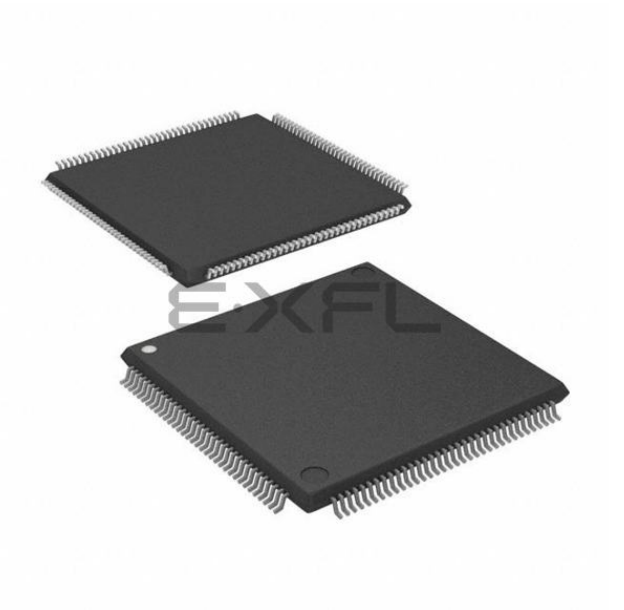


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	ARM7®
Core Size	16/32-Bit
Speed	33MHz
Connectivity	EBI/EMI, SPI, UART/USART
Peripherals	WDT
Number of I/O	54
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91m42800a-33ai-t

Table 2. AT91M42800A Pinout in BGA 144 Package

Pin#	Name
A1	PB1/NCS3
A2	NCS0
A3	NCS1
A4	GND
A5	PLLRCB
A6	GND
A7	PLLRCA
A8	GND
A9	XOUT
A10	XIN
A11	MODE0
A12	PA22/NPCSB1
B1	NUB/NWR1
B2	PB0/NCS2
B3	VDDCORE
B4	NWE/NWR0
B5	VDDPLL
B6	TDO
B7	VDDPLL
B8	NWDOVF
B9	PA26
B10	PA19/MISOB
B11	PA24/NPCSB3
B12	PA23/NPCSB2
C1	NLB/A0
C2	A1
C3	VDDIO
C4	NOE/NRD
C5	VDDIO
C6	NRST
C7	TDI
C8	VDDIO
C9	PA27/BMS
C10	VDDIO
C11	VDDCORE
C12	PA20/MOSIB

Pin#	Name
D1	A2
D2	A3
D3	A4
D4	NWAIT
D5	PA29/PME
D6	PA28
D7	TCK
D8	TMS
D9	MODE1
D10	PA25/MCKO
D11	PA21/NPCSB0
D12	PA18/SPCKB
E1	A7
E2	VDDIO
E3	A6
E4	A5
E5	GND
E6	GND
E7	GND
E8	NTRST
E9	PA13/MOSIA
E10	PA16/NPCSA2
E11	VDDIO
E12	PA17/NPCSA3
F1	A8
F2	A12
F3	A9
F4	A10
F5	GND
F6	GND
F7	GND
F8	GND
F9	PA12/MISOA
F10	PA15/NPCSA1
F11	PA11/SPCKA
F12	PA14/NPCSA0

Pin#	Name
G1	A17
G2	A16
G3	A11
G4	A13
G5	GND
G6	GND
G7	GND
G8	GND
G9	PA9/TXD1/NTRI
G10	PA10/RXD1
G11	PA8/SCK1
G12	PA7/RXD0
H1	A18
H2	VDDIO
H3	A15
H4	A14
H5	A19
H6	GND
H7	GND
H8	GND
H9	PA6/TXD0
H10	PA4/FIQ
H11	VDDIO
H12	PA5/SCK0
J1	PB5/A23/CS4
J2	D0
J3	PB4/A22/CS5
J4	PB3/A21/CS6
J5	PB2/A20/CS7
J6	D15
J7	PB6/TCLK0
J8	PB10/TIOA1
J9	PA3/IRQ3
J10	PA2/IRQ2
J11	PA0/IRQ0
J12	PA1/IRQ1

Pin#	Name
K1	D1
K2	VDDCORE
K3	VDDIO
K4	D9
K5	D10
K6	D14
K7	PB9/TCLK1
K8	PB13/TIOA2
K9	PB11/TIOB1
K10	VDDIO
K11	PB16/TIOA3
K12	PB23/TIOB5
L1	D3
L2	D2
L3	D5
L4	D8
L5	VDDIO
L6	D13
L7	PB8/TIOB0
L8	VDDIO
L9	PB17/TIOB3
L10	VDDCORE
L11	PB20/TIOB4
L12	PB22/TIOA5
M1	D4
M2	D6
M3	D7
M4	D11
M5	D12
M6	PB7/TIOA0
M7	PB12/TCLK2
M8	PB15/TCLK3
M9	PB14/TIOB2
M10	PB18/TCLK4
M11	PB19/TIOA4
M12	PB21/TCLK5

3. Pin Description

Table 3. AT91M42800A Pin Description

Module	Name	Function	Type	Active Level	Comments
EBI	A0 - A23	Address Bus	Output	–	All valid after reset
	D0 - D15	Data Bus	I/O	–	
	CS4 - CS7	Chip Select	Output	High	A23 - A20 after reset
	NCS0 - NCS3	Chip Select	Output	Low	
	NWR0	Lower Byte 0 Write Signal	Output	Low	Used in Byte Write option
	NWR1	Lower Byte 1 Write Signal	Output	Low	Used in Byte Write option
	NRD	Read Signal	Output	Low	Used in Byte Write option
	NWE	Write Enable	Output	Low	Used in Byte Select option
	NOE	Output Enable	Output	Low	Used in Byte Select option
	NUB	Upper Byte Select (16-bit SRAM)	Output	Low	Used in Byte Select option
	NLB	Lower Byte Select (16-bit SRAM)	Output	Low	Used in Byte Select option
	NWAIT	Wait Input	Input	Low	
	BMS	Boot Mode Select	Input	–	Sampled during reset
	PME	Protect Mode Enable	Input	High	PIO-controlled after reset
AIC	IRQ0 - IRQ3	External Interrupt Request	Input	–	PIO-controlled after reset
	FIQ	Fast External Interrupt Request	Input	–	PIO-controlled after reset
TC	TCLK0 - TCLK5	Timer External Clock	Input	–	PIO-controlled after reset
	TIOA0 - TIOA5	Multi-purpose Timer I/O Pin A	I/O	–	PIO-controlled after reset
	TIOB0 - TIOB5	Multi-purpose Timer I/O Pin B	I/O	–	PIO-controlled after reset
USART	SCK0 - SCK1	External Serial Clock	I/O	–	PIO-controlled after reset
	TXD0 - TXD1	Transmit Data Output	Output	–	PIO-controlled after reset
	RXD0 - RXD1	Receive Data Input	Input	–	PIO-controlled after reset
SPIA SPIB	SPCKA/SPCKB	Clock	I/O	–	PIO-controlled after reset
	MISOA/MISOB	Master In Slave Out	I/O	–	PIO-controlled after reset
	MOSIA/MOSIB	Master Out Slave In	I/O	–	PIO-controlled after reset
	NSSA/NSSB	Slave Select	Input	Low	PIO-controlled after reset
	NPCSA0 - NPCSA3 NPCSB0 - NPCSB3	Peripheral Chip Selects	Output	Low	PIO-controlled after reset
PIO	PA0 - PA29	Programmable I/O Port A	I/O	–	Input after reset
	PB0 - PB23	Programmable I/O Port B	I/O	–	Input after reset
ST	NWDOVF	Watchdog Timer Overflow	Output	Low	Open drain

11. EBI: External Bus Interface

The EBI handles the access requests performed by the ARM core or the PDC. It generates the signals that control the access to the external memory or peripheral devices. The EBI is fully programmable and can address up to 64M bytes. It has eight chip selects and a 24-bit address bus, the upper four bits of which are multiplexed with a chip select.

The 16-bit data bus can be configured to interface with 8- or 16-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing.

The EBI supports different access protocols allowing single clock cycle memory accesses.

The main features are:

- External memory mapping
- Up to 8 chip select lines
- 8- or 16-bit data bus
- Byte write or byte select lines
- Remap of boot memory
- Two different read protocols
- Programmable wait state generation
- External wait request
- Programmable data float time

The EBI User Interface is described on page 48.

11.1 External Memory Mapping

The memory map associates the internal 32-bit address space with the external 24-bit address bus.

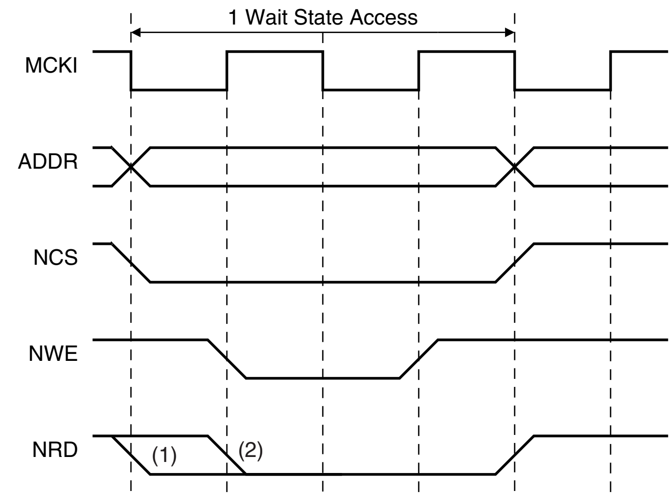
The memory map is defined by programming the base address and page size of the external memories (see registers EBI_CSR0 to EBI_CSR7 in Section 11.13 ”EBI User Interface” on page 48). Note that A0 - A23 is only significant for 8-bit memory; A1 - A23 is used for 16-bit memory.

If the physical memory device is smaller than the programmed page size, it wraps around and appears to be repeated within the page. The EBI correctly handles any valid access to the memory device within the page (see Figure 11-1 on page 24).

In the event of an access request to an address outside any programmed page, an abort signal is generated. Two types of abort are possible: instruction prefetch abort and data abort. The corresponding exception vector addresses are 0x0000000C and 0x00000010, respectively. It is up to the system programmer to program the error handling routine to use in case of an abort (see the ARM7TDMI datasheet for further information).

The chip selects can be defined to the same base address and an access to the overlapping address space asserts both NCS lines. The Chip Select Register, having the smaller number, defines the characteristics of the external access and the behaviour of the control signals.

Figure 11-13. One Wait State Access



Notes: 1. Early Read Protocol
2. Standard Read Protocol

11.11.2 Data Float Wait State

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

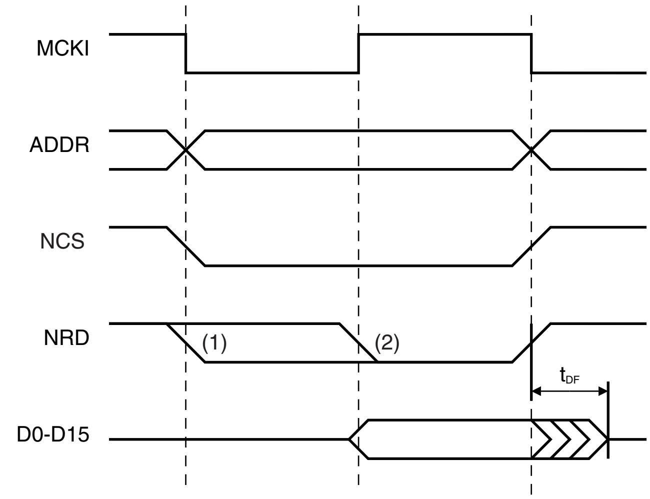
The data float output time (t_{DF}) for each external memory device is programmed in the TDF field of the EBI_CSR register for the corresponding chip select. The value (0 - 7 clock cycles) indicates the number of data float waits to be inserted and represents the time allowed for the data output to go 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 EBI keeps track of the programmed external data float time during internal accesses, to ensure that the external memory system is not accessed while it is still busy.

Internal memory accesses and consecutive accesses to the same external memory do not have added data float wait states.

Figure 11-14. Data Float Output Time



Notes: 1. Early Read Protocol
2. Standard Read Protocol

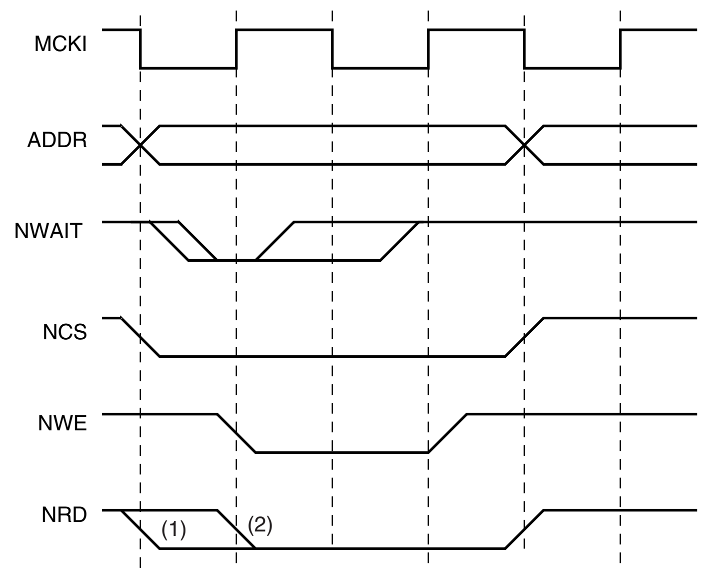
11.11.3 External Wait

The NWAIT input can be used to add wait states at any time. NWAIT is active low and is detected on the rising edge of the clock.

If NWAIT is low at the rising edge of the clock, the EBI adds a wait state and changes neither the output signals nor its internal counters and state. When NWAIT is de-asserted, the EBI finishes the access sequence.

The NWAIT signal must meet setup and hold requirements on the rising edge of the clock.

Figure 11-15. External Wait



Notes: 1. Early Read Protocol
2. Standard Read Protocol

Figure 11-19. Standard Read Protocol with t_{DF}

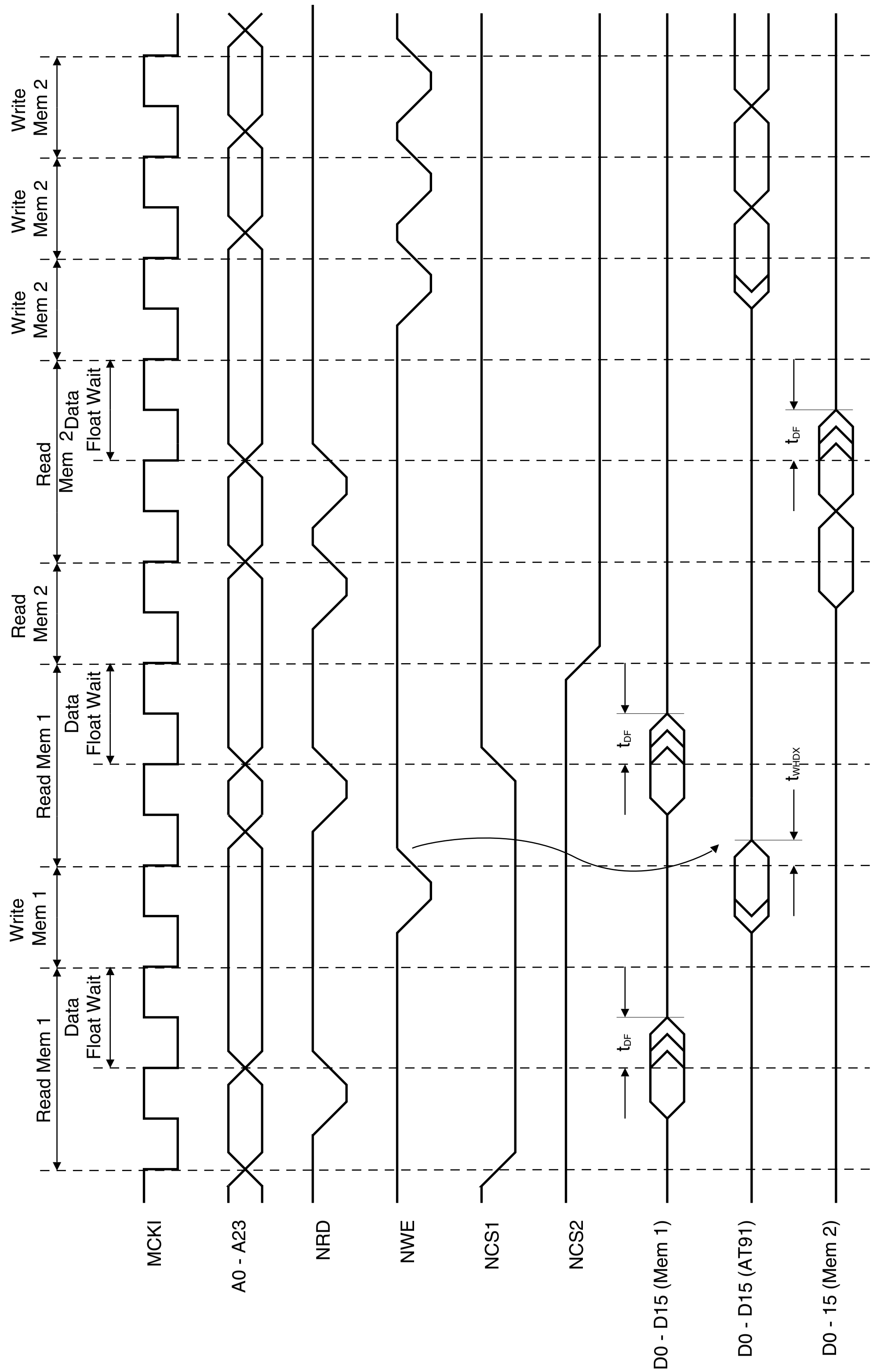
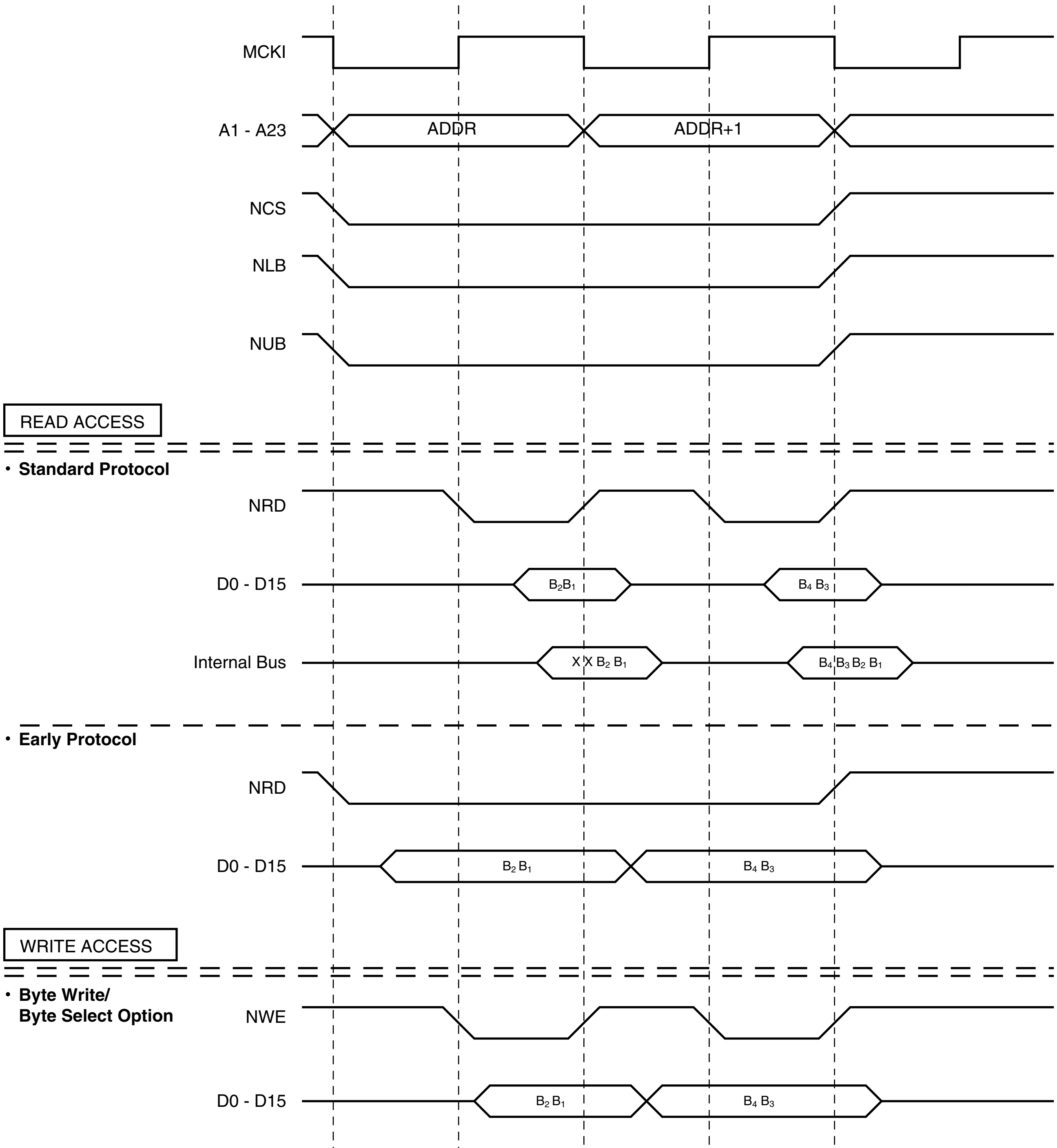


Figure 11-21. 0 Wait States, 16-bit Bus Width, Word Transfer



12.6 PMC User Interface

Base Address: 0xFFFF4000 (Code Label `PMC_BASE`)

Table 4. PMC Registers

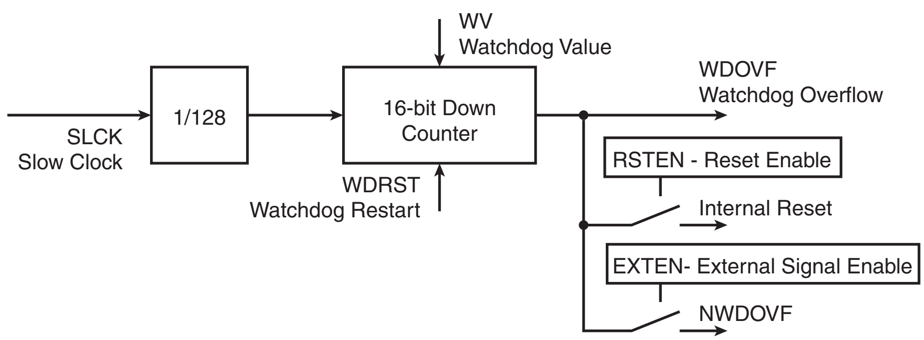
Offset	Register Name	Register Mnemonic	Access	Reset Value
0x00	System Clock Enable Register	PMC_SCER	Write-only	–
0x04	System Clock Disable Register	PMC_SCDR	Write-only	–
0x08	System Clock Status Register	PMC_SCSR	Read-only	0x00000001
0x0C	Reserved	–	–	–
0x10	Peripheral Clock Enable Register	PMC_PCER	Write-only	–
0x14	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x18	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x00000000
0x1C	Reserved	–	–	–
0x20	Clock Generator Mode Register	PMC_CGMR	Read/Write	0x00000000
0x24	Reserved	–	–	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Status Register	PMC_SR	Read-only	0x00000000
0x34	Interrupt Enable Register	PMC_IER	Write-only	–
0x38	Interrupt Disable Register	PMC_IDR	Write-only	–
0x3C	Interrupt Mask Register	PMC_IMR	Read-only	0x00000000

If an overflow does occur, the Watchdog Timer:

- Sets the WDOVF in ST_SR (Status Register) from which an interrupt can be generated
- Generates a pulse for 8 slow clock cycles on the external signal NWDOVF if the bit EXTEN in ST_WDMR is set
- Generates an internal reset if the parameter RSTEN in ST_WDMR is set
- Reloads and restarts the down counter

Writing the ST_WDMR does not reload or restart the down counter. When the ST_CR is written the watchdog is immediately reloaded from ST_WDMR and restarted. The slow clock 128 divider is also immediately reset and restarted. When the ARM7TDMI enters debug mode, the output of the slow clock divider stops, preventing any internal or external reset during the debugging phase.

Figure 13-3. Watchdog Timer



13.3 RTT: Real-time Timer

The Real-time Timer can be used to count elapsed seconds. It is built around a 20-bit counter fed by the Slow Clock divided by a programmable value. At reset this value is set to 0x8000, corresponding to feeding the real-time counter with a 1 Hz signal when the Slow Clock is 32.768 Hz. The 20-bit counter can count up to 1048576 seconds, corresponding to more than 12 days, then roll over to 0.

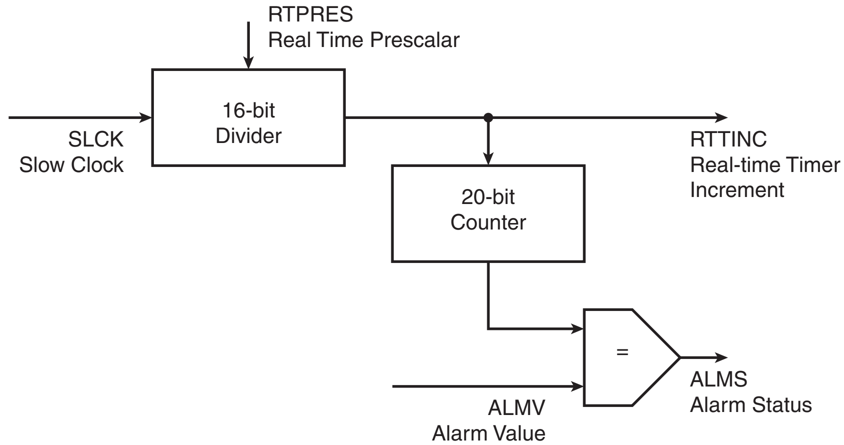
The Real-time Timer value can be read at any time in the register ST_CRTR (Current Real-time Register). As this value can be updated asynchronously to the Master Clock, it is advisable to read this register twice at the same value to improve accuracy of the returned value.

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

The bit RTTINC in ST_SR is set each time the 20-bit counter is incremented. This bit can be used to start an interrupt, or generate a one-second signal.

Writing the ST_RTMR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 20-bit counter.

Figure 13-4. Real-time Timer



Note: If RTPRES is programmed with a period less or equal to the current MCK period, the update of the RTTINC and ALMS status bits and their associated interrupt generation are unpredictable.

behavior by writing to the AIC_EOICR (End of Interrupt) before returning to the interrupted software. It also can perform other operation(s), e.g., trace possible undesirable behavior.

14.9 Protect Mode

The Protect Mode permits reading of the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system.

When a Debug Monitor or an ICE reads the AIC User Interface, the IVR could be read. This would have the following consequences in normal mode.

- 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 would be necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system. Hence the debug system would become strongly intrusive, and could cause the application to enter an undesired state.

This is avoided by using Protect mode.

The Protect mode is enabled by setting the AIC bit in the SF Protect Mode Register (see ”SF: Special Function Registers” on page 115).

When 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 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 performed in between the read and the write can cause unpredictable results. Therefore, it is strongly recommended not to set a breakpoint between these two actions, nor to stop the software.

The debug system must not write to the AIC_IVR as this would cause undesirable effects.

The following table shows the main steps of an interrupt and the order in which they are performed according to the mode:

Action	Normal Mode	Protect Mode
Calculate active interrupt (higher than current or spurious)	Read AIC_IVR	Read AIC_IVR
Determine and return the vector of the active interrupt	Read AIC_IVR	Read AIC_IVR
Memorize interrupt	Read AIC_IVR	Read AIC_IVR
Push on internal stack the current priority level	Read AIC_IVR	Write AIC_IVR
Acknowledge the interrupt(1)	Read AIC_IVR	Write AIC_IVR
No effect(2)	Write AIC_IVR	–

Notes: 1. NIRQ de-assertion and automatic interrupt clearing if the source is programmed as level sensitive.
2. Software that has been written and debugged using Protect mode will run 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.

15.20 PIO Output Data Status Register

Register Name: PIO_ODSR
Access Type: Read-only
Offset: 0x38
Reset Value: 0

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

This register shows the output data status which is programmed in PIO_SODR or PIO_CODR. The defined value is effective only if the pin is controlled by the PIO Controller and only if the pin is defined as an output.

0 = The output data for the corresponding line is programmed to 0.

1 = The output data for the corresponding line is programmed to 1.

15.21 PIO Pin Data Status Register

Register Name: PIO_PDSR
Access Type: Read-only
Offset: 0x3C
Reset Value: Undefined

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

This register shows the state of the physical pin of the chip. The pin values are always valid, regardless of whether the pins are enabled as PIO, peripheral, input or output. The register reads as follows:

0 = The corresponding pin is at logic 0.

1 = The corresponding pin is at logic 1.

Calculation of time-out duration:

Duration = Value x 4 x Bit Period

17.4 Transmitter

The transmitter has the same behavior in both synchronous and asynchronous operating modes. Start bit, data bits, parity bit and stop bits are serially shifted, lowest significant bit first, on the falling edge of the serial clock. See example in Figure 17-6.

The number of data bits is selected in the CHRL field in US_MR.

The parity bit is set according to the PAR field in US_MR.

The number of stop bits is selected in the NBSTOP field in US_MR.

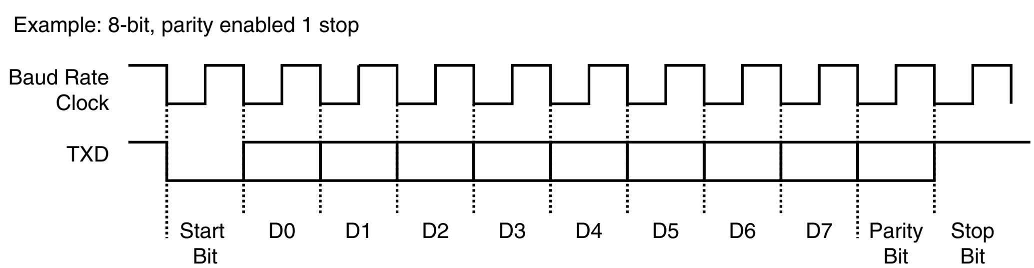
When a character is written to US_THR (Transmit Holding), it is transferred to the Shift Register as soon as it is empty. When the transfer occurs, the TXRDY bit in US_CSR is set until a new character is written to US_THR. If Transmit Shift Register and US_THR are both empty, the TXEMPTY bit in US_CSR is set.

17.4.1 Time-guard

The Time-guard function allows the transmitter to insert an idle state on the TXD line between two characters. The duration of the idle state is programmed in US_TTGR (Transmitter Time-guard). When this register is set to zero, no time-guard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in US_TTGR.

$$\begin{array}{c}\text{Idle state duration}\\\text{between two characters}\end{array} = \begin{array}{c}\text{Time-guard}\\\text{Value}\end{array} \times \begin{array}{c}\text{Bit}\\\text{Period}\end{array}$$

Figure 17-6. Synchronous and Asynchronous Modes: Character Transmission



17.5 Multi-drop Mode

When the field PAR in US_MR equals 11X (binary value), the USART is configured to run in Multi-drop mode. In this case, the parity error bit PARE in US_CSR is set when data is detected with a parity bit set to identify an address byte. PARE is cleared with the Reset Status Bits Command (RSTSTA) in US_CR. If the parity bit is detected low, identifying a data byte, PARE is not set.

The transmitter sends an address byte (parity bit set) when a Send Address Command (SENDA) is written to US_CR. In this case, the next byte written to US_THR will be transmitted as an address. After this any byte transmitted will have the parity bit cleared.

Access Type: Read/Write
Offset: 0x04
Reset Value: 0x0

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

23	22	21	20	19	18	17	16
–	–	–	–	–	CLKO	MODE9	–

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		–	–	–	–

- **USCLKS: Clock Selection (Baud Rate Generator Input Clock)**

USCLKS		Selected Clock	Code Label: `US_CLKS`
0	0	MCK	`US_CLKS_MCK`
0	1	MCK/8	`US_CLKS_MCK8`
1	0	Slow Clock	`US_CLKS_SLCK`
1	1	External (SCK)	`US_CLKS_SCK`

- **CHRL: Character Length**

CHRL		Character Length	Code Label: `US_CHRL`
0	0	Five bits	`US_CHRL_5`
0	1	Six bits	`US_CHRL_6`
1	0	Seven bits	`US_CHRL_7`
1	1	Eight bits	`US_CHRL_8`

Start, stop and parity bits are added to the character length.

- **SYNC: Synchronous Mode Select (Code Label `US_SYNC`)**

0 = USART operates in Asynchronous Mode.

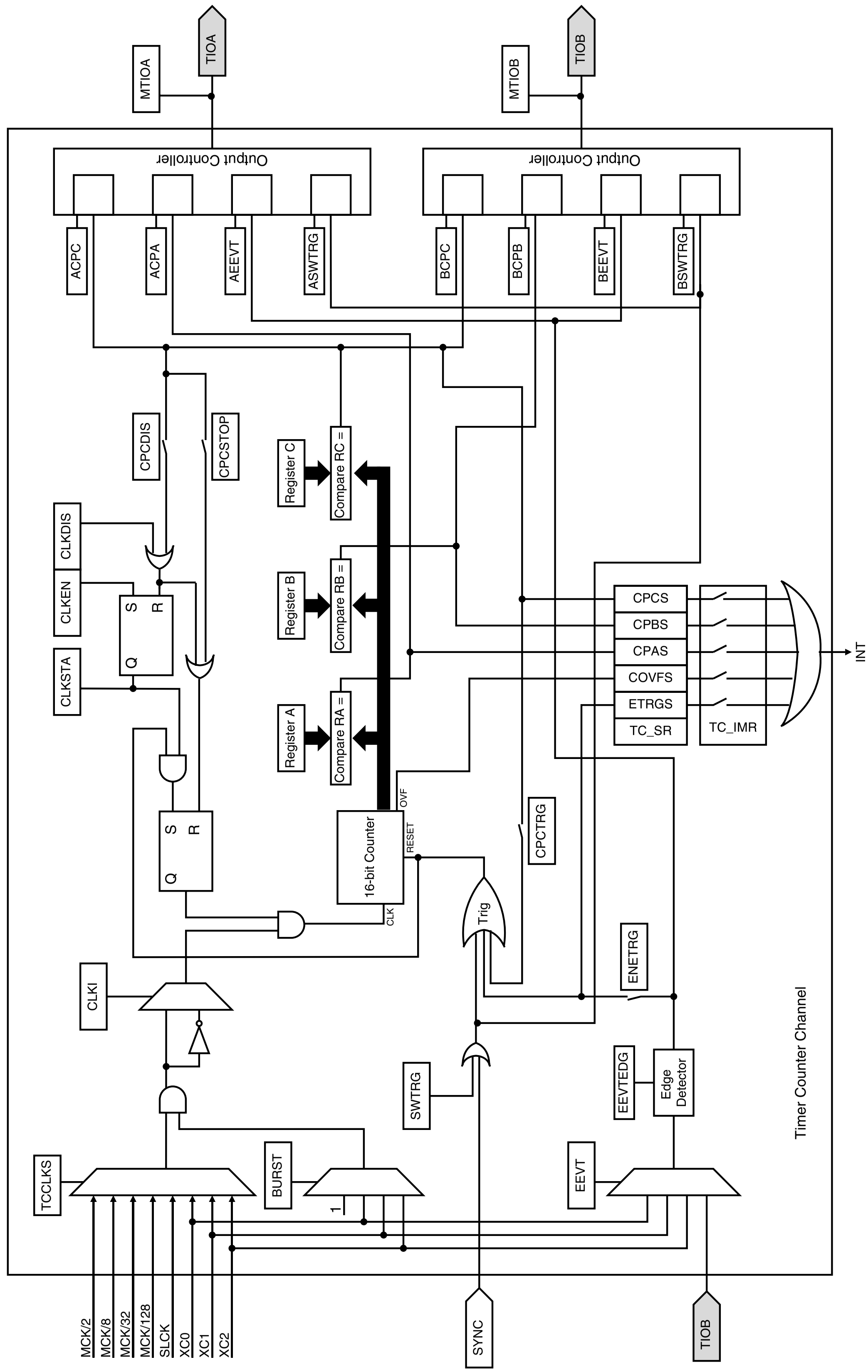
1 = USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type	Code Label: `US_PAR`
0	0	0	Even Parity	`US_PAR_EVEN`
0	0	1	Odd Parity	`US_PAR_ODD`
0	1	0	Parity forced to 0 (Space)	`US_PAR_SPACE`
0	1	1	Parity forced to 1 (Mark)	`US_PAR_MARK`
1	0	x	No parity	`US_PAR_NO`
1	1	x	Multi-drop mode	`US_PAR_MULTIDROP`

- **NBSTOP: Number of Stop Bits**

Figure 18-5. Waveform Mode



18.12 TC Register A

Register Name: TC_RA

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

Offset: 0x14

Reset Value: 0x0

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

- **RA: Register A (Code Label `TC_RA`)**

RA contains the Register A value in real time.

18.13 TC Register B

Register Name: TC_RB

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

Offset: 0x18

Reset Value: 0x0

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

- **RB: Register B (Code Label `TC_RB`)**

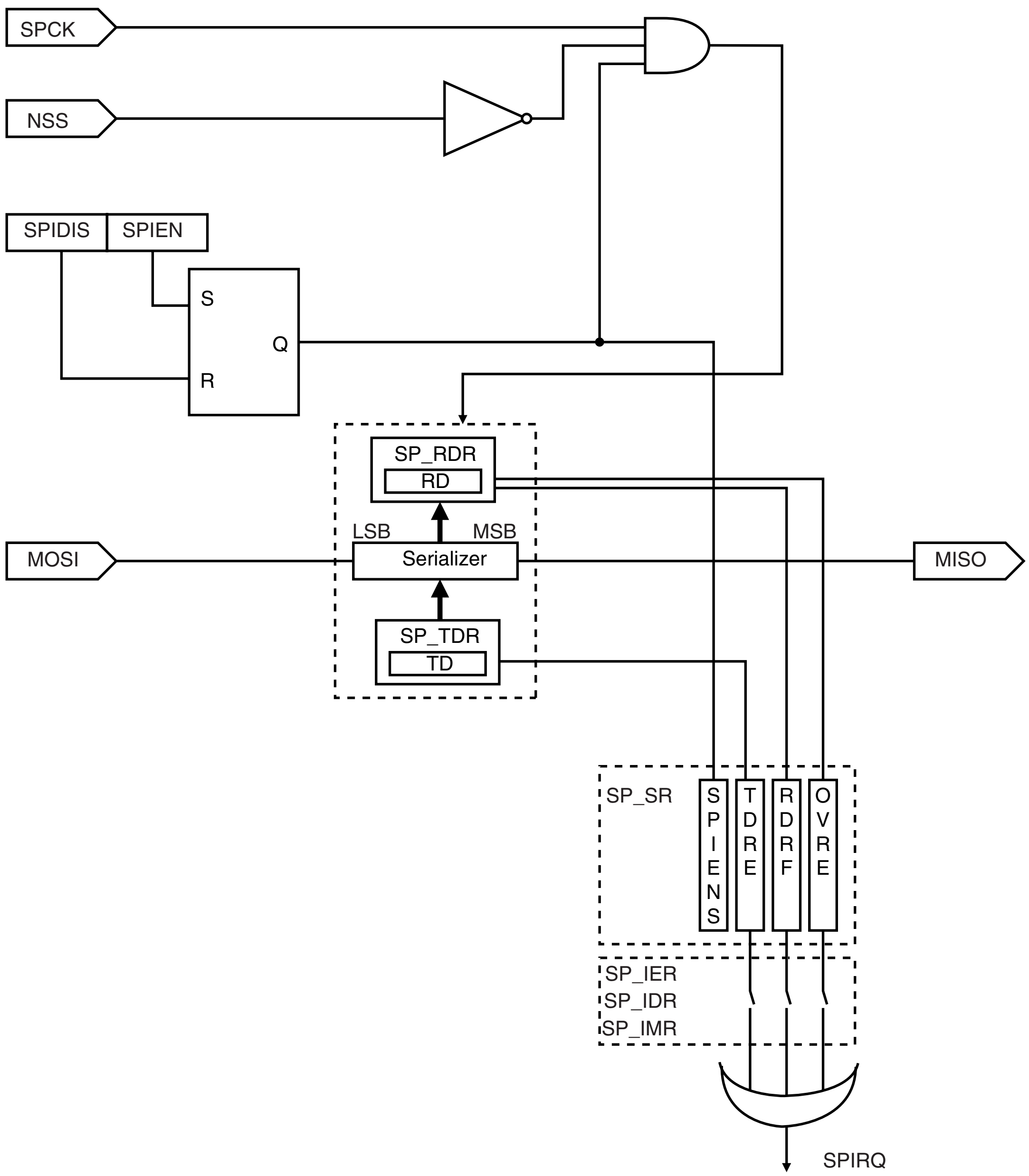
RB contains the Register B value in real time.

19.3 Slave Mode

In Slave Mode, the SPI waits for NSS to go active low before receiving the serial clock from an external master.

In slave mode CPOL, NCPHA and BITS fields of SP_CSR0 are used to define the transfer characteristics. The other Chip Select Registers are not used in slave mode.

Figure 19-4. SPI in Slave Mode



19.7 SPI Programmer's Model

SPIA Base Address: 0xFFFC8000

SPIB Base Address: 0xFFFCC000

Table 9. SPI Memory Map

Offset	Register	Name	Access	Reset State
0x00	Control Register	SP_CR	Write-only	–
0x04	Mode Register	SP_MR	Read/Write	0
0x08	Receive Data Register	SP_RDR	Read-only	0
0x0C	Transmit Data Register	SP_TDR	Write-only	–
0x10	Status Register	SP_SR	Read-only	0
0x14	Interrupt Enable Register	SP_IER	Write-only	–
0x18	Interrupt Disable Register	SP_IDR	Write-only	–
0x1C	Interrupt Mask Register	SP_IMR	Read-only	0
0x20	Receive Pointer Register	SP_RPR	Read/Write	0
0x24	Receive Counter Register	SP_RCR	Read/Write	0
0x28	Transmit Pointer Register	SP_TPR	Read/Write	0
0x2C	Transmit Counter Register	SP_TCR	Read/Write	0
0x30	Chip Select Register 0	SP_CSR0	Read/Write	0
0x34	Chip Select Register 1	SP_CSR1	Read/Write	0
0x38	Chip Select Register 2	SP_CSR2	Read/Write	0
0x3C	Chip Select Register 3	SP_CSR3	Read/Write	0

19.10 SPI Receive Data Register

Register Name: SP_RDR

Access Type: Read-only

Offset: 0x08

Reset Value: 0x0

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 (Code Label `SP_RD`)**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select Status**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

• **SCBR: Serial Clock Baud Rate (Code Label `SP_SCBR`)**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the SPI Master Clock (selected between MCK and MCK/32). The baud rate is selected by writing a value from 2 to 255 in the field SCBR. The following equation determines the SPCK baud rate:

$$\text{SPCK\_Baud\_Rate} = \frac{\text{SPI\_Master\_Clock\_frequency}}{2 \times \text{SCBR}}$$

Giving SCBR a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers may occur. At reset, baud rate is disabled.

• **DLYBS: Delay Before SPCK (Code Label `SP_DLYBS`)**

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 equation determines the delay:

NPCS_to_SPCK_Delay = DLYBS • SPI_Master_Clock_period

• **DLYBCT: Delay Between Consecutive Transfers (Code Label `SP_DLYBCT`)**

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, a delay of four SPI Master Clock periods are inserted.

Otherwise, the following equation determines the delay:

Delay_After_Transfer = 32 • DLYBCT • SPI_Master_Clock_period