



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





Atmel AVR XMEGA B1 8- and 16-bit Microcontroller

ATxmega128B1 / ATxmega64B1

Features

- High-performance, low-power Atmel® AVR® XMEGA® 8- and 16-bit Microcontroller
- Nonvolatile program and data memories
 - 64K - 128KBytes of in-system self-programmable flash
 - 4K - 8KBytes boot section
 - 2KBytes EEPROM
 - 4K - 8KBytes internal SRAM
- Peripheral features
 - Two-channel DMA controller
 - Four-channel event system
 - Three 16-bit timer/counters
 - Two timer/counters with four output compare or input capture channels
 - One timer/counter with two output compare or input capture channels
 - High resolution extensions one timer/counter
 - Advanced waveform extension (AWeX) on one timer/counter
 - Split mode on two timer/counters
 - One USB device interface
 - USB 2.0 full speed (12Mbps) and low speed (1.5Mbps) device compliant
 - 32 endpoints with full configuration flexibility
 - Two USARTs with IrDA support for one USART
 - AES and DES crypto engine
 - CRC-16 (CRC-CCITT) and CRC-32 (IEEE® 802.3) generator
 - One two-wire interface with dual address match (I²C and SMBus compatible)
 - One serial peripheral interface (SPI)
 - 16-bit Real Time Counter (RTC) with separate oscillator
 - Liquid Crystal Display (LCD)
 - Up to 4x40 segment driver
 - Built in contrast control
 - ASCII character mapping
 - Flexible SWAP of segment and common terminals buses
 - Two eight-channel, 12-bit, three hundred thousand SPS Analog to Digital Converters
 - Four Analog Comparators with window compare function, and current source feature
 - 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 interfaces
 - JTAG (IEEE 1149.1 Compliant) interface, including boundary scan
 - PDI (Program and Debug Interface)
- I/O and packages
 - 53 Programmable I/O pins
 - 100-lead TQFP, 100-ball VFBGA

- Operating voltage
 - 1.6 – 3.6V
- Operating frequency
 - 0 – 12MHz from 1.6V
 - 0 – 32MHz from 2.7V

1. Ordering Information

| Ordering Code | Flash [Bytes] | EEPROM [Bytes] | SRAM [Bytes] | Speed [MHz] | Power supply | Package ⁽¹⁾⁽²⁾⁽³⁾ | Temp. |
|---------------------------------|------------------|-------------------|-----------------|----------------|-----------------|------------------------------|---------------|
| ATxmega128B1-AU | 128K + 8K | 2K | 8K | 32 | 1.6 - 3.6V | 100A | -40°C - 85°C |
| ATxmega128B1-AUR ⁽⁴⁾ | | | | | | 7A1 | |
| ATxmega128B1-CU | | | | | | | |
| ATxmega128B1-CUR ⁽⁴⁾ | | | | | | 100A | -40°C - 105°C |
| ATxmega128B1-AN | | | | | | | |
| ATxmega128B1-ANR ⁽⁴⁾ | | | | | | | |
| ATxmega64B1-AU | 64K + 4K | 2K | 4K | 32 | 1.6 - 3.6V | 100A | -40°C - 85°C |
| ATxmega64B1-AUR ⁽⁴⁾ | | | | | | 7A1 | |
| ATxmega64B1-CU | | | | | | | |
| ATxmega64B1-CUR ⁽⁴⁾ | | | | | | 100A | -40°C - 105°C |
| ATxmega64B1-AN | | | | | | | |
| ATxmega64B1-ANR ⁽⁴⁾ | | | | | | | |

- Notes:
1. This device can also be supplied in wafer form. 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 ["Errata" on page 136](#).
 4. Tape and Reel.

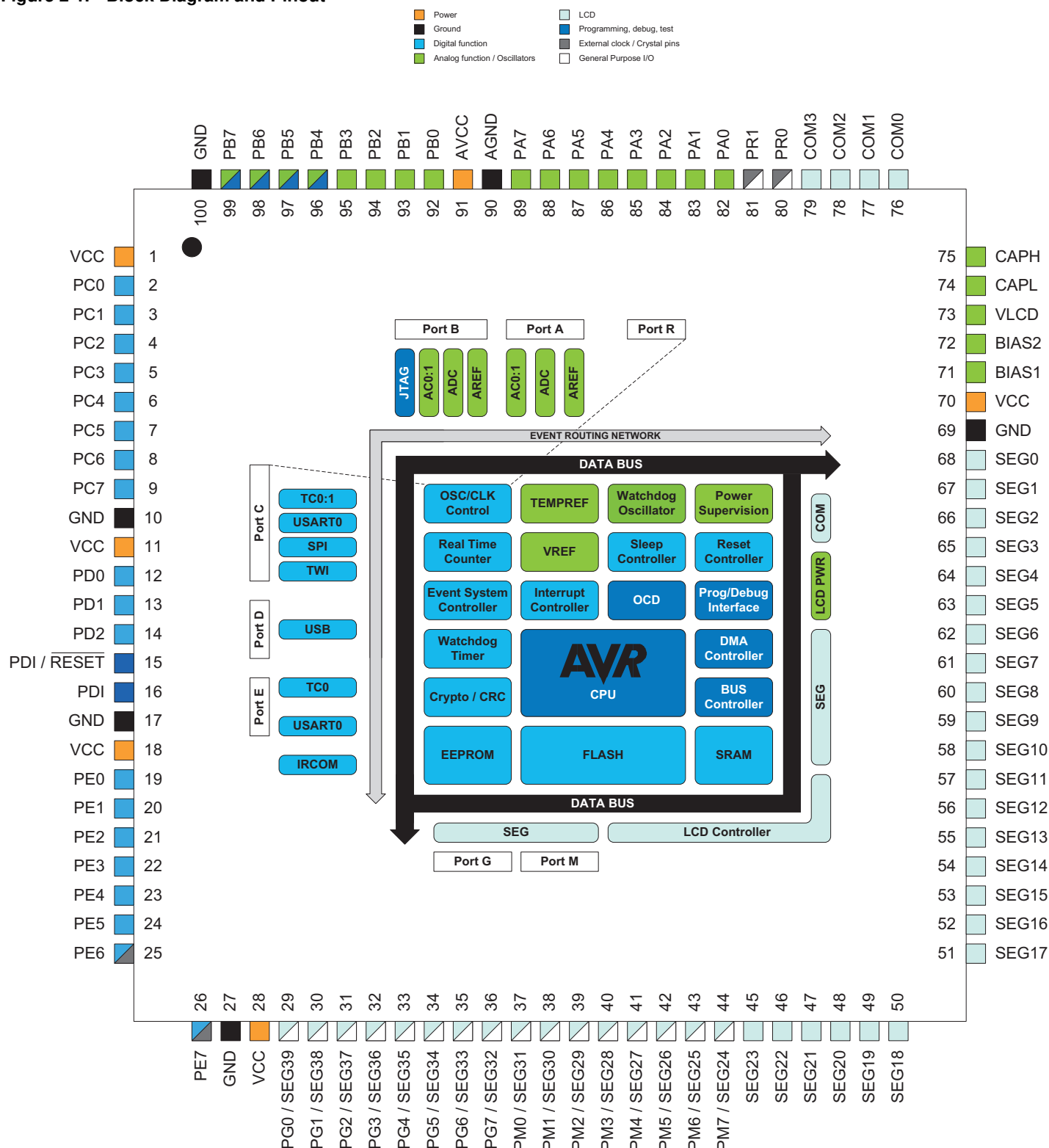
| Package Type | |
|--------------|--|
| 100A | 100-lead, 14 x 14mm body size, 1.0mm body thickness, 0.5mm lead pitch, thin profile plastic quad flat package (TQFP) |
| 7A1 | 100-ball (10x10 array), 7 x 7 x 1.0mm body, ball pitch 0.65mm, very thin fine-pitch ball grid array (VFBGA) |

Typical Applications

| | | |
|--------------------|------------------|--------------------------------|
| Industrial control | Climate control | Low power battery applications |
| Factory automation | RF and ZigBee® | Power tools |
| Building control | USB connectivity | HVAC |
| Board control | Sensor control | Utility metering |
| White goods | Optical | Medical applications |

2. Pinout/Block Diagram

Figure 2-1. Block Diagram and Pinout



Note: 1. For full details on pinout and alternate pin functions refer to "Pinout and Pin Functions" on page 54.

Figure 2-2. VFBGA Pinout

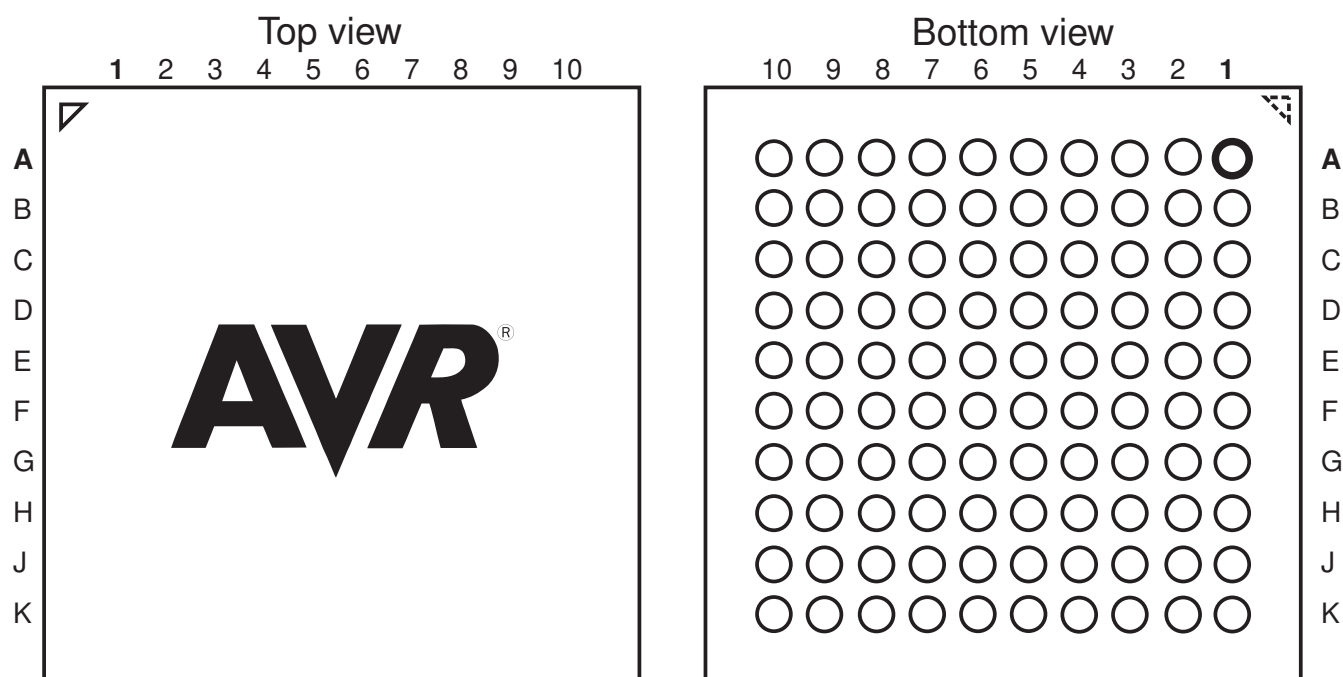


Table 2-1. VFBGA Pinout

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|-----|---------------|---------------|---------------|---------------|---------------|---------------|-------|-------|-------|
| A | PC0 | VCC | PB6 | PB2 | AVCC | PA5 | PA1 | PR1 | COM2 | CAPH |
| B | PC3 | GND | PB7 | PB4 | AGND | PA4 | PA0 | PR0 | COM1 | CAPL |
| C | PC5 | PC4 | PC1 | PB5 | PA7 | PA3 | COM3 | COM0 | BIAS2 | BIAS1 |
| D | VCC | GND | PD0 | PC2 | PB0 | PA6 | SEG0 | VLCD | GND | VCC |
| E | PD2 | PDI/ RESET | PD1 | PC6 | PB1 | PA2 | SEG1 | SEG4 | SEG3 | SEG2 |
| F | VCC | GND | PDI | PC7 | PB3 | PM2/ SEG29 | SEG10 | SEG7 | SEG6 | SEG5 |
| G | PE2 | PE1 | PE3 | PE0 | PE4 | SEG23 | SEG15 | SEG13 | SEG9 | SEG8 |
| H | PE5 | PE6 | PG1/ SEG38 | PG4/ SEG35 | PG7/ SEG32 | PM5/ SEG26 | SEG21 | SEG18 | SEG12 | SEG11 |
| J | PE7 | PG0/ SEG39 | PG3/ SEG36 | PG6/ SEG33 | PM1/ SEG30 | PM4/ SEG27 | PM7/ SEG24 | SEG20 | SEG16 | SEG14 |
| K | GND | VCC | PG2/ SEG37 | PG5/ SEG34 | PM0/ SEG31 | PM3/ SEG28 | PM6/ SEG25 | SEG22 | SEG19 | SEG17 |

3. 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 Atmel 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 Atmel AVR XMEGA B1 devices provide the following features: in-system programmable flash with read-while-write capabilities; internal EEPROM and SRAM; two-channel DMA controller, four-channel event system and programmable multilevel interrupt controller, 53 general purpose I/O lines, real-time counter (RTC); Liquid Crystal Display supporting up to 4x40 segment driver, ASCII character mapping and built-in contrast control (LCD); three flexible, 16-bit timer/counters with compare and PWM channels; two USARTs; one two-wire serial interface (TWI); one full speed USB 2.0 interface; one serial peripheral interface (SPI); AES and DES cryptographic engine; two 8-channel, 12-bit ADCs with programmable gain; four analog comparators (ACs) with window mode; 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 devices also have an IEEE std. 1149.1 compliant JTAG interface, and this can also be used for on-chip debug and programming.

The ATx devices have five software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, DMA 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, USB resume, 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 power-save mode, the LCD controller is allowed to refresh data to the panel. 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, and the LCD controller is allowed to refresh data to the panel. To further reduce power consumption, the peripheral clock to each individual peripheral can optionally be stopped in active mode and idle sleep mode.

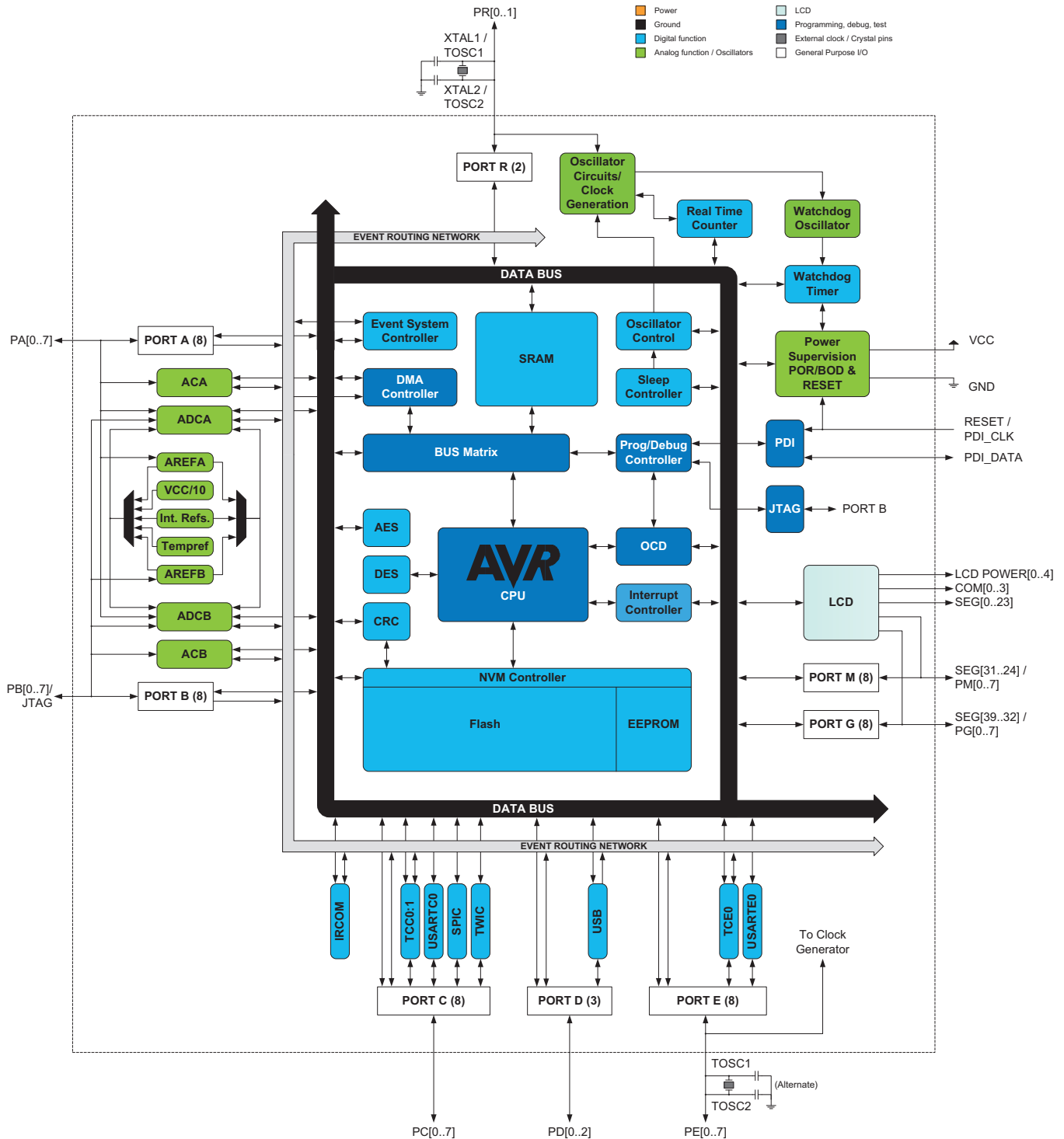
Atmel offers a free QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers.

The devices are manufactured using Atmel high-density, nonvolatile memory technology. The program flash memory can be reprogrammed in-system through the PDI or JTAG interfaces. 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 will continue to run while the application flash section is updated, providing true read-while-write operation. By combining an 8/16-bit RISC CPU with in-system, self-programmable flash, the Atmel XMEGA B1 is a powerful microcontroller family that provides a highly flexible and cost effective solution for many embedded applications.

The atmel AVR ATx 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.

3.1 Block Diagram

Figure 3-1. XMEGA B1 Block Diagram



4. Resources

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

4.1 Recommended Reading

- XMEGA B Manual
- XMEGA Application Notes

This device data sheet only contains part specific information with a short description of each peripheral and module. The XMEGA B 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.

5. 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 QTouch library is FREE and downloadable from the Atmel website at the following location: www.atmel.com/qtouchlibrary. For implementation details and other information, refer to the [Atmel QTouch Library User Guide](#) - also available for download from the Atmel website.

6. AVR CPU

6.1 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

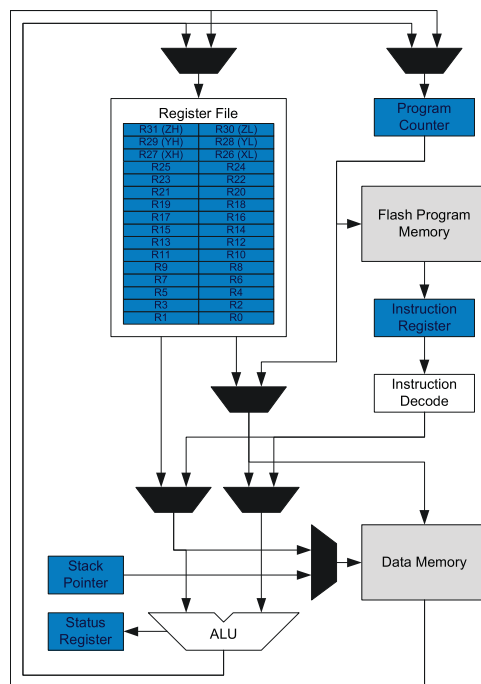
6.2 Overview

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 28.

6.3 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 pipe lining. 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 6-1. 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 and SRAM. In addition, the EEPROM can be memory mapped in the data memory.

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 memory mapping of 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.

6.4 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 are supported, and the instruction set allows for efficient implementation of 32-bit arithmetic. The hardware multiplier supports signed and unsigned multiplication and fractional format.

6.4.1 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.

6.5 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.

6.6 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.

6.7 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.

6.8 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.

7. Memories

7.1 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
 - Optional memory mapping 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
 - Safe and deterministic handling of priority between CPU, DMA controller, and other bus masters
 - Separate buses for SRAM, EEPROM and I/O memory
 - Simultaneous bus access for CPU and DMA 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

7.2 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.

7.3 Flash Program Memory

The Atmel AVR XMEGA devices contain on-chip, in-system re-programmable 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 7-1. Flash Program Memory (hexadecimal address)

| Word Address | | |
|--------------|-------------|--|
| ATxmega128B1 | ATxmega64B1 | |
| 0 | 0 | Application section (bytes) (128K/64K) |
| | | ... |
| FFFF | 77FF | |
| F000 | 7800 | Application table section (bytes) (8K/4K) |
| FFFF | 7FFF | |
| 10000 | 8000 | Boot section (bytes) (8K/4K) |
| 10FFF | 87FF | |

7.3.1 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.

7.3.2 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.

7.3.3 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. 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.

7.3.4 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 69.

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 7-1 on page 14](#).

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

Table 7-1. Device ID Bytes for XMEGA B1 Devices

| Device | Device ID bytes | | |
|--------------|-----------------|--------|--------|
| | Byte 2 | Byte 1 | Byte 0 |
| ATxmega64B1 | 52 | 96 | 1E |
| ATxmega128B1 | 4D | 97 | 1E |

7.3.5 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.

7.4 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, JTAG enable, and JTAG user ID.

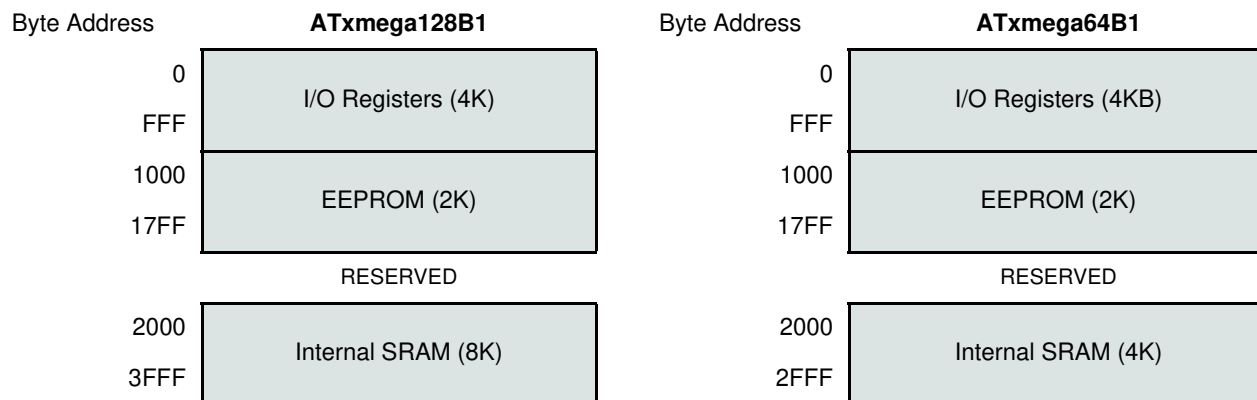
The lock bits are used to set protection levels for the different flash sections (i.e., if read and/or write access should be blocked). Lock bits can be written by external programmers and application software, but only to stricter protection levels. Chip erase is the only way to erase the lock bits. 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 unprogrammed 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 re-programmable like the flash program memory.

7.5 Data Memory

The data memory contains the I/O memory, internal SRAM and optionally memory mapped EEPROM. The data memory is organized as one continuous memory section, see [Figure 7-2 on page 15](#). To simplify development, I/O Memory, EEPROM, and SRAM will always have the same start addresses for all XMEGA devices.

Figure 7-2. Data Memory Map (hexadecimal address)



7.6 EEPROM

Atmel AVR XMEGA B1 devices have EEPROM for nonvolatile data storage. It is either addressable in a separate data space (default) or memory mapped and accessed in normal data space. The EEPROM supports both byte and page access. Memory mapped EEPROM allows highly efficient EEPROM reading and EEPROM buffer loading. When doing this, EEPROM is accessible using load and store instructions. Memory mapped EEPROM will always start at hexadecimal address 0x1000.

7.7 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 B1 is shown in the [“Peripheral Module Address Map” on page 60](#).

7.7.1 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.

7.8 Data Memory and Bus Arbitration

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

7.9 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 (DMA), 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.

7.10 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.

7.11 JTAG Disable

It is possible to disable the JTAG interface from the application software. This will prevent all external JTAG access to the device until the next device reset or until JTAG is enabled again from the application software. As long as JTAG is disabled, the I/O pins required for JTAG can be used as normal I/O pins.

7.12 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 advanced waveform extensions. As long as the lock is enabled, all related I/O registers are locked and they can not be written from the application software. The lock registers themselves are protected by the configuration change protection mechanism.

7.13 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 7-2 on page 16 shows the Flash Program Memory organization. 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 7-2. Number of Words and Pages in the Flash

| Devices | PC size | Flash | Page size | FWORD | FPAGE | Application | | Boot | |
|--------------|---------|-----------|-----------|--------|---------|-------------|--------------|------|--------------|
| | bits | bytes | words | | | Size | No. of pages | Size | No. of pages |
| ATxmega64B1 | 16 | 64K + 4K | 128 | Z[7:1] | Z[16:8] | 64K | 256 | 4K | 16 |
| ATxmega128B1 | 17 | 128K + 8K | 128 | Z[8:1] | Z[17:9] | 128K | 512 | 8K | 32 |

Table 7-3 on page 16 shows EEPROM memory organization for the XMEGA B1 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 7-3. Number of Bytes and Pages in the EEPROM

| Devices | EEPROM | Page size | E2BYTE | E2PAGE | No. of pages |
|--------------|--------|-----------|-----------|------------|--------------|
| | Size | Bytes | | | |
| ATxmega64B1 | 2K | 32 | ADDR[4:0] | ADDR[10:5] | 64 |
| ATxmega128B1 | 2K | 32 | ADDR[4:0] | ADDR[10:5] | 64 |

8. DMAC – Direct Memory Access Controller

8.1 Features

- Allows high speed 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
- Two DMA channels with separate
 - transfer triggers
 - interrupt vectors
 - addressing modes
- Programmable channel priority
- From 1 byte to 16MB of data in a single transaction
 - Up to 64KB block transfers with repeat
 - 1, 2, 4, or 8 byte burst transfers
- Multiple addressing modes
 - Static
 - Incremental
 - Decremental
- Optional reload of source and destination addresses at the end of each
 - Burst
 - Block
 - Transaction
- Optional interrupt on end of transaction
- Optional connection to CRC generator for CRC on DMA data

8.2 Overview

The two-channel direct memory access (DMA) controller can transfer data between memories and peripherals, and thus off-load 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 two 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, the channels can be interlinked so that the second takes over the transfer when the first is finished, and vice versa.

9. Event System

9.1 Features

- System for direct peripheral-to-peripheral communication and signaling
- Peripherals can directly send, receive, and react to peripheral events
 - CPU and DMA controller independent operation
 - 100% predictable signal timing
 - Short and guaranteed response time
- Four event channels for up to four different and parallel signal routings and configurations
- Events can be sent and/or used by most peripherals, clock system, and software
- Additional functions include
 - Quadrature decoders
 - Digital filtering of I/O pin state
- Works in active mode and idle sleep mode

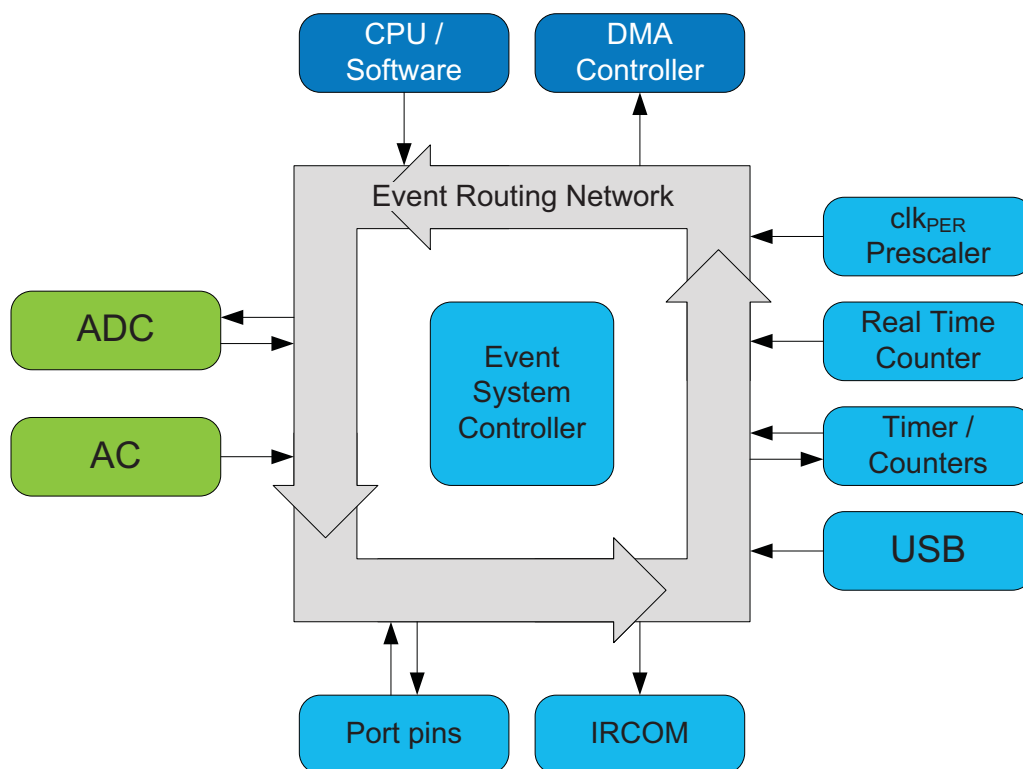
9.2 Overview

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 DMA controller resources, and is thus a powerful tool for reducing the complexity, size and execution time of application code. It also allows for synchronized timing of actions in several peripheral modules.

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 9-1 on page 19](#) 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 USB interface. It can also be used to trigger DMA transactions (DMA controller). Events can also be generated from software and the peripheral clock.

Figure 9-1. Event System Overview and Connected Peripherals



The event routing network consists of four software-configurable multiplexers that control how events are routed and used. These are called event channels, and allow for up to four parallel event configurations and routings. The maximum routing latency is two peripheral clock cycles. The event system works in both active mode and idle sleep mode.

10. System Clock and Clock Options

10.1 Features

- Fast start-up time
- Safe run-time clock switching
- Internal oscillators:
 - 32MHz run-time calibrated oscillator
 - 2MHz run-time calibrated oscillator
 - 32.768kHz calibrated oscillator
 - 32kHz Ultra Low Power (ULP) oscillator with 1kHz output
- External clock options
 - 0.4MHz - 16MHz crystal oscillator
 - 32.768kHz crystal oscillator
 - External clock
- PLL with 20MHz - 128MHz output frequency
 - Internal and external clock options and 1x to 31x multiplication
 - Lock detector
- Clock prescalers with 1x to 2048x division
- Fast peripheral clocks running at two and four times the CPU clock
- Automatic run-time calibration of internal oscillators
- External oscillator and PLL lock failure detection with optional non-maskable interrupt

10.2 Overview

Atmel AVR XMEGA 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 internal oscillators to remove frequency drift over voltage and temperature. An oscillator failure monitor can be enabled to issue a non-maskable 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 internal oscillator. During normal operation, the system clock source and prescalers can be changed from software at any time.

[Figure 10-1 on page 21](#) presents the principal clock system in the XMEGA B1 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](#).

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 internal oscillator. The other clock sources, DFLLs, and PLL, are turned off by default.

32kHz Ultra Low Power Internal Oscillator

Atmel

10.3.2 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. This oscillator can be used as a clock source for the system clock, RTC and LCD, and as the DFLL reference clock.

10.3.3 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, RTC and LCD, and as the DFLL reference clock.

10.3.4 0.4 - 16MHz Crystal Oscillator

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

10.3.5 2MHz Run-time Calibrated Internal Oscillator

The 2MHz run-time 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. A DFLL can be enabled for automatic run-time calibration of the oscillator to compensate for temperature and voltage drift and optimize the oscillator accuracy.

10.3.6 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 locked 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 30MHz and 55MHz. The production signature row contains 48MHz calibration values intended used when the oscillator is used a full-speed USB clock source.

10.3.7 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 each pin of port C can be used as input for an external clock signal. The TOSC1 and TOSC2 pins is dedicated to driving a 32.768kHz crystal oscillator.

10.3.8 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.

11. Power Management and Sleep Modes

11.1 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

11.2 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.

11.3 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.

11.3.1 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.

11.3.2 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, asynchronous port interrupts, and the USB resume interrupt.

11.3.3 Power-save Mode

Power-save mode is identical to power down, with two exceptions:

1. 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.
2. If the liquid crystal display controller (LCD) is enabled, it will keep running during sleep, and the device can wake up from LCD frame completed interrupt.

11.3.4 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, RTC, and LCD clocks are stopped. This reduces the wake-up time.

11.3.5 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.

12. System Control and Reset

12.1 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

12.2 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.

12.3 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.

12.4 Reset Sources

12.4.1 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.