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	ARM® Cortex®-M4/M4F
Core Size	32-Bit Dual-Core
Speed	120MHz
Connectivity	EBI/EMI, I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, LCD, POR, PWM, WDT
Number of I/O	74
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4c16ca-au

12.4.1.14 Fault Mask Register

Name: FAULTMASK

Access: Read/Write

Reset: 0x000000000

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
–							FAULTMASK

The FAULTMASK register prevents the activation of all exceptions except for Non-Maskable Interrupt (NMI).

- **FAULTMASK**

0: No effect.

1: Prevents the activation of all exceptions except for NMI.

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler.

12.6 Cortex-M4 Instruction Set

12.6.1 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 12-13 lists the supported instructions.

- Angle brackets, <>, enclose alternative forms of the operand
- Braces, {}, enclose optional operands
- The Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- Most instructions can use an optional condition code suffix.

For more information on the instructions and operands, see the instruction descriptions.

Table 12-13. Cortex-M4 Instructions

Mnemonic	Operands	Description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn, #imm12	Add	N,Z,C,V
ADR	Rd, label	Load PC-relative address	–
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right	N,Z,C
B	label	Branch	–
BFC	Rd, #lsb, #width	Bit Field Clear	–
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	–
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N,Z,C
BKPT	#imm	Breakpoint	–
BL	label	Branch with Link	–
BLX	Rm	Branch indirect with Link	–
BX	Rm	Branch indirect	–
CBNZ	Rn, label	Compare and Branch if Non Zero	–
CBZ	Rn, label	Compare and Branch if Zero	–
CLREX	–	Clear Exclusive	–
CLZ	Rd, Rm	Count leading zeros	–
CMN	Rn, Op2	Compare Negative	N,Z,C,V
CMP	Rn, Op2	Compare	N,Z,C,V
CPSID	i	Change Processor State, Disable Interrupts	–
CPSIE	i	Change Processor State, Enable Interrupts	–
DMB	–	Data Memory Barrier	–
DSB	–	Data Synchronization Barrier	–
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C
ISB	–	Instruction Synchronization Barrier	–
IT	–	If-Then condition block	–
LDM	Rn{!}, reglist	Load Multiple registers, increment after	–

Examples

```
LDR     R8, [R10]              ; Loads R8 from the address in R10.
LDRNE   R2, [R5, #960]!        ; Loads (conditionally) R2 from a word
                               ; 960 bytes above the address in R5, and
                               ; increments R5 by 960.
STR     R2, [R9,#const-struc]  ; const-struc is an expression evaluating
                               ; to a constant in the range 0-4095.
STRH    R3, [R4], #4           ; Store R3 as halfword data into address in
                               ; R4, then increment R4 by 4
LDRD    R8, R9, [R3, #0x20]    ; Load R8 from a word 32 bytes above the
                               ; address in R3, and load R9 from a word 36
                               ; bytes above the address in R3
STRD    R0, R1, [R8], #-16     ; Store R0 to address in R8, and store R1 to
                               ; a word 4 bytes above the address in R8,
                               ; and then decrement R8 by 16.
```

12.6.4.3 LDR and STR, Register Offset

Load and Store with register offset.

Syntax

op{*type*}{*cond*} *Rt*, [*Rn*, *Rm* {, LSL #*n*}]

where:

op is one of:

- LDR Load Register.
- STR Store Register.

type is one of:

- B unsigned byte, zero extend to 32 bits on loads.
- SB signed byte, sign extend to 32 bits (LDR only).
- H unsigned halfword, zero extend to 32 bits on loads.
- SH signed halfword, sign extend to 32 bits (LDR only).
- \- omit, for word.

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

Rt is the register to load or store.

Rn is the register on which the memory address is based.

Rm is a register containing a value to be used as the offset.

LSL *#n* is an optional shift, with *n* in the range 0 to 3.

Operation

LDR instructions load a register with a value from memory.

STR instructions store a register value into memory.

The memory address to load from or store to is at an offset from the register *Rn*. The offset is specified by the register *Rm* and can be shifted left by up to 3 bits using LSL.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See “Address Alignment”.

12.6.11.26 VSQRT

Floating-point Square Root.

Syntax

```
VSQRT{cond}.F32 Sd, Sm
```

where:

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

Sd is the destination floating-point value.

Sm is the operand floating-point value.

Operation

This instruction:

- Calculates the square root of the value in a floating-point register.
- Writes the result to another floating-point register.

Restrictions

There are no restrictions.

Condition Flags

These instructions do not change the flags.

12.12.2.3 Floating-point Context Address Register

Name: FPCAR

Access: Read/Write

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

The FPCAR holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

- **ADDRESS: Location of Unpopulated Floating-point Register Space Allocated on an Exception Stack Frame**

The location of the unpopulated floating-point register space allocated on an exception stack frame.

- **WDDIS: Watchdog Disable**

0: Enables the RSWDT.

1: Disables the RSWDT.

Table 23-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIO	I/O Lines Power Supply	Power	–	–
VDDCORE	Core Power Supply	Power	–	–
VDDPLL	PLL Power Supply	Power	–	–
GND	Ground	Ground	–	–
Clocks				
XIN	Main Clock Input	Input	–	–
Test				
TST	Test Mode Select	Input	High	Must be connected to VDDBU
PGMEN0	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN1	Test Mode Select	Input	High	Must be connected to VDDIO
PIO				
PGMNCMD	Valid command available	Input	Low	Pulled-up input at reset
PGMRDY	0: Device is busy 1: Device is ready for a new command	Output	High	Pulled-up input at reset
PGMNOE	Output Enable (active high)	Input	Low	Pulled-up input at reset
PGMNVALID	0: DATA[15:0] is in input mode 1: DATA[15:0] is in output mode	Output	Low	Pulled-up input at reset
PGMM[3:0]	Specifies DATA type (see Table 23-2)	Input	–	Pulled-up input at reset
PGMD[15:0]	Bi-directional data bus	Input/Output	–	Pulled-up input at reset

23.3.2 Signal Names

Depending on the MODE settings, DATA is latched in different internal registers.

Table 23-2. Mode Coding

MODE[3:0]	Symbol	Data
0000	CMDE	Command Register
0001	ADDR0	Address Register LSBs
0010	ADDR1	–
0011	ADDR2	–
0100	ADDR3	Address Register MSBs
0101	DATA	Data Register
Default	IDLE	No register

Figure 23-2. Parallel Programming Timing, Write Sequence

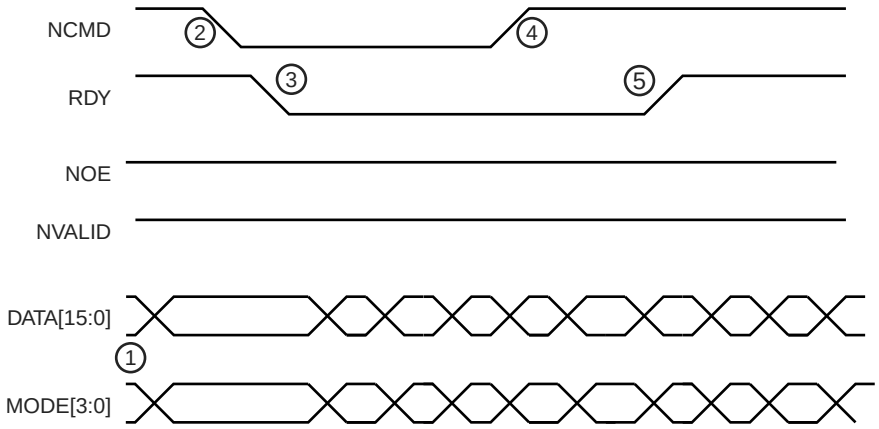


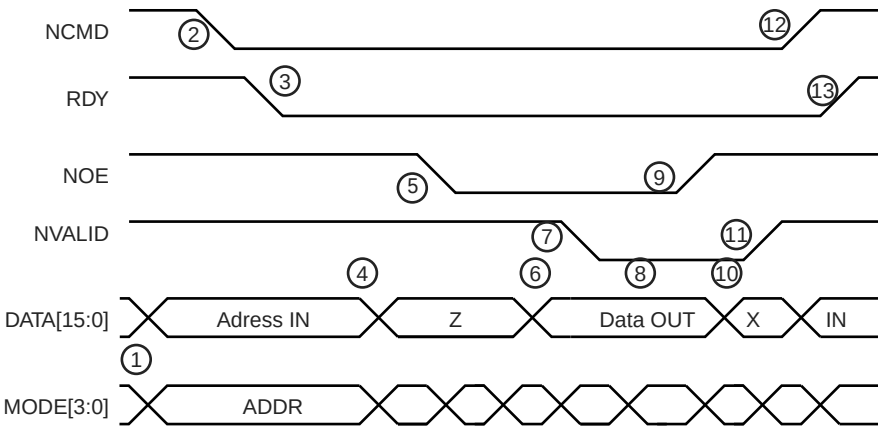
Table 23-4. Write Handshake

Step	Programmer Action	Device Action	Data I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latches MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Releases MODE and DATA signals	Executes command and polls NCMD high	Input
5	Sets NCMD signal	Executes command and polls NCMD high	Input
6	Waits for RDY high	Sets RDY	Input

23.3.4.2 Read Handshaking

For details on the read handshaking sequence, refer to Figure 23-3 and Table 23-5.

Figure 23-3. Parallel Programming Timing, Read Sequence



This allows the Bus Matrix arbiters to remove the one latency clock cycle for the fixed default master of the slave. All requests attempted by the fixed default master do not cause any arbitration latency, whereas other non-privileged masters will get one latency cycle. This technique is used for a master that mainly performs single accesses or short bursts with idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput, regardless of the number of requesting masters.

26.7 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when a conflict occurs, i.e. when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, thus arbitrating each slave specifically.

The Bus Matrix provides the user with the possibility of choosing between two arbitration types or mixing them for each slave:

1. Round-robin arbitration (default)
2. Fixed priority arbitration

The resulting algorithm may be complemented by selecting a default master configuration for each slave.

When re-arbitration must be done, specific conditions apply. See Section 26.7.1 “Arbitration Scheduling”.

26.7.1 Arbitration Scheduling

Each arbiter has the ability to arbitrate between two or more master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it
2. Single cycles: When a slave is currently doing a single access
3. End of Burst cycles: When the current cycle is the last cycle of a burst transfer. For defined burst length, predicted end of burst matches the size of the transfer but is managed differently for undefined burst length. See Section 26.7.1.1 “Undefined Length Burst Arbitration”
4. Slot cycle limit: When the slot cycle counter has reached the limit value, indicating that the current master access is too long and must be broken. See Section 26.7.1.2 “Slot Cycle Limit Arbitration”

26.7.1.1 Undefined Length Burst Arbitration

In order to prevent long AHB burst lengths that can lock the access to the slave for an excessive period of time, the user can trigger the re-arbitration before the end of the incremental bursts. The re-arbitration period can be selected from the following Undefined Length Burst Type (ULBT) possibilities:

1. Unlimited: no predetermined end of burst is generated. This value enables 1-Kbyte burst lengths.
2. 1-beat bursts: predetermined end of burst is generated at each single transfer during the INCR transfer.
3. 4-beat bursts: predetermined end of burst is generated at the end of each 4-beat boundary during INCR transfer.
4. 8-beat bursts: predetermined end of burst is generated at the end of each 8-beat boundary during INCR transfer.
5. 16-beat bursts: predetermined end of burst is generated at the end of each 16-beat boundary during INCR transfer.
6. 32-beat bursts: predetermined end of burst is generated at the end of each 32-beat boundary during INCR transfer.
7. 64-beat bursts: predetermined end of burst is generated at the end of each 64-beat boundary during INCR transfer.

30.19.6 PMC Peripheral Clock Status Register 0

Name: PMC_PCSR0

Address: 0x400E0418

Access: Read-only

31	30	29	28	27	26	25	24
PID31	–	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PIDx: Peripheral Clock x Status**

0: The corresponding peripheral clock is disabled.

1: The corresponding peripheral clock is enabled.

Note: PIDx refers to identifiers defined in the section “Peripheral Identifiers”. Other peripherals status can be read in PMC_PCSR1 (Section 30.19.26 ”PMC Peripheral Clock Status Register 1”).

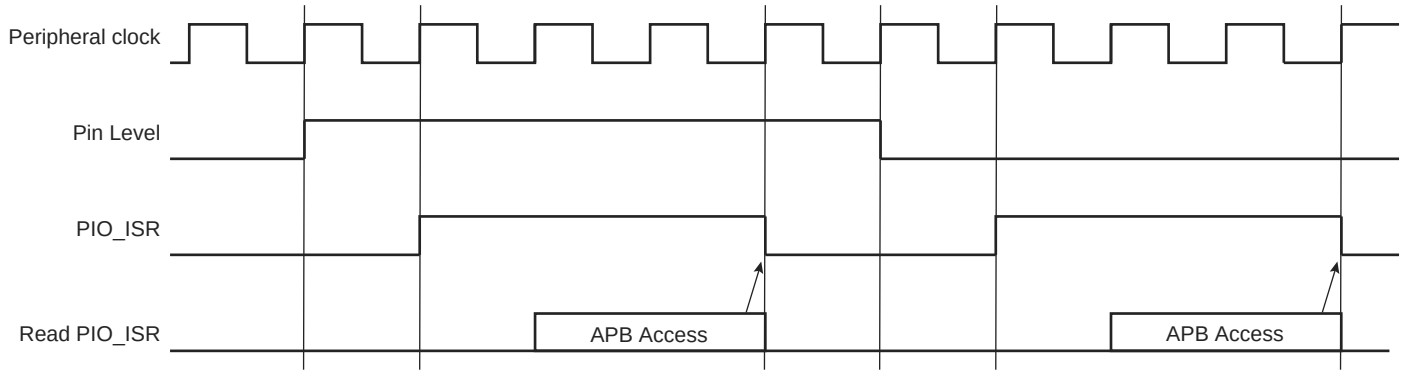
- Rising edge on PIO line 7
- Any edge on the other lines

Table 32-2 provides the required configuration for this example.

Table 32-2. Configuration for Example Interrupt Generation

Configuration	Description
Interrupt Mode	All the interrupt sources are enabled by writing 32'hFFFF_FFFF in PIO_IER. Then the additional interrupt mode is enabled for lines 0 to 7 by writing 32'h0000_00FF in PIO_AIMER.
Edge or Level Detection	Lines 3, 4 and 5 are configured in level detection by writing 32'h0000_0038 in PIO_LSR. The other lines are configured in edge detection by default, if they have not been previously configured. Otherwise, lines 0, 1, 2, 6 and 7 must be configured in edge detection by writing 32'h0000_00C7 in PIO_ESR.
Falling/Rising Edge or Low/High-Level Detection	Lines 0, 2, 4, 5 and 7 are configured in rising edge or high-level detection by writing 32'h0000_00B5 in PIO_REHLSR. The other lines are configured in falling edge or low-level detection by default if they have not been previously configured. Otherwise, lines 1, 3 and 6 must be configured in falling edge/low-level detection by writing 32'h0000_004A in PIO_FELLSR.

Figure 32-7. Input Change Interrupt Timings When No Additional Interrupt Modes



32.5.11 Programmable I/O Drive

It is possible to configure the I/O drive for pads PA0 to PA31. Refer to Section 46. "Electrical Characteristics".

32.5.12 Programmable Schmitt Trigger

It is possible to configure each input for the Schmitt trigger. By default the Schmitt trigger is active. Disabling the Schmitt trigger is requested when using the QTouch® Library.

32.5.13 I/O Lines Programming Example

The programming example shown in Table 32-3 is used to obtain the following configuration:

- 4-bit output port on I/O lines 0 to 3 (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor, no pull-down resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts

- **ONEBIT: Start Frame Delimiter Selector**

0: Start frame delimiter is COMMAND or DATA SYNC.

1: Start frame delimiter is one bit.

40.7.18 ADC Analog Control Register

Name: ADC_ACR

Address: 0x40038094

Access: Read/Write

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

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

- **IRVCE: Internal Reference Voltage Change Enable**

0 (STUCK_AT_DEFAULT): The internal reference voltage is stuck at the default value (see Section 46. ”Electrical Characteristics” for further details).

1 (SELECTION): The internal reference voltage is defined by field IRVS.

- **IRVS: Internal Reference Voltage Selection**

See Table 46-44 “Programmable Voltage Reference Selection Values” for further details.

- **FORCEREF: Force Internal Reference Voltage**

0: The internal ADC voltage reference input is connected to the ADVREF line

1: The internal ADC voltage reference input is forced to VDDIO (ONREF must be cleared).

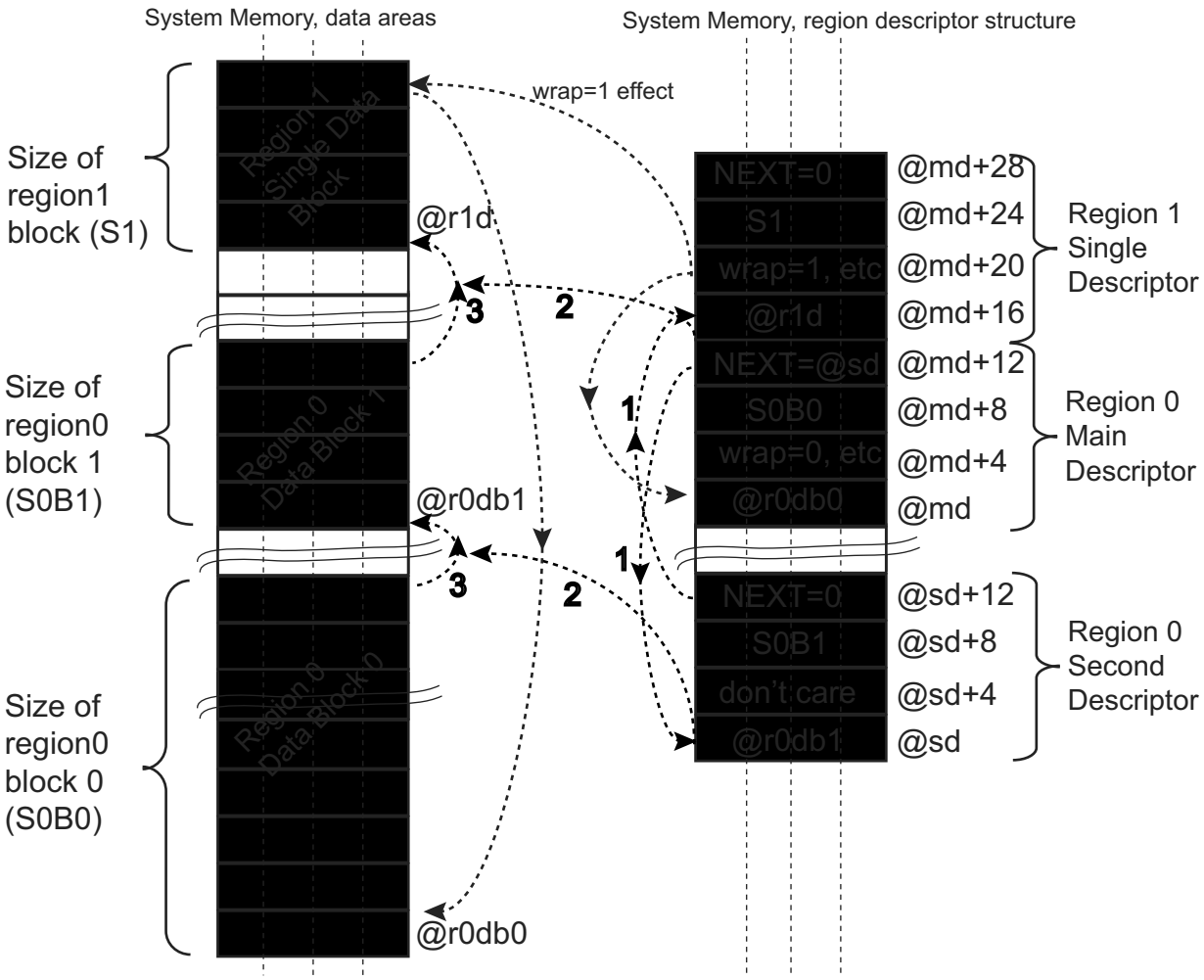
- **ONREF: Internal Voltage Reference ON**

0: The programmable voltage reference is OFF. The user can either force the internal ADC voltage reference input on the ADVREF pin or set the FORCEREF bit to connect VDDIO to the internal ADC voltage reference input.

1: The programmable voltage reference is ON and its output is connected both to the ADC voltage reference input and to the external ADVREF pin for decoupling (FORCEREF must be cleared).

Figure 42-5 shows an example of the mandatory ICM settings to monitor three memory data blocks of the system memory (defined as two regions) with one region being not contiguous (two separate areas) and one contiguous memory area. For each said region, the SHA algorithm may be independently selected (different for each region). The wrap allows continuous monitoring.

Figure 42-5. Example: Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)



42.5.2 ICM Region Descriptor Structure

The ICM Region Descriptor Area is a contiguous area of system memory that the controller and the processor can access. When the ICM controller is activated, the controller performs a descriptor fetch operation at *(ICM_DSCR) address. If the Main List contains more than one descriptor (i.e., more than one region is to be monitored), the fetch address is *(ICM_DSCR) + (RID<<4) where RID is the region identifier.

Table 42-2. Region Descriptor Structure (Main List)

Offset	Structure Member	Name
ICM_DSCR+0x000+RID*(0x10)	ICM Region Start Address	ICM_RADDR
ICM_DSCR+0x004+RID*(0x10)	ICM Region Configuration	ICM_RCFG
ICM_DSCR+0x008+RID*(0x10)	ICM Region Control	ICM_RCTRL
ICM_DSCR+0x00C+RID*(0x10)	ICM Region Next Address	ICM_RNEXT

42.6.10 ICM Hash Area Start Address Register

Name: ICM_HASH

Address: 0x40044034

Access: Read/Write

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

- **HASA: Hash Area Start Address**

This field points at the Hash memory location. The address must be a multiple of 128 bytes.

- all the registers of the pipe (USBFS_HSTPIPCFGx, USBFS_HSTPIPISRx, USBFS_HSTPIPIMRx), except its configuration (USBFS_HSTPIPCFGx.ALLOC, USBFS_HSTPIPCFGx.PBK, USBFS_HSTPIPCFGx.PSIZE, USBFS_HSTPIPCFGx.PTOKEN, USBFS_HSTPIPCFGx.PTYPE, USBFS_HSTPIPCFGx.PEPNUM, USBFS_HSTPIPCFGx.INTFRQ) and its Data Toggle Sequence field (USBFS_HSTPIPISRx.DTSEQ).

The pipe configuration remains active and the pipe is still enabled.

The pipe reset may be associated with a clear of the data toggle sequence. This can be achieved by setting the Reset Data Toggle bit in the Pipe x Control register (USBFS_HSTPIPIMRx.RSTDT) (by writing a one to the Reset Data Toggle Set bit in the Pipe x Control Set register (USBFS_HSTPIPIERx.RSTDTS)).

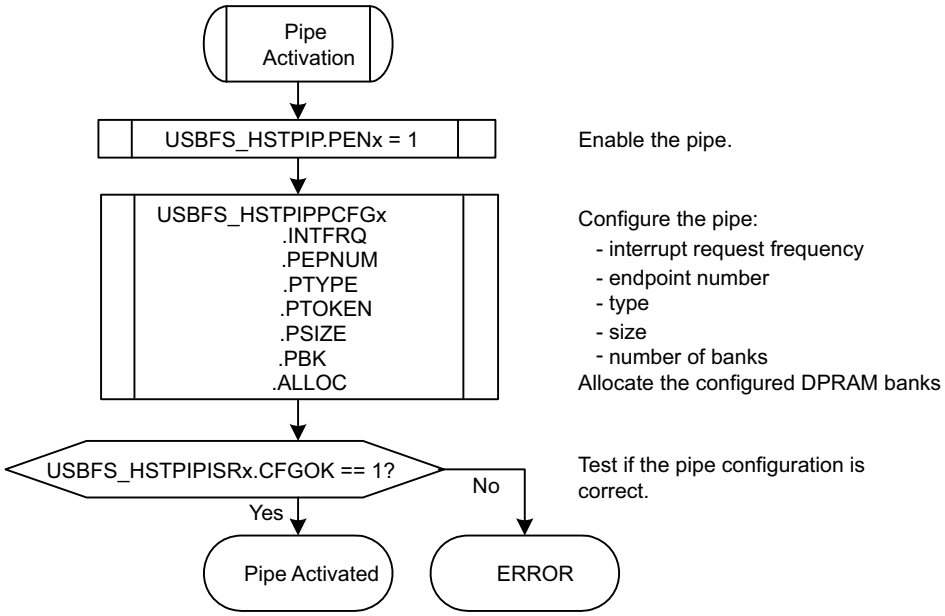
In the end, the user has to write a zero to the USBFS_HSTPIP.PRSTx bit to complete the reset operation and to start using the FIFO.

45.5.3.6 Pipe Activation

The pipe is maintained inactive and reset (see Section 45.5.3.5 ”Pipe Reset” for more details) as long as it is disabled (USBFS_HSTPIP.PENx = 0). The Data Toggle Sequence field (USBFS_HSTPIPISRx.DTSEQ) is also reset.

The algorithm represented on Figure 45-21 must be followed in order to activate a pipe.

Figure 45-21. Pipe Activation Algorithm



As long as the pipe is not correctly configured (USBFS_HSTPIPISRx.CFGOK = 0), the controller cannot send packets to the device through this pipe.

The USBFS_HSTPIPISRx.CFGOK bit is only set if the configured size and number of banks are correct as compared to their maximal allowed values for the pipe (see Table 45-1) and to the maximal FIFO size (i.e., the DPRAM size).

See Section 45.5.1.4 ”DPRAM Management” for more details about DPRAM management.

Once the pipe is correctly configured (USBFS_HSTPIPISRx.CFGOK = 1), only the USBFS_HSTPIPCFGx.PTOKEN and USBFS_HSTPIPCFGx.INTFRQ fields can be written by software. USBFS_HSTPIPCFGx.INTFRQ is meaningless for non-interrupt pipes.

45.6.2 General Status Register

Name: USBFS_SR

Address: 0x40020804

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
–	–	–	–	–	–	VBUSRQ	–

7	6	5	4	3	2	1	0
-	–	–	RDERRI	–	–	–	–

- **RDERRI: Remote Device Connection Error Interrupt (Host mode only)**

0: Cleared when USBFS_SCR.RDERRIC = 1.

1: Set when an error occurs during the Remote Device Connection. This triggers a USB interrupt if USBFS_CTRL.RDERRE = 1.

- **VBUSRQ: VBus Request (Host mode only)**

0: The VBOF output pin is driven low to disable the VBUS power supply generation. VBUSRQ is cleared when USBFS_SCR.VBUSRQC = 1 or when a VBus error occurs and USBFS_CTRL.VBUSHWC = 0.

1: The VBOF output pin is driven high to enable the VBUS power supply generation. VBUSRQ is set when USBFS_SFR.VBUSRQS = 1.

For IN transfers, it indicates the data toggle sequence that should be used for the next packet to be sent. This is not relative to the current bank.

For OUT transfers, this value indicates the last data toggle sequence received on the current bank.

By default, DTSEQ is 0b01, as if the last data toggle sequence was Data1, so the next sent or expected data toggle sequence should be Data0.

- **NBUSYBK: Number of Busy Banks**

This field is set to indicate the number of busy banks:

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

For IN endpoints, it indicates the number of banks filled by the user and ready for IN transfer. When all banks are free, this triggers a PEP_x interrupt if NBUSYBKE = 1.

For OUT endpoints, it indicates the number of banks filled by OUT transactions from the host. When all banks are busy, this triggers a PEP_x interrupt if NBUSYBKE = 1.

When the USBFS_DEVEPTIMRx.FIFOCON bit is cleared (by writing a one to the USBFS_DEVEPTIMRx.FIFOCONC bit) to validate a new bank, this field is updated two or three clock cycles later to calculate the address of the next bank.

A PEP_x interrupt is triggered if:

- for IN endpoint, USBFS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are free;
- for OUT endpoint, USBFS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are busy.

- **CURRBK: Current Bank**

This bit is set for non-control endpoints, to indicate the current bank:

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

- **RWALL: Read/Write Allowed**

This bit is set for IN endpoints when the current bank is not full, i.e., the user can write further data into the FIFO.

This bit is set for OUT endpoints when the current bank is not empty, i.e., the user can read further data from the FIFO.

This bit is never set if USBFS_DEVEPTIMRx.STALLRQ is one or in case of error.

This bit is cleared otherwise.

This bit should not be used for control endpoints.

- **CTRLDIR: Control Direction**

0: Cleared after a SETUP packet to indicate that the following packet is an OUT packet.

1: Set after a SETUP packet to indicate that the following packet is an IN packet.

45.6.17 Device Endpoint x Clear Register (Control, Bulk, Interrupt Endpoints)

Name: USBFS_DEVEPTICRx [x=0..4]

Address: 0x40020160

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
SHORTPACKETC	STALLEDIC	OVERFIC	NAKINIC	NAKOUTIC	RXSTPIC	RXOUTIC	TXINIC

This register view is relevant only if EPTYPE = 0x0, 0x2 or 0x3 in Device Endpoint x Configuration Register.

For additional information, see Section 45.6.15 ”Device Endpoint x Status Register (Control, Bulk, Interrupt Endpoints)”.

This register always reads as zero.

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

0: No effect

1: Clears the corresponding bit in USBFS_DEVEPTISRx

- **TXINIC: Transmitted IN Data Interrupt Clear**
- **RXOUTIC: Received OUT Data Interrupt Clear**
- **RXSTPIC: Received SETUP Interrupt Clear**
- **NAKOUTIC: NAKed OUT Interrupt Clear**
- **NAKINIC: NAKed IN Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **STALLEDIC: STALLed Interrupt Clear**
- **SHORTPACKETC: Short Packet Interrupt Clear**

Table 53-3. SAM4C Datasheet Rev. 11102E Revision History (Continued)

Doc. Rev. 11102E	Changes
06-Oct-14	Section 15. “Reset Controller (RSTC)”
	Section 15.4.1 “Reset Controller Overview”: changed events that that trigger assertion of reset signals by the RSTC. Changed VDD_REG_BU to VDDBU.
	Removed section “Brownout Manager”.
	Section 15.4.3.2 “Backup Reset”: replaced “core_backup_reset” with “vddcore_nreset”.
	Section 15.5.1 “Reset Controller Control Register”: modified EXTRST description.
	Section 15.5.2 “Reset Controller Status Register”: modified bit descriptions.
	Section 15.5.3 “Reset Controller Mode Register”: modified ERSTL bit description.
	Section 16. “Real-time Timer (RTT)”
	Modified Section 16.4 “Functional Description”.
	Section 16.5.1 “Real-time Timer Mode Register”: modified RTPRES description.
	Section 16.5.2 “Real-time Timer Alarm Register”: modified ALMV description.
	Section 16.5.3 “Real-time Timer Value Register”: added notes.
	Section 17. “Real-time Clock (RTC)”
	Section 17.5.7 “RTC Accurate Clock Calibration”; modified paragraph on calibration circuitry. Added paragraph on clock calibration correction.
	Section 17.6.2 “RTC Mode Register”: modified descriptions of NEGPPM, HIGHPPM and THIGH bits.
	Added Section 17.6.17 “RTC Write Protection Mode Register”.
	Section 18. “Watchdog Timer (WDT)”
	Modified Figure 18-2 “Watchdog Behavior”.
	Section 19. “Reinforced Safety Watchdog Timer (RSWDT)”
	Added Windowed Watchdog in Section 19.2 “Embedded Characteristics” and Section 19.4 “Functional Description”.
	Modified Figure 19-2 “Watchdog Behavior”.
	Section 19.5.2 “Reinforced Safety Watchdog Timer Mode Register”: added notes.
	Section 20. “Supply Controller (SUPC)”
	Section 20.4.2 “Slow Clock Generator”: updated information on entering Bypass mode using OSCBYPASS and XTSALSEL bits.
	Section 20.4.4 “Segmented LCD Voltage Regulator Control”: changed the section name and aligned references to SLCD and SLCD Controller in the text. Updated content.
	Section 20.4.6 “Supply Monitor”: Supply Monitor sampling mode, power reduction factor: replaced incorrect values of 32, 256 or 2048 by the correct values of 2, 16 and 128.
	Figure 20-4 “SAM4C16/8 Wake-up Sources”: modified to show that wake-up input detectors are based on edges. Added Figure 20-5 “SAM4C32 Wake-up Sources”.
	Section 20.4.9.1 “Force Wake-up”: corrected bit name from FWUP to FWUPS in 2nd paragraph.
	Section 20.4.9.2 “Wake-up Inputs”: corrected polarity bit name from WKUPPL to WKUPT.
	Section 20.4.9.3 “Low-power Debouncer Inputs (Tamper Detection Pins)”: added information on SAM4C32 devices. Corrected register names for WKUPTx bits and LPDBCCLR bit.
	Section 20.6.5 “Supply Controller Mode Register”: changed OSCBYPASS bit description.
	Section 20.6.8 “Supply Controller Status Register”: updated to bit descriptions as ‘cleared on read’ where applicable. Updated the bit description for WKUPIS = 1.