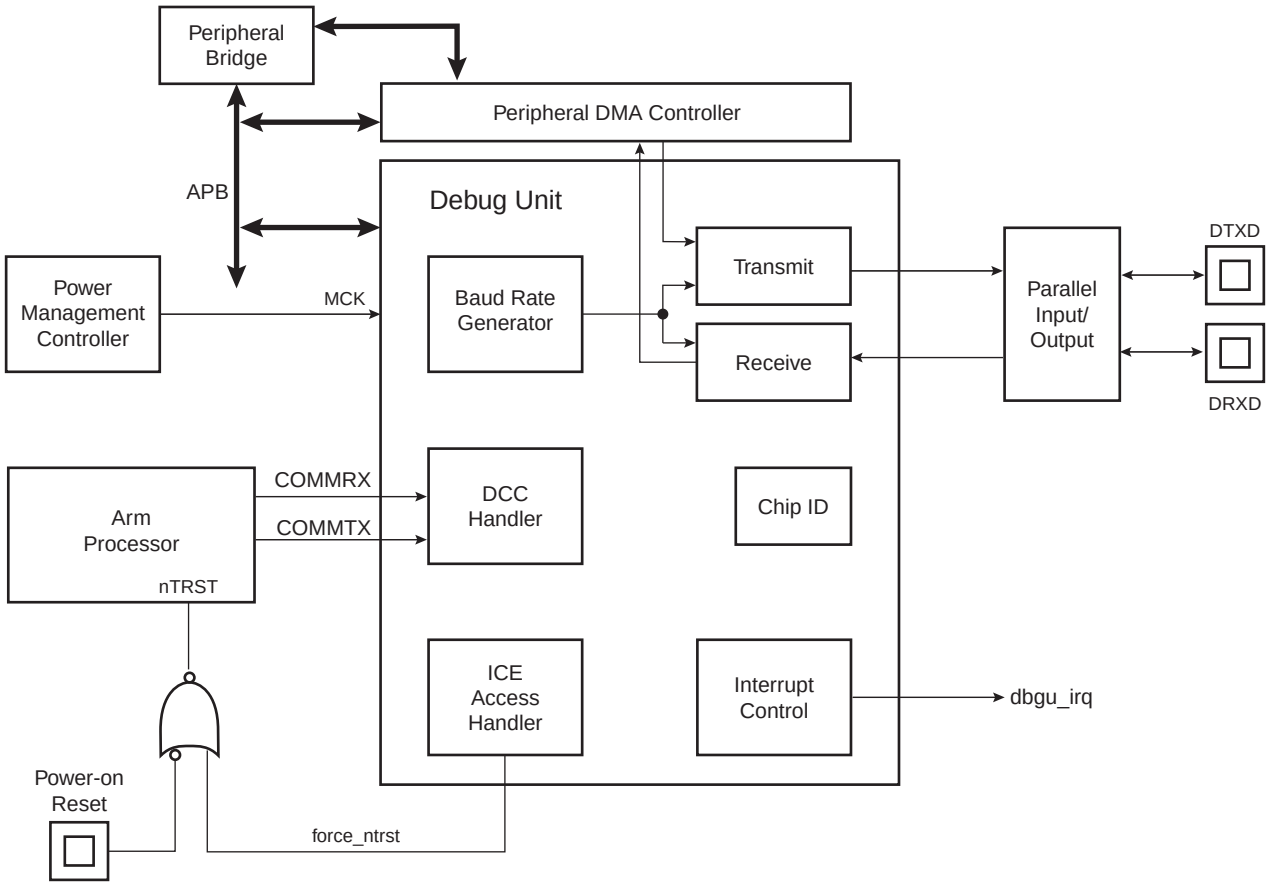
The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the Arm processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the Arm processor, making possible the handling 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.

27.2 Block Diagram

Figure 27-1: Debug Unit Functional Block Diagram

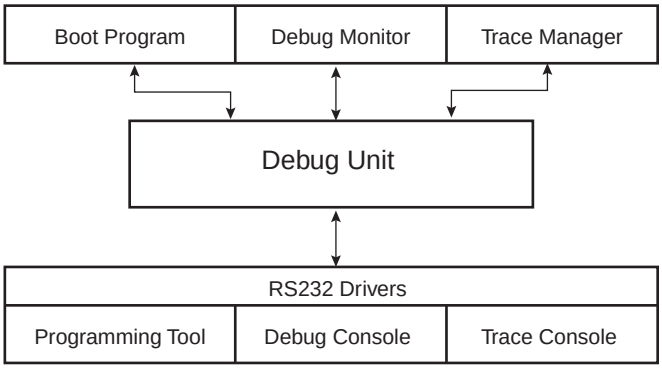


© 2017 Microchip Technology Inc.

Table 27-1: Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 27-2: Debug Unit Application Example



27.3 Product Dependencies

27.3.1 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.

27.3.2 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.

27.3.3 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 27-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

27.4 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.

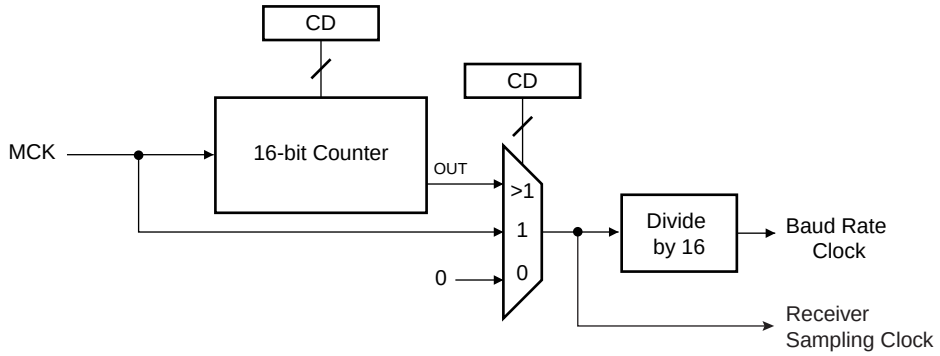
27.4.1 Baud Rate Generator

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 27-3: Baud Rate Generator



27.4.2 Receiver

27.4.2.1 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.

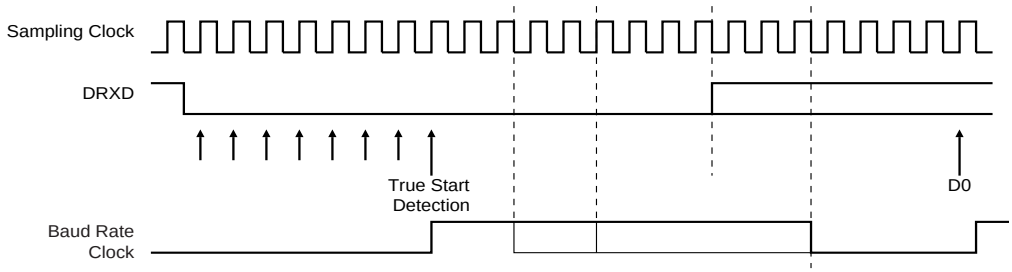
27.4.2.2 Start Detection and Data Sampling

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 midpoint 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.

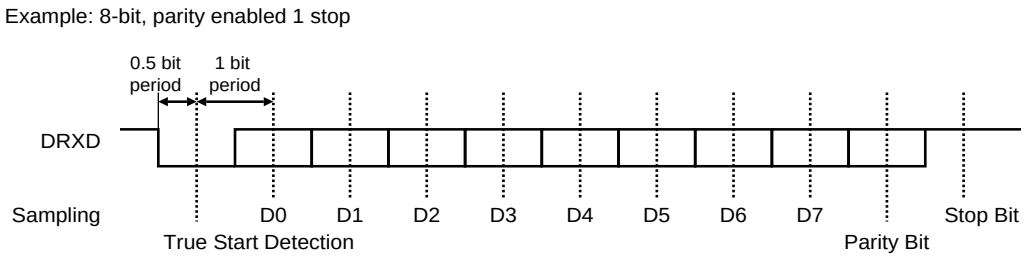
Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 27-4: Start Bit Detection



© 2017 Microchip Technology Inc.

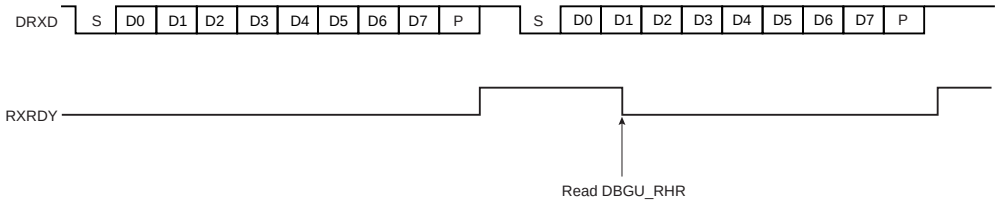
Figure 27-5: Character Reception



27.4.2.3 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.

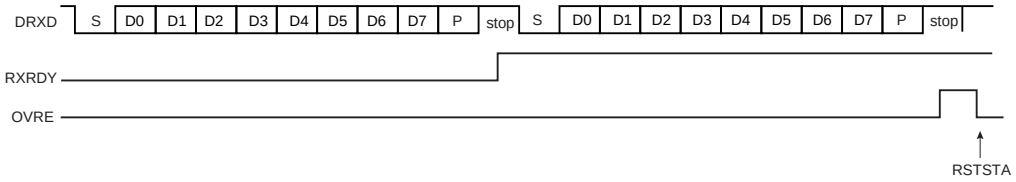
Figure 27-6: Receiver Ready



27.4.2.4 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.

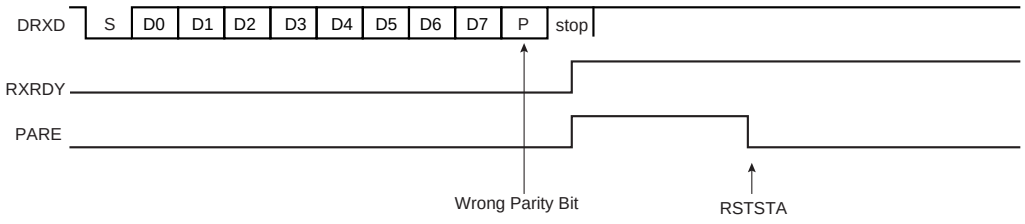
Figure 27-7: Receiver Overrun



27.4.2.5 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.

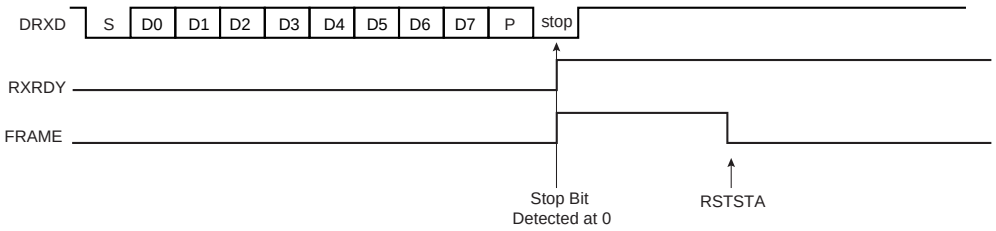
Figure 27-8: Parity Error



27.4.2.6 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 27-9: Receiver Framing Error



27.4.3 Transmitter

27.4.3.1 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.

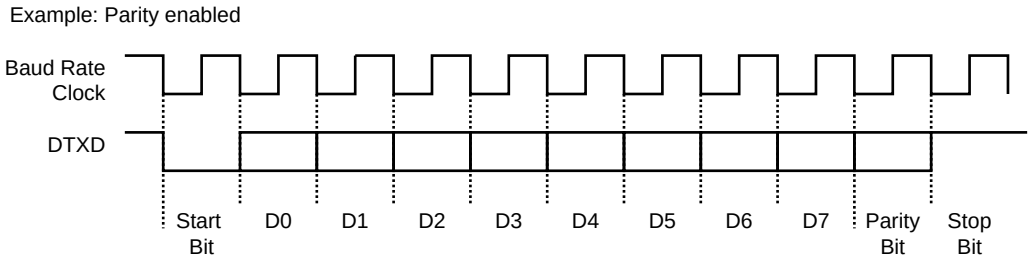
The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters 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.

27.4.3.2 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 27-10: Character Transmission



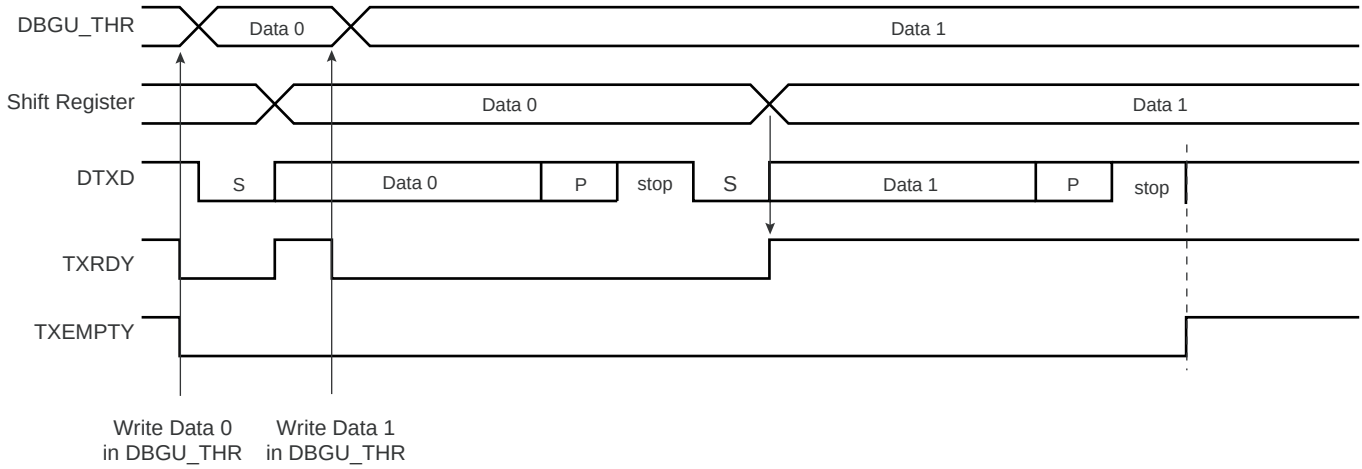
27.4.3.3 Transmitter Control

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.

© 2017 Microchip Technology Inc.

Figure 27-11: Transmitter Control



27.4.4 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.

27.4.5 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.

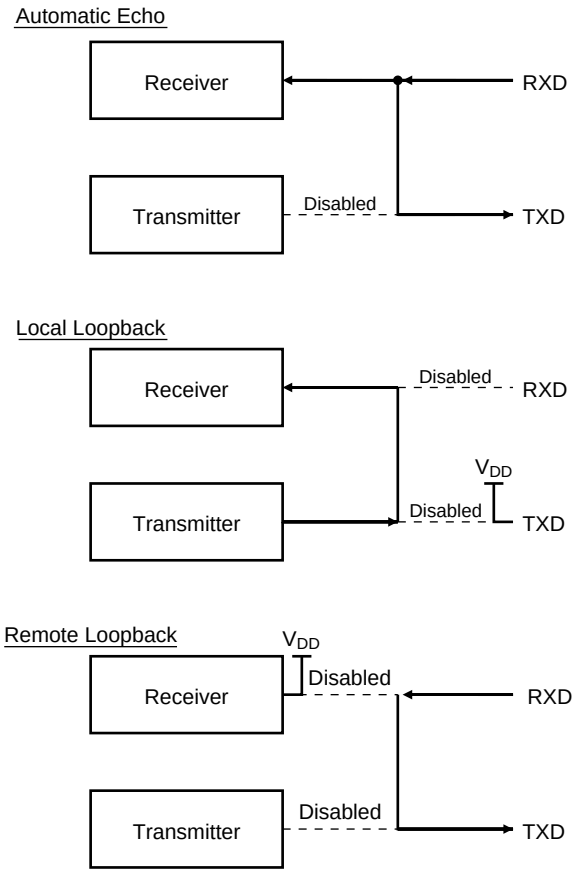
The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

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.

© 2017 Microchip Technology Inc.

Figure 27-12: Test Modes



27.4.6 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.

© 2017 Microchip Technology Inc.

27.4.7 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.

27.4.8 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.

27.5 Debug Unit (DBGU) User Interface

Table 27-2: 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
0x004C–0x00FC	Reserved	–	–	–
0x0100–0x0124	PDC Area	–	–	–

© 2017 Microchip Technology Inc.

27.5.1 Debug Unit Control Register

Name:DBGU_CR

Access: 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.

27.5.2 Debug Unit Mode Register

Name:DBGU_MR

Access: 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

© 2017 Microchip Technology Inc.

27.5.3 Debug Unit Interrupt Enable Register

Name:DBGU_IER

Access: 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.

27.5.4 Debug Unit Interrupt Disable Register

Name:DBGU_IDR

Access: 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.

© 2017 Microchip Technology Inc.

27.5.5 Debug Unit Interrupt Mask Register

Name:DBGU_IMR

Access: 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.

27.5.6 Debug Unit Status Register

Name:DBGU_SR

Access: 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.

© 2017 Microchip Technology Inc.

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.

27.5.7 Debug Unit Receiver Holding Register

Name:DBGU_RHR

Access: 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.

© 2017 Microchip Technology Inc.

27.5.8 Debug Unit Transmit Holding Register

Name:DBGU_THR

Access: 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.

27.5.9 Debug Unit Baud Rate Generator Register

Name:DBGU_BRGR

Access: 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
2–65535	MCK / (CD x 16)

© 2017 Microchip Technology Inc.

27.5.10 Debug Unit Chip ID Register

Name:DBGU_CIDR

Access: 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
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

© 2017 Microchip Technology Inc.

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
1	1	0	0	128K bytes
1	1	0	1	256K bytes
1	1	1	0	96K bytes
1	1	1	1	512K bytes

© 2017 Microchip Technology Inc.

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 NVPSIZ is ROM size NVPSIZ2 is Flash size

EXT: Extension Flag

0: Chip ID has a single register definition without extension

1: An extended Chip ID exists.

27.5.11 Debug Unit Chip ID Extension Register

Name:DBGU_EXID

Access: 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.

© 2017 Microchip Technology Inc.

27.5.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Access: 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.

© 2017 Microchip Technology Inc.

28. Parallel Input Output Controller (PIO)

28.1 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.

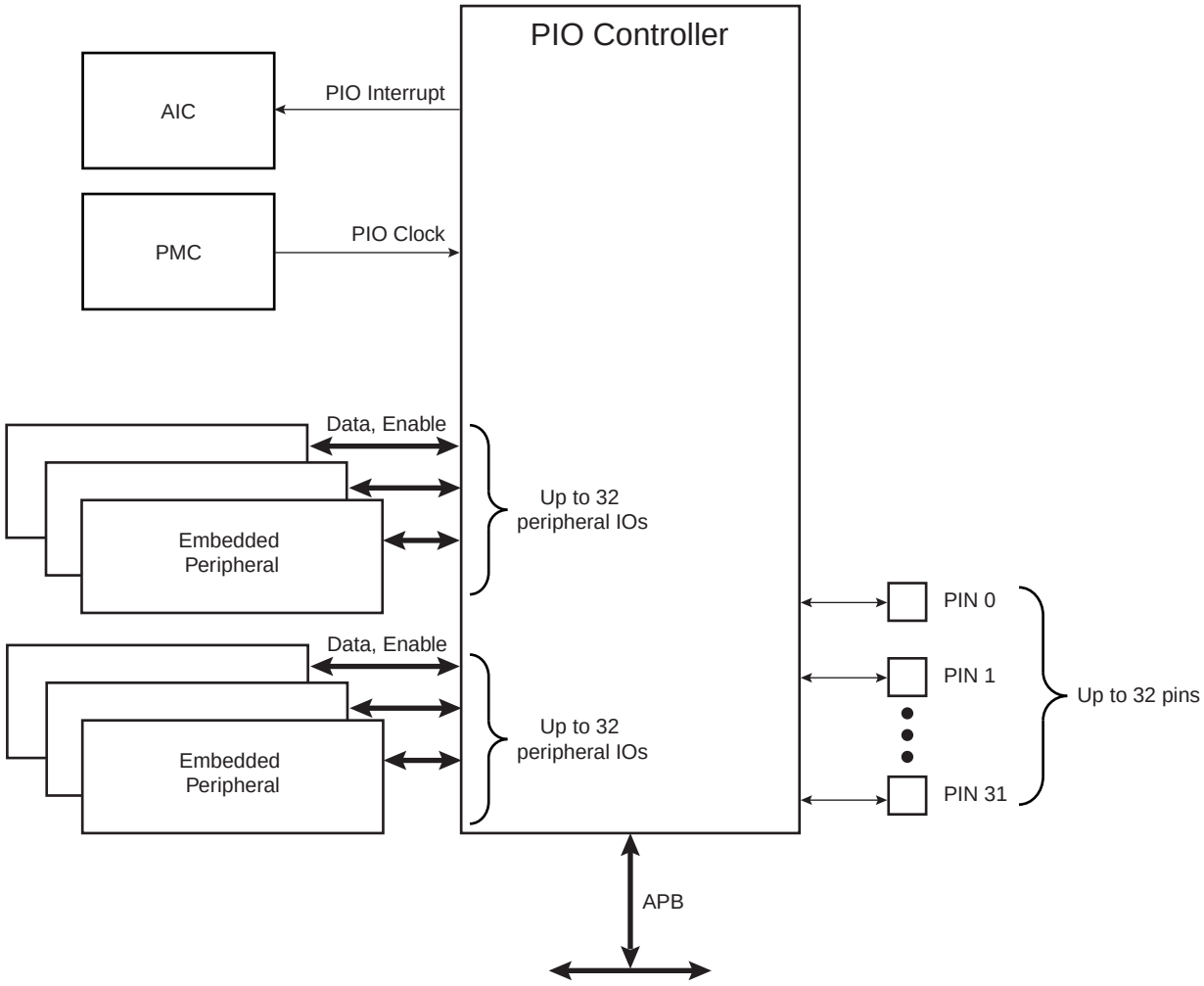
Each I/O line of the PIO Controller features:

- 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 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.

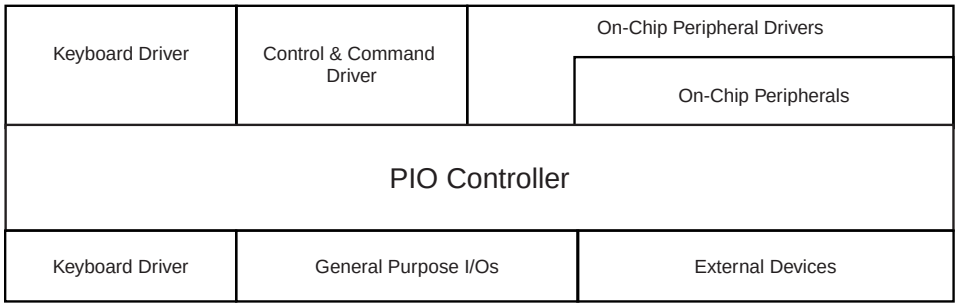
28.2 Block Diagram

Figure 28-1: Block Diagram



© 2017 Microchip Technology Inc.

Figure 28-2: Application Block Diagram



28.3 Product Dependencies

28.3.1 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.

28.3.2 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.

28.3.3 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.

28.3.4 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.

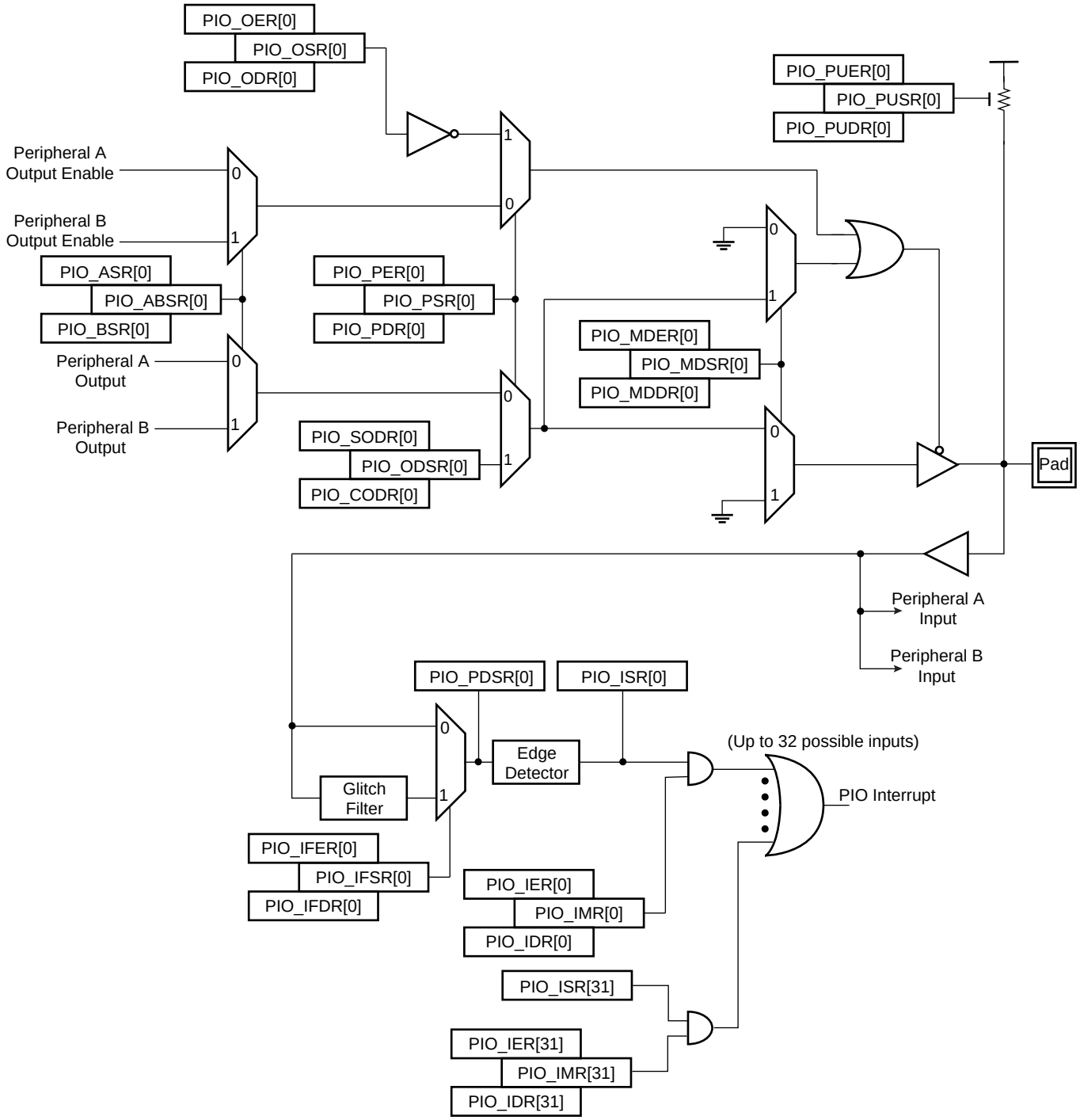
The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

© 2017 Microchip Technology Inc.

28.4 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 28-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 28-3: I/O Line Control Logic



© 2017 Microchip Technology Inc.

28.4.1 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.

28.4.2 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.

28.4.3 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.

28.4.4 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.

28.4.5 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.

28.4.6 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.

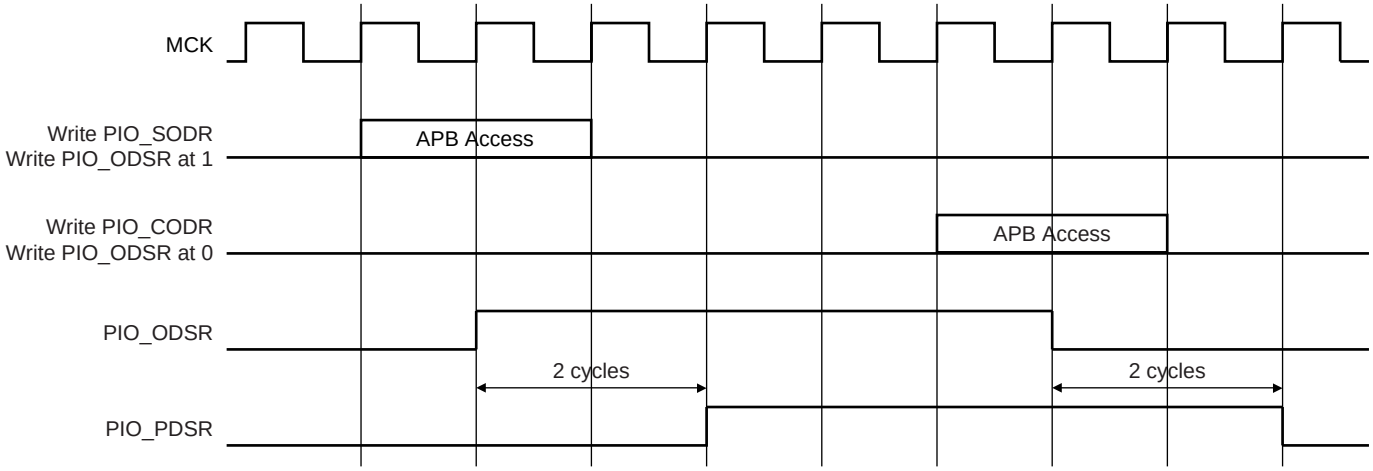
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates 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.

28.4.7 Output Line Timings

Figure 28-4 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 28-4 also shows when the feedback in PIO_PDSR is available.

Figure 28-4: Output Line Timings



28.4.8 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.

28.4.9 Input Glitch Filtering

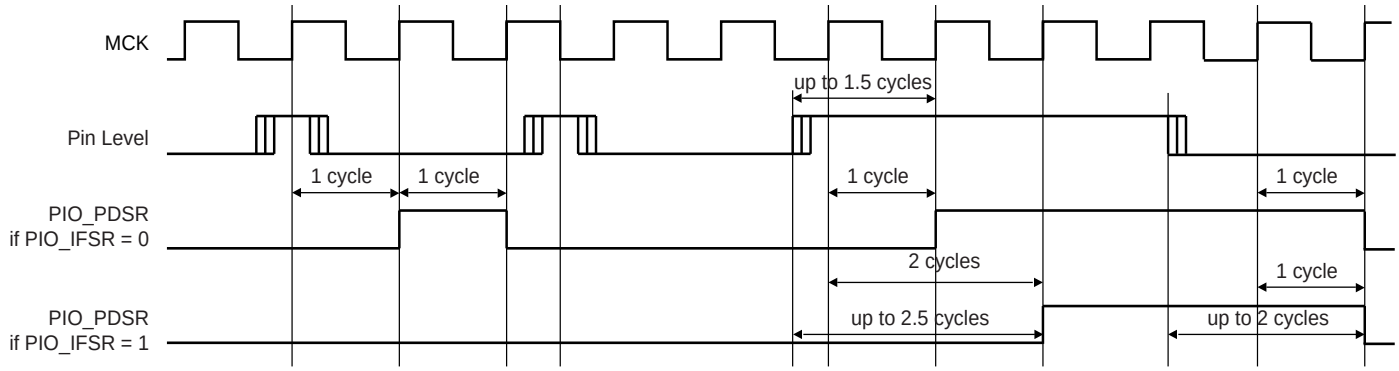
Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations 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 28-5.

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.

© 2017 Microchip Technology Inc.

Figure 28-5: Input Glitch Filter Timing



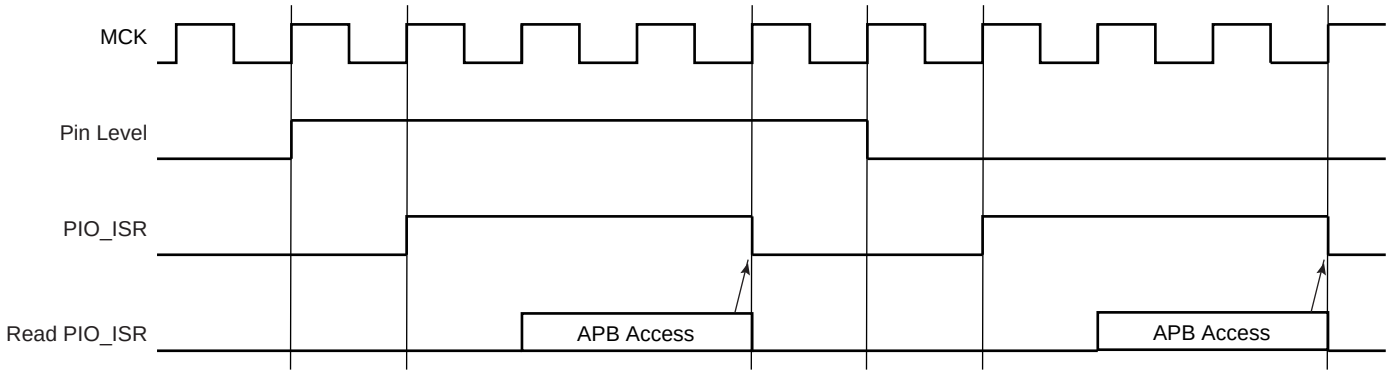
28.4.10 Input Change Interrupt

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 28-6: Input Change Interrupt Timings



28.5 I/O Lines Programming Example

The programing example as shown in Table 28-1 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

© 2017 Microchip Technology Inc.

- 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 28-1: 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

© 2017 Microchip Technology Inc.

28.6 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 28-2: 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			
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

Table 28-2: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
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			

Note 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.

© 2017 Microchip Technology Inc.

28.6.1 PIO Controller PIO Enable Register

Name:PIO_PER

Access: 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).

28.6.2 PIO Controller PIO Disable Register

Name:PIO_PDR

Access: 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).

© 2017 Microchip Technology Inc.

28.6.3 PIO Controller PIO Status Register

Name:PIO_PSR

Access: 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).

28.6.4 PIO Controller Output Enable Register

Name:PIO_OER

Access: 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.

© 2017 Microchip Technology Inc.

28.6.5 PIO Controller Output Disable Register

Name:PIO_ODR

Access: 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.

28.6.6 PIO Controller Output Status Register

Name:PIO_OSR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.7 PIO Controller Input Filter Enable Register

Name:PIO_IFER

Access: 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.

28.6.8 PIO Controller Input Filter Disable Register

Name:PIO_IFDR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.9 PIO Controller Input Filter Status Register

Name:PIO_IFSR

Access: 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.

28.6.10 PIO Controller Set Output Data Register

Name:PIO_SODR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.11 PIO Controller Clear Output Data Register

Name:PIO_CODR

Access: 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.

28.6.12 PIO Controller Output Data Status Register

Name:PIO_ODSR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.13 PIO Controller Pin Data Status Register

Name:PIO_PDSR

Access: 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.

28.6.14 PIO Controller Interrupt Enable Register

Name:PIO_IER

Access: 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.

© 2017 Microchip Technology Inc.

28.6.15 PIO Controller Interrupt Disable Register

Name:PIO_IDR

Access: 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.

28.6.16 PIO Controller Interrupt Mask Register

Name:PIO_IMR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.17 PIO Controller Interrupt Status Register

Name:PIO_ISR

Access: 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.

28.6.18 PIO Multi-driver Enable Register

Name:PIO_MDER

Access: 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.

© 2017 Microchip Technology Inc.

28.6.19 PIO Multi-driver Disable Register

Name:PIO_MDDR

Access: 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.

28.6.20 PIO Multi-driver Status Register

Name:PIO_MDSR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.21 PIO Pull Up Disable Register

Name:PIO_PUDR

Access: 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.

28.6.22 PIO Pull Up Enable Register

Name:PIO_PUER

Access: 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.

© 2017 Microchip Technology Inc.

28.6.23 PIO Pull Up Status Register

Name:PIO_PUSR

Access: 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.

28.6.24 PIO Peripheral A Select Register

Name:PIO_ASR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.25 PIO Peripheral B Select Register

Name:PIO_BSR

Access: 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.

28.6.26 PIO Peripheral A B Status Register

Name:PIO_ABSR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.27 PIO Output Write Enable Register

Name:PIO_OWER

Access: 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.

28.6.28 PIO Output Write Disable Register

Name:PIO_OWDR

Access: 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.

© 2017 Microchip Technology Inc.

28.6.29 PIO Output Write Status Register

Name:PIO_OWSR

Access: 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.

29. Serial Peripheral Interface (SPI)

29.1 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.

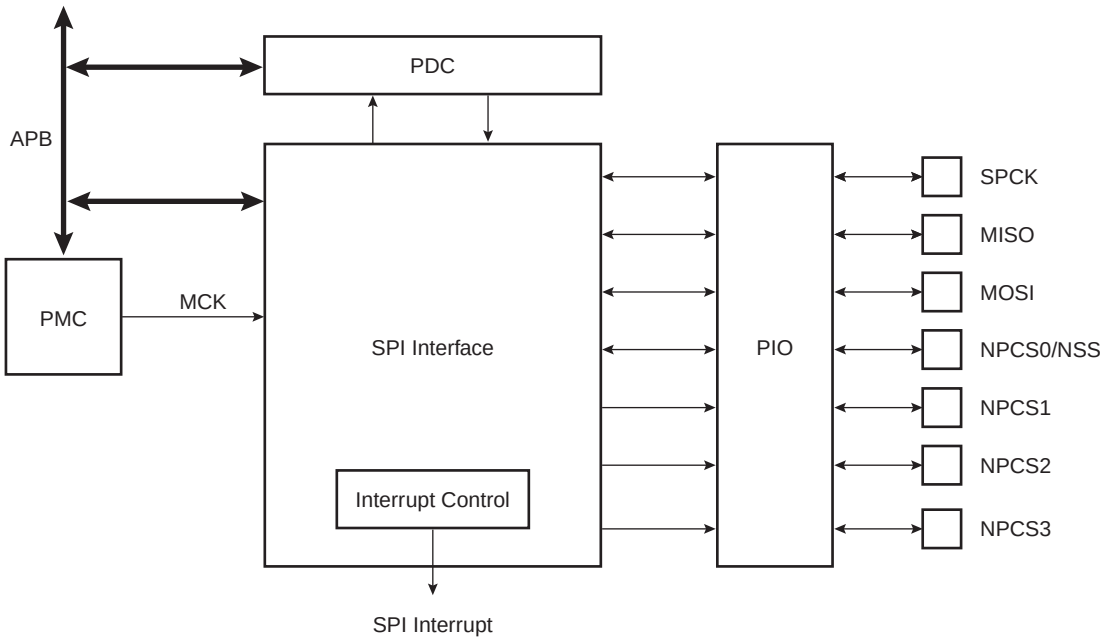
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

29.2 Block Diagram

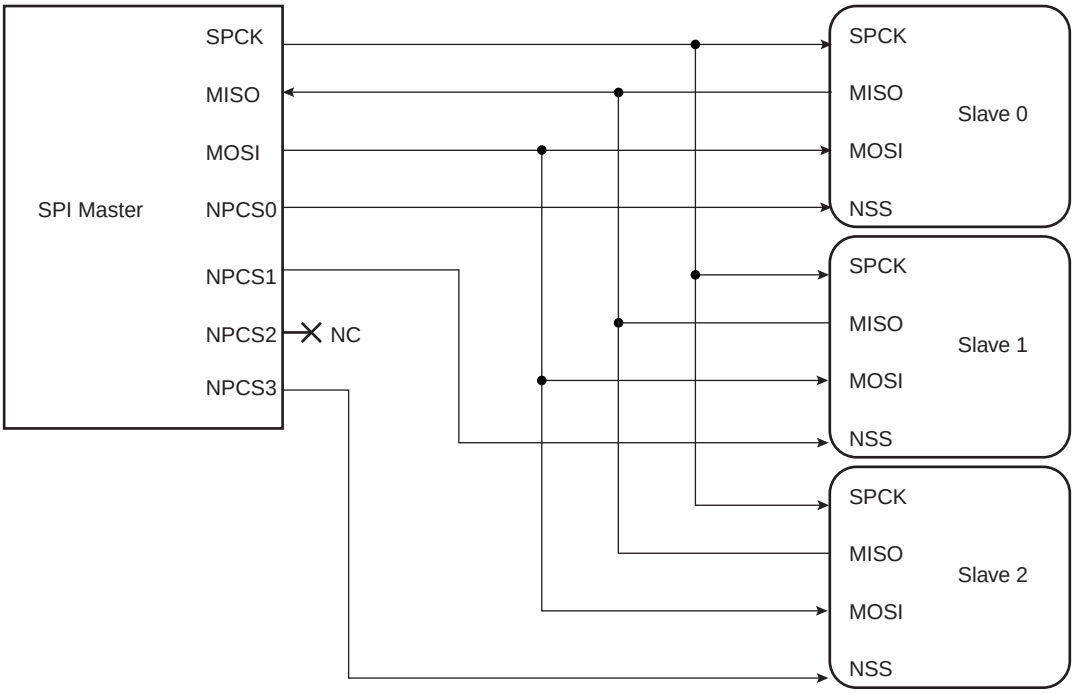
Figure 29-1: Block Diagram



© 2017 Microchip Technology Inc.

29.3 Application Block Diagram

Figure 29-2: Application Block Diagram: Single Master/Multiple Slave Implementation



29.4 Signal Description

Table 29-1: 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

29.5 Product Dependencies

29.5.1 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.

29.5.2 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.

29.5.3 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.

29.6 Functional Description

29.6.1 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.

29.6.2 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.

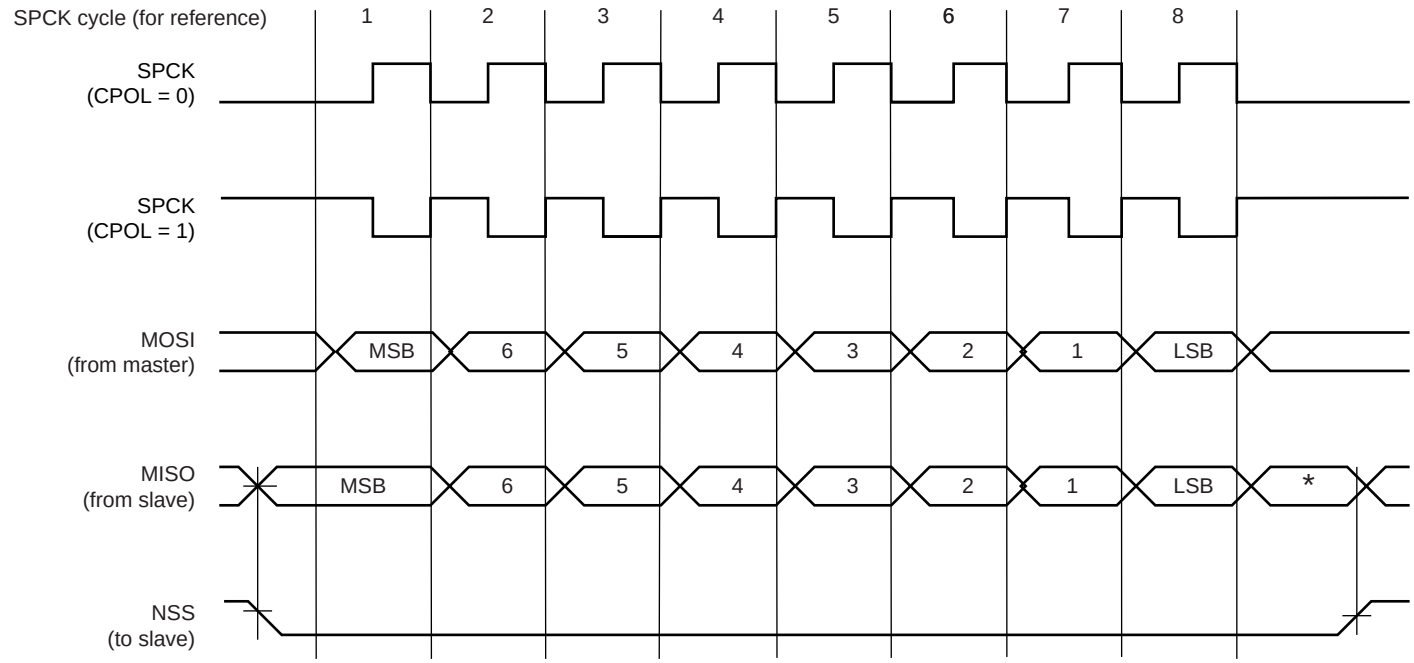
Table 29-2 shows the four modes and corresponding parameter settings.

Table 29-2: SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 29-3 and Figure 29-4 show examples of data transfers.

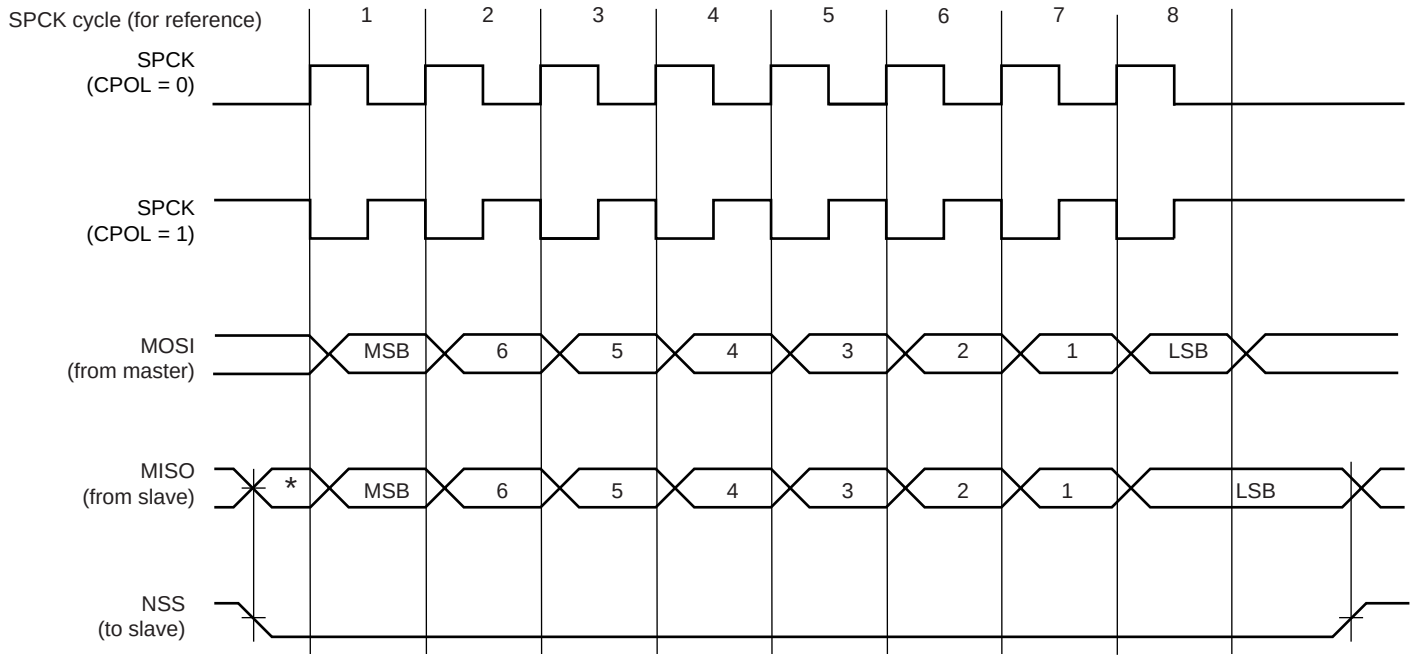
Figure 29-3: SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

© 2017 Microchip Technology Inc.

Figure 29-4: SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

29.6.3 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 writing 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. 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.

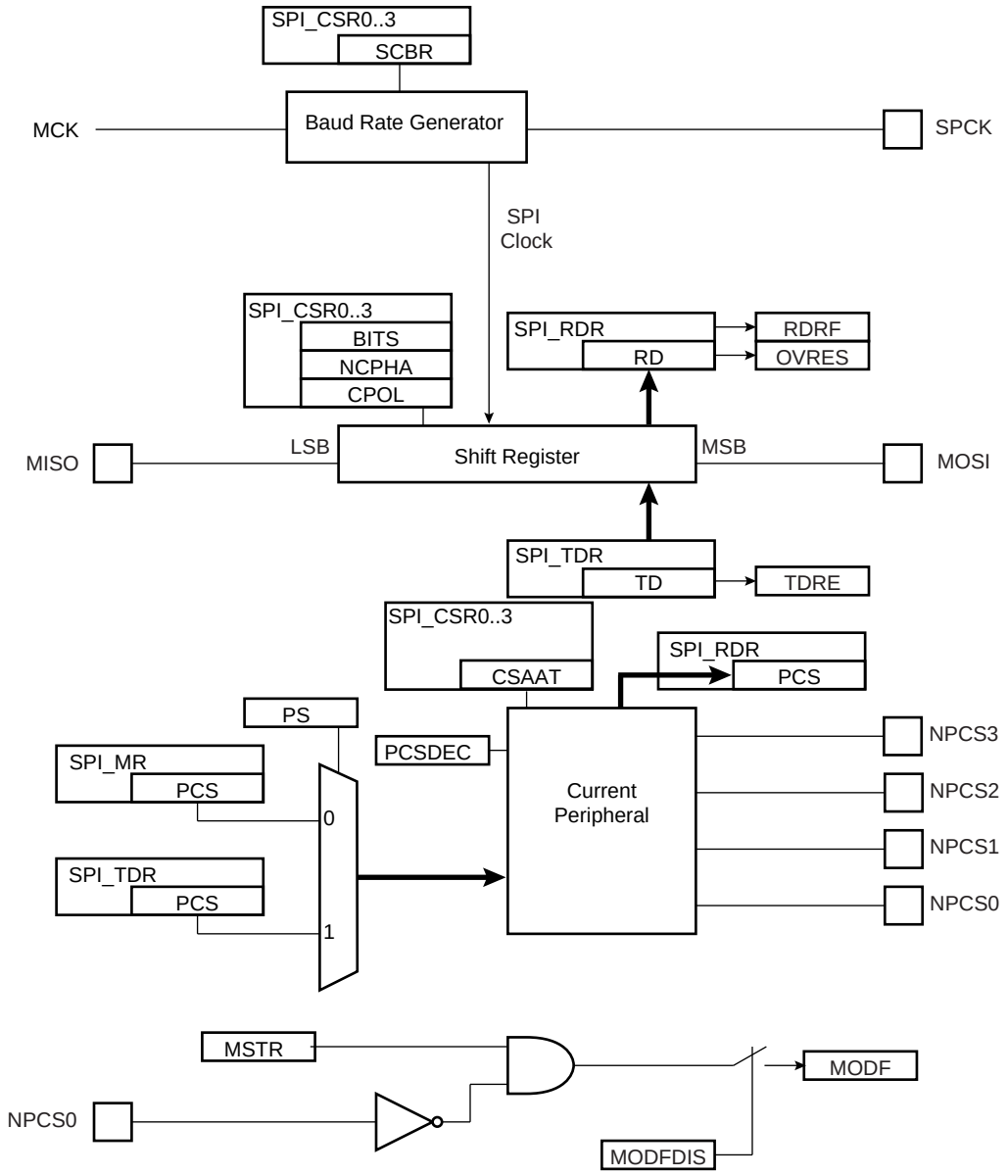
If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 29-5 shows a block diagram of the SPI when operating in Master Mode. Figure 29-6 shows a flow chart describing how transfers are handled.

© 2017 Microchip Technology Inc.

29.6.3.1 Master Mode Block Diagram

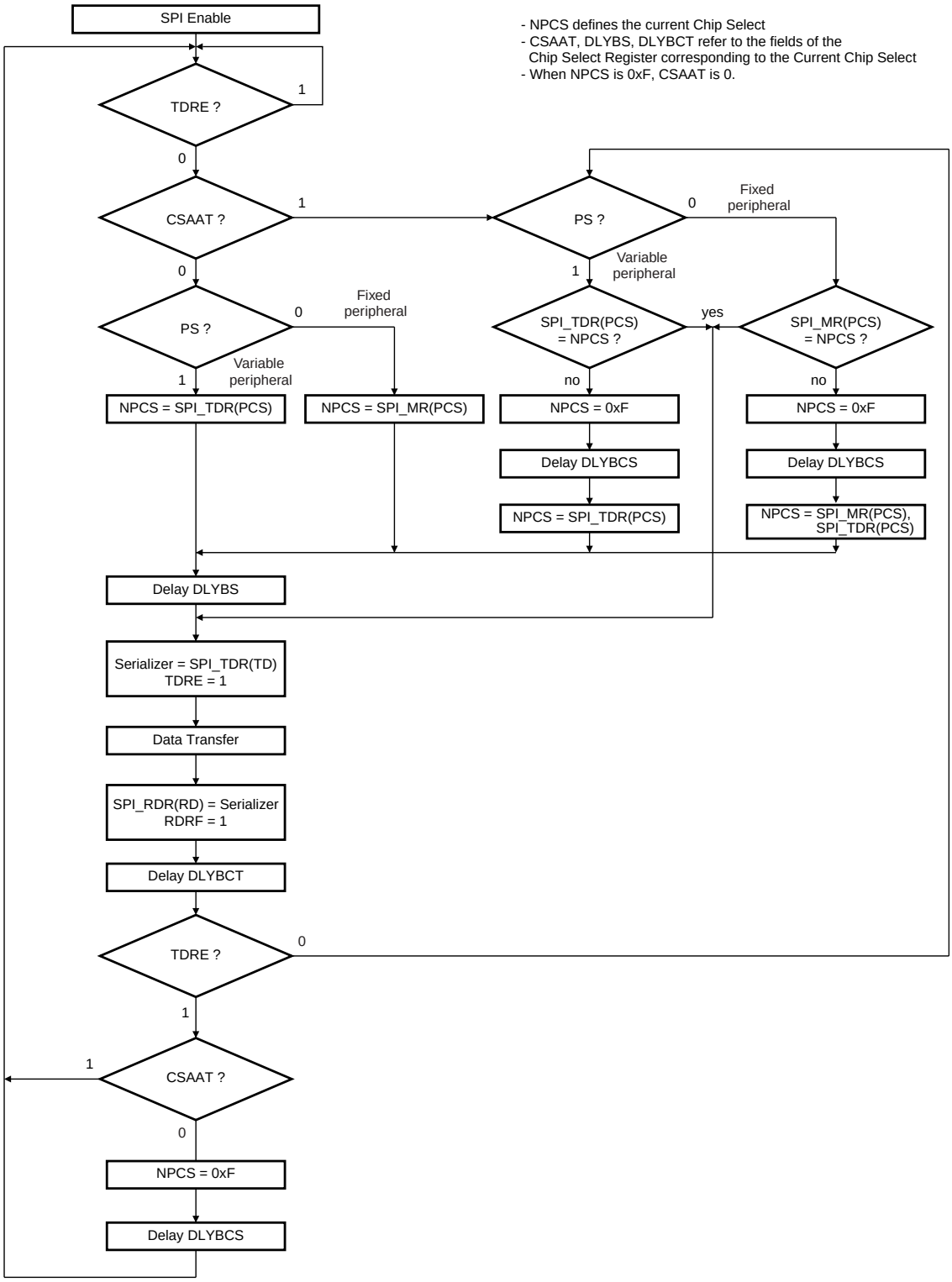
Figure 29-5: Master Mode Block Diagram



© 2017 Microchip Technology Inc.

29.6.3.2 Master Mode Flow Diagram

Figure 29-6: Master Mode Flow Diagram



29.6.3.3 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.

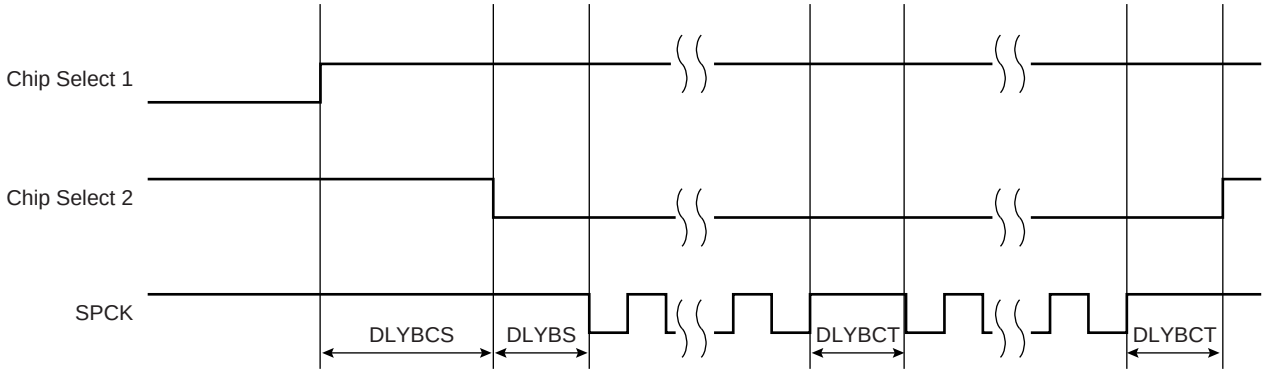
29.6.3.4 Transfer Delays

Figure 29-7 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 29-7: Programmable Delays



29.6.3.5 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.

© 2017 Microchip Technology Inc.

29.6.3.6 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.

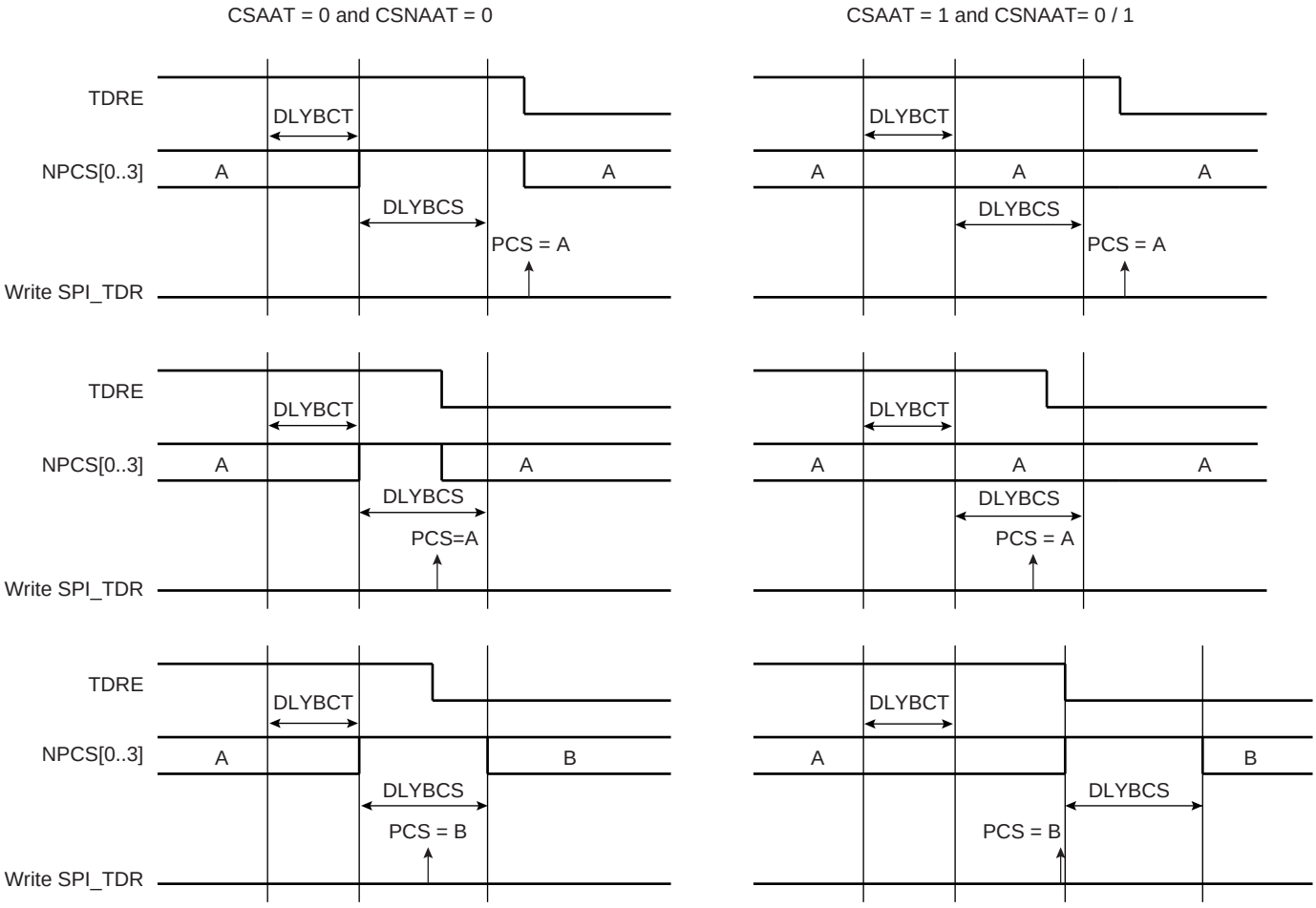
The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 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.

29.6.3.7 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 29-8: Peripheral Deselection



29.6.3.8 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).

29.6.4 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.

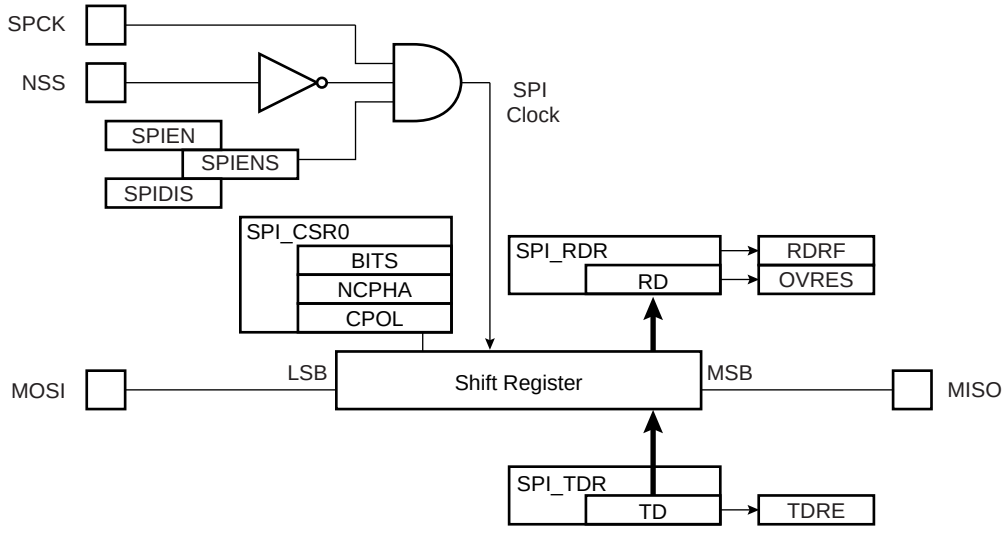
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets 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 29-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 29-9: Slave Mode Functional Block Diagram



© 2017 Microchip Technology Inc.

29.7 Serial Peripheral Interface (SPI) User Interface

Table 29-3: 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	–	–	–

© 2017 Microchip Technology Inc.

29.7.1 SPI Control Register

Name: SPI_CR

Access: 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 transfer.

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.

© 2017 Microchip Technology Inc.

29.7.2 SPI Mode Register

Name: SPI_MR

Access: 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.)

© 2017 Microchip Technology Inc.

PCS: Peripheral Chip Select

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (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:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

© 2017 Microchip Technology Inc.

29.7.3 SPI Receive Data Register

Name: SPI_RDR

Access: 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.

© 2017 Microchip Technology Inc.

29.7.4 SPI Transmit Data Register

Name: SPI_TDR

Access: 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 = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (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).

© 2017 Microchip Technology Inc.

29.7.5 SPI Status Register

Name: SPI_SR

Access: 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.

© 2017 Microchip Technology Inc.

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.

© 2017 Microchip Technology Inc.

29.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Access: 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

TXEMPTY: Transmission Registers Empty Enable

0: No effect.

1: Enables the corresponding interrupt.

29.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Access: 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

TXEMPTY: Transmission Registers Empty Disable

0: No effect.

1: Disables the corresponding interrupt.

© 2017 Microchip Technology Inc.

29.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Access: 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

TXEMPTY: Transmission Registers Empty Mask

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

© 2017 Microchip Technology Inc.

29.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Access: 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
1100	Reserved

© 2017 Microchip Technology Inc.

BITS	Bits Per Transfer
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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

30. Two-wire Interface (TWI)

30.1 Overview

The Microchip 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 Microchip 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.

The table below lists the compatibility level of the Microchip Two-wire Interface in Master Mode and a full I2C compatible device.

Table 30-1: Microchip TWI compatibility with i2C Standard

I2C Standard	Microchip 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 + SrQ

30.2 List of Abbreviations

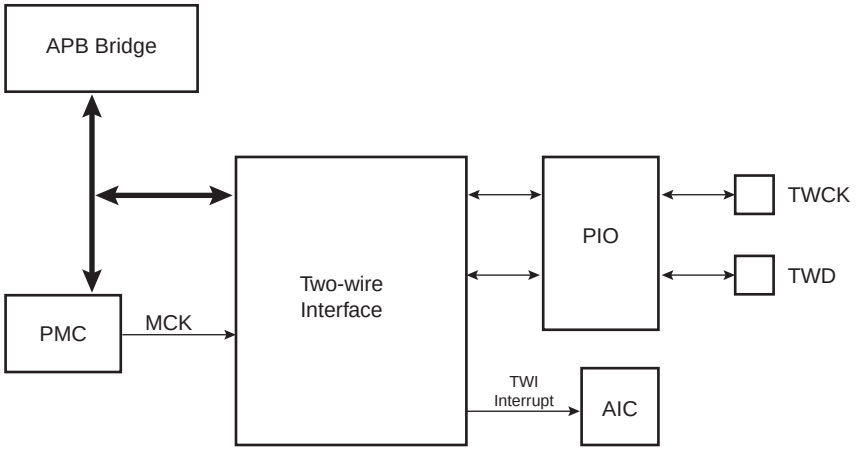
Table 30-2: Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

© 2017 Microchip Technology Inc.

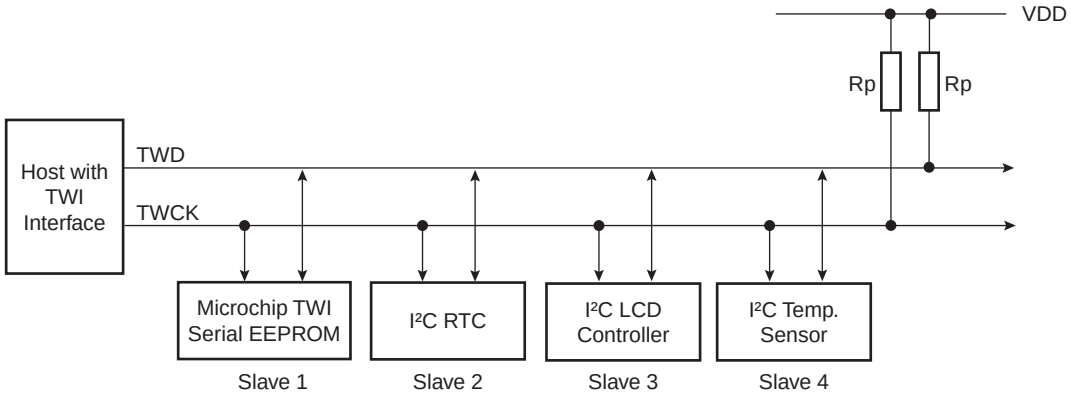
30.3 Block Diagram

Figure 30-1: Block Diagram



30.4 Application Block Diagram

Figure 30-2: Application Block Diagram



Rp: Pull up value as given by the I²C Standard

30.4.1 I/O Lines Description

Table 30-3: I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

30.5 Product Dependencies

30.5.1 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 30-2). 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 steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

30.5.2 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.

30.5.3 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.

30.6 Functional Description

30.6.1 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 30-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 30-3).

- 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 30-3: START and STOP Conditions

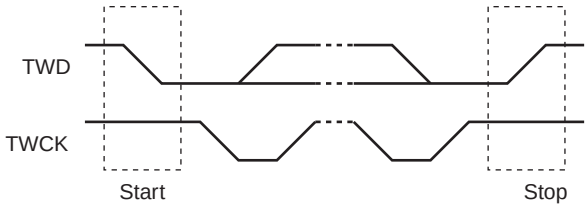
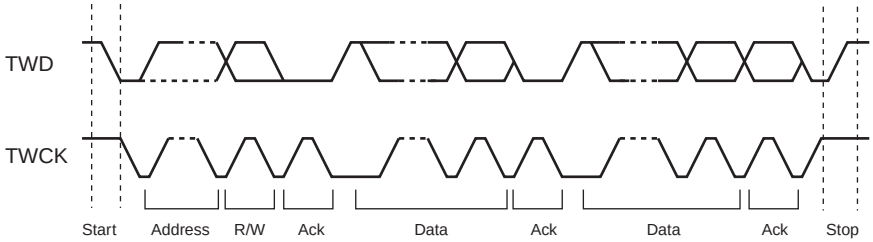


Figure 30-4: Transfer Format



30.6.2 Modes of Operation

The TWI has six modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode

© 2017 Microchip Technology Inc.

- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following sections.

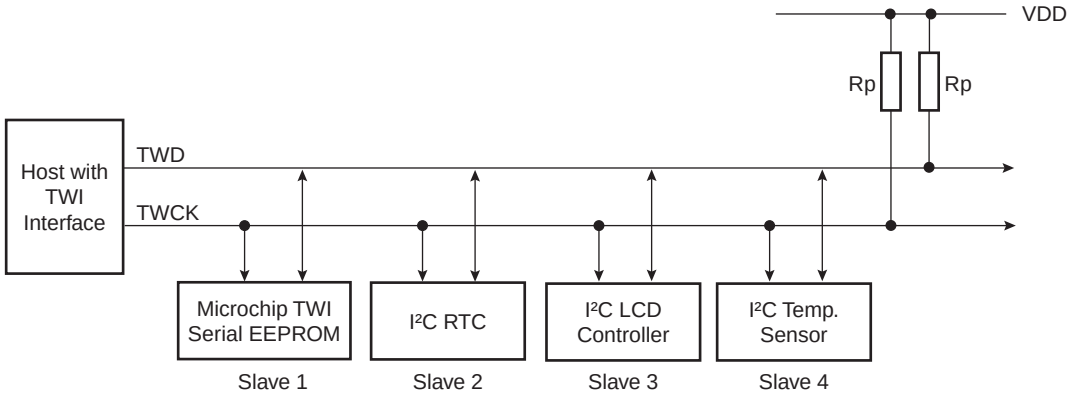
30.6.3 Master Mode

30.6.3.1 Definition

The Master is the device that starts a transfer, generates a clock and stops it.

30.6.3.2 Application Block Diagram

Figure 30-5: Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

30.6.3.3 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.

30.6.3.4 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 30-6, Figure 30-7, and Figure 30-8.

TXRDY is used as Transmit Ready for the PDC transmit channel.

Figure 30-6: Master Write with On Data Byte

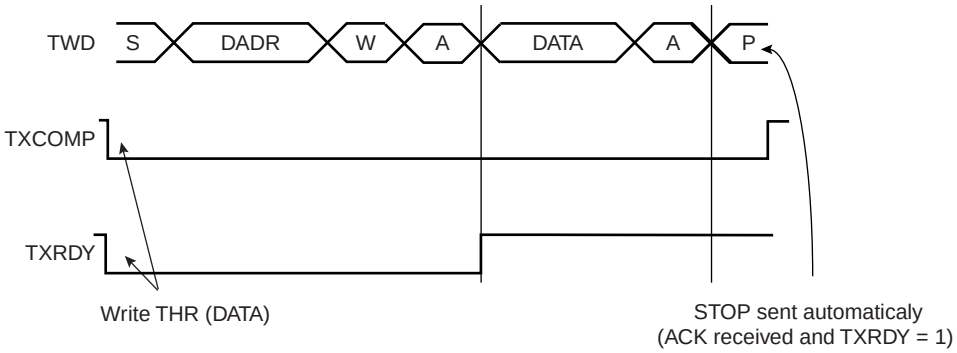


Figure 30-7: Master Write with Multiple Data Byte

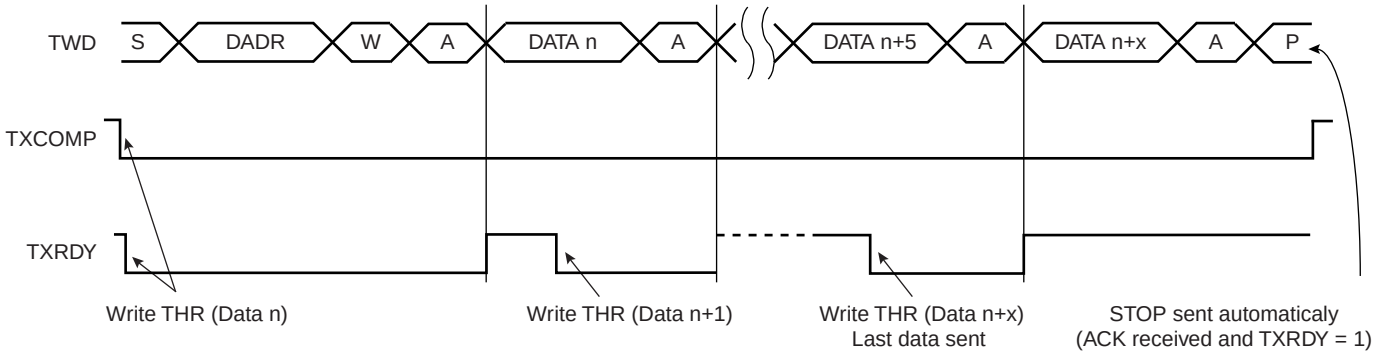
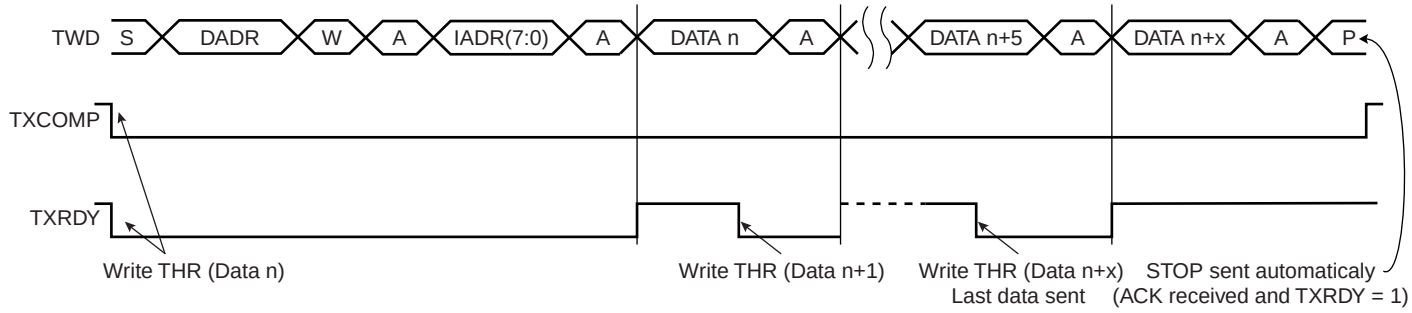


Figure 30-8: Master Write with One Byte Internal Address and Multiple Data Bytes



30.6.3.5 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 30-9. 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 30-9. 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 30-10. For Internal Address usage see Section 30.6.3.6 Internal Address.

© 2017 Microchip Technology Inc.

Figure 30-9: Master Read with One Data Byte

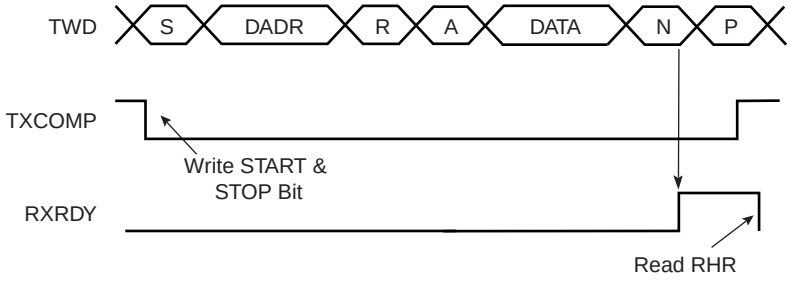
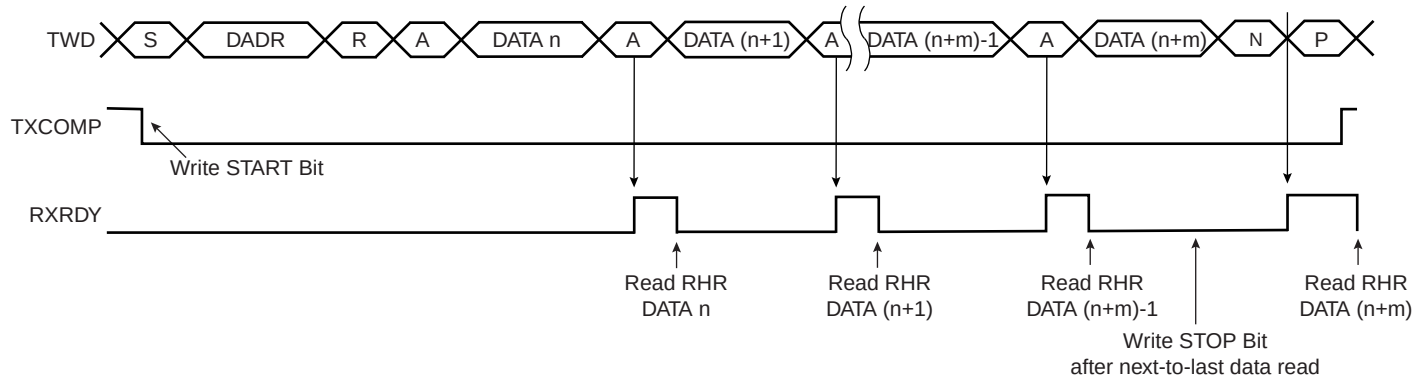


Figure 30-10: Master Read with Multiple Data Bytes



RXRDY is used as Receive Ready for the PDC receive channel.

30.6.3.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

- 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 30-12. See Figure 30-11 and Figure 30-13 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 30-11: Master Write with One, Two or Three Bytes Internal Address and One Data Byte

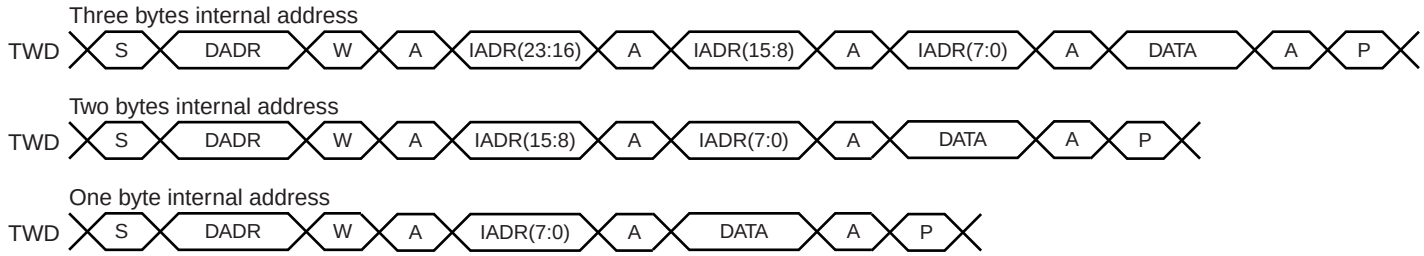
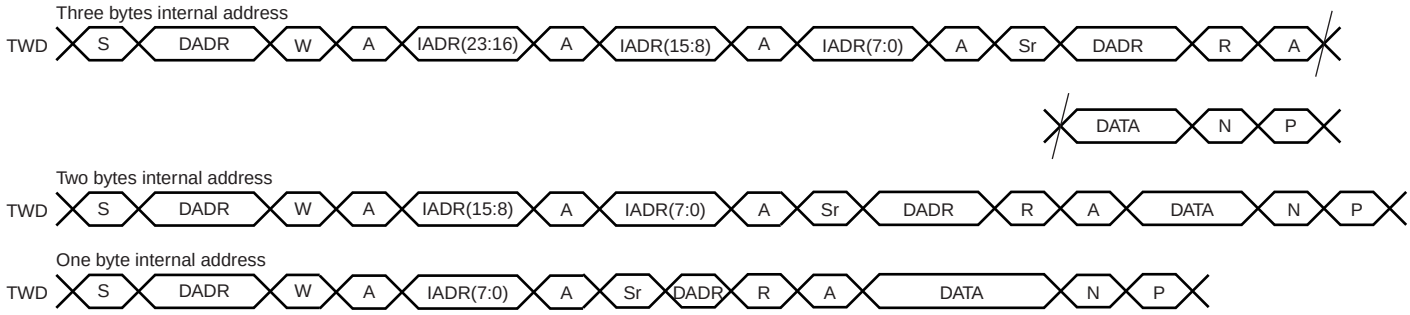


Figure 30-12: Master Read with One, Two or Three Bytes Internal Address and One Data Byte



- 10-bit Slave Addressing

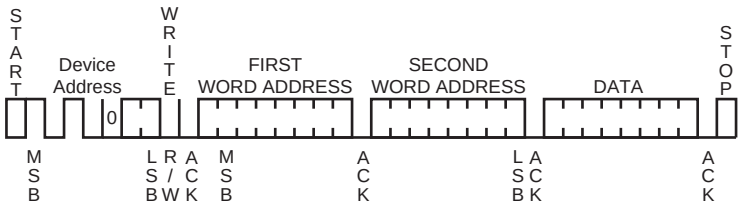
For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ**) and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

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 30-13 below shows a byte write to a Microchip AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 30-13: Internal Address Usage



30.6.3.7 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:

- 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.

© 2017 Microchip Technology Inc.

- 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.

30.6.3.8 Read/Write Flowcharts

The flowcharts in the following figures provide examples of 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 30-14: TWI Write Operation with Single Data Byte without Internal Address

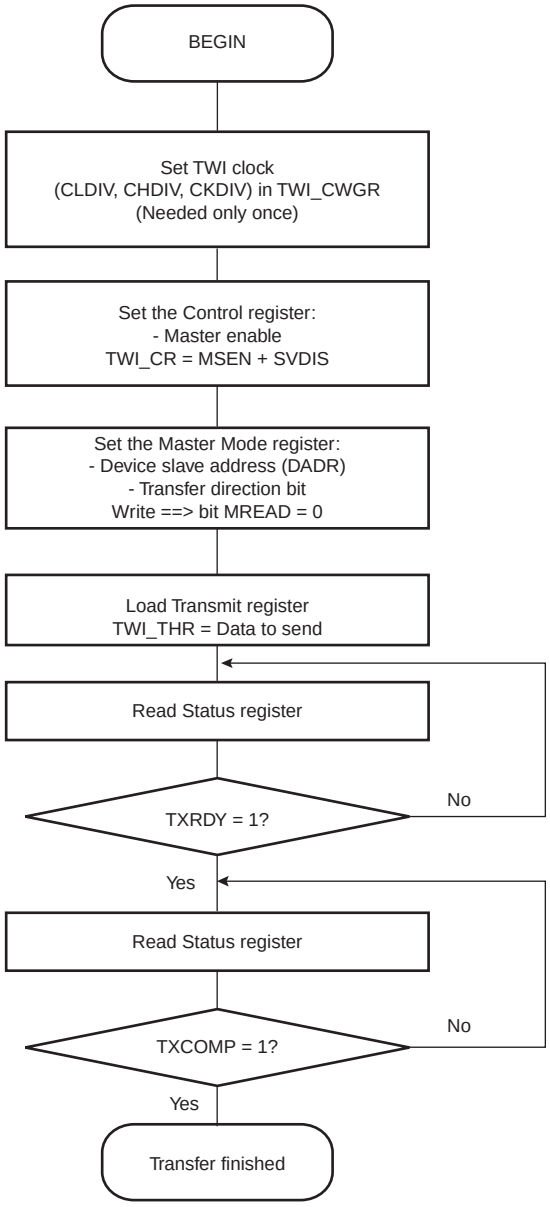
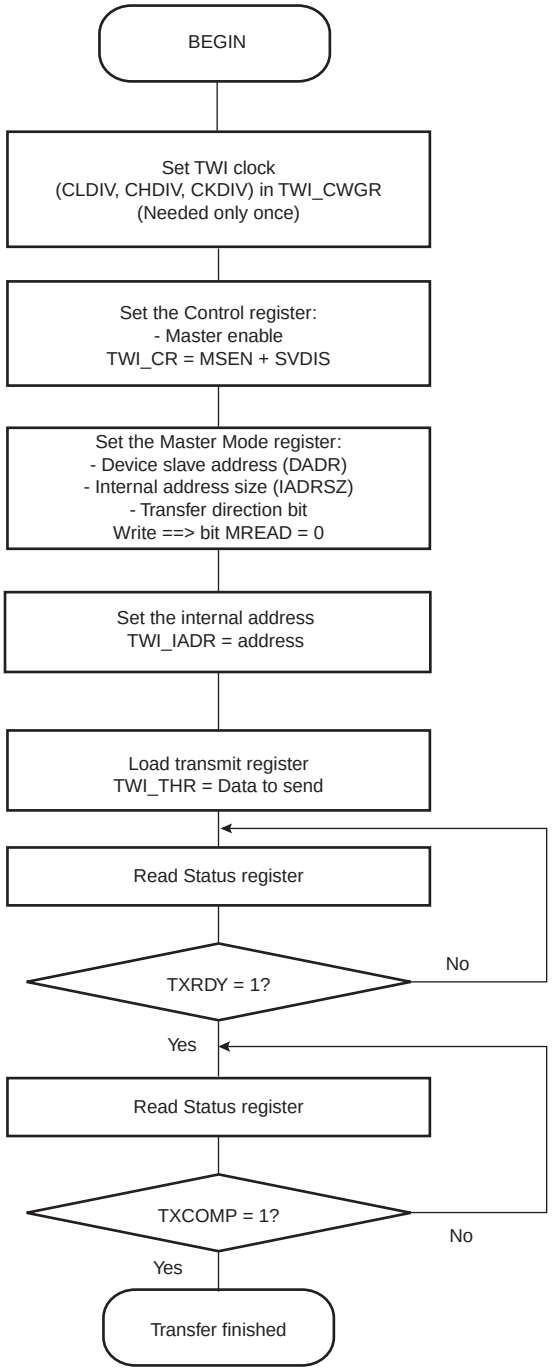


Figure 30-15: TWI Write Operation with Single Data Byte and Internal Address



© 2017 Microchip Technology Inc.

Figure 30-16: TWI Write Operation with Multiple Data Bytes with or without Internal Address

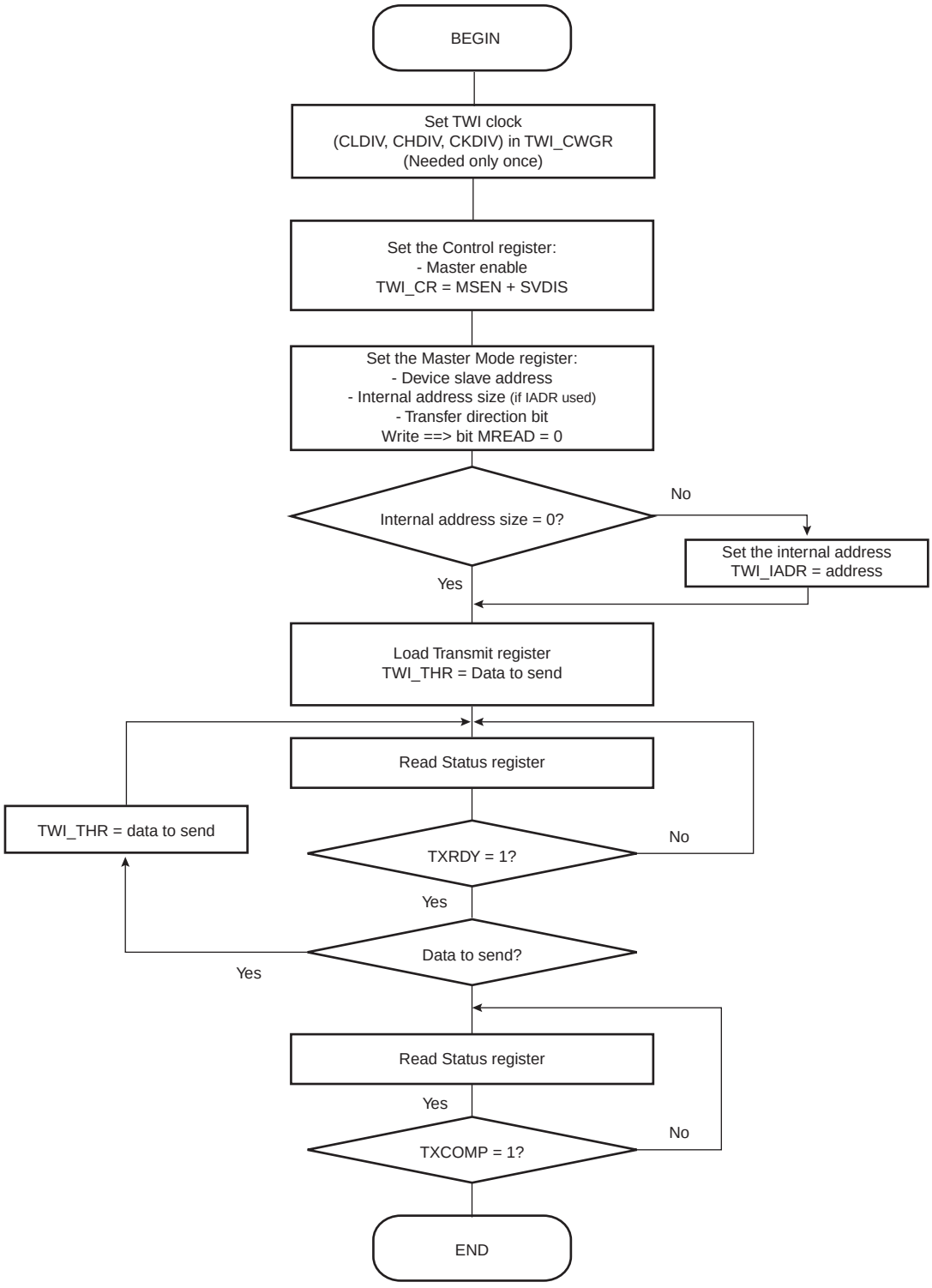
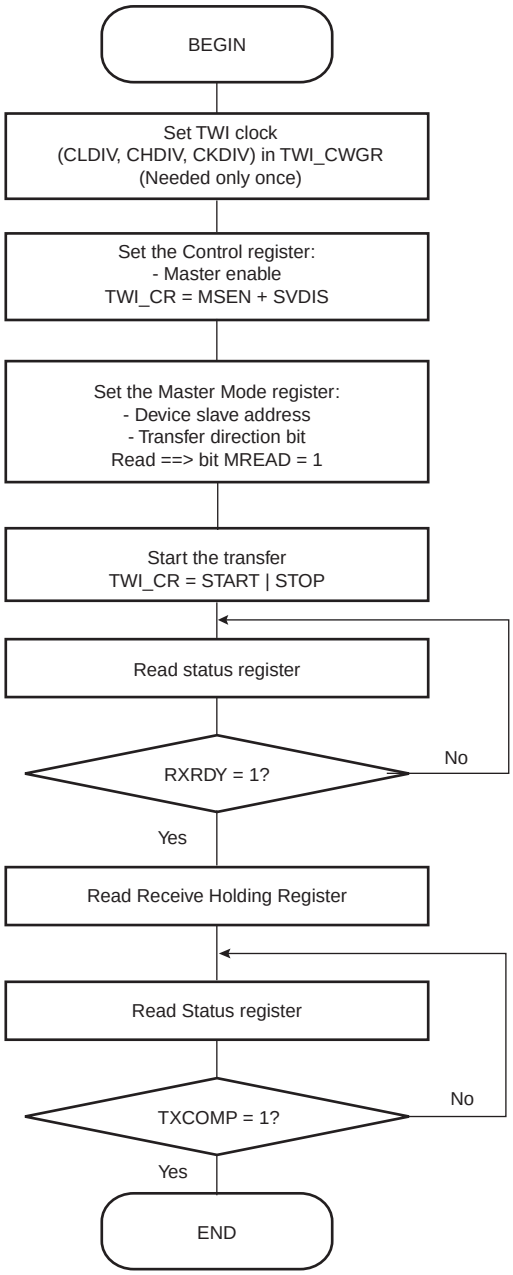


Figure 30-17: TWI Read Operation with Single Data Byte without Internal Address



© 2017 Microchip Technology Inc.

Figure 30-18: TWI Read Operation with Single Data Byte and Internal Address

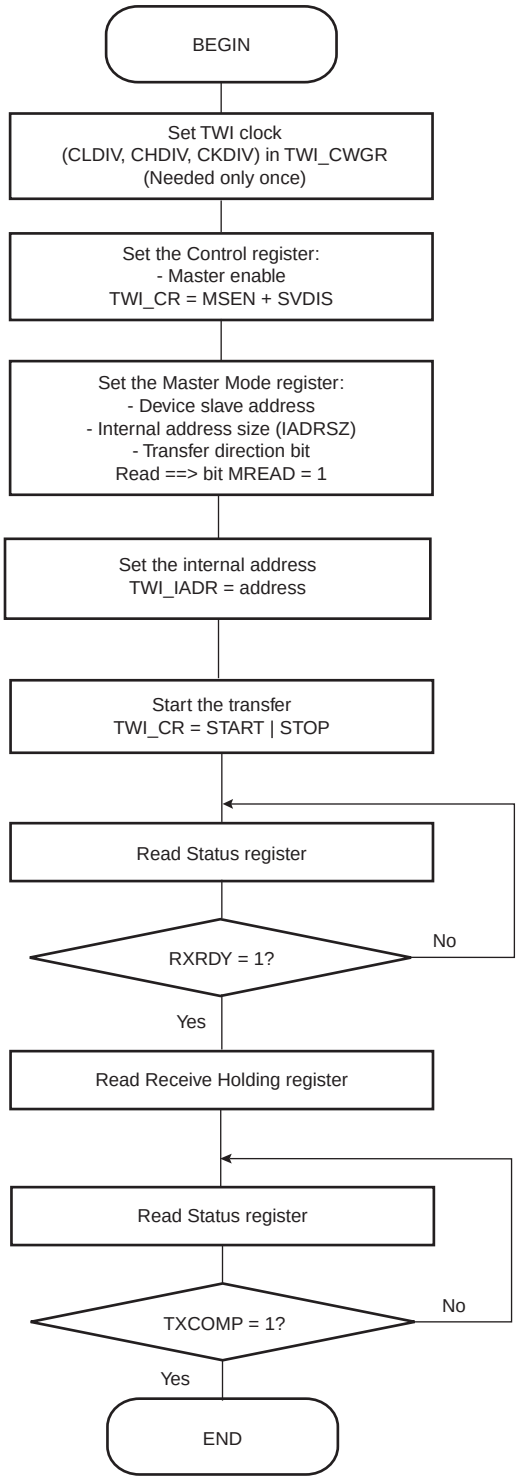
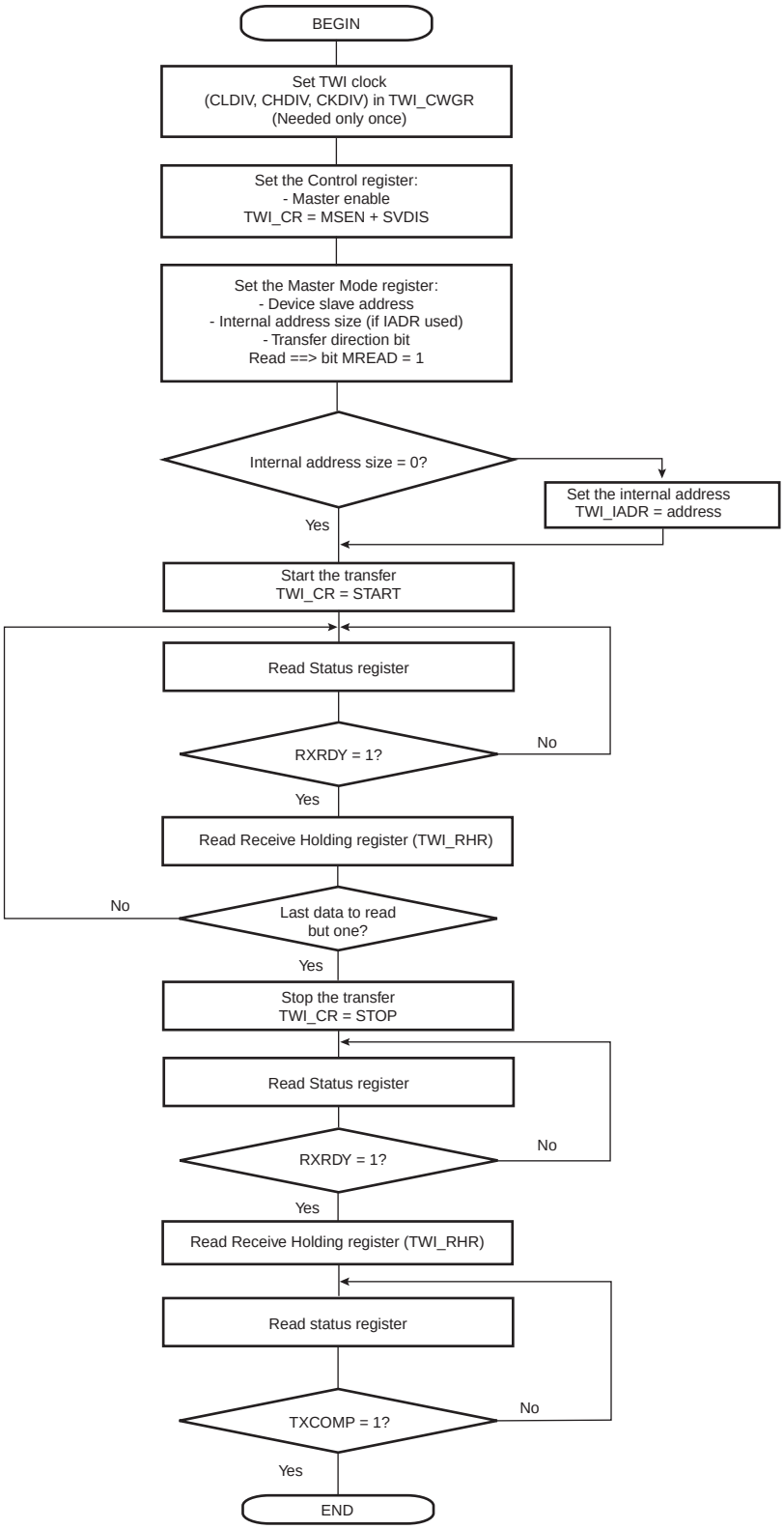


Figure 30-19: TWI Read Operation with Multiple Data Bytes with or without Internal Address



© 2017 Microchip Technology Inc.

30.6.4 Multi-master Mode

30.6.4.1 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 30-21.

30.6.4.2 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.

- 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 30-20).

Note: The state of the bus (busy or free) is not indicated in the user interface.

- 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.

© 2017 Microchip Technology Inc.

Figure 30-20: Programmer Sends Data While the Bus is Busy

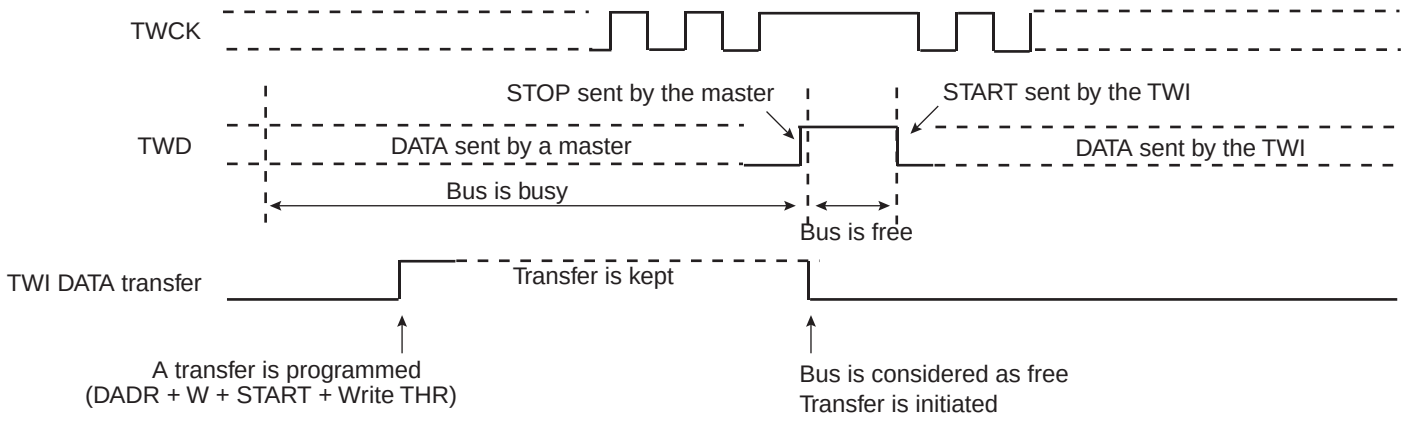
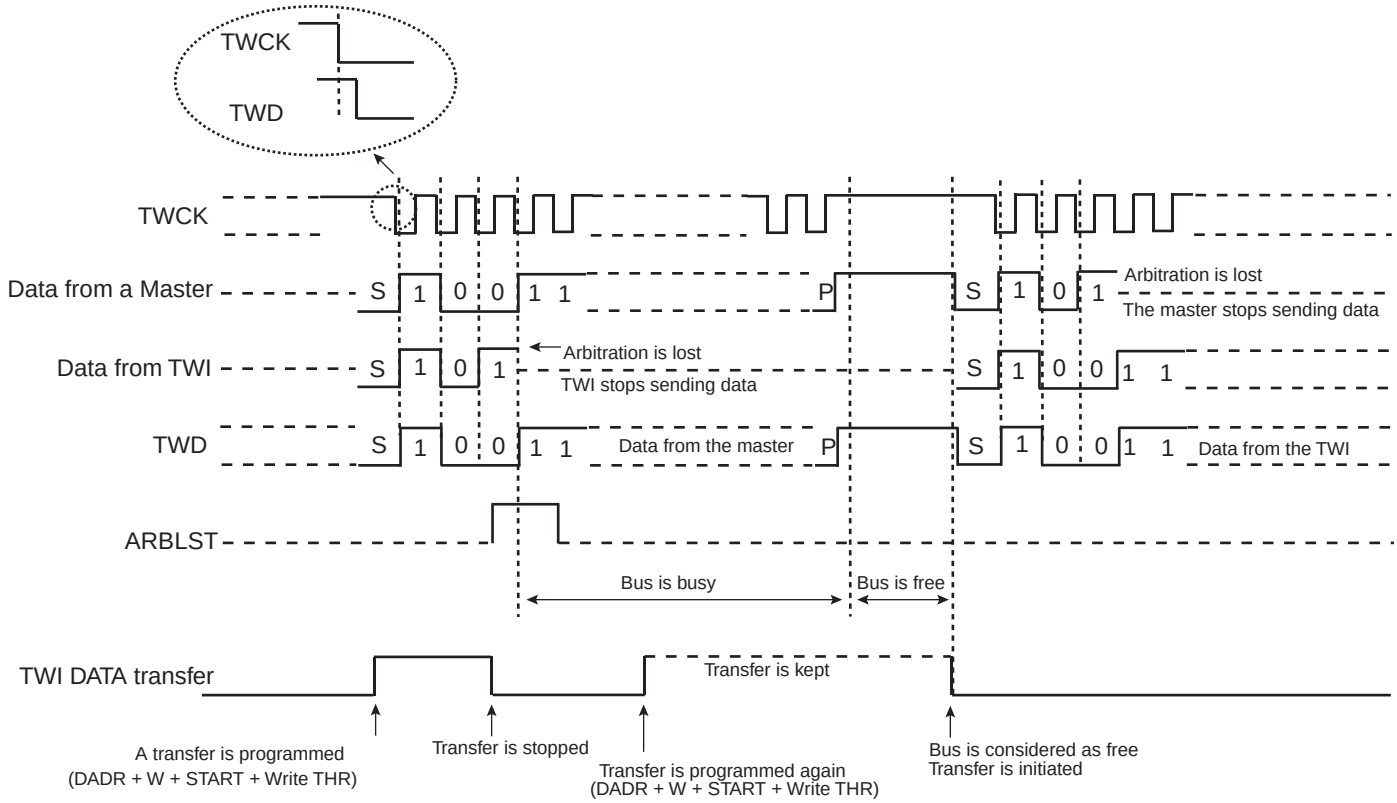


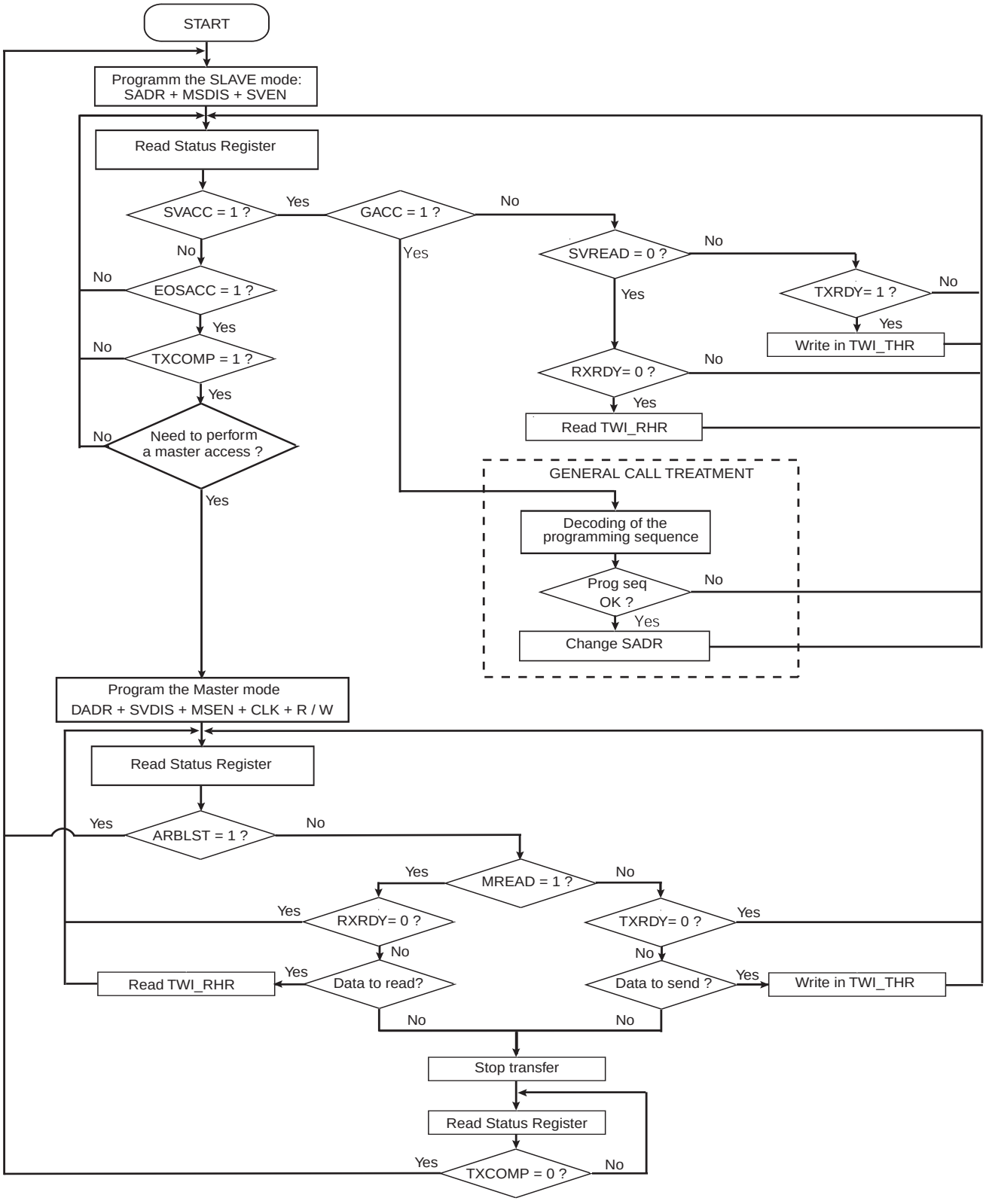
Figure 30-21: Arbitration Cases



The flowchart shown in Figure 30-22 gives an example of read and write operations in Multi-master mode.

© 2017 Microchip Technology Inc.

Figure 30-22: Multi-master Flowchart



30.6.5 Slave Mode

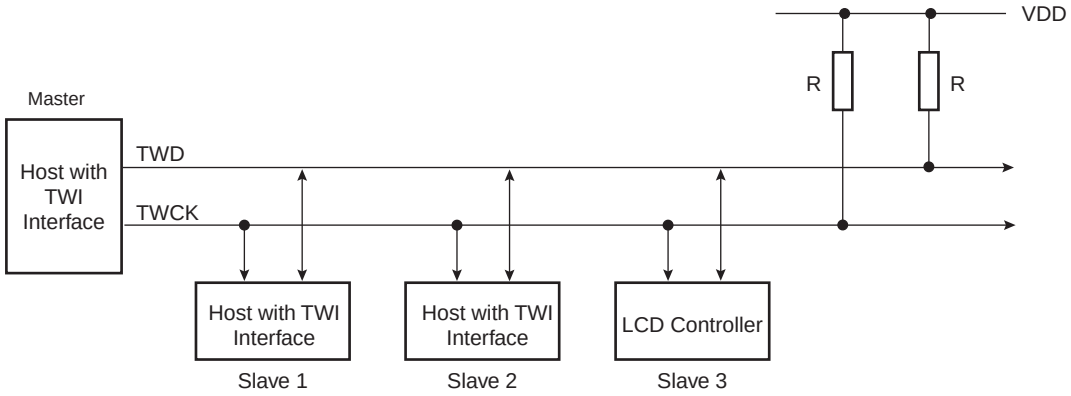
30.6.5.1 Definition

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).

30.6.5.2 Application Block Diagram

Figure 30-23: Slave Mode Typical Application Block Diagram



30.6.5.3 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.

30.6.5.4 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.

- 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 30-24.

- 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 30-25.

- Clock Synchronization Sequence

In the case where TWI_THR or TWI_RHR is not written/read in time, TWI performs a clock synchronization.

© 2017 Microchip Technology Inc.

Clock stretching information is given by the SCLWS (Clock Wait state) bit.

See Figure 30-27 and Figure 30-28.

- 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 30-26.

30.6.5.5 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.

30.6.5.6 Data Transfer

- Read Operation

The read mode is defined as a data requirement from the master.

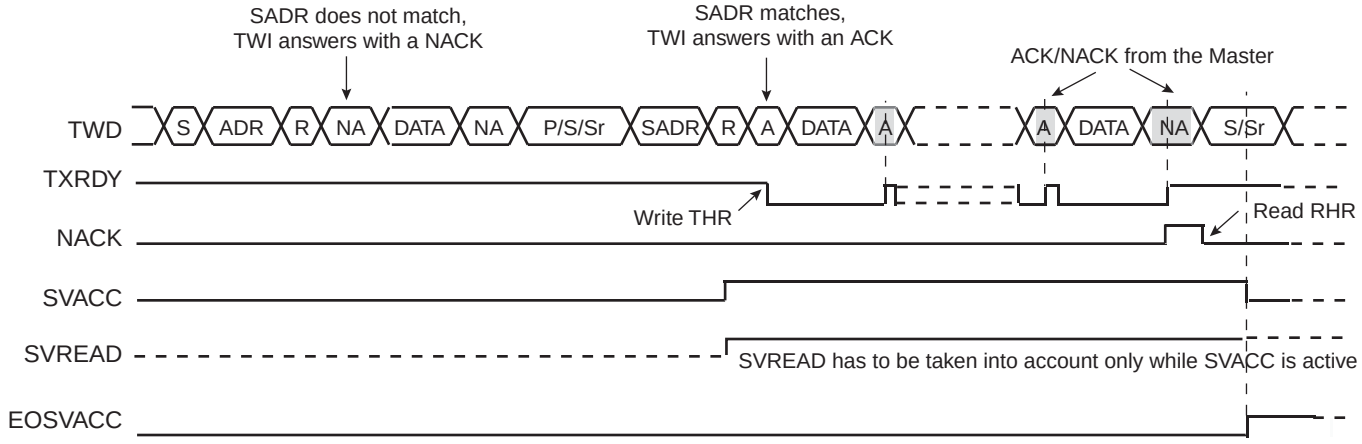
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 30-24 describes the write operation.

Figure 30-24: Read Access Ordered by a MASTER



Note 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.

- Write Operation

The write mode is defined as a data transmission from the master.

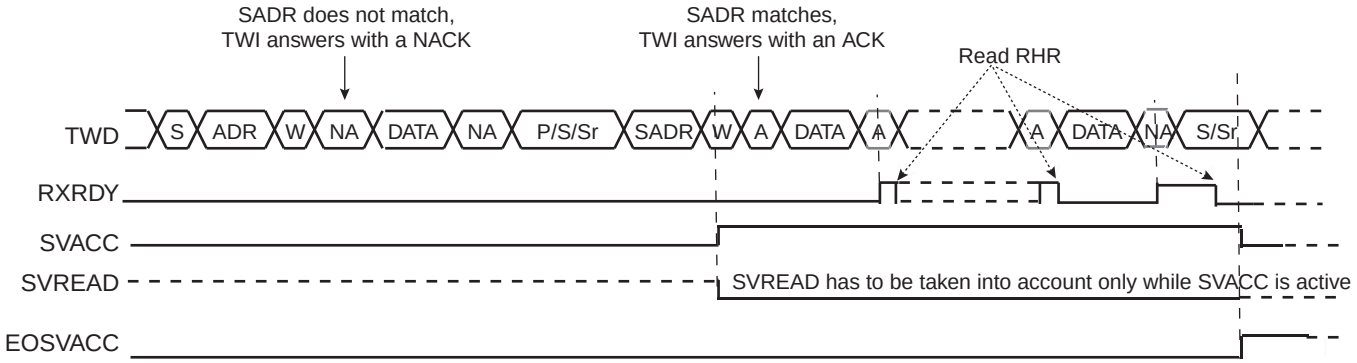
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 30-25 describes the Write operation.

Figure 30-25: Write Access Ordered by a Master



Note 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.

- General Call

The general call is performed in order to change the address of the slave.

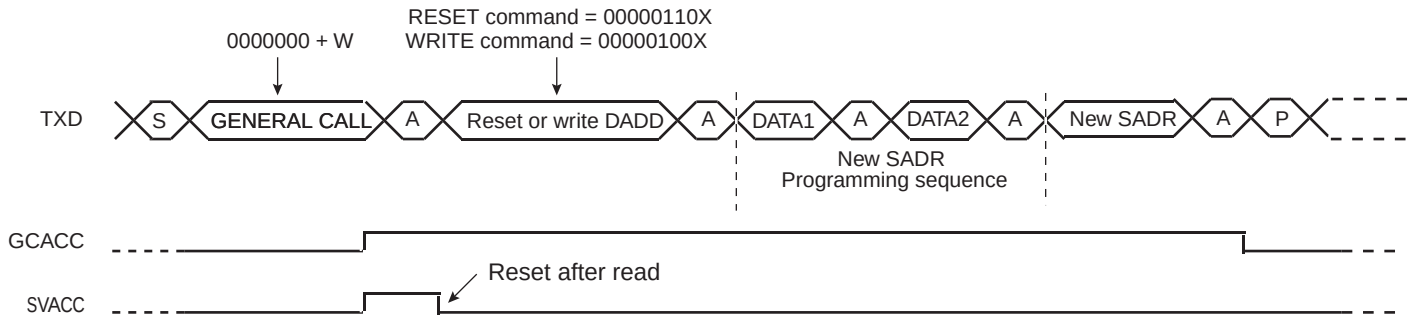
If a GENERAL CALL is detected, GACC is set.

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 30-26 describes the General Call access.

Figure 30-26: 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.

- 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.

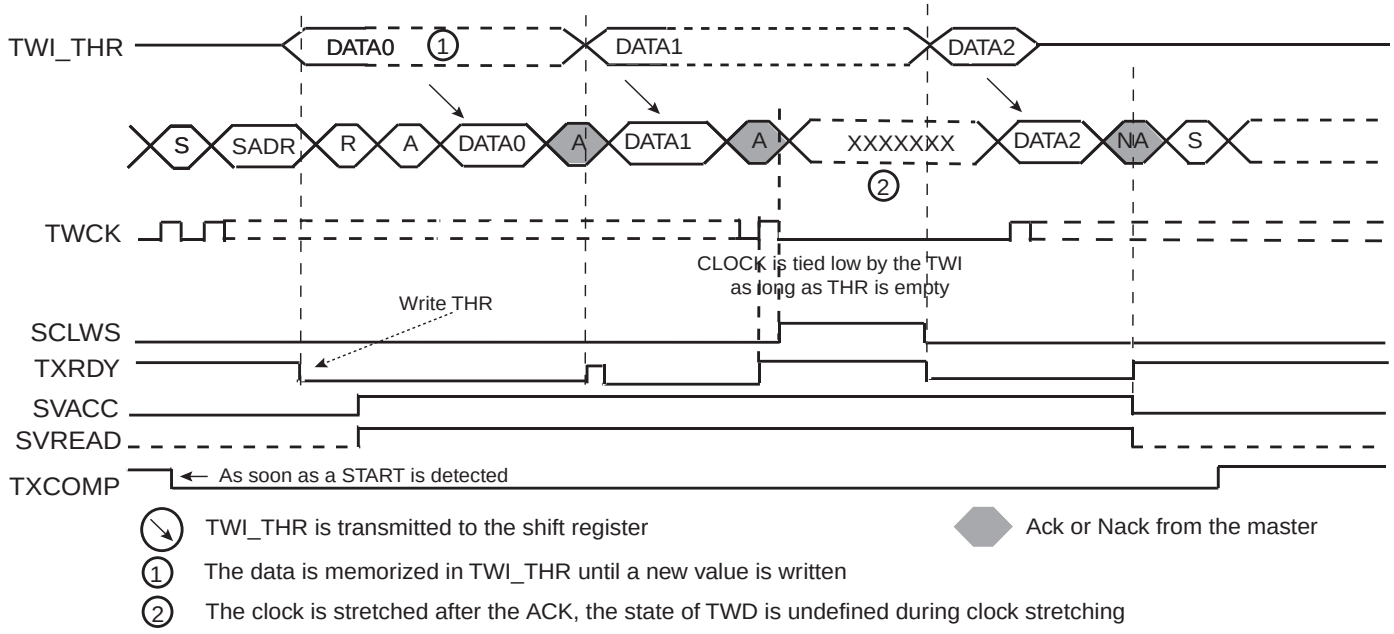
Clock Synchronization in Read Mode

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 30-27 describes the clock synchronization in Read mode.

© 2017 Microchip Technology Inc.

Figure 30-27: Clock Synchronization in Read Mode



Note 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.

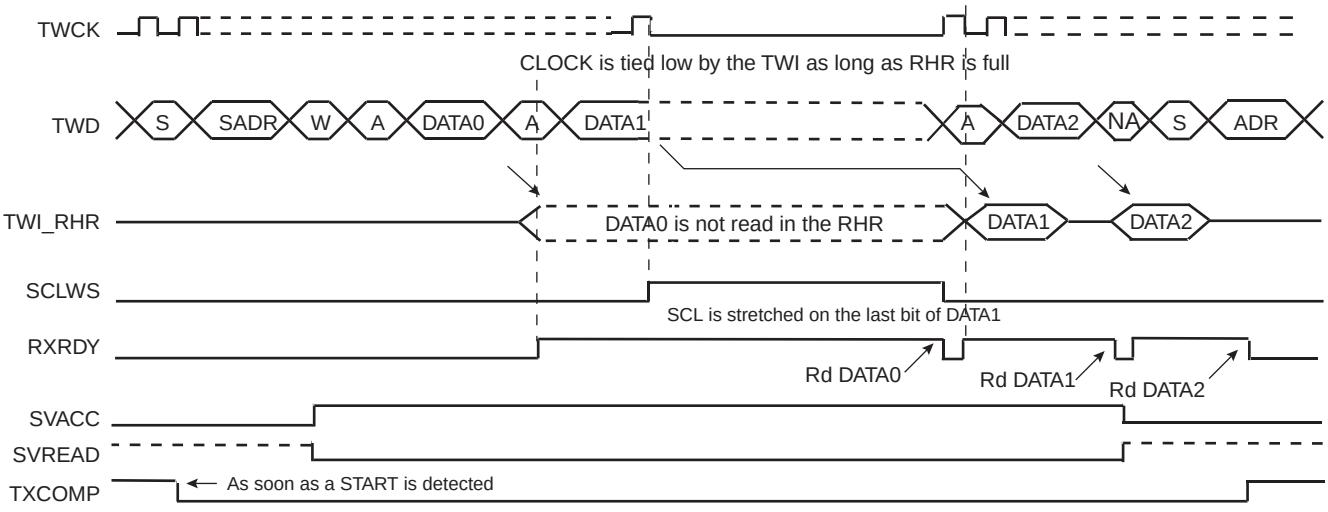
3: SCLWS is automatically set when the clock synchronization mechanism is started.

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 30-28 describes the clock synchronization in Read mode.

Figure 30-28: Clock Synchronization in Write Mode



Note 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.

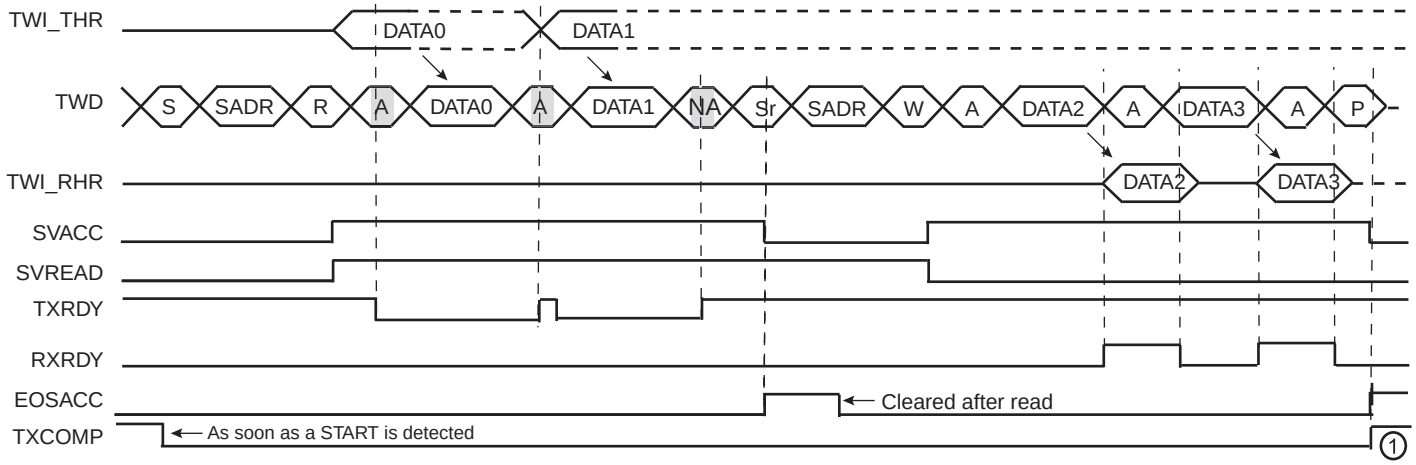
- Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 30-29 describes the repeated start + reversal from Read to Write mode.

Figure 30-29: Repeated Start + Reversal from Read to Write Mode

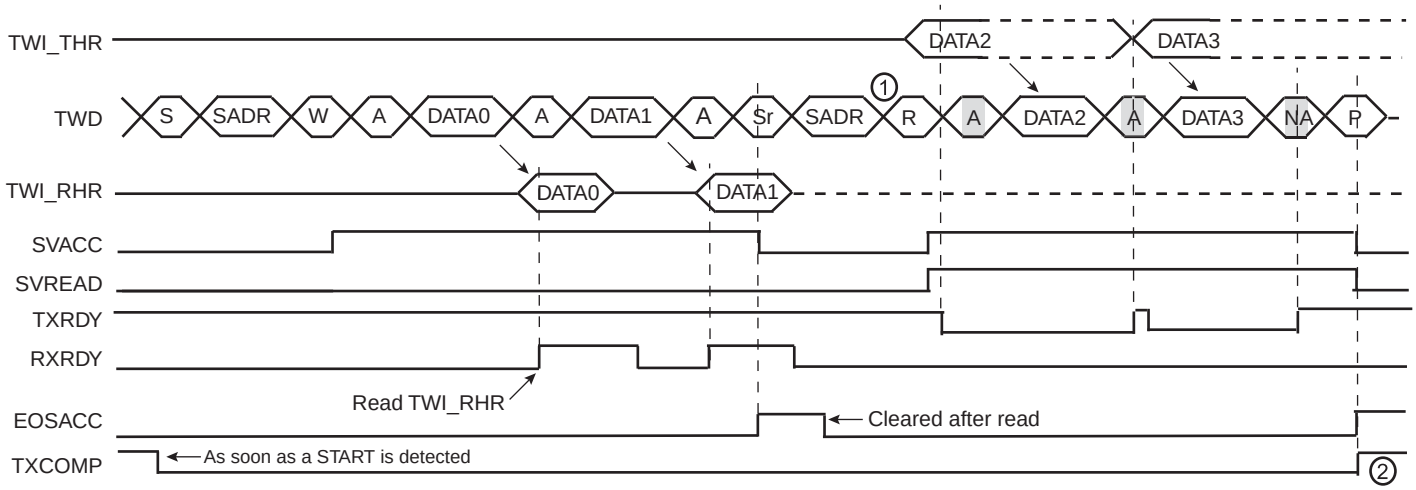


Note 1: TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.Figure 30-30 describes the repeated start + reversal from Write to Read mode.

Figure 30-30: Repeated Start + Reversal from Write to Read Mode



Note 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.

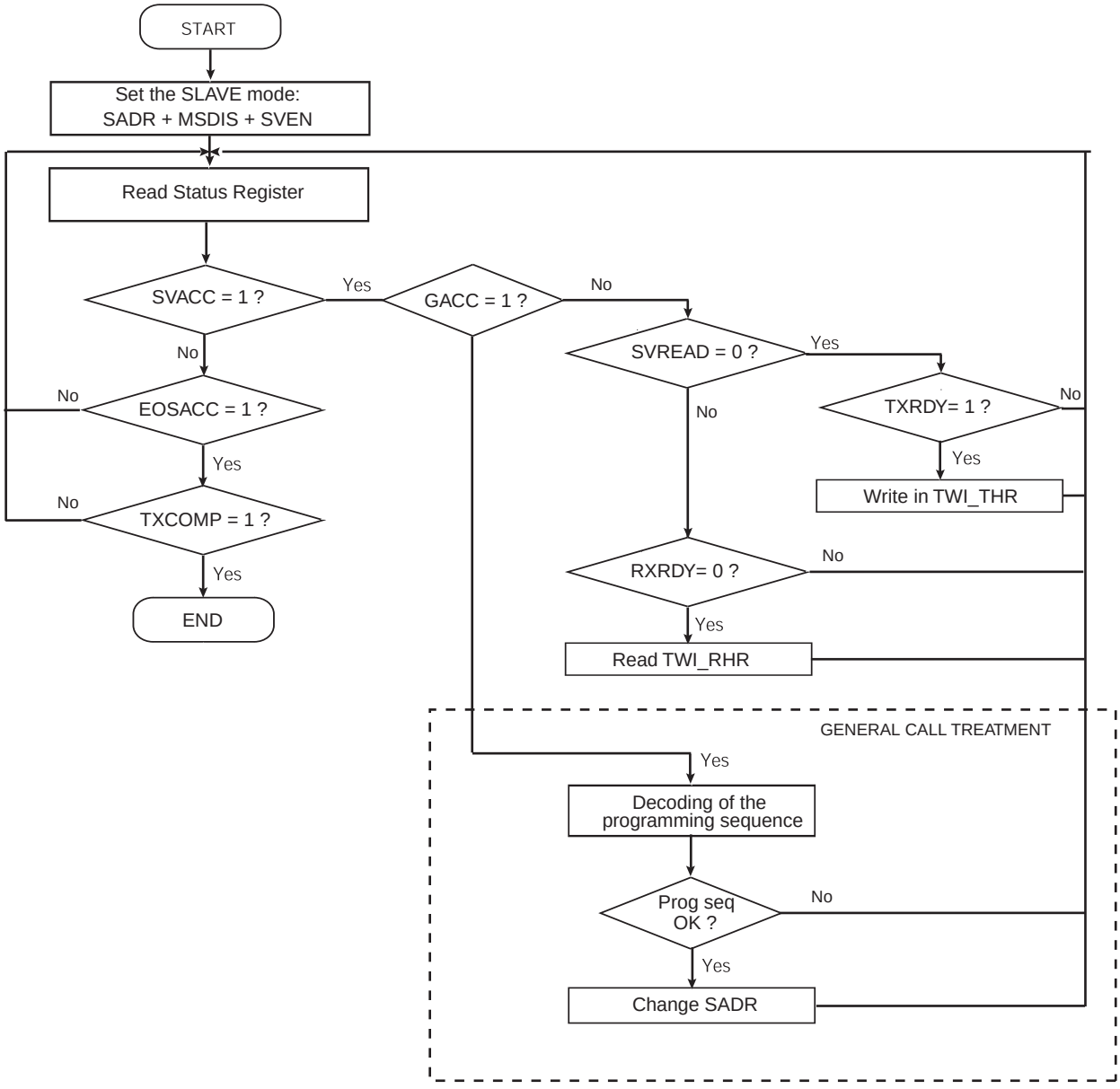
2: TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

© 2017 Microchip Technology Inc.

30.6.5.7 Read/Write Flowcharts

The flowchart shown in Figure 30-31 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 30-31: Read/Write Flowchart in Slave Mode



30.7 Two-wire Interface (TWI) User Interface

Table 30-4: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWI_CR	Write-only	–
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	–
0x28	Interrupt Disable Register	TWI_IDR	Write-only	–
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	–
0x38–0xFC	Reserved	–	–	–
0x100–0x124	Reserved for the PDC	–	–	–

© 2017 Microchip Technology Inc.

30.7.1 TWI Control Register

Name: TWI_CR

Access: 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
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.

© 2017 Microchip Technology Inc.

30.7.2 TWI Master Mode Register

Name: TWI_MMR

Access: Read/Write

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.

30.7.3 TWI Slave Mode Register

Name: TWI_SMR

Access: Read/Write

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.

© 2017 Microchip Technology Inc.

30.7.4 TWI Internal Address Register

Name: TWI_IADR

Access: Read/Write

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.

30.7.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Access: Read/Write

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.

© 2017 Microchip Technology Inc.

30.7.6 TWI Status Register

Name: TWI_SR

Access: 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
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 30-7 and in Figure 30-10.

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 30-27, Figure 30-28, Figure 30-29 and Figure 30-30.

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 30-10.

RXRDY behavior in Slave mode can be seen in Figure 30-25, Figure 30-28, Figure 30-29 and Figure 30-30.

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.

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 30-8.

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 30-24, Figure 30-27, Figure 30-29 and Figure 30-30.

© 2017 Microchip Technology Inc.

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 30-24, Figure 30-25, Figure 30-29 and Figure 30-30.

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 30-24, Figure 30-25, Figure 30-29 and Figure 30-30.

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 30-26.

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 30-27 and Figure 30-28.

© 2017 Microchip Technology Inc.

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 30-29 and Figure 30-30.

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.

30.7.7 TWI Interrupt Enable Register

Name: TWI_IER

Access: 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
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.

© 2017 Microchip Technology Inc.

30.7.8 TWI Interrupt Disable Register

Name: TWI_IDR

Access: 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
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.

30.7.9 TWI Interrupt Mask Register

Name: TWI_IMR

Access: 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
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.

© 2017 Microchip Technology Inc.

30.7.10 TWI Receive Holding Register

Name: TWI_RHR

Access: 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
RXDATA							

RXDATA: Master or Slave Receive Holding Data

30.7.11 TWI Transmit Holding Register

Name: TWI_THR

Access: 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
TXDATA							

TXDATA: Master or Slave Transmit Holding Data

© 2017 Microchip Technology Inc.

31. Universal Synchronous Asynchronous Receiver Transmitter (USART)

31.1 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.

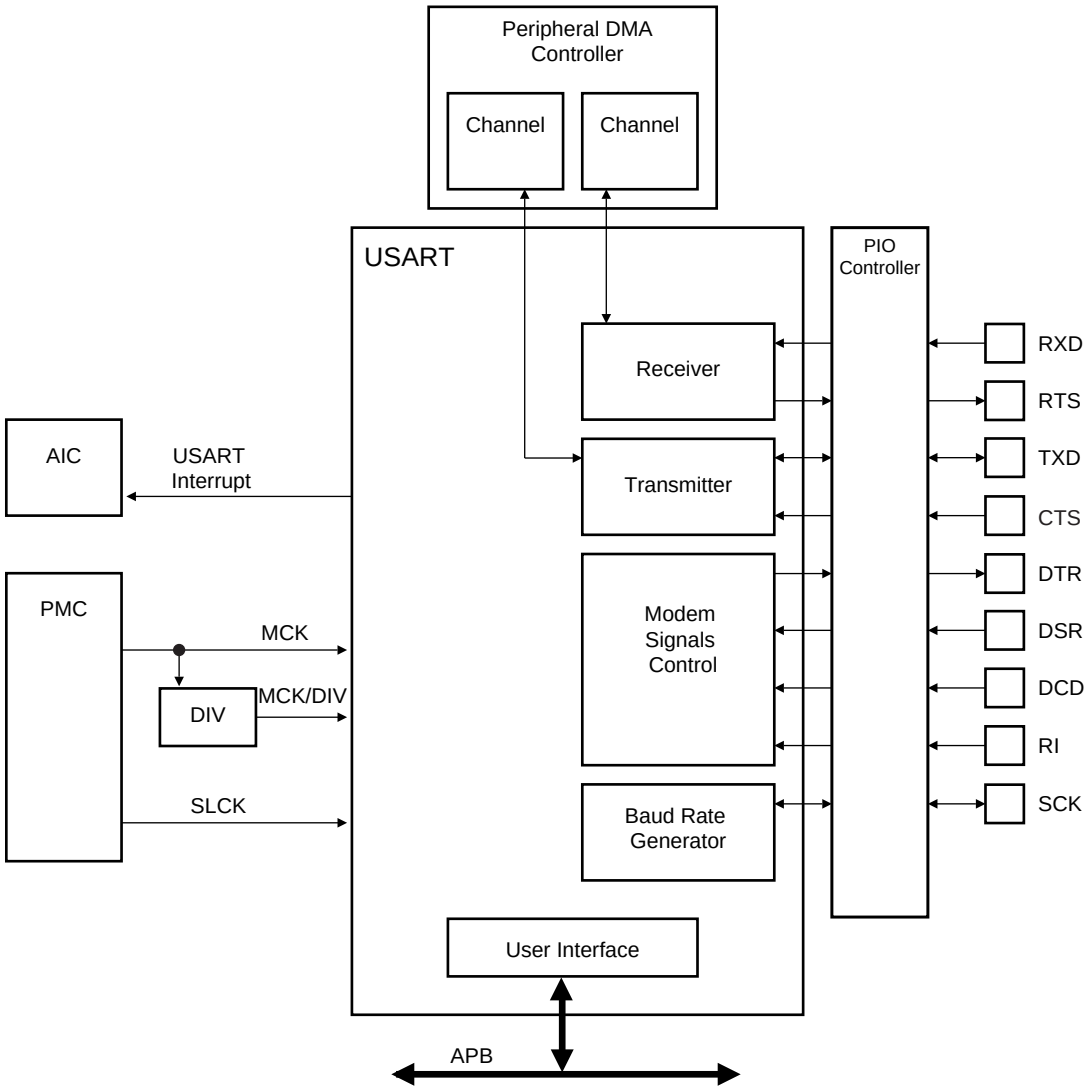
The USART features three test modes: remote loopback, local loopback and automatic echo.

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.

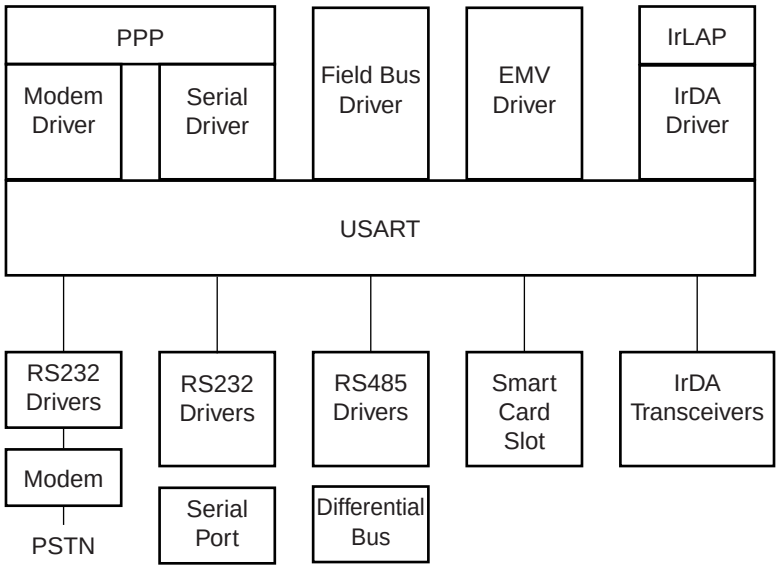
31.2 Block Diagram

Figure 31-1: USART Block Diagram



31.3 Application Block Diagram

Figure 31-2: Application Block Diagram



31.4 I/O Lines Description

Table 31-1: 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

© 2017 Microchip Technology Inc.

31.5 Product Dependencies

31.5.1 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 USART0 is 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.

31.5.2 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.

31.5.3 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.

© 2017 Microchip Technology Inc.

31.6 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

31.6.1 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:

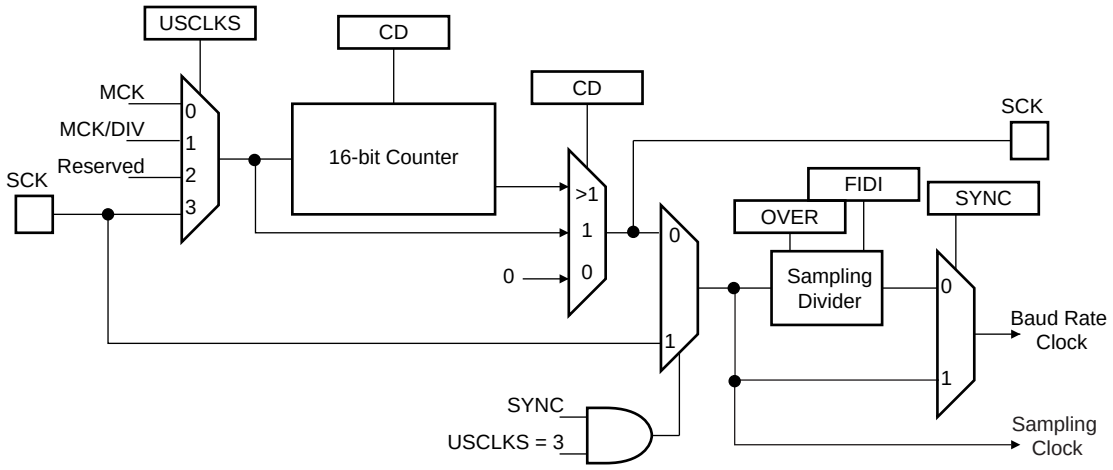
- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

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.

© 2017 Microchip Technology Inc.

Figure 31-3: Baud Rate Generator



31.6.1.1 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.

31.6.1.2 Baud Rate Calculation Example

Table 31-2 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 31-2: Baud Rate Example (OVER = 0)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
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%

Table 31-2: Baud Rate Example (OVER = 0) (Continued)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
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%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

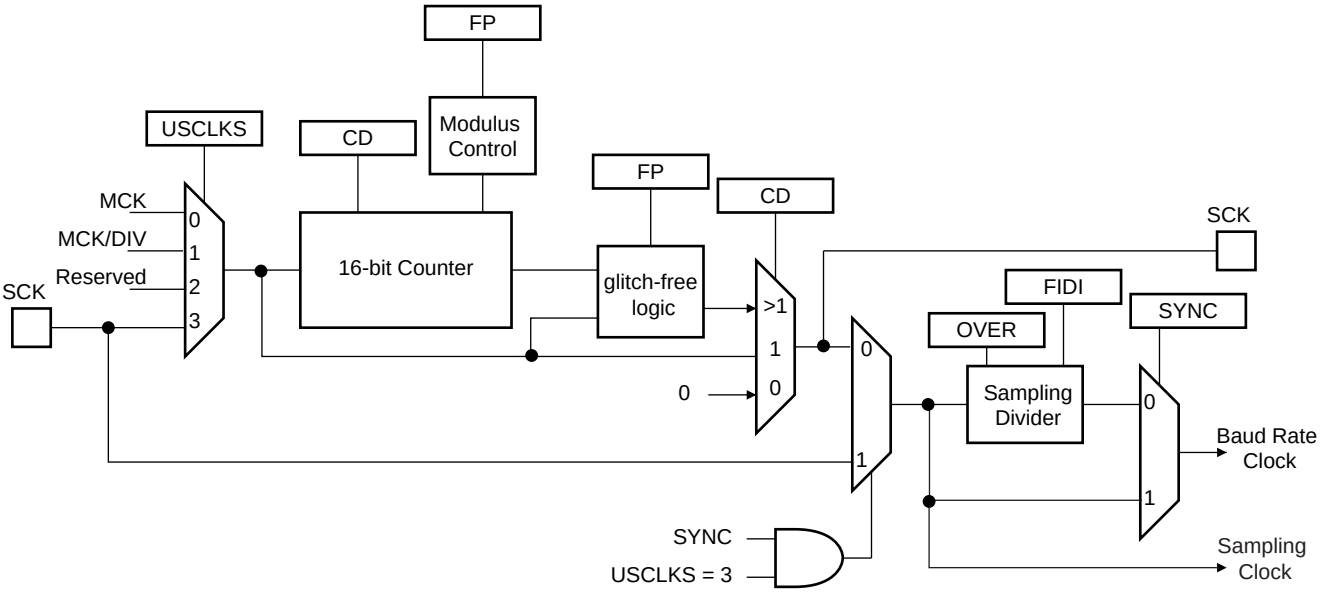
31.6.1.3 Fractional Baud Rate in Asynchronous Mode

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 in Figure 31-4.

Figure 31-4: Fractional Baud Rate Generator



© 2017 Microchip Technology Inc.

31.6.1.4 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.

31.6.1.5 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 31-3.

Table 31-3: 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 31-4.

Table 31-4: 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 31-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 31-5: Possible Values for the Fi/Di Ratio

Fi/Di	372	558	744	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

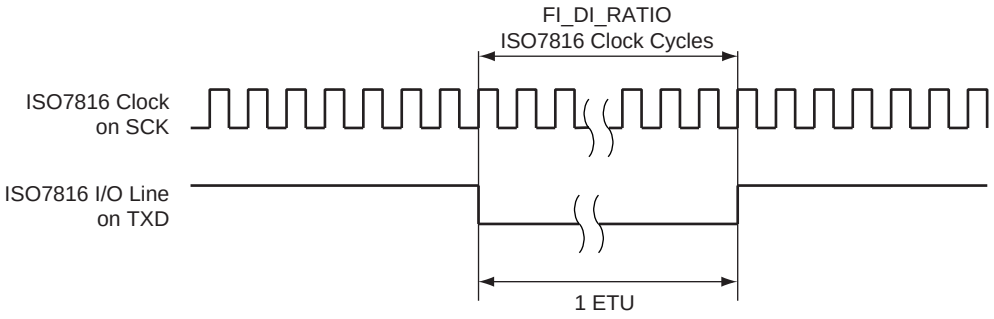
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. 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 31-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 31-5: Elementary Time Unit (ETU)



31.6.2 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.

31.6.3 Synchronous and Asynchronous Modes

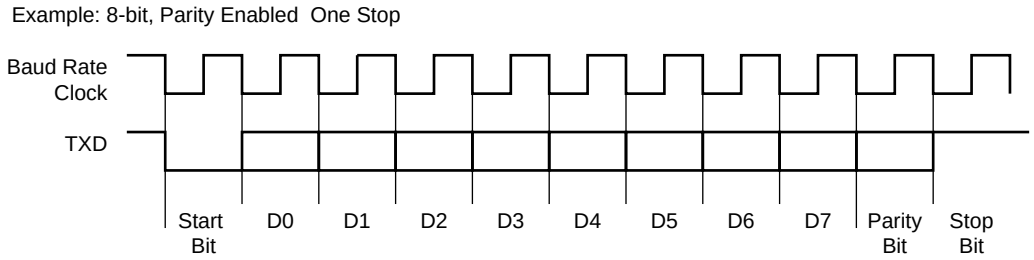
31.6.3.1 Transmitter Operations

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 number of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

© 2017 Microchip Technology Inc.

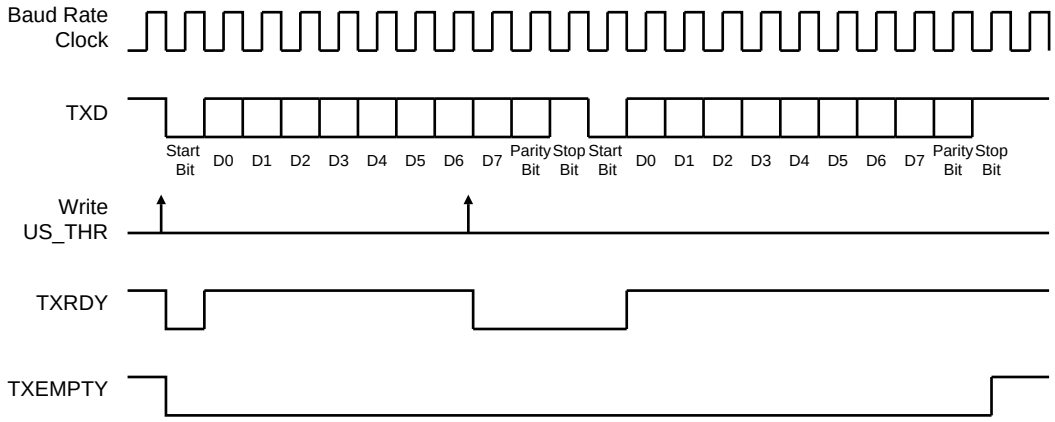
Figure 31-6: Character Transmit



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.

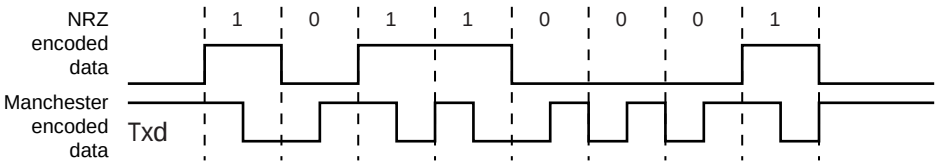
Figure 31-7: Transmitter Status



31.6.3.2 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 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 31-8 illustrates this coding scheme.

Figure 31-8: NRZ to Manchester Encoding

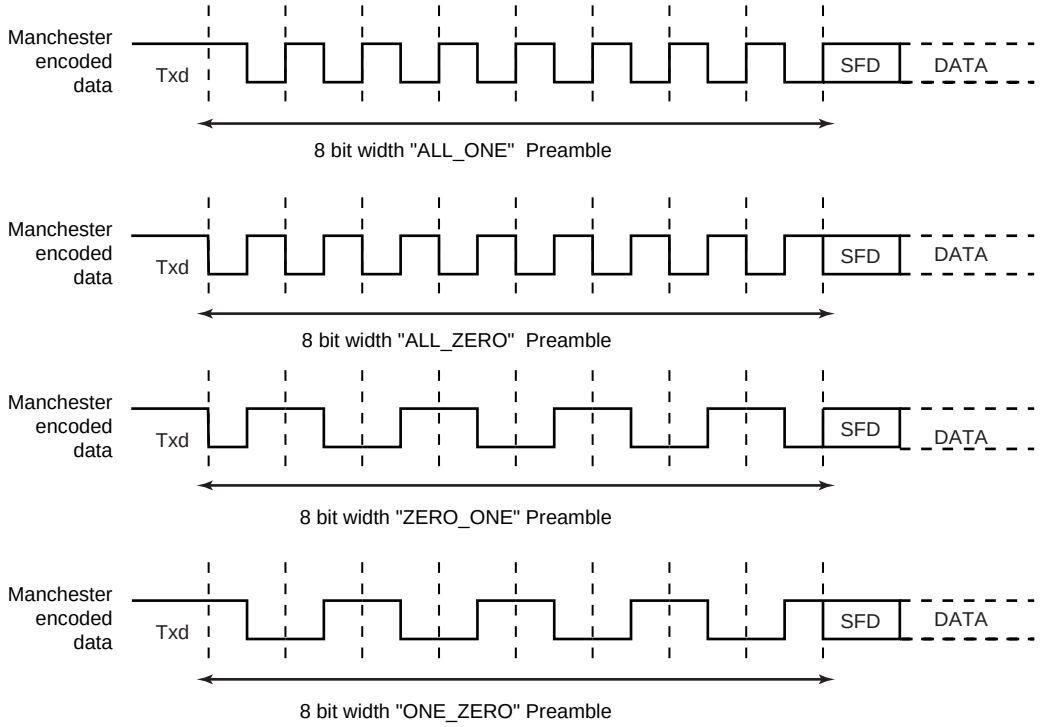


The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble 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 31-9 illustrates and defines the valid patterns. To improve flexi-

© 2017 Microchip Technology Inc.

bility, 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.

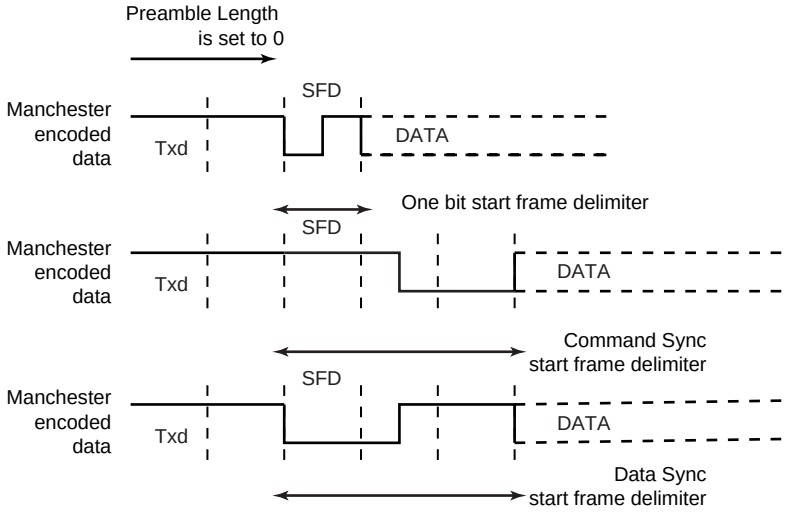
Figure 31-9: Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT field in the US_MR. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 31-10 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 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 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. The USART character format is modified and includes sync information.

© 2017 Microchip Technology Inc.

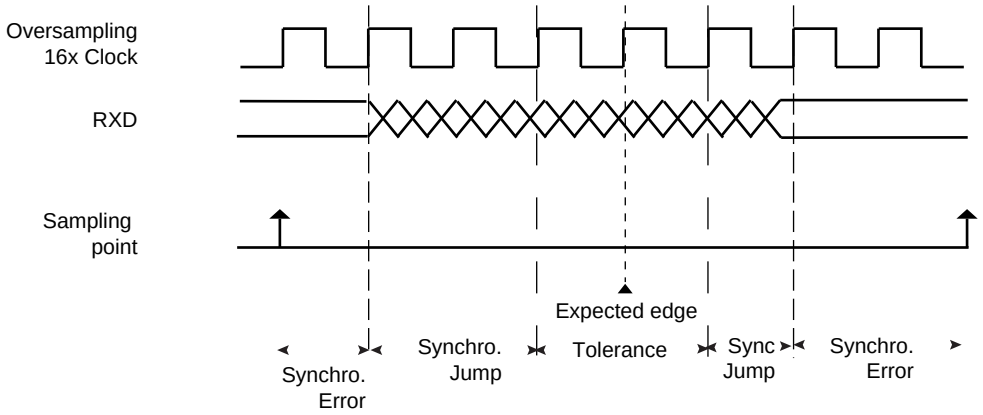
Figure 31-10: Start Frame Delimiter



31.6.3.3 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 31-11: Bit Resynchronization



31.6.3.4 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 31-12 and Figure 31-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 31-12: Asynchronous Start Detection

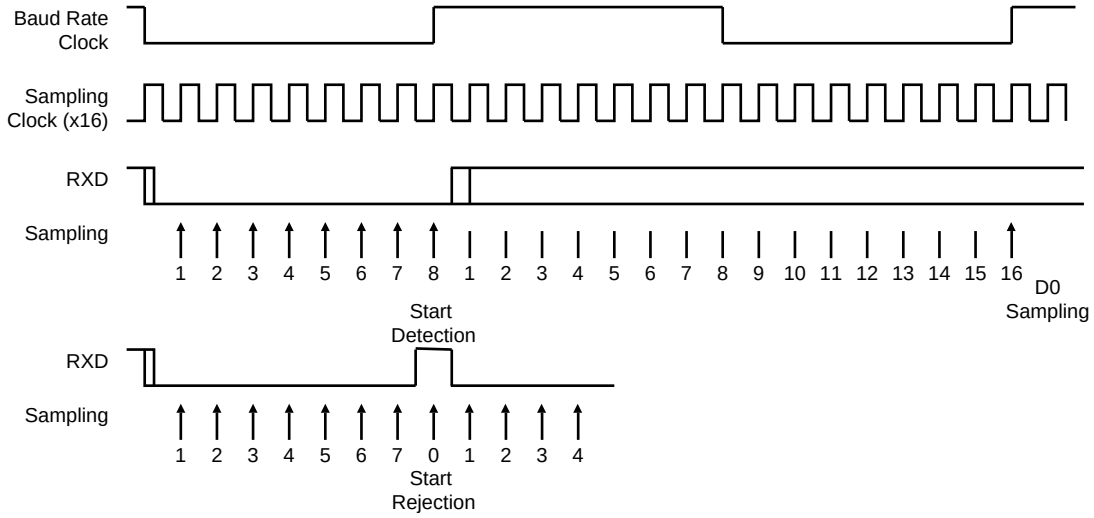
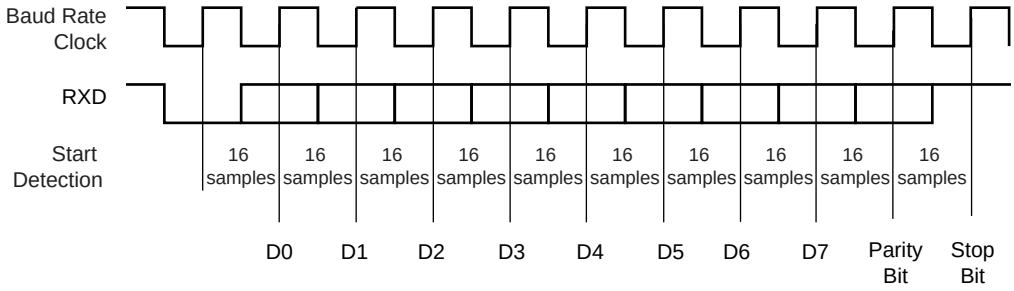


Figure 31-13: Asynchronous Character Reception

Example: 8-bit, Parity Enabled



31.6.3.5 Manchester Decoder

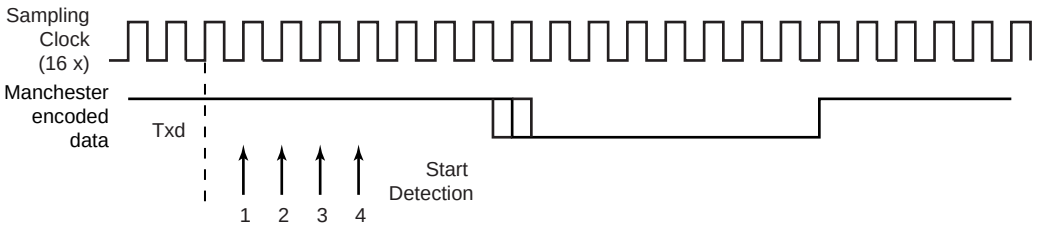
When the MAN field in US_MR is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

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 31-9 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 31-14.. The sample pulse rejection mechanism applies.

© 2017 Microchip Technology Inc.

Figure 31-14: 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 31-15 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, MANERR flag in US_CSR is raised. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. See Figure 31-16 for an example of Manchester error detection during data phase.

Figure 31-15: Preamble Pattern Mismatch

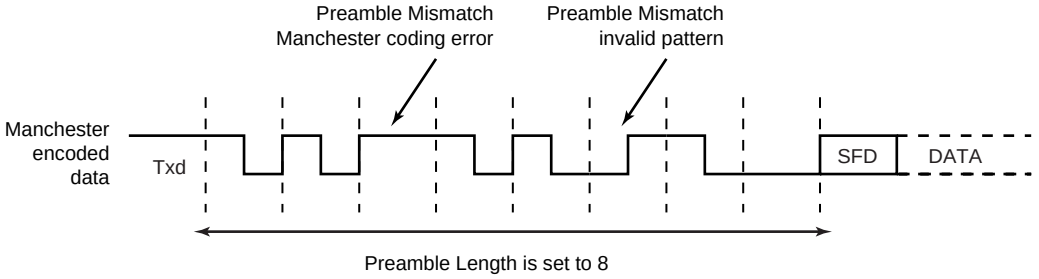
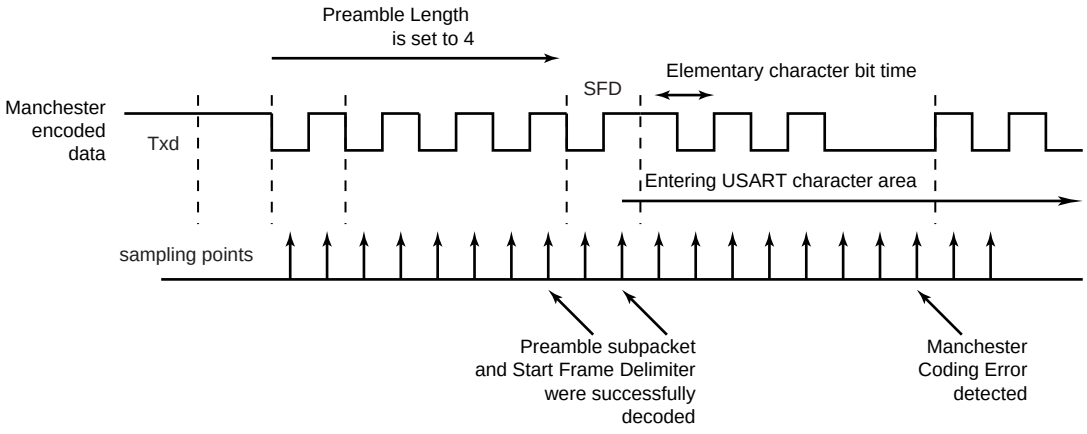


Figure 31-16: Manchester Error Flag



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 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.

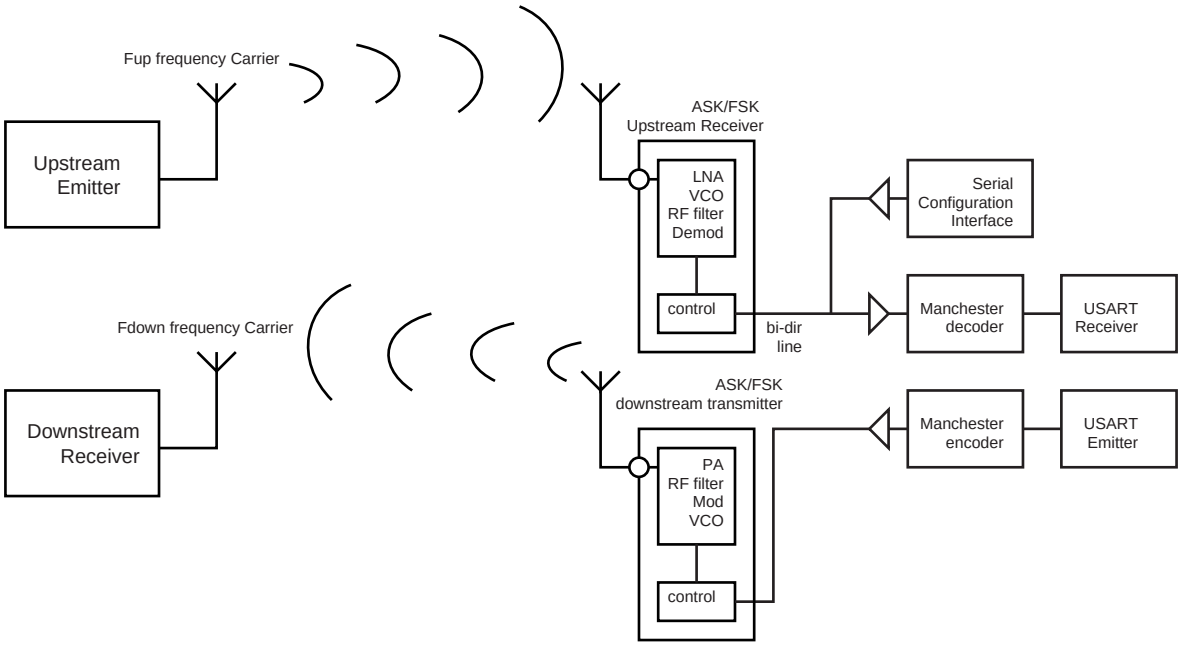
© 2017 Microchip Technology Inc.

31.6.3.6 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 31-17.

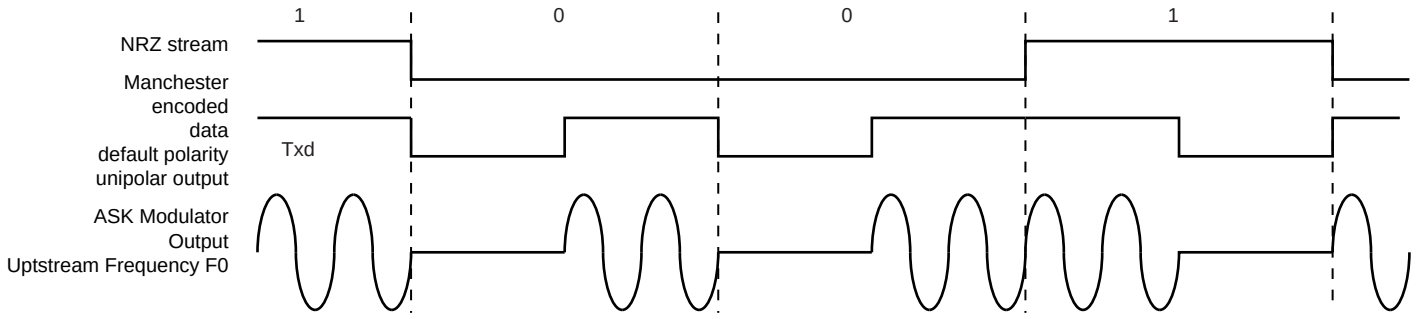
Figure 31-17: Manchester Encoded Characters RF Transmission



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 31-18 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 31-19.

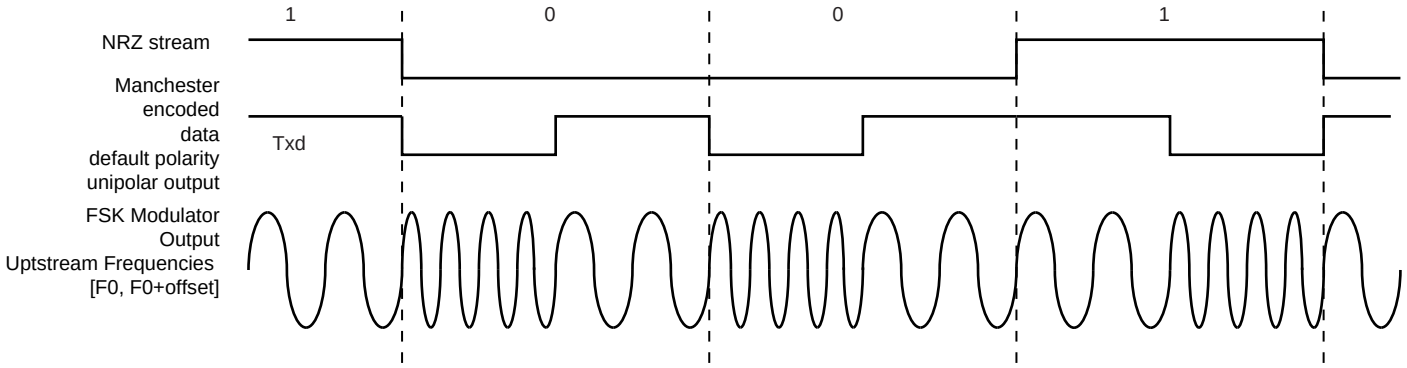
From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside 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 31-18: ASK Modulator Output



© 2017 Microchip Technology Inc.

Figure 31-19: FSK Modulator Output



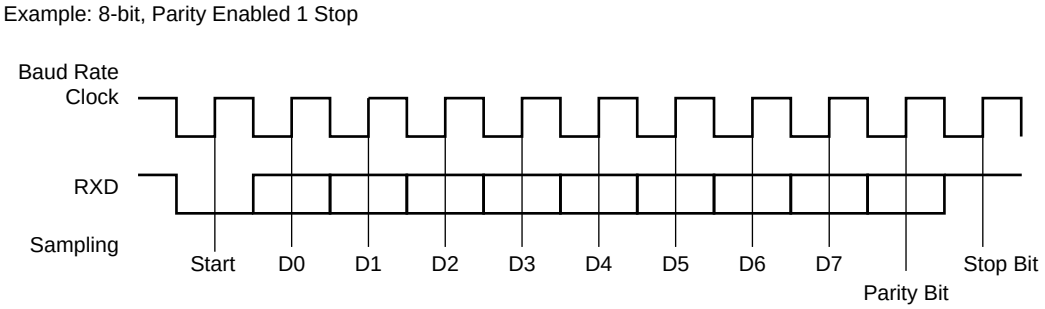
31.6.3.7 Synchronous Receiver

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 31-20 illustrates a character reception in synchronous mode.

Figure 31-20: Synchronous Mode Character Reception

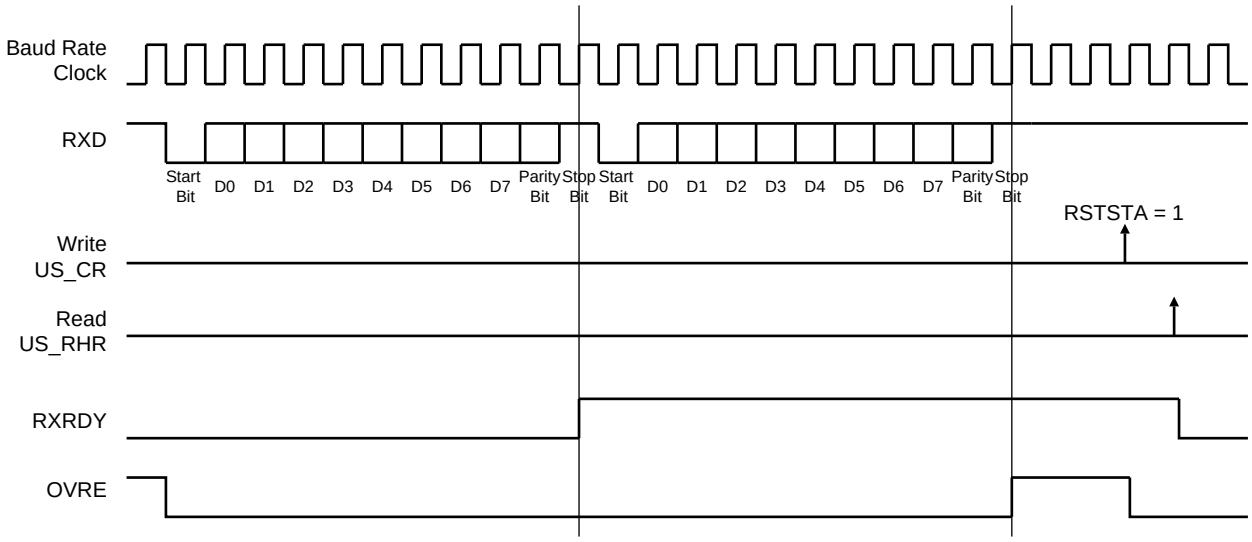


31.6.3.8 Receiver Operations

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.

© 2017 Microchip Technology Inc.

Figure 31-21: Receiver Status



31.6.3.9 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 Section 31.6.3.10 Multidrop Mode. 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 31-6 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.

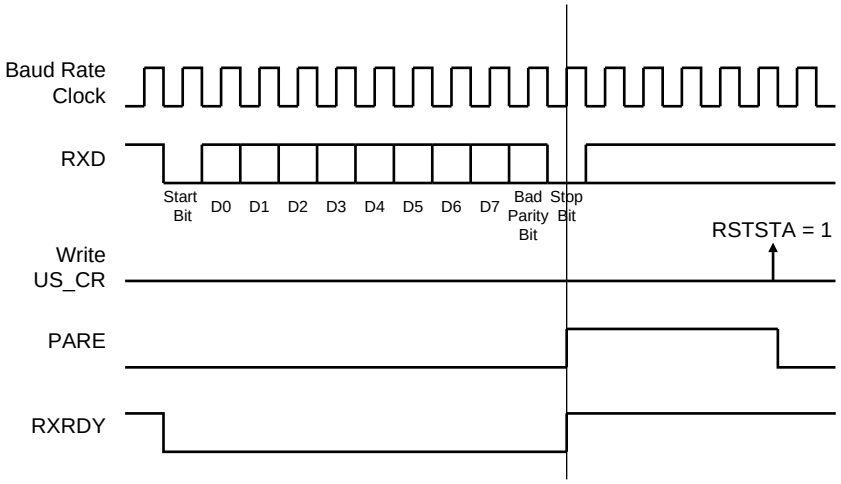
Table 31-6: Parity Bit Examples

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 31-22 illustrates the parity bit status setting and clearing.

© 2017 Microchip Technology Inc.

Figure 31-22: Parity Error



31.6.3.10 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.

31.6.3.11 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 31-23, 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.

© 2017 Microchip Technology Inc.

Figure 31-23: Timeguard Operations

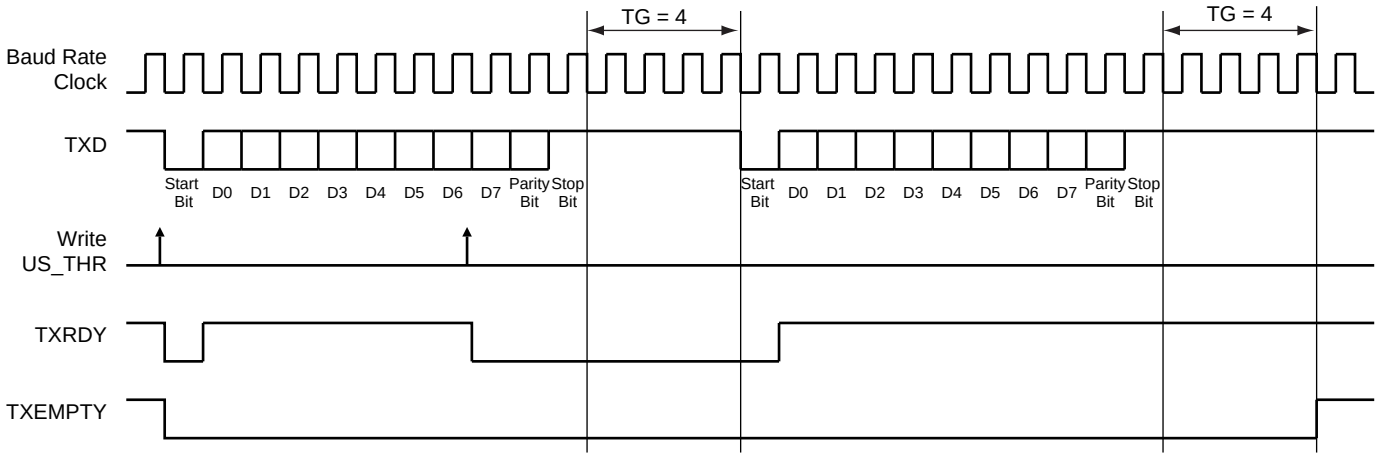


Table 31-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 31-7: Maximum Timeguard Length Depending on Baud Rate

Baud Rate (bit/s)	Bit Time (µs)	Timeguard (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

31.6.3.12 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.

© 2017 Microchip Technology Inc.

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 31-24 shows the block diagram of the Receiver Time-out feature.

Figure 31-24: Receiver Time-out Block Diagram

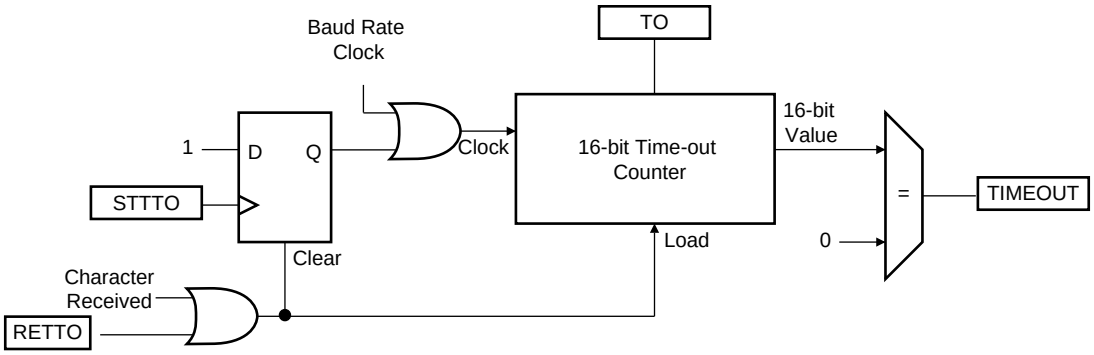


Table 31-8 gives the maximum time-out period for some standard baud rates.

Table 31-8: Maximum Time-out Period

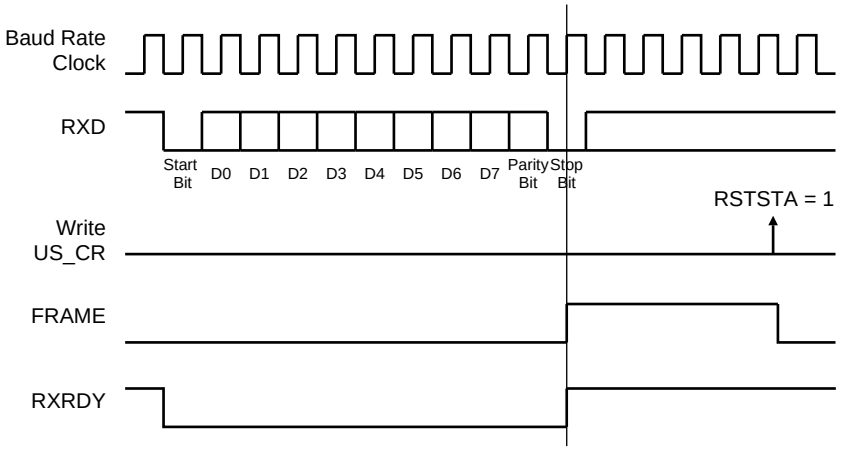
Baud Rate (bit/s)	Bit Time (µs)	Time-out (ms)
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962
56000	18	1 170
57600	17	1 138
200000	5	328

31.6.3.13 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 31-25: Framing Error Status



31.6.3.14 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.

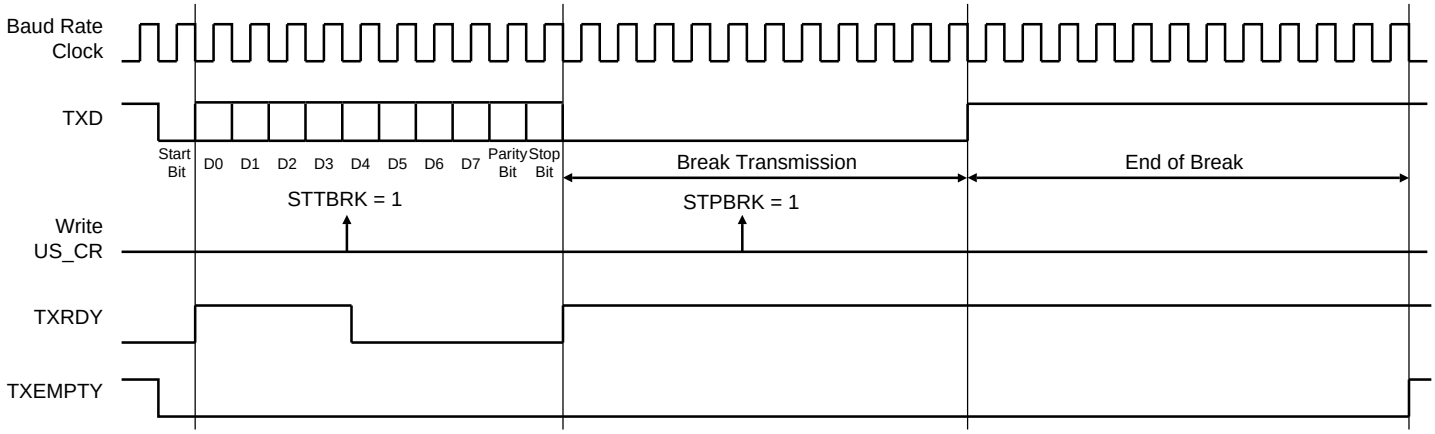
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 31-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 31-26: Break Transmission



© 2017 Microchip Technology Inc.

31.6.3.15 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.

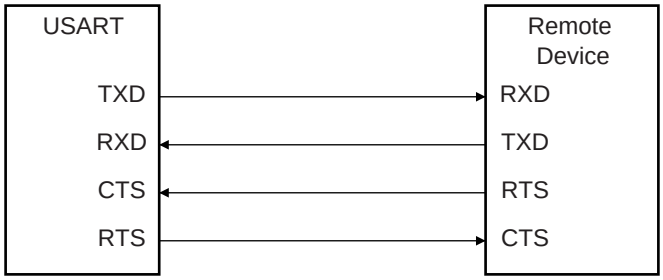
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

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.

31.6.3.16 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 31-27.

Figure 31-27: 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 31-28 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 31-28: Receiver Behavior when Operating with Hardware Handshaking

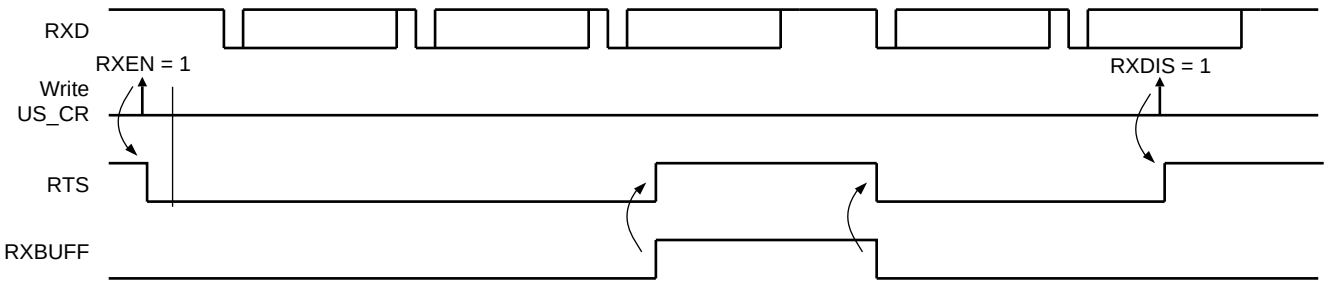
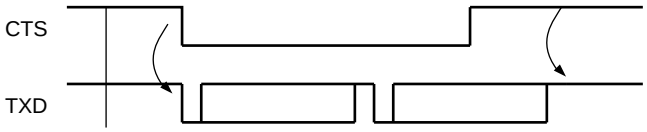


Figure 31-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 31-29: Transmitter Behavior when Operating with Hardware Handshaking



© 2017 Microchip Technology Inc.

31.6.4 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.

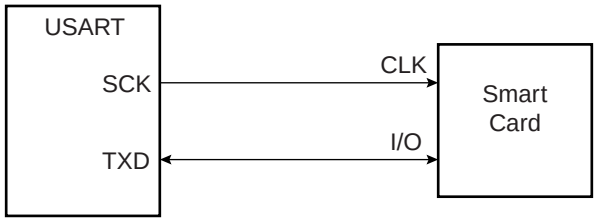
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

31.6.4.1 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 Section 31.6.1 Baud Rate Generator).

The USART connects to a smart card as shown in Figure 31-30. 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 31-30: 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 Section 31.7.2 USART Mode Register and Section PAR: Parity Type.

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).

31.6.4.2 Protocol T = 0

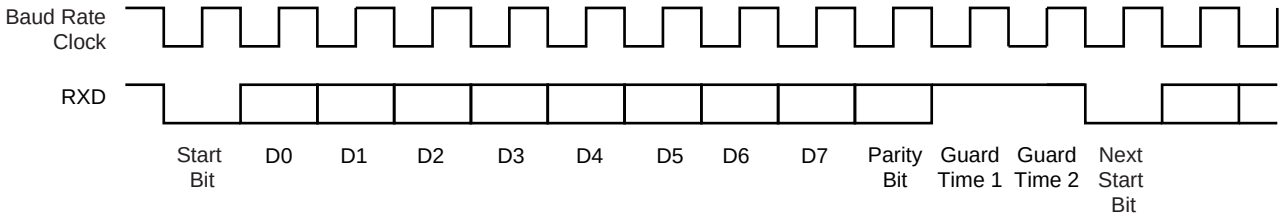
In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

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 31-31.

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 31-32. 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.

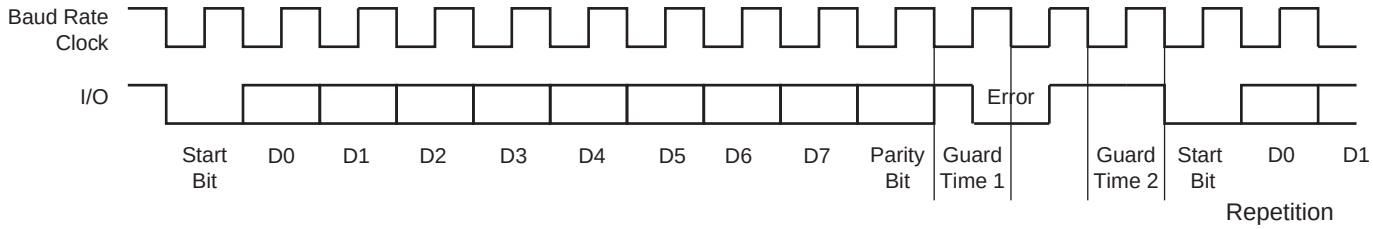
When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

Figure 31-31: T = 0 Protocol without Parity Error



© 2017 Microchip Technology Inc.

Figure 31-32: T = 0 Protocol with Parity Error



31.6.4.3 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.

31.6.4.4 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.

31.6.4.5 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.

31.6.4.6 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.

31.6.4.7 Protocol T = 1

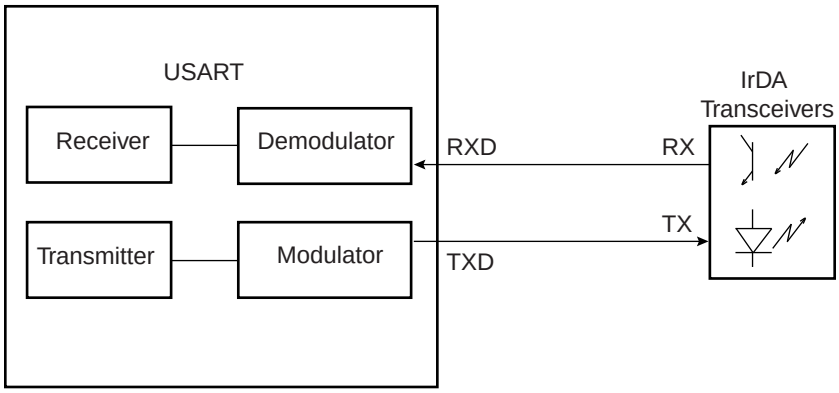
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

31.6.5 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 31-33. 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 31-33: 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

31.6.5.1 IrDA Modulation

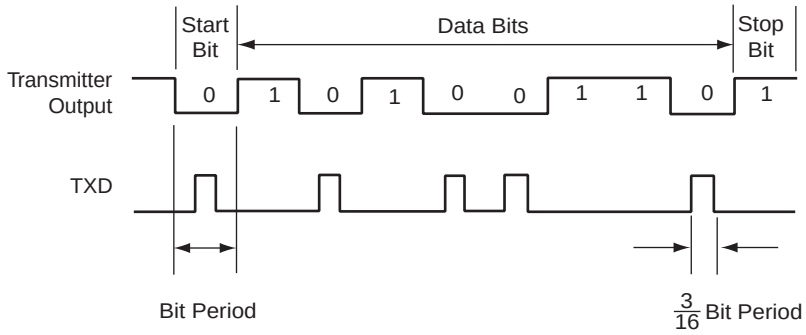
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 31-9.

Table 31-9: 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 31-34 shows an example of character transmission.

Figure 31-34: IrDA Modulation



© 2017 Microchip Technology Inc.

31.6.5.2 IrDA Baud Rate

Table 31-10 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 31-10: IrDA Baud Rate Error

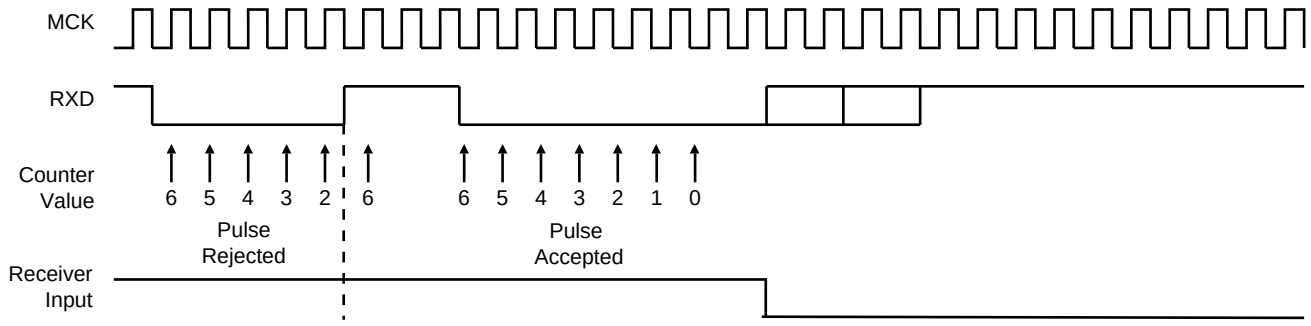
Peripheral Clock	Baud Rate (bit/s)	CD	Baud Rate Error	Pulse Time (µs)
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	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

31.6.5.3 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 31-35 illustrates the operations of the IrDA demodulator.

Figure 31-35: IrDA Demodulator Operations

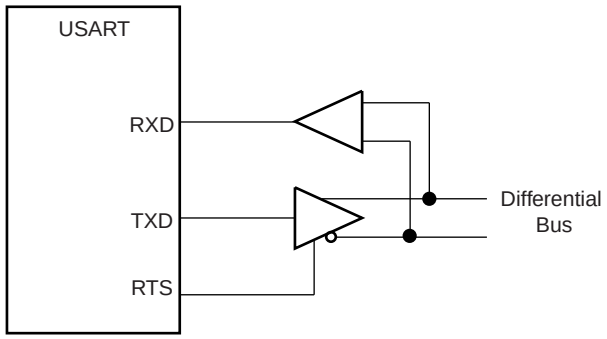


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

31.6.6 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 31-36.

Figure 31-36: Typical Connection to a RS485 Bus

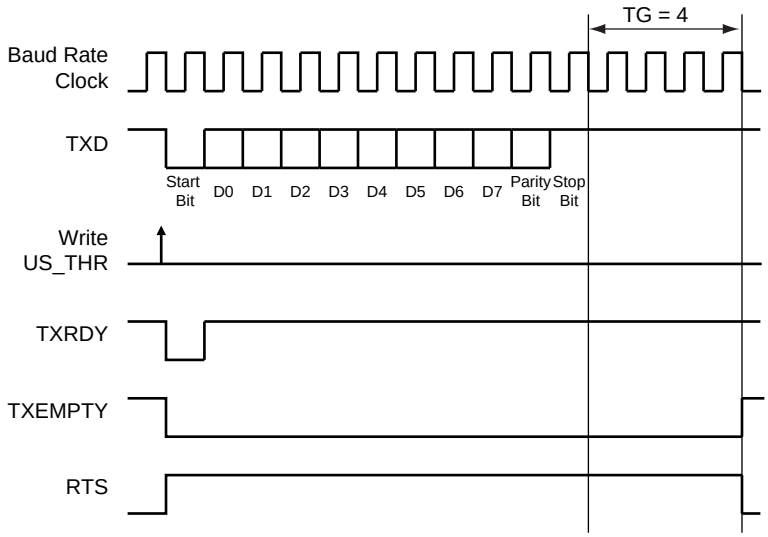


The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

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 31-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

© 2017 Microchip Technology Inc.

Figure 31-37: Example of RTS Drive with Timeguard



31.6.7 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 31-11 gives the correspondence of the USART signals with modem connection standards.

Table 31-11: 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.

© 2017 Microchip Technology Inc.

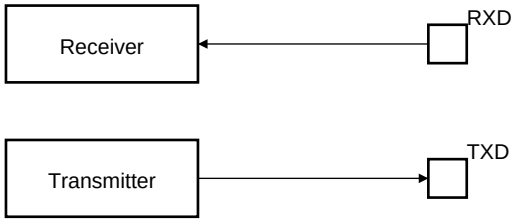
31.6.8 Test Modes

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.

31.6.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

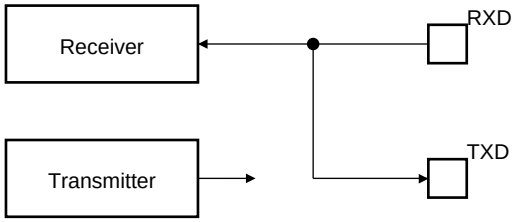
Figure 31-38: Normal Mode Configuration



31.6.8.2 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 31-39. 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.

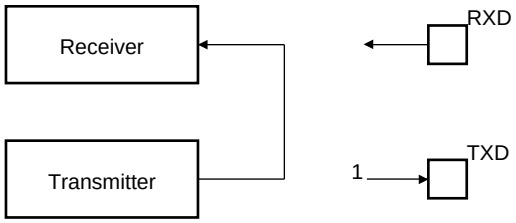
Figure 31-39: Automatic Echo Mode Configuration



31.6.8.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 31-40. 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 31-40: Local Loopback Mode Configuration

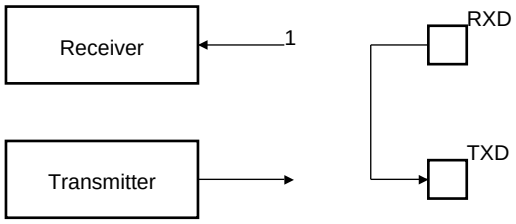


31.6.8.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 31-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

© 2017 Microchip Technology Inc.

Figure 31-41: Remote Loopback Mode Configuration



31.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 31-12: 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
0x5C–0xFC	Reserved	–	–	–
0x100–0x128	Reserved for PDC Registers	–	–	–

© 2017 Microchip Technology Inc.

31.7.1 USART Control Register

Name:US_CR

Access: 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.

© 2017 Microchip Technology Inc.

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.

© 2017 Microchip Technology Inc.

31.7.2 USART Mode Register

Name:US_MR

Access: 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.

© 2017 Microchip Technology Inc.

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.

© 2017 Microchip Technology Inc.

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.

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.

© 2017 Microchip Technology Inc.

31.7.3 USART Interrupt Enable Register

Name: US_IER

Access: 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

© 2017 Microchip Technology Inc.

31.7.4 USART Interrupt Disable Register

Name:US_IDR

Access: 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

© 2017 Microchip Technology Inc.

31.7.5 USART Interrupt Mask Register

Name:US_IMR

Access: 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

© 2017 Microchip Technology Inc.

31.7.6 USART Channel Status Register

Name:US_CSR

Access: 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).

© 2017 Microchip Technology Inc.

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.

© 2017 Microchip Technology Inc.

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.

31.7.7 USART Receive Holding Register

Name:US_RHR

Access: 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.

© 2017 Microchip Technology Inc.

31.7.8 USART Transmit Holding Register

Name:US_THR

Access: 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.

31.7.9 USART Baud Rate Generator Register

Name:US_BRGR

Access: 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			
1–65535	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.

© 2017 Microchip Technology Inc.

31.7.10 USART Receiver Time-out Register

Name:US_RTOR

Access: 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.

31.7.11 USART Transmitter Timeguard Register

Name:US_TTGR

Access: 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.

© 2017 Microchip Technology Inc.

31.7.12 USART FI DI RATIO Register

Name:US_FIDI

Access: 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
–	–	–	–	–	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.

31.7.13 USART Number of Errors Register

Name:US_NER

Access: 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.

© 2017 Microchip Technology Inc.

31.7.14 USART IrDA FILTER Register

Name:US_IF

Access: 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.

31.7.15 USART Manchester Configuration Register

Name:US_MAN

Access: 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.

© 2017 Microchip Technology Inc.

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.

32. Synchronous Serial Controller (SSC)

32.1 Overview

The Microchip Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

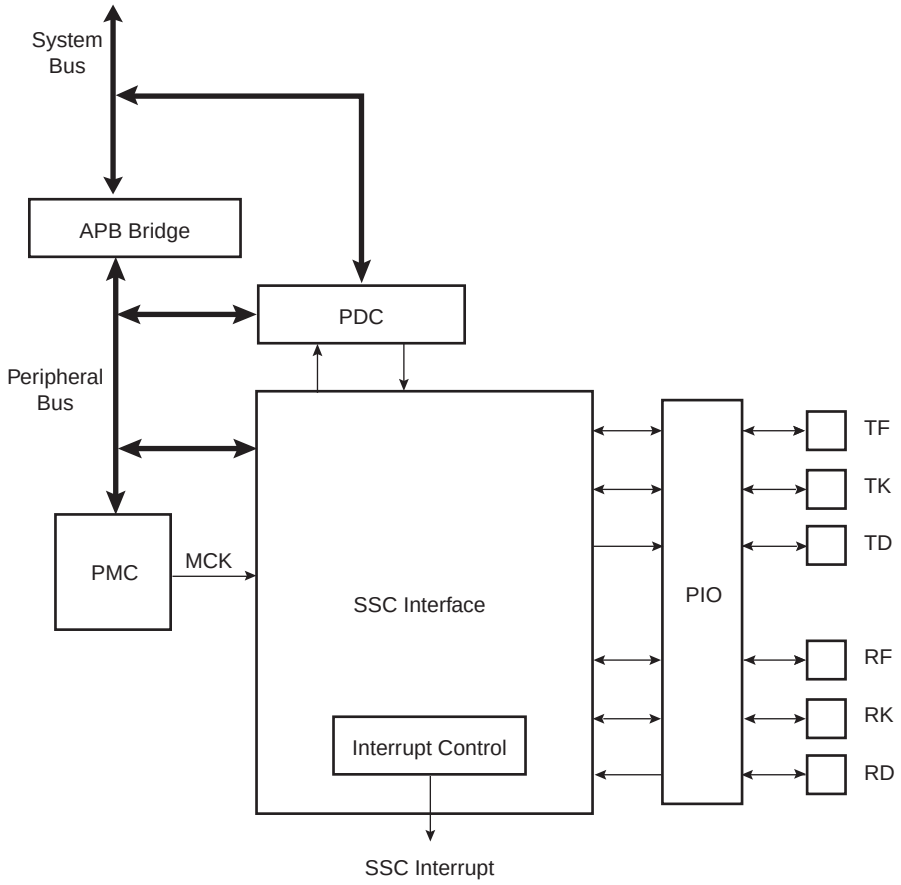
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODECs in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

32.2 Block Diagram

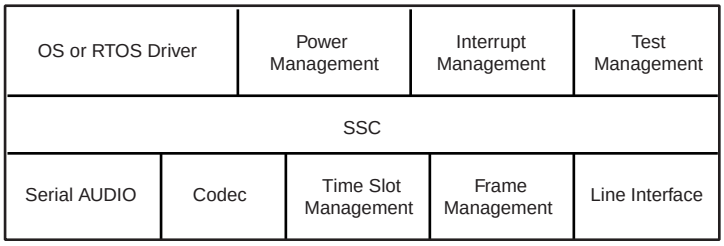
Figure 32-1: Block Diagram



© 2017 Microchip Technology Inc.

32.3 Application Block Diagram

Figure 32-2: Application Block Diagram



32.4 Pin Name List

Table 32-1: I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

32.5 Product Dependencies

32.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

32.5.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

32.5.3 Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

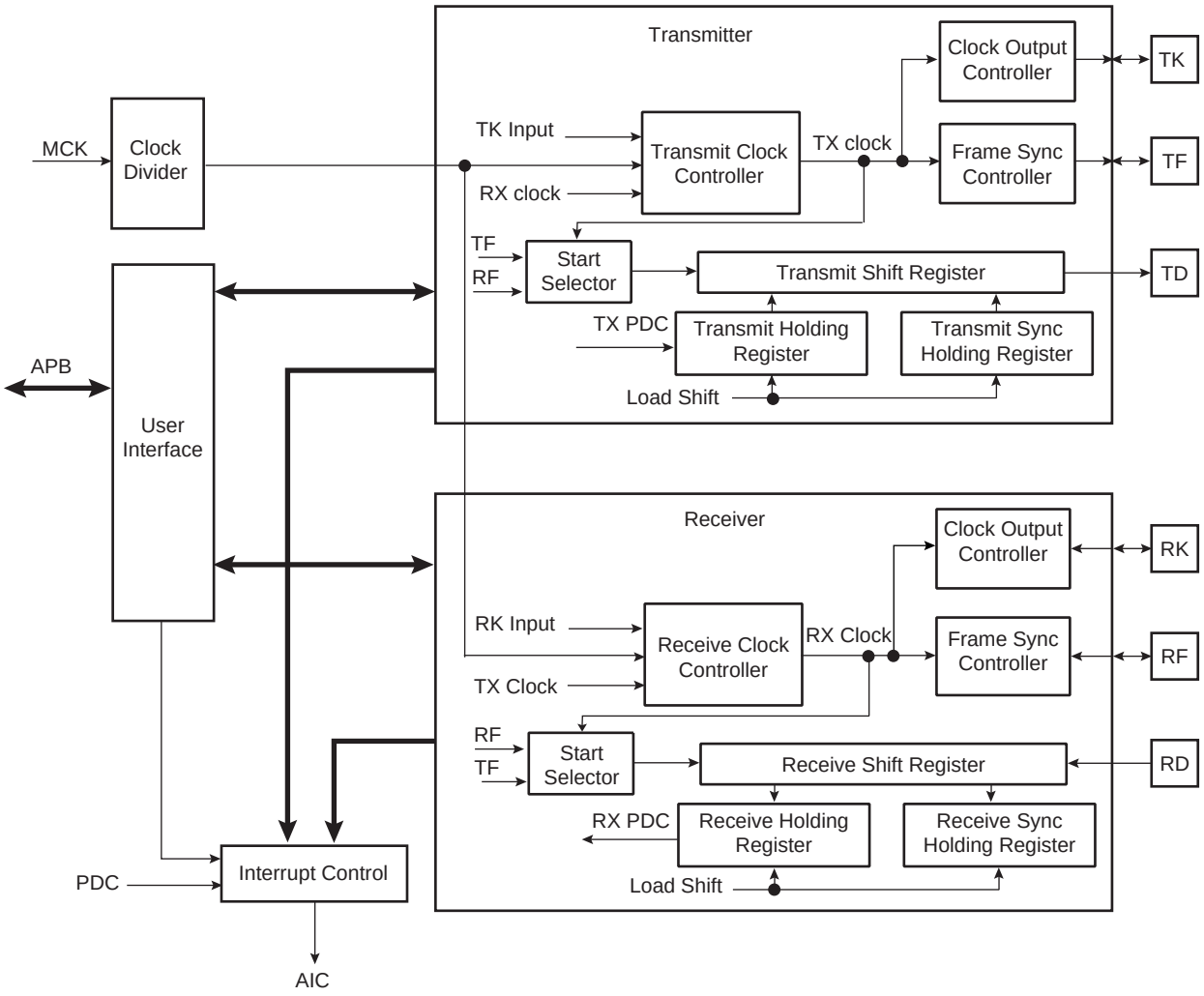
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

32.6 Functional Description

This section contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

Figure 32-3: SSC Functional Block Diagram



32.6.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

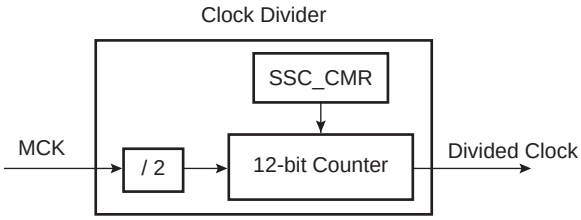
Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

© 2017 Microchip Technology Inc.

32.6.1.1 Clock Divider

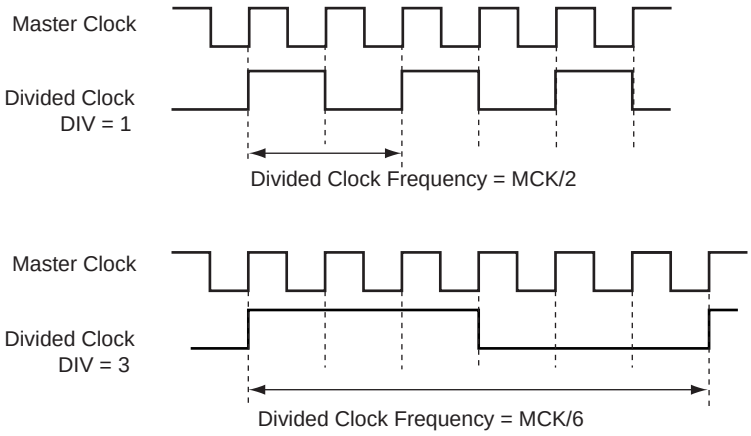
Figure 32-4: Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC_CMR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 32-5: Divided Clock Generation

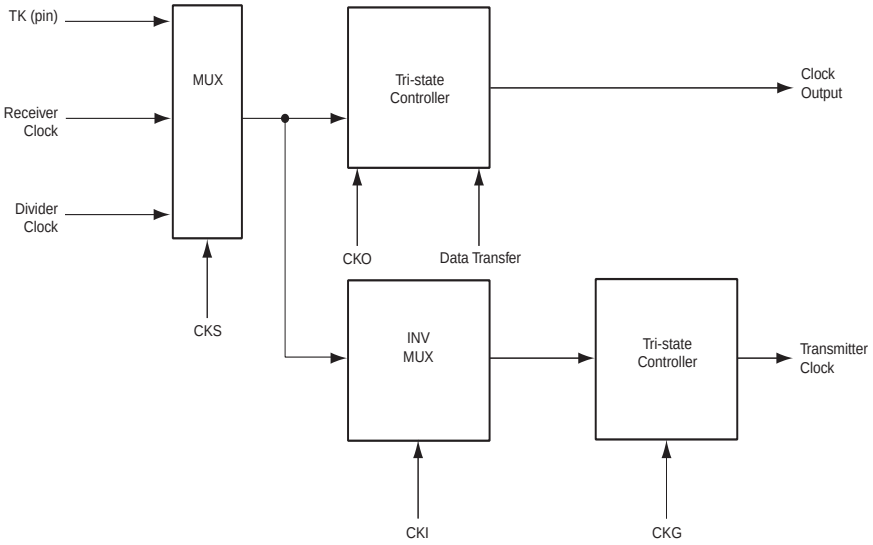


32.6.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

Figure 32-6: Transmitter Clock Management

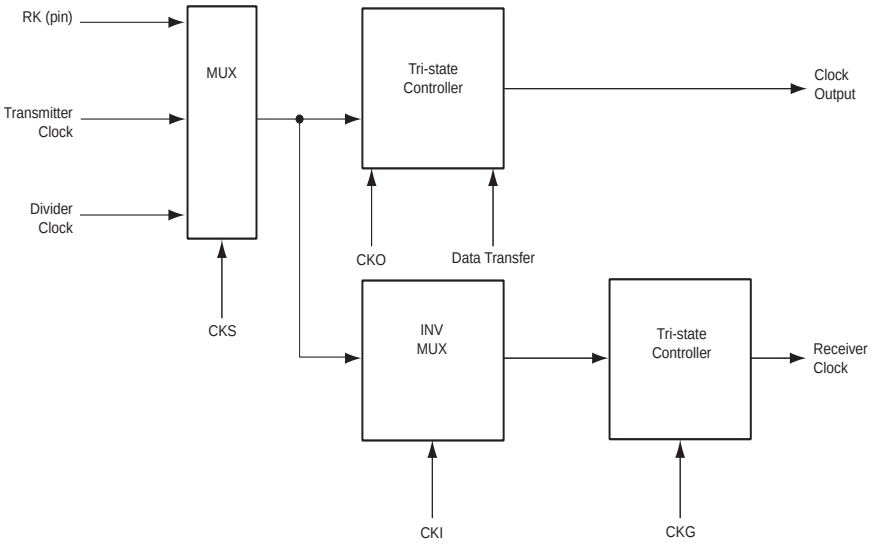


32.6.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 32-7: Receiver Clock Management



32.6.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Master Clock divided by 2 if Receiver Frame Synchro is input
- Master Clock divided by 3 if Receiver Frame Synchro is output

© 2017 Microchip Technology Inc.

In addition, the maximum clock speed allowed on the TK pin is:

- Master Clock divided by 6 if Transmit Frame Synchro is input
- Master Clock divided by 2 if Transmit Frame Synchro is output

32.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

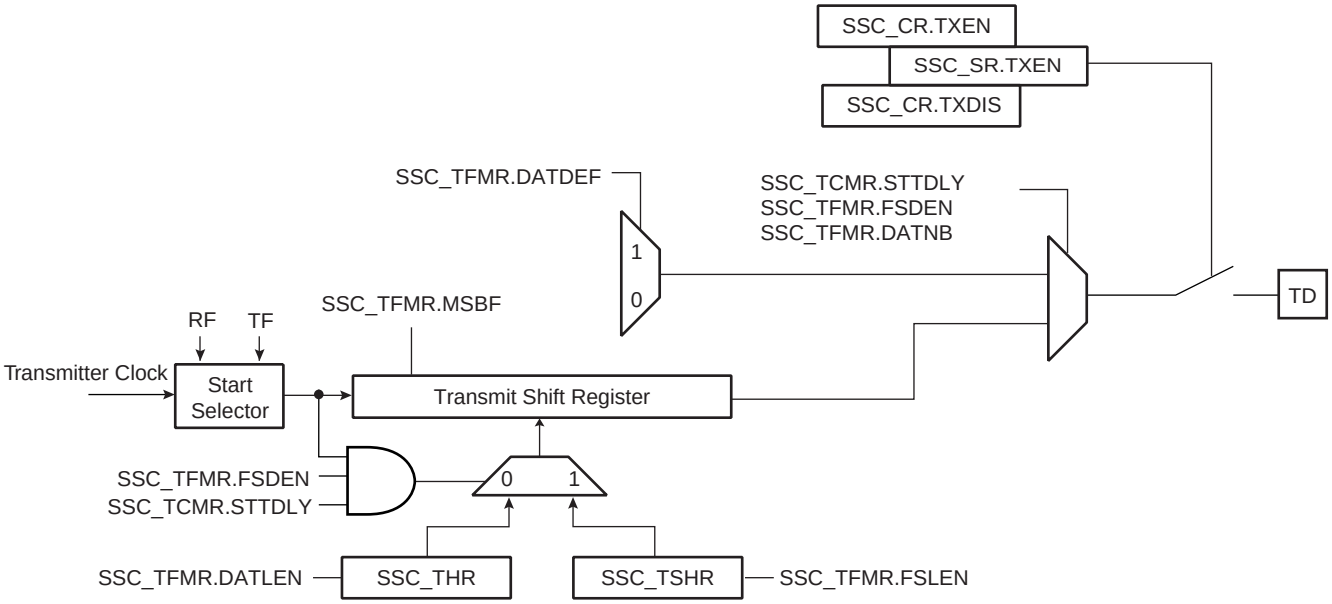
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See Section 32.6.4 Start.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See Section 32.6.5 Frame Sync.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC_SR and additional data can be loaded in the holding register.

Figure 32-8: Transmitter Block Diagram



32.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

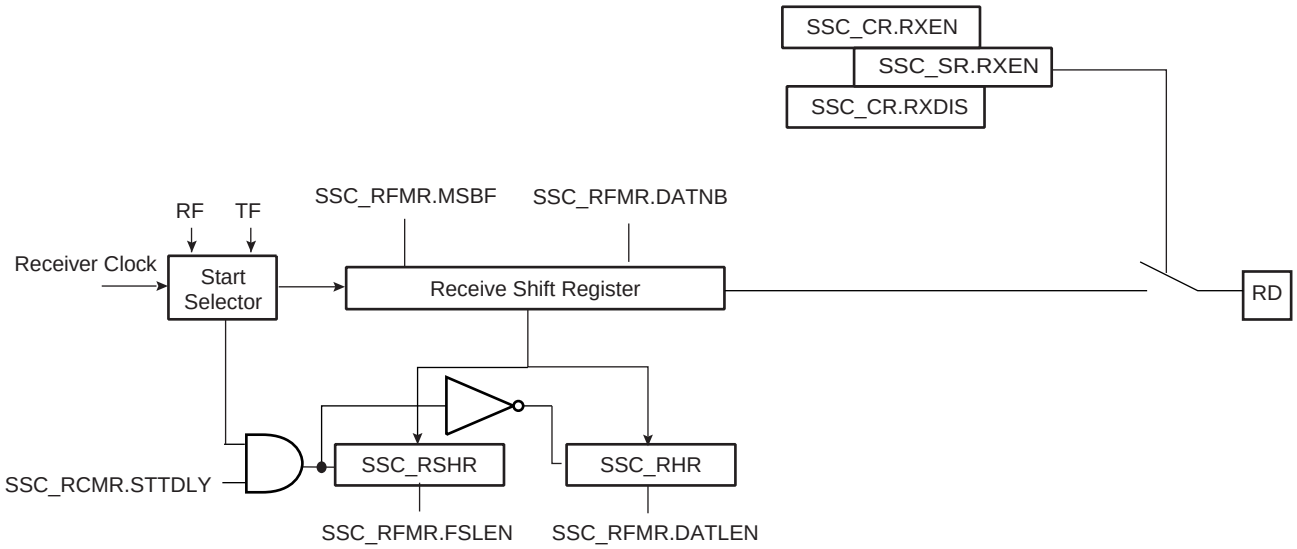
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See Section 32.6.4 Start.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See Section 32.6.5 Frame Sync.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR, the status flag OVERUN is set in SSC_SR and the receiver shift register is transferred in the RHR.

Figure 32-9: Receiver Block Diagram



32.6.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

© 2017 Microchip Technology Inc.

Figure 32-10: Transmit Start Mode

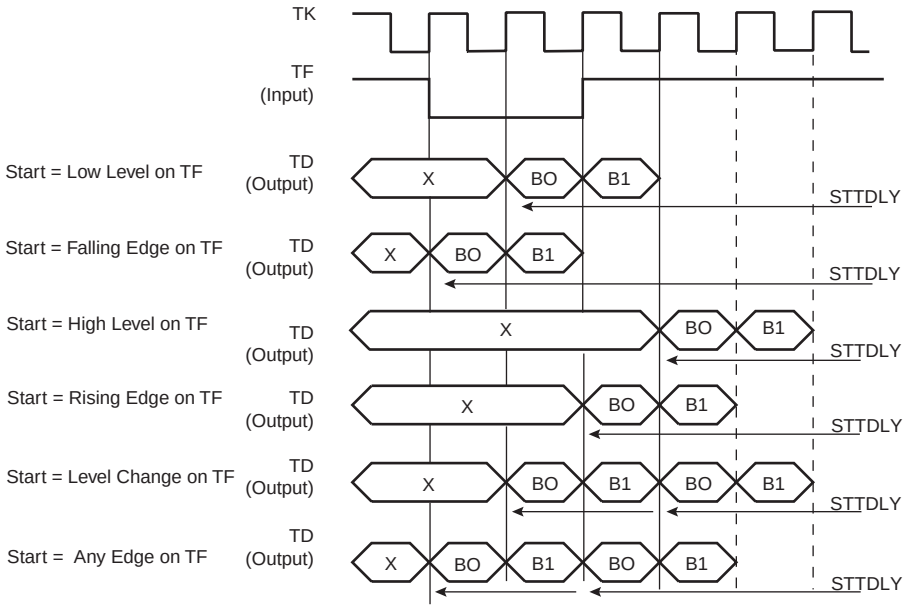
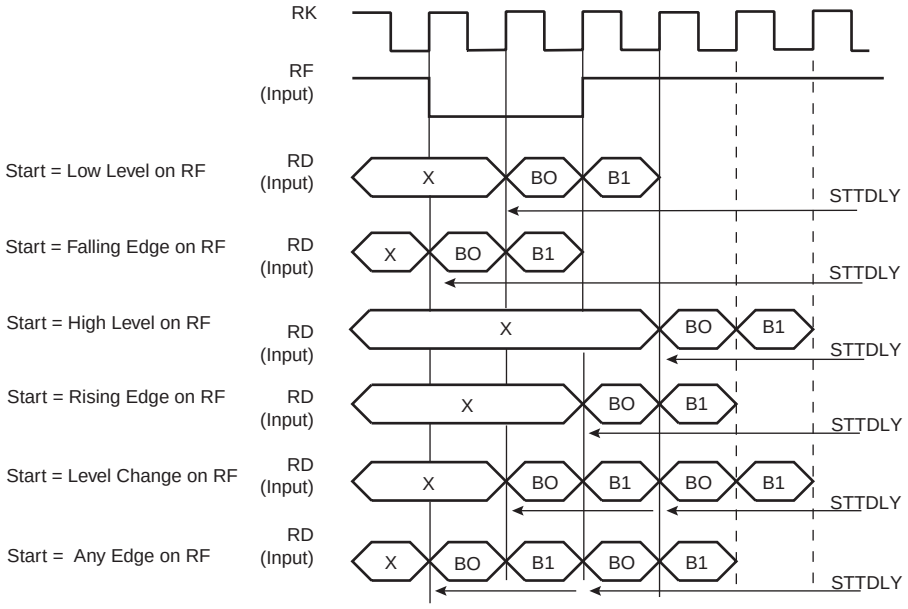


Figure 32-11: Receive Pulse/Edge Start Modes



32.6.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 256 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

32.6.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR and has a maximum value of 16.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

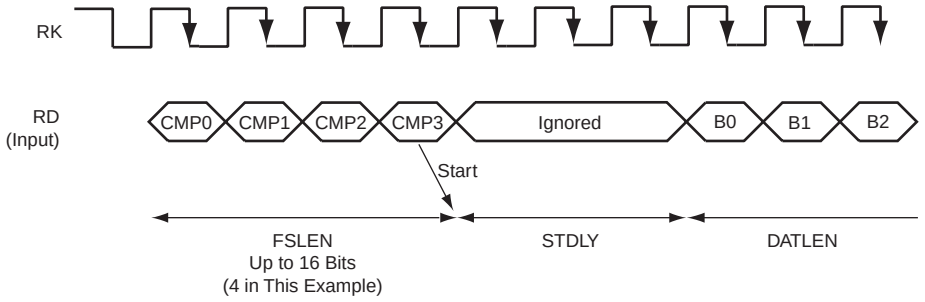
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

32.6.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

32.6.6 Receive Compare Modes

Figure 32-12: Receive Compare Modes



32.6.6.1 Compare Functions

Length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 16 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in SSC_RCMR.

© 2017 Microchip Technology Inc.

32.6.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select:

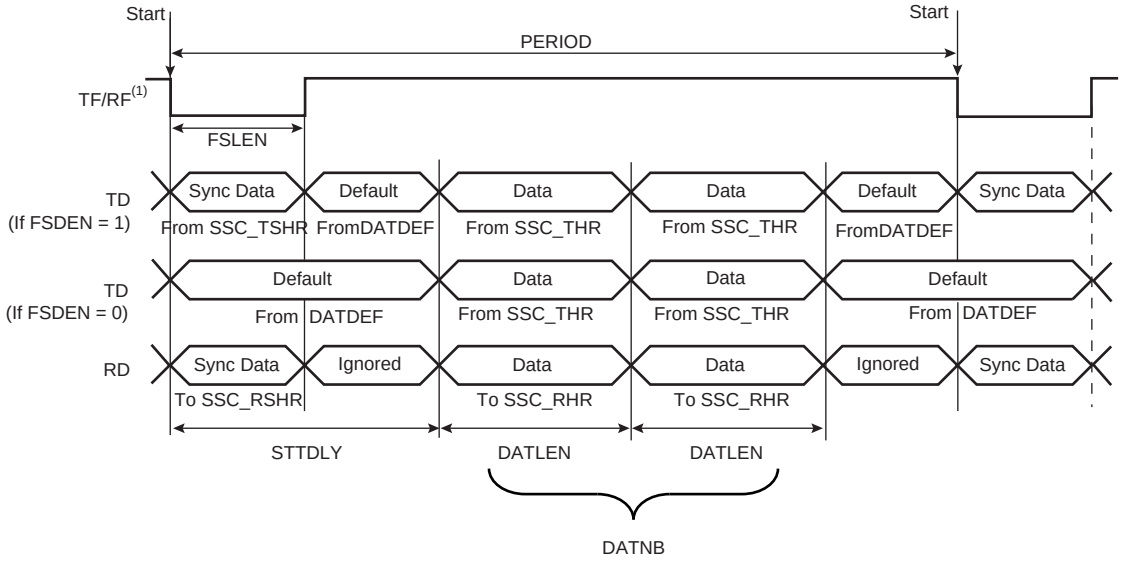
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 32-2: Data Frame Registers

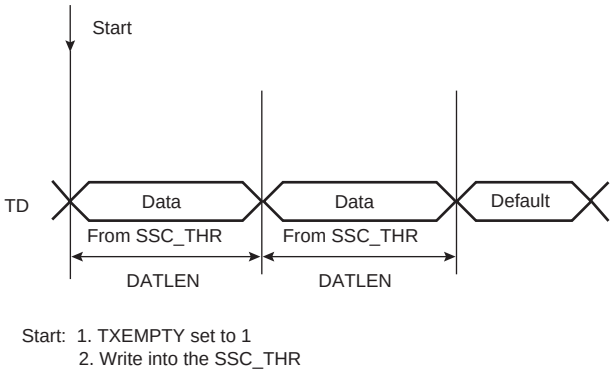
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

Figure 32-13: Transmit and Receive Frame Format in Edge/Pulse Start Modes



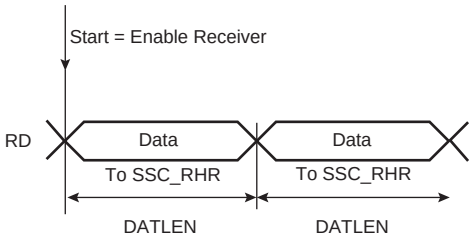
Note 1: Example of input on falling edge of TF/RF.

Figure 32-14: Transmit Frame Format in Continuous Mode



Note 1: STTDLY is set to 0. In this example, SSC_THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 32-15: Receive Frame Format in Continuous Mode



Note 1: STTDLY is set to 0.

32.6.8 Loop Mode

The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

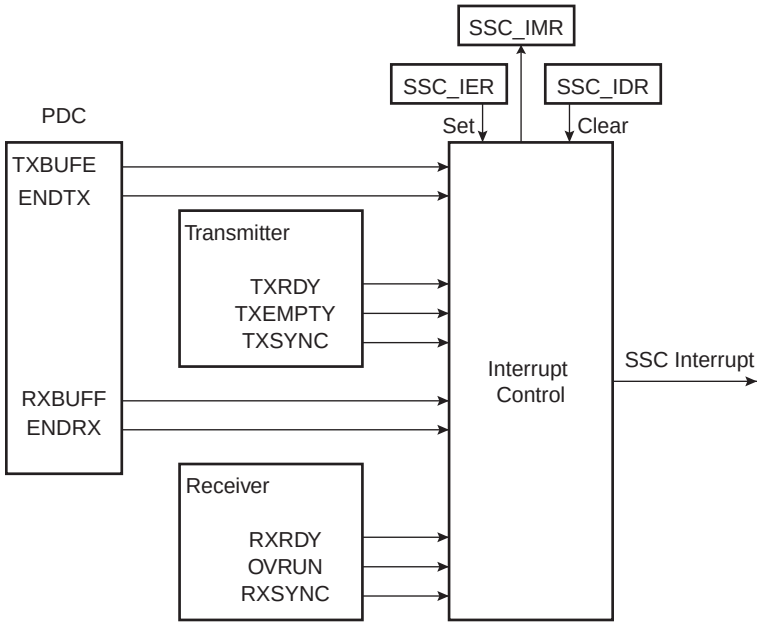
32.6.9 Interrupt

Most bits in SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing SSC_IER (Interrupt Enable Register) and SSC_IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in SSC_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

© 2017 Microchip Technology Inc.

Figure 32-16: Interrupt Block Diagram



32.7 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 32-17: Audio Application Block Diagram

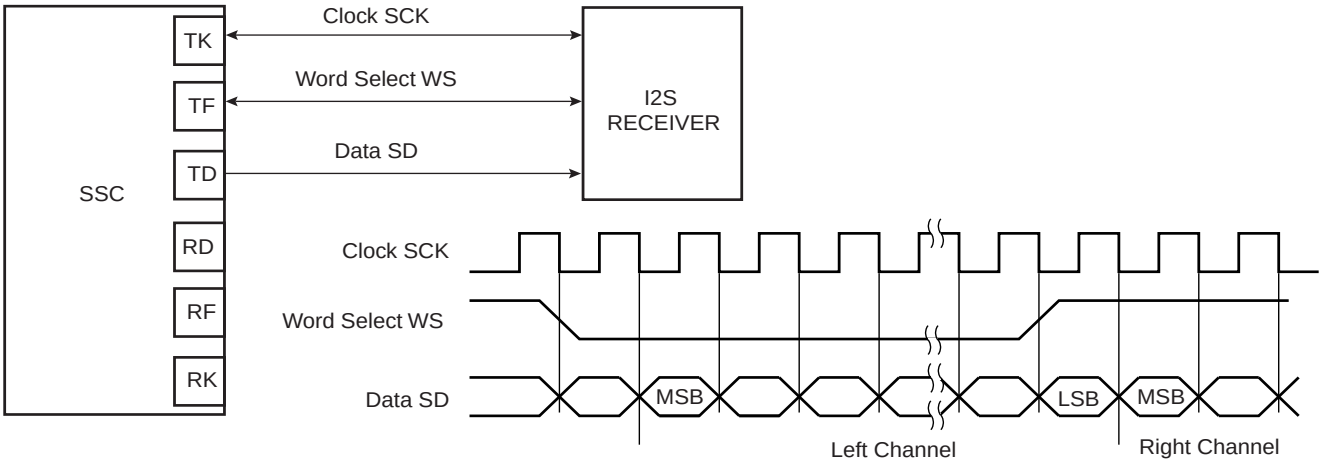


Figure 32-18: Codec Application Block Diagram

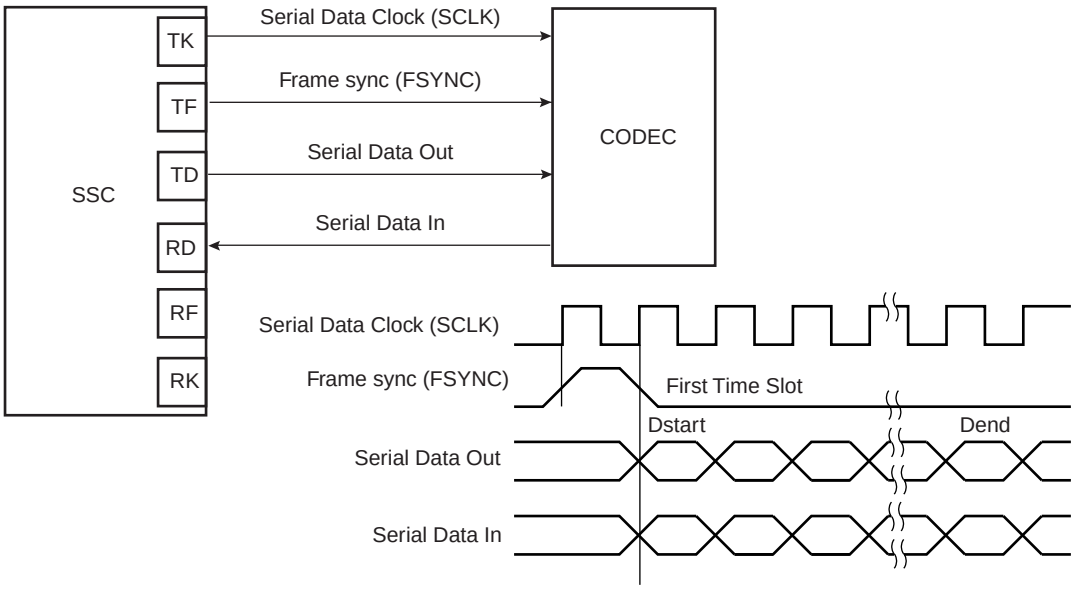
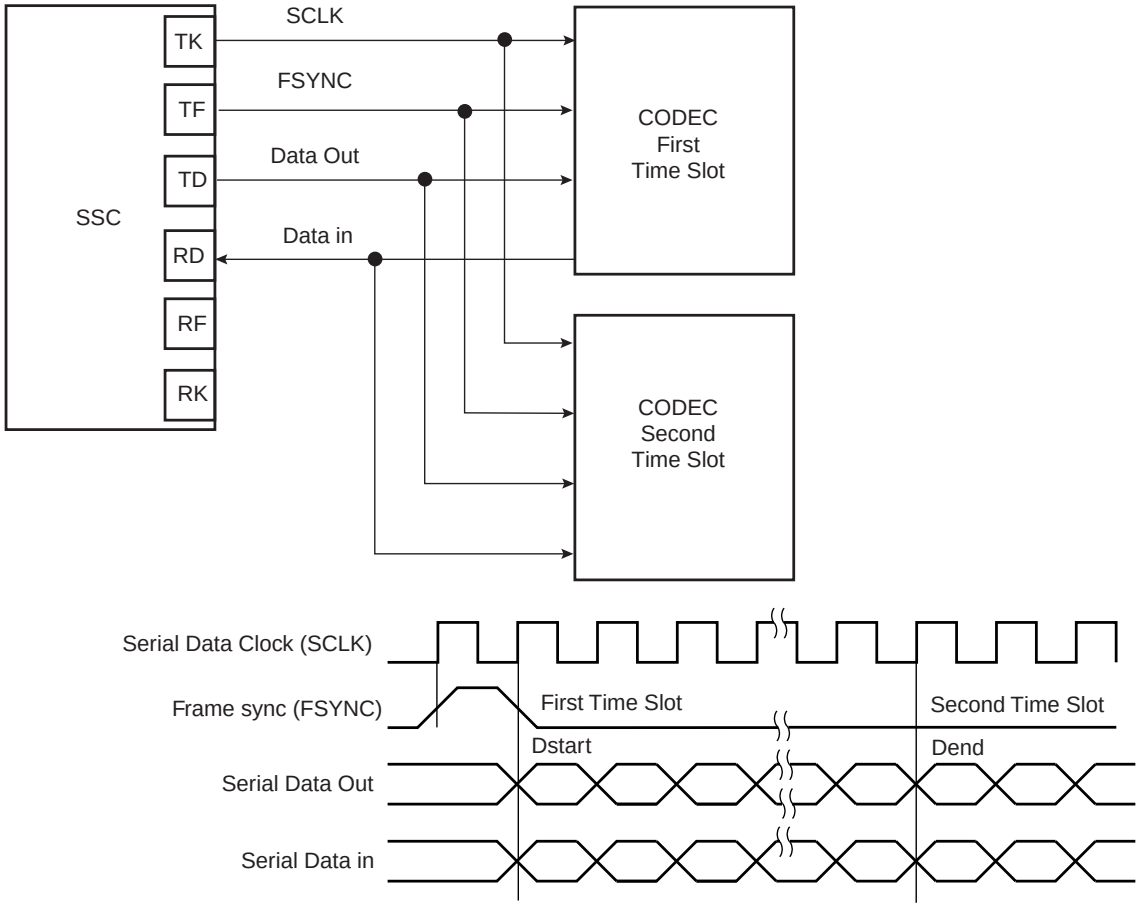


Figure 32-19: Time Slot Application Block Diagram



© 2017 Microchip Technology Inc.

32.8 Syncrhronous Serial Controller (SSC) User Interface

Table 32-3: Register Mapping

Offset	Register	Name	Access	Reset
0x0	Control Register	SSC_CR	Write	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8	Reserved	–	–	–
0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read	0x0
0x24	Transmit Holding Register	SSC_THR	Write	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write	–
0x48	Interrupt Disable Register	SSC_IDR	Write	–
0x4C	Interrupt Mask Register	SSC_IMR	Read	0x0
0x50–0xFC	Reserved	–	–	–
0x100–0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–

© 2017 Microchip Technology Inc.

32.8.1 SSC Control Register

Name:SSC_CR

Access: 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
SWRST	–	–	–	–	–	TXDIS	TXEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXDIS	RXEN

RXEN: Receive Enable

0: No effect.

1: Enables Receive if RXDIS is not set.

RXDIS: Receive Disable

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

TXEN: Transmit Enable

0: No effect.

1: Enables Transmit if TXDIS is not set.

TXDIS: Transmit Disable

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

SWRST: Software Reset

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC_CR.

© 2017 Microchip Technology Inc.

32.8.2 SSC Clock Mode Register

Name:SSC_CMR

Access: 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
–	–	–	–	DIV			
7	6	5	4	3	2	1	0
DIV							

DIV: Clock Divider

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is MCK/2 x 4095 = MCK/8190.

© 2017 Microchip Technology Inc.

32.8.3 SSC Receive Clock Mode Register

Name:SSC_RCMR

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	STOP	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

CKS: Receive Clock Selection

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

CKO: Receive Clock Output Mode Selection

CKO	Receive Clock Output Mode	RK pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3–0x7	Reserved	

CKI: Receive Clock Inversion

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

CKG: Receive Clock Gating Selection

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

© 2017 Microchip Technology Inc.

START: Receive Start Selection

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9–0xF	Reserved

STOP: Receive Stop Selection

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

STTDLY: Receive Start Delay

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

PERIOD: Receive Period Divider Selection

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.

© 2017 Microchip Technology Inc.

32.8.4 SSC Receive Frame Mode Register

Name:SSC_RFMR

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
–	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

DATLEN: Data Length

0: Forbidden value (1-bit data length not supported).
Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

LOOP: Loop Mode

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

MSBF: Most Significant Bit First

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

DATNB: Data Number per Frame

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

FSLEN: Receive Frame Sync Length

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT * 16) + 1 Receive Clock periods.

FSOS: Receive Frame Sync Output Selection

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6–0x7	Reserved	Undefined

© 2017 Microchip Technology Inc.

FSEDGE: Frame Sync Edge Detection

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

FSLEN_EXT: FSLEN Field Extension

Extends FSLEN field. For details, refer to FSLEN bit description above.

32.8.5 SSC Transmit Clock Mode Register

Name:SSC_TCMR

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	–	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

CKS: Transmit Clock Selection

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

CKO: Transmit Clock Output Mode Selection

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3–0x7	Reserved	

CKI: Transmit Clock Inversion

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

CKG: Transmit Clock Gating Selection

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

© 2017 Microchip Technology Inc.

START: Transmit Start Selection

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8–0xF	Reserved

STTDLY: Transmit Start Delay

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

PERIOD: Transmit Period Divider Selection

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 x (PERIOD+1) Transmit Clock.

© 2017 Microchip Technology Inc.

32.8.6 SSC Transmit Frame Mode Register

Name:SSC_TFMR

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

DATLEN: Data Length

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Transmit. If DATLEN is lower or equal to 7, data transfers are bytes, if DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

DATDEF: Data Default Value

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

MSBF: Most Significant Bit First

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

DATNB: Data Number per frame

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB +1).

FSLEN: Transmit Frame Sync Length

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT * 16) + 1 Transmit Clock periods.

© 2017 Microchip Technology Inc.

FSOS: Transmit Frame Sync Output Selection

FSOS	Selected Transmit Frame Sync Signal	TF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6–0x7	Reserved	Undefined

FSDEN: Frame Sync Data Enable

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

FSEDGE: Frame Sync Edge Detection

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

FSLEN_EXT: FSLEN Field Extension

Extends FSLEN field. For details, refer to FSLEN bit description above.

32.8.7 SSC Receive Holding Register

Name:SSC_RHR

Access: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

RDAT: Receive Data

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

© 2017 Microchip Technology Inc.

32.8.8 SSC Transmit Holding Register

Name:SSC_THR

Access: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

TDAT: Transmit Data

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

32.8.9 SSC Receive Synchronization Holding Register

Name:SSC_RSHR

Access: 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
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

RSDAT: Receive Synchronization Data

© 2017 Microchip Technology Inc.

32.8.10 SSC Transmit Synchronization Holding Register

Name:SSC_TSHR

Access: 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
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

TSDAT: Transmit Synchronization Data

32.8.11 SSC Receive Compare 0 Register

Name:SSC_RC0R

Access: 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
CP0							
7	6	5	4	3	2	1	0
CP0							

CP0: Receive Compare Data 0

© 2017 Microchip Technology Inc.

32.8.12 SSC Receive Compare 1 Register

Name:SSC_RC1R

Access: 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
CP1							
7	6	5	4	3	2	1	0
CP1							

CP1: Receive Compare Data 1

32.8.13 SSC Status Register

Name:SSC_SR

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

TXRDY: Transmit Ready

0: Data has been loaded in SSC_THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1: SSC_THR is empty.

TXEMPTY: Transmit Empty

0: Data remains in SSC_THR or is currently transmitted from TSR.

1: Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

ENDTX: End of Transmission

0: The register SSC_TCR has not reached 0 since the last write in SSC_TCR or SSC_TNCR.

1: The register SSC_TCR has reached 0 since the last write in SSC_TCR or SSC_TNCR.

TXBUFE: Transmit Buffer Empty

0: SSC_TCR or SSC_TNCR have a value other than 0.

1: Both SSC_TCR and SSC_TNCR have a value of 0.

RXRDY: Receive Ready

0: SSC_RHR is empty.

1: Data has been received and loaded in SSC_RHR.

OVRUN: Receive Overrun

0: No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

ENDRX: End of Reception

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

RXBUFF: Receive Buffer Full

0: SSC_RCR or SSC_RNCR have a value other than 0.

1: Both SSC_RCR and SSC_RNCR have a value of 0.

CP0: Compare 0

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

© 2017 Microchip Technology Inc.

CP1: Compare 1

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

TXSYN: Transmit Sync

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

RXSYN: Receive Sync

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

TXEN: Transmit Enable

0: Transmit is disabled.

1: Transmit is enabled.

RXEN: Receive Enable

0: Receive is disabled.

1: Receive is enabled.

32.8.14 SSC Interrupt Enable Register

Name:SSC_IER

Access: 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
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Enable

0: No effect.

1: Enables the Transmit Ready Interrupt.

TXEMPTY: Transmit Empty Interrupt Enable

0: No effect.

1: Enables the Transmit Empty Interrupt.

ENDTX: End of Transmission Interrupt Enable

0: No effect.

1: Enables the End of Transmission Interrupt.

TXBUFE: Transmit Buffer Empty Interrupt Enable

0: No effect.

1: Enables the Transmit Buffer Empty Interrupt

RXRDY: Receive Ready Interrupt Enable

0: No effect.

1: Enables the Receive Ready Interrupt.

OVRUN: Receive Overrun Interrupt Enable

0: No effect.

1: Enables the Receive Overrun Interrupt.

ENDRX: End of Reception Interrupt Enable

0: No effect.

1: Enables the End of Reception Interrupt.

RXBUFF: Receive Buffer Full Interrupt Enable

0: No effect.

1: Enables the Receive Buffer Full Interrupt.

CP0: Compare 0 Interrupt Enable

0: No effect.

1: Enables the Compare 0 Interrupt.

© 2017 Microchip Technology Inc.

CP1: Compare 1 Interrupt Enable

0: No effect.

1: Enables the Compare 1 Interrupt.

TXSYN: Tx Sync Interrupt Enable

0: No effect.

1: Enables the Tx Sync Interrupt.

RXSYN: Rx Sync Interrupt Enable

0: No effect.

1: Enables the Rx Sync Interrupt.

32.8.15 SSC Interrupt Disable Register

Name:SSC_IDR

Access: 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
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Disable

0: No effect.

1: Disables the Transmit Ready Interrupt.

TXEMPTY: Transmit Empty Interrupt Disable

0: No effect.

1: Disables the Transmit Empty Interrupt.

ENDTX: End of Transmission Interrupt Disable

0: No effect.

1: Disables the End of Transmission Interrupt.

TXBUFE: Transmit Buffer Empty Interrupt Disable

0: No effect.

1: Disables the Transmit Buffer Empty Interrupt.

RXRDY: Receive Ready Interrupt Disable

0: No effect.

1: Disables the Receive Ready Interrupt.

OVRUN: Receive Overrun Interrupt Disable

0: No effect.

1: Disables the Receive Overrun Interrupt.

ENDRX: End of Reception Interrupt Disable

0: No effect.

1: Disables the End of Reception Interrupt.

RXBUFF: Receive Buffer Full Interrupt Disable

0: No effect.

1: Disables the Receive Buffer Full Interrupt.

CP0: Compare 0 Interrupt Disable

0: No effect.

1: Disables the Compare 0 Interrupt.

© 2017 Microchip Technology Inc.

CP1: Compare 1 Interrupt Disable

0: No effect.

1: Disables the Compare 1 Interrupt.

TXSYN: Tx Sync Interrupt Enable

0: No effect.

1: Disables the Tx Sync Interrupt.

RXSYN: Rx Sync Interrupt Enable

0: No effect.

1: Disables the Rx Sync Interrupt.

32.8.16 SSC Interrupt Mask Register

Name:SSC_IMR

Access: 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
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

TXRDY: Transmit Ready Interrupt Mask

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

TXEMPTY: Transmit Empty Interrupt Mask

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

ENDTX: End of Transmission Interrupt Mask

0: The End of Transmission Interrupt is disabled.

1: The End of Transmission Interrupt is enabled.

TXBUFE: Transmit Buffer Empty Interrupt Mask

0: The Transmit Buffer Empty Interrupt is disabled.

1: The Transmit Buffer Empty Interrupt is enabled.

RXRDY: Receive Ready Interrupt Mask

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

OVRUN: Receive Overrun Interrupt Mask

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

ENDRX: End of Reception Interrupt Mask

0: The End of Reception Interrupt is disabled.

1: The End of Reception Interrupt is enabled.

RXBUFF: Receive Buffer Full Interrupt Mask

0: The Receive Buffer Full Interrupt is disabled.

1: The Receive Buffer Full Interrupt is enabled.

CP0: Compare 0 Interrupt Mask

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

© 2017 Microchip Technology Inc.

CP1: Compare 1 Interrupt Mask

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

TXSYN: Tx Sync Interrupt Mask

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

RXSYN: Rx Sync Interrupt Mask

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

33. Timer Counter (TC)

33.1 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 33-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2.

Table 33-1: Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

© 2017 Microchip Technology Inc.

33.2 Block Diagram

Figure 33-1: Timer Counter Block Diagram

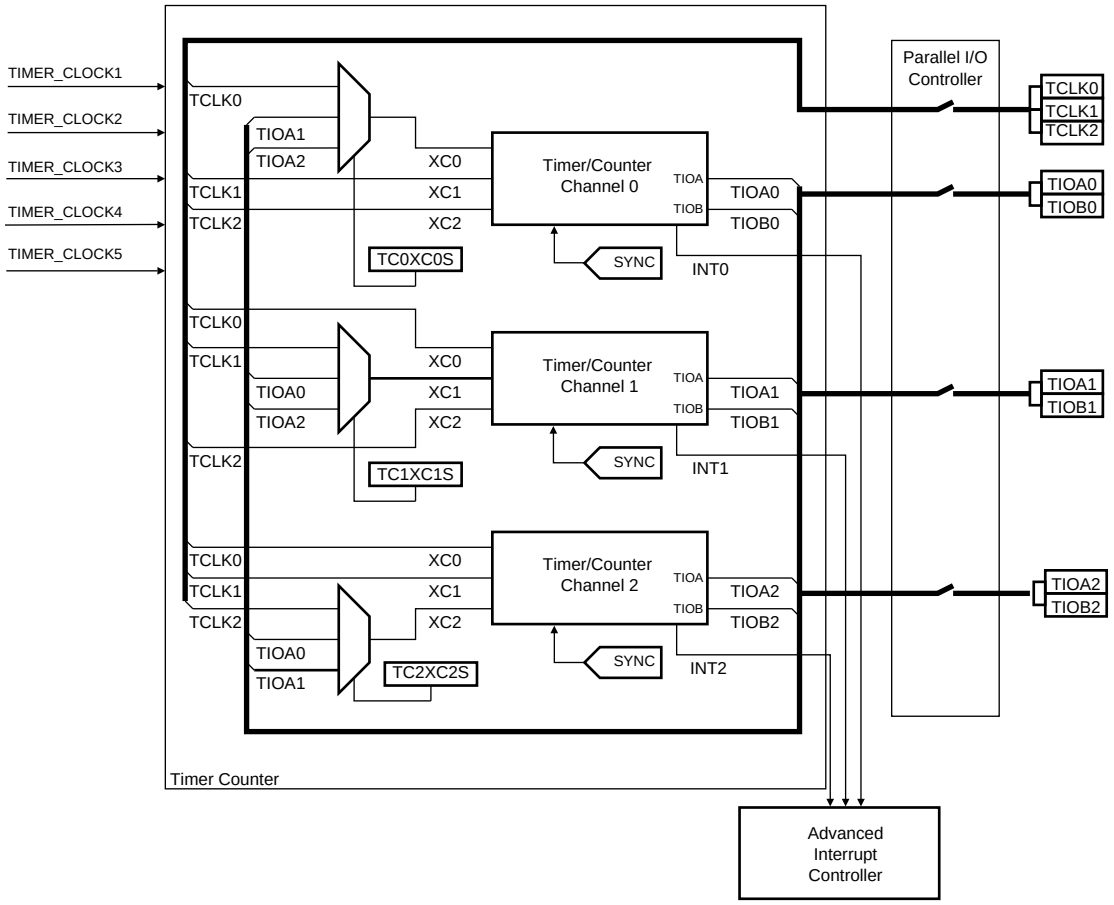


Table 33-2: Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

33.3 Pin Name List

Table 33-3: 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

33.4 Product Dependencies

33.4.1 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.

33.4.2 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.

33.4.3 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.

33.5 Functional Description

33.5.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 33-4.

33.5.2 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.

33.5.3 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 33-2.

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 33-3.

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

© 2017 Microchip Technology Inc.

Figure 33-2: Clock Chaining Selection

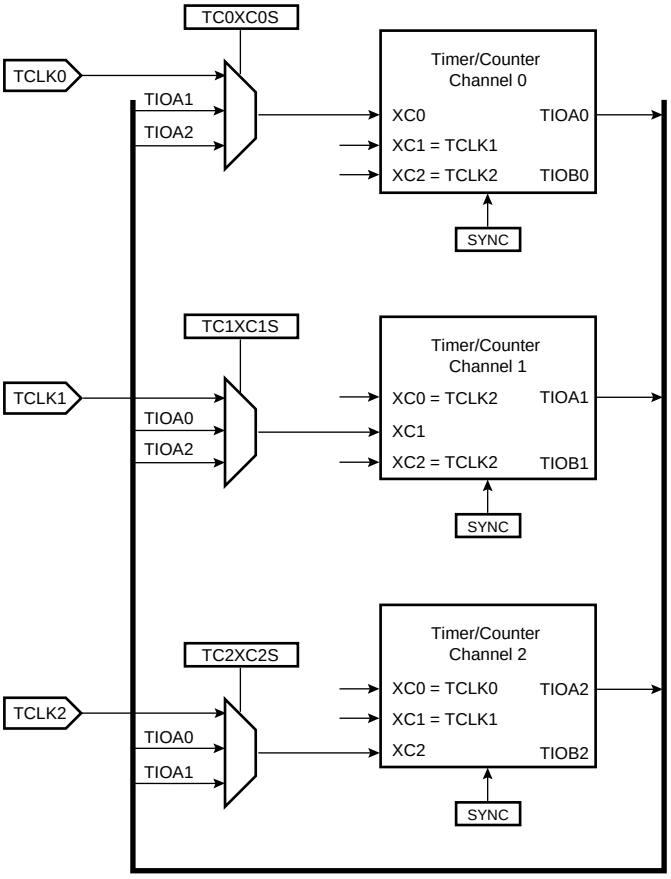
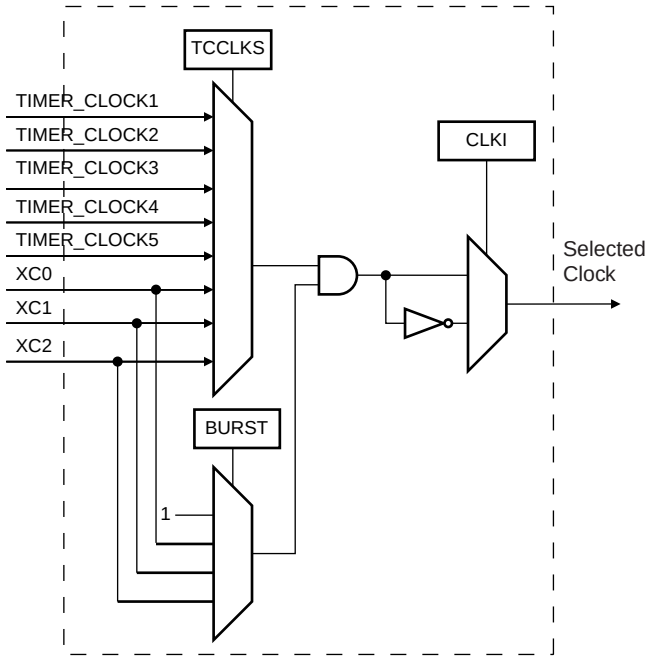


Figure 33-3: Clock Selection

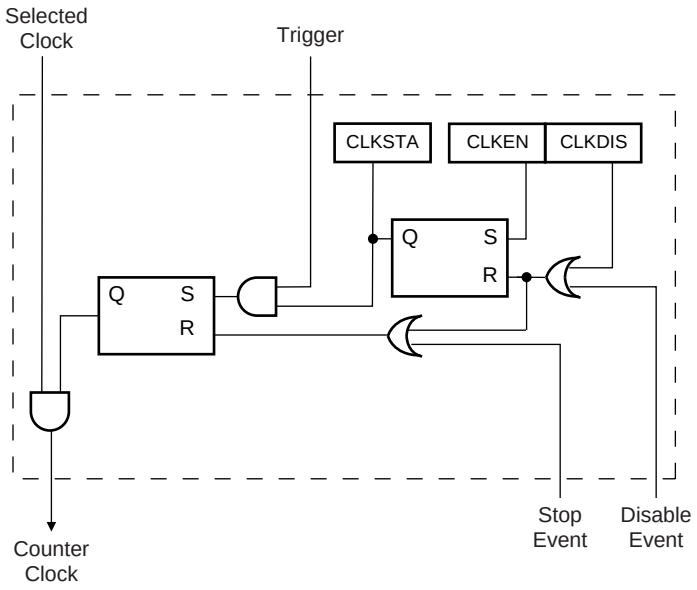


33.5.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 33-4.

- 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 33-4: Clock Control



33.5.5 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.

33.5.6 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.

© 2017 Microchip Technology Inc.

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.

33.5.7 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 33-5 shows the configuration of the TC channel when programmed in Capture Mode.

33.5.8 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.

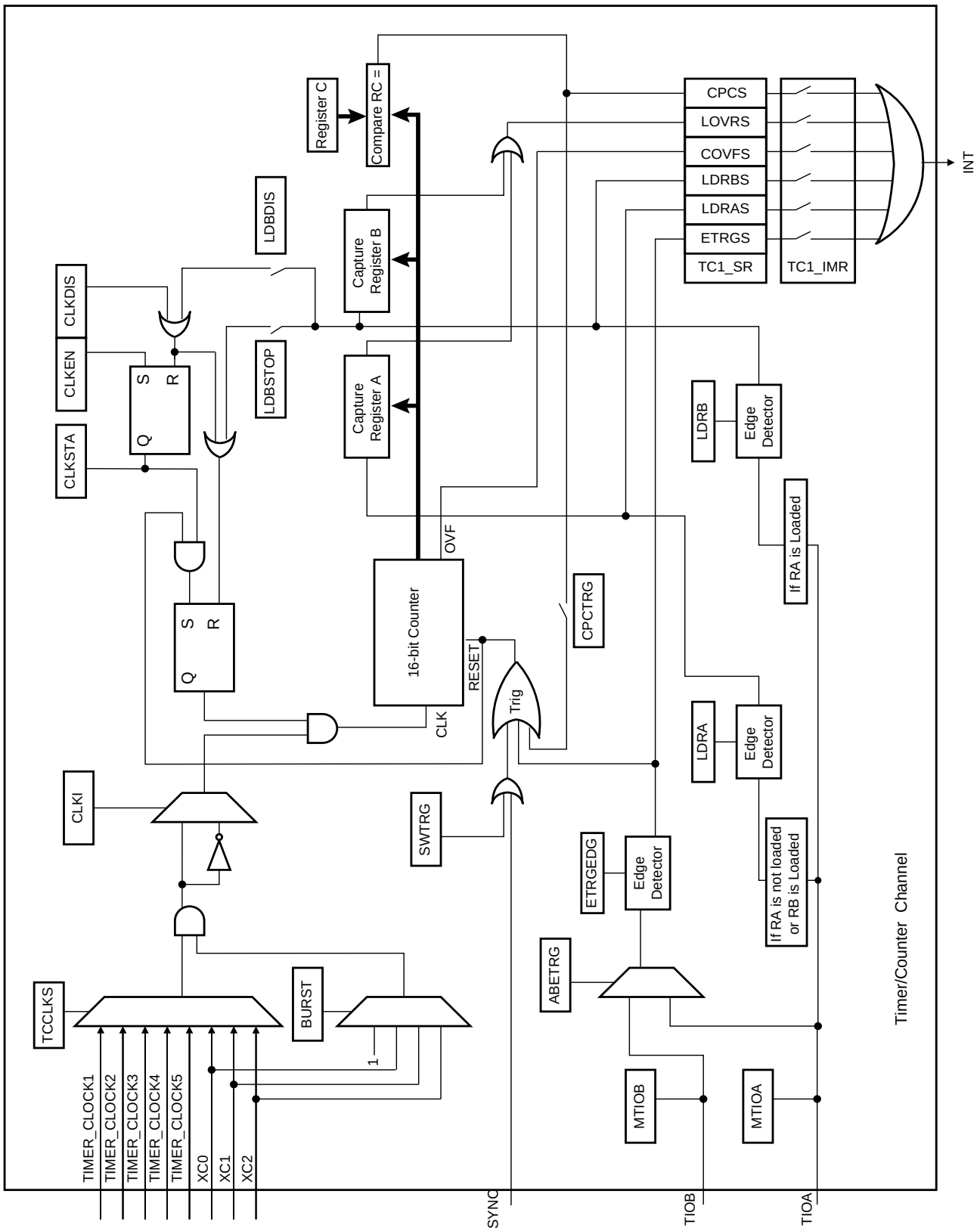
33.5.9 Trigger Conditions

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.

© 2017 Microchip Technology Inc.

Figure 33-5: Capture Mode



© 2017 Microchip Technology Inc.

33.5.10 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 33-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

33.5.11 Waveform Selection

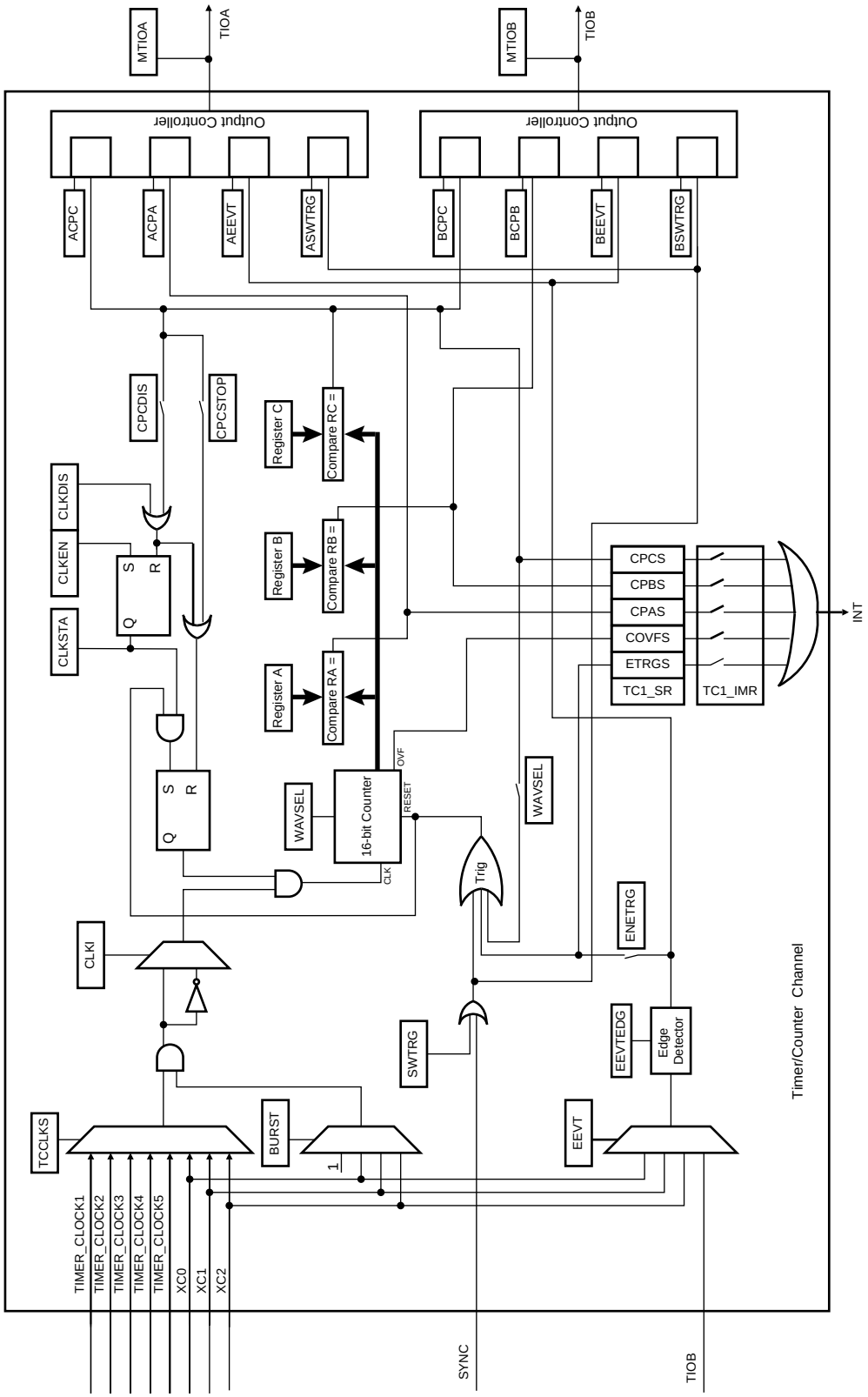
Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

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.

© 2017 Microchip Technology Inc.

Figure 33-6: Waveform Mode



© 2017 Microchip Technology Inc.

33.5.11.1 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 33-7.

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 33-8.

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 33-7: WAVSEL = 00 without trigger

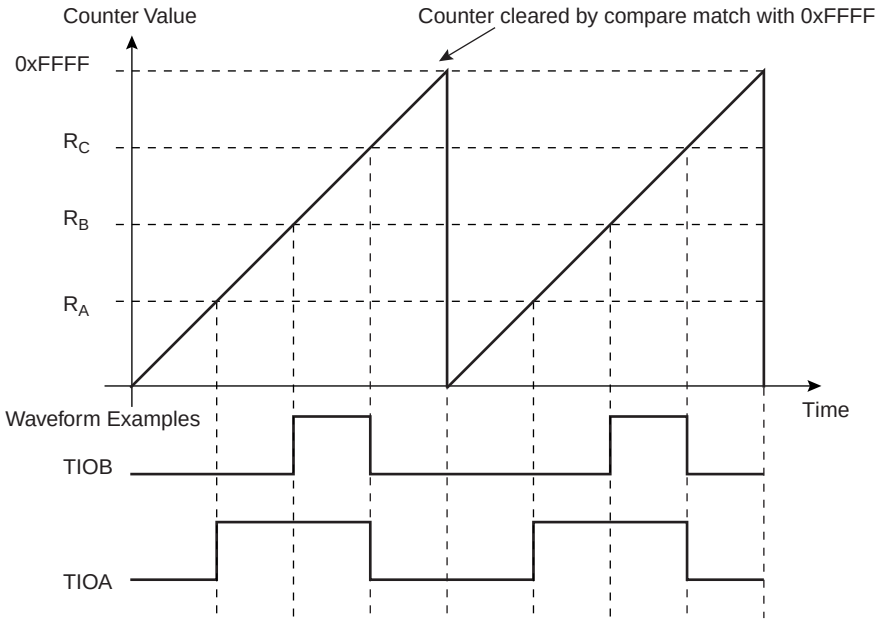
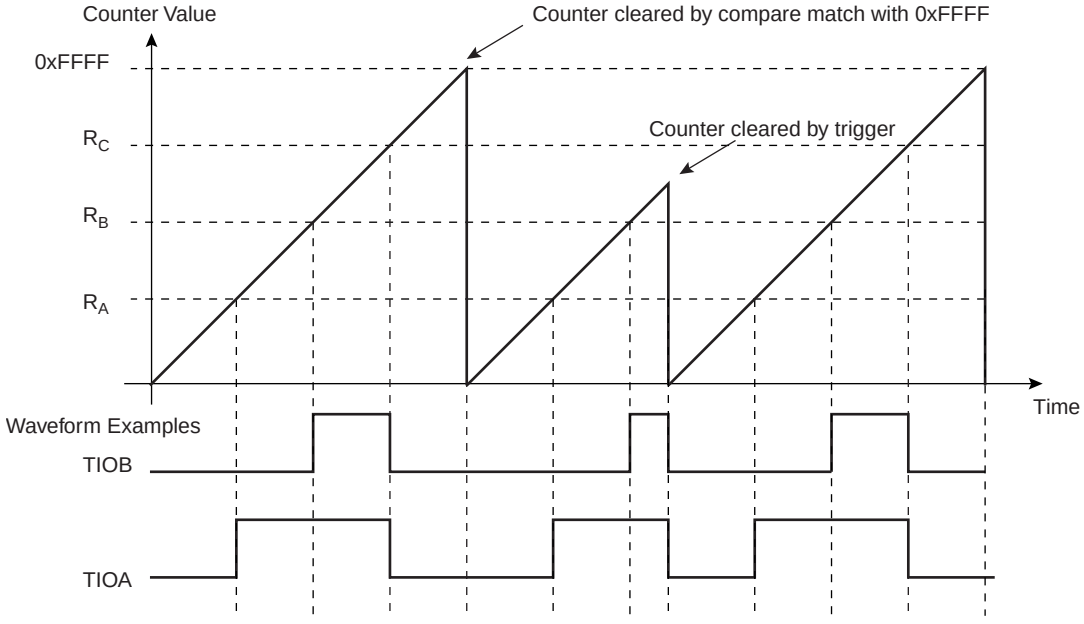


Figure 33-8: WAVSEL = 00 with trigger



33.5.11.2 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 33-9.

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 33-10.

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 33-9: WAVSEL = 10 Without Trigger

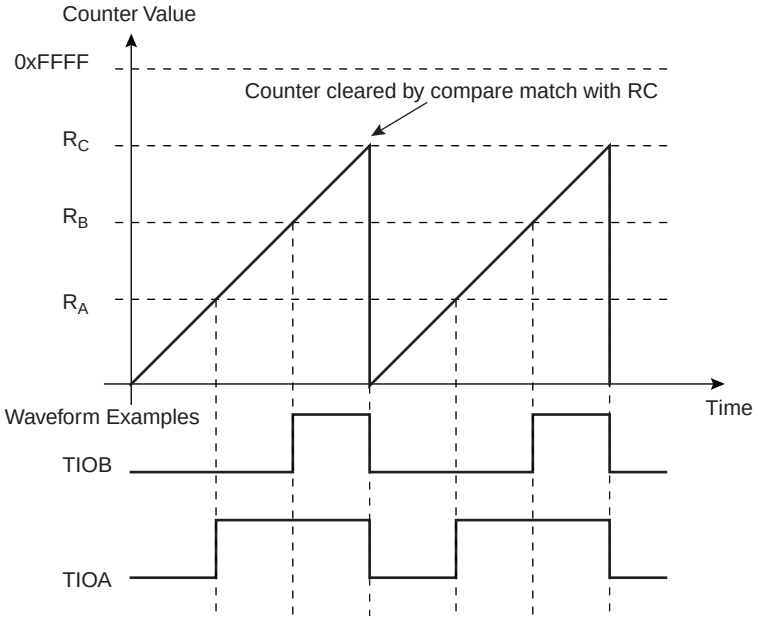
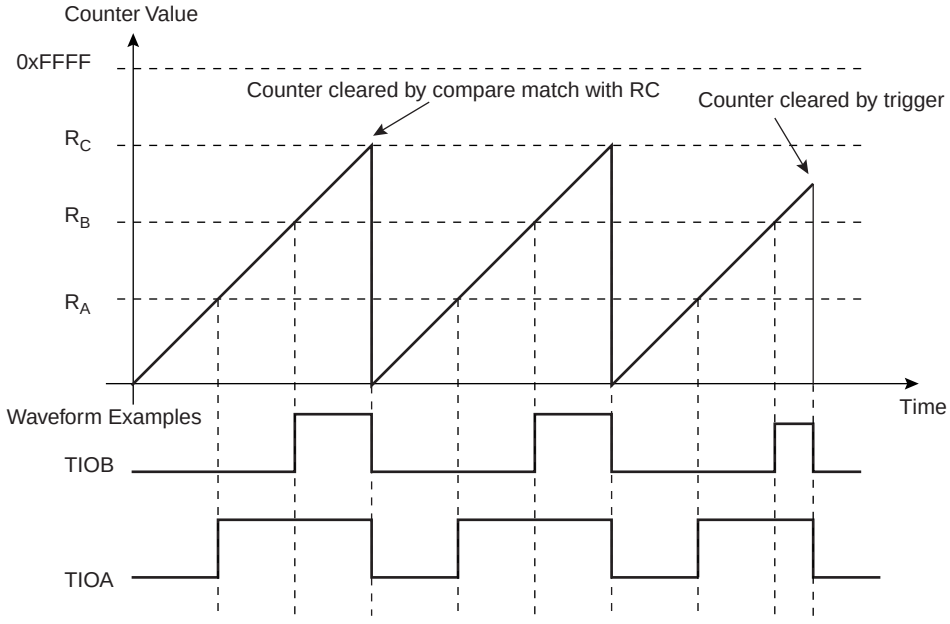


Figure 33-10: WAVSEL = 10 With Trigger



© 2017 Microchip Technology Inc.

33.5.11.3 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 33-11.

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 33-12.

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 33-11: WAVSEL = 01 Without Trigger

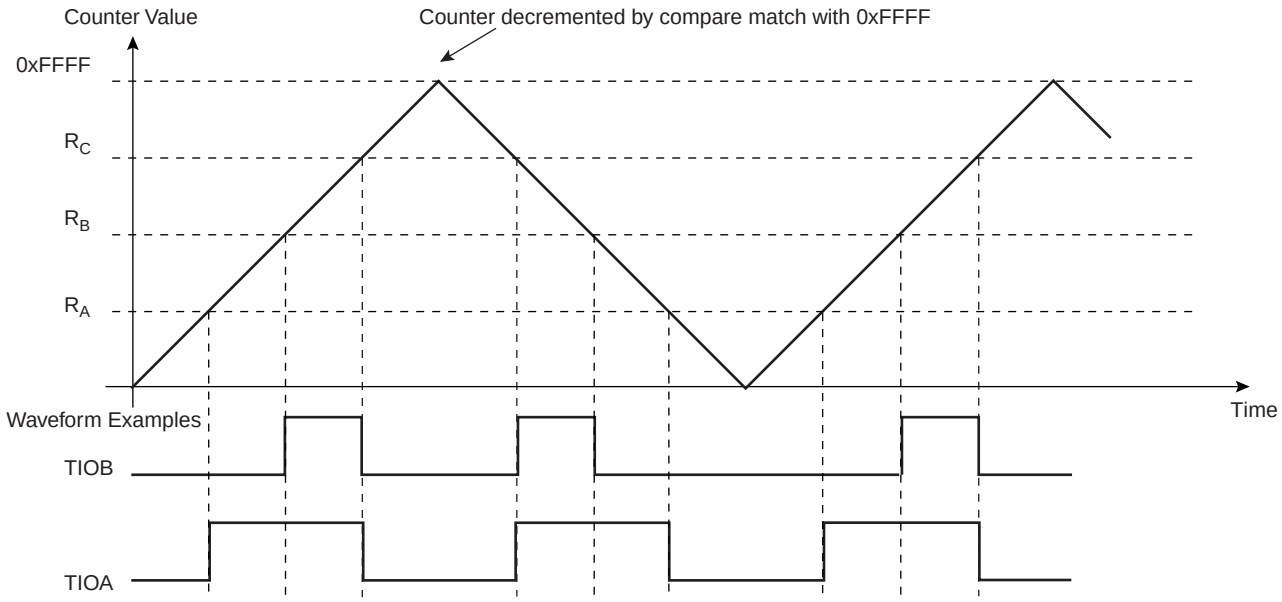
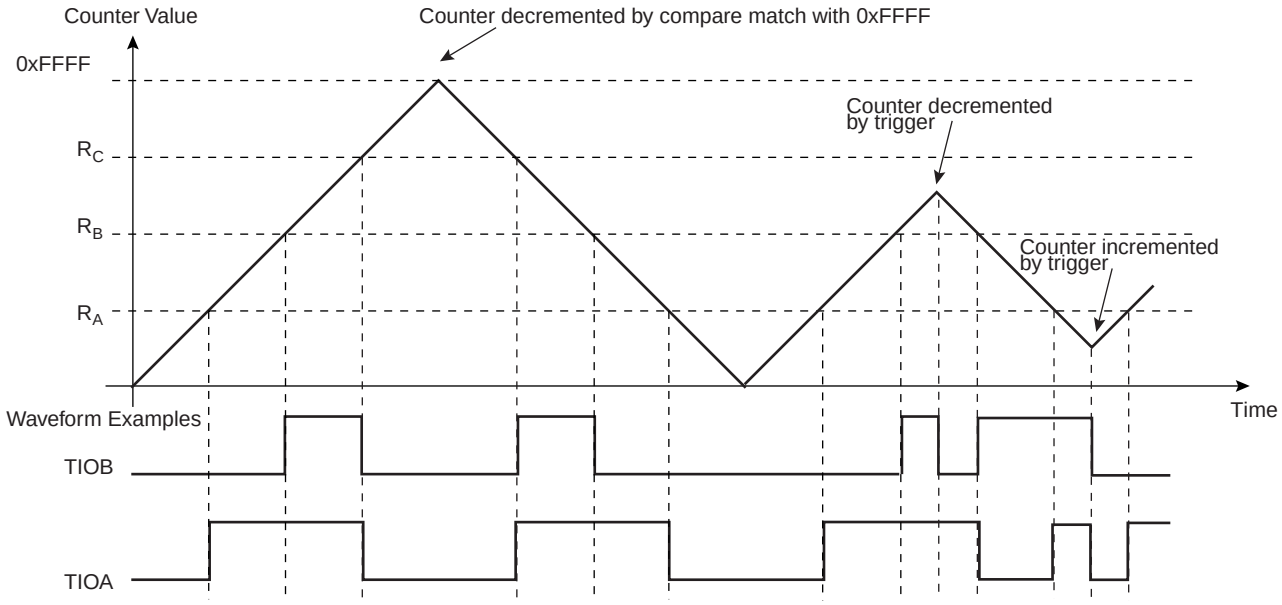


Figure 33-12: WAVSEL = 01 With Trigger



33.5.11.4 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 33-13.

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 33-14.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 33-13: WAVSEL = 11 Without Trigger

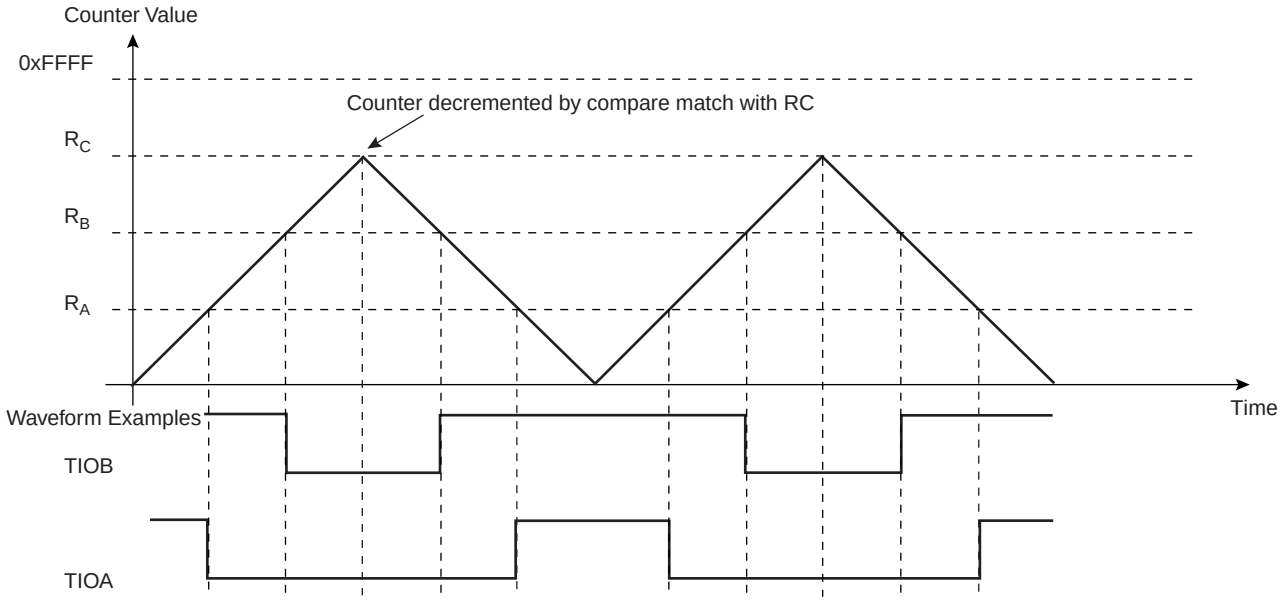
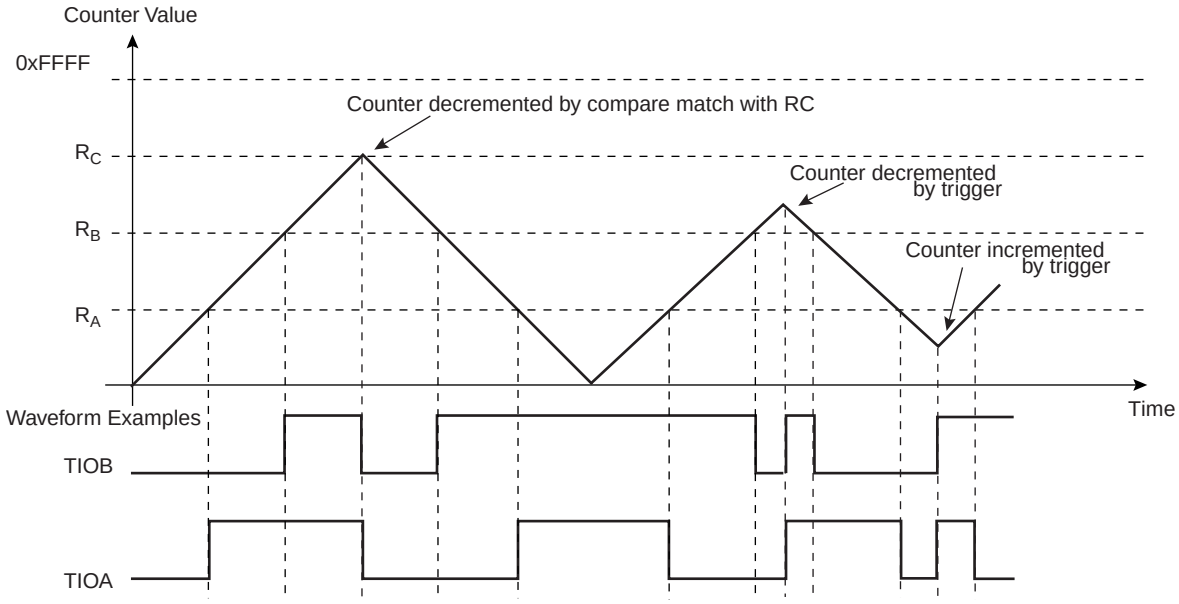


Figure 33-14: WAVSEL = 11 With Trigger



© 2017 Microchip Technology Inc.

33.5.12 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.

33.5.13 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.

© 2017 Microchip Technology Inc.

33.6 Timer Counter (TC) User Interface

Table 33-4: 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	–	–	–

Note 1: Channel index ranges from 0 to 2.

2: Read-only if WAVE = 0

© 2017 Microchip Technology Inc.

33.6.1 TC Block Control Register

Name:TC_BCR

Access: 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.

33.6.2 TC Block Mode Register

Name:TC_BMR

Access: 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

© 2017 Microchip Technology Inc.

33.6.3 TC Channel Control Register

Name:TC_CCRx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

33.6.4 TC Channel Mode Register: Capture Mode

Name:TC_CMRx [x=0..2] (WAVE = 0)

Access: 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.

© 2017 Microchip Technology Inc.

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

33.6.5 TC Channel Mode Register: Waveform Mode

Name:TC_CMRx [x=0..2] (WAVE = 1)

Access: 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.

© 2017 Microchip Technology Inc.

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

© 2017 Microchip Technology Inc.

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

© 2017 Microchip Technology Inc.

33.6.6 TC Counter Value Register

Name:TC_CVx [x=0..2]

Access: 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.

33.6.7 TC Register A

Name:TC_RAx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

33.6.8 TC Register B

Name:TC_RBx [x=0..2]

Access: 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.

33.6.9 TC Register C

Name:TC_RCx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

33.6.10 TC Status Register

Name:TC_SRx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

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.

© 2017 Microchip Technology Inc.

33.6.11 TC Interrupt Enable Register

Name:TC_IERx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

33.6.12 TC Interrupt Disable Register

Name:TC_IDRx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

33.6.13 TC Interrupt Mask Register

Name:TC_IMRx [x=0..2]

Access: 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.

© 2017 Microchip Technology Inc.

34. MultiMedia Card Interface (MCI)

34.1 Overview

The MultiMedia Card Interface (MCI) supports the MultiMedia Card (MMC) Specification V3.11, the SDIO Specification V1.1 and the SD Memory Card Specification V1.0.

The MCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The MCI supports stream, block and multi-block data read and write, and is compatible with the Peripheral DMA Controller (PDC) channels, minimizing processor intervention for large buffer transfers.

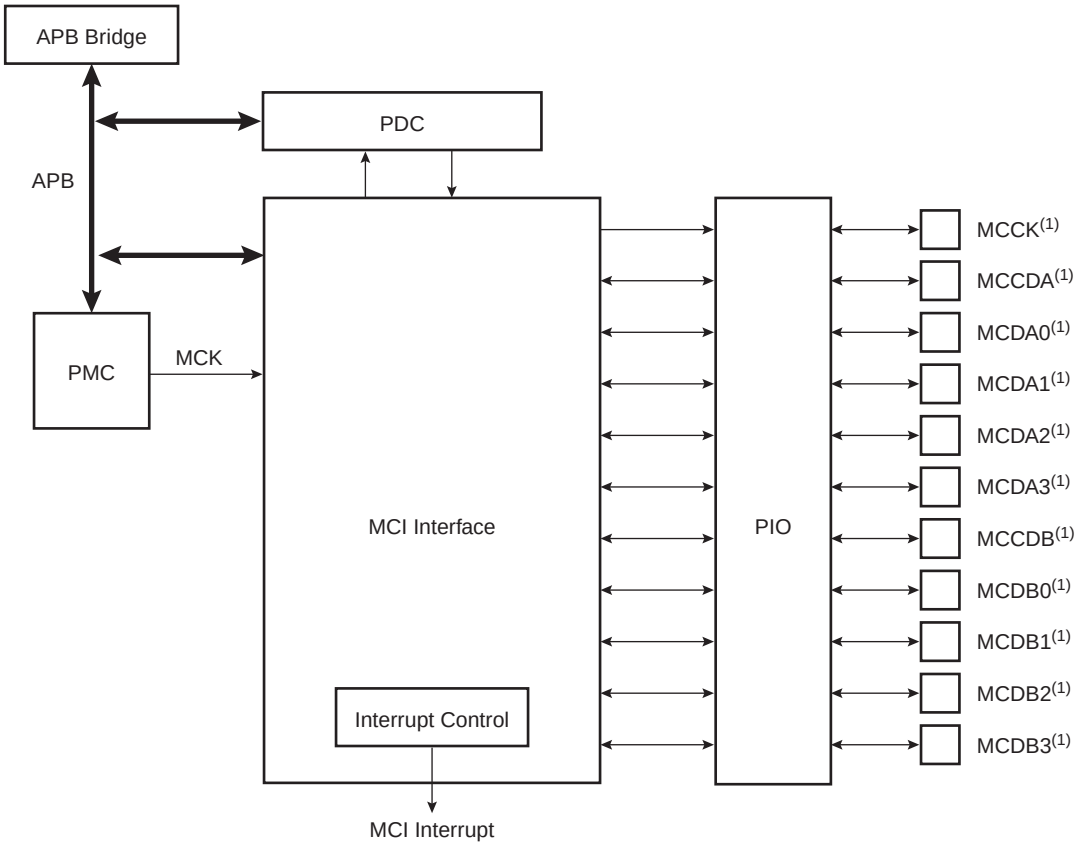
The MCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 2 slot(s). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

34.2 Block Diagram

Figure 34-1: Block Diagram

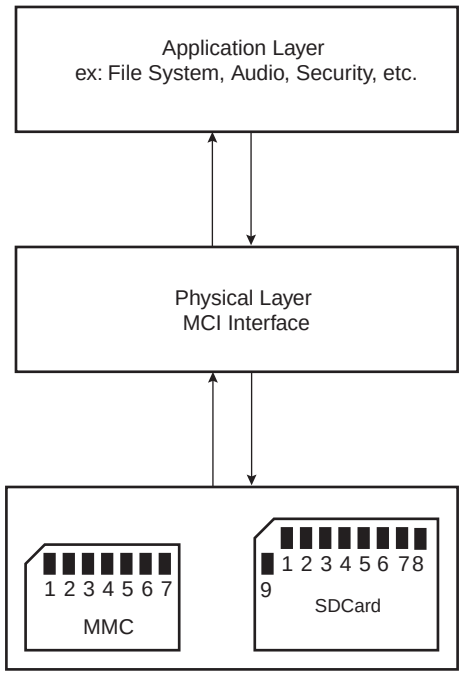


Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB,MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

© 2017 Microchip Technology Inc.

34.3 Application Block Diagram

Figure 34-2: Application Block Diagram



34.4 Pin Name List

Table 34-1: I/O Lines Description

Pin Name[2]	Pin Description	Type[1]	Comments
MCCDA/MCCDB	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0–MCDA3	Data 0..3 of Slot A	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO
MCDB0–MCDB3	Data 0..3 of Slot B	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO

Note 1: I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

2: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

34.5 Product Dependencies

34.5.1 I/O Lines

The pins used for interfacing the MultiMedia Cards or SD Cards may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to MCI pins.

34.5.2 Power Management

The MCI may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the MCI clock.

© 2017 Microchip Technology Inc.

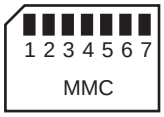
34.5.3 Interrupt

The MCI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the MCI interrupt requires programming the AIC before configuring the MCI.

34.6 Bus Topology

Figure 34-3: Multimedia Memory Card Bus Topology



The MultiMedia Card communication is based on a 7-pin serial bus interface. It has three communication lines and four supply lines.

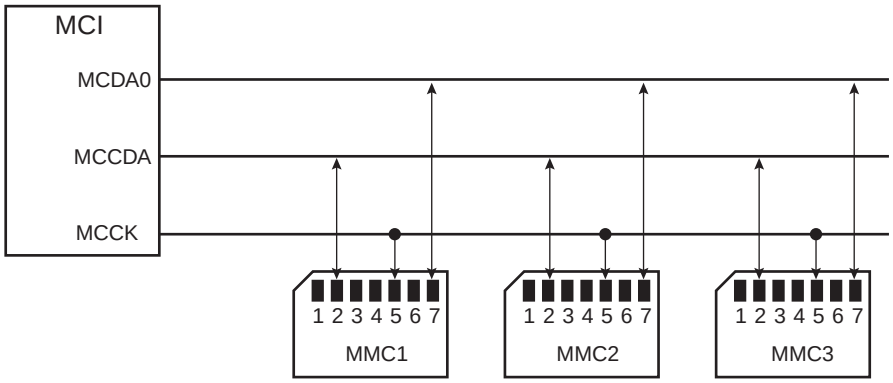
Table 34-2: Bus Topology

Pin Number	Name	Type[1]	Description	MCI Pin Name[2] (Slot z)
1	RSV	NC	Not connected	-
2	CMD	I/O/PP/OD	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0

Note 1: I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

2: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

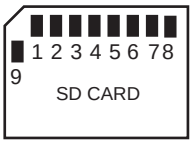
Figure 34-4: MMC Bus Connections (One Slot)



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA MCDAy to MCIx_DAy.

© 2017 Microchip Technology Inc.

Figure 34-5: SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 34-3.

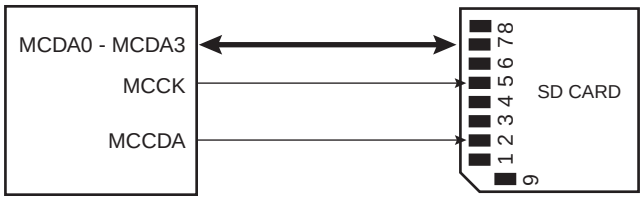
Table 34-3: SD Memory Card Bus Signals

Pin Number	Name	Type[1]	Description	MCI Pin Name[2] (Slot z)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	MCDz3
2	CMD	PP	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	MCDz0
8	DAT[1]	I/O/PP	Data line Bit 1 or Interrupt	MCDz1
9	DAT[2]	I/O/PP	Data line Bit 2	MCDz2

Note 1: I: input, O: output, PP: Push Pull, OD: Open Drain.

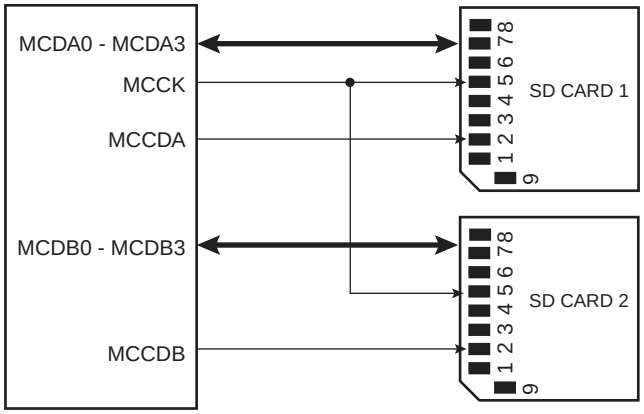
2: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

Figure 34-6: SD Card Bus Connections with One Slot



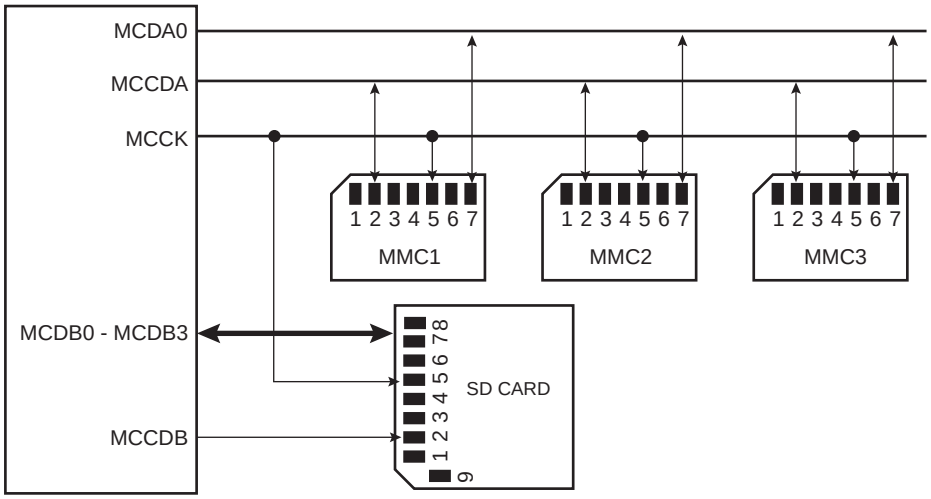
Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA MCDAy to MCIx_DAy.

Figure 34-7: SD Card Bus Connections with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK,MCCDA to MCIx_CDA, MCDAy to MCIx_DAy, MCCDB to MCIx_CDB, MCDBy to MCIx_DBy.

Figure 34-8: Mixing MultiMedia and SD Memory Cards with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCDAy to MCIx_DAy, MCCDB to MCIx_CDB, MCDBy to MCIx_DBy.

When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the MCI_SDCR. Clearing the SDCBUS bit in this register means that the width is one bit; setting it means that the width is four bits. In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

© 2017 Microchip Technology Inc.

34.7 MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command: A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- Response: A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- Data: Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the MultiMedia-Card System Specification. See also Table 34-4.

MultiMediaCard bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock MCI Clock.

Two types of data transfer commands are defined:

- Sequential commands: These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read or when a multiple block transmission has a pre-defined block count (see Section 34.7.2 Data Transfer Operation).

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

34.7.1 Command - Response Operation

After reset, the MCI is disabled and becomes valid after setting the MCIEN bit in the MCI_CR Control Register.

The PWSEN bit saves power by dividing the MCI clock by $2^{PWSDIV} + 1$ when the bus is inactive.

The two bits, RDPROOF and WRPROOF in the MCI Mode Register (MCI_MR) allow stopping the MCI Clock during read or write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

The command and the response of the card are clocked out with the rising edge of the MCI Clock.

All the timings for MultiMedia Card are defined in the MultiMediaCard System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the MCI command register. The MCI_CMDR allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					**N_{ID} Cycles**					**CID**			
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	Content	Z	Z	Z

The command ALL_SEND_CID and the fields and values for the MCI_CMDR Control Register are described in Table 34-4 and Table 34-5.

Table 34-4: ALL_SEND_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: bcr means broadcast command with response.

Table 34-5: Fields and Values for MCI_CMDR Command Register

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The MCI_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

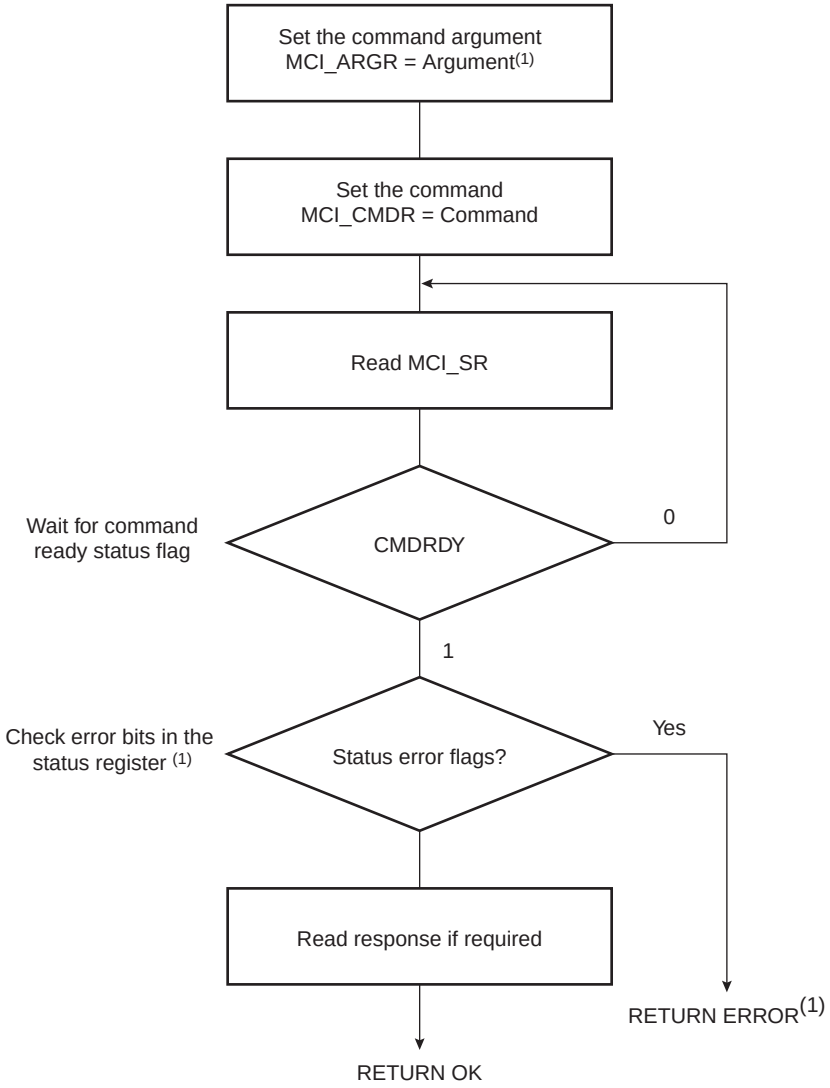
- Fill the argument register (MCI_ARGR) with the command argument.
- Set the command register (MCI_CMDR) (see Table 34-5).

The command is sent immediately after writing the command register. The status bit CMDRDY in the status register (MCI_SR) is asserted when the command is completed. If the command requires a response, it can be read in the MCI response register (MCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the interrupt enable register (MCI_IER) allows using an interrupt method.

© 2017 Microchip Technology Inc.

Figure 34-9: Command/Response Functional Flow Diagram



Note 1: If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the MultiMedia Card specification).

34.7.2 Data Transfer Operation

The MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kind of transfers can be selected setting the Transfer Type (TRTYP) field in the MCI Command Register (MCI_CMDR).

These operations can be done using the features of the Peripheral DMA Controller (PDC). If the PDCMODE bit is set in MCI_MR, then all reads and writes use the PDC facilities.

In all cases, the block length (BLKLEN field) must be defined either in the mode register MCI_MR, or in the Block Register MCI_BLKR. This field determines the size of the data block.

Enabling PDC Force Byte Transfer (PDCFBYTE bit in the MCI_MR) allows the PDC to manage with internal byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported. When PDC Force Byte Transfer is disabled, the PDC type of transfers are in words, otherwise the type of transfers are in bytes.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

- Open-ended/Infinite Multiple block read (or write):

The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.

- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

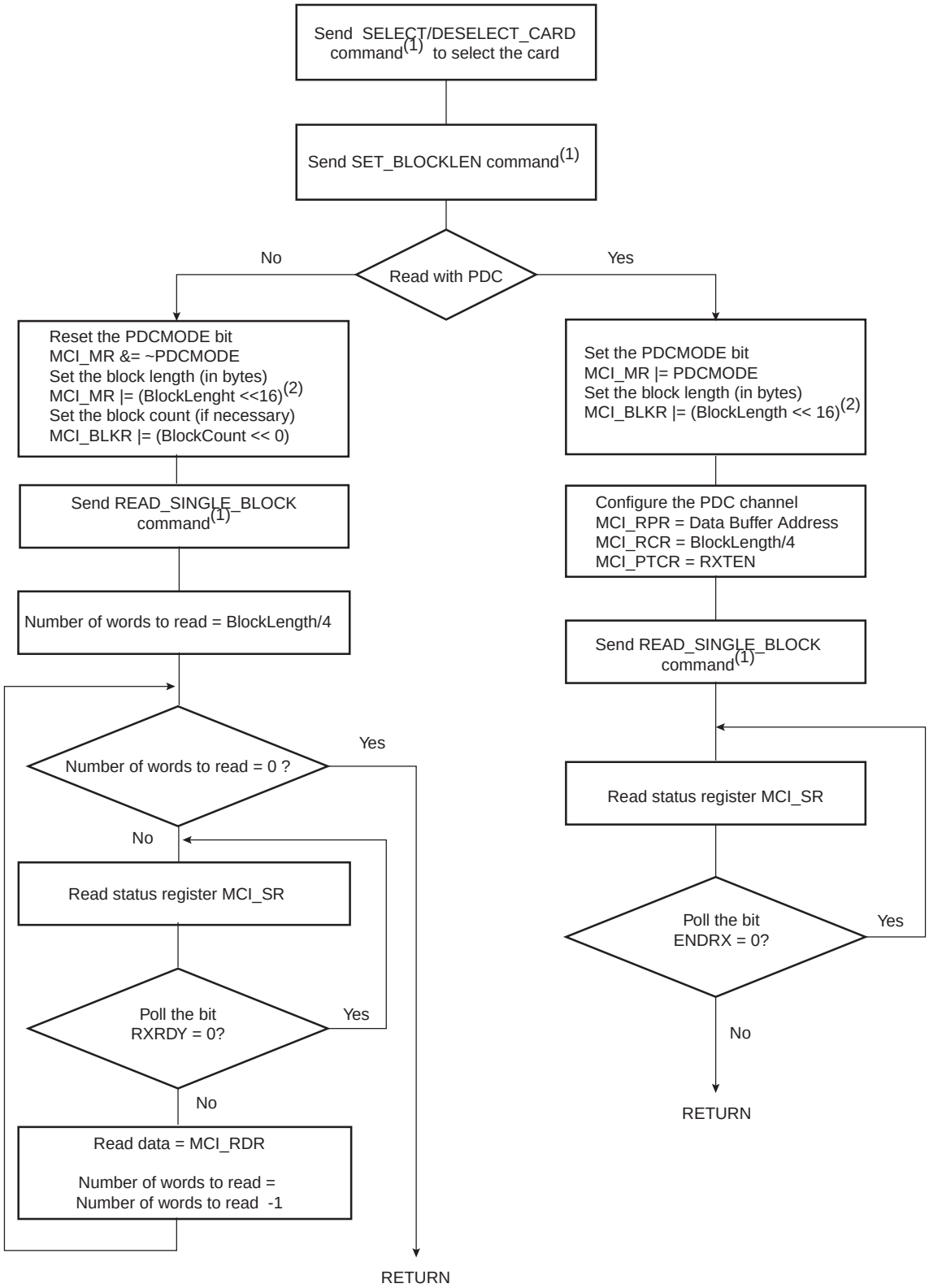
  The card will transfer (or program) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with pre-defined block count, the host must correctly program the MCI Block Register (MCI_BLKR). Otherwise the card will start an open-ended multiple block read. The BCNT field of the Block Register defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

34.7.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see Figure 34-10), a polling method is used to wait for the end of read. Similarly, the user can configure the interrupt enable register (MCI_IER) to trigger an interrupt at the end of read.

© 2017 Microchip Technology Inc.

Figure 34-10: Read Functional Flow Diagram



Note 1: It is assumed that this command has been correctly sent (see Figure 34-9).

2: This field is also accessible in the MCI Block Register (MCI_BLKR).

34.7.4 Write Operation

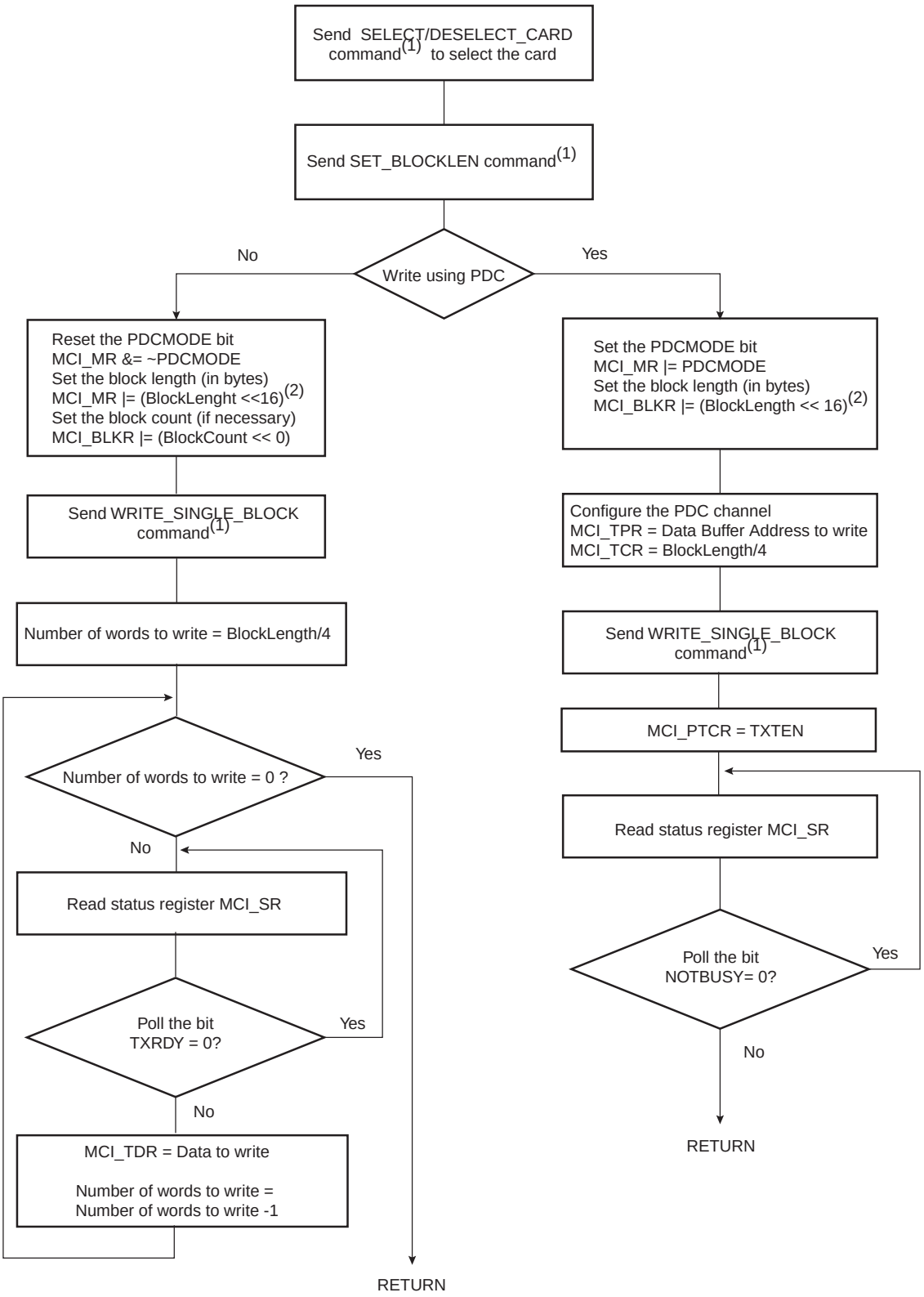
In write operation, the MCI Mode Register (MCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PDCPADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit PDCMODE enables PDC transfer.

The following flowchart shows how to write a single block with or without use of PDC facilities (see Figure 34-11). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI_IMR).

© 2017 Microchip Technology Inc.

Figure 34-11: Write Functional Flow Diagram

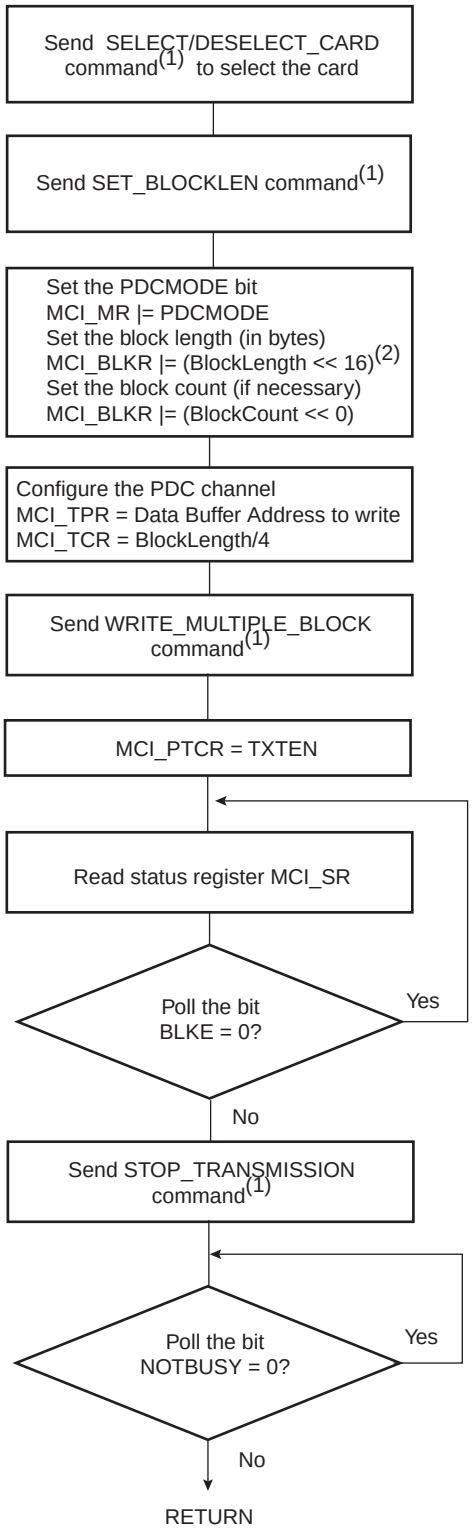


Note 1: It is assumed that this command has been correctly sent (see Figure 34-9).

2: This field is also accessible in the MCI Block Register (MCI_BLKR).

The flowchart in Figure 34-12 shows how to manage a multiple write block transfer with the PDC. Polling or interrupt method can be used to wait for the end of write according to the contents of the MCI_IMR.

Figure 34-12: Multiple Write Functional Flow Diagram



Note 1: It is assumed that this command has been correctly sent (see Figure 34-9).

2: This field is also accessible in the MCI Block Register (MCI_BLKR).

© 2017 Microchip Technology Inc.

34.8 SD/SDIO Card Operations

The MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the Multi Media Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the MultiMedia Card is the initialization process.

The SD/SDIO Card Register (MCI_SDCR) allows selection of the Card Slot and the data bus width.

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

34.8.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The TRTYP field in the MCI Command Register (MCI_CMDR) allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the MCI Block Register (MCI_BLKR). In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field (IOSPCMD) in the MCI Command Register.

34.8.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled through the MCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

© 2017 Microchip Technology Inc.

34.9 MultiMedia Card Interface (MCI) User Interface

Table 34-6: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	MCI_CR	Write-only	–
0x04	Mode Register	MCI_MR	Read/Write	0x0
0x08	Data Timeout Register	MCI_DTOR	Read/Write	0x0
0x0C	SD/SDIO Card Register	MCI_SDCR	Read/Write	0x0
0x10	Argument Register	MCI_ARGR	Read/Write	0x0
0x14	Command Register	MCI_CMDR	Write-only	–
0x18	Block Register	MCI_BLKR	Read/Write	0x0
0x1C	Reserved	–	–	–
0x20	Response Register(1)	MCI_RSPR	Read-only	0x0
0x24	Response Register(1)	MCI_RSPR	Read-only	0x0
0x28	Response Register(1)	MCI_RSPR	Read-only	0x0
0x2C	Response Register(1)	MCI_RSPR	Read-only	0x0
0x30	Receive Data Register	MCI_RDR	Read-only	0x0
0x34	Transmit Data Register	MCI_TDR	Write-only	–
0x38–0x3C	Reserved	–	–	–
0x40	Status Register	MCI_SR	Read-only	0xC0E5
0x44	Interrupt Enable Register	MCI_IER	Write-only	–
0x48	Interrupt Disable Register	MCI_IDR	Write-only	–
0x4C	Interrupt Mask Register	MCI_IMR	Read-only	0x0
0x50–0xFC	Reserved	–	–	–
0x100–0x124	Reserved for the PDC	–	–	–

Note 1: The response register can be read by N accesses at the same MCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

© 2017 Microchip Technology Inc.

34.9.1 MCI Control Register

Name: MCI_CR

Access: 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
SWRST	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

MCIEN: Multi-Media Interface Enable

0: No effect.

1: Enables the Multi-Media Interface if MCDIS is 0.

MCIDIS: Multi-Media Interface Disable

0: No effect.

1: Disables the Multi-Media Interface.

PWSEN: Power Save Mode Enable

0: No effect.

1: Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field (Mode Register MCI_MR).

PWSDIS: Power Save Mode Disable

0: No effect.

1: Disables the Power Saving Mode.

SWRST: Software Reset

0: No effect.

1: Resets the MCI. A software triggered hardware reset of the MCI interface is performed.

34.9.2 MCI Mode Register

Name: MCI_MR

Access: Read/Write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
PDCMODE	PDCPADV	PDCFBYTE	WRPROOF	RDPROOF	PWSDIV		
7	6	5	4	3	2	1	0
CLKDIV							

CLKDIV: Clock Divider

Multimedia Card Interface clock (MCCK or MCI_CK) is Master Clock (MCK) divided by (2*(CLKDIV+1)).

PWSDIV: Power Saving Divider

Multimedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.

Warning: This value must be different from 0 before enabling the Power Save Mode in the MCI_CR (MCI_PWSEN bit).

RDPROOF Read Proof Enable

Enabling Read Proof allows to stop the MCI Clock during read access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Read Proof.

1: Enables Read Proof.

WRPROOF Write Proof Enable

Enabling Write Proof allows to stop the MCI Clock during write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Write Proof.

1: Enables Write Proof.

PDCFBYTE: PDC Force Byte Transfer

Enabling PDC Force Byte Transfer allows the PDC to manage with internal byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.

Warning: BLKLEN value depends on PDCFBYTE.

0: Disables PDC Force Byte Transfer. PDC type of transfer are in words.

1: Enables PDC Force Byte Transfer. PDC type of transfer are in bytes.

PDCPADV: PDC Padding Value

0: 0x00 value is used when padding data in write transfer (not only PDC transfer).

1: 0xFF value is used when padding data in write transfer (not only PDC transfer).

PDCMODE: PDC-oriented Mode

0: Disables PDC transfer

1: Enables PDC transfer. In this case, UNRE and OVRE flags in the MCI Mode Register (MCI_SR) are deactivated after the PDC transfer has been completed.

© 2017 Microchip Technology Inc.

BLKLEN: Data Block Length

This field determines the size of the data block.

This field is also accessible in the MCI Block Register (MCI_BLKR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

34.9.3 MCI Data Timeout Register

Name: MCI_DTOR

Access: 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
–	DTOMUL			DTOCYC			

DTOCYC: Data Timeout Cycle Number

DTOMUL: Data Timeout Multiplier

These fields determine the maximum number of Master Clock cycles that the MCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

Multiplier is defined by DTOMUL as shown in the following table:

DTOMUL			Multiplier
0	0	0	1
0	0	1	16
0	1	0	128
0	1	1	256
1	0	0	1024
1	0	1	4096
1	1	0	65536
1	1	1	1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the MCI Status Register (MCI_SR) raises.

© 2017 Microchip Technology Inc.

34.9.4 MCI SDCard/SDIO Register

Name: MCI_SDCR

Access: 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
SDCBUS	–	–	–	–	–	SDCSEL	

SDCSEL: SDCard/SDIO Slot

SDCSEL		SDCard/SDIO Slot
0	0	Slot A is selected.
0	1	Slot B selected
1	0	Reserved
1	1	Reserved

SDCBUS: SDCard/SDIO Bus Width

0: 1-bit data bus

1: 4-bit data bus

34.9.5 MCI Argument Register

Name: MCI_ARGR

Access: Read/Write

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

ARG: Command Argument

© 2017 Microchip Technology Inc.

34.9.6 MCI Command Register

Name: MCI_CMDR

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	IOSPCMD	
23	22	21	20	19	18	17	16
–	–	TRTYP			TRDIR	TRCMD	
15	14	13	12	11	10	9	8
–	–	–	MAXLAT	OPDCMD	SPCMD		
7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

This register is write-protected while CMDRDY is 0 in MCI_SR. If an Interrupt command is sent, this register is only writeable by an interrupt response (field SPCMD). This means that the current command execution cannot be interrupted or modified.

CMDNB: Command Number

RSPTYP: Response Type

RSP		Response Type
0	0	No response.
0	1	48-bit response.
1	0	136-bit response.
1	1	Reserved.

SPCMD: Special Command

SPCMD			Command
0	0	0	Not a special CMD.
0	0	1	Initialization CMD: 74 clock cycles for initialization sequence.
0	1	0	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
0	1	1	Reserved.
1	0	0	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
1	0	1	Interrupt response: Corresponds to the Interrupt Mode (CMD40).

OPDCMD: Open Drain Command

0: Push pull command

1: Open drain command

MAXLAT: Max Latency for Command to Response

0: 5-cycle max latency

1: 64-cycle max latency

TRCMD: Transfer Command

TRCMD		Transfer Type
0	0	No data transfer
0	1	Start data transfer
1	0	Stop data transfer
1	1	Reserved

TRDIR: Transfer Direction

0: Write

1: Read

TRTYP: Transfer Type

TRTYP			Transfer Type
0	0	0	MMC/SDCard Single Block
0	0	1	MMC/SDCard Multiple Block
0	1	0	MMC Stream
0	1	1	Reserved
1	0	0	SDIO Byte
1	0	1	SDIO Block
1	1	0	Reserved
1	1	1	Reserved

IOSPCMD: SDIO Special Command

IOSPCMD		SDIO Special Command Type
0	0	Not a SDIO Special Command
0	1	SDIO Suspend Command
1	0	SDIO Resume Command
1	1	Reserved

© 2017 Microchip Technology Inc.

34.9.7 MCI Block Register

Name: MCI_BLKR

Access: Read/Write

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

BCNT: MMC/SDIO Block Count - SDIO Byte Count

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the MCI Command Register (MCI_CMDR):

TRTYP			Type of Transfer	BCNT Authorized Values
0	0	1	MMC/SDCard Multiple Block	From 1 to 65535: Value 0 corresponds to an infinite block transfer.
1	0	0	SDIO Byte	From 1 to 512 bytes: value 0 corresponds to a 512-byte transfer. Values from 0x200 to 0xFFFF are forbidden.
1	0	1	SDIO Block	From 1 to 511 blocks: value 0 corresponds to an infinite block transfer. Values from 0x200 to 0xFFFF are forbidden.
Other values			–	Reserved.

Warning: In SDIO Byte and Block modes, writing to the 7 last bits of BCNT field, is forbidden and may lead to unpredictable results.

BLKLEN: Data Block Length

This field determines the size of the data block.

This field is also accessible in the MCI Mode Register (MCI_MR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

34.9.8 MCI Response Register

Name: MCI_RSPR

Access: Read-only

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

RSP: Response

Note: The response register can be read by N accesses at the same MCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

© 2017 Microchip Technology Inc.

34.9.9 MCI Receive Data Register

Name: MCI_RDR

Access: Read-only

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

DATA: Data to Read

34.9.10 MCI Transmit Data Register

Name: MCI_TDR

Access: Write-only

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

DATA: Data to Write

© 2017 Microchip Technology Inc.

34.9.11 MCI Status Register

Name: MCI_SR

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

CMDRDY: Command Ready

0: A command is in progress.

1: The last command has been sent. Cleared when writing in the MCI_CMDR.

RXRDY: Receiver Ready

0: Data has not yet been received since the last read of MCI_RDR.

1: Data has been received since the last read of MCI_RDR.

TXRDY: Transmit Ready

0: The last data written in MCI_TDR has not yet been transferred in the Shift Register.

1: The last data written in MCI_TDR has been transferred in the Shift Register.

BLKE: Data Block Ended

This flag must be used only for Write Operations.

0: A data block transfer is not yet finished. Cleared when reading the MCI_SR.

1: A data block transfer has ended, including the CRC16 Status transmission.
In PDC mode (PDCMODE = 1), the flag is set when the CRC Status of the last block has been transmitted (TXBUFE already set).
Otherwise (PDCMODE = 0), the flag is set for each transmitted CRC Status.

Refer to the MMC or SD Specification for more details concerning the CRC Status.

DTIP: Data Transfer in Progress

0: No data transfer in progress.

1: The current data transfer is still in progress, including CRC16 calculation. Cleared at the end of the CRC16 calculation.

NOTBUSY: MCI Not Busy

This flag must be used only for Write Operations.

A block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line (DAT0) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.

The NOTBUSY flag allows to deal with these different states.

0: The MCI is not ready for new data transfer. Cleared at the end of the card response.

1: The MCI is ready for new data transfer. Set when the busy state on the data line has ended. This corresponds to a free internal data receive buffer of the card.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

ENDRX: End of RX Buffer

0: The Receive Counter Register has not reached 0 since the last write in MCI_RCR or MCI_RNCR.

1: The Receive Counter Register has reached 0 since the last write in MCI_RCR or MCI_RNCR.

© 2017 Microchip Technology Inc.

ENDTX: End of TX Buffer

0: The Transmit Counter Register has not reached 0 since the last write in MCI_TCR or MCI_TNCR.

1: The Transmit Counter Register has reached 0 since the last write in MCI_TCR or MCI_TNCR.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

RXBUFF: RX Buffer Full

0: MCI_RCR or MCI_RNCR has a value other than 0.

1: Both MCI_RCR and MCI_RNCR have a value of 0.

TXBUFE: TX Buffer Empty

0: MCI_TCR or MCI_TNCR has a value other than 0.

1: Both MCI_TCR and MCI_TNCR have a value of 0.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

RINDE: Response Index Error

0: No error.

1: A mismatch is detected between the command index sent and the response index received. Cleared when writing in the MCI_CMDR.

RDIRE: Response Direction Error

0: No error.

1: The direction bit from card to host in the response has not been detected.

RCRCE: Response CRC Error

0: No error.

1: A CRC7 error has been detected in the response. Cleared when writing in the MCI_CMDR.

RENDE: Response End Bit Error

0: No error.

1: The end bit of the response has not been detected. Cleared when writing in the MCI_CMDR.

RTOE: Response Time-out Error

0: No error.

1: The response time-out set by MAXLAT in the MCI_CMDR has been exceeded. Cleared when writing in the MCI_CMDR.

DCRCE: Data CRC Error

0: No error.

1: A CRC16 error has been detected in the last data block. Reset by reading in the MCI_SR.

DTOE: Data Time-out Error

0: No error.

1: The data time-out set by DTOCYC and DTOMUL in MCI_DTOR has been exceeded. Reset by reading in the MCI_SR.

OVRE: Overrun

0: No error.

1: At least one 8-bit received data has been lost (not read). Cleared when sending a new data transfer command.

UNRE: Underrun

0: No error.

1: At least one 8-bit data has been sent without valid information (not written). Cleared when sending a new data transfer command.

© 2017 Microchip Technology Inc.

SDIOIRQA: SDIO Interrupt for Slot A

0: No interrupt detected on SDIO Slot A.

1: A SDIO Interrupt on Slot A has reached. Cleared when reading the MCI_SR.

SDIOIRQB: SDIO Interrupt for Slot B

0: No interrupt detected on SDIO Slot B.

1: A SDIO Interrupt on Slot B has reached. Cleared when reading the MCI_SR.

RXBUFF: RX Buffer Full

0: MCI_RCR or MCI_RNCR has a value other than 0.

1: Both MCI_RCR and MCI_RNCR have a value of 0.

TXBUFE: TX Buffer Empty

0: MCI_TCR or MCI_TNCR has a value other than 0.

1: Both MCI_TCR and MCI_TNCR have a value of 0.

© 2017 Microchip Technology Inc.

34.9.12 MCI Interrupt Enable Register

Name: MCI_IER

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

CMDRDY: Command Ready Interrupt Enable

RXRDY: Receiver Ready Interrupt Enable

TXRDY: Transmit Ready Interrupt Enable

BLKE: Data Block Ended Interrupt Enable

DTIP: Data Transfer in Progress Interrupt Enable

NOTBUSY: Data Not Busy Interrupt Enable

ENDRX: End of Receive Buffer Interrupt Enable

ENDTX: End of Transmit Buffer Interrupt Enable

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable

SDIOIRQB: SDIO Interrupt for Slot B Interrupt Enable

RXBUFF: Receive Buffer Full Interrupt Enable

TXBUFE: Transmit Buffer Empty Interrupt Enable

RINDE: Response Index Error Interrupt Enable

RDIRE: Response Direction Error Interrupt Enable

RCRCE: Response CRC Error Interrupt Enable

RENDE: Response End Bit Error Interrupt Enable

RTOE: Response Time-out Error Interrupt Enable

DCRCE: Data CRC Error Interrupt Enable

DTOE: Data Time-out Error Interrupt Enable

OVRE: Overrun Interrupt Enable

UNRE: UnderRun Interrupt Enable

0: No effect.

1: Enables the corresponding interrupt.

© 2017 Microchip Technology Inc.

34.9.13 MCI Interrupt Disable Register

Name: MCI_IDR

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

CMDRDY: Command Ready Interrupt Disable

RXRDY: Receiver Ready Interrupt Disable

TXRDY: Transmit Ready Interrupt Disable

BLKE: Data Block Ended Interrupt Disable

DTIP: Data Transfer in Progress Interrupt Disable

NOTBUSY: Data Not Busy Interrupt Disable

ENDRX: End of Receive Buffer Interrupt Disable

ENDTX: End of Transmit Buffer Interrupt Disable

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable

SDIOIRQB: SDIO Interrupt for Slot B Interrupt Disable

RXBUFF: Receive Buffer Full Interrupt Disable

TXBUFE: Transmit Buffer Empty Interrupt Disable

RINDE: Response Index Error Interrupt Disable

RDIRE: Response Direction Error Interrupt Disable

RCRCE: Response CRC Error Interrupt Disable

RENDE: Response End Bit Error Interrupt Disable

RTOE: Response Time-out Error Interrupt Disable

DCRCE: Data CRC Error Interrupt Disable

DTOE: Data Time-out Error Interrupt Disable

OVRE: Overrun Interrupt Disable

UNRE: UnderRun Interrupt Disable

0: No effect.

1: Disables the corresponding interrupt.

© 2017 Microchip Technology Inc.

34.9.14 MCI Interrupt Mask Register

Name: MCI_IMR

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	-	-	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

CMDRDY: Command Ready Interrupt Mask

RXRDY: Receiver Ready Interrupt Mask

TXRDY: Transmit Ready Interrupt Mask

BLKE: Data Block Ended Interrupt Mask

DTIP: Data Transfer in Progress Interrupt Mask

NOTBUSY: Data Not Busy Interrupt Mask

ENDRX: End of Receive Buffer Interrupt Mask

ENDTX: End of Transmit Buffer Interrupt Mask

SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask

SDIOIRQB: SDIO Interrupt for Slot B Interrupt Mask

RXBUFF: Receive Buffer Full Interrupt Mask

TXBUFE: Transmit Buffer Empty Interrupt Mask

RINDE: Response Index Error Interrupt Mask

RDIRE: Response Direction Error Interrupt Mask

RCRCE: Response CRC Error Interrupt Mask

RENDE: Response End Bit Error Interrupt Mask

RTOE: Response Time-out Error Interrupt Mask

DCRCE: Data CRC Error Interrupt Mask

DTOE: Data Time-out Error Interrupt Mask

OVRE: Overrun Interrupt Mask

UNRE: UnderRun Interrupt Mask

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

© 2017 Microchip Technology Inc.

35. Ethernet MAC 10/100 (EMAC)

35.1 Overview

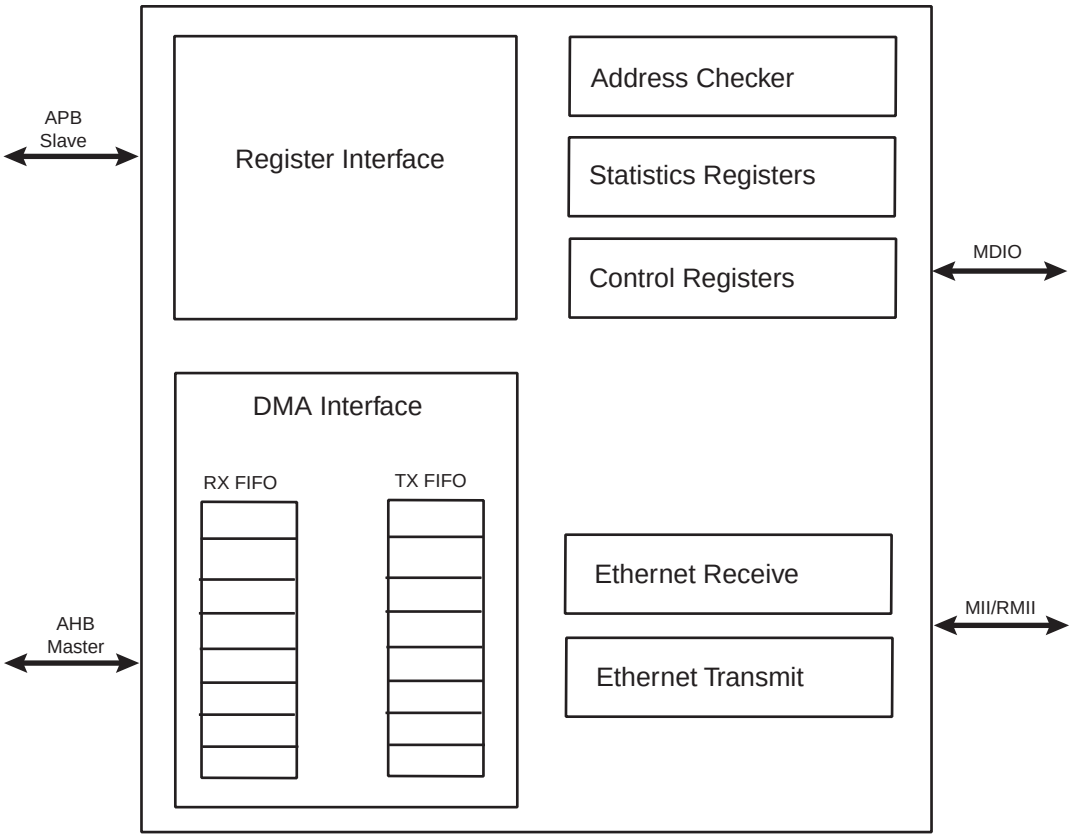
The EMAC module implements a 10/100 Ethernet MAC compatible with the IEEE 802.3 standard using an address checker, statistics and control registers, receive and transmit blocks, and a DMA interface.

The address checker recognizes four specific 48-bit addresses and contains a 64-bit hash register for matching multicast and unicast addresses. It can recognize the broadcast address of all ones, copy all frames, and act on an external address match signal.

The statistics register block contains registers for counting various types of event associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3.

35.2 Block Diagram

Figure 35-1: EMAC Block Diagram



35.3 Functional Description

The MACB has several clock domains:

- System bus clock (AHB and APB): DMA and register blocks
- Transmit clock: transmit block
- Receive clock: receive and address checker blocks

The only system constraint is 160 MHz for the system bus clock, above which MDC would toggle at above 2.5 MHz.

The system bus clock must run at least as fast as the receive clock and transmit clock (25 MHz at 100 Mbps, and 2.5 MHz at 10 Mbps).

Figure 35-1 illustrates the different blocks of the EMAC module.

The control registers drive the MDIO interface, set up DMA activity, start frame transmission and select modes of operation such as full- or half-duplex.

The receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the address checking block and DMA interface.

The transmit block takes data from the DMA interface, adds preamble and, if necessary, pad and FCS, and transmits data according to the CSMA/CD (carrier sense multiple access with collision detect) protocol. The start of transmission is deferred if CRS (carrier sense) is active.

If COL (collision) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. CRS and COL have no effect in full duplex mode.

The DMA block connects to external memory through its AHB bus interface. It contains receive and transmit FIFOs for buffering frame data. It loads the transmit FIFO and empties the receive FIFO using AHB bus master operations. Receive data is not sent to memory until the address checking logic has determined that the frame should be copied. Receive or transmit frames are stored in one or more buffers. Receive buffers have a fixed length of 128 bytes. Transmit buffers range in length between 0 and 2047 bytes, and up to 128 buffers are permitted per frame. The DMA block manages the transmit and receive framebuffer queues. These queues can hold multiple frames.

35.3.1 Clock

Synchronization module in the EMAC requires that the bus clock (hclk) runs at the speed of the macb_tx/rx_clk at least, which is 25 MHz at 100 Mbps, and 2.5 MHz at 10 Mbps.

35.3.2 Memory Interface

Frame data is transferred to and from the EMAC through the DMA interface. All transfers are 32-bit words and may be single accesses or bursts of 2, 3 or 4 words. Burst accesses do not cross sixteen-byte boundaries. Bursts of 4 words are the default data transfer; single accesses or bursts of less than four words may be used to transfer data at the beginning or the end of a buffer.

The DMA controller performs six types of operation on the bus. In order of priority, these are:

1. Receive buffer manager write
2. Receive buffer manager read
3. Transmit data DMA read
4. Receive data DMA write
5. Transmit buffer manager read
6. Transmit buffer manager write

35.3.2.1 FIFO

The FIFO depths are 128 bytes for receive and 128 bytes for transmit and are a function of the system clock speed, memory latency and network speed.

Data is typically transferred into and out of the FIFOs in bursts of four words. For receive, a bus request is asserted when the FIFO contains four words and has space for 28 more. For transmit, a bus request is generated when there is space for four words, or when there is space for 27 words if the next transfer is to be only one or two words.

Thus the bus latency must be less than the time it takes to load the FIFO and transmit or receive three words (112 bytes) of data.

At 100 Mbit/s, it takes 8960 ns to transmit or receive 112 bytes of data. In addition, six master clock cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For a 133 MHz master clock this takes 45 ns, making the bus latency requirement 8915 ns.

© 2017 Microchip Technology Inc.

35.3.2.2 Receive Buffers

Received frames, including CRC/FCS optionally, are written to receive buffers stored in memory. Each receive buffer is 128 bytes long. The start location for each receive buffer is stored in memory in a list of receive buffer descriptors at a location pointed to by the receive buffer queue pointer register. The receive buffer start location is a word address. For the first buffer of a frame, the start location can be offset by up to three bytes depending on the value written to bits 14 and 15 of the network configuration register. If the start location of the buffer is offset the available length of the first buffer of a frame is reduced by the corresponding number of bytes.

Each list entry consists of two words, the first being the address of the receive buffer and the second being the receive status. If the length of a receive frame exceeds the buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit and the offset bits, if appropriate. Bit zero of the address field is written to one to show the buffer has been used. The receive buffer manager then reads the location of the next receive buffer and fills that with receive frame data. The final buffer descriptor status word contains the complete frame status. Refer to the table below for details of the receive buffer descriptor list.

Table 35-1: Receive Buffer Descriptor Entry

Bit	Function
Word 0	
31:2	Address of beginning of buffer
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the EMAC to write data to the receive buffer. The EMAC sets this to one once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
Word 1	
31	Global all ones broadcast address detected
30	Multicast hash match
29	Unicast hash match
28	External address match
27	Reserved for future use
26	Specific address register 1 match
25	Specific address register 2 match
24	Specific address register 3 match
23	Specific address register 4 match
22	Type ID match
21	VLAN tag detected (i.e., type id of 0x8100)
20	Priority tag detected (i.e., type id of 0x8100 and null VLAN identifier)
19:17	VLAN priority (only valid if bit 21 is set)
16	Concatenation format indicator (CFI) bit (only valid if bit 21 is set)
15	End of frame - when set the buffer contains the end of a frame. If end of frame is not set, then the only other valid status are bits 12, 13 and 14.
14	Start of frame - when set the buffer contains the start of a frame. If both bits 15 and 14 are set, then the buffer contains a whole frame.
13:12	Receive buffer offset - indicates the number of bytes by which the data in the first buffer is offset from the word address. Updated with the current values of the network configuration register. If jumbo frame mode is enabled through bit 3 of the network configuration register, then bits 13:12 of the receive buffer descriptor entry are used to indicate bits 13:12 of the frame length.
11:0	Length of frame including FCS (if selected). Bits 13:12 are also used if jumbo frame mode is selected.

To receive frames, the buffer descriptors must be initialized by writing an appropriate address to bits 31 to 2 in the first word of each list entry. Bit zero must be written with zero. Bit one is the wrap bit and indicates the last entry in the list.

The start location of the receive buffer descriptor list must be written to the receive buffer queue pointer register before setting the receive enable bit in the network control register to enable receive. As soon as the receive block starts writing received frame data to the receive FIFO, the receive buffer manager reads the first receive buffer location pointed to by the receive buffer queue pointer register.

If the filter block then indicates that the frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered. If the current buffer pointer has its wrap bit set or is the 1024th descriptor, the next receive buffer location is read from the beginning of the receive descriptor list. Otherwise, the next receive buffer location is read from the next word in memory.

There is an 11-bit counter to count out the 2048 word locations of a maximum length, receive buffer descriptor list. This is added with the value originally written to the receive buffer queue pointer register to produce a pointer into the list. A read of the receive buffer queue pointer register returns the pointer value, which is the queue entry currently being accessed. The counter is reset after receive status is written to a descriptor that has its wrap bit set or rolls over to zero after 1024 descriptors have been accessed. The value written to the receive buffer pointer register may be any word-aligned address, provided that there are at least 2048 word locations available between the pointer and the top of the memory.

Section 3.6 of the AMBA 2.0 specification states that bursts should not cross 1K boundaries. As receive buffer manager writes are bursts of two words, to ensure that this does not occur, it is best to write the pointer register with the least three significant bits set to zero. As receive buffers are used, the receive buffer manager sets bit zero of the first word of the descriptor to indicate *used*. If a receive error is detected the receive buffer currently being written is recovered. Previous buffers are not recovered. Software should search through the *used* bits in the buffer descriptors to find out how many frames have been received. It should be checking the start-of-frame and end-of-frame bits, and not rely on the value returned by the receive buffer queue pointer register which changes continuously as more buffers are used.

For CRC errored frames, excessive length frames or length field mismatched frames, all of which are counted in the statistics registers, it is possible that a frame fragment might be stored in a sequence of receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working Ethernet system, there should be no excessively long frames or frames greater than 128 bytes with CRC/FCS errors. Collision fragments are less than 128 bytes long. Therefore, it is a rare occurrence to find a frame fragment in a receive buffer.

If bit zero is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the DMA block sets the buffer not available bit in the receive status register and triggers an interrupt.

If bit zero is set when the receive buffer manager reads the location of the receive buffer and a frame is being received, the frame is discarded and the receive resource error statistics register is incremented.

A receive overrun condition occurs when bus was not granted in time or because HRESP was not OK (bus error). In a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame received with an address that is recognized reuses the buffer.

If bit 17 of the network configuration register is set, the FCS of received frames shall not be copied to memory. The frame length indicated in the receive status field shall be reduced by four bytes in this case.

35.3.2.3 Transmit Buffer

Frames to be transmitted are stored in one or more transmit buffers. Transmit buffers can be between 0 and 2047 bytes long, so it is possible to transmit frames longer than the maximum length specified in IEEE Standard 802.3. Zero length buffers are allowed. The maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer register. Each list entry consists of two words, the first being the byte address of the transmit buffer and the second containing the transmit control and status. Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad is also automatically generated to take frames to a minimum length of 64 bytes. Table 35-2 defines an entry in the transmit buffer descriptor list. To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits 31 to 0 in the first word of each list entry. The second transmit buffer descriptor is initialized with control information that indicates the length of the buffer, whether or not it is to be transmitted with CRC and whether the buffer is the last buffer in the frame.

After transmission, the control bits are written back to the second word of the first buffer along with the “used” bit and other status information. Bit 31 is the “used” bit which must be zero when the control word is read if transmission is to happen. It is written to one when a frame has been transmitted. Bits 27, 28 and 29 indicate various transmit error conditions. Bit 30 is the “wrap” bit which can be set for any buffer within a frame. If no wrap bit is encountered after 1024 descriptors, the queue pointer rolls over to the start in a similar fashion to the receive queue.

© 2017 Microchip Technology Inc.

The transmit buffer queue pointer register must not be written while transmit is active. If a new value is written to the transmit buffer queue pointer register, the queue pointer resets itself to point to the beginning of the new queue. If transmit is disabled by writing to bit 3 of the network control, the transmit buffer queue pointer register resets to point to the beginning of the transmit queue. Note that disabling receive does not have the same effect on the receive queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to bit 9, the *Transmit Start* bit of the network control register. Transmit is halted when a buffer descriptor with its *used* bit set is read, or if a transmit error occurs, or by writing to the transmit halt bit of the network control register. (Transmission is suspended if a pause frame is received while the pause enable bit is set in the network configuration register.) Rewriting the start bit while transmission is active is allowed.

Transmission control is implemented with a Tx_go variable which is readable in the transmit status register at bit location 3. The Tx_go variable is reset when:

- transmit is disabled
- a buffer descriptor with its ownership bit set is read
- a new value is written to the transmit buffer queue pointer register
- bit 10, tx_halt, of the network control register is written
- there is a transmit error such as too many retries or a transmit underrun.

To set tx_go, write to bit 9, tx_start, of the network control register. Transmit halt does not take effect until any ongoing transmit finishes. If a collision occurs during transmission of a multi-buffer frame, transmission automatically restarts from the first buffer of the frame. If a “used” bit is read midway through transmission of a multi-buffer frame, this is treated as a transmit error. Transmission stops, tx_er is asserted and the FCS is bad.

If transmission stops due to a transmit error, the transmit queue pointer resets to point to the beginning of the transmit queue. Software needs to re-initialize the transmit queue after a transmit error.

If transmission stops due to a “used” bit being read at the start of the frame, the transmission queue pointer is not reset and transmit starts from the same transmit buffer descriptor when the transmit start bit is written.

Table 35-2: Transmit Buffer Descriptor Entry

Bit	Function
	Word 0
31:0	Byte Address of buffer
	Word 1
31	Used. Needs to be zero for the EMAC to read data from the transmit buffer. The EMAC sets this to one for the first buffer of a frame once it has been successfully transmitted. Software has to clear this bit before the buffer can be used again. **Note:** This bit is only set for the first buffer in a frame unlike receive where all buffers have the Used bit set once used.
30	Wrap. Marks last descriptor in transmit buffer descriptor list.
29	Retry limit exceeded, transmit error detected
28	Transmit underrun, occurs either when hresp is not OK (bus error) or the transmit data could not be fetched in time or when buffers are exhausted in mid frame.
27	Buffers exhausted in mid frame
26:17	Reserved
16	No CRC. When set, no CRC is appended to the current frame. This bit only needs to be set for the last buffer of a frame.
15	Last buffer. When set, this bit indicates the last buffer in the current frame has been reached.
14:11	Reserved
10:0	Length of buffer

35.3.3 Transmit Block

This block transmits frames in accordance with the Ethernet IEEE 802.3 CSMA/CD protocol. Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO a word at a time. Data is transmitted least significant nibble first. If necessary, padding is added to increase the frame length to 60 bytes. CRC is calculated as a 32-bit polynomial. This is inverted and appended to the end of the frame, taking the frame length to a minimum of 64 bytes. If the No CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended.

In full-duplex mode, frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half-duplex mode, the transmitter checks carrier sense. If asserted, it waits for it to de-assert and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed.

The back-off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO and a 10-bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision, 1 bit is used, after the second 2, and so on up to 10. Above 10, all 10 bits are used. An error is indicated and no further attempts are made if 16 attempts cause collisions.

If transmit DMA underruns, bad CRC is automatically appended using the same mechanism as jam insertion and the tx_er signal is asserted. For a properly configured system, this should never happen.

If the back pressure bit is set in the network control register in half duplex mode, the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit-rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half-duplex mode.

35.3.4 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 35-3: Start of an 802.3 Pause Frame

Destination Address	Source Address	Type (Mac Control Frame)	Pause Opcode	Pause Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The network configuration register contains a receive pause enable bit (13). If a valid pause frame is received, the pause time register is updated with the frame's pause time, regardless of its current contents and regardless of the state of the configuration register bit 13. An interrupt (12) is triggered when a pause frame is received, assuming it is enabled in the interrupt mask register. If bit 13 is set in the network configuration register and the value of the pause time register is non-zero, no new frame is transmitted until the pause time register has decremented to zero.

The loading of a new pause time, and hence the pausing of transmission, only occurs when the EMAC is configured for full-duplex operation. If the EMAC is configured for half-duplex, there is no transmission pause, but the pause frame received interrupt is still triggered.

A valid pause frame is defined as having a destination address that matches either the address stored in specific address register 1 or matches 0x0180C2000001 and has the MAC control frame type ID of 0x8808 and the pause opcode of 0x0001. Pause frames that have FCS or other errors are treated as invalid and are discarded. Valid pause frames received increment the Pause Frame Received statistic register.

The pause time register decrements every 512 bit times (i.e., 128 `rx_clks` in nibble mode) once transmission has stopped. For test purposes, the register decrements every `rx_clk` cycle once transmission has stopped if bit 12 (retry test) is set in the network configuration register. If the pause enable bit (13) is not set in the network configuration register, then the decrementing occurs regardless of whether transmission has stopped or not.

An interrupt (13) is asserted whenever the pause time register decrements to zero (assuming it is enabled in the interrupt mask register).

35.3.5 Receive Block

The receive block checks for valid preamble, FCS, alignment and length, presents received frames to the DMA block and stores the frames destination address for use by the address checking block. If, during frame reception, the frame is found to be too long or rx_er is asserted, a bad frame indication is sent to the DMA block. The DMA block then ceases sending data to memory. At the end of frame reception, the receive block indicates to the DMA block whether the frame is good or bad. The DMA block recovers the current receive buffer if the frame was bad. The receive block signals the register block to increment the alignment error, the CRC (FCS) error, the short frame, long frame, jabber error, the receive symbol error statistics and the length field mismatch statistics.

© 2017 Microchip Technology Inc.

The enable bit for jumbo frames in the network configuration register allows the EMAC to receive jumbo frames of up to 10240 bytes in size. This operation does not form part of the IEEE802.3 specification and is disabled by default. When jumbo frames are enabled, frames received with a frame size greater than 10240 bytes are discarded.

35.3.6 Address Checking Block

The address checking (or filter) block indicates to the DMA block which receive frames should be copied to memory. Whether a frame is copied depends on what is enabled in the network configuration register, the state of the external match pin, the contents of the specific address and hash registers and the frame's destination address. In this implementation of the EMAC, the frame's source address is not checked. Provided that bit 18 of the Network Configuration register is not set, a frame is not copied to memory if the EMAC is transmitting in half duplex mode at the time a destination address is received. If bit 18 of the Network Configuration register is set, frames can be received while transmitting in half-duplex mode.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, the LSB of the first byte of the frame, is the group/individual bit: this is *One* for multicast addresses and *Zero* for unicast. The *All Ones* address is the broadcast address, and a special case of multicast.

The EMAC supports recognition of four specific addresses. Each specific address requires two registers, specific address register bottom and specific address register top. Specific address register bottom stores the first four bytes of the destination address and specific address register top contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the specific address registers once they have been activated. The addresses are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. If a receive frame address matches an active address, the frame is copied to memory.

The following example illustrates the use of the address match registers for a MAC address of 21:43:65:87:A9:CB.

Preamble 55

SFD D5

DA (Octet0 - LSB) 21

DA(Octet 1) 43

DA(Octet 2) 65

DA(Octet 3) 87

DA(Octet 4) A9

DA (Octet5 - MSB) CB

SA (LSB) 00

SA 00

SA 00

SA 00

SA 00

SA (MSB) 43

SA (LSB) 21

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

- Base address + 0x98 0x87654321 (Bottom)
- Base address + 0x9C 0x0000CBA9 (Top)

And for a successful match to the Type ID register, the following should be set up:

- Base address + 0xB8 0x00004321

35.3.7 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized if the 'no broadcast' bit in the network configuration register is zero.

© 2017 Microchip Technology Inc.

35.3.8 Hash Addressing

The hash address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in hash register bottom and the most significant bits in hash register top.

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. The destination address is reduced to a 6-bit index into the 64-bit hash register using the following hash function. The hash function is an *exclusive or* of every sixth bit of the destination address.

hash_index[5] = da[5] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^ da[47]

hash_index[4] = da[4] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^ da[46]

hash_index[3] = da[3] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^ da[45]

hash_index[2] = da[2] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^ da[44]

hash_index[1] = da[1] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^ da[43]

hash_index[0] = da[0] ^ da[06] ^ da[12] ^ da[18] ^ da[24] ^ da[30] ^ da[36] ^ da[42]

`da[0]` represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and `da[47]` represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the hash register, then the frame is matched according to whether the frame is multicast or unicast.

A multicast match is signalled if the multicast hash enable bit is set. da[0] is 1 and the hash index points to a bit set in the hash register.

A unicast match is signalled if the unicast hash enable bit is set. da[0] is 0 and the hash index points to a bit set in the hash register.

To receive all multicast frames, the hash register should be set with all ones and the multicast hash enable bit should be set in the network configuration register.

35.3.9 Copy All Frames (or Promiscuous Mode)

If the copy all frames bit is set in the network configuration register, then all non-errored frames are copied to memory. For example, frames that are too long, too short, or have FCS errors or rx_er asserted during reception are discarded and all others are received. Frames with FCS errors are copied to memory if bit 19 in the network configuration register is set.

35.3.10 Type ID Checking

The contents of the type_id register are compared against the length/type ID of received frames (i.e., bytes 13 and 14). Bit 22 in the receive buffer descriptor status is set if there is a match. The reset state of this register is zero which is unlikely to match the length/type ID of any valid Ethernet frame.

Note: A type ID match does not affect whether a frame is copied to memory.

35.3.11 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 35-4: 802.1Q VLAN Tag

TPID (Tag Protocol Identifier) 16 bits	TCI (Tag Control Information) 16 bits
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame, adding an extra four bytes to the frame. If the VID (VLAN identifier) is null (0x000), this indicates a priority-tagged frame. The MAC can support frame lengths up to 1536 bytes, 18 bytes more than the original Ethernet maximum frame length of 1518 bytes. This is achieved by setting bit 8 in the network configuration register.

The following bits in the receive buffer descriptor status word give information about VLAN tagged frames:

- Bit 21 set if receive frame is VLAN tagged (i.e. type id of 0x8100)
- Bit 20 set if receive frame is priority tagged (i.e. type id of 0x8100 and null VID). (If bit 20 is set bit 21 is set also.)
- Bit 19, 18 and 17 set to priority if bit 21 is set
- Bit 16 set to CFI if bit 21 is set

© 2017 Microchip Technology Inc.

35.3.12 PHY Maintenance

The register EMAC_MAN enables the EMAC to communicate with a PHY by means of the MDIO interface. It is used during auto-negotiation to ensure that the EMAC and the PHY are configured for the same speed and duplex configuration.

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the network status register (about 2000 MCK cycles later when bit ten is set to zero, and bit eleven is set to one in the network configuration register). An interrupt is generated as this bit is set. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO.

Reading during the shift operation returns the current contents of the shift register. At the end of management operation, the bits have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bits[31:28] should be written as 0x0011. For a description of MDC generation, see the network configuration register in the Section 35.5.1 Network Control Register.

35.3.13 Media Independent Interface

The Ethernet MAC is capable of interfacing to both RMII and MII Interfaces. The RMII bit in the EMAC_USRIO register controls the interface that is selected. When this bit is set, the RMII interface is selected, else the MII interface is selected.

The MII and RMII interface are capable of both 10Mb/s and 100Mb/s data rates as described in the IEEE 802.3u standard. The signals used by the MII and RMII interfaces are described in Table 35-5.

Table 35-5: Pin Configuration

Pin Name	MII	RMII
ETXCK_EREFCK	ETXCK: Transmit Clock	EREFCK: Reference Clock
ECRS	ECRS: Carrier Sense	
ECOL	ECOL: Collision Detect	
ERXDV	ERXDV: Data Valid	ECRSDV: Carrier Sense/Data Valid
ERX0–ERX3	ERX0–ERX3: 4-bit Receive Data	ERX0–ERX1: 2-bit Receive Data
ERXER	ERXER: Receive Error	ERXER: Receive Error
ERXCK	ERXCK: Receive Clock	
ETXEN	ETXEN: Transmit Enable	ETXEN: Transmit Enable
ETX0–ETX3	ETX0–ETX3: 4-bit Transmit Data	ETX0–ETX1: 2-bit Transmit Data
ETXER	ETXER: Transmit Error	

The intent of the RMII is to provide a reduced pin count alternative to the IEEE 802.3u MII. It uses 2 bits for transmit (ETX0 and ETX1) and two bits for receive (ERX0 and ERX1). There is a Transmit Enable (ETXEN), a Receive Error (ERXER), a Carrier Sense (ECRS_DV), and a 50 MHz Reference Clock (ETXCK_EREFCK) for 100Mb/s data rate.

35.3.13.1 RMII Transmit and Receive Operation

The same signals are used internally for both the RMII and the MII operations. The RMII maps these signals in a more pin-efficient manner. The transmit and receive bits are converted from a 4-bit parallel format to a 2-bit parallel scheme that is clocked at twice the rate. The carrier sense and data valid signals are combined into the ECRSDV signal. This signal contains information on carrier sense, FIFO status, and validity of the data. Transmit error bit (ETXER) and collision detect (ECOL) are not used in RMII mode.

35.4 Programming Interface

35.4.1 Initialization

35.4.1.1 Configuration

Initialization of the EMAC configuration (e.g., loop-back mode, frequency ratios) must be done while the transmit and receive circuits are disabled. See the description of the network control register and network configuration register earlier in this document.

To change loop-back mode, the following sequence of operations must be followed:

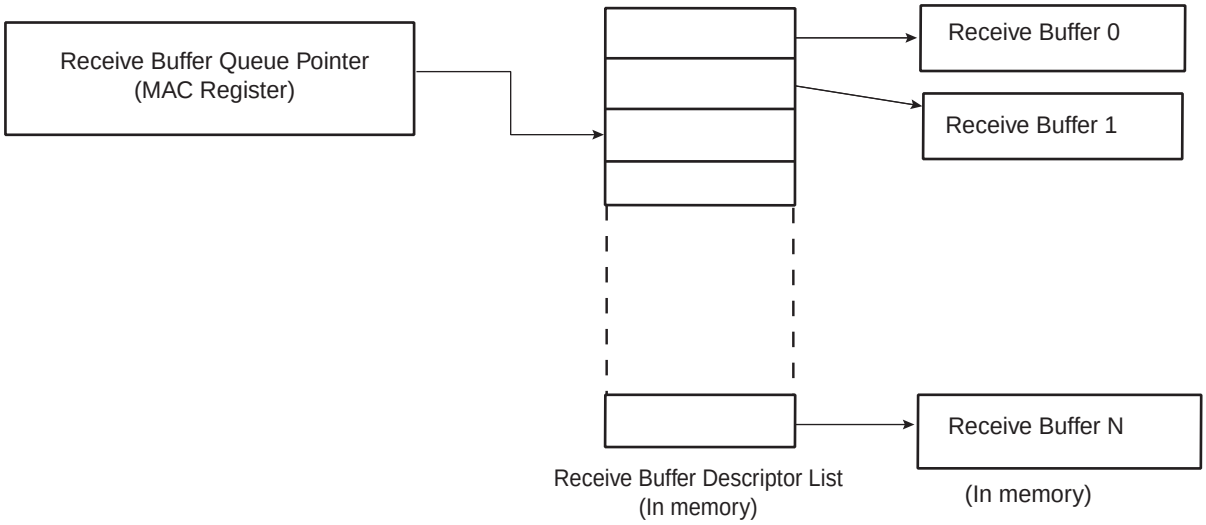
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop-back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: These writes to network control register cannot be combined in any way.

35.4.1.2 Receive Buffer List

Receive data is written to areas of data (i.e., buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries as defined in Section 35.3.2.2 “Receive Buffers”. It points to this data structure.

Figure 35-2: Receive Buffer List



To create the list of buffers:

1. Allocate a number (*n*) of buffers of 128 bytes in system memory.
2. Allocate an area 2*n* words for the receive buffer descriptor entry in system memory and create *n* entries in this list. Mark all entries in this list as owned by EMAC, i.e., bit 0 of word 0 set to 0.
3. If less than 1024 buffers are defined, the last descriptor must be marked with the wrap bit (bit 1 in word 0 set to 1).
4. Write address of receive buffer descriptor entry to EMAC register `receive_buffer` queue pointer.
5. The receive circuits can then be enabled by writing to the address recognition registers and then to the network control register.

35.4.1.3 Transmit Buffer List

Transmit data is read from areas of data (the buffers) in system memory These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries (as defined in Table 35-2) that points to this data structure.

To create this list of buffers:

1. Allocate a number (*n*) of buffers of between 1 and 2047 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 2*n* words for the transmit buffer descriptor entry in system memory and create N entries in this list. Mark all entries in this list as owned by EMAC, i.e. bit 31 of word 1 set to 0.

© 2017 Microchip Technology Inc.

3. If fewer than 1024 buffers are defined, the last descriptor must be marked with the wrap bit — bit 30 in word 1 set to 1.
4. Write address of transmit buffer descriptor entry to EMAC register transmit_buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the network control register.

35.4.1.4 Address Matching

The EMAC register-pair hash address and the four specific address register-pairs must be written with the required values. Each register-pair comprises a bottom register and top register, with the bottom register being written first. The address matching is disabled for a particular register-pair after the bottom-register has been written and re-enabled when the top register is written. See Section 35.3.6 “Address Checking Block” for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

35.4.1.5 Interrupts

There are 14 interrupt conditions that are detected within the EMAC. These are ORed to make a single interrupt. Depending on the overall system design, this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler (refer to Section 26. “Advanced Interrupt Controller (AIC)”). To ascertain which interrupt has been generated, read the interrupt status register. Note that this register clears itself when read. At reset, all interrupts are disabled. To enable an interrupt, write to interrupt enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to interrupt disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read interrupt mask register: if the bit is set to 1, the interrupt is disabled.

35.4.1.6 Transmitting Frames

To set up a frame for transmission:

1. Enable transmit in the network control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous, varying byte lengths can be used as long as they conclude on byte borders.
3. Set-up the transmit buffer list.
4. Set the network control register to enable transmission and enable interrupts.
5. Write data for transmission into these buffers.
6. Write the address to transmit buffer descriptor queue pointer.
7. Write control and length to word one of the transmit buffer descriptor entry.
8. Write to the transmit start bit in the network control register.

35.4.1.7 Receiving Frames

When a frame is received and the receive circuits are enabled, the EMAC checks the address and, in the following cases, the frame is written to system memory:

- if it matches one of the four specific address registers.
- if it matches the hash address function.
- if it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.
- if the EMAC is configured to copy all frames.

The register receive buffer queue pointer points to the next entry (see Table 35-1) and the EMAC uses this as the address in system memory to write the frame to. Once the frame has been completely and successfully received and written to system memory, the EMAC then updates the receive buffer descriptor entry with the reason for the address match and marks the area as being owned by software. Once this is complete an interrupt receive complete is set. Software is then responsible for handling the data in the buffer and then releasing the buffer by writing the ownership bit back to 0.

If the EMAC is unable to write the data at a rate to match the incoming frame, then an interrupt receive overrun is set. If there is no receive buffer available, i.e., the next buffer is still owned by software, the interrupt receive buffer not available is set. If the frame is not successfully received, a statistic register is incremented and the frame is discarded without informing software.

© 2017 Microchip Technology Inc.

35.5 Ethernet MAC 10/100 (EMAC) User Interface

Table 35-6: Register Mapping

Offset	Register	Name	Access	Reset
0x00	Network Control Register	EMAC_NCR	Read/Write	0
0x04	Network Configuration Register	EMAC_NCFG	Read/Write	0x800
0x08	Network Status Register	EMAC_NSR	Read-only	–
0x0C	Reserved			
0x10	Reserved			
0x14	Transmit Status Register	EMAC_TSR	Read/Write	0x0000_0000
0x18	Receive Buffer Queue Pointer Register	EMAC_RBQP	Read/Write	0x0000_0000
0x1C	Transmit Buffer Queue Pointer Register	EMAC_TBQP	Read/Write	0x0000_0000
0x20	Receive Status Register	EMAC_RSR	Read/Write	0x0000_0000
0x24	Interrupt Status Register	EMAC_ISR	Read/Write	0x0000_0000
0x28	Interrupt Enable Register	EMAC_IER	Write-only	–
0x2C	Interrupt Disable Register	EMAC_IDR	Write-only	–
0x30	Interrupt Mask Register	EMAC_IMR	Read-only	0x0000_3FFF
0x34	Phy Maintenance Register	EMAC_MAN	Read/Write	0x0000_0000
0x38	Pause Time Register	EMAC_PTR	Read/Write	0x0000_0000
0x3C	Pause Frames Received Register	EMAC_PFR	Read/Write	0x0000_0000
0x40	Frames Transmitted Ok Register	EMAC_FTO	Read/Write	0x0000_0000
0x44	Single Collision Frames Register	EMAC_SCF	Read/Write	0x0000_0000
0x48	Multiple Collision Frames Register	EMAC_MCF	Read/Write	0x0000_0000
0x4C	Frames Received Ok Register	EMAC_FRO	Read/Write	0x0000_0000
0x50	Frame Check Sequence Errors Register	EMAC_FCSE	Read/Write	0x0000_0000
0x54	Alignment Errors Register	EMAC_ALE	Read/Write	0x0000_0000
0x58	Deferred Transmission Frames Register	EMAC_DTF	Read/Write	0x0000_0000
0x5C	Late Collisions Register	EMAC_LCOL	Read/Write	0x0000_0000
0x60	Excessive Collisions Register	EMAC_ECOL	Read/Write	0x0000_0000
0x64	Transmit Underrun Errors Register	EMAC_TUND	Read/Write	0x0000_0000
0x68	Carrier Sense Errors Register	EMAC_CSE	Read/Write	0x0000_0000
0x6C	Receive Resource Errors Register	EMAC_RRE	Read/Write	0x0000_0000
0x70	Receive Overrun Errors Register	EMAC_ROV	Read/Write	0x0000_0000
0x74	Receive Symbol Errors Register	EMAC_RSE	Read/Write	0x0000_0000
0x78	Excessive Length Errors Register	EMAC_ELE	Read/Write	0x0000_0000
0x7C	Receive Jabbers Register	EMAC_RJA	Read/Write	0x0000_0000
0x80	Undersize Frames Register	EMAC_USF	Read/Write	0x0000_0000
0x84	SQE Test Errors Register	EMAC_STE	Read/Write	0x0000_0000
0x88	Received Length Field Mismatch Register	EMAC_RLE	Read/Write	0x0000_0000

© 2017 Microchip Technology Inc.

Table 35-6: Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x90	Hash Register Bottom [31:0] Register	EMAC_HRB	Read/Write	0x0000_0000
0x94	Hash Register Top [63:32] Register	EMAC_HRT	Read/Write	0x0000_0000
0x98	Specific Address 1 Bottom Register	EMAC_SA1B	Read/Write	0x0000_0000
0x9C	Specific Address 1 Top Register	EMAC_SA1T	Read/Write	0x0000_0000
0xA0	Specific Address 2 Bottom Register	EMAC_SA2B	Read/Write	0x0000_0000
0xA4	Specific Address 2 Top Register	EMAC_SA2T	Read/Write	0x0000_0000
0xA8	Specific Address 3 Bottom Register	EMAC_SA3B	Read/Write	0x0000_0000
0xAC	Specific Address 3 Top Register	EMAC_SA3T	Read/Write	0x0000_0000
0xB0	Specific Address 4 Bottom Register	EMAC_SA4B	Read/Write	0x0000_0000
0xB4	Specific Address 4 Top Register	EMAC_SA4T	Read/Write	0x0000_0000
0xB8	Type ID Checking Register	EMAC_TID	Read/Write	0x0000_0000
0xC0	User Input/Output Register	EMAC_USRIO	Read/Write	0x0000_0000
0xC8–0xFC	Reserved	–	–	–

© 2017 Microchip Technology Inc.

35.5.1 Network Control Register

Name:EMAC_NCR

Access: 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
–	–	–	–	–	THALT	TSTART	BP
7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TE	RE	LLB	LB

LB: LoopBack

Asserts the loopback signal to the PHY.

LLB: Loopback local

Connects `txd` to `rxd`, `tx_en` to `rx_dv`, forces full duplex and drives `rx_clk` and `tx_clk` with `pclk` divided by 4. `rx_clk` and `tx_clk` may glitch as the EMAC is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back.

RE: Receive enable

When set, enables the EMAC to receive data. When reset, frame reception stops immediately and the receive FIFO is cleared. The receive queue pointer register is unaffected.

TE: Transmit enable

When set, enables the Ethernet transmitter to send data. When reset transmission, stops immediately, the transmit FIFO and control registers are cleared and the transmit queue pointer register resets to point to the start of the transmit descriptor list.

MPE: Management port enable

Set to one to enable the management port. When zero, forces MDIO to high impedance state and MDC low.

CLRSTAT: Clear statistics registers

This bit is write only. Writing a one clears the statistics registers.

INCSTAT: Increment statistics registers

This bit is write only. Writing a one increments all the statistics registers by one for test purposes.

WESTAT: Write enable for statistics registers

Setting this bit to one makes the statistics registers writable for functional test purposes.

BP: Back pressure

If set in half duplex mode, forces collisions on all received frames.

TSTART: Start transmission

Writing one to this bit starts transmission.

THALT: Transmit halt

Writing one to this bit halts transmission as soon as any ongoing frame transmission ends.

© 2017 Microchip Technology Inc.

35.5.2 Network Configuration Register

Name:EMAC_NCFG

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	IRXFCS	EFRHD	DRFCS	RLCE

15	14	13	12	11	10	9	8
RBOF		PAE	RTY	CLK		–	BIG

7	6	5	4	3	2	1	0
UNI	MTI	NBC	CAF	JFRAME	–	FD	SPD

SPD: Speed

Set to 1 to indicate 100 Mbit/s operation, 0 for 10 Mbit/s. The value of this pin is reflected on the `speed` pin.

FD: Full Duplex

If set to 1, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the `half_duplex` pin.

CAF: Copy All Frames

When set to 1, all valid frames are received.

JFRAME: Jumbo Frames

Set to one to enable jumbo frames of up to 10240 bytes to be accepted.

NBC: No Broadcast

When set to 1, frames addressed to the broadcast address of all ones are not received.

MTI: Multicast Hash Enable

When set, multicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

UNI: Unicast Hash Enable

When set, unicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

BIG: Receive 1536 bytes frames

Setting this bit means the EMAC receives frames up to 1536 bytes in length. Normally, the EMAC would reject any frame above 1518 bytes.

CLK: MDC clock divider

Set according to system clock speed. This determines by what number system clock is divided to generate MDC. For conformance with 802.3, MDC must not exceed 2.5MHz (MDC is only active during MDIO read and write operations).

CLK	MDC
00	MCK divided by 8 (MCK up to 20 MHz)
01	MCK divided by 16 (MCK up to 40 MHz)
10	MCK divided by 32 (MCK up to 80 MHz)
11	MCK divided by 64 (MCK up to 160 MHz)

© 2017 Microchip Technology Inc.

RTY: Retry test

Must be set to zero for normal operation. If set to one, the back off between collisions is always one slot time. Setting this bit to one helps testing the too many retries condition. Also used in the pause frame tests to reduce the pause counters decrement time from 512 bit times, to every `rx_clk` cycle.

PAE: Pause Enable

When set, transmission pauses when a valid pause frame is received.

RBOF: Receive Buffer Offset

Indicates the number of bytes by which the received data is offset from the start of the first receive buffer.

RBOF	Offset
00	No offset from start of receive buffer
01	One-byte offset from start of receive buffer
10	Two-byte offset from start of receive buffer
11	Three-byte offset from start of receive buffer

RLCE: Receive Length field Checking Enable

When set, frames with measured lengths shorter than their length fields are discarded. Frames containing a type ID in bytes 13 and 14 — length/type ID = 0600 — are not be counted as length errors.

DRFCS: Discard Receive FCS

When set, the FCS field of received frames are not be copied to memory.

EFRHD:

Enable Frames to be received in half-duplex mode while transmitting.

IRXFCS: Ignore RX FCS

When set, frames with FCS/CRC errors are not rejected and no FCS error statistics are counted. For normal operation, this bit must be set to 0.

© 2017 Microchip Technology Inc.

35.5.3 Network Status Register

Name:EMAC_NSR

Access: 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
–	–	–	–	–	IDLE	MDIO	–

MDIO

Returns status of the mdio_in pin. Use the PHY maintenance register for reading managed frames rather than this bit.

IDLE

0: The PHY logic is running.

1: The PHY management logic is idle (i.e., has completed).

35.5.4 Transmit Status Register

Name:EMAC_TSR

Access: 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
–	UND	COMP	BEX	TGO	RLE	COL	UBR

This register, when read, provides details of the status of a transmit. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

UBR: Used Bit Read

Set when a transmit buffer descriptor is read with its used bit set. Cleared by writing a one to this bit.

COL: Collision Occurred

Set by the assertion of collision. Cleared by writing a one to this bit.

RLE: Retry Limit exceeded

Cleared by writing a one to this bit.

TGO: Transmit Go

If high transmit is active.

BEX: Buffers exhausted mid frame

If the buffers run out during transmission of a frame, then transmission stops, FCS shall be bad and tx_er asserted. Cleared by writing a one to this bit.

COMP: Transmit Complete

Set when a frame has been transmitted. Cleared by writing a one to this bit.

UND: Transmit Underrun

Set when transmit DMA was not able to read data from memory, either because the bus was not granted in time, because a not OK `hresp(bus error)` was returned or because a used bit was read midway through frame transmission. If this occurs, the transmitter forces `bad` CRC. Cleared by writing a one to this bit.

© 2017 Microchip Technology Inc.

35.5.5 Receive Buffer Queue Pointer Register

Name:EMAC_RBQP

Access: Read/Write

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

This register points to the entry in the receive buffer queue (descriptor list) currently being used. It is written with the start location of the receive buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set.

Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits.

Receive buffer writes also comprise bursts of two words and, as with transmit buffer reads, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

ADDR: Receive buffer queue pointer address

Written with the address of the start of the receive queue, reads as a pointer to the current buffer being used.

35.5.6 Transmit Buffer Queue Pointer Register

Name:EMAC_TBQP

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register points to the entry in the transmit buffer queue (descriptor list) currently being used. It is written with the start location of the transmit buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set. This register can only be written when bit 3 in the transmit status register is low.

As transmit buffer reads consist of bursts of two words, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

ADDR: Transmit buffer queue pointer address

Written with the address of the start of the transmit queue, reads as a pointer to the first buffer of the frame being transmitted or about to be transmitted.

© 2017 Microchip Technology Inc.

35.5.7 Receive Status Register

Name:EMAC_RSR

Access: 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
–	–	–	–	–	OVR	REC	BNA

This register, when read, provides details of the status of a receive. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

BNA: Buffer Not Available

An attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA rereads the pointer each time a new frame starts until a valid pointer is found. This bit is set at each attempt that fails even if it has not had a successful pointer read since it has been cleared.

Cleared by writing a one to this bit.

REC: Frame Received

One or more frames have been received and placed in memory. Cleared by writing a one to this bit.

OVR: Receive Overrun

The DMA block was unable to store the receive frame to memory, either because the bus was not granted in time or because a not OK `hresp(bus error)` was returned. The buffer is recovered if this happens.

Cleared by writing a one to this bit.

35.5.8 Interrupt Status Register

Name:EMAC_ISR

Access: 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
–	–	PTZ	PFR	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

MFD: Management Frame Done

The PHY maintenance register has completed its operation. Cleared on read.

RCOMP: Receive Complete

A frame has been stored in memory. Cleared on read.

RXUBR: Receive Used Bit Read

Set when a receive buffer descriptor is read with its used bit set. Cleared on read.

TXUBR: Transmit Used Bit Read

Set when a transmit buffer descriptor is read with its used bit set. Cleared on read.

TUND: Ethernet Transmit Buffer Underrun

The transmit DMA did not fetch frame data in time for it to be transmitted or `hresp` returned not OK. Also set if a used bit is read mid-frame or when a new transmit queue pointer is written. Cleared on read.

RLE: Retry Limit Exceeded

Cleared on read.

TXERR: Transmit Error

Transmit buffers exhausted in mid-frame - transmit error. Cleared on read.

TCOMP: Transmit Complete

Set when a frame has been transmitted. Cleared on read.

ROVR: Receive Overrun

Set when the receive overrun status bit gets set. Cleared on read.

HRESP: Hresp not OK

Set when the DMA block sees a `bus error`. Cleared on read.

PFR: Pause Frame Received

Indicates a valid pause has been received. Cleared on a read.

PTZ: Pause Time Zero

Set when the pause time register, 0x38 decrements to zero. Cleared on a read.

© 2017 Microchip Technology Inc.

35.5.9 Interrupt Enable Register

Name:EMAC_IER

Access: 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
–	–	PTZ	PFR	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

MFD: Management Frame sent

Enable management done interrupt.

RCOMP: Receive Complete

Enable receive complete interrupt.

RXUBR: Receive Used Bit Read

Enable receive used bit read interrupt.

TXUBR: Transmit Used Bit Read

Enable transmit used bit read interrupt.

TUND: Ethernet Transmit Buffer Underrun

Enable transmit underrun interrupt.

RLE: Retry Limit Exceeded

Enable retry limit exceeded interrupt.

TXERR

Enable transmit buffers exhausted in mid-frame interrupt.

TCOMP: Transmit Complete

Enable transmit complete interrupt.

ROVR: Receive Overrun

Enable receive overrun interrupt.

HRESP: Hresp not OK

Enable Hresp not OK interrupt.

PFR: Pause Frame Received

Enable pause frame received interrupt.

PTZ: Pause Time Zero

Enable pause time zero interrupt.

© 2017 Microchip Technology Inc.

35.5.10 Interrupt Disable Register

Name:EMAC_IDR

Access: 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
–	–	PTZ	PFR	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

MFD: Management Frame sent

Disable management done interrupt.

RCOMP: Receive Complete

Disable receive complete interrupt.

RXUBR: Receive Used Bit Read

Disable receive used bit read interrupt.

TXUBR: Transmit Used Bit Read

Disable transmit used bit read interrupt.

TUND: Ethernet Transmit Buffer Underrun

Disable transmit underrun interrupt.

RLE: Retry Limit Exceeded

Disable retry limit exceeded interrupt.

TXERR

Disable transmit buffers exhausted in mid-frame interrupt.

TCOMP: Transmit Complete

Disable transmit complete interrupt.

ROVR: Receive Overrun

Disable receive overrun interrupt.

HRESP: Hresp not OK

Disable Hresp not OK interrupt.

PFR: Pause Frame Received

Disable pause frame received interrupt.

PTZ: Pause Time Zero

Disable pause time zero interrupt.

© 2017 Microchip Technology Inc.

35.5.11 Interrupt Mask Register

Name:EMAC_IMR

Access: 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
–	–	PTZ	PFR	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

MFD: Management Frame sent

Management done interrupt masked.

RCOMP: Receive Complete

Receive complete interrupt masked.

RXUBR: Receive Used Bit Read

Receive used bit read interrupt masked.

TXUBR: Transmit Used Bit Read

Transmit used bit read interrupt masked.

TUND: Ethernet Transmit Buffer Underrun

Transmit underrun interrupt masked.

RLE: Retry Limit Exceeded

Retry limit exceeded interrupt masked.

TXERR

Transmit buffers exhausted in mid-frame interrupt masked.

TCOMP: Transmit Complete

Transmit complete interrupt masked.

ROVR: Receive Overrun

Receive overrun interrupt masked.

HRESP: Hresp not OK

Hresp not OK interrupt masked.

PFR: Pause Frame Received

Pause frame received interrupt masked.

PTZ: Pause Time Zero

Pause time zero interrupt masked.

© 2017 Microchip Technology Inc.

35.5.12 PHY Maintenance Register

Name:EMAC_MAN

Access: Read/Write

31	30	29	28	27	26	25	24
SOF		RW		PHYA			

23	22	21	20	19	18	17	16
PHYA		REGA				CODE	

15	14	13	12	11	10	9	8
DATA							

7	6	5	4	3	2	1	0
DATA							

DATA

For a write operation this is written with the data to be written to the PHY.

After a read operation this contains the data read from the PHY.

CODE:

Must be written to 10. Reads as written.

REGA: Register Address

Specifies the register in the PHY to access.

PHYA: PHY Address

RW: Read/Write

10 is read; 01 is write. Any other value is an invalid PHY management frame

SOF: Start of frame

Must be written 01 for a valid frame.

© 2017 Microchip Technology Inc.

35.5.13 Pause Time Register

Name:EMAC_PTR

Access: 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
PTIME							

7	6	5	4	3	2	1	0
PTIME							

PTIME: Pause Time

Stores the current value of the pause time register which is decremented every 512 bit times.

35.5.14 Hash Register Bottom

Name:EMAC_HRB

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR:

Bits 31:0 of the hash address register. See “Hash Addressing”.

© 2017 Microchip Technology Inc.

35.5.15 Hash Register Top

Name:EMAC_HRT

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR:

Bits 63:32 of the hash address register. See “Hash Addressing”.

35.5.16 Specific Address 1 Bottom Register

Name:EMAC_SA1B

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

© 2017 Microchip Technology Inc.

35.5.17 Specific Address 1 Top Register

Name:EMAC_SA1T

Access: 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
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

The most significant bits of the destination address, that is bits 47 to 32.

35.5.18 Specific Address 2 Bottom Register

Name:EMAC_SA2B

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

© 2017 Microchip Technology Inc.

35.5.19 Specific Address 2 Top Register

Name:EMAC_SA2T

Access: 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
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

The most significant bits of the destination address, that is bits 47 to 32.

35.5.20 Specific Address 3 Bottom Register

Name:EMAC_SA3B

Access: Read/Write

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

ADDR

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

© 2017 Microchip Technology Inc.

35.5.21 Specific Address 3 Top Register

Name:EMAC_SA3T

Access: 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
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

The most significant bits of the destination address, that is bits 47 to 32.

35.5.22 Specific Address 4 Bottom Register

Name:EMAC_SA4B

Access: Read/Write

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

ADDR

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

© 2017 Microchip Technology Inc.

35.5.23 Specific Address 4 Top Register

Name:EMAC_SA4T

Access: 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
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

The most significant bits of the destination address, that is bits 47 to 32.

35.5.24 Type ID Checking Register

Name:EMAC_TID

Access: 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
TID							

7	6	5	4	3	2	1	0
TID							

TID: Type ID checking

For use in comparisons with received frames TypeID/Length field.

© 2017 Microchip Technology Inc.

35.5.25 User Input/Output Register

Name:EMAC_USRIO

Access: 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
–	–	–	–	–	–	CLKEN	RMII

RMII

When set, this bit enables the RMII operation mode. When reset, it selects the MII mode.

CLKEN

When set, this bit enables the transceiver input clock.

Setting this bit to 0 reduces power consumption when the treasurer is not used.

35.5.26 EMAC Statistic Registers

These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data. The receive statistics registers are only incremented when the receive enable bit is set in the network control register. To write to these registers, bit 7 must be set in the network control register. The statistics register block contains the following registers.

35.5.26.1 Pause Frames Received Register

Name:EMAC_PFR

Access: 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
FROK							

7	6	5	4	3	2	1	0
FROK							

FROK: Pause Frames Received OK

A 16-bit register counting the number of good pause frames received. A good frame has a length of 64 to 1518 (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

© 2017 Microchip Technology Inc.

35.5.26.2 Frames Transmitted OK Register

Name:EMAC_FTO

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
FTOK							

15	14	13	12	11	10	9	8
FTOK							

7	6	5	4	3	2	1	0
FTOK							

FTOK: Frames Transmitted OK

A 24-bit register counting the number of frames successfully transmitted, i.e., no underrun and not too many retries.

35.5.26.3 Single Collision Frames Register

Name:EMAC_SCF

Access: 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
SCF							

7	6	5	4	3	2	1	0
SCF							

SCF: Single Collision Frames

A 16-bit register counting the number of frames experiencing a single collision before being successfully transmitted, i.e., no underrun.

© 2017 Microchip Technology Inc.

35.5.26.4 Multicollision Frames Register

Name:EMAC_MCF

Access: 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
MCF							

7	6	5	4	3	2	1	0
MCF							

MCF: Multicollision Frames

A 16-bit register counting the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

© 2017 Microchip Technology Inc.

35.5.26.5 Frames Received OK Register

Name:EMAC_FRO

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
FROK							

15	14	13	12	11	10	9	8
FROK							

7	6	5	4	3	2	1	0
FROK							

FROK: Frames Received OK

A 24-bit register counting the number of good frames received, i.e., address recognized and successfully copied to memory. A good frame is of length 64 to 1518 bytes (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

© 2017 Microchip Technology Inc.

35.5.26.6 Frames Check Sequence Errors Register

Name:EMAC_FCSE

Access: 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
FCSE							

FCSE: Frame Check Sequence Errors

An 8-bit register counting frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes.

© 2017 Microchip Technology Inc.

35.5.26.7 Alignment Errors Register

Name:EMAC_ALE

Access: 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
ALE							

ALE: Alignment Errors

An 8-bit register counting frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes.

© 2017 Microchip Technology Inc.

35.5.26.8 Deferred Transmission Frames Register

Name:EMAC_DTF

Access: 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
DTF							

7	6	5	4	3	2	1	0
DTF							

DTF: Deferred Transmission Frames

A 16-bit register counting the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit underrun.

35.5.26.9 Late Collisions Register

Name:EMAC_LCOL

Access: 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
LCOL							

LCOL: Late Collisions

An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

© 2017 Microchip Technology Inc.

35.5.26.10 Excessive Collisions Register

Name:EMAC_ECOL

Access: 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
EXCOL							

EXCOL: Excessive Collisions

An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

35.5.26.11 Transmit Underrun Errors Register

Name:EMAC_TUND

Access: 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
TUND							

TUND: Transmit Underruns

An 8-bit register counting the number of frames not transmitted due to a transmit DMA underrun. If this register is incremented, then no other statistics register is incremented.

© 2017 Microchip Technology Inc.

35.5.26.12 Carrier Sense Errors Register

Name:EMAC_CSE

Access: 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
CSE							

CSE: Carrier Sense Errors

An 8-bit register counting the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no underrun). Only incremented in half-duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

© 2017 Microchip Technology Inc.

35.5.26.13 Receive Resource Errors Register

Name:EMAC_RRE

Access: 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
RRE							

7	6	5	4	3	2	1	0
RRE							

RRE: Receive Resource Errors

A 16-bit register counting the number of frames that were address matched but could not be copied to memory because no receive buffer was available.

© 2017 Microchip Technology Inc.

35.5.26.14 Receive Overrun Errors Register

Name:EMAC_ROV

Access: 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
ROVR							

ROVR: Receive Overrun

An 8-bit register counting the number of frames that are address recognized but were not copied to memory due to a receive DMA overrun.

35.5.26.15 Receive Symbol Errors Register

Name:EMAC_RSE

Access: 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
RSE							

RSE: Receive Symbol Errors

An 8-bit register counting the number of frames that had `rx_er` asserted during reception. Receive symbol errors are also counted as an FCS or alignment error if the frame is between 64 and 1518 bytes in length (1536 if bit 8 is set in the network configuration register). If the frame is larger, it is recorded as a jabber error.

© 2017 Microchip Technology Inc.

35.5.26.16 Excessive Length Errors Register

Name:EMAC_ELE

Access: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
EXL							

EXL: Excessive Length Errors

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length but do not have either a CRC error, an alignment error nor a receive symbol error.

35.5.26.17 Receive Jabbers Register

Name:EMAC_RJA

Access: 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
RJB							

RJB: Receive Jabbers

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error.

© 2017 Microchip Technology Inc.

35.5.26.18 Undersize Frames Register

Name:EMAC_USF

Access: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
USF							

USF: Undersize frames

An 8-bit register counting the number of frames received less than 64 bytes in length but do not have either a CRC error, an alignment error or a receive symbol error.

35.5.26.19 SQE Test Errors Register

Name:EMAC_STE

Access: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
SQER							

SQER: SQE test errors

An 8-bit register counting the number of frames where `col` was not asserted within 96 bit times (an interframe gap) of `tx_en` being deasserted in half duplex mode.

© 2017 Microchip Technology Inc.

35.5.26.20 Received Length Field Mismatch Register

Name:EMAC_RLE

Access: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
RLFM							

RLFM: Receive Length Field Mismatch

An 8-bit register counting the number of frames received that have a measured length shorter than that extracted from its length field. Checking is enabled through bit 16 of the network configuration register. Frames containing a type ID in bytes 13 and 14 (i.e., length/type ID ≥ 0x0600) are not counted as length field errors, neither are excessive length frames.

36. USB Device Port (UDP)

36.1 Overview

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1 Mbyte/s) by working with endpoints with two banks of DPR.

Table 36-1: USB Endpoint Description

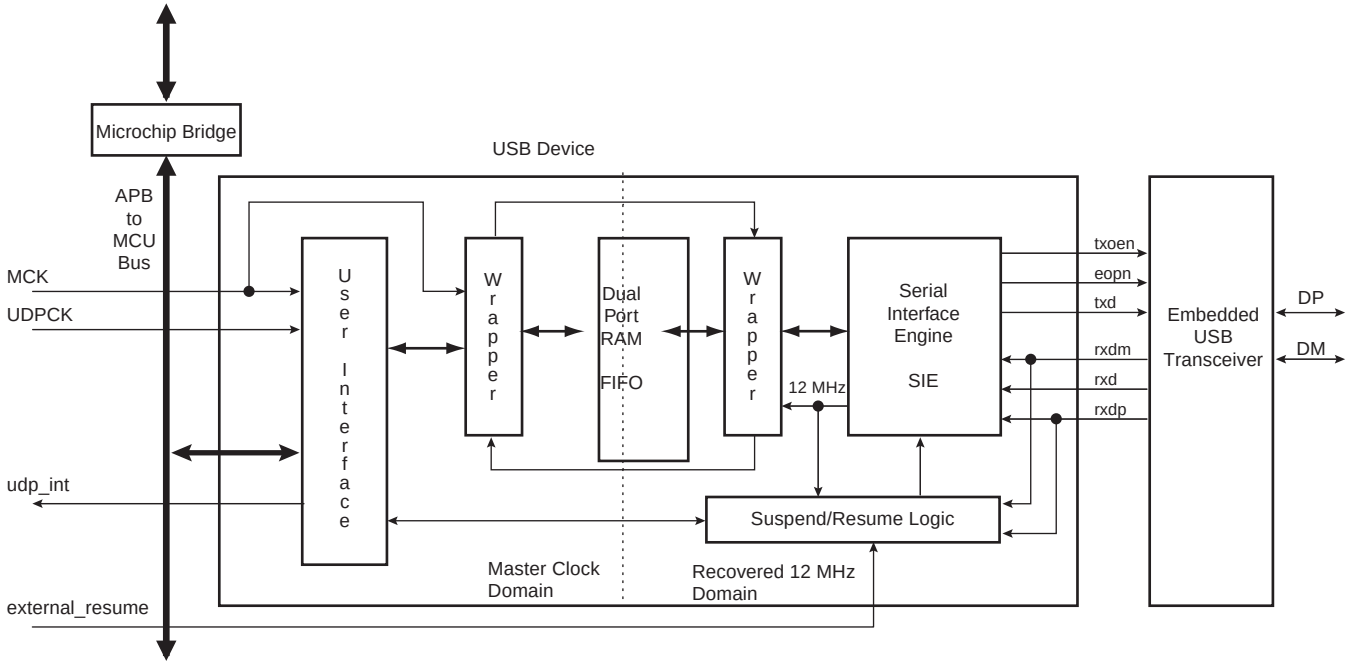
Endpoint Number	Mnemonic	Dual-Bank(1)	Max. Endpoint Size	Endpoint Type
0	EP0	No	64	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	512	Bulk/Iso/Interrupt
5	EP5	Yes	512	Bulk/Iso/Interrupt

Note 1: The Dual-Bank function provides two banks for an endpoint. This feature is used for ping-pong mode.

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake up to the USB host controller.

36.2 Block Diagram

Figure 36-1: Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the Master Clock domain (MCK) and a 48 MHz clock (UDPCK) used by the 12 MHz domain.

© 2017 Microchip Technology Inc.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

The signal external_resume is optional. It allows the UDP peripheral to wake up once in system mode. The host is then notified that the device asks for a resume. This optional feature must be also negotiated with the host during the enumeration.

36.3 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DP and DM are available from the product boundary.

One I/O line may be used by the application to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the pullup on DP must be disabled in order to prevent feeding current to the host. The application should disconnect the transceiver, then remove the pullup.

36.3.1 I/O Lines

DP and DM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

36.3.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of ± 0.25%.

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface, and the UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain).

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXCV register.

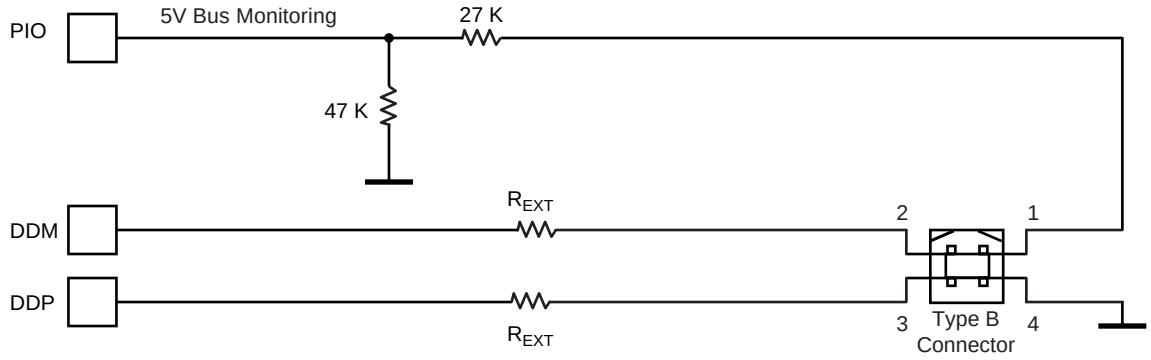
36.3.3 Interrupt

The USB device interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the USB device interrupt requires programming the AIC before configuring the UDP.

36.4 Typical Connection

Figure 36-2: Board Schematic to Interface Device Peripheral



36.4.1 USB Device Transceiver

The USB device transceiver is embedded in the product. A few discrete components are required as follows:

- the application detects all device states as defined in chapter 9 of the USB specification;
  - VBUS monitoring
- to reduce power consumption the host is disconnected
- for line termination.

36.4.2 VBUS Monitoring

VBUS monitoring is required to detect host connection. VBUS monitoring is done using a standard PIO with internal pullup disabled. When the host is switched off, it should be considered as a disconnect, the pullup must be disabled in order to prevent powering the host through the pull-up resistor.

When the host is disconnected and the transceiver is enabled, then DDP and DDM are floating. This may lead to over consumption. A solution is to enable the integrated pulldown by disabling the transceiver (TXVDIS = 1) and then remove the pullup (PUON = 0).

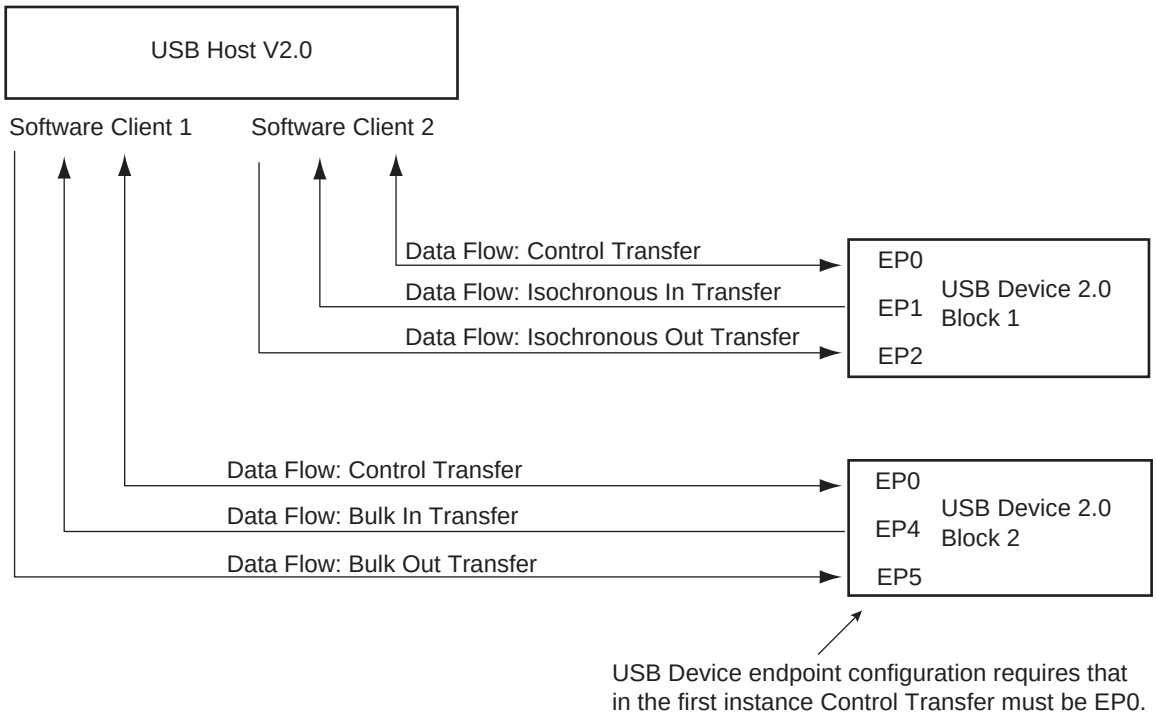
A termination serial resistor must be connected to DP and DM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

36.5 Functional Description

36.5.1 USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB device through a set of communication flows.

Figure 36-3: Example of USB V2.0 Full-speed Communication Control



The Control Transfer endpoint EP0 is always used when a USB device is first configured (USB v. 2.0 specifications).

© 2017 Microchip Technology Inc.

36.5.1.1 USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 36-2: USB Communication Flow

Transfer	Direction	Bandwidth	Supported Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	512	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤ 64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

36.5.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are three kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction

36.5.1.3 USB Transfer Event Definitions

As indicated below, transfers are sequential events carried out on the USB bus.

Table 36-3: USB Transfer Events

Transfer		Transaction
Direction	**Type**	
CONTROL (bidirectional)	Control(1)(3)	Setup transaction → Data IN transactions → Status OUT transaction
		Setup transaction → Data OUT transactions → Status IN transaction
		Setup transaction → Status IN transaction
IN (device toward host)	Interrupt IN	Data IN transaction → Data IN transaction
	Isochronous IN(2)	
	Bulk IN	
OUT (host toward device)	Interrupt OUT	Data OUT transaction → Data OUT transaction
	Isochronous OUT(2)	
	Bulk OUT	

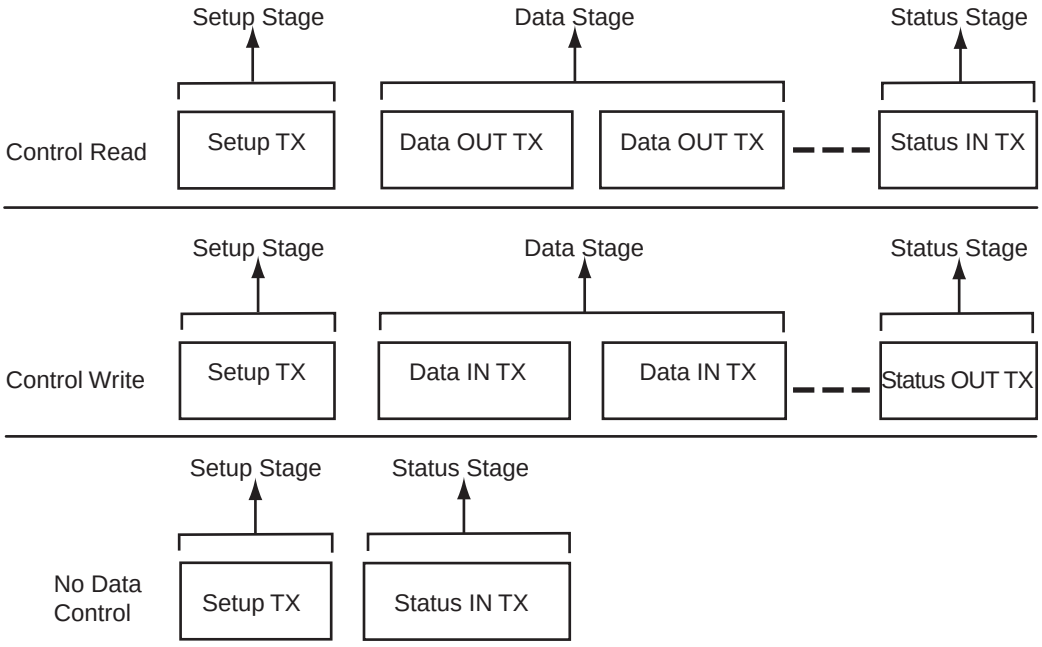
Note 1: Control transfer must use endpoints with no ping-pong attributes.

2: Isochronous transfers must use endpoints with ping-pong attributes.

3: Control transfers can be aborted using a stall handshake.

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

Figure 36-4: Control Read and Write Sequences



Note 1: During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.

2: During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

36.5.2 Handling Transactions with USB V2.0 Device Peripheral

36.5.2.1 Setup Transaction

Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

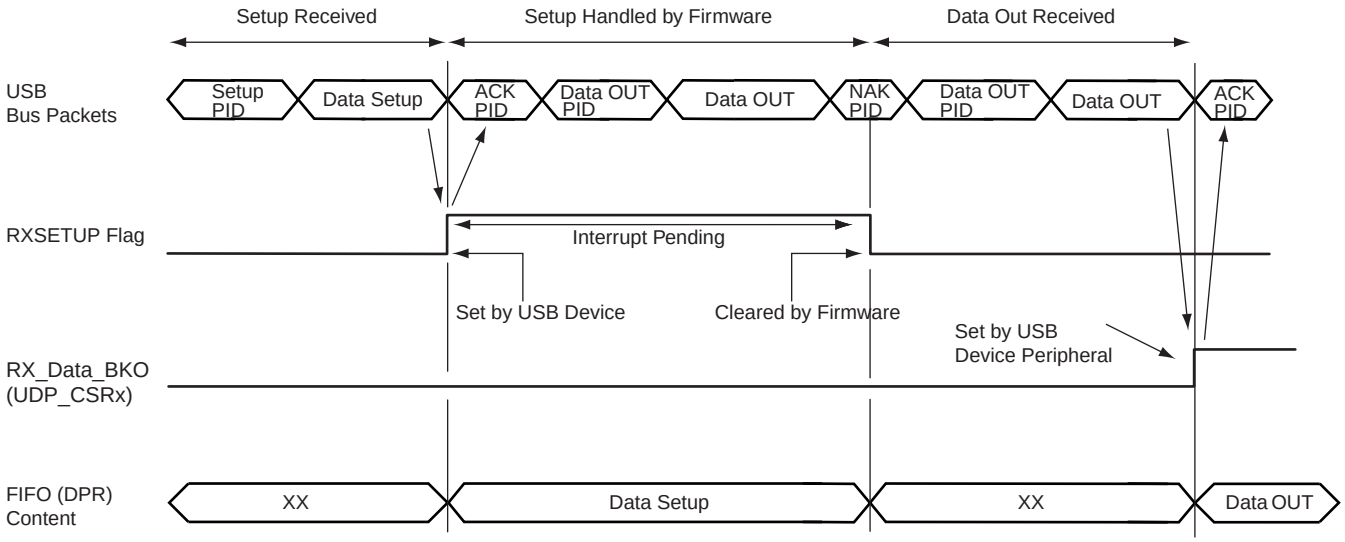
When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet?
- RXSETUP is set in the UDP_CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

© 2017 Microchip Technology Inc.

Figure 36-5: Setup Transaction Followed by a Data OUT Transaction



36.5.2.2 Data IN Transaction

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

- Using Endpoints Without Ping-pong Attributes

To perform a Data IN transaction using a non ping-pong endpoint:

1. The application checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_CSRx register (TXPKTRDY must be cleared).
2. The application writes the first packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
3. The application notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. The application is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.
5. The microcontroller writes the second packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
6. The microcontroller notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
7. The application clears the TXCOMP in the endpoint's UDP_CSRx.

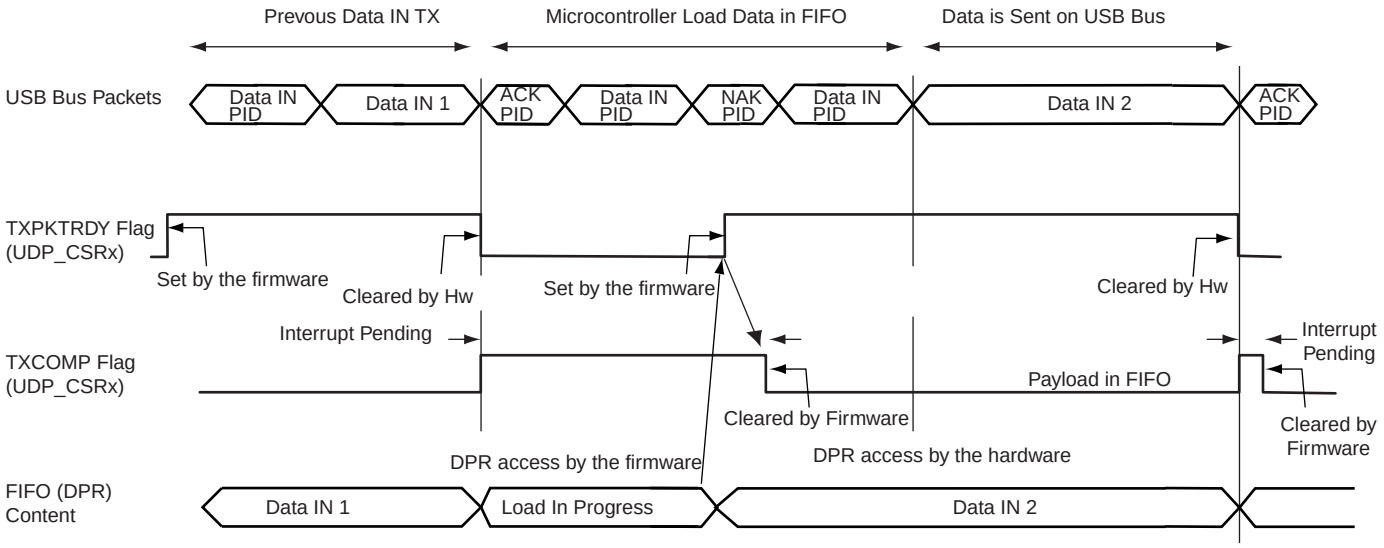
After the last packet has been sent, the application must clear TXCOMP once this has been set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

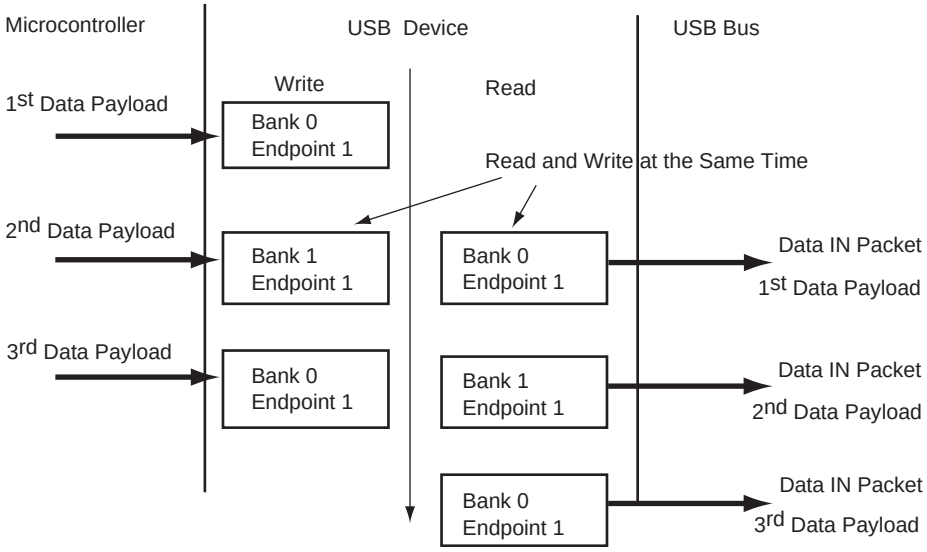
Figure 36-6: Data IN Transfer for Non Ping-pong Endpoint



- Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. This also allows handling the maximum bandwidth defined in the USB specification during bulk transfer. To be able to guarantee a constant or the maximum bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 36-7: Bank Swapping Data IN Transfer for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

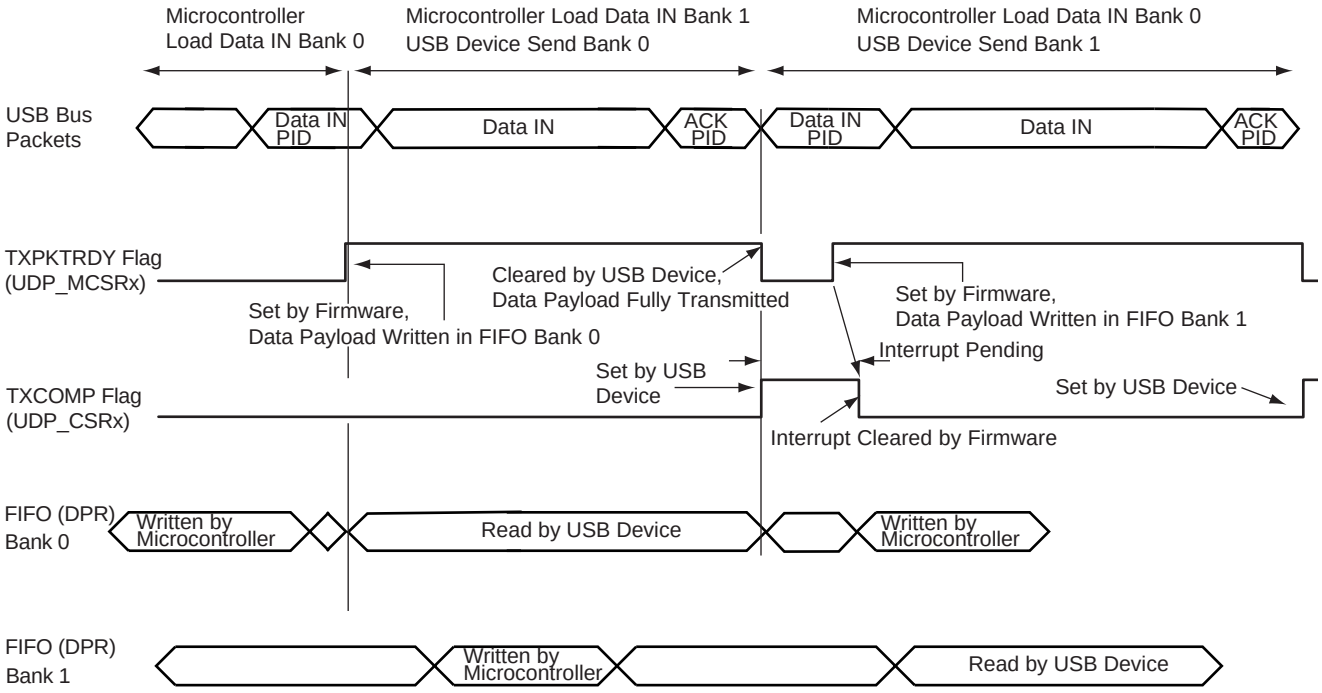
1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1),

© 2017 Microchip Technology Inc.

writing zero or more byte values in the endpoint's UDP_FDRx register.

5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent, raising TXPKTRDY in the endpoint's UDP_CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 36-8: Data IN Transfer for Ping-pong Endpoint



Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set too long, some Data IN packets may be NACKed, reducing the bandwidth.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

36.5.2.3 Data OUT Transaction

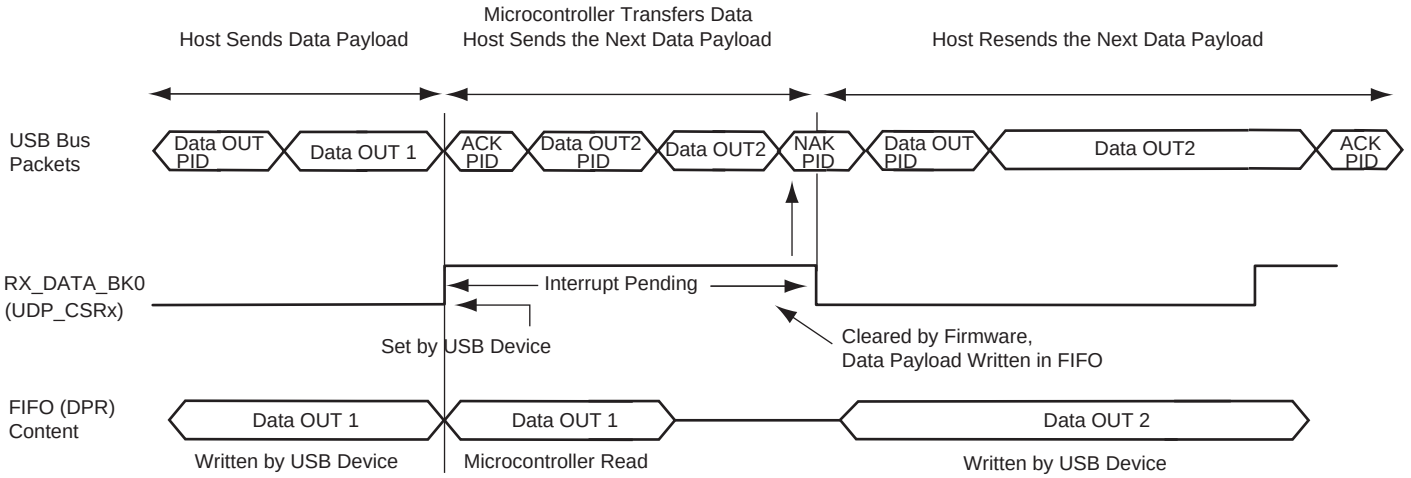
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

- Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

Figure 36-9: Data OUT Transfer for Non Ping-pong Endpoints

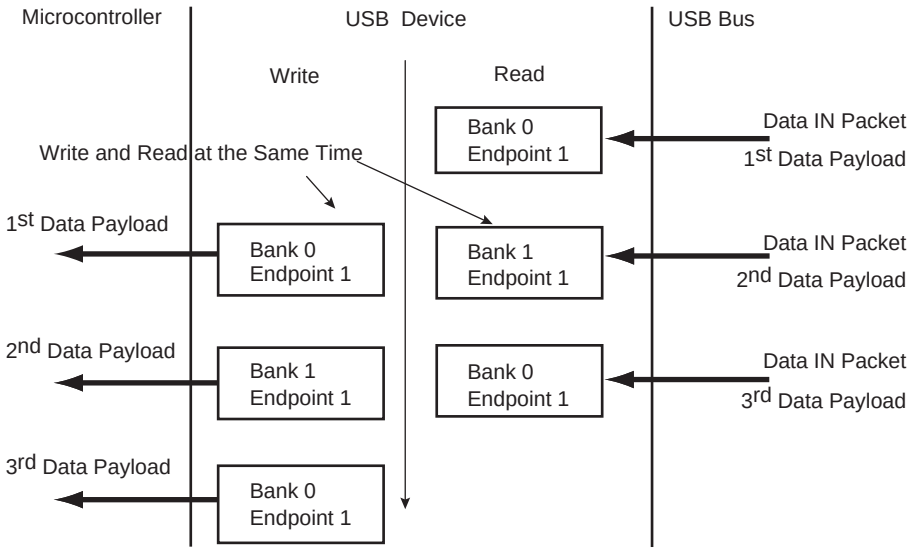


An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

- Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 36-10: Bank Swapping in Data OUT Transfers for Ping-pong Endpoints



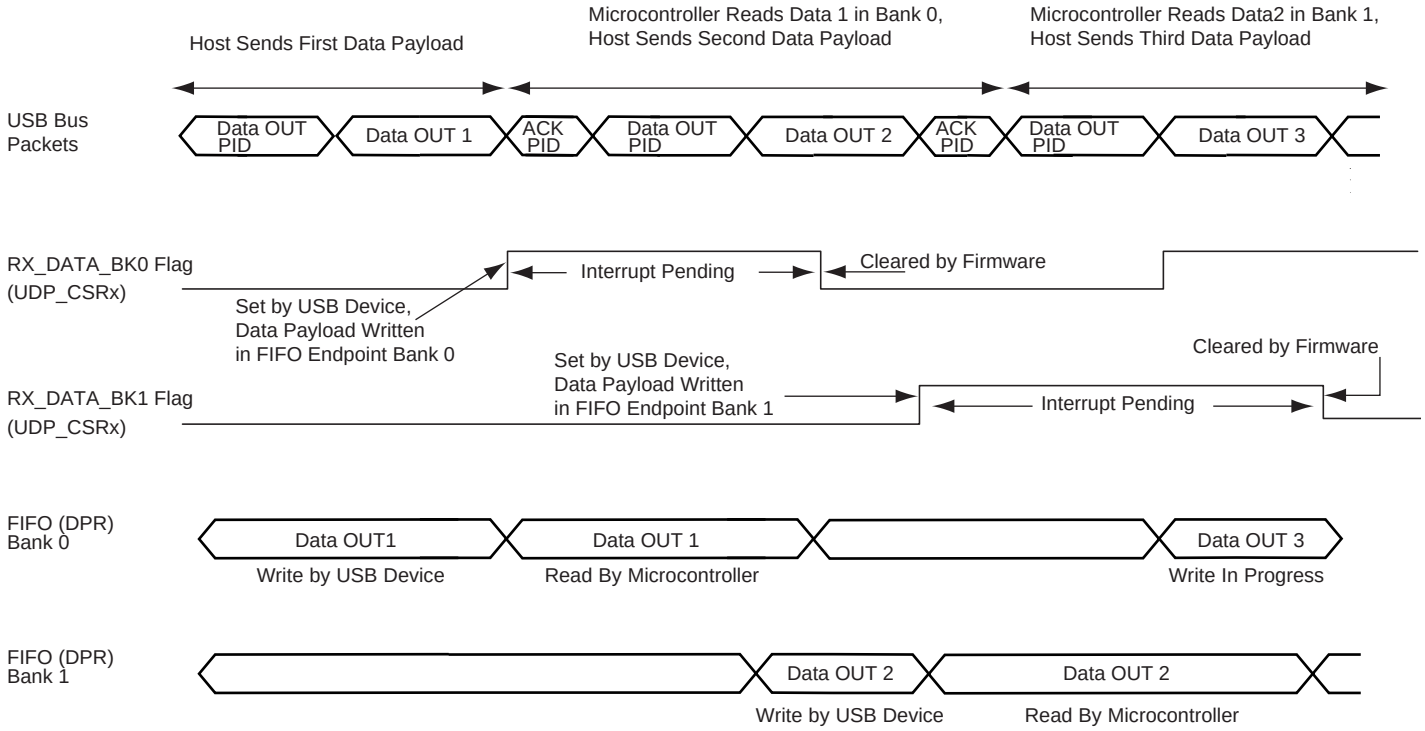
When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.

© 2017 Microchip Technology Inc.

6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.
10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_FDRx register.
11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

Figure 36-11: Data OUT Transfer for Ping-pong Endpoint



Note: An interrupt is pending while the RX_DATA_BK0 or RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

36.5.2.4 Stall Handshake

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0.*)

- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_CSRx endpoint's register.
2. The host receives the stall packet.
3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 36-12: Stall Handshake (Data IN Transfer)

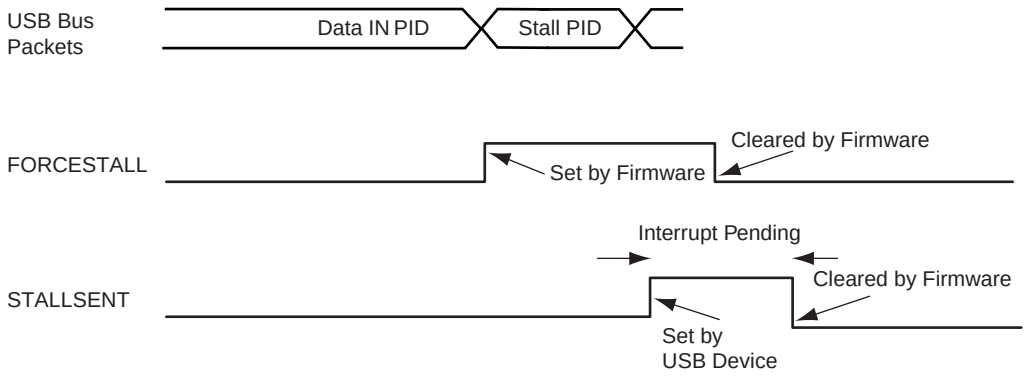
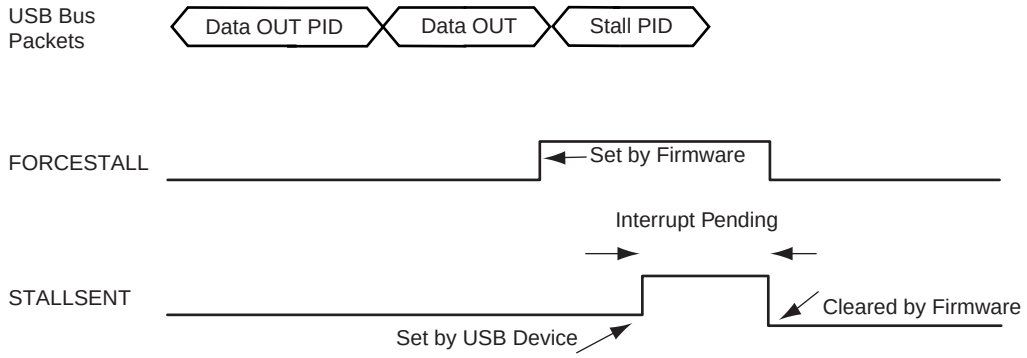


Figure 36-13: Stall Handshake (Data OUT Transfer)



36.5.2.5 Transmit Data Cancellation

Some endpoints have dual-banks whereas some endpoints have only one bank. The procedure to cancel transmission data held in these banks is described below.

To see the organization of dual-bank availability refer to Table 36-1 "USB Endpoint Description".

- Endpoints Without Dual-Banks

There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 36.6.9 "UDP Reset Endpoint Register".)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY so that no packet is ready to be sent
  - Reset the endpoint to clear the FIFO (pointers). (See Section 36.6.9 "UDP Reset Endpoint Register".)
- Endpoints With Dual-Banks

There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See Section 36.6.9 "UDP Reset Endpoint Register".)

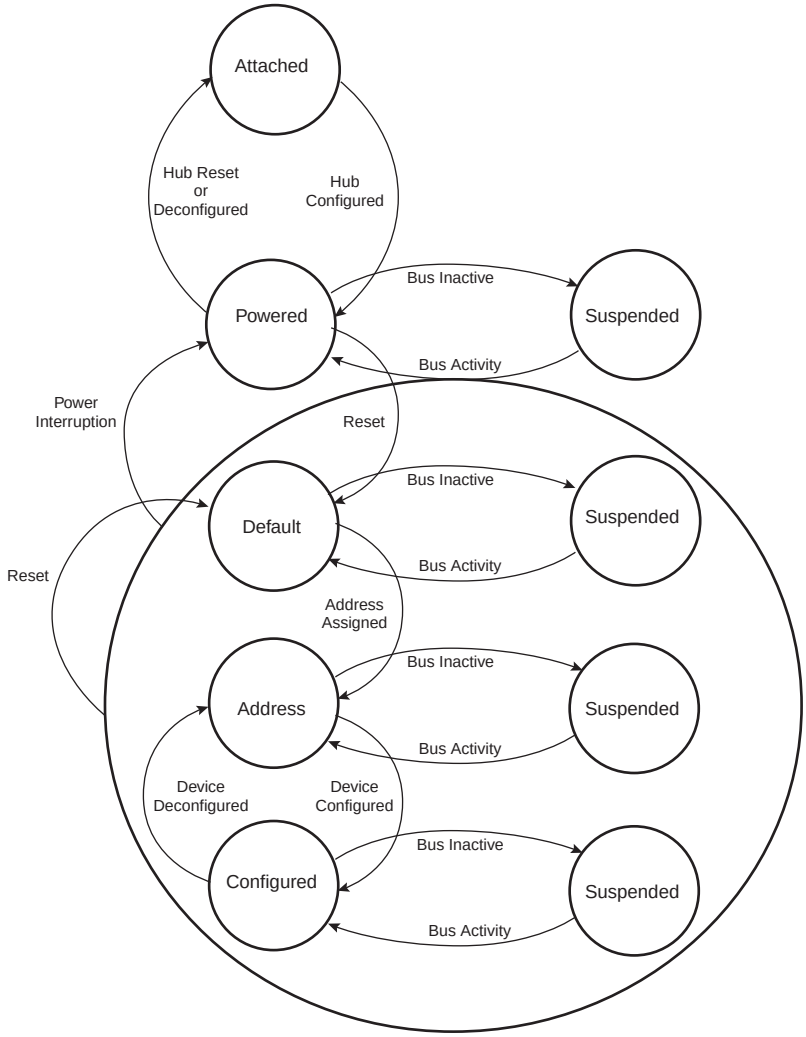
© 2017 Microchip Technology Inc.

- TXPKTRDY has already been set:
  - Clear TXPKTRDY and read it back until actually read at 0.
  - Set TXPKTRDY and read it back until actually read at 1.
  - Clear TXPKTRDY so that no packet is ready to be sent.
  - Reset the endpoint to clear the FIFO (pointers). (See Section 36.6.9 "UDP Reset Endpoint Register".)

36.5.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 36-14: USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 µA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake up feature is not mandatory for all devices and must be negotiated with the host.

36.5.3.1 Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

36.5.3.2 Entering Attached State

When no device is connected, the USB DP and DM signals are tied to GND by 15 KΩ pull-down resistors integrated in the hub downstream ports. When a device is attached to a hub downstream port, the device connects a 1.5 KΩ pull-up resistor on DP. The USB bus line goes into IDLE state, DP is pulled up by the device 1.5 KΩ resistor to 3.3V and DM is pulled down by the 15 KΩ resistor of the host. To enable integrated pullup, the PUON bit in the UDP_TXVC register must be set.

Warning: To write to the UDP_TXVC register, MCK clock must be enabled on the UDP. This is done in the Power Management Controller.

After pullup connection, the device enters the powered state. In this state, the UDPCK and MCK must be enabled in the Power Management Controller. The transceiver can remain disabled.

36.5.3.3 From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmaskable flag ENDBUSRES is set in the register UDP_ISR and an interrupt is triggered.

Once the ENDBUSRES interrupt has been triggered, the device enters Default State. In this state, the UDP software must:

- Enable the default endpoint, setting the EPEDS flag in the UDP_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP_IER. The enumeration then begins by a control transfer.
- Configure the interrupt mask register which has been reset by the USB reset detection
- Enable the transceiver clearing the TXVDIS flag in the UDP_TXVC register.

In this state UDPCK and MCK must be enabled.

Warning: Each time an ENDBUSRES interrupt is triggered, the Interrupt Mask Register and UDP_CSR registers have been reset.

36.5.3.4 From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state.

Warning: Before the device enters in address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP_GLB_STAT register, sets its new address, and sets the FEN bit in the UDP_FADDR.

36.5.3.5 From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx registers and, optionally, enabling corresponding interrupts in the UDP_IER.

36.5.3.6 Entering in Suspend State

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR.This flag is cleared by writing to the UDP_ICR. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500uA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks can be switched off. Resume event is asynchronously detected. MCK and UDPCK can be switched off in the Power Management controller and the USB transceiver can be disabled by setting the TXVDIS field in the UDP_TXVC register.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. Switching off MCK for the UDP peripheral must be one of the last operations after writing to the UDP_TXVC and acknowledging the RXSUSP.

36.5.3.7 Receiving a Host Resume

In suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks are disabled (however the pullup shall not be removed).

Once the resume is detected on the bus, the WAKEUP signal in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR is set. This interrupt may be used to wake up the core, enable PLL and main oscillators and configure clocks.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. MCK for the UDP must be enabled before clearing the WAKEUP bit in the UDP_ICR and clearing TXVDIS in the UDP_TXVC register.

© 2017 Microchip Technology Inc.

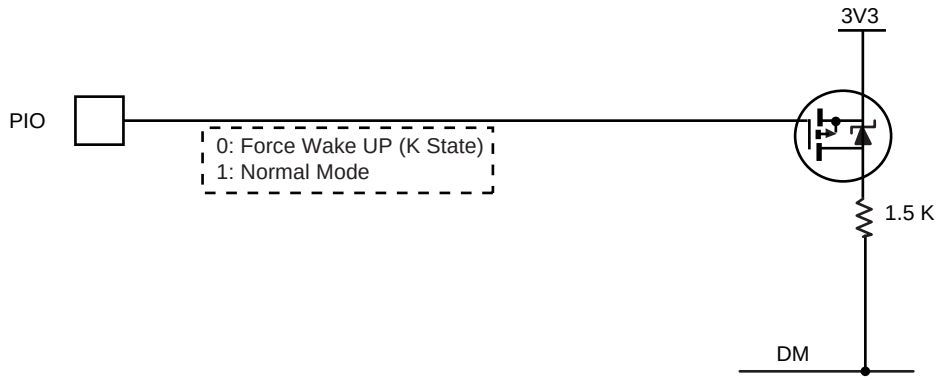
36.5.3.8 Sending a Device Remote Wakeup

In Suspend state it is possible to wake up the host sending an external resume.

- The device must wait at least 5 ms after being entered in suspend before sending an external resume.
- The device has 10 ms from the moment it starts to drain current and it forces a K state to resume the host.
- The device must force a K state from 1 to 15 ms to resume the host

To force a K state to the bus (DM at 3.3V and DP tied to GND), it is possible to use a transistor to connect a pullup on DM. The K state is obtained by disabling the pullup on DP and enabling the pullup on DM. This should be under the control of the application.

Figure 36-15: Board Schematic to Drive a K State



36.6 USB Device (UDP) User Interface

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers, including the UDP_TXVC register.

Table 36-4: Register Mapping

Offset	Register	Name	Access	Reset
0x000	Frame Number Register	UDP_FRM_NUM	Read-only	0x0000_0000
0x004	Global State Register	UDP_GLB_STAT	Read/Write	0x0000_0000
0x008	Function Address Register	UDP_FADDR	Read/Write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_IER	Write-only	
0x014	Interrupt Disable Register	UDP_IDR	Write-only	
0x018	Interrupt Mask Register	UDP_IMR	Read-only	0x0000_1200
0x01C	Interrupt Status Register	UDP_ISR	Read-only	–(1)
0x020	Interrupt Clear Register	UDP_ICR	Write-only	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_RST_EP	Read/Write	0x0000_0000
0x02C	Reserved	–	–	–
0x030 + 0x4 * (ept_num - 1)	Endpoint Control and Status Register	UDP_CSR	Read/Write	0x0000_0000
0x050 + 0x4 * (ept_num - 1)	Endpoint FIFO Data Register	UDP_FDR	Read/Write	0x0000_0000
0x070	Reserved	–	–	–
0x074	Transceiver Control Register	UDP_TXVC(2)	Read/Write	0x0000_0100
0x078–0xFC	Reserved	–	–	–

Note 1: Reset values are not defined for UDP_ISR.

2: See Warning above the "Register Mapping" on this page.

© 2017 Microchip Technology Inc.

36.6.1 UDP Frame Number Register

Name:UDP_FRM_NUM

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	FRM_OK	FRM_ERR

15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		

7	6	5	4	3	2	1	0
FRM_NUM							

FRM_NUM[10:0]: Frame Number as Defined in the Packet Field Formats

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value Updated at the SOF_EOP (Start of Frame End of Packet).

FRM_ERR: Frame Error

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

FRM_OK: Frame OK

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

© 2017 Microchip Technology Inc.

36.6.2 UDP Global State Register

Name:UDP_GLB_STAT

Access: 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
–	–	–	–	RSMINPR	–	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

FADDEN: Function Address Enable

Read:

0: Device is not in address state.

1: Device is in address state.

Write:

0: No effect, only a reset can bring back a device to the default state.

1: Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_FADDR must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

CONFG: Configured

Read:

0: Device is not in configured state.

1: Device is in configured state.

Write:

0: Sets device in a non configured state

1: Sets device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

RSMINPR: Resume Interrupt Request

Read:

0: No effect.

1: The pin “send_resume” is set to one. A Send Resume request has been detected and the device can send a Remote Wake Up.

© 2017 Microchip Technology Inc.

36.6.3 UDP Function Address Register

Name:UDP_FADDR

Access: 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
–	–	–	–	–	–	–	FEN
7	6	5	4	3	2	1	0
–	FADD						

FADD[6:0]: Function Address Value

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

FEN: Function Enable

Read:

0: Function endpoint disabled.

1: Function endpoint enabled.

Write:

0: Disables function endpoint.

1: Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

© 2017 Microchip Technology Inc.

36.6.4 UDP Interrupt Enable Register

Name:UDP_IER

Access: 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
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Enable Endpoint 0 Interrupt

EP1INT: Enable Endpoint 1 Interrupt

EP2INT: Enable Endpoint 2Interrupt

EP3INT: Enable Endpoint 3 Interrupt

EP4INT: Enable Endpoint 4 Interrupt

EP5INT: Enable Endpoint 5 Interrupt

0: No effect.

1: Enables corresponding Endpoint Interrupt.

RXSUSP: Enable UDP Suspend Interrupt

0: No effect.

1: Enables UDP Suspend Interrupt.

RXRSM: Enable UDP Resume Interrupt

0: No effect.

1: Enables UDP Resume Interrupt.

EXTRSM: Enable External Resume Interrupt

0: No effect.

1: Enables External Resume Interrupt.

SOFINT: Enable Start Of Frame Interrupt

0: No effect.

1: Enables Start Of Frame Interrupt.

WAKEUP: Enable UDP bus Wakeup Interrupt

0: No effect.

1: Enables USB bus Interrupt.

© 2017 Microchip Technology Inc.

36.6.5 UDP Interrupt Disable Register

Name:UDP_IDR

Access: 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
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Disable Endpoint 0 Interrupt

EP1INT: Disable Endpoint 1 Interrupt

EP2INT: Disable Endpoint 2 Interrupt

EP3INT: Disable Endpoint 3 Interrupt

EP4INT: Disable Endpoint 4 Interrupt

EP5INT: Disable Endpoint 5 Interrupt

0: No effect.

1: Disables corresponding Endpoint Interrupt.

RXSUSP: Disable UDP Suspend Interrupt

0: No effect.

1: Disables UDP Suspend Interrupt.

RXRSM: Disable UDP Resume Interrupt

0: No effect.

1: Disables UDP Resume Interrupt.

EXTRSM: Disable External Resume Interrupt

0: No effect.

1: Disables External Resume Interrupt.

SOFINT: Disable Start Of Frame Interrupt

0: No effect.

1: Disables Start Of Frame Interrupt

WAKEUP: Disable USB Bus Interrupt

0: No effect.

1: Disables USB Bus Wakeup Interrupt.

© 2017 Microchip Technology Inc.

36.6.6 UDP Interrupt Mask Register

Name:UDP_IMR

Access: 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
–	–	WAKEUP	BIT12	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Mask Endpoint 0 Interrupt

EP1INT: Mask Endpoint 1 Interrupt

EP2INT: Mask Endpoint 2 Interrupt

EP3INT: Mask Endpoint 3 Interrupt

EP4INT: Mask Endpoint 4 Interrupt

EP5INT: Mask Endpoint 5 Interrupt

0: Corresponding Endpoint Interrupt is disabled.

1: Corresponding Endpoint Interrupt is enabled.

RXSUSP: Mask UDP Suspend Interrupt

0: UDP Suspend Interrupt is disabled.

1: UDP Suspend Interrupt is enabled.

RXRSM: Mask UDP Resume Interrupt.

0: UDP Resume Interrupt is disabled.

1: UDP Resume Interrupt is enabled.

EXTRSM: Mask External Resume Interrupt

0: UDP External Resume Interrupt is disabled.

1: UDP External Resume Interrupt is enabled.

SOFINT: Mask Start Of Frame Interrupt

0: Start of Frame Interrupt is disabled.

1: Start of Frame Interrupt is enabled.

BIT12: UDP_IMR Bit 12

Bit 12 of UDP_IMR cannot be masked and is always read at 1.

WAKEUP: USB Bus WAKEUP Interrupt

0: USB Bus Wakeup Interrupt is disabled.

1: USB Bus Wakeup Interrupt is enabled.

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_IMR is enabled.

© 2017 Microchip Technology Inc.

36.6.7 UDP Interrupt Status Register

Name:UDP_ISR

Access: 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
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

EP0INT: Endpoint 0 Interrupt Status

EP1INT: Endpoint 1 Interrupt Status

EP2INT: Endpoint 2 Interrupt Status

EP3INT: Endpoint 3 Interrupt Status

EP4INT: Endpoint 4 Interrupt Status

EP5INT: Endpoint 5 Interrupt Status

0: No Endpoint0 Interrupt pending.

1: Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_CSR0 bit.

RXSUSP: UDP Suspend Interrupt Status

0: No UDP Suspend Interrupt pending.

1: UDP Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

RXRSM: UDP Resume Interrupt Status

0: No UDP Resume Interrupt pending.

1:UDP Resume Interrupt has been raised.

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ICR.

EXTRSM: UDP External Resume Interrupt Status

0: No UDP External Resume Interrupt pending.

1: UDP External Resume Interrupt has been raised.

© 2017 Microchip Technology Inc.

SOFINT: Start of Frame Interrupt Status

0: No Start of Frame Interrupt pending.

1: Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

ENDBUSRES: End of BUS Reset Interrupt Status

0: No End of Bus Reset Interrupt pending.

1: End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

WAKEUP: UDP Resume Interrupt Status

0: No Wakeup Interrupt pending.

1: A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined, the application must clear this bit by setting the WAKEUP flag in the UDP_ICR.

© 2017 Microchip Technology Inc.

36.6.8 UDP Interrupt Clear Register

Name:UDP_ICR

Access: 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
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

RXSUSP: Clear UDP Suspend Interrupt

0: No effect.

1: Clears UDP Suspend Interrupt.

RXRSM: Clear UDP Resume Interrupt

0: No effect.

1: Clears UDP Resume Interrupt.

EXTRSM: Clear UDP External Resume Interrupt

0: No effect.

1: Clears UDP External Resume Interrupt.

SOFINT: Clear Start Of Frame Interrupt

0: No effect.

1: Clears Start Of Frame Interrupt.

ENDBUSRES: Clear End of Bus Reset Interrupt

0: No effect.

1: Clears End of Bus Reset Interrupt.

WAKEUP: Clear Wakeup Interrupt

0: No effect.

1: Clears Wakeup Interrupt.

© 2017 Microchip Technology Inc.

36.6.9 UDP Reset Endpoint Register

Name:UDP_RST_EP

Access: 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
–	–	EP5	EP4	EP3	EP2	EP1	EP0

EP0: Reset Endpoint 0

EP1: Reset Endpoint 1

EP2: Reset Endpoint 2

EP3: Reset Endpoint 3

EP4: Reset Endpoint 4

EP5: Reset Endpoint 5

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP_CSRx.It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0: No reset.

1: Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx register.

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

© 2017 Microchip Technology Inc.

36.6.10 UDP Endpoint Control and Status Register

Name:UDP_CSRx [x = 0..5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCESTALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

```
//! Clear flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_clr_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] &= ~(flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) == (flags) ); \
   }
//! Set flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_set_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] |= (flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) != (flags) ); \
   }
```

Note: In a preemptive environment, set or clear the flag and wait for a time of 1 UDPCK clock cycle and 1peripheral clock cycle. However, RX_DATA_BLK0, TXPKTRDY, RX_DATA_BK1 require wait times of 3 UDPCK clock cycles and 3 peripheral clock cycles before accessing DPR.

TXCOMP: Generates an IN Packet with Data Previously Written in the DPR

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0: Clear the flag, clear the interrupt.

1: No effect.

Read (Set by the USB peripheral):

0: Data IN transaction has not been acknowledged by the Host.

1: Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

© 2017 Microchip Technology Inc.

RX_DATA_BK0: Receive Data Bank 0

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0: Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1: To leave the read value unchanged.

Read (Set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 0.

1: A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

RXSETUP: Received Setup

This flag generates an interrupt while it is set to one.

Read:

0: No setup packet available.

1: A setup data packet has been sent by the host and is available in the FIFO.

Write:

0: Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1: No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

STALLSENT: Stall Sent (Control, Bulk Interrupt Endpoints)/ISOERROR (Isochronous Endpoints)

This flag generates an interrupt while it is set to one.

STALLSENT: This ends a STALL handshake.

Read:

0: The host has not acknowledged a STALL.

1: Host has acknowledged the stall.

Write:

0: Resets the STALLSENT flag, clears the interrupt.

1: No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

ISOERROR: A CRC error has been detected in an isochronous transfer.

Read:

0: No error in the previous isochronous transfer.

1: CRC error has been detected, data available in the FIFO are corrupted.

Write:

0: Resets the ISOERROR flag, clears the interrupt.

1: No effect.

© 2017 Microchip Technology Inc.

TXPKTRDY: Transmit Packet Ready

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0: There is no data to send.

1: The data is waiting to be sent upon reception of token IN.

Write:

0: Can be used in the procedure to cancel transmission data. (See Section 36.5.2.5 ”Transmit Data Cancellation”)

1: A new data payload has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)

Read:

0: Normal state.

1: Stall state.

Write:

0: Return to normal state.

1: Send STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0: Notifies USB device that data have been read in the FIFO’s Bank 1.

1: To leave the read value unchanged.

Read (Set by the USB peripheral):

0: No data packet has been received in the FIFO's Bank 1.

1: A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

DIR: Transfer Direction (only available for control endpoints)

Read/Write

0: Allows Data OUT transactions in the control data stage.

1: Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

© 2017 Microchip Technology Inc.

EPTYPE[2:0]: Endpoint Type

Read/Write

000	Control
001	Isochronous OUT
101	Isochronous IN
010	Bulk OUT
110	Bulk IN
011	Interrupt OUT
111	Interrupt IN

DTGLE: Data Toggle

Read-only

0: Identifies DATA0 packet.

1: Identifies DATA1 packet.

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

EPEDS: Endpoint Enable Disable

Read:

0: Endpoint disabled.

1: Endpoint enabled.

Write:

0: Disables endpoint.

1: Enables endpoint.

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

RXBYTECNT[10:0]: Number of Bytes Available in the FIFO

Read-only

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx register.

© 2017 Microchip Technology Inc.

36.6.11 UDP FIFO Data Register

Name:UDP_FDRx [x = 0..5]

Access: 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
FIFO_DATA							

FIFO_DATA[7:0]: FIFO Data Value

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

© 2017 Microchip Technology Inc.

36.6.12 UDP Transceiver Control Register

Name:UDP_TXVC

Access: 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
–	–	–	–	–	–	PUON	TXVDIS
7	6	5	4	3	2	1	0
–	–	–	–	–	–	_	_

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

TXVDIS: Transceiver Disable

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS field.

To enable the transceiver, TXVDIS must be cleared.

PUON: Pullup On

0: The 1.5KΩ integrated pullup on DP is disconnected.

1: The 1.5 KΩ integrated pullup on DP is connected.

Note: If the USB pullup is not connected on DP, the user should not write in any UDP register other than the UDP_TXVC register. This is because if DP and DM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

© 2017 Microchip Technology Inc.

37. USB Host Port (UHP)

37.1 Overview

The USB Host Port (UHP) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed and Low-speed protocols.

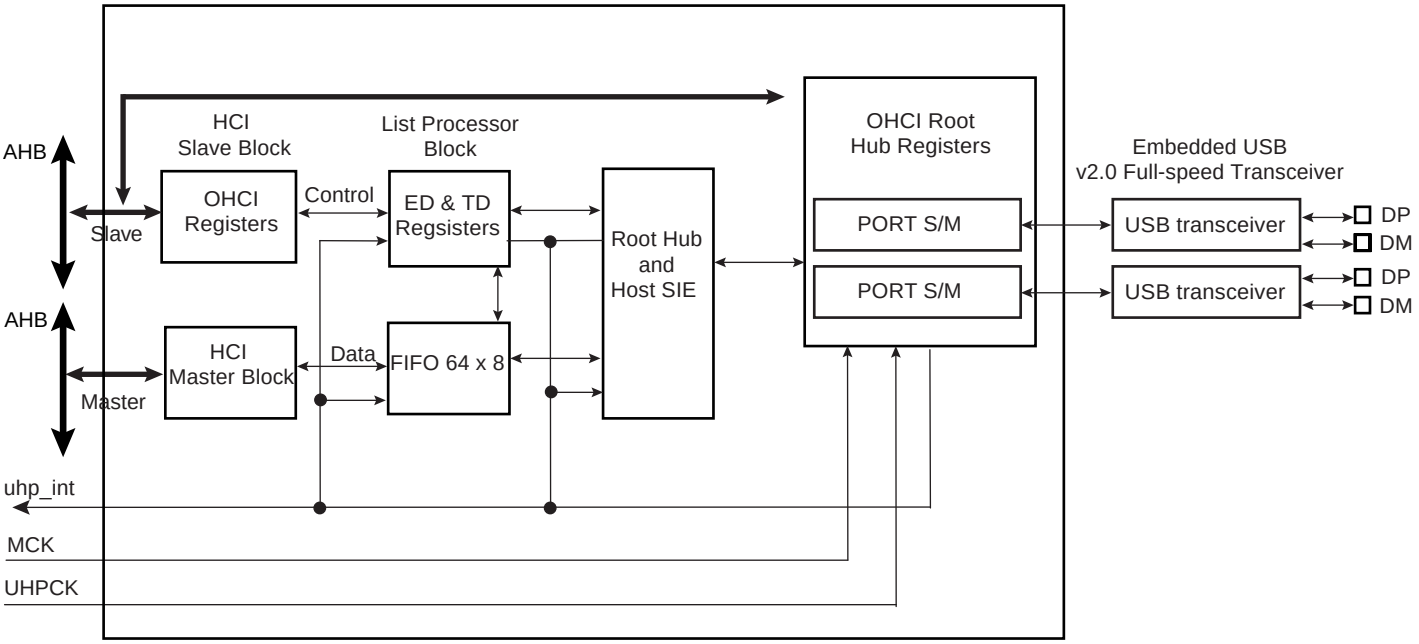
The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB “tiered star” topology.

The USB Host Port controller is fully compliant with the Open HCI specification. The USB Host Port User Interface (registers description) can be found in the Open HCI Rev 1.0 Specification available on www.hp.com. The standard OHCI USB stack driver can be easily ported to Microchip’s architecture in the same way all existing class drivers run without hardware specialization.

This means that all standard class devices are automatically detected and available to the user application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mouses.

37.2 Block Diagram

Figure 37-1: Block Diagram



Access to the USB host operational registers is achieved through the AHB bus slave interface. The Open HCI host controller initializes master DMA transfers through the ASB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer Descriptor

Memory access errors (abort, misalignment) lead to an “Unrecoverable Error” indicated by the corresponding flag in the host controller operational registers.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

USB physical transceivers are integrated in the product and driven by the root hub’s ports.

Over current protection on ports can be activated by the USB host controller. Microchip’s standard product does not dedicate pads to external over current protection.

37.3 Product Dependencies

37.3.1 I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

37.3.2 Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of ± 0.25%.

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the master clock MCK used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

37.3.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling USB host interrupts requires programming the AIC before configuring the UHP.

37.4 Functional Description

Refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

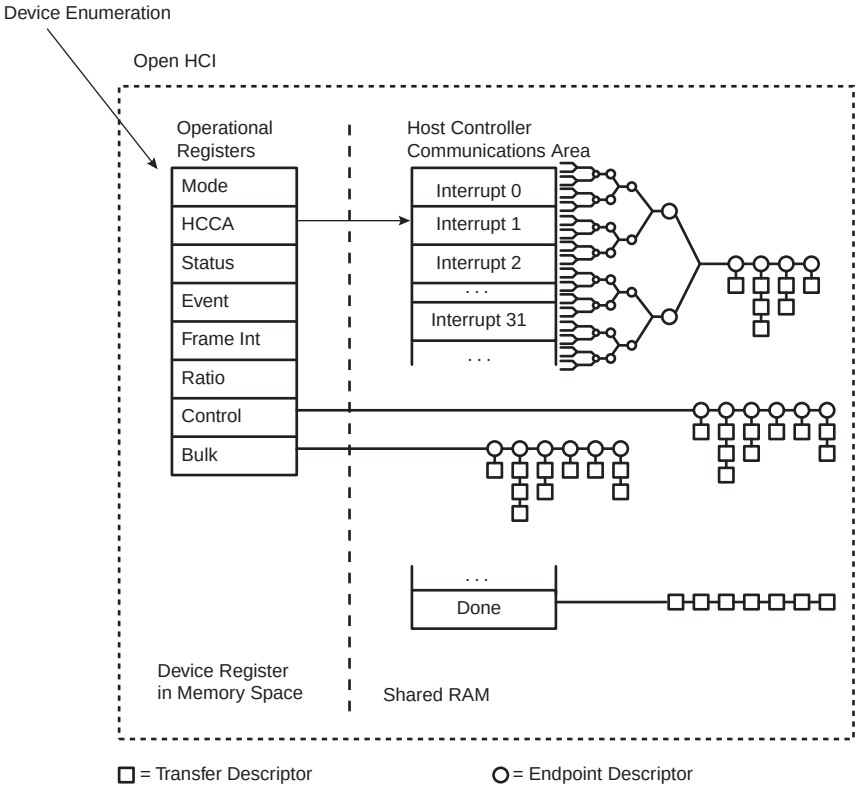
37.4.1 Host Controller Interface

There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the USB Host Controller. The Host Controller is the target for all communications on this channel. The operational registers contain control, status and list pointer registers. They are mapped in the memory mapped area. Within the operational register set there is a pointer to a location in the processor address space named the Host Controller Communication Area (HCCA). The HCCA is the second communication channel. The host controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint Descriptor lists, the head pointer to the done queue and status information associated with start-of-frame processing.

The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

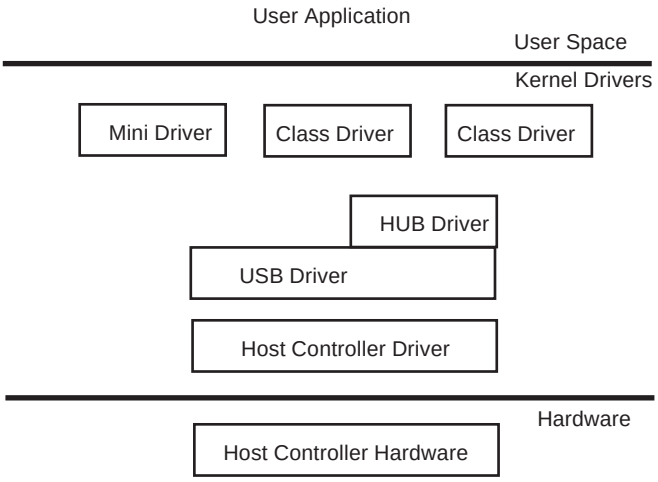
© 2017 Microchip Technology Inc.

Figure 37-2: USB Host Communication Channels



37.4.2 Host Controller Driver

Figure 37-3: USB Host Drivers



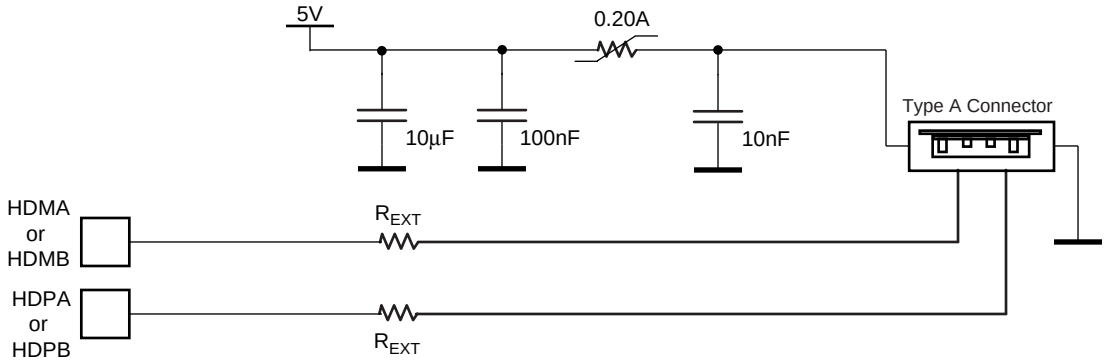
USB Handling is done through several layers as follows:

- Host controller hardware and serial engine: Transmits and receives USB data on the bus.
- Host controller driver: Drives the Host controller hardware and handles the USB protocol.
- USB Bus driver and hub driver: Handles USB commands and enumeration. Offers a hardware independent interface.
- Mini driver: Handles device specific commands.
- Class driver: Handles standard devices. This acts as a generic driver for a class of devices, for example the HID driver.

© 2017 Microchip Technology Inc.

37.5 Typical Connection

Figure 37-4: Board Schematic to Interface UHP Device Controller



A termination serial resistor must be connected to HDP and HDM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

© 2017 Microchip Technology Inc.

38. Image Sensor Interface (ISI)

38.1 Overview

The Image Sensor Interface (ISI) connects a CMOS-type image sensor to the processor and provides image capture in various formats. It does data conversion, if necessary, before the storage in memory through DMA.

The ISI supports color CMOS image sensor and grayscale image sensors with a reduced set of functionalities.

In grayscale mode, the data stream is stored in memory without any processing and so is not compatible with the LCD controller.

Internal FIFOs on the preview and codec paths are used to store the incoming data. The RGB output on the preview path is compatible with the LCD controller. This module outputs the data in RGB format (LCD compatible) and has scaling capabilities to make it compliant to the LCD display resolution (See Table 38-3).

Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

It supports two modes of synchronization:

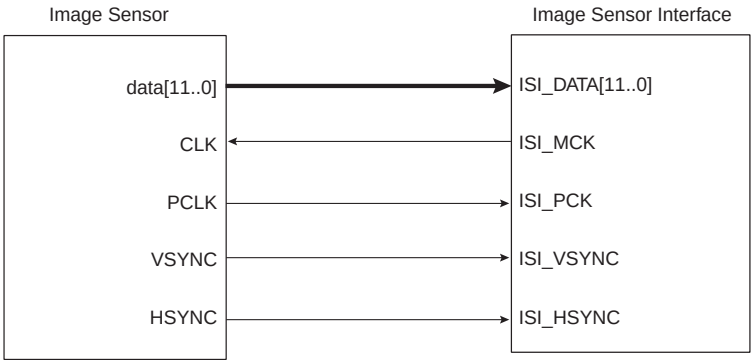
1. The hardware with ISI_VSYNC and ISI_HSYNC signals
2. The International Telecommunication Union Recommendation *ITU-R BT.656-4* Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) synchronization sequence.

Using EAV/SAV for synchronization reduces the pin count (ISI_VSYNC, ISI_HSYNC not used). The polarity of the synchronization pulse is programmable to comply with the sensor signals.

Table 38-1: I/O Description

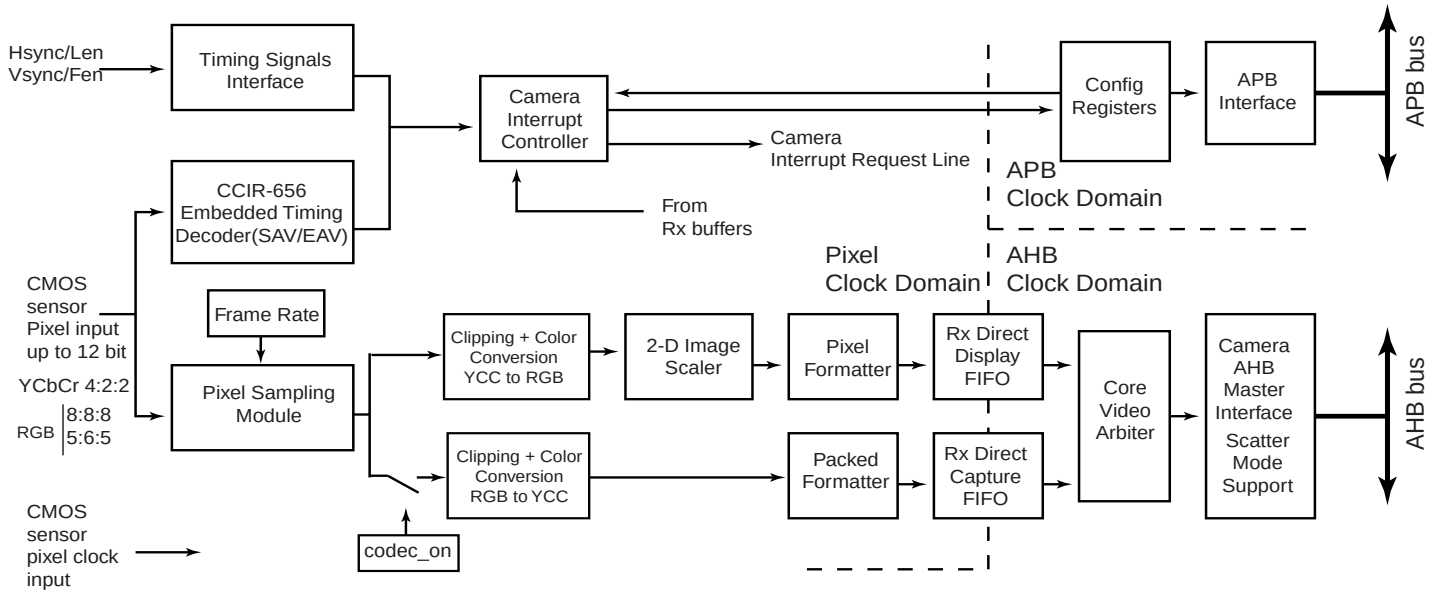
Signal	Dir	Description
ISI_VSYNC	IN	Vertical Synchronization
ISI_HSYNC	IN	Horizontal Synchronization
ISI_DATA[11..0]	IN	Sensor Pixel Data
ISI_MCK	OUT	Master Clock Provided to the Image Sensor
ISI_PCK	IN	Pixel Clock Provided by the Image Sensor

Figure 38-1: ISI Connection Example



38.2 Block Diagram

Figure 38-2: Image Sensor Interface Block Diagram



38.3 Functional Description

The Image Sensor Interface (ISI) supports direct connection to the ITU-R BT. 601/656 8-bit mode compliant sensors and up to 12-bit gray-scale sensors. It receives the image data stream from the image sensor on the 12-bit data bus.

This module receives up to 12 bits for data, the horizontal and vertical synchronizations and the pixel clock. The reduced pin count alternative for synchronization is supported for sensors that embed SAV (start of active video) and EAV (end of active video) delimiters in the data stream.

The Image Sensor Interface interrupt line is generally connected to the Advanced Interrupt Controller and can trigger an interrupt at the beginning of each frame and at the end of a DMA frame transfer. If the SAV/EAV synchronization is used, an interrupt can be triggered on each delimiter event.

For 8-bit color sensors, the data stream received can be in several possible formats: YCbCr 4:2:2, RGB 8:8:8, RGB 5:6:5 and may be processed before the storage in memory. The data stream may be sent on both preview path and codec path if the bit CODEC_ON in the ISI_CR1 is one. To optimize the bandwidth, the codec path should be enabled only when a capture is required.

In grayscale mode, the input data stream is stored in memory without any processing. The 12-bit data, which represent the grayscale level for the pixel, is stored in memory one or two pixels per word, depending on the GS_MODE bit in the ISI_CR2 register. The codec datapath is not available when grayscale image is selected.

A frame rate counter allows users to capture all frames or 1 out of every 2 to 8 frames.

38.3.1 Data Timing

The two data timings using horizontal and vertical synchronization and EAV/SAV sequence synchronization are shown in Figure 38-3 and Figure 38-4.

In the VSYNC/HSYNC synchronization, the valid data is captured with the active edge of the pixel clock (ISI_PCK), after SFD lines of vertical blanking and SLD pixel clock periods delay programmed in the control register.

The ITU-RBT.656-4 defines the functional timing for an 8-bit wide interface.

There are two timing reference signals, one at the beginning of each video data block SAV (0xFF000080) and one at the end of each video data block EAV(0xFF00009D). Only data sent between EAV and SAV is captured. Horizontal blanking and vertical blanking are ignored. Use of the SAV and EAV synchronization eliminates the ISI_VSYNC and ISI_HSYNC signals from the interface, thereby reducing the pin count. In order to retrieve both frame and line synchronization properly, at least one line of vertical blanking is mandatory.

© 2017 Microchip Technology Inc.

Figure 38-3: HSYNC and VSYNC Synchronization

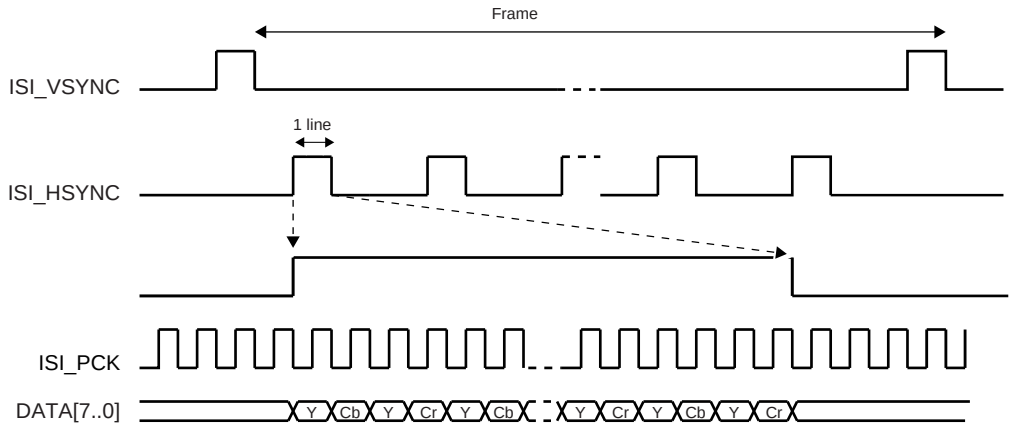
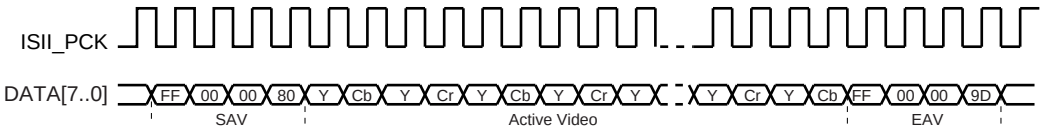


Figure 38-4: SAV and EAV Sequence Synchronization



38.3.2 Data Ordering

The RGB color space format is required for viewing images on a display screen preview, and the YCbCr color space format is required for encoding.

All the sensors do not output the YCbCr or RGB components in the same order. The ISI allows the user to program the same component order as the sensor, reducing software treatments to restore the right format.

Table 38-2: Data Ordering in YCbCr Mode

Mode	Byte 0	Byte 1	Byte 2	Byte 3
Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
Mode1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
Mode2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
Mode3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

Table 38-3: RGB Format in Default Mode, RGB_CFG = 00, No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R7(i)	R6(i)	R5(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	G7(i)	G6(i)	G5(i)	G4(i)	G3(i)	G2(i)	G1(i)	G0(i)
	Byte 2	B7(i)	B6(i)	B5(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 3	R7(i+1)	R6(i+1)	R5(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)

Table 38-3: RGB Format in Default Mode, RGB_CFG = 00, No Swap

RGB 5:6:5	Byte 0	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)	G5(i)	G4(i)	G3(i)
	Byte 1	G2(i)	G1(i)	G0(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 2	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)
	Byte 3	G2(i+1)	G1(i+1)	G0(i+1)	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)

Table 38-4: RGB Format, RGB_CFG = 10 (Mode 2), No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 5:6:5	Byte 0	G2(i)	G1(i)	G0(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)	G5(i)	G4(i)	G3(i)
	Byte 2	G2(i+1)	G1(i+1)	G0(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
	Byte 3	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)

Table 38-5: RGB Format in Default Mode, RGB_CFG = 00, Swap Activated

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)	R5(i)	R6(i)	R7(i)
	Byte 1	G0(i)	G1(i)	G2(i)	G3(i)	G4(i)	G5(i)	G6(i)	G7(i)
	Byte 2	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	B5(i)	B6(i)	B7(i)
	Byte 3	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)	R5(i+1)	R6(i+1)	R7(i+1)
RGB 5:6:5	Byte 0	G3(i)	G4(i)	G5(i)	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)
	Byte 1	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	G0(i)	G1(i)	G2(i)
	Byte 2	G3(i+1)	G4(i+1)	G5(i+1)	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)
	Byte 3	B0(i+1)	B1(i+1)	B2(i+1)	B3(i+1)	B4(i+1)	G0(i+1)	G1(i+1)	G2(i+1)

The RGB 5:6:5 input format is processed to be displayed as RGB 5:5:5 format, compliant with the 16-bit mode of the LCD controller.

38.3.3 Clocks

The sensor master clock (ISI_MCK) can be generated either by the Advanced Power Management Controller (APMC) through a Programmable Clock output or by an external oscillator connected to the sensor.

None of the sensors embeds a power management controller, so providing the clock by the APMC is a simple and efficient way to control power consumption of the system.

Care must be taken when programming the system clock. The ISI has two clock domains, the system bus clock and the pixel clock provided by sensor. The two clock domains are not synchronized, but the system clock must be faster than pixel clock.

38.3.4 Preview Path

38.3.4.1 Scaling, Decimation (Subsampling)

This module resizes captured 8-bit color sensor images to fit the LCD display format. The resize module performs only downscaling. The same ratio is applied for both horizontal and vertical resize, then a fractional decimation algorithm is applied.

The decimation factor is a multiple of 1/16 and values 0 to 15 are forbidden.

© 2017 Microchip Technology Inc.

Table 38-6: Decimation Factor

Dec value	0->15	16	17	18	19	...	124	125	126	127
Dec Factor	X	1	1.063	1.125	1.188	...	7.750	7.813	7.875	7.938

Table 38-7: Decimation and Scaler Offset Values

INPUT / OUTPUT		352*288	640*480	800*600	1280*1024	1600*1200	2048*1536
VGA 640*480	F	NA	16	20	32	40	51
QVGA 320*240	F	16	32	40	64	80	102
CIF 352*288	F	16	26	33	56	66	85
QCIF 176*144	F	16	53	66	113	133	170

Example:

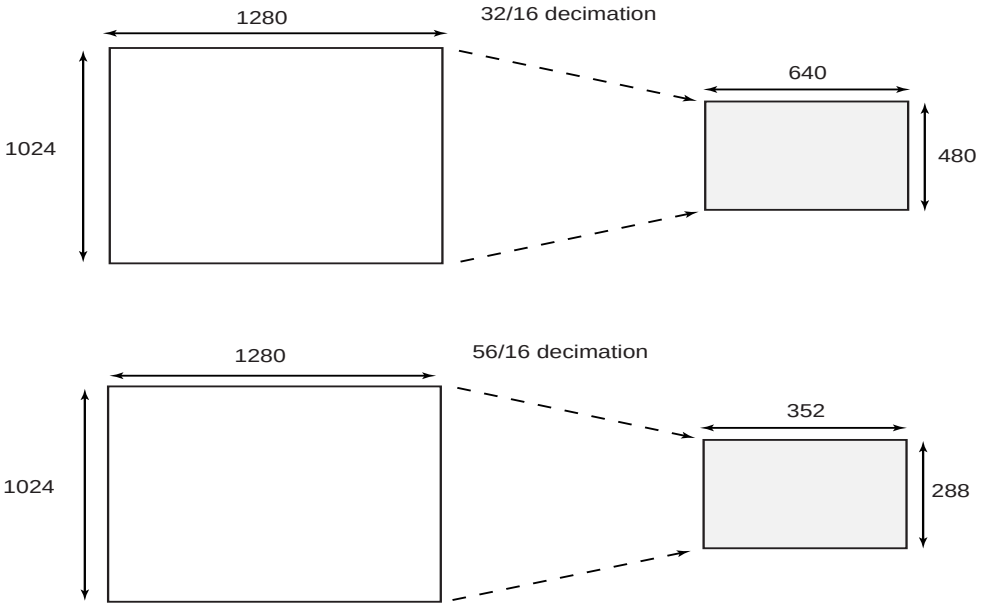
Input 1280*1024 Output = 640*480

Hratio = 1280/640 = 2

Vratio = 1024/480 = 2.1333

The decimation factor is 2 so 32/16.

Figure 38-5: Resize Examples



38.3.4.2 Color Space Conversion

This module converts YCrCb or YUV pixels to RGB color space. Clipping is performed to ensure that the samples value do not exceed the allowable range. The conversion matrix is defined below and is fully programmable:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} C_0 & 0 & C_1 \\ C_0 & -C_2 & -C_3 \\ C_0 & C_4 & 0 \end{bmatrix} \times \begin{bmatrix} Y - Y_{off} \\ C_b - C_{boff} \\ C_r - C_{roff} \end{bmatrix}$$

Example of programmable value to convert YCrCb to RGB:

$$\begin{cases} R = 1.164 \cdot (Y - 16) + 1.596 \cdot (C_r - 128) \\ G = 1.164 \cdot (Y - 16) - 0.813 \cdot (C_r - 128) - 0.392 \cdot (C_b - 128) \\ B = 1.164 \cdot (Y - 16) + 2.107 \cdot (C_b - 128) \end{cases}$$

An example of programmable value to convert from YUV to RGB:

$$\begin{cases} R = Y + 1.596 \cdot V \\ G = Y - 0.394 \cdot U - 0.436 \cdot V \\ B = Y + 2.032 \cdot U \end{cases}$$

38.3.4.3 Memory Interface

Preview datapath contains a data formatter that converts 8:8:8 pixel to RGB 5:5:5 format compliant with 16-bit format of the LCD controller. In general, when converting from a color channel with more bits to one with fewer bits, formatter module discards the lower-order bits. Example: Converting from RGB 8:8:8 to RGB 5:6:5, it discards the three LSBs from the red and blue channels, and two LSBs from the green channel. When grayscale mode is enabled, two memory format are supported. One mode supports 2 pixels per word, and the other mode supports 1 pixel per word.

Table 38-8: Grayscale Memory Mapping Configuration for 12-bit Data

GS_MODE	DATA[31:24]	DATA[23:16]	DATA[15:8]	DATA[7:0]
0	P_0[11:4]	P_0[3:0], 0000	P_1[11:4]	P_1[3:0], 0000
1	P_0[11:4]	P_0[3:0], 0000	0	0

38.3.4.4 FIFO and DMA Features

Both preview and Codec datapaths contain FIFOs, asynchronous buffers that are used to safely transfer formatted pixels from Pixel clock domain to AHB clock domain. A video arbiter is used to manage FIFO thresholds and triggers a relevant DMA request through the AHB master interface. Thus, depending on FIFO state, a specified length burst is asserted. Regarding AHB master interface, it supports Scatter DMA mode through linked list operation. This mode of operation improves flexibility of image buffer location and allows the user to allocate two or more frame buffers. The destination frame buffers are defined by a series of Frame Buffer Descriptors (FBD). Each FBD controls the transfer of one entire frame and then optionally loads a further FBD to switch the DMA operation at another frame buffer address. The FBD is defined by a series of two words. The first one defines the current frame buffer address, and the second defines the next FBD memory location. This DMA transfer mode is only available for preview datapath and is configured in the ISI_PPFBD register that indicates the memory location of the first FBD.

The primary FBD is programmed into the camera interface controller. The data to be transferred described by an FBD requires several burst access. In the example below, the use of 2 ping-pong frame buffers is described.

© 2017 Microchip Technology Inc.

Example

The first FBD, stored at address 0x30000, defines the location of the first frame buffer.

Destination Address: frame buffer ID0 0x02A000

Next FBD address: 0x30010

Second FBD, stored at address 0x30010, defines the location of the second frame buffer.

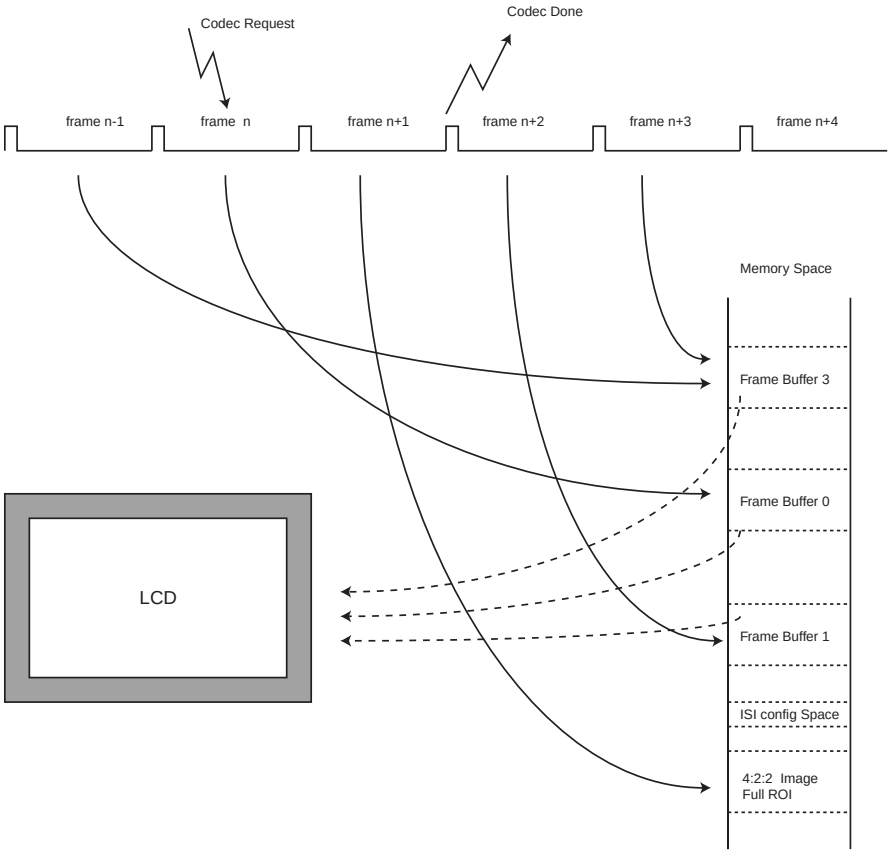
Destination Address: frame buffer ID1 0x3A000

Transfer width: 32 bit

Next FBD address: 0x30000, wrapping to first FBD.

Using this technique, several frame buffers can be configured through the linked list. Figure 38-6 illustrates a typical three frame buffer application. Frame n is mapped to frame buffer 0, frame n+1 is mapped to frame buffer 1, frame n+2 is mapped to Frame buffer 2, further frames wrap. A codec request occurs, and the full-size 4:2:2 encoded frame is stored in a dedicated memory space.

Figure 38-6: Three Frame Buffers Application and Memory Mapping



38.3.5 Codec Path

38.3.5.1 Color Space Conversion

Depending on user selection, this module can be bypassed so that input YCrCb stream is directly connected to the format converter module. If the RGB input stream is selected, this module converts RGB to YCrCb color space with the formulas given below:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & -C_4 & -C_5 \\ -C_6 & -C_7 & C_8 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y_{off} \\ Cr_{off} \\ Cb_{off} \end{bmatrix}$$

An example of coefficients is given below:

$$\begin{cases} Y = 0.257 \cdot R + 0.504 \cdot G + 0.098 \cdot B + 16 \\ C_r = 0.439 \cdot R - 0.368 \cdot G - 0.071 \cdot B + 128 \\ C_b = -0.148 \cdot R - 0.291 \cdot G + 0.439 \cdot B + 128 \end{cases}$$

38.3.5.2 Memory Interface

Dedicated FIFO are used to support packed memory mapping. YCrCb pixel components are sent in a single 32-bit word in a contiguous space (packed). Data is stored in the order of natural scan lines. Planar mode is not supported.

38.3.5.3 DMA Features

Unlike preview datapath, codec datapath DMA mode does not support linked list operation. Only the CODEC_DMA_ADDR register is used to configure the frame buffer base address.

© 2017 Microchip Technology Inc.

38.4 Image Sensor Interface (ISI) User Interface

Table 38-9: ISI Memory Mapping

Offset	Register	Register	Access	Reset
0x00	ISI Control 1 Register	ISI_CR1	Read/Write	0x00000002
0x04	ISI Control 2 Register	ISI_CR2	Read/Write	0x00000000
0x08	ISI Status Register	ISI_SR	Read-only	0x00000000
0x0C	ISI Interrupt Enable Register	ISI_IER	Read/Write	0x00000000
0x10	ISI Interrupt Disable Register	ISI_IDR	Read/Write	0x00000000
0x14	ISI Interrupt Mask Register	ISI_IMR	Read/Write	0x00000000
0x18	Reserved	–	–	–
0x1C	Reserved	–	–	–
0x20	ISI Preview Size Register	ISI_PSIZE	Read/Write	0x00000000
0x24	ISI Preview Decimation Factor Register	ISI_PDECF	Read/Write	0x00000010
0x28	ISI Preview Primary FBD Register	ISI_PPFBD	Read/Write	0x00000000
0x2C	ISI Codec DMA Base Address Register	ISI_CDBA	Read/Write	0x00000000
0x30	ISI CSC YCrCb To RGB Set 0 Register	ISI_Y2R_SET0	Read/Write	0x6832cc95
0x34	ISI CSC YCrCb To RGB Set 1 Register	ISI_Y2R_SET1	Read/Write	0x00007102
0x38	ISI CSC RGB To YCrCb Set 0 Register	ISI_R2Y_SET0	Read/Write	0x01324145
0x3C	ISI CSC RGB To YCrCb Set 1 Register	ISI_R2Y_SET1	Read/Write	0x01245e38
0x40	ISI CSC RGB To YCrCb Set 2 Register	ISI_R2Y_SET2	Read/Write	0x01384a4b
0x44–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: Several parts of the ISI controller use the pixel clock provided by the image sensor (ISI_PCK). Thus the user must first program the image sensor to provide this clock (ISI_PCK) before programming the Image Sensor Controller.

38.4.1 ISI Control 1 Register

Name: ISI_CR1

Access: Read/Write

31	30	29	28	27	26	25	24
SFD							
23	22	21	20	19	18	17	16
SLD							
15	14	13	12	11	10	9	8
CODEC_ON	THMASK		FULL	–	FRATE		
7	6	5	4	3	2	1	0
CRC_SYNC	EMB_SYNC	–	PIXCLK_POL	VSYNC_POL	HSYNC_POL	ISI_DIS	ISI_RST

ISI_RST: Image Sensor Interface Reset

Write-only. Refer to bit SOFTRST in "ISI Status Register" for soft reset status.

0: No action

1: Resets the image sensor interface.

ISI_DIS: Image Sensor Disable:

0: Enable the image sensor interface.

1: Finish capturing the current frame and then shut down the module.

HSYNC_POL: Horizontal Synchronization Polarity

0: HSYNC active high

1: HSYNC active low

VSYNC_POL: Vertical sYnchronization Polarity

0: VSYNC active high

1: VSYNC active low

PIXCLK_POL: Pixel Clock Polarity

0: Data is sampled on rising edge of pixel clock

1: Data is sampled on falling edge of pixel clock

EMB_SYNC: Embedded Synchronization

0: Synchronization by HSYNC, VSYNC

1: Synchronization by embedded synchronization sequence SAV/EAV

CRC_SYNC: Embedded Synchronization

0: No CRC correction is performed on embedded synchronization

1: CRC correction is performed. if the correction is not possible, the current frame is discarded and the CRC_ERR is set in the status register.

FRATE: Frame Rate [0..7]

0: All the frames are captured, else one frame every FRATE+1 is captured.

FULL: Full Mode is Allowed

1: Both codec and preview datapaths are working simultaneously

© 2017 Microchip Technology Inc.

THMASK: Threshold Mask

0: 4, 8 and 16 AHB bursts are allowed

1: 8 and 16 AHB bursts are allowed

2: Only 16 AHB bursts are allowed

CODEC_ON: Enable the Codec Path Enable Bit

Write-only.

0: The codec path is disabled

1: The codec path is enabled and the next frame is captured. Refer to bit CDC_PND in “ISI Status Register” .

SLD: Start of Line Delay

SLD pixel clock periods to wait before the beginning of a line.

SFD: Start of Frame Delay

SFD lines are skipped at the beginning of the frame.

© 2017 Microchip Technology Inc.

38.4.2 ISI Control 2 Register

Name: ISI_CR2

Access: Read/Write

31	30	29	28	27	26	25	24
RGB_CFG		YCC_SWAP		–	IM_HSIZE		

23	22	21	20	19	18	17	16
IM_HSIZE							

15	14	13	12	11	10	9	8
COL_SPACE	RGB_SWAP	GRAYSCALE	RGB_MODE	GS_MODE	IM_VSIZE		

7	6	5	4	3	2	1	0
IM_VSIZE							

IM_VSIZE: Vertical Size of the Image Sensor [0..2047]

Vertical size = IM_VSIZE + 1

GS_MODE

0: 2 pixels per word

1: 1 pixel per word

RGB_MODE: RGB Input Mode

0: RGB 8:8:8 24 bits

1: RGB 5:6:5 16 bits

GRAYSCALE

0: Grayscale mode is disabled

1: Input image is assumed to be grayscale coded

RGB_SWAP

0: D7 -> R7

1: D0 -> R7

The RGB_SWAP has no effect when the grayscale mode is enabled.

COL_SPACE: Color Space for The Image Data

0: YCbCr

1: RGB

IM_HSIZE: Horizontal Size of the Image Sensor [0..2047]

Horizontal size = IM_HSIZE + 1

YCC_SWAP: Defines the YCC Image Data

YCC_SWAP	Byte 0	Byte 1	Byte 2	Byte 3
00: Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
01: Mode1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
10: Mode2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
11: Mode3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

© 2017 Microchip Technology Inc.

RGB_CFG: Defines RGB Pattern when RGB_MODE is set to 1

RGB_CFG	Byte 0	Byte 1	Byte 2	Byte 3
00: Default	R/G(MSB)	G(LSB)/B	R/G(MSB)	G(LSB)/B
01: Mode1	B/G(MSB)	G(LSB)/R	B/G(MSB)	G(LSB)/R
10: Mode2	G(LSB)/R	B/G(MSB)	G(LSB)/R	B/G(MSB)
11: Mode3	G(LSB)/B	R/G(MSB)	G(LSB)/B	R/G(MSB)

If RGB_MODE is set to RGB 8:8:8, then RGB_CFG = 0 implies RGB color sequence, else it implies BGR color sequence.

38.4.3 ISI Status Register

Name: ISI_SR

Access: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP
7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	CDC_PND	SOFTRST	DIS	SOF

SOF: Start of Frame

0: No start of frame has been detected.

1: A start of frame has been detected.

DIS: Image Sensor Interface Disable

0: The image sensor interface is enabled.

1: The image sensor interface is disabled and stops capturing data. The DMA controller and the core can still read the FIFOs.

SOFTRST: Software Reset

0: Software reset not asserted or not completed.

1: Software reset has completed successfully.

CDC_PND: Codec Request Pending

0: No request asserted.

1: A codec request is pending. If a codec request is asserted during a frame, the CDC_PND bit rises until the start of a new frame. The capture is completed when the flag FO_C_EMP = 1.

CRC_ERR: CRC Synchronization Error

0: No CRC error in the embedded synchronization frame (SAV/EAV)

1: The CRC_SYNC is enabled in the control register and an error has been detected and not corrected. The frame is discarded and the ISI waits for a new one.

FO_C_OVF: FIFO Codec Overflow

0: No overflow

1: An overrun condition has occurred in input FIFO on the codec path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO.

FO_P_OVF: FIFO Preview Overflow

0: No overflow

1: An overrun condition has occurred in input FIFO on the preview path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO.

FO_P_EMP

0:The DMA has not finished transferring all the contents of the preview FIFO.

1:The DMA has finished transferring all the contents of the preview FIFO.

© 2017 Microchip Technology Inc.

FO_C_EMP

0: The DMA has not finished transferring all the contents of the codec FIFO.

1: The DMA has finished transferring all the contents of the codec FIFO.

FR_OVR: Frame Rate Overrun

0: No frame overrun.

1: Frame overrun, the current frame is being skipped because a vsync signal has been detected while flushing FIFOs.

38.4.4 Interrupt Enable Register

Name: ISI_IER

Access: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP
7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

SOF: Start of Frame

1: Enables the Start of Frame interrupt.

DIS: Image Sensor Interface Disable

1: Enables the DIS interrupt.

SOFTRST: Soft Reset

1: Enables the Soft Reset Completion interrupt.

CRC_ERR: CRC Synchronization Error

1: Enables the CRC_SYNC interrupt.

FO_C_OVF: FIFO Codec Overflow

1: Enables the codec FIFO overflow interrupt.

FO_P_OVF: FIFO Preview Overflow

1: Enables the preview FIFO overflow interrupt.

FO_P_EMP

1: Enables the preview FIFO empty interrupt.

FO_C_EMP

1: Enables the codec FIFO empty interrupt.

FR_OVR: Frame Overrun

1: Enables the Frame overrun interrupt.

© 2017 Microchip Technology Inc.

38.4.5 ISI Interrupt Disable Register

Name: ISI_IDR

Access: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP
7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

SOF: Start of Frame

1: Disables the Start of Frame interrupt.

DIS: Image Sensor Interface Disable

1: Disables the DIS interrupt.

SOFTRST

1: Disables the soft reset completion interrupt.

CRC_ERR: CRC Synchronization Error

1: Disables the CRC_SYNC interrupt.

FO_C_OVF: FIFO Codec Overflow

1: Disables the codec FIFO overflow interrupt.

FO_P_OVF: FIFO Preview Overflow

1: Disables the preview FIFO overflow interrupt.

FO_P_EMP

1: Disables the preview FIFO empty interrupt.

FO_C_EMP

1: Disables the codec FIFO empty interrupt.

FR_OVR

1: Disables frame overrun interrupt.

© 2017 Microchip Technology Inc.

38.4.6 ISI Interrupt Mask Register

Name: ISI_IMR

Access: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP
7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

SOF: Start of Frame

0: The Start of Frame interrupt is disabled.

1: The Start of Frame interrupt is enabled.

DIS: Image Sensor Interface Disable

0: The DIS interrupt is disabled.

1: The DIS interrupt is enabled.

SOFTRST

0: The soft reset completion interrupt is enabled.

1: The soft reset completion interrupt is disabled.

CRC_ERR: CRC Synchronization Error

0: The CRC_SYNC interrupt is disabled.

1: The CRC_SYNC interrupt is enabled.

FO_C_OVF: FIFO Codec Overflow

0: The codec FIFO overflow interrupt is disabled.

1: The codec FIFO overflow interrupt is enabled.

FO_P_OVF: FIFO Preview Overflow

0: The preview FIFO overflow interrupt is disabled.

1: The preview FIFO overflow interrupt is enabled.

FO_P_EMP

0: The preview FIFO empty interrupt is disabled.

1: The preview FIFO empty interrupt is enabled.

FO_C_EMP

0: The codec FIFO empty interrupt is disabled.

1: The codec FIFO empty interrupt is enabled.

FR_OVR: Frame Rate Overrun

0: The frame overrun interrupt is disabled.

1: The frame overrun interrupt is enabled.

© 2017 Microchip Technology Inc.

38.4.7 ISI Preview Register

Name: ISI_PSIZE

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PREV_HSIZE	

23	22	21	20	19	18	17	16
PREV_HSIZE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	PREV_VSIZE	

7	6	5	4	3	2	1	0
PREV_VSIZE							

PREV_VSIZE: Vertical Size for the Preview Path

Vertical Preview size = PREV_VSIZE + 1 (480 max only in RGB mode).

PREV_HSIZE: Horizontal Size for the Preview Path

Horizontal Preview size = PREV_HSIZE + 1 (640 max only in RGB mode).

© 2017 Microchip Technology Inc.

38.4.8 ISI Preview Decimation Factor Register

Name: ISI_PDECF

Access: 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
DEC_FACTOR							

DEC_FACTOR: Decimation Factor

DEC_FACTOR is 8-bit width, range is from 16 to 255. Values from 0 to 16 do not perform any decimation.

© 2017 Microchip Technology Inc.

38.4.9 ISI Preview Primary FBD Register

Name: ISI_PPFBD

Access: Read/Write

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

PREV_FBD_ADDR: Base Address for Preview Frame Buffer Descriptor

Written with the address of the start of the preview frame buffer queue, reads as a pointer to the current buffer being used. The frame buffer is forced to word alignment.

38.4.10 ISI Codec DMA Base Address Register

Name: ISI_CDBA

Access: Read/Write

31	30	29	28	27	26	25	24
CODEC_DMA_ADDR							

23	22	21	20	19	18	17	16
CODEC_DMA_ADDR							

15	14	13	12	11	10	9	8
CODEC_DMA_ADDR							

7	6	5	4	3	2	1	0
CODEC_DMA_ADDR							

CODEC_DMA_ADDR: Base Address for Codec DMA

This register contains codec datapath start address of buffer location.

© 2017 Microchip Technology Inc.

38.4.11 ISI Color Space Conversion YCrCb to RGB Set 0 Register

Name: ISI_Y2R_SET0

Access: Read/Write

31	30	29	28	27	26	25	24
C3							
23	22	21	20	19	18	17	16
C2							
15	14	13	12	11	10	9	8
C1							
7	6	5	4	3	2	1	0
C0							

C0: Color Space Conversion Matrix Coefficient C0

C0 element, default step is 1/128, ranges from 0 to 1.9921875.

C1: Color Space Conversion Matrix Coefficient C1

C1 element, default step is 1/128, ranges from 0 to 1.9921875.

C2: Color Space Conversion Matrix Coefficient C2

C2 element, default step is 1/128, ranges from 0 to 1.9921875.

C3: Color Space Conversion Matrix Coefficient C3

C3 element default step is 1/128, ranges from 0 to 1.9921875.

© 2017 Microchip Technology Inc.

38.4.12 ISI Color Space Conversion YCrCb to RGB Set 1 Register

Name: ISI_Y2R_SET1

Access: 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
–	Cboff	Croff	Yoff	–	–	–	C4

C4

C4: Color Space Conversion Matrix Coefficient C4

C4 element default step is 1/128, ranges from 0 to 3.9921875.

Yoff: Color Space Conversion Luminance Default Offset

0: No offset.

1: Offset = 128.

Croff: Color Space Conversion Red Chrominance Default Offset

0: No offset.

1: Offset = 16.

Cboff: Color Space Conversion Blue Chrominance Default Offset

0: No offset.

1: Offset = 16.

© 2017 Microchip Technology Inc.

38.4.13 ISI Color Space Conversion RGB to YCrCb Set 0 Register

Name: ISI_R2Y_SET0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Roff

23	22	21	20	19	18	17	16
C2							

15	14	13	12	11	10	9	8
C1							

7	6	5	4	3	2	1	0
C0							

C0: Color Space Conversion Matrix Coefficient C0

C0 element default step is 1/256, from 0 to 0.49609375.

C1: Color Space Conversion Matrix Coefficient C1

C1 element default step is 1/128, from 0 to 0.9921875.

C2: Color Space Conversion Matrix Coefficient C2

C2 element default step is 1/512, from 0 to 0.2480468875.

Roff: Color Space Conversion Red Component Offset

0: No offset.

1: Offset = 16.

38.4.14 ISI Color Space Conversion RGB to YCrCb Set 1 Register

Name: ISI_R2Y_SET1

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Goff

23	22	21	20	19	18	17	16
C5							

15	14	13	12	11	10	9	8
C4							

7	6	5	4	3	2	1	0
C3							

C3: Color Space Conversion Matrix Coefficient C3

C0 element default step is 1/128, ranges from 0 to 0.9921875.

C4: Color Space Conversion Matrix Coefficient C4

C1 element default step is 1/256, ranges from 0 to 0.49609375.

C5: Color Space Conversion Matrix Coefficient C5

C1 element default step is 1/512, ranges from 0 to 0.2480468875.

Goff: Color Space Conversion Green Component Offset

0: No offset.

1: Offset = 128.

© 2017 Microchip Technology Inc.

38.4.15 ISI Color Space Conversion RGB to YCrCb Set 2 Register

Name: ISI_R2Y_SET2

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Boff

23	22	21	20	19	18	17	16
C8							

15	14	13	12	11	10	9	8
C7							

7	6	5	4	3	2	1	0
C6							

C6: Color Space Conversion Matrix Coefficient C6

C6 element default step is 1/512, ranges from 0 to 0.2480468875.

C7: Color Space Conversion Matrix coefficient C7

C7 element default step is 1/256, ranges from 0 to 0.49609375.

C8: Color Space Conversion Matrix Coefficient C8

C8 element default step is 1/128, ranges from 0 to 0.9921875.

Boff: Color Space Conversion Blue Component Offset

0: No offset.

1: Offset = 128.

© 2017 Microchip Technology Inc.

39. Analog-to-Digital Converter (ADC)

39.1 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.

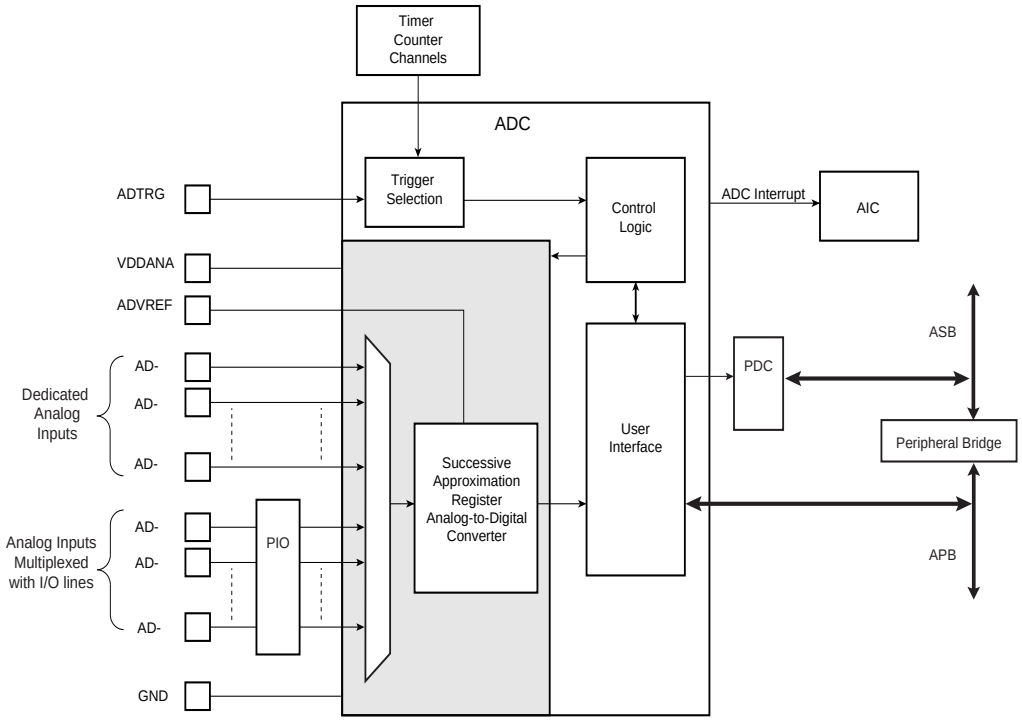
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

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.

39.2 Block Diagram

Figure 39-1: Analog-to-Digital Converter Block Diagram



© 2017 Microchip Technology Inc.

39.3 Signal Description

Table 39-1: ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0–AD3	Analog input channels
ADTRG	External trigger

39.4 Product Dependencies

39.4.1 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.

39.4.2 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.

39.4.3 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.

39.4.4 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.

39.4.5 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.

39.4.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

39.5 Functional Description

39.5.1 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” 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.

39.5.2 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.

© 2017 Microchip Technology Inc.

39.5.3 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 and of the LDATA field in the ADC_LCDR 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.

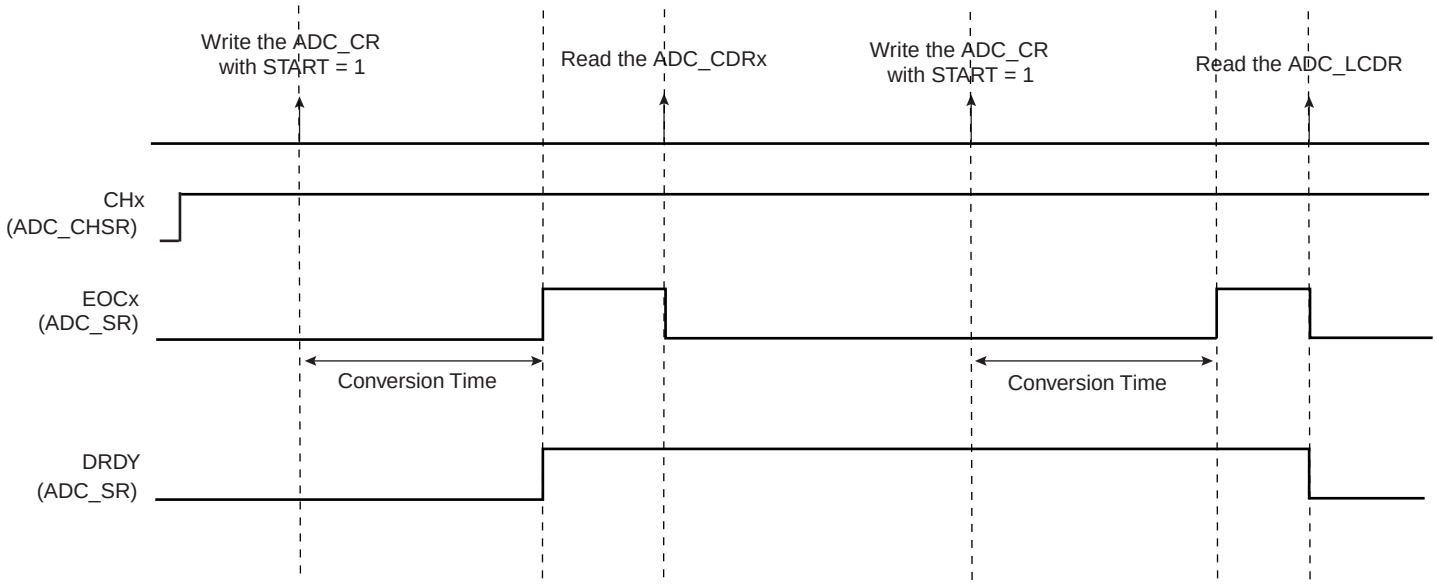
39.5.4 Conversion Results

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 39-2: EOCx and DRDY Flag Behavior



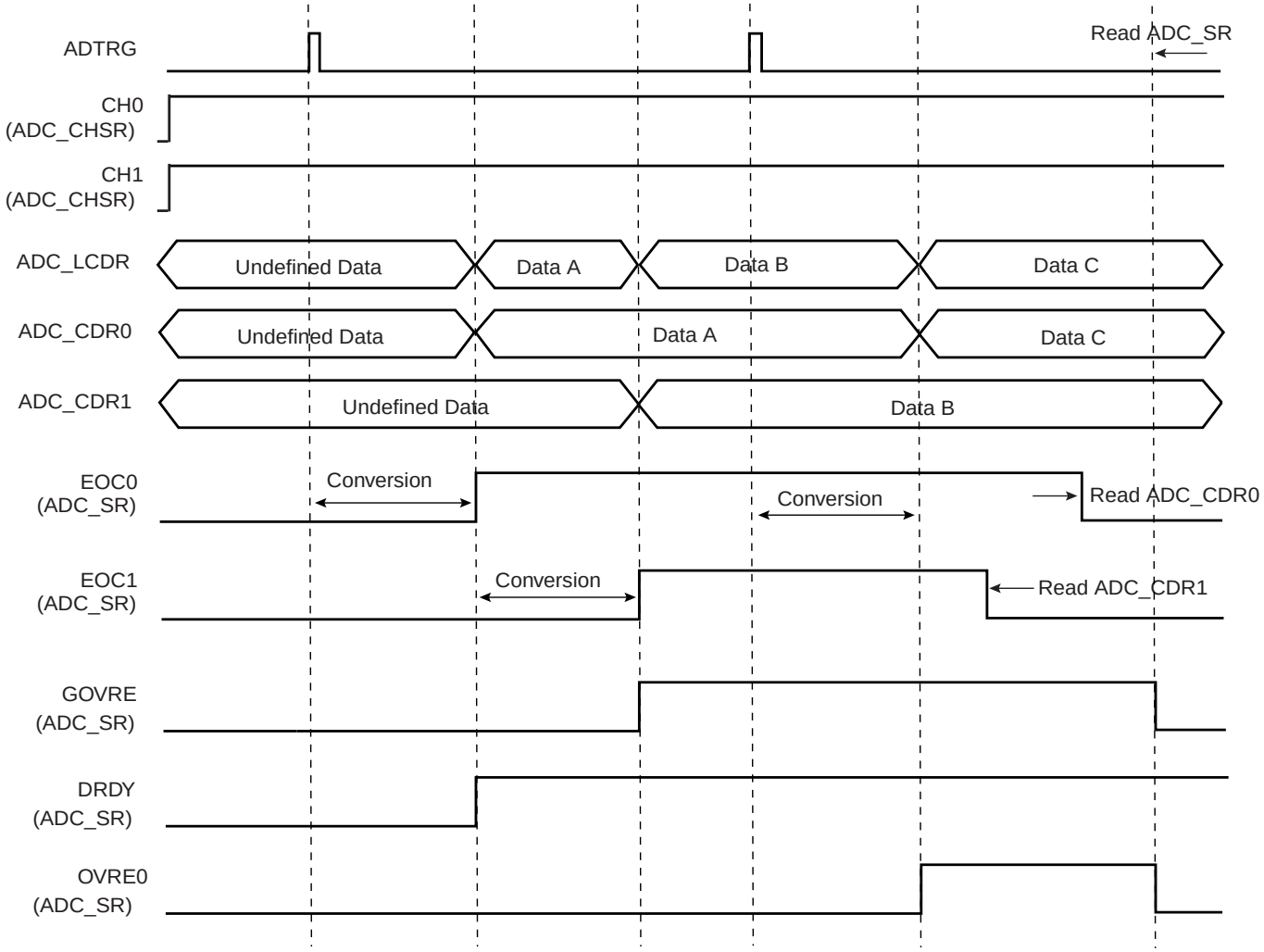
If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

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.

© 2017 Microchip Technology Inc.

Figure 39-3: 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.

39.5.5 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.

39.5.6 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.

39.5.7 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 bitfield SHTIM 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. Refer to Section 40.7 “ADC Characteristics” in Section 40. “Electrical Characteristics”.

© 2017 Microchip Technology Inc.

39.6 Analog-to-Digital Converter (ADC) User Interface

Table 39-2: 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
...	...	...	...	...
0x3C	Channel Data Register 3	ADC_CDR3	Read-only	0x00000000
0x44–0xFC	Reserved	–	–	–

39.6.1 ADC Control Register

Name:ADC_CR

Access: 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.

© 2017 Microchip Technology Inc.

39.6.2 ADC Mode Register

Name:ADC_MR

Access: 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)

© 2017 Microchip Technology Inc.

STARTUP: Start Up Time

Startup Time = (STARTUP+1) * 8 / ADCClock

SHTIM: Sample & Hold Time

Sample & Hold Time = (SHTIM+1) / ADCClock

© 2017 Microchip Technology Inc.

39.6.3 ADC Channel Enable Register

Name:ADC_CHER

Access: 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.

39.6.4 ADC Channel Disable Register

Name:ADC_CHDR

Access: 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.

© 2017 Microchip Technology Inc.

39.6.5 ADC Channel Status Register

Name:ADC_CHSR

Access: 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.

39.6.6 ADC Status Register

Name:ADC_SR

Access: 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.

© 2017 Microchip Technology Inc.

39.6.7 ADC Last Converted Data Register

Name:ADC_LCDR

Access: 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.

39.6.8 ADC Interrupt Enable Register

Name:ADC_IER

Access: 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.

© 2017 Microchip Technology Inc.

39.6.9 ADC Interrupt Disable Register

Name:ADC_IDR

Access: 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.

39.6.10 ADC Interrupt Mask Register

Name:ADC_IMR

Access: 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.

© 2017 Microchip Technology Inc.

39.6.11 ADC Channel Data Register

Name:ADC_CDRx

Access: 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.

40. Electrical Characteristics

40.1 Absolute Maximum Ratings

Table 40-1: Absolute Maximum Ratings*

Operating Temperature (Industrial).....-40°C to + 85°C
Junction Temperature.....+125°C
Storage Temperature.....-40°C to + 150°C
Voltage on Input Pins with Respect to Ground .....-0.3V to VDDIO+0.3V (+4V max)
Maximum Operating Voltage (VDDCORE, VDDPLL and VDDBU).....1.2V
Maximum Operating Voltage (VDDIOM and VDDIOP).....4.0V
Total DC Output Current on all I/O lines.....350 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.

© 2017 Microchip Technology Inc.

40.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified.

Table 40-2: DC Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
$V_{VDDCORE}$	DC Supply Core			0.9	1.0	1.1	V
V_{VDDBU}	DC Supply Backup			0.9	1.0	1.1	V
V_{VDDPLL}	DC Supply PLL			0.9	1.0	1.1	V
V_{VDDOSC}	DC Supply Oscillator			1.65	–	3.6	V
V_{VDDIOM}	DC Supply Memory I/Os	Selectable by software		1.65/3.0	1.8/3.3	1.95/3.6	V
V_{VDDIOP}	DC Supply Peripheral I/Os			1.65	–	3.6	V
V_{VDDANA}	DC Supply Analog			3.0	3.3	3.6	V
V_{VDDUSB}	DC Supply USB			3.0	3.3	3.6	V
V_{IL}	Input Low-level Voltage	V_{VDDIO} from 3.0V to 3.6V		-0.3	–	0.8	V
		V_{VDDIO} from 1.65V to 1.95V		-0.3	–	$0.3 \times V_{VDDIO}$	V
V_{IH}	Input High-level Voltage	V_{VDDIO} from 3.0V to 3.6V		2	–	V_{VDDIO} + 0.3	V
		V_{VDDIO} from 1.65V to 1.95V		$0.7 \times V_{VDDIO}$	–	V_{VDDIO} + 0.3	V
V_{OL}	Output Low-level Voltage	I_O Max, V_{VDDIO} from 3.0V to 3.6V		–	–	0.4	V
		CMOS (I_O < 0.3 mA) V_{VDDIO} from 1.65V to 1.95V		–	–	0.1	V
		TTL (I_O Max) V_{VDDIO} from 1.65V to 1.95V		–	–	0.4	V
V_{OH}	Output High-level Voltage	I_O Max, V_{VDDIO} from 3.0V to 3.6V		V_{VDDIO} - 0.4	–	–	V
		CMOS (I_O < 0.3 mA) V_{VDDIO} from 1.65V to 1.95V		V_{VDDIO} - 0.1	–	–	V
		TTL (I_O Max) V_{VDDIO} from 1.65V to 1.95V		V_{VDDIO} - 0.4	–	–	V
R_{PULLUP}	Pull-up Resistance	PA0–PA31, PB0–PB31, PC0–PC3, NTRST and NRST		40	75	190	kΩ
		PC4–PC31 V_{VDDIOM} in 1.8V range		240	–	1000	
		PC4–PC31 V_{VDDIOM} in 3.3V range		140	–	450	
I_O	Output Current	PA0–PA31, PB0–PB31, PC0–PC3		–	–	8	mA
		PC4–PC31 in 3.3V range		–	–	2	
		PC4–PC31 in 1.8V range		–	–	4	
I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.0V, MCK = 0 Hz, excluding POR	T_A = 25°C	–	25	–	mA
		All inputs driven TMS, TDI, TCK, NRST = 1	T_A = 85°C	–	–	100	
		On V_{VDDBU} = 1.0V, Logic cells consumption, excluding POR	T_A = 25°C	–	9	–	µA
		All inputs driven WKUP = 0	T_A = 85°C	–	–	25	
I_{IN_lkg}	3.3V I/O Input Leakage Current	V_{IN} = 3.3V or 0		–	±1	–	µA

40.3 Power Consumption

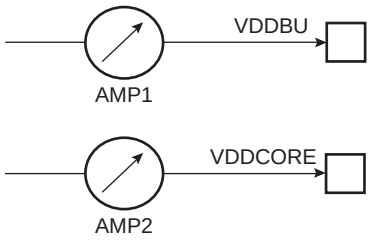
- Typical power consumption of PLLs, Slow Clock and Main Oscillator
- Power consumption of power supply in four different modes: Active, Idle, Ultra Low-power and Backup
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock

40.3.1 Power Consumption versus Modes

The values in Table 40-3 and Table 40-4 are estimated values of the power consumption with operating conditions as follows:

- $V_{DDIOM} = V_{DDIOP} = 3.3V$
- $V_{DDPLL} = 1.0V$
- $V_{DDCORE} = V_{DDBU} = 1.0V$
- $T_A = 25°C$
- There is no consumption on the I/Os of the device

Figure 40-1: Measures Schematics



These figures represent the power consumption estimated on the power supplies.

Table 40-3: Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active	Arm Core clock is 400 MHz. MCK is 133 MHz. All peripheral clocks de-activated. onto AMP2	50	mA
Idle	Idle state, waiting an interrupt. All peripheral clocks de-activated. onto AMP2	20	mA
Ultra low power	Arm Core clock is 500 Hz. All peripheral clocks de-activated. onto AMP2	8	mA
Backup	Device only V_{DDBU} powered onto AMP1	9	µA

© 2017 Microchip Technology Inc.

Table 40-4: Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
PIO Controller	2.5	µA/MHz
USART	7.0	
UHP	5.4	
UDP	4.9	
ADC	4.1	
TWI	4.6	
SPI	4.0	
MCI	6.4	
SSC	7.0	
Timer Counter Channels	1.8	
ISI	4.6	
EMAC	34.6	

40.4 Clock Characteristics

40.4.1 Processor Clock Characteristics

Table 40-5: Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	$V_{VDDCORE}$ = 0.9V T_A = 85°C	–	400	MHz

40.4.2 Master Clock Characteristics

Table 40-6: Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	$V_{VDDCORE}$ = 0.9V T_A = 85°C	–	133	MHz

Note 1: The system clock is the maximum clock at which the system is able to run. It is given by the smallest value of the processor clock, internal bus clock, EBI clock.

40.4.3 XIN Clock Characteristics

Table 40-7: XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPXIN})$	XIN Clock Frequency		–	50	MHz
t_{CPXIN}	XIN Clock Period		20	–	ns
t_{CHXIN}	XIN Clock High Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
t_{CLXIN}	XIN Clock Low Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	ns
C_{IN}	XIN Input Capacitance	Main Oscillator in Bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR). See "PMC Clock Generator Main Oscillator Register".	–	25	pF
R_{IN}	XIN Pull-down Resistor		–	1000	kΩ
V_{IN}	XIN Voltage		V_{DDOSC}	V_{DDOSC}	V

40.4.4 I/O Slew Rates

The IOSR and VDDIOMSEL bits in the EBI_CSA register allow the user to program the rising and falling times of SDRAM signals. Rising/Falling time is given between 30% and 70%.

Table 40-8: SDRAM Clock

VDDIOMSEL	IOSR	Conditions	Rising time (ns)	Falling time (ns)
0	0	1.8V range, FAST slew rate, C_{LOAD} = 10 pF	0.8	1.0
0	1	1.8V range, SLOW slew rate, C_{LOAD} = 10 pF	1.6	1.5
1	0	3.3V range, FAST slew rate, C_{LOAD} = 10 pF	0.8	0.8
1	1	3.3V range, SLOW slew rate, C_{LOAD} = 10 pF	1.5	1.6

Table 40-9: SDRAM Data, Address and Control

VDDIOMSEL	IOSR	Conditions	Rising time (ns)	Falling time (ns)
0	0	1.8V range, FAST slew rate, C_{LOAD} = 30 pF	1.5	1.5
0	1	1.8V range, SLOW slew rate, C_{LOAD} = 30 pF	1.7	1.8
1	0	3.3V range, FAST slew rate, C_{LOAD} = 50 pF	2.8	3.1
1	1	3.3V range, SLOW slew rate, C_{LOAD} = 50 pF	6.0	5.7

© 2017 Microchip Technology Inc.

40.5 Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

40.5.1 32 kHz Oscillator Characteristics

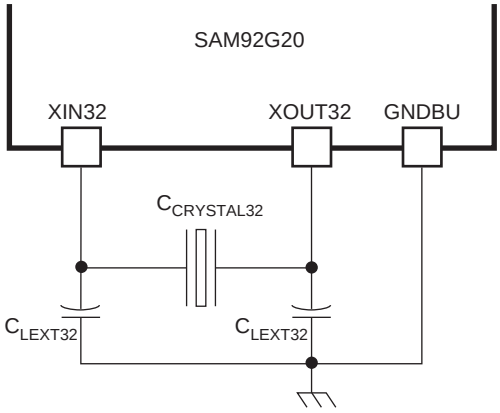
Table 40-10: 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			–	32.768	–	kHz
$C_{CRYSTAL32}$	Load Capacitance	Crystal @ 32.768 kHz		6	–	12.5	pF
C_{LEXT32}[2]	External Load Capacitance	$C_{CRYSTAL32}$ = 6 pF		–	6	–	pF
		$C_{CRYSTAL32}$ = 12.5 pF		–	19	–	pF
	Duty Cycle			40	–	60	%
t_{START}	Startup Time	V_{DDBU} = 1.0V, R_S = 50 kΩ[1]	$C_{CRYSTAL32}$ = 6 pF	–	–	300	ms
			$C_{CRYSTAL32}$ = 12.5 pF	–	–	900	ms
		V_{DDBU} = 1.0V R_S = 100 kΩ[1]	$C_{CRYSTAL32}$ = 6 pF	–	–	600	ms
			$C_{CRYSTAL32}$ = 12.5 pF	–	–	1200	ms

Note 1: R_S is the equivalent series resistance.

2: C_{LEXT32} is determined by taking into account internal, parasitic and package load capacitance.

Figure 40-2: 32 kHz Oscillator Schematic



40.5.2 32 kHz Crystal Characteristics

Table 40-11: 32 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Crystal @ 32.768 kHz	–	50	100	kΩ
C_m	Motional Capacitance	Crystal @ 32.768 kHz	–	–	3	fF
C_{SHUNT}	Shunt Capacitance	Crystal @ 32.768 kHz	–	–	2	pF

40.5.3 RC Oscillator Characteristics

Table 40-12: RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRCz})$	Crystal Oscillator Frequency		20	32	44	kHz
	Duty Cycle		45	–	55	%
t_{START}	Startup Time		–	–	75	µs

40.5.4 Slow Clock Selection

Table 40-13 defines the states for OSCSEL signal.

Table 40-13: Slow Clock Selection

OSCSEL	Slow Clock	Startup Time (Max)
0	Internal RC	150 µs
1	External 32768 Hz	1200 ms

The startup counter delay for the slow clock oscillator depends on the OSCSEL signal. The 32768 Hz startup delay is 1200 ms whereas it is 150 µs for the internal RC oscillator (refer to Table 40-13). The pin OSCSEL must be tied either to GND or VDDBU for correct operation of the device.

© 2017 Microchip Technology Inc.

40.5.5 Main Oscillator Characteristics

Table 40-14: Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
1/(t_{CPMAIN})	Crystal Oscillator Frequency		3	16	20	MHz
$C_{CRYSTAL}$	Crystal Load Capacitance		12.5	–	17.5	pF
C_{LEXT}[7]	External Load Capacitance	$C_{CRYSTAL}$ = 12.5 pF[6]	–	16	–	pF
		$C_{CRYSTAL}$ = 17.5 pF[6]	–	26	–	pF
	Duty Cycle		40	50	60	%
t_{START}	Startup Time	V_{DDOSC} = 3 to 3.6V C_S = 3 pF[1] 1/(t_{CPMAIN}) = 3 MHz C_S = 7 pF[1] 1/(t_{CPMAIN}) = 8 MHz C_S = 7 pF[1] 1/(t_{CPMAIN}) = 16 MHz C_S = 7 pF[1] 1/(t_{CPMAIN}) = 20 MHz	–	–	 14.5 4 1.4 1	ms
I_{DDST}	Standby Current Consumption	Standby mode	–	–	1	µA
P_{ON}	Drive Level	@ 3 MHz @ 8 MHz @ 16 MHz @ 20 MHz	–	–	15 30 50 50	µW
$I_{DD\ ON}$	Current Dissipation	@ 3 MHz[2] @ 8 MHz[3] @ 16 MHz[4] @ 20 MHz[5]	–	280 380 500 580	380 510 630 750	µA

Note 1: C_{SHUNT} is the shunt capacitance.

2: R_S = 100 to 200 Ω; C_{SHUNT} = 2.0 to 2.5 pF; C_m = 2 to 1.5 pF (typ, worst case) using 1 kΩ serial resistor on XOUT.

3: R_S = 50 to 100 Ω; C_{SHUNT} = 2.0 to 2.5 pF; C_m = 4 to 3 pF (typ, worst case).

4: R_S = 25 to 50 Ω; C_{SHUNT} = 2.5 to 3.0 pF; C_m = 7 to 5 pF (typ, worst case).

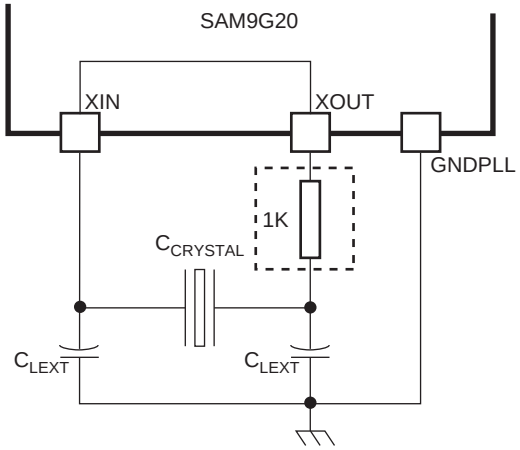
5: R_S = 20 to 50 Ω; C_{SHUNT} = 3.2 to 4.0 pF; C_m = 10 to 8 pF (typ, worst case).

6: Additional user load capacitance should be subtracted from C_{LEXT}.

7: C_{LEXT} is determined by taking into account internal, parasitic and package load capacitance.

© 2017 Microchip Technology Inc.

Figure 40-3: Main Oscillator Schematic



40.5.6 Crystal Characteristics

Table 40-15: Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Fundamental @ 3 MHz	–	–	200	Ω
		Fundamental @ 8 MHz			100	
		Fundamental @ 16 MHz			80	
		Fundamental @ 20 MHz			50	
C_m	Motional Capacitance		–	–	8	pF
C_{SHUNT}	Shunt Capacitance		–	–	7	pF

40.5.7 PLL Characteristics

Table 40-16: PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	Refer to following table	400	–	800	MHz
f_{IN}	Input Frequency		2	–	32	MHz
I_{PLL}	Current Consumption	Active mode	–	3.6	8	mA
		Standby mode	–	–	1	µA
t_{START}	Startup Time		–	–	50	µs

© 2017 Microchip Technology Inc.

Following configuration of ICPLLA and OUTA must be done for each PLLA frequency range.

Table 40-17: PLLA Frequency Regarding ICPLLA and OUTA

PLL Frequency Range (MHz)	ICPLLA	OUTA	
745–800	0	0	0
695–750	0	0	1
645–700	0	1	0
595–650	0	1	1
545–600	1	0	0
495–550	1	0	1
445–500	1	1	0
400–450	1	1	1

Table 40-18: PLLB Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output Frequency	Field CKGR_PLL.OUT = 00	30	–	100	MHz
f_{in}	Input Frequency		2	–	32	MHz
I_{PLL}	Current Consumption	Active mode @ 100 MHz	–	–	1.2	mA
		Standby mode	–	–	1	µA
t_{START}	Startup Time		–	–	100	µs

40.6 I/O Characteristics

Criteria used to define the maximum frequency of the I/Os:

- Output duty cycle (40%–60%)
- Minimum output swing: 100 mV to VDDIO - 100 mV
- Addition of rising and falling time inferior to 75% of the period

Table 40-19: I/O Characteristics

Symbol	Parameter	Conditions		Min	Max	Unit
FreqMax	VDDIOP powered pins frequency	3.3V domain(1)	Max. external load = 20 pF	–	66	MHz
			Max. external load = 40 pF	–	66	MHz
		1.8V domain(2)	Max. external load = 20 pF	–	66	MHz
			Max. external load = 40 pF	–	66	MHz

Note 1: V_{VDDIOP} from 3.0V to 3.6V

2: V_{VDDIOP} from 1.65V to 1.95V

40.7 ADC Characteristics

Table 40-20: ADC Characteristics 9-bit Mode

Code	Parameter	Condition & Notes	Min	Typ	Max	Unit
R	Resolution		–	9	–	Bits
INL	Integral Non-linearity		–	–	±2	LSB
DNL	Differential Non-linearity	No missing code	-1	–	+2	LSB
E_O	Offset Error	Not including V_{REF}N error	–	–	±3	LSB
E_G	Gain Error		-1.5	–	3.5	LSB
f_{CLK}	Clock Frequency	No missing code	–	–	5	MHz
	Conversion Rate:					
	Conversion Time	Clock Frequency = 5 MHz		2		µs
	Track-and-Hold Acquisition Time	see Note (1)	500			ns
	Throughput Rate	@ f_{CLK} = 5 MHz with TTH = 1 µs			330	ksps

Table 40-21: ADC Characteristics 10-bit Mode

Code	Parameter	Condition & Notes	Min	Typ	Max	Unit
R	Resolution		–	10	–	Bits
INL	Integral Non-linearity		–	–	±2	LSB
DNL	Differential Non-linearity	No missing code	-1	–	±1	LSB
E_O	Offset Error	Not including V_{REF}N error		–	±3	LSB
E_G	Gain Error		-1.5	–	3.5	LSB
f_{CLK}	Clock Frequency	No missing code	–	–	1	MHz
	Conversion Rate:					
	Conversion Time	Clock Frequency = 5 MHz		10		µs
	Track-and-Hold Acquisition Time	see Note (1)	500			ns
	Throughput Rate	@ f_{CLK} = 5 MHz with TTH = 1 µs			95	ksps

Note 1: In worst case, the Track-and-Hold Acquisition Time is given by:

$$TTH\,(\mu s) = 1.2 + (0.09 \times Z_{IN})(kOhm)$$

In case of very high input impedance, this value must be respected in order to guarantee the correct converted value. An internal input current buffer supplies the current required for the low input impedance (1 mA max).

To achieve optimal performance of the ADC, the analog power supply VDDANA and the ADVREF input voltage must be decoupled with a 4.7 µF capacitor in parallel with a 100 nF capacitor.

© 2017 Microchip Technology Inc.

Table 40-22: External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Unit
ADVREF Input Voltage Range		2.4	–	VDDANA	V
ADVREF Average Current		–	–	220	µA
Current Consumption on VDDANA		–	300	620	µA

Table 40-23: Analog Inputs

Parameter	Min	Typ	Max	Unit
Input Voltage Range	0	–	ADVREF	V
Input Leakage Current	–	–	1	µA
Input Capacitance	–	8	–	pF

40.8 USB Transceiver Characteristics

Table 40-24: USB Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level		–	–	0.8	V
V_{IH}	High Level		2.0	–	–	V
V_{DI}	Differential Input Sensitivity	|(D+) - (D-)|	0.2	–	–	V
V_{CM}	Differential Input Common Mode Range		0.8	–	2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line	–	–	9.18	pF
I_{lkg}	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	- 10	–	+ 10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%	–	27	–	Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	0.0	–	0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	2.8	–	3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measurement conditions described in Figure 40-23	1.3	–	2.0	V
Pull-up and Pull-down Resistor						
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)		0.900	–	1.575	kΩ
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)		1.425	–	3.090	kΩ
R_{PD}	Bus Pull-down resistor		14.25	–	24.8	kΩ

© 2017 Microchip Technology Inc.

40.9 Core Power Supply POR Characteristics

Table 40-25: Power-On-Reset Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T+}	Threshold Voltage Rising	Minimum Slope of +1.0V/100ms	0.5	0.7	0.89	V
V_{T-}	Threshold Voltage Falling		0.4	0.6	0.85	V
t_{RST}	Reset Time		30	70	130	µs

40.10 EBI Timings

SMC timings are given in Max (T_A = 85°C, VDDCORE = 0.9V) corner.

Timings are given assuming a capacitance load on data, control and address pads as defined in Table 40-26.

Table 40-26: Capacitance Load

	Corner
I/O Supply	Max
3.3V	50 pF
1.8V	30 pF

In the following tables t_{CPMCK} is MCK period.

40.10.1 Read Timings

Table 40-27: SMC Read Signals - NRD Controlled (READ_MODE = 1)

		Min		
Symbol	Parameter	1.8V VDDIOM Supply	3.3V VDDIOM Supply	Unit
NO HOLD SETTINGS (nrd hold = 0)				
SMC_1	Data Setup before NRD High	13.0	11.4	ns
SMC_2	Data Hold after NRD High	0	0	ns
HOLD SETTINGS (nrd hold ≠ 0)				
SMC_3	Data Setup before NRD High	9.6	8.0	ns
SMC_4	Data Hold after NRD High	0	0	ns
HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold = 0)				
SMC_5	NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 Valid before NRD High	(nrd setup + nrd pulse) * t_{CPMCK} - 0.4	(nrd setup + nrd pulse) * t_{CPMCK} - 0.4	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} + 0.5	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} + 0.4	ns
SMC_7	NRD Pulse Width	nrd pulse * t_{CPMCK} + 0.2	nrd pulse * t_{CPMCK} + 0.1	ns

© 2017 Microchip Technology Inc.

Table 40-28: SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
NO HOLD SETTINGS (ncs rd hold = 0)				
SMC_8	Data Setup before NCS High	12.6	11.0	ns
SMC_9	Data Hold after NCS High	0	0	ns
HOLD SETTINGS (ncs rd hold ≠ 0)				
SMC_{10}	Data Setup before NCS High	9.2	7.6	ns
SMC_{11}	Data Hold after NCS High	0	0	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)				
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS High	(ncs rd setup + ncs rd pulse) * t_{CPMCK} - 0.8	(ncs rd setup + ncs rd pulse) * t_{CPMCK} - 0.6	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup) * t_{CPMCK} -0.2	(ncs rd setup + ncs rd pulse - nrd setup) * t_{CPMCK} -0.3	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} + 0.1	ncs rd pulse length * t_{CPMCK} - 0.1	ns

40.10.2 Write Timings

Table 40-29: SMC Write Signals - NWE controlled (WRITE_MODE = 1)

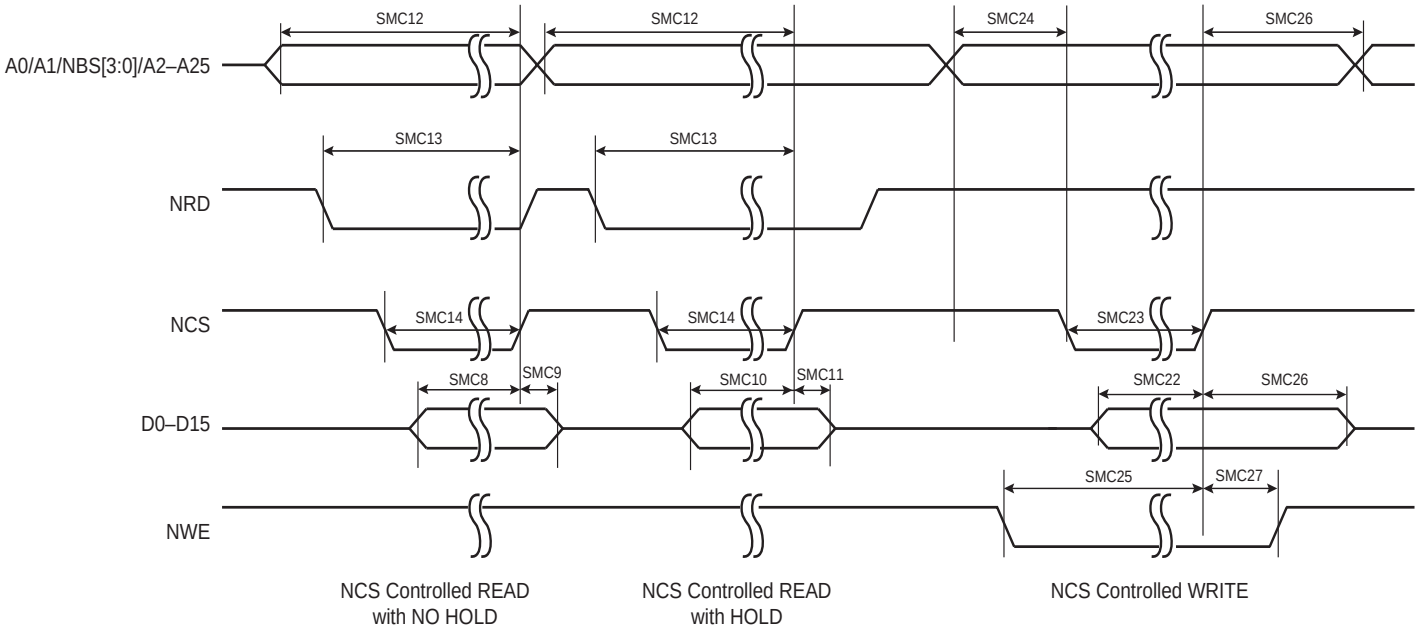
Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)				
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} - 0.8	nwe pulse * t_{CPMCK} - 0.8	ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} - 0.4	nwe pulse * t_{CPMCK} - 0.4	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NWE low	nwe setup * t_{CPMCK} - 0.5	nwe setup * t_{CPMCK} - 0.5	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 0.2	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 0.2	ns
HOLD SETTINGS (nwe hold ≠ 0)				
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 change	nwe hold * t_{CPMCK} - 0.4	nwe hold * t_{CPMCK} - 0.4	ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold) * t_{CPMCK} -0.3	(nwe hold - ncs wr hold) * t_{CPMCK} - 0.3	ns
NO HOLD SETTINGS (nwe hold = 0)				
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25, NCS change(1)	3.4	3.1	ns

Note 1: hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 40-30: SMC Write NCS Controlled (WRITE_MODE = 0)

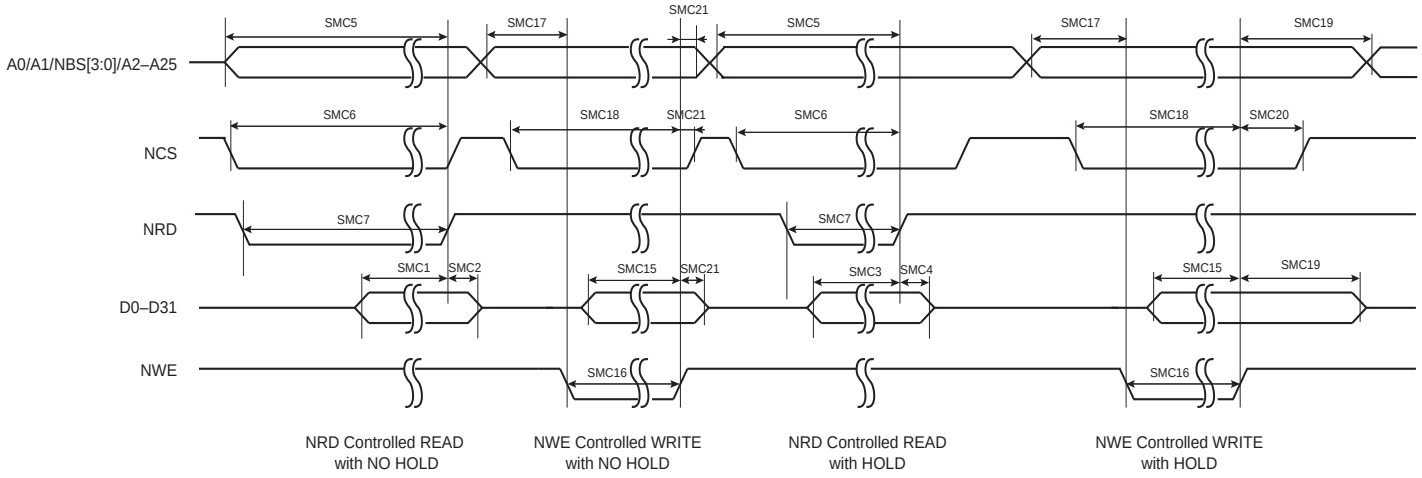
Symbol	Parameter	Min		Unit
		1.8V VDDIOM Supply	3.3V VDDIOM Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} - 0.8	ncs wr pulse * t_{CPMCK} - 0.7	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} + 0.1	ncs wr pulse * t_{CPMCK} - 0.1	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS low	ncs wr setup * t_{CPMCK} - 0.6	ncs wr setup * t_{CPMCK} - 0.6	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse) * t_{CPMCK} - 0.4	(ncs wr setup - nwe setup + ncs pulse) * t_{CPMCK} - 0.3	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25, change	ncs wr hold * t_{CPMCK} - 0.2	ncs wr hold * t_{CPMCK} - 0.2	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold) * t_{CPMCK} - 0.2	(ncs wr hold - nwe hold) * t_{CPMCK} - 0.2	ns

Figure 40-4: SMC Timings - NCS Controlled Read and Write



© 2017 Microchip Technology Inc.

Figure 40-5: SMC Timings - NRD Controlled Read and NWE Controlled Write



40.11 SDRAMC Timings

Timings are given assuming a capacitance load on data, control and address pads as defined in Table 40-31, as well as the SDCK pad as defined in Table 40-32.

Table 40-31: Capacitance Load on data, control and address pads

	Corner
I/O Supply	Max
3.3V	50 pF
1.8V	30 pF

Table 40-32: Capacitance Load on SDCK pad

	Corner
I/O Supply	Max
3.3V	10 pF
1.8V	10 pF

The SDRAM Controller satisfies the timings of standard PC100, PC133 (3.3V supply) and Mobile SDRAM (1.8V supply) that are given in Table 40-33.

Table 40-33: SDRAMC Characteristics

Timings Standard	Parameter	Supply	Min	Max	Unit
PC100	SDRAM Controller Clock Frequency	3.3V	–	100	MHz
	Control/Address/Data In Setup(1)(2)		2	–	ns
	Control/Address/Data In Hold(1)(2)		1	–	ns
	Data Out Access time after SDCK rising		–	6	ns
	Data Out change time after SDCK rising		3	–	ns
PC133	SDRAM Controller Clock Frequency	3.3V	–	133	MHz
	Control/Address/Data In Setup(1)(2)		1.5	–	ns
	Control/Address/Data In Hold(1)(2)		0.8	–	ns
	Data Out Access time after SDCK rising		–	5.4	ns
	Data Out change time after SDCK rising		3.0	–	ns
Mobile SDRAM	SDRAM Controller Clock Frequency	1.8V	–	133 / 100 (3)	MHz
	Control/Address/Data In Setup(1)(2)		2.0	–	ns
	Control/Address/Data In Hold(1)(2)		1.0	–	ns
	Data Out Access time after SDCK rising		–	6.0 / 8.0 (3)	ns
	Data Out change time after SDCK rising		3.0	–	ns

Note 1: Control is the set of following signals: SDCKE, SDCS, RAS, CAS, SDA10, BAx, DQMx, and SDWE.

2: Address is the set of A0–A9, A11–A13.

3: 133 MHz with CAS Latency = 3, 100 MHz with CAS Latency = 2.

© 2017 Microchip Technology Inc.

40.12 Peripheral Timings

40.12.1 EMAC

Timings are given assuming a capacitance load on data and clock as defined in Table 40-34.

Table 40-34: Capacitance Load on Data, Clock Pads

I/O Supply	Corner
	Max
3.3V	20 pF
1.8V	20 pF

The Ethernet controller satisfies the timings of standard in Max corner.

Table 40-35: EMAC Signals Relative to EMDC

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_1$	Setup for EMDIO from EMDC rising	10	–
$EMAC_2$	Hold for EMDIO from EMDC rising	10	–
$EMAC_3$	EMDIO toggling from EMDC rising	0	300

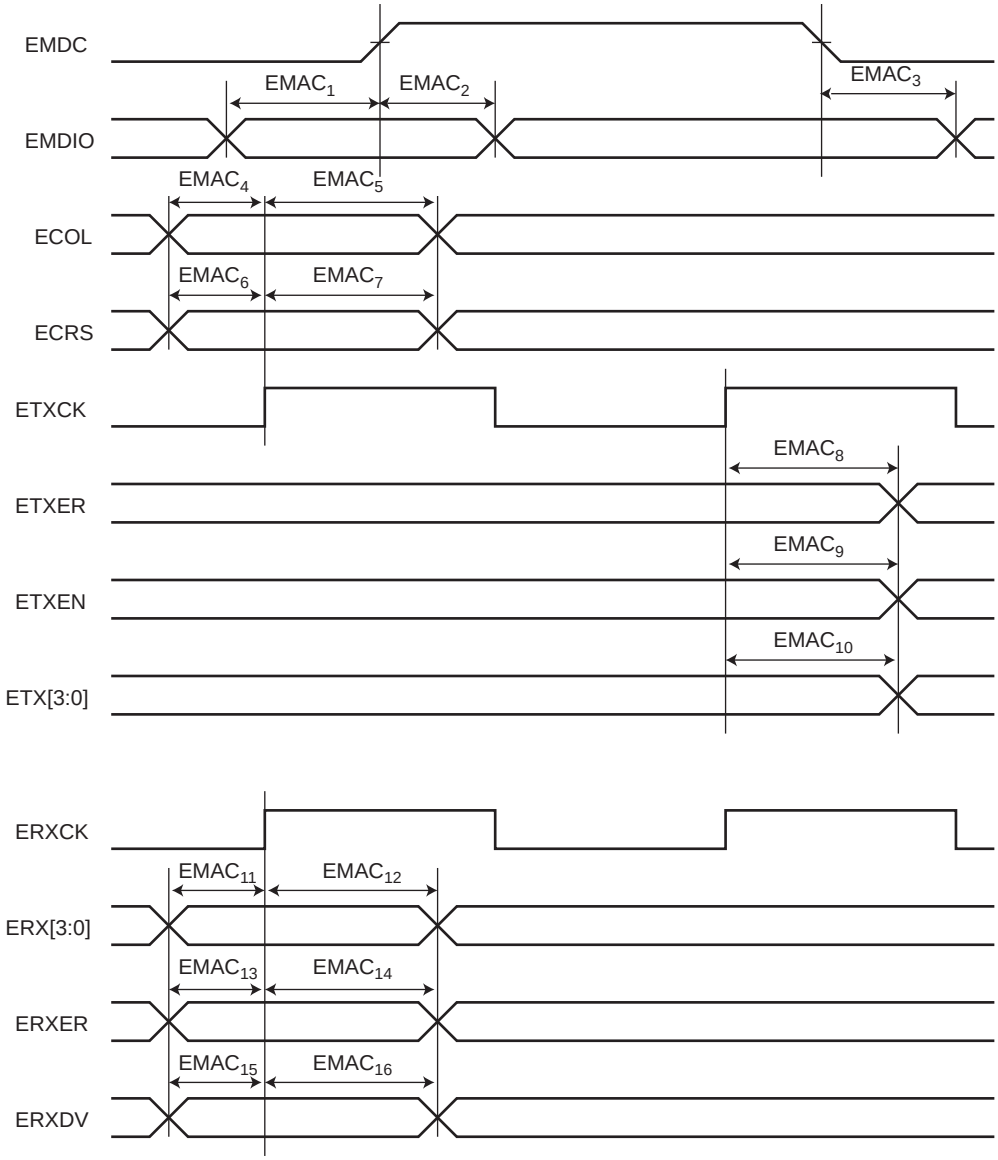
40.12.1.1 MII Mode

Table 40-36: EMAC MII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_4$	Setup for ECOL from ETXCK rising	10	–
$EMAC_5$	Hold for ECOL from ETXCK rising	10	–
$EMAC_6$	Setup for ECRS from ETXCK rising	10	–
$EMAC_7$	Hold for ECRS from ETXCK rising	10	–
$EMAC_8$	ETXER toggling from ETXCK rising	10	25
$EMAC_9$	ETXEN toggling from ETXCK rising	10	25
$EMAC_{10}$	ETX toggling from ETXCK rising	10	25
$EMAC_{11}$	Setup for ERX from ERXCK	10	–
$EMAC_{12}$	Hold for ERX from ERXCK	10	–
$EMAC_{13}$	Setup for ERXER from ERXCK	10	–
$EMAC_{14}$	Hold for ERXER from ERXCK	10	–
$EMAC_{15}$	Setup for ERXDV from ERXCK	10	–
$EMAC_{16}$	Hold for ERXDV from ERXCK	10	–

Note 1: VDDIO from 3.0V to 3.6V, maximum external capacitor = 20 pF

Figure 40-6: EMAC MII Mode



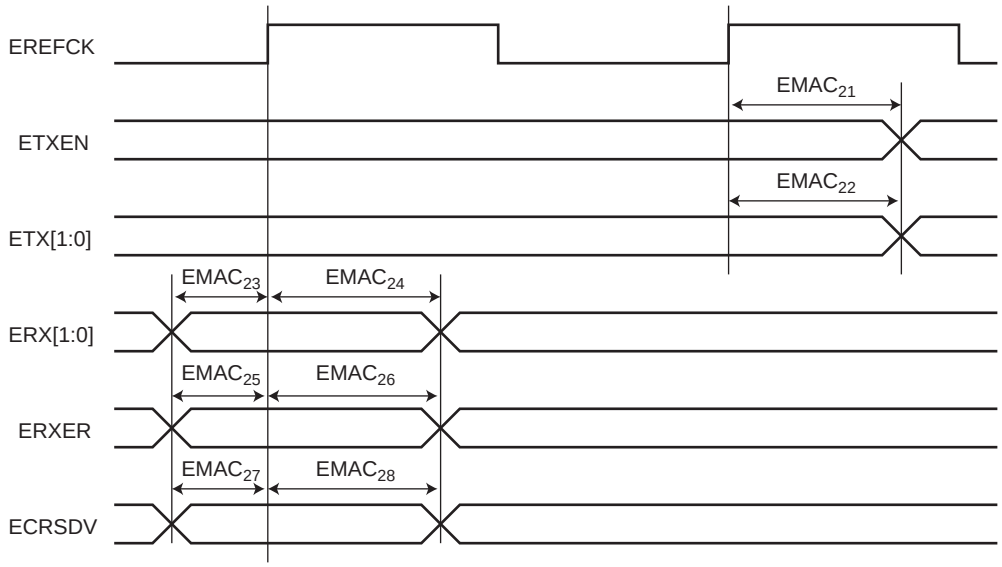
© 2017 Microchip Technology Inc.

40.12.1.2 RMII Mode

Table 40-37: RMII Mode

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_{21}$	ETXEN toggling from EREFCK rising	2	16
$EMAC_{22}$	ETX toggling from EREFCK rising	2	16
$EMAC_{23}$	Setup for ERX from EREFCK rising	4	–
$EMAC_{24}$	Hold for ERX from EREFCK rising	2	–
$EMAC_{25}$	Setup for ERXER from EREFCK rising	4	–
$EMAC_{26}$	Hold for ERXER from EREFCK rising	2	–
$EMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	–
$EMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	–

Figure 40-7: EMAC RMII Timings



40.12.2 SPI

40.12.2.1 Maximum SPI Frequency

The following formulas give maximum SPI frequency in Master read and write modes and in Slave read and write modes.

- Master Write Mode

  The SPI is only sending data to a slave device such as an LCD, for example. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 40.6 “I/O Characteristics”), the maximum SPI frequency is the one from the pad.

- Master Read Mode

$$f_{SPCK}Max = \frac{1}{SPI_0(orSPI_3) + t_{valid}}$$

  t_{valid} is the slave time response to output data after deleting an SPCK edge. For a non-volatile memory with t_{valid} (or t_V) = 12 ns Max, f_{SPCK}Max = 37.7 MHz @ V_{DDIO} = 3.3V.

- Slave Read Mode

  In slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings SPI_7/SPI_8(or SPI_{10}/SPI_{11}). Since this gives a frequency well above the pad limit, the limit in slave read mode is given by SPCK pad.

© 2017 Microchip Technology Inc.

- Slave Write Mode

$$f_{SPCK}Max = \frac{1}{2x(SPI_{6max}(orSPI_{9max}) + t_{su})}$$

For 3.3V I/O domain and SPI6, f_{SPCK}Max = 18.7 MHz. t_{su} is the setup time from the master before sampling data.

40.12.2.2 SPI Timings

SPI timings are given assuming a capacitance load on MISO, SPCK and MOSI as defined in Table 40-38.

Table 40-38: Capacitance Load for MISO, SPCK and MOSI

	Corner
Supply	Max
1.8V/3.3V	20 pF

Figure 40-8: SPI Master Mode 1 and 2

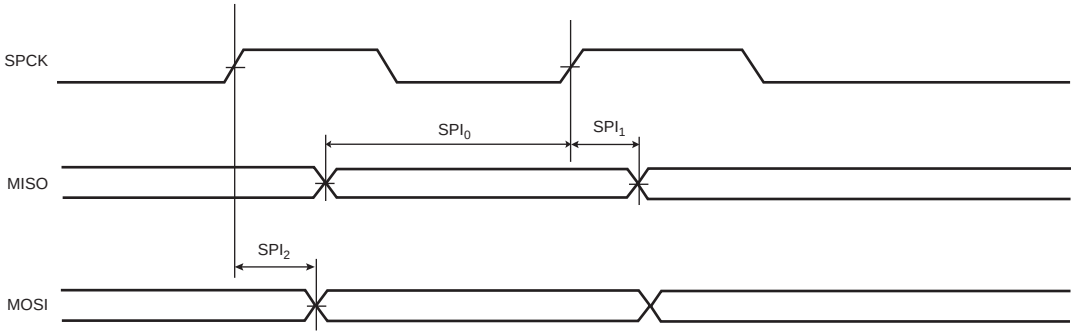
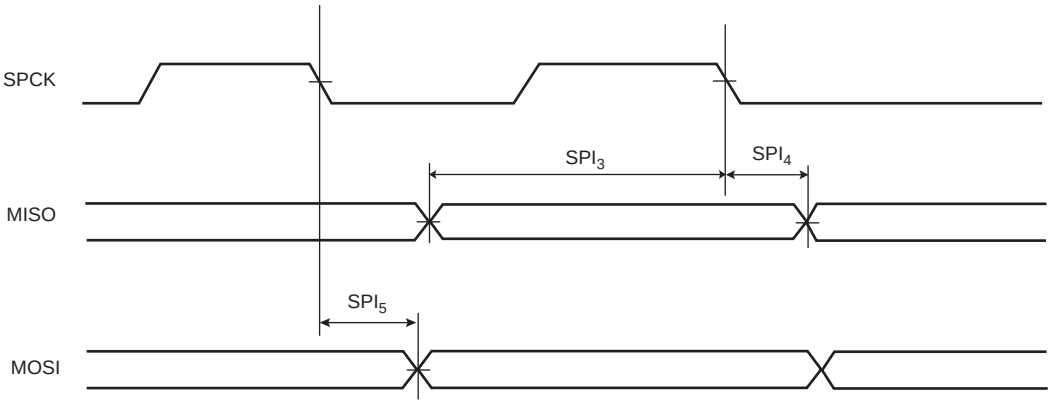


Figure 40-9: SPI Master Mode 0 and3



© 2017 Microchip Technology Inc.

Figure 40-10: SPI Slave Mode 0 and 3

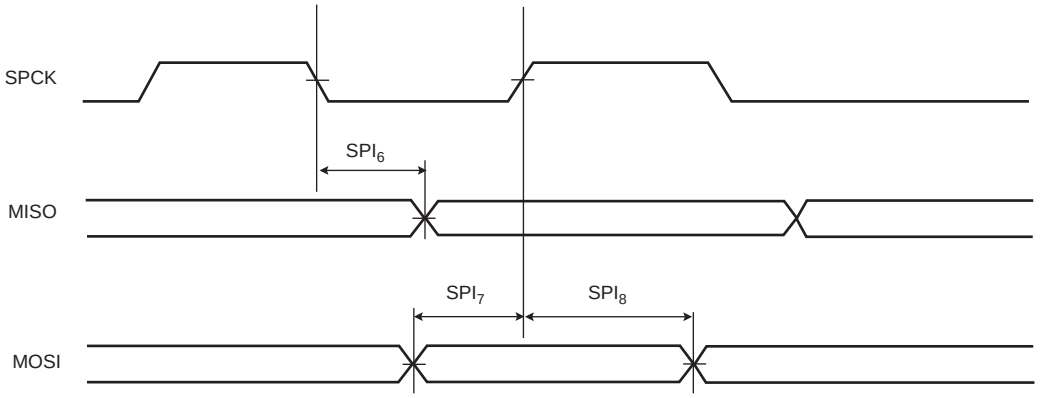


Figure 40-11: SPI Slave Mode 1 and 2

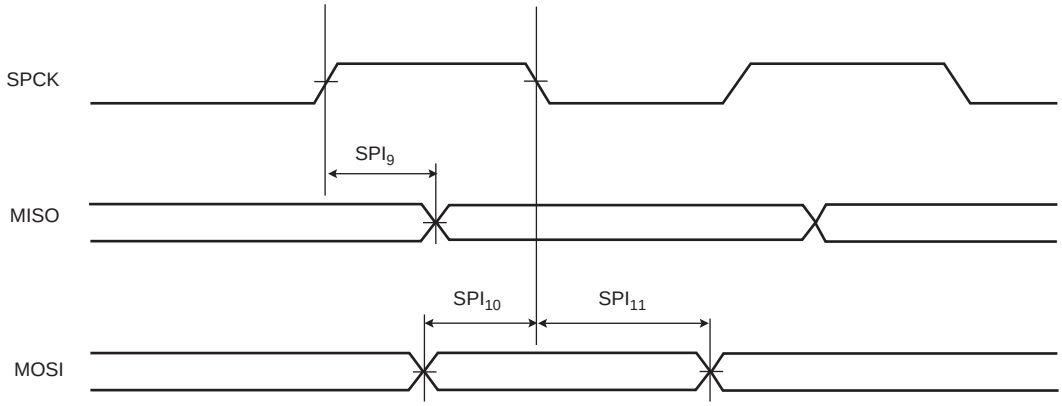


Figure 40-12: SPI Slave Mode - NPCS Timings

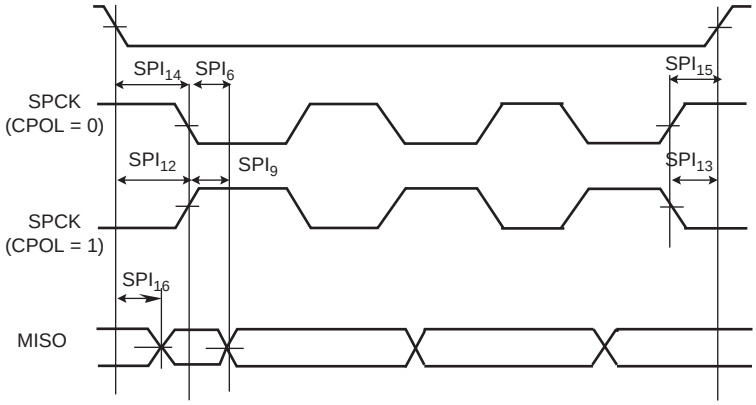


Table 40-39: SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_0	MISO Setup time before SPCK rises	Master mode Max corner	14.5 + 0.5 * t_{CPMCK}	–	ns
SPI_1	MISO Hold time after SPCK rises		-11.6 - 0.5 * t_{CPMCK}	–	ns
SPI_2	SPCK rising to MOSI		-0.6	–	ns
SPI_3	MISO Setup time before SPCK falls		14.7 + 0.5 * t_{CPMCK}	–	ns
SPI_4	MISO Hold time after SPCK falls		-11.7 - 0.5 * t_{CPMCK}	–	ns
SPI_5	SPCK falling to MOSI		-0.7	–	ns
SPI_6	SPCK falling to MISO	Slave mode Max corner	14.7	–	ns
SPI_7	MOSI Setup time before SPCK rises		5.8	–	ns
SPI_8	MOSI Hold time after SPCK rises		-2.7	–	ns
SPI_9	SPCK rising to MISO		14.7	–	ns
SPI_{10}	MOSI Setup time before SPCK falls		23.8	–	ns
SPI_{11}	MOSI Hold time after SPCK falls		15.3	–	ns
SPI_{12}	NPCS0 setup to SPCK rising		-8.3	–	ns
SPI_{13}	NPCS0 hold after SPCK falling		10.9	–	ns
SPI_{14}	NPCS0 setup to SPCK falling		-8.3	–	ns
SPI_{15}	NPCS0 hold after SPCK rising		10.9	–	ns
SPI_{16}	NPCS0 falling to MISO valid		14.7	10.7	ns

Note 1: C_{LOAD} is 8 pF for MISO and 6 pF for SPCK and MOSI.

40.12.3 SSC Timings

Timings are given assuming a capacitance load on clock and data signals as defined in Table 40-40.

Table 40-40: Capacitance Load

	Corner
Supply	**Max**
3.3V	30 pF
1.8V	20 pF

© 2017 Microchip Technology Inc.

Figure 40-13: SSC Transmitter, TK and TF in Output

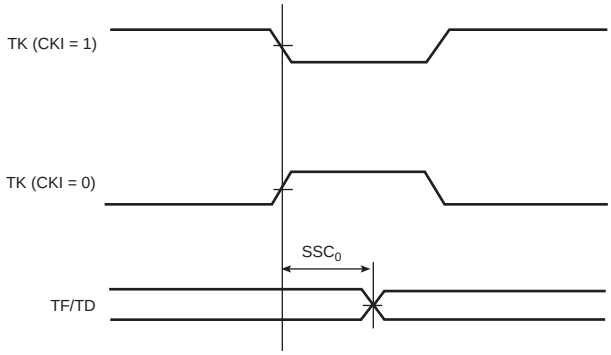


Figure 40-14: SSC Transmitter, TK in Input and TF in Output

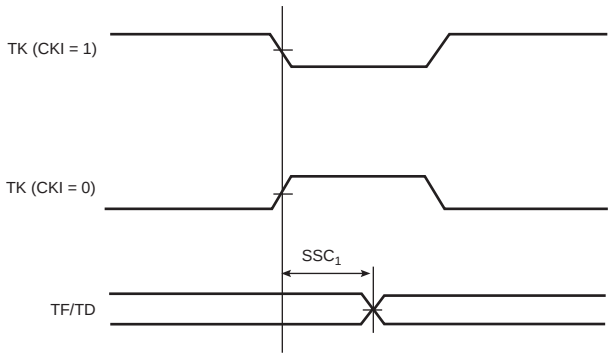


Figure 40-15: SSC Transmitter, TK in Output and TF in Input

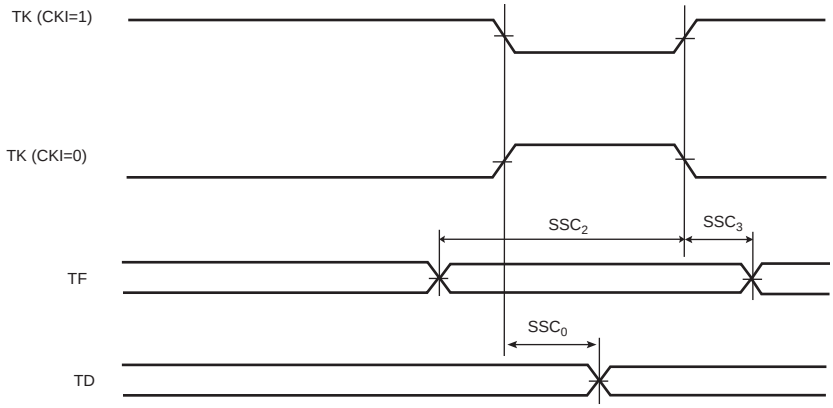


Figure 40-16: SSC Transmitter, TK and TF in Input

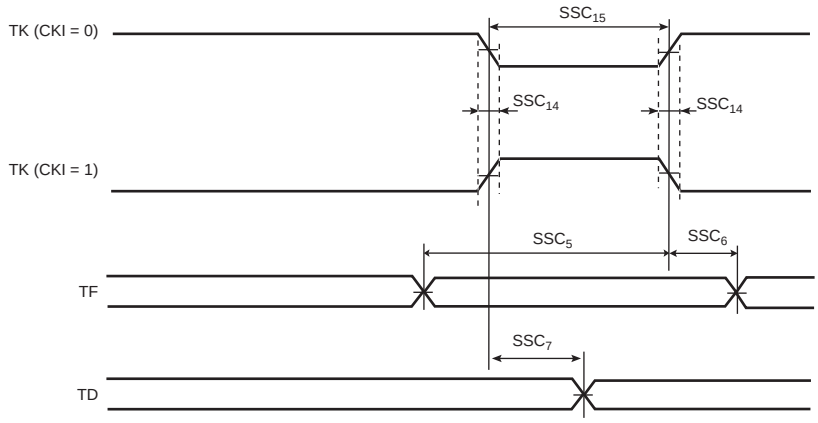


Figure 40-17: SSC Receiver RK and RF in Input

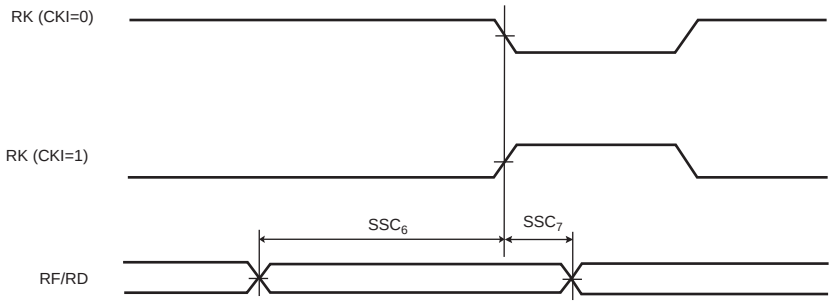
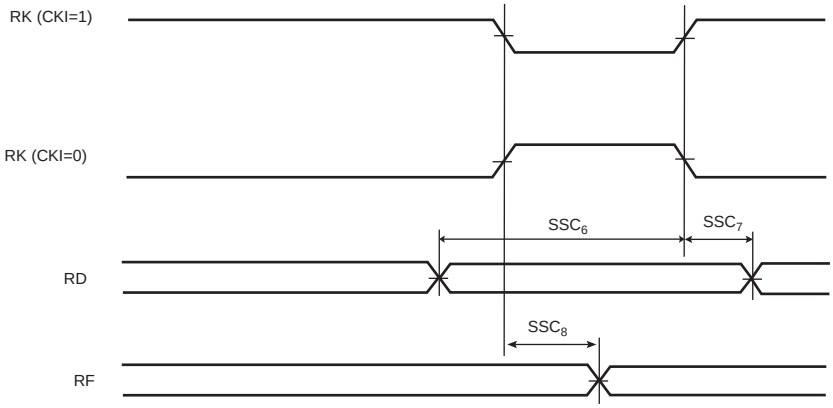


Figure 40-18: SSC Receiver, RK in input and RF in Output



© 2017 Microchip Technology Inc.

Figure 40-19: SSC Receiver, RK and RF in Output

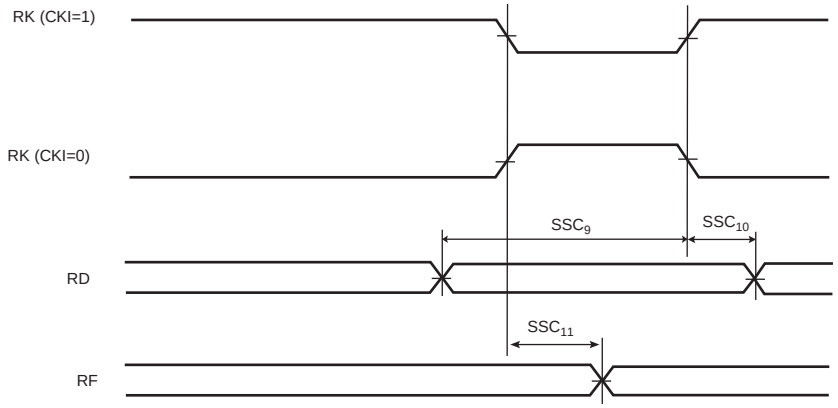


Figure 40-20: SSC Receiver, RK in Output and RF in Input

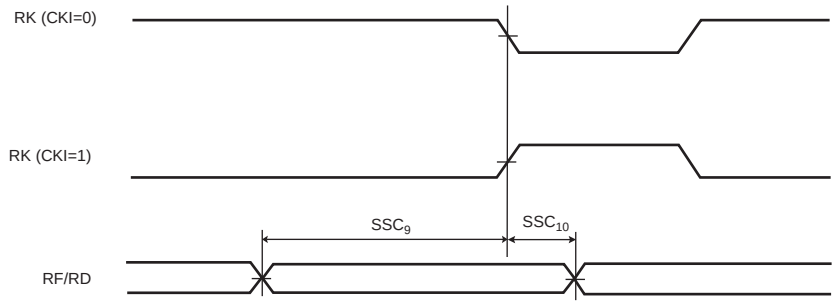


Table 40-41: SSC Timings 3.3V Domain

Symbol	Parameter	Conditions	Min	Max	Unit
		Transmitter			
SSC_0	TK edge to TF/TD (TK output, TF output)		-2.7	3.7	ns
SSC_1	TK edge to TF/TD (TK input, TF output)		-2.7	3.7	ns
SSC_2	TF setup time before TK edge (TK output)		15.5 - t_{CPMCK}	–	ns
SSC_3	TF hold time after TK edge (TK output)		t_{CPMCK} - 9.5	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)		-2.6	3.8	ns
		STTDLY = 0 START = 4, 5 or 7	-2.6 + (2 × t_{CPMCK})	3.8 + (2 × t_{CPMCK})	
SSC_5	TF setup time before TK edge (TK input)		6.0 - t_{CPMCK}	–	ns
SSC_6	TF hold time after TK edge (TK input)		t_{CPMCK} - 2.0	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)		7.7	11.3	ns
		STTDLY = 0 START = 4, 5 or 7	7.7 + (3 × t_{CPMCK})	11.3 + (3 × t_{CPMCK})	
		Receiver			
SSC_8	RF/RD setup time before RK edge (RK input)		6.2 - t_{CPMCK}	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)		t_{CPMCK} + 6.1	–	ns
SSC_{10}	RK edge to RF (RK input)		7.3	16.5	ns

© 2017 Microchip Technology Inc.

Table 40-41: SSC Timings 3.3V Domain (Continued)

Symbol	Parameter	Conditions	Min	Max	Unit
SSC_{11}	RF/RD setup time before RK edge (RK output)		15.9 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)		t_{CPMCK} - 6.5	–	ns
SSC_{13}	RK edge to RF (RK output)		-2.8	–	ns
SSC_{14}(1)	TK rise time or fall time	10 to 90%	–	10	ns
SSC_{15}(1)	TK low or high time	$V_{TK}>V_{IH}$ or $V_{TK}<V_{IL}$	3.4 + (3 x t_{CPMCK})	–	ns

Note 1: Timings SSC_{14} and SSC_{15} also apply when RK is used instead of TK (SSC_TCMR.CKS = RK).

Table 40-42: SSC Timings 1.8V Domain

Symbol	Parameter	Conditions	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)		-3.2	4.2(1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)		-3.2(1)	4.2(1)	ns
SSC_2	TF setup time before TK edge (TK output)		21.2 - t_{CPMCK}	–	ns
SSC_3	TF hold time after TK edge (TK output)		t_{CPMCK} - 13.4	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)		-3.1(1)	4.3(1)	ns
		STTDLY = 0 START = 4, 5 or 7	-3.1 + ($2 \times t_{CPMCK}$)(1)	4.3 + ($2 \times t_{CPMCK}$)(1)	
SSC_5	TF setup time before TK edge (TK input)		7.4 - t_{CPMCK}	–	ns
SSC_6	TF hold time after TK edge (TK input)		t_{CPMCK} - 0.7	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)		11.6	17.0	ns
		STTDLY = 0 START = 4, 5 or 7	11.6 + ($3 \times t_{CPMCK}$)	17.0 + ($3 \times t_{CPMCK}$)	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)		7.4 - t_{CPMCK}	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)		t_{CPMCK} + 7.4	–	ns
SSC_{10}	RK edge to RF (RK input)		11.3	22.3	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)		21.6 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)		t_{CPMCK} - 10.5	–	ns
SSC_{13}	RK edge to RF (RK output)		-3.2	–	ns
SSC_{14}(1)	TK rise time or fall time	10 to 90%	–	10	ns
SSC_{15}(1)	TK low or high time	$V_{TK}>V_{IH}$ or $V_{TK}<V_{IL}$	3.6 + (3 x t_{CPMCK})	–	ns

Note 1: Timings SSC_{14} and SSC_{15} also apply when RK is used instead of TK (SSC_TCMR.CKS = RK).

© 2017 Microchip Technology Inc.

40.12.4 ISI Timings

Timings are given assuming a capacitance load on clock and data signals as defined in Table 40-43.

Table 40-43: Capacitance Load

Supply	Corner
	Max
3.3V	30 pF
1.8V	20 pF

Figure 40-21: ISI Timing Diagram

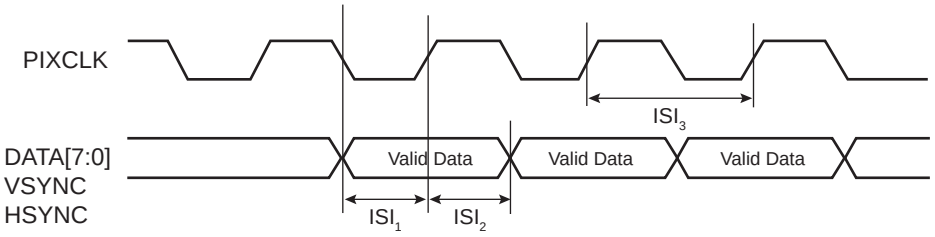


Table 40-44: ISI Timings with Peripheral Supply 3.3V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	4.5	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	1.3	–	ns
ISI_3	PIXCLK frequency	–	66	MHz

Table 40-45: ISI Timings with Peripheral Supply 1.8V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	6.4	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	2.0	–	ns
ISI_3	PIXCLK frequency	–	55	MHz

40.12.5 MCI Timings

Capacitance loads on data and clock are given in Table 40-46.

Figure 40-22: MCI Timings

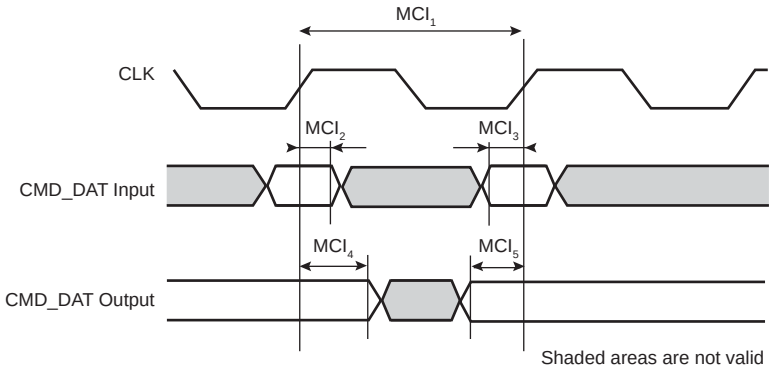


Table 40-46: MCI Timings

Symbol	Parameter	C_{LOAD}	Min	Max	Unit
MCI_1	CLK frequency at Data transfer Mode	25 pF	–	25	MHz
		100 pF	–	20	
		250 pF	–	20	
	CLK frequency at Identification Mode	–	–	400	kHz
	CLK Low time	100 pF	10	–	ns
	CLK High time	100 pF	10	–	ns
	CLK Rise time	100 pF	–	10	ns
	CLK Fall time	100 pF	–	10	ns
	CLK Low time	250 pF	50	–	ns
	CLK High time	250 pF	50	–	ns
	CLK Rise time	250 pF	–	50	ns
	CLK Fall time	250 pF	–	50	ns
MCI_2	Input hold time	–	3	–	ns
MCI_3	Input setup time	–	3	–	ns
MCI_4	Output change after CLK rising	–	5	–	ns
MCI_5	Output valid before CLK rising	–	5	–	ns

© 2017 Microchip Technology Inc.

40.12.6 UDP Timings

Figure 40-23: USB Data Signal Rise and Fall Times

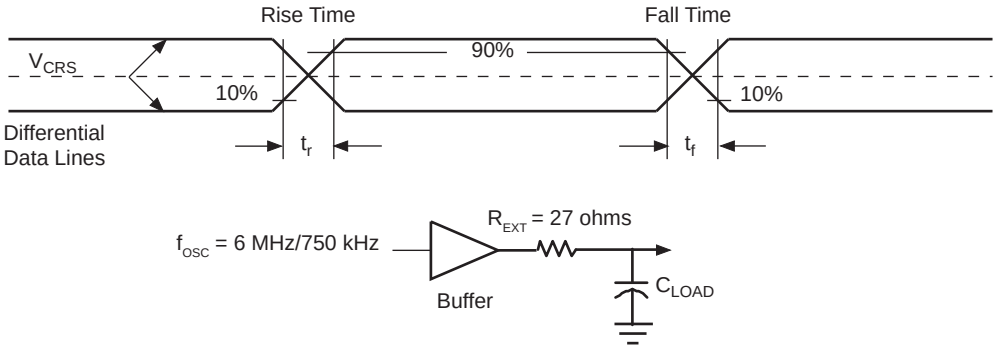


Table 40-47: In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_r	Transition Rise Time	C_{LOAD} = 50 pF	4	–	20	ns
t_f	Transition Fall Time	C_{LOAD} = 50 pF	4	–	20	ns
t_{rfm}	Rise/Fall time Matching		90	–	111.11	%

41. Mechanical Characteristics

41.1 217-ball LFBGA Package Drawing

Figure 41-1: 217-ball LFBGA Package Drawing

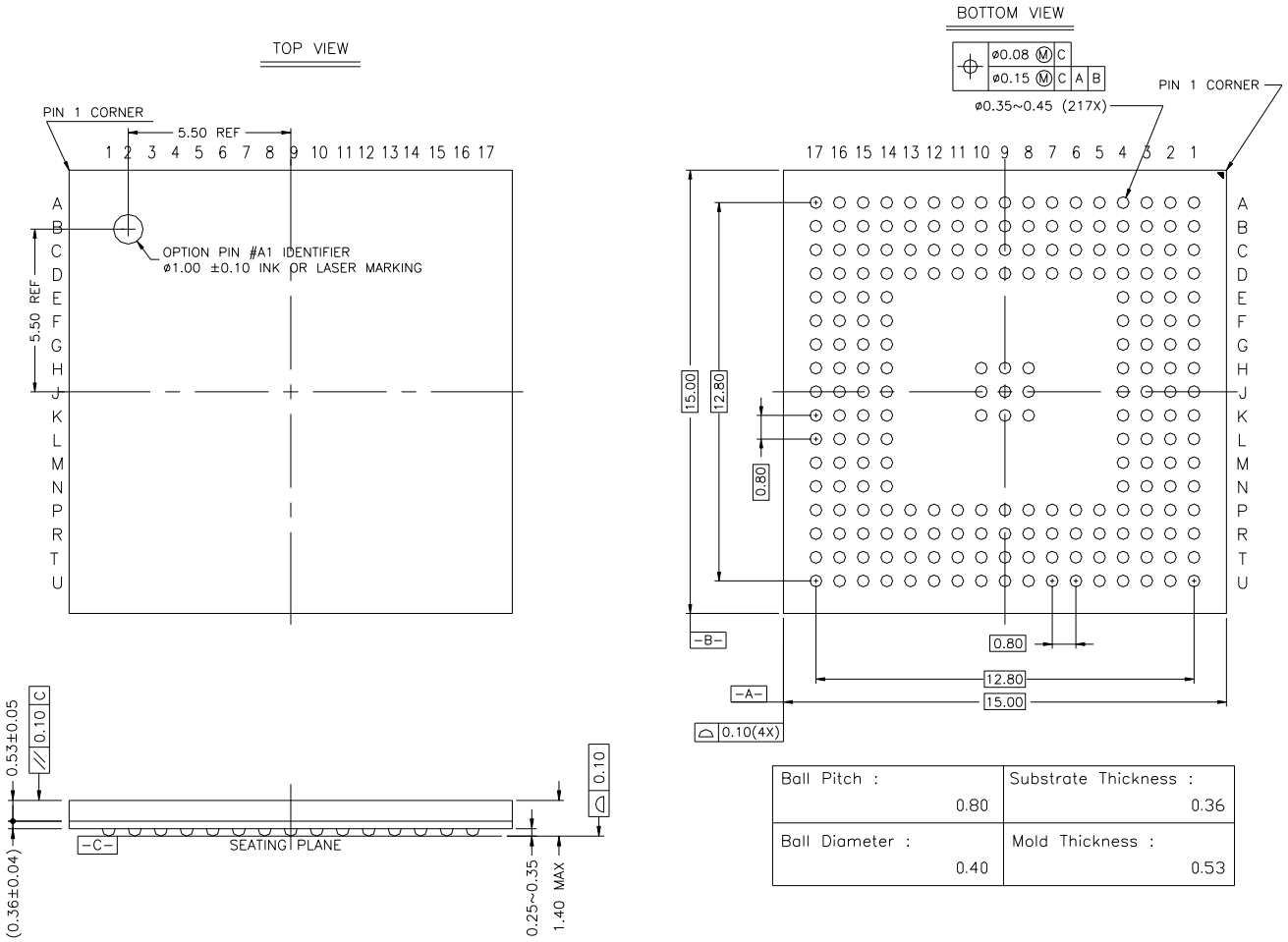


Table 41-1: 217-ball LFBGA Soldering Information

Ball Land	0.43 mm +/- 0.05
Soldering Mask Opening	0.30 mm +/- 0.05

Table 41-2: Device and 217-ball LFBGA Package Maximum Weight

450	mg

Table 41-3: 217-ball LFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 41-4: 217-ball LFBGA Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

© 2017 Microchip Technology Inc.

41.1.1 Soldering Profile – 217-ball LFBGA

Table 41-5 gives the recommended soldering profile from J-STD-20.

Table 41-5: Soldering Profile – 217-ball LFBGA

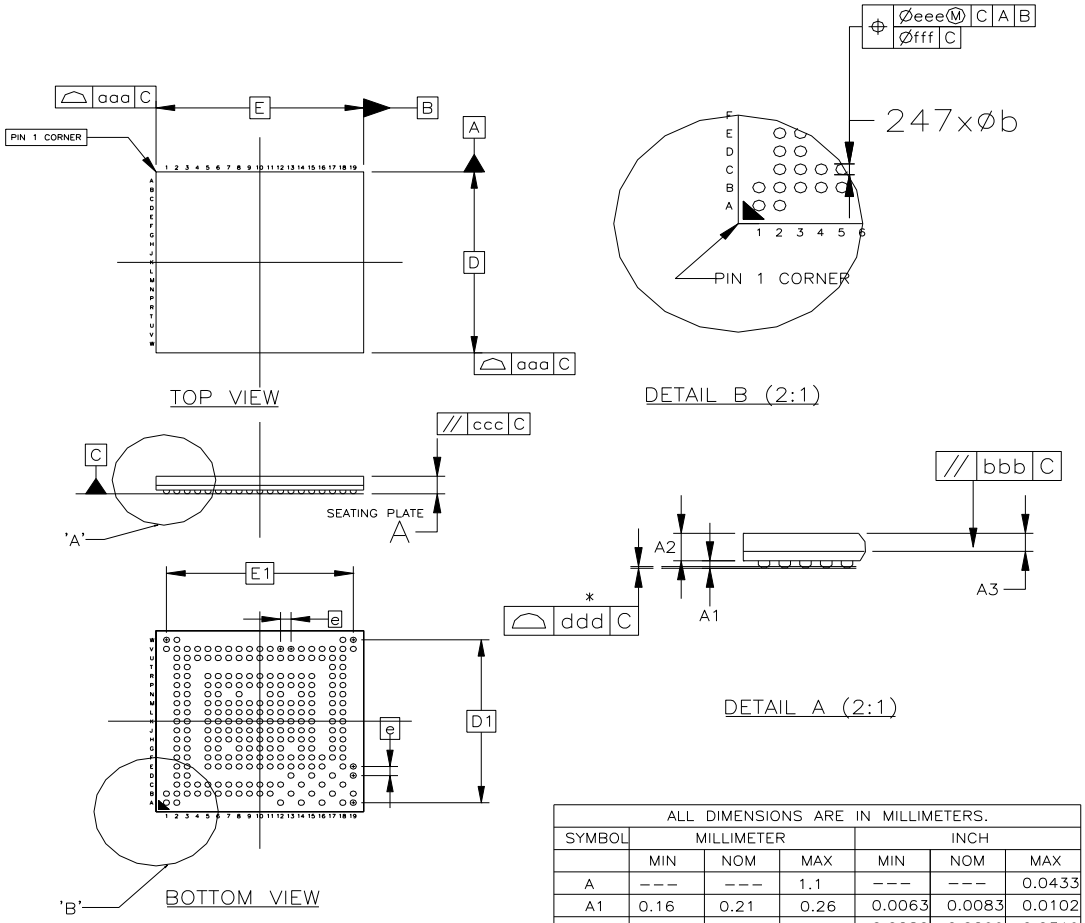
Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260 +0°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

41.2 247-ball TFBGA Package Drawing

Figure 41-2: 247-ball TFBGA Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.1	---	---	0.0433
A1	0.16	0.21	0.26	0.0063	0.0083	0.0102
A2	0.71	0.76	0.81	0.0280	0.0299	0.0319
A3	0.50 BASIC			0.0197BASIC		
D	9.90	10.00	10.10	0.3898	0.3937	0.3976
D1	9 BASIC			0.3543BASIC		
E	9.90	10.00	10.10	0.3898	0.3937	0.3976
E1	9 BASIC			0.3543BASIC		
e	0.5 BASIC			0.0197 BASIC		
b	0.25	0.3	0.35	0.0098	0.0118	0.0138
aaa	0.15			0.0059		
bbb	0.20			0.0078		
ccc	0.20			0.0079		
ddd	0.08			0.0031		
eee	0.15			0.0059		
fff	0.05			0.0020		

Table 41-6: 247-ball TFBGA Ball Information

Ball Pitch	0.50 mm ± 0.05
Ball Diameter	0.30 mm ± 0.05

Table 41-7: 247-ball TFBGA Soldering Information

Ball Land	0.35 mm ± 0.05
Solder Mask Opening	0.27 mm ± 0.05

© 2017 Microchip Technology Inc.

Table 41-8: Device and 247-ball TFBGA Package Maximum Weight

177	mg

Table 41-9: 247-ball TFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 41-10: 247-ball TFBGA Package Reference

JEDEC Drawing Reference	none
JESD97 Classification	e1

41.2.1 Soldering Profile – 247-ball TFBGA

Table 41-5 gives the recommended soldering profile from J-STD-20.

Table 41-11: Soldering Profile – 247-ball TFBGA

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 +0°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

42. Marking

All devices are marked with the company logo and the ordering code.

Additional marking has the following format:

where

- “YY”: Manufactory year
- “WW”: Manufactory week
- “C”: Assembly country code (optional)
- “V”: Revision
- “XXXXXXX”: Lot number

© 2017 Microchip Technology Inc.

43. Ordering Information

Table 43-1: SAM9G20 Ordering Information

MRL A Ordering Code	MRL B Ordering Code	Package	Carrier Type	Operating Temperature Range
AT91SAM9G20-CU	AT91SAM9G20B-CU	LFBGA217	Tray	Industrial -40°C to 85°C
–	AT91SAM9G20B-CFU	TFBGA247	Tray	Industrial -40°C to 85°C

44. Errata

Errata is gathered in two sections:

Section 44.1 “SAM9G20 Errata - Revision A Parts”

Section 44.2 “SAM9G20 Errata - Revision B Parts”

44.1 SAM9G20 Errata - Revision A Parts

44.1.1 Analog-to-digital Converter (ADC)

44.1.1.1 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); to start an analog-to-digital conversion, in order put ADC into sleep mode at the end of this conversion.

44.1.2 Boot ROM

44.1.2.1 Boot ROM: ROM Code, Internal RC Usage with SAM-BA Boot through USB

Booting from the internal RC oscillator (OSCSEL = 0) prevents using SAM-BA Boot through the USB device interface.

Problem Fix/Workaround

Use SAM-BA Boot through DBGU (or AT91SAM-ICE JTAG-ICE interface with SAM-BA GUI) if the internal RC oscillator must be used.

Boot from the 32 kHz external oscillator (OSCEL = 1) to use SAM-BA Boot through the USB.

44.1.2.2 Boot ROM: MCCK Remains Active

The ROM boot sequence is as follows:

- DataFlash on SPI0 NPCS0
- DataFlash on SPI0 NPCS1
- NAND FLash
- SDCard
- TWI
- SAM-BA Boot

MCCK is not disabled after SDCard boot program. This does not affect the TWI boot program but must be considered by any applications using SAM-BA boot.

Problem Fix/Workaround

None.

44.1.3 Error Correction Code Controller (ECC)

44.1.3.1 ECC: 1-bit ECC per 512 Words

1-bit ECC per 512 words is not functional.

Problem Fix/Workaround

Perform the ECC computation by software.

44.1.3.2 ECC: Incomplete parity status when error in ECC parity

When a single correctable error is detected in ECC value, the error is located in ECC Parity register's field which contains a 1 in the 24 least significant bits except when the error is located in the 12th or the 24th bit. In this case these bits are always stuck at 0.

A Single correctable error is detected but it is impossible to correct it.

Problem Fix/Workaround

None.

© 2017 Microchip Technology Inc.

44.1.3.3 ECC: Unsupported hardware ECC on 16-bit NAND Flash

Hardware ECC on 16-bit NAND Flash is not supported.

Problem Fix/Workaround

Perform the ECC by software.

44.1.4 MCI

44.1.4.1 MCI: Busy Signal of R1b responses is not taken in accoun

The busy status of the card during the response (R1b) is ignored for the commands CMD7, CMD28, CMD29, CMD38, CMD42, CMD56. Additionally, for commands CMD42 and CMD56 a conflict can occur on data line0 if the MCI sends data to the card while the card is still busy.The behavior is correct for CMD12 command (STOP_TRANSFER).

Problem Fix/Workaround

None.

44.1.4.2 MCI: SDIO Interrupt does not work with slots other than A

If there is 1-bit data bus width on slots other than slot A, the SDIO interrupt can not be captured. The sample is made on the wrong data line.

Problem Fix/Workaround

None

44.1.4.3 MCI: Data Timeout Error Flag

As the data Timeout error flag checking the Naac timing cannot rise, the MCI can be stalled waiting indefinitely the Data start bit.

Problem Fix/Workaround

A STOP command must be sent with a software timeout.

44.1.4.4 MCI: Data Write Operation and number of bytes

The Data Write operation with a number of bytes less than 12 is impossible.

Problem Fix/Workaround

The PDC counters must always be equal to 12 bytes for data transfers lower than 12 bytes. The BLKLEN or BCNT field are used to specify the real count number.

44.1.4.5 MCI: Flag Reset is not correct in half duplex mode

In half duplex mode, the reset of the flags ENDRX, RXBUFF, ENDTX and TXBUFE can be incorrect. These flags are reset correctly after a PDC channel enable.

Problem Fix/Workaround

Enable the interrupts related to ENDRX, ENDTX, RXBUFF and TXBUFE only after enabling the PDC channel by writing PDC_TXTEN or PDC_RXTEN.

44.1.5 Reset Controller (RSTC)

44.1.5.1 RSTC: Reset During SDRAM Accesses

When a User Reset occurs during SDRAM read access, the SDRAM clock is turned off while data are ready to be read on the data bus. The SDRAM maintains the data until the clock restarts.

If the user Reset is programmed to assert a general reset, the data maintained by the SDRAM leads to a data bus conflict and adversely affects the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

Problem Fix/Workaround

1. Avoid User Reset to generate a system reset.
2. Trap the User Reset with an interrupt.

In the interrupt routine, Power Down SDRAM properly and perform Peripheral and Processor Reset with software in assembler.

Example with libV3.

- The main code:

```
//user reset interrupt setting
    // Configure AIC controller to handle System peripheral interrupts
    AT91F_AIC_ConfigureIt (
        AT91C_BASE_AIC,                       // AIC base address
        AT91C_ID_SYS,                        // System peripheral ID
        AT91C_AIC_PRIOR_HIGHEST,                 // Max priority
        AT91C_AIC_SRCTYPE_INT_EDGE_TRIGGERED, // Level sensitive
        sysc_handler );

    // Enable SYSC interrupt in AIC
    AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);

*AT91C_RSTC_RMR = (0xA5<<24) | (0x4<<8) | AT91C_RSTC_URSTIEN;
```

- The C SYS handler:

```
extern void soft_user_reset(void);

void sysc_handler(void){
//check if interrupt comes from RSTC
  if( (*AT91C_RSTC_RSR & AT91C_RSTC_URSTS ) == AT91C_RSTC_URSTS){
    soft_user_reset();
    //never reached
    while(1);

  }
}
```

- Assembly code is mandatory for the following sequence as Arm instructions need to be pipelined.
  The assembler routine:

```
AREA       TEST, CODE
    INCLUDE AT91SAM9xxx.inc
    EXPORTsoft_user_reset

soft_user_reset
;disable IRQs
    MRS r0, CPSR
    ORR r0, r0, #0x80
    MSR CPSR_c, r0

;change refresh rate to block all data accesses
    LDR r0, =AT91C_SDRAMC_TR
    LDR r1, =1
    STR r1, [r0]

;prepare power down command
    LDR r0, =AT91C_SDRAMC_LPR
    LDR r1, =2

;prepare proc_reset and periph_reset
    LDR r2, =AT91C_RSTC_RCR
    LDR r3, =0xA5000005

;perform power down command
    STR r1, [r0]
;perform proc_reset and periph_reset (in the ARM pipeline)
    STR r3, [r2]

    END
```


© 2017 Microchip Technology Inc.

44.1.6 Serial Peripheral Interface (SPI)

44.1.6.1 SPI: Bad Serial Clock Generation on second chip_select when SCBR = 1, CPOL = 1 and NCPHA = 0

If the SPI is used in the following configuration:

- Master mode
- CPOL = 1 and NCPHA = 0
- multiple chip selects 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 not equal to 1
- transmit with the slowest chip select and then with the fastest one

then an additional pulse will be generated on output PSCK 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 CPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

44.1.6.2 SPI: Baudrate set to 1

When Baudrate is set to 1 (i.e., when serial clock frequency equals the system clock frequency), and when the fields BITS (number of bits to be transmitted) equals an ODD value (in this case 9,11,13 or 15), an additional pulse is generated on output SPCK. No error occurs if BITS field equals 8,10,12,14 or 16 and Baudrate = 1.

Problem Fix/Workaround

None.

44.1.7 Serial Synchronous Controller (SSC)

44.1.7.1 SSC: Unexpected RK clock cycle when RK outputs a clock during data transfer

When the SSC receiver is used in the following configuration:

- the internal clock divider is used (CKS = 0 and DIV different from 0),
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

then, at the end of the data, the RK pin is set in high impedance which may be interpreted as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

44.1.7.2 SSC: Incorrect first RK clock cycle when RK outputs a clock during data transfer

When the SSC receiver is used in the following configuration:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

then the first clock cycle time generated by the RK pin is equal to MCK/(2 x (DIV +1)) instead of MCK/(2 x DIV).

Problem Fix/Workaround

None.

44.1.7.3 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

Problem Fix/Workaround

None.

44.1.7.4 SSC: Periodic Transmission Limitations in Master Mode

If Last Significant Bit is sent first (MSBF = 0) the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

44.1.8 Shutdown Controller (SHDWC)

44.1.8.1 SHDWC: SHDN Signal may be Driven to Low Level Voltage During Device Power-on

If only VDDBU is powered during boot sequence (No VDDCORE), the SHDN signal may be driven to low level voltage after a delay.This delay is linked to the startup time of the slow clock selected by OSCSEL signal.

If SHDN pin is connected to the Enable pin (EN) of the VDDCORE regulator, VDDCORE establishment does not occur and the system does not start.

Problem Fix/Workaround

1. VDDCORE must be established within the delay corresponding to the startup time of the slow clock selected by OSCSEL.
2. Add a glue logic to latch the rising edge of the SHDN signal. The reset of the latch output (EN_REG) can be connected to a PIO and used to enter the shutdown mode.

44.1.9 Static Memory Controller (SMC)

44.1.9.1 SMC: Chip Select Parameters Modification

The user must not change the configuration parameters of an SMC Chip Select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification.

For example, the modification of the Chip Select 0 (CS0) parameters, while fetching the code from a memory connected on this CS0, may lead to unpredictable behavior.

Problem Fix/Workaround

The code used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another Chip Select

44.1.10 System Controller (SYSC)

44.1.10.1 SYSC: Possible Event Loss when reading RTT_SR

If an event (RTTINC or ALMS) occurs within the same slow clock cycle as when the RTT_SR is read, the corresponding bit might be cleared. This can lead to the loss of this event.

Problem Fix/Workaround

The software must handle an RTT event as an interrupt and should not poll RTT_SR.

44.1.11 Two-wire Interface (TWI)

44.1.11.1 TWI: RXRDY Flag is not reset by a SOFTWARE Reset

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.

44.1.12 UHP

44.1.12.1 UHP: Non-ISO IN Transfers

Conditions:

Consider the following sequence:

1. The Host controller issues an IN token.
2. The Device provides the IN data in a short packet.
3. The Host controller writes the received data to the system memory.
4. The Host controller is now supposed to carry out two Write transactions (TD status write and TD retirement write) to the system memory in order to complete the status update.

© 2017 Microchip Technology Inc.

5. The Host controller raises the request for the first write transaction. By the time the transaction is completed, a frame boundary is crossed.
6. After completing the first write transaction, the Host controller skips the second write transaction.

Consequence: When this error occurs, the Host controller tries the same IN token again.

Problem Fix/Workaround

This problem can be avoided if the system guarantees that the status update can be completed within the same frame.

44.1.12.2 UHP: ISO OUT transfers

Conditions:

Consider the following sequence:

1. The Host controller sends an ISO OUT token after fetching 16 bytes of data from the system memory.
2. When the Host controller is sending the ISO OUT data, because of system latencies, remaining bytes of the packet are not available. This results in a buffer underrun condition.
3. While there is an underrun condition, if the Host controller is in the process of bit-stuffing, it causes the Host controller to hang.

Consequence: After the failure condition, the Host controller stops sending the SOF. This causes the connected device to go into suspend state.

Problem Fix/Workaround

This problem can be avoided if the system can guarantee that no buffer underrun occurs during the transfer.

44.1.12.3 UHP: Remote Wakeup Event

Conditions:

When a Remote Wakeup event occurs on a downstream port, the OHCI Host controller begins sending resume signaling to the device. The Host controller is supposed to send this resume signaling for 20 ms. However, if the driver sets the HcControl.HCFS into USBOPERATIONAL state during the resume event, then the Host controller terminates sending the resume signal with an EOP to the device.

Consequence: If the Device does not recognize the resume (< 20 ms) event then the Device remains in suspend state.

Problem Fix/Workaround

Host stack can do a port resume after it sets the HcControl.HCFS to USBOPERATIONAL.

44.1.13 USART

44.1.13.1 USART: TXD Signal is floating in Modem and Hardware Handshaking mode.

TXD signal should be pulled up in Modem and Hardware Handshaking mode.

Problem Fix/Workaround

TXD is multiplexed with PIO which integrates a pull up resistor. This internal pull-up must be enabled.

44.1.13.2 USART: DCD is Active High instead of Low

The DCD signal is active at High level in the USART Modem Mode .

DCD should be active at Low level.

Problem Fix/Workaround

Add an inverter.

44.1.14 Power Management Controller (PMC)

44.1.14.1 PMC: PMC bad frequency after MDIV switching

If MDIV and another field (CSS or PRES) are changed at the same, the clock frequency may not be correct.

Problem Fix/Workaround

For each clock switching, the user must take care to:

- change fields CSS, MDIV, PRES one by one
- wait MCKRDY bit setting in PMC_SR before changing PMC_MCKR
- ensure each transitory frequency value is in operational range for PCK and MCK

44.1.15 External Bus Interface (EBI)

44.1.15.1 EBI: After reset A25 is not functionnal

After reset of the SAM9G20 device the PC10/A25 line is set into general purpose (GPIO) mode and not in peripheral mode.

This result to prevent any access to upper memory address range (0xD0000000) as the A25 EBI line is not functionnal.

Problem Fix/Workaround

According to your system memory mapping, set into your low level initialization the PC10 line in peripheral A mode.

44.2 SAM9G20 Errata - Revision B Parts

44.2.1 Analog-to-digital Converter (ADC)

44.2.1.1 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); to start an analog-to-digital conversion, in order put ADC into sleep mode at the end of this conversion.

44.2.2 Boot ROM

44.2.2.1 Boot ROM: ROM Code, Internal RC Usage with SAM-BA Boot through USB

Booting from the internal RC oscillator (OSCSEL = 0) prevents using SAM-BA Boot through the USB device interface.

Problem Fix/Workaround

Use SAM-BA Boot through DBGU (or AT91SAM-ICE JTAG-ICE interface with SAM-BA GUI) if the internal RC oscillator must be used.

Boot from the 32 kHz external oscillator (OSCEL = 1) to use SAM-BA Boot through the USB.

44.2.2.2 Boot ROM: MCCK Remains Active

The ROM boot sequence is as follows:

- DataFlash on SPI0 NPCS0
- DataFlash on SPI0 NPCS1
- NAND FLash
- SDCard
- TWI
- SAM-BA Boot

MCCK is not disabled after SDCard boot program. This does not affect the TWI boot program but must be considered by any applications using SAM-BA boot.

Problem Fix/Workaround

None.

44.2.3 Error Correction Code Controller (ECC)

44.2.3.1 ECC: 1-bit ECC per 512 Words

1-bit ECC per 512 words is not functional.

Problem Fix/Workaround

Perform the ECC computation by software.

44.2.3.2 ECC: Incomplete parity status when error in ECC parity

When a single correctable error is detected in ECC value, the error is located in ECC Parity register's field which contains a 1 in the 24 least significant bits except when the error is located in the 12th or the 24th bit. In this case these bits are always stuck at 0.

A Single correctable error is detected but it is impossible to correct it.

Problem Fix/Workaround

None.

© 2017 Microchip Technology Inc.

44.2.3.3 ECC: Unsupported hardware ECC on 16-bit NAND Flash

Hardware ECC on 16-bit NAND Flash is not supported.

Problem Fix/Workaround

Perform the ECC by software.

44.2.4 MCI

44.2.4.1 MCI: Busy Signal of R1b responses is not taken in account

The busy status of the card during the response (R1b) is ignored for the commands CMD7, CMD28, CMD29, CMD38, CMD42, CMD56. Additionally, for commands CMD42 and CMD56 a conflict can occur on data line0 if the MCI sends data to the card while the card is still busy.The behavior is correct for CMD12 command (STOP_TRANSFER).

Problem Fix/Workaround

None.

44.2.4.2 MCI: SDIO Interrupt does not work with slots other than A

If there is 1-bit data bus width on slots other than slot A, the SDIO interrupt can not be captured. Th sample is made on the wrong data line.

Problem Fix/Workaround

None

44.2.4.3 MCI: Data Timeout Error Flag

As the data Timeout error flag checking the Naac timing cannot rise, the MCI can be stalled waiting indefinitely the Data start bit.

Problem Fix/Workaround

A STOP command must be sent with a software timeout.

44.2.4.4 MCI: Data Write Operation and number of bytes

The Data Write operation with a number of bytes less than 12 is impossible.

Problem Fix/Workaround

The PDC counters must always be equal to 12 bytes for data transfers lower than 12 bytes. The BLKLEN or BCNT field are used to specify the real count number.

44.2.4.5 MCI: Flag Reset is not correct in half duplex mode

In half duplex mode, the reset of the flags ENDRX, RXBUFF, ENDTX and TXBUFE can be incorrect. These flags are reset correctly after a PDC channel enable.

Problem Fix/Workaround

Enable the interrupts related to ENDRX, ENDTX, RXBUFF and TXBUFE only after enabling the PDC channel by writing PDC_TXTEN or PDC_RXTEN.

44.2.5 Reset Controller (RSTC)

44.2.5.1 RSTC: Reset During SDRAM Accesses

When a User Reset occurs during SDRAM read access, the SDRAM clock is turned off while data are ready to be read on the data bus. The SDRAM maintains the data until the clock restarts.

If the user Reset is programmed to assert a general reset, the data maintained by the SDRAM leads to a data bus conflict and adversely affects the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

Problem Fix/Workaround

1. Avoid User Reset to generate a system reset.
2. Trap the User Reset with an interrupt.

In the interrupt routine, Power Down SDRAM properly and perform Peripheral and Processor Reset with software in assembler.

Example with libV3.

- The main code:

```
//user reset interrupt setting
    // Configure AIC controller to handle SSC interrupts
    AT91F_AIC_ConfigureIt (
        AT91C_BASE_AIC,                       // AIC base address
        AT91C_ID_SYS,                        // System peripheral ID
        AT91C_AIC_PRIOR_HIGHEST,                 // Max priority
        AT91C_AIC_SRCTYPE_INT_EDGE_TRIGGERED, // Level sensitive
        sysc_handler );

    // Enable SYSC interrupt in AIC
    AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);

*AT91C_RSTC_RMR = (0xA5<<24) | (0x4<<8) | AT91C_RSTC_URSTIEN;
```

- The C SYS handler:

```
extern void soft_user_reset(void);

void sysc_handler(void){
//check if interrupt comes from RSTC
  if( (*AT91C_RSTC_RSR & AT91C_RSTC_URSTS ) == AT91C_RSTC_URSTS){
    soft_user_reset();
    //never reached
    while(1);

  }
}
```

- Assembly code is mandatory for the following sequence as Arm instructions need to be pipelined.
  The assembler routine:

```
AREA       TEST, CODE
    INCLUDE AT91SAM9xxx.inc
    EXPORTsoft_user_reset

soft_user_reset
;disable IRQs
    MRS r0, CPSR
    ORR r0, r0, #0x80
    MSR CPSR_c, r0

;change refresh rate to block all data accesses
    LDR r0, =AT91C_SDRAMC_TR
    LDR r1, =1
    STR r1, [r0]

;prepare power down command
    LDR r0, =AT91C_SDRAMC_LPR
    LDR r1, =2

;prepare proc_reset and periph_reset
    LDR r2, =AT91C_RSTC_RCR
    LDR r3, =0xA5000005
```


© 2017 Microchip Technology Inc.

```
;perform power down command
    STR r1, [r0]
;perform proc_reset and periph_reset (in the ARM pipeline)
    STR r3, [r2]

    END
```

44.2.6 Serial Peripheral Interface (SPI)

44.2.6.1 SPI: Bad Serial Clock Generation on second chip_select when SCBR = 1, CPOL = 1 and NCPHA = 0

If the SPI is used in the following configuration:

- master mode
- CPOL = 1 and NCPHA = 0
- multiple chip selects 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 not equal to 1
- transmit with the slowest chip select and then with the fastest one

then an additional pulse will be generated on output PSCK 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 CPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

44.2.6.2 SPI: Baudrate set to 1

When Baudrate is set to 1 (i.e., when serial clock frequency equals the system clock frequency), and when the fields BITS (number of bits to be transmitted) equals an ODD value (in this case 9,11,13 or 15), an additional pulse is generated on output SPCK. No error occurs if BITS field equals 8,10,12,14 or 16 and Baudrate = 1.

Problem Fix/Workaround

None.

44.2.7 Serial Synchronous Controller (SSC)

44.2.7.1 SSC: Unexpected RK clock cycle when RK outputs a clock during data transfer

When the SSC receiver is used in the following configuration:

- the internal clock divider is used (CKS = 0 and DIV different from 0),
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

then, at the end of the data, the RK pin is set in high impedance which may be interpreted as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

44.2.7.2 SSC: Incorrect first RK clock cycle when RK outputs a clock during data transfer

When the SSC receiver is used in the following configuration:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0)

then the first clock cycle time generated by the RK pin is equal to MCK/(2 x (DIV +1)) instead of MCK/(2 x DIV).

Problem Fix/Workaround

None.

44.2.7.3 SSC: Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

Problem Fix/Workaround

None.

44.2.7.4 SSC: Periodic Transmission Limitations in Master Mode

If Last Significant Bit is sent first (MSBF = 0) the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

44.2.8 Shutdown Controller (SHDWC)

44.2.8.1 SHDWC: SHDN Signal may be Driven to Low Level Voltage During Device Power-on

If only VDDBU is powered during boot sequence (No VDDCORE), the SHDN signal may be driven to low level voltage after a delay.This delay is linked to the startup time of the slow clock selected by OSCSEL signal.

If SHDN pin is connected to the Enable pin (EN) of the VDDCORE regulator, VDDCORE establishment does not occur and the system does not start.

Problem Fix/Workaround

1. VDDCORE must be established within the delay corresponding to the startup time of the slow clock selected by OSCSEL.
2. Add a glue logic to latch the rising edge of the SHDN signal. The reset of the latch output (EN_REG) can be connected to a PIO and used to enter the shutdown mode.

44.2.9 Static Memory Controller (SMC)

44.2.9.1 SMC: Chip Select Parameters Modification

The user must not change the configuration parameters of an SMC Chip Select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification.

For example, the modification of the Chip Select 0 (CS0) parameters, while fetching the code from a memory connected on this CS0, may lead to unpredictable behavior.

Problem Fix/Workaround

The code used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another Chip Select

44.2.10 System Controller (SYSC)

44.2.10.1 SYSC: Possible Event Loss when reading RTT_SR

If an event (RTTINC or ALMS) occurs within the same slow clock cycle as when the RTT_SR is read, the corresponding bit might be cleared. This can lead to the loss of this event.

Problem Fix/Workaround

The software must handle an RTT event as an interrupt and should not poll RTT_SR.

44.2.11 Two-wire Interface (TWI)

44.2.11.1 TWI: RXRDY Flag is not reset by a SOFTWARE Reset

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.

© 2017 Microchip Technology Inc.

44.2.12 UHP

44.2.12.1 UHP: Non-ISO IN Transfers

Conditions:

Consider the following sequence:

1. The Host controller issues an IN token.
2. The Device provides the IN data in a short packet.
3. The Host controller writes the received data to the system memory.
4. The Host controller is now supposed to carry out two Write transactions (TD status write and TD retirement write) to the system memory in order to complete the status update.
5. The Host controller raises the request for the first write transaction. By the time the transaction is completed, a frame boundary is crossed.
6. After completing the first write transaction, the Host controller skips the second write transaction.

Consequence: When this error occurs, the Host controller tries the same IN token again.

Problem Fix/Workaround

This problem can be avoided if the system guarantees that the status update can be completed within the same frame.

44.2.12.2 UHP: ISO OUT transfers

Conditions:

Consider the following sequence:

1. The Host controller sends an ISO OUT token after fetching 16 bytes of data from the system memory.
2. When the Host controller is sending the ISO OUT data, because of system latencies, remaining bytes of the packet are not available. This results in a buffer underrun condition.
3. While there is an underrun condition, if the Host controller is in the process of bit-stuffing, it causes the Host controller to hang.

Consequence: After the failure condition, the Host controller stops sending the SOF. This causes the connected device to go into suspend state.

Problem Fix/Workaround

This problem can be avoided if the system can guarantee that no buffer underrun occurs during the transfer.

44.2.12.3 UHP: Remote Wakeup Event

Conditions:

When a Remote Wakeup event occurs on a downstream port, the OHCI Host controller begins sending resume signaling to the device. The Host controller is supposed to send this resume signaling for 20 ms. However, if the driver sets the HcControl.HCFS into USBOPERATIONAL state during the resume event, then the Host controller terminates sending the resume signal with an EOP to the device.

Consequence: If the Device does not recognize the resume (< 20 ms) event then the Device remains in suspend state.

Problem Fix/Workaround

Host stack can do a port resume after it sets the HcControl.HCFS to USBOPERATIONAL.

44.2.13 USART

44.2.13.1 USART: TXD Signal is floating in Modem and Hardware Handshaking Mode.

TXD signal should be pulled up in Modem and Hardware Handshaking mode.

Problem Fix/Workaround

TXD is multiplexed with PIO which integrates a pull up resistor. This internal pull-up must be enabled.

44.2.13.2 USART: DCD is Active High instead of Low

The DCD signal is active at High level in the USART Modem Mode .

DCD should be active at Low level.

Problem Fix/Workaround

Add an inverter.

44.2.14 Power Management Controller (PMC)

44.2.14.1 PMC: PMC bad frequency after MDIV switching

If MDIV and another field (CSS or PRES) are changed at the same, the clock frequency may not be correct.

Problem Fix/Workaround

For each clock switching, the user must take care to:

- change fields CSS, MDIV, PRES one by one
- wait MCKRDY bit setting in PMC_SR before changing PMC_MCKR
- ensure each transitory frequency value is in operational range for PCK and MCK

44.2.15 External Bus Interface (EBI)

44.2.15.1 EBI: After reset A25 is not functionnal

After reset of the SAM9G20 device the PC10/A25 line is set into general purpose (GPIO) mode and not in peripheral mode.

This result to prevent any access to upper memory address range (0xD0000000) as the A25 EBI line is not functionnal.

Problem Fix/Workaround

According to your system memory mapping, set into your low level initialization the PC10 line in peripheral A mode.

© 2017 Microchip Technology Inc.

45. Revision History

In the tables that follow, the most recent version appears first.

Issue Date	Changes
Oct-17	General - Template update: Moved from Atmel to Microchip template. - The datasheet is assigned a new document number (DS60001516) and revision letter is reset to A. --- Document number DS60001516 revision A corresponds to what would have been 6384 revision G. - ISBN number assigned.
	Section 40. “Electrical Characteristics” Table 40-45 ”ISI Timings with Peripheral Supply 1.8V”: corrected ISI_1 and ISI_2 minimum values Table 40-41 ”SSC Timings 3.3V Domain” and Table 40-42 ”SSC Timings 1.8V Domain”: removed footnote “For output signals (TF, TD, RF), Min and Max access times are defined, etc.” Removed Figure 40-21: “Min and Max Access for SSC0” Updated the following with SSC_{14} and SSC_{15} content: Figure 40-16: “SSC Transmitter, TK and TF in Input” Table 40-41 ”SSC Timings 3.3V Domain” Table 40-42 ”SSC Timings 1.8V Domain”
	Section 43. “Ordering Information” Updated Table 43-1 ”SAM9G20 Ordering Information”

Doc. Rev. 6384F	Comments
09-Nov-15	In document title, “Thumb Microcontrollers” changed to “Embedded MPU” Product name AT91SAM9G20 updated to SAM9G20 General editorial and formatting changes throughout
	Section “Description” Updated text
	Section “Features” In package information, “RoHS-compliant” updated to “Green-compliant”
	Section 1. “Block Diagram” Figure 1-1: “SAM9G20 Block Diagram”: - minor update in System Controller and backup area - signal range “SCK0–SCK2” corrected to “SCK0–SCK3”
	Section 2. “Signal Description” At end of section, deleted note “No PLLRCA line present on the AT91SAM9G20.”
	Section 3. “Package and Pinout” Updated first sentence
	Section 4. “Power Considerations” Updated Section 4.1 “Power Supplies” Added Section 4.3 “Power Sequence Requirements” Updated Section 4.3.1 “Power-up Sequence”
	Section 7. “Memories” Figure 7-1: “SAM9G20 Memory Mapping”: two blocks “MATRIX” and “CCFG” replaced by single “MATRIX” block
	Section 8. “System Controller” Section 8.3 “Shutdown Controller”: “SHWDN pin” corrected to “SHDN pin”
	Section 11. “Debug and Test” Updated Figure 11-2: “Application Debug and Trace Environment Example”
	Section 12. “Boot Program” Section 12.5 “Serial Flash Boot”: in second paragraph, “SPCK at 8 MHz” corrected to “SPCK at 1 MHz” Section 12.6 “DataFlash Boot Sequence”: in second paragraph, “SPCK at 8 MHz” corrected to “SPCK at 1 MHz” Section 12.7 “NAND Flash Boot”: updated first paragraph to specify first block must be guaranteed Section 12.8 “SDCard Boot”: added note “The bootable SDCard slot is Slot A.” Section 12.10.3 “USB Device Port”: removed reference to “Windows XP”
	Section 18. “SAM9G20 Bus Matrix” Section 18.5.4 “Bus Matrix Master Remap Control Register”: deleted reset value line Section 18.6.1 “EBI Chip Select Assignment Register”: deleted reset value line; updated IOSR bit description
	Section 19. “SAM9G20 External Bus Interface” Section 19.6.6.2 “CFCE1 and CFCE2 Signals”: updated text

© 2017 Microchip Technology Inc.

Doc. Rev. 6384F	Comments (Continued)
09-Nov-15	Section 21. “SDRAM Controller (SDRAMC)” Table 21-8, “Register Mapping”: access “Read” corrected to “Read/Write” for SDRAMC_MDR Removed reset value from register description sections (reset values are provided in Table 21-8, “Register Mapping”)
	Section 22. “Error Correction Code Controller (ECC)” Table 22-1, “Register Mapping”: for ECC Control Register, removed reset value (register is write-only); ECC_CTRL corrected to ECC_CR
	Section 23. “Peripheral DMA Controller (PDC)” Table 23-1, “Register Mapping”: removed reset value from PERIPH_PTCR (register is write-only)
	Section 25. “Power Management Controller (PMC)” Updated Section 25.3 “Processor Clock Controller” PMC_PLLICPR access “Write-only” corrected to “Read/Write” in Table 25-3, “Register Mapping” and in Section 25.9.17 “PLL Charge Pump Current Register”
	Section 26. “Advanced Interrupt Controller (AIC)” Removed reset value from register description sections (reset values are provided in Table 26-2, “Register Mapping”)
	Section 30. “Two-wire Interface (TWI)” Table 30-4, “Register Mapping”: removed reset value from TWI_THR (register is write-only) Removed reset value from register description sections (reset values are provided in Table 30-4, “Register Mapping”)
	Section 31. “Universal Synchronous Asynchronous Receiver Transmitter (USART)” Table 31-5, “Possible Values for the Fi/Di Ratio”: in top row, replaced “774” with “744” Section 31.6.3.5 “Manchester Decoder”: “MANE flag in US_CSR” corrected to “MANERR flag in US_CSR” Figure 31-32: “T = 0 Protocol with Parity Error”: added diagram Table 31-10, “IrDA Baud Rate Error”: in header row, added “bit/s” to Baud Rate and added “µs” to Pulse Time Removed ‘MANE’ name from bit 24 in Section 31.7.3 “USART Interrupt Enable Register”, Section 31.7.4 “USART Interrupt Disable Register” and Section 31.7.5 “USART Interrupt Mask Register” Section 31.7.12 “USART FI DI RATIO Register”: removed reset value (reset value provided in Table 31-12, “Register Mapping”)
	Section 32. “Synchronous Serial Controller (SSC)” Section 32.6.1.1 “Clock Divider”: at end of section, deleted untitled Table 33-2
	Section 34. “MultiMedia Card Interface (MCI)” Figure 34-12: “Multiple Write Functional Flow Diagram”: adjusted frame dimensions to fully display previously hidden content at top and bottom of diagram
	Section 37. “USB Host Port (UHP)” Section 37.1 “Overview”: updated link to Hewlett-Packard website
	Section 38. “Image Sensor Interface (ISI)” Removed reset value from register description sections (reset values are provided in Table 38-9, “ISI Memory Mapping”)

Doc. Rev. 6384F	Comments (Continued)
09-Nov-15	Section 40. “Electrical Characteristics” Added Section 40.6 “I/O Characteristics” Section 40.9 “Core Power Supply POR Characteristics”: transferred three sections “Power Sequence Requirements”, “Power-up Sequence” and “Power-down Sequence” to Section 4. “Power Considerations” Added Section 40.12.2.1 “Maximum SPI Frequency” Table 40-38, “Capacitance Load for MISO, SPCK and MOSI”: supply “1.8V” corrected to “1.8V/3.3V” Table 40-39, “SPI Timings”: deleted instances of “VDDIOP in 1.8V range”; deleted footnote C_{LOAD} is 8 pF for MISO and 6 pF for SPCK and MOSI.” Section 40.12.3 “SSC Timings”: replaced single “SSC Timings” table with two updated tables (Table 40-41, “SSC Timings 3.3V Domain” and Table 40-42, “SSC Timings 1.8V Domain”) Table 40-44, “ISI Timings with Peripheral Supply 3.3V”: ISI_3 PIXCLK frequency max value “TBD” updated to “66 MHz” Table 40-45, “ISI Timings with Peripheral Supply 1.8V”: ISI_3 PIXCLK frequency max value “TBD” updated to “55 MHz”
	Section 41.1 “217-ball LFBGA Package Drawing” Table 41-5, “Soldering Profile – 217-ball LFBGA”: removed column “BGA247 green-compliant Package”
	Migrated previous section 44.1 “Marking” to Section 42. “Marking”
	Section 43. “Ordering Information” Table 43-1, “SAM9G20 Ordering Information”: replaced column “Package Type” with “Carrier Type”
	Section 44. “Errata” Section 44.1 “SAM9G20 Errata - Revision A Parts” - added Section 44.1.3.2 “ECC: Incomplete parity status when error in ECC parity” - added Section 44.1.3.3 “ECC: Unsupported hardware ECC on 16-bit NAND Flash” - added Section 44.1.14.1 “PMC: PMC bad frequency after MDIV switching” - added Section 44.1.15.1 “EBI: After reset A25 is not functionnal” Section 44.2 “SAM9G20 Errata - Revision B Parts” - added Section 44.2.3.2 “ECC: Incomplete parity status when error in ECC parity” - added Section 44.2.3.3 “ECC: Unsupported hardware ECC on 16-bit NAND Flash” - added Section 44.2.14.1 “PMC: PMC bad frequency after MDIV switching” - added Section 44.2.15.1 “EBI: After reset A25 is not functionnal”

© 2017 Microchip Technology Inc.

Doc. Rev 6384E	Comments	Change Request Ref.
	Product Overview:	
	Section 4. “Power Considerations”, Removed the section “Power Consumption”.	6945
	Section 5. “I/O Line Considerations”, Removed the section “Slow Clock Selection”.	
	Boot Progam:	
	Section 12.1 “Overview”, 2nd sentence of 6th paragraph updated...”formatted SDCard on SlotA”.	6243
	CKGR:	
	Section 24.5.1 “Divider and Phase Lock Loop Programming”, removed and replaced last line with reference to PLL Characteristics in Product Electrical Characteristics section.	6666/6870
	PMC:	
	Section 25.9.16 “PMC Interrupt Mask Register”, PCKRDYx bitfield description functional sense of values 0 and 1, reversed.	6311
	SHDWC:	
	Section 17.6.3 “Shutdown Status Register”, bitfield 16 contains RTTWK.	6583
	SMC:	
	Table 20-8, “Register Mapping”, SMC_CYCLE reset is 0x00030003.	6742
	Table 20.8.6, “Reset Values of Timing Parameters”, replace redundant Table 21-5 with ref. to Table 21-8.	
	Electrical Characteristics:	
	“SPI”, Figure 40-8, Figure 40-9, Figure 40-10, Figure 40-11, titles to SPI timing diagrams simplified.	6872
	Table 40-2, “DC Characteristics”, I_{SC} values updated.	6905
	Table 40-2, “DC Characteristics”, I_{IN} 3.3V I/O Input leakage current row added.	6605
	Table 40.10.1, “Power-up Sequence”, updated the sentence; ”VDDIO must be established first (>0.7V)...”	6443
	Table 40-7, “XIN Clock Electrical Characteristics”, V_{IN} row, Min Max cells updated.	6595
	Table 40-11, “32 kHz Crystal Characteristics”, motional capacitance units are fF not pF.	6708
	Table 40-16, “PLLA Characteristics”, added; T, Startup Time row.	6673
	Table 40-18, “PLLB Characteristics”, added; T, Startup Time row.	
	ERRATA:	
	Section 44.1 “SAM9G20 Errata - Revision A Parts”,	
	Section 44.1.5.1 “RSTC: Reset During SDRAM Accesses”, updated.	
	Section 44.2 “SAM9G20 Errata - Revision B Parts”,	6728
	Section 44.2.5.1 “RSTC: Reset During SDRAM Accesses”, updated.	

Doc. Rev 6384D	Comments	Change Request Ref.
	Section 41.6 “ADC”, former Channel Conversion TIme and ADC Clock table and former Transfer Characteritics table replaced by Table 41-18, “ADC Characteristics 9-bit Mode” and Table 41-19, “ADC Characteristics 10-bit Mode”.	6262
	Section 44.2 “AT91SAM9G20 Errata - Revision “A” Parts”, added Section 44.2.2.1 “Boot ROM: ROM Code, Internal RC Usage with SAM-BA Boot through USB”.	6238

Doc. Rev 6384B	Comments	Change Request Ref.
	Overview “Features” on page 1, Debug Unit (DBGU) updated. Section 10.4.3 “USART”, “Optional Manchester Encoding” added to list of USART features. Section 8.1.1.1 “BMS = 1, Boot on Embedded ROM”, – SDCard, (boot ROM does not support high capacity SDCards) clarification added. Signal Description, Table 3-1, added GNDPLL to table.	 5846 5931 5935 6022
	Section 6.6 “Shutdown Logic Pins”, updated with external pull-up requirement.	rfo(1)
	AT91SAM9G20 Boot Program Section 13.3 “Device Initialization”, updated “Enable the user reset” removed from initialization steps. Section 13.8 “SDCard Boot” added (high capactiy SDCards not support by boot ROM) Section 13.11 “Hardware and Software Constraints” added footnote (1) “Boot ROM does not support high capacity SDCards.”	 5590 5934 5935
	DBGU Section 28.1 “Overview”, added text: “The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication.”	 5846/ 5847
	PMC Section 26.9.11 “PMC Master Clock Register”: bitfiedld description for “PDIV: Processor Clock Division” , Processor Clock replaced Master Clock in description.	 6384
	Electrical Characteristics: Section 41.8 “Core Power Supply POR Characteristics”, updated with caution on power-up, Figure 41-2. “VDDCORE and VDDIO Constraints at Startup” updated, and Board Schematics added to section.	 5943

© 2017 Microchip Technology Inc.

Doc. Rev 6384B	Comments (Continued)	Change Request Ref.
	41.11.2 “SPI” Timings: The titles of the figures listed below have been updated. Figure 41-7. “SPI Master Mode with (CPOL = 0 and NCPHA = 1) or (CPOL =1 and NCPHA = 0)” Figure 41-8. “SPI Master Mode with (CPOL = NCPHA = 0) or (CPOL and NCPHA = 1)” Figure 41-9. “SPI Slave Mode with (CPOL = 0 and NCPHA = 1) or (CPOL = 1 and NCPHA = 0)” Figure 41-10. “SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL = NCPHA = 1)”	5592
	Table 41-4, “Power Consumption by Peripheral in Active Mode” updatedvalues in Consumption column. Table 41-15, “PLLA Characteristics” I_{PLL} active mode, current consumption is 8 mA.	5771
	Table 41-3, “Power Consumption for Different Modes” Active mode, “all peripheral clocks de-activated”.	5802
	Errata: Section 44.2.9 “Static Memory Controller (SMC)”, added to datasheet with the errata listed below: Section 44.2.9.1 “SMC: Chip Select Parameters Modification”	5642

Doc. Rev	Comments	Change Request Ref.
6384A	First issue	

Note 1: “rfo” indicates changes requested during document review and approval loop.

The Microchip Web Site

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Design Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

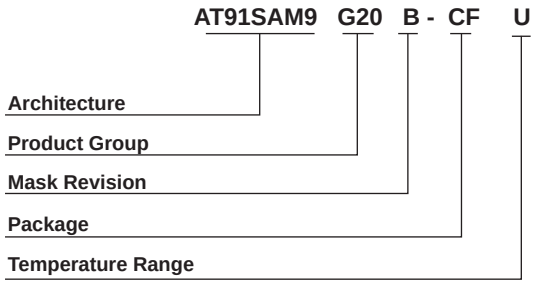
Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support

© 2017 Microchip Technology Inc.

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Architecture:	AT91SAM9 =	Arm926EJ-S processor
Product Group:	G20 =	General-purpose microprocessor
Mask Revision:	B	
Package:	C =	LFBGA217
	CF =	TFBGA247
Temperature Range:	U =	-40°C to +85°C (Industrial)

Example:

a) AT91SAM9G20B-CFU = Arm Arm926EJ-S general-purpose microprocessor, 247-ball, industrial temperature, BGA package.

Note 1: Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-2169-6

© 2017 Microchip Technology Inc.

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2017 Microchip Technology Inc.