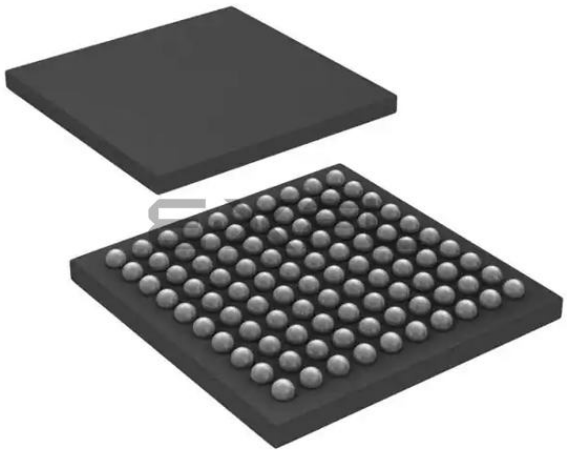


Welcome to E-XFL.COM

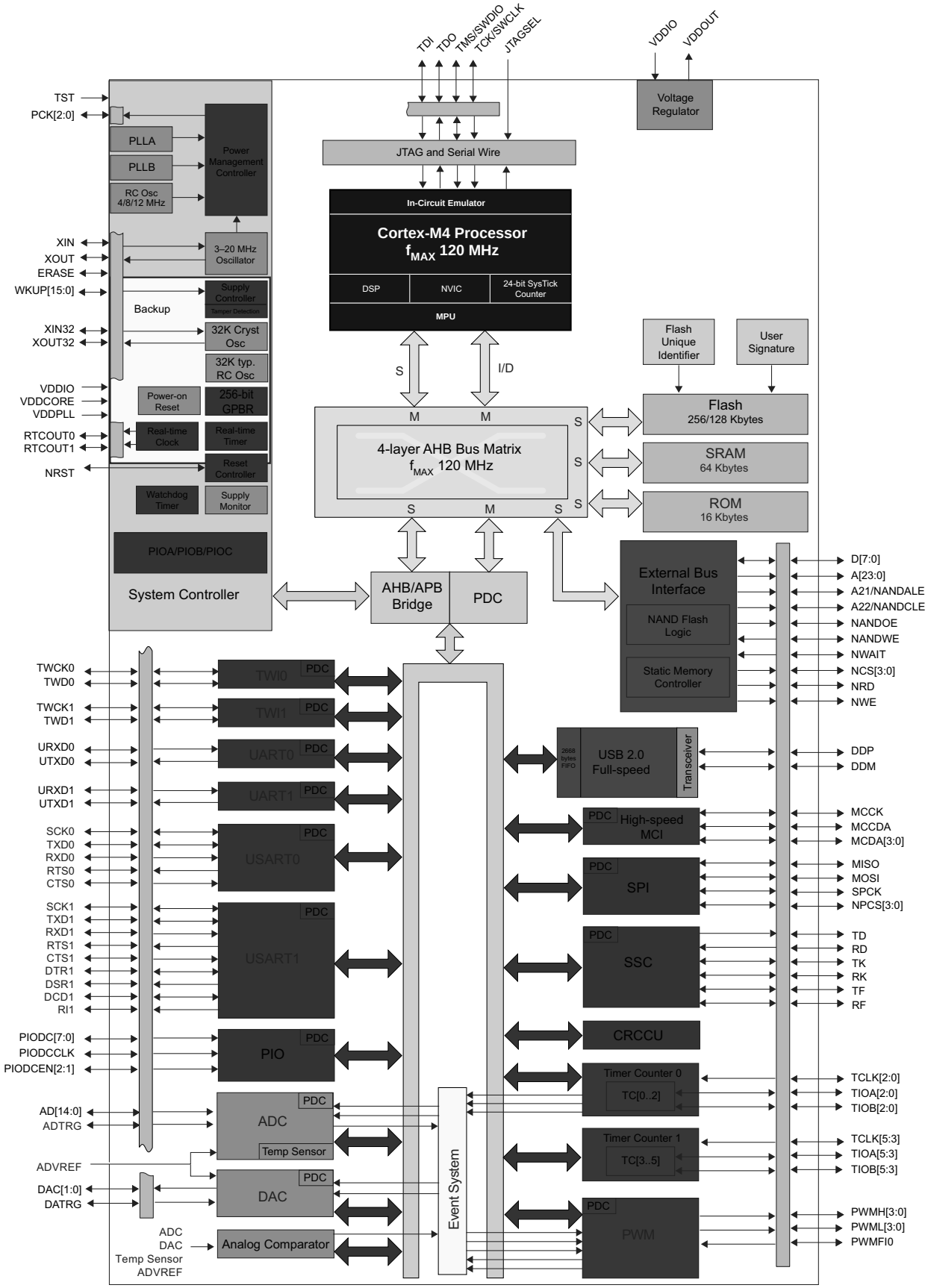
What is "Embedded - Microcontrollers"?

"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	ARM® Cortex®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	2MB (2M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	160K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-VFBGA
Supplier Device Package	100-VFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sd32cb-cfnr

Figure 2-5. SAM4S4/S2 100-pin Version Block Diagram



6. Input/Output Lines

The SAM4S has several kinds of input/output (I/O) lines such as general purpose I/Os (GPIO) and system I/Os. GPIOs can have alternate functionality due to multiplexing capabilities of the PIO controllers. The same PIO line can be used whether in I/O mode or by the multiplexed peripheral. System I/Os include pins such as test pins, oscillators, erase or analog inputs.

6.1 General Purpose I/O Lines

GPIO Lines are managed by PIO controllers. All I/Os have several input or output modes such as pull-up or pull-down, input Schmitt triggers, multi-drive (open-drain), glitch filters, debouncing or input change interrupt. Programming of these modes is performed independently for each I/O line through the PIO controller user interface. For more details, refer to Section 31. “Parallel Input/Output Controller (PIO)”.

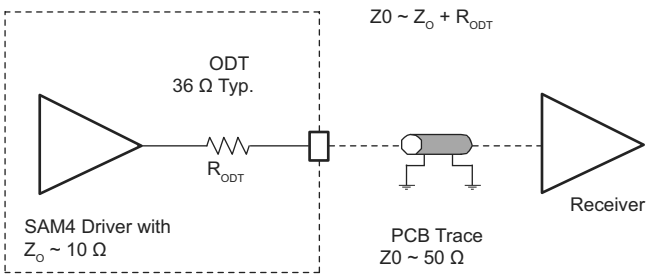
Some GPIOs can have alternate function as analog input. When the GPIO is set in analog mode, all digital features of the I/O are disabled.

The input/output buffers of the PIO lines are supplied through VDDIO power supply rail.

The SAM4S embeds high-speed pads able to handle up to 70 MHz for HSMCI (MCK/2), 70 MHz for SPI clock lines and 46 MHz on other lines. See Section 44.12 “AC Characteristics” for more details. Typical pull-up and pull-down value is 100 kΩ for all I/Os.

Each I/O line also embeds an ODT (On-Die Termination), (see Figure 6-1). It consists of an internal series resistor termination scheme for impedance matching between the driver output (SAM4S) and the PCB trace impedance preventing signal reflection. The series resistor helps to reduce IOs switching current (di/dt) thereby reducing in turn, EMI. It also decreases overshoot and undershoot (ringing) due to inductance of interconnect between devices or between boards. In conclusion ODT helps diminish signal integrity issues.

Figure 6-1. On-Die Termination



12.4.1.4 General-purpose Registers

R0–R12 are 32-bit general-purpose registers for data operations.

12.4.1.5 Stack Pointer

The *Stack Pointer* (SP) is register R13. In Thread mode, bit[1] of the Control Register indicates the stack pointer to use:

- 0 = *Main Stack Pointer* (MSP). This is the reset value.
- 1 = *Process Stack Pointer* (PSP).

On reset, the processor loads the MSP with the value from address 0x00000000.

12.4.1.6 Link Register

The *Link Register* (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.

12.4.1.7 Program Counter

The *Program Counter* (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

12.6.3.8 Instruction Width Selection

There are many instructions that can generate either a 16-bit encoding or a 32-bit encoding depending on the operands and destination register specified. For some of these instructions, the user can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 32-bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

If the user specifies an instruction width suffix and the assembler cannot generate an instruction encoding of the requested width, it generates an error.

Note: In some cases, it might be necessary to specify the .W suffix, for example if the operand is the label of an instruction or literal data, as in the case of branch instructions. This is because the assembler might not automatically generate the right size encoding.

To use an instruction width suffix, place it immediately after the instruction mnemonic and condition code, if any. The example below shows instructions with the instruction width suffix.

```
BCS.W  label      ; creates a 32-bit instruction even for a short
                  ; branch
ADDS.W R0, R0, R1 ; creates a 32-bit instruction even though the same
                  ; operation can be done by a 16-bit instruction
```

Operation

The register access operation in MSR depends on the privilege level. Unprivileged software can only access the APSR. See “Application Program Status Register” . Privileged software can access all special registers.

In unprivileged software writes to unallocated or execution state bits in the PSR are ignored.

Note: When the user writes to BASEPRI_MAX, the instruction writes to BASEPRI only if either:
Rn is non-zero and the current BASEPRI value is 0
Rn is non-zero and less than the current BASEPRI value.

See “MRS” .

Restrictions

Rn must not be SP and must not be PC.

Condition Flags

This instruction updates the flags explicitly based on the value in *Rn*.

Examples

```
MSR  CONTROL, R1 ; Read R1 value and write it to the CONTROL register
```

12.6.11.8 NOP

No Operation.

Syntax

`NOP{cond}`

where:

cond is an optional condition code, see “Conditional Execution” .

Operation

NOP does nothing. NOP is not necessarily a time-consuming NOP. The processor might remove it from the pipeline before it reaches the execution stage.

Use NOP for padding, for example to place the following instruction on a 64-bit boundary.

Condition Flags

This instruction does not change the flags.

Examples

```
NOP  ; No operation
```

12.9.1.11 System Handler Priority Register 3

Name: SCB_SHPR3

Access: Read/Write

31	30	29	28	27	26	25	24
PRI_15							
23	22	21	20	19	18	17	16
PRI_14							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PRI_15: Priority**

Priority of system handler 15, SysTick exception.

- **PRI_14: Priority**

Priority of system handler 14, PendSV.

16.6.1 RTC Control Register

Name: RTC_CR

Address: 0x400E1460

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

- **UPDTIM: Update Request Time Register**

0: No effect.

1: Stops the RTC time counting.

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

- **UPDCAL: Update Request Calendar Register**

0: No effect.

1: Stops the RTC calendar counting.

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

- **TIMEVSEL: Time Event Selection**

The event that generates the flag TIMEV in RTC_SR depends on the value of TIMEVSEL.

Value	Name	Description
0	MINUTE	Minute change
1	HOUR	Hour change
2	MIDNIGHT	Every day at midnight
3	NOON	Every day at noon

16.6.6 RTC Calendar Alarm Register

Name: RTC_CALALR

Address: 0x400E1474

Access: Read/Write

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

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

Note: To change one of the DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_CALALR. The first access clears the enable corresponding to the field to change (DATEEN, MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (DATE, MONTH). The third access is required to re-enable the field by writing 1 in DATEEN, MTHEN fields.

- **MONTH: Month Alarm**

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

- **MTHEN: Month Alarm Enable**

0: The month-matching alarm is disabled.

1: The month-matching alarm is enabled.

- **DATE: Date Alarm**

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

- **DATEEN: Date Alarm Enable**

0: The date-matching alarm is disabled.

1: The date-matching alarm is enabled.

17.4 Functional Description

The Watchdog Timer is 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 can either disable the WDT by setting bit WDT_MR.WDDIS or reprogram the WDT to meet the maximum watchdog period the application requires.

If the watchdog is restarted by writing into the Control Register (WDT_CR), WDT_MR must not be programmed during a period of time of three slow clock periods following the WDT_CR write access. In any case, programming a new value in WDT_MR automatically initiates a restart instruction.

WDT_MR can be written only once. Only a processor reset resets it. Writing WDT_MR reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the watchdog at regular intervals before the timer underflow occurs, by setting bit WDT_CR.WDRSTT. The watchdog counter is then immediately reloaded from WDT_MR and restarted, and the slow clock 128 divider is reset and restarted. 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 bit WDT_MR.WDRSTEN is set. Moreover, the bit WDUNF is set in the 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 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 WDT_SR.WDERR is updated 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).

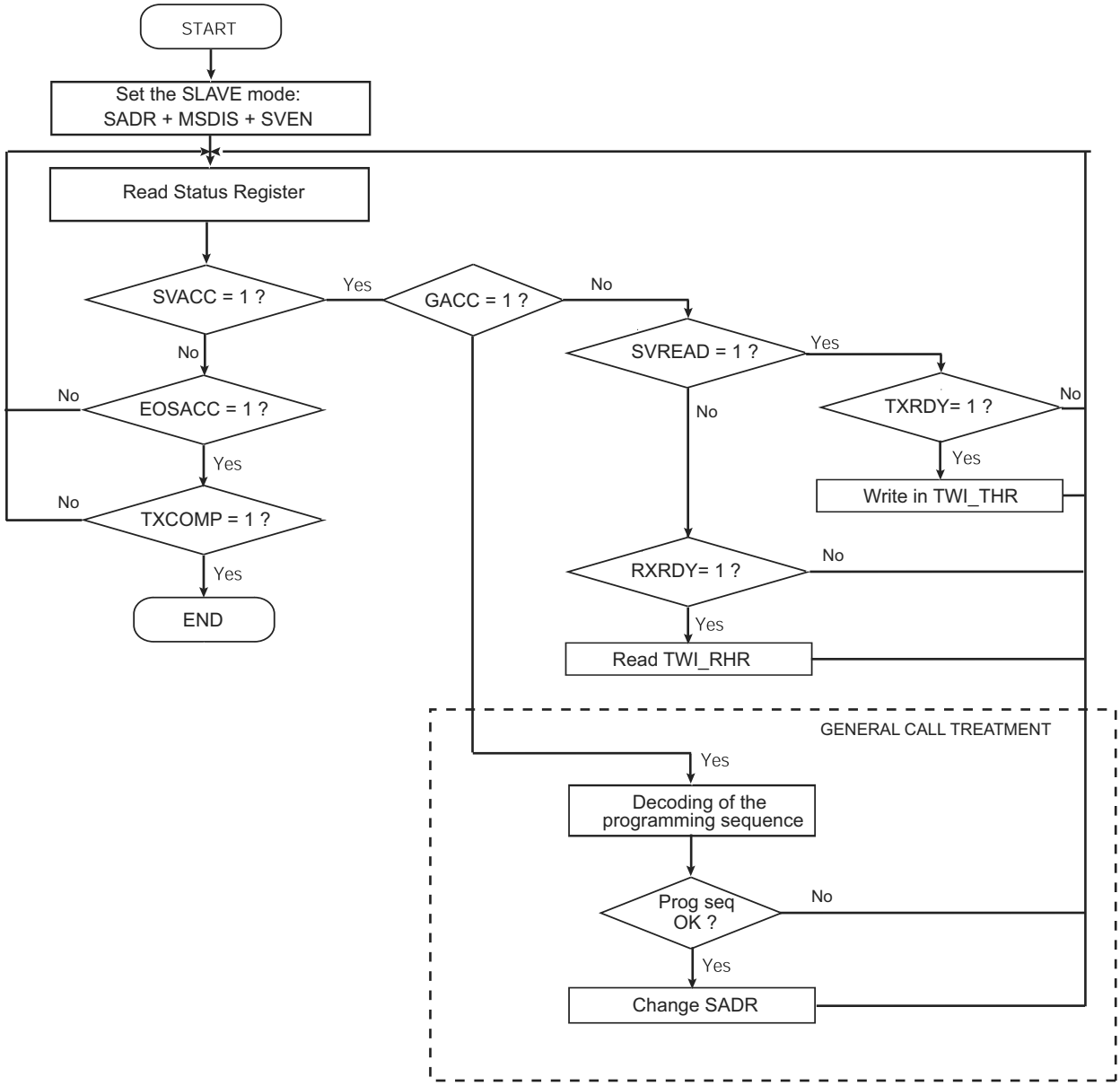
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDT_MR.WDFIEN is set. The signal “wdt_fault” to the Reset Controller causes a watchdog reset if the WDRSTEN bit is set as already explained in the Reset Controller documentation. In this case, 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 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 WDT_MR.

Figure 34-31. Read Write Flowchart in Slave Mode



- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Clear TIMEOUT Flag and Start Time-out After Next Character Received**

0: No effect.

1: Starts waiting for a character before enabling the time-out counter. Immediately disables a time-out period in progress. 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 ITER 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: Start Time-out Immediately**

0: No effect

1: Immediately restarts time-out period.

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR to 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Pin Control**

0: No effect.

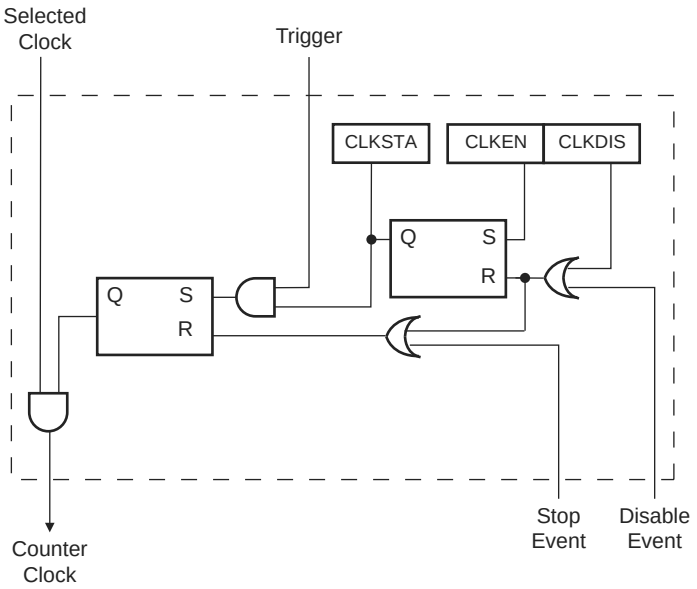
1: Drives RTS pin to 0 if US_MR.USART_MODE field = 0.

- **RTSDIS: Request to Send Pin Control**

0: No effect.

1: Drives RTS pin to 1 if US_MR.USART_MODE field = 0.

Figure 37-4. Clock Control



37.6.5 Operating Modes

Each channel can operate independently 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_CMR.

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.

37.6.6 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.

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.

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 the 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 bit ENETRG in the TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the peripheral clock period in order to be detected.

38.8.2 Data Transfer Operation

The High Speed MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kinds of transfer can be selected setting the Transfer Type (TRTYP) field in the HSMCI Command Register (HSMCI_CMDR).

These operations can be done using the features of the Peripheral DMA Controller (PDC). If the PDCMODE bit is set in HSMCI_MR, then all reads and writes use the PDC facilities.

In all cases, the block length (BLKLEN field) must be defined either in the HSMCI Mode Register (HSMCI_MR) or in the HSMCI Block Register (HSMCI_BLKR). This field determines the size of the data block.

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 predefined 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 predefined block count, the host must correctly program the HSMCI Block Register (HSMCI_BLKR). Otherwise the card will start an open-ended multiple block read. The BCNT field of the HSMCI_BLKR 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.

38.8.3 Read Operation

The following flowchart (Figure 38-8) shows how to read a single block with or without use of PDC facilities. In this example, a polling method is used to wait for the end of read. Similarly, the user can configure the HSMCI Interrupt Enable Register (HSMCI_IER) to trigger an interrupt at the end of read.

38.14.14HSMCI Interrupt Disable Register

Name: HSMCI_IDR

Address: 0x40000048

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	–	–
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

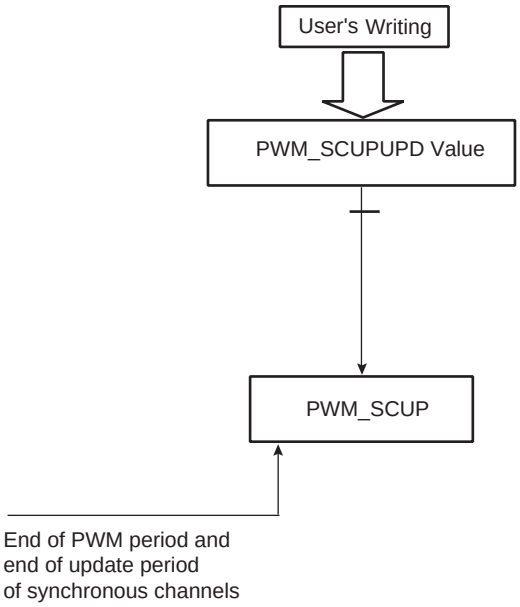
The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **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**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Disable**
- **CSRCV: Completion Signal received 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**

Figure 39-19. Synchronized Update of Update Period Value of Synchronous Channels



39.6.5.5 Changing the Comparison Value and the Comparison Configuration

It is possible to change the comparison values and the comparison configurations while the channel 0 is enabled (see Section 39.6.3 “PWM Comparison Units”).

To prevent unexpected comparison match, the user must use the PWM Comparison x Value Update Register (PWM_CMPVUPDx) and the PWM Comparison x Mode Update Register (PWM_CMPMUPDx) to change, respectively, the comparison values and the comparison configurations while the channel 0 is still enabled. These registers hold the new values until the end of the comparison update period (when CUPRCNT is equal to CUPR in PWM Comparison x Mode Register (PWM_CMPMx) and the end of the current PWM period, then update the values for the next period.

CAUTION: The write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

Note: If the update registers PWM_CMPVUPDx and PWM_CMPMUPDx are written several times between two updates, only the last written value are taken into account.

39.7.33 PWM Comparison x Value Update Register

Name: PWM_CMPVUPDx

Address: 0x40020134 [0], 0x40020144 [1], 0x40020154 [2], 0x40020164 [3], 0x40020174 [4], 0x40020184 [5], 0x40020194 [6], 0x400201A4 [7]

Access: Write-only

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

23	22	21	20	19	18	17	16
CVUPD							

15	14	13	12	11	10	9	8
CVUPD							

7	6	5	4	3	2	1	0
CVUPD							

This register acts as a double buffer for the CV and CVM values. This prevents an unexpected comparison x match.

Only the first 16 bits (channel counter size) of field CVUPD are significant.

- **CVUPD: Comparison x Value Update**

Define the comparison x value to be compared with the counter of the channel 0.

- **CVMUPD: Comparison x Value Mode Update**

0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.

1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is not relevant if the counter of the channel 0 is left-aligned (CALG = 0 in PWM Channel Mode Register)

CAUTION: The write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

41.7.1 ACC Control Register

Name: ACC_CR

Address: 0x40040000

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

- **SWRST: Software Reset**

0: No effect.

1: Resets the module.

Table 44-59. Dynamic Performance Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
SNR	Signal to Noise Ratio	DAC Clock (f_{DAC}) = 50 MHz, f_S = 2 MHz, f_{IN} = 241 kHz, ADC_ACR.IBCTL = 01, FFT using 1024 points or more, Frequency band = [10 kHz–1 MHz] Nyquist conditions fulfilled	2.4V < V_{DDIN} < 2.7V	50	62	70	dB
			2.7V < V_{DDIN} < 3.6V	62	70	74	
THD	Total Harmonic Distortion		2.4V < V_{DDIN} < 2.7V	-78	-64	-60	dB
			2.7V < V_{DDIN} < 3.6V	-80	-74	-72	
SINAD	Signal to Noise and Distortion		2.4V < V_{DDIN} < 2.7V	50	60	70	dB
			2.7V < V_{DDIN} < 3.6V	62	68	73	
ENOB	Effective Number of Bits		2.4V < V_{DDIN} < 2.7V	8	10	12	bit
			2.7V < V_{DDIN} < 3.6V	10	11	12	

Table 44-60. Analog Outputs

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OR}	Voltage Range		(1/6) × V_{ADVREF}	–	(5/6) × V_{ADVREF}	V
SR	Slew Rate	Channel output current versus slew rate (IBCTL for DAC0 or DAC1, noted IBCTLCHx) R_{LOAD} = 10 kΩ, 0 pF < C_{LOAD}< 50 pF IBCTLCHx = 00 IBCTLCHx = 01 IBCTLCHx = 10 IBCTLCHx = 11	–	 2.7 5.3 8 10.7	–	V/µs
	Output Channel Current Consumption	No resistive load IBCTLCHx = 00 IBCTLCHx = 01 IBCTLCHx = 10 IBCTLCHx = 11	–	 0.23 0.45 0.67 0.89	–	mA
t_{sa}	Settling Time	R_{LOAD} = 10 kΩ, 0 pF < C_{LOAD}< 50 pF	–	–	0.5	µs
R_{LOAD}	Output load resistor		10	–	–	kΩ
C_{LOAD}	Output load capacitor		–	30	50	pF

45.3 100-ball VFBGA Mechanical Characteristics

Figure 45-3. 100-ball VFBGA Package Drawing

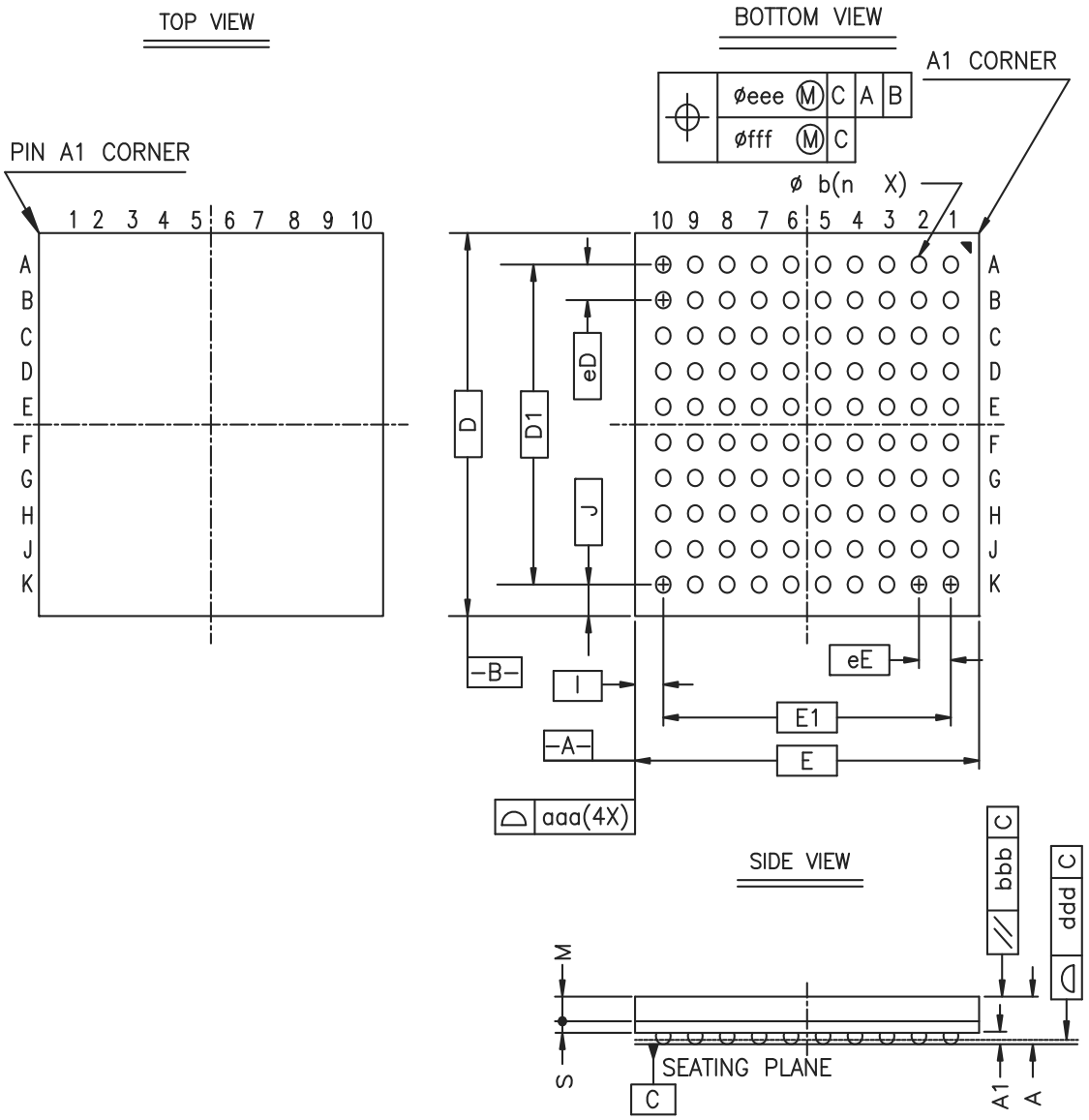


Table 45-8. VFBGA Package Dimensions

		Symbol	Common Dimensions (mm)
Package:			VFBGA
Body Size:	X	E	7.000 ± 0.100
	Y	D	7.000 ± 0.100
Ball Pitch:	X	eE	0.650
	Y	eD	0.650
Total Thickness:		A	1.000 max
Mold Thickness:		M	0.450 ref.
Substrate Thickness:		S	0.210 ref.

Table of Contents