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	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/atsam4c8ca-au

Each sector is organized in pages of 512 bytes.

For sector 0:

- The small sector 0 has 16 pages of 512 bytes, 8 Kbytes in total
- The small sector 1 has 16 pages of 512 bytes, 8 Kbytes in total
- The larger sector has 96 pages of 512 bytes, 48 Kbytes in total

From sector 1 to n:

The rest of the array is composed of 64-Kbyte sectors where each sector comprises 128 pages of 512 bytes. Refer to Figure 8-3 below.

Figure 8-3. Flash Sector Organization

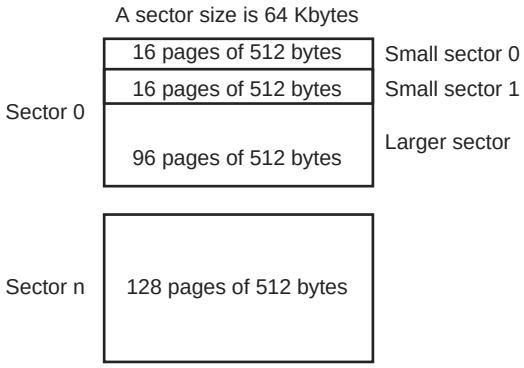


Table 8-1. SAM4C Flash Size

Device	Flash (Kbytes)
SAM4C4	256
SAM4C8	512
SAM4C16	1024
SAM4C32	2048 (2 × 1024)

Use a DSB followed by an ISB instruction or exception return to ensure that the new MPU configuration is used by subsequent instructions.

12.4.2.5 Bit-banding

A bit-band region maps each word in a *bit-band alias* region to a single bit in the *bit-band region*. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions.

The memory map has two 32 MB alias regions that map to two 1 MB bit-band regions:

- Accesses to the 32 MB SRAM alias region map to the 1 MB SRAM bit-band region, as shown in Table 12-6.
- Accesses to the 32 MB peripheral alias region map to the 1 MB peripheral bit-band region, as shown in Table 12-7.

Table 12-6. SRAM Memory Bit-banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x20000000–0x200FFFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit-addressable through bit-band alias.
0x22000000–0x23FFFFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit-band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 12-7. Peripheral Memory Bit-banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x40000000–0x400FFFFF	Peripheral bit-band alias	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit-addressable through bit-band alias.
0x42000000–0x43FFFFFF	Peripheral bit-band region	Data accesses to this region are remapped to bit-band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

Notes:
1. A word access to the SRAM or peripheral bit-band alias regions map to a single bit in the SRAM or peripheral bit-band region.
2. Bit-band accesses can use byte, halfword, or word transfers. The bit-band transfer size matches the transfer size of the instruction making the bit-band access.

The following formula shows how the alias region maps onto the bit-band region:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)
bit_word_addr = bit_band_base + bit_word_offset
```

where:

- `Bit_word_offset` is the position of the target bit in the bit-band memory region.
- `Bit_word_addr` is the address of the word in the alias memory region that maps to the targeted bit.
- `Bit_band_base` is the starting address of the alias region.
- `Byte_offset` is the number of the byte in the bit-band region that contains the targeted bit.
- `Bit_number` is the bit position, 0–7, of the targeted bit.

Figure 12-4 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FFFFE0 maps to bit[0] of the bit-band byte at 0x200FFFFF: 0x23FFFFE0 = 0x22000000 + (0xFFFFF*32) + (0*4).
- The alias word at 0x23FFFFFC maps to bit[7] of the bit-band byte at 0x200FFFFF: 0x23FFFFFC = 0x22000000 + (0xFFFFF*32) + (7*4).

12.6.4.5 LDR, PC-relative

Load register from memory.

Syntax

```
LDR{type}{cond} Rt, label
LDRD{cond} Rt, Rt2, label          ; Load two words
```

where:

type is one of:

- B unsigned byte, zero extend to 32 bits.
- SB signed byte, sign extend to 32 bits.
- H unsigned halfword, zero extend to 32 bits.
- SH signed halfword, sign extend to 32 bits.
- - omit, for word.

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

Rt is the register to load or store.

Rt2 is the second register to load or store.

label is a PC-relative expression. See “PC-relative Expressions”.

Operation

LDR loads a register with a value from a PC-relative memory address. The memory address is specified by a label or by an offset from the PC.

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”.

label must be within a limited range of the current instruction. The table below shows the possible offsets between *label* and the PC.

Table 12-19. Offset Ranges

Instruction Type	Offset Range
Word, halfword, signed halfword, byte, signed byte	-4095 to 4095
Two words	-1020 to 1020

The user might have to use the .W suffix to get the maximum offset range. See “Instruction Width Selection”.

Restrictions

In these instructions:

- *Rt* can be SP or PC only for word loads
- *Rt2* must not be SP and must not be PC
- *Rt* must be different from *Rt2.*

13.4 Cross Triggering Debug Events

Cross Triggering (CT) as shown in Figure 13-2 is an Atmel module that enables two cores to send and receive debug events to and from each other. This module is used to debug two applications at the same time (one application running on each core).

CT enables core 0 (or 1) to trigger a debug event (halt) to core 1 (or 0) to enter Debug mode. The debug event can be sent when the core 0 (or 1) enters Debug mode (such as breakpoint) or at run-time.

Once core 0 (or 1) gets out of Debug mode, it releases core 1 (0) from Debug mode as well.

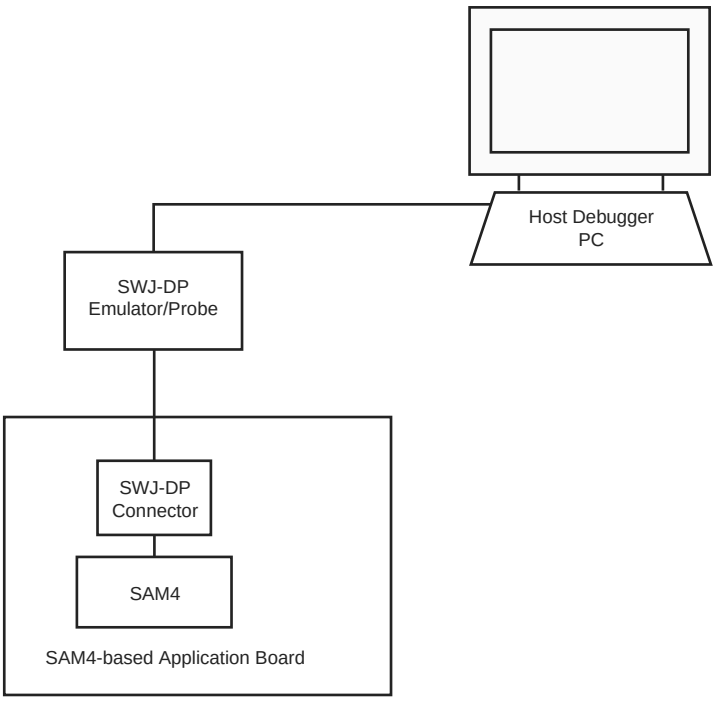
The Cross Triggering configuration is located in the Special Function Register in the Matrix User Interface.

13.5 Application Examples

13.5.1 Debug Environment

Figure 13-3 shows a complete debug environment example. The SWJ-DP interface is used for standard debugging functions, such as downloading code and single-stepping through the program, as well as viewing core and peripheral registers.

Figure 13-3. Application Debug Environment Example



13.5.2 Test Environment

Figure 13-4 shows an example of a test environment (JTAG Boundary scan). Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

16.5.2 Real-time Timer Alarm Register

Name: RTT_AR

Address: 0x400E1434

Access: Read/Write

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

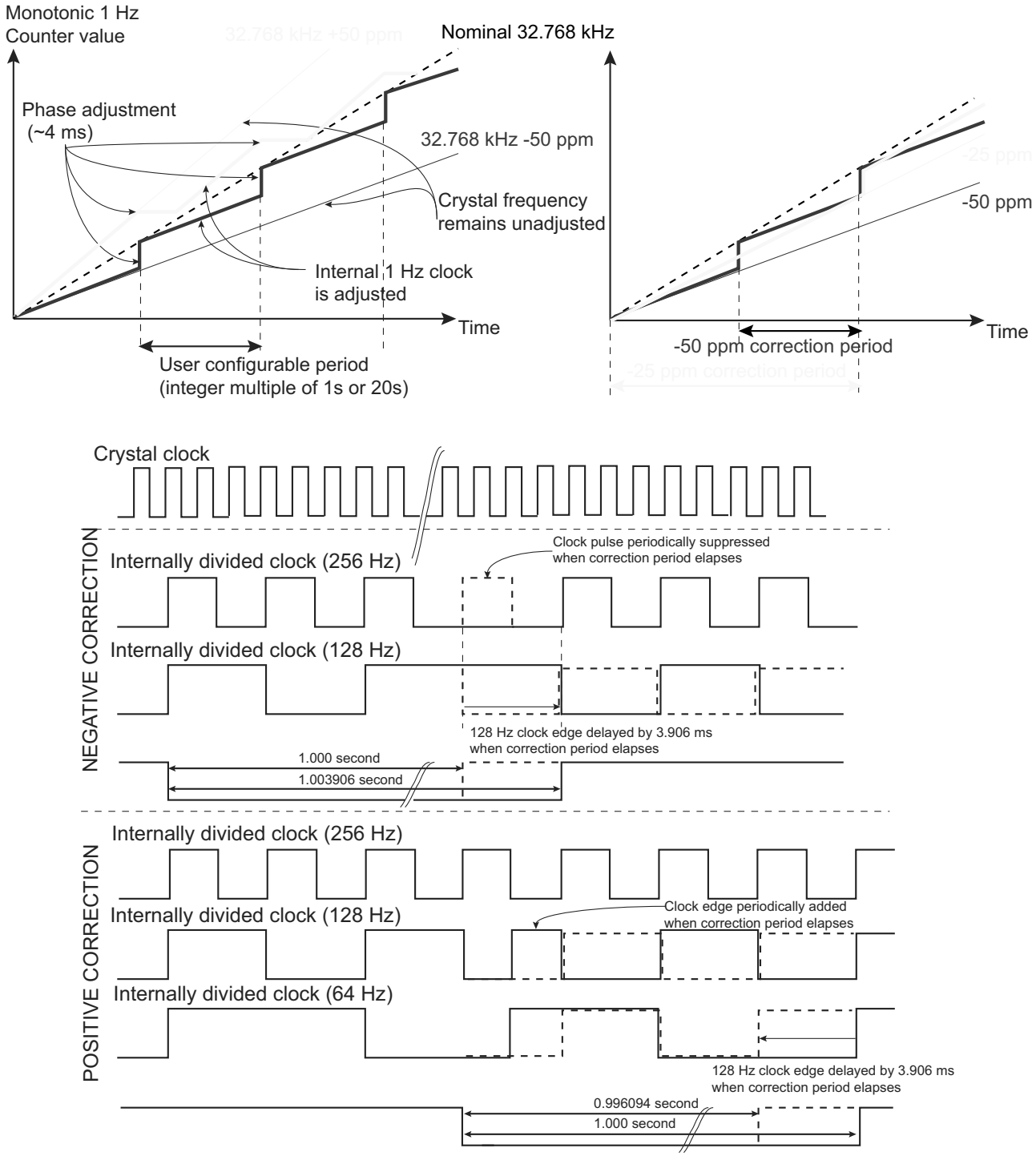
7	6	5	4	3	2	1	0
ALMV							

- **ALMV: Alarm Value**

When the CRTV value in RTT_VR equals the ALMV field, the ALMS flag is set in RTT_SR. As soon as the ALMS flag rises, the CRTV value equals ALMV+1 (refer to Figure 16-2).

Note: The alarm interrupt must be disabled (ALMIEN must be cleared in RTT_MR) when writing a new ALMV value.

Figure 17-5. Calibration Circuitry Waveforms



The inaccuracy of a crystal oscillator at typical room temperature (±20 ppm at 20–25 °C) can be compensated if a reference clock/signal is used to measure such inaccuracy. This kind of calibration operation can be set up during the final product manufacturing by means of measurement equipment embedding such a reference clock. The correction of value must be programmed into the (RTC_MR), and this value is kept as long as the circuitry is powered (backup area). Removing the backup power supply cancels this calibration. This room temperature calibration can be further processed by means of the networking capability of the target application.

To ease the comparison of the inherent crystal accuracy with the reference clock/signal during manufacturing, an internal prescaled 32.768 kHz clock derivative signal can be assigned to drive RTC output. To accommodate the measure, several clock frequencies can be selected among 1 Hz, 32 Hz, 64 Hz, 512 Hz.

The clock calibration correction drives the internal RTC counters but can also be observed in the RTC output when one of the following three frequencies 1 Hz, 32 Hz or 64 Hz is configured. The correction is not visible in the RTC output if 512 Hz frequency is configured.

In any event, this adjustment does not take into account the temperature variation.

The frequency drift (up to -200 ppm) due to temperature variation can be compensated using a reference time if the application can access such a reference. If a reference time cannot be used, a temperature sensor can be placed close to the crystal oscillator in order to get the operating temperature of the crystal oscillator. Once obtained, the temperature may be converted using a lookup table (describing the accuracy/temperature curve of the crystal oscillator used) and RTC_MR configured accordingly. The calibration can be performed on-the-fly. This adjustment method is not based on a measurement of the crystal frequency/drift and therefore can be improved by means of the networking capability of the target application.

If no crystal frequency adjustment has been done during manufacturing, it is still possible to do it. In the case where a reference time of the day can be obtained through LAN/WAN network, it is possible to calculate the drift of the application crystal oscillator by comparing the values read on RTC Time Register (RTC_TIMR) and programming the HIGHPPM and CORRECTION fields on RTC_MR according to the difference measured between the reference time and those of RTC_TIMR.

17.5.8 Waveform Generation

Waveforms can be generated by the RTC in order to take advantage of the RTC inherent prescalers while the RTC is the only powered circuitry (Low-power mode of operation, Backup mode) or in any active mode. Going into Backup or Low-power operating modes does not affect the waveform generation outputs.

The RTC output (RTCOUT0) has a source driver selected among seven possibilities.

The first selection choice sticks the associated output at 0 (This is the reset value and it can be used at any time to disable the waveform generation).

Selection choices 1 to 4 respectively select 1 Hz, 32 Hz, 64 Hz and 512 Hz.

32 Hz or 64 Hz can drive, for example, a TN LCD backplane signal while 1 Hz can be used to drive a blinking character like “:” for basic time display (hour, minute) on TN LCDs.

Selection choice 5 provides a toggling signal when the RTC alarm is reached.

Selection choice 6 provides a copy of the alarm flag, so the associated output is set high (logical 1) when an alarm occurs and immediately cleared when software clears the alarm interrupt source.

Selection choice 7 provides a 1 Hz periodic high pulse of 15 µs duration that can be used to drive external devices for power consumption reduction or any other purpose.

PIO line associated to RTC output is automatically selecting these waveforms as soon as RTC_MR corresponding fields OUT0 differ from 0.

Figure 27-16. Early Read Wait State: Write with No Hold Followed by Read with No Setup

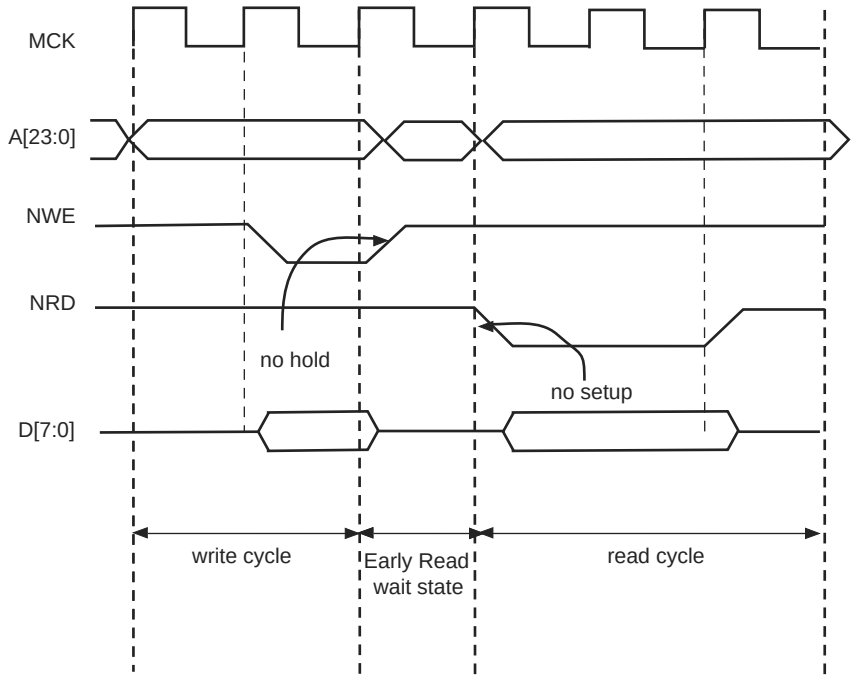
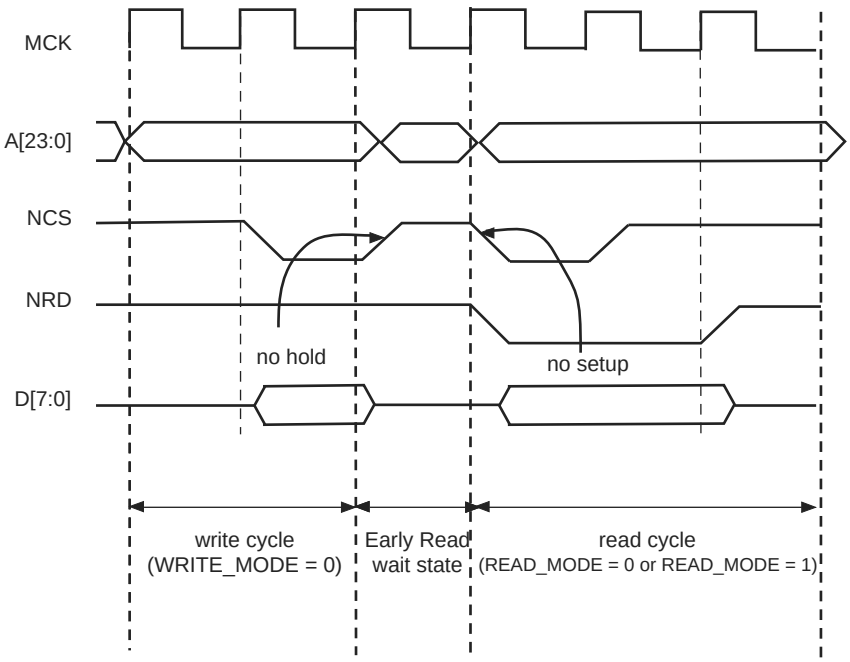


Figure 27-17. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup



28. Peripheral DMA Controller (PDC)

28.1 Description

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals and the target memories. The link between the PDC and a serial peripheral is operated by the AHB to APB bridge.

The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono-directional channels (receive-only or transmit-only) contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for the current transfer and one set (pointer, counter) for the next transfer. The bidirectional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by the current transmit, next transmit, current receive and next receive.

Using the PDC decreases processor overhead by reducing its intervention during the transfer. This lowers significantly the number of clock cycles required for a data transfer, improving microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.

28.2 Embedded Characteristics

- Performs Transfers to/from APB Communication Serial Peripherals
- Supports Half-duplex and Full-duplex Peripherals

30.19.8 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Address: 0x400E0424

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

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

- **MAINF: Main Clock Frequency**

Gives the number of main clock cycles within 16 slow clock periods. To calculate the frequency of the measured clock: $f_{MAINCK} = (MAINF \times f_{SLCK}) / 16$

where frequency is in MHz.

- **MAINFRDY: Main Clock Frequency Measure Ready**

0: MAINF value is not valid or the measured oscillator is disabled or a measure has just been started by means of RCMEAS.

1: The measured oscillator has been enabled previously and MAINF value is available.

Note: To ensure that a correct value is read on the MAINF field, the MAINFRDY flag must be read at 1 then another read access must be performed on the register to get a stable value on the MAINF field.

- **RCMEAS: Restart Main Clock Source Frequency Measure (write-only)**

0: No effect.

1: Restarts measuring of the frequency of the main clock source. MAINF will carry the new frequency as soon as a low to high transition occurs on the MAINFRDY flag.

The measure is performed on the main frequency (i.e. not limited to RC oscillator only), but if the main clock frequency source is the 3 to 20 MHz crystal oscillator, the restart of measuring is not needed because of the well known stability of crystal oscillators.

32.6.43 PIO Rising Edge/High-Level Select Register

Name: PIO_REHLSR

Address: 0x400E0ED4 (PIOA), 0x400E10D4 (PIOB), 0x4800C0D4 (PIOC), 0x400E12D4 (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: Rising Edge/High-Level Interrupt Selection**

0: No effect.

1: The interrupt source is set to a rising edge detection or high-level detection event, depending on PIO_ELSR.

- **LLB: Local Loopback Enable**

0: Local loopback path disabled.

1: Local loopback path enabled.

LLB controls the local loopback on the data shift register for testing in Master mode only (MISO is internally connected on MOSI).

- **PCS: Peripheral Chip Select**

This field is only used if fixed peripheral select is active (PS = 0).

If SPI_MR.PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don’t care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

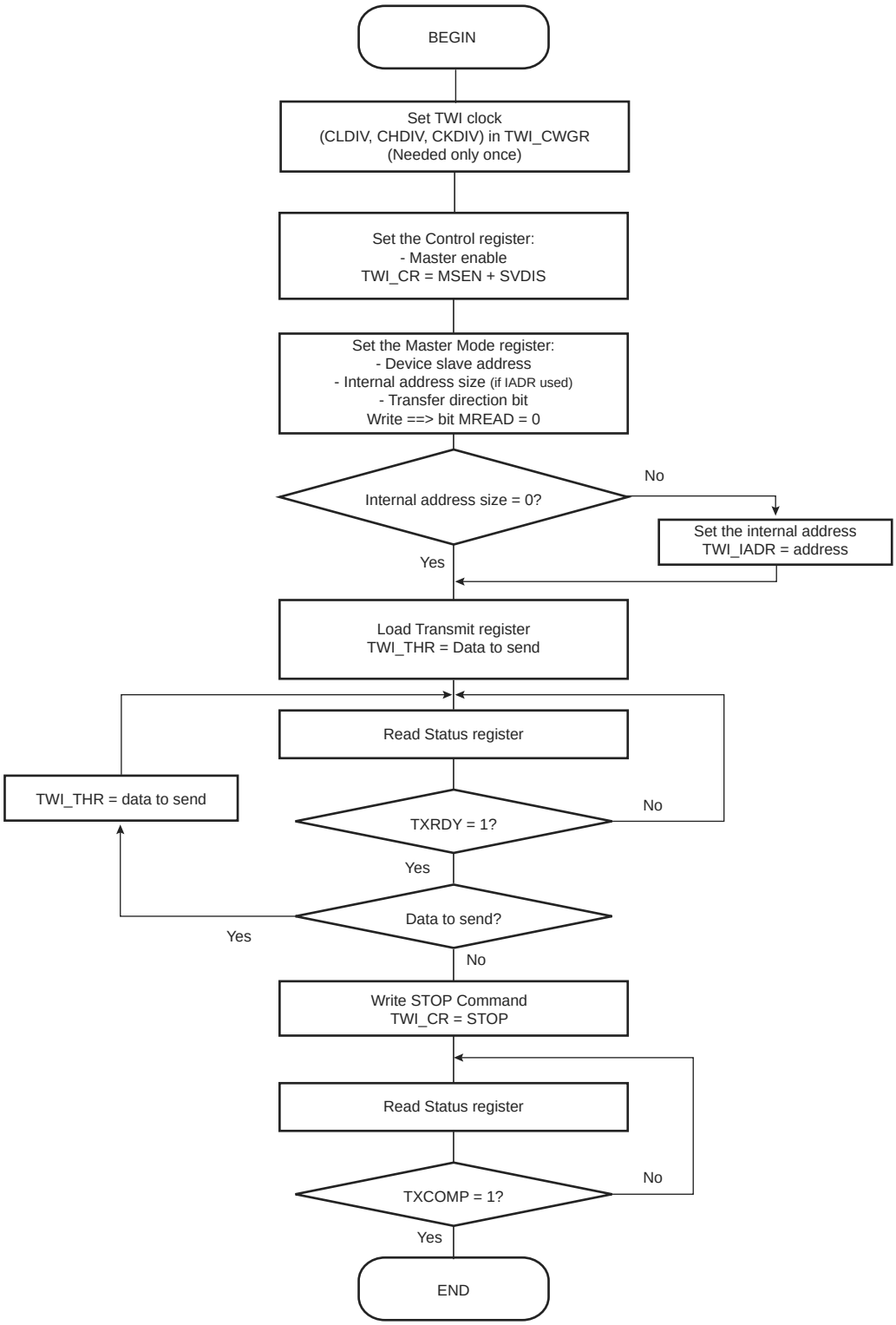
This field defines the delay between the inactivation and the activation of NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is lower than 6, six peripheral clock periods are inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{\text{DLYBCS}}{f_{\text{peripheral clock}}}$$

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



35.6.4 UART Interrupt Disable Register

Name: UART_IDR

Address: 0x400E060C (0), 0x4800400C (1)

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
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

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

0: No effect.

1: Disables the corresponding interrupt.

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**

- **TXBUFE: Transmission Buffer Empty**

0: The buffer empty signal from the transmitter PDC channel is inactive.

1: The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0: The buffer full signal from the receiver PDC channel is inactive.

1: The buffer full signal from the receiver PDC channel is active.

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

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

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

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out (cleared by writing a one to bit US_CR.STTTO)**

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

- **TXEMPTY: Transmitter Empty (cleared by writing US_THR)**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **ITER: Max Number of Repetitions Reached (cleared by writing a one to bit US_CR.RSTIT)**

0: Maximum number of repetitions has not been reached since the last RSTIT.

1: Maximum number of repetitions has been reached since the last RSTIT.

- **TXBUFE: TX Buffer Empty (cleared by writing US_TCR or US_TNCR)**

0: US_TCR or US_TNCR have a value other than 0[(1)].

1: Both US_TCR and US_TNCR have a value of 0[(1)].

- **RXBUFF: RX Buffer Full (cleared by writing US_RCR or US_RNCR)**

0: US_RCR or US_RNCR have a value other than 0[(1)].

1: Both US_RCR and US_RNCR have a value of 0[(1)].

Note: 1. US_RCR, US_RNCR, US_TCR and US_TNCR are PDC registers.

- **NACK: Non Acknowledge Interrupt (cleared by writing a one to bit US_CR.RSTNACK)**

0: Non acknowledge has not been detected since the last RSTNACK.

1: At least one non acknowledge has been detected since the last RSTNACK.

- **CTSIC: Clear to Send Input Change Flag (cleared on read)**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **CTS: Image of CTS Input**

0: CTS input is driven low.

1: CTS input is driven high.

- **MANERR: Manchester Error (cleared by writing a one to the bit US_CR.RSTSTA)**

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

- **ETRGS: External Trigger Status (cleared on read)**

0: External trigger has not occurred since the last read of the Status Register.

1: External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0: Clock is disabled.

1: Clock is enabled.

- **MTIOA: TIOA Mirror**

0: TIOA is low. If TC_CMRx.WAVE = 0, this means that TIOA pin is low. If TC_CMRx.WAVE = 1, this means that TIOA is driven low.

1: TIOA is high. If TC_CMRx.WAVE = 0, this means that TIOA pin is high. If TC_CMRx.WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0: TIOB is low. If TC_CMRx.WAVE = 0, this means that TIOB pin is low. If TC_CMRx.WAVE = 1, this means that TIOB is driven low.

1: TIOB is high. If TC_CMRx.WAVE = 0, this means that TIOB pin is high. If TC_CMRx.WAVE = 1, this means that TIOB is driven high.

39.8.6 SLCDC Interrupt Enable Register

Name: SLCDC_IER

Address: 0x4003C020

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
–	–	–	–	–	DIS	–	ENDFRAME

- **ENDFRAME: End of Frame Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **DIS: SLCDC Disable Completion Interrupt Enable**

0: No effect.

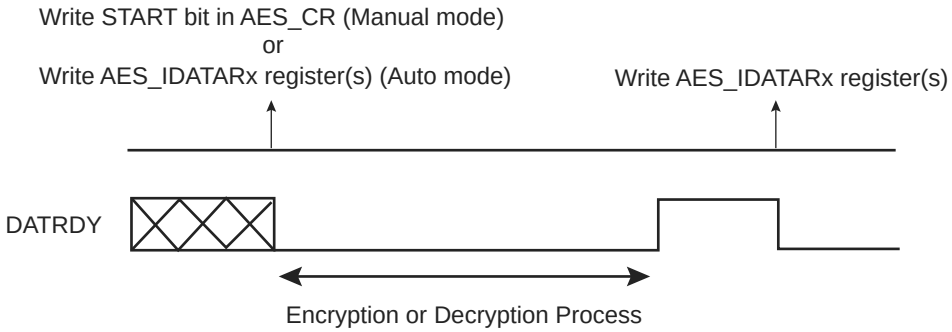
1: Enables the corresponding interrupt.

If AES_MR.LOD = 1

This mode is optimized to process AES CPC-MAC operating mode.

The DATRDY flag is cleared when at least one AES_IDATAR is written (see Figure 41-2). No more AES_ODATAR reads are necessary between consecutive encryptions/decryptions.

Figure 41-2. Manual and Auto Modes with AES_MR.LOD = 1



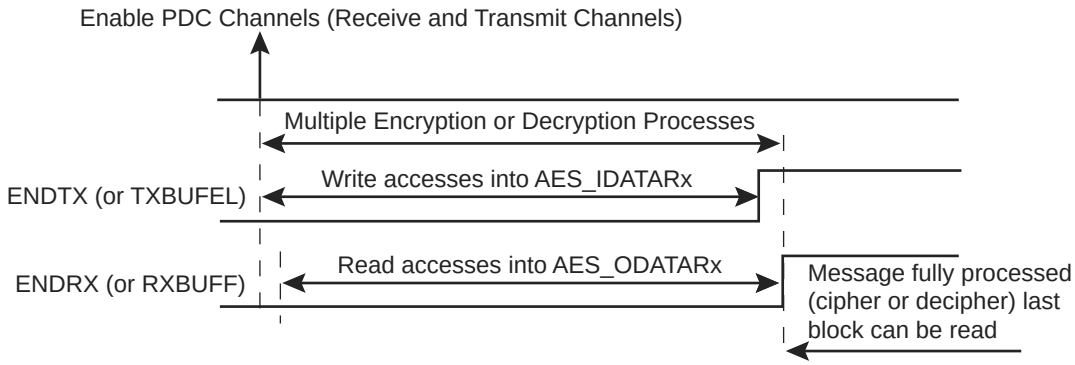
41.4.5.2 PDC Mode

If AES_MR.LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where AES_MR.LOD = 1 mode is recommended.

The end of the encryption/decryption is indicated when the ENDRX (or RXBUFF) flag is raised (see Figure 41-3).

Figure 41-3. PDC Transfer with AES_MR.LOD = 0



If AES_MR.LOD = 1

This mode is optimized to process AES CBC-MAC operating mode.

The user must first wait for the ENDTX (or TXBUFE) flag to be raised, then for DATRDY to ensure that the encryption/decryption is completed (see Figure 41-4).

In this case, no receive buffers are required.

The output data are only available on the AES_ODATARx.

Figure 45-17. Example of an OUT Endpoint with one Data Bank

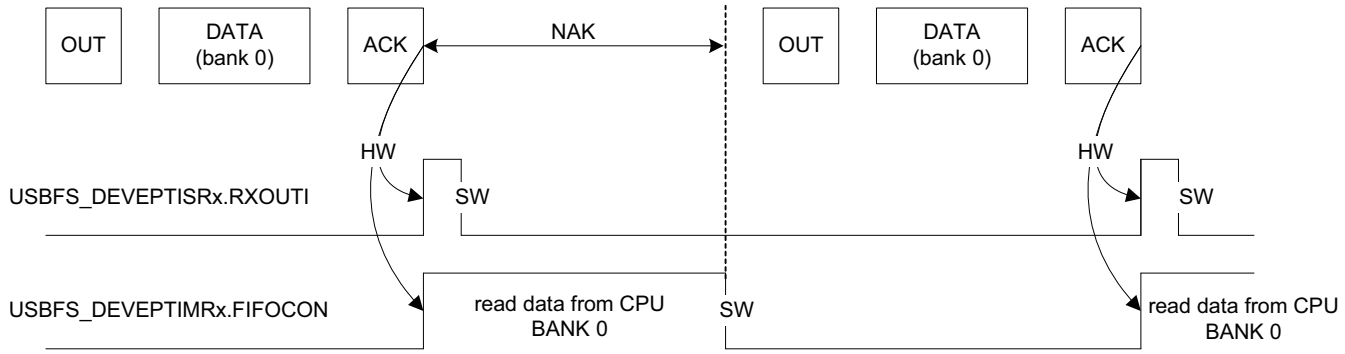
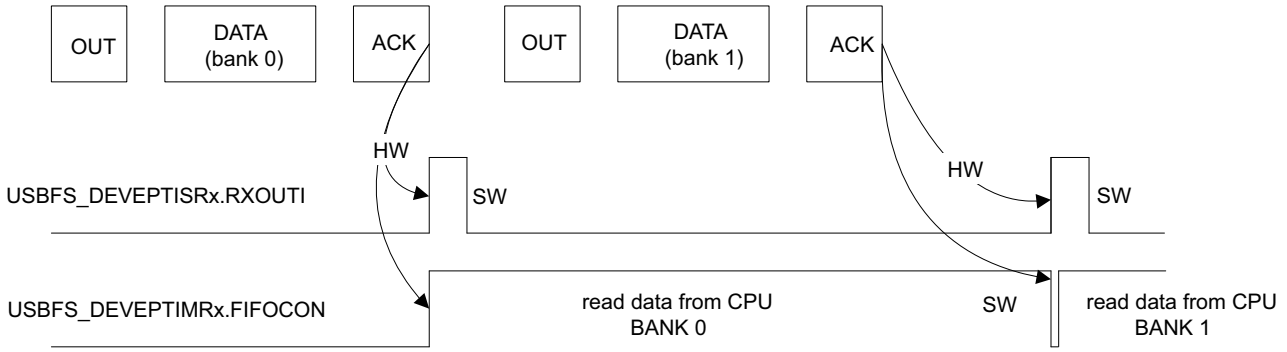


Figure 45-18. Example of an OUT Endpoint with two Data Banks



Detailed Description

The data is read as follows:

- When the bank is full, USBFS_DEVEPTISRx.RXOUTI and USBFS_DEVEPTIMRx.FIFOCON are set, which triggers a PEP_x interrupt if USBFS_DEVEPTIMRx.RXOUTE = 1.
- The user acknowledges the interrupt by writing a one to USBFS_DEVEPTICRx.RXOUTIC in order to clear USBFS_DEVEPTISRx.RXOUTI.
- The user can read the byte count of the current bank from USBFS_DEVEPTISRx.BYCT to know how many bytes to read, rather than polling USBFS_DEVEPTISRx.RWALL.
- The user reads the data from the current bank by using the USBFIFOnDATA register, until all the expected data frame is read or the bank is empty (in which case USBFS_DEVEPTISRx.RWALL is cleared and USBFS_DEVEPTISRx.BYCT reaches zero).
- The user frees the bank and switches to the next bank (if any) by clearing USBFS_DEVEPTIMRx.FIFOCON.

If the endpoint uses several banks, the current one can be read while the following one is being written by the host. Then, when the user clears USBFS_DEVEPTIMRx.FIFOCON, the following bank can already be read and USBFS_DEVEPTISRx.RXOUTI is set immediately.

45.5.2.14 Underflow

Underflow errors exist only for isochronous IN/OUT endpoints. An underflow error sets the Underflow Interrupt (USBFS_DEVEPTISRx.UNDERFI) bit, which triggers a PEP_x interrupt if the Underflow Interrupt Enable (USBFS_DEVEPTIMRx.UNDERFE) bit is one.

- An underflow can occur during the IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBFS.