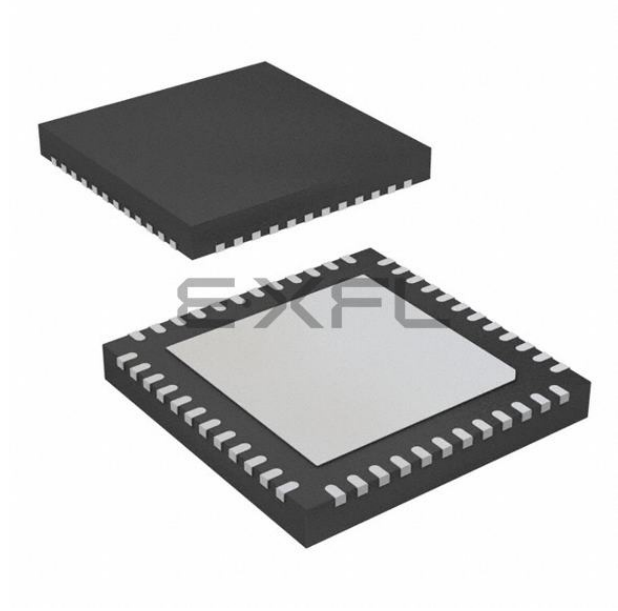


Welcome to E-XFL.COM

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.

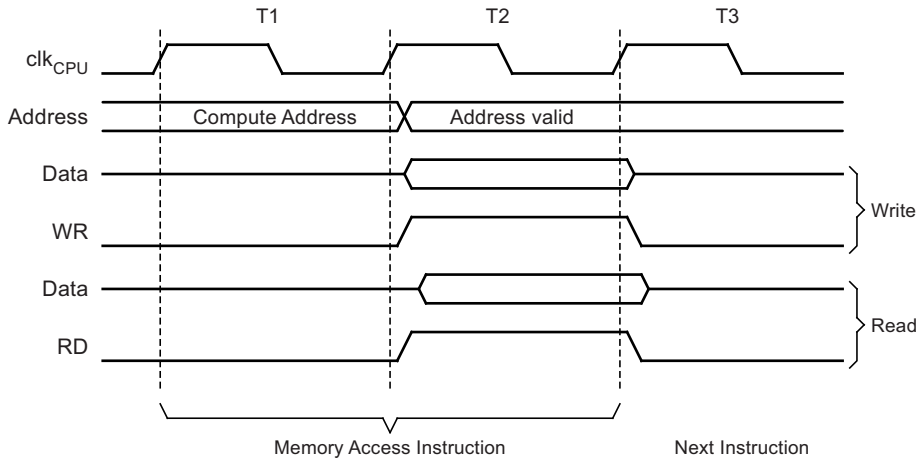
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TC)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/ata6612c-plqw

5.5.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 5-10.

Figure 5-10. On-chip Data SRAM Access Cycles



5.5.3 EEPROM Data Memory

The Atmel® ATA6612C/ATA6613C contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register.

The section Section 5.24 “Memory Programming” on page 253 contains a detailed description on EEPROM programming in SPI or parallel programming mode.

5.5.3.1 EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 5-3 on page 43. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See Section 5.5.3.5 “Preventing EEPROM Corruption” on page 44 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM control register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

5.5.3.2 The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	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	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATA6612C/ATA6613C and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM address registers – EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255/511/511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

5.5.3.3 The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

5.5.3.4 The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bits 7..6 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATA6612C/ATA6613C and will always read as zero

- **Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits**

The EEPROM programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the erase and write operations in two different operations.

The programming times for the different modes are shown in Table 5-2 on page 42. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 5-2. EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4ms	Erase and write in one operation (atomic operation)
0	1	1.8ms	Erase only
1	0	1.8ms	Write only
1	1	–	Reserved for future use

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM ready interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM ready interrupt generates a constant interrupt when EEPE is cleared.

- **Bit 2 – EEMPE: EEPROM Master Write Enable**

  The EEMPE bit determines whether setting EEPE to one causes the EEPROM to be written. When EEMPE is set, setting EEPE within four clock cycles will write data to the EEPROM at the selected address If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

- **Bit 1 – EEPE: EEPROM Write Enable**

  The EEPROM write enable signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEPE becomes zero.
2. Wait until SELFPRGEN in SPMCSR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMPE bit while writing a zero to EEPE in EECR.
6. Within four clock cycles after setting EEMPE, write a logical one to EEPE.

The EEPROM can not be programmed during a CPU write to the flash memory. The software must check that the flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a boot loader allowing the CPU to program the flash. If the flash is never being updated by the CPU, step 2 can be omitted. See Section 5.23 “Boot Loader Support – Read-While-Write Self-Programming, Atmel ATA6612C and ATA6613C” on page 240 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM master write enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM read enable signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

5.10.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 5-23 on page 78, the digital input signal can be clamped to ground at the input of the Schmitt trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 5.10.3 Alternate Port Functions on page 81.

If a logic high level (one) is present on an asynchronous external interrupt pin configured as interrupt on rising edge, falling edge, or any logic change on pin while the external interrupt is not enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned sleep mode, as the clamping in these sleep mode produces the requested logic change.

5.10.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (reset, active mode and idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

5.10.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 5-26 on page 82 shows how the port pin control signals from the simplified Figure 5-23 on page 78 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR® microcontroller family.

5.15.8 8-bit Timer/Counter Register Description

5.15.8.1 Timer/Counter Control Register A – TCCR2A

Bit	7	6	5	4	3	2	1	0	
	COM2A**1**	COM2A**0**	COM2B**1**	COM2B**0**	–	–	**WGM21**	**WGM20**	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM2A1:0: Compare Match Output A Mode**

These bits control the output compare pin (OC2A) behavior. If one or both of the COM2A1:0 bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC2A pin must be set in order to enable the output driver.

When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM22:0 bit setting. Table 5-59 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 5-59. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC2A on compare match
1	0	Clear OC2A on compare match
1	1	Set OC2A on compare match

Table 5-60 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 5-60. Compare Output Mode, Fast PWM Mode(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC0A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match, set OC2A at TOP
1	1	Set OC2A on compare match, clear OC2A at TOP

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 5.15.6.3 “Fast PWM Mode” on page 144 for more details.

Table 5-61 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 5-61. Compare Output Mode, Phase Correct PWM Mode()

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC2A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match when up-counting. Set OC2A on compare match when down-counting.
1	1	Set OC2A on compare match when up-counting. Clear OC2A on compare match when down-counting.

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 5.15.6.4 “Phase Correct PWM Mode” on page 145 for more details.

- **Bits 5:4 – COM2B1:0: Compare Match Output B Mode**

These bits control the output compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver.

When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. Table 5-62 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 5-62. Compare Output Mode, non-PWM Mode

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on compare match
1	0	Clear OC2B on compare match
1	1	Set OC2B on compare match

Table 5-63 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to fast PWM mode.

Table 5-63. Compare Output Mode, Fast PWM Mode(1)

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match, set OC2B at TOP
1	1	Set OC2B on compare match, clear OC2B at TOP

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 5.15.6.4 “Phase Correct PWM Mode” on page 145 for more details.

Table 5-64 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 5-64. Compare Output Mode, Phase Correct PWM Mode(1)

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match when up-counting. Set OC2B on compare match when down-counting.
1	1	Set OC2B on compare match when up-counting. Clear OC2B on compare match when down-counting.

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 5.15.6.4 “Phase Correct PWM Mode” on page 145 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATA6612C/ATA6613C and will always read as zero.

5.17.5.2 Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZn = 7), the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example[1][2]

```
USART_Transmit:
        ; Wait for empty transmit buffer
        sbis    UCSRnA,UDREn
        rjmp    USART_Transmit
        ; Copy 9th bit from r17 to TXB8
        cbi     UCSRnB,TXB8
        sbrc    r17,0
        sbi     UCSRnB,TXB8
        ; Put LSB data (r16) into buffer, sends the data
        out     UDRn,r16
        ret
```

C Code Example[1][2]

```
void USART_Transmit (unsigned int data)
{
        /* Wait for empty transmit buffer */
        while (!(UCSRnA & (1<<UDREn)))
        ;
        /* Copy 9th bit to TXB8 */
        UCSRnB &= ~(1<<TXB8);
        if (data & 0x0100)
                UCSRnB |= (1<<TXB8);
        /* Put data into buffer, sends the data */
        UDRn = data;
}
```

Notes:
1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB register is used after initialization.
2. The example code assumes that the part specific header file is included.
   For I/O registers located in extended I/O map, IN , OUT , SBIS , SBIC , CBI , and SBI instructions must be replaced with instructions that allow access to extended I/O. Typically LDS and STS combined with SBRS , SBRC , SBR , and CBR .

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

5.17.5.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART data register empty (UDREn) and transmit complete (TXCn). Both flags can be used for generating interrupts.

The data register empty (UDREn) flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always write this bit to zero when writing the UCSRnA register.

When the data register empty interrupt enable (UDRIEn) bit in UCSRnB is written to one, the USART data register empty interrupt will be executed as long as UDREn is set (provided that global interrupts are enabled). UDREn is cleared by writing UDRn. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to UDRn in order to clear UDREn or disable the data register empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The transmit complete (TXCn) flag bit is set one when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXCn flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

Figure 5-105. ADC Timing Diagram, Auto Triggered Conversion

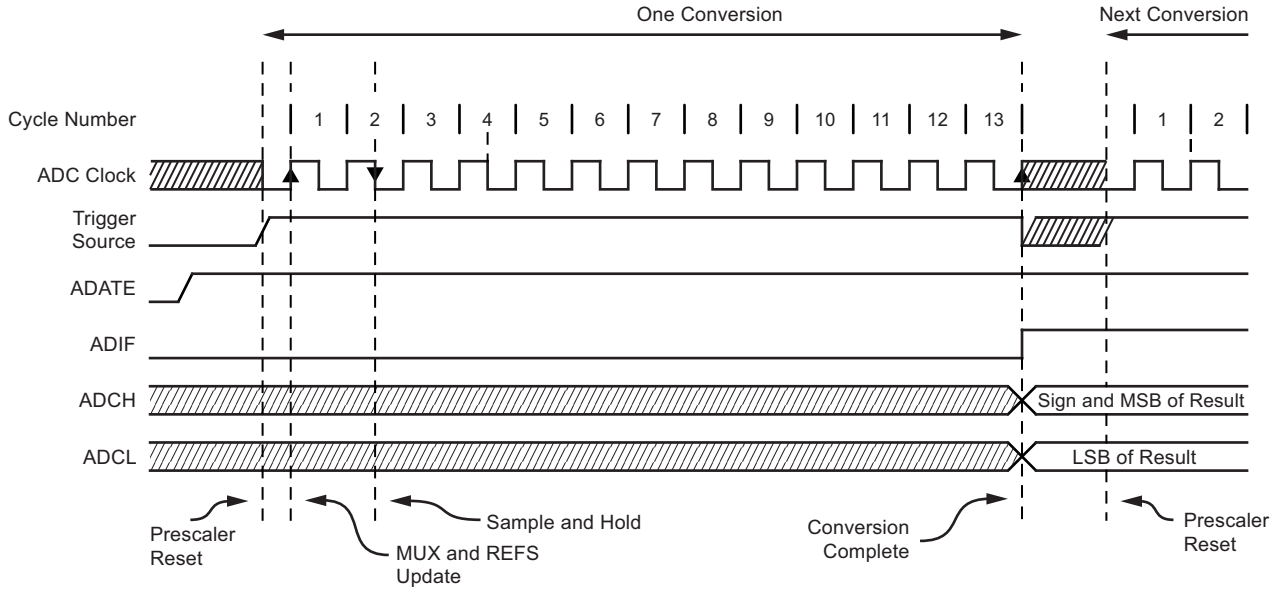


Figure 5-106. ADC Timing Diagram, Free Running Conversion

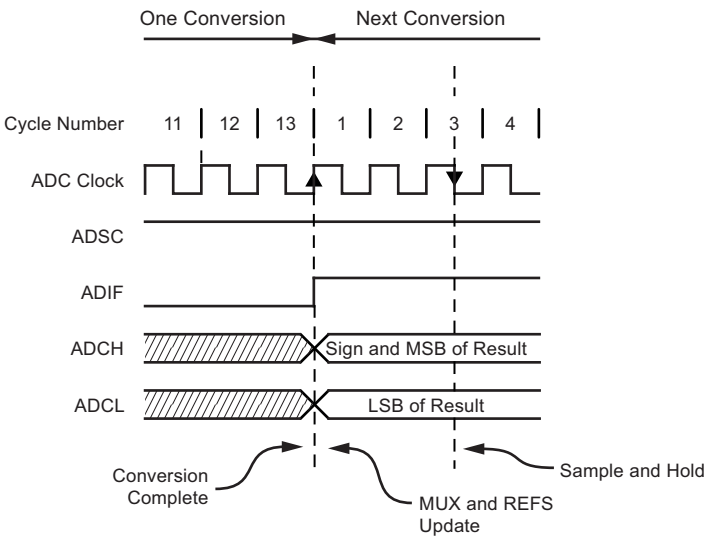
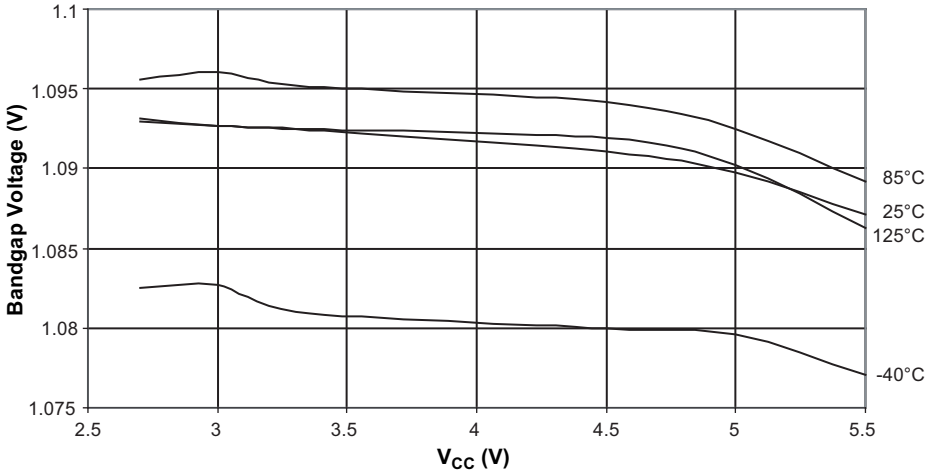


Table 5-98. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5

Figure 6-29. Bandgap Voltage versus V_{CC}



6.3.1.7 Peripheral Units

Figure 6-30. Analog to Digital Converter GAIN versus V_{CC}

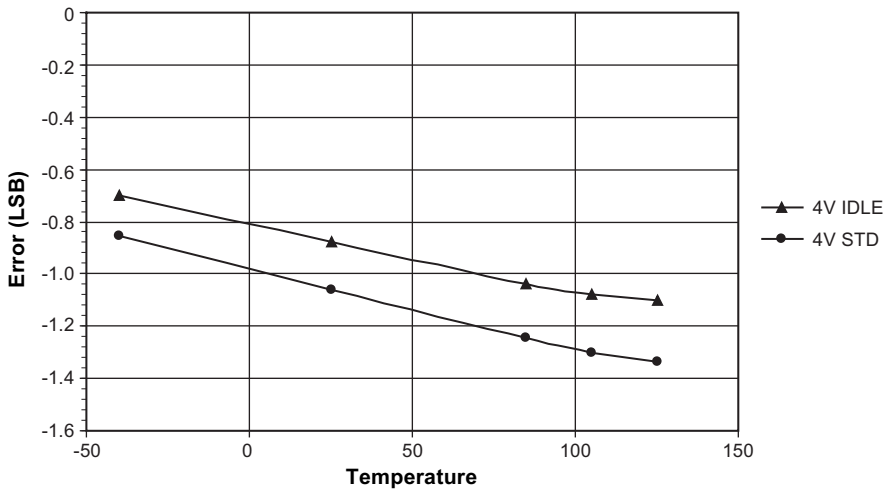


Figure 6-31. Analog to Digital Converter OFFSET versus V_{CC}