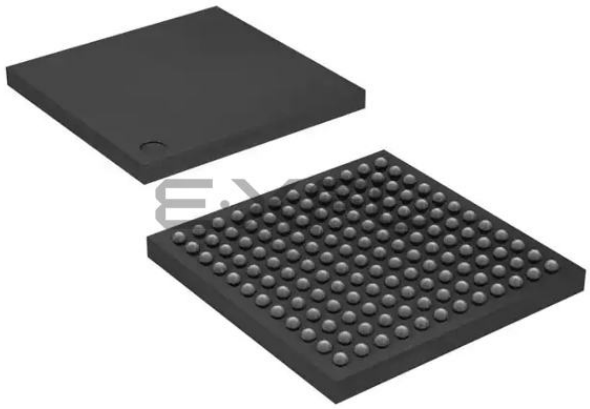


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-M4 |
| Core Size | 32-Bit Single-Core |
| Speed | 120MHz |
| Connectivity | CANbus, EBI/EMI, Ethernet, IrDA, SD, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, POR, PWM, WDT |
| Number of I/O | 117 |
| 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 16x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 144-LFBGA |
| Supplier Device Package | 144-LFBGA (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsam4e8ea-cu |

# 8. Real-time Event Management

The events generated by peripherals are designed to be directly routed to peripherals managing/using these events without processor intervention. Peripherals receiving events contain logic by which to select the one required.

## 8.1 Embedded Characteristics

- Timers, PWM, IO peripherals generate event triggers which are directly routed to event managers such as AFEC or DACC, for example, to start measurement/conversion without processor intervention.
- UART, USART, SPI, TWI, PWM, HSMCI, AES, AFEC, DACC, PIO, TIMER (capture mode) also generate event triggers directly connected to Peripheral DMA Controller (PDC) for data transfer without processor intervention.
- Parallel capture logic is directly embedded in PIO and generates trigger event to PDC to capture data without processor intervention.
- PWM security events (faults) are in combinational form and directly routed from event generators (AFEC, ACC, PMC, TIMER) to PWM module.
- PWM output comparators generate events directly connected to TIMER.
- PMC security event (clock failure detection) can be programmed to switch the MCK on reliable main RC internal clock without processor intervention.

## 8.2 Real-time Event Mapping

**Table 8-1. Real-time Event Mapping List**

| Function | Application | Description | Event Source | Event Destination |
|---|---|---|---|---|
| Security | General-purpose | Immediate GPBR clear (asynchronous) on Tamper detection through WKUP0/1 IO pins (1) | Parallel Input/Output Controller (PIO): WKUP0/1 | General Purpose Backup Registers (GPBR) |
| Safety | General-purpose | Automatic Switch to reliable main RC oscillator in case of Main Crystal Clock Failure (2) | Power Management Controller (PMC) | PMC |
| | General-purpose, motor control | Puts the PWM Outputs in Safe Mode (Main Crystal Clock Failure Detection) (2)(3) | PMC | Pulse Width Modulation (PWM) |
| | Motor control | Puts the PWM Outputs in Safe Mode (Overcurrent sensor, ...) (3)(4) | Analog Comparator Controller (ACC) | |
| | | Puts the PWM Outputs in Safe Mode (Overspeed, Overcurrent detection ...) (3)(5) | Analog-Front-End-Controller (AFEC0/1) | |
| | | Puts the PWM Outputs in Safe Mode (Overspeed detection through TIMER Quadrature Decoder) (3)(6) | Timer Counter (TC) | |
| | General-purpose, motor control | Puts the PWM Outputs in Safe Mode (General Purpose Fault Inputs) (3) | PIO | |
| Image capture | Low-cost image sensor | PC is embedded in PIO (Capture Image from Sensor directly to System Memory) (7) | PIO | DMA |

#### 11.4.1.10 Interrupt Program Status Register

**Name:** IPSR

**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 |
|---|---|---|---|---|---|---|---|
| – | | | | | | | ISR_NUMBER |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISR_NUMBER | | | | | | | |

The IPSR contains the exception type number of the current *Interrupt Service Routine* (ISR).

- **ISR_NUMBER: Number of the Current Exception**

0 = Thread mode

1 = Reserved

2 = NMI

3 = Hard fault

4 = Memory management fault

5 = Bus fault

6 = Usage fault

7–10 = Reserved

11 = SVCall

12 = Reserved for Debug

13 = Reserved

14 = PendSV

15 = SysTick

16 = IRQ0

49 = IRQ46

See “Exception Types” for more information.

#### 11.4.1.11 Execution Program Status Register

**Name:** EPSR

**Access:** Read/Write

**Reset:** 0x000000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | | | | | ICI/IT | | T |

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

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | | | | | | | |

The EPSR contains the Thumb state bit, and the execution state bits for either the *If-Then* (IT) instruction, or the *Interruptible-Continuable Instruction* (ICI) field for an interrupted load multiple or store multiple instruction.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in the application software are ignored. Fault handlers can examine the EPSR value in the stacked PSR to indicate the operation that is at fault. See “Exception Entry and Return” .

- **ICI: Interruptible-continuable Instruction**

When an interrupt occurs during the execution of an LDM, STM, PUSH, POP, VLDM, VSTM, VPUSH, or VPOP instruction, the processor:

– Stops the load multiple or store multiple instruction operation temporarily

– Stores the next register operand in the multiple operation to EPSR bits[15:12].

After servicing the interrupt, the processor:

– Returns to the register pointed to by bits[15:12]

– Resumes the execution of the multiple load or store instruction.

When the EPSR holds the ICI execution state, bits[26:25,11:10] are zero.

- **IT: If-Then Instruction**

Indicates the execution state bits of the IT instruction.

The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See “IT”  for more information.

- **T: Thumb State**

The Cortex-M4 processor only supports the execution of instructions in Thumb state. The following can clear the T bit to 0:

– Instructions BLX, BX and POP{PC}

– Restoration from the stacked xPSR value on an exception return

– Bit[0] of the vector value on an exception entry or reset.

Attempting to execute instructions when the T bit is 0 results in a fault or lockup. See “Lockup”  for more information.

- Add the two multiplication results to the signed 64-bit value in *RdLo* and *RdHi* to create the resulting 64-bit product.
- Write the 64-bit product in *RdLo* and *RdHi*.

Restrictions

In these instructions:

- Do not use SP and do not use PC.
- *RdHi* and *RdLo* must be different registers.

Condition Flags

These instructions do not affect the condition code flags.

Examples

```
SMLAL    R4, R5, R3, R8  ; Multiplies R3 and R8, adds R5:R4 and writes to
                         ; R5:R4
SMLALBT  R2, R1, R6, R7  ; Multiplies bottom halfword of R6 with top
                         ; halfword of R7, sign extends to 32-bit, adds
                         ; R1:R2 and writes to R1:R2
SMLALTB  R2, R1, R6, R7  ; Multiplies top halfword of R6 with bottom
                         ; halfword of R7,sign extends to 32-bit, adds R1:R2
                         ; and writes to R1:R2
SMLALD   R6, R8, R5, R1  ; Multiplies top halfwords in R5 and R1 and bottom
                         ; halfwords of R5 and R1, adds R8:R6 and writes to
                         ; R8:R6
SMLALDX  R6, R8, R5, R1  ; Multiplies top halfword in R5 with bottom
                         ; halfword of R1, and bottom halfword of R5 with
                         ; top halfword of R1, adds R8:R6 and writes to
                         ; R8:R6.
```

#### 11.6.9.1 BFC and BFI

Bit Field Clear and Bit Field Insert.

Syntax

```
BFC{cond} Rd, #lsb, #width
BFI{cond} Rd, Rn, #lsb, #width
```

where:

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

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32-*lsb*.

Operation

BFC clears a bitfield in a register. It clears *width* bits in *Rd*, starting at the low bit position *lsb*. Other bits in *Rd* are unchanged.

BFI copies a bitfield into one register from another register. It replaces *width* bits in *Rd* starting at the low bit position *lsb*, with *width* bits from *Rn* starting at bit[0]. Other bits in *Rd* are unchanged.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the flags.

Examples

```
BFC   R4, #8, #12       ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI   R9, R2, #8, #12   ; Replace bit 8 to bit 19 (12 bits) of R9 with
                        ; bit 0 to bit 11 from R2.
```

#### 11.6.12.3 DMB

Data Memory Barrier.

Syntax

```
DMB{cond}
```

where:

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

Operation

DMB acts as a data memory barrier. It ensures that all explicit memory accesses that appear, in program order, before the DMB instruction are completed before any explicit memory accesses that appear, in program order, after the DMB instruction. DMB does not affect the ordering or execution of instructions that do not access memory.

Condition Flags

This instruction does not change the flags.

Examples

```
DMB  ; Data Memory Barrier
```

#### 11.8.3.4 Interrupt Clear-pending Registers

**Name:** NVIC_ICPRx [x=0..7]

**Access:** Read/Write

**Reset:** 0x000000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLRPEND | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| CLRPEND | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CLRPEND | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLRPEND | | | | | | | |

These registers remove the pending state from interrupts, and show which interrupts are pending.

- **CLRPEND: Interrupt Clear-pending**

Write:

0: No effect.

1: Removes the pending state from an interrupt.

Read:

0: Interrupt is not pending.

1: Interrupt is pending.

Note: Writing a 1 to an ICPR bit does not affect the active state of the corresponding interrupt.

- **INVPC: Invalid PC Load Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” . It is caused by an invalid PC load by EXC_RETURN:

0: No invalid PC load usage fault.

1: The processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value.

When this bit is set to 1, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC.

- **NOCP: No Coprocessor Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” . The processor does not support coprocessor instructions:

0: No usage fault caused by attempting to access a coprocessor.

1: The processor has attempted to access a coprocessor.

- **UNALIGNED: Unaligned Access Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” .

0: No unaligned access fault, or unaligned access trapping not enabled.

1: The processor has made an unaligned memory access.

Enable trapping of unaligned accesses by setting the UNALIGN_TRP bit in the SCB_CCR to 1. See “Configuration and Control Register” . Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of the setting of UNALIGN_TRP.

- **DIVBYZERO: Divide by Zero Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” .

0: No divide by zero fault, or divide by zero trapping not enabled.

1: The processor has executed an SDIV or UDIV instruction with a divisor of 0.

When the processor sets this bit to 1, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Enable trapping of divide by zero by setting the DIV_0_TRP bit in the SCB_CCR to 1. See “Configuration and Control Register” .

### 14.5.3 Real-time Timer Value Register

**Name:** RTT_VR

**Address:** 0x400E1838

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CRTV | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| CRTV | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CRTV | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CRTV | | | | | | | |

- **CRTV: Current Real-time Value**

Returns the current value of the Real-time Timer.

Note: As CRTV can be updated asynchronously, it must be read twice at the same value.

The status of lock bits can be returned by the EEFC. The ‘Get Lock Bit’ sequence is the following:

1. Execute the ‘Get Lock Bit’ command by writing EEFC_FCR.FCMD with the GLB command. Field EEFC_FCR.FARG is meaningless.
2. Lock bits can be read by the software application in EEFC_FRR. The first word read corresponds to the 32 first lock bits, next reads providing the next 32 lock bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

For example, if the third bit of the first word read in EEFC_FRR is set, the third lock region is locked.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

Note: Access to the Flash in read is permitted when a ‘Set Lock Bit’, ‘Clear Lock Bit’ or ‘Get Lock Bit’ command is executed.

#### 20.4.3.5 GPNVM Bit

GPNVM bits do not interfere with the embedded Flash memory plane. For more details, refer to the section “Memories” of this datasheet.

The ‘Set GPNVM Bit’ sequence is the following:

1. Execute the ‘Set GPNVM Bit’ command by writing EEFC_FCR.FCMD with the SGPB command and EEFC_FCR.FARG with the number of GPNVM bits to be set.
2. When the GPNVM bit is set, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.
3. The result of the SGPB command can be checked by running a ‘Get GPNVM Bit’ (GGPB) command.

Note: The value of the FARG argument passed together with SGPB command must not exceed the higher GPNVM index available in the product. Flash data content is not altered if FARG exceeds the limit. Command Error is detected only if FARG is greater than 8.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

It is possible to clear GPNVM bits previously set. The ‘Clear GPNVM Bit’ sequence is the following:

1. Execute the ‘Clear GPNVM Bit’ command by writing EEFC_FCR.FCMD with the CGPB command and EEFC_FCR.FARG with the number of GPNVM bits to be cleared.
2. When the clear completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note: The value of the FARG argument passed together with CGPB command must not exceed the higher GPNVM index available in the product. Flash data content is not altered if FARG exceeds the limit. Command Error is detected only if FARG is greater than 8.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

The status of GPNVM bits can be returned by the EEFC. The sequence is the following:

1. Execute the ‘Get GPNVM Bit’ command by writing EEFC_FCR.FCMD with the GGPB command. Field EEFC_FCR.FARG is meaningless.
2. GPNVM bits can be read by the software application in EEFC_FRR. The first word read corresponds to the 32 first GPNVM bits, following reads provide the next 32 GPNVM bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

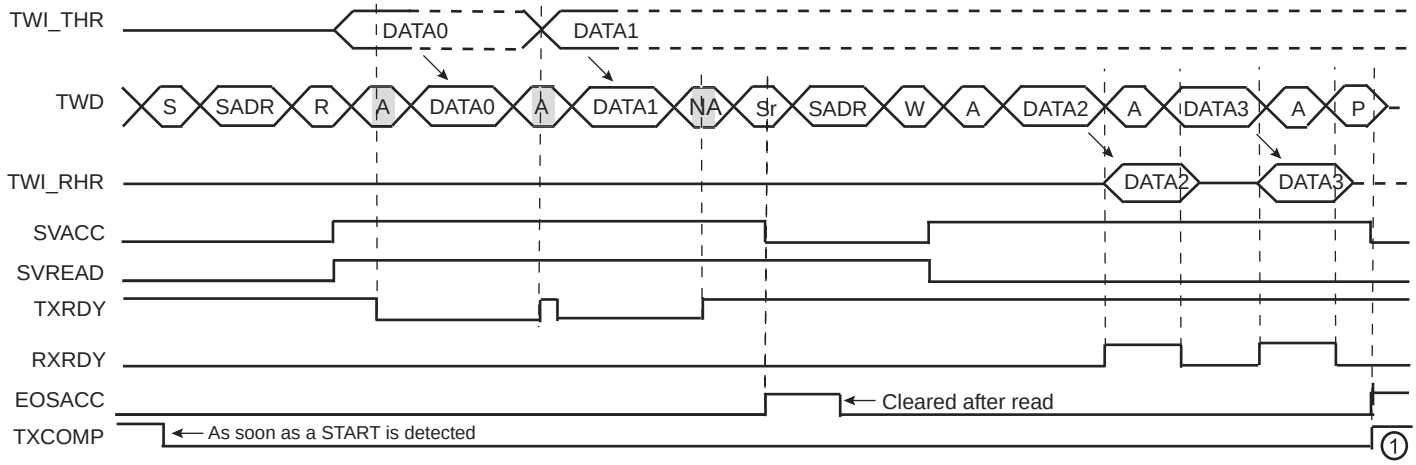
*Reversal After a Repeated Start*

**Reversal of Read to Write**

The master initiates the communication by a read command and finishes it by a write command.

Figure 35-28 describes the repeated start + reversal from Read to Write mode.

**Figure 35-28. Repeated Start + Reversal from Read to Write Mode**



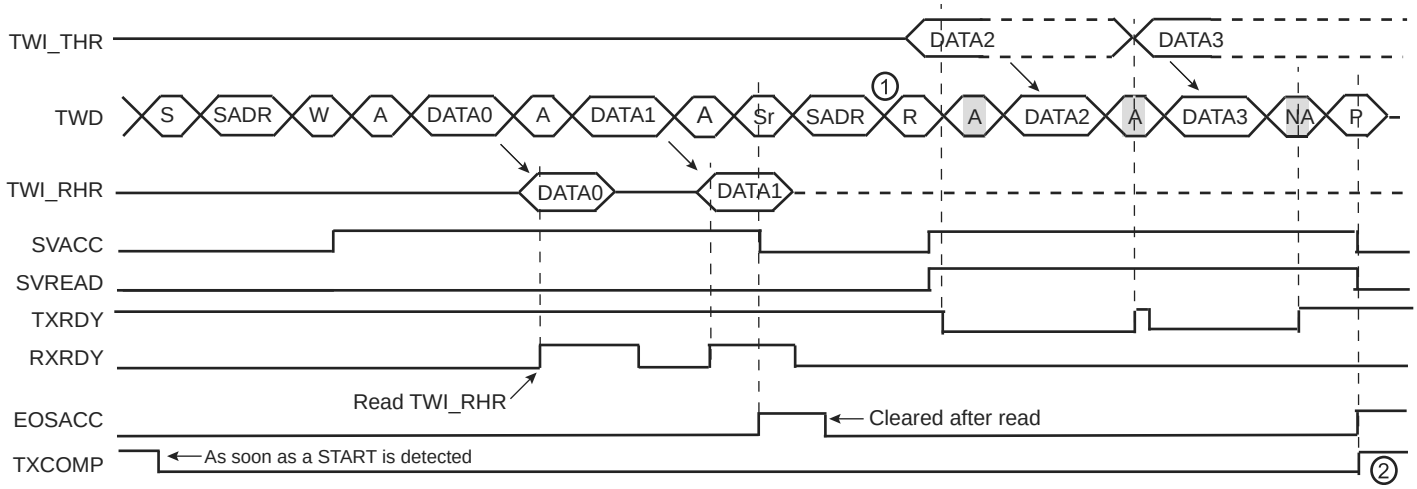

Note: 1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

**Reversal of Write to Read**

The master initiates the communication by a write command and finishes it by a read command.

Figure 35-29 describes the repeated start + reversal from Write to Read mode.

**Figure 35-29. Repeated Start + Reversal from Write to Read Mode**




Notes: 1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

### 35.7.5.5 Using the Peripheral DMA Controller (PDC) in Slave Mode

The use of the PDC significantly reduces the CPU load.

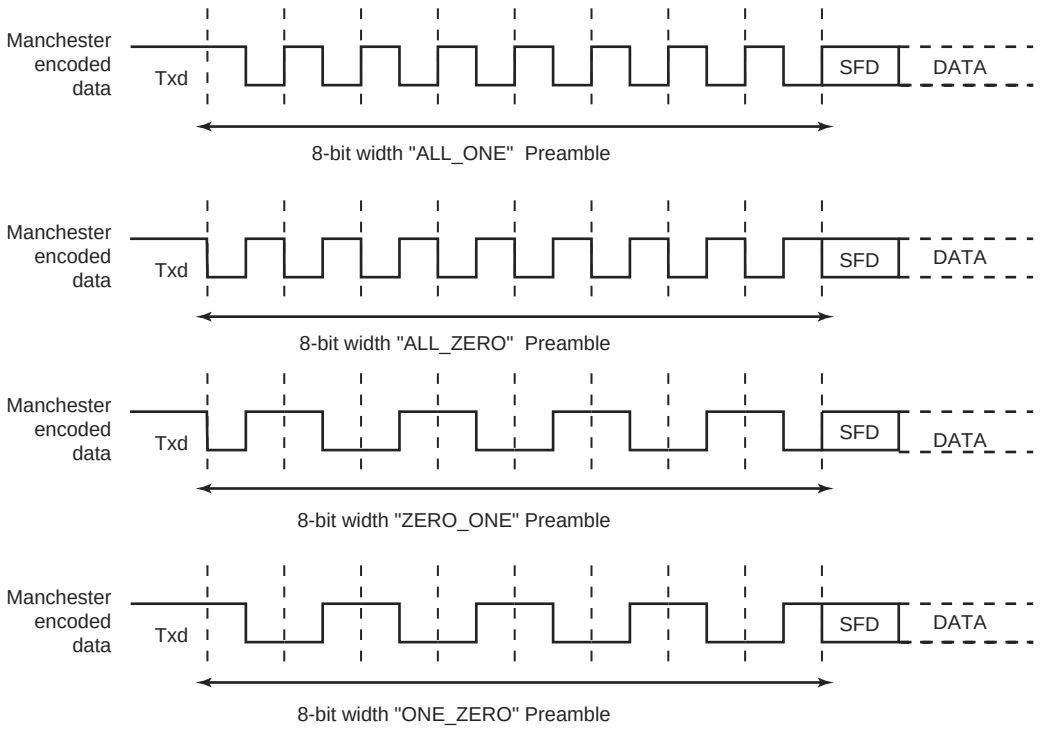
*Data Transmit with the PDC in Slave Mode*

The following procedure shows an example of data transmission with PDC.

- Offers Buffer Transfer without Processor Intervention
- Register Write Protection

predefined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 37-8 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to 1, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

**Figure 37-8. Preamble Patterns, Default Polarity Assumed**




A start frame delimiter is to be configured using the ONEBIT bit in the US_MR. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 37-9 illustrates these patterns. If the start frame delimiter, also known as the start bit, is one bit, (ONEBIT = 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT to 0), a sequence of three bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC bit in the US_MR is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC bit in US_MR must be set to 1. In this case, the MODSYNC bit in the US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR. The USART character format is modified and includes sync information.

### 37.7.13 USART Receive Holding Register

**Name:** US_RHR

**Address:** 0x400A0018 (0), 0x400A4018 (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 |
| RXSYNH | – | – | – | – | – | – | RXCHR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXCHR | | | | | | | |

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last character received is a data.

1: Last character received is a command.

### 39.7.5 PWM Interrupt Enable Register 1

**Name:** PWM_IER1

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | FCHID3 | FCHID2 | FCHID1 | FCHID0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | CHID3 | CHID2 | CHID1 | CHID0 |

- **CHIDx: Counter Event on Channel x Interrupt Enable**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Enable**

### 40.11.1 Boot Procedure, Processor Mode

1. Configure the HSMCI data bus width programming SDCBUS Field in the HSMCI_SDCR. The BOOT_BUS_WIDTH field located in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks, writing BLKLEN and BCNT fields of the HSMCI_BLKR.
3. Issue the Boot Operation Request command by writing to the HSMCI_CMDR with SPCMD field set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
4. The BOOT_ACK field located in the HSMCI_CMDR must be set to one, if the BOOT_ACK field of the MMC device located in the Extended CSD register is set to one.
5. Host processor can copy boot data sequentially as soon as the RXRDY flag is asserted.
6. When Data transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR with SPCMD field set to BOOTEND.

## 40.12 HSMCI Transfer Done Timings

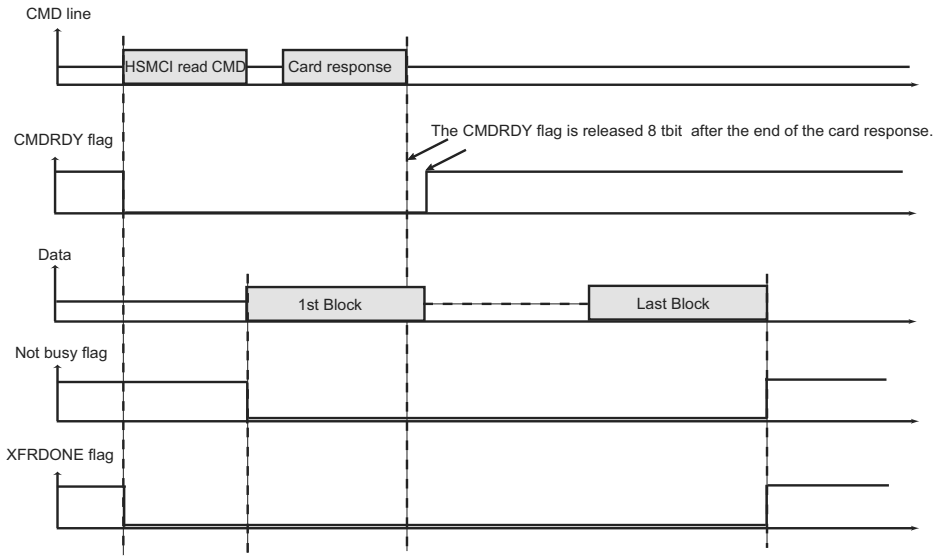
### 40.12.1 Definition

The XFRDONE flag in the HSMCI_SR indicates exactly when the read or write sequence is finished.

### 40.12.2 Read Access

During a read access, the XFRDONE flag behaves as shown in Figure 40-11.

**Figure 40-11. XFRDONE During a Read Access**




### 40.12.3 Write Access

During a write access, the XFRDONE flag behaves as shown in Figure 40-12.

### 41.7.9 UDP Reset Endpoint Register

**Name:** UDP_RST_EP

**Address:** 0x40084028

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**
- **EP6: Reset Endpoint 6**
- **EP7: Reset Endpoint 7**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the UDP_CSRx. It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

**Warning:** This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0: No reset

1: Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

### 42.8.45 GMAC PTP Peer Event Frame Transmitted Nanoseconds Register

**Name:** GMAC_PEFTN

**Address:** 0x400341F4

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | RUD | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| RUD | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RUD | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RUD | | | | | | | |

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

### 46.7.4 ADC Transfer Function

The first operation of the ADC is a sampling function relative to a common mode voltage. The common mode voltage ($V_{CM}$) is equal to $V_{ADVREF}/2$ when the bits OFFx = 1, in Differential and in Single-ended mode. When the bits OFFx = 0, sampling is done versus $V_{ADVREF}/4$ for gain = 2, and $V_{ADVREF}/8$ for gain = 4, in Single-ended mode only.

The code in AFEC_CDR is a 12-bit positive integer. The internal DAC is set for the code 2047.

#### 46.7.4.1 Differential Mode

A differential input voltage $V_I = V_{I+} - V_{I-}$ can be applied between two selected differential pins, e.g., AD0 and AD1. The ideal code Ci is calculated by using the following formula and rounding the result to the nearest positive integer.

$$Ci = \frac{4096}{V_{ADVREF}} \times V_I \times Gain + 2047$$

Table 46-31 is a computation example for the above formula, where $V_{ADVREF}$ = 3V.

**Table 46-31. Input Voltage Values in Differential Mode**

| Ci | Gain = 0.5 | Gain = 1 | Gain = 2 |
|---|---|---|---|
| 0 | -3 | -1.5 | -0.75 |
| 2047 | 0 | 0 | 0 |
| 4095 | 3 | 1.5 | 0.75 |

#### 46.7.4.2 Single-ended Mode

A single input voltage $V_I$ can be applied to selected pins, e.g., AD0 or AD1. The ideal code Ci is calculated by using the following formula and rounding the result to the nearest positive integer.

The single-ended ideal code conversion formula for OFFx = 1 is:

$$Ci = \frac{4096}{V_{ADVREF}} \times \left(V_I - \frac{V_{ADVREF}}{2}\right) \times Gain + 2047$$

Table 46-32 is a computation example for the above formula, where $V_{ADVREF}$ = 3V.

**Table 46-32. Input Voltage Values in Single-ended Mode, OFFx = 1**

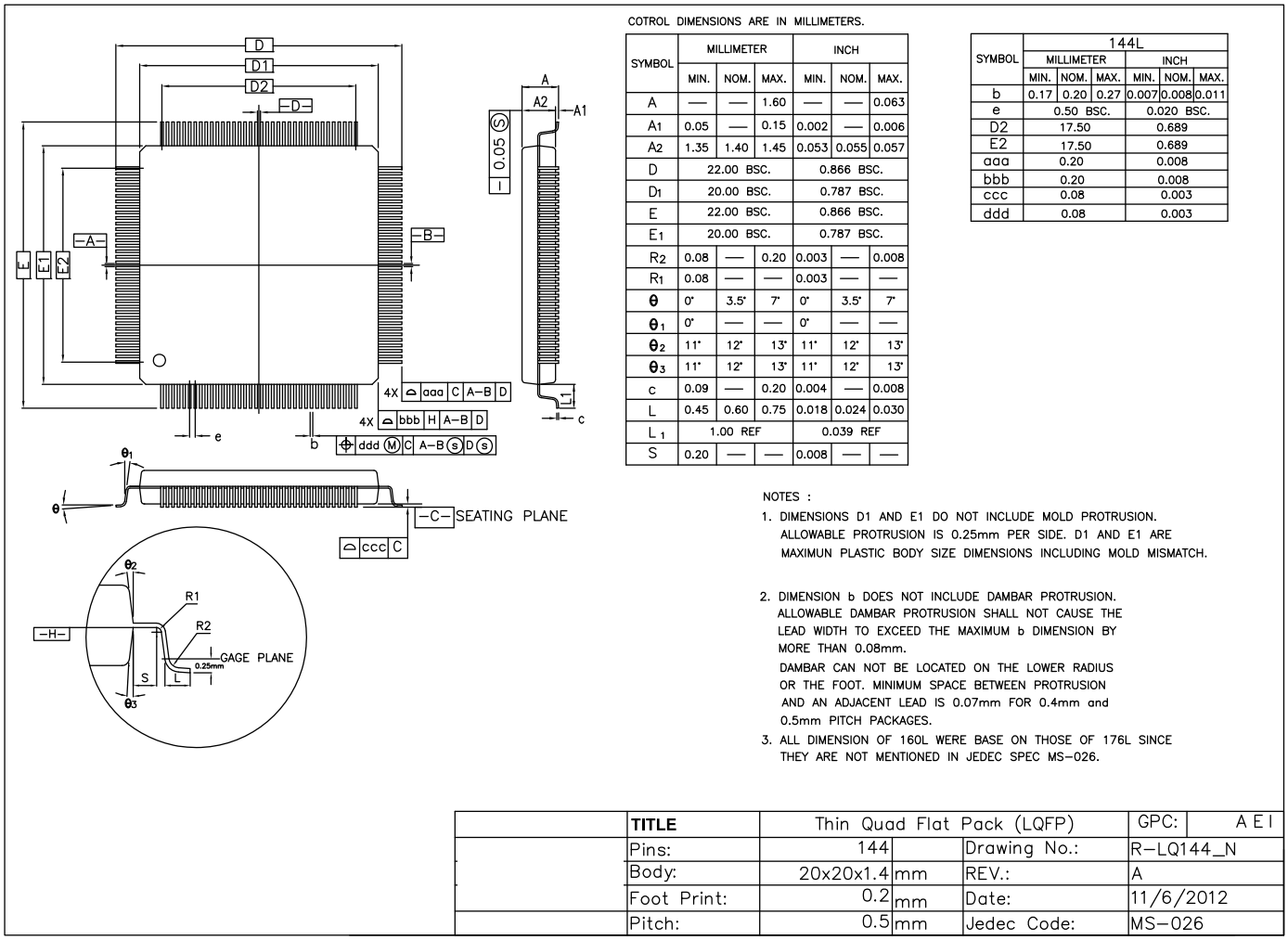
| Ci | Gain = 1 | Gain = 2 | Gain = 4 |
|---|---|---|---|
| 0 | 0 | 0.75 | 1.125 |
| 2047 | 1.5 | 1.5 | 1.5 |
| 4095 | 3 | 2.25 | 1.875 |

The single-ended ideal code conversion formula for OFFx = 0 is:

$$Ci = V_I \times Gain \times \frac{4096}{V_{ADVREF}} - 1$$

## 47.4 144-lead LQFP Package Drawing

**Figure 47-4. 144-lead LQFP Package Drawing**




COTROL DIMENSIONS ARE IN MILLIMETERS.

| SYMBOL | MILLIMETER | | | INCH | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | — | — | 1.60 | — | — | 0.063 |
| A1 | 0.05 | — | 0.15 | 0.002 | — | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| D | 22.00 BSC. | | | 0.866 BSC. | | |
| D1 | 20.00 BSC. | | | 0.787 BSC. | | |
| E | 22.00 BSC. | | | 0.866 BSC. | | |
| E1 | 20.00 BSC. | | | 0.787 BSC. | | |
| R2 | 0.08 | — | 0.20 | 0.003 | — | 0.008 |
| R1 | 0.08 | — | — | 0.003 | — | — |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| θ1 | 0° | — | — | 0° | — | — |
| θ2 | 11° | 12° | 13° | 11° | 12° | 13° |
| θ3 | 11° | 12° | 13° | 11° | 12° | 13° |
| c | 0.09 | — | 0.20 | 0.004 | — | 0.008 |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | 1.00 REF | | | 0.039 REF | | |
| S | 0.20 | — | — | 0.008 | — | — |

| SYMBOL | 144L | | | | | |
|---|---|---|---|---|---|---|
| | MILLIMETER | | | INCH | | |
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| b | 0.17 | 0.20 | 0.27 | 0.007 | 0.008 | 0.011 |
| e | 0.50 BSC. | | | 0.020 BSC. | | |
| D2 | 17.50 | | | 0.689 | | |
| E2 | 17.50 | | | 0.689 | | |
| aaa | 0.20 | | | 0.008 | | |
| bbb | 0.20 | | | 0.008 | | |
| ccc | 0.08 | | | 0.003 | | |
| ddd | 0.08 | | | 0.003 | | |

NOTES :

1. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUN PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
2. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm. DAMBAR CAN NOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD IS 0.07mm FOR 0.4mm and 0.5mm PITCH PACKAGES.
3. ALL DIMENSION OF 160L WERE BASE ON THOSE OF 176L SINCE THEY ARE NOT MENTIONED IN JEDEC SPEC MS-026.

| | TITLE | Thin Quad Flat Pack (LQFP) | | | GPC: AEI |
|---|---|---|---|---|---|
| | Pins: | 144 | | Drawing No.: | R-LQ144_N |
| | Body: | 20x20x1.4 | mm | REV.: | A |
| | Foot Print: | 0.2 | mm | Date: | 11/6/2012 |
| | Pitch: | 0.5 | mm | Jedec Code: | MS-026 |

**Table 47-10. Device and LQFP Package Maximum Weight (Preliminary)**

| | | |
|---|---|---|
| SAM4E | 900 | mg |

**Table 47-11. LQFP Package Reference**

| | |
|---|---|
| JEDEC Drawing Reference | MS-026-C |
| JESD97 Classification | e3 |

**Table 47-12. LQFP Package Characteristics**

| | |
|---|---|
| Moisture Sensitivity Level | 3 |

This package respects the recommendations of the NEMI User Group.