Microchip Technology - ATSAM4C32CA-AUR Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

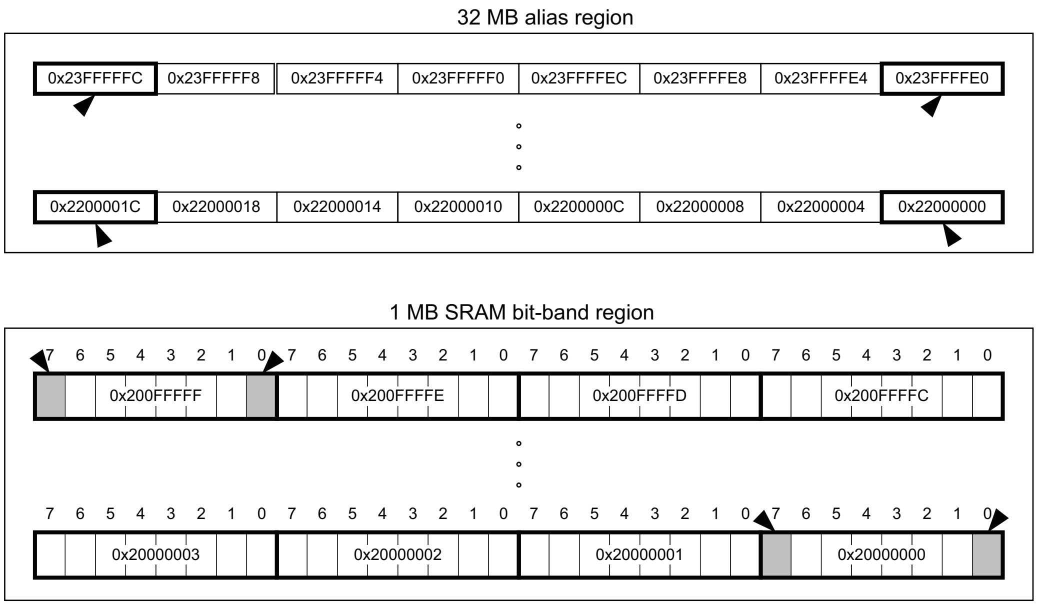
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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/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	2MB (2M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	304K 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/atsam4c32ca-aur

- The alias word at 0x22000000 maps to bit[0] of the bit-band byte at 0x20000000: 0x22000000 = 0x22000000 + (0*32) + (0*4).
- The alias word at 0x2200001C maps to bit[7] of the bit-band byte at 0x20000000: 0x2200001C = 0x22000000+ (0*32) + (7*4).

Figure 12-4. Bit-band Mapping



Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit.

Bits[31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region:

- 0x00000000 indicates that the targeted bit in the bit-band region is set to 0
- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

Directly Accessing a Bit-band Region

“Behavior of Memory Accesses” describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

12.4.2.6 Memory Endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0–3 hold the first stored word, and bytes 4–7 hold the second stored word. “Little-endian Format” describes how words of data are stored in memory.

Little-endian Format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:

12.6.6.12 SDIV and UDIV

Signed Divide and Unsigned Divide.

Syntax

```
SDIV{cond} {Rd,} Rn, Rm
UDIV{cond} {Rd,} Rn, Rm
```

where:

cond	is an optional condition code, see “Conditional Execution”.
Rd	is the destination register. If *Rd* is omitted, the destination register is *Rn*.
Rn	is the register holding the value to be divided.
Rm	is a register holding the divisor.

Operation

SDIV performs a signed integer division of the value in *Rn* by the value in *Rm*.

UDIV performs an unsigned integer division of the value in *Rn* by the value in *Rm*.

For both instructions, if the value in *Rn* is not divisible by the value in *Rm*, the result is rounded towards zero.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
    SDIV  R0, R2, R4  ; Signed divide, R0 = R2/R4
    UDIV  R8, R8, R1  ; Unsigned divide, R8 = R8/R1
```

12.6.11.1 VABS

Floating-point Absolute.

Syntax

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

where:

cond	is an optional condition code, see “Conditional Execution”.
Sd, Sm	are the destination floating-point value and the operand floating-point value.

Operation

This instruction:

1. Takes the absolute value of the operand floating-point register.
2. Places the results in the destination floating-point register.

Restrictions

There are no restrictions.

Condition Flags

The floating-point instruction clears the sign bit.

Examples

```
        VABS.F32 S4, S6
```

12.6.11.14 VMOV Register

Copies the contents of one register to another.

Syntax

```
VMOV{cond}.F64 Dd, Dm
VMOV{cond}.F32 Sd, Sm
```

where:

cond	is an optional condition code, see “Conditional Execution”.
Dd	is the destination register, for a doubleword operation.
Dm	is the source register, for a doubleword operation.
Sd	is the destination register, for a singleword operation.
Sm	is the source register, for a singleword operation.

Operation

This instruction copies the contents of one floating-point register to another.

Restrictions

There are no restrictions

Condition Flags

These instructions do not change the flags.

12.6.11.28 VSTR

Floating-point Store.

Syntax

```
    VSTR{cond}{.32} Sd, [Rn{, #imm}]
    VSTR{cond}{.64} Dd, [Rn{, #imm}]
```

where

cond	is an optional condition code, see “Conditional Execution”.
32, 64	are the optional data size specifiers.
Sd	is the source register for a singleword store.
Dd	is the source register for a doubleword store.
Rn	is the base register. The SP can be used.
imm	is the + or - immediate offset used to form the address. Values are multiples of 4 in the range 0–1020. *imm* can be omitted, meaning an offset of +0.

Operation

This instruction:

- Stores a single extension register to memory, using an address from an ARM core register, with an optional offset, defined in *imm*.

Restrictions

The restrictions are:

- The use of PC for *Rn* is deprecated.

Condition Flags

These instructions do not change the flags.

12.6.12.4 DSB

Data Synchronization Barrier.

Syntax

```
        DSB{cond}
```

where:

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

Operation

DSB acts as a special data synchronization memory barrier. Instructions that come after the DSB, in program order, do not execute until the DSB instruction completes. The DSB instruction completes when all explicit memory accesses before it complete.

Condition Flags

This instruction does not change the flags.

Examples

```
            DSB ; Data Synchronisation Barrier
```

12.6.12.5 ISB

Instruction Synchronization Barrier.

Syntax

```
        ISB{cond}
```

where:

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

Operation

ISB acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the ISB are fetched from memory again, after the ISB instruction has been completed.

Condition Flags

This instruction does not change the flags.

Examples

```
            ISB  ; Instruction Synchronisation Barrier
```

28.5.3 Transmit Pointer Register

Name: PERIPH_TPR

Access: Read/Write

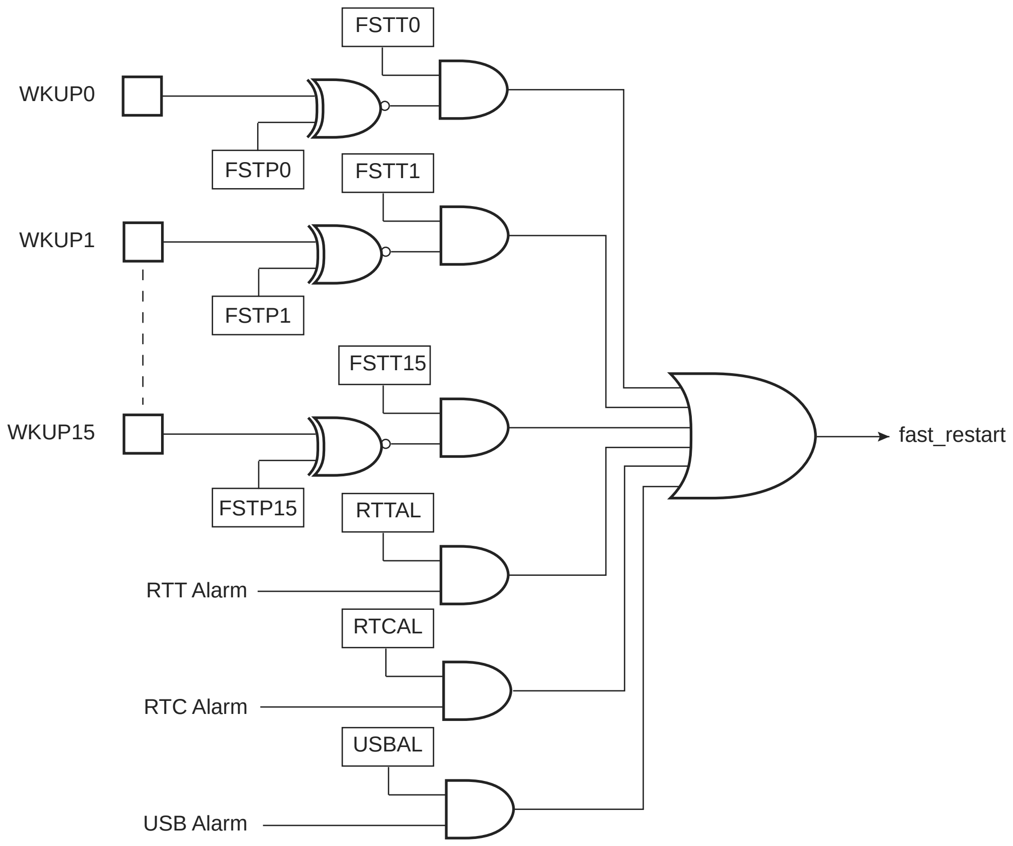
31	30	29	28	27	26	25	24
TXPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXPTR							

- **TXPTR: Transmit Counter Register**

TXPTR must be set to transmit buffer address.

When a half-duplex peripheral is connected to the PDC, RXPTR = TXPTR.

Figure 30-5. Fast Startup Circuitry



Each wake-up input pin and alarm can be enabled to generate a fast startup event by setting the corresponding bit in PMC_FSMR.

The user interface does not provide any status for fast startup, but the user can easily recover this information by reading the PIO Controller and the status registers of the RTC, RTT and USB Controller.

30.12 Main Processor Startup from Embedded Flash

The inherent start-up time of the embedded Flash cannot provide a fast startup of the system.

If system fast start-up time is not required, the first instruction after a Wait mode exit can be located in the embedded Flash. Under these conditions, prior to entering Wait mode, the Flash controller must be programmed to perform access in 0 wait-state. Refer to Section 22. ”Enhanced Embedded Flash Controller (EEFC)”.

The procedure and conditions to enter Wait mode and the circuitry to exit Wait mode are strictly the same as fast startup (refer to Section 30.11 ”Main Processor Fast Startup”).

30.13 Coprocessor Sleep Mode

The coprocessor enters Sleep mode by executing the WaitForInterrupt (WFI) instruction of the coprocessor. Any enabled interrupt can wake the processor up.

30.14 Main Clock Failure Detector

The clock failure detector monitors the 3 to 20 MHz crystal oscillator or ceramic resonator-based oscillator to identify a failure of this oscillator when selected as main clock.

The clock failure detector can be enabled or disabled by bit CFDEN in CKGR_MOR. After a VDDCORE reset, the detector is disabled. However, if the oscillator is disabled (MOSCXTEN = 0), the detector is also disabled.

30.18 Register Write Protection

To prevent any single software error from corrupting PMC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PMC Write Protection Mode Register (PMC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the PMC Write Protection Status Register (PMC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the PMC_WPSR.

The following registers can be write-protected:

- PMC System Clock Enable Register
- PMC System Clock Disable Register
- PMC Peripheral Clock Enable Register 0
- PMC Peripheral Clock Disable Register 0
- PMC Clock Generator Main Oscillator Register
- PMC Clock Generator PLLA Register
- PMC Clock Generator PLLB Register
- PMC Master Clock Register
- PMC USB Clock Register
- PMC Programmable Clock Register
- PMC Fast Startup Mode Register
- PMC Fast Startup Polarity Register
- PMC Coprocessor Fast Startup Mode Register
- PMC Peripheral Clock Enable Register 1
- PMC Peripheral Clock Disable Register 1
- PMC Oscillator Calibration Register

30.19.25PMC Peripheral Clock Disable Register 1

Name: PMC_PCDR1

Address: 0x400E0504

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
–	–	–	–	PID43	PID42	PID41	PID40

7	6	5	4	3	2	1	0
PID39	PID38	PID37	PID36	PID35	PID34	PID33	PID32

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

- **PIDx: Peripheral Clock x Disable**

0: No effect.

1: Disables the corresponding peripheral clock.

Note: The values for PIDx are defined in the section “Peripheral Identifiers”.

32.6.27 PIO Input Filter Slow Clock Enable Register

Name: PIO_IFSCER

Address: 0x400E0E84 (PIOA), 0x400E1084 (PIOB), 0x4800C084 (PIOC), 0x400E1284 (PIOD)

Access: Write-only

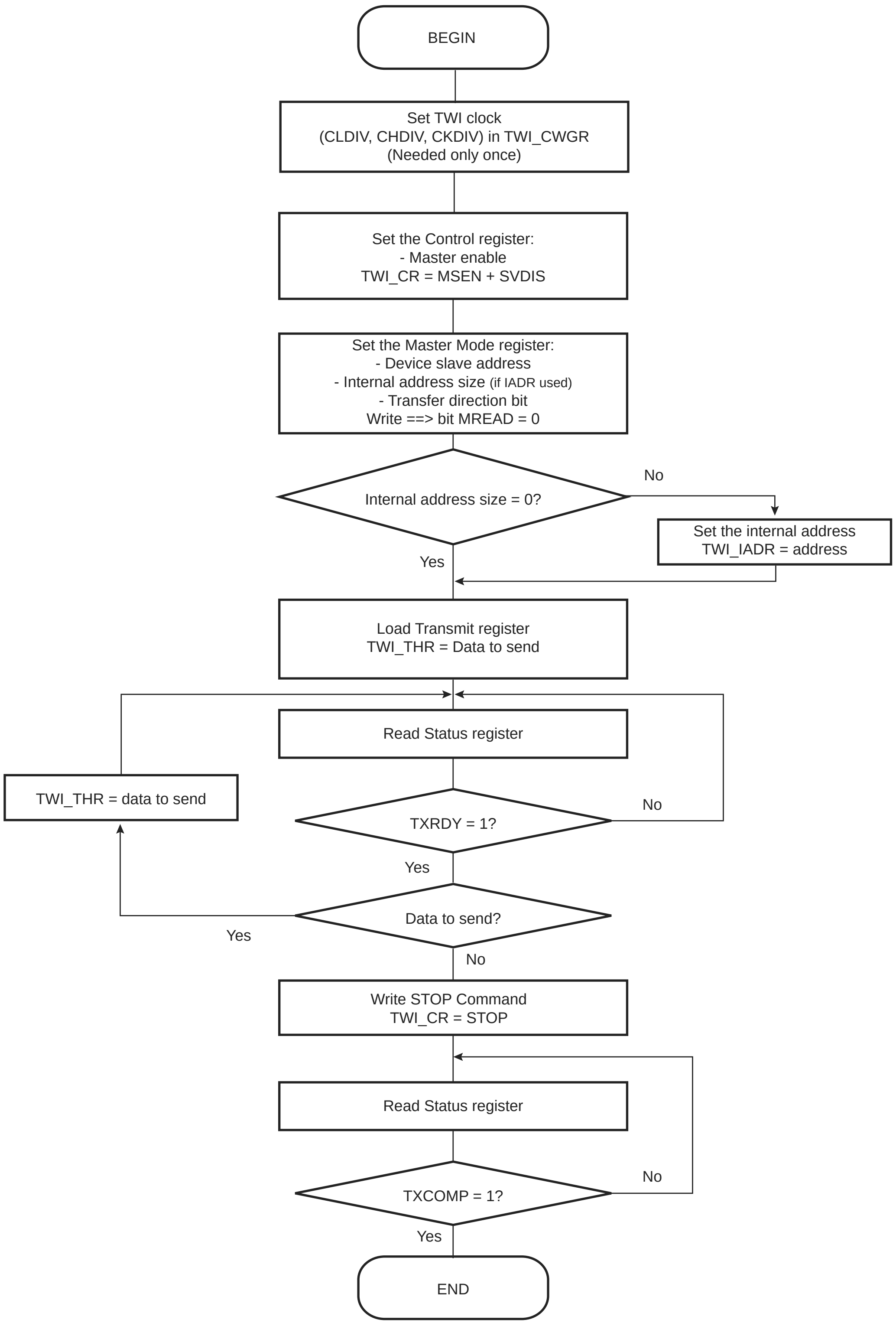
31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Slow Clock Debouncing Filtering Select**

0: No effect.

1: The debouncing filter is able to filter pulses with a duration < $t_{div\_slck}/2$.

Figure 34-17. TWI Write Operation with Multiple Data Bytes with or without Internal Address



34.8.6 TWI Status Register

Name: TWI_SR

Address: 0x40018020 (0), 0x4001C020 (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
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCLWS	ARBLST	NACK

7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed (cleared by writing TWI_THR)**

TXCOMP used in Master mode:

0: During the length of the current frame.

1: When both holding register and internal shifter are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 34-7 and in Figure 34-9.

TXCOMP used in Slave mode:

0: As soon as a START is detected.

1: After a STOP or a REPEATED START + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 34-28, Figure 34-29, Figure 34-30 and Figure 34-31.

- **RXRDY: Receive Holding Register Ready (cleared by reading TWI_RHR)**

0: No character has been received since the last TWI_RHR read operation.

1: A byte has been received in the TWI_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 34-9.

RXRDY behavior in Slave mode can be seen in Figure 34-26, Figure 34-29, Figure 34-30 and Figure 34-31.

- **TXRDY: Transmit Holding Register Ready (cleared by writing TWI_THR)**

TXRDY used in Master mode:

0: The transmit holding register has not been transferred into internal shifter. Set to 0 when writing into TWI_THR.

1: As soon as a data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

TXRDY behavior in Master mode can be seen in Figure 34-5, Figure 34-6 and Figure 34-7.

TXRDY used in Slave mode:

0: As soon as data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1: It indicates that the TWI_THR is empty and that data has been transmitted and acknowledged.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16 × Oversampling

1: 8 × Oversampling

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITER is asserted.

Note: MAX_ITERATION field must be set to 0 if DSNACK is cleared.

- **INVDATA: Inverted Data**

0: The data field transmitted on TXD line is the same as the one written in US_THR or the content read in US_RHR is the same as RXD line. Normal mode of operation.

1: The data field transmitted on TXD line is inverted (voltage polarity only) compared to the value written on US_THR or the content read in US_RHR is inverted compared to what is received on RXD line (or ISO7816 IO line). Inverted mode of operation, useful for contactless card application. To be used with configuration bit MSBF.

- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**

0: User defined configuration of command or data sync field depending on MODSYNC value.

1: The sync field is updated when a character is written into US_THR.

- **MAX_ITERATION: Maximum Number of Automatic Iteration**

0–7: Defines the maximum number of iterations in mode ISO7816, protocol T = 0.

- **FILTER: Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

- **MAN: Manchester Encoder/Decoder Enable**

0: Manchester encoder/decoder are disabled.

1: Manchester encoder/decoder are enabled.

- **MODSYNC: Manchester Synchronization Mode**

0:The Manchester start bit is a 0 to 1 transition

1: The Manchester start bit is a 1 to 0 transition.

41.4.6.3 GCM Processing

GCM processing comprises three phases:

1. Processing the Additional Authenticated Data (*AAD*), hash computation only.
2. Processing the Ciphertext (*C*), hash computation + ciphering/deciphering.
3. Generating the Tag using length of *AAD*, length of *C* and J_0 (see NIST documentation for details).

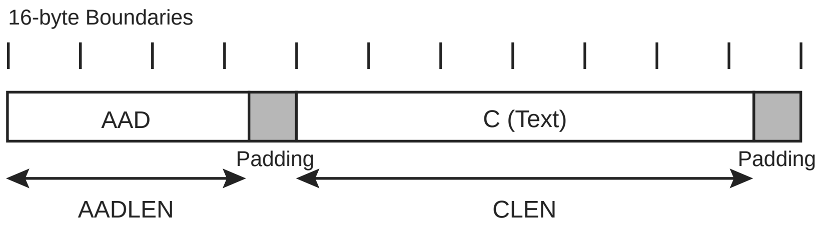
The Tag generation can be done either automatically, after the end of *AAD*/*C* processing if TAG_EN bit is set in the AES_MR or done manually, using the GHASH field in AES_GHASHRx (see below “Processing a Complete Message with Tag Generation” and “Manual GCM Tag Generation” for details).

Processing a Complete Message with Tag Generation

Use this procedure only if J_0 four LSB bytes ≠ 0xFFFFFFFF.

NOTE: In the case where J_0 four LSB bytes = 0xFFFFFFFF or if the value is unknown, use the procedure described in “Processing a Complete Message without Tag Generation” followed by the procedure in “Manual GCM Tag Generation”.

Figure 41-6. Full Message Alignment



To process a complete message with Tag generation, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to ‘1’ (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. See Section 41.4.6.2 “Key Writing and Automatic Hash Subkey Calculation” for details.
3. Calculate the J_0 value as described in NIST documentation $J_0 = IV \parallel 0^{31} \parallel 1$ when $\text{len}(IV) = 96$ and $J_0 = \text{GHASH}_H(IV \parallel 0^{s+64} \parallel [\text{len}(IV)]_{64})$ if $\text{len}(IV) \neq 96$. See “Processing a Message with only AAD (GHASHH)” for J_0 generation.
4. Set IV in AES_IVRx with inc32(J_0) (J_0 + 1 on 32 bits).
5. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR.
6. Fill the IDATA field of AES_IDATARx with the message to process according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Wait for TAGRDY to be set (use interrupt if needed), then read the TAG field of AES_TAGRx to obtain the authentication tag of the message.

Processing a Complete Message without Tag Generation

Processing a message without generating the Tag can be used to customize the Tag generation, or to process a fragmented message. To manually generate the GCM Tag, refer to “Manual GCM Tag Generation”.

To process a complete message without Tag generation, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to ‘0’ (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash

45.5.4 USB DMA Operation

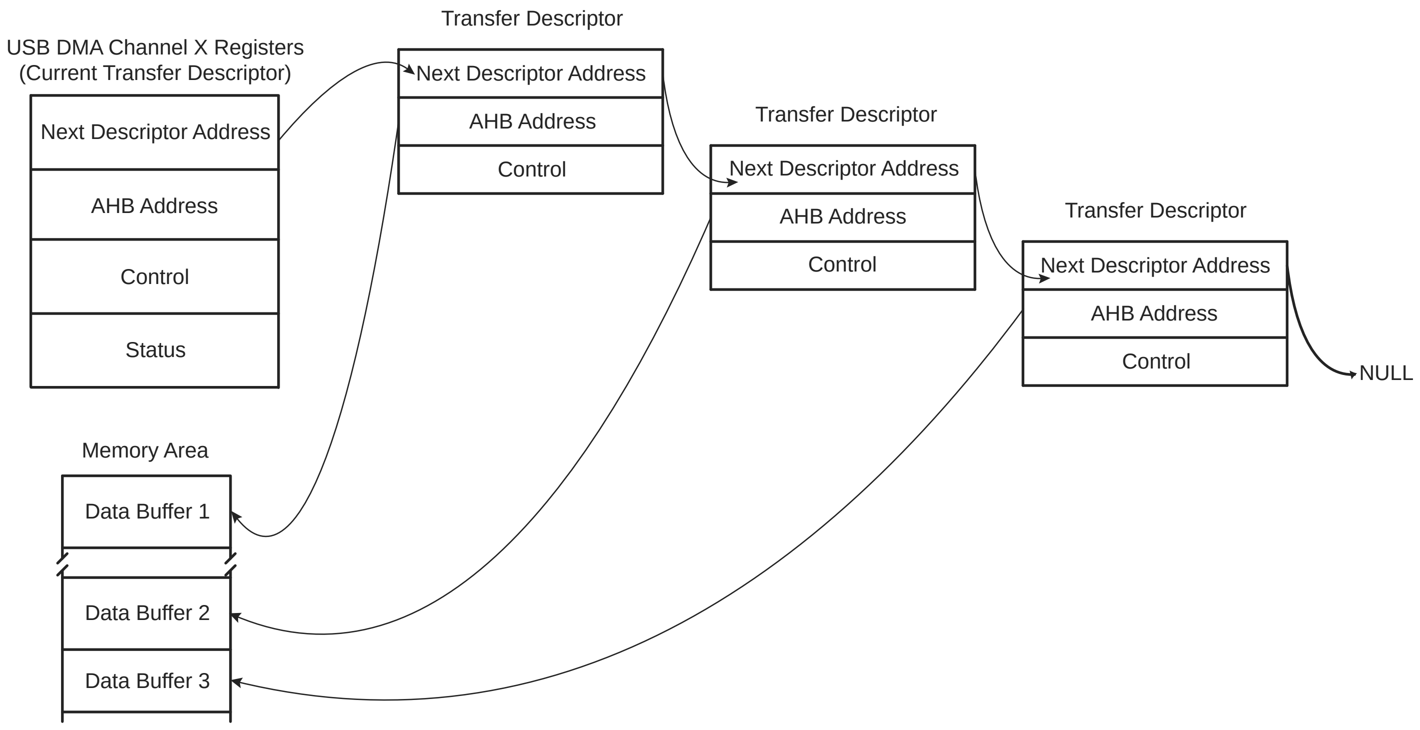
USB packets of any length may be transferred when required by the USBFS. These transfers always feature sequential addressing. Such characteristics mean that in case of high USBFS throughput, both AHB ports benefit from “incrementing burst of unspecified length” since the average access latency of AHB slaves can then be reduced.

The DMA uses word “incrementing burst of unspecified length” of up to 256 beats for both data transfers and channel descriptor loading. A burst may last on the AHB busses for the duration of a whole USB packet transfer, unless otherwise broken by the AHB arbitration or the AHB 1-Kbyte boundary crossing.

Packet data AHB bursts may be locked on a DMA buffer basis for drastic overall AHB bus bandwidth performance boost with paged memories. This prevents large AHB bursts from being broken in case of conflict with other AHB bus masters, thus avoiding the access latencies due to memory row changes. This means up to 16 words single cycle unbroken AHB bursts for bulk pipes/endpoints and 256 words single cycle unbroken bursts for isochronous pipes/endpoints. This maximal burst length is then controlled by the lowest programmed USB Pipe/Endpoint Size (USBFS_HSTPIPCFGx.PSIZE / USBFS_DEVEPTCFGx.EPSIZE) and the Buffer Byte Length (USBFS_HSTDMACONTROLx.BUFF_LENGTH / USBFS_DEVDMACONTROLx.BUFF_LENGTH) fields.

The USBFS average throughput can reach nearly 12 Mbps. Its average access latency decreases as burst length increases due to the zero wait-state side effect of unchanged pipe/endpoint. Word access allows reducing the AHB bandwidth required for the USB by four, as compared to native byte access. If at least 0 wait-state word burst capability is also provided by the other DMA AHB bus slaves, each DMA AHB bus needs less than 1.1% bandwidth allocation for full USB bandwidth usage at 33 MHz, and less than 0.6% at 66 MHz.

Figure 45-27. Example of a DMA Chained List



For IN endpoints:

0: Cleared (by writing a one to the USBFS_DEVEPTIDRx.FIFOCONC bit) to send the FIFO data and to switch to the next bank.

1: Set when the current bank is free, at the same time as USBFS_DEVEPTISRx.TXINI.

For OUT endpoints:

0: Cleared (by writing a one to the USBFS_DEVEPTIDRx.FIFOCONC bit) to free the current bank and to switch to the next bank.

1: Set when the current bank is full, at the same time as USBFS_DEVEPTISRx.RXOUTI.

- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request**

This bit is set when USBFS_DEVEPTIERx.EPDISHDMAS = 1. This pauses the on-going DMA channel x transfer on any Endpoint x interrupt (PEP_x), whatever the state of the Endpoint x Interrupt Enable bit (PEP_x).

The user then has to acknowledge or to disable the interrupt source (e.g. USBFS_DEVEPTISRx.RXOUTI) or to clear the EPDISHDMA bit (by writing a one to the USBFS_DEVEPTIDRx.EPDISHDMAC bit) in order to complete the DMA transfer.

In Ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer does not start (not requested).

If the interrupt is not associated to a new system-bank packet (USBFS_DEVEPTISRx.NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.

This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.

- **RSTDT: Reset Data Toggle**

This bit is set when USBFS_DEVEPTIERx.RSTDTS = 1. This clears the data toggle sequence, i.e., sets to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.

This bit is cleared instantaneously.

The user does not have to wait for this bit to be cleared.

- **STALLRQ: STALL Request**

0: Cleared when a new SETUP packet is received or when USBFS_DEVEPTIDRx.STALLRQC = 0.

1: Set when USBFS_DEVEPTIERx.STALLRQS = 1. This requests to send a STALL handshake to the host.

- **PFREEZE: Pipe Freeze**

0: Cleared when USBFS_HSTPIPIDR.PFREEZEC = 1. This enables the pipe request generation.

1: Set when one of the following conditions is met:

- USBFS_HSTPIPIER.PFREEZES = 1
- The pipe is not configured.
- A STALL handshake has been received on the pipe.
- An error has occurred on the pipe (USBFS_HSTPIPISR.PERRI = 1).
- (INRQ+1) In requests have been processed.
- A Pipe Reset (USBFS_HSTPIP.PRSTx rising) has occurred.
- A Pipe Enable (USBFS_HSTPIP.PEN rising) has occurred.

This freezes the pipe request generation.

- **RSTDT: Reset Data Toggle**

0: No reset of the Data Toggle is ongoing.

1: Set when USBFS_HSTPIPIER.RSTDTS = 1. This resets the Data Toggle to its initial value for the current pipe.

47.3 Soldering Profile

Table 47-5 gives the recommended soldering profile from J-STD-020C.

Table 47-5. 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.

47.4 Packaging Resources

This section provides 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 Data Sheet http://www.atmel.com/about/quality/package.aspx

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

Doc. Rev. 11102E	Changes
06-Oct-14	Section 49. “Ordering Information” Updated Table 49-1 “Ordering Codes for SAM4C Devices”.
	Section 50. “SAM4C16/8 Errata Revision A (MRL A) Parts” Removed erratum on SUPC: LCD End of Frame Disable Does Not Work. Added Section 50.5.2 “RSWDT Windowing Mode”, Section 50.7.1 “Unpredictable Software Behavior When Entering Sleep Mode”, Section 50.8.1 “CORE 1 Systick Counter Erratic Behavior” and Section 50.9.1 “SRCB Bit in CKGR_PLLB Register”.
	Added Section 51. “SAM4C16/8 Errata Revision B (MRL B) Parts”.
	Added Section 52. “SAM4C32 Errata Revision A (MRL A) Parts”.

Table 53-4. SAM4C Datasheet Rev. 11102D Revision History

Doc. Rev. 11102D	Changes
14-Apr-14	Section 46. “Electrical Characteristics” Table 46-1 “Absolute Maximum Ratings*”: removed junction temperature. Table 46-21 “VDDIO Supply Monitor”: modified min and max values for parameter ACC. Table 46-26 “4/8/12 MHz RC Oscillators Characteristics”: modified conditions for ACC4, ACC8 and ACC12. Modified max values for ACC8 and ACC12. Figure 45-18 “Measurement Setup for Configuration C and D”: added note below figure. Table 46-49 “SAM4C16/8 Typical Current Consumption Values for Backup Mode Configurations C and D”: modified ‘Conditions’ column to VDDIO. Table 46-59 “SAM4C8/16 Test Setup 3 Current Consumption”: modified values for 128-bit Flash Access, Cache Enabled columns.
	Section 50. “SAM4C16/8 Errata Revision A (MRL A) Parts” Added erratum on Flash Memory: “Flash: Incorrect Flash Read May Occur Depending on VDDIO Voltage and Flash Wait State”

Table 53-5. SAM4C Datasheet Rev. 11102C Revision History

Doc. Rev. 11102C	Changes
16-Jan-14	Table 5-2 “Low-power Mode Configuration Summary”: modified notes (4) and (5).
	Section 46. “Electrical Characteristics” Removed section 45.5.17.2 ‘10-bit ADC with Averager’. Table 46-65 “Power Consumption on VDDCORE(1)”: all consumption values modified except AES. Removed section 45.7.6 ‘Low-power Mode Wake-up Time’.