**Atmel AVR**

[](http://en.wikipedia.org/wiki/File:AVR-logo.png)

The **AVR** is a [modified Harvard architecture](http://en.wikipedia.org/wiki/Modified_Harvard_architecture) [8-bit](http://en.wikipedia.org/wiki/8-bit) [RISC](http://en.wikipedia.org/wiki/Reduced_instruction_set_computer) single chip [microcontroller](http://en.wikipedia.org/wiki/Microcontroller) which was developed by [Atmel](http://en.wikipedia.org/wiki/Atmel) in 1996. The AVR was one of the first microcontroller families to use on-chip [flash memory](http://en.wikipedia.org/wiki/Flash_memory) for program storage, as opposed to [one-time programmable ROM](http://en.wikipedia.org/wiki/Programmable_read-only_memory), [EPROM](http://en.wikipedia.org/wiki/EPROM), or [EEPROM](http://en.wikipedia.org/wiki/EEPROM) used by other microcontrollers at the time.

**AVR** was developed in the year 1996 by Atmel Corporation. The architecture of **AVR** was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for **Alf-Egil Bogen Vegard Wollan RISC microcontroller**, also known as **A**dvanced **V**irtual **R**ISC. The AT90S8515 was the first microcontroller which was based on **AVR architecture** however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

**AVR microcontrollers** are available in three categories:

1.      **TinyAVR** – Less memory, small size, suitable only for simpler applications

2.      **MegaAVR** – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.

3.      **XmegaAVR** – Used commercially for complex applications, which require large program memory and high speed.

The following table compares the above mentioned AVR series of microcontrollers:

|  |  |  |  |
| --- | --- | --- | --- |
| **Series Name** | **Pins** | **Flash Memory** | **Special Feature** |
| **TinyAVR** | 6-32 | 0.5-8 KB | Small in size |
| **MegaAVR** | 28-100 | 4-256KB | Extended peripherals |
| **XmegaAVR** | 44-100 | 16-384KB | DMA , Event System included |

**Device overview**

The AVR is a modified Harvard architecture machine where program and data is stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.

**Basic families**

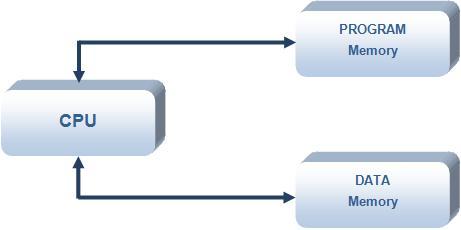
AVRs are generally classified into five broad groups:

* **tinyAVR** — the ATtiny series
  + 0.5–8 kB program memory
  + 6–32-pin package
  + Limited peripheral set
* **megaAVR** — the ATmega series
  + 4–256 kB program memory
  + 28–100-pin package
  + Extended instruction set (Multiply instructions and instructions for handling larger program memories)
  + Extensive peripheral set
* **XMEGA** — the ATxmega series
  + 16–384 kB program memory
  + 44–64–100-pin package (A4, A3, A1)
  + Extended performance features, such as DMA, "Event System", and cryptography support.
  + Extensive peripheral set with DACs
* **Application-specific AVR**
  + megaAVRs with special features not found on the other members of the AVR family, such as LCD controller, [USB](http://en.wikipedia.org/wiki/Universal_Serial_Bus) controller, advanced PWM, CAN etc.
* **FPSLIC™ (AVR with FPGA)**
  + [FPGA](http://en.wikipedia.org/wiki/Field-programmable_gate_array) 5K to 40K gates
  + SRAM for the AVR program code, unlike all other AVRs
  + AVR core can run at up to 50 MHz [[5]](http://en.wikipedia.org/wiki/Atmel_AVR#cite_note-4)
* **32-bit AVRs**

**Architecture of AVR**

The AVR microcontrollers are based on the advanced RISC architecture and consist of 32 x 8-bit general purpose working registers. Within one single clock cycle, AVR can take inputs from two general purpose registers and put them to ALU for carrying out the requested operation, and transfer back the result to an arbitrary register. The ALU can perform arithmetic as well as logical operations over the inputs from the register or between the register and a constant. Single register operations like taking a complement can also be executed in ALU. We can see that AVR does not have any register like accumulator as in 8051 family of microcontrollers; the operations can be performed between any of the registers and can be stored in either of them.

AVR follows Harvard Architecture format in which the processor is equipped with separate memories and buses for Program and the Data information. Here while an instruction is being executed, the next instruction is pre-fetched from the program memory.



Since AVR can perform single cycle execution, it means that AVR can execute 1 million instructions per second if cycle frequency is 1MHz. The higher is the operating frequency of the controller, the higher will be its processing speed. We need to optimize the power consumption with processing speed and hence need to select the operating frequency accordingly.

There are two flavors for Atmega16 microcontroller:

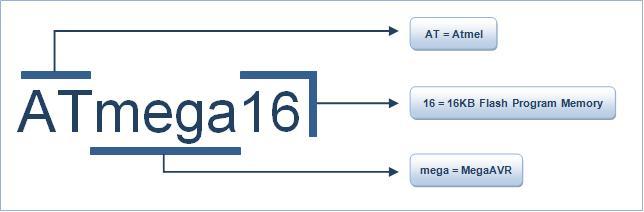
1.   **Atmega16**:- Operating frequency range is 0 – 16 MHz.

2.      **Atmega16L**:- Operating frequency range is 0 – 8 MHz.

If we are using a crystal of 8 MHz = 8 x 106 Hertz = 8 Million cycles, then AVR can execute 8 million instructions.

**Naming Convention.!**

The **AT** refers to Atmel the manufacturer, **Mega** means that the microcontroller belong to MegaAVR category, **16** signifies the memory of the controller, which is 16KB.

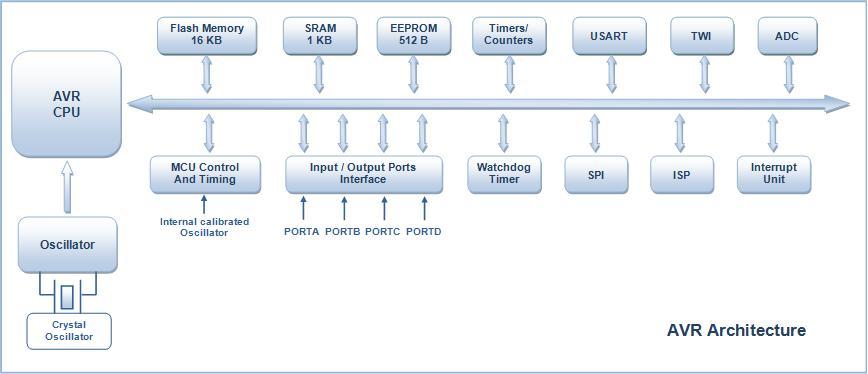


**Architecture Diagram: Atmega16**

Following points explain the building blocks of **Atmega16 architecture**:

·           **I/O Ports:** Atmega16 has four (PORTA, PORTB, PORTC and PORTD) **8-bit** input-output ports.

·           **Internal Calibrated Oscillator:** Atmega16 is equipped with an internal oscillator for driving its clock. By default Atmega16 is set to operate at internal calibrated oscillator of 1 MHz. The maximum frequency of internal oscillator is 8Mhz. Alternatively, ATmega16 can be operated using an external crystal oscillator with a maximum frequency of 16MHz. In this case you need to modify the fuse bits. (Fuse Bits will be explained in a separate tutorial).



·           **ADC Interface:** Atmega16 is equipped with an 8 channel [ADC](http://www.engineersgarage.com/embedded/avr-microcontroller-projects/adc-circuit) (**Analog to Digital Converter**) with a resolution of 10-bits. ADC reads the analog input for e.g., a sensor input and converts it into digital information which is understandable by the microcontroller.

·           **Timers/Counters:** Atmega16 consists of two 8-bit and one 16-bit timer/counter. Timers are useful for generating precision actions for e.g., creating time delays between two operations.

·           **Watchdog Timer:** Watchdog timer is present with internal oscillator. Watchdog timer continuously monitors and resets the controller if the code gets stuck at any execution action for more than a defined time interval.

·           **Interrupts:** Atmega16 consists of 21 interrupt sources out of which four are external. The remaining are internal interrupts which support the peripherals like USART, ADC, Timers etc.

·           **USART:** **Universal Synchronous and Asynchronous Receiver and Transmitter** interface is available for interfacing with external device capable of communicating serially (data transmission bit by bit).

·           **General Purpose Registers:** Atmega16 is equipped with 32 general purpose registers which are coupled directly with the Arithmetic Logical Unit (ALU) of CPU.

·           **Memory:** Atmega16 consist of three different memory sections:

1.         **Flash EEPROM**: Flash EEPROM or simple flash memory is used to store the program dumped or burnt by the user on to the microcontroller. It can be easily erased electrically as a single unit. Flash memory is non-volatile i.e., it retains the program even if the power is cut-off. Atmega16 is available with 16KB of in system programmable Flash EEPROM.

2.        **Byte Addressable EEPROM**: This is also a nonvolatile memory used to store data like values of certain variables. Atmega16 has 512 bytes of EEPROM, this memory can be useful for storing the lock code if we are designing an application like electronic door lock.

3.        **SRAM**: Static Random Access Memory, this is the volatile memory of microcontroller i.e., data is lost as soon as power is turned off. Atmega16 is equipped with 1KB of internal SRAM. A small portion of SRAM is set aside for general purpose registers used by CPU and some for the peripheral subsystems of the microcontroller.

·           **ISP:** AVR family of controllers have **In System Programmable** Flash Memory which can be programmed without removing the IC from the circuit, ISP allows to reprogram the controller while it is in the application circuit.

·          **SPI:** **Serial Peripheral Interface**, SPI port is used for serial communication between two devices on a common clock source. The data transmission rate of SPI is more than that of USART.

·           **TWI: Two Wire Interface** (TWI) can be used to set up a network of devices, many devices can be connected over TWI interface forming a network, the devices can simultaneously transmit and receive and have their own unique address.

·           **DAC:** Atmega16 is also equipped with a **Digital to Analog Converter** (DAC) interface which can be used for reverse action performed by ADC. DAC can be used when there is a need of converting a digital signal to analog signal.

**Various microcontroller of MegaAVR series:**

ATmega8 and Atmega32 are other members of MegaAVR series controllers. They are quite similar to ATmega16 in architecture. Low power version MegaAVR controllers are also available in markets. The following table shows the comparison between different members of MegaAVR family:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Part Name** | **ROM** | **RAM** | **EEPROM** | **I/0 Pins** | **Timer** | **Interrupts** | **Operation Voltage** | **Operating frequency** | **Packaging** |
| **ATmega8** | 8KB | 1KB | 512B | 23 | 3 | 19 | 4.5-5.5 V | 0-16 MHz | 28 |
| **ATmega8L** | 8KB | 1KB | 512B | 23 | 3 | 19 | 2.7-5.5 V | 0-8 MHz | 28 |
| **ATmega16** | 16KB | 1KB | 512B | 32 | 3 | 21 | 4.5-5.5 V | 0-16 MHz | 40 |
| **ATmega16L** | 16KB | 1KB | 512B | 32 | 3 | 21 | 2.7-5.5 V | 0-8 MHz | 40 |
| **ATmega32** | 32KB | 2KB | 1KB | 32 | 3 | 21 | 4.5-5.5 V | 0-16 MHz | 40 |
| **ATmega32L** | 32KB | 2KB | 1KB | 32 | 3 | 21 | 2.7-5.5 V | 0-8 MHz | 40 |