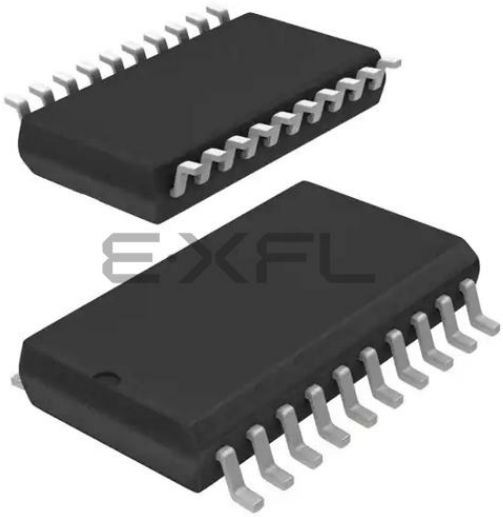


Welcome to E-XFL.COM

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	16MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	18
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 20x10b; D/A 3x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny3216-sfr

8. AVR CPU

8.1 Features

- 8-Bit, High-Performance AVR RISC CPU:
  - 135 instructions
  - Hardware multiplier
- 32 8-Bit Registers Directly Connected to the Arithmetic Logic Unit (ALU)
- Stack in RAM
- Stack Pointer Accessible in I/O Memory Space
- Direct Addressing of up to 64 KB of Unified Memory:
  - Entire Flash accessible with all `LD`/`ST` instructions
- True 16/24-Bit Access to 16/24-Bit I/O Registers
- Efficient Support for 8-, 16-, and 32-Bit Arithmetic
- Configuration Change Protection for System Critical Features

8.2 Overview

All AVR devices use the 8-bit AVR CPU. The CPU is able to access memories, perform calculations, control peripherals, and execute instructions in the program memory. Interrupt handling is described in a separate section.

Related Links

6. Memories
9. NVMCTRL - Nonvolatile Memory Controller
13. CPUINT - CPU Interrupt Controller

8.3 Architecture

In order to maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate buses for program and data. Instructions in the program memory are executed with single-level pipelining. While one instruction is being executed, the next instruction is prefetched from the program memory. This enables instructions to be executed on every clock cycle.

15.2 Register Summary - PORTMUX

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0			LUT1	LUT0		EVOUT2	EVOUT1	EVOUT0
0x01	CTRLB	7:0						SPI0		USART0
0x02	CTRLC	7:0			TCA05	TCA04	TCA03	TCA02	TCA01	TCA00
0x03	CTRLD	7:0							TCB1	TCB0

15.3 Register Description

Changes of the signal on a pin can trigger an interrupt. The exact conditions are defined by writing to the Input/Sense bit field (ISC) in PORT.PINnCTRL.

When setting or changing interrupt settings, take these points into account:

- If an INVEN bit is toggled in the same cycle as the interrupt setting, the edge caused by the inversion toggling may not cause an interrupt request.
- If an input is disabled while synchronizing an interrupt, that interrupt may be requested on re-enabling the input, even if it is re-enabled with a different interrupt setting.
- If the interrupt setting is changed while synchronizing an interrupt, that interrupt may not be accepted.
- Only a few pins support full asynchronous interrupt detection, see I/O Multiplexing and Considerations. These limitations apply for waking the system from sleep:

Interrupt Type	Fully Asynchronous Pins	Other Pins
BOTHEDGES	Will wake the system	Will wake the system
RISING	Will wake the system	Will not wake the system
FALLING	Will wake the system	Will not wake the system
LEVEL	Will wake the system	Will wake the system

Related Links

5. I/O Multiplexing and Considerations

16.3.3 Interrupts

Table 16-3. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	PORTx	PORT A, B, C interrupt	INTn in PORT.INTFLAGS is raised as configured by ISC bit in PORT.PINnCTRL.

Each port pin n can be configured as an interrupt source. Each interrupt can be individually enabled or disabled by writing to ISC in PORT.PINCTRL.

When an interrupt condition occurs, the corresponding interrupt flag is set in the Interrupt Flags register of the peripheral (*peripheral*.INTFLAGS).

An interrupt request is generated when the corresponding interrupt is enabled and the interrupt flag is set. The interrupt request remains active until the interrupt flag is cleared. See the peripheral's INTFLAGS register for details on how to clear interrupt flags.

20.5.10 Interrupt Control Register - Normal Mode

Name: INTCTRL
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CMP2	CMP1	CMP0				OVF
Access		R/W	R/W	R/W				R/W
Reset		0	0	0				0

Bit 6 – CMP2 Compare Channel 2 Interrupt Enable
See CMP0.

Bit 5 – CMP1 Compare Channel 1 Interrupt Enable
See CMP0.

Bit 4 – CMP0 Compare Channel 0 Interrupt Enable
Writing the CMPn bits to '1' enable compare interrupt from channel n.

Bit 0 – OVF Timer Overflow/Underflow Interrupt Enable
Writing the OVF bit to '1' enables overflow interrupt.

20.7.3 Control C - Split Mode

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

Bit	7	6	5	4	3	2	1	0
		HCMP2OV	HCMP1OV	HCMP0OV		LCMP2OV	LCMP1OV	LCMP0OV
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – HCMP2OV High byte Compare 2 Output Value
See LCMP0OV.

Bit 5 – HCMP1OV High byte Compare 1 Output Value
See LCMP0OV.

Bit 4 – HCMP0OV High byte Compare 0 Output Value
See LCMP0OV.

Bit 2 – LCMP2OV Low byte Compare 2 Output Value
See LCMP0OV.

Bit 1 – LCMP1OV Low byte Compare 1 Output Value
See LCMP0OV.

Bit 0 – LCMP0OV Low byte Compare 0 Output Value
The LCMPnOV/HCMPn bits allow direct access to the waveform generator's output compare value when the timer/counter is not enabled. This is used to set or clear the WOn output value when the timer/counter is not running.

20.7.12 Low Byte Timer Period Register - Split Mode

Name: LPER
Offset: 0x26
Reset: 0x00
Property: -

The TCAn.LPER register contains the TOP value of low byte timer.

Bit	7	6	5	4	3	2	1	0
	LPER[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 7:0 – LPER[7:0] Period Value Low Byte Timer
These bits hold the TOP value of low byte timer.

23.11.3 Interrupt Control

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

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

Bit 1 – CMP Compare Match Interrupt Enable
Enable interrupt-on-compare match (i.e., when the Counter value (CNT) matches the Compare value (CMP)).

Bit 0 – OVF Overflow Interrupt Enable
Enable interrupt-on-counter overflow (i.e., when the Counter value (CNT) matched the Period value (PER) and wraps around to zero).

24.3.2.10 Multiprocessor Communication Mode

The Multiprocessor Communication mode (MCPM) effectively reduces the number of incoming frames that have to be handled by the receiver in a system with multiple microcontrollers communicating via the same serial bus. This mode is enabled by writing a '1' to the MCPM bit in the Control B register (USARTn.CTRLB). In this mode, a dedicated bit in the frames is used to indicate whether the frame is an address or data frame type.

If the receiver is set up to receive frames that contain five to eight data bits, the first Stop bit is used to indicate the frame type. If the receiver is set up for frames with nine data bits, the ninth bit is used to indicate frame type. When the frame type bit is one, the frame contains an address. When the frame type bit is zero, the frame is a data frame. If 5- to 8-bit character frames are used, the transmitter must be set to use two Stop bits, since the first Stop bit is used for indicating the frame type.

If a particular slave MCU has been addressed, it will receive the following data frames as usual, while the other slave MCUs will ignore the frames until another address frame is received.

24.3.2.10.1 Using Multiprocessor Communication Mode

The following procedure should be used to exchange data in Multiprocessor Communication mode (MPCM):

1. All slave MCUs are in Multiprocessor Communication mode.
2. The master MCU sends an address frame, and all slaves receive and read this frame.
3. Each slave MCU determines if it has been selected.
4. The addressed MCU will disable MPCM and receive all data frames. The other slave MCUs will ignore the data frames.
5. When the addressed MCU has received the last data frame, it must enable MPCM again and wait for a new address frame from the master.

The process then repeats from step 2.

Using any of the 5- to 8-bit character frame formats is impractical, as the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver must use the same character size setting.

24.3.2.11 IRCOM Mode of Operation

The IRCOM mode enables IrDA® 1.4 compliant modulation and demodulation for baud rates up to 115.2 kbps. When IRCOM mode is enabled, Double-Speed mode cannot be used for the USART.

24.3.2.11.1 Overview

A USART can be configured in infrared communication mode (IRCOM) that is IrDA compatible with baud rates up to 115.2 kbps. When enabled, the IRCOM mode enables infrared pulse encoding/decoding for the USART.

A USART is set in IRCOM mode by writing 0x2 to the CMODE bits in USARTn.CTRLC. The data on the TX/RX pins is the inverted value of the transmitted/received infrared pulse. It is also possible to select an event channel from the Event System as an input for the IRCOM receiver. This will disable the RX input from the USART pin.

For transmission, three pulse modulation schemes are available:

- 3/16 of the baud rate period
- Fixed programmable pulse time based on the peripheral clock frequency
- Pulse modulation disabled

24.5.5 USART Status Register

Name: STATUS
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIF	TXCIF	DREIF	RXSIF	ISFIF		BDF	WFB
Access	R	R/W	R	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit 7 – RXCIF USART Receive Complete Interrupt Flag
This flag is set to ‘1’ when there is unread data in the receive buffer and cleared when the receive buffer is empty (i.e. does not contain any unread data). When the receiver is disabled, the receive buffer will be flushed and consequently, the RXCIF will become '0'.

When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from RXDATA in order to clear the RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt.

Bit 6 – TXCIF USART Transmit Complete Interrupt Flag
This flag is set when the entire frame in the Transmit Shift register has been shifted out and there are no new data in the transmit buffer (TXDATA).

This flag is automatically cleared when the transmit complete interrupt vector is executed. The flag can also be cleared by writing a ‘1’ to its bit location.

Bit 5 – DREIF USART Data Register Empty Flag
The DREIF indicates if the transmit buffer (TXDATA) is ready to receive new data. The flag is set to ‘1’ when the transmit buffer is empty and is ‘0’ when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift register. DREIF is set after a Reset to indicate that the transmitter is ready. Always write this bit to ‘0’ when writing the STATUS register.

DREIF is cleared to ‘0’ by writing TXDATAL. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to TXDATA in order to clear DREIF or disable the Data Register Empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.

Bit 4 – RXSIF USART Receive Start Interrupt Flag
The RXSIF flag indicates a valid Start condition on RxD line. The flag is set when the system is in standby modes and a high (IDLE) to low (START) valid transition is detected on the RxD line. If the start detection is not enabled, the RXSIF will always be read as '0'. This flag can only be cleared by writing a ‘1’ to its bit location. This flag is not used in the Master SPI mode operation.

Bit 3 – ISFIF Inconsistent Sync Field Interrupt Flag
This bit is set when the auto-baud is enabled and the sync field bit time is too fast or too slow to give a valid baud setting. It will also be set when USART is set to LINAUTO mode and the SYNC character differ from data value 0x55.

Writing a ‘1’ to this bit will clear the flag and bring the USART back to Idle state.

The CRCSCAN can be enabled during the internal Reset initialization to ensure the Flash is OK before letting the CPU execute code. If the CRCSCAN fails during the internal Reset initialization, the CPU is not allowed to start normal code execution - the device remains in Reset state instead of executing code with unexpected behavior. The full source settings are available during the internal Reset initialization. See the Fuse description for more information.

If the CRCSCAN was enabled during the internal Reset initialization, the CRC Control A and B registers will reflect this when normal code execution is started:

- The ENABLE bit in CRCSCAN.CTRLA will be '1'
- The MODE bit field in CRCSCAN.CTRLB will be non-zero
- The SRC bit field in CRCSCAN.CTRLB will reflect the checked section(s).

The CRCSCAN can be enabled during Reset by configuring the CRCSRC fuse in FUSE.SYSCFG0.

Related Links

27.5.1 CTRLA

27.5.2 CTRLB

6.10 Configuration and User Fuses (FUSE)

12.3.2.2 Reset Time

27.3.2 Operation

The CRC is operating in Priority mode: the CRC peripheral has priority access to the Flash and will stall the CPU until completed.

In Priority mode, the CRC fetches a new word (16-bit) on every third main clock cycle, or when the CRC peripheral is configured to do a scan from start-up.

27.3.2.1 Checksum

The pre-calculated checksum must be present in the last location of the section to be checked. If the BOOT section should be checked, the 16-bit checksum must be saved in the last two bytes of the BOOT section, and similarly for APPLICATION and entire Flash. Table 27-2 shows explicitly how the checksum should be stored for the different sections. Also, see the CRCSCAN.CTRLB register description for how to configure which section to check and the device fuse description for how to configure the BOOTEND and APPEND fuses.

Table 27-2. How to Place the Pre-Calculated 16-Bit Checksum in Flash

Section to Check	CHECKSUM[15:8]	CHECKSUM[7:0]
BOOT	FUSE_BOOTEND*256-2	FUSE_BOOTEND*256-1
BOOT and APPLICATION	FUSE_APPEND*256-2	FUSE_APPEND*256-1
Full Flash	FLASHEND-1	FLASHEND

27.3.3 Interrupts

Table 27-3. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	NMI	Non-Maskable Interrupt	Generated on CRC failure

When the interrupt condition occurs, the OK flag in the Status register (CRCSCAN.STATUS) is cleared to '0'.

Value	Name	Description
0xA	USART0	USART XCK input source
0xB	SPI0	SPI SCK input source
0xC	AC1	AC1 OUT input source
0xD	TCB1	TCB 1 W0 input source
0xE	AC2	AC2 OUT input source
Other	-	Reserved

28.5.8 TRUTHn

Name: TRUTH
Offset: 0x08 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	TRUTH[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 – TRUTH[7:0] Truth Table
These bits define the value of truth logic as a function of inputs IN[2:0].

30.4 Register Summary - ADCn

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTBY					RESSEL	FREERUN	ENABLE
0x01	CTRLB	7:0						SAMPNUM[2:0]		
0x02	CTRLC	7:0		SAMPCAP	REFSEL[1:0]			PRESC[2:0]		
0x03	CTRLD	7:0	INITDLY[2:0]			ASDV	SAMPDLY[3:0]			
0x04	CTRLE	7:0						WINCM[2:0]		
0x05	SAMPCTRL	7:0				SAMPLEN[4:0]				
0x06	MUXPOS	7:0				MUXPOS[4:0]				
0x07	Reserved									
0x08	COMMAND	7:0								STCONV
0x09	EVCTRL	7:0								STARTEI
0x0A	INTCTRL	7:0							WCOMP	RESRDY
0x0B	INTFLAGS	7:0							WCOMP	RESRDY
0x0C	DBGCTRL	7:0								DBGRUN
0x0D	TEMP	7:0	TEMP[7:0]							
0x0E ... 0x0F	Reserved									
0x10	RES	7:0	RES[7:0]							
		15:8	RES[15:8]							
0x12	WINLT	7:0	WINLT[7:0]							
		15:8	WINLT[15:8]							
0x14	WINHT	7:0	WINHT[7:0]							
		15:8	WINHT[15:8]							
0x16	CALIB	7:0								DUTYCYC

30.5 Register Description

30.5.7 MUXPOS

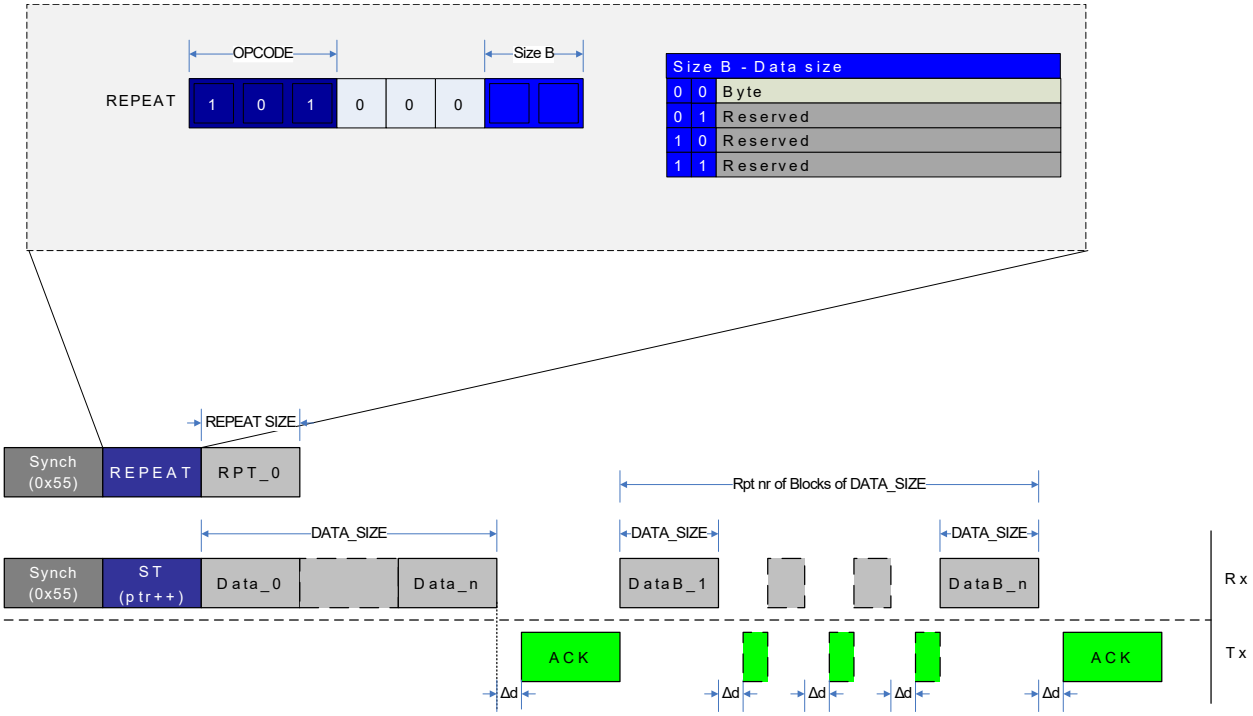
Name: MUXPOS
Offset: 0x06
Reset: 0x00
Property: -

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

Bits 4:0 – MUXPOS[4:0] MUXPOS
This bit field selects which single-ended analog input is connected to the ADC. If these bits are changed during a conversion, the change will not take effect until this conversion is complete.

Value	Name	Description
0x00	AIN0	ADC input pin 0
0x01	AIN1	ADC input pin 1
0x02	AIN2	ADC input pin 2
0x03	AIN3	ADC input pin 3
0x04	AIN4	ADC input pin 4
0x05	AIN5	ADC input pin 5
0x06	AIN6	ADC input pin 6
0x07	AIN7	ADC input pin 7
0x08	AIN8	ADC input pin 8
0x09	AIN9	ADC input pin 9
0x0A	AIN10	ADC input pin 10
0x0B	AIN11	ADC input pin 11
0x1B	PTC	ADC0: Reserved / ADC1: DAC2
0x1C	DAC0	DAC0
0x1D	INTREF	Internal reference (from VREF peripheral)
0x1F	GND	0V (GND)
Other	-	Reserved

Figure 33-15. `REPEAT` Instruction Operation



The figure above gives an example of repeat operation with an `ST` instruction using pointer post-increment operation. After the `REPEAT` instruction is sent with `RPT_0` = ***n***, the first `ST` instruction is issued with SYNCH and Instruction frame, while the next ***n*** `ST` instructions are executed by only sending in data bytes according to the `ST` operand `DATA_SIZE`, and maintaining the Acknowledge (ACK) handshake protocol.

If using indirect addressing instructions (`LD`/`ST`) it is recommended to always use the pointer post increment option when combined with `REPEAT`. Otherwise, the same address will be accessed in all repeated access operations. For direct addressing instructions (`LDS`/`STS`), the address must always be transmitted as specified in the instruction protocol, before data can be received (`LDS`) or sent (`STS`).

33.3.3.8 `KEY` - Set Activation KEY

The `KEY` instruction is used for communicating KEY bytes to the UPDI, opening up for executing protected features on the device. See Table 33-5 for an overview of functions that are activated by KEYs. For the `KEY` instruction, only 64-bit KEY size is supported. If the System Information Block (SIB) field of the `KEY` instruction is set, the `KEY` instruction returns the SIB instead of expecting incoming KEY bytes. Maximum supported size for SIB is 128 bits.

33.5.1 Status A

Name: STATUSA
Offset: 0x00
Reset: 0x10
Property: -

Bit	7	6	5	4	3	2	1	0
	UPDIREV[3:0]							
Access	R	R	R	R				
Reset	0	0	0	1				

Bits 7:4 – UPDIREV[3:0] UPDI Revision
These bits are read-only and contain the revision of the current UPDI implementation.

Table 37-9. Power Consumption of Peripherals

Peripheral	Conditions	Typ.(1)	Unit
BOD	Continuous	19	µA
	Sampling @ 1 kHz	1.2	
TCA	16-bit count @ 1 MHz	12.6	µA
TCB	16-bit count @ 1 MHz	7.4	µA
RTC	16-bit count	1.2	µA
WDT		0.7	µA
OSC20M		125	µA
AC	Fast mode(2)	92	µA
	Low-power mode(2)	45	µA
ADC	50 ksps	325	µA
	100 ksps	340	µA
XOSC32K	C_L=7.5 pF	0.5	µA
USART	Enable @ 9600 Baud	13	µA
SPI (Master)	Enable @ 100 kHz	2.1	µA
TWI (Master)	Enable @ 100 kHz	23.9	µA
TWI (Slave)	Enable @ 100 kHz	17.1	µA
Flash programming	Erase Operation	1.5	mA
	Write Operation	3.0	

Note:

1. Current consumption of the module only. To calculate the total power consumption of the system, add this value to the base power consumption as listed in *Power Consumption*.
2. CPU in Standby mode.

Related Links

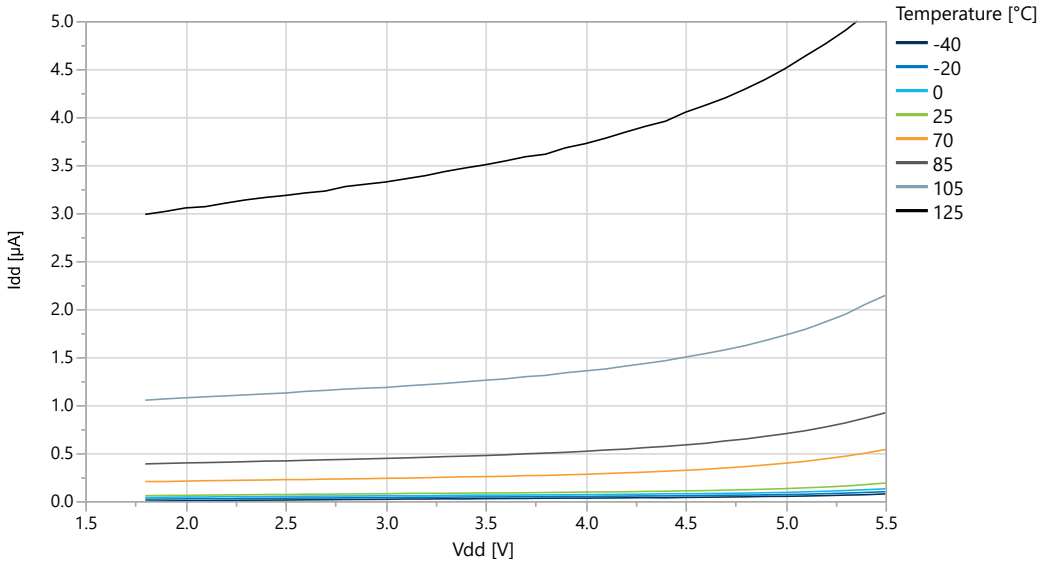
37.4 Power Consumption for ATtiny1616

37.8 BOD and POR Characteristics

Table 37-10. Power Supply Characteristics

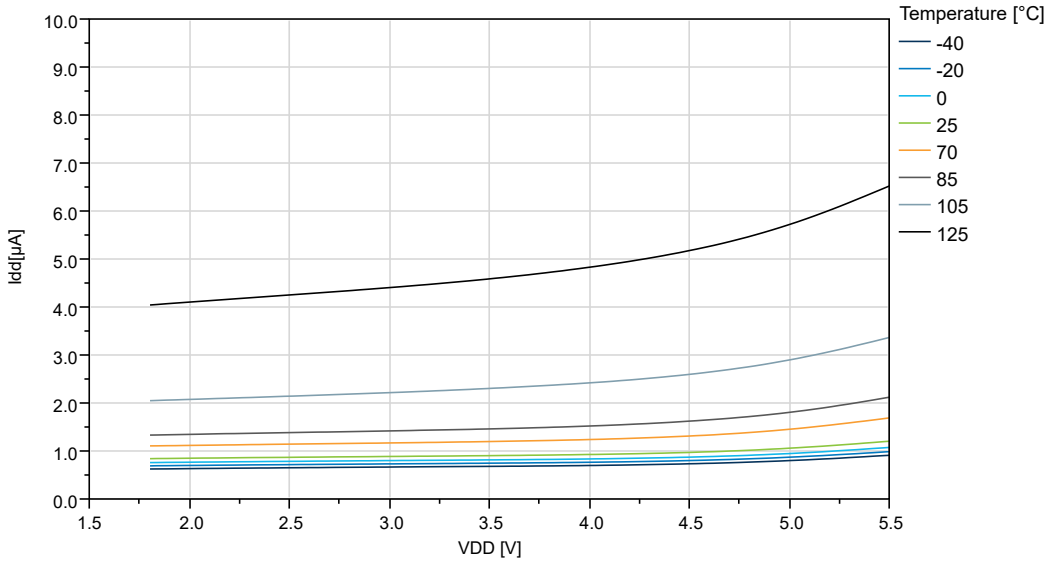
Symbol	Description	Condition	Min.	Typ.	Max.	Unit
SRON	Power-on Slope		-	-	100	V/ms

Figure 38-11. ATtiny1616 Power-Down Mode Supply Current vs. V_{DD} (all functions disabled)



38.1.4 Supply Currents in Standby Mode for ATtiny1616

Figure 38-12. ATtiny1616 Standby Mode Supply Current vs. V_{DD} (RTC Running with External 32 KHz Osc.)



38.4 BOD Characteristics

BOD Current vs. V_{DD}

Figure 38-49. BOD Current vs. V_{DD} (Continuous Mode Enabled)

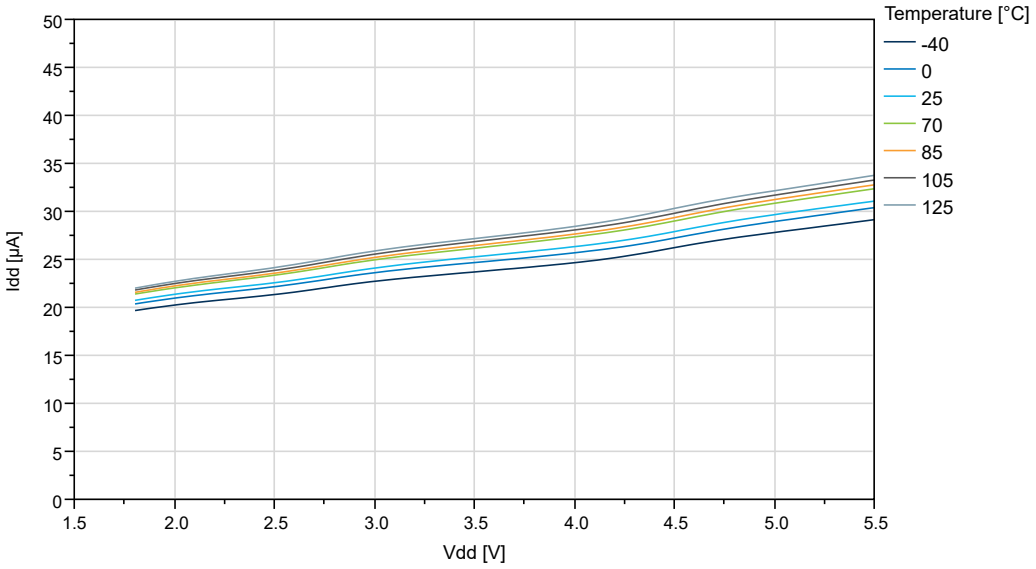
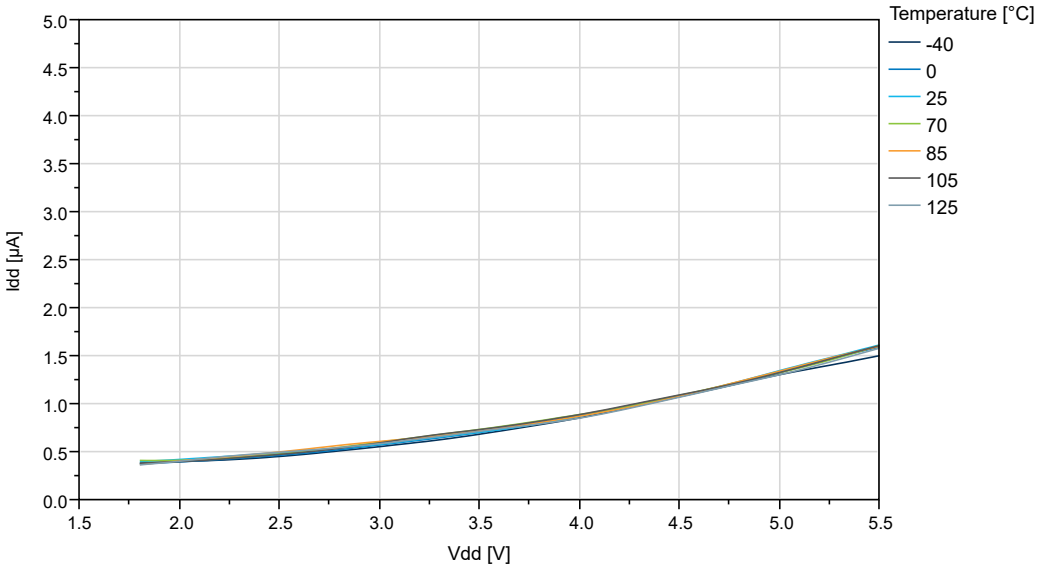
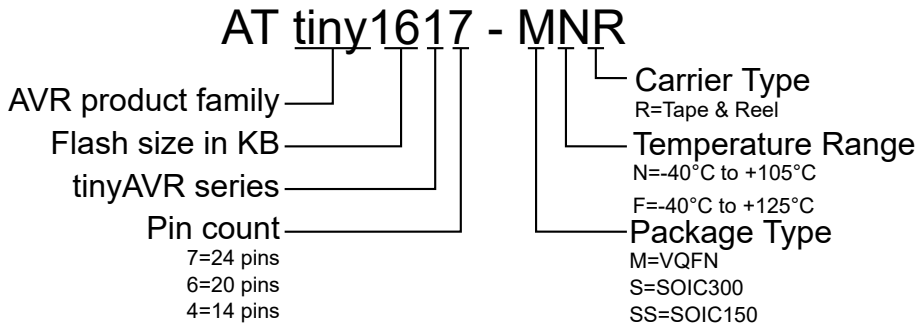


Figure 38-50. BOD Current vs. V_{DD} (Sampled BOD at 125 Hz)



Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.