Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

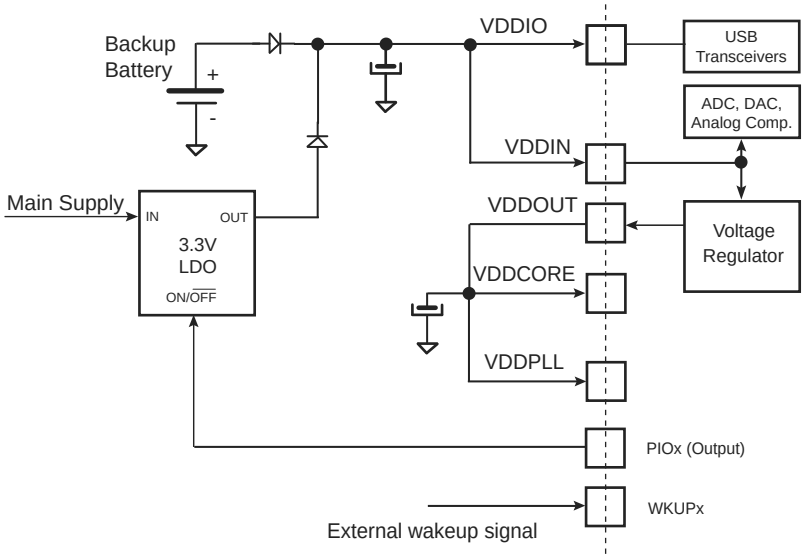
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	47
Program Memory Size	1MB (1M 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 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sa16ba-au

Figure 5-4 provides an example of the powering scheme when using a backup battery. Since the PIO state is preserved when in Backup mode, any free PIO line can be used to switch off the external regulator by driving the PIO line at low level (PIO is input, pull-up enabled after backup reset). External wake-up of the system can be from a push button or any signal. See Section 5.7 “Wake-up Sources” for further details.

Figure 5-4. Backup Battery



Note: The two diodes provide a “switchover circuit” (for illustration purpose) between the backup battery and the main supply when the system is put in backup mode.

Note: Restrictions:
For USB, VDDIO needs to be greater than 3.0V.
For ADC, DAC and Analog Comparator, VDDIN needs to be greater than 2.4V.

5.5 Active Mode

Active mode is the normal running mode with the core clock running from the fast RC oscillator, the main crystal oscillator or the PLLA. The Power Management Controller can be used to adapt the frequency and to disable the peripheral clocks.

5.6 Low-power Modes

The SAM4S has the following low-power modes: Backup mode, Wait mode and Sleep mode.

Note: The Wait For Event instruction (WFE) of the Cortex-M4 core can be used to enter any of the low-power modes, however, this may add complexity in the design of application state machines. This is due to the fact that the WFE instruction goes along with an event flag of the Cortex core (cannot be managed by the software application). The event flag can be set by interrupts, a debug event or an event signal from another processor. Since it is possible for an interrupt to occur just before the execution of WFE, WFE takes into account events that happened in the past. As a result, WFE prevents the device from entering Wait mode if an interrupt event has occurred.
Atmel has made provision to avoid using the WFE instruction. The workarounds to ease application design are as follows:
- For Backup mode, switch off the voltage regulator and configure the VROFF bit in the Supply Controller Control Register (SUPC_CR).
- For Wait mode, configure the WAITMODE bit in the PMC Clock Generator Main Oscillator Register of the Power Management Controller (PMC)
- For Sleep mode, use the Wait for Interrupt (WFI) instruction.

Complete information is available in Table 5-1 “Low-power Mode Configuration Summary".

12.4.1.17 Exceptions and Interrupts

The Cortex-M4 processor supports interrupts and system exceptions. The processor and the *Nested Vectored Interrupt Controller* (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses the Handler mode to handle all exceptions except for reset. See “Exception Entry” and “Exception Return” for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” for more information.

12.4.1.18 Data Types

The processor supports the following data types:

- 32-bit words
- 16-bit halfwords
- 8-bit bytes
- The processor manages all data memory accesses as little-endian. Instruction memory and *Private Peripheral Bus* (PPB) accesses are always little-endian. See “Memory Regions, Types and Attributes” for more information.

12.4.1.19 Cortex Microcontroller Software Interface Standard (CMSIS)

For a Cortex-M4 microcontroller system, the *Cortex Microcontroller Software Interface Standard* (CMSIS) defines:

- A common way to:
  - Access peripheral registers
  - Define exception vectors
- The names of:
  - The registers of the core peripherals
  - The core exception vectors
- A device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M4 processor.

The CMSIS simplifies the software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note: This document uses the register short names defined by the CMSIS. In a few cases, these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Section 12.5.3 ”Power Management Programming Hints”
- Section 12.6.2 ”CMSIS Functions”
- Section 12.8.2.1 ”NVIC Programming Hints”.

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” .

12.4.3.3 Exception Handlers

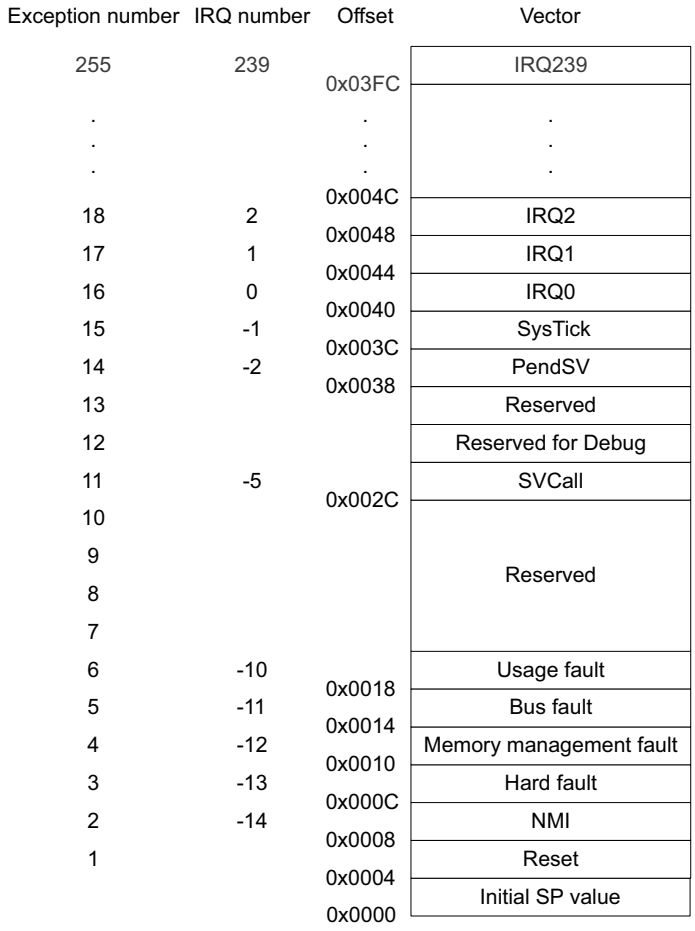
The processor handles exceptions using:

- Interrupt Service Routines (ISRs)
  Interrupts IRQ0 to IRQ34 are the exceptions handled by ISRs.
- Fault Handlers
  Hard fault, memory management fault, usage fault, bus fault are fault exceptions handled by the fault handlers.
- System Handlers
  NMI, PendSV, SVCall SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

12.4.3.4 Vector Table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 12-6 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code.

Figure 12-6. Vector Table



On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the SCB_VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFFF80, see “Vector Table Offset Register” .

If S is specified:

- These instructions update the N and Z flags according to the result
- The C flag is updated to the last bit shifted out, except when the shift length is 0, see “Shift Operations” .

Examples

```
ASR    R7, R8, #9  ; Arithmetic shift right by 9 bits
SLS    R1, R2, #3  ; Logical shift left by 3 bits with flag update
LSR    R4, R5, #6  ; Logical shift right by 6 bits
ROR    R4, R5, R6  ; Rotate right by the value in the bottom byte of R6
RRX    R4, R5      ; Rotate right with extend.
```

12.6.5.4 CLZ

Count Leading Zeros.

Syntax

`CLZ{cond} Rd, Rm`

where:

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

Rd is the destination register.

Rm is the operand register.

Operation

The CLZ instruction counts the number of leading zeros in the value in *Rm* and returns the result in *Rd*. The result value is 32 if no bits are set and zero if bit[31] is set.

Restrictions

Do not use SP and do not use PC.

Condition Flags

This instruction does not change the flags.

Examples

```
CLZ     R4,R9
CLZNE   R2,R3
```

```
op{XY}{cond} RdLo, RdHi, Rn, Rm
op{X}{cond} RdLo, RdHi, Rn, Rm
```

where:

op — is one of:

MLAL Signed Multiply Accumulate Long.

SMLAL Signed Multiply Accumulate Long (halfwords, X and Y).

X and Y specify which halfword of the source registers *Rn* and *Rm* are used as the first and second multiply operand:

If *X* is B, then the bottom halfword, bits [15:0], of *Rn* is used.
If *X* is T, then the top halfword, bits [31:16], of *Rn* is used.

If *Y* is B, then the bottom halfword, bits [15:0], of *Rm* is used.
If *Y* is T, then the top halfword, bits [31:16], of *Rm* is used.

SMLALD Signed Multiply Accumulate Long Dual.

SMLALDX Signed Multiply Accumulate Long Dual Reversed.

If the *X* is omitted, the multiplications are bottom × bottom and top × top.

If *X* is present, the multiplications are bottom × top and top × bottom.

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

RdHi, RdLo — are the destination registers.
RdLo is the lower 32 bits and *RdHi* is the upper 32 bits of the 64-bit integer.
For SMLAL, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD and SMLALDX, they also hold the accumulating value.

Rn, Rm — are registers holding the first and second operands.

Operation

The SMLAL instruction:

- Multiplies the two’s complement signed word values from *Rn* and *Rm*.
- Adds the 64-bit value in *RdLo* and *RdHi* to the resulting 64-bit product.
- Writes the 64-bit result of the multiplication and addition in *RdLo* and *RdHi*.

The SMLALBB, SMLALBT, SMLALTB and SMLALTT instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from *Rn* and *Rm*.
- Adds the resulting sign-extended 32-bit product to the 64-bit value in *RdLo* and *RdHi*.
- Writes the 64-bit result of the multiplication and addition in *RdLo* and *RdHi*.

The non-specified halfwords of the source registers are ignored.

The SMLALD and SMLALDX instructions interpret the values from *Rn* and *Rm* as four halfword two’s complement signed 16-bit integers. These instructions:

- If *X* is not present, multiply the top signed halfword value of *Rn* with the top signed halfword of *Rm* and the bottom signed halfword values of *Rn* with the bottom signed halfword of *Rm*.
- Or if *X* is present, multiply the top signed halfword value of *Rn* with the bottom signed halfword of *Rm* and the bottom signed halfword values of *Rn* with the top signed halfword of *Rm*.
- Add the two multiplication results to the signed 64-bit value in *RdLo* and *RdHi* to create the resulting 64-bit product.
- Write the 64-bit product in *RdLo* and *RdHi*.

Restrictions

In these instructions:

12.10.1.1 SysTick Control and Status Register

Name: SYST_CSR

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	CLKSOURCE	TICKINT	ENABLE

The SysTick SYST_CSR enables the SysTick features.

- **COUNTFLAG: Count Flag**

Returns 1 if the timer counted to 0 since the last time this was read.

- **CLKSOURCE: Clock Source**

Indicates the clock source:

0: External Clock.

1: Processor Clock.

- **TICKINT: SysTick Exception Request Enable**

Enables a SysTick exception request:

0: Counting down to zero does not assert the SysTick exception request.

1: Counting down to zero asserts the SysTick exception request.

The software can use COUNTFLAG to determine if SysTick has ever counted to zero.

- **ENABLE: Counter Enable**

Enables the counter:

0: Counter disabled.

1: Counter enabled.

When ENABLE is set to 1, the counter loads the RELOAD value from the SYST_RVR and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

20. Enhanced Embedded Flash Controller (EEFC)

20.1 Description

The Enhanced Embedded Flash Controller (EEFC) provides the interface of the Flash block with the 32-bit internal bus.

Its 128-bit or 64-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

20.2 Embedded Characteristics

- Increases Performance in Thumb-2 Mode with 128-bit or 64-bit-wide Memory Interface up to 120 MHz
- Code Loop Optimization
- 256 Lock Bits, Each Protecting a Lock Region
- GPNVMx General-purpose GPNVM Bits
- One-by-one Lock Bit Programming
- Commands Protected by a Keyword
- Erase the Entire Flash
- Erase by Plane
- Erase by Sector
- Erase by Page
- Provides Unique Identifier
- Provides 512-byte User Signature Area
- Supports Erasing before Programming
- Locking and Unlocking Operations
- Supports Read of the Calibration Bits

20.3 Product Dependencies

20.3.1 Power Management

The Enhanced Embedded Flash Controller (EEFC) is continuously clocked. The Power Management Controller has no effect on its behavior.

20.3.2 Interrupt Sources

The EEFC interrupt line is connected to the interrupt controller. Using the EEFC interrupt requires the interrupt controller to be programmed first. The EEFC interrupt is generated only if the value of bit EEFC_FMR.FRDY is 1.

Table 20-1. Peripheral IDs

Instance	ID
EFC0	6
EFC1	7

24.4 Device Initialization

Initialization follows the steps described below:

1. Stack setup
2. Set up the Embedded Flash Controller
3. External Clock detection (crystal or external clock on XIN)
4. If external crystal or clock with supported frequency, allow USB activation
5. Else, does not allow USB activation and use internal 12 MHz RC oscillator
6. Main oscillator frequency detection if no external clock detected
7. Switch Master Clock on Main Oscillator
8. C variable initialization
9. PLLA setup: PLLA is initialized to generate a 48 MHz clock
10. Disable the Watchdog
11. Initialization of UART0 (115200 bauds, 8, N, 1)
12. Initialization of the USB Device Port (in case USB activation allowed)
13. Wait for one of the following events
    1. Check if USB device enumeration has occurred
    2. Check if characters have been received in UART0
14. Jump to SAM-BA Monitor (see Section 24.5 ”SAM-BA Monitor”)

25.8.7 Write Protection Status Register

Name: MATRIX_WPSR

Address: 0x400E03E8

Access: Read-only

Reset: See Table 25-4

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the MATRIX_WPSR.

1: A write protection violation has occurred since the last read of the MATRIX_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Prior to instructing the device to enter Wait mode:

1. Select the fast RC oscillator as the master clock source (the CSS field in PMC_MCKR must be written to 1).
2. Disable the PLL if enabled.
3. Clear the internal wake-up sources.

The system enters Wait mode either by setting the WAITMODE bit in CKGR_MOR, or by executing the WaitForEvent (WFE) instruction of the processor while the LPM bit is at 1 in PMC_FSMR. Immediately after setting the WAITMODE bit or using the WFE instruction, wait for the MCKRDY bit to be set in PMC_SR.

A fast startup is enabled upon the detection of a programmed level on one of the 16 wake-up inputs (WKUP) or upon an active alarm from the RTC, RTT and USB Controller. The polarity of the 16 wake-up inputs is programmable by writing the PMC Fast Startup Polarity Register (PMC_FSPR).

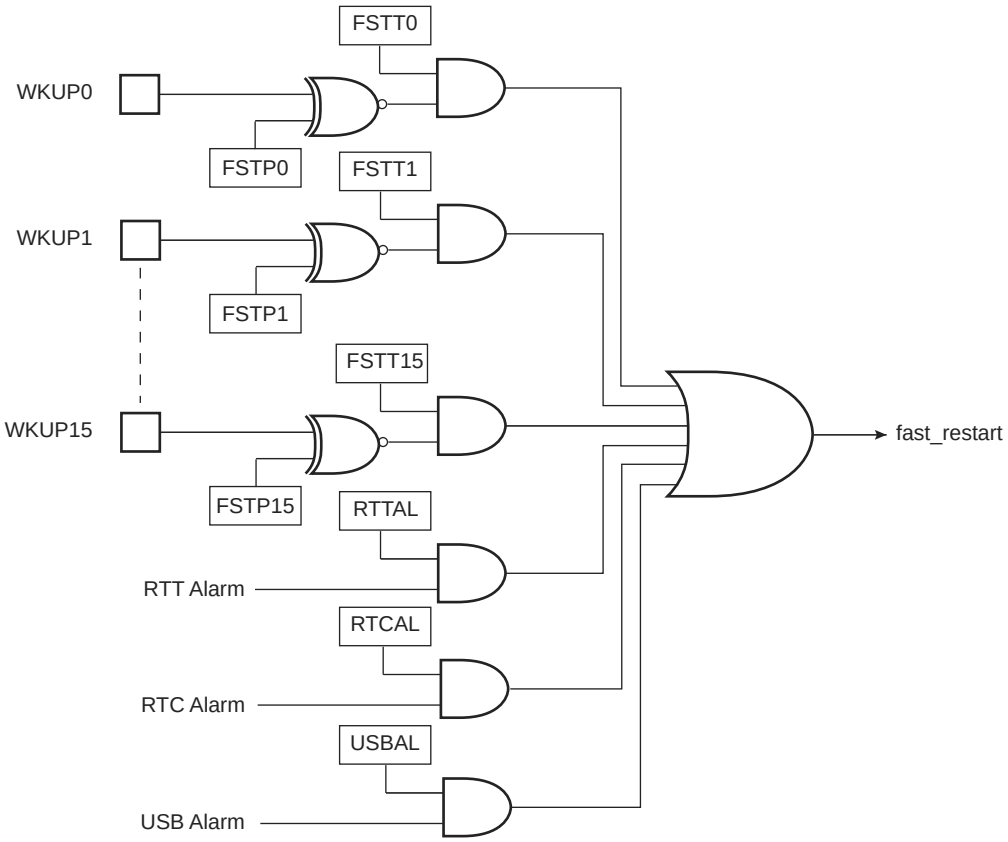
The fast startup circuitry, as shown in Figure 29-4, is fully asynchronous and provides a fast startup signal to the PMC. As soon as the fast startup signal is asserted, the embedded 4/8/12 MHz fast RC oscillator restarts automatically.

When entering Wait mode, the embedded Flash can be placed in one of the Low-power modes (Deep-power-down or Standby modes) depending on the configuration of the FLPM field in the PMC_FSMR. The FLPM field can be programmed at anytime and its value will be applied to the next Wait mode period.

The power consumption reduction is optimal when configuring 1 (Deep-power-down mode) in field FLPM. If 0 is programmed (Standby mode), the power consumption is slightly higher than in Deep-power-down mode.

When programming 2 in field FLPM, the Wait mode Flash power consumption is equivalent to that of the Active mode when there is no read access on the Flash.

Figure 29-4. Fast Startup Circuitry



automatically forces the fast RC oscillator to be the source clock for MAINCK. If the fast RC oscillator is disabled when a clock failure detection occurs, it is automatically re-enabled by the clock failure detection mechanism.

It takes two slow RC oscillator clock cycles to detect and switch from the main oscillator, to the fast RC oscillator if the source master clock (MCK) is main clock (MAINCK), or three slow clock RC oscillator cycles if the source of MCK is PLLACKor PLLBCK.

A clock failure detection activates a fault output that is connected to the Pulse Width Modulator (PWM) Controller. With this connection, the PWM controller is able to force its outputs and to protect the driven device, if a clock failure is detected.

The user can know the status of the clock failure detector at any time by reading the FOS bit in PMC_SR.

This fault output remains active until the defect is detected and until it is cleared by the bit FOCLR in the PMC Fault Output Clear Register (PMC_FOCR).

29.17.11PMC Master Clock Register

Name: PMC_MCKR

Address: 0x400E0430

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	PLLBDIV2	PLLADIV2	–	–	–	–

7	6	5	4	3	2	1	0
–	PRES			–	–	CSS	

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **CSS: Master Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLA Clock is selected
3	PLLB_CLK	PLLBClock is selected

- **PRES: Processor Clock Prescaler**

Value	Name	Description
0	CLK_1	Selected clock
1	CLK_2	Selected clock divided by 2
2	CLK_4	Selected clock divided by 4
3	CLK_8	Selected clock divided by 8
4	CLK_16	Selected clock divided by 16
5	CLK_32	Selected clock divided by 32
6	CLK_64	Selected clock divided by 64
7	CLK_3	Selected clock divided by 3

- **PLLADIV2: PLLA Divisor by 2**

PLLADIV2	PLLA Clock Division
0	PLLA clock frequency is divided by 1.
1	PLLA clock frequency is divided by 2.

35.6.7 UART Receiver Holding Register

Name: UART_RHR

Address: 0x400E0618 (0), 0x400E0818 (1)

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

36.7.9 USART Interrupt Mask Register

Name: US_IMR

Address: 0x40024010 (0), 0x40028010 (1)

Access: Read-only

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

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITER	TXEMPTY	TIMEOUT

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

For SPI specific configuration, see Section 36.7.10 ”USART Interrupt Mask Register (SPI_MODE)”.

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

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask (available in all USART modes of operation)**
- **ENDTX: End of Transmit Buffer Interrupt Mask (available in all USART modes of operation)**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITER: Max Number of Repetitions Reached Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask (available in all USART modes of operation)**
- **RXBUFF: Receive Buffer Full Interrupt Mask (available in all USART modes of operation)**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**

37.6.13 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

37.6.14 Quadrature Decoder

37.6.14.1 Description

The quadrature decoder (QDEC) is driven by TIOA0, TIOB0, TIOB1 input pins and drives the timer/counter of channel 0 and 1. Channel 2 can be used as a time base in case of speed measurement requirements (refer to Figure 37-15).

When writing a 0 to bit QDEN of the TC_BMR, the QDEC is bypassed and the IO pins are directly routed to the timer counter function. See

TIOA0 and TIOB0 are to be driven by the two dedicated quadrature signals from a rotary sensor mounted on the shaft of the off-chip motor.

A third signal from the rotary sensor can be processed through pin TIOB1 and is typically dedicated to be driven by an index signal if it is provided by the sensor. This signal is not required to decode the quadrature signals PHA, PHB.

Field TCCLKS of TC_CMRx must be configured to select XC0 input (i.e., 0x101). Field TC0XC0S has no effect as soon as the QDEC is enabled.

Either speed or position/revolution can be measured. Position channel 0 accumulates the edges of PHA, PHB input signals giving a high accuracy on motor position whereas channel 1 accumulates the index pulses of the sensor, therefore the number of rotations. Concatenation of both values provides a high level of precision on motion system position.

In Speed mode, position cannot be measured but revolution can be measured.

Inputs from the rotary sensor can be filtered prior to down-stream processing. Accommodation of input polarity, phase definition and other factors are configurable.

Interruptions can be generated on different events.

A compare function (using TC_RC) is available on channel 0 (speed/position) or channel 1 (rotation) and can generate an interrupt by means of the CPCS flag in the TC_SRx.

38.14.19HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx [x=0..255]

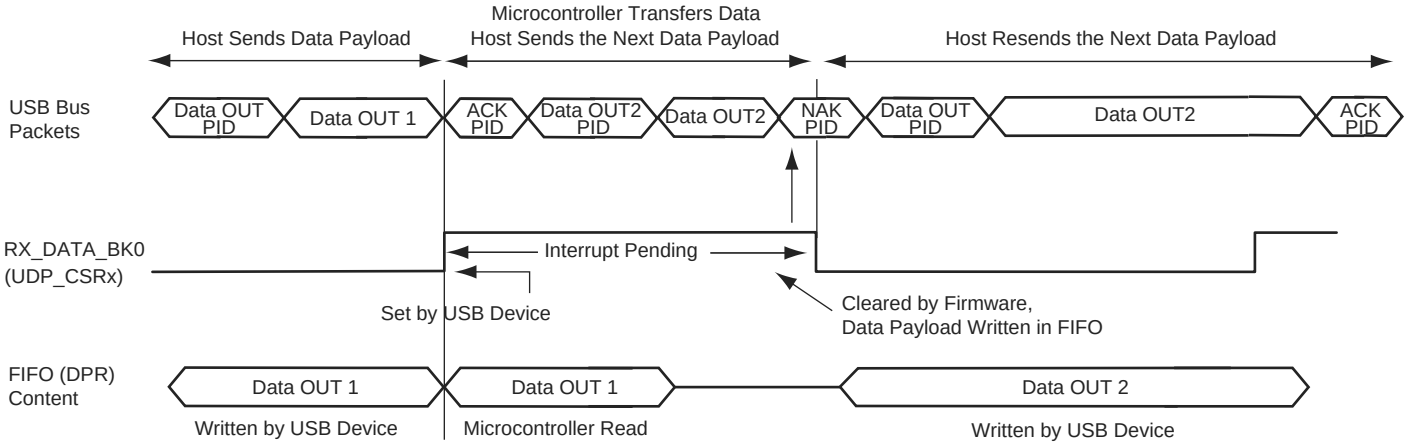
Address: 0x40000200

Access: Read/Write

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

- **DATA: Data to Read or Data to Write**

Figure 40-9. Data OUT Transfer for Non Ping-pong Endpoints

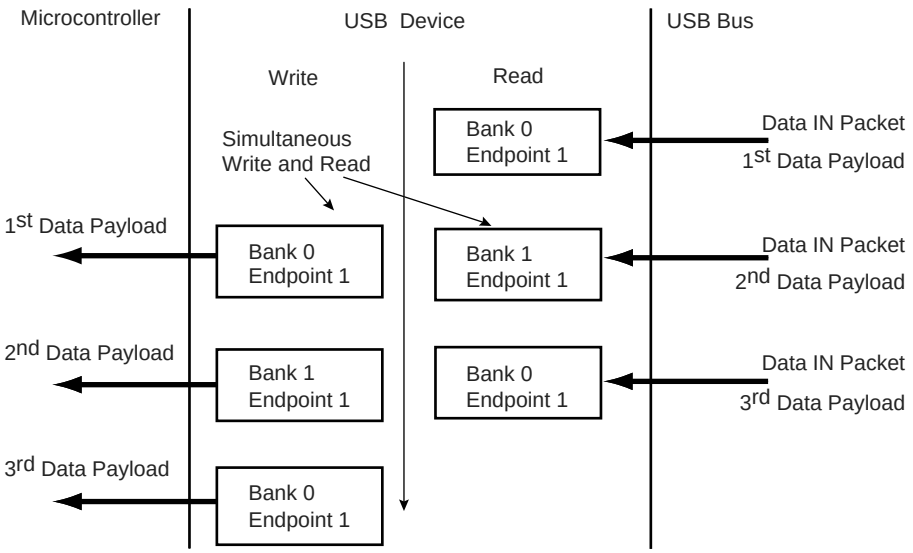


An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory is not possible after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 40-10. Bank Swapping in Data OUT Transfers for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_CSRx. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.

43.7.9 DACC Interrupt Mask Register

Name: DACC_IMR

Address: 0x4003C02C

Access: Read-only

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

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

0: Corresponding interrupt is not enabled

1: Corresponding interrupt is enabled

- **TXRDY: Transmit Ready Interrupt Mask**
- **EOC: End of Conversion Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

45.9 Soldering Profile

Table 45-31 gives the recommended soldering profile from J-STD-020C.

Table 45-31. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ± 25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: The package is certified to be backward compatible with Pb/Sn soldering profile.

A maximum of three reflow passes is allowed per component.

45.10 Packaging Resources

Land Pattern Definition.

Refer to the following IPC Standards:

- IPC-7351A and IPC-782 (*Generic Requirements for Surface Mount Design and Land Pattern Standards*) http://landpatterns.ipc.org/default.asp
- Atmel Green and RoHS Policy and Package Material Declaration Datasheet available on www.atmel.com

Table 49-3. SAM4S Datasheet Rev. 11100I Revision History (Continued)

Doc. Date	Changes
03-Apr-15	**Section 18., “Supply Controller (SUPC)”** Updated Figure 18-1 “Supply Controller Block Diagram”. Section 18.4.2, “Slow Clock Generator”: modified last paragraph with information on entering Bypass mode. Modfied Section 18.4.7.1, “Wake-up Inputs”, Section 18.5.5, “Supply Controller Mode Register”, Section 18.5.7, “Supply Controller Wake-up Inputs Register” and Section 18.5.8, “Supply Controller Status Register” Section 18.4.7.2, “Low-power Tamper Detection and Anti-Tampering”: corrected ‘...LPDBCCLR bit must be set in SUPC_MR’ to ‘...LPDBCCLR bit must be set in SUPC_WUMR’Updated Figure 18-4 “Wake-up Sources” Section 18.5.2, “Supply Controller (SUPC) User Interface”: corrected reset value for SUPC_MR.
	Section 19., “General Purpose Backup Registers (GPBR)” Modified Section 19.1, “Description” and Section 19.2, “Embedded Characteristics” Table 19-1 “Register Mapping”: added reset value 0x00000000 for all registers SYS_GPBRx Section 19.3.1, “General Purpose Backup Register x”: inserted sentence “These registers are reset at first power-up and on each loss of VVDIO”
	Section 20., “Enhanced Embedded Flash Controller (EEFC)”. Updated Section 20.2, “Embedded Characteristics” Added Figure 20-1 Flash Memory Areas Section 20.3.2, “Interrupt Sources”: changed last sentence Section 20.4.1, “Embedded Flash Organization”: corrected instance of command name “Get descriptor” to “Get Flash Descriptor” Figure 20-7 Command State Chart: replaced two instances of “MC_FSR” with “EEFC_FSR” Modified Section 20.4.3.2, “Write Commands”, Section 20.4.3.3, “Erase Commands”, Section 20.4.3.8, “Unique Identifier Area” and Section 20.4.3.9, “User Signature Area” Section 20.5, “Enhanced Embedded Flash Controller (EEFC) User Interface”: deleted address offsets from individual register descriptions (offsets are provided in Section 20-6, “Register Mapping”) Modified Section 20.5.1, “EEFC Flash Mode Register”, Section 20.5.2, “EEFC Flash Command Register” and Section 20.5.3, “EEFC Flash Status Register”: added new field ‘ZERO’ (register bits 26:25) Updated Section 20-6, “Register Mapping”
	Section 21., “Fast Flash Programming Interface (FFPI)” Section 21-1, “16-bit Parallel Programming Interface”: renamed figure
	Section 22., “Cortex-M Cache Controller (CMCC)” Section 22.4.3, “Cache Performance Monitoring”: corrected “MODE field of the CMCC_CFG register” to “MODE field of the CMCC_MCFG register” Table 22-1 "Register Mapping": removed CMCC_CTRL reset value for this write-only register; replaced “Cache ...” with “Cache Controller ...” in “Register” column contents Updated Section 22.5.1, “Cache Controller Type Register” Updated bit descriptions in Section 22.5.3, “Cache Controller Control Register”, Section 22.5.4, “Cache Controller Status Register”, Section 22.5.5, “Cache Controller Maintenance Register 0”, Section 22.5.8, “Cache Controller Monitor Enable Register” and Section 22.5.9, “Cache Controller Monitor Control Register”. Section 22.5.7, “Cache Controller Monitor Configuration Register”: changed access from Write-only to Read/Write. Section 22.5.8, “Cache Controller Monitor Enable Register”: removed reset value from several write-only registers (reset values are found in Table 22-1 "Register Mapping") and changed access from Write-only to Read/Write.