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	512KB (512K 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/atsam4c8cb-aur

1. Configuration Summary

The SAM4C devices differ in memory size, package and features. Table 1-1 summarizes the different device configurations.

Table 1-1. Configuration Summary

Feature	SAM4C32E	SAM4C32C	SAM4C16C	SAM4C8C	SAM4C4C
Flash	2048 Kbytes	2048 Kbytes	1024 Kbytes	512 Kbytes	256 Kbytes
SRAM	256 + 32 + 16 Kbytes		128 + 16 + 8 Kbytes		
Package	LQFP 144	LQFP 100			
Number of PIOs	106	74			
External Bus Interface	16-bit data				
16-bit Timer	6 channels				
16-bit PWM	4 channels				
UART/USART	2/5				
SPI (1)	2/5 + 5				
TWI	2				
10-bit ADC Channels(2)	8				
USB Full Speed	Host + Device	–			
Cryptography	AES, CPKCC, ICM (SHA), TRNG				
Segmented LCD	50 segments × 6 commons				
Anti-Tampering Inputs	4				
Flash Page Size	512 bytes				
Flash Pages	2 × 2048		2048	1024	512
Flash Lock Region Size	8 Kbytes				
Flash Lock Bits	256 (128 + 128)		128	64	32

Notes: 1. 2/5 + 5 = Number of SPI Controllers / Number of Chip Selects + Number of USARTs with SPI mode.
2. One channel is reserved for internal temperature sensor and one channel for VDDBU measurement.

12.4.1.8 Program Status Register

Name: PSR

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI/IT		T

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
ICI/IT						–	ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

The *Program Status Register* (PSR) combines:

- *Application Program Status Register* (APSR)
- *Interrupt Program Status Register* (IPSR)
- *Execution Program Status Register* (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR.

The PSR accesses these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- Read of all the registers using PSR with the MRS instruction
- Write to the APSR N, Z, C, V and Q bits using APSR_nzcvq with the MSR instruction.

The PSR combinations and attributes are:

Name	Access	Combination
PSR	Read/Write(1)(2)	APSR, EPSR, and IPSR
IEPSR	Read-only	EPSR and IPSR
IAPSR	Read/Write(1)	APSR and IPSR
EAPSR	Read/Write(2)	APSR and EPSR

Notes: 1. The processor ignores writes to the IPSR bits.
2. Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

See the instruction descriptions “MRS” and “MSR” for more information about how to access the program status registers.

12.6.8.1 PKHBT and PKHTB

Pack Halfword

Syntax

```
op{cond} {Rd}, Rn, Rm {, LSL #imm}
op{cond} {Rd}, Rn, Rm {, ASR #imm}
```

where:

op — is one of:

PKHBT Pack Halfword, bottom and top with shift.

PKHTB Pack Halfword, top and bottom with shift.

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

Rd — is the destination register.

Rn — is the first operand register

Rm — is the second operand register holding the value to be optionally shifted.

imm — is the shift length. The type of shift length depends on the instruction:
For PKHBT
LSL a left shift with a shift length from 1 to 31, 0 means no shift.
For PKHTB
ASR an arithmetic shift right with a shift length from 1 to 32,
a shift of 32-bits is encoded as 0b00000.

Operation

The PKHBT instruction:

1. Writes the value of the bottom halfword of the first operand to the bottom halfword of the destination register.
2. If shifted, the shifted value of the second operand is written to the top halfword of the destination register.

The PKHTB instruction:

1. Writes the value of the top halfword of the first operand to the top halfword of the destination register.
2. If shifted, the shifted value of the second operand is written to the bottom halfword of the destination register.

Restrictions

Rd must not be SP and must not be PC.

Condition Flags

This instruction does not change the flags.

Examples

```
PKHBT   R3, R4, R5 LSL #0  ; Writes bottom halfword of R4 to bottom halfword of
                           ; R3, writes top halfword of R5, unshifted, to top
                           ; halfword of R3
PKHTB   R4, R0, R2 ASR #1  ; Writes R2 shifted right by 1 bit to bottom halfword
                           ; of R4, and writes top halfword of R0 to top
                           ; halfword of R4.
```

12.8 Nested Vectored Interrupt Controller (NVIC)

This section describes the NVIC and the registers it uses. The NVIC supports:

- Up to 41 interrupts
- A programmable priority level of 0–15 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level detection of interrupt signals
- Dynamic reprioritization of interrupts
- Grouping of priority values into group priority and subpriority fields
- Interrupt tail-chaining
- An external *Non-maskable interrupt* (NMI)

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.

12.8.1 Level-sensitive Interrupts

The processor supports level-sensitive interrupts. A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically, this happens because the ISR accesses the peripheral, causing it to clear the interrupt request.

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts”). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer requires servicing.

12.8.1.1 Hardware and Software Control of Interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is HIGH and the interrupt is not active
- The NVIC detects a rising edge on the interrupt signal
- A software writes to the corresponding interrupt set-pending register bit, see “Interrupt Set-pending Registers”, or to the NVIC_STIR to make an interrupt pending, see “Software Trigger Interrupt Register”.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit.
  For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

If the user disables a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault.

The user can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

When MODE is equal to CMDE, then a new command (strobed on DATA[15:0] signals) is stored in the command register.

Table 23-3. Command Bit Coding

DATA[15:0]	Symbol	Command Executed
0x0011	READ	Read Flash
0x0012	WP	Write Page Flash
0x0022	WPL	Write Page and Lock Flash
0x0032	EWP	Erase Page and Write Page
0x0042	EWPL	Erase Page and Write Page then Lock
0x0013	EA	Erase All
0x0014	SLB	Set Lock Bit
0x0024	CLB	Clear Lock Bit
0x0015	GLB	Get Lock Bit
0x0034	SGPB	Set General Purpose NVM bit
0x0044	CGPB	Clear General Purpose NVM bit
0x0025	GGPB	Get General Purpose NVM bit
0x0054	SSE	Set Security Bit
0x0035	GSE	Get Security Bit
0x001F	WRAM	Write Memory
0x001E	GVE	Get Version

23.3.3 Entering Parallel Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply the supplies as described in Table 23-1.
2. If an external clock is available, apply it to XIN within the VDDCORE POR reset time-out period, as defined in the section “Electrical Characteristics”.
3. Wait for the end of this reset period.
4. Start a read or write handshaking.

23.3.4 Programmer Handshaking

A handshake is defined for read and write operations. When the device is ready to start a new operation (RDY signal set), the programmer starts the handshake by clearing the NCMD signal. The handshaking is completed once the NCMD signal is high and RDY is high.

23.3.4.1 Write Handshaking

For details on the write handshaking sequence, refer to Figure 23-2 and Table 23-4.

25.3 Product Dependencies

25.3.1 Power Management

The Interprocessor Communication module is not continuously clocked. The IPC interface is clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the IPC clock.

25.3.2 Interrupt Line

The IPC module has an interrupt line connected to the Interrupt Controller. Handling interrupts requires programming the Interrupt Controller before configuring the IPC.

25.4 Functional Description

25.4.1 Interrupt Sources

Table 25-1. Peripheral IDs

Instance	ID
IPC0	31
IPC1	39

25.4.1.1 Interrupt Generation

Interrupt sources can be individually generated by writing respectively the IPC_ISCR and IPC_ICCR registers.

25.4.1.2 Interrupt Source Control

Each interrupt source (IRQ0 to IRQ31) can be enabled or disabled by using the command registers: IPC_IECR (Interrupt Enable Command Register) and IPC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling of an instruction. The interrupt mask can be read in the IPC_IMR register. All IPC interrupts can be enabled/disabled, thus configuring the IPC Interrupt mask register. Each pending and unmasked IPC interrupt asserts the IPC output interrupt line. A disabled interrupt does not affect servicing of other interrupts.

25.4.1.3 Interrupt Status

The IPC_IECR and IPC_IDCR registers are used to determine which interrupt sources are active/inhibited to generate an interrupt output. The IPC_IMR register is a status of the interrupt source selection (a result from write into the IPC_IECR and IPC_IDCR registers). The IPC_ISCR and IPC_ICCR registers are used to activate/inhibit interrupt sources. The IPC_IPR register is a status register giving active interrupt sources.

The IPC_ISR register reports which interrupt source(s) is(are) currently asserting an interrupt output. IPC_ISR is basically equivalent to an AND between the IPC_IPR and IPC_IMR registers.

25.5.7 IPC Interrupt Status Register

Name: IPC_ISR

Address: 0x4004C018 (0), 0x48014018 (1)

Access: Read-only

31	30	29	28	27	26	25	24
IRQ31	IRQ30	IRQ29	IRQ28	IRQ27	IRQ26	IRQ25	IRQ24
23	22	21	20	19	18	17	16
IRQ23	IRQ22	IRQ21	IRQ20	IRQ19	IRQ18	IRQ17	IRQ16
15	14	13	12	11	10	9	8
IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8
7	6	5	4	3	2	1	0
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

- **IRQ0-IRQ31: Current Interrupt Identifier**

0: The corresponding interrupt source is not currently asserting the interrupt output.

1: The corresponding interrupt source is currently asserting the interrupt output.

26.9.3 Bus Matrix Priority Registers A For Slaves

Name: MATRIX_PRASx [x=0..8]

Address: 0x400E0280 (0)[0], 0x400E0288 (0)[1], 0x400E0290 (0)[2], 0x400E0298 (0)[3], 0x400E02A0 (0)[4], 0x400E02A8 (0)[5], 0x400E02B0 (0)[6], 0x400E02B8 (0)[7], 0x400E02BC (0)[8], 0x48010080 (1)[0], 0x48010088 (1)[1], 0x48010090 (1)[2], 0x48010098 (1)[3], 0x480100A0 (1)[4], 0x480100A8 (1)[5], 0x480100B0 (1)[6], 0x480100B8 (1)[7], 0x480100BC (1)[8]

Access: Read/Write

31	30	29	28	27	26	25	24
–		M7PR		–		M6PR	

23	22	21	20	19	18	17	16
–		M5PR		–		M4PR	

15	14	13	12	11	10	9	8
–		M3PR		–		M2PR	

7	6	5	4	3	2	1	0
–		M1PR		–		M0PR	

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

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-robin arbitration is used in the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used in intermediate priority pools (MxPR = 1) and (MxPR = 2).

See Section 26.7.2 “Arbitration Priority Scheme” for details.

27.16 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 27-9. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 27-9, “CS_number” denotes the chip select number. 16 bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing any one of the SMC_MODE registers.

Table 27-9. Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x01010101
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00030003
0x10 x CS_number + 0x0C	SMC MODE Register	SMC_MODE	Read/Write	0x10000003
0x80	SMC OCMS MODE Register	SMC_OCMS	Read/Write	0x00000000
0x84	SMC OCMS KEY1 Register	SMC_KEY1	Write Once	0x00000000
0x88	SMC OCMS KEY2 Register	SMC_KEY2	Write Once	0x00000000
0xE4	SMC Write Protection Mode Register	SMC_WPMR	Read/Write	0x00000000
0xE8	SMC Write Protection Status Register	SMC_WPSR	Read-only	0x00000000
0xEC-0xFC	Reserved	–	–	–

Notes: 1. All unlisted offset values are considered as ‘reserved’.

32.6.48 PIO I/O Drive Register

Name: PIO_DRIVER

Address: 0x400E0F18 (PIOA), 0x400E1118 (PIOB), 0x4800C118 (PIOC), 0x400E1318 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
LINE31	LINE30	LINE29	LINE28	LINE27	LINE26	LINE25	LINE24
23	22	21	20	19	18	17	16
LINE23	LINE22	LINE21	LINE20	LINE19	LINE18	LINE17	LINE16
15	14	13	12	11	10	9	8
LINE15	LINE14	LINE13	LINE12	LINE11	LINE10	LINE9	LINE8
7	6	5	4	3	2	1	0
LINE7	LINE6	LINE5	LINE4	LINE3	LINE2	LINE1	LINE0

- **LINEx [x=0..31]: Drive of PIO Line x**

Value	Name	Description
0	LOW_DRIVE	Lowest drive
1	HIGH_DRIVE	Highest drive

33.8.1 SPI Control Register

Name: SPI_CR

Address: 0x40008000 (0), 0x48000000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
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	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI Enable**

0: No effect.

1: Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0: No effect.

1: Disables the SPI.

All pins are set in Input mode after completion of the transmission in progress, if any.

If a transfer is in progress when SPIDIS is set, the SPI completes the transmission of the shifter register and does not start any new transfer, even if the SPI_THR is loaded.

Note: If both SPIEN and SPIDIS are equal to one when the SPI_CR is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0: No effect.

1: Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in Slave mode after software reset.

PDC channels are not affected by software reset.

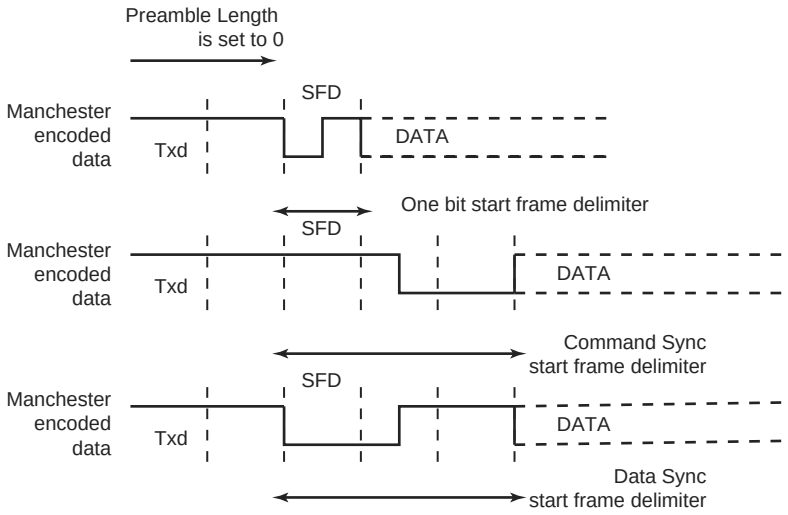
- **LASTXFER: Last Transfer**

0: No effect.

1: The current NPCS is de-asserted after the character written in TD has been transferred. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Refer to Section 33.7.3.5 “Peripheral Selection” for more details.

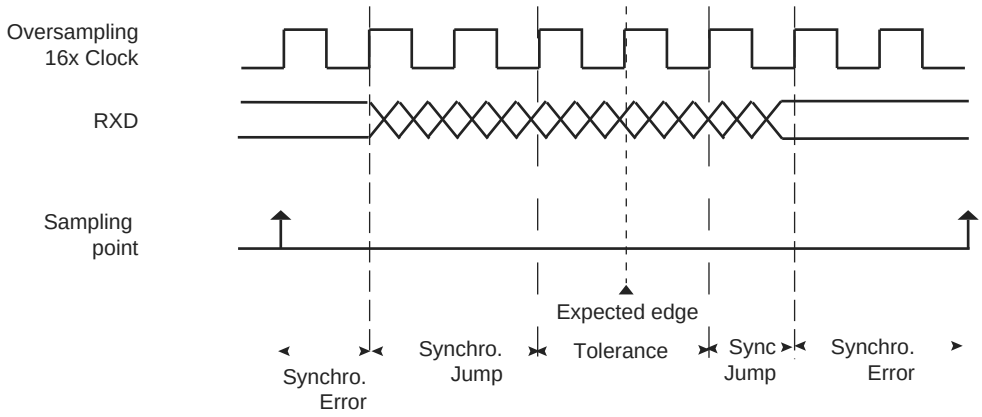
Figure 36-9. Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 36-10. Bit Resynchronization



36.6.3.3 Asynchronous Receiver

If the USART is programmed in Asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the baud rate clock, depending on the OVER bit in the US_MR.

The receiver samples the RXD line. If the line is sampled during one half of a bit time to 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16 (OVER = 0), a start is detected at the eighth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 16 oversampling clock cycles. If the oversampling is 8 (OVER = 1), a start bit is detected at the fourth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 8 oversampling clock cycles.

36.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 36-14. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	0x0
0x0018	Receive Holding Register	US_RHR	Read-only	0x0
0x001C	Transmit Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x002C–0x003C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	0x0
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x0050	Manchester Configuration Register	US_MAN	Read/Write	0x30011004
0x0054–0x005C	Reserved	–	–	–
0x0060–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	US_WPMR	Read/Write	0x0
0x00E8	Write Protection Status Register	US_WPSR	Read-only	0x0
0x00EC–0x00FC	Reserved	–	–	–
0x0100–0x0128	Reserved for PDC Registers	–	–	–

36.7.11 USART Channel Status Register

Name: US_CSR

Address: 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2), 0x40030014 (3), 0x40034014 (4)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANERR
23	22	21	20	19	18	17	16
CTS	–	–	–	CTSIC	–	–	–
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 36.7.12 ”USART Channel Status Register (SPI_MODE)”.

- **RXRDY: Receiver Ready (cleared by reading US_RHR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break (cleared by writing a one to bit US_CR.RSTSTA)**

0: No break received or end of break detected since the last RSTSTA.

1: Break received or end of break detected since the last RSTSTA.

- **ENDRX: End of RX Buffer (cleared by writing US_RCR or US_RNCR)**

0: The Receive Counter Register has not reached 0 since the last write in US_RCR or US_RNCR(1).

1: The Receive Counter Register has reached 0 since the last write in US_RCR or US_RNCR(1).

- **ENDTX: End of TX Buffer (cleared by writing US_TCR or US_TNCR)**

0: The Transmit Counter Register has not reached 0 since the last write in US_TCR or US_TNCR(1).

1: The Transmit Counter Register has reached 0 since the last write in US_TCR or US_TNCR(1).

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

40.6.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the LOWRES bit in ADC_MR. By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the LOWRES bit, the ADC switches to the lowest resolution and the conversion results can be read in the lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding Channel Data register (ADC_CDR) and of the LDATA field in the Last Converted Data register (ADC_LCDR) read 0.

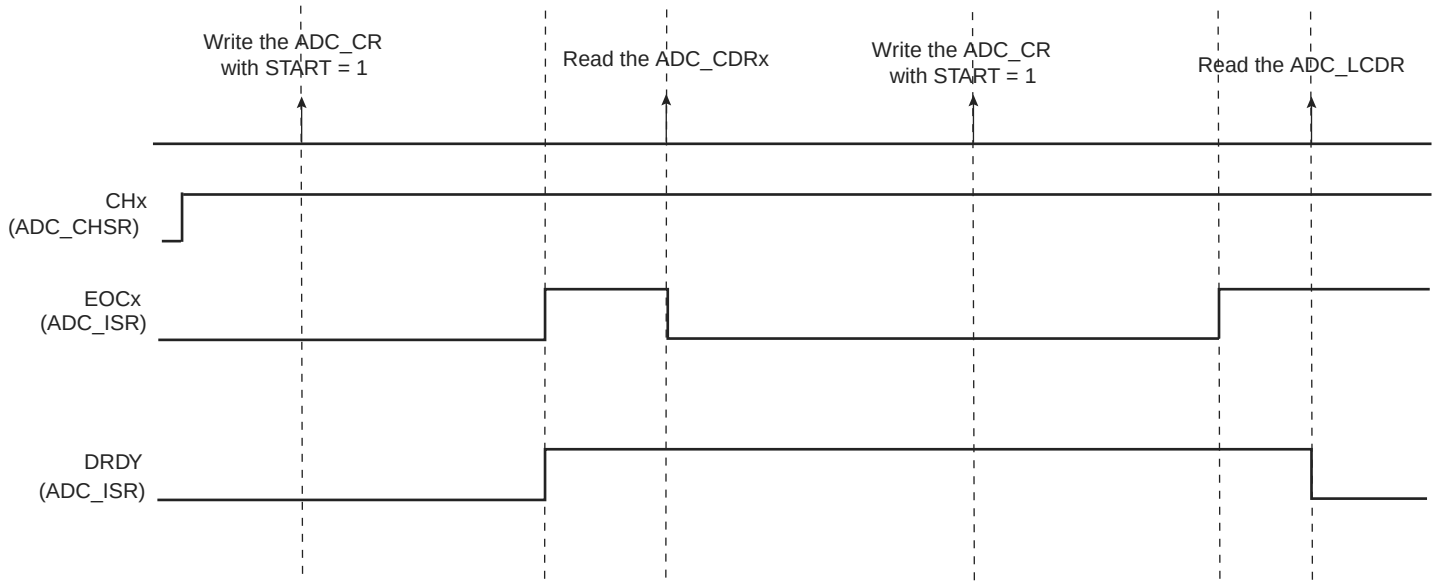
40.6.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in ADC_CDRx of the current channel and in ADC_LCDR. By setting the TAG bit in the Extended Mode register (ADC_EMR), ADC_LCDR presents the channel number associated with the last converted data in the CHNB field.

The EOCx and DRDY bits in the Interrupt Status register (ADC_ISR) are set. In the case of a connected PDC channel, DRDY rising triggers a data request. In any case, both EOC and DRDY can trigger an interrupt.

Reading one ADC_CDRx clears the corresponding EOCx bit. Reading ADC_LCDR clears the DRDY bit.

Figure 40-3. EOCx and DRDY Flag Behavior



If ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVREx) flag is set in the Overrun Status register (ADC_OVER).

New data converted when DRDY is high sets the GOVRE bit in ADC_ISR.

The OVREx flag is automatically cleared when ADC_OVER is read, and GOVRE flag is automatically cleared when ADC_ISR is read.

Figure 41-4. PDC Transfer with AES_MR.LOD = 1

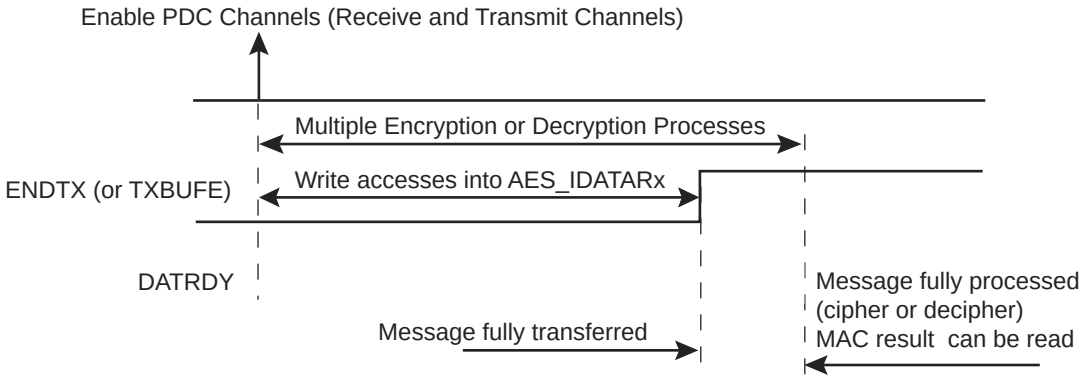


Table 41-4 summarizes the different cases.

Table 41-4. Last Output Data Mode Behavior versus Start Modes

Sequence	Manual and Auto Modes		PDC Mode	
	AES_MR.LOD = 0	AES_MR.LOD = 1	AES_MR.LOD = 0	AES_MR.LOD = 1
DATRDY Flag Clearing Condition(1)	At least one AES_ODATAR must be read	At least one AES_IDATAR must be written	Not used	Managed by the PDC
End of Encryption/Decryption Notification	DATRDY	DATRDY	ENDRX (or RXBUFF)	ENDTX (or TXBUFE) then DATRDY
Encrypted/Decrypted Data Result Location	In the AES_ODATARx	In the AES_ODATARx	At the address specified in the AES_RPR	In the AES_ODATARx

Note: 1. Depending on the mode, there are other ways of clearing the DATRDY flag. See Section 41.5.6 “AES Interrupt Status Register”.

Warning: In PDC mode, reading the AES_ODATARx before the last data transfer may lead to unpredictable results.

41.4.6 Galois/Counter Mode (GCM)

41.4.6.1 Description

GCM comprises the AES engine in CTR mode along with a universal hash function (GHASH engine) that is defined over a binary Galois field to produce a message authentication tag (the AES CTR engine and the GHASH engine are depicted in Figure 41-5).

The GHASH engine processes data packets after the AES operation. GCM provides assurance of the confidentiality of data through the AES Counter mode of operation for encryption. Authenticity of the confidential data is assured through the GHASH engine. GCM can also provide assurance of data that is not encrypted. Refer to the *NIST Special Publication 800-38D* for more complete information.

GCM can be used with or without the PDC master. Messages may be processed as a single complete packet of data or they may be broken into multiple packets of data over time.

GCM processing is computed on 128-bit input data fields. There is no support for unaligned data. The AES key length can be whatever length is supported by the AES module.

The recommended programming procedure when using PDC is described in Section 41.4.6.3 “GCM Processing”.

- **ECIEN: End Bit Condition Interrupt (Default Enabled)**

0: The ICM_ISR REC[*i*] flag is set when the descriptor having the EOM bit set is processed.

1: The ICM_ISR REC[*i*] flag remains cleared even if the setting condition is met.

- **SUIEN: Monitoring Status Updated Condition Interrupt (Default Enabled)**

0: The ICM_ISR RSU[*i*] flag is set when the corresponding descriptor is loaded from memory to ICM.

1: The ICM_ISR RSU[*i*] flag remains cleared even if the setting condition is met.

- **PROCDLY: Processing Delay**

Value	Name	Description
0	SHORTEST	SHA processing runtime is the shortest one
1	LONGEST	SHA processing runtime is the longest one

When SHA1 algorithm is processed, the runtime period is either 85 or 209 clock cycles.

When SHA256 or SHA224 algorithm is processed, the runtime period is either 72 or 194 clock cycles.

- **ALGO: SHA Algorithm**

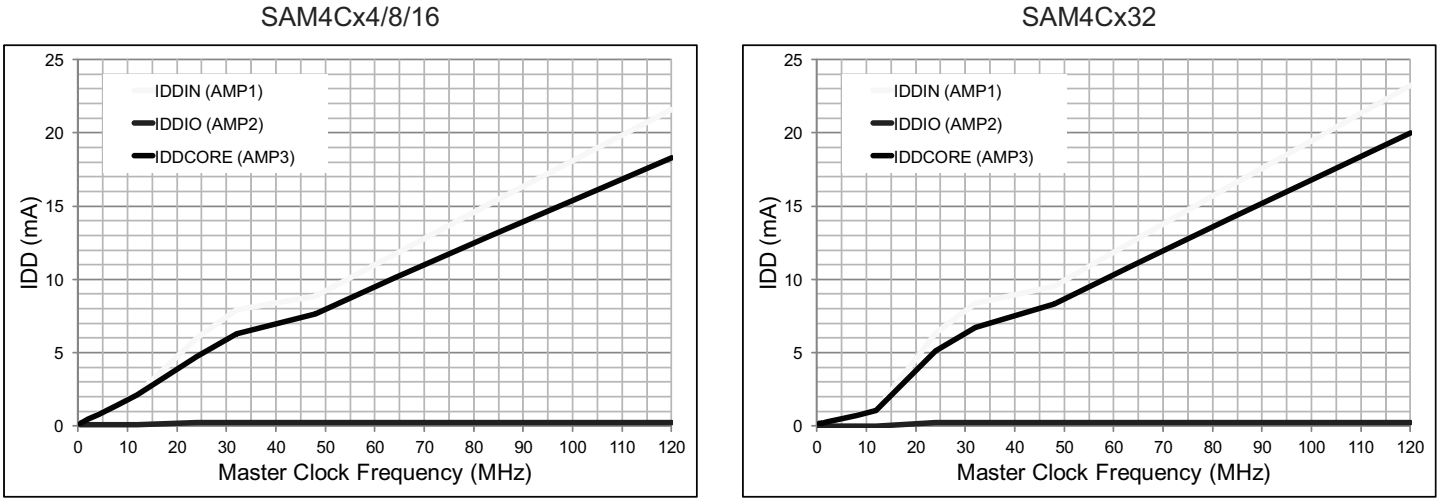
Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
4	SHA224	SHA224 algorithm processed

Values which are not listed in the table must be considered as “reserved”.

- **MRPROT: Memory Region AHB Protection**

This field indicates the value of HPROT AHB signal when the ICM reads the memory region to be monitored (refer to HPROT signal encoding available on www.arm.com).

Figure 46-30. Typical Current Consumption in Active Mode (Test Setup 4)



46.7.4.5 Test Setup 5: DSP Application (Five Cascaded 4th Order Biquad Filters) from ARM CMSIS DSP Library

- Application running on Core 0 (CM4P0) out of Flash in 128-bit Access mode with and without cache enabled. Cache is enabled above 0 WS.
- Sub-system 1 Master Clock (CPBMCK) and Core Clock (CPHCLK) stopped and in reset.
- VDDIO = VDDIN = 3V

Table 46-65. SAM4C4/8/16 Test Setup 5 Current Consumption

Clock (MHz)	128-bit Flash Access						64-bit Flash Access						Unit
	Cache Enabled			Cache Disabled			Cache Enabled			Cache Disabled			
	IDD_IN (AMP1)	IDD_I0 (AMP2)	IDD_CORE (AMP3)	IDD_IN (AMP1)	IDD_I0 (AMP2)	IDD_CORE (AMP3)	IDD_IN (AMP1)	IDD_I0 (AMP2)	IDD_CORE (AMP3)	IDD_IN (AMP1)	IDD_I0 (AMP2)	IDD_CORE (AMP3)	
120	23.2	0.31	19.9	26.3	2.1	23.1	23.2	0.31	19.9	21.2	1.7	18.0	mA
100	19.3	0.31	16.6	23.7	2.0	21.0	19.3	0.31	16.6	18.9	1.7	16.2	
84	16.3	0.31	14.1	21.2	1.9	19.0	16.3	0.31	14.1	17.5	1.7	15.3	
64	12.9	0.31	11.2	17.2	1.8	15.5	12.9	0.31	11.2	14.8	1.6	13.1	
48	9.9	0.31	8.6	13.9	1.6	12.7	9.9	0.31	8.6	12.3	1.6	11.1	
32	7.5	0.31	5.8	10.6	1.4	9.0	7.5	0.31	5.8	9.9	1.4	8.2	
24	5.7	0.31	4.4	8.7	1.2	7.5	5.7	0.31	4.4	8.1	1.3	6.9	
12	2.6	0.08	2.6	4.0	0.82	3.9	2.6	0.08	2.6	3.5	0.8	3.4	
8	1.7	0.08	1.7	2.7	0.70	2.7	1.7	0.08	1.7	2.4	0.8	2.4	
4	0.89	0.08	0.88	1.6	0.51	1.6	0.89	0.08	0.88	1.3	0.7	1.3	
2	0.56	0.08	0.55	0.96	0.39	0.95	0.56	0.08	0.55	0.78	0.54	0.76	
1	0.55	0.08	0.54	0.67	0.20	0.66	0.55	0.08	0.54	0.68	0.37	0.67	