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®-M3
Core Size	32-Bit Single-Core
Speed	84MHz
Connectivity	CANbus, EBI/EMI, Ethernet, I²C, IrDA, LINbus, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	103
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	100K 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-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam3x8ea-au

10.4.4 Exceptions and interrupts

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the *Nested Vectored Interrupt Controller* (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See “Exception entry” on page 76 and “Exception return” on page 76 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller” on page 152 for more information.

10.4.5 Data types

The processor:

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- supports 64-bit data transfer instructions.
- manages all data memory accesses as little-endian. Instruction memory and *Private Peripheral Bus* (PPB) accesses are always little-endian. See “Memory regions, types and attributes” on page 62 for more information.

10.4.6 The Cortex Microcontroller Software Interface Standard

For a Cortex-M3 microcontroller system, the *Cortex Microcontroller Software Interface Standard* (CMSIS) defines:

- a common way to:
  - access peripheral registers
  - define exception vectors
- the names of:
  - the registers of the core peripherals
  - the core exception vectors
- a device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M3 processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system.

CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- “Power management programming hints” on page 81
- “Intrinsic functions” on page 85
- “The CMSIS mapping of the Cortex-M3 NVIC registers” on page 153
- “NVIC programming hints” on page 164.

10.5.4.2 DSB

The *Data Synchronization Barrier* (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. See “DSB” on page 144.

10.5.4.3 ISB

The *Instruction Synchronization Barrier* (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See “ISB” on page 145.

Use memory barrier instructions in, for example:

- MPU programming:
  - Use a DSB instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
  - Use an ISB instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an ISB instruction is not required.
- Vector table. If the program changes an entry in the vector table, and then enables the corresponding exception, use a DMB instruction between the operations. This ensures that if the exception is taken immediately after being enabled the processor uses the new exception vector.
- Self-modifying code. If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. This ensures subsequent instruction execution uses the updated program.
- Memory map switching. If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. This ensures subsequent instruction execution uses the updated memory map.
- Dynamic exception priority change. When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. This ensures the change takes effect on completion of the DSB instruction.
- Using a semaphore in multi-master system. If the system contains more than one bus master, for example, if another processor is present in the system, each processor must use a DMB instruction after any semaphore instructions, to ensure other bus masters see the memory transactions in the order in which they were executed.

Memory accesses to Strongly-ordered memory, such as the system control block, do not require the use of DMB instructions.

10.5.5 Bit-banding

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

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

- accesses to the 32MB SRAM alias region map to the 1MB SRAM bit-band region, as shown in Table 10-6
- accesses to the 32MB peripheral alias region map to the 1MB peripheral bit-band region, as shown in Table 10-7.

10.12.9 CLREX

Clear Exclusive.

10.12.9.1 Syntax

```
CLREX{cond}
```

where:

cond is an optional condition code, see “Conditional execution” on page 91.

10.12.9.2 Operation

Use CLREX to make the next STREX, STREXB, or STREXH instruction write 1 to its destination register and fail to perform the store. It is useful in exception handler code to force the failure of the store exclusive if the exception occurs between a load exclusive instruction and the matching store exclusive instruction in a synchronization operation.

See “Synchronization primitives” on page 69 for more information.

10.12.9.3 Condition flags

These instructions do not change the flags.

10.12.9.4 Examples

```
CLREX
```

14.6.9 RTC Interrupt Enable Register

Name: RTC_IER

Address: 0x400E1A80

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	CALEN	TIMEN	SECEN	ALREN	ACKEN

- **ACKEN: Acknowledge Update Interrupt Enable**

0: No effect.

1: The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0: No effect.

1: The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0: No effect.

1: The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0: No effect.

1: The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0: No effect.

1: The selected calendar event interrupt is enabled.

25.10.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 25-10. The write cycle starts with the address setting on the memory address bus.

25.10.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

1. NWE_SETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge.
2. NWE_PULSE: The NWE pulse length is the time between NWE falling edge and NWE rising edge.
3. NWE_HOLD: The NWE hold time is defined as the hold time of address and data after the NWE rising edge.

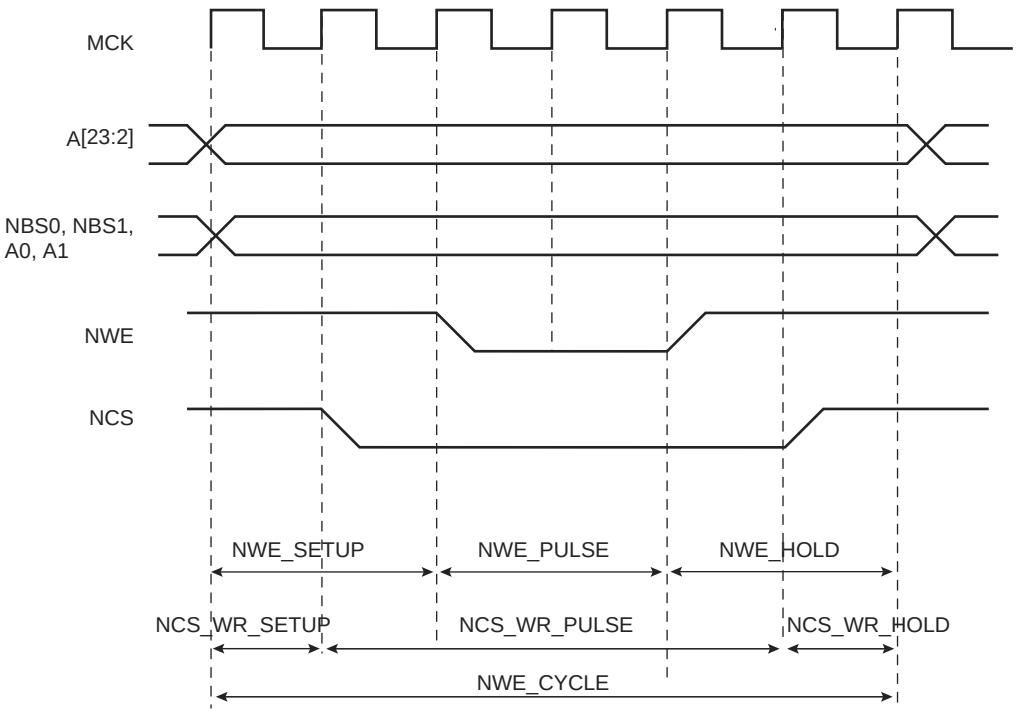
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

25.10.3.2 NCS Waveforms

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

1. NCS_WR_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCS_WR_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge.
3. NCS_WR_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 25-10. Write Cycle



28.15.14 PMC Interrupt Enable Register

Name: PMC_IER

Address: 0x400E0660

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU–	–	–	MCKRDY	–	LOCKA	MOSCXTS

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Enable**
- **LOCKA: PLLA Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **LOCKU: UTMI PLL Lock Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Enable**
- **MOSCRCS: Main On-Chip RC Status Interrupt Enable**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**

- **CFDEV: Clock Failure Detector Event**

0 = No clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

1 = At least one clock failure detection of the main on-chip RC oscillator clock has occurred since the last read of PMC_SR.

- **CFDS: Clock Failure Detector Status**

0 = A clock failure of the main on-chip RC oscillator clock is not detected.

1 = A clock failure of the main on-chip RC oscillator clock is detected.

- **FOS: Clock Failure Detector Fault Output Status**

0 = The fault output of the clock failure detector is inactive.

1 = The fault output of the clock failure detector is active.

Value	Name	Description
13		Reserved
14	2048K	2048K bytes
15		Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8K bytes
2	16K	16K bytes
3	32K	32K bytes
4		Reserved
5	64K	64K bytes
6		Reserved
7	128K	128K bytes
8		Reserved
9	256K	256K bytes
10	512K	512K bytes
11		Reserved
12	1024K	1024K bytes
13		Reserved
14	2048K	2048K bytes
15		Reserved

- **SRAMSIZ: Internal SRAM Size**

Value	Name	Description
0	48K	48K bytes
1	1K	1K bytes
2	2K	2K bytes
3	6K	6K bytes
4	24K	24K bytes
5	4K	4K bytes
6	80K	80K bytes
7	160K	160K bytes
8	8K	8K bytes
9	16K	16K bytes
10	32K	32K bytes
11	64K	64K bytes
12	128K	128K bytes

31.7.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Address: 0x400E0E24 (PIOA), 0x400E1024 (PIOB), 0x400E1224 (PIOC), 0x400E1424 (PIOD), 0x400E1624 (PIOE), 0x400E1824 (PIOF)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

32.8.2 SPI Mode Register

Name: SPI_MR

Address: 0x40008004 (0), 0x4000C004 (1)

Access: Read-write

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	WDRBT	MODFDIS	–	PCSDEC	PS	MSTR

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

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **WDRBT: Wait Data Read Before Transfer**

0 = No Effect. In master mode, a transfer can be initiated whatever the state of the Receive Data Register is.

1 = In Master Mode, a transfer can start only if the Receive Data Register is empty, i.e. does not contain any unread data. This mode prevents overrun error in reception.

36.6.11.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 2^{32}-1. Once 2^{32}-1 has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 36-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 36-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 36-7. WAVSEL = 00 without trigger

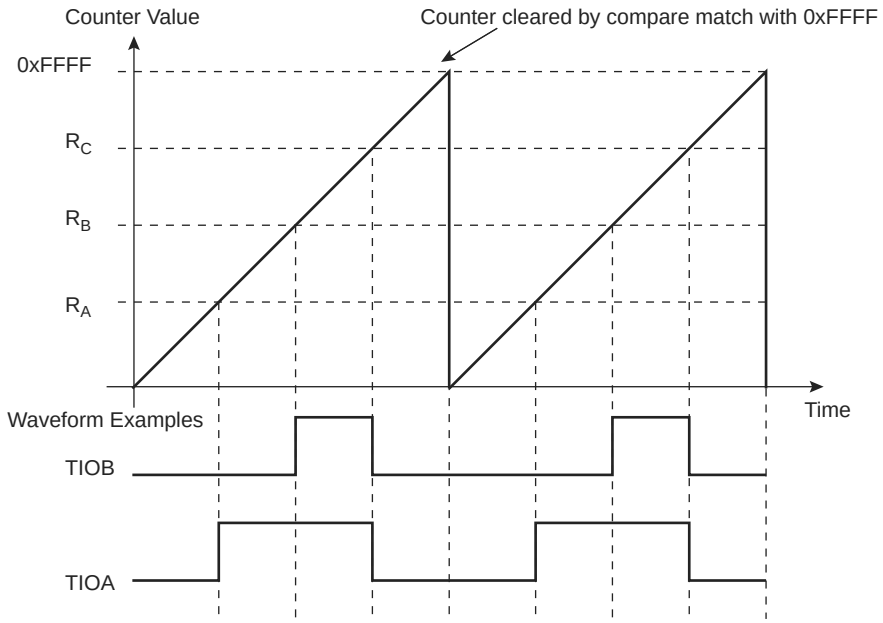
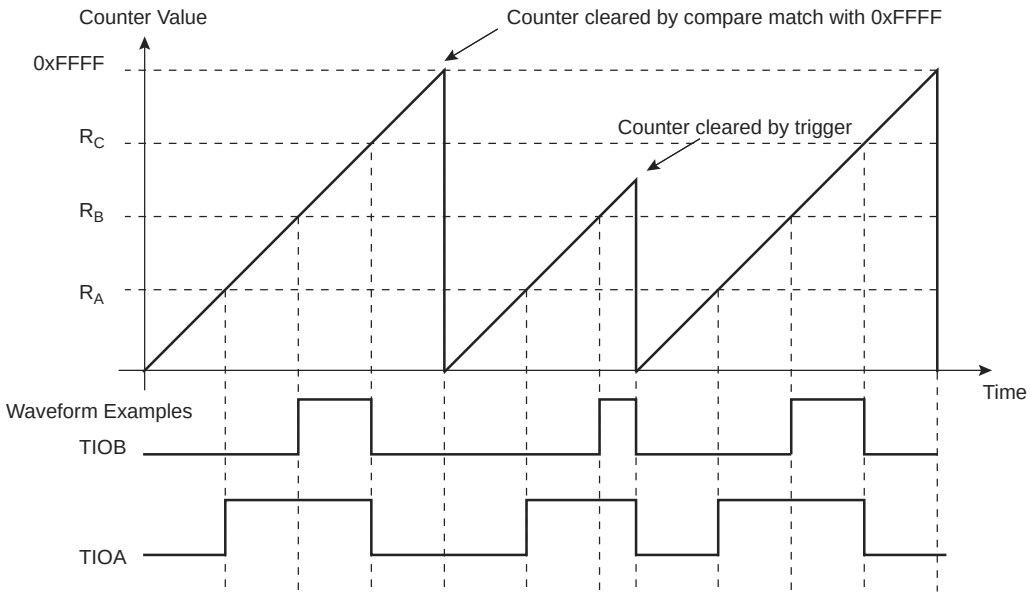


Figure 36-8. WAVSEL = 00 with Trigger



37.10 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, 8 bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

37.10.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8kB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are set to 0.

37.10.2 Executing an ATA Interrupt Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8kB of DATA with nIEN field set to zero to enable the command completion signal in the device.
2. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
3. Wait for Completion Signal Received Interrupt.

37.10.3 Aborting an ATA Command

If the host needs to abort an ATA command prior to the completion signal it must send a special command to avoid potential collision on the command line. The SPCMD field of the HSMCI_CMDR must be set to 3 to issue the CE-ATA completion Signal Disable Command.

37.10.4 CE-ATA Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, such as RW_MULTIPLE_REGISTER (CMD60).
- CRC is invalid for an MMC command or response.
- CRC16 is invalid for an MMC data packet.
- ATA Status register reflects an error by setting the ERR bit to one.
- The command completion signal does not arrive within a host specified time out period.

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event. The recommended error recovery procedure after a timeout is:

- Issue the command completion signal disable if nIEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received.
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39).

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another timeout, the next step is to issue

By using buffer registers PWM_OSSUPD and PWM_OSCUPD, the output selection of PWM outputs is done synchronously to the channel counter, at the beginning of the next PWM period.

By using registers PWM_OSS and PWM_OSC, the output selection of PWM outputs is done asynchronously to the channel counter, as soon as the register is written.

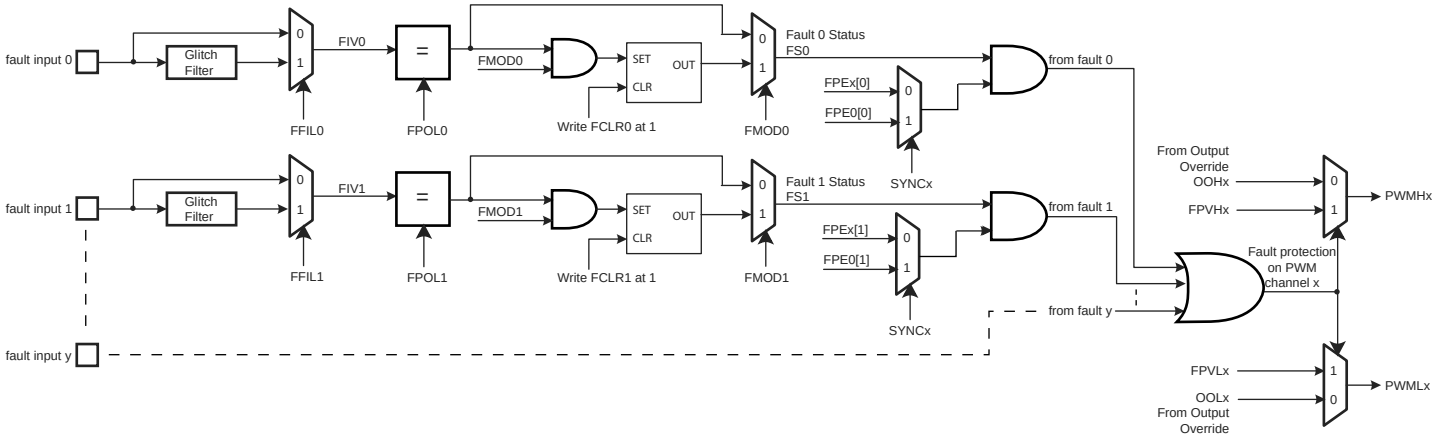
The value of the current output selection can be read in PWM_OS.

While overriding PWM outputs, the channel counters continue to run, only the PWM outputs are forced to user defined values.

38.6.2.6 Fault Protection

6 inputs provide fault protection which can force any of the PWM output pair to a programmable value. This mechanism has priority over output overriding.

Figure 38-9. Fault Protection



The polarity level of the fault inputs is configured by the FPOL field in the “PWM Fault Mode Register” (PWM_FMR). For fault inputs coming from internal peripherals such as ADC, Timer Counter, to name but a few, the polarity level must be FPOL = 1. For fault inputs coming from external GPIO pins the polarity level depends on the user's implementation.

The configuration of the Fault Activation Mode (FMOD bit in PWMC_FMR) depends on the peripheral generating the fault. If the corresponding peripheral does not have “Fault Clear” management, then the FMOD configuration to use must be FMOD = 1, to avoid spurious fault detection. Check the corresponding peripheral documentation for details on handling fault generation.

The fault inputs can be glitch filtered or not in function of the FFIL field in the PWM_FMR register. When the filter is activated, glitches on fault inputs with a width inferior to the PWM master clock (MCK) period are rejected.

A fault becomes active as soon as its corresponding fault input has a transition to the programmed polarity level. If the corresponding bit FMOD is set to 0 in the PWM_FMR register, the fault remains active as long as the fault input is at this polarity level. If the corresponding FMOD bit is set to 1, the fault remains active until the fault input is not at this polarity level anymore and until it is cleared by writing the corresponding bit FCLR in the “PWM Fault Clear Register” (PWM_FSCR). By reading the “PWM Fault Status Register” (PWM_FSR), the user can read the current level of the fault inputs by means of the field FIV, and can know which fault is currently active thanks to the FS field.

Each fault can be taken into account or not by the fault protection mechanism in each channel. To be taken into account in the channel x, the fault y must be enabled by the bit FPEx[y] in the “PWM Fault Protection Enable Registers” (PWM_FPE1/2). However the synchronous channels (see Section 38.6.2.7 “Synchronous Channels”) do not use their own fault enable bits, but those of the channel 0 (bits FPE0[y]).

38.7.13 PWM Interrupt Enable Register 2

Name: PWM_IER2

Address: 0x40094034

Access: Write-only

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

23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0

15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0

7	6	5	4	3	2	1	0
–	–	–	–	UNRE	TXBUFE	ENDTX	WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Enable**
- **ENDTX: PDC End of TX Buffer Interrupt Enable**
- **TXBUFE: PDC TX Buffer Empty Interrupt Enable**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Enable**
- **CMPMx: Comparison x Match Interrupt Enable**
- **CMPUx: Comparison x Update Interrupt Enable**

39.3 Block Diagram

The UOTGHS provides a hardware device to interface a USB link to a data flow stored in a dual-port RAM (DPRAM).

In normal operation (SPDCONF = 1), the UTMI transceiver requires the UTMI PLL (480 MHz). In case of Full or Low speed only, for a lower consumption (SPDCONF = 0), the UTMI transceiver only requires 48 MHz.

Figure 39-1. UOTGHS Block Diagram

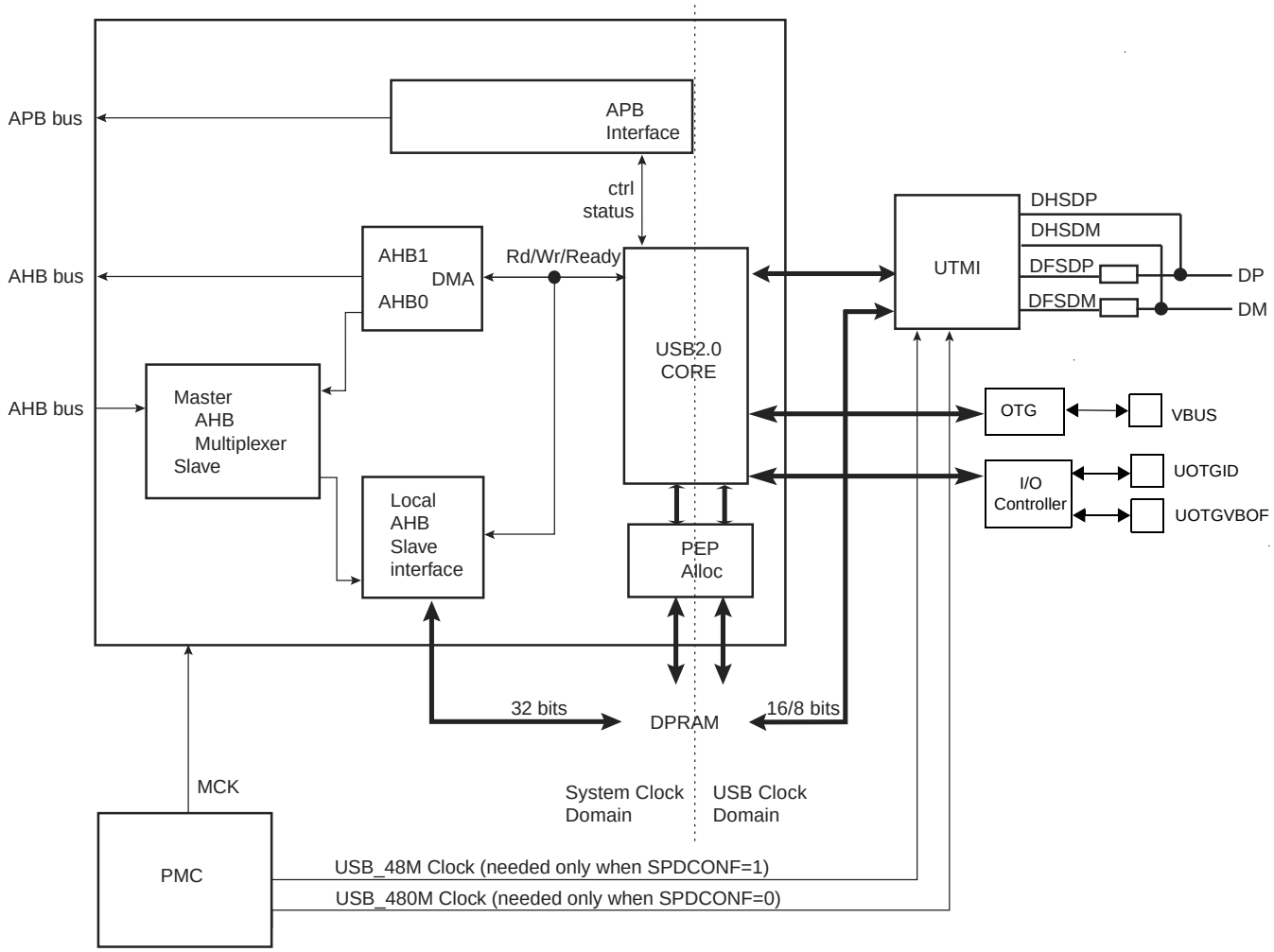
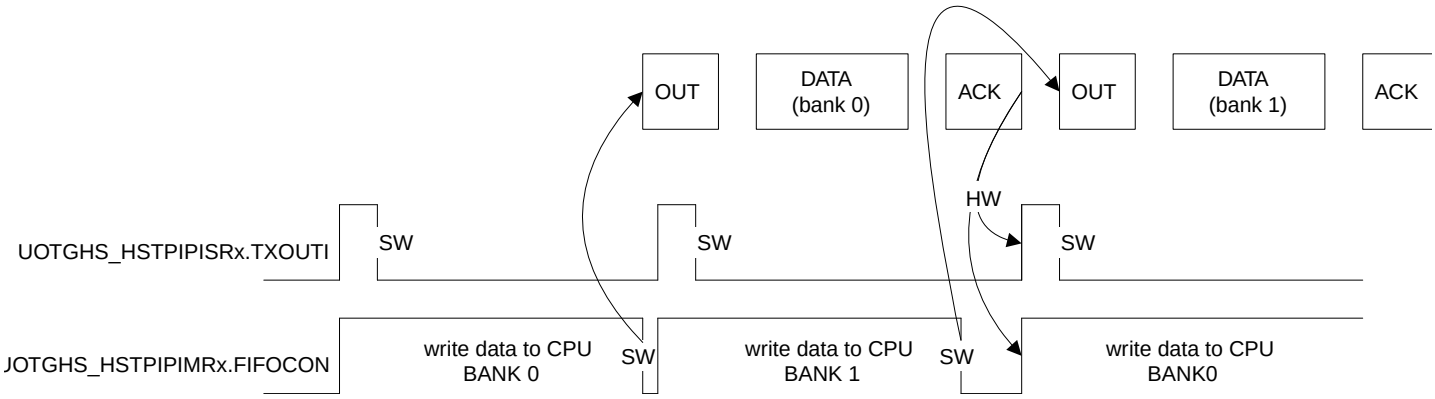


Figure 39-28. Example of an OUT Pipe with two Data Banks and a Bank Switching Delay



39.5.3.12 CRC Error

This error exists only for isochronous IN pipes. It sets the CRC Error Interrupt (UOTGHS_HSTPIPISRx.CRCERRI) bit, which triggers a PEP_x interrupt if then the CRC Error Interrupt Enable (UOTGHS_HSTPIPIMRx.CRCERRE) bit is one.

A CRC error can occur during IN stage if the UOTGHS detects a corrupted received packet. The IN packet is stored in the bank as if no CRC error had occurred (UOTGHS_HSTPIPISRx.RXINI is set).

39.5.3.13 Interrupts

See the structure of the USB host interrupt system on Figure 39-6 on page 1059.

There are two kinds of host interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

Global interrupts

The processing host global interrupts are:

- The Device Connection Interrupt (UOTGHS_HSTISR.DCONNI)
- The Device Disconnection Interrupt (UOTGHS_HSTISR.DDISCI)
- The USB Reset Sent Interrupt (UOTGHS_HSTISR.RSTI)
- The Downstream Resume Sent Interrupt (UOTGHS_HSTISR.RSMEDI)
- The Upstream Resume Received Interrupt (UOTGHS_HSTISR.RXRSMI)
- The Host Start of Frame Interrupt (UOTGHS_HSTISR.HSOFI)
- The Host Wake-Up Interrupt (UOTGHS_HSTISR.HWUPI)
- The Pipe x Interrupt (UOTGHS_HSTISR.PEP_x)
- The DMA Channel x Interrupt (UOTGHS_HSTISR.DMAxINT)

There is no exception host global interrupt.

Pipe interrupts

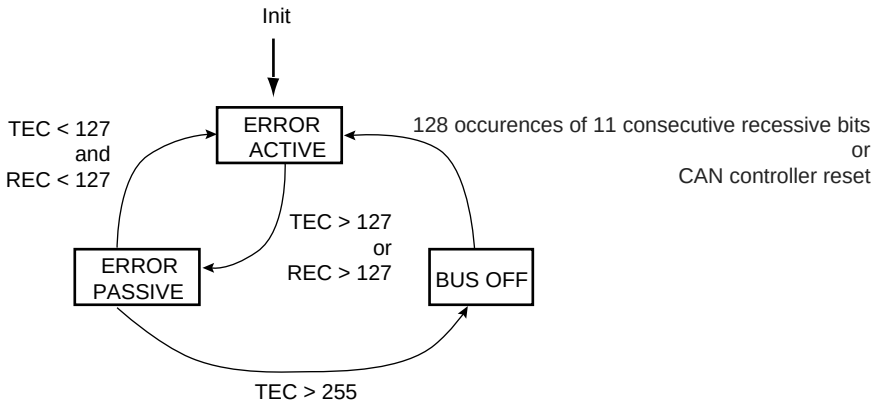
The processing host pipe interrupts are:

- The Received IN Data Interrupt (UOTGHS_HSTPIPISRx.RXINI)
- The Transmitted OUT Data Interrupt (UOTGHS_HSTPIPISRx.TXOUTI)
- The Transmitted SETUP Interrupt (UOTGHS_HSTPIPISRx.TXSTPI)
- The Short Packet Interrupt (UOTGHS_HSTPIPISRx.SHORTPACKETI)
- The Number of Busy Banks (UOTGHS_HSTPIPISRx.NBUSYBK) interrupt

Fault Confinement

To distinguish between temporary and permanent failures, every CAN controller has two error counters: REC (Receive Error Counter) and TEC (Transmit Error Counter). The two counters are incremented upon detected errors and are decremented upon correct transmissions or receptions, respectively. Depending on the counter values, the state of the node changes: the initial state of the CAN controller is Error Active, meaning that the controller can send Error Active flags. The controller changes to the Error Passive state if there is an accumulation of errors. If the CAN controller fails or if there is an extreme accumulation of errors, there is a state transition to Bus Off.

Figure 40-7. Line Error Mode



An error active unit takes part in bus communication and sends an active error frame when the CAN controller detects an error.

An error passive unit cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit waits before initiating further transmission.

A bus off unit is not allowed to have any influence on the bus.

For fault confinement, two errors counters (TEC and REC) are implemented. These counters are accessible via the CAN_ECR register. The state of the CAN controller is automatically updated according to these counter values. If the CAN controller is in Error Active state, then the ERRA bit is set in the CAN_SR register. The corresponding interrupt is pending while the interrupt is not masked in the CAN_IMR register. If the CAN controller is in Error Passive Mode, then the ERRP bit is set in the CAN_SR register and an interrupt remains pending while the ERRP bit is set in the CAN_IMR register. If the CAN is in Bus Off Mode, then the BOFF bit is set in the CAN_SR register. As for ERRP and ERRA, an interrupt is pending while the BOFF bit is set in the CAN_IMR register.

When one of the error counters values exceeds 96, an increased error rate is indicated to the controller through the WARN bit in CAN_SR register, but the node remains error active. The corresponding interrupt is pending while the interrupt is set in the CAN_IMR register.

Refer to the Bosch CAN specification v2.0 for details on fault confinement.

Error Interrupt Handler

WARN, BOFF, ERRA and ERRP (CAN_SR) represent the current status of the CAN bus and are not latched. They reflect the current TEC and REC (CAN_ECR) values as described in Section “Fault Confinement” on page 1196.

Based on that, if these bits are used as an interrupt, the user can enter into an interrupt and not see the corresponding status register if the TEC and REC counter have changed their state. When entering Bus Off Mode, the only way to exit from this state is 128 occurrences of 11 consecutive recessive bits or a CAN controller reset.

– Internal timer counter overflow interrupt: This interrupt is generated when the internal timer rolls over.
– Timestamp interrupt: This interrupt is generated after the reception or the transmission of a start of frame or an end of frame. The value of the internal counter is copied in the CAN_TIMESTP register.

All interrupts are cleared by clearing the interrupt source except for the internal timer counter overflow interrupt and the timestamp interrupt. These interrupts are cleared by reading the CAN_SR register.

40.8.3 CAN Controller Message Handling

40.8.3.1 Receive Handling

Two modes are available to configure a mailbox to receive messages. In *Receive Mode*, the first message received is stored in the mailbox data register. In *Receive with Overwrite Mode*, the last message received is stored in the mailbox.

Simple Receive Mailbox

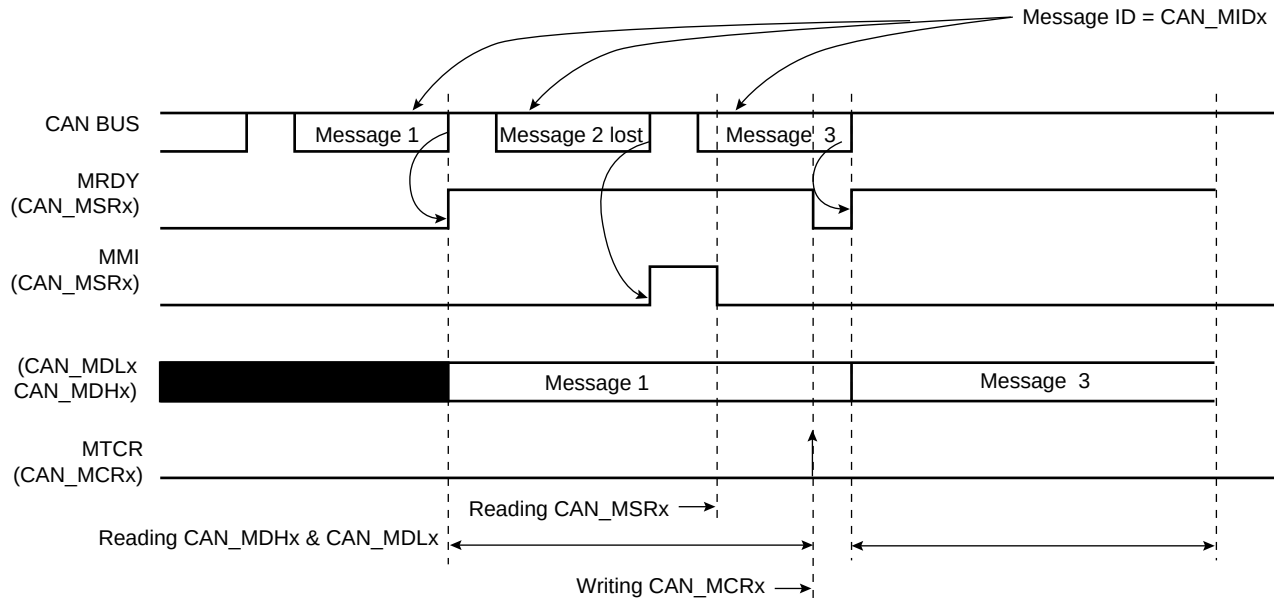
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first message is received. When the first message has been accepted by the mailbox, the MRDY flag is set. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked depending on the mailbox flag in the CAN_IMR global register.

Message data are stored in the mailbox data register until the software application notifies that data processing has ended. This is done by asking for a new transfer command, setting the MTCR flag in the CAN_MCRx register. This automatically clears the MRDY signal.

The MMI flag in the CAN_MSRx register notifies the software that a message has been lost by the mailbox. This flag is set when messages are received while MRDY is set in the CAN_MSRx register. This flag is cleared by reading the CAN_MSRs register. A receive mailbox prevents from overwriting the first message by new ones while MRDY flag is set in the CAN_MSRx register. See Figure 40-11.

Figure 40-11. Receive Mailbox



Note: In the case of ARM architecture, CAN_MSRx, CAN_MDLx, CAN_MDHx can be read using an optimized ldm assembler instruction.

- **MMI: Mailbox Message Ignored**

0: No message has been ignored during the previous transfer
1: At least one message has been ignored during the previous transfer
Cleared by reading the CAN_MSRx register.

Mailbox Object Type	Description
Receive	Set when at least two messages intended for the mailbox have been sent. The first one is available in the mailbox data register. Others have been ignored. A mailbox with a lower priority may have accepted the message.
Receive with overwrite	Set when at least two messages intended for the mailbox have been sent. The last one is available in the mailbox data register. Previous ones have been lost.
Transmit	Reserved
Consumer	A remote frame has been sent by the mailbox but several messages have been received. The first one is available in the mailbox data register. Others have been ignored. Another mailbox with a lower priority may have accepted the message.
Producer	A remote frame has been received, but no data are available to be sent.

45.7.1 Static Performance Characteristics

Minimal code = 0

Maximal code = 4095

ADC resolution = 12 bits (4096)

Temperature range -40 to 100 °C

f_{ADC} = 2 MHz, ADC_ACR.IBCTL = 01

In Table 45-32 and Table 45-33, the LSB is relative to analog scale for 12-bit ADC:

- Single-ended (ex: ADVREF = 3.0V),
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV
  - Gain = 4, LSB = (750mV / 4096) = 183 µV
- Differential (ex: ADVREF = 3.0V),
  - Gain = 0.5, LSB = (6.0V / 4096) = 1465 µV
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (750mV / 4096) = 366 µV

In Table 45-34, the LSB is relative to analog scale for 10-bit ADC: LSB = 3.0V/1024.

Table 45-32. INL, DNL, 12-bit mode

Symbol	Parameter	Conditions(1)	Min	Typ	Max	Unit
INL	Integral Non-linearity	VDDIN 2.4V to < 3.0V Differential, DIFF = 1, OFF = x, GAIN = xx	-2.2	±1	+2.2	LSB
		VDDIN 2.4V to < 3.0V Single-ended, DIFF = 0, OFF = x, GAIN = xx	-1.8	±1	+1.8	LSB
		VDDIN 3.0V to < 3.6V Differential, DIFF = 1, OFF = x, GAIN = xx	-1.4	±1	+1.4	LSB
		VDDIN 3.3V to < 3.6V Single-ended, DIFF = 0, OFF = x, GAIN = xx	-1.2	±1	+1.2	LSB
DNL	Differential Non-linearity	VDDIN 2.4V to < 3.0V Differential, DIFF = 1, OFF = x, GAIN = xx	-0.8	±0.5	+0.8	LSB
		VDDIN 2.4V to < 3.0V Single-ended, DIFF = 0, OFF = x, GAIN = xx	-0.8	±0.5	+0.8	LSB
		VDDIN 3.0V to < 3.6V Differential, DIFF = 1, OFF = x, GAIN = xx	-0.7	±0.5	+0.7	LSB
		VDDIN 3.0V to < 3.6V Single-ended, DIFF = 0, OFF = x, GAIN = xx	-0.7	±0.5	+0.7	LSB

Note: 1. ‘x’ in conditions stands for digital 0 or 1

DRAFT