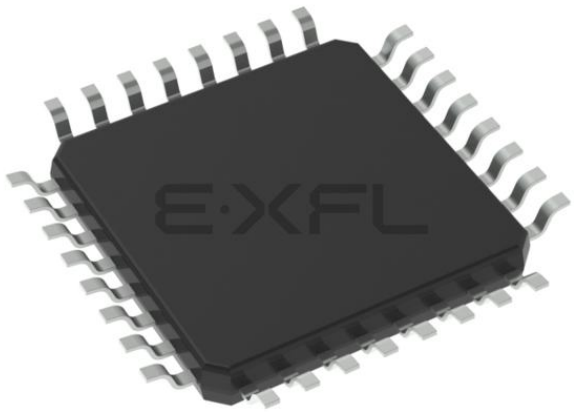


Welcome to E-XFL.COM

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	8/16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	26
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.6V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atxmega8e5-anr

1. Ordering Information

Ordering Code	Package[1][2][3]	Flash [Bytes]	EEPROM [Bytes]	SRAM [Bytes]	Speed [MHz]	Power supply [V]	Temp. [°C]
ATxmega8E5-AU	32A (7x7mm TQFP)	8K + 2K	512	1K	32	1.6 – 3.6	-40 – 85
ATxmega8E5-AUR[4]							
ATxmega8E5-MU	32Z (5x5mm VQFN)						
ATxmega8E5-MUR[4]							
ATxmega8E5-M4U	32MA (4x4mm UQFN)						
ATxmega8E5-M4UR[4]							
ATxmega16E5-AU	32A (7x7mm TQFP)	16K + 4K	512	2K	32	1.6 – 3.6	-40 – 85
ATxmega16E5-AUR[4]							
ATxmega16E5-MU	32Z (5x5mm VQFN)						
ATxmega16E5-MUR[4]							
ATxmega16E5-M4U	32MA (4x4mm UQFN)						
ATxmega16E5-M4UR[4]							
ATxmega32E5-AU	32A (7x7mm TQFP)	32K + 4K	1K	4K	32	1.6 – 3.6	-40 – 85
ATxmega32E5AUR[4]							
ATxmega32E5-MU	32Z (5x5mm VQFN)						
ATxmega32E5-MUR[4]							
ATxmega32E5-M4U	32MA (4x4mm UQFN)						
ATxmega32E5-M4UR[4]							
ATxmega8E5-AN	32A (7x7mm TQFP)	8K + 2K	512	1K	32	1.6 – 3.6	-40 – 105
ATxmega8E5-ANR[4]							
ATxmega8E5-MN	32Z (5x5mm VQFN)						
ATxmega8E5-MNR[4]							
ATxmega8E5-M4UN	32MA (4x4mm UQFN)						
ATxmega8E5-M4UNR[4]							
ATxmega16E5-AN	32A (7x7mm TQFP)	16K + 4K	512	2K	32	1.6 – 3.6	-40 – 105
ATxmega16E5-ANR[4]							
ATxmega16E5-MN	32Z (5x5mm VQFN)						
ATxmega16E5-MNR[4]							
ATxmega16E5-M4UN	32MA (4x4mm UQFN)						
ATxmega16E5-M4UNR[4]							

15. Interrupts and Programmable Multilevel Interrupt Controller

15.1 Features

- Short and predictable interrupt response time
- Separate interrupt configuration and vector address for each interrupt
- Programmable multilevel interrupt controller
  - Interrupt prioritizing according to level and vector address
  - Three selectable interrupt levels for all interrupts: low, medium, and high
  - Selectable, round-robin priority scheme within low-level interrupts
  - Non-maskable interrupts for critical functions
- Interrupt vectors optionally placed in the application section or the boot loader section

15.2 Overview

Interrupts signal a change of state in peripherals, and this can be used to alter program execution. Peripherals can have one or more interrupts, and all are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition is present. The programmable multilevel interrupt controller (PMIC) controls the handling and prioritizing of interrupt requests. When an interrupt request is acknowledged by the PMIC, the program counter is set to point to the interrupt vector, and the interrupt handler can be executed.

All peripherals can select between three different priority levels for their interrupts: low, medium, and high. Interrupts are prioritized according to their level and their interrupt vector address. Medium-level interrupts will interrupt low-level interrupt handlers. High-level interrupts will interrupt both medium- and low-level interrupt handlers. Within each level, the interrupt priority is decided from the interrupt vector address, where the lowest interrupt vector address has the highest interrupt priority. Low-level interrupts have an optional round-robin scheduling scheme to ensure that all interrupts are serviced within a certain amount of time.

Non-maskable interrupts (NMI) are also supported, and can be used for system critical functions.

15.3 Interrupt Vectors

The interrupt vector is the sum of the peripheral's base interrupt address and the offset address for specific interrupts in each peripheral. The base addresses for the Atmel AVR XMEGA E5 devices are shown in Table 15-1. Offset addresses for each interrupt available in the peripheral are described for each peripheral in the XMEGA AU manual. For peripherals or modules that have only one interrupt, the interrupt vector is shown in Table 15-1. The program address is the word address.

Table 15-1. Peripheral Module Address Map

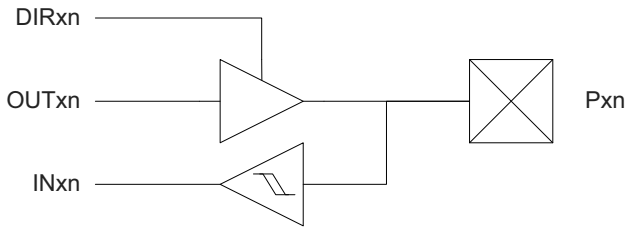
Program address (base address)	Source	Interrupt description
0x0000	RESET	
0x0002	OSCF_INT_vect	Crystal oscillator failure and PLL lock failure interrupt vector (NMI)
0x0004	PORTR_INT_vect	Port R Interrupt vector
0x0006	EDMA_INT_base	EDMA Controller Interrupt base
0x000E	RTC_INT_base	Real time counter interrupt base
0x0012	PORTC_INT_vect	Port C interrupt vector
0x0014	TWIC_INT_base	Two-wire interface on Port C interrupt base
0x0018	TCC4_INT_base	Timer/counter 4 on port C interrupt base

16.3 Output Driver

All port pins (Pxn) have programmable output configuration. The port pins also have configurable slew rate limitation to reduce electromagnetic emission.

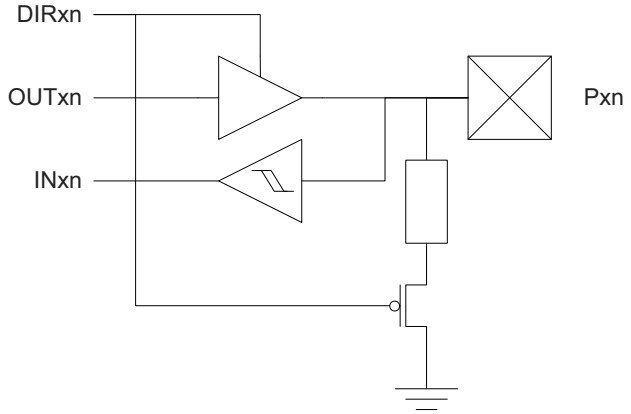
16.3.1 Push-pull

Figure 16-1. I/O Configuration - Totem-pole



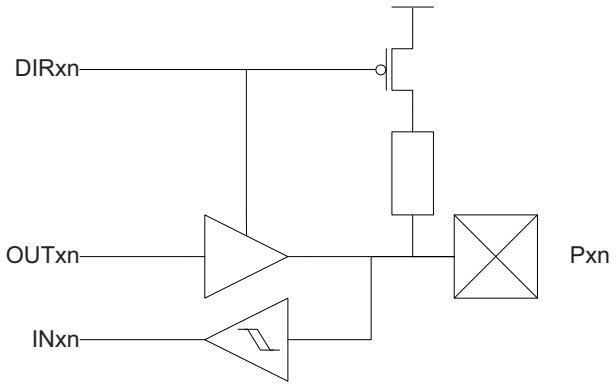
16.3.2 Pull-down

Figure 16-2. I/O Configuration - Totem-pole with Pull-down (on input)



16.3.3 Pull-up

Figure 16-3. I/O Configuration - Totem-pole with Pull-up (on input)



The clock generator includes a fractional baud rate generator that is able to generate a wide range of USART baud rates from any system clock frequencies. This removes the need to use an external crystal oscillator with a specific frequency to achieve a required baud rate. It also supports external clock input in synchronous slave operation.

An IRCOM module can be enabled for one USART to support IrDA 1.4 physical compliant pulse modulation and demodulation for baud rates up to 115.2Kbps.

One USART can be connected to the XMEGA Custom Logic module (XCL). When used with the XCL, the data length within an USART/SPI frame can be controlled by the peripheral counter (PEC) within the XCL. This enables configurable frame length up to 256 bits. In addition, the TxD/RxD data can be encoded/decoded before the signal is fed into the USART receiver, or after the signal is output from transmitter when the USART is connected to XCL LUT outputs.

When the USART is set in master SPI mode, all USART-specific logic is disabled, leaving the transmit and receive buffers, shift registers, and baud rate generator enabled. The registers are used in both modes, but their functionality differs for some control settings. Pin control and interrupt generation are identical in both modes.

PORTC and PORTD each has one USART. Notation of these peripherals are USARTC0 and USARTD0, respectively.

28. ADC – 12-bit Analog to Digital Converter

28.1 Features

- 12-bit resolution
- Up to 300 thousand samples per second
  - Down to 2.3µs conversion time with 8-bit resolution
  - Down to 3.35µs conversion time with 12-bit resolution
- Differential and single-ended input
  - Up to 16 single-ended inputs
  - 16x8 differential inputs with optional gain
- Built-in differential gain stage
  - 1/2x, 1x, 2x, 4x, 8x, 16x, 32x, and 64x gain options
- Single, continuous and scan conversion options
- Four internal inputs
  - Internal temperature sensor
  - DAC output
  - AV_{CC} voltage divided by 10
  - 1.1V bandgap voltage
- Internal and external reference options
- Compare function for accurate monitoring of user defined thresholds
- Offset and gain correction
- Averaging
- Over-sampling and decimation
- Optional event triggered conversion for accurate timing
- Optional interrupt/event on compare result
- Optional EDMA transfer of conversion results

28.2 Overview

The ADC converts analog signals to digital values. The ADC has 12-bit resolution and is capable of converting up to 300 thousand samples per second (ksps). The input selection is flexible, and both single-ended and differential measurements can be done. For differential measurements, an optional gain stage is available to increase the dynamic range. In addition, several internal signal inputs are available. The ADC can provide both signed and unsigned results.

The ADC measurements can either be started by application software or an incoming event from another peripheral in the device. The ADC measurements can be started with predictable timing, and without software intervention. It is possible to use EDMA to move ADC results directly to memory or peripherals when conversions are done.

Both internal and external reference voltages can be used. An integrated temperature sensor is available for use with the ADC. The output from the DAC, $AV_{CC}/10$, and the bandgap voltage can also be measured by the ADC.

The ADC has a compare function for accurate monitoring of user defined thresholds with minimum software intervention required.

When operation in noisy conditions, the average feature can be enabled to increase the ADC resolution. Up to 1024 samples can be averaged, enabling up to 16-bit resolution results. In the same way, using the over-sampling and decimation mode, the ADC resolution is increased up to 16-bits, which results in up to 4-bit extra lsb resolution. The ADC includes various calibration options. In addition to standard production calibration, the user can enable the offset and gain correction to improve the absolute ADC accuracy.

XCKn	Transfer Clock for USART n
RXDn	Receiver Data for USART n
TXDn	Transmitter Data for USART n
SS	Slave Select for SPI
MOSI	Master Out Slave In for SPI
MISO	Master In Slave Out for SPI
SCK	Serial Clock for SPI

32.1.6 Oscillators, Clock, and Event

TOSCn	Timer Oscillator pin n
XTALn	Input/Output for Oscillator pin n
CLKOUT	Peripheral Clock Output
EVOUT	Event Channel Output
RTCOUT	RTC Clock Source Output

32.1.7 Debug/System Functions

RESET	Reset pin
PDI_CLK	Program and Debug Interface Clock pin
PDI_DATA	Program and Debug Interface Data pin

32.2 Alternate Pin Functions

The tables below show the primary/default function for each pin on a port in the first column, the pin number in the second column, and then all alternate pin functions in the remaining columns. The head row shows what peripheral that enable and use the alternate pin functions.

For better flexibility, some alternate functions also have selectable pin locations for their functions, this is noted under the first table where this apply.

Table 32-1. PORT A – Alternate Functions

PORT A	Pin#	ADCA POS/ GAINPOS	ADCA NEG/ GAINNEG	DACA	ACA POS	ACA NEG	ACA OUT	REFA
PA0	6	ADC 0	ADC 0		AC0	AC0		AREF
PA1	5	ADC 1	ADC 1		AC1	AC1		
PA2	4	ADC 2	ADC 2	DAC0	AC2			
PA3	3	ADC 3	ADC 3	DAC1	AC3	AC3		
PA4	2	ADC 4	ADC 4		AC4			
PA5	31	ADC 5	ADC 5		AC5	AC5		
PA6	30	ADC 6	ADC 6		AC6		AC1OUT	
PA7	29	ADC 7	ADC 7			AC7	AC0OUT	

Table 32-2. PORT C – Alternate Functions

PORT C	Pin #	TCC4	WEXC	TCC5	USARTC0	SPIC	TWI	XCL (LUT)	EXTCLK	AC OUT
PC0	16	OC4A	OC4ALS				SDA	IN1/OUT0		
PC1	15	OC4B	OC4AHS		XCK0		SCL	IN2		
PC2	14	OC4C	OC4BLS		RXD0			IN0		
PC3	13	OC4D	OC4BHS		TXD0			IN3		
PC4	12	OC4A	OC4CLS	OC5A		$\overline{SS}$		IN1/OUT0	EXTCLK	
PC5	11	OC4B	OC4CHS	OC5B	XCK0	SCK		IN2		
PC6	10	OC4C	OC4DLS		RXD0	MISO		IN0		AC1OUT
PC7	9	OC4D	OC4DHS		TXD0	MOSI		IN3		AC0OUT

Table 32-3. Debug – Program and Debug Functions

DEBUG	Pin #	PROG
RESET	8	PDI CLOCK
PDI	7	PDI DATA

34. Instruction Set Summary

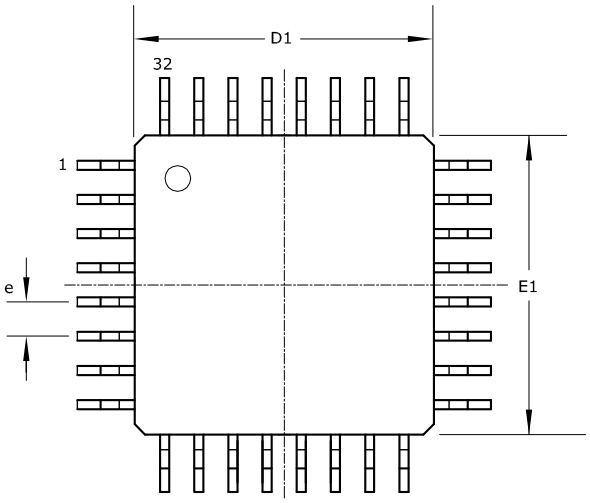
Mnemonics	Operands	Description	Operation	Flags	#Clocks
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One’s Complement	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two’s Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2
DES	K	Data Encryption	if (H = 0) then R15:R0 ← Encrypt(R15:R0, K) else if (H = 1) then R15:R0 ← Decrypt(R15:R0, K)		1/2
Branch instructions					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2
EIJMP		Extended Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	2 / 3(1)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ICALL		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2 / 3[1]
EICALL		Extended Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	3[1]
CALL	k	call Subroutine	PC ← k	None	3 / 4[1]
RET		Subroutine Return	PC ← STACK	None	4 / 5[1]
RETI		Interrupt Return	PC ← STACK	I	4 / 5[1]
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	2 / 3 / 4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	2 / 3 / 4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data transfer instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1

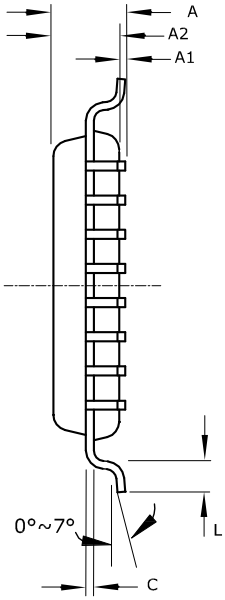
35. Packaging Information

35.1 32A

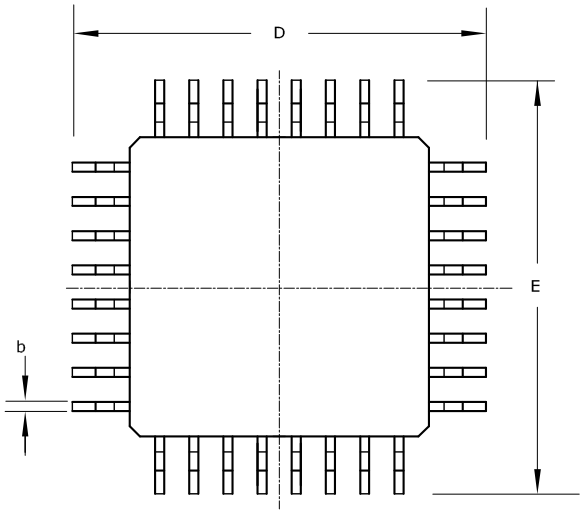
DRAWINGS NOT SCALED



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS

(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	-----	-----	1.20	
A1	0.05	-----	0.15	
A2	0.95	1.00	1.05	
D/E	8.75	9.00	9.25	
D1/E1	6.90	7.00	7.10	2
C	0.09	-----	0.20	
L	0.45	-----	0.75	
b	0.30	-----	0.45	
e	0.80 TYP			
n	32			

Notes :
1. This drawing is for general information only. Refer to JEDEC Drawing MS-026, Variation ABA.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10mm maximum.

02/29/2012

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	MA, 32 Lds - 0.80mm Pitch, 7x7x1.00mm Body size Thin Profile Plastic Quad Flat Package (TQFP)	AUT	MA	C

Table 36-26. External Clock with Prescaler (1) for System Clock

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$1/t_{CK}$	Clock Frequency (2)	V_{CC} = 1.6 - 1.8V	0		90	MHz
		V_{CC} = 2.7 - 3.6V	0		142	
t_{CK}	Clock Period	V_{CC} = 1.6 - 1.8V	11			ns
		V_{CC} = 2.7 - 3.6V	7			
t_{CH}	Clock High Time	V_{CC} = 1.6 - 1.8V	4.5			
		V_{CC} = 2.7 - 3.6V	2.4			
t_{CL}	Clock Low Time	V_{CC} = 1.6 - 1.8V	4.5			
		V_{CC} = 2.7 - 3.6V	2.4			
t_{CR}	Rise Time (for maximum frequency)	V_{CC} = 1.6 - 1.8V			1.5	
		V_{CC} = 2.7 - 3.6V			1.0	
t_{CF}	Fall Time (for maximum frequency)	V_{CC} = 1.6 - 1.8V			1.5	
		V_{CC} = 2.7 - 3.6V			1.0	
Δt_{CK}	Change in period from one clock cycle to the next				10	%

Notes: 1. System Clock Prescalers must be set so that maximum CPU clock frequency for device is not exceeded.
2. The maximum frequency vs. supply voltage is linear between 1.6V and 2.7V, and the same applies for all other parameters with supply voltage conditions.

36.13.7 External 16MHz Crystal Oscillator and XOSC Characteristics

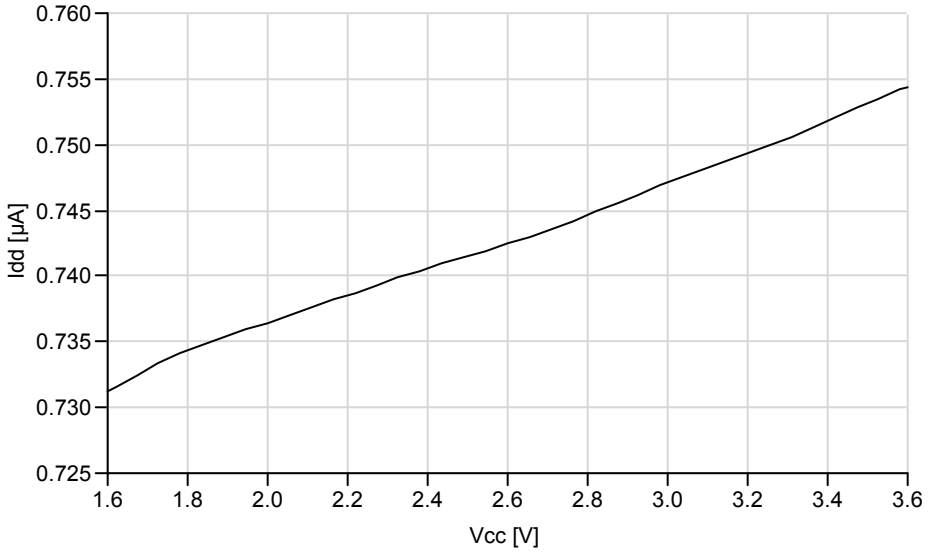
Table 36-27. External 16MHz Crystal Oscillator and XOSC Characteristics

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
	Cycle to cycle jitter	XOSCPWR=0	FRQRANGE=0		<10		ns
			FRQRANGE=1, 2, or 3		<1		
		XOSCPWR=1			<1		
	Long term jitter	XOSCPWR=0	FRQRANGE=0		<6		
			FRQRANGE=1, 2, or 3		<0.5		
		XOSCPWR=1			<0.5		

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
	Frequency error	XOSCPWR=0	FRQRANGE=0		<0.1		%
			FRQRANGE=1		<0.05		
			FRQRANGE=2 or 3		<0.005		
		XOSCPWR=1			<0.005		
	Duty cycle	XOSCPWR=0	FRQRANGE=0		40		
			FRQRANGE=1		42		
			FRQRANGE=2 or 3		45		
		XOSCPWR=1			48		
R_Q	Negative impedance (1)	XOSCPWR=0, FRQRANGE=0	0.4MHz resonator, CL=100pF				Ω
			1MHz crystal, CL=20pF				
			2MHz crystal, CL=20pF				
		XOSCPWR=0, FRQRANGE=1, CL=20pF	2MHz crystal				
			8MHz crystal				
			9MHz crystal				
		XOSCPWR=0, FRQRANGE=2, CL=20pF	8MHz crystal				
			9MHz crystal				
			12MHz crystal				
		XOSCPWR=0, FRQRANGE=3, CL=20pF	9MHz crystal				
			12MHz crystal				
			16MHz crystal				
		XOSCPWR=1, FRQRANGE=0, CL=20pF	9MHz crystal				
			12MHz crystal				
			16MHz crystal				
		XOSCPWR=1, FRQRANGE=1, CL=20pF	9MHz crystal				
			12MHz crystal				
			16MHz crystal				
		XOSCPWR=1, FRQRANGE=2, CL=20pF	12MHz crystal				
			16MHz crystal				
		XOSCPWR=1, FRQRANGE=3, CL=20pF	12MHz crystal				
			16MHz crystal				
	ESR	SF=Safety factor				min(RQ) /SF	kΩ

Figure 37-19.Power-down Mode Supply Current vs. Temperature

Sampled BOD with Watchdog Timer running on ULP oscillator



37.1.4 Power-save Mode Supply Current

Figure 37-20.Power-save Mode Supply Current vs. V_{CC}

Real Time Counter enabled and running from 1.024kHz output of 32.768kHz TOSC

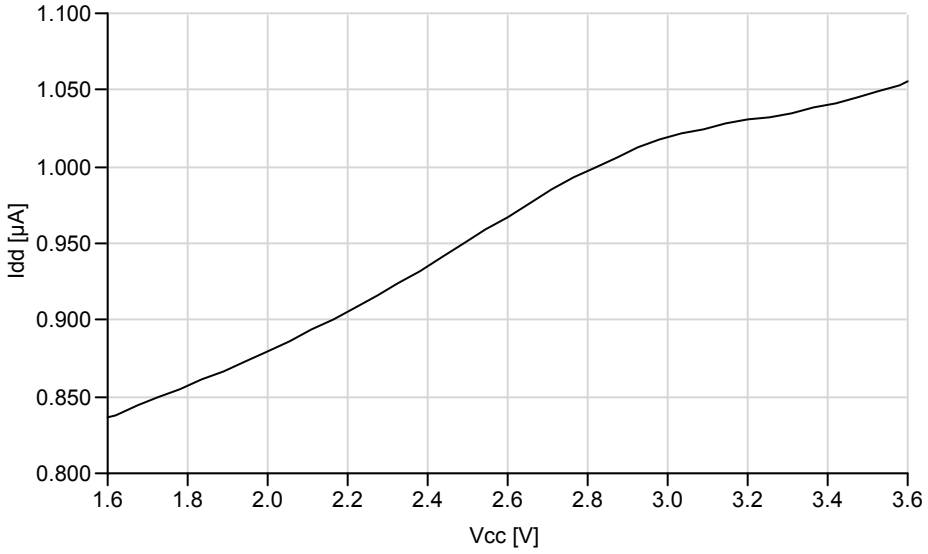


Figure 37-35.I/O Pin Input Threshold Voltage vs. V_{CC}

V_{IH} I/O pin read as "1"

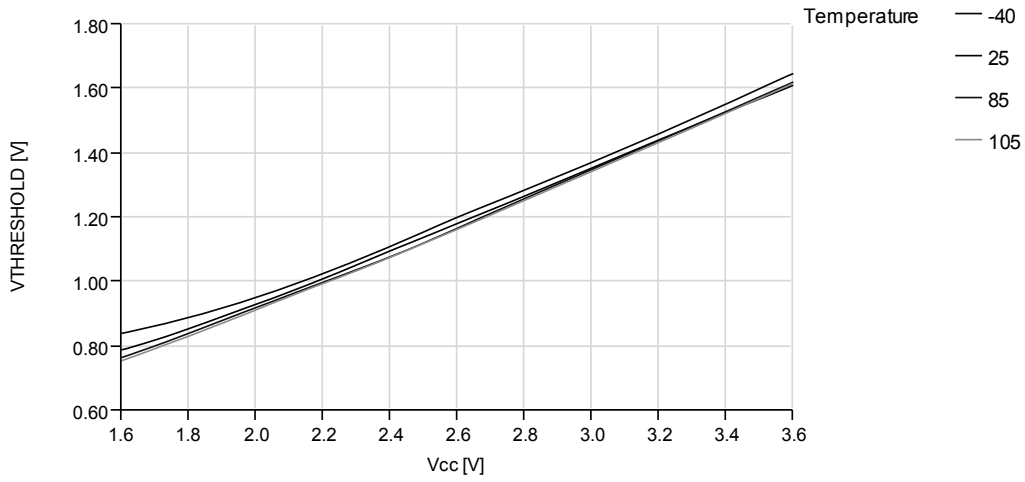


Figure 37-36.I/O Pin Input Threshold Voltage vs. V_{CC}

V_{IL} I/O pin read as "0"

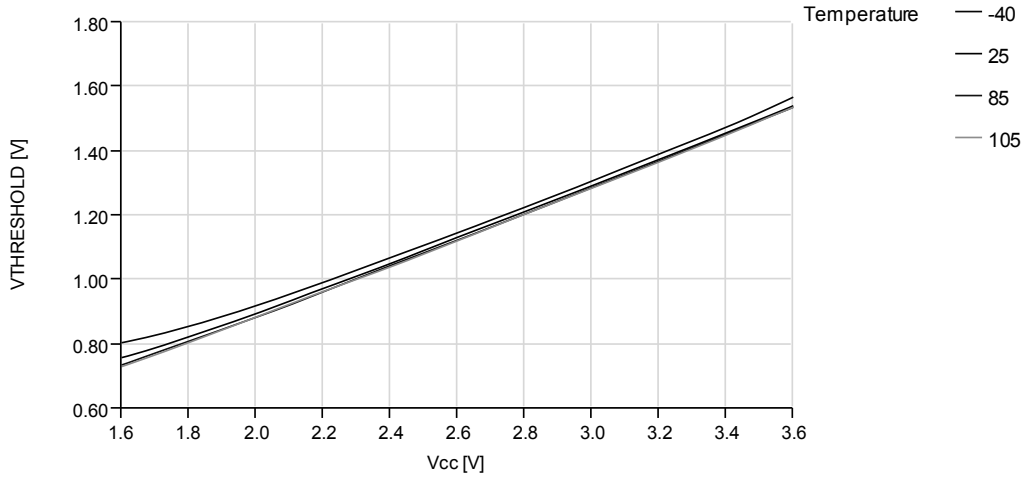


Figure 37-45. ADC Offset Error vs. V_{REF}

T = 25°C, V_{CC} = 3.6V, ADC sample rate = 300ksps

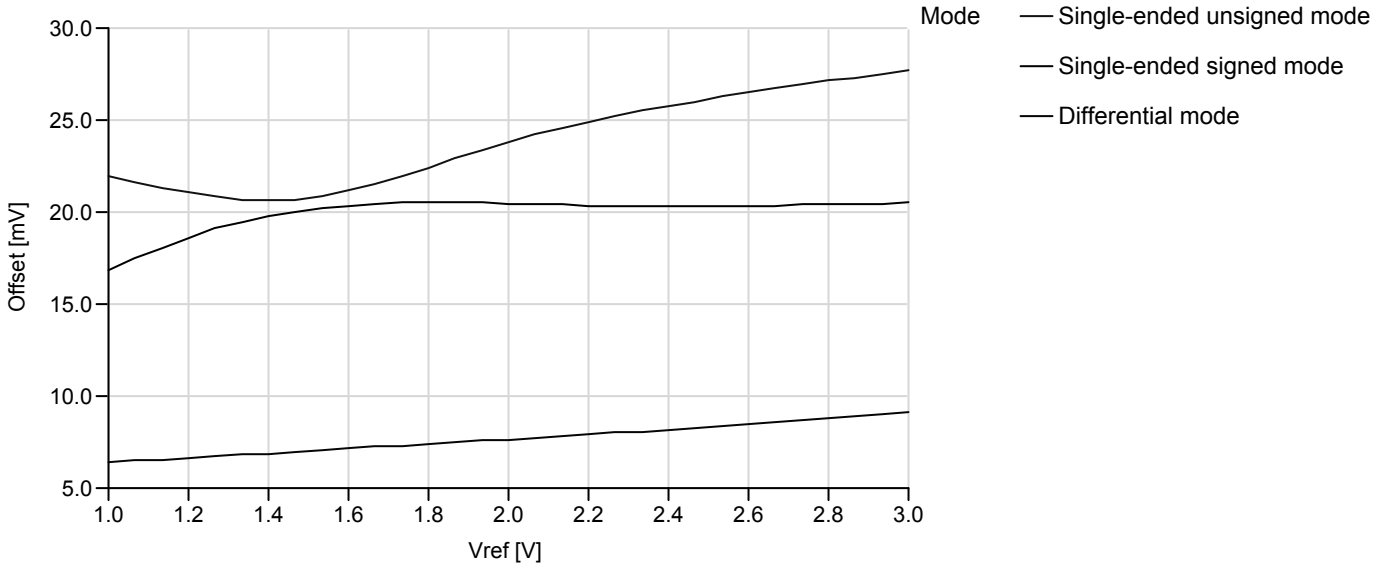


Figure 37-46.ADC Gain Error vs. Temperature

V_{CC} = 3.6V, V_{REF} = external 1.0V, sample rate = 300ksps

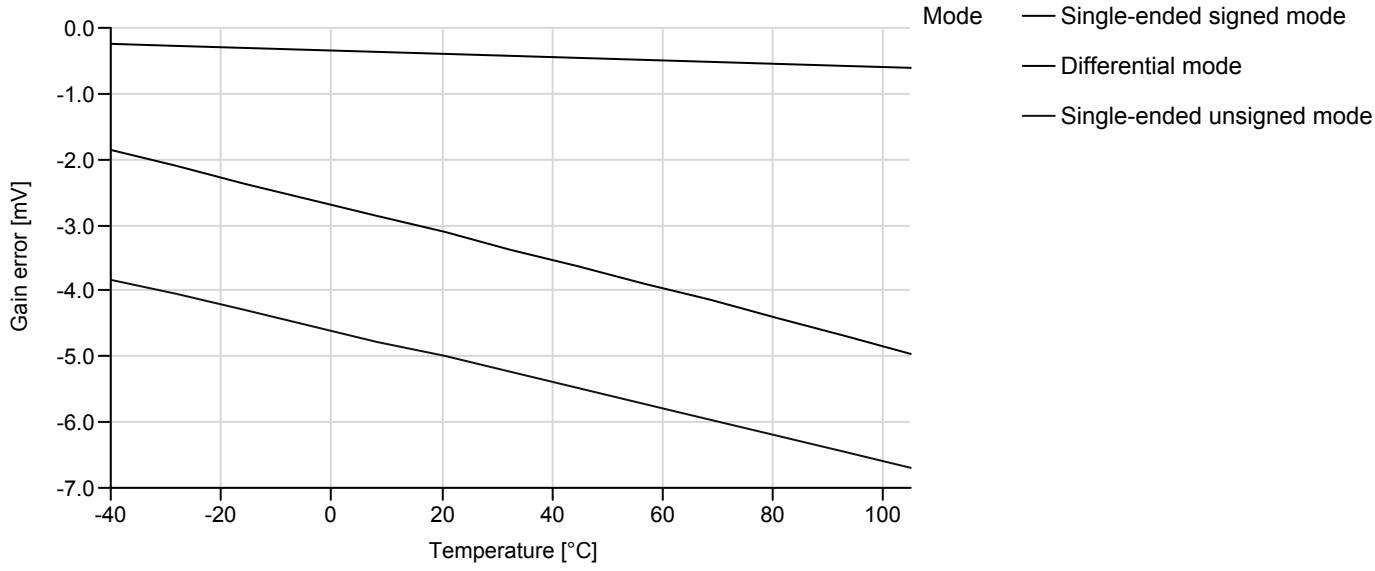
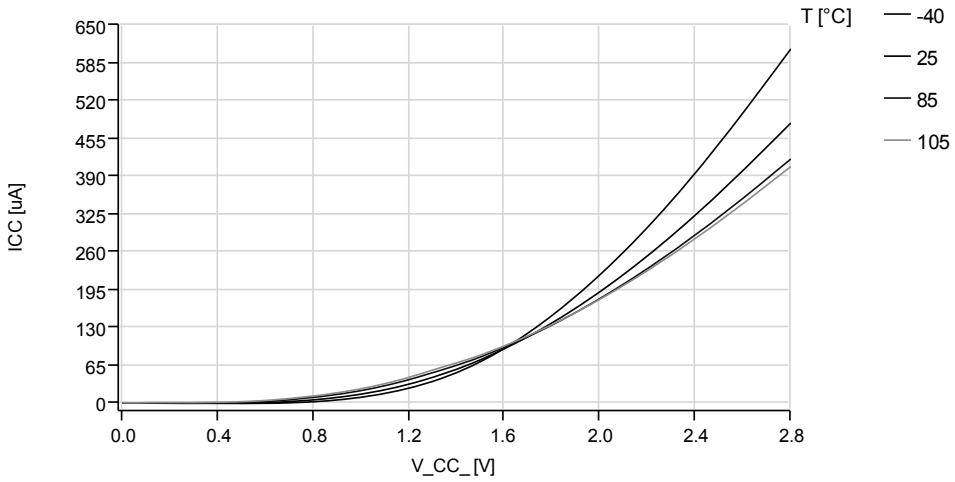


Figure 37-69.Power-on Reset Current Consumption vs. V_{CC}

BOD level = 3.0V, enabled in sampled mode



37.10.3 8MHz Internal Oscillator

Figure 37-74. 8MHz Internal Oscillator Frequency vs. Temperature

Normal mode

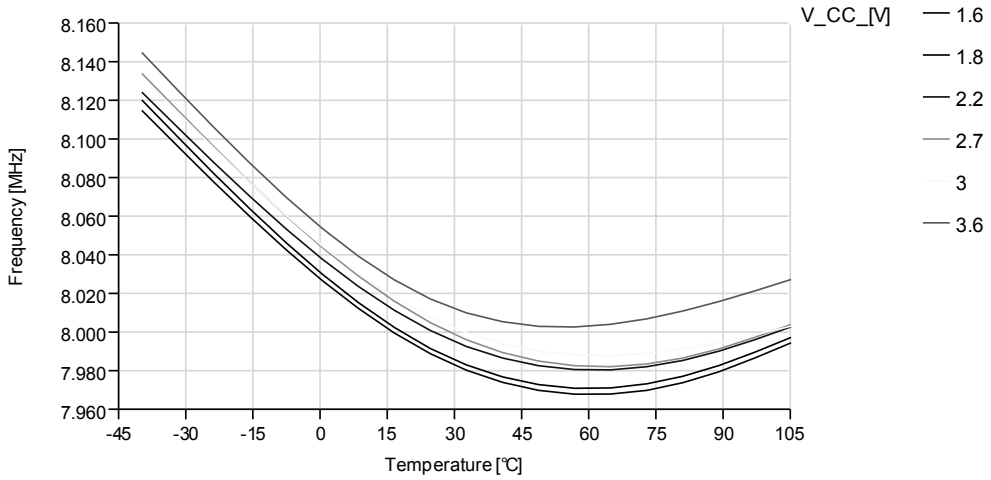


Figure 37-75. 8MHz Internal Oscillator Frequency vs. Temperature

Low power mode

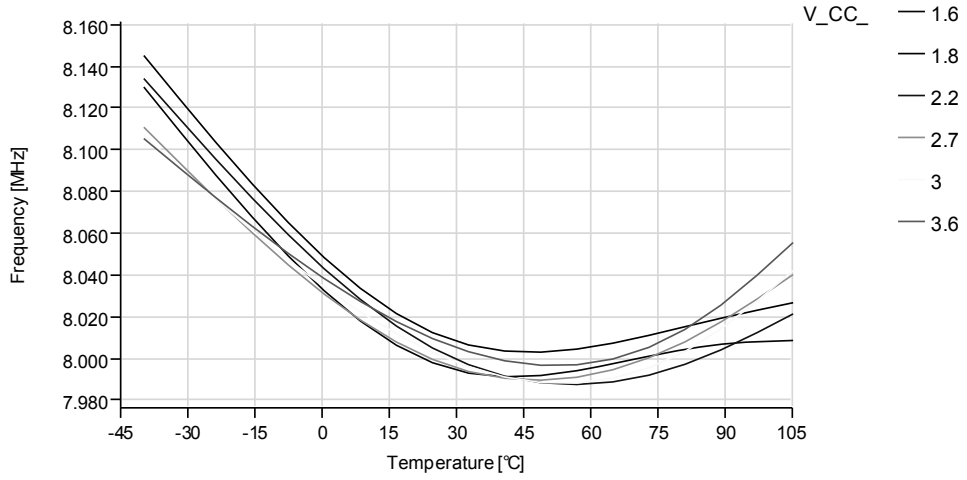
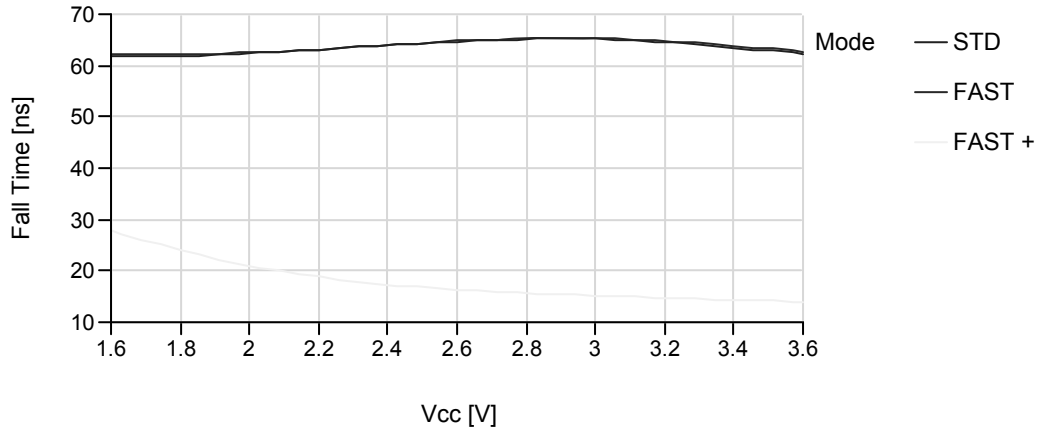
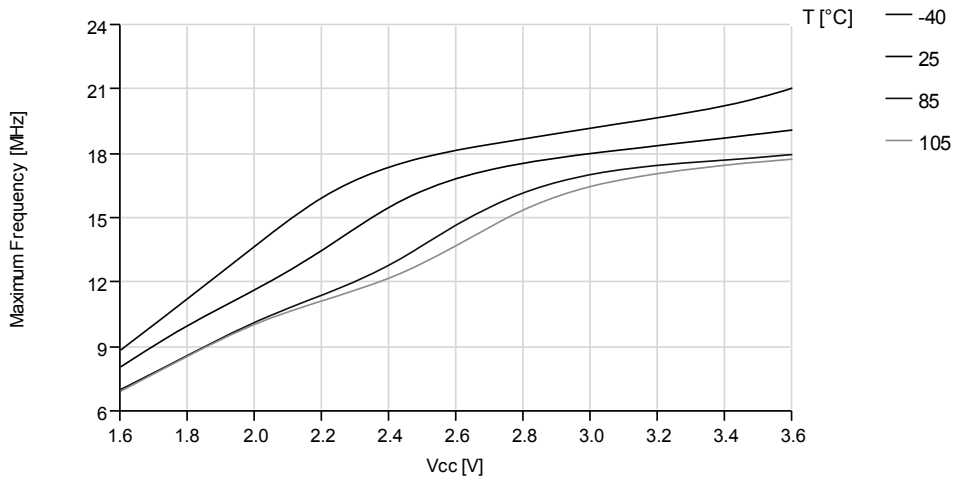


Figure 37-84. SDA Fall Time vs. V_{CC}



37.12 PDI Characteristics

Figure 37-85. Maximum PDI Frequency vs. V_{CC}



38. Errata – ATxmega32E5 / ATxmega16E5 / ATxmega8E5

38.1 Rev. B

- DAC: AREF on PD0 is not available for the DAC
- ADC: Offset correction fails in unsigned mode
- EEPROM write and Flash write operations fails under 2.0V
- TWI Master or slave remembering data
- TWI SM bus level one Master or slave remembering data
- Temperature Sensor not calibrated
- Automatic port override on PORT C
- Sext timer is not implemented in slave mode

Issue: DAC: AREF on PD0 is not available for the DAC

The AREF external reference input on pin PD0 is not available for the DAC.

Workaround:

No workaround. Only AREF on pin PA0 can be used as external reference input for the DAC.

Issue: ADC: Offset correction fails in unsigned mode

In single ended, unsigned mode, a problem appears in low saturation (zero) when the offset correction is activated. The offset is removed from result and when a negative result appears, the result is not correct.

Workaround:

No workaround, but avoid using this correction method to cancel ΔV effect.

Issue: EEPROM write and Flash write operations fails under 2.0V

EEPROM write and Flash write operations are limited from 2.0V to 3.6V. Other functionalities operates from 1.6V to 3.6V.

Workaround:

None.

Issue: TWI master or slave remembering data

If a write is made to Data register, prior to Address register, the TWI design sends the data as soon as the write to Address register is made. But the send data will be always 0x00.

Workaround:

None.

Issue: ADC: Averaging is failing when channel scan is enabled

For a correct operation, the averaging must complete on the on-going channel before incrementing the input offset. In the current implementation, the input offset is incremented after the ADC sampling is done.

Workaround:

None.

Issue: ADC: Averaging in single conversion requires multiple conversion triggers

For a normal operation, an unique start of conversion trigger starts a complete average operation. Then, for N-samples average operation, we should have:

- One start of conversion
- N conversions + average
- Optional interrupt when the Nth conversion/last average is completed

On silicon we need:

- N start of conversion

The two additional steps are well done.

Workaround:

- Set averaging configuration
- N starts of conversion by polling the reset of START bit
- Wait for interrupt flag (end of averaging)

Issue: ADC accumulator sign extends the result in unsigned mode averaging

In unsigned mode averaging, when the msb is going high(1), measurements are considered as negative when right shift is used. This sets the unused most significant bits once the shift is done.

Workaround:

Mask to zero the unused most significant bits once shift is done.

Issue: ADC: Free running average mode issue

In free running mode the ADC stops the ongoing averaging as soon as free running bit is disabled. This creates the need to flush the ADC before starting the next conversion since one or two conversions might have taken place in the internal accumulator.

Workaround:

Disable and re-enable the ADC before the start of next conversion in free running average mode.

Issue: ADC: Event triggered conversion in averaging mode

If the ADC is configured as event triggered in averaging mode, then a single event does not complete the entire averaging as it should be.

Workaround:

In the current revision, N events are needed for completing averaging on N samples.