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	EBI/EMI, I²C, SPI, UART/USART, USB OTG
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	48
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at90usb1287-aur

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly code example (1)

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEPE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from Data Register
  in   r16,EEDR
  ret
```

C code example (1)

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from Data Register */
  return EEDR;
}
```

Note: 1. See “About code examples” on page 10.

6.3.5 Preventing EEPROM corruption

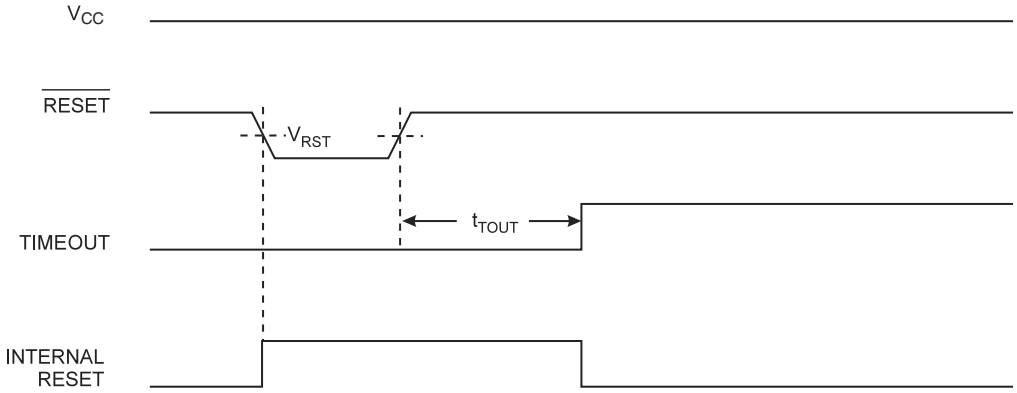
During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

Figure 9-4. External reset during operation.



9.5 Brown-out detection

Atmel AT90USB64/128 has an on-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

Table 9-2. BODLEVEL fuse coding.

BODLEVEL 2..0 Fuses	Min. V_{BOT}	Typ. V_{BOT}	Max. V_{BOT}	Units
111	BOD disabled			
110	Reserved			
101				
100				
011	2.4	2.6	2.8	V
010	3.2	3.4	3.6	
001	3.3	3.5	3.7	
000	4.1	4.3	4.5	

Table 9-3. Brown-out characteristics.

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{HYST}	Brown-out detector hysteresis		50		mV
t_{BOD}	Min. pulse width on brown-out reset				ns
I_{BOD}	Brown-out detector consumption		25		µA

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-5 on page 61), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-5 on page 61), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 9-1 on page 58.

Table 12-1. Interrupt sense control (1).

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

Note: 1. n = 3, 2, 1or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Table 12-2. Asynchronous external interrupt characteristics.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

12.0.2 EICRB – External Interrupt Control Register B

Bit	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – ISC71, ISC70 - ISC41, ISC40: External Interrupt 7 - 4 Sense Control Bits**

The External Interrupts 7 - 4 are activated by the external pins INT7:4 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in Table 12-3. The value on the INT7:4 pins are sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. Observe that CPU clock frequency can be lower than the XTAL frequency if the XTAL divider is enabled. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low.

Table 12-3. Interrupt sense control (1).

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request.
1	0	The falling edge between two samples of INTn generates an interrupt request.
1	1	The rising edge between two samples of INTn generates an interrupt request.

Note: 1. n = 7, 6, 5 or 4.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

12.0.3 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The double buffered Output Compare Registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B). See “Output compare unit” on page 100. for details. The Compare Match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

14.1.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, that is, TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in the table below are also used extensively throughout the document.

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

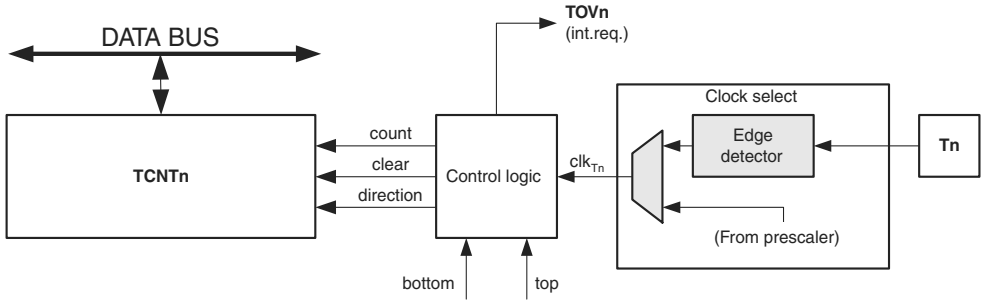
14.2 Timer/Counter clock sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler, see “Timer/Counter0, Timer/Counter1, and Timer/Counter3 prescalers” on page 96.

14.3 Counter unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 14-2 shows a block diagram of the counter and its surroundings.

Figure 14-2. Counter unit block diagram.



- **Bit 2 – OCF0B: Timer/Counter 0 Output Compare B Match Flag**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

- **Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 14-7, “Waveform Generation Mode bit description.” on page 111.

The following code examples show how to do an atomic read of the TCNTn Register contents. Reading any of the OCRnA/B/C or ICRn Registers can be done by using the same principle.

Assembly code example [1]

```
TIM16_ReadTCNTn:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNTn into r17:r16
  in  r16,TCNTnL
  in  r17,TCNTnH
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C code example [1]

```
unsigned int TIM16_ReadTCNTn( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Read TCNTn into i */
  i = TCNTn;
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Note: 1. See “About code examples” on page 10.

The assembly code example returns the TCNTn value in the r17:r16 register pair.

When the OCnA, OCnB or OCnC is connected to the pin, the function of the COMnx1:0 bits is dependent of the WGMn3:0 bits setting. Table 15-1 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to a normal or a CTC mode (non-PWM).

Table 15-1. Compare Output mode, non-PWM.

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	Toggle OCnA/OCnB/OCnC on compare match.
1	0	Clear OCnA/OCnB/OCnC on compare match (set output to low level).
1	1	Set OCnA/OCnB/OCnC on compare match (set output to high level).

Table 15-2 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the fast PWM mode.

Table 15-2. Compare Output mode, fast PWM.

COMnA1/COMnB1/ COMnC0	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at TOP
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at TOP

Note: A special case occurs when OCRnA/OCRnB/OCRnC equals TOP and COMnA1/COMnB1/COMnC1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See “Fast PWM mode” on page 104. for more details.

Table 15-3 on page 138 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct and frequency correct PWM mode.

Table 15-3. Compare Output mode, phase correct and phase and frequency correct PWM.

COMnA1/COMnB/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGM13:0 = 8, 9 10 or 11: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1	0	Clear OCnA/OCnB/OCnC on compare match when up-counting. Set OCnA/OCnB/OCnC on compare match when counting down.
1	1	Set OCnA/OCnB/OCnC on compare match when up-counting. Clear OCnA/OCnB/OCnC on compare match when counting down.

Note: A special case occurs when OCRnA/OCRnB/OCRnC equals TOP and COMnA1/COMnB1//COMnC1 is set. See “Phase correct PWM mode” on page 106. for more details.

- **Bit 1:0 – WGMn1:0: Waveform Generation mode**

Combined with the WGMn3:2 bits found in the TCCRnB Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 15-4 on page 138. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See “Modes of operation” on page 103.).

Table 15-4. Waveform Generation mode bit description [1].

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter mode of operation	TOP	Update of OCRnx at	TOVn flag set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, phase and frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICRn	TOP	BOTTOM

Figure 16-9 shows the same timing data, but with the prescaler enabled.

Figure 16-9. Timer/Counter timing diagram, with prescaler ($f_{clk\_I/O}/8$).

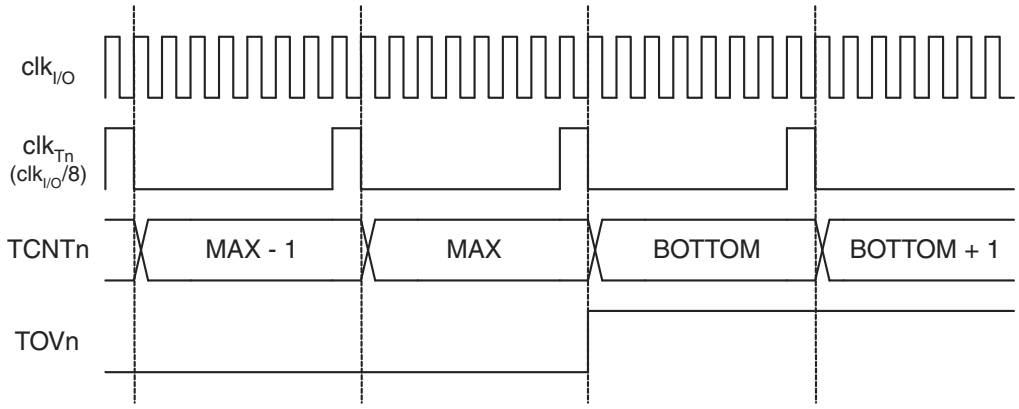


Figure 16-10 shows the setting of OCF2A in all modes except CTC mode.

Figure 16-10. Timer/Counter timing diagram, setting of OCF2A, with prescaler ($f_{clk\_I/O}/8$).

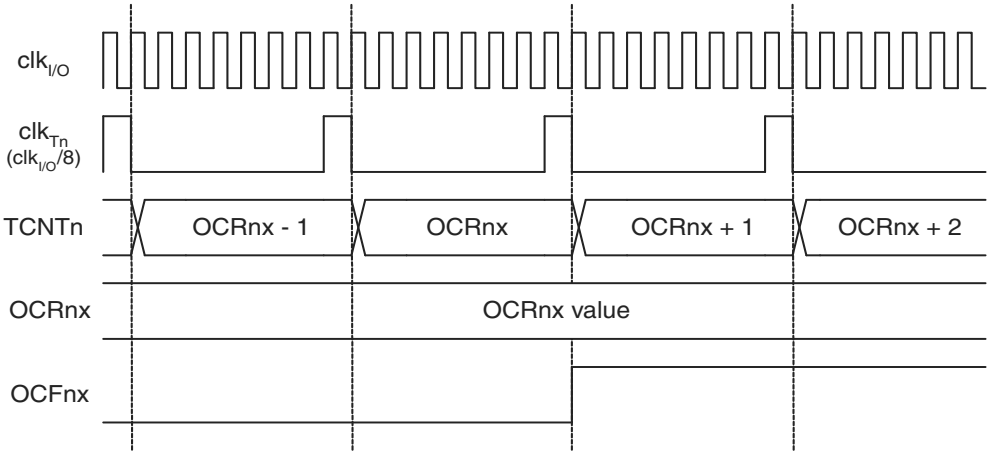


Figure 16-11 on page 156 shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 18-3. SPI transfer format with CPHA = 0.

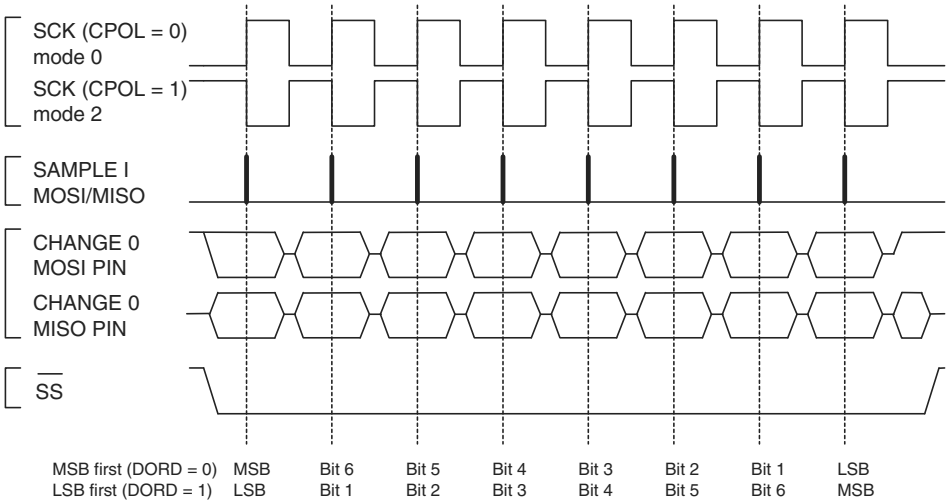
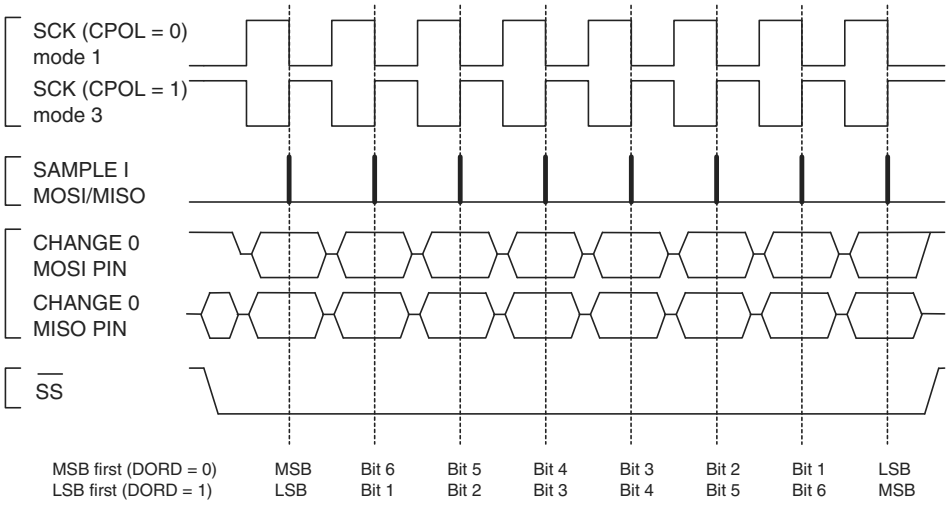


Figure 18-4. SPI transfer format with CPHA = 1.



mal USART operation. The MSPIM is enabled when both UMSELn bits are set to one. The UDORDn, UCPHAn, and UCPOLn can be set in the same write operation where the MSPIM is enabled.

Table 20-3. UMSELn bits settings.

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)

- **Bit 5:3 - Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnC is written.

- **Bit 2 - UDORDn: Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the Frame Formats section page 4 for details.

- **Bit 1 - UCPHAn: Clock Phase**

The UCPHAn bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCKn. Refer to the SPI Data Modes and Timing section page 4 for details.

- **Bit 0 - UCPOLn: Clock Polarity**

The UCPOLn bit sets the polarity of the XCKn clock. The combination of the UCPOLn and UCPHAn bit settings determine the timing of the data transfer. Refer to the SPI Data Modes and Timing section page 4 for details.

20.6.5 UBRRnL and UBRRnH – USART MSPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode is identical to normal USART operation. See “UBRRLn and UBRRHn – USART baud rate registers” on page 197.

20.7 AVR USART MSPIM vs. AVR SPI

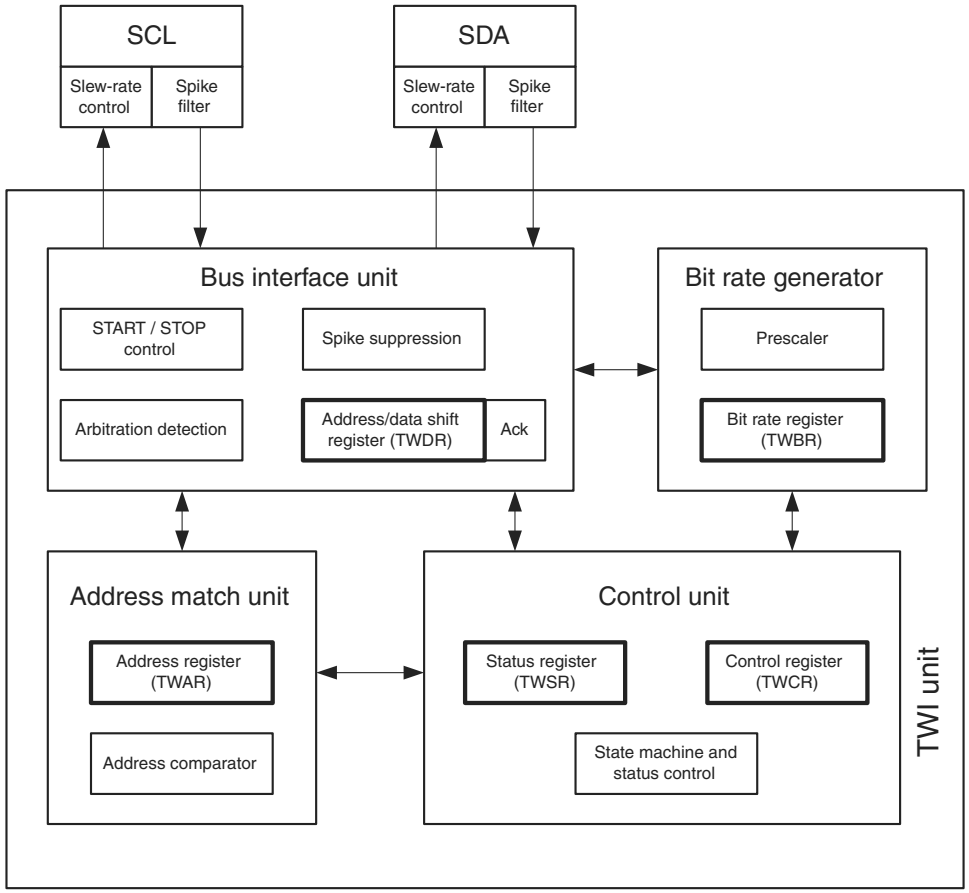
The USART in MSPIM mode is fully compatible with the AVR SPI regarding:

- Master mode timing diagram.
- The UCPOLn bit functionality is identical to the SPI CPOL bit
- The UCPHAn bit functionality is identical to the SPI CPHA bit
- The UDORDn bit functionality is identical to the SPI DORD bit

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer
- The USART in MSPIM mode receiver includes an additional buffer level
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode

Figure 21-9. Overview of the TWI module.



21.5.1 SCL and SDA pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

21.5.2 Bit Rate Generator unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

- **Bit 5 – TWSTA: TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a Master on the 2-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit and will always read as zero.

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

21.6.3 TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

- **3-1 – Reserved**

The value read from these bits is always 0. Do not set these bits.

- **0 – UVREGE: USB pad regulator Enable**

Set to enable the USB pad regulator. Clear to disable the USB pad regulator.

Bit	7	6	5	4	3	2	1	0	
	USBE	HOST	FRZCLK	OTGPADE	-	-	IDTE	VBUSTE	USBCON
Read/write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	1	0	0	0	0	0	

- **7 – USBE: USB macro Enable bit**

Set to enable the USB controller. Clear to disable and reset the USB controller, to disable the USB transceiver and to disable the USB controller clock inputs.

- **6 – HOST: HOST bit**

Set to enable the Host mode. Clear to enable the device mode.

- **5 – FRZCLK: Freeze USB Clock bit**

Set to disable the clock inputs (the ”Resume Detection” is still active). This reduces the power consumption. Clear to enable the clock inputs.

- **4 – OTGPADE: OTG Pad Enable**

Set to enable the OTG pad. Clear to disable the OTG pad. The OTG pad is actually the VBUS pad.

Note that this bit can be set/cleared even if USBE=0. That allows the VBUS detection even if the USB macro is disabled. This pad must be enabled in both Host and Device modes in order to allow USB operation (attaching, transmitting...).

- **3-2 – Reserved**

The value read from these bits is always 0. Do not set these bits.

- **1 – IDTE: ID Transition Interrupt Enable bit**

Set this bit to enable the ID Transition interrupt generation. Clear this bit to disable the ID Transition interrupt generation.

- **0 – VBUSTE: VBUS Transition Interrupt Enable bit**

Set this bit to enable the VBUS Transition interrupt generation.
Clear this bit to disable the VBUS Transition interrupt generation.

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	SPEED		ID	VBUS	USBSTA
Read/write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	1	0	1	0	

- **7-4 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- **4 - NAKOUTI - NAK OUT Received Interrupt flag**

Set by hardware when a NAK handshake has been sent in response of a OUT/PING request from the host. This triggers an USB interrupt if NAKOUTE is sent.

Shall be cleared by software. Setting by software has no effect.

- **3 - RXSTPI - Received SETUP Interrupt flag**

Set by hardware to signal that the current bank contains a new **valid SETUP packet**. An interrupt (EPINTx) is triggered (if enabled).

Shall be cleared by software to handshake the interrupt. Setting by software has no effect.

This bit is inactive (cleared) if the endpoint is an IN endpoint.

- **2 - RXOUTI / KILLBK - Received OUT Data Interrupt flag**

Set by hardware to signal that the current bank contains a new packet. An interrupt (EPINTx) is triggered (if enabled).

Shall be cleared by software to handshake the interrupt. Setting by software has no effect.

Kill Bank IN bit

Set this bit to kill the last written bank.

Cleared by hardware when the bank is killed. Clearing by software has no effect.

See page 271 for more details on the Abort.

- **1 - STALLEDI - STALLEDI Interrupt flag**

Set by hardware to signal that a STALL handshake has been sent, or that a CRC error has been detected in a OUT isochronous endpoint.

Shall be cleared by software. Setting by software has no effect.

- **0 - TXINI - Transmitter Ready Interrupt flag**

Set by hardware to signal that the current bank is free and can be filled. An interrupt (EPINTx) is triggered (if enabled).

Shall be cleared by software to handshake the interrupt. Setting by software has no effect.

This bit is inactive (cleared) if the endpoint is an OUT endpoint.

Bit	7	6	5	4	3	2	1	0	
	FLERRE	NAKINE	-	NAKOUTE	RXSTPE	RXOUTE	STALLEDE	TXINE	UEIENX
Read/write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **7 - FLERRE - Flow Error Interrupt Enable flag**

Set to enable an endpoint interrupt (EPINTx) when OVERFI or UNDERFI are sent.

Clear to disable an endpoint interrupt (EPINTx) when OVERFI or UNDERFI are sent.

- **6 - NAKINE - NAK IN Interrupt Enable bit**

Set to enable an endpoint interrupt (EPINTx) when NAKINI is set.

Clear to disable an endpoint interrupt (EPINTx) when NAKINI is set.

Bit	7	6	5	4	3	2	1	0	
	-	HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI	UHINT
Read/write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **7 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- **6 - HWUPI - Host Wake-Up Interrupt**

Set by hardware when a non-idle state is detected on the USB bus.This interrupt should be enable only to wake up the CPU core from power down mode.
Shall be clear by software to acknowledge the interrupt. Setting by software has no effect.

- **5 - HSOFI - Host Start Of Frame Interrupt**

Set by hardware when a SOF is issued by the Host controller. This triggers a USB interrupt when HSOFE is set. When using the host controller in low speed mode, this bit is also set when a keep-alive is sent.

Shall be cleared by software to acknowledge the interrupt. Setting by software has no effect.

- **4 - RXRSMI - Upstream Resume Received Interrupt**

Set by hardware when an Upstream Resume has been received from the Device.

Shall be cleared by software. Setting by software has no effect.

- **3 - RSMEDI - Downstream Resume Sent Interrupt**

Set by hardware when a Downstream Resume has been sent to the Device.

Shall be cleared by software. Setting by software has no effect.

- **2 - RSTI - USB Reset Sent Interrupt**

Set by hardware when a USB Reset has been sent to the Device.
Shall be cleared by software. Setting by software has no effect.

- **1 - DDISCI - Device Disconnection Interrupt**

Set by hardware when the device has been removed from the USB bus.

Shall be cleared by software. Setting by software has no effect.

- **0 - DCONNI - Device Connection Interrupt**

Set by hardware when a new device has been connected to the USB bus.
Shall be cleared by software. Setting by software has no effect.

Bit	7	6	5	4	3	2	1	0	
		HWUPE	HSOFE	RXRSME	RSMEDE	RSTE	DDISCE	DCONNE	UHIEN
Read/write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **7 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register – Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in “Bibliography” on page 332.

27.4 Using the Boundary-scan chain

A complete description of the Boundary-scan capabilities are given in the section “IEEE 1149.1 (JTAG) boundary-scan” on page 333.

27.5 Using the on-chip debug system

As shown in Figure 27-1 on page 328, the hardware support for on-chip debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- Break Point unit
- Communication interface between the CPU and JTAG system

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Break Point Unit implements Break on Change of Program Flow, Single Step Break, two Program Memory Break Points, and two combined Break Points. Together, the four Break Points can be configured as either:

- Four single program memory break points
- Three single program memory break point + one single data memory break point
- Two single program memory break points + two single data memory break points
- Two single program memory break points + one program memory break point with mask (“range Break Point”)

28. IEEE 1149.1 (JTAG) boundary-scan

28.1 Features

- **JTAG (IEEE std. 1149.1 compliant) interface**
- **Boundary-scan capabilities according to the JTAG standard**
- **Full scan of all port functions as well as analog circuitry having off-chip connections**
- **Supports the optional IDCODE instruction**
- **Additional public AVR_RESET instruction to reset the AVR**

28.2 System overview

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long Shift Register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the four TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the Data Register path will show the ID-Code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. Entering reset, the outputs of any port pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external $\overline{\text{RESET}}$ pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN Fuse must be programmed and the JTD bit in the I/O Register MCUCR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

28.3 Data registers

The Data Registers relevant for Boundary-scan operations are:

- Bypass Register
- Device Identification Register
- Reset Register
- Boundary-scan Chain

Note: 1. The timing requirements shown in Figure 30-7 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 30-13. Parallel programming characteristics, V_{CC} = 5V ±10%.

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			
t_{XHXL}	XTAL1 Pulse Width High	150			
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			
t_{XLPH}	XTAL1 Low to PAGEL high	0			
t_{PLXH}	PAGEL low to XTAL1 high	150			
t_{BVPH}	BS1 Valid before PAGEL High	67			
t_{PHPL}	PAGEL Pulse Width High	150			
t_{PLBX}	BS1 Hold after PAGEL Low	67			
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			
t_{BVWL}	BS2/1 Valid to $\overline{WR}$ Low	67			
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High (1)	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase (2)	7.5		9	
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

30.7 Serial downloading

Both the Flash and EEPROM memory arrays can be programmed using a serial programming bus while $\overline{RESET}$ is pulled to GND. The serial programming interface consists of pins SCK, PDI (input) and PDO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 30-14 on page 374, the pin mapping for serial programming is listed. Not all packages use the SPI pins dedicated for the internal Serial Peripheral Interface - SPI.

Table 30-17. JTAG programming instruction set.

a = address high bits, **b** = address low bits, **c** = address extended bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care.

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip Erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for Chip Erase Complete	0110011_10000000	xxxxx**o**x_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
2c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
2d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
2e. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2f. Load Data High Byte	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2g. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2i. Poll for Page Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
3c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
3d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
3e. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	 Low byte High byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(10)
4c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)