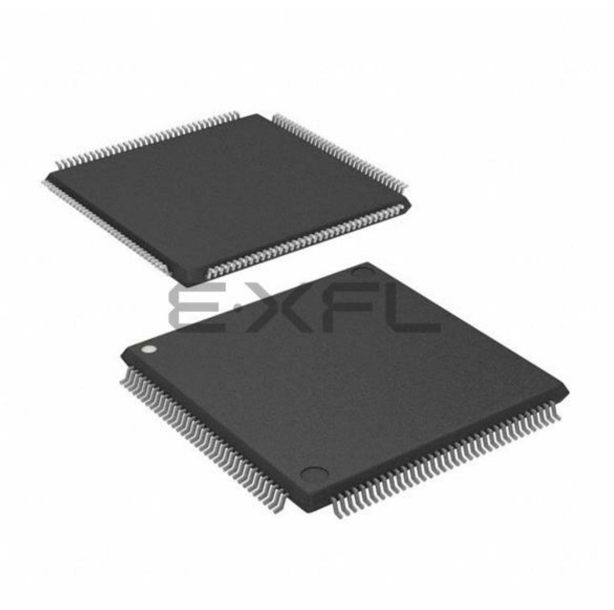


Welcome to E-XFL.COM

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.

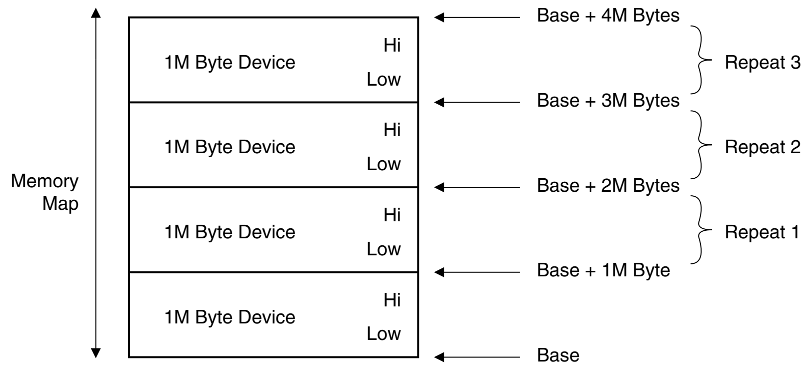
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
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-33au

Table 2. AT91M42800A Pinout in BGA 144 Package

Pin#	Name	Pin#	Name	Pin#	Name	Pin#	Name
A1	PB1/NCS3	D1	A2	G1	A17	K1	D1
A2	NCS0	D2	A3	G2	A16	K2	VDDCORE
A3	NCS1	D3	A4	G3	A11	K3	VDDIO
A4	GND	D4	NWAIT	G4	A13	K4	D9
A5	PLLRCB	D5	PA29/PME	G5	GND	K5	D10
A6	GND	D6	PA28	G6	GND	K6	D14
A7	PLLRCA	D7	TCK	G7	GND	K7	PB9/TCLK1
A8	GND	D8	TMS	G8	GND	K8	PB13/TIOA2
A9	XOUT	D9	MODE1	G9	PA9/TXD1/NTRI	K9	PB11/TIOB1
A10	XIN	D10	PA25/MCKO	G10	PA10/RXD1	K10	VDDIO
A11	MODE0	D11	PA21/NPCSB0	G11	PA8/SCK1	K11	PB16/TIOA3
A12	PA22/NPCSB1	D12	PA18/SPCKB	G12	PA7/RXD0	K12	PB23/TIOB5
B1	NUB/NWR1	E1	A7	H1	A18	L1	D3
B2	PB0/NCS2	E2	VDDIO	H2	VDDIO	L2	D2
B3	VDDCORE	E3	A6	H3	A15	L3	D5
B4	NWE/NWR0	E4	A5	H4	A14	L4	D8
B5	VDDPLL	E5	GND	H5	A19	L5	VDDIO
B6	TDO	E6	GND	H6	GND	L6	D13
B7	VDDPLL	E7	GND	H7	GND	L7	PB8/TIOB0
B8	NWDOVF	E8	NTRST	H8	GND	L8	VDDIO
B9	PA26	E9	PA13/MOSIA	H9	PA6/TXD0	L9	PB17/TIOB3
B10	PA19/MISOB	E10	PA16/NPCSA2	H10	PA4/FIQ	L10	VDDCORE
B11	PA24/NPCSB3	E11	VDDIO	H11	VDDIO	L11	PB20/TIOB4
B12	PA23/NPCSB2	E12	PA17/NPCSA3	H12	PA5/SCK0	L12	PB22/TIOA5
C1	NLB/A0	F1	A8	J1	PB5/A23/CS4	M1	D4
C2	A1	F2	A12	J2	D0	M2	D6
C3	VDDIO	F3	A9	J3	PB4/A22/CS5	M3	D7
C4	NOE/NRD	F4	A10	J4	PB3/A21/CS6	M4	D11
C5	VDDIO	F5	GND	J5	PB2/A20/CS7	M5	D12
C6	NRST	F6	GND	J6	D15	M6	PB7/TIOA0
C7	TDI	F7	GND	J7	PB6/TCLK0	M7	PB12/TCLK2
C8	VDDIO	F8	GND	J8	PB10/TIOA1	M8	PB15/TCLK3
C9	PA27/BMS	F9	PA12/MISOA	J9	PA3/IRQ3	M9	PB14/TIOB2
C10	VDDIO	F10	PA15/NPCSA1	J10	PA2/IRQ2	M10	PB18/TCLK4
C11	VDDCORE	F11	PA11/SPCKA	J11	PA0/IRQ0	M11	PB19/TIOA4
C12	PA20/MOSIB	F12	PA14/NPCSA0	J12	PA1/IRQ1	M12	PB21/TCLK5

Figure 11-1. External Memory Smaller than Page Size



11.2 Abort Status

When an abort is generated, the EBI_AASR (Abort Address Status Register) and the EBI_ASR (Abort Status Register) provide the details of the source causing the abort. Only the last abort is saved and registers are left in the last abort status. After the reset, the registers are initialized to 0.

The following are saved:

In EBI_AASR:

- The address at which the abort is generated

In EBI_ASR:

- Whether or not the processor has accessed an undefined address in the EBI address space
- Whether or not the processor required an access at a misaligned address
- The size of the access (byte, word or half-word)
- The type of the access (read, write or code fetch)

11.3 EBI Behavior During Internal Accesses

When the ARM core performs accesses in the internal memories or the embedded peripherals, the EBI signals behave as follows:

- The address lines remain at the level of the last external access.
- The data bus is tri-stated.
- The control signals remain in an inactive state.

12.10 PMC Peripheral Clock Enable Register

Register Name: PMC_PCER
Access Type: Write-only
Offset: 0x10

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

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

15	14	13	12	11	10	9	8
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2

7	6	5	4	3	2	1	0
TC1	TC0	SPIB	SPIA	US1	US0	–	–

- **Peripheral Clock Enable**

0 = No effect.

1 = Enables the peripheral clock.

12.11 PMC Peripheral Clock Disable Register

Register Name: PMC_PCDR
Access Type: Write-only
Offset: 0x14

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

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

15	14	13	12	11	10	9	8
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2

7	6	5	4	3	2	1	0
TC1	TC0	SPIB	SPIA	US1	US0	–	–

- **Peripheral Clock Disable**

0 = No effect.

1 = Disables the peripheral clock.

12.12 PMC Peripheral Clock Status Register

Register Name: PMC_PCSR
Access Type: Read-only
Offset: 0x1C
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
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2

7	6	5	4	3	2	1	0
TC1	TC0	SPIB	SPIA	US1	US0	–	–

- **Peripheral Clock Status**

0 = Peripheral clock is disabled.

1 = Peripheral clock is enabled.

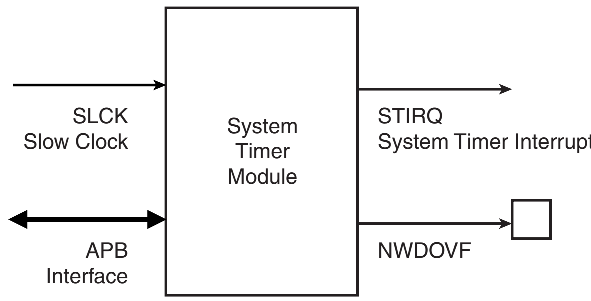
13. ST: System Timer

The System Timer module integrates three different free-running timers:

- A Period Interval Timer setting the base time for an Operating System.
- A Watchdog Timer having capabilities to reset the system in case of software deadlock.
- A Real-time Timer counting elapsed seconds.

These timers count using the Slow Clock. Typically, this clock has a frequency of 32.768 kHz.

Figure 13-1. System Timer Module

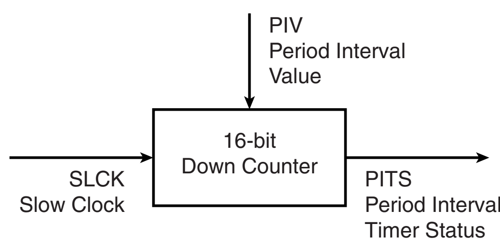


13.1 PIT: Period Interval Timer

The Period Interval Timer can be used to provide periodic interrupts for use by operating systems. It is built around a 16-bit down counter, which is preloaded by a value programmed in ST_PIMR (Period Interval Mode Register). When the PIT counter reaches 0, the bit PITS is set in ST_SR (Status Register), and an interrupt is generated, if it is enabled.

The counter is then automatically reloaded and restarted. Writing to the ST_PIMR at any time immediately reloads and restarts the down counter with the new programmed value.

Figure 13-2. Period Interval Timer



Note: If ST_PIMR is programmed with a period less or equal to the current MCK period, the update of the PITS status bit and its associated interrupt generation are unpredictable.

13.2 WDT: Watchdog Timer

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock.

It is built around a 16-bit down counter loaded with the value defined in ST_WDMR (Watchdog Mode Register). It uses the Slow Clock divided by 128. This allows the maximum watchdog period to be 256 seconds (with a typical Slow Clock of 32.768 kHz).

In normal operation, the user reloads the watchdog at regular intervals before the timer overflow occurs. This is done by writing to the ST_CR (Control Register) with the bit WDRST set.

14. AIC: Advanced Interrupt Controller

The AT91M42800A has an 8-level priority, individually maskable, vectored interrupt controller. This feature substantially reduces the software and real-time overhead in handling internal and external interrupts.

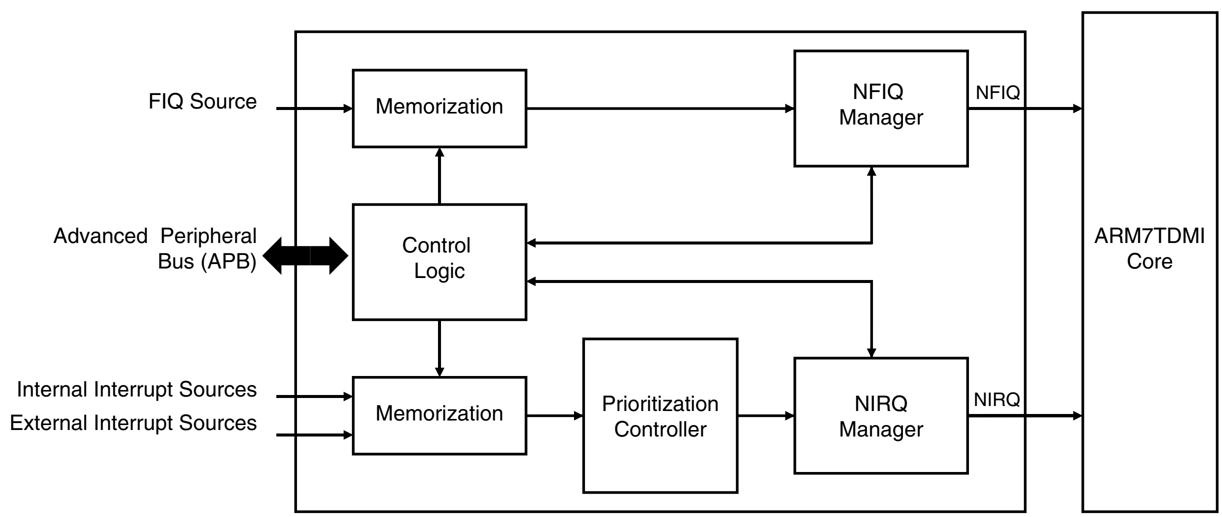
The interrupt controller is connected to the NFIQ (fast interrupt request) and the NIRQ (standard interrupt request) inputs of the ARM7TDMI processor. The processor's NFIQ line can only be asserted by the external fast interrupt request input: FIQ. The NIRQ line can be asserted by the interrupts generated by the on-chip peripherals and the external interrupt request lines: IRQ0 to IRQ3.

The 8-level priority encoder allows the customer to define the priority between the different NIRQ interrupt sources.

Internal sources are programmed to be level sensitive or edge triggered. External sources can be programmed to be positive or negative edge triggered or high- or low-level sensitive.

The interrupt sources are listed in Table 14-1 and the AIC programmable registers in Table 6.

Figure 14-1. Interrupt Controller Block Diagram



Note: After a hardware reset, the external interrupt sources pins are controlled by the Controller. They must be configured to be controlled by the peripheral before being used.

loads the program counter with the interrupt handler address stored in the AIC_IVR register. Execution is then vectored to the interrupt handler corresponding to the current interrupt.

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

The current interrupt is the interrupt with the highest priority when the Interrupt Vector Register (AIC_IVR) is read. The value read in the AIC_IVR corresponds to the address stored in the Source Vector Register (AIC_SVR) of the current interrupt. Each interrupt source has its corresponding AIC_SVR. In order to take advantage of the hardware interrupt vectoring it is necessary to store the address of each interrupt handler in the corresponding AIC_SVR, at system initialization.

14.2 Priority Controller

The NIRQ line is controlled by an 8-level priority encoder. Each source has a programmable priority level of 7 to 0. Level 7 is the highest priority and level 0 the lowest.

When the AIC receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the lowest interrupt source number (see Table 14-1) is serviced first.

The current priority level is defined as the priority level of the current interrupt at the time the register AIC_IVR is read (the interrupt which will be serviced).

In the case when a higher priority unmasked interrupt occurs while an interrupt already exists, there are two possible outcomes depending on whether the AIC_IVR has been read.

- If the NIRQ line has been asserted but the AIC_IVR has not been read, then the processor will read the new higher priority interrupt handler address in the AIC_IVR register and the current interrupt level is updated.
- If the processor has already read the AIC_IVR then the NIRQ line is reasserted. When the processor has authorized nested interrupts to occur and reads the AIC_IVR again, it reads the new, higher priority interrupt handler address. At the same time the current priority value is pushed onto a first-in last-out stack and the current priority is updated to the higher priority.

When the end of interrupt command register (AIC_EOICR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Hence at the end of a higher priority interrupt, the AIC returns to the previous state corresponding to the preceding lower priority interrupt which had been interrupted.

14.3 Interrupt Handling

The interrupt handler must read the AIC_IVR as soon as possible. This de-asserts the NIRQ request to the processor and clears the interrupt in case it is programmed to be edge triggered. This permits the AIC to assert the NIRQ line again when a higher priority unmasked interrupt occurs.

At the end of the interrupt service routine, the end of interrupt command register (AIC_EOICR) must be written. This allows pending interrupts to be serviced.

14.4 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.

15.5 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.

15.6 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.

15.7 Multi-driver (Open Drain)

Each I/O can be programmed for multi-driver option. This means that the I/O is configured as open drain (can only drive a low level) in order to support external drivers on the same pin. An external pull-up is necessary to guarantee a logic level of one when the pin is not being driven.

Registers PIO_MDER (Multi-Driver Enable) and PIO_MDDR (Multi-Driver Disable) control this option. Multi-driver can be selected whether the I/O pin is controlled by the PIO Controller or the peripheral. PIO_MDSR (Multi-Driver Status) indicates which pins are configured to support external drivers.

15.9 PIO Enable Register

Register Name: PIO_PER
Access Type: Write-only
Offset: 0x00

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 enable individual pins to be controlled by the PIO Controller instead of the associated peripheral. When the PIO is enabled, the associated peripheral (if any) is held at logic zero.

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

15.10 PIO Disable Register

Register Name: PIO_PDR
Access Type: Write-only
Offset: 0x04

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 control of individual pins. When the PIO control is disabled, the normal peripheral function is enabled on the corresponding pin.

0 = No effect.

1 = Disables PIO control (enables peripheral control) on the corresponding pin.

15.17 PIO Input Filter Status Register

Register Name: PIO_IFSR

Access Type: Read-only

Offset: 0x28

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 indicates which pins have glitch filters selected. It is updated when PIO outputs are enabled or disabled by writing to PIO_IFER or PIO_IFDR.

0 = Filter is not selected on the corresponding input.

1 = Filter is selected on the corresponding input (peripheral and PIO).

15.28 PIO Multi-drive Status Register

Register Name: PIO_MDSR

Access Type: Read-only

Offset: 0x58

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 indicates which pins are configured with open drain drivers.

0 = PIO is not configured as an open drain.

1 = PIO is configured as an open drain.

16.5 SF Protect Mode Register

Register Name: SF_PMR
Offset: 0x18
Reset Value: 0x00000000

31	30	29	28	27	26	25	24
PMRKEY							

23	22	21	20	19	18	17	16
PMRKEY							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	AIC	–	–	–	–	–

- **AIC: AIC Protect Mode Enable (Code Label `SF_AIC`)**

0 = The Advanced Interrupt Controller runs in Normal Mode.

1 = The Advanced Interrupt Controller runs in Protect Mode.

See Section 14.9 ”Protect Mode” on page 85.

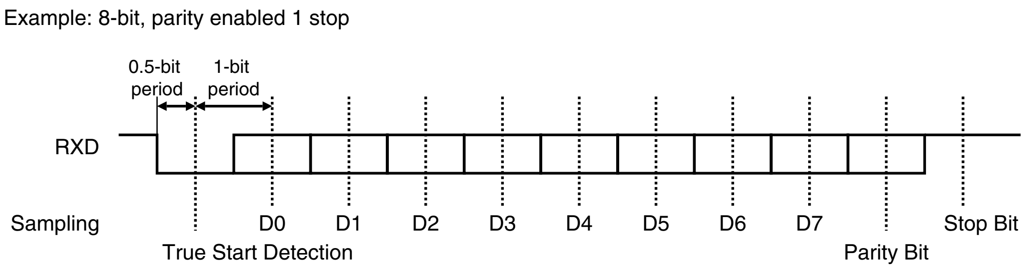
- **PMRKEY: Protect Mode Register Key**

Used only when writing SF_PMR. PMRKEY reads 0.

0x27A8: Write access in SF_PMR is allowed.

Other value: Write access in SF_PMR is prohibited.

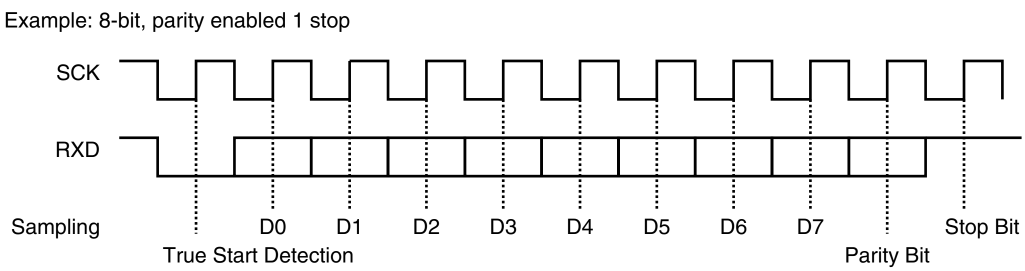
Figure 17-4. Asynchronous Mode: Character Reception



17.3.2 Synchronous Receiver

When configured for synchronous operation (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. Data bits, parity bit and stop bit are sampled and the receiver waits for the next start bit. See example in Figure 17-5.

Figure 17-5. Synchronous Mode: Character Reception



17.3.3 Receiver Ready

When a complete character is received, it is transferred to the US_RHR and the RXRDY status bit in US_CSR is set. If US_RHR has not been read since the last transfer, the OVRE status bit in US_CSR is set.

17.3.4 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 US_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in US_CSR is set.

17.3.5 Framing Error

If a character is received with a stop bit at low level and with at least one data bit at high level, a framing error is generated. This sets FRAME in US_CSR.

17.3.6 Time-out

This function allows an idle condition on the RXD line to be detected. The maximum delay for which the USART should wait for a new character to arrive while the RXD line is inactive (high level) is programmed in US_RTOR (Receiver Tim-out). When this register is set to 0, no time-out is detected. Otherwise, the receiver waits for a first character and then initializes a counter which is decremented at each bit period and reloaded at each byte reception. When the counter reaches 0, the TIMEOUT bit in US_CSR is set. The user can restart the wait for a first character with the STTTO (Start Time-out) bit in US_CR.

17.12 USART Mode Register

Name: US_MR

17.14 USART Interrupt Disable Register

Name: US_IDR

Access Type: Write-only

Offset: 0x0C

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
–	–	–	–	–	–	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Disable RXRDY Interrupt (Code Label `US_RXRDY`)**

0 = No effect.

1 = Disables RXRDY Interrupt.

- **TXRDY: Disable TXRDY Interrupt (Code Label `US_TXRDY`)**

0 = No effect.

1 = Disables TXRDY Interrupt.

- **RXBRK: Disable Receiver Break Interrupt (Code Label `US_RXBRK`)**

0 = No effect.

1 = Disables Receiver Break Interrupt.

- **ENDRX: Disable End of Receive Transfer Interrupt (Code Label `US_ENDRX`)**

0 = No effect.

1 = Disables End of Receive Transfer Interrupt.

- **ENDTX: Disable End of Transmit Transfer Interrupt (Code Label `US_ENDTX`)**

0 = No effect.

1 = Disables End of Transmit Transfer Interrupt.

- **OVRE: Disable Overrun Error Interrupt (Code Label `US_OVRE`)**

0 = No effect.

1 = Disables Overrun Error Interrupt.

- **FRAME: Disable Framing Error Interrupt (Code Label `US_FRAME`)**

0 = No effect.

1 = Disables Framing Error Interrupt.

- **PARE: Disable Parity Error Interrupt (Code Label `US_PARE`)**

0 = No effect.

1 = Disables Parity Error Interrupt.

- **TIMEOUT: Disable Time-out Interrupt (Code Label `US_TIMEOUT`)**

0 = No effect.

1 = Disables Receiver Time-out Interrupt.

- **TXEMPTY: Disable TXEMPTY Interrupt (Code Label `US_TXEMPTY`)**

0 = No effect.

1 = Disables TXEMPTY Interrupt.

- **COMMTX: Disable ARM7TDMI ICE Debug Communication Channel Transmit Interrupt**

This bit is implemented for USART0 only.

0 = No effect.

1 = Disables COMMTX Interrupt.

- **COMMRX: Disable ARM7TDMI ICE Debug Communication Channel Receive Interrupt**

This bit is implemented for USART0 only.

0 = No effect.

1 = Disables COMMRX Interrupt.

The tables below show which parameter in TC_CMR is used to define the effect of each event.

Parameter	TIOA Event
ASWTRG	Software Trigger
AEEVT	External Event
ACPC	RC Compare
ACPA	RA Compare

Parameter	TIOB Event
BSWTRG	Software Trigger
BEEVT	External Event
BCPC	RC Compare
BCPB	RB Compare

If two or more events occur at the same time, the priority level is defined as follows:

1. Software Trigger
2. External Event
3. RC Compare
4. RA or RB Compare

18.4.4 Status

The following bits in the status register are significant in Waveform mode:

- CPAS: RA Compare Status
  There has been a RA Compare match at least once since the last read of the status
- CPBS: RB Compare Status
  There has been a RB Compare match at least once since the last read of the status
- CPCS: RC Compare Status
  There has been a RC Compare match at least once since the last read of the status
- COVFS: Counter Overflow
  Counter has attempted to count past $FFFF since the last read of the status
- ETRGS: External Trigger
  External trigger has been detected since the last read of the status

18.7 TC Block Mode Register

Register Name: TC_BMR

Access Type: Read/Write

Offset: 0xC4

Reset Value: 0x0

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S		Signal Connected to XC0	Code Label: `TC_TC0XC0S`
0	0	TCLK0	`TC_TCLK0XC0`
0	1	None	`TC_NONEXC0`
1	0	TIOA1	`TC_TIOA1XC0`
1	1	TIOA2	`TC_TIOA2XC0`

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1	Code Label: `TC_TC1XC1S`
0	0	TCLK1	`TC_TCLK1XC1`
0	1	None	`TC_NONEXC1`
1	0	TIOA0	`TC_TIOA0XC1`
1	1	TIOA2	`TC_TIOA2XC1`

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2	Code Label: `TC_TC2XC2S`
0	0	TCLK2	`TC_TCLK2XC2`
0	1	None	`TC_NONEXC2`
1	0	TIOA0	`TC_TIOA0XC2`
1	1	TIOA1	`TC_TIOA1XC2`

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

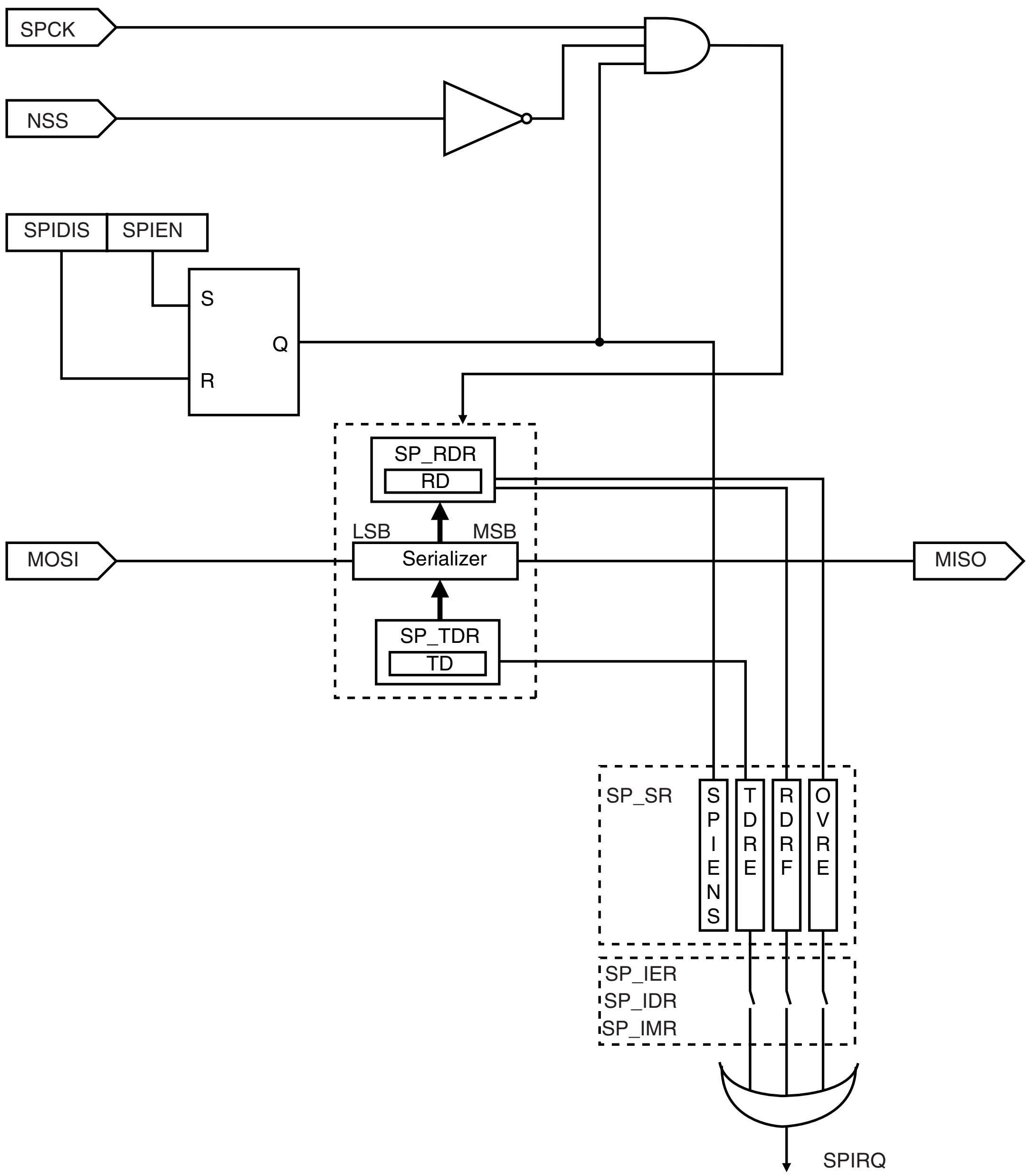


Table 20-1. Boundary-scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
85	D6	IN/OUT	OUTPUT
84			INPUT
83	D5	IN/OUT	OUTPUT
47	A11	OUTPUT	OUTPUT
46	A10	OUTPUT	OUTPUT
45	A9	OUTPUT	OUTPUT
44	A8	OUTPUT	OUTPUT
43	A[11:8]	OUTPUT	CTRL
42	A7	OUTPUT	OUTPUT
41	A6	OUTPUT	OUTPUT
40	A5	OUTPUT	OUTPUT
39	A4	OUTPUT	OUTPUT
38	A[7:4]	OUTPUT	CTRL
37	A3	OUTPUT	OUTPUT
36	A2	OUTPUT	OUTPUT
35	A1	OUTPUT	OUTPUT
34	NLB/A0	OUTPUT	OUTPUT
33	A[3:0]	OUTPUT	CTRL
32	PB1/NCS3	IN/OUT	OUTPUT
31			INPUT
30			CTRL
29	PB0/NCS2	IN/OUT	OUTPUT
28			INPUT
27			CTRL
26	NCS1	OUTPUT	OUTPUT
25	NCS0	IN/OUT	OUTPUT
24			CTRL
23	NUB/NWR1	IN/OUT	OUTPUT
22			INPUT
50	A13	OUTPUT	OUTPUT
49	A12	OUTPUT	OUTPUT
48	A[15:12]	OUTPUT	CTRL
21	NWE/NWR0	IN/OUT	OUTPUT
20			INPUT
19	NOE/NRD	IN/OUT	OUTPUT
18			INPUT
17	NOE/NRD NEW/NWR0 NUB/NWR1 NCS1	IN/OUT	CTRL
16	NWAIT	INPUT	INPUT
15	PA29/PME	IN/OUT	OUTPUT
14			INPUT
13			CTRL
12	PA28	IN/OUT	OUTPUT
11			INPUT
10			CTRL
9	NRST	INPUT	INPUT
8	PA27/BMS	IN/OUT	OUTPUT
7			INPUT
6			CTRL
5	NWDOVF	OUTPUT	OUTPUT
5	NWDOVF	OUTPUT	OUTPUT
4			CTRL
3	PA26	IN/OUT	OUTPUT
2			INPUT
1			CTRL