AVR Microcontroller Programming

Embedded Workshop 8/26/2015

AVR Microcontroller Programming

Please sign in

Name	Member Y/N	Email address

The Basics Steps of Microcontroller Programming

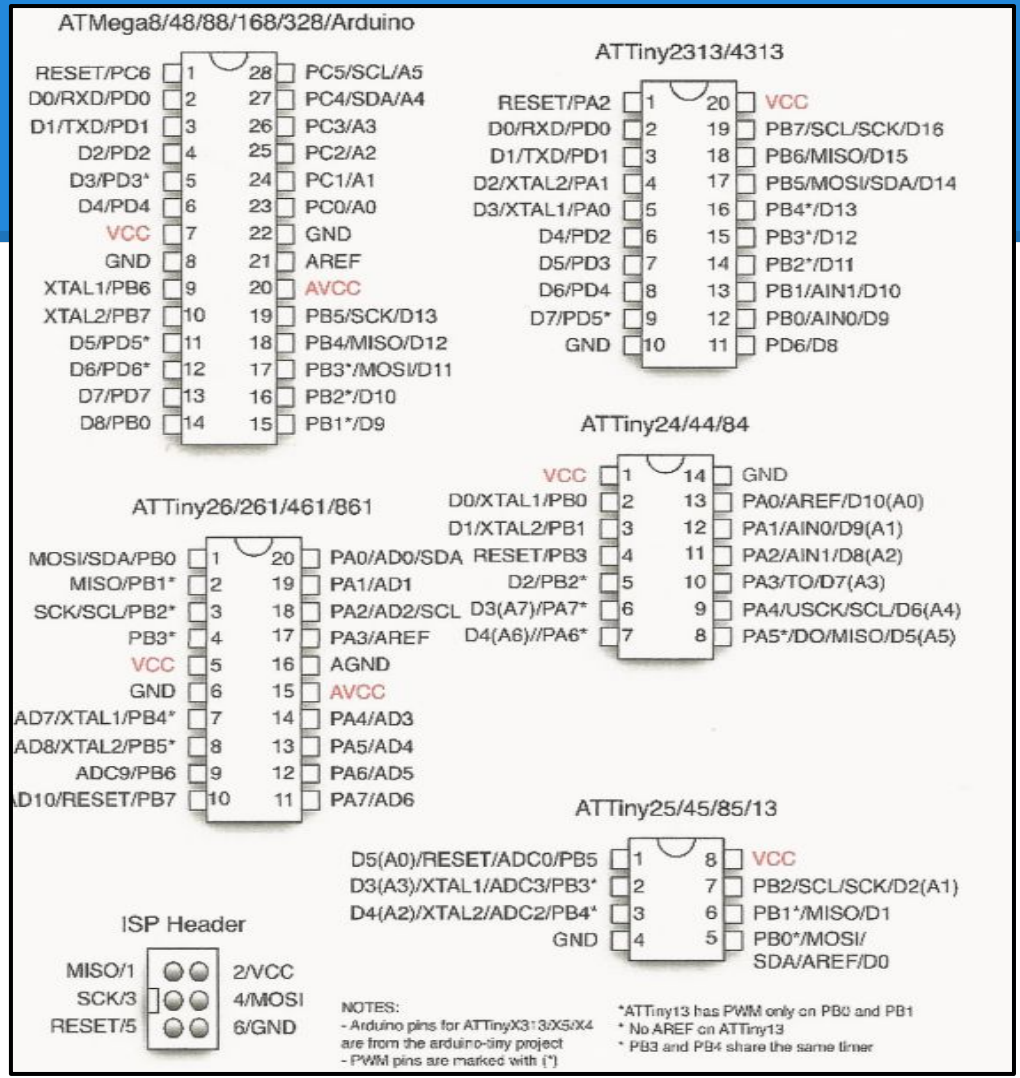
Agenda

1. Overview of the Atmel AVR Chipset.
2. Overview of Programming the AVR Chipset.
3. Overview of the different types of Programmers and Target boards.
   a. USBTiny, USBasp, Arduino as ISP etc…
4. Overview of Writing and Compiling Code with the different types of IDE's.
   a. Arduino IDE, WinAVR (AVRDUDE), Atmel Studio 6
5. Software Change Control.
6. Building an Arduino Shield Target Board.
7. How to program the ATTiny84/85 using the Arduino Uno and a Breadboard.

AVR Chipset

AVRs are generally classified into following:

- **tinyAVR** — the ATtiny series
  - 2–8 kB program memory
  - 6–32-pin package
- **megaAVR** — the ATmega series
  - 4–512 kB program memory
  - 28–100-pin package
  - Extended instruction set
  - 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 ADCs



AVR Ports and Pins

Ports B,C,D {76543210}

Port B 0b**00000000 -** B0 - B7

Port C 0b**x0000000 -** C0 - C6

Port D 0b**00000000 -** D0 - D7

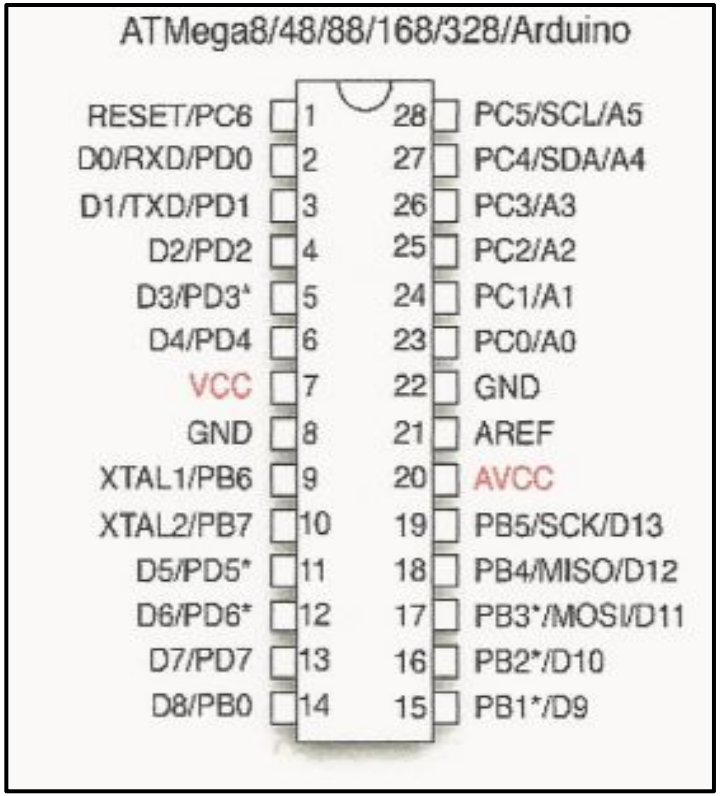
RS232 PD0-RX(2) PD1-TX(3)

I2c PC4-SDA PC5-SCL

SPI PB3-MOSI PB4-MISO PB5-SCK

Reset PC6(1)

Atmega 8/48/88/168/328								
PortB	B0	B1	B2	B3	B4	B5	B6	B7
Pin	14	15	16	17	18	19	9	10
PortC	C0	C1	C2	C3	C4	C5	C6	X
Pin	23	24	25	26	27	28	1	X
PortD	D0	D1	D2	D3	D4	D5	D6	D7
Pin	2	3	4	5	6	11	12	13

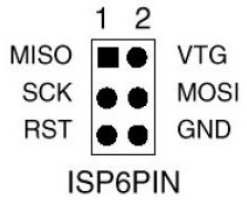


What is an AVR Programmer

AVR programmer connects to your computer's via an USB Cable and communicates with your programming software through a virtual COM port using the AVR In-Service Programmer protocol.

Programming software: **Atmel Studio, AVRDUDE, Arduino IDE, Programmers NotePad** etc.

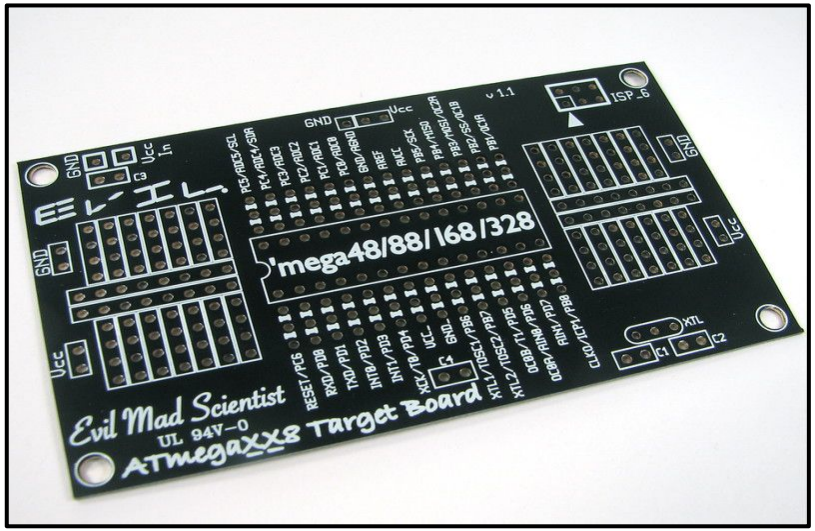
AVR Programmers use SPI protocol (MISO, MOSI, SCK). Some use JTAG for debugging.



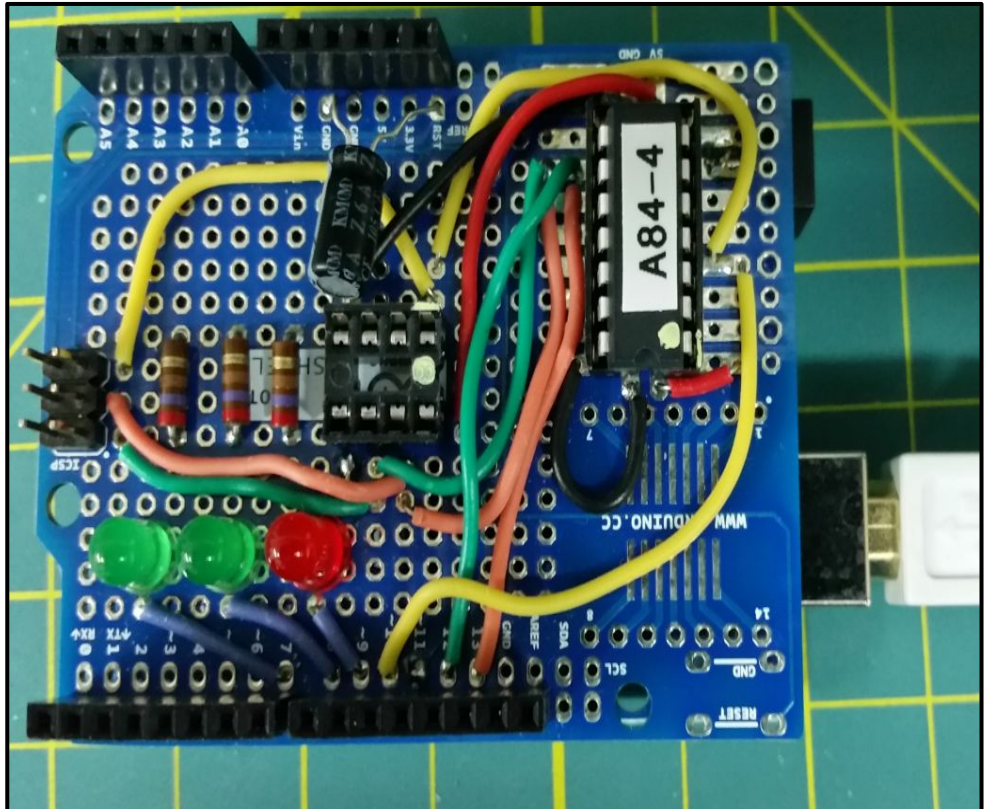
ISP6PIN

The programmer connects to your target device via an 6-pin cable or the older, 10-pin cable.

Target Board is used to hold the chip. AVR chip can stay soldered on a PCB while reprogramming.



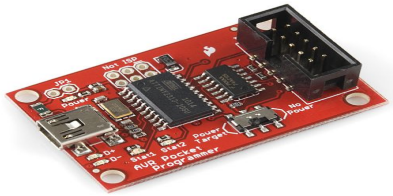
AVR Programmer - ISP Six Pin



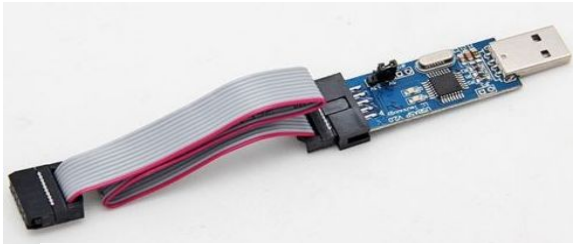
Arduino	ISP	Atiny85/45	
13	3	7	SCK
12	1	6	MISO
11	4	5	MOSI
10	5	1	Reset
VCC	2	8	VCC
GND	6	4	GND

Examples of AVR programmers

Pocket AVR
Sparkfun



USBasp

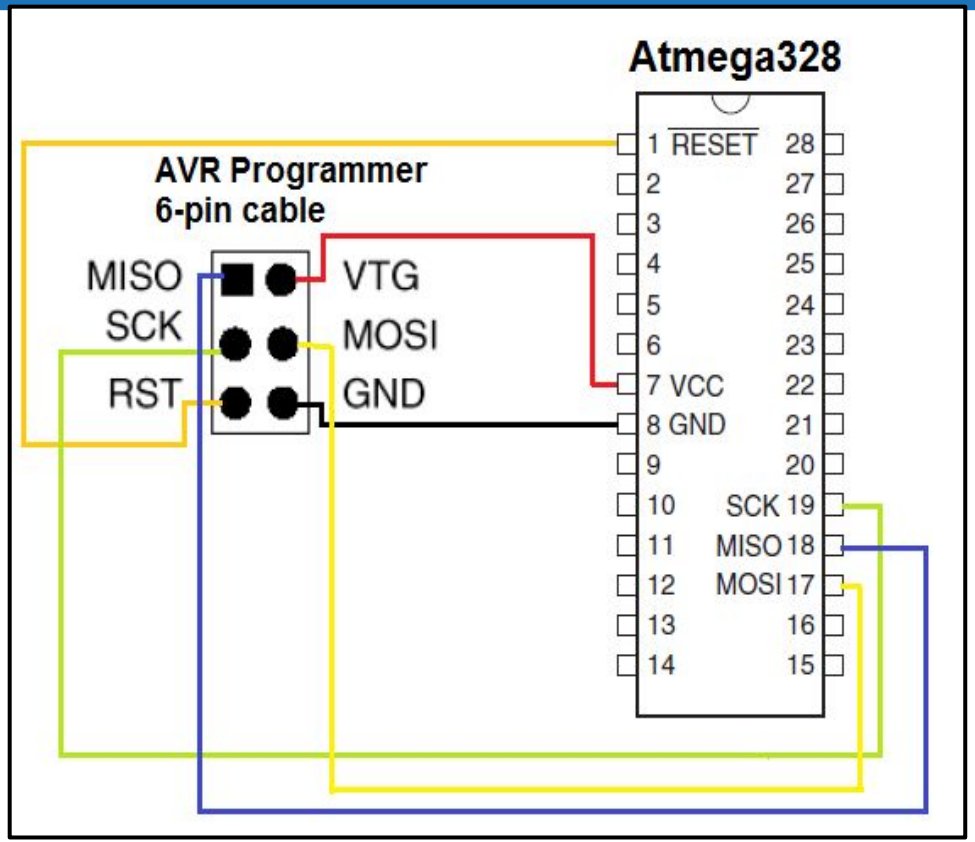


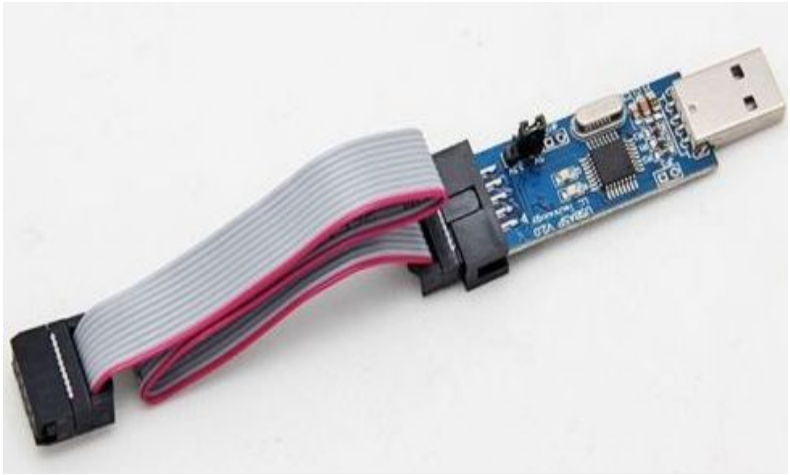
AVR
STK500

USBTinyISP

AVRISP

What is an AVR Programmer





1 = Vcc
2 = MISO
3 = MOSI
4 = SCK
5 = GND
6 = RST

Write code for your Microcontroller

The first step is to write your program code. This is usually done is C. It can also be done in assembly language and some compilers support other languages as well.

The AVR processors was designed with the efficient execution of compiled C code and have several built-in pointers for the task.

Commonly used Software IDE:

Arduino IDE, Notepad++, Eclipse, Atmel Studio

Example Code: Arduino Fade

```
int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

```
#include <avr/IO.h>
#include <util/delay.h>


int main(void)
{

    // This is written for the ATmel attiny using the usbtiny programmer
    // at the chip level

    // This written for the ATtiny 84 or Attiny 44
    //        ATtiny 84/44
    //                    +-----+
    // (+)VCC           1|     |14 GND
    // Analog       P10 2|     |13 P0 Analog
    // Analog       P9  3|     |12 P1 Analog
    // Reset            4|     |11 P2 Analog
    // PWM Analog   P8  5|     |10 P3 Analog
    // PWM Analog   P7  6|     |9  P4 SCK Analog
    // MOSI  Analog P6  7|     |8  P5 MISO Analog PWM
    //                    +-----+
    //
    // ATtiny 44/84 P0 - P10
    // Using Pin 0 Port B
    // http://www.newbiehack.com/
    // Lesson 7
    // Bits for portB   76543210
    //                 0b00000001

    // Set the DDRB Port B input or outputs
    DDRB = 0b00000001;  // Data Direction Register DDR PortB set pins to
    DDRB |= 1 << PINB1; // Set DDRB PIN 1 as an output
    DDRB |= 1 << PINB2; // Set DDRB PIN 2 as an output

    while (1)
    {

    PORTB = 0b00000001; // set pin0 on port b High
    _delay_ms(100);     // delay 1ms
    PORTB = 0b00000000; // set pin0 on port b LOW
    _delay_ms(10);      // delay 1ms
    PORTB ^= 1 << PINB1; // Xor symbol ^
    PORTB ^= 1 << PINB2; // Xor symbol ^
    PORTB ^= 1 << PINB0; // Xor symbol ^
    _delay_us(1000);
    _delay_ms(100);


    }
```

```
//        ATtiny 85/45
//             +-----+
// Reset     1|     |8 VCC
// Analog P3 2|     |7 P2 SCK
// Analog P4 3|     |6 P1 MISO
// GND       4|     |5 P0 MOSI
//             +-----+


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(0, OUTPUT);
}


// the loop function runs over and over again forever
void loop() {
  digitalWrite(0, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);              // wait for a second
  digitalWrite(0, LOW);    // turn the LED off by making the voltage LOW
  delay(500);              // wait for a second
}
```

Compile the Code

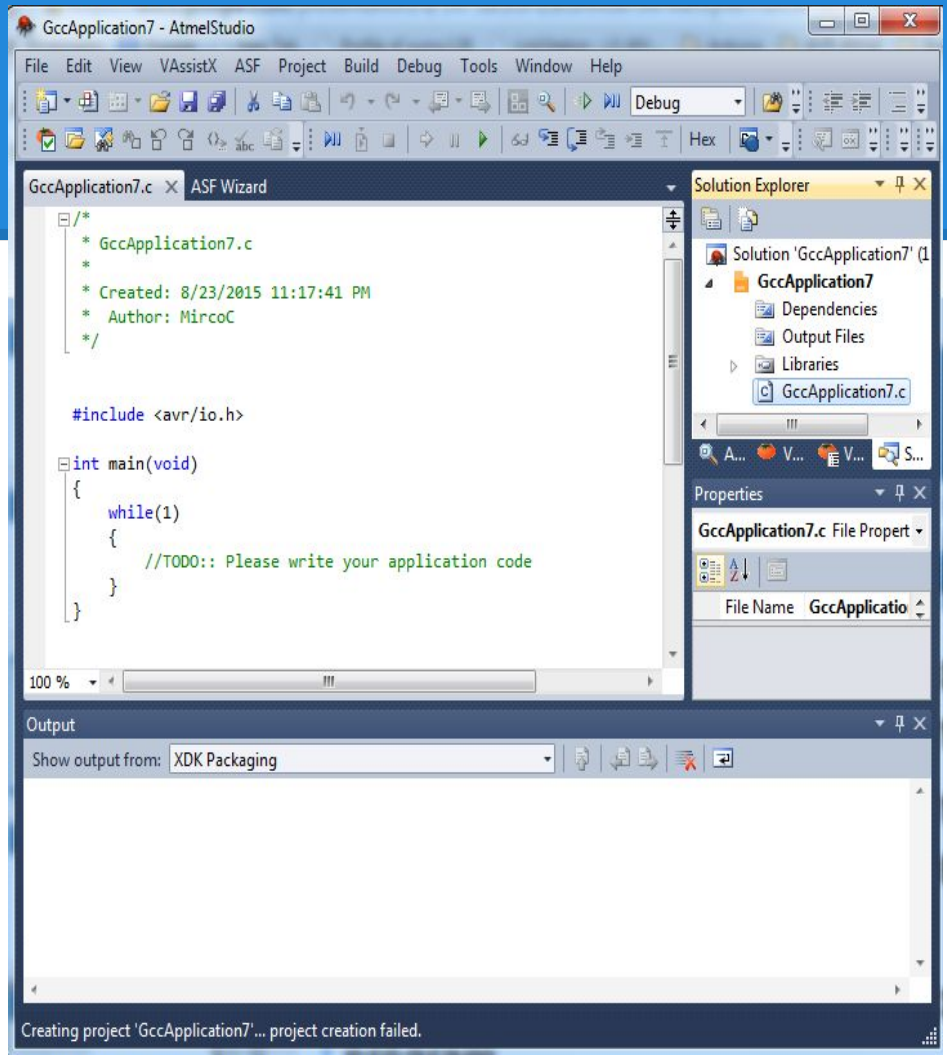
Before uploading your program to your microcontroller it needs to be compiled.

Compiling converts the code from human readable code to machine readable code.

Arduino uses it IDE to compile your program.

Other popular compilers for Atmel AVR chips is avr-gcc.

After compilation you will have one or more files containing machine code.



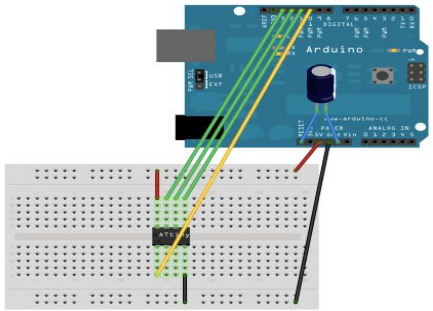
Upload the Compiled Code & Flash Memory

The AVR chip has a small amount of nonvolatile flash memory. Program instructions are stored in the nonvolatile flash memory.

You can use a dedicated programmer such as the following:

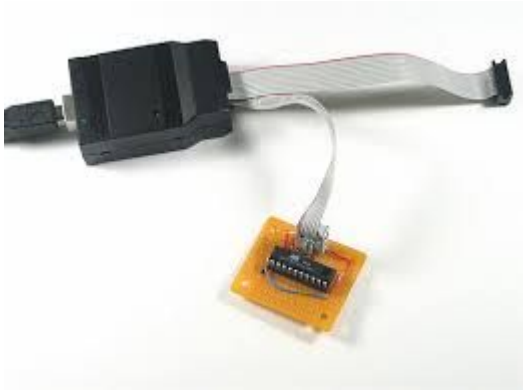
STK600 , STK500, STK200, AVRISP, AVRISP mkII, USBtiny, JTAGICE mkI,

Arduino uno.(Use a program ArduinoISP for uploading file).



USBtiny simple USB programmer, (uses AVRDUDE)

You need a physical connection from your computer to your microcontroller.



AVRDUDE.EXE

The Arduino IDE uses avrdude in the background. (Arduino as ISP = avrisp)
http://www.nongnu.org/avrdude/user-manual/avrdude_4.html#Option-Descriptions

C:\WinAVR-20100110\bin

Read only: avrdude.conf

avrdude -p t84 -c usbtiny -e -U Flash:w"main.hex" -v

avrdude -p m328p -c avrisp -e -U Flash:w:"blink.h" -vvvv

avrdude -p t85 -c usbasp -e -U Flash:w:"blink.h" -vvvv

-p (microcontroller)

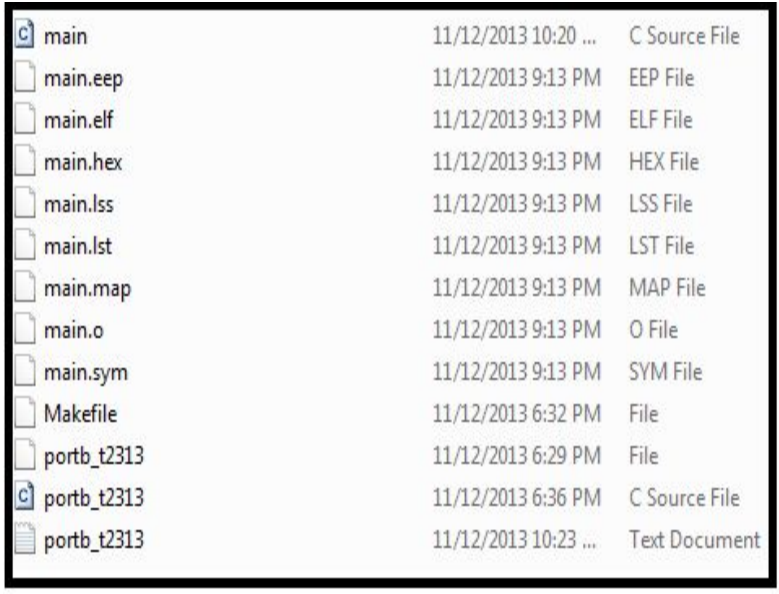
-c (programmer)

-e (erase)

-U <memtype>:r:w:v <filename>

-v verbose

http://www.ladyada.net/learn/avr/avrdude.html



AVRDUDE.EXE

`-U <memtype>:r|w|v:<filename>[:format]:`

The important part. This is where we actually get around to telling **avrdude** how to put the data onto the chip. This command is rather complex, but we'll break it down.

<memtype> - can be **flash**, **eeprom, hfuse** (high fuse), **lfuse** (low fuse), or **efuse** (extended fuse)

r|w|v - can be **r** (read), **w** (write), **v** (verify)

<filename> - the input (writing or verifying) or output file (reading)

[:format] - optional, the format of the file. You can leave this off for writing, but for reading use **i** for Intel Hex (the prevailing standard)

For example:

- To write a file called **firmware.hex** to the flash use the command: **-U flash:w:firmware.hex**
- To verify a file called **mydata.eep** from the eeprom use the command **-U eeprom:v:mydata.eep**
- To read the low fuse into a file use the command **-U lfuse:r: lfusefile.hex:i**

```
C:\WinAVR-20100110\bin>avrdude -p t85 -c usbtiny

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.05s

avrdude: Device signature = 0x1e930b

avrdude: safemode: Fuses OK

avrdude done.  Thank you.


C:\WinAVR-20100110\bin>avrdude -p t85 -c usbtiny

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.03s

avrdude: Device signature = 0x1e930c
avrdude: Expected signature for ATtiny85 is 1E 93 0B
         Double check chip, or use -F to override this check.

avrdude done.  Thank you.
```

Programming with WinAVR

Open programmers notepad.

All Programs > WinAVR-2010010 > Programmers Notepad

Select C/C++, Write your code, Save you code as a Bink.C file.

Modify the **Makefile** template by selecting in Windows

All Programs > WinAVR-2010010 >

MFile[WinAVR] (Template pops open)

Select MCU Type > ATtiny85

Change Programmer > stk500v2

Enable Editing of Makefile > usbtiny

Change Port > usb

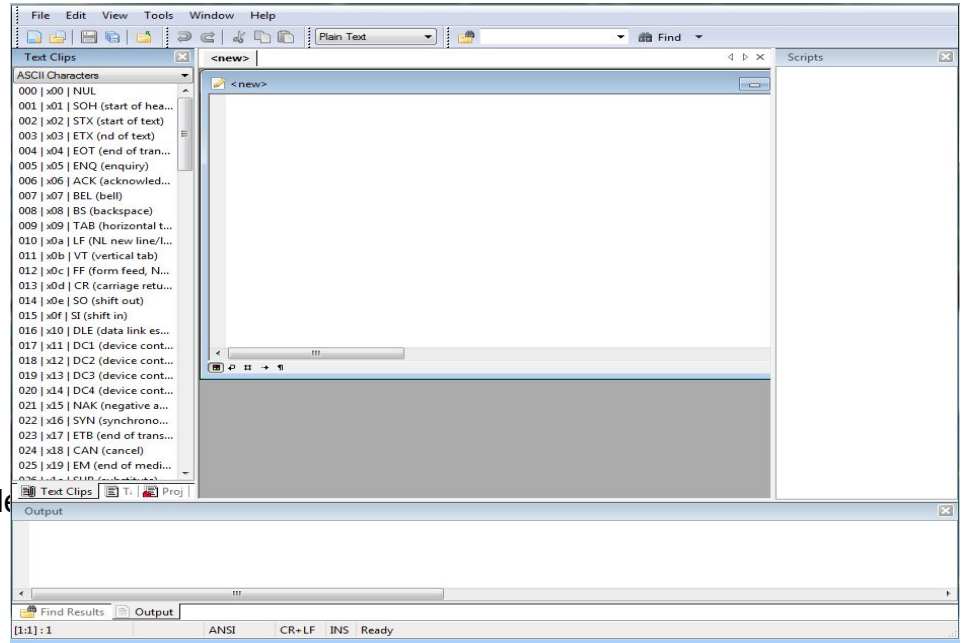
File Save as into the folder where the Blink.C

Plug programmer is plugged into laptop USB and 6 pin cable is plugged into the ISP Target Bd.

Tools > [WinARV] Make All

Watch for errors in Results Box - No Errors proceed

Tools > [WinARV] Program



Software Tracking and Change Control

Device	Program	Date	File Size	Max Size	Available	Location	Issues - Notes
Attiny44-1	LM35_Temp_Atiny85_84_v1	10/13/2013	2700	4096	1396	Clear Bd	Used arduino1.0.5 version to compile
Attiny44-3	LM35_Temp_Atiny85_84_v1	10/13/2013	2700	4096	1396	Clear Bd	Used arduino1.0.5 version to compile
Attiny44-2	N/A		0	4096	4096	?	
Attiny84-1	LM35_Temp_Atiny44_V2_Release1	10/12/2013	2700	8192	5492	ISP BD 2	Used arduino1.0.5 version to compile 500ms r
Attiny84-2	LM35_Temp_Atiny85_84_v1	10/13/2013	2700	8192	5492	LM35ProtoBd	
Attiny84-3	main.c (Atiny84_blink)	10/24/2013	144	8192	8048	85-84 Target Bd	WinAVR Programmer Notepad
Attiny84-4	LM35_Temp_Atiny84_44_v3	12/22/2013	2700	8192	5492	Chip Holder	
Attiny45-1	Blank		0	4096	4096		Future Purchase
Attiny45-2	Blank		0	4096	4096		Future Purchase
Attiny45-3	Blank		0	4096	4096		Future Purchase
Attiny85-1	LM35_Temp_Atiny85_84_v1	10/13/2013	2624	8192	5568	ISP BD 1	Used arduino1.0.5 version to compile
Attiny85-2	LM35_Temp_Atiny85_84_v1	10/13/2013	2624	8192	5568	ArduinoBD2	Used arduino1.0.5 version to compile
Attiny85-3	LM35_Temp_Atiny85_84_v1	10/14/2013	2624	8192	5568	BB 4	Used arduino1.0.5 version to compile
Attiny85-4	Blank		0	8192	8192	Chip Hold	
Attiny85-5	LM35_Temp_Atiny85_84_v1	12/18/2013	2624	8192	5568	ISP BD1	Used arduino1.0.5 version to compile
Attiny85-6	Blank		0	8192	8192	Chip Hold	
Attiny85-7	Blink2	11/6/2013	958	8192	7234	830 BD5 Clear	Used arduino1.0.5 version to compile
Attiny85-8	Blank		0	8192	8192	Chip Hold	
Attiny85-9	Blank		0	8192			
Attiny85-0	Blank		0	8192			
Attiny2313-1	Arduino_7_seg_cd4511_tiny2313	10/13/2013	1026	2048			
Attiny2313-2	Arduino_7_seg_cd4511_tiny2313	10/13/2013	1026	2048			
Attiny2313-3	main.c (Atiny2313) portb_t2313	11/12/2013	134	2048			
Attiny2313-4	Blank			2048			
Attiny2313-5	Blank			2048			

Track all MCU's on a spreadsheet or other change control tool.

Number or label MCU's.

Number or label Target boards.