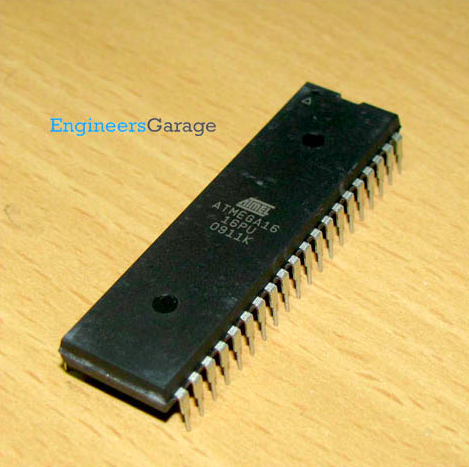
AT MEGA-16

* **ATmega16** is an 8-bit high performance microcontroller of Atmel’s Mega AVR family with low power consumption.
* Atmega16 is based on enhanced **RISC** (Reduced Instruction Set Computing)architecture with 131 powerful instructions. Most of the instructions execute in one machine cycle. Atmega16 can work on a maximum frequency of 16MHz.
* ATmega16 is a 40 pin microcontroller. There are **32 I/O** (input/output) lines which are divided into four 8-bit ports designated as **PORTA, PORTB, PORTC and PORTD**.
* **PIN DESCRIPTION:**

|  |  |
| --- | --- |
| PIN1 | I/O , T0 ( Timer0 External Counter Input) ,XCK : USART External Clock I/O |
| PIN2 | I/O, T1 (Timer1 External Counter Input) |
| PIN3 | I/O, AIN0: Analog Comparator Positive Input , INT2: External Interrupt 2 Input |
| PIN4 | I/O, AIN1: Analog Comparator Negative Input, OC0 : Timer0 Output Compare Match Output |
| PIN9 | Reset Pin, Active Low Reset |
| PIN10 | VCC=+5V |
| PIN11 | GND |
| PIN12 | XTAL2 |
| PIN13 | XTAL1 |
| PIN14 | (RXD) ,I/O PIN 0,USART Serial Communication Interface |
| PIN15 | (TXD) ,I/O Pin 1,USART Serial Communication Interface |
| PIN16 | (INT0),I/O Pin 2, External Interrupt INT0 |
| PIN17 | (INT1),I/O Pin 3, External Interrupt INT1 |
| PIN18 | (OC1B),I/O Pin 4, PWM Channel Outputs |
| PIN19 | (OC1A),I/O Pin 5, PWM Channel Outputs |
| PIN20 | (ICP), I/O Pin 6, Timer/Counter1 Input Capture Pin |
| PIN21 | (OC2),I/O Pin 7,Timer/Counter2 Output Compare Match Output |
| PIN22 | (SCL),I/O Pin 0,TWI Interface |
| PIN23 | (SDA),I/O Pin 1,TWI Interface |
| PIN24-27 | JTAG INTERFACE |
| PIN28 : | (TOSC1),I/O Pin 6,Timer Oscillator Pin 1 |
| PIN29 : | (TOSC2),I/O Pin 7,Timer Oscillator Pin 2 |
| PIN30 : | AVCC (for ADC) |
| PIN31 : | GND (for ADC) |
| PIN33 – PIN40 | PAx: I/O,ADCx (Where x is 7 – 0) |

* We can Program **AT Mega16** using the Arduino IDE , but for that we need to do a few additions to it. So,

**STEP 1: pins\_arduino.h file**

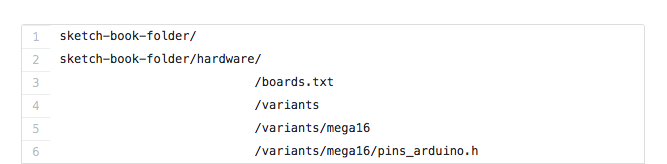
* + - The developers of Arduino, have cleanly separated the pin definitions into a separate files. This allows you to easily add support for new non-Arduino AVR microcontrollers.

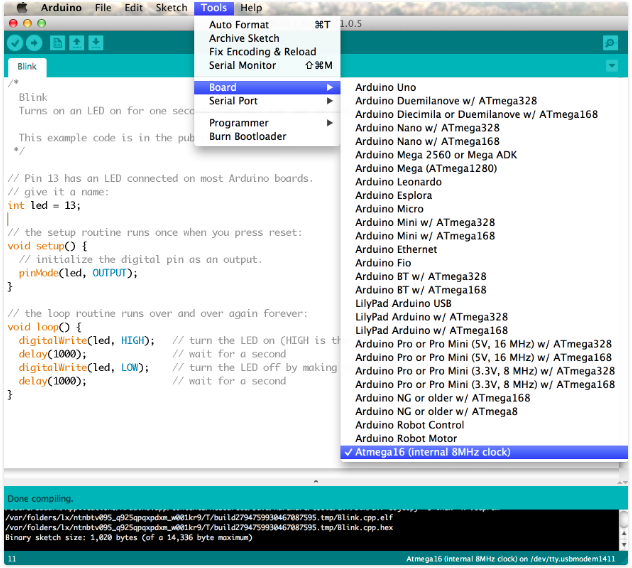
* + - To add support for ATmega 16A, just copy the Uno’s ***pin\_arduino.h*** file from ***hardware/arduino/ variants/standard/*** folder in Arduino installation directory and start modifying it.
    - In the ***pin\_arduino.h*** file, you need to change the following things.
      * The number of digital pins
      * The number of analog pins
      * Analog pin mappings and const for Analog pins (actual values for A0, A1)
      * Digital pin to PCICR mapping
      * Pin to port (PORTA, PORTB etc) mapping
      * Pin to timer mapping
      * The link to the code is :<https://github.com/sudar/arduino-extra-cores/blob/master/variants/mega16/pins_arduino.h>

**STEP 2: pins\_arduino.h file**

* + - The Arduino IDE get details about your board from a file named boards.txt. This simple text file, has the details about your board and also instructs the Arduino IDE to use the proper parameters and fuses while compiling and uploading programs.
    - To let the Arduino IDE, know about our new microcontroller, we need to create a new boards.txt file which has the following information about the microcontroller.
      * Upload protocol
      * Upload speed (baudrate)
      * Upload fuses
      * Clock frequency
      * Maximum flash memory size
      * Which variant (pins\_arduino.h) file to use
      * The link to the .txt file is : *https://github.com/sudar/arduino-extra-cores/blob/master/boards.txt*

**STEP 2: Integration with Arduino IDE**

* + Once we have created the pins\_arduino.h and boards.txt file, we should then place them in a proper directory structure, which the Arduino IDE could understand.
  + First, create a directory called hardware in your sketchbook directory. After that create a new directory inside it to keep all our files. Name it arduino-extra-cores. Place the boards.txt file inside this directory. Now create a directory named variants and then create another directory with the name of your microcontroller. Place the pins\_arduino.h file inside this directory.
  + Once you have done this, the directory structure should look like this.
    - After doing this, restart your Arduino IDE and you should see the new entry that you created in the boards.txt file under the Boards menu.
    - To compile and upload Arduino programs to your new microcontroller, you just have to select the new entry from the Boards menu, everything else will be taken care by the Arduino IDE.



* Now, You can code on The AT MEGA16 like a normal Arduino.

***Important Links:***

* **DataSheet** -<http://ww1.microchip.com/downloads/en/DeviceDoc/doc2466.pdf>
* *<http://www.instructables.com/id/Programming-ATmega16A-using-arduino-IDE/>*
* *<https://electronicsforu.com/resources/learn-electronics/atmega16-pin-diagram-description>*

***Projects:***

* GPS System - <https://electronicsforu.com/electronics-projects/hardware-diy/microcontroller-atmega16a-based-gps-receiver>
* *<http://www.ablab.in/led-blinking-with-avr-atmega16-microcontroller/>*