How to use WinAVR for the Microrobot AVR Products(Rev 0.1)

Contents

What is WinAVR?

WinAVR (pronounced "whenever") is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. It includes the GNU GCC compiler for C and C++.

How to Install?

Go to http://sourceforge.net/projects/winavr and download the latest version of WinAVR. Run the file you've downloaded.

Warning : There are many different versions of AVR GCC. Installing more than one version of AVR GCC including 'Maro GCC' causes a problem. Uninstall any existing version of AVR GCC before installing a new version.

How to Use WinAVR

So far, WinAVR supports only the DOS command-line platform. The user should be familiar with DOS commands before using it. During the WinAVR installation, the program installer changes and/or adds some settings in your PC. You can see the added options using the 'set' DOS command.

Once WinAVR is installed, the user can call the installed programs from any folders. It is recommended to create a new folder for each source code for the purpose of simplicity.

1) Download sample source codes at

http://www.microrobotna.com/download/AVR_Source_Codes.zip and copy the file to the WinAVR folder and unzip it.

2) The following folders are created in the C:\WinAVR\AVR_Source_Codes\. The folders are named after the CPU boards.

Inchworm(AT90S4433)
Inchworm(Maro_GCC)
Inchworm(mega)
MR8
MR16
MR161
MR162
MR163
MR2313
MR4433
MR8515
MR8535(AT90S)
MR8535(mega)
MR-Servo8
Owl_Robot(AT90S4433)
Owl_Robot(Maro_GCC)
Owl_Robot(mega)

Note: In the future, there might be some more folders.
Refer to Owl_Robot(mega) or Inchworm(mega) source for the MR-SERVO8 board.

Each folder contains three or more files. The following is the MR2313 folder's contents.

makefile
MR2313.c
MR2313.hex
(Note: The name 'MR2313' above varies in each folder.)

All three files are text format files. You can open them and see the contents.

```
.
.
.
```

```
# MCU name
#MCU = at90s8515
#MCU = at90s8535
#MCU = at90s4433
MCU = at90s2313
#MCU = atmega163

# Output format. (can be srec, ihex, binary)
FORMAT = ihex

# Target file name (without extension).
TARGET = MR2313

# Optimization level, can be [0, 1, 2, 3, s]. 0 turns off optimization.
# (Note: 3 is not always the best optimization level. See avr-libc FAQ.)
OPT = 1
#OPT = s
.
.
.
```

The portion of code above is a sample from a 'makefile'. This is the only area you might consider modifying.

MCU = Your CPU Name.

FORMAT = ihex (Do not change.)

TARGET = Your source code name without extension. If you create new source code and want to compile it, you have to change this entry to your source file name.

OPT = 1 (Change at your own risk. In certain cases, this optimization option may cause unpredictable results. In that case, try other options.)

```
/*-----------------------------------------------------------------------------
 * File: MR2313.C
 * Description: Turns on and off all the ports every 0.5 sec.
 * X-tal frequency = 8 MHz
 *
 * MICROROBOT NA Inc.(www.microrobotna.com)
 * Free Open Source. Free as in 'Free beer'.
 * You can do whatever you want with this stuff.
 * Don't even worry about buying a beer. ~ha ha
 * - James Jeong.
 *---------------------------------------------------------------------------*/

#include <avr/io.h>

#define LED1		PB5

typedef unsigned char byte;
typedef unsigned int word;

// 1msec UNIT delay function
void delay_1ms(unsigned int i)
{
		word j;
		while(i--)
		{
```

```
                j=14268;   // 10Mhz Exteranl Crystal
                while(j--);
        }
}

void ports_init(void)
{
        DDRB = 0xff;            //Configures PORTB as an output port.
        DDRD = 0xff;            //Configures PORTD as an output port.
}

void ports_set(void)
{
        PORTB = 0xff;           //Outputs 0xff to PORTB.
        PORTD = 0xff;           //Outputs 0xff to PORTD.
}

void ports_clear(void)
{
        PORTB = 0;              //Outputs 0 to PORTB.
        PORTD = 0;              //Outputs 0 to PORTD.
}

void start_signal(void)
{
        byte c;
        for(c=5; c>0; --c)
        {
                PORTB &= ~_BV(LED1);  // Bit clear.= Turn On LED1.
                delay_1ms(20);        // 0.2 sec delay.
                PORTB |= _BV(LED1);   // Bit set.= Turn Off LED1.
                delay_1ms(20);        // 0.2 sec delay.
        }
}

int main(void)
{
        ports_init();//Ports Initialization
        start_signal(); // Toggle LED1 five times.

        while(1)      // Keeps toggling all ports every 0.5 sec.
        {
                ports_set();
                delay_1ms(50);
                ports_clear();
                delay_1ms(50);
        }
}
```

The above is example source code.

3) Launch the DOS command line platform.

4) Go to your WinAVR directory and select the folder which is the same as your board name.

5) Type the command below and press ‘Enter’ to compile the source code.

make all

6) Launch the PonyProg2000 program and download the generated .hex file to your board. (Refer to ‘How to use PonyProg for Microrobot AVR Products(Eng).pdf’ for details.)

7) To erase the files generated by the compiler, use the following command :

```
make clean
```

How the Sample Source Code Works

All the sample codes are basically the same. Once the program is run on the CPU board, LED1 is toggled 5 times every 0.2 sec and all the ports are continuously toggled every 0.5 sec. Actually LED1 is a downloading indicator but it can also be used as a general purpose LED with a jumper block. After downloading the .hex file, disconnect the cable from the board and short pins 8 and 10 on the downloading header. This connects LED1 to the MOSI pin (usually PB5). Refer to Fig 1 and Fig 2 below.

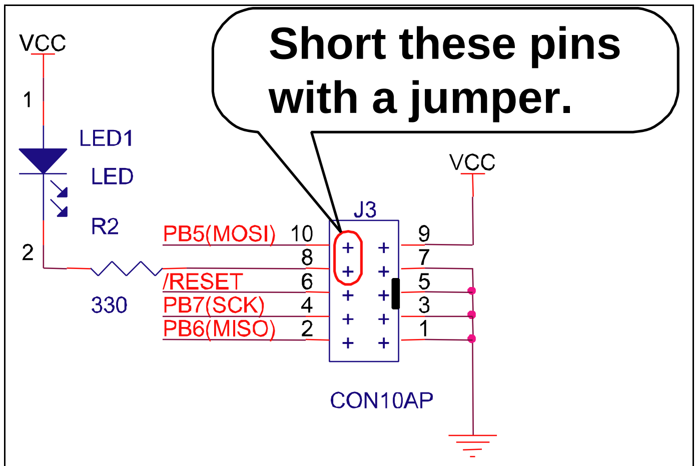

Fig 1. Schematic Diagram of Download Header.

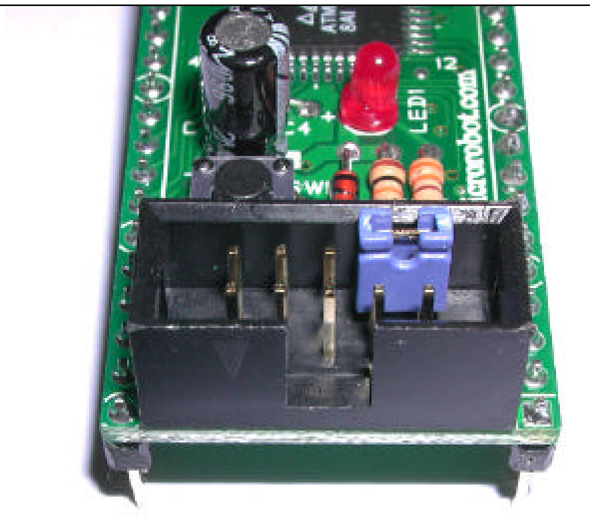

Fig 2. Insert a jumper to use LED1 as a general purpose LED.

Useful Tips

- Just follow the above instructions first before studying the makefile, make and avr-gcc program. These are quite complicated. You don't have to understand them thoroughly the first time. Take your time, you will learn the process gradually.
- If it seems that your AVR CPU is kind of slow, check the bit configuration. Refer to "Security Bit Settings for ATMega Family.pdf" for details.
- Read make.txt which comes with WinAVR.
- http://www.gnu.org/manual/manual.html : Compiler and make manuals.
- Refer to your AVR gcc manual (C:\WinAVR\doc\avr-libc\avr-libc-user-manual\index.html). This file comes with WinAVR.
- Use the "Programmers Notepad" that comes with WinAVR. It is quite a cool editor.