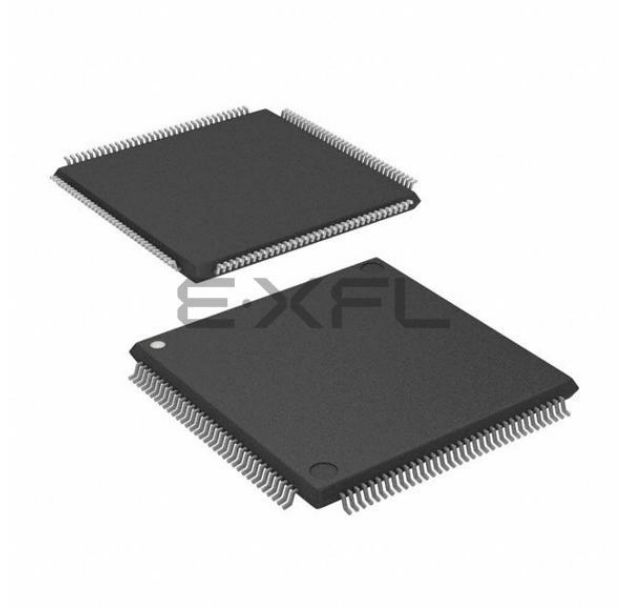


**Welcome to E-XFL.COM**

## 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 | CANbus, EBI/EMI, Ethernet, 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) | 1.62V ~ 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-LQFP |
| Supplier Device Package | 144-LQFP (20x20) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsame70q20b-ant |

Since this function is executed from ROM, this allows Flash programming (such as sector write) to be done by code running in Flash.

The IAP function entry point is retrieved by reading the NMI vector in ROM (0x00800008).

This function takes two arguments as parameters:

- the index of the Flash bank to be programmed: 0 for EEFC0, 1 for EEFC1. For devices with only one bank, this parameter has no effect and can be either 0 or 1, only EEFC0 will be accessed.
- the command to be sent to the EEFC Command register.

This function returns the value of the EEFC_FSR register.

An example of IAP software code follows:

```
// Example: How to write data in page 200 of the flash memory using ROM IAP
function

flash_page_num = 200

flash_cmd = 0

flash_status = 0

eefc_index = 0 (0 for EEFC0, 1 for EEFC1)

// Initialize the function pointer (retrieve function address from NMI
vector)*/

iap_function_address = 0x00800008

// Fill the Flash page buffer at address 200 with the data to be written

for i=0, i < page_size, i++ do

flash_sector_200_address[i] = your_data[i]

// Prepare the command to be sent to the EEFC Command register: key, page
number and write command

flash_cmd = (0x5A << 24) | (flash_page_num << 8) | flash_write_command;

// Call the IAP function with the right parameters and retrieve the status in
flash_status after completion

flash_status = iap_function (eefc_index, flash_cmd);
```

| Value | Description |
|---|---|
| 0 | The detection of a low level on the NRST pin does not generate a user reset. |
| 1 | The detection of a low level on the NRST pin triggers a user reset. |

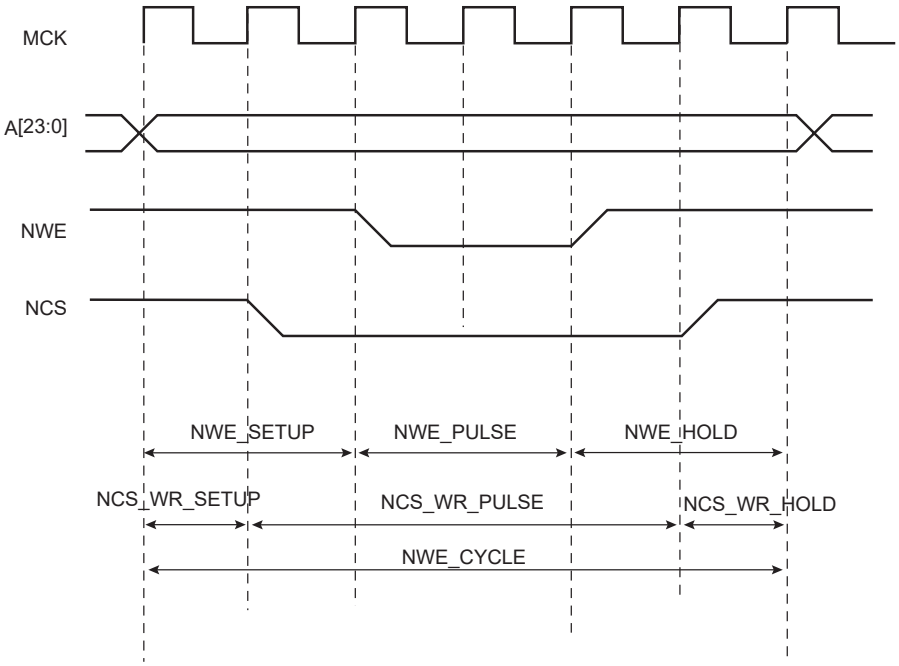
- NWE_SETUP—the NWE setup time is defined as the setup of address and data before the NWE falling edge;
- NWE_PULSE—the NWE pulse length is the time between NWE falling edge and NWE rising edge;
- NWE_HOLD—the NWE hold time is defined as the hold time of address and data after the NWE rising edge.

#### 35.9.3.2 NCS Waveforms

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

- ncs_wr_setup—the NCS setup time is defined as the setup time of address before the NCS falling edge.
- ncs_wr_pulse—the NCS pulse length is the time between NCS falling edge and NCS rising edge;
- ncs_wr_hold—the NCS hold time is defined as the hold time of address after the NCS rising edge.

**Figure 35-13. Write Cycle**




#### 35.9.3.3 Write Cycle

The write_cycle time is defined as the total duration of the write cycle; that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is defined as:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD,

as well as

NWE_CYCLE = NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. The NWE_CYCLE field is common to both the NWE and NCS signals, thus the timing period is of the same duration.

NWE_CYCLE, NWE_SETUP, and NWE_PULSE implicitly define the NWE_HOLD value as:

NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

| Peripheral Name | Transfer Type | HW Interface Number (XDMAC_CC.PERID) |
|---|---|---|
| I2SC0 | Receive Left | 45 |
| I2SC1 | Transmit Left | 46 |
| I2SC1 | Receive Left | 47 |
| I2SC0 | Transmit Right | 48 |
| I2SC0 | Receive Right | 49 |
| I2SC1 | Transmit Right | 50 |
| I2SC1 | Receive Right | 51 |

## 36.5 Functional Description

### 36.5.1 Basic Definitions

**Source Peripheral**: Slave device, memory mapped on the interconnection network, from where the XDMAC reads data. The source peripheral teams up with a destination peripheral to form a channel. A data read operation is scheduled when the peripheral transfer request is asserted.

**Destination Peripheral**: Slave device, memory mapped on the interconnection network, to which the XDMAC writes. A write data operation is scheduled when the peripheral transfer request is asserted.

**Channel**: The data movement between source and destination creates a logical channel.

**Transfer Type**: The transfer is hardware-synchronized when it is paced by the peripheral hardware request, otherwise the transfer is self-triggered (memory to memory transfer).

### 36.5.2 Transfer Hierarchy Diagram

**XDMAC Master Transfer**: The Master Transfer is composed of a linked list of blocks. The channel address, control and configuration registers can be modified at the inter block boundary. The descriptor structure modifies the channel registers conditionally. Interrupts can be generated on a per block basis or when the end of linked list event occurs.

**XDMAC Block**: An XDMAC block is composed of a programmable number of microblocks. The channel configuration registers remain unchanged at the inter microblock boundary. The source and destination addresses are conditionally updated with a programmable signed number.

**XDMAC Microblock**: The microblock is composed of a programmable number of data. The channel configuration registers remain unchanged at the data boundary. The data address may be fixed (a FIFO location, a peripheral transmit or receive register), incrementing (a memory-mapped area) by a programmable signed number.

**XDMAC Burst and Incomplete Burst**: In order to improve the overall performance when accessing dynamic external memory, burst access is mandatory. Each data of the microblock is considered as a part of a memory burst. The programmable burst value indicates the largest memory burst allowed on a per channel basis. When the microblock length is not an integral multiple of the burst size, an incomplete burst is performed to read or write the last trailing bytes.

**XDMAC Chunk and Incomplete Chunk**: When a peripheral synchronized transfer is activated, the microblock splits into a number of data chunks. The chunk size is programmable. The larger the chunk is,

### 36.9.28 XDMAC Channel x Configuration Register [x = 0..23]

**Name:** XDMAC_CC
**Offset:** 0x78 + n*0x40 [n=0..23]
**Reset:** 0x00000000
**Property:** Read/Write

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | | PERID[6:0] | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | WRIP | RDIP | INITD | | DAM[1:0] | | SAM[1:0] | |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | DIF | SIF | DWIDTH[1:0] | | CSIZE[2:0] | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MEMSET | SWREQ | | DSYNC | | MBSIZE[1:0] | | TYPE |
| Access | R/W | R/W | | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | | 0 | | 0 | 0 | 0 |

**Bits 30:24 – PERID[6:0]** Channel x Peripheral Hardware Request Line Identifier
This field contains the peripheral hardware request line identifier. PERID refers to identifiers defined in "DMA Controller Peripheral Connections".

**Bit 23 – WRIP** Write in Progress (this bit is read-only)
0 (DONE): No active write transaction on the bus.

1 (IN_PROGRESS): A write transaction is in progress.

**Bit 22 – RDIP** Read in Progress (this bit is read-only)
0 (DONE): No active read transaction on the bus.

1 (IN_PROGRESS): A read transaction is in progress.

**Bit 21 – INITD** Channel Initialization Done (this bit is read-only)
0 (IN_PROGRESS): Channel initialization is in progress.

1 (TERMINATED): Channel initialization is completed.

Note: When set to 0, XDMAC_CUBC.UBLEN and XDMAC_CNDA.NDA field values are unreliable each time a descriptor is being updated. See 36.8 XDMAC Software Requirements.

**Bits 19:18 – DAM[1:0]** Channel x Destination Addressing Mode

2. Allocate an area 8N bytes for the transmit buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GMAC, i.e., bit 31 of word 1 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit 30 in word 1 set to 1).
4. Write address of transmit buffer descriptor list and control information to GMAC register transmit buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the Network Control register.

**Note:** The queue pointers must be initialized and point to USED descriptors for all queues including those not intended for use.

#### 38.7.1.4 Address Matching

The GMAC Hash register pair and the four Specific Address register pairs must be written with the required values. Each register pair comprises of a bottom register and top register, with the bottom register being written first. The address matching is disabled for a particular register pair after the bottom register has been written and re-enabled when the top register is written. Each register pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

As an example, to set Specific Address register 1 to recognize destination address 21:43:65:87:A9:CB, the following values are written to Specific Address register 1 bottom and Specific Address register 1 top:

- Specific Address register 1 bottom bits 31:0 (0x98): 0x8765_4321.
- Specific Address register 1 top bits 31:0 (0x9C): 0x0000_CBA9.
  **Note:** The address matching is the first level of filtering. If there is a match, the screeners are the next level of filtering for routing the data to the appropriate queue. See Priority Queueing in the DMA for more details.

#### 38.7.1.5 PHY Maintenance

The PHY Maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the Network Status register (about 2000 MCK cycles later when bits 18:16 are set to 010 in the Network Configuration register). An interrupt is generated as this bit is set.

During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each Management Data Clock (MDC) cycle. This causes the transmission of a PHY management frame on MDIO. See section 22.2.4.5 of the IEEE 802.3 standard.

Reading during the shift operation will return the current contents of the shift register. At the end of the management operation the bits will have shifted back to their original locations. For a read operation the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The Management Data Clock (MDC) should not toggle faster than 2.5 MHz (minimum period of 400 ns), as defined by the IEEE 802.3 standard. MDC is generated by dividing down MCK. Three bits in the Network Configuration register determine by how much MCK should be divided to produce MDC.

#### 38.7.1.6 Interrupts

There are 18 interrupt conditions that are detected within the GMAC. The conditions are ORed to make multiple interrupts. Depending on the overall system design this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler. Refer to the device interrupt controller documentation to identify that it is the GMAC that is generating the interrupt. To ascertain which interrupt, read the Interrupt Status register. Note that in the default configuration this register will clear itself after being read, though this may be configured to be write-one-to-clear if desired.

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0620 | GMAC_IDRPQ1 | 7:0 | TCOMP | TFC | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0624 | GMAC_IDRPQ2 | 7:0 | TCOMP | TFC | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0628 | GMAC_IDRPQ3 | 7:0 | TCOMP | TFC | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x062C | GMAC_IDRPQ4 | 7:0 | TCOMP | TFC | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0630 | GMAC_IDRPQ5 | 7:0 | TCOMP | TFC | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0634 ... 0x063F | Reserved | | | | | | | | | |
| 0x0640 | GMAC_IMRPQ1 | 7:0 | TCOMP | AHB | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0644 | GMAC_IMRPQ2 | 7:0 | TCOMP | AHB | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0648 | GMAC_IMRPQ3 | 7:0 | TCOMP | AHB | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x064C | GMAC_IMRPQ4 | 7:0 | TCOMP | AHB | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0650 | GMAC_IMRPQ5 | 7:0 | TCOMP | AHB | RLEX | | | RXUBR | RCOMP | |
| | | 15:8 | | | | | HRESP | ROVR | | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x0654 ... 0x06DF | Reserved | | | | | | | | | |

The USBHS can be frozen when the USB line is in the Suspend mode, by writing a one to the USBHS_CTRL.FRZCLK bit, which reduces power consumption.

In this case, it is still possible to access the following:

- USBHS_CTRL.FRZCLK, USBHS_CTRL.USBE and USBHS_DEVCTRL.LS bits

Moreover, when USBHS_CTRL.FRZCLK = 1, only the asynchronous interrupt sources can trigger the USB interrupt:

- Wakeup Interrupt (USBHS_DEVISR.WAKEUP)
- Host Wakeup Interrupt (USBHS_HSTISR.HWUPI)

#### 39.5.1.4 Speed Control

Device Mode

When the USB interface is in Device mode, the speed selection (Full-speed or High-speed) is performed automatically by the USBHS during the USB reset according to the host speed capability. At the end of the USB reset, the USBHS enables or disables high-speed terminations and pull-up.

It is possible to set the USBHS to Full-speed or Low-speed mode via USBHS_DEVCTRL.LS and USBHS_DEVCTRL.SPDCONF.

Host Mode

When the USB interface is in Host mode, internal pull-down resistors are connected on both D+ and D- and the interface detects the speed of the connected device, which is reflected by the Speed Status (USBHS_SR.SPEED) field.

#### 39.5.1.5 DPRAM Management

Pipes and endpoints can only be allocated in ascending order, from pipe/endpoint 0 to the last pipe/endpoint to be allocated. The user should therefore configure them in the same order.

The allocation of a pipe/endpoint x starts when the Endpoint Memory Allocate bit in the Endpoint x Configuration register (USBHS_DEVEPTCFGx.ALLOC) is written to one. Then, the hardware allocates a memory area in the DPRAM and inserts it between the x - 1 and x+ 1 pipes/endpoints. The x+ 1 pipe/endpoint memory window slides up and its data is lost. Note that the following pipe/endpoint memory windows (from x+ 2) do not slide.

Disabling a pipe, by writing a zero to the Pipe x Enable bit in the Host Pipe register (USBHS_HSTPIP.PENx), or disabling an endpoint, by writing a zero to the Endpoint x Enable bit in the Device Endpoint register (USBHS_DEVEPT.EPENx), does not reset the USBHS_DEVEPTCFGx.ALLOC bit or the Pipe/Endpoint configuration:

- Pipe Configuration
  - Pipe Banks (USBHS_HSTPIPCFGx.PBK)
  - Pipe Size (USBHS_HSTPIPCFGx.PSIZE)
  - Pipe Token (USBHS_HSTPIPCFGx.PTOKEN)
  - Pipe Type (USBHS_HSTPIPCFGx.PTYPE)
  - Pipe Endpoint Number (USBHS_HSTPIPCFGx.PEPNUM)
  - Pipe Interrupt Request Frequency (USBHS_HSTPIPCFGx.INTFRQ)
- Endpoint Configuration
  - Endpoint Banks (USBHS_DEVEPTCFGx.EPBK)
  - Endpoint Size (USBHS_DEVEPTCFGx. EPSIZE)

#### 39.5.3.4 USB Reset

The USBHS sends a USB bus reset when the user writes a one to the Send USB Reset bit in the Host General Control register (USBHS_HSTCTRL.RESET). The USB Reset Sent Interrupt bit in the Host Global Interrupt Status register (USBHS_HSTISR.RSTI) is set when the USB reset has been sent. In this case, all pipes are disabled and de-allocated.

If the bus was previously in a “Suspend” state (the Start of Frame Generation Enable (USBHS_HSTCTRL.SOFE) bit is zero), the USBHS automatically switches to the “Resume” state, the Host Wakeup Interrupt (USBHS_HSTISR.HWUPI) bit is set and the USBHS_HSTCTRL.SOFE bit is set in order to generate SOFs or micro SOFs immediately after the USB reset.

At the end of the reset, the user should check the USBHS_SR.SPEED field to know the speed running according to the peripheral capability (LS.FS/HS).

#### 39.5.3.5 Pipe Reset

A pipe can be reset at any time by writing a one to the Pipe x Reset (USBHS_HSTPIP.PRSTx) bit. This is recommended before using a pipe upon hardware reset or when a USB bus reset has been sent. This resets:

- the internal state machine of the pipe,
- the receive and transmit bank FIFO counters,
- all the registers of the pipe (USBHS_HSTPIPCFGx, USBHS_HSTPIPISRx, USBHS_HSTPIPIMRx), except its configuration (USBHS_HSTPIPCFGx.ALLOC, USBHS_HSTPIPCFGx.PBK, USBHS_HSTPIPCFGx.PSIZE, USBHS_HSTPIPCFGx.PTOKEN, USBHS_HSTPIPCFGx.PTYPE, USBHS_HSTPIPCFGx.PEPNUM, USBHS_HSTPIPCFGx.INTFRQ) and its Data Toggle Sequence field (USBHS_HSTPIPISRx.DTSEQ).

The pipe configuration remains active and the pipe is still enabled.

The pipe reset may be associated with a clear of the data toggle sequence. This can be achieved by setting the Reset Data Toggle bit in the Pipe x Control register (USBHS_HSTPIPIMRx.RSTDT) (by writing a one to the Reset Data Toggle Set bit in the Pipe x Control Set register (USBHS_HSTPIPIERx.RSTDTS)).

In the end, the user has to write a zero to the USBHS_HSTPIP.PRSTx bit to complete the reset operation and to start using the FIFO.

#### 39.5.3.6 Pipe Activation

The pipe is maintained inactive and reset (see "Pipe Reset" for more details) as long as it is disabled (USBHS_HSTPIP.PENx = 0). The Data Toggle Sequence field (USBHS_HSTPIPISRx.DTSEQ) is also reset.

The algorithm represented in the following figure must be followed to activate a pipe.

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0600 | USBHS_HSTPIPIER4 (INTPIPES) | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0600 | USBHS_HSTPIPIER4 (ISOPIPES) | 7:0 | SHORTPACKETIES | CRCERRES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0604 | USBHS_HSTPIPIER5 | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | TXSTPES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0604 | USBHS_HSTPIPIER5 (INTPIPES) | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0604 | USBHS_HSTPIPIER5 (ISOPIPES) | 7:0 | SHORTPACKETIES | CRCERRES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0608 | USBHS_HSTPIPIER6 | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | TXSTPES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0608 | USBHS_HSTPIPIER6 (INTPIPES) | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x0608 | USBHS_HSTPIPIER6 (ISOPIPES) | 7:0 | SHORTPACKETIES | CRCERRES | OVERFIES | NAKEDES | PERRES | UNDERFIES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |
| 0x060C | USBHS_HSTPIPIER7 | 7:0 | SHORTPACKETIES | RXSTALLDES | OVERFIES | NAKEDES | PERRES | TXSTPES | TXOUTES | RXINES |
| | | 15:8 | | | | NBUSYBKES | | | | |
| | | 23:16 | | | | | | RSTDTS | PFREEZES | PDISHDMAS |
| | | 31:24 | | | | | | | | |

### 40.14.20 HSMCI FIFOx Memory Aperture

**Name:** HSMCI_FIFOx [x=0..255]
**Offset:** 0x00
**Reset:** 0
**Property:** R/W

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | DATA[31:24] | | | | | | | |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

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

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

**Bits 31:0 – DATA[31:0]** Data to Read or Data to Write

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

The WPVS bit is automatically cleared after reading TWIHS_WPSR.

The following registers can be write-protected:

- TWIHS Clock Waveform Generator Register

| Offset | Name | Bit Pos. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 31:24 | LINHTE | LINSTE | LINSNRE | LINCE | LINIPE | LINISFE | LINBE | |
| 0x0C | US_IDR (LON_MODE) | 7:0 | LCRCE | LSFE | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | | | | | | UNRE | TXEMPTY | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | LBLOVFE | LRXD | LFET | LCOL | LTXD |
| 0x10 | US_IMR | 7:0 | PARE | FRAME | OVRE | | | RXBRK | TXRDY | RXRDY |
| | | 15:8 | | | NACK | | | ITER | TXEMPTY | TIMEOUT |
| | | 23:16 | | | | | CTSIC | DCDIC | DSRIC | RIIC |
| | | 31:24 | | | | | | | | MANE |
| 0x10 | US_IMR (SPI_MODE) | 7:0 | | | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | | | | | | UNRE | TXEMPTY | |
| | | 23:16 | | | | | NSSE | | | |
| | | 31:24 | | | | | | | | |
| 0x10 | US_IMR (LIN_MODE) | 7:0 | PARE | FRAME | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | LINTC | LINID | LINBK | | | | TXEMPTY | TIMEOUT |
| | | 23:16 | | | | | | | | |
| | | 31:24 | LINHTE | LINSTE | LINSNRE | LINCE | LINIPE | LINISFE | LINBE | |
| 0x10 | US_IMR (LON_MODE) | 7:0 | LCRCE | LSFE | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | | | | | | UNRE | TXEMPTY | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | LBLOVFE | LRXD | LFET | LCOL | LTXD |
| 0x14 | US_CSR | 7:0 | PARE | FRAME | OVRE | | | RXBRK | TXRDY | RXRDY |
| | | 15:8 | | | NACK | | | ITER | TXEMPTY | TIMEOUT |
| | | 23:16 | CTS | DCD | DSR | RI | CTSIC | DCDIC | DSRIC | RIIC |
| | | 31:24 | | | | | | | | MANERR |
| 0x14 | US_CSR (SPI_MODE) | 7:0 | | | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | | | | | | UNRE | TXEMPTY | |
| | | 23:16 | NSS | | | | NSSE | | | |
| | | 31:24 | | | | | | | | |
| 0x14 | US_CSR (LIN_MODE) | 7:0 | PARE | FRAME | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | LINTC | LINID | LINBK | | | | TXEMPTY | TIMEOUT |
| | | 23:16 | LINBLS | | | | | | | |
| | | 31:24 | LINHTE | LINSTE | LINSNRE | LINCE | LINIPE | LINISFE | LINBE | |
| 0x14 | US_CSR (LON_MODE) | 7:0 | LCRCE | LSFE | OVRE | | | | TXRDY | RXRDY |
| | | 15:8 | | | | | | UNRE | TXEMPTY | |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | LBLOVFE | LRXD | LFET | LCOL | LTXD |
| 0x18 | US_RHR | 7:0 | RXCHR[7:0] | | | | | | | |
| | | 15:8 | RXSYNH | | | | | | | RXCHR[8:8] |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x1C | US_THR | 7:0 | TXCHR[7:0] | | | | | | | |
| | | 15:8 | TXSYNH | | | | | | | TXCHR[8:8] |
| | | 23:16 | | | | | | | | |
| | | 31:24 | | | | | | | | |
| 0x20 | US_BRGR | 7:0 | CD[7:0] | | | | | | | |

### 46.7.33 USART LIN Identifier Register

**Name:** US_LINIR
**Offset:** 0x0058
**Reset:** 0x0
**Property:** Read/Write(1)

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

| 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 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

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

**Bits 7:0 – IDCHR[7:0]** Identifier Character
If USART_MODE = 0xA (master node configuration):

IDCHR is Read/Write and its value is the identifier character to be transmitted.

If USART_MODE = 0xB (slave node configuration):

IDCHR is Read-only and its value is the last identifier character that has been received.

### 49.6.1 MCAN Core Release Register

**Name:** MCAN_CREL
**Offset:** 0x00
**Reset:** 0xrrrddddd
**Property:** Read-only

Due to clock domain crossing, there is a delay between when a register bit or field is written and when the related status register bits are updated.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | REL[3:0] | | | | STEP[3:0] | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | SUBSTEP[3:0] | | | | YEAR[3:0] | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

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

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

**Bits 31:28 – REL[3:0]** Core Release
One digit, BCD-coded.

**Bits 27:24 – STEP[3:0]** Step of Core Release
One digit, BCD-coded.

**Bits 23:20 – SUBSTEP[3:0]** Sub-step of Core Release
One digit, BCD-coded.

**Bits 19:16 – YEAR[3:0]** Timestamp Year
One digit, BCD-coded. This field is set by generic parameter on MCAN synthesis.

**Bits 15:8 – MON[7:0]** Timestamp Month
Two digits, BCD-coded. This field is set by generic parameter on MCAN synthesis.

**Bits 7:0 – DAY[7:0]** Timestamp Day
Two digits, BCD-coded. This field is set by generic parameter on MCAN synthesis.

### 49.6.30 MCAN Receive Buffer Configuration

**Name:** MCAN_RXBC
**Offset:** 0xAC
**Reset:** 0x00000000
**Property:** Read/Write

| 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 |
|---|---|---|---|---|---|---|---|---|
| | RBSA[13:6] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Bits 15:2 – RBSA[13:0]** Receive Buffer Start Address
Configures the start address of the Receive Buffers section in the Message RAM (32-bit word address, see Message RAM Configuration). Also used to reference debug messages A,B,C.

Write RBSA with the bits [15:2] of the 32-bit address.

### 51.7.10 PWM DMA Register

**Name:** PWM_DMAR
**Offset:** 0x24
**Reset:** –
**Property:** Write-only

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 |
|---|---|---|---|---|---|---|---|---|
| | DMADUTY[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 |
|---|---|---|---|---|---|---|---|---|
| | DMADUTY[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 |
|---|---|---|---|---|---|---|---|---|
| | DMADUTY[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |

**Bits 23:0 – DMADUTY[23:0]** Duty-Cycle Holding Register for DMA Access
Each write access to PWM_DMAR sequentially updates PWM_CDTYUPDx.CDTYUPD with DMADUTY (only for channel configured as synchronous). See “Method 3: Automatic write of duty-cycle values and automatic trigger of the update” .

| Value | Description |
|---|---|
| 0 | Clears CHNB in AFEC_LCDR. |
| 1 | Appends the channel number to the conversion result in AFEC_LCDR. |

**Bits 18:16 – RES[2:0]** Resolution

| Value | Name | Description |
|---|---|---|
| 0 | NO_AVERAGE | 12-bit resolution, AFE sample rate is maximum (no averaging). |
| 1 | LOW_RES | 10-bit resolution, AFE sample rate is maximum (no averaging). |
| 2 | OSR4 | 13-bit resolution, AFE sample rate divided by 4 (averaging). |
| 3 | OSR16 | 14-bit resolution, AFE sample rate divided by 16 (averaging). |
| 4 | OSR64 | 15-bit resolution, AFE sample rate divided by 64 (averaging). |
| 5 | OSR256 | 16-bit resolution, AFE sample rate divided by 256 (averaging). |

**Bits 13:12 – CMPFILTER[1:0]** Compare Event Filtering
Number of consecutive compare events necessary to raise the flag = CMPFILTER+1.

When programmed to ‘0’, the flag rises as soon as an event occurs.

**Bit 9 – CMPALL** Compare All Channels

| Value | Description |
|---|---|
| 0 | Only the channel indicated in CMPSEL field is compared. |
| 1 | All channels are compared. |

**Bits 7:3 – CMPSEL[4:0]** Comparison Selected Channel
If CMPALL = 0: CMPSEL indicates which channel has to be compared.

If CMPALL = 1: No effect.

**Bits 1:0 – CMPMODE[1:0]** Comparison Mode

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Generates an event when the converted data is lower than the low threshold of the window. |
| 1 | HIGH | Generates an event when the converted data is higher than the high threshold of the window. |
| 2 | IN | Generates an event when the converted data is in the comparison window. |
| 3 | OUT | Generates an event when the converted data is out of the comparison window. |

The DATA code in AFEC_CDR is up to 16-bit positive integer or two's complement (signed integer).

The code does not exceed 4095 when the field AFEC_EMR.RES=0 (12-bit mode, no averaging).

#### 59.8.4.1 Differential Mode (12-bit mode)

A differential input voltage $V_{IN} = V_{INP} - V_{INN}$ can be applied between two selected differential pins, e.g. AFE0_AD0 and AFE0_AD1.The ideal code $C_i$ is calculated by using the following formula and rounding the result to the nearest positive integer.

$$C_i = \frac{4096}{V_{\mathrm{VREFP}}} \times V_{\mathrm{IN}} \times \mathrm{Gain} + (2047)$$

For the other resolution defined by RES, the code $C_i$ is extended to the corresponding resolution.

The table below is a computation example for the above formula, where $V_{VREFP}$ = 3V.

**Table 59-32. Input Voltage Values in Differential Mode, Nonsigned Output**

| $C_i$ | | Gain | | |
|---|---|---|---|---|
| Signed | Nonsigned | 1 | 2 | 4 |
| -2048 | 0 | -3 | -1.5 | -0.75 |
| 0 | 2047 | 0 | 0 | 0 |
| 2047 | 4095 | 3 | 1.5 | 0.75 |

#### 59.8.4.2 Single-ended Mode (12-bit mode)

A single input voltage $V_{IN}$ can be applied to selected pins, e.g. AFE0_AD0 or AFE0_AD1. The ideal code $C_i$ is calculated using the following formula and rounding the result to the nearest positive integer.

The single-ended ideal code conversion formula is:

$$C_i = \frac{4096}{V_{\mathrm{VREFP}}} \times (V_{\mathrm{IN}} - V_{\mathrm{DAC}}) \times \mathrm{Gain} + 2047$$

For the other resolution defined by RES, the code $C_i$ is extended to the corresponding resolution.

The table below is a computation example for the above formula, where $V_{VREFP}$ = 3V:

**Table 59-33. Input Voltage Values in Single-ended Mode**

| $C_i$ | | Gain | | |
|---|---|---|---|---|
| Signed | Nonsigned | 1 | 2 | 4 |
| -2048 | 0 | 0 | 0.75 | 1.125 |
| 0 | 2047 | 1.5 | 1.5 | 1.5 |
| 2047 | 4095 | 3 | 2.25 | 1.875 |

#### 59.8.4.3 Example of LSB Computation

The LSB is relative to the analog scale $V_{VREFP}$.

The term LSB expresses the quantization step in volts, also used for one AFE code variation.

- Single-ended (SE) (ex: $V_{VREFP}$ = 3.0V)
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV

| Symbol | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | Unit |
|---|---|---|---|---|---|---|
| | Parameter | Min | | Max | | |
| NO HOLD Settings (NWE_HOLD = 0) | | | | | | |
| $SMC_{21}$ | NWE High to Data OUT, NBS0/A0 NBS1, A1, A2–A25, NCS change(1) | 2.1 | 1.5 | – | – | ns |

**Note:**
Hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “NCS_WR_HOLD length” or “NWE_HOLD length”

**Table 59-63. SMC Write NCS Controlled (WRITE_MODE = 0)**

| Symbol | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | Unit |
|---|---|---|---|---|---|---|
| | Parameter | Min | | Max | | |
| $SMC_{22}$ | Data Out Valid before NCS High | NCS_WR_PULSE × $t_{CPMCK}$ - 2.8 | NCS_WR_PULSE × $t_{CPMCK}$ - 3.9 | — | — | ns |
| $SMC_{23}$ | NCS Pulse Width | NCS_WR_PULSE × $t_{CPMCK}$ - 0.9 | NCS_WR_PULSE × $t_{CPMCK}$ - 0.2 | — | — | ns |
| $SMC_{24}$ | A0–A22 valid before NCS low | NCS_WR_SETUP × $t_{CPMCK}$ - 4.0 | NCS_WR_SETUP × $t_{CPMCK}$ - 4.6 | — | — | ns |
| $SMC_{25}$ | NWE low before NCS high | (NCS_WR_SETUP - NWE_SETUP + NCS pulse) × $t_{CPMCK}$ - 4.6 | (NCS_WR_SETUP - NWE_SETUP + NCS pulse) × $t_{CPMCK}$ - 4.6 | — | — | ns |
| $SMC_{26}$ | NCS High to Data Out, A0–A25, change | NCS_WR_HOLD × $t_{CPMCK}$ - 4.4 | NCS_WR_HOLD × $t_{CPMCK}$ - 3.4 | — | — | ns |
| $SMC_{27}$ | NCS High to NWE Inactive | (NCS_WR_HOLD - NWE_HOLD) × $t_{CPMCK}$ - 2.8 | (NCS_WR_HOLD - NWE_HOLD) × $t_{CPMCK}$ - 2.4 | — | — | ns |

Timings are given in the 3.3V domain, with VDDIO from 2.85V to 3.6V, maximum external capacitor = 50 pF.

Timings are given assuming a capacitance load on data, control and address pads.

In the tables that follow, $t_{CPMCK}$ is MCK period.