**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®-M7 |
| Core Size | 32-Bit Single-Core |
| Speed | 300MHz |
| Connectivity | EBI/EMI, I²C, IrDA, LINbus, MMC/SD/SDIO, QSPI, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT |
| Number of I/O | 114 |
| Program Memory Size | 1MB (1M x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 384K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.08V ~ 3.6V |
| Data Converters | A/D 24x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°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/atsams70q20b-cnt |

# 12. Event System

The events generated by peripherals (source) are designed to be directly routed to peripherals (destination) using these events without processor intervention. The trigger source can be programmed in the destination peripheral.

## 12.1 Embedded Characteristics

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

## 12.2 Real-time Event Mapping

**Table 12-1. Real-time Event Mapping List**

| Function | Application | Description | Event Source | Event Destination |
|---|---|---|---|---|
| Safety | General-purpose | Automatic switch to reliable main RC oscillator in case of main crystal clock failure (see **Note 1**) | Power Management Controller (PMC) | PMC |
| | General-purpose, motor control, power factor correction (PFC) | Puts the PWM outputs in Safe mode in case of main crystal clock failure (see **Notes 1, 2**) | PMC | Pulse Width Modulation 0 and 1 (PWM0 and PWM1) |
| | Motor control, PFC | Puts the PWM outputs in Safe mode (overcurrent detection, etc.) (see **Notes 2, 3**) | Analog Comparator Controller (ACC) | PWM0 and PWM1 |
| | Motor control, PFC | Puts the PWM outputs in Safe mode (overspeed, overcurrent detection, etc.) (see **Notes 2, 4**) | Analog Front-End Controller (AFEC0) | PWM0 and PWM1 |
| | | | AFEC1 | PWM0 and PWM1 |

| Function | Application | Description | Event Source | Event Destination |
|---|---|---|---|---|
| | | | PWM0 Comparator Output OC2 | TC2 TIOA8 and TIOB8 |
| | | | PWM1 Comparator Output OC0 | TC3 TIOA9 and TIOB9 |
| | | | PWM1 Comparator Output OC1 | TC3 TIOA10 and TIOB10 |
| Audio clock recovery from Ethernet | Audio | GMAC GTSUCOMP signal adaptation via TC (TC_EMR.TRIGSRCB) in order to drive the clock reference of the external PLL for the audio clock | GMAC GTSUCOMP | TC3 TIOB11 |
| Direct Memory Access | General-purpose | Peripheral trigger event generation to transfer data to/ from system memory (see **Note 18**) | USART, UART, TWIHS, SPI, QSPI, AFEC, TC (Capture), SSC, HSMCI, DAC, AES, PWM, PIO, I2SC | XDMA |

**Note:**

1. Refer to 31.15 Main Crystal Oscillator Failure Detection.
2. Refer to 51.5.4 Fault Inputs and 51.6.2.7 Fault Protection.
3. Refer to 54.6.4 Fault Mode.
4. Refer to 54.5.4 Fault Output.
5. Refer to 23.4.9.2 Low-power Tamper Detection and Anti-Tampering and 29.3.1 SYS_GPBRx.
6. Refer to 50.6.18 Fault Mode.
7. Refer to 51.7.49 PWM_ETRGx.
8. Refer to 51.6.5 PWM External Trigger Mode.
9. Refer to 52.6.6 Conversion Triggers and 52.7.2 AFEC_MR.
10. Refer to 58.10 Temperature Sensor.
11. Refer to 27.5.8 Waveform Generation.
12. Refer to 51.7.36 PWM_CMPVx and 51.6.4 PWM Event Lines.
13. Refer to 53.7.3 DACC_TRIGR.
14. Refer to 51.6.3 PWM Comparison Units and 51.6.4 PWM Event Lines.
15. Refer to 32.5.14 Parallel Capture Mode.
16. Refer to 51.6.2.2 Comparator.
17. Refer to 50.6.14 Synchronization with PWM.
18. Refer to 36. DMA Controller (XDMAC).

**Bit 1 – ALREN** Alarm Interrupt Enable

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | The alarm interrupt is enabled. |

**Bit 0 – ACKEN** Acknowledge Update Interrupt Enable

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | The acknowledge for update interrupt is enabled. |

e. Program PMC_MCKR.CSS.

f. Wait for PMC_SR.MCKRDY to be set.

If a new value for PMC_MCKR.CSS corresponds to MAINCK or SLCK:

a. Program PMC_MCKR.CSS.

b. Wait for PMC_SR.MCKRDY to be set.

c. Program PMC_MCKR.PRES.

d. Wait for PMC_SR.MCKRDY to be set.

If CSS, MDIV or PRES are modified at any stage, the MCKRDY bit goes low to indicate that MCK and HCLK are not yet ready. The user must wait for MCKRDY bit to be set again before using MCK and HCLK.

Note: If PLLA clock was selected as MCK and the user decides to modify it by writing a new value into CKGR_PLLAR, the MCKRDY flag will go low while PLLA is unlocked. Once PLLA is locked again, LOCKA goes high and MCKRDY is set.
While PLLA is unlocked, MCK selection is automatically changed to SLCK for PLLA. For further information, see "Clock Switching Waveforms".

MCK is MAINCK divided by 2.

8. Select the Programmable clocks (PCKx):
PCKx are controlled via registers PMC_SCER, PMC_SCDR and PMC_SCSR.

   PCKx can be enabled and/or disabled via PMC_SCER and PMC_SCDR. Three PCKx can be used. PMC_SCSR indicates which PCKx is enabled. By default all PCKx are disabled.

   PMC_PCKx registers are used to configure PCKx.

   PMC_PCKx.CSS is used to select the PCKx divider source. Several clock options are available:

   – MAINCK
   – SLCK
   – MCK
   – PLLACK
   – UPLLCKDIV
   SLCK is the default clock source.

   PMC_PCKx.PRES is used to control the PCKx prescaler. It is possible to choose between different values (1 to 256). PCKx output is prescaler input divided by PRES. By default, the PRES value is cleared which means that PCKx is equal to Slow clock.

   Once PMC_PCKx has been configured, the corresponding PCKx must be enabled and the user must wait for PMC_SR.PCKRDYx to be set. This can be done either by polling PMC_SR.PCKRDYx or by waiting for the interrupt line to be raised if the associated interrupt source (PCKRDYx) has been enabled in PMC_IER. All parameters in PMC_PCKx can be programmed in a single write operation.

   If the PMC_PCKx.CSS and PMC_PCKx.PRES parameters are to be modified, the corresponding PCKx must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable PCKx and wait for the PCKRDYx bit to be set.

9. Enable the peripheral clocks

These additional modes are:

- Rising edge detection
- Falling edge detection
- Low-level detection
- High-level detection

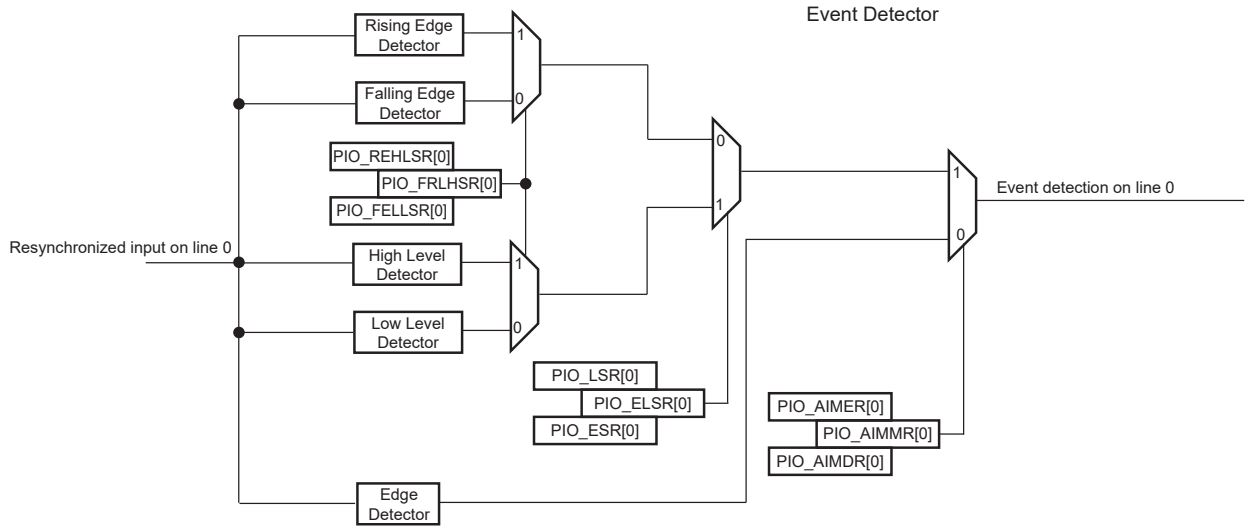
In order to select an additional interrupt mode:

- The type of event detection (edge or level) must be selected by writing in the Edge Select Register (PIO_ESR) and Level Select Register (PIO_LSR) which select, respectively, the edge and level detection. The current status of this selection is accessible through the Edge/Level Status Register (PIO_ELSR).
- The polarity of the event detection (rising/falling edge or high/low-level) must be selected by writing in the Falling Edge/Low-Level Select Register (PIO_FELLSR) and Rising Edge/High-Level Select Register (PIO_REHLSR) which allow to select falling or rising edge (if edge is selected in PIO_ELSR) edge or high- or low-level detection (if level is selected in PIO_ELSR). The current status of this selection is accessible through the Fall/Rise - Low/High Status Register (PIO_FRLHSR).

When an input edge or level is detected on an I/O line, the corresponding bit in the Interrupt Status Register (PIO_ISR) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted.The interrupt signals of the 32 channels are ORed-wired together to generate a single interrupt signal to the interrupt controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled. When an Interrupt is enabled on a “level”, the interrupt is generated as long as the interrupt source is not cleared, even if some read accesses in PIO_ISR are performed.

**Figure 32-6. Event Detector on Input Lines (Figure Represents Line 0)**




Example of interrupt generation on following lines:

- Rising edge on PIO line 0
- Falling edge on PIO line 1
- Rising edge on PIO line 2
- Low-level on PIO line 3

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0478 | XDMAC_CC16 | 7:0 | MEMSET | SWREQ | | DSYNC | | MBSIZE[1:0] | | TYPE |
| | | 15:8 | | DIF | SIF | DWIDTH[1:0] | | CSIZE[2:0] | | |
| | | 23:16 | WRIP | RDIP | INITD | | DAM[1:0] | | SAM[1:0] | |
| | | 31:24 | | PERID[6:0] | | | | | | |
| 0x047C | XDMAC_CDS_MSP16 | 7:0 | SDS_MSP[7:0] | | | | | | | |
| | | 15:8 | SDS_MSP[15:8] | | | | | | | |
| | | 23:16 | DDS_MSP[7:0] | | | | | | | |
| | | 31:24 | DDS_MSP[15:8] | | | | | | | |
| 0x0480 | XDMAC_CSUS16 | 7:0 | SUBS[7:0] | | | | | | | |
| | | 15:8 | SUBS[15:8] | | | | | | | |
| | | 23:16 | SUBS[23:16] | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0484 | XDMAC_CDUS16 | 7:0 | DUBS[7:0] | | | | | | | |
| | | 15:8 | DUBS[15:8] | | | | | | | |
| | | 23:16 | DUBS[23:16] | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0488 ... 0x048F | Reserved | | | | | | | | | |
| 0x0490 | XDMAC_CIE17 | 7:0 | | ROIE | WBIE | RBIE | FIE | DIE | LIE | BIE |
| | | 15:8 | | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0494 | XDMAC_CID17 | 7:0 | | ROID | WBEID | RBEID | FID | DID | LID | BID |
| | | 15:8 | | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0498 | XDMAC_CIM17 | 7:0 | | ROIM | WBEIM | RBEIM | FIM | DIM | LIM | BIM |
| | | 15:8 | | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x049C | XDMAC_CIS17 | 7:0 | | ROIS | WBEIS | RBEIS | FIS | DIS | LIS | BIS |
| | | 15:8 | | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x04A0 | XDMAC_CSA17 | 7:0 | SA[7:0] | | | | | | | |
| | | 15:8 | SA[15:8] | | | | | | | |
| | | 23:16 | SA[23:16] | | | | | | | |
| | | 31:24 | SA[31:24] | | | | | | | |
| 0x04A4 | XDMAC_CDA17 | 7:0 | DA[7:0] | | | | | | | |
| | | 15:8 | DA[15:8] | | | | | | | |
| | | 23:16 | DA[23:16] | | | | | | | |
| | | 31:24 | DA[31:24] | | | | | | | |
| 0x04A8 | XDMAC_CNDA17 | 7:0 | NDA[5:0] | | | | | | | NDAIF |

### 36.9.10 XDMAC Global Channel Status Register

**Name:** XDMAC_GS
**Offset:** 0x24
**Reset:** 0x00000000
**Property:** Read-only

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 – ST** XDMAC Channel x Status

| Value | Description |
|---|---|
| 0 | This bit indicates that the channel x is disabled. |
| 1 | This bit indicates that the channel x is enabled. If a channel disable request is issued, this bit remains asserted until pending transaction is completed. |

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xB8 | GMAC_WOL | 7:0 | IP[7:0] | | | | | | | |
| | | 15:8 | IP[15:8] | | | | | | | |
| | | 23:16 | | | | | MTI | SA1 | ARP | MAG |
| | | 31:24 | | | | | | | | |
| 0xBC | GMAC_IPGS | 7:0 | FL[7:0] | | | | | | | |
| | | 15:8 | FL[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0xC0 | GMAC_SVLAN | 7:0 | VLAN_TYPE[7:0] | | | | | | | |
| | | 15:8 | VLAN_TYPE[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | ESVLAN | | | | | | | |
| 0xC4 | GMAC_TPFCP | 7:0 | PEV[7:0] | | | | | | | |
| | | 15:8 | PQ[7:0] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0xC8 | GMAC_SAMB1 | 7:0 | ADDR[7:0] | | | | | | | |
| | | 15:8 | ADDR[15:8] | | | | | | | |
| | | 23:16 | ADDR[23:16] | | | | | | | |
| | | 31:24 | ADDR[31:24] | | | | | | | |
| 0xCC | GMAC_SAMT1 | 7:0 | ADDR[7:0] | | | | | | | |
| | | 15:8 | ADDR[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0xD0 ... 0xDB | Reserved | | | | | | | | | |
| 0xDC | GMAC_NSC | 7:0 | NANOSEC[7:0] | | | | | | | |
| | | 15:8 | NANOSEC[15:8] | | | | | | | |
| | | 23:16 | | | NANOSEC[21:16] | | | | | |
| | | 31:24 | | | | | | | | |
| 0xE0 | GMAC_SCL | 7:0 | SEC[7:0] | | | | | | | |
| | | 15:8 | SEC[15:8] | | | | | | | |
| | | 23:16 | SEC[23:16] | | | | | | | |
| | | 31:24 | SEC[31:24] | | | | | | | |
| 0xE4 | GMAC_SCH | 7:0 | SEC[7:0] | | | | | | | |
| | | 15:8 | SEC[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0xE8 | GMAC_EFTSH | 7:0 | RUD[7:0] | | | | | | | |
| | | 15:8 | RUD[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0xEC | GMAC_EFRSH | 7:0 | RUD[7:0] | | | | | | | |
| | | 15:8 | RUD[15:8] | | | | | | | |
| | | 23:16 | | | | | | | | |

### 38.8.79 GMAC Receive Overruns Register

**Name:** GMAC_ROE
**Offset:** 0x1A4
**Reset:** 0x00000000
**Property:** -

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | RXOVR[9:8] | |
| Access | | | | | | | R | R |
| Reset | | | | | | | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXOVR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 9:0 – RXOVR[9:0]** Receive Overruns
This bit field counts the number of frames that are address recognized but were not copied to memory due to a receive overrun.

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 15:8 | | FIFOCON | KILLBK | NBUSYBKE | | ERRORTRANSE | DATAXE | MDATAE |
| | | 23:16 | | | | | | RSTDT | | EPDISHDMA |
| | | 31:24 | | | | | | | | |
| 0x01E8 ... 0x01EF | Reserved | | | | | | | | | |
| 0x01F0 | USBHS_DEVEPTIER0 | 7:0 | SHORTPACKETES | STALLEDES | OVERFES | NAKINES | NAKOUTES | RXSTPES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | | | |
| | | 23:16 | | | | | STALLRQS | RSTDTS | NYETDISS | EPDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x01F0 | USBHS_DEVEPTIER0 (ISOENPT) | 7:0 | SHORTPACKETES | CRCERRES | OVERFES | HBISOFLUSHES | HBISOINERRES | UNDERFES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | ERRORTRANSES | DATAXES | MDATAES |
| | | 23:16 | | | | | | RSTDTS | | EPDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x01F4 | USBHS_DEVEPTIER1 | 7:0 | SHORTPACKETES | STALLEDES | OVERFES | NAKINES | NAKOUTES | RXSTPES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | | | |
| | | 23:16 | | | | | STALLRQS | RSTDTS | NYETDISS | EPDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x01F4 | USBHS_DEVEPTIER1 (ISOENPT) | 7:0 | SHORTPACKETES | CRCERRES | OVERFES | HBISOFLUSHES | HBISOINERRES | UNDERFES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | ERRORTRANSES | DATAXES | MDATAES |
| | | 23:16 | | | | | | RSTDTS | | EPDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x01F8 | USBHS_DEVEPTIER2 | 7:0 | SHORTPACKETES | STALLEDES | OVERFES | NAKINES | NAKOUTES | RXSTPES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | | | |
| | | 23:16 | | | | | STALLRQS | RSTDTS | NYETDISS | EPDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x01F8 | USBHS_DEVEPTIER2 (ISOENPT) | 7:0 | SHORTPACKETES | CRCERRES | OVERFES | HBISOFLUSHES | HBISOINERRES | UNDERFES | RXOUTES | TXINES |
| | | 15:8 | | FIFOCONS | KILLBKS | NBUSYBKES | | ERRORTRANSES | DATAXES | MDATAES |
| | | 23:16 | | | | | | RSTDTS | | EPDISHDMAS |
| | | 31:24 | | | | | | | | |

| Value | Description |
|---|---|
| 0 | No channel register is loaded after the end of the channel transfer. |
| 1 | The channel controller loads the next descriptor after the end of the current transfer, i.e., when the USBHS_DEVDMASTATUS.CHANN_ENB bit is reset. |

**Bit 0 – CHANN_ENB** Channel Enable Command

| Value | Description |
|---|---|
| 0 | The DMA channel is disabled at end of transfer and no transfer occurs upon request. This bit is also cleared by hardware when the channel source bus is disabled at end of buffer.<br>If the LDNXT_DSC bit has been cleared by descriptor loading, the firmware must set the corresponding CHANN_ENB bit to start the described transfer, if needed.<br>If the LDNXT_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both USBHS_DEVDMASTATUS.CHANN_ENB and CHANN_ACT flags read as 0.<br>If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the USBHS_DEVDMASTATUS.CHANN_ENB bit is cleared.<br>If the LDNXT_DSC bit is set at or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded. |
| 1 | The USBHS_DEVDMASTATUS.CHANN_ENB bit is set, thus enabling the DMA channel data transfer. Then, any pending request starts the transfer. This may be used to start or resume any requested transfer. |

### 40.14.19 HSMCI Write Protection Status Register

**Name:** HSMCI_WPSR
**Offset:** 0xE8
**Reset:** 0x0
**Property:** Read-only

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | WPVSRC[15:8] | | | | | | | |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | WPVSRC[7:0] | | | | | | | |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

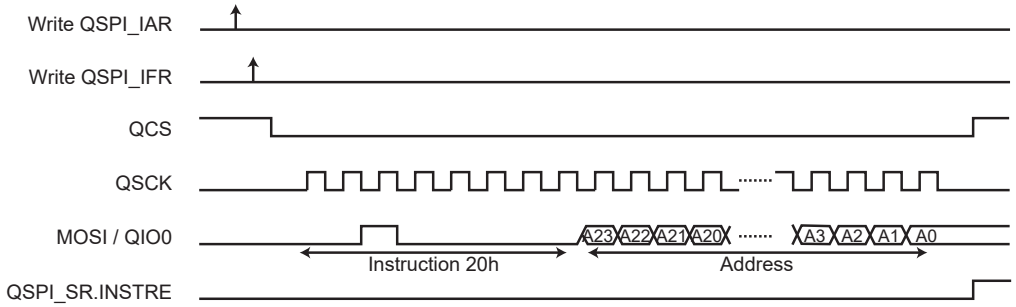
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | WPVS |
| Access | | | | | | | | |
| Reset | | | | | | | | 0 |

**Bits 23:8 – WPVSRC[15:0]** Write Protection Violation Source
When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

**Bit 0 – WPVS** Write Protection Violation Status

| Value | Description |
|---|---|
| 0 | No write protection violation has occurred since the last read of the HSMCI_WPSR. |
| 1 | A write protection violation has occurred since the last read of the HSMCI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC. |

**Figure 42-13. Instruction Transmission Waveform 3**



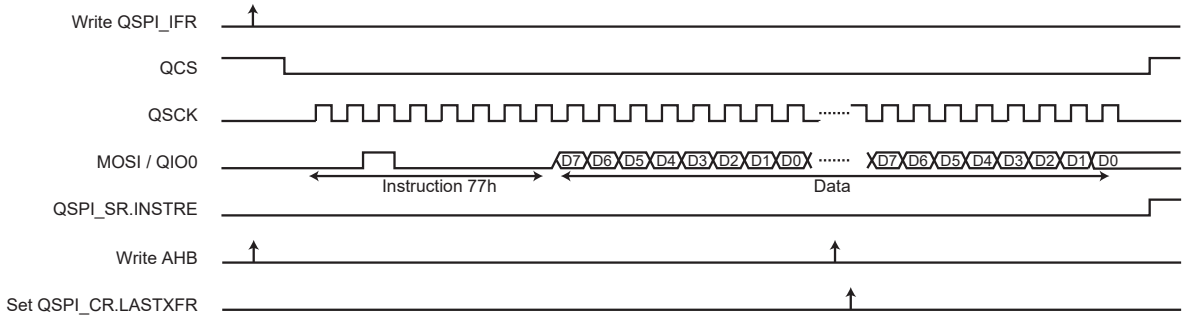

Example 4:

Instruction in Single-bit SPI, without address, without option, with data write in Single-bit SPI.

Command: SET BURST (77h)

- Write 0x0000_0077 in QSPI_ICR.
- Write 0x0000_2090 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Write data in the system bus memory space (0x80000000).
  The address of system bus write accesses is not used.
- Write a ‘1’ to QSPI_CR.LASTXFR.
- Wait for QSPI_SR.INSTRE to rise.

**Figure 42-14. Instruction Transmission Waveform 4**




Example 5:

Instruction in Single-bit SPI, with address in Dual SPI, without option, with data write in Dual SPI.

Command: BYTE/PAGE PROGRAM (02h)

- Write 0x0000_0002 in QSPI_ICR.
- Write 0x0000_30B3 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Write data in the QSPI system bus memory space (0x80000000).
  The address of the first system bus write access is sent in the instruction frame.
  The address of the next system bus write accesses is not used.
- Write a ‘1’ to QSPI_CR.LASTXFR.
- Wait for QSPI_SR.INSTRE to rise.

### 42.7.3 QSPI Receive Data Register

**Name:** QSPI_RDR
**Offset:** 0x08
**Reset:** 0x00000000
**Property:** Read-only

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | RD[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RD[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – RD[15:0]** Receive Data
Data received by the QSPI is stored in this register right-justified. Unused bits read zero.

See Master Write with One-, Two- or Three-Byte Internal Address and One Data Byte and Internal Address Usage for the master write operation with internal address.

The three internal address bytes are configurable through TWIHS_MMR.

If the slave device supports only a 7-bit address, i.e., no internal address, IADRSZ must be set to 0.

The table below shows the abbreviations used in the figures below.

**Table 43-4. Abbreviations**

| Abbreviation | Definition |
|---|---|
| S | Start |
| Sr | Repeated Start |
| P | Stop |
| W | Write |
| R | Read |
| A | Acknowledge |
| NA | Not Acknowledge |
| DADR | Device Address |
| IADR | Internal Address |

**Figure 43-10. Master Write with One-, Two- or Three-Byte Internal Address and One Data Byte**

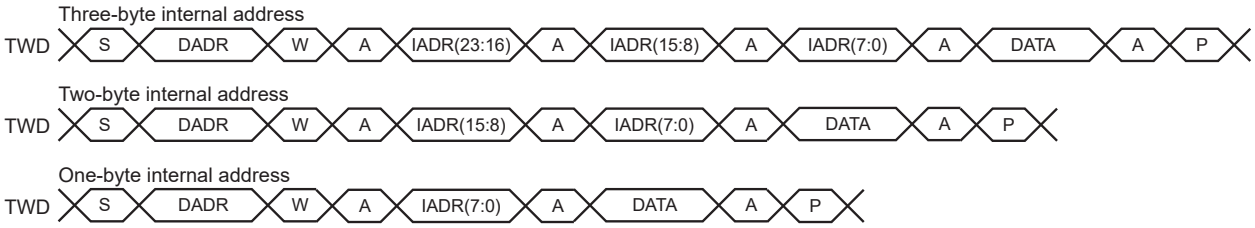
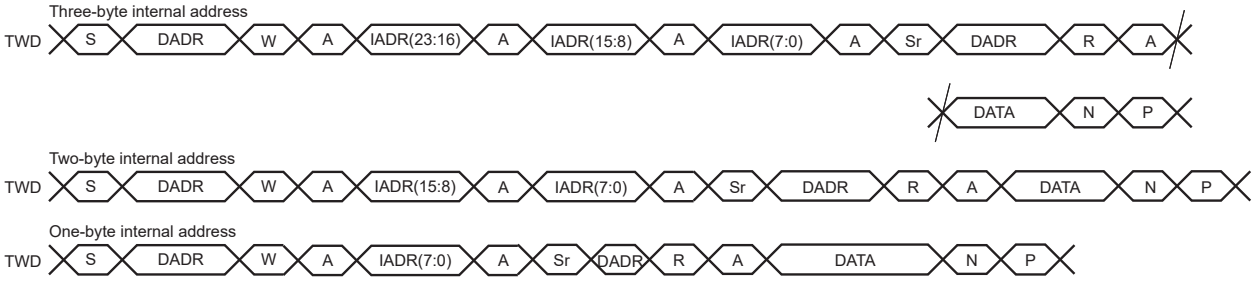



**Figure 43-11. Master Read with One-, Two- or Three-Byte Internal Address and One Data Byte**




#### 43.6.3.5.2 10-bit Slave Addressing

For a slave address higher than seven bits, configure the address size (IADRSZ) and set the other slave address bits in the Internal Address register (TWIHS_IADR). The two remaining internal address bytes, IADR[15:8] and IADR[23:16], can be used the same way as in 7-bit slave addressing.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)

A byte-wide value sent by the receiving (Rx) MediaLB Device on the MLBS line, after Command is sent. This status response provides a hardware handshaking mechanism and signals other control information, such as transmission errors, back to the sender.

- Data:

The physical channel contains Data and is sent by the Tx MediaLB Device during the same physical channel in which Command is sent. This physical channel data must be transmitted left-justified, MSB first, most significant byte first. Note the Rx Device might return a status of busy, wherein the Tx Device must retransmit the same data in the next physical channel associated with the logical channel.

To dynamically configure ChannelAddresses for logical channels, a DeviceAddress can be pre-defined for MediaLB Devices. The DeviceAddress is a 16-bit address used in the System Channel with the MLBScan command to detect which MediaLB Devices exist.

#### 48.6.1.1 Channel Addresses

A MediaLB logical channel is defined as all physical channels associated with a single ChannelAddress. A logical channel on MediaLB is unidirectional; therefore, a single MediaLB Device sends data on a logical channel to one or more receiving Devices. If two Devices require bidirectional communication, then two MediaLB logical channels are required.

A ChannelAddress is 16-bits wide. Of the 16-bits, ChannelAddress (CA) bits 15 through 9 and the LSB are always zero. Only the eight bits CA[8:1] vary. A delay of one physical channel exists between the occurrence of the ChannelAddress and the actual physical channel granted. The 0x01FE ChannelAddress is defined as the FRAMESYNC pattern, where the end of the pattern determines the byte boundary, the physical channel boundary, and indicates that the MediaLB frame starts one physical channel later (PC0). The 0x0000 ChannelAddress is defined as the BusIdle state, which indicates that the corresponding physical channel is not assigned and not used by any Device. All odd ChannelAddresses are reserved; therefore, the LSB of a valid ChannelAddress is always zero. The MLBS line is in a consistent known state when not driven by any Device. For 3-pin MediaLB, this is achieved with the required weak pull-down.

**Table 48-3. MediaLB ChannelAddresses**

| ChannelAddress (1) | Description |
|---|---|
| 0x0000 | BusIdle - Indicates that the physical channel is not being used, not assigned. |
| 0x0002..0x007E | 63 ChannelAddresses - defines the logical channels used in normal operation (3-pin MediaLB) |
| 0x0080..0x01FC | Reserved |
| 0x01FE | FRAMESYNC - MediaLB frame alignment and System Channel ChannelAddress |
| 0x0200..0xFFFF | Reserved |

Note: 1. All odd ChannelAddresses are reserved (LSB must be zero for valid ChannelAddresses).

#### 48.6.1.2 Device Addresses

DeviceAddresses are 16-bits wide, must be pre-assigned, and must be unique for each MediaLB Device. Of the 16-bits, DeviceAddress (DA) bits 15 through 9 and the LSB are always zero. Only the eight bits DA[8:1] vary. At the request of the EHC, DeviceAddresses can be scanned for by the MediaLB Controller to dynamically determine which Devices exist on MediaLB. DeviceAddresses are only used with the MLBScan command in the System Channel and are never assigned to physical channels. Once a Device is found, the ChannelAddresses used in normal operation can be assigned.

- Finish any active MLB transfer
- Disable MLB (clear the MLBEN and MLBPEN bits in MLB_MLBC0)
- Disable HBI (clear all bits in MLB_HCMR0 and MLB_HCMR1, clear EN bit in MLB_HCTL)
- Mask AHB interrupts (clear all bits in MLB_ACMR0 and MLB_ACMR1)

For information on configuring the MLB IP if the clocks are re-enabled, see Section “Configure the Hardware”.

### 51.7.9 PWM Sync Channels Mode Register

**Name:** PWM_SCM
**Offset:** 0x20
**Reset:** 0x00000000
**Property:** Read/Write

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the PWM Write Protection Status Register.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | PTRCS[2:0] | | | PTRM | | | UPDM[1:0] | |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SYNC3 | SYNC2 | SYNC1 | SYNC0 |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 23:21 – PTRCS[2:0]** DMA Controller Transfer Request Comparison Selection
Selection of the comparison used to set the flag WRDY and the corresponding DMA Controller transfer request.

**Bit 20 – PTRM** DMA Controller Transfer Request Mode

| UPDM | PTRM | WRDY Flag and DMA Controller Transfer Request |
|---|---|---|
| 0 | x | The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are never set to ‘1’. |
| 1 | x | The WRDY flag in PWM Interrupt Status Register 2 is set to ‘1’ as soon as the update period is elapsed, the DMA Controller transfer request is never set to ‘1’. |
| 2 | 0 | The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are set to ‘1’ as soon as the update period is elapsed. |
| | 1 | The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are set to ‘1’ as soon as the selected comparison matches. |

**Bits 17:16 – UPDM[1:0]** Synchronous Channels Update Mode

### 51.7.31 PWM Spread Spectrum Update Register

**Name:** PWM_SSPUP
**Offset:** 0xA4
**Reset:** –
**Property:** Write-only

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the PWM Write Protection Status Register.

This register acts as a double buffer for the SPRD value. This prevents an unexpected waveform when modifying the spread spectrum limit value.

Only the first 16 bits (channel counter size) are significant.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

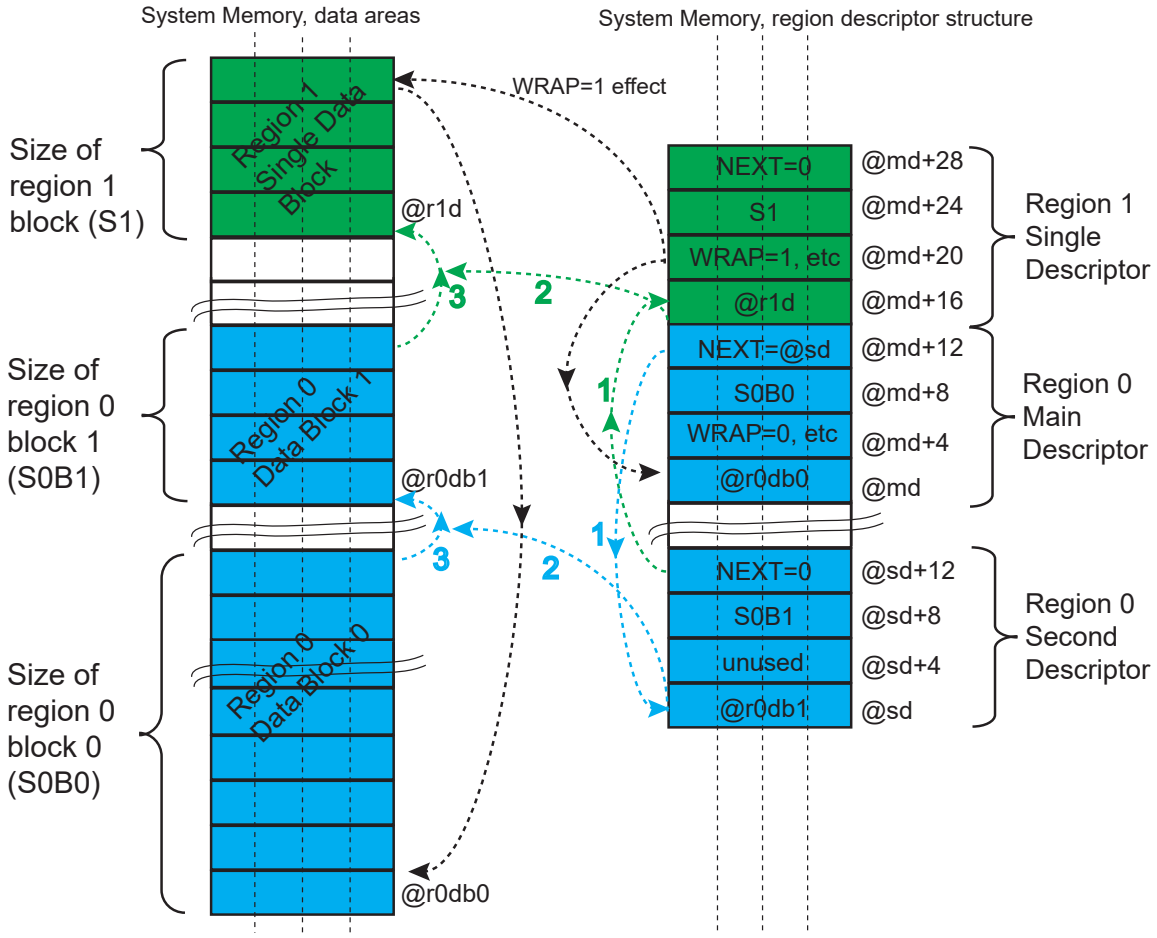
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | SPRDUP[23:16] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SPRDUP[15:8] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SPRDUP[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |

**Bits 23:0 – SPRDUP[23:0]** Spread Spectrum Limit Value Update
The spread spectrum limit value defines the range for the spread spectrum counter. It is introduced in order to achieve constant varying period for the output waveform.

**Figure 55-5. Example: Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)**




### 55.5.2 ICM Region Descriptor Structure

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

**Table 55-1. Region Descriptor Structure (Main List)**

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