

Atmel Studio 6

for Basics of Microprocessor technology



Alexey Tukalo,
EFA12SF,
Information Technology,
Savonia University of Applied Sciences

February 2, 2015

Contents

1	Assembling the development kit	2
1.1	STK600 connection	2
1.2	Atmel Studio Settings	2
1.3	Atmel Studio Programming	2
2	Operating the IO pins	5
3	Programs	5
3.1	The first program	5
3.2	The second program	6
3.3	The third program	7

1 Assembling the development kit

First of all we have to study the process of STK600 connection via AVR Dragon to PC.

1.1 STK600 connection

We have to connect PORT A to switches and PORT B to leds. We also have to check that VTARGET, RESET, AREF0 and AREF1 jumpers are on the places.

After that we should make a connection between AVR DRAGON¹ and STK600 via JTAG interface. At the last step the STK600 and Dragon need to be plugged to the PC through USB cable, it also provides enough power for the devices.

1.2 Atmel Studio Settings

We have to open Device Programming window on the Atmel Studio and choose AVR Dragon in Tool's combo-box, ATmega2560 should be placed as a Device on the next form, and Interface has to be JTAG, of course. Apply button has to be pressed after that.

In Fuses tab turn on OCDEN, JTAGEN, SPIEN and turn off all other ones. Set BODLEVEL Disabled, BOOTSZ 4096_1F000 and SUT_CKSEL EXT_XOSC_3MHZ_8MHZ_1KCK_0MS. Go to Lock bits and set all of them to NO_LOCK mode. Press program button.

1.3 Atmel Studio Programming

It is possible to start new project now. The STK600-ATmega2560 template can be found in Atmel-Boards folder on the New Project creation window.

After that we can write the testing code into C file.

```
#include <asf.h>

int i=0;

void main(void)
{
    DDRA = 0x00; // PORTA is input

    PORTB = 0xff; // PORTB off all leds
    DDRB = 0xff; // PORTB is output

    while(1)
    {
        PORTB = PINA; // Led on
        i++;
    }
}
```

To debug the code user has to press CTRL+F5 or go to Debug→Start Debugging. After that the Atmel would upload your program to the device and start it with debugging mode.

¹programator for AVR CPU based boards, allows to use hardware debugging

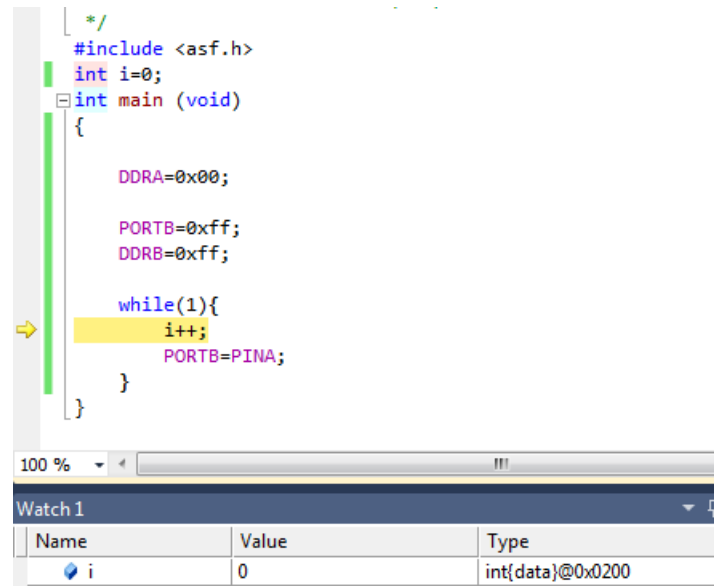


Figure 1: The first loop of the program `i` is still equal initialization value

There are a lot of useful tools provided for debugging, for example it is possible to follow the variables values through Watch window, Figure 1.

There is the first program loop before increment, variable `i` has an initial value 0.

Figure 2 shows us that the value became 1, after increment.

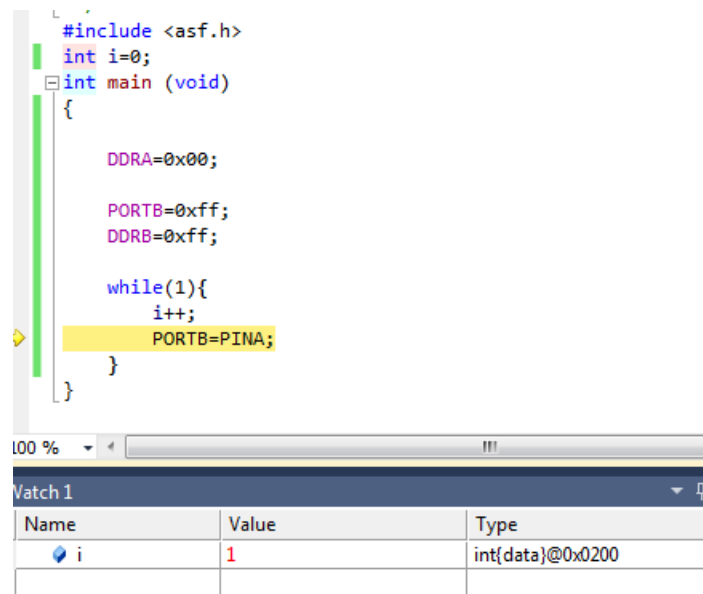
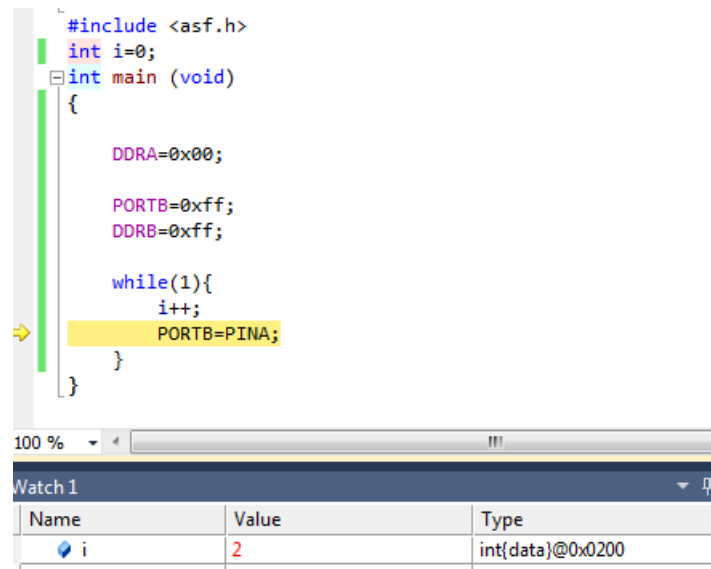


Figure 2: The end of the first loop of the program the variable was incremented

And the variable is again incremented in the second program loop.



```

#include <asf.h>
int i=0;
int main (void)
{
    DDRA=0x00;

    PORTB=0xff;
    DDRB=0xff;

    while(1){
        i++;
        PORTB=PINA;
    }
}

```

100 %

Name	Value	Type
i	2	int[data]@0x0200

Figure 3: i was again incremented at the second loop

We also can see changes on the registers. When one of the switches is pressed the correspondent value in register was forced to change.

R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x02
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0xCF
R25	0x00
R26	0x02

Figure 4: R24 was changed from 0x00 to 0xCF

When the button is released the value became initial value.

2 Operating the IO pins

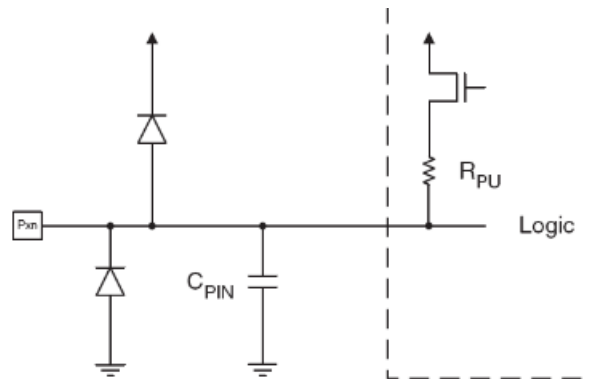


Figure 5: Diagram picture

In the figure there is written a general form of port pins for all AVR digital I/O ports. All of them have their individual selectable pull-up resistor with a supply-voltage invariant resistance, few protection diodes to both Vcc and Ground. There are three I/O memory address locations for each port: Data Register PORTxn (Pxn), Data Direction Register DDRx and Port Input Pins PINx. A lower case x represents the port number, and a lower case n represents the bit number.

Example:

If we will define that PORTB (PORT[x] form) will handle LEDs, that means that each pin of port B (PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7) have each own Led accordingly LED port (LED0, LED1, LED2, LED3, LED4, LED5, LED6, LED7). If we setted SWITCHES to PORTA (PORT[x] form), that means that (PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7) configured to read a signal from switches (SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7) and write them to A register. And after that if we want to switch on the led with the same number as a number of pressed button, we have to read the data from PINA (PINx form) and set the value to the Data Register of Switches (PORTB).

```
PORTB=PINA;//LedOn
```

Or in some cases we can select a necessary bit just calling that in this form: PORTB2

where we call the bit number 2 in PORTB Data Register.

Port Input Pins I/O location is only for reading, two other locations can be used for reading/writing.

However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

3 Programs

3.1 The first program

```
#include <asf.h>
#include <util/delay.h>

#define SetBit(port, bit) {port = port &~( 1 << bit);}
```

```

#define ResetBit(port, bit) {port = port | ( 1 << bit);}

void buttonsOnOff(int i);

int main (void)
{
    int i;
        board_init();
        DDRA = 0x00;
        PORTB = 0xff;
        DDRB = 0xff;

    while(1)
    {

        for (i=0;i<8;i++)
            buttonsOnOff(i);

        for (i=7;i>=0;i--)
            buttonsOnOff(i);

    }

}

void buttonsOnOff(int i){
    int j, stopLoop=1;

        SetBit(PORTB, i);
    for (j=0;j<50;j++){
        stopLoop=PINA!=0xff?—stopLoop: stopLoop;
        while(PINA!=0xff);
        while(stopLoop<0){
            stopLoop=PINA!=0xff?—stopLoop: stopLoop;
            while(PINA!=0xff);
        }
        _delay_ms(10);
    }
    ResetBit(PORTB, i);
}

```

3.2 The second program

```

#include <asf.h>
#include <util/delay.h>

#define SetBit( port, bit) { port = port &~( 1 << bit); }
#define ResetBit( port, bit) { port = port | ( 1 << bit); }

void AnotherWay(int i, int j);

void oneWay(int i, int j)
{

```

```

    int k;
    for (i=j; i<8; i++)
    {
        SetBit(PORTB, i);
        for (k=0; k<300; k++){
            if (PINA!=0xff){
                ResetBit(PORTB, i);
                while (PINA!=0xff);
                j=i;
                AnotherWay(i, j);
            }
            _delay_ms(10);
        }
        ResetBit(PORTB, i);
    }
    oneWay(0, 0);
}

void AnotherWay(int i, int j)
{
    int k;
    if (j<0)
        j=8;
    for (i=j-1; i>=0; i--)
    {
        SetBit(PORTB, i);
        for (k=0; k<300; k++){
            if (PINA!=0xff){
                ResetBit(PORTB, i);
                while (PINA!=0xff);
                j=i;
                oneWay(i, j);
            }
            _delay_ms(10);
        }
        ResetBit(PORTB, i);
    }
    AnotherWay(8, 8);
}

int main (void)
{
    int j=0, i=j;
    board_init();
    DDRA = 0x00;
    PORTB = 0xff;
    DDRB = 0xff;
    oneWay(i, j);
}

```

3.3 The third program

```

#include <asf.h>
#include <util/delay.h>

```



```
#include <math.h>

#define F_CPU 16000000

int main (void)
{
    unsigned char bCounter=0;
    board_init();

    while(1)
        if(PINA!=0xff) // a key is pressed
        {
            bCounter++;
            while(PINA!=0xff); // wait until released
            PORTB=~bCounter;
        }

    return 0;
}

// board_init

void board_init(void)
{
    DDRA = 0x00; // input
    PORTA = 0xff; // pull up to 5V resistor to switch

    DDRB = 0xff; // output for leds
    PORTB = 0xff; // turn off
}
```