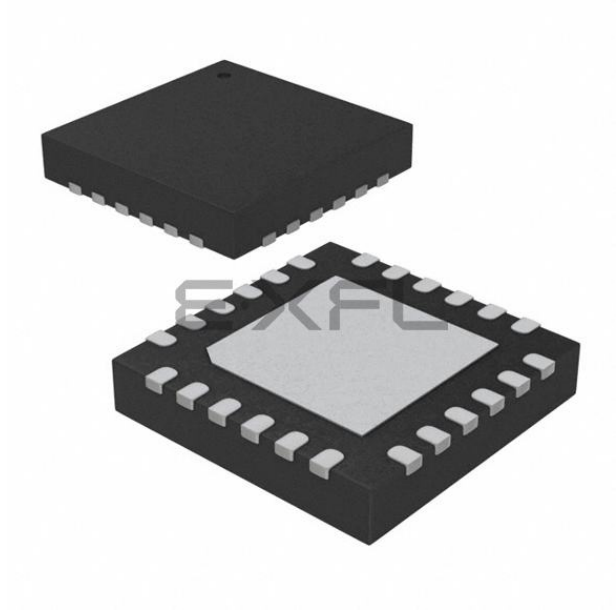


Welcome to E-XFL.COM

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	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	22
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	24-VFQFN Exposed Pad
Supplier Device Package	24-VQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny1607-mnr

10.5.1 Main Clock Control A

Name: MCLKCTRLA
Offset: 0x00
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	CLKOUT						CLKSEL[1:0]	
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – CLKOUT System Clock Out
When this bit is written to '1', the system clock is output to CLKOUT pin.

When the device is in a Sleep mode, there is no clock output unless a peripheral is using the system clock.

Bits 1:0 – CLKSEL[1:0] Clock Select
This bit field selects the source for the Main Clock (CLK_MAIN).

Value	Name	Description
0x0	OSC20M	16/20 MHz internal oscillator
0x1	OSCULP32K	32 KHz internal ultra low-power oscillator
0x2	Reserved	Reserved
0x3	EXTCLK	External clock

Table 13-2. Interrupt Priority Levels

Priority	Level	Source
Highest	Non-Maskable Interrupt (NMI)	Device dependent and statically assigned
...	High Priority (Level 1)	One vector is optionally user selectable as Level 1
Lowest	Normal Priority (Level 0)	The remaining interrupt vectors

13.3.2.5 Scheduling of Normal Priority Interrupts

13.3.2.5.1 Non-Maskable Interrupts (NMI)

An NMI will be executed regardless of the setting of the I bit in CPU.SREG, and it will never change the I bit. No other interrupt can interrupt an NMI handler. If more than one NMI is requested at the same time, priority is static according to the interrupt vector address, where the lowest address has the highest priority.

Which interrupts are non-maskable is device-dependent and not subject to configuration. Non-maskable interrupts must be enabled before they can be used. Refer to the Interrupt Vector Mapping of the device for available NMI lines.

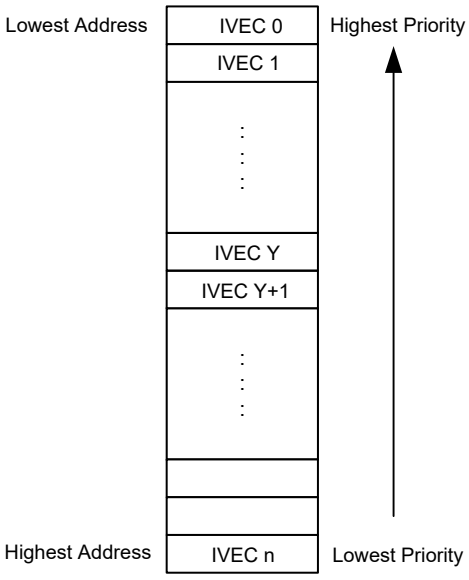
Related Links

7.2 Interrupt Vector Mapping

13.3.2.5.2 Static Scheduling

If several level 0 interrupt requests are pending at the same time, the one with the highest priority is scheduled for execution first. The CPUINT.LVL0PRI register makes it possible to change the default priority. The Reset value for CPUINT.LVL0PRI is zero, resulting in a default priority as shown in the following figure. As the figure shows, IVEC0 has the highest priority, and IVECn has the lowest priority.

Figure 13-3. Static Scheduling when CPUINT.LVL0PRI is Zero



The default priority can be changed by writing to the CPUINT.LVL0PRI register. The value written to the register will identify the vector number with the lowest priority. The next interrupt vector in IVEC will have the highest priority, see the following figure. In this figure, the value Y has been written to

14.3.2.4 Software Event

In a software event, the CPU will “strobe” an event channel by inverting the current value for one system clock cycle.

A software event is triggered on a channel by writing a '1' to the respective Strobe bit in the appropriate Channel Strobe register:

- Software events on asynchronous channel *l* are initiated by writing a '1' to the ASYNCSTROBE[*l*] bit in the Asynchronous Channel Strobe register (EVSYS.ASYNCSTROBE).
- Software events on synchronous channel *k* are initiated by writing a '1' to the SYNCSTROBE[*k*] bit in the Synchronous Channel Strobe register (EVSYS.SYNCSTROBE).

Software events are no different to those produced by event generator peripherals with respect to event users: when the bit is written to '1', an event will be generated on the respective channel, and received and processed by the event user.

14.3.3 Interrupts

Not applicable.

14.3.4 Sleep Mode Operation

When configured, the Event System will work in all sleep modes. One exception is software events that require a system clock.

14.3.5 Debug Operation

This peripheral is unaffected by entering Debug mode.

Related Links

30. UPDI - Unified Program and Debug Interface

14.3.6 Synchronization

Asynchronous events are synchronized and handled by the compatible event users. Event user peripherals not compatible with asynchronous events can only be configured to listen to synchronous event channels.

14.3.7 Configuration Change Protection

Not applicable.

19. WDT - Watchdog Timer

19.1 Features

- Issues a System Reset if the Watchdog Timer is not Cleared Before its Time-out Period
- Operating Asynchronously from System Clock Using an Independent Oscillator
- Using the 1 KHz Output of the 32 KHz Ultra Low-Power Oscillator (OSCULP32K)
- 11 Selectable Time-out Periods, from 8 ms to 8s
- Two Operation modes:
  - Normal mode
  - Window mode
- Configuration Lock to Prevent Unwanted Changes
- Closed Period Timer Activation After First WDT Instruction for Easy Setup

19.2 Overview

The Watchdog Timer (WDT) is a system function for monitoring correct program operation. It allows the system to recover from situations such as runaway or deadlocked code, by issuing a Reset. When enabled, the WDT is a constantly running timer configured to a predefined time-out period. If the WDT is not reset within the time-out period, it will issue a system Reset. The WDT is reset by executing the `WDR` (Watchdog Timer Reset) instruction from software.

The WDT has two modes of operation; Normal mode and Window mode. The settings in the Control A register (WDT.CTRLA) determine the mode of operation.

A Window mode defines a time slot or "window" inside the time-out period during which the WDT must be reset. If the WDT is reset outside this window, either too early or too late, a system Reset will be issued. Compared to the Normal mode, the Window mode can catch situations where a code error causes constant `WDR` execution.

When enabled, the WDT will run in Active mode and all Sleep modes. It is asynchronous (i.e., running from a CPU independent clock source). For this reason, it will continue to operate and be able to issue a system Reset even if the main clock fails.

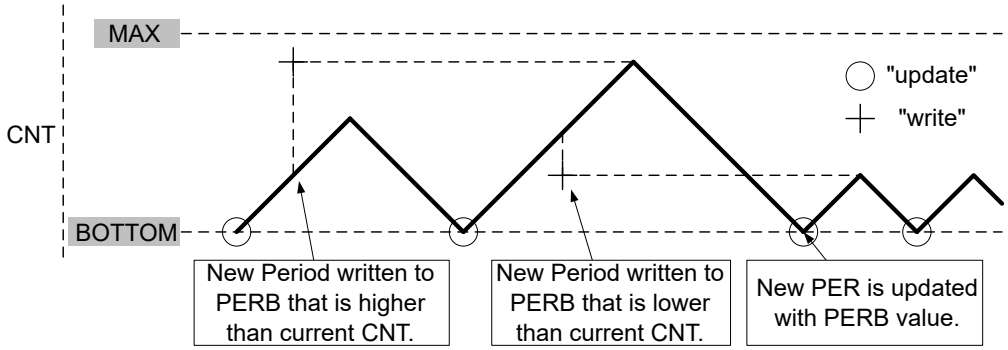
The CCP mechanism ensures that the WDT settings cannot be changed by accident. For increased safety, a configuration for locking the WDT settings is available.

Related Links

8.5.7 Configuration Change Protection (CCP)

With Buffering: When double buffering is used, the buffer can be written at any time and still maintain correct operation. The TCAn.PER is always updated on the UPDATE condition, as shown for dual-slope operation in the figure below. This prevents wrap-around and the generation of odd waveforms.

Figure 20-8. Changing the Period Using Buffering



20.3.3.4 Compare Channel

Each Compare Channel n continuously compares the counter value (TCAn.CNT) with the Compare n register (TCAn.CMPn). If TCAn.CNT equals TCAn.CMPn, the comparator n signals a match. The match will set the Compare Channel's Interrupt flag at the next timer clock cycle, and the optional interrupt is generated.

The Compare n Buffer register (TCAn.CMPnBUF) provides double buffer capability equivalent to that for the period buffer. The double buffering synchronizes the update of the TCAn.CMPn register with the buffer value to either the TOP or BOTTOM of the counting sequence, according to the UPDATE condition. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses for glitch-free output.

20.3.3.4.1 Waveform Generation

The compare channels can be used for waveform generation on the corresponding port pins. To make the waveform visible on the connected port pin, the following requirements must be met:

1. A Waveform Generation mode must be selected by writing the WGMODE bit field in TCAn.CTRLB.
2. The TCA is counting clock ticks, not events (CNTEI=0 in TCAn.EVCTRL).
3. The compare channels used must be enabled (CMPnEN=1 in TCAn.CTRLB). This will override the corresponding port pin output register. An alternative pin can be selected by writing to the respective TCA Waveform Output n bit (TCA0n) in the Control C register of the Port Multiplexer (PORTMUX.CTRLC).
4. The direction for the associated port pin n must be configured as an output (PORTx.DIR[n]=1).
5. Optional: Enable inverted waveform output for the associated port pin n (INVEN=1 in PORTx.PINn).

20.3.3.4.2 Frequency (FRQ) Waveform Generation

For frequency generation, the period time (T) is controlled by a TCAn.CMPn register instead of the Period register (TCAn.PER). The waveform generation output WG is toggled on each compare match between the TCAn.CNT and TCAn.CMPn registers.

20.5.15 Period Register - Normal Mode

Name: PER
Offset: 0x26
Reset: 0xFFFF
Property: -

TCAn.PER contains the 16-bit TOP value in the timer/counter.

The TCAn.PERL and TCAn.PERH register pair represents the 16-bit value, TCAn.PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01.

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

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

Bits 15:8 – PER[15:8] Periodic High Byte
These bits hold the MSB of the 16-bit period register.

Bits 7:0 – PER[7:0] Periodic Low Byte
These bits hold the LSB of the 16-bit period register.

23.3.2.1.2 External Clock

An External clock (XCK) is used in Synchronous Slave mode operation. The XCK clock input is sampled on the peripheral clock frequency and the maximum XCK clock frequency (f_{XCK}) is limited by the following:

$$f_{XCK} < \frac{f_{CLK\_PER}}{4}$$

For each high and low period, the XCK clock cycles must be sampled twice by the peripheral clock. If the XCK clock has jitter, or if the high/low period duty cycle is not 50/50, the maximum XCK clock speed must be reduced accordingly.

23.3.2.1.3 Double Speed Operation

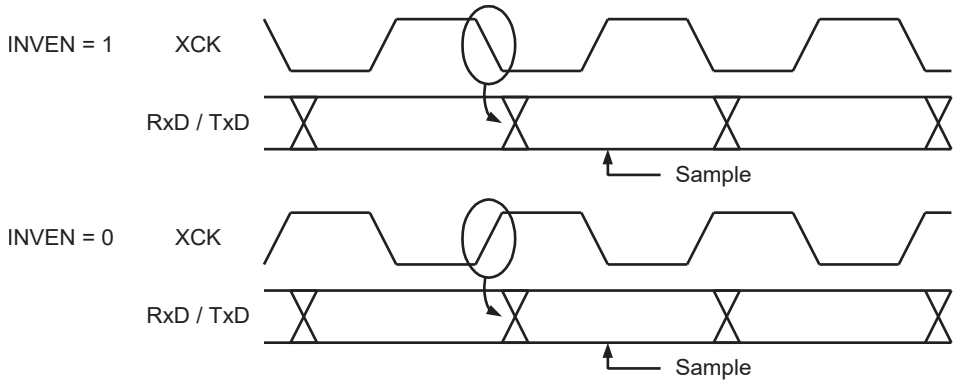
Double speed operation allows for higher baud rates under asynchronous operation with lower peripheral clock frequencies. This operation mode is enabled by writing the RXMODE bit in the Control B register (USARTn.CTRLB) to CLK2X.

When enabled, the baud rate for a given asynchronous baud rate setting shown in Table 23-2 will be doubled. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery. This requires a more accurate baud rate setting and peripheral clock. See 23.3.2.4.6 Asynchronous Data Reception for more details.

23.3.2.1.4 Synchronous Clock Operation

When Synchronous mode is used, the XCK pin controls whether the transmission clock is input (Slave mode) or output (Master mode). The corresponding port pin must be set to output for Master mode or to input for Slave mode (PORTx.DIR[n]). The normal port operation of the XCK pin will be overridden. The dependency between the clock edges and data sampling or data change is the same. Data input (on RxD) is sampled at the XCK clock edge which is opposite the edge where data output (TxD) is changed.

Figure 23-3. Synchronous Mode XCK Timing

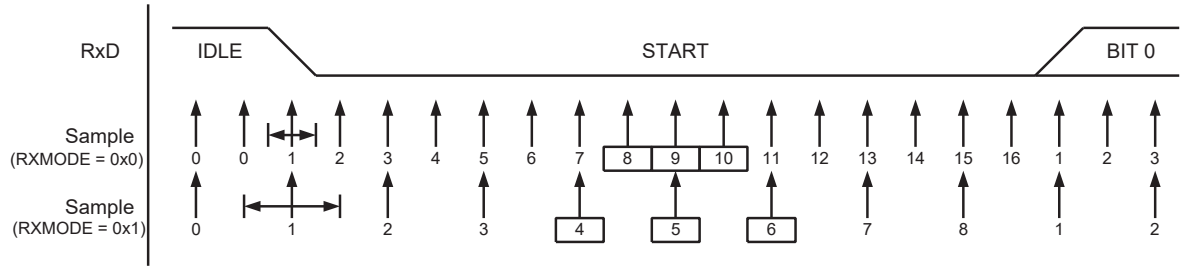


The I/O pin can be inverted by writing a '1' to the Inverted I/O Enable bit (INVEN) in the Pin n Control register of the port peripheral (PORTx.PINnCTRL). Using the inverted I/O setting for the corresponding XCK port pin, the XCK clock edges used for data sampling and data change can be selected. If inverted I/O is disabled (INVEN=0), data will be changed at the rising XCK clock edge and sampled at the falling XCK clock edge. If inverted I/O is enabled (INVEN=1), data will be changed at the falling XCK clock edge and sampled at the rising XCK clock edge.

23.3.2.1.5 Master SPI Mode Clock Generation

For Master SPI mode operation, only internal clock generation is supported. This is identical to the USART Synchronous Master mode, and the baud rate or BAUD setting is calculated using the same equations (see Table 23-2).

Figure 23-7. Start Bit Sampling



When the clock recovery logic detects a high-to-low (i.e., idle-to-start) transition on the RxD line, the Start bit detection sequence is initiated. Sample 1 denotes the first zero-sample, as shown in the figure. The clock recovery logic then uses three subsequent samples (samples 8, 9, and 10 in Normal mode, samples 4, 5, and 6 in Double-Speed mode) to decide if a valid Start bit is received:

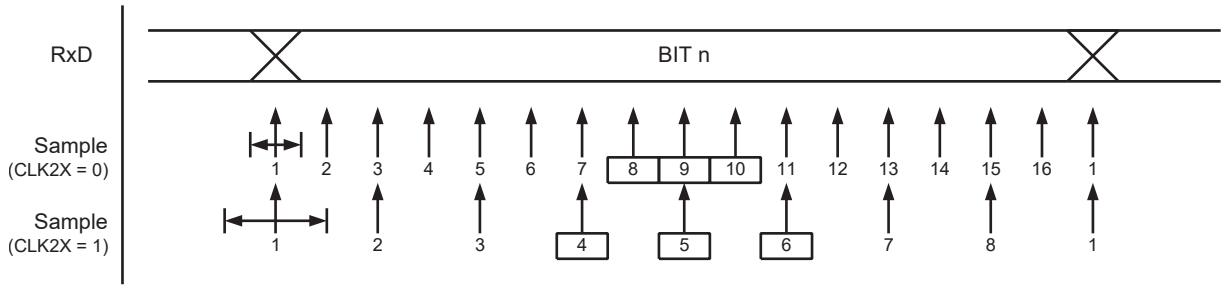
- If two or three samples have a low level, the Start bit is accepted. The clock recovery unit is synchronized, and the data recovery can begin.
- If two or three samples have a high level, the Start bit is rejected as a noise spike, and the receiver looks for the next high-to-low transition.

The process is repeated for each Start bit.

Asynchronous Data Recovery

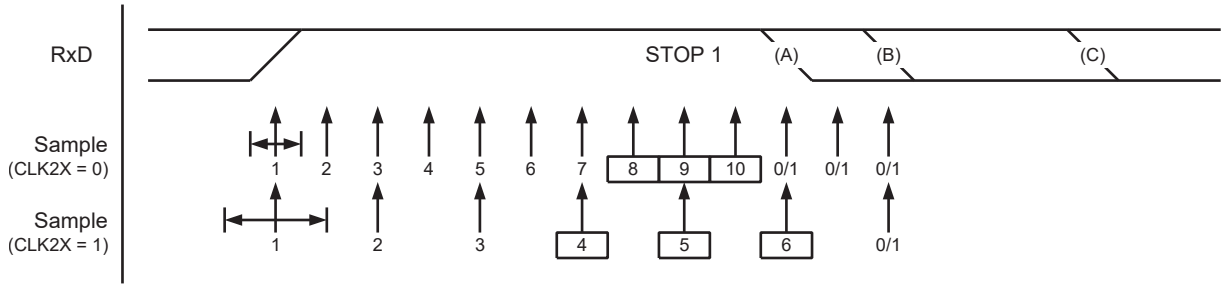
The data recovery unit uses sixteen samples in Normal mode and eight samples in Double-Speed mode for each bit. The following figure shows the sampling process of data and parity bits.

Figure 23-8. Sampling of Data and Parity Bits



As for Start bit detection, an identical majority voting technique is used on the three center samples for deciding of the logic level of the received bit. The process is repeated for each bit until a complete frame is received. It includes the first Stop bit but excludes additional ones. If the sampled Stop bit is a '0' value, the Frame Error (FERR in USARTn.RXDATAH) flag will be set. The next figure shows the sampling of the Stop bit in relation to the earliest possible beginning of the next frame's Start bit.

Figure 23-9. Stop Bit and Next Start Bit Sampling



A new high-to-low transition indicating the Start bit of a new frame can come right after the last of the bits used for majority voting. For Normal-Speed mode, the first low-level sample can be at the point marked

23.5.3 Transmit Data Register Low Byte

Name: TXDATAL
Offset: 0x02
Reset: 0x00
Property: R/W

The Transmit Data Buffer (TXB) register will be the destination for data written to the USARTn.TXDATAL register location.

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the transmitter and set to '0' by the receiver.

The transmit buffer can only be written when the DREIF flag in the USARTn.STATUS register is set. Data written to DATA when the DREIF flag is not set will be ignored by the USART transmitter. When data is written to the transmit buffer, and the transmitter is enabled, the transmitter will load the data into the Transmit Shift register when the Shift register is empty. The data is then transmitted on the TxD pin.

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
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

Bits 7:0 – DATA[7:0] Transmit Data Register

23.5.11 Debug Control Register

Name: DBGCTRL
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run

Value	Description
0	The peripheral is halted in Break Debug mode and ignores events
1	The peripheral will continue to run in Break Debug mode when the CPU is halted

24.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	BUFEN	BUFWR				SSD	MODE[1:0]	
Access	R/W	R/W				R/W	R/W	R/W
Reset	0	0				0	0	0

Bit 7 – BUFEN Buffer Mode Enable
Writing this bit to '1' enables Buffer mode, meaning two buffers in receive direction, one buffer in transmit direction, and separate interrupt flags for both transmit complete and receive complete.

Bit 6 – BUFWR Buffer Mode Wait for Receive
When writing this bit to '0' the first data transferred will be a dummy sample.

Value	Description
0	One SPI transfer must be completed before the data is copied into the Shift register.
1	When writing to the data register when the SPI is enabled and $\overline{SS}$ is high, the first write will go directly to the Shift register.

Bit 2 – SSD Slave Select Disable
When this bit is set and when operating as SPI Master (MASTER=1 in SPIn.CTRLA), $\overline{SS}$ does not disable Master mode.

Value	Description
0	Enable the Slave Select line when operating as SPI Master
1	Disable the Slave Select line when operating as SPI Master

Bits 1:0 – MODE[1:0] Mode
These bits select the Transfer mode. The four combinations of SCK phase and polarity with respect to the serial data are shown in the table below. These bits decide whether the first edge of a clock cycle (leading edge) is rising or falling and whether data setup and sample occur on the leading or trailing edge. When the leading edge is rising, the SCK signal is low when idle, and when the leading edge is falling, the SCK signal is high when idle.

Value	Name	Description
0x0	0	Leading edge: Rising, sample Trailing edge: Falling, setup
0x1	1	Leading edge: Rising, setup Trailing edge: Falling, sample
0x2	2	Leading edge: Falling, sample Trailing edge: Rising, setup
0x3	3	Leading edge: Falling, setup Trailing edge: Rising, sample

25.5.8 Master DATA

Name: MDATA
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
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

Bits 7:0 – DATA[7:0] Data
The bit field gives direct access to the master's physical Shift register which is used both to shift data out onto the bus (write) and to shift in data received from the bus (read).

The direct access implies that the Data register cannot be accessed during byte transmissions. Built-in logic prevents any write access to this register during the shift operations. Reading valid data or writing data to be transmitted can only be successfully done when the bus clock (SCL) is held low by the master (i.e., when the CLKHOLD bit in the Master Status register (TWIn.MSTATUS) is set). However, it is not necessary to check the CLKHOLD bit in software before accessing this register if the software keeps track of the present protocol state by using interrupts or observing the interrupt flags.

Accessing this register assumes that the master clock hold is active, auto-triggers bus operations dependent of the state of the Acknowledge Action Command bit (ACKACT) in TWIn.MSTATUS and type of register access (read or write).

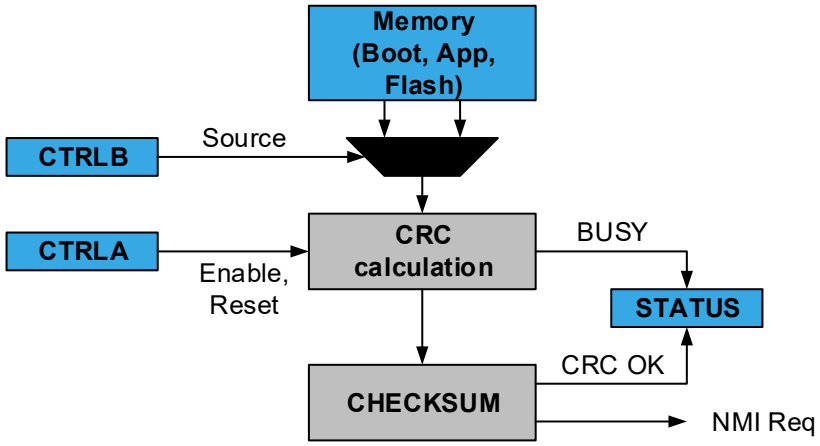
A write access to this register will, independent of ACKACT in TWIn.MSTATUS, command the master to perform a byte transmit operation on the bus directly followed by receiving the Acknowledge bit from the slave. When the Acknowledge bit is received, the Master Write Interrupt Flag (WIF) in TWIn.MSTATUS is set regardless of any bus errors or arbitration. If operating in a multi-master environment, the interrupt handler or application software must check the Arbitration Lost Status Flag (ARBLOST) in TWIn.MSTATUS before continuing from this point. If the arbitration was lost, the application software must decide to either abort or to resend the packet by rewriting this register. The entire operation is performed (i.e., all bits are clocked), regardless of winning or losing arbitration before the write interrupt flag is set. When arbitration is lost, only '1's are transmitted for the remainder of the operation, followed by a write interrupt with ARBLOST flag set.

Both TWI Master Interrupt Flags are cleared automatically when this register is written. However, the Master Arbitration Lost and Bus Error flags are left unchanged.

Reading this register triggers a bus operation, dependent on the setting of the Acknowledge Action Command bit (ACKACT) in TWIn.MSTATUS. Normally the ACKACT bit is preset to either ACK or NACK before the register read operation. If ACK or NACK action is selected, the transmission of the acknowledge bit precedes the release of the clock hold. The clock is released for one byte, allowing the slave to put one byte of data on the bus. The Master Read Interrupt flag RIF in TWIn.MSTATUS is then set if the procedure was successfully executed. However, if arbitration was lost when sending NACK, or a bus error occurred during the time of operation, the Master Write Interrupt flag (WIF) is set instead. Observe that the two Master Interrupt Flags are mutually exclusive (i.e., both flags will not be set simultaneously).

26.2.1 Block Diagram

Figure 26-2. Cyclic Redundancy Check Block Diagram



26.2.2 System Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 26-1. System Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	No	-
Debug	Yes	UPDI

Related Links

11.2.2.1 Clocks
26.2.2.3 Interrupts

26.2.2.1 Clocks

This peripheral depends on the peripheral clock.

Related Links

10. CLKCTRL - Clock Controller

26.2.2.2 I/O Lines and Connections

Not applicable.

26.2.2.3 Interrupts

Using the interrupts of this peripheral requires the interrupt controller to be configured first.

Related Links

13. CPUINT - CPU Interrupt Controller
8.7.3 SREG
26.3.3 Interrupts

The Sequential block is clocked by the same clock as the corresponding LUT. This is configured by the Clock Source bit (CLKSRC) in the LUT n Control A register (CCL.LUTnCTRLA).

When the even LUT (LUT0) is disabled, the latch is asynchronously cleared, during which the flip-flop Reset signal (R) is kept enabled for one clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on the rising edge of the clock, as shown in the respective *Characteristics* tables.

Gated D Flip-Flop (DFF)

The D-input is driven by the even LUT output (LUT0), and the G-input is driven by the odd LUT output (LUT1).

Figure 27-6. D Flip-Flop

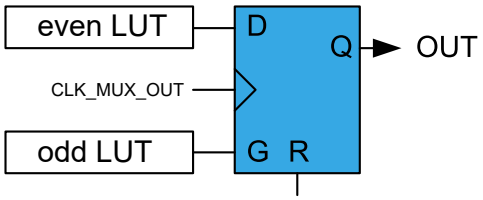


Table 27-3. DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
	0	X	Hold state (no change)

JK Flip-Flop (JK)

The J-input is driven by the even LUT output (LUT0), and the K-input is driven by the odd LUT output (LUT1).

Figure 27-7. JK Flip-Flop

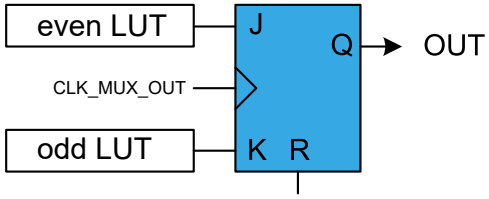


Table 27-4. JK Characteristics

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

27.5.2 Sequential Control 0

Name: SEQCTRL0
Offset: 0x01 [ID-00000485]
Reset: 0x00
Property: Enable-Protected

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

Bits 3:0 – SEQSEL[3:0] Sequential Selection
These bits select the sequential configuration.

Value	Name	Description
0x0	DISABLE	Sequential logic is disabled
0x1	DFF	D flip-flop
0x2	JK	JK flip-flop
0x3	LATCH	D latch
0x4	RS	RS latch
Other	-	Reserved

6. In ADCn.CTRLC select $SAMPCAP = 1$
7. Acquire the temperature sensor output voltage by starting a conversion.
8. Process the measurement result as described below.

The measured voltage has a linear relationship to the temperature. Due to process variations, the temperature sensor output voltage varies between individual devices at the same temperature. The individual compensation factors are determined during the production test and saved in the Signature Row:

- SIGROW.TEMPSENSE0 is a gain/slope correction
- SIGROW.TEMPSENSE1 is an offset correction

In order to achieve accurate results, the result of the temperature sensor measurement must be processed in the application software using factory calibration values. The temperature (in Kelvin) is calculated by this rule:

```
Temp = (((RESH << 8) | RESL) - TEMPSENSE1) * TEMPSENSE0) >> 8
```

RESH and RESL are the high and low bytes of the Result register (ADCn.RES), and TEMPSENSEn are the respective values from the Signature row.

It is recommended to follow these steps in user code:

```
int8_t sigrow_offset = SIGROW.TEMPSENSE1;  // Read signed value from signature row
uint8_t sigrow_gain = SIGROW.TEMPSENSE0;    // Read unsigned value from signature row
uint16_t adc_reading = 0;   // ADC conversion result with 1.1 V internal reference

uint32_t temp = adc_reading - sigrow_offset;
temp *= sigrow_gain;  // Result might overflow 16 bit variable (10bit+8bit)
temp += 0x80;               // Add 1/2 to get correct rounding on division below
temp >>= 8;                 // Divide result to get Kelvin
uint16_t temperature_in_K = temp;
```

Related Links

6.10.2.3 TEMPSENSEn

29.3.2.7 Window Comparator Mode

The ADC can raise the WCOMP flag in the Interrupt and Flag register (ADCn.INTFLAG) and request an interrupt (WCOMP) when the result of a conversion is above and/or below certain thresholds. The available modes are:

- The result is under a threshold
- The result is over a threshold
- The result is inside a window (above a lower threshold, but below the upper one)
- The result is outside a window (either under the lower or above the upper threshold)

The thresholds are defined by writing to the Window Comparator Threshold registers (ADCn.WINLT and ADCn.WINHT). Writing to the Window Comparator mode bit field (WINCM) in the Control E register (ADCn.CTRLE) selects the conditions when the flag is raised and/or the interrupt is requested.

Assuming the ADC is already configured to run, follow these steps to use the Window Comparator mode:

1. Choose which Window Comparator to use (see the WINCM description in ADCn.CTRLE), and set the required threshold(s) by writing to ADCn.WINLT and/or ADCn.WINHT.
2. Optional: enable the interrupt request by writing a '1' to the Window Comparator Interrupt Enable bit (WCOMP) in the Interrupt Control register (ADCn.INTCTRL).

vary, depending on the status of the oscillator when the UPDI is enabled. After this duration, the data line will be released by the UPDI and pulled high.

When the debugger detects that the line is high, the initial SYNCH character (0x55) must be sent to properly enable the UPDI for communication. If the Start bit of the SYNCH character is not sent well within maximum T_{DebZ}, the UPDI will disable itself, and the enable sequence must be repeated. This time is based on counted cycles on the 4 MHz UPDI clock, which is the default when enabling the UPDI. The disable is performed to avoid the UPDI being enabled unintentionally.

After successful SYNCH character transmission, the first instruction frame can be transmitted.

Related Links

31.17 UPDI Timing

31.17 UPDI Timing

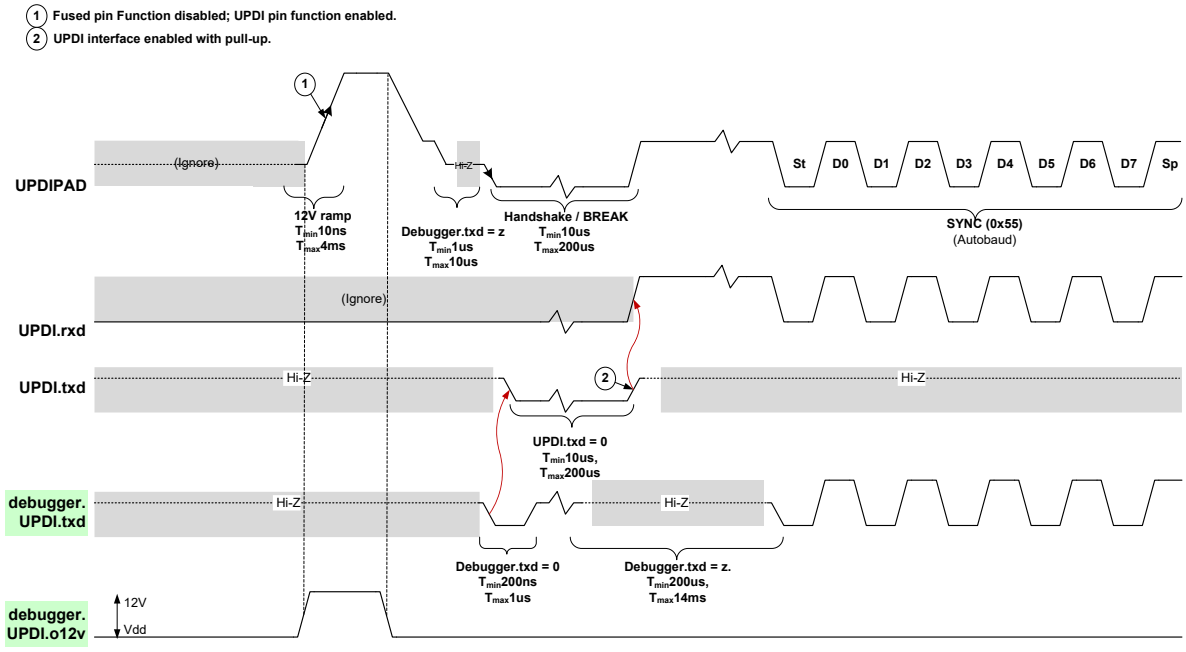
30.3.2.2 UPDI Enable with 12V Override of $\overline{\text{RESET}}$ Pin

GPIO or Reset functionality on the $\overline{\text{RESET}}$ pin can be overridden by the UPDI by using 12V programming. By applying a 12V pulse to the $\overline{\text{RESET}}$ pin, the pin functionality is switched to UPDI, independent of RSTPINCFG in FUSE.SYSCFG0. It is recommended to always reset the device before starting the 12V enable sequence.

During power-up, the Power-on Reset (POR) must be released before the 12V pulse can be applied. The duration of the pulse is recommended in the range from 100 µs to 1 ms, before tri-stating. When applying the rising edge of the 12V pulse, the UPDI will be reset. After tri-stating, the UPDI will remain in Reset until the $\overline{\text{RESET}}$ pin is driven low by the debugger. This will release the UPDI Reset and initiate the same enable sequence as explained in 30.3.2.1 UPDI Enable with Fuse Override of RESET Pin.

The following figure shows the 12V enable sequence.

Figure 30-6. UPDI Enable Sequence by 12V Programming



When enabled by 12V, only a POR will disable the UPDI configuration on the $\overline{\text{RESET}}$ pin, and restore the default setting. If issuing a UPDI Disable command through the UPDIDIS bit in UPDI.CTRLB, the UPDI will be reset and the clock request will be canceled, but the $\overline{\text{RESET}}$ pin will remain in UPDI configuration.

31.11 USART

Figure 31-3. USART in SPI Mode - Timing Requirements in Master Mode

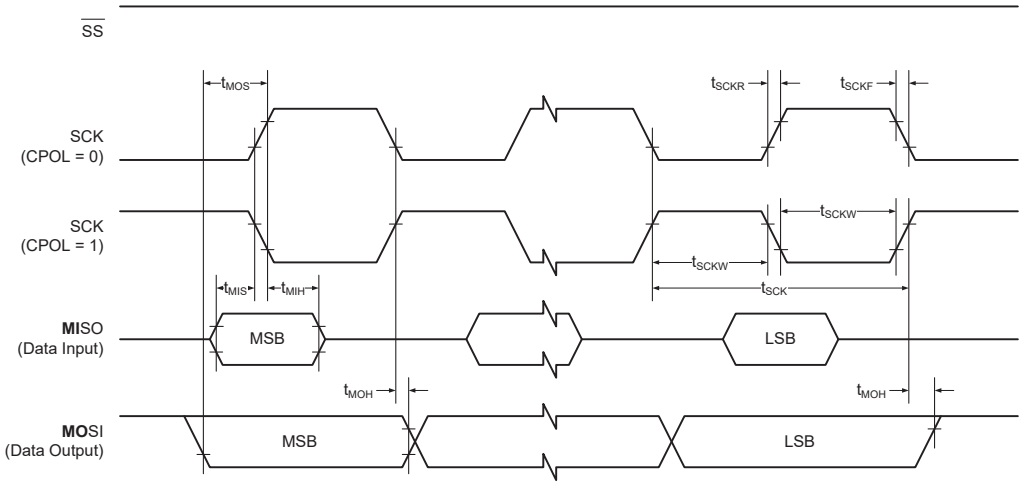


Table 31-16. USART in SPI Master Mode - Timing Characteristics

Symbol	Description	Condition	Min.	Typ.	Max.	Unit
f_{SCK}	SCK clock frequency	Master	-	-	10	MHz
t_{SCK}	SCK period	Master	100	-	-	ns
t_{SCKW}	SCK high/low width	Master	-	$0.5 \times t_{SCK}$	-	ns
t_{SCKR}	SCK rise time	Master	-	2.7	-	ns
t_{SCKF}	SCK fall time	Master	-	2.7	-	ns
t_{MIS}	MISO setup to SCK	Master	-	10	-	ns
t_{MIH}	MISO hold after SCK	Master	-	10	-	ns
t_{MOS}	MOSI setup to SCK	Master	-	$0.5 \times t_{SCK}$	-	ns
t_{MOH}	MOSI hold after SCK	Master	-	1.0	-	ns

Abbreviation	Description
INT	Interrupt
IrDA	Infrared Data Association
IVEC	Interrupt Vector
LSB	Least Significant Byte
LSb	Least Significant bit
LUT	Look Up Table
MBIST	Memory Built-in Self-test
MSB	Most Significant Byte
MSb	Most Significant bit
NACK	Not Acknowledge
NMI	Non-maskable interrupt
NVM	Nonvolatile Memory
NVMCTRL	Nonvolatile Memory Controller
OPAMP	Operation Amplifier
OSC	Oscillator
PC	Program Counter
PER	Period
POR	Power-on Reset
PORT	I/O Pin Configuration
PTC	Peripheral Touch Controller
PWM	Pulse-width Modulation
RAM	Random Access Memory
REF	Reference
REQ	Request
RISC	Reduced Instruction Set Computer
RSTCTRL	Reset Controller
RTC	Real-time Counter
RX	Receiver/Receive
SERCOM	Serial Communication Interface
SLPCTRL	Sleep Controller
SMBus	System Management Bus
SP	Stack Pointer