Abu Dhabi University

Microprocessors and Firmware Programming

Lab Report 3
Introduction to AVR Studio and HAPSIM

Author:
Muhammad Obaidullah
1030313

Supervisor:
Dr. Mohammed Assad Ghazal

Section 1

June 16, 2012

Abstract

In this lab we were introduced to AVR Studio, HAPSIM, and Basic ATMega16 connection through an example of a simple blinking LED problem.

1 Introduction

In First Exercise, we were introduced to the AVR Studio IDE, and HAPSIM. We were taken through steps of how to install and run AVR Studio and HAPSIM. Additionally, we coded a simple blinking LED program and simulated it in HAPSIM.

In Second Exercise, we were introduced to the basic connections of ATMega16. E.g. VCC, GND, XTAL etc. Then we connected the JTAG pins of the ATMega16 to the JTAG MKII programmer. Finally, we downloaded our code on the ATMega16 using the JTAG MKII programmer.

2 Experiment Set-up

The ATMega16 chip was already mounted on a safety bracket. We had to place the bracket with the micro-controller on to the breadboard. Then we connected the micro-processor to the JTAG MKII programmer and the LEDs as shown in the *Figure 1* and *Figure 2*.

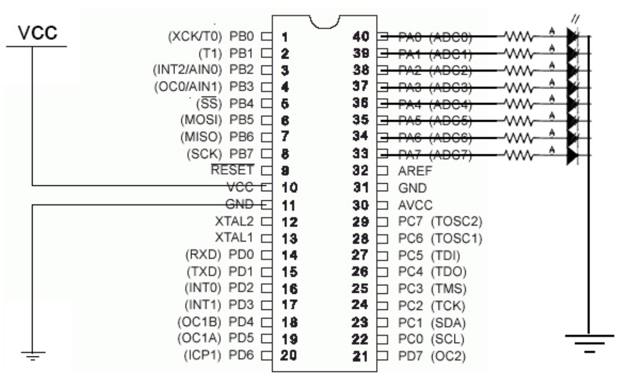


Figure 1: This is how we connect LEDs to the ATMega16

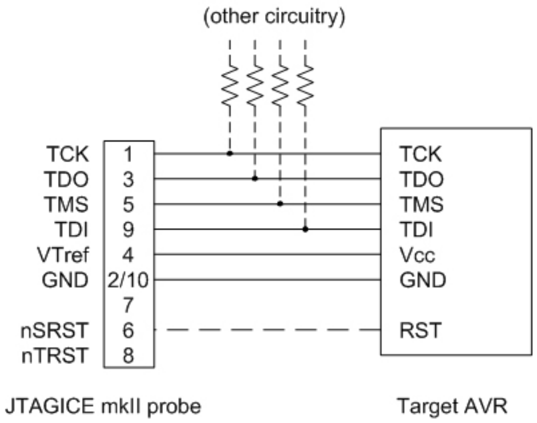


Figure 2: Figure showing connections from ATMega16 to the JTAG pins

3 List of Equipment used

- ATMega16 micro-controller chip.
- JTAG MKII programmer.
- Wires.
- Breadboard.
- Mounting bracket for micro-controller.

- 8 LEDs.
- 8 330 Ω resistors.
- 5V power supply.
- AVR Studio IDE.
- HAPSIM.

4 Procedure

4.1 Exersice 1

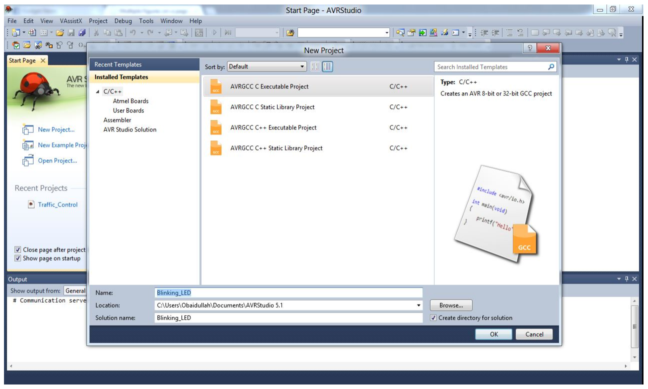
- Open setup files and install AVR studio and HAPSIM by following the screen instructions.
- Start AVR Studio and click on File/New/New Project.
- Write the following code into the AVR .c file.

```
#include <avr/io.h>
#include <util/delay.h>
#define LED_Off 0
#define LED_On 1
int main()
{
int current = 0;
PORTA = 0xFF;
while(1)
{
switch (current)
{
case LED_Off:
{
PORTA = 0x00;
current = LED_On;
_delay_ms(500);
}
case LED_On:
{
PORTA = 0xFF;
current = LED_Off;
_delay_ms(500);
}
}

}
}
```

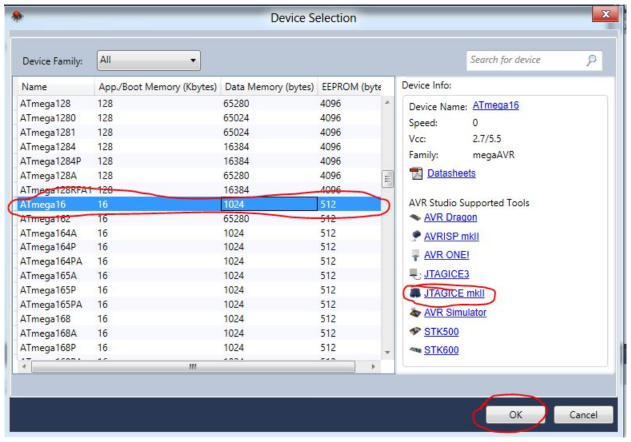
- Open HAPSIM.
- Choose your Micro-controller, in this case its ATMega16.
- open 10 LEDs.

- Goto Options/LED settings and change the options as shown in fiqure 3.
- Simulate your code by opening the .c file you saved from AVR studio.


Start Page - AVRStudio
New Project
Recent Templates
Installed Templates
C/C++
Atmel Boards
User Boards
Assembler
AVR Studio Solution
AVRGCC C Executable Project
AVRGCC C Static Library Project
AVRGCC C++ Executable Project
AVRGCC C++ Static Library Project
Type: C/C++
Creates an AVR 8-bit or 32-bit GCC project
OK
Cancel

Name: Blinking_LED
Location: C:\Users\Obaidullah\Documents\AVRStudio 5.1
Solution name: Blinking_LED
Create directory for solution

Figure 3: Creating a new project in AVR Studio

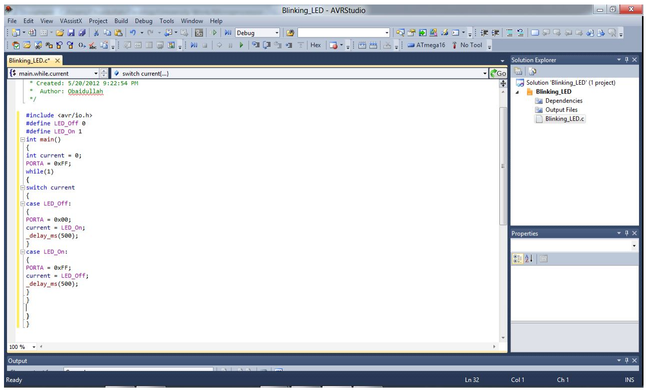


Device Family: All

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (byte
ATmega128	128	65280	4096
ATmega1280	128	65024	4096
ATmega1281	128	65024	4096
ATmega1284	128	16384	4096
ATmega1284P	128	16384	4096
ATmega128A	128	65280	4096
ATmega128RFA1	128	16384	4096
ATmega16	16	1024	512
ATmega162	16	65280	512
ATmega164A	16	1024	512
ATmega164P	16	1024	512
ATmega164PA	16	1024	512
ATmega165A	16	1024	512
ATmega165P	16	1024	512
ATmega165PA	16	1024	512
ATmega168	16	1024	512
ATmega168A	16	1024	512
ATmega168P	16	1024	512

Device Info:
Device Name: ATmega16
Speed: 0
Vcc: 2.7/5.5
Family: megaAVR

Figure 4: Choosing the right debugger/programmer is essential



```
 * Created: 5/20/2012 9:22:54 PM
 *  Author: Obaidullah
 */

#include <avr/io.h>
#define LED_Off 0
#define LED_On 1
int main()
{
int current = 0;
PORTA = 0xFF;
while(1)
{
switch current
{
case LED_Off:
{
PORTA = 0x00;
current = LED_On;
_delay_ms(500);
}
case LED_On:
{
PORTA = 0xFF;
current = LED_Off;
_delay_ms(500);
}
}

}
}
```

Figure 5: Writing the code in AVR Studio

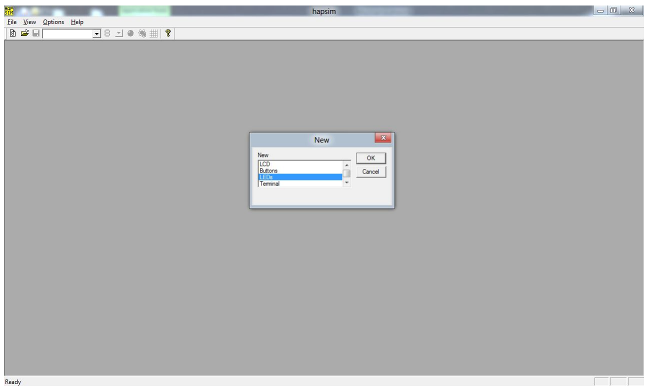
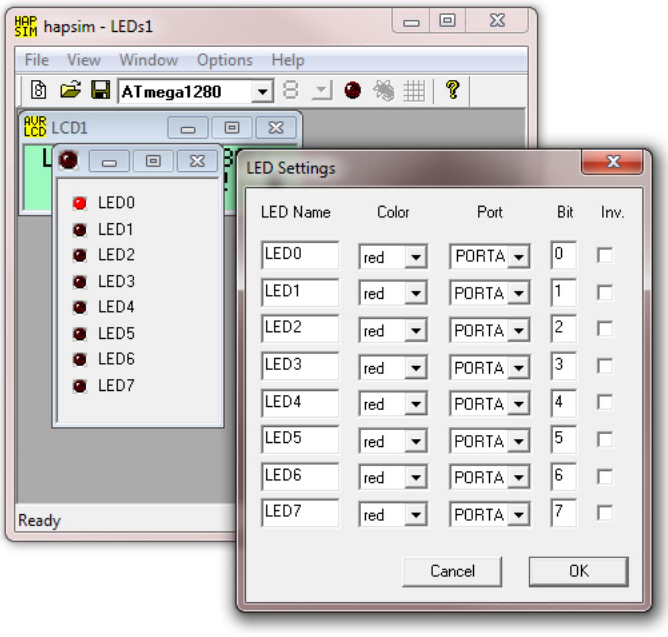


Figure 6: Creating LEDs in HAPSIM

4.2 Exercise 2

- Mount the Micro-controller with the bracket onto the breadboard.
- Connect the micro-controller pins to the appropriate JTAG pins as shown in *Fiqure 1*.
- Start AVR Studio and find the program you were previously working on.
- Compile and build the code.
- If the program doesn't work, Troubleshoot the problems and try again.



LED Name	Color	Port	Bit	Inv.
LED0	red	PORTA	0	☐
LED1	red	PORTA	1	☐
LED2	red	PORTA	2	☐
LED3	red	PORTA	3	☐
LED4	red	PORTA	4	☐
LED5	red	PORTA	5	☐
LED6	red	PORTA	6	☐
LED7	red	PORTA	7	☐

Figure 7: Change all the ports to PORT A because you are using PORT A as your output port for the LEDS and then change the bit number sequentially for all the bits of Port A

5 Results and Discussions

At the end of these exercises we got the following results:-

- The AVR and HAPSIM are easy to install and the version of AVR Studio to be installed is 4.19.
- The program we uploaded to the micro-controller is stored in the flash memory and gets re-written when we upload another one from build and compile button in AVR Studio.
- The code we ran on the micro-controller allowed the LEDs connected to the Port A to be On for 0.5 seconds and Off for another 0.5 seconds.
- For using a software delay you have to include a header file “util/delay.h”.
- “delay..ms()” is the command for software delay where the CPU is kept busy doing a useless loop for the given milli-seconds

6 Conclusion

- All the LEDs have to be connected to the Port A because, we have set the Port A as output and not the other ports.
- The LEDs have to be connected in series with a 330 Ω resistor in order to prevent LED from being burned down of high current coming from the micro-processor pin.
- Micro-controller is very sensitive to minute changes in voltages and voltage spikes. Thus, we needed to have a mounting bracket for it.

- AVR Dragon cannot program as many micro-controllers as JTAG MKII.
- AVR Dragon and JTAG MKII are the programmers/debuggers which provide a bridge between computer's USB interface to the micro-controller pins.
- By using the software delay we are keeping the micro-controller busy for the prescribed time but we can use interrupts and hardware timers which come with the micro-controller to do the same task and that will keep the micro-controller free to do other tasks.