**ABSTRACT:**

Battery Monitoring System for electric vehicle has been developed for monitoring battery level using LCD. It was designed on a low cost microcontroller. Voltage, current aretransferred to microcontroller, and then data of battery is transferred using embedded technology on display. Battery monitoring is displayed on LCD display. The monitoring system was able to show real-time data of voltageand parameters in LCD display simultaneously.

**Introduction:**

Batteries are widely used on many applications as energy source on mobile and portable electronic devices. The battery monitoring system (BMS) is one of main component on electric vehicle. It is used for monitoring voltage, current and temperature of battery during charging or discharging process. There are many researches remote monitoring systems. The microcontroller technology based on based on LCD (LCD) has been developed for monitoring battery on electric vehicle. Other research about microcontroller technology based on Embedded technology has been done and applied for battery management system. Embedded technonologys a short-range microcontroller technology that operates in inbuilt oscillator

Block Diagram:

Vehicle switch

Voltage sensor

Microcontroller

LCD

motor

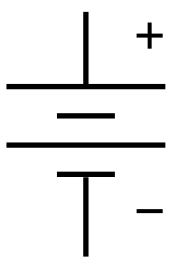
battery

**BATTERY**

An electrical **battery** is one or more [electrochemical cells](http://en.wikipedia.org/wiki/Electrochemical_cell) that convert stored chemical [energy](http://en.wikipedia.org/wiki/Energy) into electrical energy.[.](http://en.wikipedia.org/wiki/Battery_%28electricity%29#cite_note-Webster-0) Since the invention of the first battery (or "[voltaic pile](http://en.wikipedia.org/wiki/Voltaic_pile)") in 1800 by [Alessandro Volta](http://en.wikipedia.org/wiki/Alessandro_Volta) and especially since the technically improved [Daniel cell](http://en.wikipedia.org/wiki/Daniell_cell) in 1836, batteries have become a common power source for many household and industrial applications. According to a 2005 estimate, the worldwide battery industry generates [US$](http://en.wikipedia.org/wiki/United_States_dollar)48 [billion](http://en.wikipedia.org/wiki/1000000000_%28number%29) in sales each year, with 6% annual growth.

There are two types of batteries: [primary batteries](http://en.wikipedia.org/wiki/Primary_battery) (disposable batteries), which are designed to be used once and discarded, and [secondary batteries](http://en.wikipedia.org/wiki/Secondary_battery) (rechargeable batteries), which are designed to be recharged and used multiple times. Batteries come in many sizes, from miniature cells used to power [hearing aids](http://en.wikipedia.org/wiki/Hearing_aid) and wristwatches to battery banks the size of rooms that provide standby power for [telephone exchanges](http://en.wikipedia.org/wiki/Telephone_exchange) and computer [data centers](http://en.wikipedia.org/wiki/Data_center).

|  |
| --- |
|  |

[](http://en.wikipedia.org/wiki/File:Battery_symbol2.svg)

The [symbol](http://en.wikipedia.org/wiki/Electronic_symbol) for a battery in a [circuit diagram](http://en.wikipedia.org/wiki/Circuit_diagram). It originated as a schematic drawing of the earliest type of battery, a voltaic pile. In strict terms, a battery is a collection of multiple electrochemical cells, but in popular usage *battery* often refers to a single cell.[.](http://en.wikipedia.org/wiki/Battery_%28electricity%29#cite_note-Webster-0) For example, a 1.5-volt AAA battery is a single 1.5-volt cell, and a 9-volt battery has six 1.5-volt cells in series. The first electrochemical cell was developed by the [Italian](http://en.wikipedia.org/wiki/Italians) physicist [Alessandro Volta](http://en.wikipedia.org/wiki/Alessandro_Volta) in 1792, and in 1800 he invented the first battery, a "pile" of many cells in series.

**Principle of operation**

A voltaic cell for demonstration purposes. In this example the two half-cells are linked by a [salt bridge](http://en.wikipedia.org/wiki/Salt_bridge) separator that permits the transfer of ions, but not water molecules.

A battery is a device that converts chemical energy directly to electrical energy. It consists of a number of voltaic cells; each voltaic cell consists of two [half-cells](http://en.wikipedia.org/wiki/Half-cell) connected in series by a conductive electrolyte containing anions and cations. One half-cell includes electrolyte and the electrode to which [anions](http://en.wikipedia.org/wiki/Ion#Ions) (negatively charged ions) migrate, i.e., the [anode](http://en.wikipedia.org/wiki/Anode) or negative electrode; the other half-cell includes electrolyte and the electrode to which [cations](http://en.wikipedia.org/wiki/Ion#Ions) (positively charged ions) migrate, i.e., the [cathode](http://en.wikipedia.org/wiki/Cathode) or positive electrode. In the [redox](http://en.wikipedia.org/wiki/Redox) reaction that powers the battery, cations are reduced (electrons are added) at the cathode, while anions are oxidized (electrons are removed) at the anode. The electrodes do not touch each other but are electrically connected by the [electrolyte](http://en.wikipedia.org/wiki/Electrolyte). Some cells use two half-cells with different electrolytes. A separator between half-cells allows ions to flow, but prevents mixing of the electrolytes.

Each half-cell has an electromotive force (or emf), determined by its ability to drive electric current from the interior to the exterior of the cell. The net emf of the cell is the difference between the emfs of its half-cells, as first recognized by Volta. Therefore, if the electrodes have emfs \mathcal{E}_1and\mathcal{E}_2, then the net emf is\mathcal{E}_{2}-\mathcal{E}_{1}; in other words, the net emf is the difference between the [reduction potentials](http://en.wikipedia.org/wiki/Reduction_potential) of the [half-reactions](http://en.wikipedia.org/wiki/Half-reaction).

**Battery capacity and discharging:**

A device to check battery voltage.

A battery's **capacity** is the amount of [electric charge](http://en.wikipedia.org/wiki/Electric_charge) it can store. The more electrolyte and electrode material there is in the cell the greater the capacity of the cell. A small cell has less capacity than a larger cell with the same chemistry, and they develop the same open-circuit voltage.

Because of the chemical reactions within the cells, the capacity of a battery depends on the discharge conditions such as the magnitude of the current (which may vary with time), the allowable terminal voltage of the battery, temperature, and other factors. The available capacity of a battery depends upon the rate at which it is discharged. If a battery is discharged at a relatively high rate, the available capacity will be lower than expected.

The capacity printed on a battery is usually the product of 20 hours multiplied by the constant current that a new battery can supply for 20 hours at 68 F° (20 C°), down to a specified terminal voltage per cell. A battery rated at 100 A·h will deliver 5 A over a 20-hour period at [room temperature](http://en.wikipedia.org/wiki/Room_temperature). However, if discharged at 50 A, it will have a lower capacity.

The relationship between current, discharge time, and capacity for a lead acid battery is approximated (over a certain range of current values)

By [Peukert's law](http://en.wikipedia.org/wiki/Peukert%27s_law):

t = \frac {Q_P} {I^k}

Where

Q_P Is the capacity when discharged at a rate of 1 amp.

I Is the current drawn from battery ([A](http://en.wikipedia.org/wiki/Amperes)).

t Is the amount of time (in hours) that a battery can sustain.

k Is a constant around 1.3

For low values of *I* internal self-discharge must be included.

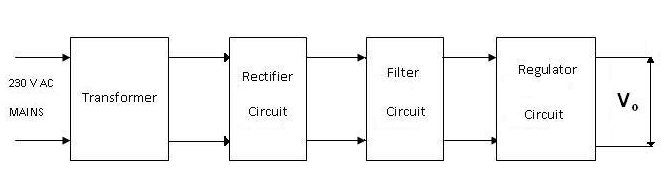
Internal energy losses and limited rate of diffusion of ions through the electrolyte cause the [efficiency](http://en.wikipedia.org/wiki/Efficient_energy_use) of a real battery to vary at different discharge rates. When discharging at low rate, the battery's energy is delivered more efficiently than at higher discharge rates, but if the rate is very low, it will partly self-discharge during the long time of operation, again lowering its efficiency.

Installing batteries with different A·h ratings will not affect the operation of a device (except for the time it will work for) rated for a specific voltage unless the load limits of the battery are exceeded. High-drain loads such as [digital cameras](http://en.wikipedia.org/wiki/Digital_camera) can result in delivery of less total energy, as happens with alkaline batteries.[[31]](http://en.wikipedia.org/wiki/Battery_%28electricity%29#cite_note-bu50-30) For example, a battery rated at 2000 mA·h for a 10- or 20-hour discharge would not sustain a current of 1 A for a full two hours as its stated capacity implies.

Block diagram of power supply

Block diagram of power supply

**Block diagram of power supply**

POWER SUPPLY

The given block diagram includes following:

**Transformer:** A transformer is an electro-magnetic static device, which transfers electrical energy from one circuit to another, either at the same voltage or at different voltage but at the same frequency.

**Rectifier:** The function of the rectifier is to convert AC to DC current or voltage. Usually in the rectifier circuit full wave bridge rectifier is used.

**Filter:** The Filter is used to remove the pulsated AC. A filter circuit uses capacitor and inductor. The function of the capacitor is to block the DC voltage and bypass the AC voltage. The function of the inductor is to block the AC voltage and bypass the DC voltage.

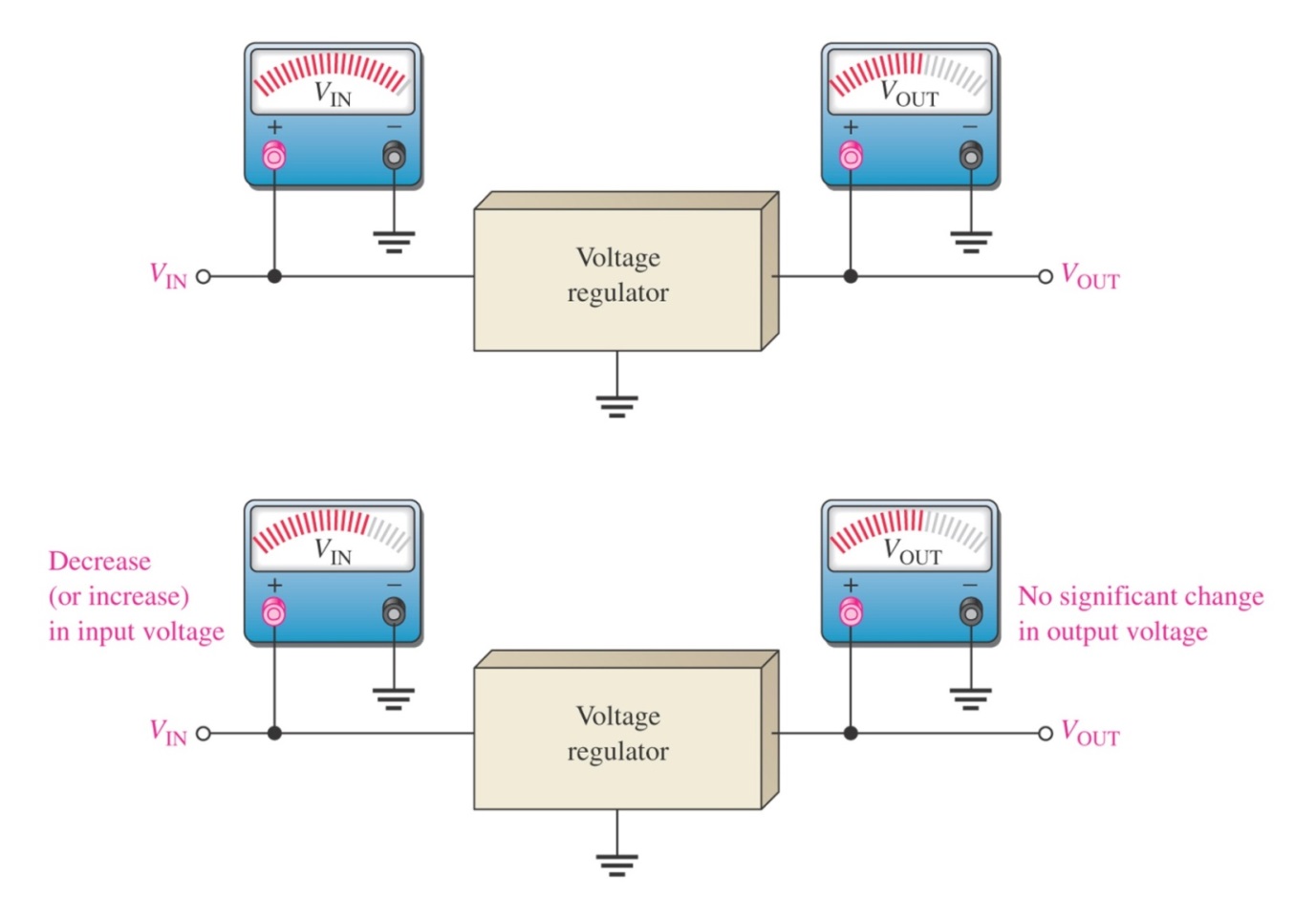
**Voltage Regulator:** Voltage regulator constitutes an indispensable part of the power supply section of any electronic systems. The main advantage of the regulator ICs is that it regulates or maintains the output constant, in spite of the variation in the input supply.

**Voltage Regulation**

* Two basic categories of voltage regulation are:
  + line regulation
  + load regulation

**Line Regulation**

* The purpose of line regulation is to maintain a nearly constant output voltage when the input voltage varies.

****

**Load regulation:**

* A change in load current (due to a varying RL) has practically no effect on the output voltage of a regulator (within certain limits)
* The purpose of load regulation is to maintain a nearly constant output voltage when the load varies.
* Load regulation can be defined as the percentage change in the output voltage from no-load (NL) to full-load (FL).



Where:

* VNL = the no-load output voltage
* VFL = the full-load output voltage

**Types of Regulator:**

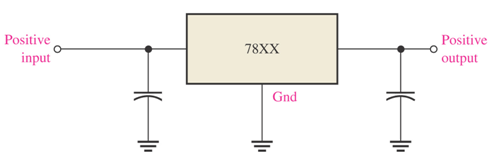
* Fundamental classes of voltage regulators are **linear regulators** and **switching regulators**.
* Two basic types of linear regulator are the **series regulator** and the **shunt regulator**.
* The series regulator is connected in **series** with the load and the shunt regulator is connected in **parallel** with the load.

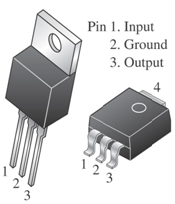
**IC Voltage Regulators:**

* Regulation circuits in integrated circuit form are widely used.
* Their operation is no different but they are treated as a single device with associated components.
* These are generally three terminal devices that provide a positive or negative output.
* Some types have variable voltage outputs.
* A typical 7800 series voltage regulator is used for positive voltages.
* The 7900 series are negative voltage regulators.
* These voltage regulators when used with heatsinks can safely produce current values of 1A and greater.
* The capacitors act as line filtration.
* Several types of both linear (series and shunt) and switching regulators are available in integrated circuit (IC) form.
* Single IC regulators contain the circuitry for:
* Reference source
* Comparator amplifier
* Control device
* Overload protection
* Generally, the linear regulators are three-terminal devices that provides either positive or negative output voltages that can be either fixed or adjustable

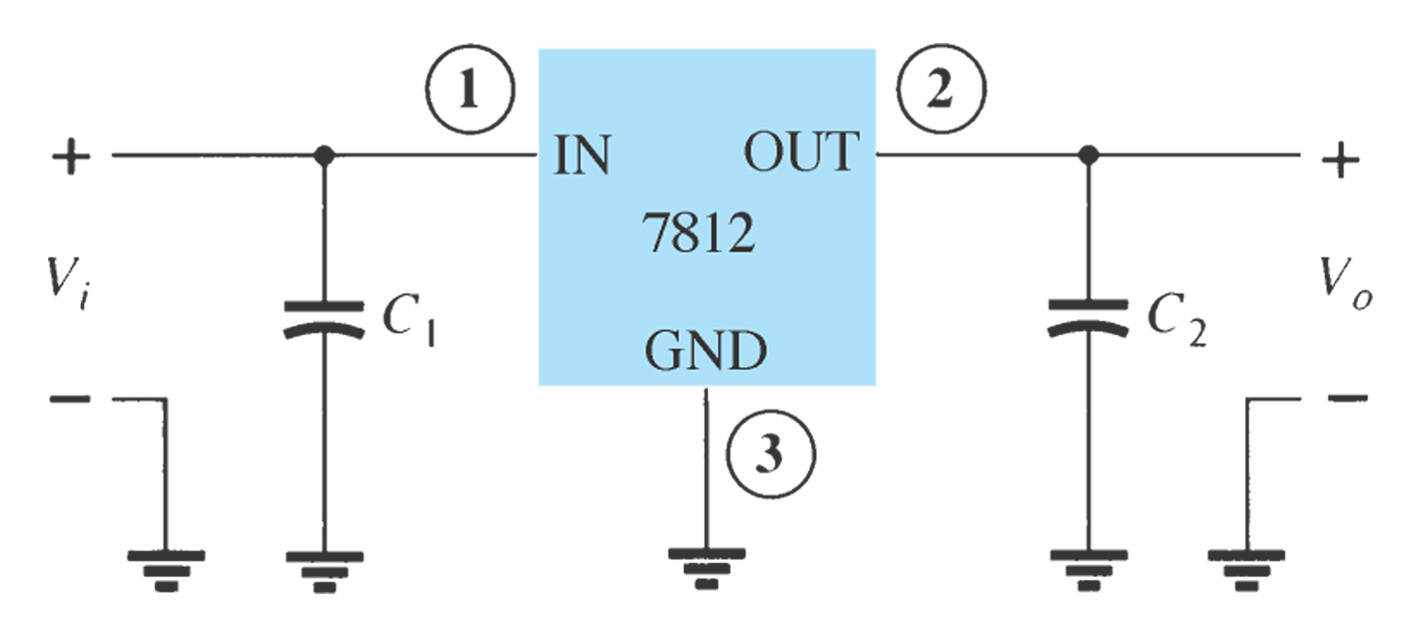
**Fixed Voltage Regulator:**

* The fixed voltage regulator has an unregulated dc input voltage Vi applied to one input terminal, a regulated output dc voltage Vo from a second terminal, and the third terminal connected to ground.
* The series 78XX regulators are the three-terminal devices that provide a fixed positive output voltage.





* An unregulated input voltage Vi is filtered by a capacitor C1 and connected to the IC’s IN terminal.
* The IC’s OUT terminal provides a regulated +12 V, which is filtered by capacitor C2.
* The third IC terminal is connected to ground (GND



|  |  |  |
| --- | --- | --- |
| **IC Part** | **Output Voltage (V)** | **Minimum Vi (V)** |
| 7805 | +5 | +7.3 |
| 7806 | +6 | +8.3 |
| 7808 | +8 | +10.5 |
| 7810 | +10 | +12.5 |
| 7812 | +12 | +14.5 |
| 7815 | +15 | +17.7 |
| 7818 | +18 | +21.0 |
| 7824 | +24 | +27.1 |

* Voltage regulators keep a constant dc output despite input voltage or load changes.
* The two basic categories of voltage regulators are linear and switching.
* The two types of linear voltage regulators are series and shunt.
* The three types of switching are step-up, step-down, and inverting.
* Switching regulators are more efficient than linear making them ideal for low voltage high current applications.
* IC regulators are available with fixed positive or negative output voltages or variable negative or positive output voltages.
* Both linear and switching type regulators are available in IC form.
* Current capacity of a voltage regulator can be increased with an external pass transistor.

**MICROCONTROLLER – ATMEGA328:**

**Features**

• High-performance, Low-power AVR® 8-bit Microcontroller

• Advanced RISC Architecture

– 130 Powerful Instructions – Most Single-clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 16 MIPS Throughput at 16 MHz

– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory segments

– 8K Bytes of In-System Self-programmable Flash program memory

– 512 Bytes EEPROM

– 1K Byte Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C(1)

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

• Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture

Mode

– Real Time Counter with Separate Oscillator

– Three PWM Channels

– 8-channel ADC in TQFP and QFN/MLF package

Eight Channels 10-bit Accuracy

– 6-channel ADC in PDIP package

Six Channels 10-bit Accuracy

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby

• I/O and Packages

– 23 Programmable I/O Lines

– 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF

• Operating Voltages

– 2.7 - 5.5V (ATMEGA328L)

– 4.5 - 5.5V (ATMEGA328)

• Speed Grades

– 0 - 8 MHz (ATMEGA328L)

– 0 - 16 MHz (ATMEGA328)

• Power Consumption at 4 MHz, 3V, 25°C

– Active: 3.6 mA

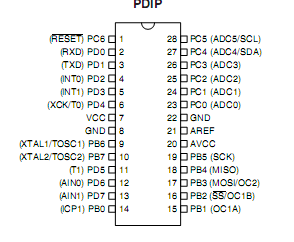
– Idle Mode: 1.0 mA

– Power-down Mode: 0.5 µA

**AVR Core:**

The AVR core combines a rich instruction set with 32 general purpose working registers. All the32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATMEGA328 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM; Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. The device is manufactured using Atmel’s high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. 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-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the AtmelATMEGA328 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications. The ATMEGA328 AVR is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

**Pin Configurations:**



**Disclaimer:**

* Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology.
* Min and Max values will be available after the device is characterized.

**Pin Descriptions:**

**VCC**  Digital supply voltage.

**GND**  Ground.

**Port B (PB7..PB0)**

**XTAL1/XTAL2/TOSC1/TOSC2**

* Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).
* The Port B output buffers have symmetrical drive characteristics with both high sink and source capability.
* As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated.
* The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.
* Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
* Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.
* If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

**Port C (PC5..PC0)**

* Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).
* The Port C output buffers have symmetrical drive characteristics with both high sink and source capability.
* As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated.
* The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**PC6/RESET**

* If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin.
* Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.
* If the RSTDISBL Fuse is un programmed, PC6 is used as a Reset input.
* A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running.
* Shorter pulses are not guaranteed to generate a Reset.

**Port D (PD7..PD0)**

* Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).
* The Port D output buffers have symmetrical drive characteristics with both high sink and source capability.
* As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated.
* The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**RESET**

* Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running.
* Shorter pulses are not guaranteed to generate a reset.

**AVCC**

* AVCC is the supply voltage pin for the A/D Converter, Port C (3..0), and ADC (7..6). It should be externally connected to VCC, even if the ADC is not used.
* If the ADC is used, it should be connected to VCC through a low-pass filter. Note that Port C (5..4) use digital supply voltage, VCC.

**AREF**

* AREF is the analog reference pin for the A/D Converter.

**ADC7..6 (TQFP and QFN/MLF Package Only)**

* In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels

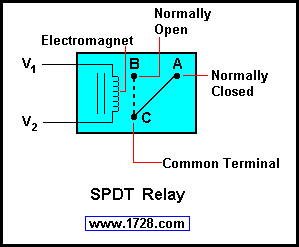
**4 LCD:**

* A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector.
* Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other.
* Without the liquid crystals between them, light passing through one would be blocked by the other.
* The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

**Pin Diagram of LCD:**



**RELAY:**



**Note: In the above diagram, a relay uses an electromagnet.**

* This is a device consisting of a coil of wire wrapped around an iron core.
* When electricity is applied to the coil of wire it becomes magnetic, hence the term electromagnet.
* The A B and C terminals are an SPDT switch controlled by the electromagnet.
* When electricity is applied to V1 and V2, the electromagnet acts upon the SPDT switch so that the B and C terminals are connected.
* When the electricity is disconnected, then the A and C terminals are connected.
* It is important to note that the electromagnet is magnetically linked to the switch but the two are NOT linked electrically.

**SOFTWARE:**

**Atmel AVR**

[](http://en.wikipedia.org/wiki/File:ATmega8_01_Pengo.jpg)

**Atmel ATMEGA328 in** [**28-pin narrow DIP**](http://en.wikipedia.org/wiki/PDIP)

* 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.

**Device overview**

* The AVR is a modified Harvard architecture machine where program and data are 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 AT tiny series
  + 0.5–8 kB program memory
  + 6–32-pin package
  + Limited peripheral set
* **Mega AVR** — the AT mega 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 AT mega 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**

[**AVR32**](http://en.wikipedia.org/wiki/AVR32)**:**

* + In 2006 Atmel released microcontrollers based on the new, 32-bit, [AVR32](http://en.wikipedia.org/wiki/AVR32) architecture.
  + They include [SIMD](http://en.wikipedia.org/wiki/SIMD) and [DSP](http://en.wikipedia.org/wiki/Digital_signal_processor) instructions, along with other audio and video processing features. This 32-bit family of devices is intended to compete with the [ARM](http://en.wikipedia.org/wiki/ARM_architecture) based processors.
  + The instruction set is similar to other RISC cores, but is not compatible with the original AVR or any of the various ARM cores.

**Device architecture:**

* [Flash](http://en.wikipedia.org/wiki/Flash_memory), [EEPROM](http://en.wikipedia.org/wiki/EEPROM), and [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory) are all integrated onto a single chip, removing the need for external memory in most applications.
* Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices.
* Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

**Program memory:**

* Program instructions are stored in volatile flash.
* Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.
* The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash while the ATmega32x line has 32 kB).
* There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash.
* However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

**Internal data memory:**

The data [address space](http://en.wikipedia.org/wiki/Address_space) consists of the [register file](http://en.wikipedia.org/wiki/Register_file), I/O registers, and [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory).

**Internal registers:**

**[](http://en.wikipedia.org/wiki/File:AVR_ATXMEGA_128A1.JPG)**

**Atmel ATxmega328A1 in 100-pin** [**TQFP**](http://en.wikipedia.org/wiki/TQFP) **package:**

* The AVRs have 32 byte registers and are classified as 8-bit RISC devices.
* In most variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (000016–001F16) followed by the 64 I/O registers (002016–005F16).
* Actual SRAM starts after these register sections (address 006016). (Note that the I/O register space may be larger on some more extensive devices, in which case the [memory mapped I/O](http://en.wikipedia.org/wiki/Memory-mapped_I/O) registers will occupy a portion of the SRAM address space.)
* Even though there are separate addressing schemes and optimized opcodes for register file and I/O register access, all can still be addressed and manipulated as if they were in SRAM.
* In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM.
* Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space.
* Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (000016–0FFF16).
* As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions).
* Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space which can be used optionally for mapping the internal EEPROM to the data address space (100016–1FFF16).
* The actual SRAM is located after these ranges, starting at 200016.

**EEPROM:**

* Almost all AVR microcontrollers have internal [EEPROM](http://en.wikipedia.org/wiki/EEPROM) for semi-permanent data storage.
* Like flash memory, EEPROM can maintain its contents when electrical power is removed.
* In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space.
* It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions which makes EEPROM access much slower than other internal RAM.
* However, some devices in the SecureAVR (AT90SC) family [[6]](http://en.wikipedia.org/wiki/Atmel_AVR#cite_note-5) use a special EEPROM mapping to the data or program memory depending on the configuration.
* The XMEGA family also allows the EEPROM to be mapped into the data address space.
* Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well designed EEPROM write routine should compare the contents of an EEPROM address with desired contents and only perform an actual write if contents need to be changed.

**Program execution:**

* Atmel's AVRs have a two stage, single level [pipeline](http://en.wikipedia.org/wiki/Pipeline_%28computing%29) design. This means the next machine instruction is fetched as the current one is executing.
* Most instructions take just one or two clock cycles, making AVRs relatively fast among the [eight-bit](http://en.wikipedia.org/wiki/Eight-bit) microcontrollers.
* The AVR family of processors was designed with the efficient execution of compiled C code in mind and has several built-in pointers for the task.

**Instruction set:**

[**Atmel AVR instruction set**](http://en.wikipedia.org/wiki/Atmel_AVR_instruction_set)

* The [AVR Instruction Set](http://en.wikipedia.org/wiki/Atmel_AVR_instruction_set) is more [orthogonal](http://en.wikipedia.org/wiki/Orthogonal_instruction_set) than those of most eight-bit microcontrollers, in particular the [8051 clones](http://en.wikipedia.org/wiki/Intel_8051) and [PIC microcontrollers](http://en.wikipedia.org/wiki/PIC_microcontroller) with which AVR competes today. However, it is not completely regular:
* [Pointer registers](http://en.wikipedia.org/wiki/Pointer_register) X, Y, and Z have addressing capabilities that are different from each other.
* [Register](http://en.wikipedia.org/wiki/Processor_register) locations R0 to R15 have different addressing capabilities than register locations R16 to R31.
* I/O ports 0 to 31 have different addressing capabilities than I/O ports 32 to 63.
* CLR affects flags, while SER does not, even though they are complementary instructions. CLR set all bits to zero and SER sets them to one. (Note that CLR is pseudo-op for EOR R, R; and SER is short for LDI R,$FF. Math operations such as EOR modify flags while moves/loads/stores/branches such as LDI do not.)
* Accessing read-only data stored in the program memory (flash) requires special LPM instructions; the flash bus is otherwise reserved for instruction memory.
* Additionally, some chip-specific differences affect code generation. Code pointers (including return addresses on the stack) are two bytes long on chips with up to 128 Kbytes of flash memory, but three bytes long on larger chips; not all chips have hardware multipliers; chips with over 8 Kbytes of flash have branch and call instructions with longer ranges; and so forth.
* The mostly-regular instruction set makes programming it using C (or even Ada) compilers fairly straightforward. [GCC](http://en.wikipedia.org/wiki/GNU_Compiler_Collection) has included AVR support for quite some time, and that support is widely used.
* In fact, Atmel solicited input from major developers of compilers for small microcontrollers, to determine the instruction set features that were most useful in a compiler for high-level languages.

**MCU speed:**

* The AVR line can normally support clock speeds from 0-20 MHz, with some devices reaching 32 MHz Lower powered operation usually requires a reduced clock speed.
* All recent (Tiny, Mega, and X mega, but not 90S) AVRs feature an on-chip oscillator, removing the need for external clocks or resonator circuitry.
* Some AVRs also have a system clock prescaler that can divide down the system clock by up to 1024.
* This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.
* Since all operations (excluding literals) on registers R0 - R31 are single cycle, the AVR can achieve up to 1 [MIPS](http://en.wikipedia.org/wiki/Million_instructions_per_second) per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS.
* Loads and stores to/from memory take 2 cycles, branching takes 2 cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices.

**Development:**

* AVRs have a large following due to the free and inexpensive development tools available, including reasonably priced development boards and free development software.
* The AVRs are sold under various names that share the same basic core but with different peripheral and memory combinations.
* Compatibility between chips in each family is fairly good, although I/O controller features may vary.

**Features:**

**Current AVRs offer a wide range of features:**

* Multifunction, bi-directional general-purpose I/O ports with configurable, built-in [pull-up resistors](http://en.wikipedia.org/wiki/Pull-up_resistor)
* Multiple internal oscillators, including RC oscillator without external parts
* Internal, self-programmable instruction [flash memory](http://en.wikipedia.org/wiki/Flash_memory) up to 256 KB (384 KB on X Mega)
  + [In-system programmable](http://en.wikipedia.org/wiki/In-system_programming) using serial/parallel low-voltage proprietary interfaces or [JTAG](http://en.wikipedia.org/wiki/JTAG)
  + Optional boot code section with independent lock bits for protection
* On-chip debugging (OCD) support through JTAG or [debug WIRE](http://en.wikipedia.org/wiki/DebugWIRE) on most devices
  + The JTAG signals (TMS, TDI, TDO, and TCK) are multiplexed on [GPIOs](http://en.wikipedia.org/wiki/General_Purpose_Input/Output). These pins can be configured to function as JTAG or GPIO depending on the setting of a fuse bit, which can be programmed via ISP or HVSP. By default, AVRs with JTAG come with the JTAG interface enabled.
  + [Debug WIRE](http://en.wikipedia.org/wiki/DebugWIRE) uses the /RESET pin as a bi-directional communication channel to access on-chip debug circuitry. It is present on devices with lower pin counts, as it only requires one pin.
* Internal data [EEPROM](http://en.wikipedia.org/wiki/EEPROM) up to 4 KB
* Internal [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory) up to 16 KB (32 KB on X Mega)
* External 64 KB little endian data space on certain models, including the Mega8515 and Mega162.
  + The external data space is overlaid with the internal data space, such that the full 64 KB address space does not appear on the external bus. Accesses to e.g. address 010016 will access internal RAM, not the external bus.
  + In certain members of the X Mega series, the external data space has been enhanced to support both SRAM and SDRAM. As well, the data addressing modes have been expanded to allow up to 16 MB of data memory to be directly addressed.
  + AVRs generally do not support executing code from external memory. Some [ASSPs](http://en.wikipedia.org/wiki/Application-specific_standard_product) using the AVR core do support external program memory.
* 8-bit and 16-bit timers
  + [PWM](http://en.wikipedia.org/wiki/Pulse-width_modulation) output (some devices have an enhanced PWM peripheral which includes a dead-time generator)
  + Input capture
* Analog comparator
* 10 or 12-bit [A/D converters](http://en.wikipedia.org/wiki/Analog-to-digital_converter), with multiplex of up to 16 channels
* 12-bit [D/A converters](http://en.wikipedia.org/wiki/Digital-to-analog_converter)
* A variety of serial interfaces, including
  + [I²C](http://en.wikipedia.org/wiki/I%C2%B2C) compatible Two-Wire Interface (TWI)
  + Synchronous/asynchronous serial peripherals ([UART](http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter)/USART) (used with [RS-232](http://en.wikipedia.org/wiki/RS-232), [RS-485](http://en.wikipedia.org/wiki/RS-485), and more)
  + [Serial Peripheral Interface Bus](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus) (SPI)
  + Universal Serial Interface (USI) for two or three-wire synchronous data transfer
* [Brownout](http://en.wikipedia.org/wiki/Power_outage) detection
* [Watchdog timer](http://en.wikipedia.org/wiki/Watchdog_timer) (WDT)
* Multiple power-saving sleep modes
* Lighting and motor control ([PWM](http://en.wikipedia.org/wiki/Pulse-width_modulation)-specific) controller models
* [CAN](http://en.wikipedia.org/wiki/Controller_area_network) controller support
* [USB](http://en.wikipedia.org/wiki/Universal_Serial_Bus) controller support
  + Proper full-speed (12 Mbit/s) hardware & Hub controller with embedded AVR.
  + Also freely available low-speed (1.5 Mbit/s) ([HID](http://en.wikipedia.org/wiki/Human_interface_device)) [bit banging](http://en.wikipedia.org/wiki/Bit-banging) software emulations
* [Ethernet](http://en.wikipedia.org/wiki/Ethernet) controller support
* [LCD](http://en.wikipedia.org/wiki/Liquid_crystal_display) controller support
* Low-voltage devices operating down to 1.8 V (to 0.7 V for parts with built-in DC–DC up converter)
* picoPower devices
* [DMA](http://en.wikipedia.org/wiki/Direct_memory_access) controllers and "event system" peripheral communication.
* Fast cryptography support for [AES](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard) and [DES](http://en.wikipedia.org/wiki/Data_Encryption_Standard)

**Programming interfaces:**

* There are many means to load program code into an AVR chip. The methods to program AVR chips vary from AVR family to family.

**ISP:**

* The In-system programming (ISP) programming method is functionally performed through [SPI](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus), plus some twiddling of the Reset line.
* As long as the SPI pins of the AVR aren't connected to anything disruptive, the AVR chip can stay soldered on a [PCB](http://en.wikipedia.org/wiki/Printed_circuit_board) while reprogramming.
* All that's needed is a 6-pin connector and programming adapter. This is the most common way to develop with an AVR.
* The Atmel AVR ISP mkII device connects to a computer's USB port and performs in-system programming using Atmel's software.
* AVRDUDE (AVR DownloderUploaDEr) runs on [Linux](http://en.wikipedia.org/wiki/Linux), [FreeBSD](http://en.wikipedia.org/wiki/FreeBSD), Windows, and [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X), and supports a variety of in-system programming hardware, including Atmel AVR ISP mkII, Atmel JTAG ICE, older Atmel serial-port based programmers, and various third-party and "do-it-yourself" programmers.

**PDI:**

* The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of XMEGA devices.
* The PDI supports high-speed programming of all non-volatile memory (NVM) spaces; flash, EEPROM, fuses, lock-bits and the User Signature Row.
* This is done by accessing the XMEGA NVM controller through the PDI interface, and executing NVM controller commands.
* The PDI is a 2-pin interface using the Reset pin for clock input (PDI\_CLK) and a dedicated data pin (PDI\_DATA) for input and output.

**High voltage:**

* High-voltage serial programming (hvsp) is mostly the backup mode on smaller AVRs.
* An 8-pin AVR package doesn't leave many unique signal combinations to place the AVR into a programming mode.
* A 12 volt signal, however, is something the AVR should only see during programming and never during normal operation.

**Parallel:**

* Parallel programming is considered the "final resort" and may be the only way to fix AVR chips with bad fuse settings.
* Parallel programming may be faster and beneficial when programming many AVR devices for production use.

**Boot loader:**

* Most AVR models can reserve a [boot loader](http://en.wikipedia.org/wiki/Bootloader) region, 256 B to 4 KB, where re-programming code can reside.
* At reset, the boot loader runs first, and does some user-programmed determination whether to re-program, or jump to the main application.
* The code can re-program through any interface available, it could read an encrypted binary through an Ethernet adapter like [PXE](http://en.wikipedia.org/wiki/Preboot_Execution_Environment).
* Atmel has application notes and code pertaining to many bus interfaces.

**ROM:**

* The AT90SC series of AVRs are available with a factory mask-ROM rather than flash for program memory.
* Because of the large up-front cost and minimum order quantity, a mask-ROM is only cost-effective for high production runs.

**A Wire:**

* A Wire is a new one-wire debug interface available on the new UC3L AVR32 devices.

**Debugging interfaces:**

* The AVR offers several options for debugging, mostly involving on-chip debugging while the chip is in the target system.

**Debug WIRE:**

* Debug WIREis Atmel's solution for providing on-chip debug capabilities via a single microcontroller pin. It is particularly useful for lower pin count parts which cannot provide the four "spare" pins needed for JTAG.
* The JTAGICE mkII, mkIII and the AVR Dragon support debug WIRE. debug WIRE was developed after the original JTAGICE release, and now clones support it.

**JTAG:**

* [JTAG](http://en.wikipedia.org/wiki/JTAG) provides access to on-chip debugging functionality while the chip is running in the target system.
* JTAG allows accessing internal memory and registers, setting breakpoints on code, and single-stepping execution to observe system behavior.

**Atmel provides a series of JTAG adapters for the AVR:**

1. The [JTAGICE 3](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=17213&category_id=163&family_id=607&subfamily_id=2138) is the latest member of the JTAGICE family (JTAGICE mkIII). It supports JTAG, aWire, SPI, and PDI interfaces.
2. The [JTAGICE mkII](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3353) replaces the JTAGICE, and is similarly priced. The JTAGICE mkII interfaces to the PC via USB, and supports both JTAG and the newer debug WIRE interface. Numerous 3rd-party clones of the Atmel JTAGICE mkII device started shipping after Atmel released the communication protocol.
3. The [AVR Dragon](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3891) is a low-cost (approximately $50) substitute for the JTAGICE mkII for certain target parts. The AVR Dragon provides in-system serial programming, high-voltage serial programming and parallel programming, as well as JTAG or debug WIRE emulation for parts with 32 KB of program memory or less. ATMEL changed the debugging feature of AVR Dragon with the latest firmware of AVR STUDIO 4 - AVR STUDIO 5 and now it supports devices over 32KB of program memory.

4. The [JTAGICE adapter](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2737) interfaces to the PC via a standard serial port. The JTAGICE has been [End-Of-Life](http://en.wikipedia.org/wiki/End-of-life_%28product%29), though it is still supported in AVR Studio and other tools.

5. JTAG can also be used to perform a [Boundary Scan](http://en.wikipedia.org/wiki/Boundary_scan) test, which tests the electrical connections between AVRs and other Boundary Scan capable chips in a system. Boundary scan is well-suited for a production line; the hobbyist is probably better off testing with a multimeter or oscilloscope.

**Development tools and evaluation kits:**

* Official Atmel AVR development tools and evaluation kits consists of a number of starter kits and debugging tools with support for most AVR devices:

**STK600 starter kit:**

* The STK600 starter kit and development system is an update to the STK500.
* The STK600 uses a base board, a signal routing board, and a target board.
* The base board is similar to the STK500, in that it provides a power supply, clock, in-system programming, an RS-232 port and a CAN (Controller Area Network, an automotive standard) port via DB9 connectors, and stake pins for all of the GPIO signals from the target device.
* The target boards have [ZIF](http://en.wikipedia.org/wiki/Zero_insertion_force) sockets for [DIP](http://en.wikipedia.org/wiki/Dual_in-line_package), [SOIC](http://en.wikipedia.org/wiki/Small-outline_integrated_circuit), [QFN](http://en.wikipedia.org/wiki/Quad_Flat_No_leads_package), or [QFP](http://en.wikipedia.org/wiki/Quad_Flat_Package) packages, depending on the board.
* The signal routing board sits between the base board and the target board, and routes the signals to the proper pin on the device board. There are many different signal routing boards that could be used with a single target board, depending on what device is in the ZIF socket.
* The STK600 interfaces with the PC via USB, leaving the RS-232 port available for the target microcontroller.
* A 4 pin header on the STK600 labeled 'RS-232 spare' can connect any TTL level USART port on the chip to the onboard MAX232 chip.
* The MAX232 is a TTL to RS-232 signal level converter to communicate with PC's.
* The pins are RX, TX, CTS, and RTS.

**STK500 starter kit:**

* The STK500 starter kit and development system features ISP and high voltage programming (HVP) for all AVR devices, either directly or through extension boards.
* The board is fitted with DIP sockets for all AVRs available in DIP packages.
* STK500 Expansion Modules: Several expansion modules are available for the STK500 board:
* STK501 - Adds support for microcontrollers in 64-pin TQFP packages.
* STK502 - Adds support for LCD AVRs in 64-pin TQFP packages.
* STK503 - Adds support for microcontrollers in 100-pin TQFP packages.
* STK504 - Adds support for LCD AVRs in 100-pin TQFP packages.
* STK505 - Adds support for 14 and 20-pin AVRs.
* STK520 - Adds support for 14 and 20, and 32-pin microcontrollers from the AT90PWM and AT mega family.
* STK524 - Adds support for the ATmega32M1/C1 32-pin CAN/LIN/Motor Control family.
* STK525 - Adds support for the AT90USB microcontrollers in 64-pin TQFP packages.
* STK526 - Adds support for the AT90USB microcontrollers in 32-pin TQFP packages

**STK200 starter kit**

* The STK200 starter kit and development system can use AVR chips via DIL-40/20/8 and features 4 MHz clock source, 8x Light-emitting diodes, 8x input buttons, [RS-232](http://en.wikipedia.org/wiki/RS-232) port, option for 32k [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory) and numerous general I/O.
* Programmed can be done with a dongle connected to the parallel-port and the ISP socket.
* Software wise programs can be compiled with [*avr-gcc*](http://en.wikipedia.org/wiki/GNU_Compiler_Collection), Simulated with *simulavr*, Downloaded with *avrdude/avrice* on [BSD](http://en.wikipedia.org/wiki/FreeBSD) and Linux. Assembler is available with *avra/tpasm*.
* GNU debugger is available with *avr-gdb*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Support microcontrollers (according to the manual):** | | | | | |
| **Chip** | [**Flash size**](http://en.wikipedia.org/wiki/Flash_memory) | [**EEPROM**](http://en.wikipedia.org/wiki/EEPROM) | [**SRAM**](http://en.wikipedia.org/wiki/Static_random-access_memory) | **Frequency [MHz]** | [**Package**](http://en.wikipedia.org/wiki/Chip_carrier) |
| AT90S1200 | 1k | 64 | 0 | 12 | [PDIP-20](http://en.wikipedia.org/wiki/Dual_in-line_package) |
| AT90S2313 | 2k | 128 | 128 | 10 | PDIP-20 |
| AT90S/LS2323 | 2k | 128 | 128 | 10 | PDIP-8 |
| AT90S/LS2343 | 2k | 128 | 128 | 10 | PDIP-8 |
| AT90S4414 | 4k | 256 | 256 | 8 | PDIP-40 |
| AT90S/LS4434 | 4k | 256 | 256 | 8 | PDIP-40 |
| AT90S8515 | 8k | 512 | 512 | 8 | PDIP-40 |
| AT90S/LS8535 | 8k | 512 | 512 | 8 | PDIP-40 |

**AVR ISP and AVR ISP mkII:**

* The AVR ISP and AVR ISP mkII are inexpensive tools allowing all AVRs to be programmed via [ICSP](http://en.wikipedia.org/wiki/In-circuit_serial_programming).
* The AVR ISP connects to a PC via a serial port, and draws power from the target system. The AVR ISP allows using either of the "standard" ICSP pin outs, either the 10-pin or 6-pin connector. The AVR ISP has been discontinued, replaced by the AVR ISP mkII.
* The AVR ISP mkII connects to a PC via USB, and draws power from USB. [LEDs](http://en.wikipedia.org/wiki/Light-emitting_diode) visible through the translucent case indicate the state of target power.

**AVR Dragon:**

**[](http://en.wikipedia.org/wiki/File:AvrDragon.png)**

**AVR Dragon with** [**ISP programming cable**](http://en.wikipedia.org/wiki/In-System_Programming)**:**

* The Atmel Dragon is an inexpensive tool which connects to a PC via USB.
* The Dragon can program all AVRs via JTAG, HVP, PDI, or ICSP.
* The Dragon also allows debugging of all AVRs via JTAG, PDI, or Debug Wire; a previous limitation to devices with 32 KB or less program memory has been removed in AVRstudio 4.18.
* The Dragon has a small prototype area which can accommodate an 8, 28, or 40-pin AVR, including connections to power and programming pins.
* There is no area for any additional circuitry, although this can be provided by a third-party product called the "[Dragon Rider](http://www.ecrostech.com/AtmelAvr/DragonRider/)".

**JTAGICE mkI:**

* The [JTAG](http://en.wikipedia.org/wiki/JTAG) in Circuit Emulator (JTAGICE) debugging tool supports on-chip debugging (OCD) of AVRs with a JTAG interface. The original JTAGICE mkI uses an RS-232 interface to a PC, and can only program AVR's with a JTAG interface.
* The JTAGICE mkI is no longer in production; however it has been replaced by the JTAGICE mkII.

**JTAGICE mkII:**

The JTAGICE mkII debugging tool supports on-chip debugging (OCD) of AVRs with SPI, JTAG, PDI, and debug WIRE interfaces. The debug Wire interface enables debugging using only one pin (the Reset pin), allowing debugging of applications running on low pin-count microcontrollers.

The JTAGICE mkII connects using USB, but there is an alternate connection via serial port, which requires using a separate power supply. In addition to JTAG, the mkII supports ISP programming (using 6-pin or 10-pin adapters). Both the USB and serial links use a variant of the STK500 protocol.

**Butterfly demo board**

**[](http://en.wikipedia.org/wiki/File:ATmega169-MLF.jpg):**

**Atmel ATmega169 in 64-pad** [**MLF**](http://en.wikipedia.org/wiki/MicroLeadFrame) **package:**

[**AVR Butterfly**](http://en.wikipedia.org/wiki/AVR_Butterfly)**:**

* The very popular AVR Butterfly demonstration board is a self-contained, battery-powered computer running the Atmel AVR ATmega169V microcontroller.
* It was built to show-off the AVR family, especially a new built-in LCD interface. The board includes the LCD screen, joystick, speaker, serial port, real time clock (RTC), flash memory chip, and both temperature and voltage sensors. Earlier versions of the AVR Butterfly also contained a CdS[photoresistor](http://en.wikipedia.org/wiki/Photoresistor); it is not present on Butterfly boards produced after June 2006 to allow [RoHS](http://en.wikipedia.org/wiki/RoHS) compliance.
* The small board has a shirt pin on its back so it can be worn as a name badge.
* The AVR Butterfly comes preloaded with software to demonstrate the capabilities of the microcontroller. Factory firmware can scroll your name, display the sensor readings, and show the time.
* The AVR Butterfly also has a piezo speaker that can be used to reproduce sounds and music.
* The AVR Butterfly demonstrates LCD driving by running a 14-segment, six alpha-numeric character displays.
* However, the LCD interface consumes many of the I/O pins.
* The Butterfly's ATmega169 CPU is capable of speeds up to 8 MHz; however it is factory set by software to 2 MHz to preserve the button battery life.
* A pre-installed boot loader program allows the board to be re-programmed via a standard RS-232 serial plug with new programs that users can write with the free Atmel IDE tools.

**AT90USBKey**

* This small board, about half the size of a business card, is priced at slightly more than an AVR Butterfly.
* It includes an AT90USB1287 with [USB On-The-Go](http://en.wikipedia.org/wiki/USB_On-The-Go) (OTG) support, 16 MB of [Data Flash](http://en.wikipedia.org/wiki/DataFlash), LEDs, a small joystick, and a temperature sensor.
* The board includes software which lets it act as a USB Mass Storage device (its documentation is shipped on the Data Flash), a USB joystick, and more. To support the USB host capability, it must be operated from a battery; but when running as a USB peripheral, it only needs the power provided over USB.
* Only the JTAG port uses conventional 2.54 mm pin out. All the other AVR I/O ports require more compact 1.27 mm headers.
* The AVR Dragon can both program and debug since the 32 kb limitation was removed in AVR Studio 4.18, and the JTAGICE mkII is capable of both programming and debugging the processor.
* The processor can also be programmed through USB from a Windows or Linux host, using the USB "Device Firmware Update" protocols.
* Atmel ships proprietary (source code included but distribution restricted) example programs and a USB protocol stack with the device.
* [LUFA](http://www.fourwalledcubicle.com/LUFA.php) is a third-party [free software](http://en.wikipedia.org/wiki/Free_software) ([MIT license](http://en.wikipedia.org/wiki/MIT_license)) USB protocol stack for the USBKey and other.

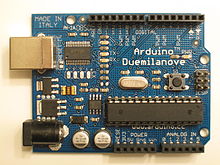
**Raven microcontroller kit:**

* The RAVEN kit supports microcontroller development using Atmel's [IEEE 802.15.4](http://en.wikipedia.org/wiki/IEEE_802.15.4) chipsets, for [ZigBee](http://en.wikipedia.org/wiki/ZigBee) and other microcontroller stacks.
* It resembles a pair of microcontroller more-powerful Butterfly cards, plus a microcontrollerUSBKey; and costing about that much (under $US100).
* All these boards support JTAG-based development.
* The kit includes two AVR Raven boards, each with 2.4 GHz transceiver supporting IEEE 802.15.4 (and a freely licensed ZigBee stack).
* The radios are driven with ATmega1284p processors, which are supported by a custom segmented LCD display driven by an ATmega3290p processor. Raven peripherals resemble the Butterfly: piezo speaker, Data Flash (bigger), external EEPROM, sensors, 32 kHz crystal for [RTC](http://en.wikipedia.org/wiki/Real-time_clock), and so on. These are intended for use in developing remote sensor nodes, to control relays, or whatever is needed.
* The USB stick uses an AT90USB1287 for connections to a USB host and to the 2.4 GHz microcontroller links.
* These are intended to monitor and control the remote nodes, relying on host power rather than local batteries.

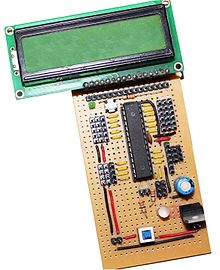
**Third-party programmers:**

* A wide variety of third-party programming and debugging tools are available for the AVR.
* These devices use various interfaces, including RS-232, PC parallel port, and USB. [AVR Freaks](http://www.avrfreaks.net/) has a comprehensive list.

**Atmel AVR usage:**

[](http://en.wikipedia.org/wiki/File:Arduino_Duemilanove_0509.JPG)

**Atmel AVR Atmega328 28-pin DIPS on an Arduino Duemilanove board:**

[](http://en.wikipedia.org/wiki/File:Atmega8_Development_Board.jpg)

* AVRs have been used in various automotive applications such as security, safety, power train and entertainment systems.
* Atmel AVR ATMEGA328 28-pin DIP on a  
  [custom designed development board](http://www.robotplatform.com)
* Atmel has recently launched a new publication "Atmel Automotive Compilation" to help developers with automotive applications. Some current usages are in BMW, Daimler-Chrysler and TRW.
* The [Arduino](http://en.wikipedia.org/wiki/Arduino)[physical computing](http://en.wikipedia.org/wiki/Physical_computing) platform is based on an ATmega328 microcontroller (ATmega168 or ATMEGA328 in older board versions than the Diecimila).
* The ATmega1280 and ATmega2560, with more pin out and memory capabilities, have also been employed to develop the Arduino Mega platform.
* Arduino boards can be used with its language and [IDE](http://en.wikipedia.org/wiki/Integrated_development_environment), or with more conventional programming environments ([C](http://en.wikipedia.org/wiki/C_%28programming_language%29), [assembler](http://en.wikipedia.org/wiki/Assembly_language), etc.) as just standardized and widely available AVR platforms.
* USB-based AVRs have been used in the Microsoft Xbox hand controllers. The link between the controllers and Xbox is USB.
* Numerous companies produce AVR-based microcontroller boards intended for use by hobbyists, robot builders, experimenters and small system developers including: [Cubloc](http://www.cubloc.com), [gnusb](http://gnusb.sourceforge.net), [Basic X](http://www.basicx.com/), [Oak Micros](http://www.oakmicros.com/), [ZX Microcontrollers](http://www.zbasic.net/), and [my AVR](http://www.myavr.com/). There is also a large community of [Arduino-compatible boards](http://en.wikipedia.org/wiki/Arduino-compatible_boards) supporting similar users.
* Few hobbyists prefer making their own version of board from scratch.
* [Schneider Electric](http://en.wikipedia.org/wiki/Schneider_Electric) produces the M3000 Motor and Motion Control Chip, incorporating an Atmel AVR Core and an Advanced Motion Controller for use in a variety of motion applications.

**FPGA clones:**

* With the growing popularity of [FPGAs](http://en.wikipedia.org/wiki/Field-programmable_gate_array) among the open source community, people have started developing open source processors compatible with the AVR instruction set. The [Open Cores](http://en.wikipedia.org/wiki/OpenCores) website lists the following major AVR clone projects:
* [pAVR](http://opencores.org/project,pavr), written in [VHDL](http://en.wikipedia.org/wiki/VHDL), is aimed at creating the fastest and maximally featured AVR processor, by implementing techniques not found in the original AVR processor such as deeper pipelining.
* [avr\_core](http://opencores.org/project,avr_core), written in [VHDL](http://en.wikipedia.org/wiki/VHDL), is a clone aimed at being as close as possible to the ATmega103.
* [Navre](http://opencores.org/project,navre), written in [Verilog](http://en.wikipedia.org/wiki/Verilog), implements all [Classic Core](http://en.wikipedia.org/wiki/Atmel_AVR_instruction_set) instructions and is aimed at high performance and low resource usage. It does not support interrupts.

**Window for choosing target device**:



* Next, Micro Vision must be instructed to generate a HEX file upon program compilation.
* A HEX file is a standard file format for storing executable code that is to be loaded onto the microcontroller. In the “Project Workspace” pane at the left, right–click on “Target 1” and select “Options for ‘Target 1’ ”.
* Under the “Output” tab of the resulting options dialog, ensure that both the “Create Executable” and “Create HEX File” options are checked.
* Then click “OK”.

**Project Options Dialog:**



* Next, a file must be added to the project that will contain the project code. To do this, expand the “Target 1” heading, right–click on the “Source Group 1” folder, and select “Add files”
* Create a new blank file (the file name should end in “.asm”), select it, and click “Add.”
* The new file should now appear in the “Project Workspace” pane under the “Source Group 1” folder.
* Double-click on the newly created file to open it in the editor. All code for this lab will go in this file.
* To compile the program, first save all source files by clicking on the “Save All” button, and then click on the “Rebuild All Target Files” to compile the program as shown in the figure below.
* If any errors or warnings occur during compilation, they will be displayed in the output window at the bottom of the screen.
* All errors and warnings will reference the line and column number in which they occur along with a description of the problem so that they can be easily located.
* Note that only errors indicate that the compilation failed, warnings do not (though it is generally a good idea to look into them anyway).

**Project Workspace Pane:**



**“Save All” and “Build All Target Files” Buttons**

****

* At the left side of the debugger window, a table is displayed containing several key parameters about the simulated microcontroller, most notably the elapsed time (circled in the figure below).
* Just above that, there are several buttons that control code execution. The “Run” button will cause the program to run continuously until a breakpoint is reached, whereas the “Step Into” button will execute the next line of code and then pause (the current position in the program is indicated by a yellow arrow to the left of the code).

**PROGRAMMER:**

* The programmer used is a powerful programmer for the Atmel 89 series of microcontrollers that includes
* 89C51/52/55
* 89S51/52/55
* And many more.
* It is simple to use & low cost, yet powerful flash microcontroller programmer for the Atmel 89 series.
* It will Program, Read and Verify Code Data, Write Lock Bits, Erase and Blank Check.
* All fuse and lock bits are programmable.
* This programmer has intelligent onboard firmware and connects to the serial port.
* It can be used with any type of computer and requires no special hardware.
* All that is needed is a serial communication port which all computers have.
* All devices also have a number of lock bits to provide various levels of software and programming protection.
* These lock bits are fully programmable using this programmer.
* Locks bits are useful to protect the program to be read back from microcontroller only allowing erase to reprogram the microcontroller.
* Major parts of this programmer are Serial Port, Power Supply and Firmware microcontroller.
* Serial data is sent and received from 9 pin connector and converted to/from TTL logic/RS232 signal levels by MAX232 chip.
* A Male to Female serial port cable, connects to the 9 pin connector of hardware and another side connects to back of computer.
* All the programming **‘intelligence’** is built into the programmer so you do not need any special hardware to run it.
* Programmer comes with window based software for easy programming of the devices.

**μVision3 Debugger window:**



|  |
| --- |
|  |