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	96MHz
Connectivity	EBI/EMI, I²C, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	57
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	52K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 4x10b, 4x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam3u4ca-au

12.5.1.3 Active

An exception that is being serviced by the processor but has not completed.

An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.

12.5.1.4 Active and pending

The exception is being serviced by the processor and there is a pending exception from the same source.

12.5.2 Exception types

The exception types are:

12.5.2.1 Reset

Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.

12.5.2.2 Non Maskable Interrupt (NMI)

A non maskable interrupt (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2.

NMIs cannot be:

- Masked or prevented from activation by any other exception.
- Preempted by any exception other than Reset.

12.5.2.3 Hard fault

A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.

12.5.2.4 Memory management fault

A memory management fault is an exception that occurs because of a memory protection related fault. The MPU or the fixed memory protection constraints determines this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to *Execute Never* (XN) memory regions, even if the MPU is disabled.

12.5.2.5 Bus fault

A bus fault is an exception that occurs because of a memory related fault for an instruction or data memory transaction. This might be from an error detected on a bus in the memory system.

12.5.2.6 Usage fault

A usage fault is an exception that occurs because of a fault related to instruction execution. This includes:

- an undefined instruction
- an illegal unaligned access
- invalid state on instruction execution
- an error on exception return.

The following can cause a usage fault when the core is configured to report them:

- an unaligned address on word and halfword memory access
- division by zero.

12.13.1 MUL, MLA, and MLS

Multiply, Multiply with Accumulate, and Multiply with Subtract, using 32-bit operands, and producing a 32-bit result.

12.13.1.1 Syntax

```
MUL{S}{cond} {Rd,} Rn, Rm ; Multiply
MLA{cond} Rd, Rn, Rm, Ra  ; Multiply with accumulate
MLS{cond} Rd, Rn, Rm, Ra  ; Multiply with subtract
```

where:

cond is an optional condition code, see “Conditional execution” on page 87.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see “Conditional execution” on page 87.

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn, Rm are registers holding the values to be multiplied.

Ra is a register holding the value to be added or subtracted from.

12.13.1.2 Operation

The MUL instruction multiplies the values from *Rn* and *Rm*, and places the least significant 32 bits of the result in *Rd*.

The MLA instruction multiplies the values from *Rn* and *Rm*, adds the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The MLS instruction multiplies the values from *Rn* and *Rm*, subtracts the product from the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The results of these instructions do not depend on whether the operands are signed or unsigned.

12.13.1.3 Restrictions

In these instructions, do not use SP and do not use PC.

If you use the S suffix with the MUL instruction:

- *Rd*, *Rn*, and *Rm* must all be in the range R0 to R7
- *Rd* must be the same as *Rm*
- you must not use the *cond* suffix.

12.13.1.4 Condition flags

If S is specified, the MUL instruction:

- updates the N and Z flags according to the result
- does not affect the C and V flags.

12.13.1.5 Examples

```
MUL     R10, R2, R5     ; Multiply, R10 = R2 x R5
MLA     R10, R2, R1, R5 ; Multiply with accumulate, R10 = (R2 x R1) + R5
MULS    R0, R2, R2      ; Multiply with flag update, R0 = R2 x R2
MULLT   R2, R3, R2      ; Conditionally multiply, R2 = R3 x R2
MLS     R4, R5, R6, R7  ; Multiply with subtract, R4 = R7 - (R5 x R6)
```

13.5.5 DWT (Data Watchpoint and Trace)

The DWT contains four comparators which can be configured to generate the following:

- PC sampling packets at set intervals
- PC or Data watchpoint packets
- Watchpoint event to halt core

The DWT contains counters for the items that follow:

- Clock cycle (CYCCNT)
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep Cycles
- CPI (all instruction cycles except for the first cycle)
- Interrupt overhead

13.5.6 ITM (Instrumentation Trace Macrocell)

The ITM is an application driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated by three different sources with several priority levels:

- **Software trace**: Software can write directly to ITM stimulus registers. This can be done thanks to the “printf” function. For more information, refer to Section 13.5.6.1 “How to Configure the ITM”.
- **Hardware trace**: The ITM emits packets generated by the DWT.
- **Time stamping**: Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp.

13.5.6.1 How to Configure the ITM

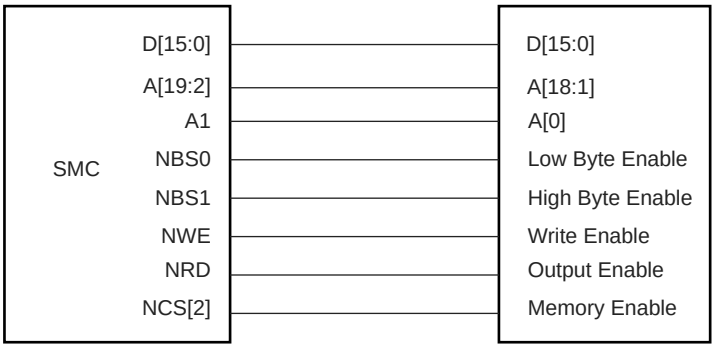
The following example describes how to output trace data in asynchronous trace mode.

- Configure the TPIU for asynchronous trace mode (refer to Section 13.5.6.3 “5.4.3. How to Configure the TPIU”)
- Enable the write accesses into the ITM registers by writing “0xC5ACCE55” into the Lock Access Register (Address: 0xE0000FB0)
- Write 0x00010015 into the Trace Control Register:
  - Enable ITM
  - Enable Synchronization packets
  - Enable SWO behavior
  - Fix the ATB ID to 1
- Write 0x1 into the Trace Enable Register:
  - Enable the Stimulus port 0
- Write 0x1 into the Trace Privilege Register:
  - Stimulus port 0 only accessed in privileged mode (Clearing a bit in this register will result in the corresponding stimulus port being accessible in user mode.)
- Write into the Stimulus port 0 register: TPIU (Trace Port Interface Unit)

  The TPIU acts as a bridge between the on-chip trace data and the Instruction Trace Macrocell (ITM).

  The TPIU formats and transmits trace data off-chip at frequencies asynchronous to the core.

Figure 24-5. Memory Connection for a 16-bit Data Bus



24.12.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If only the timing registers are modified (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, the user must validate the modification by writing the SMC_MODE register, even if no change was made on the mode parameters.

30.7.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See “Start” on page 560.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See “Frame Sync” on page 561.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SSC_SR and the receiver shift register is transferred in the RHR register.

Figure 30-9. Receiver Block Diagram

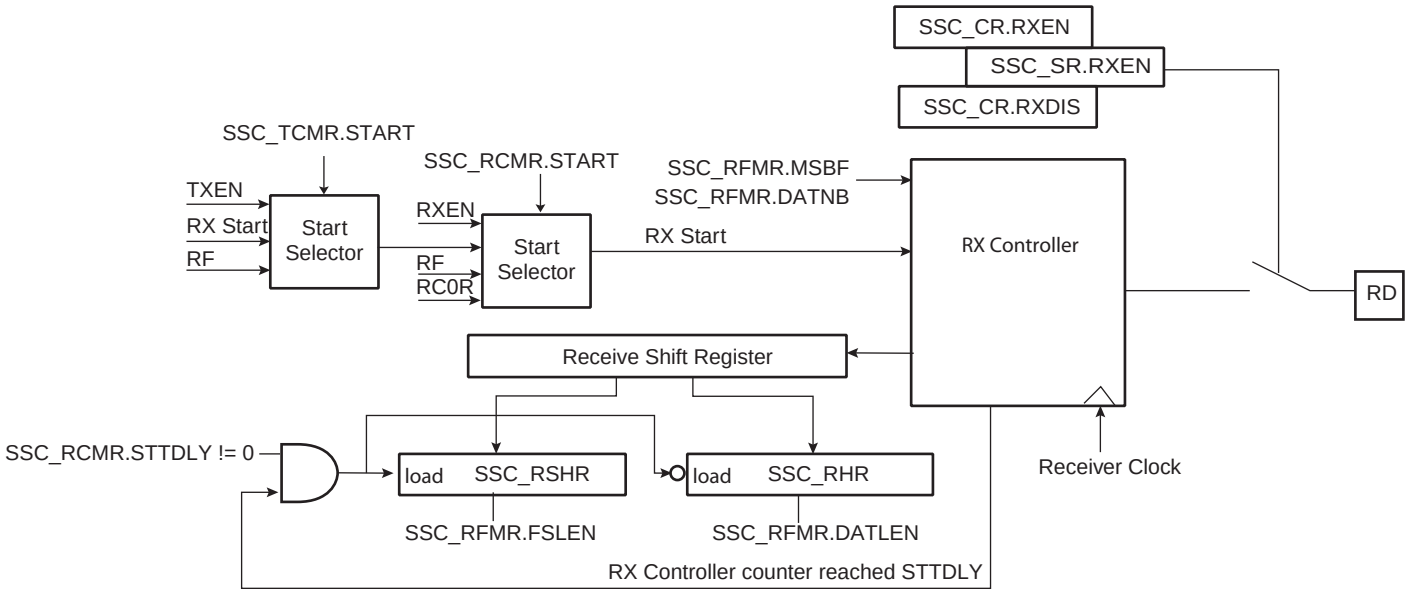
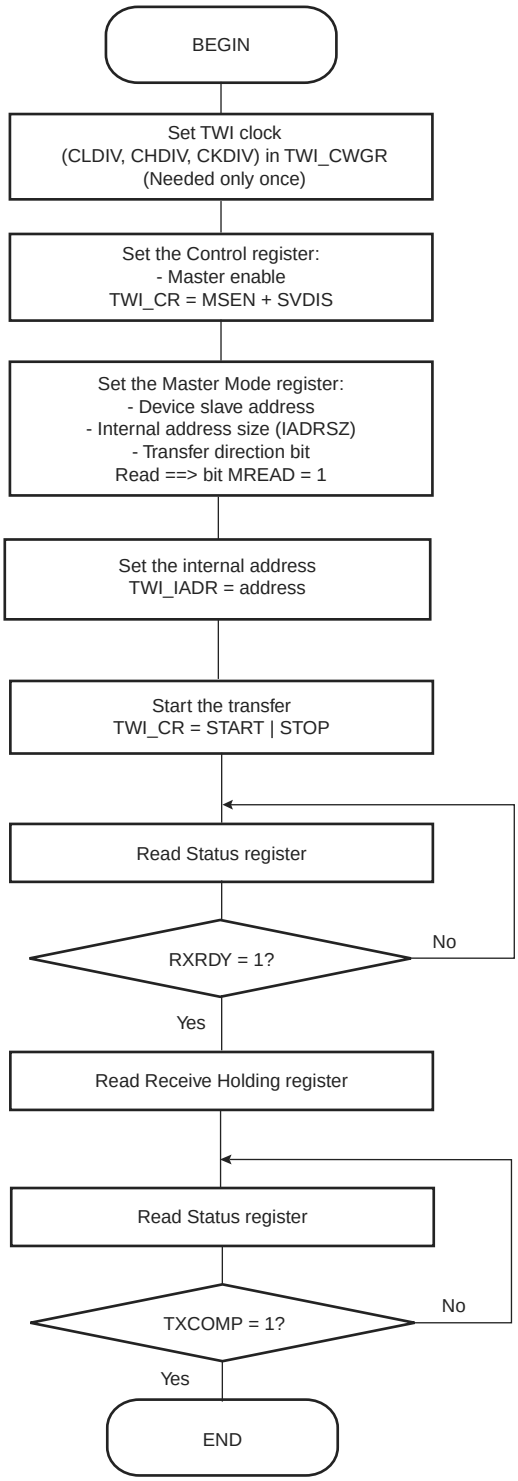


Figure 32-19. TWI Read Operation with Single Data Byte and Internal Address



32.11.3 TWI Slave Mode Register

Name: TWI_SMR

Address: 0x40084008 (0), 0x40088008 (1)

Access: Read-write

Reset: 0x00000000

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

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

34.7.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains to 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises. Then, the user can either:

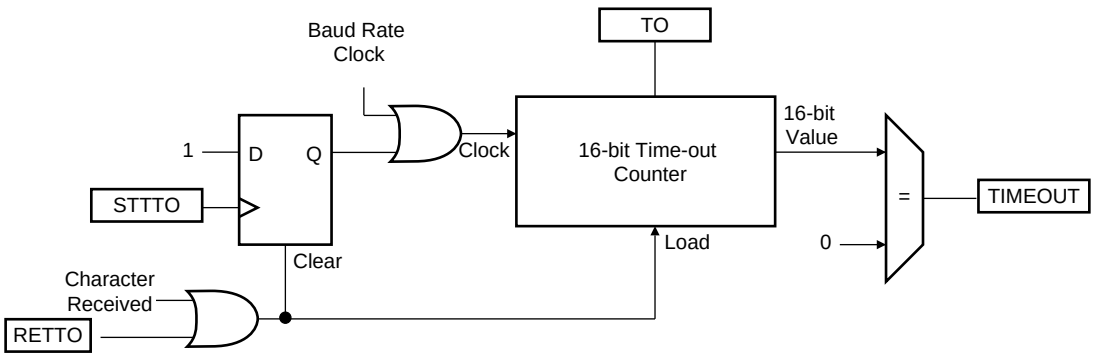
- Stop the counter clock until a new character is received. This is performed by writing the Control Register (US_CR) with the STTTO (Start Time-out) bit to 1. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.
- Obtain an interrupt while no character is received. This is performed by writing US_CR with the RETTO (Reload and Start Time-out) bit to 1. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 34-24 shows the block diagram of the Receiver Time-out feature.

Figure 34-24. Receiver Time-out Block Diagram



If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in Figure 34-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

Figure 34-31. T = 0 Protocol without Parity Error

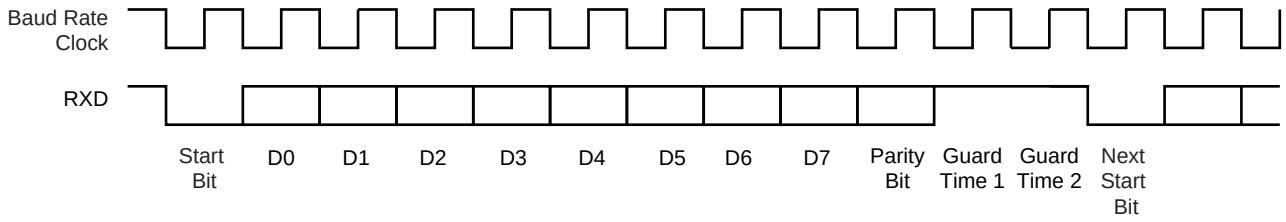
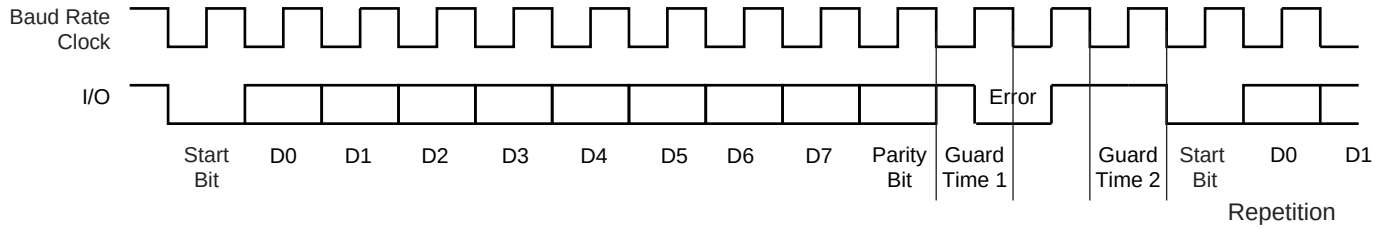


Figure 34-32. T = 0 Protocol with Parity Error



Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is to 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred and the RXRDY bit does rise.

Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (US_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (US_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US_CSR can be cleared by writing the Control Register with the RSIT bit to 1.

34.8.9 USART Baud Rate Generator Register

Name: US_BRGR

Address: 0x40090020 (0), 0x40094020 (1), 0x40098020 (2), 0x4009C020 (3)

Access: Read-write

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

23	22	21	20	19	18	17	16
–	–	–	–	–	FP		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 753.

- **CD: Clock Divider**

	USART_MODE ≠ ISO7816			
	SYNC = 0		**SYNC = 1 or USART_MODE = SPI (Master or Slave)**	**USART_MODE = ISO7816**
CD	**OVER = 0**	**OVER = 1**		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/(16*CD)	Baud Rate = Selected Clock/(8*CD)	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/(FI_DI_RATIO*CD)

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

35.6.14.5 Speed Measurement

When SPEEDEN is set in the TC_BMR, the speed measure is enabled on channel 0.

A time base must be defined on channel 2 by writing the TC_RC2 period register. Channel 2 must be configured in Waveform mode (WAVE bit set) in TC_CMR2. The WAVSEL field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. Field ACPC must be defined at 0x11 to toggle TIOA output.

This time base is automatically fed back to TIOA of channel 0 when QDEN and SPEEDEN are set.

Channel 0 must be configured in Capture mode (WAVE = 0 in TC_CMR0). The ABETRG bit of TC_CMR0 must be configured at 1 to select TIOA as a trigger for this channel.

EDGTRG must be set to 0x01, to clear the counter on a rising edge of the TIOA signal and field LDRA must be set accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01). As a consequence, at the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring bits CLKEN and SWTRG in the TC_CCR.

The speed can be read on field RA in TC_RA0.

Channel 1 can still be used to count the number of revolutions of the motor.

35.7.11 TC Interrupt Mask Register

Name: TC_IMRx [x=0..2]

Address: 0x4008002C (0)[0], 0x4008006C (0)[1], 0x400800AC (0)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0: The Counter Overflow Interrupt is disabled.

1: The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0: The Load Overrun Interrupt is disabled.

1: The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0: The RA Compare Interrupt is disabled.

1: The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0: The RB Compare Interrupt is disabled.

1: The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0: The RC Compare Interrupt is disabled.

1: The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0: The Load RA Interrupt is disabled.

1: The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0: The Load RB Interrupt is disabled.

1: The Load RB Interrupt is enabled.

36.14 High Speed Multimedia Card Interface (HSMCI) User Interface

Table 36-7. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	HSMCI_CR	Write	–
0x04	Mode Register	HSMCI_MR	Read-write	0x0
0x08	Data Timeout Register	HSMCI_DTOR	Read-write	0x0
0x0C	SD/SDIO Card Register	HSMCI_SDCR	Read-write	0x0
0x10	Argument Register	HSMCI_ARGR	Read-write	0x0
0x14	Command Register	HSMCI_CMDR	Write	–
0x18	Block Register	HSMCI_BLKR	Read-write	0x0
0x1C	Completion Signal Timeout Register	HSMCI_CSTOR	Read-write	0x0
0x20	Response Register(1)	HSMCI_RSPR	Read	0x0
0x24	Response Register(1)	HSMCI_RSPR	Read	0x0
0x28	Response Register(1)	HSMCI_RSPR	Read	0x0
0x2C	Response Register(1)	HSMCI_RSPR	Read	0x0
0x30	Receive Data Register	HSMCI_RDR	Read	0x0
0x34	Transmit Data Register	HSMCI_TDR	Write	–
0x38 - 0x3C	Reserved	–	–	–
0x40	Status Register	HSMCI_SR	Read	0xC0E5
0x44	Interrupt Enable Register	HSMCI_IER	Write	–
0x48	Interrupt Disable Register	HSMCI_IDR	Write	–
0x4C	Interrupt Mask Register	HSMCI_IMR	Read	0x0
0x50	DMA Configuration Register	HSMCI_DMA	Read-write	0x00
0x54	Configuration Register	HSMCI_CFG	Read-write	0x00
0x58-0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	HSMCI_WPMR	Read-write	–
0xE8	Write Protection Status Register	HSMCI_WPSR	Read-only	–
0xEC - 0xFC	Reserved	–	–	–
0x100-0x1FC	Reserved	–	–	–
0x200	FIFO Memory Aperture0	HSMCI_FIFO0	Read-write	0x0
...	...	...	...	...
0x5FC	FIFO Memory Aperture255	HSMCI_FIFO255	Read-write	0x0

Note: 1. The response register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

37.7.28 PWM Event Line x Register

Name: PWM_ELMRx

Address: 0x4008C07C

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

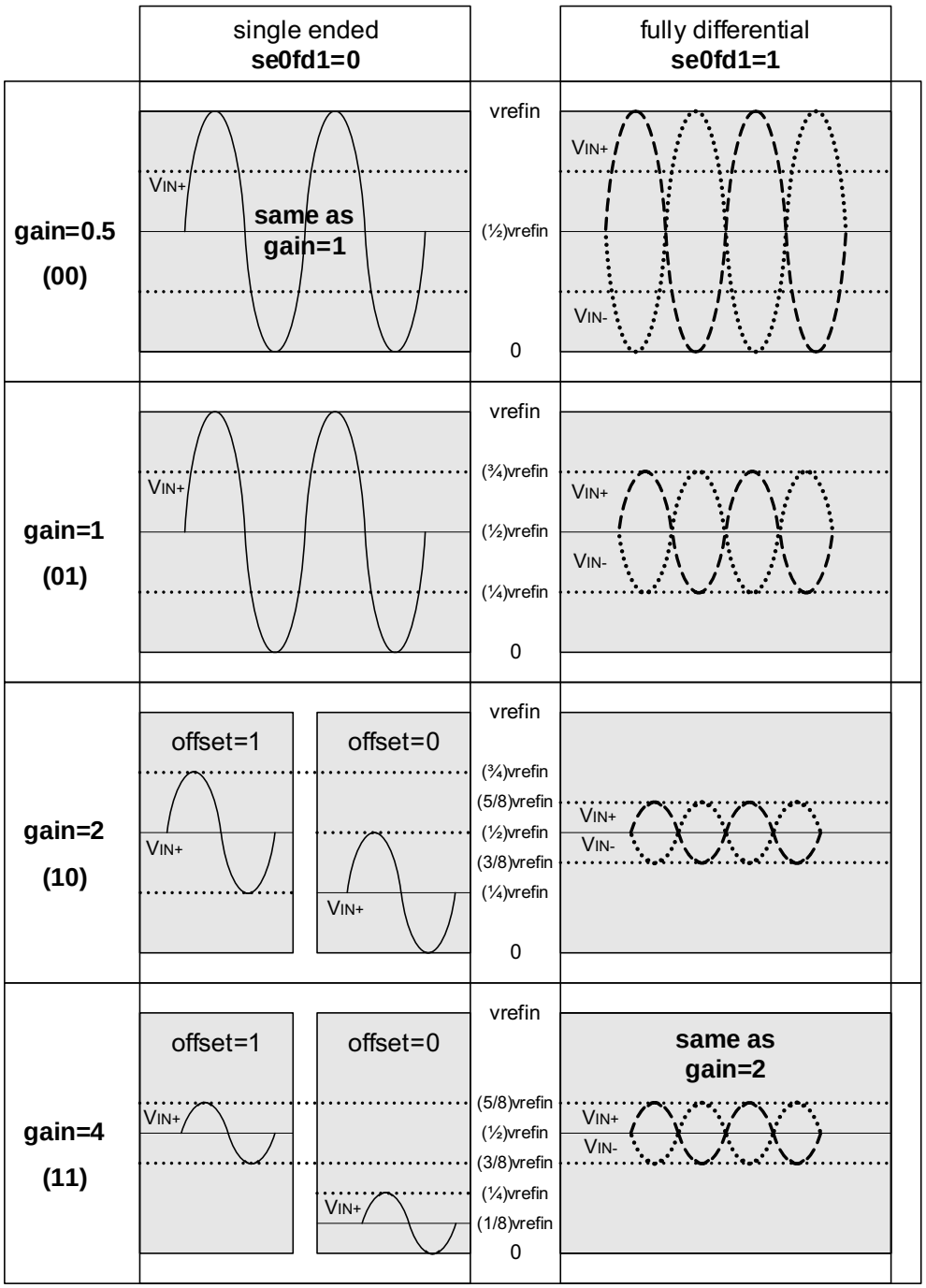
7	6	5	4	3	2	1	0
CSEL7	CSEL6	CSEL5	CSEL4	CSEL3	CSEL2	CSEL1	CSEL0

- **CSELy: Comparison y Selection**

0 = A pulse is not generated on the event line x when the comparison y matches.

1 = A pulse is generated on the event line x when the comparison y match.

Figure 40-2. Analog Full Scale Ranges in Single Ended/Differential Applications Versus Gain and Offset



40.5.6 Power Consumption Adjustment

The power consumption of the ADC12B can be adjusted through a 2-bit bias control (IBCTL bit in ADC12B_ACR register) providing possibilities for smart optimization of power and effective resolution relative to the application speed request.

Please refer to the Electrical Characteristics of the product datasheet for further details.

40.6.11 ADC12B Interrupt Disable Register

Name: ADC12B_IDR

Address: 0x400A8028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**
- **OVREx: Overrun Error Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

41. Analog-to-Digital Converter (ADC)

41.1 Description

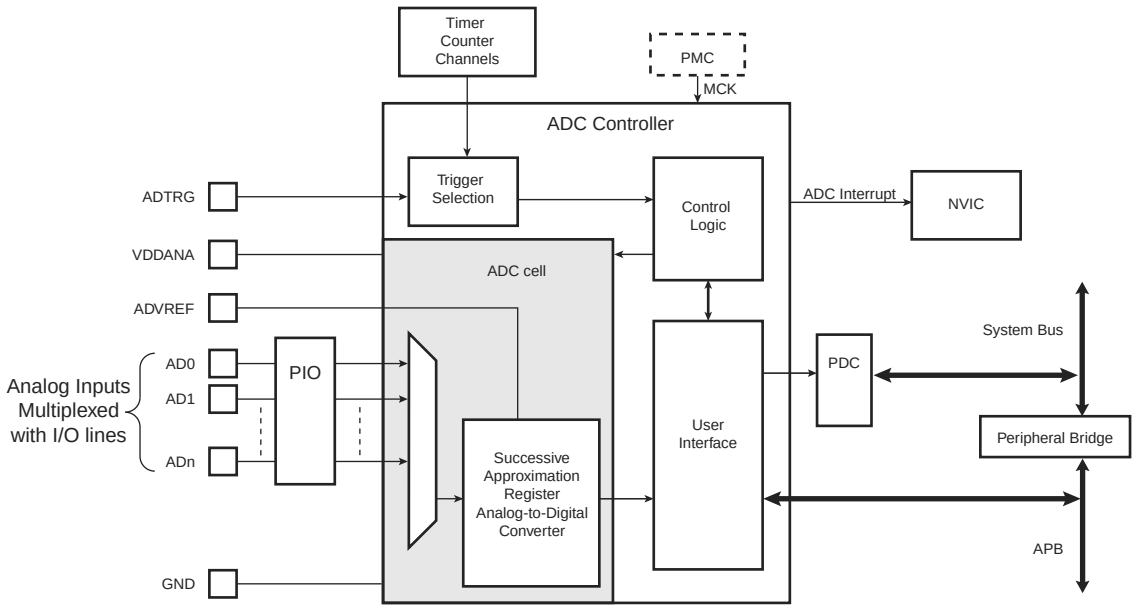
The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an 8-to-1 analog multiplexer, making possible the analog-to-digital conversions of 8 analog lines. The conversions extend from 0V to ADVREF. The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) or PWM Event lines are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

41.2 Block Diagram

Figure 41-1. Analog-to-Digital Converter Block Diagram



41.3 Signal Description

Table 41-1. ADC Pin Description

Pin Name	Description
AD0 - AD7	Analog input channels
ADTRG	External trigger

Table 42-34. Gain Error, Offset Error, 12-bit Mode, VDDANA Supply Voltage Conditions(1)

Symbol	Parameter	Conditions(2)	Min	Typ	Max	Unit
E_G	Gain error	VDDANA 2.4V to < 3.6V Differential, DIFF = 1, OFF = x, GAIN = xx	-1.56 -64	-0.56 -23	+0.29 +12	% LSB
		VDDANA 2.4V to < 3.6V Single ended, DIFF = 0, OFF = x, GAIN = xx	-1.56 -64	-0.56 -23	+0.78 +32	% LSB
E_O	Offset error	VDDANA 2.4V to < 3.6V Differential, DIFF = 1, OFF = x, GAIN = xx	-30		+64	LSB
		VDDANA 2.4V to < 3.6V Single ended, DIFF = 0, OFF = x, GAIN = xx	-60		+80	LSB

Notes: 1. Offset and Gain errors are given without calibration. A software calibration can be done to reduce Gain and offset errors.

2. 'x' in conditions represents digital 0 or 1

Figure 42-14. Offset and Gain Definitions

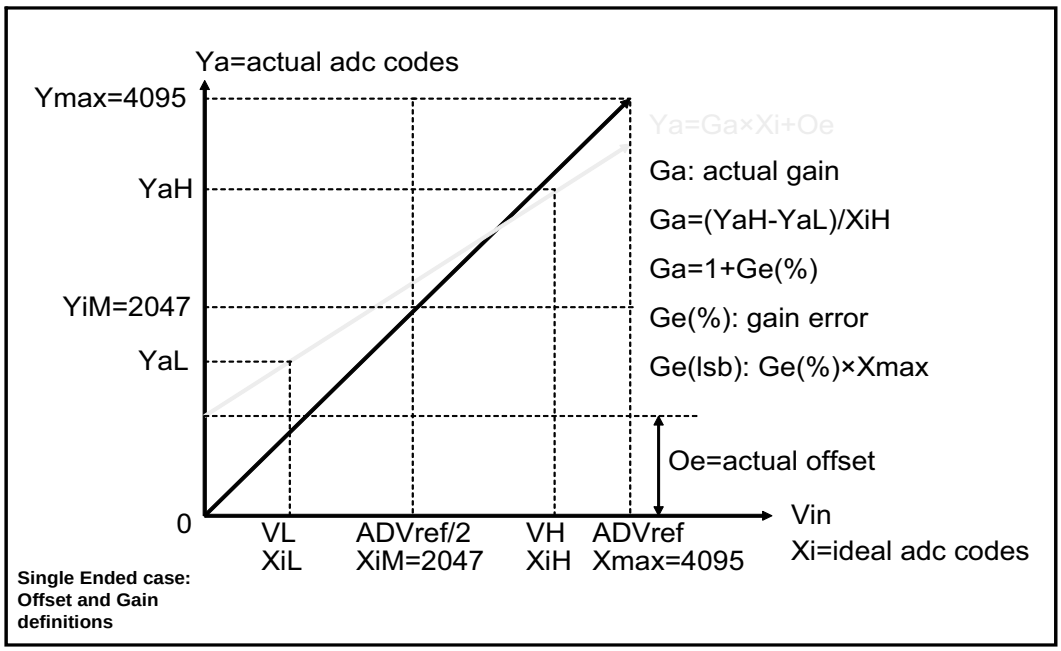


Table 42-35. Static Performance Characteristics - 10-bit mode (1)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Resolution			10		bit
INL	Integral Non-linearity		-1	±0.5	+1	LSB
DNL	Differential Non-linearity	No missing code	-1	±0.5	+1	LSB
E_O	Offset Error	All gain, Differential or Single-ended, no calibration	-8	+3	+20	LSB
E_G	Gain Error	All gain, Differential or Single-ended, no calibration	-16	-6	+3	LSB

Note: 1. Single-ended or differential mode, any gain values.

46.1.2 12-bit Analog-to-Digital Converter (ADC12B)

46.1.2.1 ADC12B: Current Consumption in Backup Mode on VDDANA

In Backup mode, the current consumption on VDDANA is around 1.0 mA instead of 0.1 µA

Problem Fix/Workaround

Four workarounds are possible:

1. Do not supply VDDANA and VDDIO in Backup mode using an external switch managed by SHDN pin.
2. Do not supply VDDANA in Backup mode using an external switch managed by the SHDN and set all PIOs with ADC inputs (PA22, PA30, PB3–PB8, PC15–PC18, PC28–C21) at low level (either externally or by software).
3. Do not supply VDDANA in Backup mode using an external switch managed by any PIO and set all PIOs with ADC inputs (PA22, PA30, PB3–PB8, PC15–PC18, PC28–C21) at low level (either externally or by software). Since the PIO state is preserved when in backup mode, any free PIO line can be used to switch off the external switch by driving the PIO line at low level (PIO is input, pull-up enabled after backup reset).
4. Use Wait mode instead of Backup mode.

46.1.2.2 ADC: Trigger Launches only One Conversion

A start command initiates a conversion sequence of one channel but not all activated channels as expected.

Problem Fix/Workaround

Send as many start commands as the number of activated channels, or use free run mode.

46.1.2.3 ADC12B: Saturation

When the ADC12B works in saturation (measurements below 0V or above AD12BVREF) the results may be erratic, the value deviation can be around 30 LSB to the expected data.

Problem Fix/Workaround

None.

46.1.2.4 ADC: Wrong First Conversions

The first conversions done by the ADC may be erroneous if the maximum gain (x4 in single ended or x2 in differential mode) is not used. The issue appears after the power-up or if a conversion has not occured for 1 minute.

Problem Fix/Workaround

Three workarounds are possible:

1. Perform 16 dummy conversions on one channel (whatever conditions used in term of setup of gain, single/differential, offset, and channel selected). The next conversions will be correct for any channels and any settings. Note that these dummy conversions need to be performed if no conversion has occured for 1 minute or for a new chip startup.
2. Perform a dummy conversion on a single ended channel on which an external voltage of ADVREF/2 (±10%) is applied. Use the following conditions for this conversion: gain at 4, offset set at 1. The next conversions will be correct for any channels and any settings. Note that this dummy conversion needs to be performed if no conversion has occured for 1 minute or for a new chip startup.
3. Perform a dummy conversion on a differential channel on which the two inputs are connected together and connected to any voltage (from 0 to ADVREF). Use the following conditions for this conversion: gain at 4, offset set at 1. The next conversions will be correct for any channels and any settings. Note that this dummy conversion needs to be performed if no conversion has occured for 1 minute or for a new chip startup.