Intro to AVR

Made by Kustiawanto Halim in 2016

Abstract. AVR is a family of microcontrollers developed by Atmel beginning in 1996. These are modified Harvard architecture 8-bit RISC single-chip microcontrollers. AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time. AVR microcontrollers find many applications as embedded systems; they are also used in the Arduino line of open source board designs.

1. Getting Started

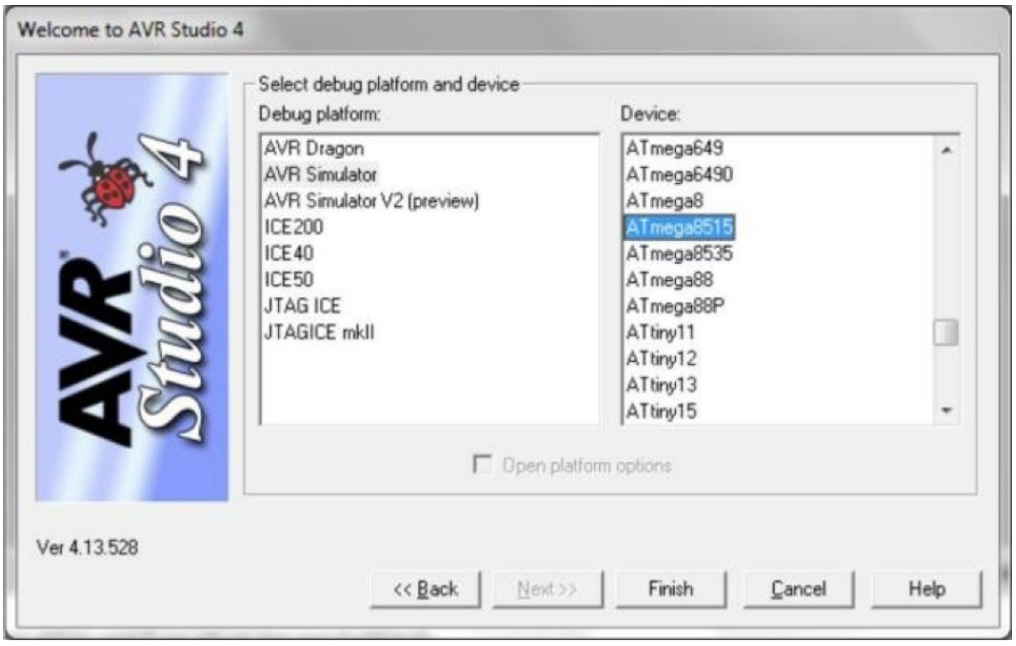
These are the essential steps needed to create a new project in AVR Studio.

a. Open AVR Studio on your computer. Download it at http://ww1.microchip.com/downloads/archive/AvrStudio4Setup.exe

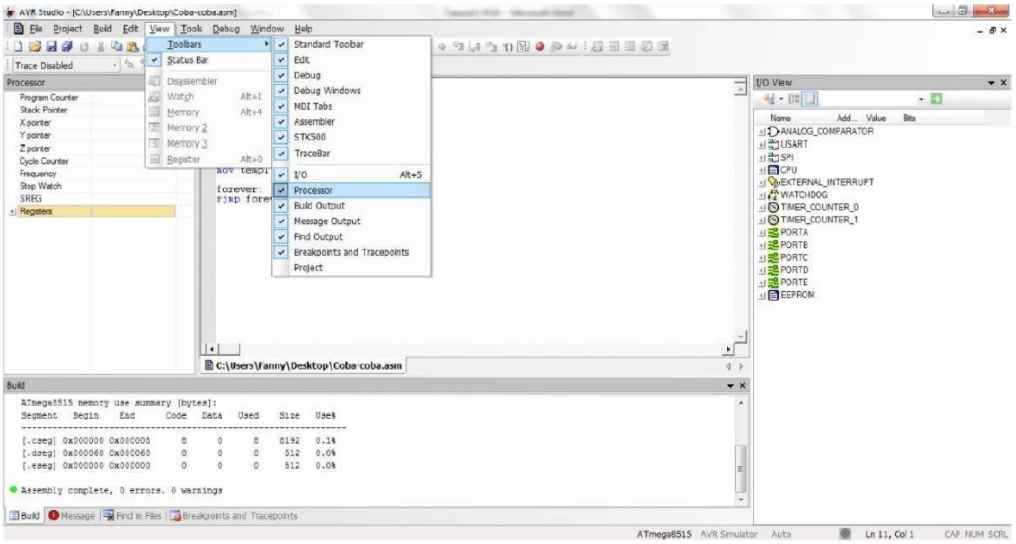
b. Choose New Project

c. In Project Type window, choose Atmel AVR Assembler. Write your project name, and choose the location of your project. Then click next.

d. In Debug Platform window, choose AVR Simulator. In Device Tab, choose ATmega8515. Then click finish.



e. Make sure Processor, Register, Program Memory, and Data Memory window is opened. You can open the window tab by accessing View » Toolbars » Processor.



f. For Register, Program Memory, Data Memory can be opened after doing Debugging process.

g. Now you have successfully created a new AVR Studio Project

2. Introduction to Register

Registers are special storage capacity of 8 bits, which is located in the datapath in the Processor. A register can store values from 0 to 255, or a value of -128 to +127 (using the 7th bit as the sign bit), or a value that represents the ASCII characters. In addition, registers are also useful as a flag, which each of eight bits in the register does not represent a value, but we treat them as a single bit indicating yes/no decisions. There are 32 General Purpose Registers in AVR, they are named R0- R31. This register can be named as you wish (like a variable).

The very first step in writing your assembly language in AVR is to make sure we include the correct header of the device that we use. In this case, we use ATmega8515.

```
.include "m8515def.inc"
```

To rename a register, you can simply write:

```
.def registerku = r16
```

Now, look at the following instruction:

```
ldi registerku, $aa
```

The instruction above will load the value of $aa to register R16 (Load Immediate). This instruction will take a constant value and put it in the desired register. An instruction can also involve two register at the same time. One of the instructions is mov. This instruction will copy the value of a register to another register.

```
.def registerku = r16
.def registerkamu = r15

ldi registerku, $aa
mov registerkamu, registerku
```

ldi (load a constant value to register) operation can only use R16-R31 as destination register (You can't use R0- R15 for ldi operation). For further explanation about register and how to use it, you can check ICO slide in SCeLE.

3. Simple Program

This is a simple AVR program using simple operation.

```
.include "m8515def.inc"
.def temp = r16
.def temp1 = r15
.equ param = 4      ; constant initialization
ldi temp, $0A
subi temp, param
mov temp1, temp     ; substraction operation
forever:
rjmp forever        ;infinite loop
```

To run the following code of program, we need to Build/Assemble your

program. Bellow is several step to Build/Assemble your program:

1. Choose Menu Build » Build (F7) or click on shortcut icon
2. If the program can be build successfully (no error found), proceed by choosing Menu Build » Build and Run or using the following shortcut icon
3. Click View » Memory View » Memory2, View » Memory3, and View » Register.
4. Now you can trace the code by running the program step by step. To run the program step by step you can hit F11.