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	75MHz
Connectivity	EBI/EMI, UART/USART
Peripherals	WDT
Number of I/O	32
Program Memory Size	2MB (1M x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 1.95V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	121-TFBGA
Supplier Device Package	121-BGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91fr40162s-cj

7.1 System Peripherals

7.1.1 PS: Power-saving

The power-saving feature optimizes power consumption, enabling the software to stop the ARM7TDMI clock (idle mode), restarting it when the module receives an interrupt (or reset). It also enables on-chip peripheral clocks to be enabled and disabled individually, matching power consumption and application needs.

7.1.2 AIC: Advanced Interrupt Controller

The Advanced Interrupt Controller has an 8-level priority, individually maskable, vectored interrupt controller, and drives the NIRQ and NFIQ pins of the ARM7TDMI from:

- The external fast interrupt line (FIQ)
- The three external interrupt request lines (IRQ0 - IRQ2)
- The interrupt signals from the on-chip peripherals

The AIC is extensively programmable offering maximum flexibility, and its vectoring features reduce the real-time overhead in handling interrupts.

The AIC also features a spurious vector detection feature, which reduces spurious interrupt handling to a minimum, and a protect mode that facilitates the debug capabilities.

7.1.3 PIO: Parallel I/O Controller

The AT91FR40162S has 32 programmable I/O lines. Six pins are dedicated as general-purpose I/O pins. Other I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins. The PIO controller enables generation of an interrupt on input change and insertion of a simple input glitch filter on any of the PIO pins.

7.1.4 WD: Watchdog

The Watchdog is built around a 16-bit counter and is used to prevent system lock-up if the software becomes trapped in a deadlock. It can generate an internal reset or interrupt, or assert an active level on the dedicated pin NWDOVF. All programming registers are password-protected to prevent unintentional programming.

7.1.5 SF: Special Function

The AT91FR40162S provides registers that implement the following special functions.

- Chip Identification
- RESET Status
- Protect Mode

9. Peripheral Memory Map

Figure 9-1. Peripheral Memory Map

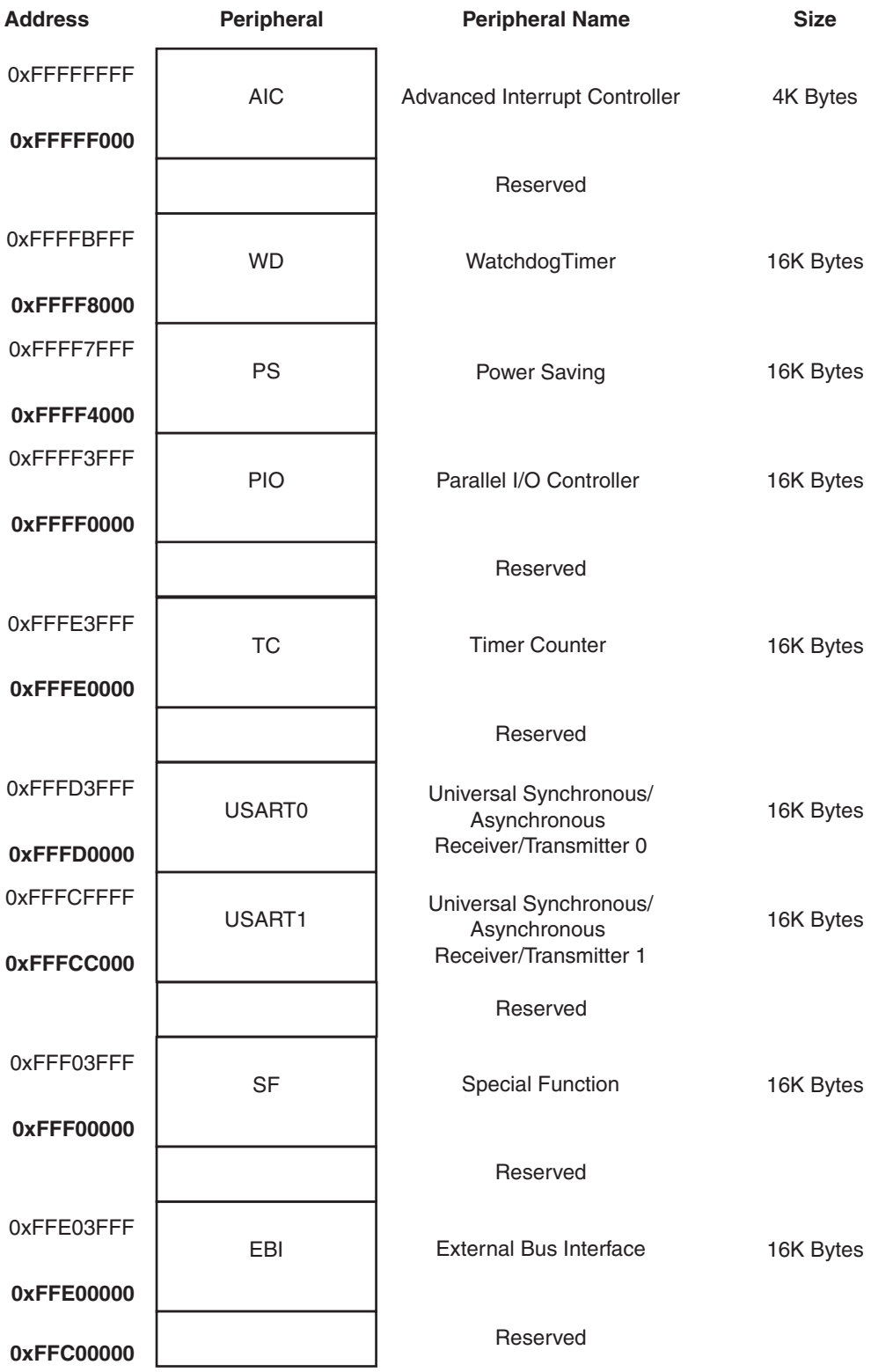


Figure 10-19. Standard Read Protocol with t_{DF}

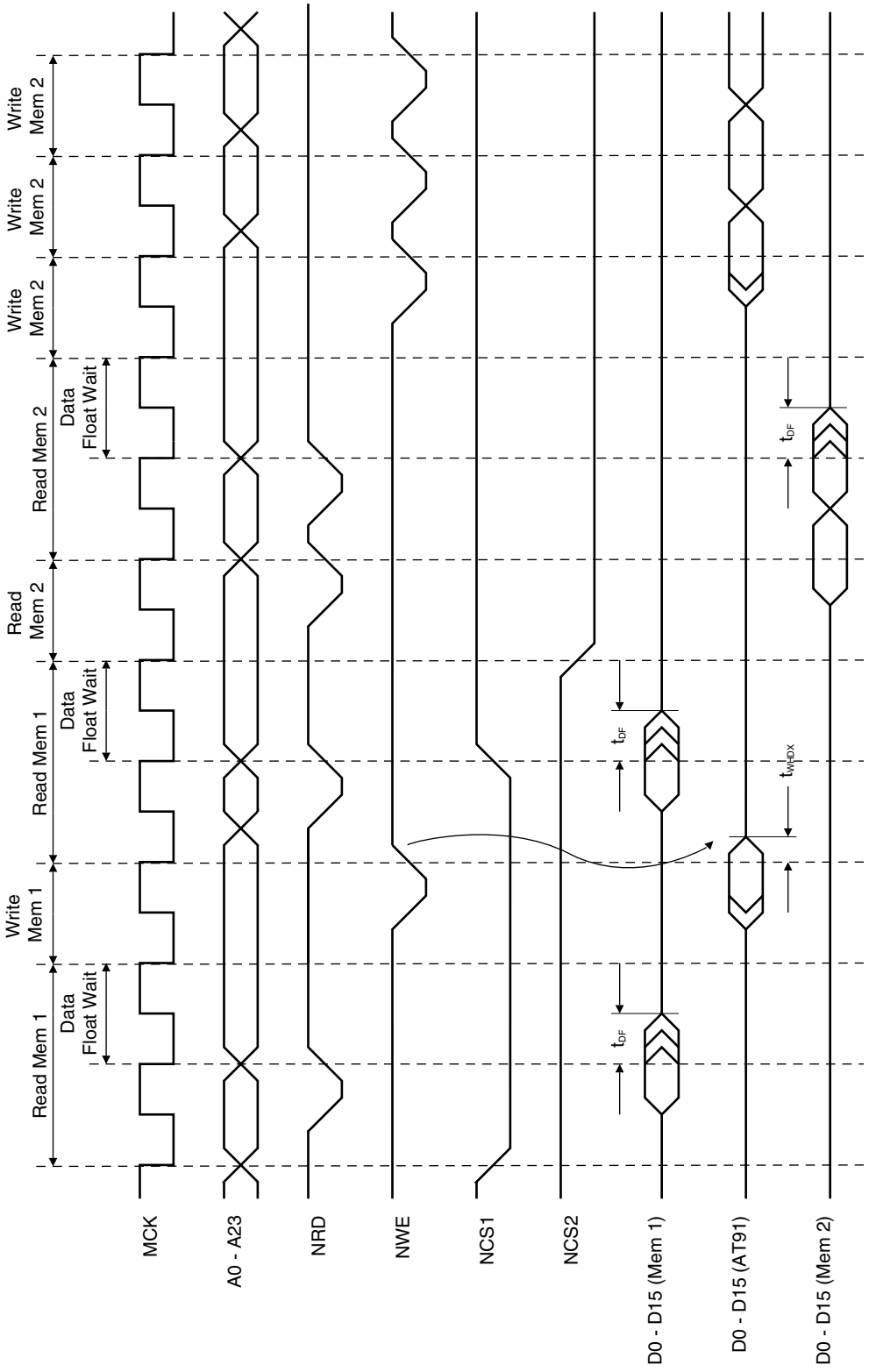


Figure 10-27. 0 Wait States, 16-bit Bus Width, Byte Transfer

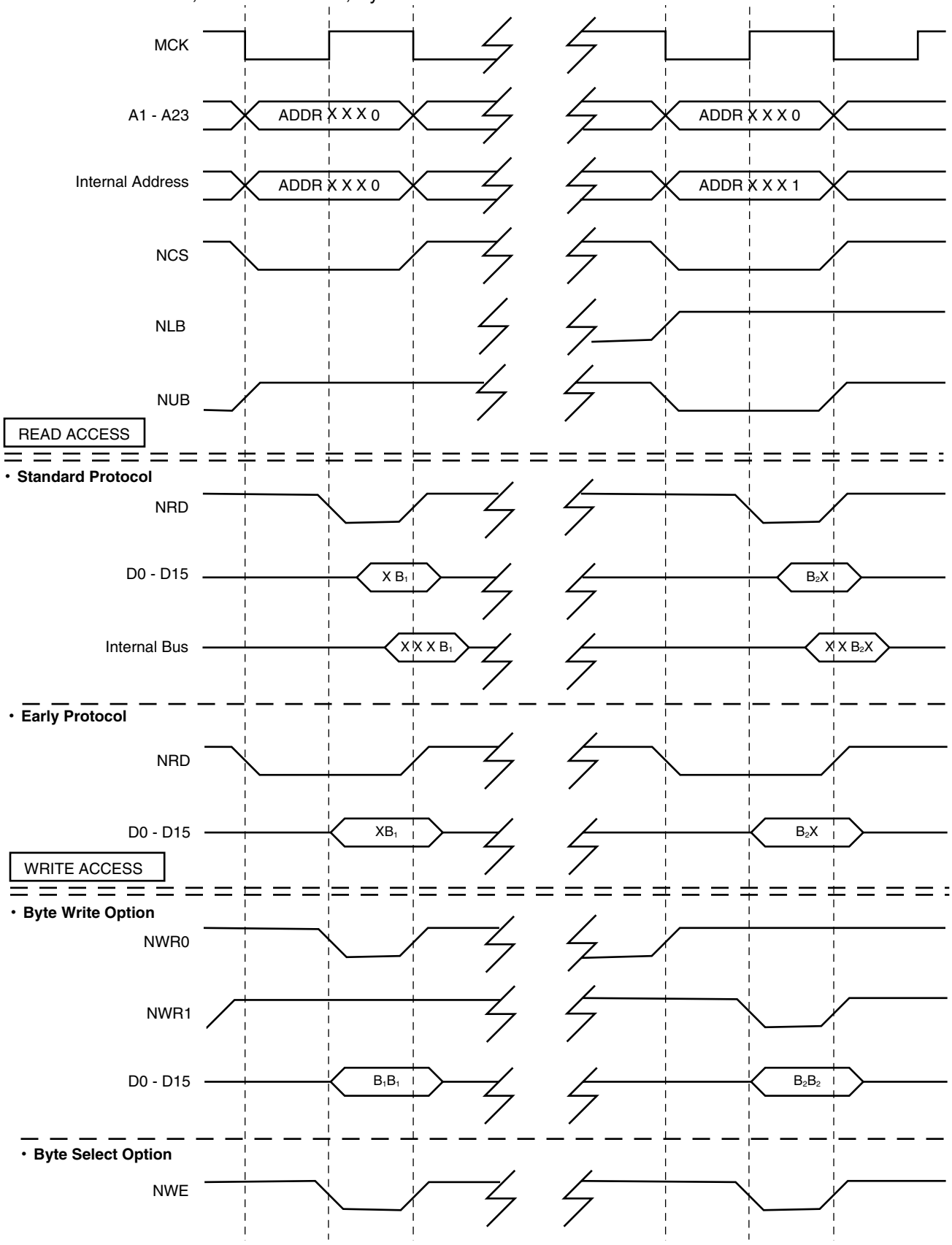
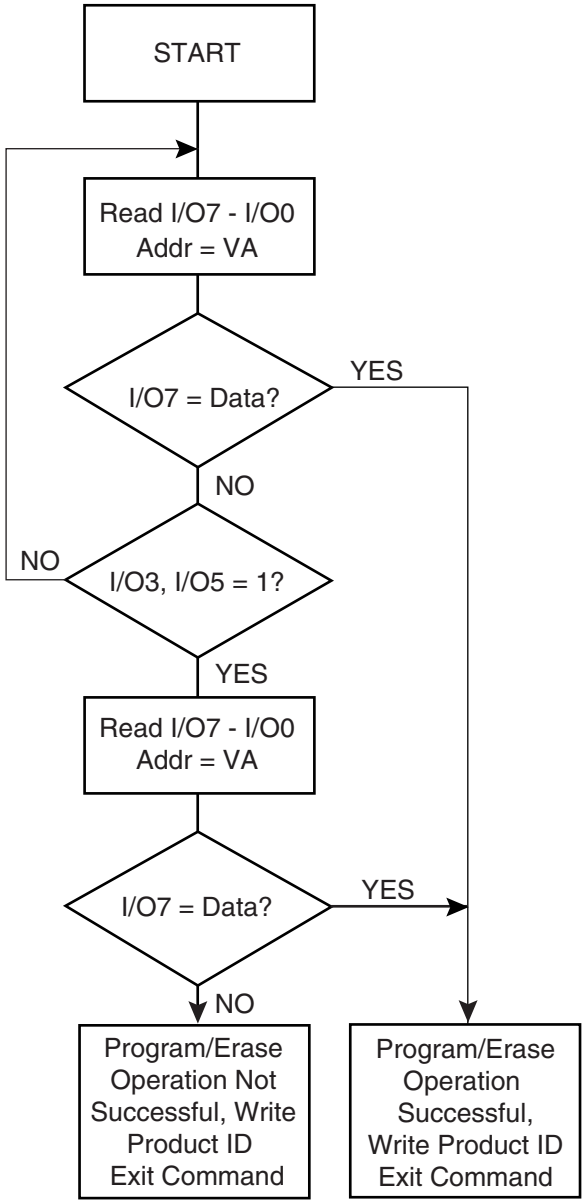
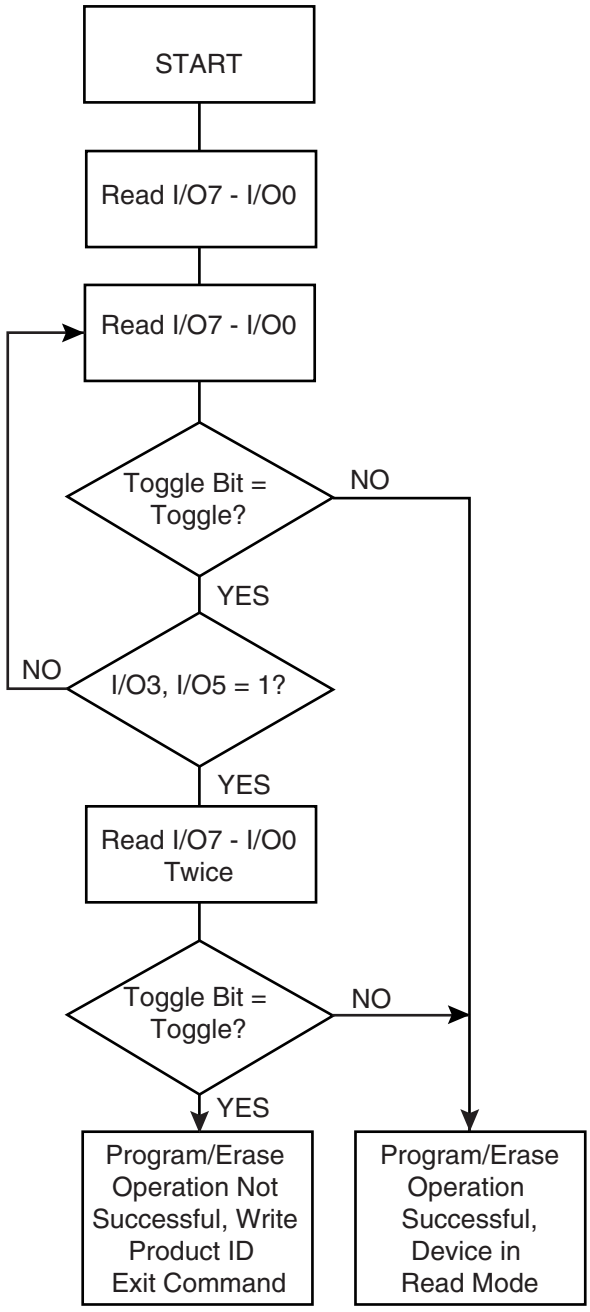


Figure 11-3. $\overline{\text{Data}}$ Polling Algorithm (Configuration Register = 01)



Notes: 1. VA = Valid address for programming. During a sector erase operation, a valid address is any sector address within the sector being erased. During chip erase, a valid address is any non-protected sector address.
2. I/O7 should be rechecked even if I/O5 = “1” because I/O7 may change simultaneously with I/O5.

Figure 11-4. Toggle Bit Algorithm (Configuration Register = 00)



Note: The system should recheck the toggle bit even if I/O5 = “1” because the toggle bit may stop toggling as I/O5 changes to “1”.

12.2.3 PS Peripheral Clock Disable Register

Name: PS_PCDR

Access: Write-only

Offset: 0x08

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PIO
7	6	5	4	3	2	1	0
–	TC2	TC1	TC0	US1	US0	–	–

- **US0: USART 0 Clock Disable**

0 = No effect.

1 = Disables the USART 0 clock.

- **US1: USART 1 Clock Disable**

0 = No effect.

1 = Disables the USART 1 clock.

- **TC0: Timer Counter 0 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 0 clock.

- **TC1: Timer Counter 1 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 1 clock.

- **TC2: Timer Counter 2 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 2 clock.

- **PIO: Parallel IO Clock Disable**

0 = No effect.

1 = Disables the Parallel IO clock.

13.5 Interrupt Masking

Each interrupt source, including FIQ, can be enabled or disabled using the command registers AIC_IECR and AIC_IDCR. The interrupt mask can be read in the read-only register AIC_IMR. A disabled interrupt does not affect the servicing of other interrupts.

13.6 Interrupt Clearing and Setting

All interrupt sources which are programmed to be edge triggered (including FIQ) can be individually set or cleared by respectively writing to the registers AIC_ISCR and AIC_ICCR. This function of the interrupt controller is available for auto-test or software debug purposes.

13.7 Fast Interrupt Request

The external FIQ line is the only source which can raise a fast interrupt request to the processor. Therefore, it has no priority controller.

The external FIQ line can be programmed to be positive or negative edge triggered or high- or low-level sensitive in the AIC_SMR0 register.

The fast interrupt handler address can be stored in the AIC_SVR0 register. The value written into this register is available by reading the AIC_FVR register when an FIQ interrupt is raised. By storing the following instruction at address 0x0000001C, the processor will load the program counter with the interrupt handler address stored in the AIC_FVR register.

```
ldr     PC,[PC,# -&F20]
```

Alternatively the interrupt handler can be stored starting from address 0x0000001C as described in the ARM7TDMI datasheet.

13.8 Software Interrupt

Interrupt source 1 of the advanced interrupt controller is a software interrupt. It must be programmed to be edge triggered in order to set or clear it by writing to the AIC_ISCR and AIC_ICCR.

This is totally independent of the SWI instruction of the ARM7TDMI processor.

13.9 Spurious Interrupt

When the AIC asserts the NIRQ line, the ARM7TDMI enters IRQ Mode and the interrupt handler reads the IVR. It may happen that the AIC de-asserts the NIRQ line after the core has taken into account the NIRQ assertion and before the read of the IVR.

This behavior is called a Spurious Interrupt.

The AIC is able to detect these Spurious Interrupts and returns the Spurious Vector when the IVR is read. The Spurious Vector can be programmed by the user when the vector table is initialized.

A spurious interrupt may occur in the following cases:

- With any sources programmed to be level sensitive, if the interrupt signal of the AIC input is de-asserted at the same time as it is taken into account by the ARM7TDMI.
- If an interrupt is asserted at the same time as the software is disabling the corresponding source through AIC_IDCR (this can happen due to the pipelining of the ARM core).

13.13.4 AIC FIQ Vector Register

Register Name: AIC_FVR

Access Type: Read-only

Reset Value: 0

Offset: 0x104

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 which corresponds to FIQ.

13.13.5 AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read-only

Reset Value: 0

Offset: 0x108

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 IRQ Identifier (Code Label `AIC_IRQID`)**

The Interrupt Status Register returns the current interrupt source number.

14.4 Interrupts

Each parallel I/O can be programmed to generate an interrupt when a level change occurs. This is controlled by the PIO_IER (Interrupt Enable) and PIO_IDR (Interrupt Disable) registers which enable/disable the I/O interrupt by setting/clearing the corresponding bit in the PIO_IMR. When a change in level occurs, the corresponding bit in the PIO_ISR (Interrupt Status) is set whether the pin is used as a PIO or a peripheral and whether it is defined as input or output. If the corresponding interrupt in PIO_IMR (Interrupt Mask) is enabled, the PIO interrupt is asserted.

When PIO_ISR is read, the register is automatically cleared.

14.5 User Interface

Each individual I/O is associated with a bit position in the Parallel I/O user interface registers. Each of these registers are 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero.

14.6.15 PIO Interrupt Disable Register

Register Name: PIO_IDR

Access Type: Write-only

Offset: 0x44

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 is used to disable PIO interrupts on the corresponding pin. It has effect whether the PIO is enabled or not.

1 = Disables the interrupt on the corresponding pin. Logic level changes are still detected.

0 = No effect.

14.6.16 PIO Interrupt Mask Register

Register Name: PIO_IMR

Access Type: Read-only

Reset Value: 0

Offset: 0x48

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 which pins have interrupts enabled. It is updated when interrupts are enabled or disabled by writing to PIO_IER or PIO_IDR.

1 = Interrupt is enabled on the corresponding input pin.

0 = Interrupt is not enabled on the corresponding input pin.

16.2 SF User Interface

Chip ID Base Address = 0xFFF00000 (Code Label `SF_BASE`)

Table 16-2. SF Memory Map

Offset	Register	Name	Access	Reset State
0x00	Chip ID Register	SF_CIDR	Read-only	Hardwired
0x04	Chip ID Extension Register	SF_EXID	Read-only	Hardwired
0x08	Reset Status Register	SF_RSR	Read-only	See register description
0x10	Reserved	–	–	–
0x14	Reserved	–	–	–
0x18	Protect Mode Register	SF_PMR	Read/Write	0x0

- **VDSIZ: Volatile Data Memory Size**

VDSIZ				Size	Code Label
					SF_VDSIZ
0	0	0	0	None	SF_VDSIZ_NONE
0	0	0	1	1K bytes	SF_VDSIZ_1K
0	0	1	0	2K bytes	SF_VDSIZ_2K
0	1	0	0	4K bytes	SF_VDSIZ_4K
1	0	0	0	8K bytes	SF_VDSIZ_8K
Others				Reserved	–

- **ARCH: Chip Architecture (Code Label `SF_ARCH`)**

Code of Architecture: Two BCD digits.

0100 0000	AT91x40yyy	Code Label
		SF_ARCH_AT91x40

- **NVPTYP: Non Volatile Program Memory Type**

NVPTYP			Type	Code Label
				SF_NVPTYP
0	0	0	Reserved	–
0	0	1	“F” Series	SF_NVPTYP_M
1	x	x	Reserved	–
1	0	0	“R” Series	SF_NVPTYP_R

- **EXT: Extension Flag (Code Label `SF_EXT`)**

0 = Chip ID has a single register definition without extensions

1 = An extended Chip ID exists (to be defined in the future).

17.2 Pin Description

Each USART channel has the following external signals:

Table 17-1.

Name	Description
SCK	USART Serial clock can be configured as input or output: SCK is configured as input if an External clock is selected (USCLKS[1] = 1) SCK is driven as output if the External Clock is disabled (USCLKS[1] = 0) and Clock output is enabled (CLKO = 1)
TXD	Transmit Serial Data is an output
RXD	Receive Serial Data is an input

Notes: 1. After a hardware reset, the USART pins are not enabled by default (see "PIO: Parallel I/O Controller" on page 91). The user must configure the PIO Controller before enabling the transmitter or receiver.
2. If the user selects one of the internal clocks, SCK can be configured as a PIO.

17.10.2 USART Mode Register

Name: US_MR

Access Type: Read/Write

Reset Value: 0

Offset: 0x04

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	X	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.

17.10.13 USART Receive Counter Register

Name: US_RCR

Access Type: Read/Write

Reset Value: 0

Offset: 0x34

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	4920	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**

RXCTR must be loaded with the size of the receive buffer.

0: Stop Peripheral Data Transfer dedicated to the receiver.

1 - 65535: Start Peripheral Data transfer if RXRDY is active.

17.10.14 USART Transmit Pointer Register

Name: US_TPR

Access Type: Read/Write

Reset Value: 0

Offset: 0x38

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

- **TXPTR: Transmit Pointer**

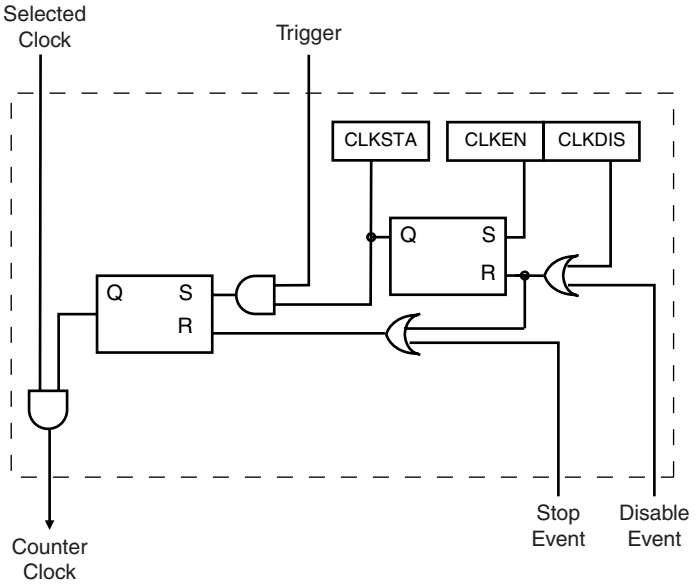
TXPTR must be loaded with the address of the transmit buffer.

18.3.3 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped.

- 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 18-3. Clock Control



18.3.4 Timer Counter Operating Modes

Each Timer Counter channel can independently operate in two different modes:

- Capture Mode allows measurement on signals
- Waveform Mode allows wave generation

The Timer Counter Operating Mode is programmed with the WAVE bit in the TC 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.

18.3.5 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.

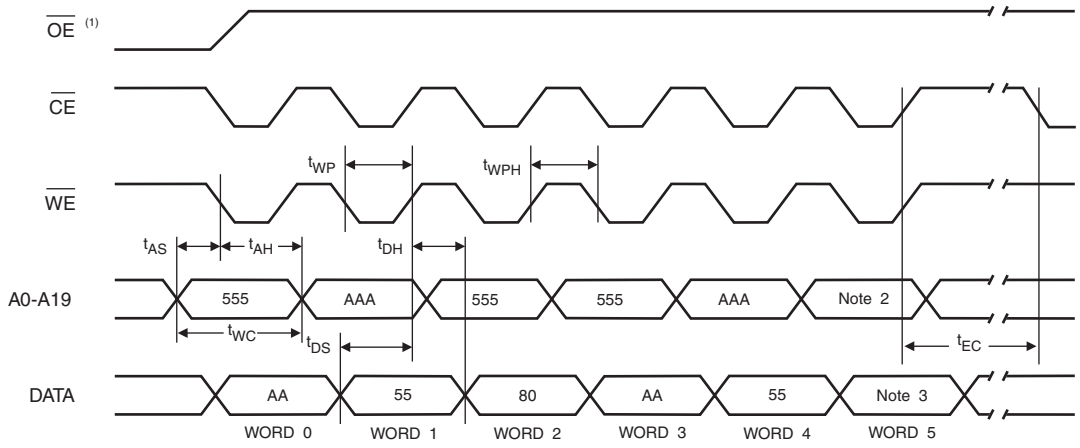
The Timer Counter channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then 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 system clock (MCK) period in order to be detected.

Whatever the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value may not read zero just after a trigger, especially when a low frequency signal is selected as the clock.

20.6.2 Sector or Chip Erase Cycle Waveforms

Figure 20-17. Sector or Chip Erase Cycle Waveforms



Notes:
1. $\overline{OE}$ must be high only when $\overline{WE}$ and $\overline{CE}$ are both low.
2. For chip erase, the address should be 555. For sector erase, the address depends on what sector is to be erased. (See footnote (3) of Table 11-2, “Command Definition Table,” on page 61.)
3. For chip erase, the data should be 10H, and for sector erase, the data should be 30H.