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, PWM, WDT
Number of I/O	6
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.154", 3.90mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny402-ssnr

8.5.6 Accessing 16-Bit Registers

The AVR data bus is 8-bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

This ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the register.

Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can be read and written directly from user software.

8.5.7 Configuration Change Protection (CCP)

System critical I/O register settings are protected from accidental modification. Flash self-programming (via store to NVM controller) is protected from accidental execution. This is handled globally by the Configuration Change Protection (CCP) register.

Changes to the protected I/O registers or bits, or execution of protected instructions, are only possible after the CPU writes a signature to the CCP register. The different signatures are listed in the description of the CCP register (CPU.CCP).

There are two modes of operation: one for protected I/O registers, and one for the protected self-programming.

Related Links

CCP

8.5.7.1 Sequence for Write Operation to Configuration Change Protected I/O Registers

In order to write to registers protected by CCP, these steps are required:

1. The software writes the signature that enables change of protected I/O registers to the CCP bit field in the CPU.CCP register.
2. Within four instructions, the software must write the appropriate data to the protected register. Most protected registers also contain a write enable/change enable/lock bit. This bit must be written to '1' in the same operation as the data are written.

   The protected change is immediately disabled if the CPU performs write operations to the I/O register or data memory, if load or store accesses to Flash, NVMCTRL, EEPROM are conducted, or if the `SLEEP` instruction is executed.

8.5.7.2 Sequence for Execution of Self-Programming

In order to execute self-programming (the execution of writes to the NVM controller's command register), the following steps are required:

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
							CLKSEL[1:0]	
Access		R	R	R	R	R	R/W	R/W
Reset		0	0	0	0	0	0	0

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

13.3.2.3 Interrupt Response Time

The minimum interrupt response time for all enabled interrupts is three CPU clock cycles: one cycle to finish the ongoing instruction, two cycles to store the Program Counter to the stack, and three cycles(1) to jump to the interrupt handler (`JMP`).

After the Program Counter is pushed on the stack, the program vector for the interrupt is executed. See the figure below, first diagram.

The jump to the interrupt handler takes three clock cycles(1). If an interrupt occurs during execution of a multicycle instruction, this instruction is completed before the interrupt is served. See the figure below, second diagram.

If an interrupt occurs when the device is in sleep mode, the interrupt execution response time is increased by five clock cycles. In addition, the response time is increased by the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four to five clock cycles, depending on the size of the Program Counter. During these clock cycles, the Program Counter is popped from the stack and the Stack Pointer is incremented. See the figure above, third diagram.

16.7.3 Input Value

Name: IN
Offset: 0x02
Reset: 0x00
Property: -

Writing to the Virtual PORT registers has the same effect as writing to the regular registers, but allows for memory-specific instructions, such as bit-manipulation instructions, which are not valid for the extended I/O memory space where the regular PORT registers reside.

Bit	7	6	5	4	3	2	1	0
	IN[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 – IN[7:0] Input Value
This bit field holds the value present on the pins if the digital input buffer is enabled.
Writing to a bit of VPORT.IN will toggle the corresponding bit in VPORT.OUT.

20.6 Register Summary - TCA in Split Mode (CTRLD.SPLITM=1)

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0					CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0				HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
0x02	CTRLC	7:0				HCMP0OV		LCMP2OV	LCMP1OV	LCMP0OV
0x03	CTRLD	7:0								SPLITM
0x04	CTRLECLR	7:0					CMD[1:0]		CMDEN[1:0]	
0x05	CTRLESET	7:0					CMD[1:0]		CMDEN[1:0]	
0x06 ... 0x09	Reserved									
0x0A	INTCTRL	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0B	INTFLAGS	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0C ... 0x0D	Reserved									
0x0E	DBGCTRL	7:0								DBGRUN
0x0F ... 0x1F	Reserved									
0x20	LCNT	7:0	LCNT[7:0]							
0x21	HCNT	7:0	HCNT[7:0]							
0x22 ... 0x25	Reserved									
0x26	LPER	7:0	LPER[7:0]							
0x27	HPER	7:0	HPER[7:0]							
0x28	LCMP0	7:0	LCMP[7:0]							
0x29	HCMP0	7:0	HCMP[7:0]							
0x2A	LCMP1	7:0	LCMP[7:0]							
0x2B	HCMP1	7:0	HCMP[7:0]							
0x2C	LCMP2	7:0	LCMP[7:0]							
0x2D	HCMP2	7:0	HCMP[7:0]							

20.7 Register Description - Split Mode

20.7.2 Control B - Split Mode

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

Bit	7	6	5	4	3	2	1	0
				HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
Access				R/W		R/W	R/W	R/W
Reset				0		0	0	0

Bit 4 – HCMP0EN High byte Compare 0 Enable
See LCMP0EN.

Bit 2 – LCMP2EN Low byte Compare 2 Enable
See LCMP0EN.

Bit 1 – LCMP1EN Low byte Compare 1 Enable
See LCMP0EN.

Bit 0 – LCMP0EN Low byte Compare 0 Enable
Setting the LCMPnEN/HCMPnEN bits in the FRQ or PWM Waveform Generation mode of operation will override the port output register for the corresponding WOn pin.

20.7.6 Control Register E Set - Split Mode

Name: CTRLESET
Offset: 0x05
Reset: 0x00
Property: -

The individual Status bit can be set by writing a '1' to its bit location. This allows each bit to be set without the use of a read-modify-write operation on a single register.

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

Bits 3:2 – CMD[1:0] Command
These bits are used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero. The CMD bits must be used together with CMDEN. Using the reset command requires that both low-byte and high-byte timer/counter is selected.

Value	Name	Description
0x0	NONE	No command
0x1	-	Reserved
0x2	RESTART	Force restart
0x3	RESET	Force hard Reset (ignored if TC is enabled)

Bits 1:0 – CMDEN[1:0] Command enable
These bits are used to indicate for which timer/counter the command (CMD) is valid.

Value	Name	Description
0x0	NONE	None
0x1	LOW	Command valid for low-byte T/C
0x2	HIGH	Command valid for high-byte T/C
0x3	BOTH	Command valid for both low-byte and high-byte T/C

20.7.9 Debug Control Register

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00
Property: -

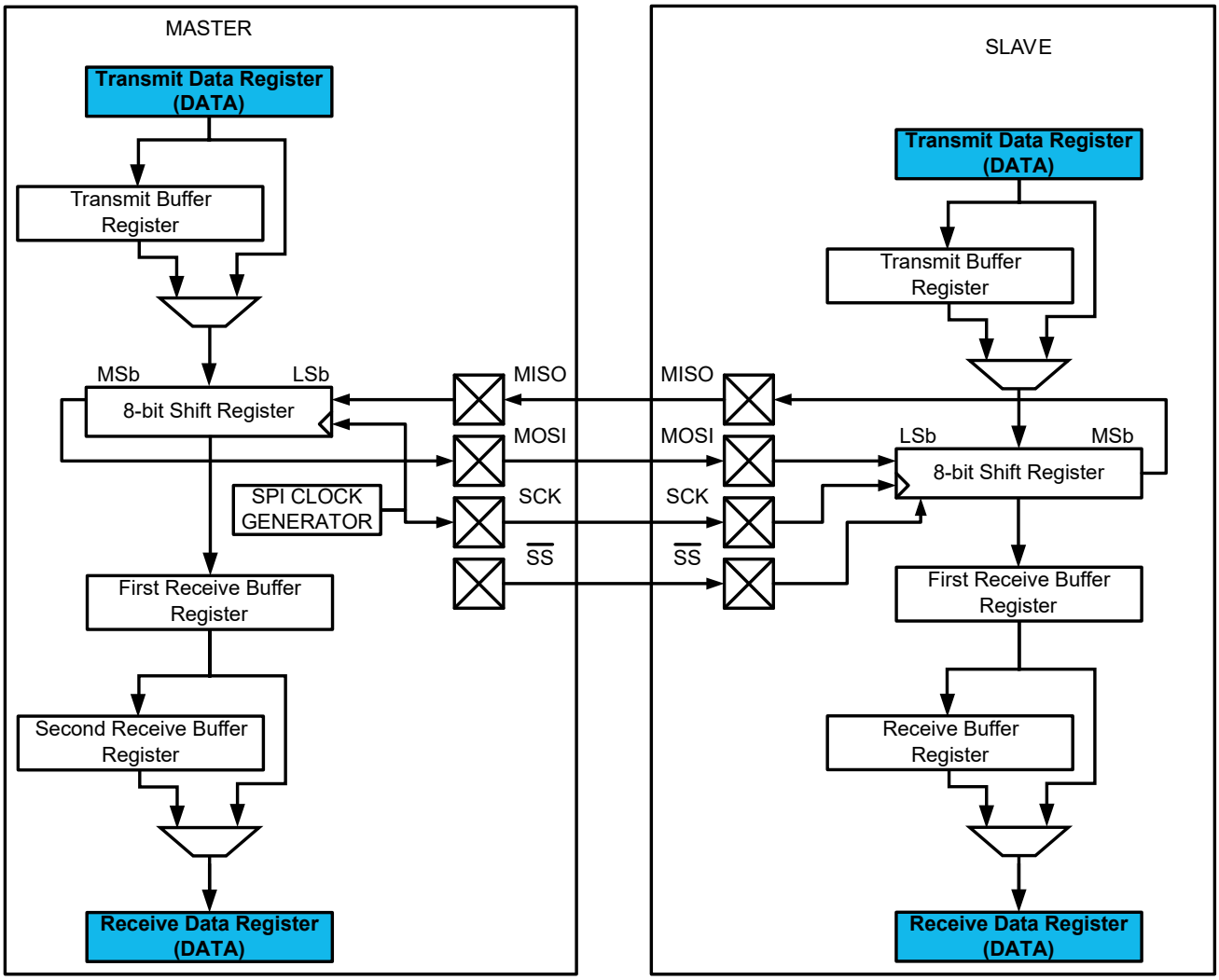
Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Run in Debug

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

24.2.1 Block Diagram

Figure 24-1. SPI Block Diagram



The SPI is built around an 8-bit Shift register that will shift data out and in at the same time. The Transmit Data register and the Receive Data register are not physical registers but are mapped to other registers when written or read: Writing the Transmit Data register (SPIn.DATA) will write the Shift register in Normal mode and the Transmit Buffer register in Buffer mode. Reading the Receive Data register (SPIn.DATA) will read the First Receive Buffer register in normal mode and the Second Receive Data register in Buffer mode.

In Master mode the SPI has a clock generator to generate the SCK clock. In Slave mode the received SCK clock is synchronized and sampled to trigger the shifting of data in the Shift register.

25.5.2 Debug Control

Name: DBGCTRL
Offset: 0x02
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.

27.5.3 LUT n Control A

Name: LUTCTRLA
Offset: 0x05 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	EDGEDET	CLKSRC	FILTSEL[1:0]		OUTEN			ENABLE
Access	R/W	R/W	R/W	R/W	R/W			R/W
Reset	0	0	0	0	0			0

Bit 7 – EDGEDET Edge Detection

Value	Description
0	Edge detector is disabled
1	Edge detector is enabled

Bit 6 – CLKSRC Clock Source Selection
This bit selects whether the peripheral clock (CLK_PER) or any input present on input line 2 (IN[2]) is used as clock (CLK_MUX_OUT) for an LUT.

The CLK_MUX_OUT of the even LUT is used for clocking the Sequential block of an LUT pair.

Value	Description
0	CLK_PER is clocking the LUT
1	IN[2] is clocking the LUT

Bits 5:4 – FILTSEL[1:0] Filter Selection
These bits select the LUT output filter options:

Value	Name	Description
0x0	DISABLE	Filter disabled
0x1	SYNCH	Synchronizer enabled
0x2	FILTER	Filter enabled
0x3	-	Reserved

Bit 3 – OUTEN Output Enable
This bit enables the LUT output to the LUTnOUT pin. When written to '1', the pin configuration of the PORT I/O Controller is overridden.

Value	Description
0	Output to pin disabled
1	Output to pin enabled

Bit 0 – ENABLE LUT Enable

Value	Description
0	The LUT is disabled
1	The LUT is enabled

28.4 Register Summary - AC

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY	OUTEN	INTMODE[1:0]			HYSMODE[1:0]		ENABLE
0x01	Reserved									
0x02	MUXCTRLA	7:0	INVERT				MUXPOS		MUXNEG[1:0]	
0x03 ... 0x05	Reserved									
0x06	INTCTRL	7:0								CMP
0x07	STATUS	7:0				STATE				CMP

28.5 Register Description

29. Analog-to-Digital Converter (ADC)

29.1 Features

- 10-Bit Resolution
- ±2 LSB Absolute Accuracy
- 6.5 - 260 µs Conversion Time
- Up to 115 ksps at 10-Bit Resolution (150 ksps at 8-bit)
- Up to Six Multiplexed Single-ended Input Channels
- Temperature Sensor Input Channel
- 0V to V_{DD} ADC Input Voltage Range
- Multiple Internal ADC Reference Voltages Between 0.55V and V_{DD}
- Free-running or Single Conversion mode
- Interrupt Available on ADC Conversion Complete
- Optional Event Triggered Conversion
- Optional Interrupt on Conversion Results
- Compare Function for Accurate Monitoring or User-Defined Thresholds (Window Comparator mode)
- Accumulation up to 64 Samples per Conversion

29.2 Overview

The Analog-to-Digital Converter (ADC) peripheral features a 10-bit Successive Approximation ADC (SAR), with a sampling rate up to 115 ksps at 10-bit resolution (150 ksps at 8-bit). The ADC is connected to a 6-channel analog multiplexer, which allows twelve single-ended voltage inputs. The single-ended voltage inputs refer to 0V (GND). The ADC input channel can either be internal (e.g. a voltage reference) or external through the analog input pins.

An ADC conversion can be started by software or by using the Event System (EVSYS) to route an event from other peripherals, making it possible to do a periodic sampling of input signals, trigger an ADC conversion on a special condition, or trigger an ADC conversion in Standby Sleep mode. A window compare feature is available for monitoring the input signal and can be configured to only trigger an interrupt on user-defined thresholds for under, over, inside, or outside a window, with minimum software intervention required.

The ADC input signal is fed through a sample-and-hold circuit that ensures that the input voltage to the ADC is held at a constant level during sampling.

The ADC supports sampling in bursts where a configurable number of conversion results are accumulated into a single ADC result (Sample Accumulation). Further, a sample delay can be configured to tune the ADC sampling frequency associated with a single burst. This is to tune the sampling frequency away from any harmonic noise aliased with the ADC sampling frequency (within the burst) from the sampled signal. An automatic sampling delay variation feature can be used to randomize this delay to slightly change the time between samples.

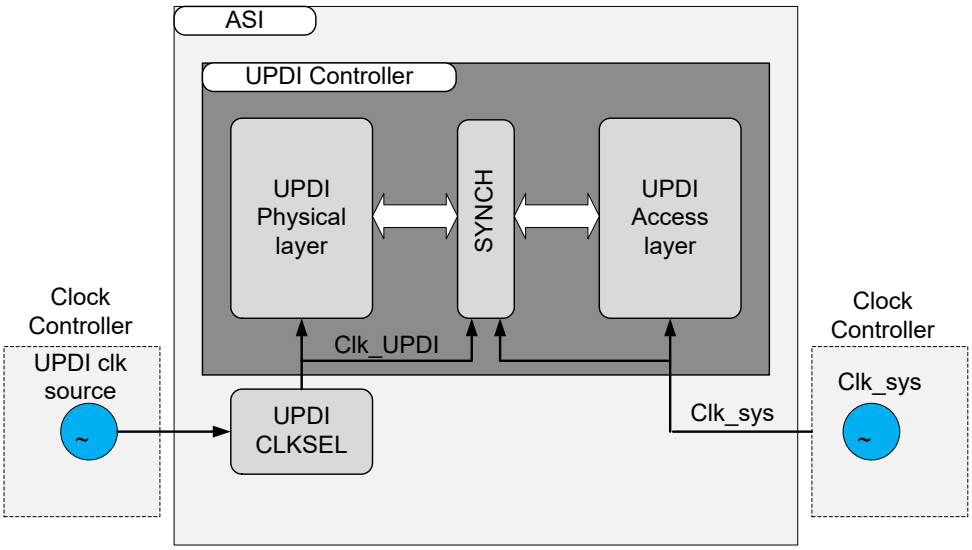
Internal reference voltages from Voltage Reference (VREF) or V_{DD} are provided on-chip.

This device has one instance of the ADC peripheral; ADC0.

30.2.2.1 Clocks

The UPDI Physical (UPDI PHY) layer and UPDI Access (UPDI ACC) layer can operate on different clock domains. The UPDI PHY layer clock is derived from an internal oscillator, and the UPDI ACC layer clock is the same as the system clock. There is a synchronization boundary between the UPDI PHY layer and the UPDI ACC layer, which ensures correct operation between the clock domains. The UPDI clock output frequency is selected through the ASI, and the default UPDI clock start-up frequency is 4 MHz after enabling the UPDI. The UPDI clock frequency is changed by writing the UPDICLKSEL bits in the ASI_CTRLA register.

Figure 30-2. UPDI Clock Domains



Related Links

Clock Controller (CLKCTRL)

30.2.2.2 I/O Lines and Connections

To operate the UPDI, the $\overline{\text{RESET}}$ pin must be set to UPDI mode. This is not done through the port I/O pin configuration as regular I/O pins, but through setting the $\overline{\text{RESET}}$ Pin Configuration (RSTPINCFG) bits in FUSE.SYSCFG0 as described in UPDI Enable with Fuse Override of RESET Pin, or by following the UPDI 12V enable sequence from UPDI Enable with 12V Override of RESET Pin. Pull enable, input enable and output enable settings are automatically controlled by the UPDI when active.

30.2.2.3 Events

The events of this peripheral are connected to the Event System.

Related Links

Event System (EVSYS)

30.2.2.4 Power Management

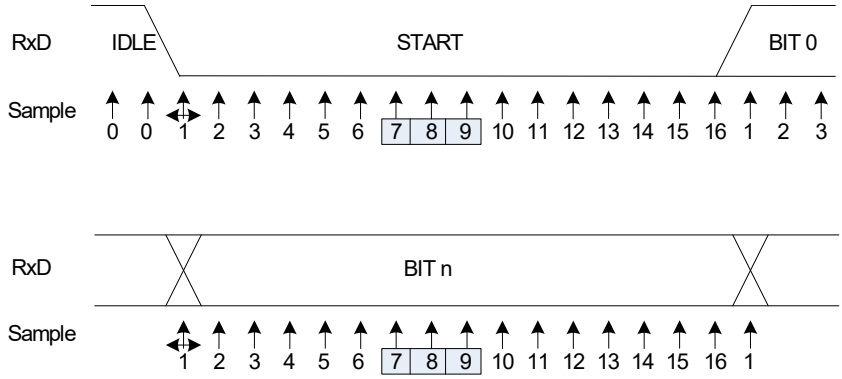
The UPDI physical layer continues to operate in any Sleep mode and is always accessible for a connected debugger, but read/write access to the system bus is restricted in Sleep modes where the CPU clock is switched off. The UPDI can be enabled at any time, independent of the system Sleep state. See Sleep Mode Operation for details on UPDI operation during Sleep modes.

30.3.1.1 UPDI UART

All transmission and reception of serial data on the UPDI is achieved using the UPDI frames presented in Figure 30-3. Communication is initiated from the master (debugger) side, and every transmission must start with a SYNCH character upon which the UPDI can recover the transmission baud rate, and store this setting for the coming data. The baud rate set by the SYNCH character will be used for both reception and transmission for the instruction byte received after the SYNCH. See UPDI Instruction Set for details on when the next SYNCH character is expected in the instruction stream.

There is no writable baud rate register in the UPDI, so the baud rate sampled from the SYNCH character is used for data recovery by sampling the Start bit, and performing a majority vote on the middle samples. This process is repeated for all bits in the frame, including the parity bit and two Stop bits. The baud generator uses 16 samples, and the majority voting is done on sample 7, 8, and 9.

Figure 30-4. UPDI UART Start Bit and Data/Parity/Stop Bit Sampling



The transmission baud rate must be set up in relation to the selected UPDI clock, which can be adjusted by UPDICLKSEL in UPDI.ASI_CTRLA. See Table 30-2 for recommended maximum and minimum baud rate settings.

Table 30-2. Recommended UART Baud Rate Based on UPDICLKSEL Setting

UPDICLKSEL[1:0]	MAX Recommended Baud Rate	MIN Recommended Baud Rate
0x1 (16 MHz)	0.9 Mbps	0.300 kbps
0x2 (8 MHz)	450 kbps	0.150 kbps
0x3 (4 MHz) - Default	225 kbps	0.075 kbps

The UPDI Baud Rate Generator utilizes fractional baud counting to minimize the transmission error. With the fixed frame format used by the UPDI, the maximum and recommended receiver transmission error limits can be seen in the following table:

Table 30-3. Receiver Baud Rate Error

Data + Parity Bits	R_{slow}	R_{fast}	Max. Total Error [%]	Recommended Max. RX Error [%]
9	96.39	104.76	+4.76/-3.61	+1.5/-1.5

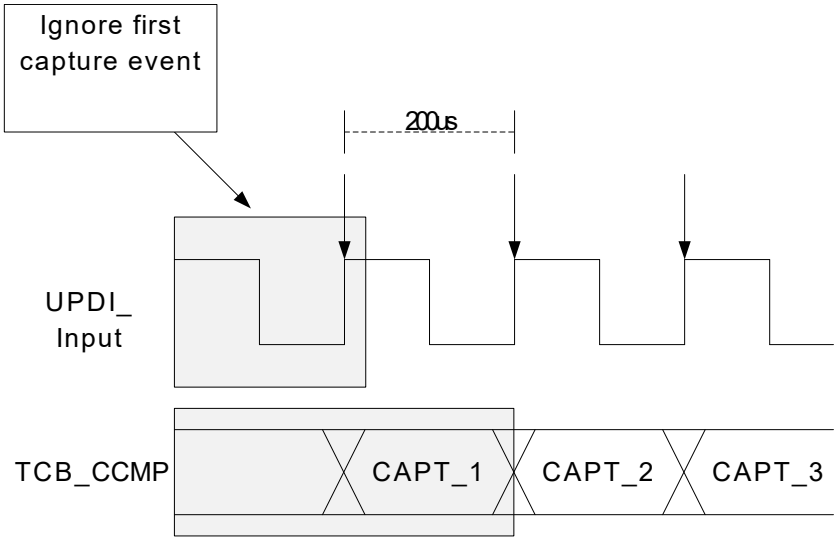
30.3.1.2 BREAK Character

The BREAK character is used to reset the internal state of the UPDI to the default setting. This is useful if the UPDI enters an error state due to a communication error, or when synchronization between the debugger and the UPDI is lost.

on the input event should be used for the measurement. See the figure below for an example using 10 kbps UPDI SYNCH character pulses, giving a capture window of 200 µs for the timer.

- It is possible to readout the captured value directly after the SYNCH character, by reading the TCBn.CCMP register or the value can be written to memory by the CPU once the capture is done.

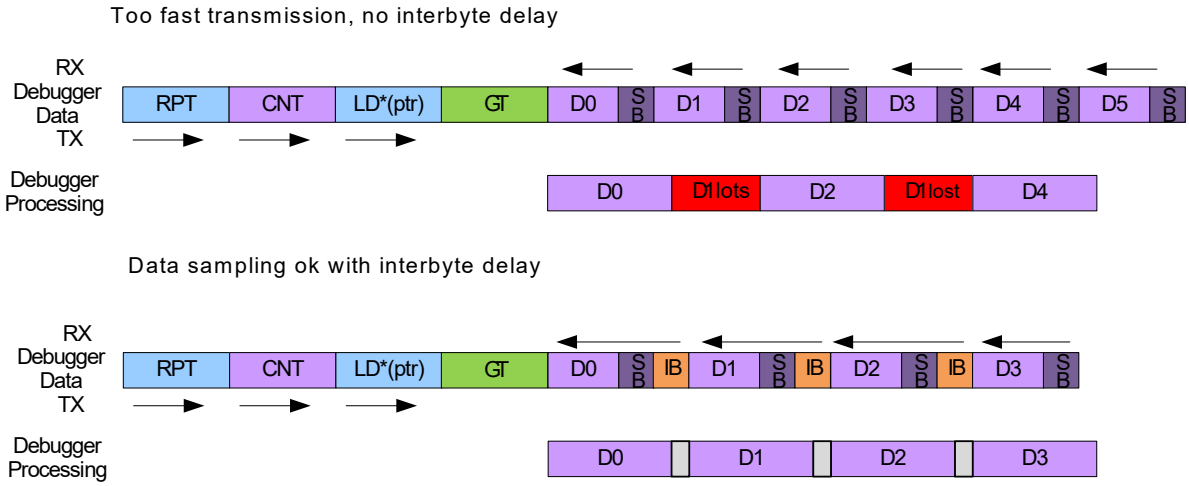
Figure 30-17. UPDI System Clock Measurement Events



30.3.5 Interbyte Delay

When loading data with the UPDI, or reading out the System Information Block, the output data will normally come out with two IDLE bits between each transmitted byte for a multibyte transfer. Depending on the application on the receiver side, data might be coming out too fast when there are no extra IDLE bits between each byte. By enabling the IBDLY feature in UPDI.CTRLB, two extra Stop bits will be inserted between each byte to relax the sampling time for the debugger. Interbyte delay works in the same way as a guard time, by inserting extra IDLE bits, but only a fixed number of IDLE bits and only for multibyte transfers. The first transmitted byte after a direction change will be subject to the regular Guard Time before it is transmitted, and the interbyte delay is not added to this time.

Figure 30-18. Interbyte Delay Example with LD and RPT



In Figure 30-18, GT denotes the Guard Time insertion, SB are Stop bits and IB is the inserted interbyte delay. The rest of the frames are data and instructions.

Figure 32-23. : I/O Pin Output Voltage vs. Sink Current (T=25°C)

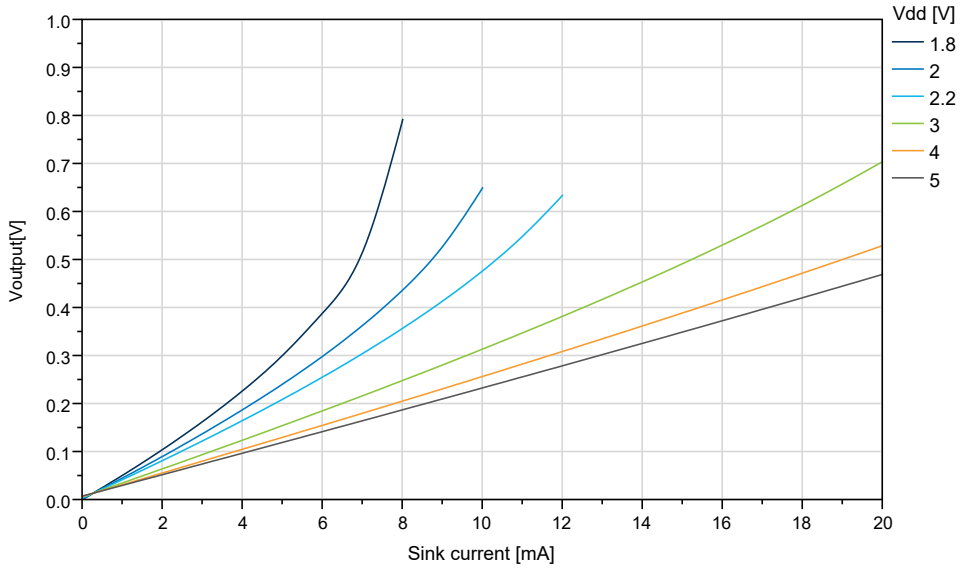
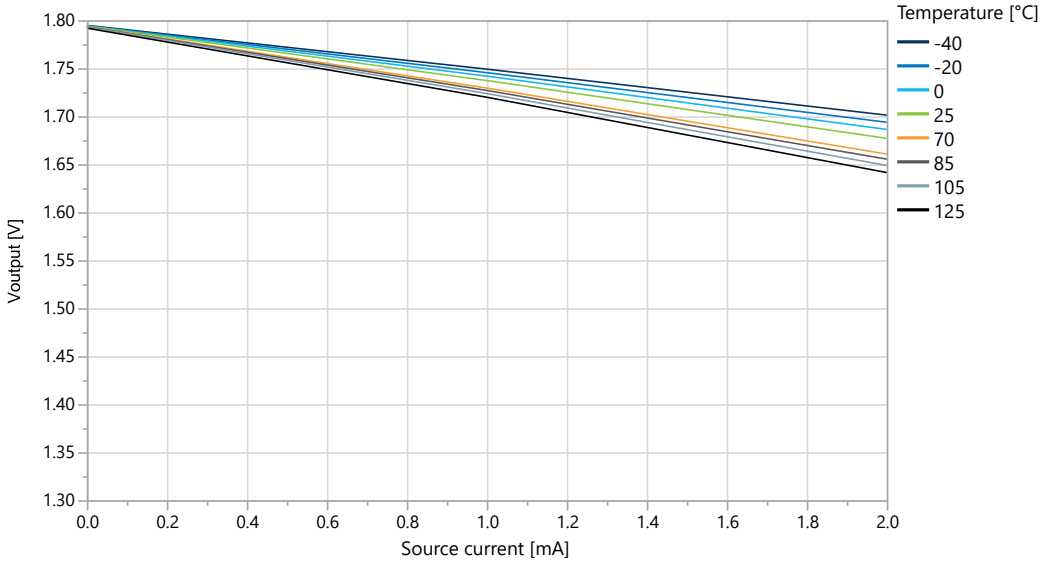


Figure 32-24. I/O Pin Output Voltage vs. Source Current (V_{DD}=1.8V)



32.5 ADC Characteristics

Figure 32-41. Absolute Accuracy vs. V_{DD} (115 ksps) at T=25°C

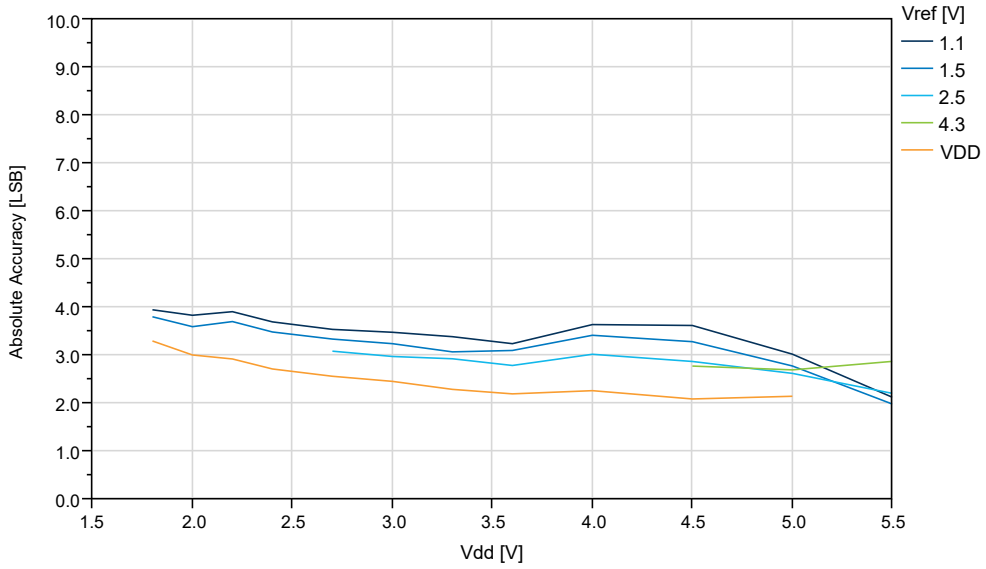
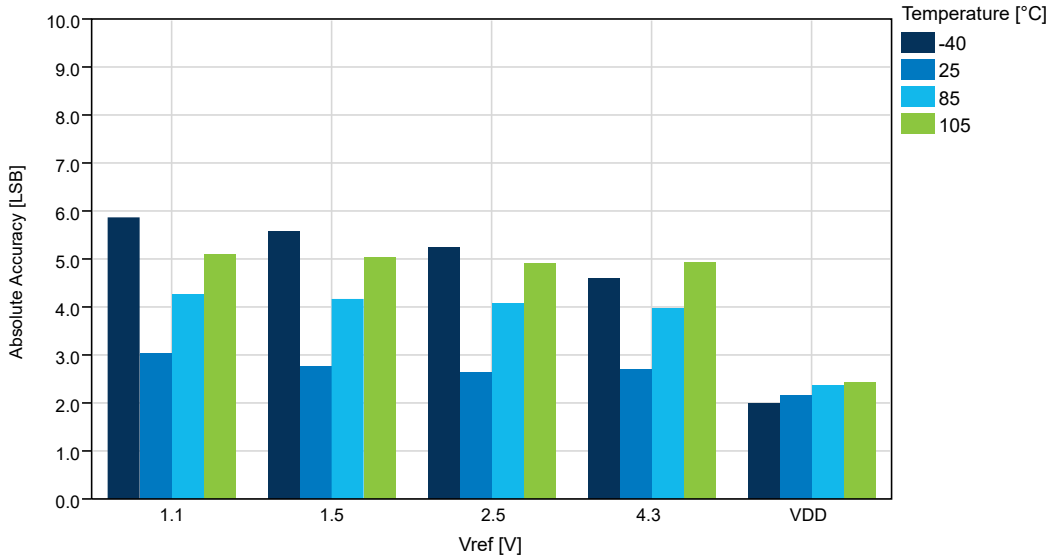


Figure 32-42. Absolute Accuracy vs. V_{ref} (V_{DD}=5.0V, 115 ksps)



32.8 OSCULP32K Characteristics

Figure 32-61. OSCULP32K Internal Oscillator Frequency vs. Temperature

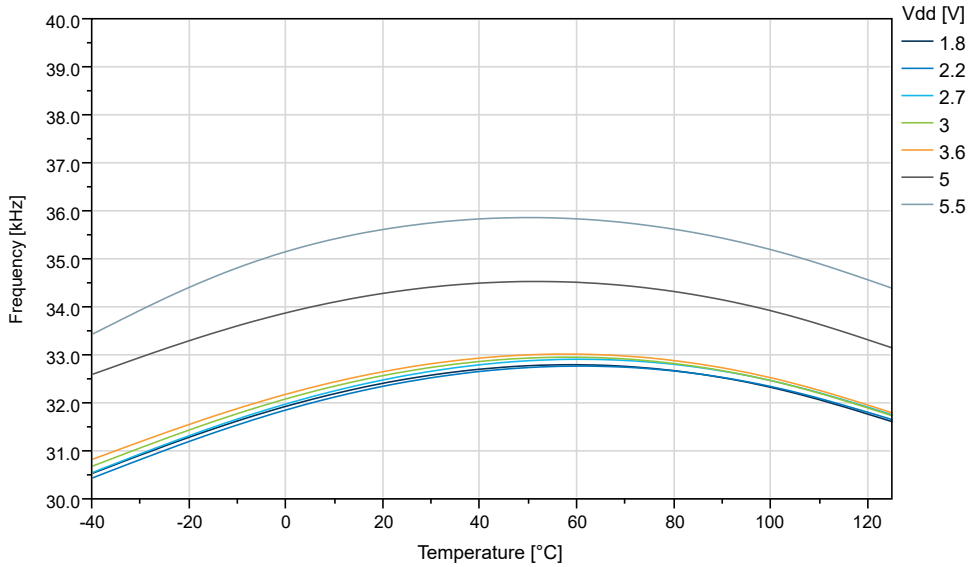
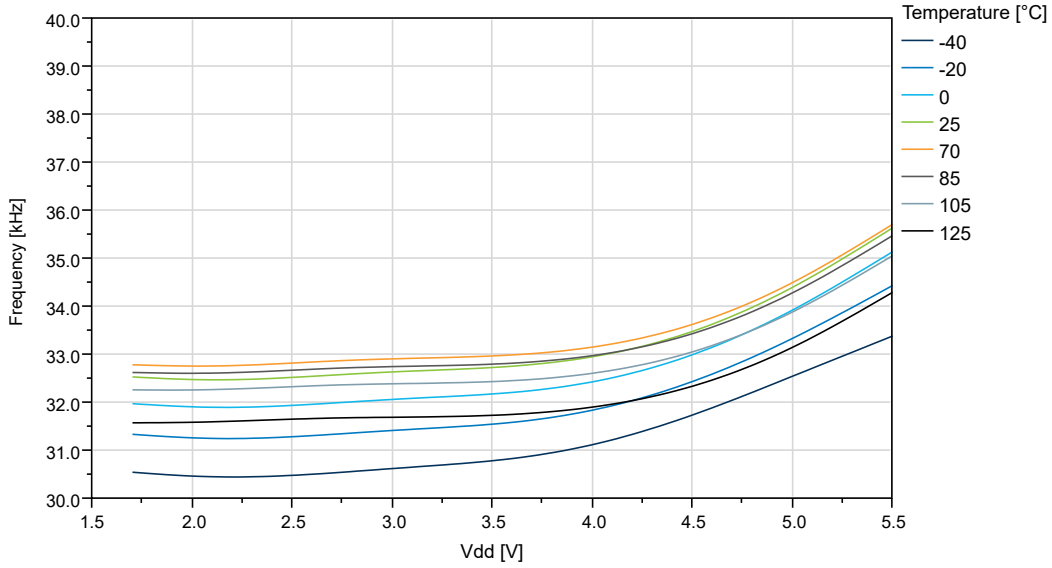


Figure 32-62. OSCULP32K Internal Oscillator Frequency vs. V_{DD}



Symbol	Description
GHz	1 GHz = 10^9 Hz = 1,000,000,000 Hz
s	second
ms	millisecond
µs	microsecond
ns	nanosecond

37.4 Registers and Bits

Table 37-4. Register and Bit Mnemonics

Symbol	Description
R/W	Read/Write accessible register bit. The user can read from and write to this bit.
R	Read-only accessible register bit. The user can only read this bit. Writes will be ignored.
W	Write-only accessible register bit. The user can only write this bit. Reading this bit will return an undefined value.
BIT	Bit names are shown in uppercase. (Example ENABLE)
FIELD[n:m]	A set of bits from bit n down to m. (Example: PINA[3:0] = {PINA3, PINA2, PINA1, PINA0}
Reserved	Reserved bits are unused and reserved for future use. For compatibility with future devices, always write reserved bits to zero when the register is written. Reserved bits will always return zero when read. Reserved bit field values must not be written to a bit field. A reserved value will not be read from a read-only bit field. Do not write any value to reserved bits of a fuse.
PERIPHERAL*i*	If several instances of a peripheral exist, the peripheral name is followed by a number to indicate the number of the instance in the range 0-n. PERIPHERAL0 denotes one specific instance.
Reset	Value of a register after a power Reset. This is also the value of registers in a peripheral after performing a software Reset of the peripheral, except for the Debug Control registers.
SET/CLR	Registers with SET/CLR suffix allows the user to clear and set bits in a register without doing a read-modify-write operation. These registers always come in pairs. Writing a ‘1’ to a bit in the CLR register will clear the corresponding bit in both registers, while writing a ‘1’ to a bit in the SET register will set the corresponding bit in both registers. Both registers will return the same value when read. If both registers are written simultaneously, the write to the CLR register will take precedence.