Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

8/16-bit Atmel AVR XMEGA Microcontrollers

ATxmega32E5 / ATxmega16E5 / ATxmega8E5

DATASHEET

Features

- High-performance, low-power Atmel® AVR® XMEGA® 8/16-bit Microcontroller
- Nonvolatile program and data memories
  - 8K –32KB of in-system self-programmable flash
  - 2K – 4KB boot section
  - 512Bytes – 1KB EEPROM
  - 1K – 4KB internal SRAM
- Peripheral features
  - Four-channel enhanced DMA controller with 8/16-bit address match
  - Eight-channel event system
    - Asynchronous and synchronous signal routing
    - Quadrature encoder with rotary filter
- Three 16-bit timer/counters
  - One timer/counter with four output compare or input capture channels
  - Two timer/counter with two output compare or input capture channels
  - High resolution extension enabling down to 4ns PWM resolution
  - Waveform extension for control of motor, LED, lighting, H-bridge, high drives, and more
  - Fault extension for safe and deterministic handling and/or shut-down of external driver
- CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3) generator
- XMEGA Custom Logic (XCL) module with timer, counter and logic functions
  - Two 8-bit timer/counters with capture/compare and 16-bit cascade mode
  - Connected to one USART to support custom data frame length
  - Connected to I/O pins and event system to do programmable logic functions
    - MUX, AND, NAND, OR, NOR, XOR, XNOR, NOT, D-Flip-Flop, D Latch, RS Latch
- Two USARTs with full-duplex and single wire half-duplex configuration
  - Master SPI mode
  - Support custom protocols with configurable data frame length up to 256-bit
  - System wake-up from deep sleep modes when used with internal 8MHz oscillator
- One two-wire interface with dual address match (I^2C and SMBus compatible)
  - Bridge configuration for simultaneous master and slave operation
  - Up to 1MHz bus speed support
- One serial peripheral interface (SPI)
- 16-bit real time counter with separate oscillator and digital correction
- One sixteen-channel, 12-bit, 300ksps Analog to Digital Converter with:
  - Offset and gain correction
  - Averaging
  - Over-sampling and decimation
- One two-channel, 12-bit, 1Msps Digital to Analog Converter
- Two Analog Comparators with window compare function and current sources
- External interrupts on all general purpose I/O pins
- Programmable watchdog timer with separate on-chip ultra low power oscillator
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal and external clock options with PLL
  - Programmable multilevel interrupt controller
  - Five sleep modes
  - Programming and debug interface
    - PDI (Program and Debug Interface)
- I/O and Packages
  - 26 programmable I/O pins
  - 7x7mm 32-lead TQFP
  - 5x5mm 32-lead VQFN
  - 4x4mm 32-lead UQFN
- Operating Voltage
  - 1.6 – 3.6V
- Operating frequency
  - 0 – 12MHz from 1.6V
  - 0 – 32MHz from 2.7V

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]							

Ordering Code	Package[1][2][3]	Flash [Bytes]	EEPROM [Bytes]	SRAM [Bytes]	Speed [MHz]	Power supply [V]	Temp. [°C]
ATxmega32E5-AN	32A (7x7mm TQFP)	32K + 4K	1K	4K	32	1.6 – 3.6	-40 – 105
ATxmega32E5ANR[4]							
ATxmega32E5-MN	32Z (5x5mm VQFN)						
ATxmega32E5-MNR[4]							
ATxmega32E5-M4UN	32MA (4x4mm UQFN)						
ATxmega32E5-M4UNR[4]							

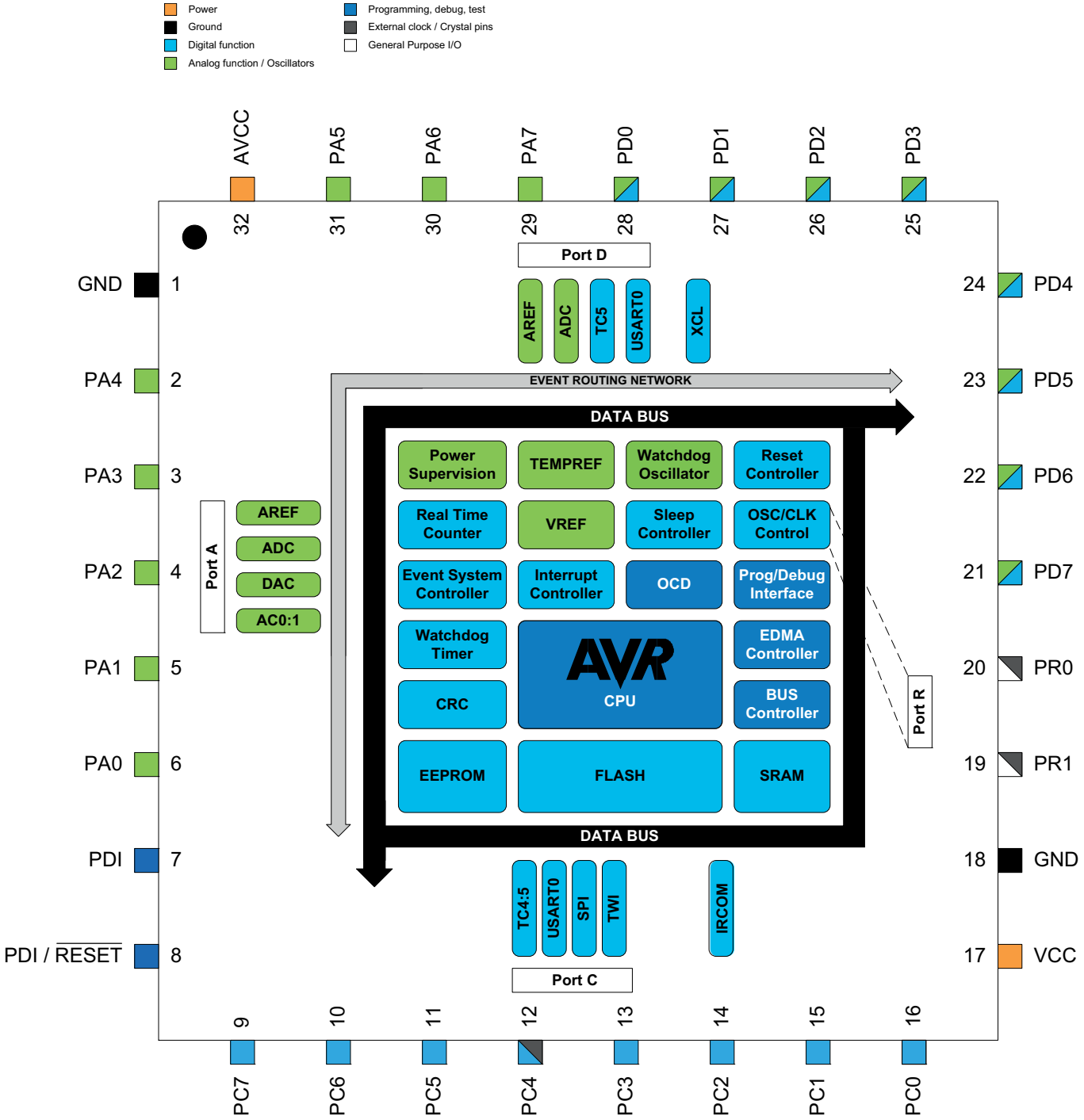
Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. For packaging information, see "Packaging Information" on page 68.
4. Tape and Reel.

Package Type	
32A	32-lead, 7x7mm body size, 1.0mm body thickness, 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)
32Z	32-lead, 0.5mm pitch, 5x5mm Very Thin quad Flat No Lead Package (VQFN) Sawn
32MA	32-lead, 0.4mm pitch, 4x4x0.60mm Ultra Thin Quad No Lead (UQFN) Package

2. Typical Applications

Board controller	Sensor control	Motor control
User interface	Industrial control	Ballast control, Inverters
Communication bridges	Battery charger	Utility metering
Appliances		

3. Pinout and Block Diagram



Notes: 1. For full details on pinout and alternate pin functions refer to "Pinout and Pin Functions" on page 57.

4. Overview

The Atmel AVR XMEGA is a family of low power, high performance, and peripheral rich 8/16-bit microcontrollers based on the AVR enhanced RISC architecture. By executing instructions in a single clock cycle, the AVR XMEGA devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

The AVR CPU combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in a single instruction, executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs many times faster than conventional single-accumulator or CISC based microcontrollers.

The AVR XMEGA E5 devices provide the following features: in-system programmable flash with read-while-write capabilities; internal EEPROM and SRAM; four-channel enhanced DMA (EDMA) controller; eight-channel event system with asynchronous event support; programmable multilevel interrupt controller; 26 general purpose I/O lines; CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3) generators; one XMEGA Custom Logic module with timer, counter and logic functions (XCL); 16-bit real-time counter (RTC) with digital correction; three flexible, 16-bit timer/counters with compare and PWM channels; two USARTs; one two-wire serial interface (TWI) allowing simultaneous master and slave; one serial peripheral interface (SPI); one sixteen-channel, 12-bit ADC with programmable gain, offset and gain correction, averaging, over-sampling and decimation; one 2-channel 12-bit DAC; two analog comparators (ACs) with window mode and current sources; programmable watchdog timer with separate internal oscillator; accurate internal oscillators with PLL and prescaler; and programmable brown-out detection.

The program and debug interface (PDI), a fast, two-pin interface for programming and debugging, is available.

The AVR XMEGA E5 devices have five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, EDMA controller, event system, interrupt controller, and all peripherals to continue functioning. The power-down mode saves the SRAM and register contents, but stops the oscillators, disabling all other functions until the next TWI, or pin-change interrupt, or reset. In power-save mode, the asynchronous real-time counter continues to run, allowing the application to maintain a timer base while the rest of the device is sleeping. In standby mode, the external crystal oscillator keeps running while the rest of the device is sleeping. This allows very fast startup from the external crystal, combined with low power consumption. In extended standby mode, both the main oscillator and the asynchronous timer continue to run. In each power save, standby or extended standby mode, the low power mode of the internal 8MHz oscillator allows very fast startup time combined with very low power consumption.

To further reduce power consumption, the peripheral clock to each individual peripheral can optionally be stopped in active mode and idle sleep mode and low power mode of the internal 8MHz oscillator can be enabled.

Atmel offers a free QTouch library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The devices are manufactured using Atmel high-density, nonvolatile memory technology. The program flash memory can be reprogrammed in-system through the PDI. A boot loader running in the device can use any interface to download the application program to the flash memory. The boot loader software in the boot flash section can continue to run. By combining an 8/16-bit RISC CPU with in-system, self-programmable flash, the AVR XMEGA is a powerful microcontroller family that provides a highly flexible and cost effective solution for many embedded applications.

All Atmel AVR XMEGA devices are supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, programmers, and evaluation kits.

5. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

5.1 Recommended Reading

- XMEGA E Manual
- XMEGA Application Notes

This device data sheet only contains part specific information with a short description of each peripheral and module. The XMEGA E Manual describes the modules and peripherals in depth. The XMEGA application notes contain example code and show applied use of the modules and peripherals.

All documentations are available from www.atmel.com/avr.

6. Capacitive Touch Sensing

The Atmel QTouch® library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR® microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression™ (AKS™) technology for unambiguous detection of key events. The QTouch library includes support for the QTouch and QMatrix acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

The Atmel QTouch library is FREE and downloadable from the Atmel website at the following location: http://www.atmel.com/tools/QTOUCHLIBRARY.aspx. For implementation details and other information, refer to the Atmel QTouch library user guide - also available for download from the Atmel website.

7. CPU

7.1 Features

- 8/16-bit, high-performance Atmel AVR RISC CPU
- 142 instructions
- Hardware multiplier
- 32x8-bit registers directly connected to the ALU
- Stack in RAM
- Stack pointer accessible in I/O memory space
- Direct addressing of up to 16MB of program memory and 16MB of data memory
- True 16/24-bit access to 16/24-bit I/O registers
- Efficient support for 8-, 16-, and 32-bit arithmetic
- Configuration change protection of system-critical features

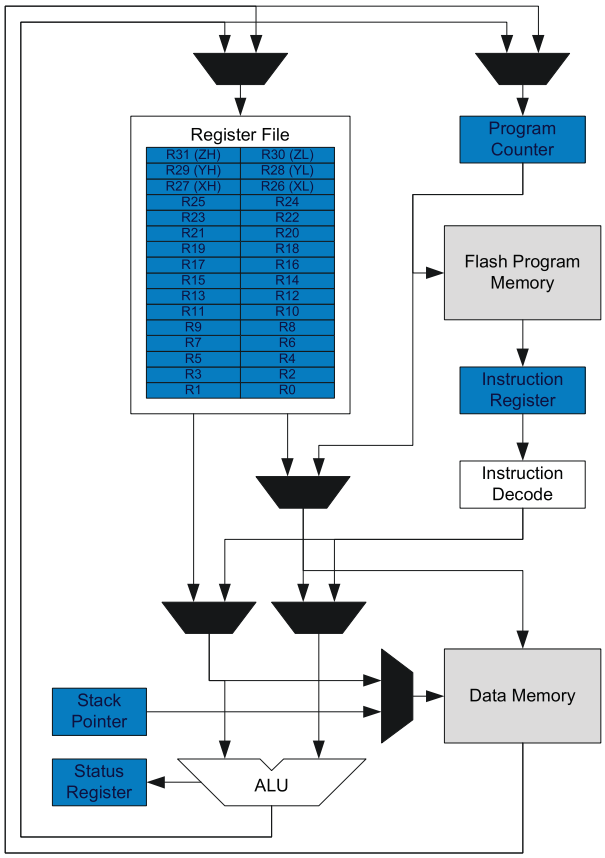
7.2 Overview

All 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 28.

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.

Figure 7-1. Block Diagram of the AVR CPU Architecture



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 in the ALU. After an arithmetic operation, the status register is updated to reflect information about the result of the operation.

The ALU is directly connected to the fast-access register file. The 32 x 8-bit general purpose working registers all have single clock cycle access time allowing single-cycle arithmetic logic unit (ALU) operation between registers or between a register and an immediate. Six of the 32 registers can be used as three 16-bit address pointers for program and data space addressing, enabling efficient address calculations.

The memory spaces are linear. The data memory space and the program memory space are two different memory spaces.

The data memory space is divided into I/O registers, SRAM, and memory mapped EEPROM.

All I/O status and control registers reside in the lowest 4KB addresses of the data memory. This is referred to as the I/O memory space. The lowest 64 addresses can be accessed directly, or as the data space locations from 0x00 to 0x3F. The rest is the extended I/O memory space, ranging from 0x0040 to 0x0FFF. I/O registers here must be accessed as data space locations using load (LD/LDS/LDD) and store (ST/STS/STD) instructions.

The SRAM holds data. Code execution from SRAM is not supported. It can easily be accessed through the five different addressing modes supported in the AVR architecture. The first SRAM address is 0x2000.

Data addresses 0x1000 to 0x1FFF are reserved for EEPROM.

The program memory is divided in two sections, the application program section and the boot program section. Both sections have dedicated lock bits for write and read/write protection. The SPM instruction that is used for self-programming of the application flash memory must reside in the boot program section. The application section contains an application table section with separate lock bits for write and read/write protection. The application table section can be used for save storing of nonvolatile data in the program memory.

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 arithmetic. 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.

7.7 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

after reset, and the initial value is the highest address of the internal SRAM. If the SP is changed, it must be set to point above address 0x2000, and it must be defined before any subroutine calls are executed or before interrupts are enabled.

During interrupts or subroutine calls, the return address is automatically pushed on the stack. The return address can be two or three bytes, depending on program memory size of the device. For devices with 128KB or less of program memory, the return address is two bytes, and hence the stack pointer is decremented/incremented by two. For devices with more than 128KB of program memory, the return address is three bytes, and hence the SP is decremented/incremented by three. The return address is popped off the stack when returning from interrupts using the RETI instruction, and from subroutine calls using the RET instruction.

The SP is decremented by one when data are pushed on the stack with the PUSH instruction, and incremented by one when data is popped off the stack using the POP instruction.

To prevent corruption when updating the stack pointer from software, a write to SPL will automatically disable interrupts for up to four instructions or until the next I/O memory write.

7.8 Register File

The register file consists of 32 x 8-bit general purpose working registers with single clock cycle access time. The register file supports the following input/output schemes:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Six of the 32 registers can be used as three 16-bit address register pointers for data space addressing, enabling efficient address calculations. One of these address pointers can also be used as an address pointer for lookup tables in flash program memory.

8. Memories

8.1 Features

- Flash program memory
  - One linear address space
  - In-system programmable
  - Self-programming and boot loader support
  - Application section for application code
  - Application table section for application code or data storage
  - Boot section for application code or bootloader code
  - Separate read/write protection lock bits for all sections
  - Built in fast CRC check of a selectable flash program memory section
- Data memory
  - One linear address space
  - Single-cycle access from CPU
  - SRAM
  - EEPROM
    - Byte and page accessible
    - Memory mapped for direct load and store
  - I/O memory
    - Configuration and status registers for all peripherals and modules
    - Four bit-accessible general purpose registers for global variables or flags
  - Bus arbitration
    - Deterministic handling of priority between CPU, EDMA controller, and other bus masters
  - Separate buses for SRAM, EEPROM, and I/O memory
    - Simultaneous bus access for CPU and EDMA controller
- Production signature row memory for factory programmed data
  - ID for each microcontroller device type
  - Serial number for each device
  - Calibration bytes for factory calibrated peripherals
- User signature row
  - One flash page in size
  - Can be read and written from software
  - Content is kept after chip erase

8.2 Overview

The Atmel AVR architecture has two main memory spaces, the program memory and the data memory. Executable code can reside only in the program memory, while data can be stored in the program memory and the data memory. The data memory includes the internal SRAM, and EEPROM for nonvolatile data storage. All memory spaces are linear and require no memory bank switching. Nonvolatile memory (NVM) spaces can be locked for further write and read/write operations. This prevents unrestricted access to the application software.

A separate memory section contains the fuse bytes. These are used for configuring important system functions, and can only be written by an external programmer.

The available memory size configurations are shown in “Ordering Information” on page 2”. In addition, each device has a Flash memory signature row for calibration data, device identification, serial number etc.

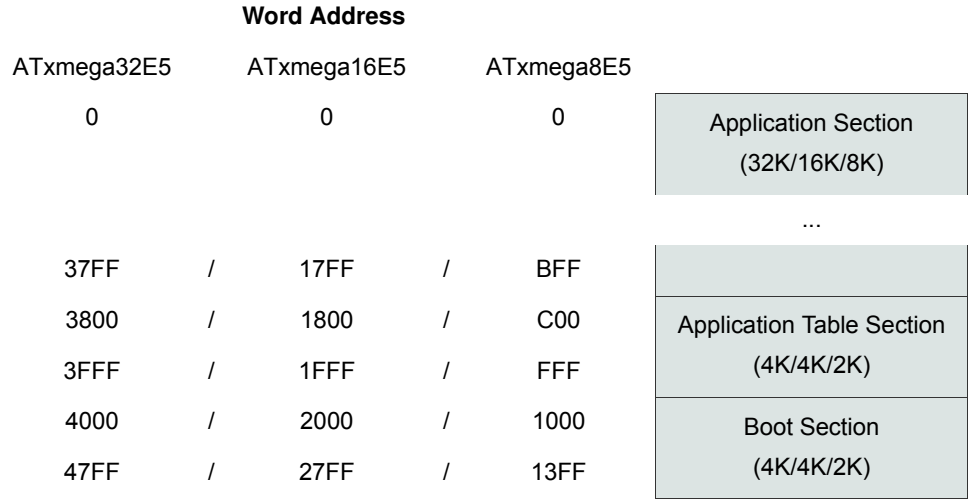
8.3 Flash Program Memory

The Atmel AVR XMEGA devices contain on-chip, in-system reprogrammable flash memory for program storage. The flash memory can be accessed for read and write from an external programmer through the PDI or from application software running in the device.

All AVR CPU instructions are 16 or 32 bits wide, and each flash location is 16 bits wide. The flash memory is organized in two main sections, the application section and the boot loader section. The sizes of the different sections are fixed, but device-dependent. These two sections have separate lock bits, and can have different levels of protection. The store program memory (SPM) instruction, which is used to write to the flash from the application software, will only operate when executed from the boot loader section.

The application section contains an application table section with separate lock settings. This enables safe storage of nonvolatile data in the program memory.

Figure 8-1. Flash Program Memory (hexadecimal address)



8.3.1 Application Section

The Application section is the section of the flash that is used for storing the executable application code. The protection level for the application section can be selected by the boot lock bits for this section. The application section can not store any boot loader code since the SPM instruction cannot be executed from the application section.

8.3.2 Application Table Section

The application table section is a part of the application section of the flash memory that can be used for storing data. The size is identical to the boot loader section. The protection level for the application table section can be selected by the boot lock bits for this section. The possibilities for different protection levels on the application section and the application table section enable safe parameter storage in the program memory. If this section is not used for data, application code can reside here.

8.3.3 Boot Loader Section

While the application section is used for storing the application code, the boot loader software must be located in the boot loader section because the SPM instruction can only initiate programming when executing from this section. When programming, the CPU is halted, waiting for the flash operation to complete. The SPM instruction can access the entire flash, including the boot loader section itself. The protection level for the boot loader section can be selected by the boot loader lock bits. If this section is not used for boot loader software, application code can be stored here.

8.3.4 Production Signature Row

The production signature row is a separate memory section for factory programmed data. It contains calibration data for functions such as oscillators and analog modules. Some of the calibration values will be automatically loaded to the corresponding module or peripheral unit during reset. Other values must be loaded from the signature row and written to

the corresponding peripheral registers from software. For details on calibration conditions, refer to “Electrical Characteristics” on page 71.

The production signature row also contains an ID that identifies each microcontroller device type and a serial number for each manufactured device. The serial number consists of the production lot number, wafer number, and wafer coordinates for the device. The device ID for the available devices is shown in Table 8-1.

The production signature row cannot be written or erased, but it can be read from application software and external programmers.

Table 8-1. Device ID Bytes for Atmel AVR XMEGA E5 Devices

Device	Device ID bytes		
	Byte 2	Byte 1	Byte 0
ATxmega32E5	4C	95	1E
ATxmega16E5	45	94	1E
ATxmega8E5	41	93	1E

8.3.5 User Signature Row

The user signature row is a separate memory section that is fully accessible (read and write) from application software and external programmers. It is one flash page in size, and is meant for static user parameter storage, such as calibration data, custom serial number, identification numbers, random number seeds, etc. This section is not erased by chip erase commands that erase the flash, and requires a dedicated erase command. This ensures parameter storage during multiple program/erase operations and on-chip debug sessions.

8.4 Fuses and Lock Bits

The fuses are used to configure important system functions, and can only be written from an external programmer. The application software can read the fuses. The fuses are used to configure reset sources such as brownout detector and watchdog, startup configuration, etc.

The lock bits are used to set protection levels for the different flash sections (i.e., if read and/or write access should be blocked). Lock bits can be written by external programmers and application software, but only to stricter protection levels. Chip erase is the only way to erase the lock bits. To ensure that flash contents are protected even during chip erase, the lock bits are erased after the rest of the flash memory has been erased.

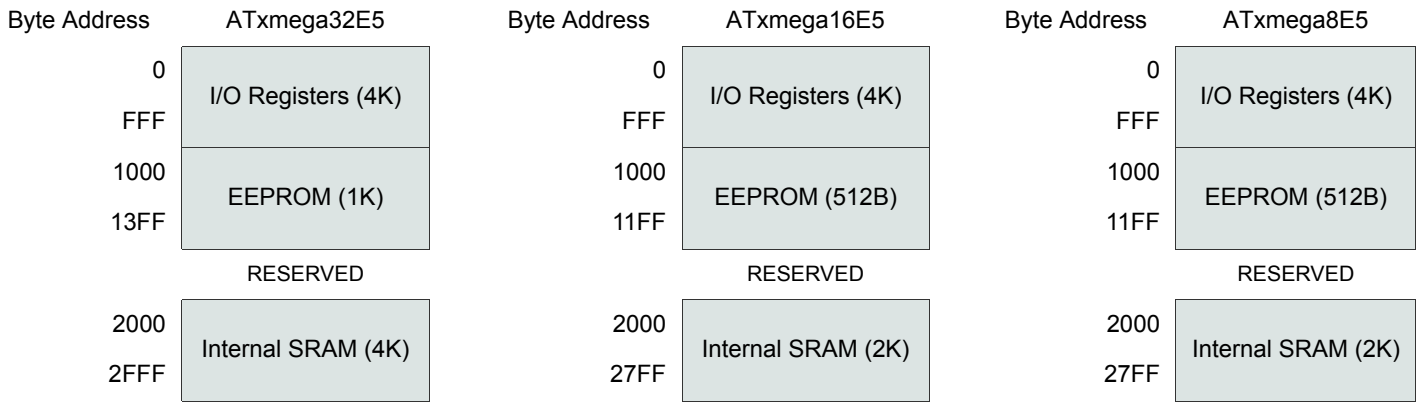
An un-programmed fuse or lock bit will have the value one, while a programmed fuse or lock bit will have the value zero.

Both fuses and lock bits are reprogrammable like the flash program memory.

8.5 Data Memory

The data memory contains the I/O memory, internal SRAM and EEPROM. The data memory is organized as one continuous memory section, see Table 8-2 on page 15. To simplify development, I/O Memory, EEPROM and SRAM will always have the same start addresses for all XMEGA devices.

Figure 8-2. Data Memory Map (hexadecimal value)



8.6 EEPROM

Atmel AVR XMEGA E5 devices have EEPROM for nonvolatile data storage. It is memory mapped and accessed in normal data space. The EEPROM supports both byte and page access. EEPROM allows highly efficient EEPROM reading and EEPROM buffer loading. When doing this, EEPROM is accessible using load and store instructions. EEPROM will always start at hexadecimal address 0x1000.

8.7 I/O Memory

The status and configuration registers for peripherals and modules, including the CPU, are addressable through I/O memory locations. All I/O locations can be accessed by the load (LD/LDS/LDD) and store (ST/STS/STD) instructions, which are used to transfer data between the 32 registers in the register file and the I/O memory. The IN and OUT instructions can address I/O memory locations in the range of 0x00 to 0x3F directly. In the address range 0x00 - 0x1F, single-cycle instructions for manipulation and checking of individual bits are available.

The I/O memory address for all peripherals and modules in XMEGA E5 is shown in the "Peripheral Module Address Map" on page 61.

8.7.1 General Purpose I/O Registers

The lowest four I/O memory addresses are reserved as general purpose I/O registers. These registers can be used for storing global variables and flags, as they are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

8.8 Data Memory and Bus Arbitration

Since the data memory is organized as three separate sets of memories, the different bus masters (CPU, EDMA controller read and EDMA controller write, etc.) can access different memory sections at the same time.

8.9 Memory Timing

Read and write access to the I/O memory takes one CPU clock cycle. A write to SRAM takes one cycle, and a read from SRAM takes two cycles. For burst read (EDMA), new data are available every cycle. EEPROM page load (write) takes one cycle, and three cycles are required for read. For burst read, new data are available every second cycle. Refer to the instruction summary for more details on instructions and instruction timing.

8.10 Device ID and Revision

Each device has a three-byte device ID. This ID identifies Atmel as the manufacturer of the device and the device type. A separate register contains the revision number of the device.

8.11 I/O Memory Protection

Some features in the device are regarded as critical for safety in some applications. Due to this, it is possible to lock the I/O register related to the clock system, the event system, and the waveform extensions. As long as the lock is enabled, all related I/O registers are locked and they cannot be written from the application software. The lock registers themselves are protected by the configuration change protection mechanism.

8.12 Flash and EEPROM Page Size

The flash program memory and EEPROM data memory are organized in pages. The pages are word accessible for the flash and byte accessible for the EEPROM.

Table 8-2 shows the Flash Program Memory organization and Program Counter (PC) size. Flash write and erase operations are performed on one page at a time, while reading the Flash is done one byte at a time. For Flash access the Z-pointer (Z[m:n]) is used for addressing. The most significant bits in the address (FPAGE) give the page number and the least significant address bits (FWORD) give the word in the page.

Table 8-2. Number of Words and Pages in the Flash

Devices	PC size	Flash size	Page Size	FWORD	FPAGE	Application		Boot	
	bits	bytes	words			Size	No. of pages	Size	No. of pages
ATxmega32E5	15	32K+4K	64	Z[6:0]	Z[14:7]	32K	256	4K	32
ATxmega16E5	14	16K+4K	64	Z[6:0]	Z[13:7]	16K	128	4K	32
ATxmega8E5	13	8K+2K	64	Z[6:0]	Z[12:7]	8K	64	2K	16

Table 8-3 shows EEPROM memory organization for the Atmel AVR XMEGA E5 devices. EEPROM write and erase operations can be performed one page or one byte at a time, while reading the EEPROM is done one byte at a time. For EEPROM access the NVM address register (ADDR[m:n]) is used for addressing. The most significant bits in the address (E2PAGE) give the page number and the least significant address bits (E2BYTE) give the byte in the page.

Table 8-3. Number of Words and Pages in the EEPROM

Devices	EEPROM	Page Size	E2BYTE	E2PAGE	No. of Pages
	Size	bytes			
ATxmega32E5	1K	32	ADDR[4:0]	ADDR[10:5]	32
ATxmega16E5	512Bytes	32	ADDR[4:0]	ADDR[10:5]	16
ATxmega8E5	512Bytes	32	ADDR[4:0]	ADDR[10:5]	16

9. EDMA – Enhanced DMA Controller

9.1 Features

- The EDMA Controller allows data transfers with minimal CPU intervention
  - from data memory to data memory
  - from data memory to peripheral
  - from peripheral to data memory
  - from peripheral to peripheral
- Four peripheral EDMA channels with separate:
  - transfer triggers
  - interrupt vectors
  - addressing modes
  - data matching
- Two peripheral channels can be combined to one standard channel with separate:
  - transfer triggers
  - interrupt vectors
  - addressing modes
  - data search
- Programmable channel priority
- From 1byte to 128KB of data in a single transaction
  - Up to 64K block transfer with repeat
  - 1 or 2 bytes burst transfers
- Multiple addressing modes
  - Static
  - Increment
- Optional reload of source and destination address at the end of each
  - Burst
  - Block
  - Transaction
- Optional Interrupt on end of transaction
- Optional connection to CRC Generator module for CRC on EDMA data

9.2 Overview

The four-channel enhanced direct memory access (EDMA) 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 EDMA channels enable up to four independent and parallel transfers.

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

Data transfers are done in continuous bursts of 1 or 2 bytes. They build block transfers of configurable size from 1 byte to 64KB. Repeat option can be used to repeat once each block transfer for single transactions up to 128KB. Source and destination addressing can be static or incremental. 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 EDMA transfers.

The four EDMA channels have individual configuration and control settings. This includes 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 EDMA controller detects an error on an EDMA channel.

To enable flexibility in transfers, channels can be interlinked so that the second takes over the transfer when the first is finished.

The EDMA controller supports extended features such as double buffering, data match for peripherals and data search for SRAM or EEPROM.

The EDMA controller supports two types of channel. Each channel type can be selected individually.

10. Event System

10.1 Features

- System for direct peripheral-to-peripheral communication and signaling
- Peripherals can directly send, receive, and react to peripheral events
  - CPU and EDMA controller independent operation
  - 100% predictable signal timing
  - Short and guaranteed response time
  - Synchronous and asynchronous event routing
- Eight event channels for up to eight different and parallel signal routing and configurations
- Events can be sent and/or used by most peripherals, clock system, and software
- Additional functions include
  - Quadrature decoder with rotary filtering
  - Digital filtering of I/O pin state with configurable filter
  - Simultaneous synchronous and asynchronous events provided to peripheral
- Works in all sleep modes

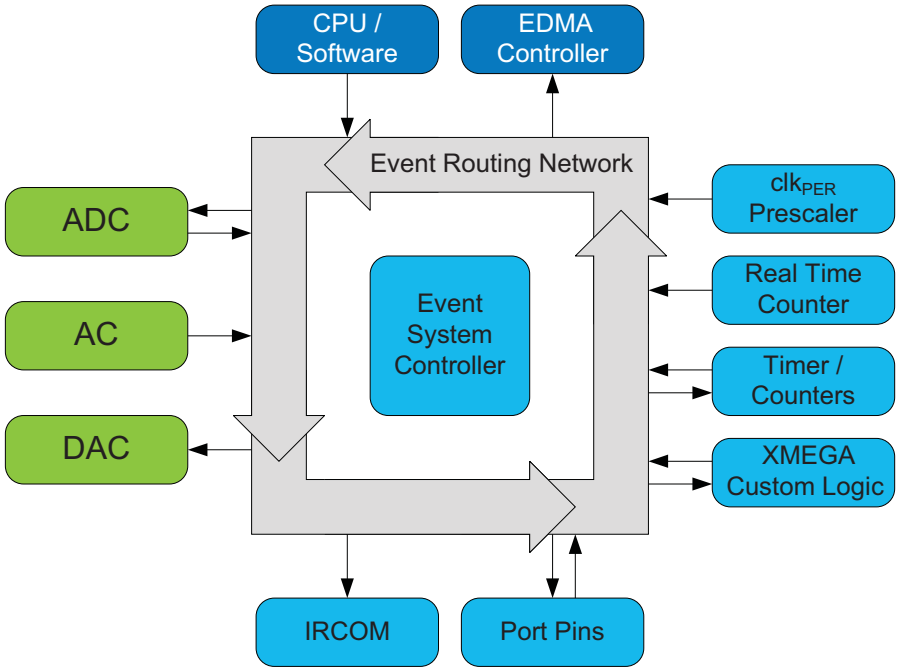
10.2 Overview

The event system enables direct peripheral-to-peripheral communication and signaling. It allows a change in one peripheral's state to automatically trigger actions in other peripherals. It is designed to provide a predictable system for short and predictable response times between peripherals. It allows for autonomous peripheral control and interaction without the use of interrupts, CPU, or EDMA controller resources, and is thus a powerful tool for reducing the complexity, size and execution time of application code. It allows for synchronized timing of actions in several peripheral modules. The event system enables also asynchronous event routing for instant actions in peripherals.

A change in a peripheral's state is referred to as an event, and usually corresponds to the peripheral's interrupt conditions. Events can be directly passed to other peripherals using a dedicated routing network called the event routing network. How events are routed and used by the peripherals is configured in software.

Figure 10-1 shows a basic diagram of all connected peripherals. The event system can directly connect together analog and digital converters, analog comparators, I/O port pins, the real-time counter, timer/counters, IR communication module (IRCOM), and XMEGA Custom Logic (programmable logic) block (XCL). It can also be used to trigger EDMA transactions (EDMA controller). Events can also be generated from software and peripheral clock.

Figure 10-1. Event System Overview and Connected Peripherals



The event routing network consists of eight software-configurable multiplexers that control how events are routed and used. These are called event channels, and allow up to eight parallel event configurations and routing. The maximum routing latency of an external event is two peripheral clock cycles due to re-synchronization, but several peripherals can directly use the asynchronous event without any clock delay. The event system works in all power sleep modes, but only asynchronous events can be routed in sleep modes where the system clock is not available.

11. System Clock and Clock options

11.1 Features

- Fast start-up time
- Safe run-time clock switching
- Internal Oscillators:
  - 32MHz run-time calibrated and tuneable oscillator
  - 8MHz calibrated oscillator with 2MHz output option and fast start-up
  - 32.768kHz calibrated oscillator
  - 32kHz Ultra Low Power (ULP) oscillator with 1kHz output
- External clock options
  - 0.4 - 16MHz Crystal Oscillator
  - 32kHz crystal oscillator with digital correction
  - External clock input in selectable pin location
- PLL with 20 - 128MHz output frequency
  - Internal and external clock options and 1 to 31x multiplication
  - Lock detector
- Clock Prescalers with 1x to 2048x division
- Fast peripheral clocks running at two and four times the CPU clock frequency
- Automatic Run-Time Calibration of the 32MHz internal oscillator
- External oscillator and PLL lock failure detection with optional non maskable interrupt

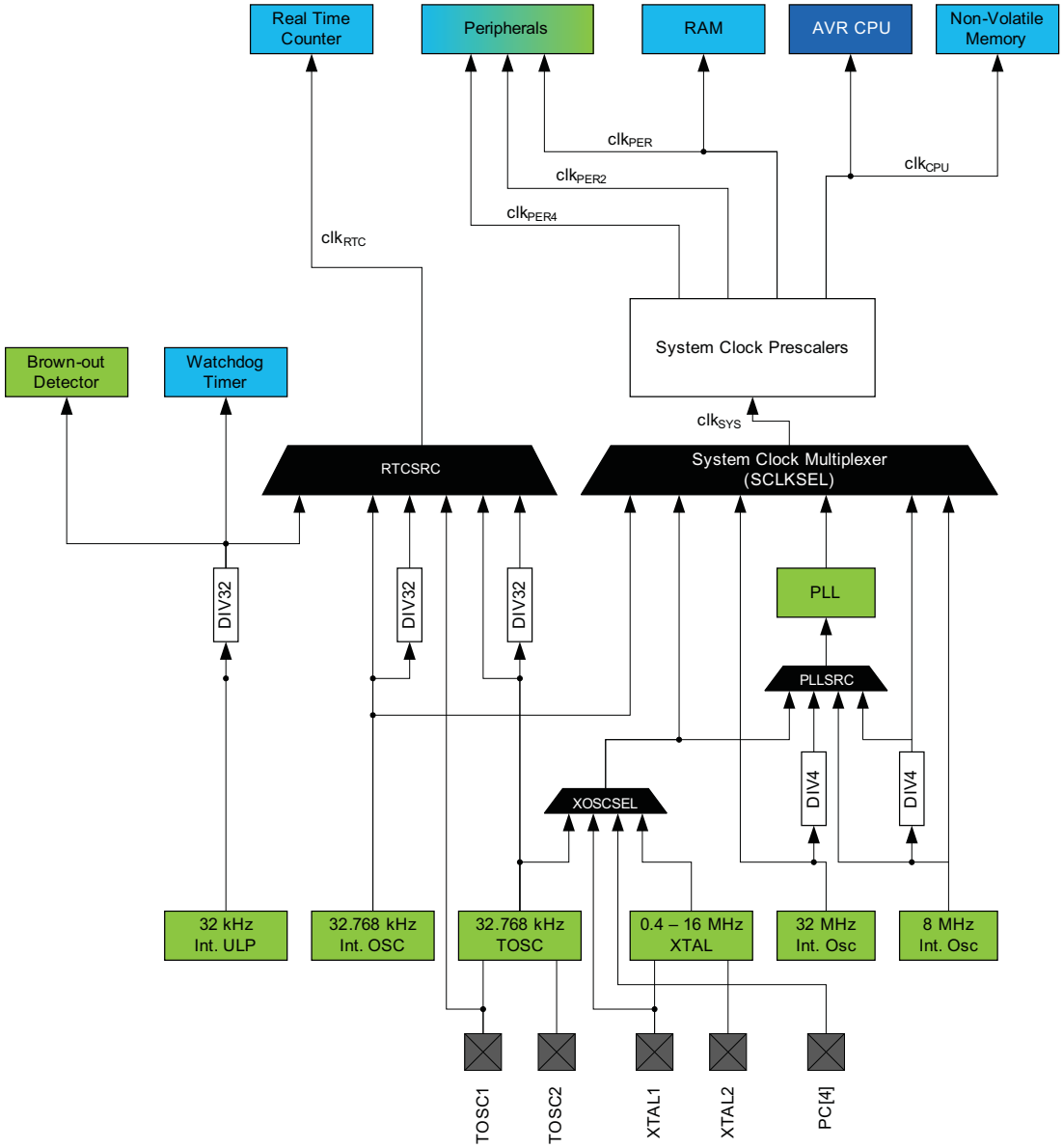
11.2 Overview

Atmel AVR XMEGA E5 devices have a flexible clock system supporting a large number of clock sources. It incorporates both accurate internal oscillators and external crystal oscillator and resonator support. A high-frequency phase locked loop (PLL) and clock prescalers can be used to generate a wide range of clock frequencies. A calibration feature (DFLL) is available, and can be used for automatic run-time calibration of the 32MHz internal oscillator to remove frequency drift over voltage and temperature. An oscillator failure monitor can be enabled to issue a nonmaskable interrupt and switch to the internal oscillator if the external oscillator or PLL fails.

When a reset occurs, all clock sources except the 32kHz ultra low power oscillator are disabled. After reset, the device will always start up running from the 2MHz output of the 8MHz internal oscillator. During normal operation, the system clock source and prescalers can be changed from software at any time.

Figure 11-1 on page 21 presents the principal clock system in the XMEGA E5 family of devices. Not all of the clocks need to be active at a given time. The clocks for the CPU and peripherals can be stopped using sleep modes and power reduction registers, as described in “Power Management and Sleep Modes” on page 23.

Figure 11-1. The Clock System, Clock Sources, and Clock Distribution



11.3 Clock Sources

The clock sources are divided in two main groups: internal oscillators and external clock sources. Most of the clock sources can be directly enabled and disabled from software, while others are automatically enabled or disabled, depending on peripheral settings. After reset, the device starts up running from the 2MHz output of the 8MHz internal oscillator. The other clock sources, DFLL and PLL, are turned off by default.

The internal oscillators do not require any external components to run. For details on characteristics and accuracy of the internal oscillators, refer to the device datasheet.

11.3.1 32kHz Ultra Low Power Internal Oscillator

This oscillator provides an approximate 32kHz clock. The 32kHz ultra low power (ULP) internal oscillator is a very low power clock source, and it is not designed for high accuracy. The oscillator employs a built-in prescaler that provides a 1kHz output. The oscillator is automatically enabled/disabled when it is used as clock source for any part of the device. This oscillator can be selected as the clock source for the RTC.

11.3.2 32.768kHz Calibrated Internal Oscillator

This oscillator provides an approximate 32.768kHz clock. It is calibrated during production to provide a default frequency close to its nominal frequency. The calibration register can also be written from software for run-time calibration of the oscillator frequency. The oscillator employs a built-in prescaler, which provides both a 32.768kHz output and a 1.024kHz output.

11.3.3 32.768kHz Crystal Oscillator

A 32.768kHz crystal oscillator can be connected between the TOSC1 and TOSC2 pins and enables a dedicated low frequency oscillator input circuit. A low power mode with reduced voltage swing on TOSC2 is available. This oscillator can be used as a clock source for the system clock and RTC, and as the DFLL reference clock.

11.3.4 0.4 - 16MHz Crystal Oscillator

This oscillator can operate in four different modes optimized for different frequency ranges, all within 0.4 - 16MHz.

11.3.5 8MHz Calibrated Internal Oscillator

The 8MHz calibrated internal oscillator is the default system clock source after reset. It is calibrated during production to provide a default frequency close to its nominal frequency. The calibration register can also be written from software for run-time calibration of the oscillator frequency. The oscillator employs a built-in prescaler, with 2MHz output. The default output frequency at start-up and after reset is 2MHz. A low power mode option can be used to enable fast system wake-up from power-save mode. In all other modes, the low power mode can be enabled to significantly reduce the power consumption of the internal oscillator.

11.3.6 32MHz Run-time Calibrated Internal Oscillator

The 32MHz run-time calibrated internal oscillator is a high-frequency oscillator. It is calibrated during production to provide a default frequency close to its nominal frequency. A digital frequency looked loop (DFLL) can be enabled for automatic run-time calibration of the oscillator to compensate for temperature and voltage drift and optimize the oscillator accuracy. This oscillator can also be adjusted and calibrated to any frequency between 30 and 55MHz.

11.3.7 External Clock Sources

The XTAL1 and XTAL2 pins can be used to drive an external oscillator, either a quartz crystal or a ceramic resonator. XTAL1 or pin 4 of port C (PC4) can be used as input for an external clock signal. The TOSC1 and TOSC2 pins are dedicated to driving a 32.768kHz crystal oscillator.

11.3.8 PLL with 1x-31x Multiplication Factor

The built-in phase locked loop (PLL) can be used to generate a high-frequency system clock. The PLL has a user-selectable multiplication factor of from 1 to 31. In combination with the prescalers, this gives a wide range of output frequencies from all clock sources.

12. Power Management and Sleep Modes

12.1 Features

- Power management for adjusting power consumption and functions
- Five 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 EDMA 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.

12.3.3 Power-save Mode

Power-save mode is identical to power down, with one exception. If the real-time counter (RTC) is enabled, it will keep running during sleep, and the device can also wake up from either an RTC overflow or compare match interrupt. Low power mode option of 8MHz internal oscillator enables instant oscillator wake-up time. This reduces the MCU wake-up time or enables the MCU wake-up from UART bus.

12.3.4 Standby Mode

Standby mode is identical to power down, with the exception that the enabled system clock sources are kept running while the CPU, peripheral, and RTC clocks are stopped. This reduces the wake-up time. The low power option of 8MHz internal oscillator can be enabled to further reduce the power consumption.

12.3.5 Extended Standby Mode

Extended standby mode is identical to power-save mode, with the exception that the enabled system clock sources are kept running while the CPU and peripheral clocks are stopped. This reduces the wake-up time. The low power option of 8MHz internal oscillator can be enabled to further reduce the power consumption.

13. System Control and Reset

13.1 Features

- Reset the microcontroller and set it to initial state when a reset source goes active
- Multiple reset sources that cover different situations
  - Power-on reset
  - External reset
  - Watchdog reset
  - Brownout reset
  - PDI reset
  - Software reset
- Asynchronous operation
  - No running system clock in the device is required for reset
- Reset status register for reading the reset source from the application code

13.2 Overview

The reset system issues a microcontroller reset and sets the device to its initial state. This is for situations where operation should not start or continue, such as when the microcontroller operates below its power supply rating. If a reset source goes active, the device enters and is kept in reset until all reset sources have released their reset. The I/O pins are immediately tri-stated. The program counter is set to the reset vector location, and all I/O registers are set to their initial values. The SRAM content is kept. However, if the device accesses the SRAM when a reset occurs, the content of the accessed location can not be guaranteed.

After reset is released from all reset sources, the default oscillator is started and calibrated before the device starts running from the reset vector address. By default, this is the lowest program memory address, 0, but it is possible to move the reset vector to the lowest address in the boot section.

The reset functionality is asynchronous, and so no running system clock is required to reset the device. The software reset feature makes it possible to issue a controlled system reset from the user software.

The reset status register has individual status flags for each reset source. It is cleared at power-on reset, and shows which sources have issued a reset since the last power-on.

13.3 Reset Sequence

A reset request from any reset source will immediately reset the device and keep it in reset as long as the request is active. When all reset requests are released, the device will go through three stages before the device starts running again:

- Reset counter delay
- Oscillator startup
- Oscillator calibration

If another reset requests occurs during this process, the reset sequence will start over again.

13.4 Reset Sources

13.4.1 Power-on Reset

A power-on reset (POR) is generated by an on-chip detection circuit. The POR is activated when the V_{CC} rises and reaches the POR threshold voltage (V_{POT}), and this will start the reset sequence.

The POR is also activated to power down the device properly when the V_{CC} falls and drops below the V_{POT} level. The V_{POT} level is higher for falling V_{CC} than for rising V_{CC}. Consult the datasheet for POR characteristics data.