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	EBI/EMI, I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	78
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	1.6V ~ 3.6V
Data Converters	A/D 16x12b; D/A 4x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atxmega128a1-au

7. AVR CPU

7.1 Features

- 8/16-bit high performance AVR RISC Architecture
  - 138 instructions
  - Hardware multiplier
- 32x8-bit registers directly connected to the ALU
- Stack in SRAM
- Stack Pointer accessible in I/O memory space
- Direct addressing of up to 16M Bytes of program and data memory
- True 16/24-bit access to 16/24-bit I/O registers
- Support for 8-, 16- and 32-bit Arithmetic
- Configuration Change Protection of system critical features

7.2 Overview

All Atmel AVR XMEGA devices use the 8/16-bit AVR CPU. The main function of the CPU is to execute the code and perform all calculations. The CPU is able to access memories, perform calculations, control peripherals, and execute the program in the flash memory. Interrupt handling is described in a separate section, refer to “Interrupts and Programmable Multilevel Interrupt Controller” on page 29.

7.3 Architectural Overview

In order to maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate memories and 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 pre-fetched from the program memory. This enables instructions to be executed on every clock cycle. For details of all AVR instructions, refer to http://www.atmel.com/avr.

7.4 ALU - Arithmetic Logic Unit

The arithmetic logic unit (ALU) supports arithmetic and logic operations between registers or between a constant and a register. Single-register operations can also be executed. The ALU operates in direct connection with all 32 general purpose registers. In a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed and the result is stored in the register file. After an arithmetic or logic operation, the status register is updated to reflect information about the result of the operation.

ALU operations are divided into three main categories – arithmetic, logical, and bit functions. Both 8- and 16-bit arithmetic is supported, and the instruction set allows for efficient implementation of 32-bit aritmetic. The hardware multiplier supports signed and unsigned multiplication and fractional format.

7.4.1 Hardware Multiplier

The multiplier is capable of multiplying two 8-bit numbers into a 16-bit result. The hardware multiplier supports different variations of signed and unsigned integer and fractional numbers:

- Multiplication of unsigned integers
- Multiplication of signed integers
- Multiplication of a signed integer with an unsigned integer
- Multiplication of unsigned fractional numbers
- Multiplication of signed fractional numbers
- Multiplication of a signed fractional number with an unsigned one

A multiplication takes two CPU clock cycles.

7.5 Program Flow

After reset, the CPU starts to execute instructions from the lowest address in the flash program memory ‘0.’ The program counter (PC) addresses the next instruction to be fetched.

Program flow is provided by conditional and unconditional jump and call instructions capable of addressing the whole address space directly. Most AVR instructions use a 16-bit word format, while a limited number use a 32-bit format.

During interrupts and subroutine calls, the return address PC is stored on the stack. The stack is allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. After reset, the stack pointer (SP) points to the highest address in the internal SRAM. The SP is read/write accessible in the I/O memory space, enabling easy implementation of multiple stacks or stack areas. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR CPU.

7.6 Status Register

The status register (SREG) contains information about the result of the most recently executed arithmetic or logic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the instruction set reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The status register is not automatically stored when entering an interrupt routine nor restored when returning from an interrupt. This must be handled by software.

The status register is accessible in the I/O memory space.

7.6.1 Stack and Stack Pointer

The stack is used for storing return addresses after interrupts and subroutine calls. It can also be used for storing temporary data. The stack pointer (SP) register always points to the top of the stack. It is implemented as two 8-bit registers that are accessible in the I/O memory space. Data are pushed and popped from the stack using the PUSH and POP instructions. The stack grows from a higher memory location to a lower memory location. This implies that pushing data onto the stack decreases the SP, and popping data off the stack increases the SP. The SP is automatically loaded

9. DMAC - Direct Memory Access Controller

9.1 Features

- Allows High-speed data transfer
  - From memory to peripheral
  - From memory to memory
  - From peripheral to memory
  - From peripheral to peripheral
- 4 Channels
- From 1 byte and up to 16M bytes transfers in a single transaction
- Multiple addressing modes for source and destination address
  - Increment
  - Decrement
  - Static
- 1, 2, 4, or 8 byte Burst Transfers
- Programmable priority between channels

9.2 Overview

The four-channel direct memory access (DMA) controller can transfer data between memories and peripherals, and thus offload these tasks from the CPU. It enables high data transfer rates with minimum CPU intervention, and frees up CPU time. The four DMA channels enable up to four independent and parallel transfers.

The DMA controller can move data between SRAM and peripherals, between SRAM locations and directly between peripheral registers. With access to all peripherals, the DMA controller can handle automatic transfer of data to/from communication modules. The DMA controller can also read from memory mapped EEPROM.

Data transfers are done in continuous bursts of 1, 2, 4, or 8 bytes. They build block transfers of configurable size from 1 byte to 64KB. A repeat counter can be used to repeat each block transfer for single transactions up to 16MB. Source and destination addressing can be static, incremental or decremental. Automatic reload of source and/or destination addresses can be done after each burst or block transfer, or when a transaction is complete. Application software, peripherals, and events can trigger DMA transfers.

The four DMA channels have individual configuration and control settings. This include source, destination, transfer triggers, and transaction sizes. They have individual interrupt settings. Interrupt requests can be generated when a transaction is complete or when the DMA controller detects an error on a DMA channel.

To allow for continuous transfers, two channels can be interlinked so that the second takes over the transfer when the first is finished, and vice versa.

12. Power Management and Sleep Modes

12.1 Features

- Power management for adjusting power consumption and functions
- 5 sleep modes
  - Idle
  - Power-down
  - Power-save
  - Standby
  - Extended standby
- Power reduction register to disable clock and turn off unused peripherals in active and idle modes

12.2 Overview

Various sleep modes and clock gating are provided in order to tailor power consumption to application requirements. This enables the Atmel AVR XMEGA microcontroller to stop unused modules to save power.

All sleep modes are available and can be entered from active mode. In active mode, the CPU is executing application code. When the device enters sleep mode, program execution is stopped and interrupts or a reset is used to wake the device again. The application code decides which sleep mode to enter and when. Interrupts from enabled peripherals and all enabled reset sources can restore the microcontroller from sleep to active mode.

In addition, power reduction registers provide a method to stop the clock to individual peripherals from software. When this is done, the current state of the peripheral is frozen, and there is no power consumption from that peripheral. This reduces the power consumption in active mode and idle sleep modes and enables much more fine-tuned power management than sleep modes alone.

12.3 Sleep Modes

Sleep modes are used to shut down modules and clock domains in the microcontroller in order to save power. XMEGA microcontrollers have five different sleep modes tuned to match the typical functional stages during application execution. A dedicated sleep instruction (SLEEP) is available to enter sleep mode. Interrupts are used to wake the device from sleep, and the available interrupt wake-up sources are dependent on the configured sleep mode. When an enabled interrupt occurs, the device will wake up and execute the interrupt service routine before continuing normal program execution from the first instruction after the SLEEP instruction. If other, higher priority interrupts are pending when the wake-up occurs, their interrupt service routines will be executed according to their priority before the interrupt service routine for the wake-up interrupt is executed. After wake-up, the CPU is halted for four cycles before execution starts.

The content of the register file, SRAM and registers are kept during sleep. If a reset occurs during sleep, the device will reset, start up, and execute from the reset vector.

12.3.1 Idle Mode

In idle mode the CPU and nonvolatile memory are stopped (note that any ongoing programming will be completed), but all peripherals, including the interrupt controller, event system and DMA controller are kept running. Any enabled interrupt will wake the device.

12.3.2 Power-down Mode

In power-down mode, all clocks, including the real-time counter clock source, are stopped. This allows operation only of asynchronous modules that do not require a running clock. The only interrupts that can wake up the MCU are the two-wire interface address match interrupt and asynchronous port interrupts, e.g pin change.

14. Interrupts and Programmable Multilevel Interrupt Controller

14.1 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

14.2 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.

14.3 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 A1U devices are shown in Table 14-1. 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 14-1. The program address is the word address.

Table 14-1. Reset and Interrupt vectors

Program Address (Base Address)	Source	Interrupt Description
0x000	RESET	
0x002	OSCF_INT_vect	Crystal Oscillator Failure Interrupt vector (NMI)
0x004	PORTC_INT_base	Port C Interrupt base
0x008	PORTR_INT_base	Port R Interrupt base
0x00C	DMA_INT_base	DMA Controller Interrupt base
0x014	RTC_INT_base	Real Time Counter Interrupt base
0x018	TWIC_INT_base	Two-Wire Interface on Port C Interrupt base

17. AWeX - Advanced Waveform Extension

17.1 Features

- Output with complementary output from each Capture channel
- Four Dead Time Insertion (DTI) Units, one for each Capture channel
- 8-bit DTI Resolution
- Separate High and Low Side Dead-Time Setting
- Double Buffered Dead-Time
- Event Controlled Fault Protection
- Single Channel Multiple Output Operation (for BLDC motor control)
- Double Buffered Pattern Generation

17.2 Overview

The advanced waveform extension (AWeX) provides extra functions to the timer/counter in waveform generation (WG) modes. It is primarily intended for use with different types of motor control and other power control applications. It enables low- and high side output with dead-time insertion and fault protection for disabling and shutting down external drivers. It can also generate a synchronized bit pattern across the port pins.

Each of the waveform generator outputs from the Timer/Counter 0 are split into a complimentary pair of outputs when any AWeX features are enabled. These output pairs go through a dead-time insertion (DTI) unit that generates the non-inverted low side (LS) and inverted high side (HS) of the WG output with dead-time insertion between LS and HS switching. The DTI output will override the normal port value according to the port override setting.

The pattern generation unit can be used to generate a synchronized bit pattern on the port it is connected to. In addition, the WG output from compare channel A can be distributed to and override all the port pins. When the pattern generator unit is enabled, the DTI unit is bypassed.

The fault protection unit is connected to the event system, enabling any event to trigger a fault condition that will disable the AWeX output. The event system ensures predictable and instant fault reaction, and gives great flexibility in the selection of fault triggers.

The AWeX is available for TCC0 and TCE0. The notation of these are AWEXC and AWEXE.

19. RTC - 16-bit Real-Time Counter

19.1 Features

- 16-bit resolution
- Selectable clock source
  - 32.768kHz external crystal
  - External clock
  - 32.768kHz internal oscillator
  - 32kHz internal ULP oscillator
- Programmable 10-bit clock prescaling
- One compare register
- One period register
- Clear counter on period overflow
- Optional interrupt/event on overflow and compare match

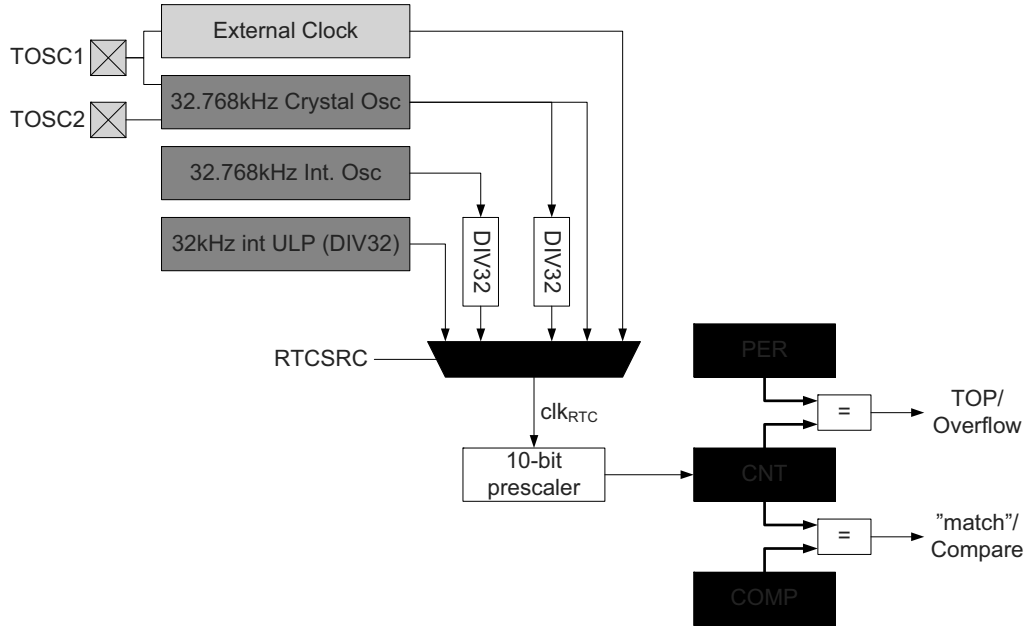
19.2 Overview

The 16-bit real-time counter (RTC) is a counter that typically runs continuously, including in low-power sleep modes, to keep track of time. It can wake up the device from sleep modes and/or interrupt the device at regular intervals.

The reference clock is typically the 1.024kHz output from a high-accuracy crystal of 32.768kHz, and this is the configuration most optimized for low power consumption. The faster 32.768kHz output can be selected if the RTC needs a resolution higher than 1ms. The RTC can also be clocked from an external clock signal, the 32.768kHz internal oscillator or the 32kHz internal ULP oscillator.

The RTC includes a 10-bit programmable prescaler that can scale down the reference clock before it reaches the counter. A wide range of resolutions and time-out periods can be configured. With a 32.768kHz clock source, the maximum resolution is 30.5µs, and time-out periods can range up to 2000 seconds. With a resolution of 1s, the maximum timeout period is more than18 hours (65536 seconds). The RTC can give a compare interrupt and/or event when the counter equals the compare register value, and an overflow interrupt and/or event when it equals the period register value.

Figure 19-1. Real Time Counter overview



CLK	SDRAM Clock	(SDRAM)
$\overline{DQM}$	Data Mask Signal/Output Enable	(SDRAM)
$\overline{RAS}$	Row Access Strobe	(SDRAM)
2P	2 Port Interface	
3P	3 Port Interface	

30.1.5 Timer/Counter and AWEX functions

OCnx	Output Compare Channel x for Timer/Counter n
$\overline{OCnx}$	Inverted Output Compare Channel x for Timer/Counter n
OCnxLS	Output Compare Channel x Low Side for Timer/Counter n
OCnxHS	Output Compare Channel x High Side for Timer/Counter n

30.1.6 Communication functions

SCL	Serial Clock for TWI
SDA	Serial Data for TWI
SCLIN	Serial Clock In for TWI when external driver interface is enabled
SCLOUT	Serial Clock Out for TWI when external driver interface is enabled
SDAIN	Serial Data In for TWI when external driver interface is enabled
SDAOUT	Serial Data Out for TWI when external driver interface is enabled
XCKn	Transfer Clock for USART n
RXDn	Receiver Data for USART n
TXDn	Transmitter Data for USART n
$\overline{SS}$	Slave Select for SPI
MOSI	Master Out Slave In for SPI
MISO	Master In Slave Out for SPI
SCK	Serial Clock for SPI

30.1.7 Oscillators, Clock and Event

n	Timer Oscillator pin n
XTALn	Input/Output for inverting Oscillator pin n
CLKOUT	Peripheral Clock Output
EVOUT	Event Channel 0 Output

30.1.8 Debug/System functions

$\overline{\text{RESET}}$	Reset pin
PDI_CLK	Program and Debug Interface Clock pin
PDI_DATA	Program and Debug Interface Data pin
TCK	JTAG Test Clock
TDI	JTAG Test Data In
TDO	JTAG Test Data Out
TMS	JTAG Test Mode Select

31. Peripheral Module Address Map

The address maps show the base address for each peripheral and module in XMEGA A1. For complete register description and summary for each peripheral module, refer to the XMEGA A Manual.

Table 31-1. Peripheral Module Address Map

Base Address	Name	Description
0x0000	GPIO	General Purpose IO Registers
0x0010	VPORT0	Virtual Port 0
0x0014	VPORT1	Virtual Port 1
0x0018	VPORT2	Virtual Port 2
0x001C	VPORT3	Virtual Port 3
0x0030	CPU	CPU
0x0040	CLK	Clock Control
0x0048	SLEEP	Sleep Controller
0x0050	OSC	Oscillator Control
0x0060	DFLLRC32M	DFLL for the 32 MHz Internal RC Oscillator
0x0068	DFLLRC2M	DFLL for the 2 MHz RC Oscillator
0x0070	PR	Power Reduction
0x0078	RST	Reset Controller
0x0080	WDT	Watch-Dog Timer
0x0090	MCU	MCU Control
0x00A0	PMIC	Programmable Multilevel Interrupt Controller
0x00B0	PORTCFG	Port Configuration
0x00C0	AES	AES Module
0x0100	DMA	DMA Controller
0x0180	EVSYS	Event System
0x01C0	NVM	Non Volatile Memory (NVM) Controller
0x0200	ADCA	Analog to Digital Converter on port A
0x0240	ADCB	Analog to Digital Converter on port B
0x0300	DACA	Digital to Analog Converter on port A
0x0320	DACB	Digital to Analog Converter on port B
0x0380	ACA	Analog Comparator pair on port A
0x0390	ACB	Analog Comparator pair on port B
0x0400	RTC	Real Time Counter
0x0440	EBI	External Bus Interface
0x0480	TWIC	Two Wire Interface on port C
0x0490	TWID	Two Wire Interface on port D

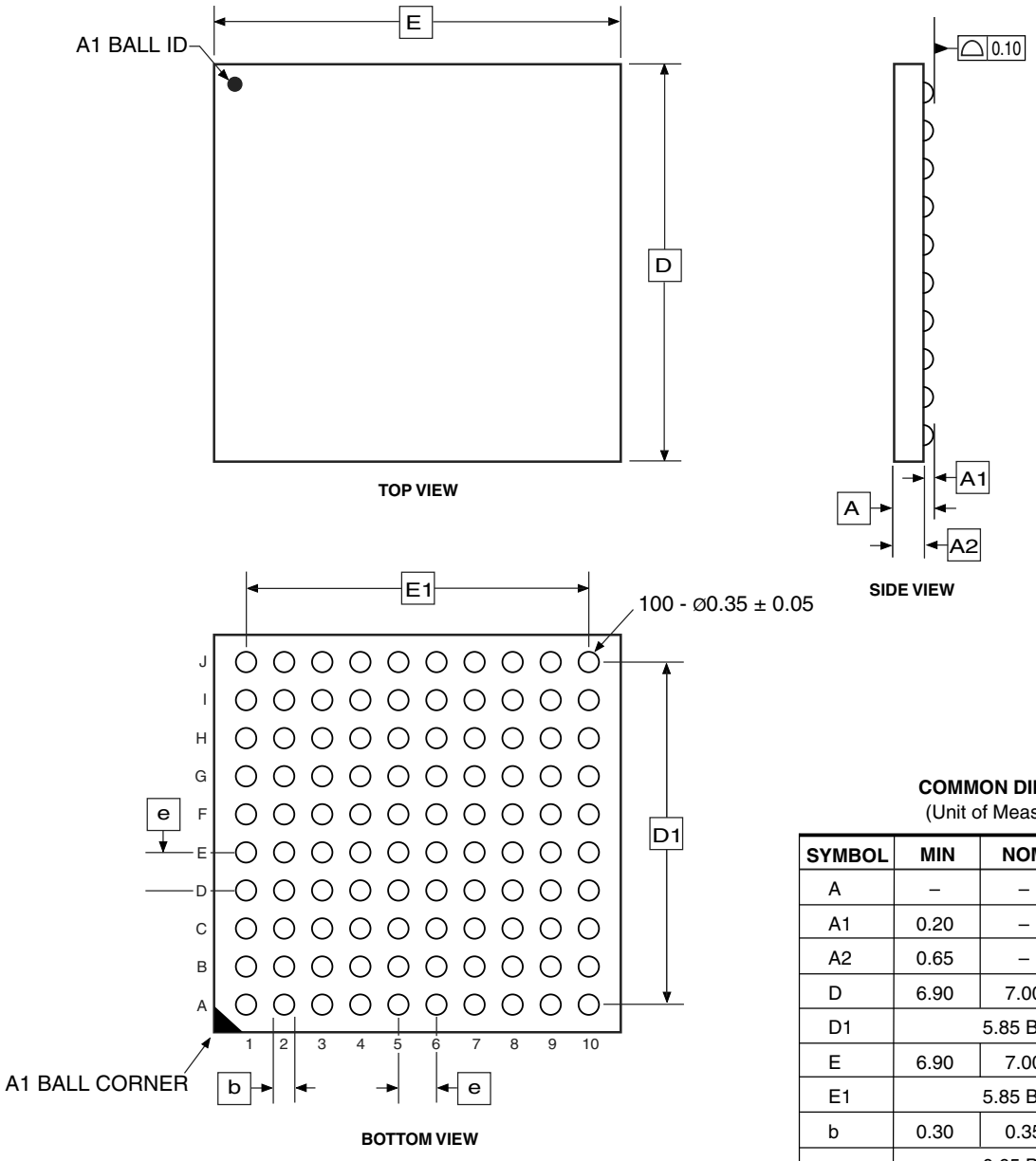
32. Instruction Set Summary

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 else if (H = 1) then R15:R0	← ←	Encrypt(R15:R0, K) 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) PC(21:16)	← ←	Z, 0	None	2
EIJMP		Extended Indirect Jump to (Z)	PC(15:0) PC(21:16)	← ←	Z, EIND	None	2
JMP	k	Jump	PC	←	k	None	3

Mnemonics	Operands	Description	Operation	Flags	#Clocks
MCU Control Instructions					
BREAK		Break	(See specific descr. for BREAK)	None	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

Notes: 1. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface.
2. One extra cycle must be added when accessing Internal SRAM.

33.3 100C2



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.00	
A1	0.20	–	–	
A2	0.65	–	–	
D	6.90	7.00	7.10	
D1		5.85 BSC		
E	6.90	7.00	7.10	
E1		5.85 BSC		
b	0.30	0.35	0.40	
e		0.65 BSC		

12/23/08

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	**100C2,** 100-ball (10 x 10 Array), 0.65 mm Pitch, 7.0 x 7.0 x 1.0 mm, Very Thin, Fine-Pitch Ball Grid Array Package (VFBGA)	CIF	100C2	A

35.7 Bod Thresholds

Figure 35-15.BOD Thresholds vs. Temperature

BOD Level = 1.6V

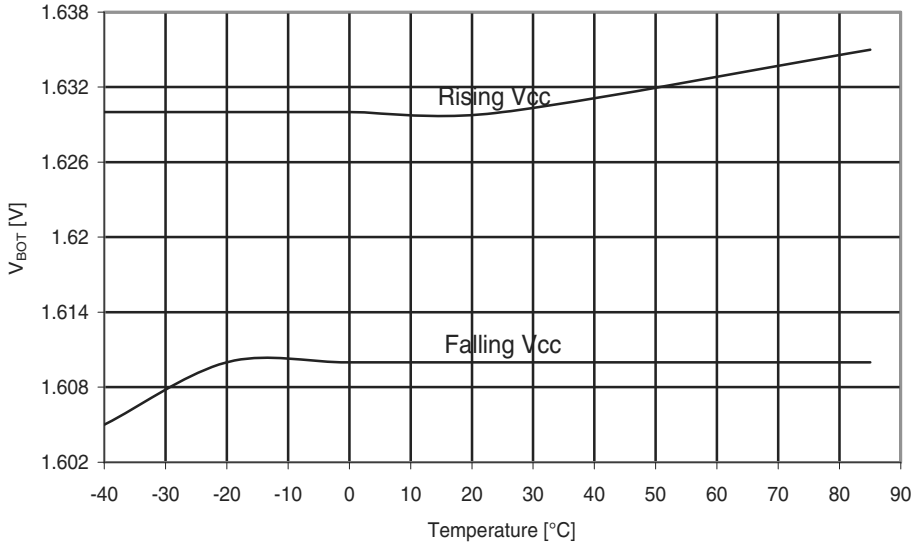


Figure 35-16.BOD Thresholds vs. Temperature

BOD Level = 2.9V

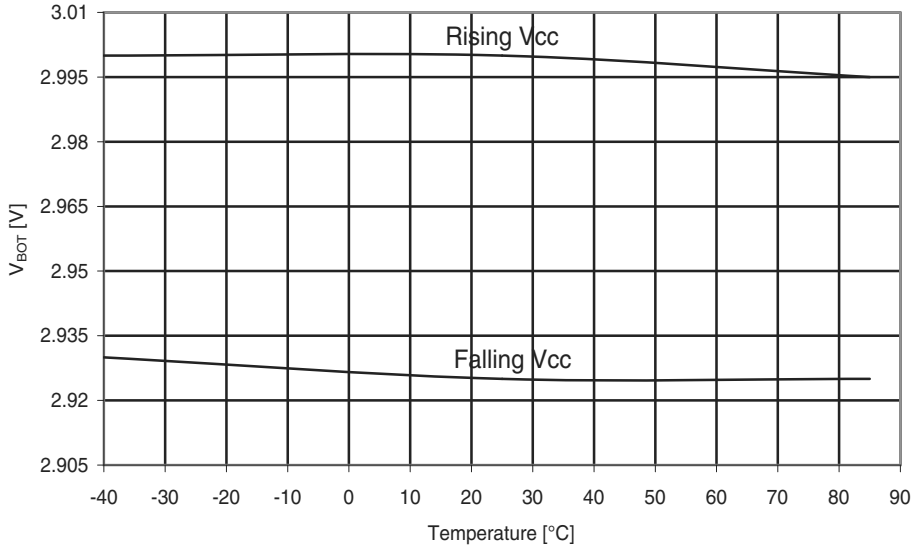


Figure 35-25.Internal 32 MHz Oscillator CalA Calibration Step Size

T = -40 to 85°C, V_{CC} = 3V

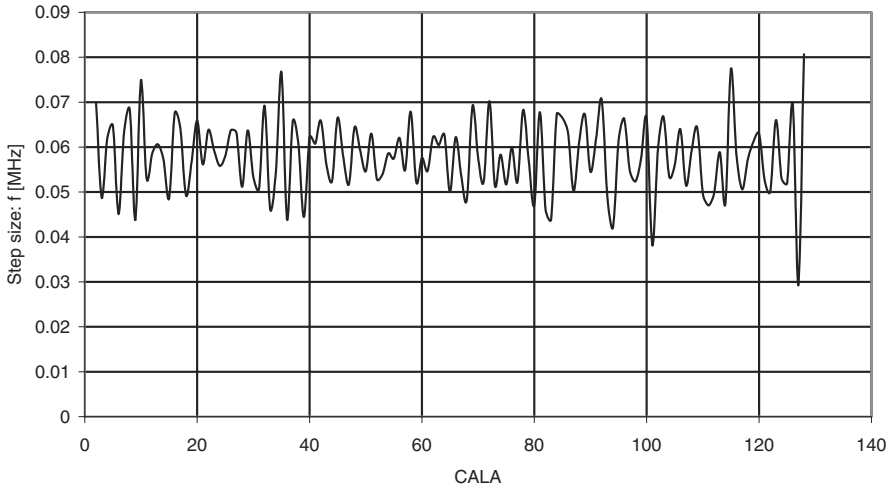
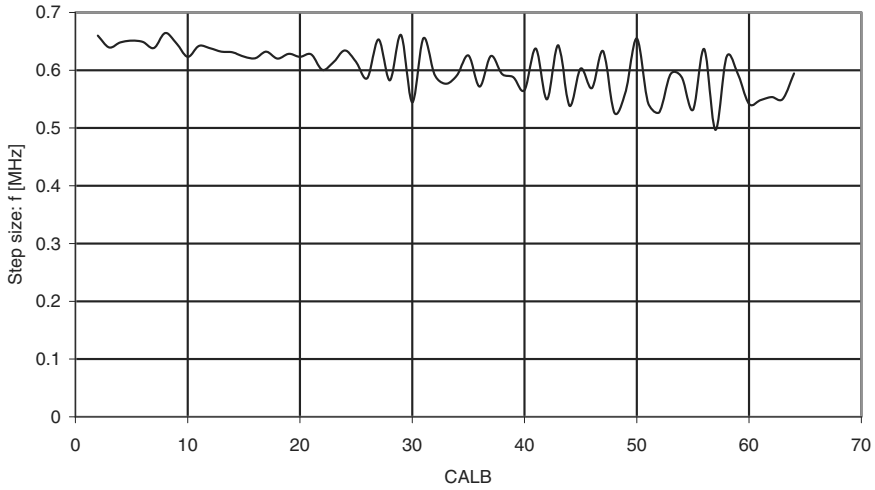


Figure 35-26.Internal 32 MHz Oscillator CalB Calibration Step Size

T = -40 to 85°C, V_{CC} = 3V



- TWI START condition at bus timeout will cause transaction to be dropped
- TWI Data Interrupt Flag erroneously read as set
- WDR instruction inside closed window will not issue reset

1. Bandgap voltage input for the ACs cannot be changed when used for both ACs simultaneously

If the Bandgap voltage is selected as input for one Analog Comparator (AC) and then selected/deselected as input for another AC, the first comparator will be affected for up to 1 µs and could potentially give a wrong comparison result.

Problem fix/Workaround

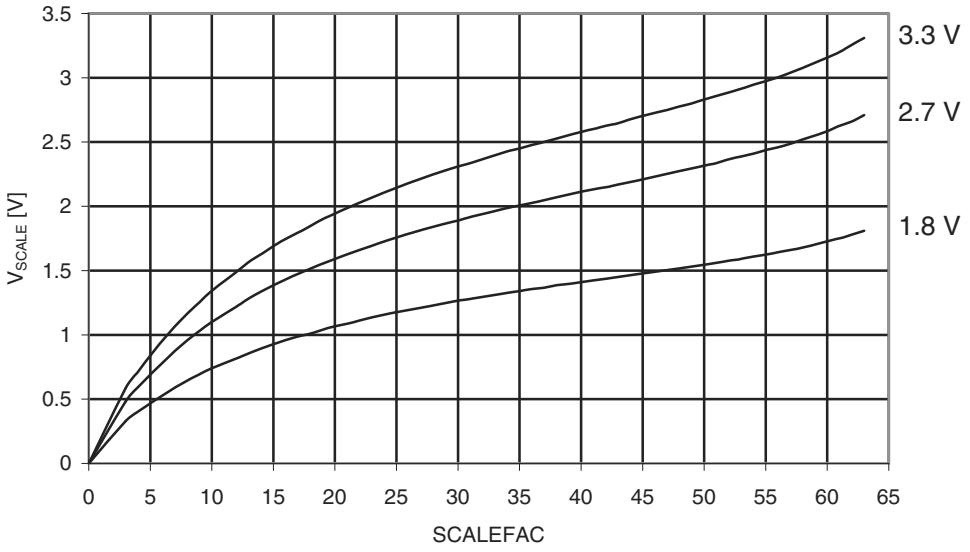
If the Bandgap is required for both ACs simultaneously, configure the input selection for both ACs before enabling any of them.

2. VCC voltage scaler for AC is non-linear

The 6-bit VCC voltage scaler in the Analog Comparators is non-linear.

Figure 36-1. Analog Comparator Voltage Scaler vs. Scalefac

T = 25°C



Problem fix/Workaround

Use external voltage input for the analog comparator if accurate voltage levels are needed

3. The ADC has up to ±2 LSB inaccuracy

The ADC will have up to ±2 LSB inaccuracy, visible as a saw-tooth pattern on the input voltage/ output value transfer function of the ADC. The inaccuracy increases with increasing voltage reference reaching ±2 LSB with 3V reference.

None.

8. Accuracy lost on first three samples after switching input to ADC gain stage

Due to memory effect in the ADC gain stage, the first three samples after changing input channel must be disregarded to achieve 12-bit accuracy.

Problem fix/Workaround

Run three ADC conversions and discard these results after changing input channels to ADC gain stage.

9. The input difference between two succeeding ADC samples is limited by VREF

If the difference in input between two samples changes more than the size of the reference, the ADC will not be able to convert the data correctly. Two conversions will be required before the conversion is correct.

Problem fix/Workaround

Discard the first conversion if input is changed more than VREF, or ensure that the input never changes more then VREF.

10. Increased noise when using internal 1.0V reference at low temperature

When operating at below 0°C and using internal 1.0V reference the RMS noise will be up 4 LSB, Peak-to-peak noise up to 25 LSB.

Problem fix/Workaround

Use averaging to remove noise.

11. Configuration of PGM and CWCM not as described in XMEGA A Manual

Enabling Common Waveform Channel Mode will enable Pattern generation mode (PGM), but not Common Waveform Channel Mode.

Enabling Pattern Generation Mode (PGM) and not Common Waveform Channel Mode (CWCM) will enable both Pattern Generation Mode and Common Waveform Channel Mode.

Problem fix/Workaround

PGM	CWCM	Description
0	0	PGM and CWCM disabled
0	1	PGM enabled
1	0	PGM and CWCM enabled
1	1	PGM enabled

12 PWM is not restarted properly after a fault in cycle-by-cycle mode

When the AWeX fault restore mode is set to cycle-by-cycle, the waveform output will not return to normal operation at first update after fault condition is no longer present.

18. DAC has up to ±10 LSB noise in Sampled Mode

If the DAC is running in Sample and Hold (S/H) mode and conversion for one channel is done at maximum rate (i.e. the DAC is always busy doing conversion for this channel), this will block refresh signals to the second channel.

Problem fix/Workaround

When using the DAC in S/H mode, ensure that none of the channels is running at maximum conversion rate, or ensure that the conversion rate of both channels is high enough to not require refresh.

19. Conversion lost on DAC channel B in event triggered mode

If during dual channel operation channel 1 is set in auto trigged conversion mode, channel 1 conversions are occasionally lost. This means that not all data-values written to the Channel 1 data register are converted.

Problem fix/Workaround

Keep the DAC conversion interval in the range 000-001 (1 and 3 CLK), and limit the Peripheral clock frequency so the conversion internal never is shorter than 1.5 µs.

20. Both DFLLs and both oscillators have to be enabled for one to work

In order to use the automatic runtime calibration for the 2 MHz or the 32 MHz internal oscillators, the DFLL for both oscillators and both oscillators have to be enabled for one to work.

Problem fix/Workaround

Enable both DFLLs and both oscillators when using automatic runtime calibration for either of the internal oscillators.

21. Access error when multiple bus masters are accessing SDRAM

If one bus master (CPU and DMA channels) is using the EBI to access an SDRAM in burst mode and another bus master is accessing the same row number in a different BANK of the SDRAM in the cycle directly after the burst access is complete, the access for the second bus master will fail.

Problem fix/Workaround

Do not put stack pointer in SDRAM and use DMA Controller in 1 byte burst mode if CPU and DMA Controller are required to access SDRAM at the same time.

22. EEPROM page buffer always written when NVM DATA0 is written

If the EEPROM is memory mapped, writing to NVM DATA0 will corrupt data in the EEPROM page buffer.

Problem fix/Workaround

Before writing to NVM DATA0, for example when doing software CRC or flash page buffer write, check if EEPROM page buffer active loading flag (EELOAD) is set. Do not write NVM DATA0 when EELOAD is set.

23. Pending full asynchronous pin change interrupts will not wake the device

37.9 8067G – 11/2008

1. Updated “Block Diagram” on page 6.
2. Updated feature list in “Memories” on page 12.
3. Updated “Programming and Debugging” on page 54.
4. Updated “Peripheral Module Address Map” on page 62. IRCOM has address 0x8F0.
5. Added “Electrical Characteristics” on page 73.
6. Added “Typical Characteristics” on page 82.
7. Added “ATxmega64A1and ATxmega128A1 rev. H” on page 96.
8. Updated “ATxmega64A1 and ATxmega128A1 rev. G” on page 107.

37.10 8067F – 09/2008

1. Updated “Features” on page 1
2. Updated “Ordering Information” on page 2
3. Updated Figure 7-1 on page 11 and Figure 7-2 on page 11.
4. Updated Table 7-2 on page 15.
5. Updated “Features” on page 48 and “Overview” on page 48.
6. Removed “Interrupt Vector Summary” section from datasheet.

37.11 8067E – 08/2008

1. Changed Figure 2-1’s title to “Block diagram and pinout” on page 3.
2. Updated Figure 2-2 on page 4.
3. Updated Table 29-2 on page 51 and Table 29-3 on page 52.

37.12 8067D – 07/2008

1. Updated “Ordering Information” on page 2.
2. Updated “Peripheral Module Address Map” on page 62.
3. Inserted “Interrupt Vector Summary” on page 56.

37.13 8067C – 06/2008

1.	Updated the Front page and “Features” on page 1.
2.	Updated the “DC Characteristics” on page 73.
3.	Updated Figure 3-1 on page 6.
4.	Added “Flash and EEPROM Page Size” on page 15.
5.	Updated Table 33-6 on page 72 with new data: Gain Error, Offset Error and Signal -to-Noise Ratio (SNR).
6.	Updated Errata “ATxmega64A1 and ATxmega128A1 rev. G” on page 107.

37.14 8067B – 05/2008

1.	Updated “Pinout/Block Diagram” on page 3 and “Pinout and Pin Functions” on page 55.
2.	Added XMEGA A1 Block Diagram, Figure 3-1 on page 6.
3.	Updated “Overview” on page 5 included the XMEGA A1 explanation text on page 6.
4.	Updated AVR CPU “Features” on page 8.
5.	Updated Event System block diagram, Figure 10-1 on page 20.
6.	Updated “Interrupts and Programmable Multilevel Interrupt Controller” on page 29.
7.	Updated “AC - Analog Comparator” on page 52.
8.	Updated “Alternate Pin Function Description” on page 55.
9.	Updated “Alternate Pin Functions” on page 58.
10.	Updated “Typical Characteristics” on page 82.
11.	Updated “Ordering Information” on page 2.
12.	Updated “Overview” on page 5.
13.	Updated Figure 6-1 on page 8.
14.	Inserted a new Figure 16-1 on page 37.
15.	Updated Speed grades in “Speed” on page 75.
16.	Added a new ATxmega384A1 device in “Features” on page 1, updated “Ordering Information” on page 2 and “Memories” on page 12.
17.	Replaced the Figure 3-1 on page 6 by a new XMEGA A1 detailed block diagram.
18.	Inserted Errata “ATxmega64A1 and ATxmega128A1 rev. G” on page 107.

37.15 8067A – 02/2008

1.	Initial revision.