MILESTONE 1: INTRO TO THE MCU

LAB OBJECTIVES

1. Understand how to set up the development environment for Atmel MCU.
2. Gain a familiarity with the AT90USBKey microcontroller.
3. Understand how to program and execute the program on the microcontroller.

DELIVERABLES

1. Demonstrate the ability to upload a program to the microcontroller to the teaching assistant
2. Knight Rider Display by function of waiting
   a) Set up and create a LED display for a 'Knight Ride' by a function of waiting, see http://www.youtube.com/watch?v=bis0AWuZCrs .
   b) Provide a demonstration of your 'knight rider display' with 8 LEDs continuously appearing to shift from bit 0 to bit 7 repeatedly.
3. Complete Milestone 1 assignment at the end of this document.

INSTRUCTIONS

1. SET UP THE ENVIRONMENT

If you have your own computer with a USB port on it, and you want to set up the development environment for the microcontroller, please following the instructions. Note that these files do not support systems after XP (Vista or Win7).

a) Do not plug in your AT90USBKey yet. Download and install the following software:
   i. Download and install WinAVR from the following URL: http://sourceforge.net/projects/winavr/files/ . WinAVR provides source files (.h etc.) for AVR Studio 4. So it should be installed before the installation of the AVR Studio 4.
   ii. Download and install AVR Studio 4.18 SP1 from the following URL: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725 . The software supplies integrated development environment. If asked to installed the driver, click on Yes.
   iii. Download and install FLIP 3 from the following URL: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886 . This software will be used to load our new programs (machine language) onto the USB key. Note that do not forget the location of the installation.

b) Plug in your AT90USBKey. Depress the RST and HWB buttons at the same time on the microcontroller and then FIRST release the RST button then SECOND release the HWB

button. If you don't know which button is which, look at the white lettering next to the button. Now you will be asked to install the driver of the USB Key. The location of the driver is in '…FLIP/ usb'.

c) Open FLIP. If asked to install virtual machine, download and install from the following URL: http://java.com/en/download/index.jsp

2. HARDWARE SUMMARY

After ordering your own ATUSBKey you will probably be wondering what comes in the box. The AT90USBKey Demo-Kit contains the following components:

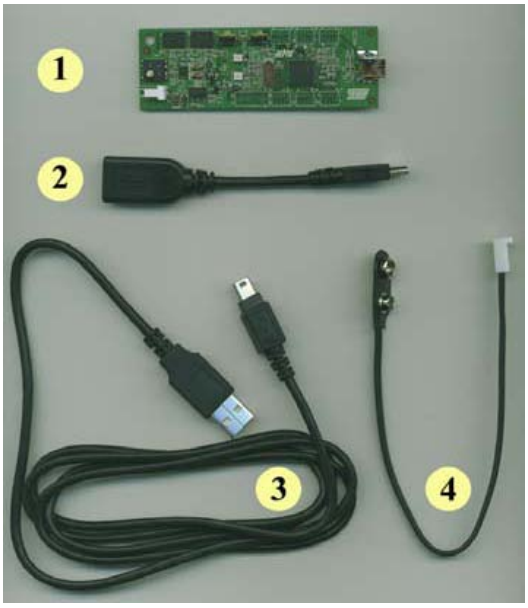
1. AVR USB Key - USB On-The-Go Microcontroller

- 128KB Flash Program Memory
- 8KB SRAM
- 4KB EEPROM
- 8 Channel 10-bit A/D-converter
- JTAG interface for on-chip-debug
- Up to 16MIPS throughput at 16MHz
- 2.7 – 5.5 Volt operation
- USB full/low speed device/host/OTG interface
- 4+1-ways joystick
- 2 Bi-Color LEDs
- Temperature sensor
- System clock: 8 MHz crystal
- Onboard reset button
- Onboard HWD button (force bootloader section execution)

2. Host Adapter

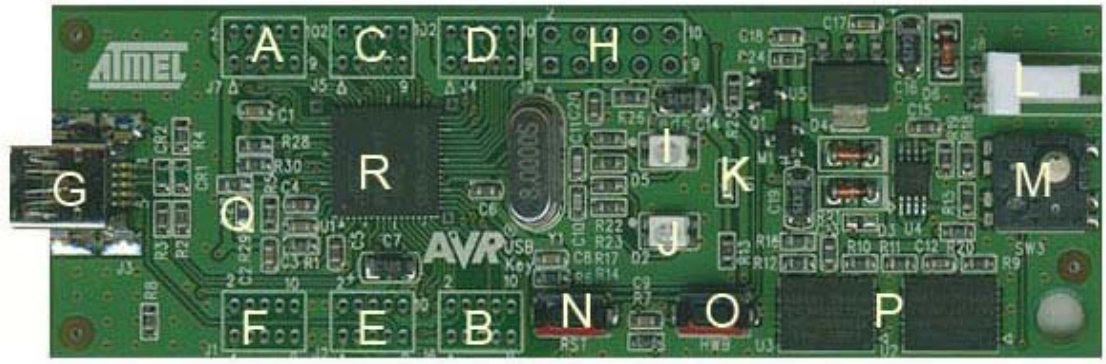
3. USB Cable

4. Battery Clip



The cables are pretty basic. Use (3) to connect your key to the development platform, and use (4) to connect the key to a power supply once you are done programming

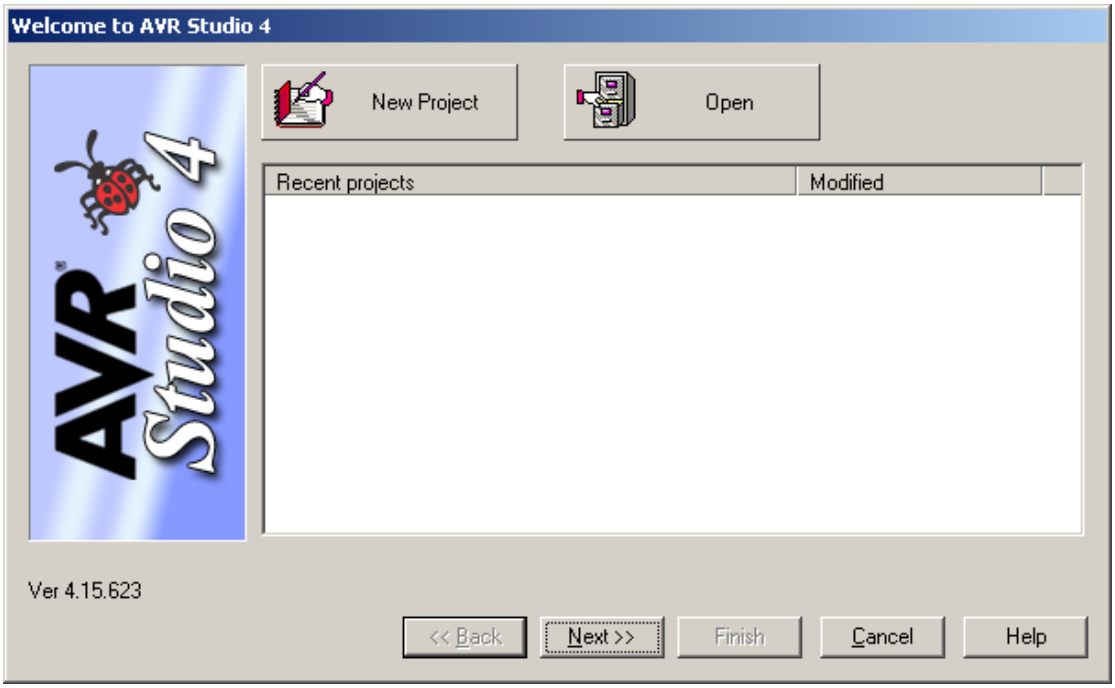
The USB key, however, is more complicated and deserves to have its key components spelled out. The AVR USB Key has the following layout:



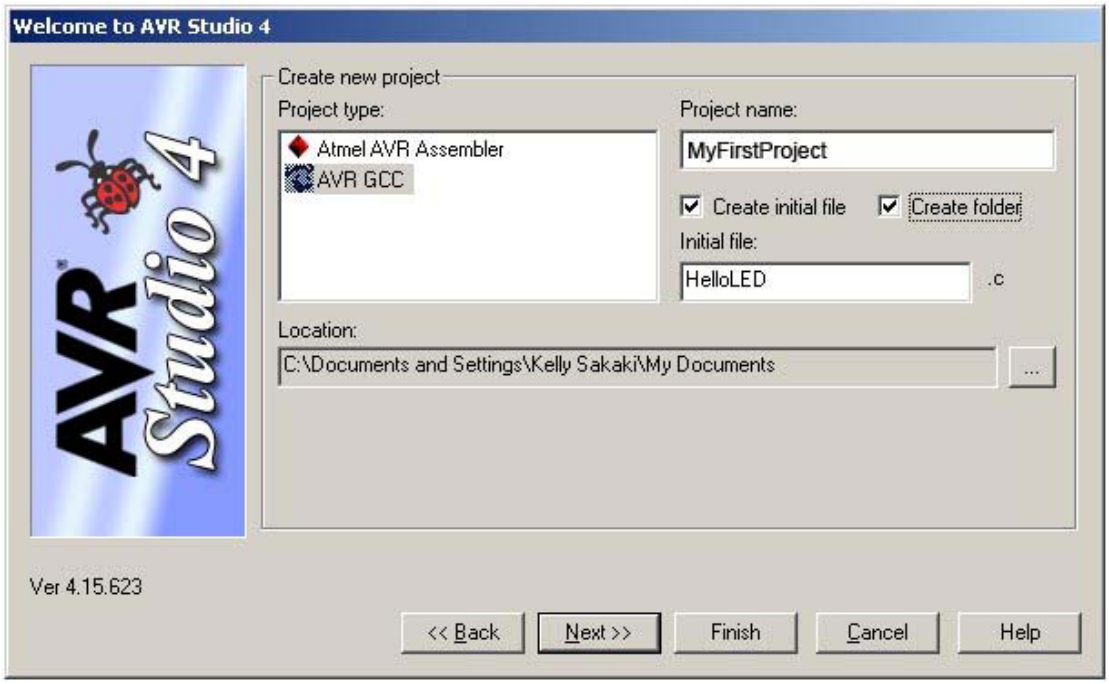
LABEL	Description
A-F	Ports A through F
G	USB MiniAB connector
H	JTAG port
I-J	Bi-colored LEDs (Red/Green LEDs)
K	Power LED
L	External power connector
M	4+1 way joystick (four directions + push button)
N	On board reset button
O	On board HWB button to force bootloader section execution on reset
P	Data flash memory
Q	Temperature sensor
R	AT90USB1287

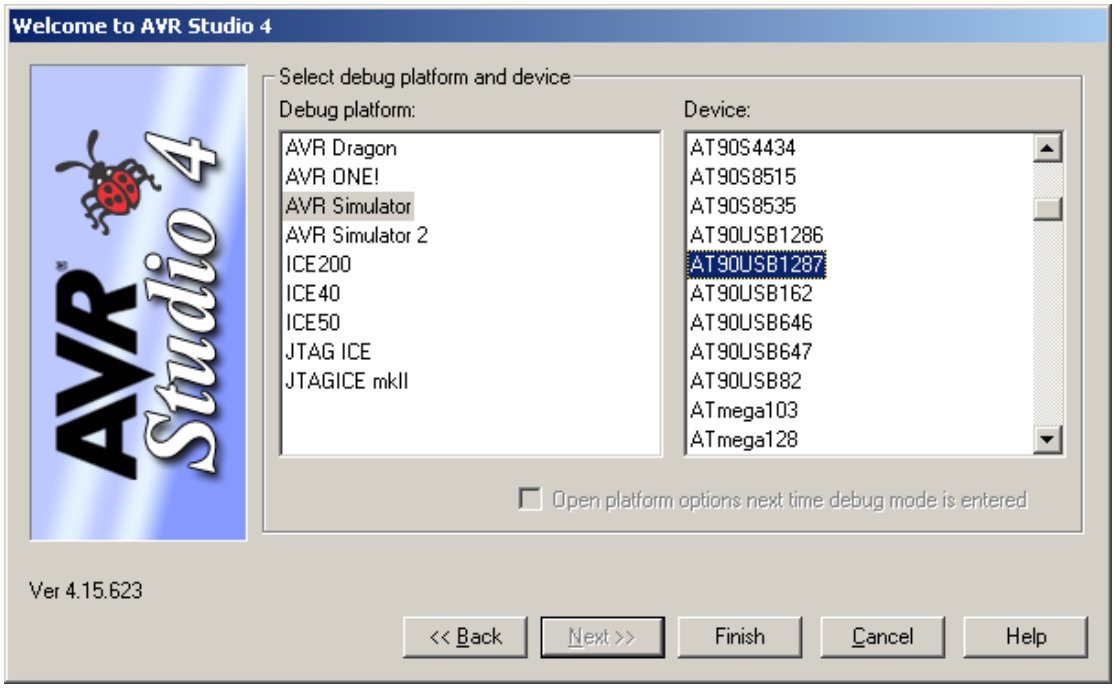
3. CREATING YOUR FIRST PROJECT

a) Open AVR Studio 4, and click on New Project select Next



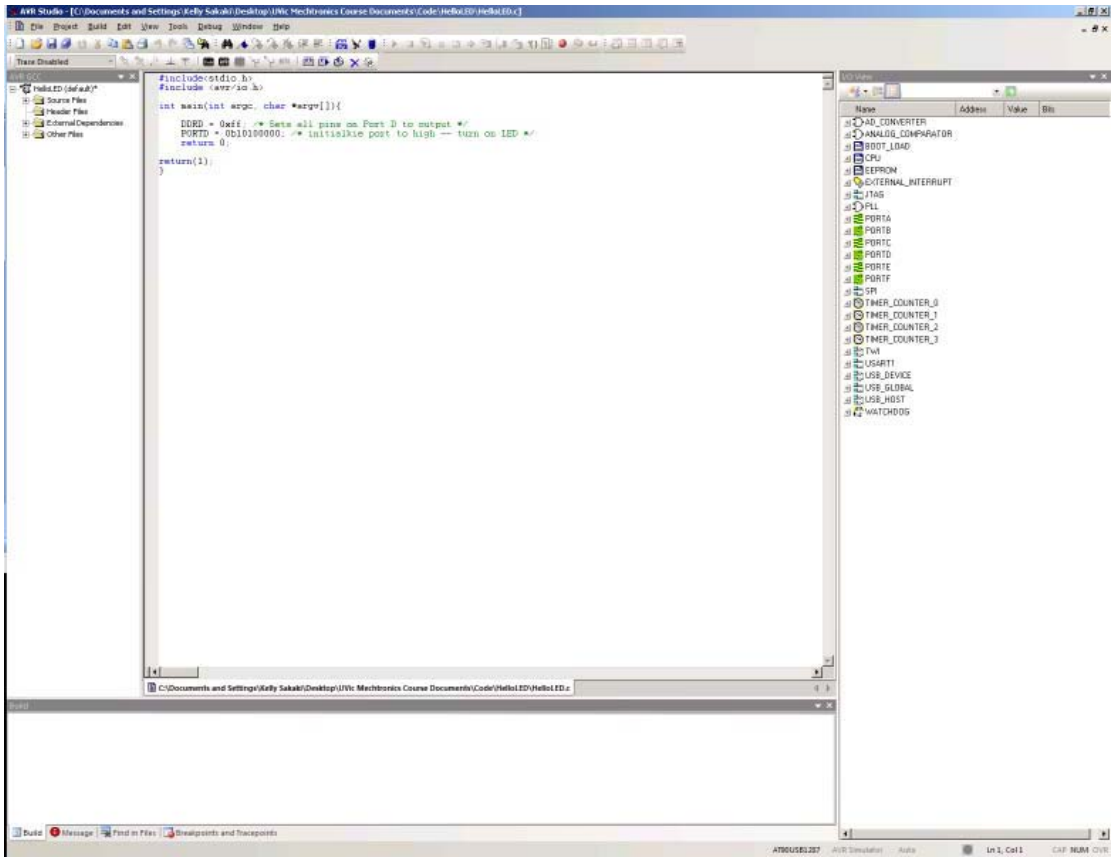
Click on AVR GCC to use the AVR GCC compiler (for C programming). If you wanted to use Assembly language you would have clicked on Atmel AVR Assembler. Fill in the remaining boxes in the following screen and select a location where you want to store your documents. I suggest creating a C:\Documents and Settings\[Your computer]\My Documents\MECH458 \Software\ folder to hold all of your project files. If you are in the lab you will have to store the files in your network drive.





Select AVR Simulator, and then select the microcontroller that is on the ATUSB90 demo board which is the AT90USB1287 then click next.

You should see the following screen:



The IDE is broken up into 4 areas:

1. The programming window
2. The directory structure of your project

3. The results after you compile your project

4. The inputs and outputs of the microcontroller and simulated peripheral devices on the microcontroller

WINDOW 1: PROGRAMMING AREA

This is where you will write all of your code. You can have multiple files open and also access any of these files that you have open by selecting the tab in the bottom left hand corner of Window 1.

WINDOW 2: DIRECTORY STRUCTURE AND PROJECT FILES/OBJECTS

This window shows you all the entire structure of your project. You can:

- See all of your files that are associated with this project,
- Add/delete C files or H (header) files to your project
- Rename files or directories using the left click option

WINDOW 3: COMPILER RESULTS AND DEBUGGING INFORMATION

This window will show you the progress of the AVR C compiler when you 'build' your project (*i.e.* create the executable file). If you have errors in your project, the compiler will tell you, provide a line number the error resides on and indicate a hint of what the problem may be. These errors will be highlighted in RED. You can click on the error and the IDE will go to the nearest location it think s the error is on. Warnings are suggestions that the compiler has found and MAY cause problems during runtime. These are highlighted in yellow, and it is advised to fix these before you upload and execute your program on your microcontroller.

WINDOW 4: INPUT AND OUTPUT VIEW

This is a VERY USEFUL window, and allows you to see all of the registers of the microcontroller, and how each register's bits are broken up. You will become more familiar with this window as you start programming more frequently. The registers' status can also indicate whether the bits are activated or not activated in the AVR Simulator.

b) Write the following code in the programming area and ensure that the information block is filled out:

```
################################################################################
# MILESTONE : 1
# PROGRAM : 1
# PROJECT : MyFirstProject
# GROUP : X
# NAME 1 : First Name, Last Name, Student ID
# NAME 2 : First Name, Last Name, Student ID
# DESC : This program does…[write out a brief summary]
# DATA
# REVISED
################################################################################
```

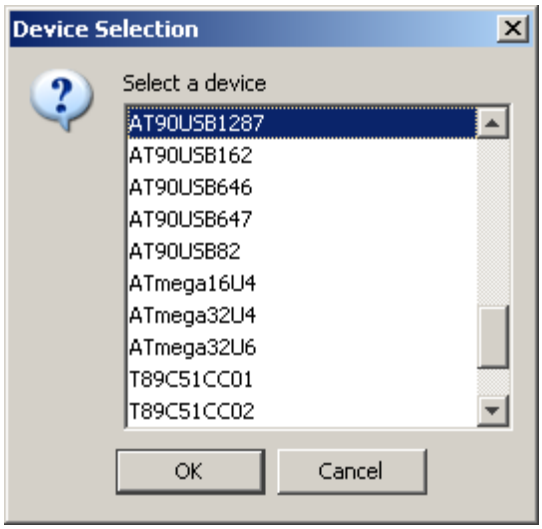
```
#include <stdlib.h> // the header of the general purpose standard library of C programming language
#include <avr/io.h>// the header of    i/o port
################### MAIN ROUTINE ###################
int main(int argc, char *argv[]){
DDRD = 0b11111111; /* Sets all pins on Port D to output */
PORTD = 0b10100000; /* initialize port to high – turn on LEDs */
return (0); //This line returns a 0 value to the calling program
// generally means no error was returned
}
```

Click Build /Build, but WE NEED TO BE SPEEDY so just hit F7.

Good! So what you've done now is create the written program. You have also compiled the program. This means you have created the machine language that the microcontroller understands and will need so it can execute your program.

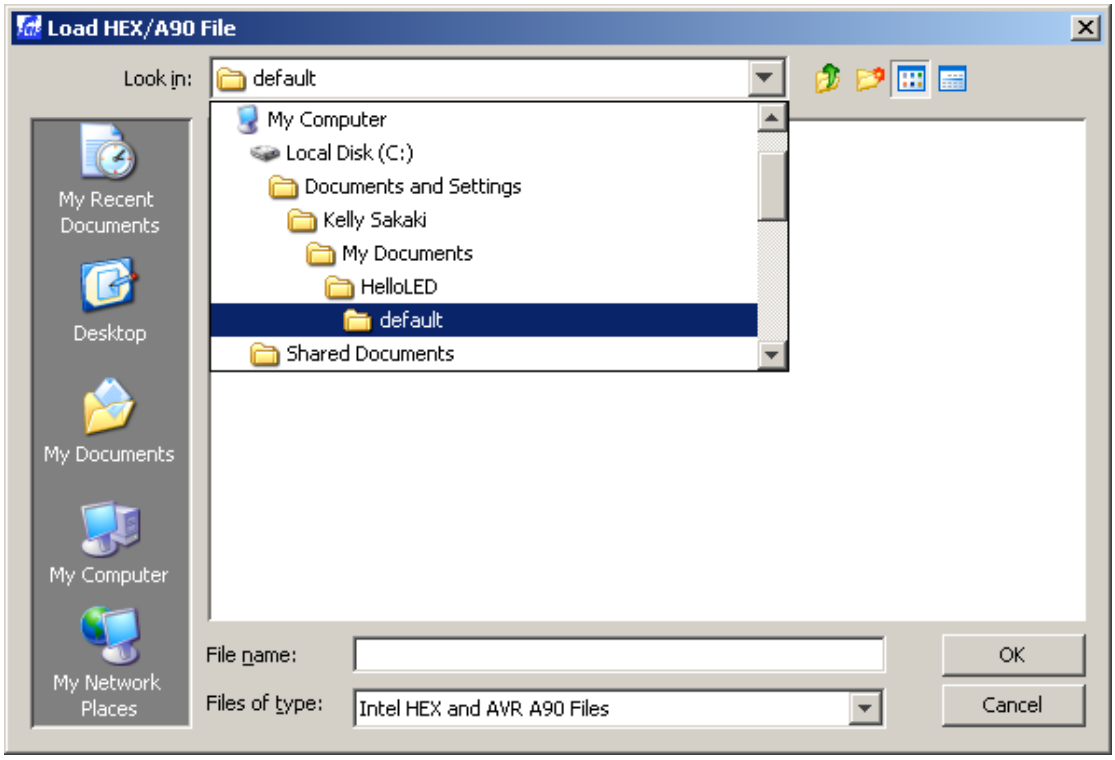
Open FLIP by clicking on the desktop icon, or opening FLIP from the Windows Start menu. Select the type of microcontroller that we will upload the program.

Device/Select



c) UPLOAD YOUR PROGRAM TO THE MICROCONTROLLER

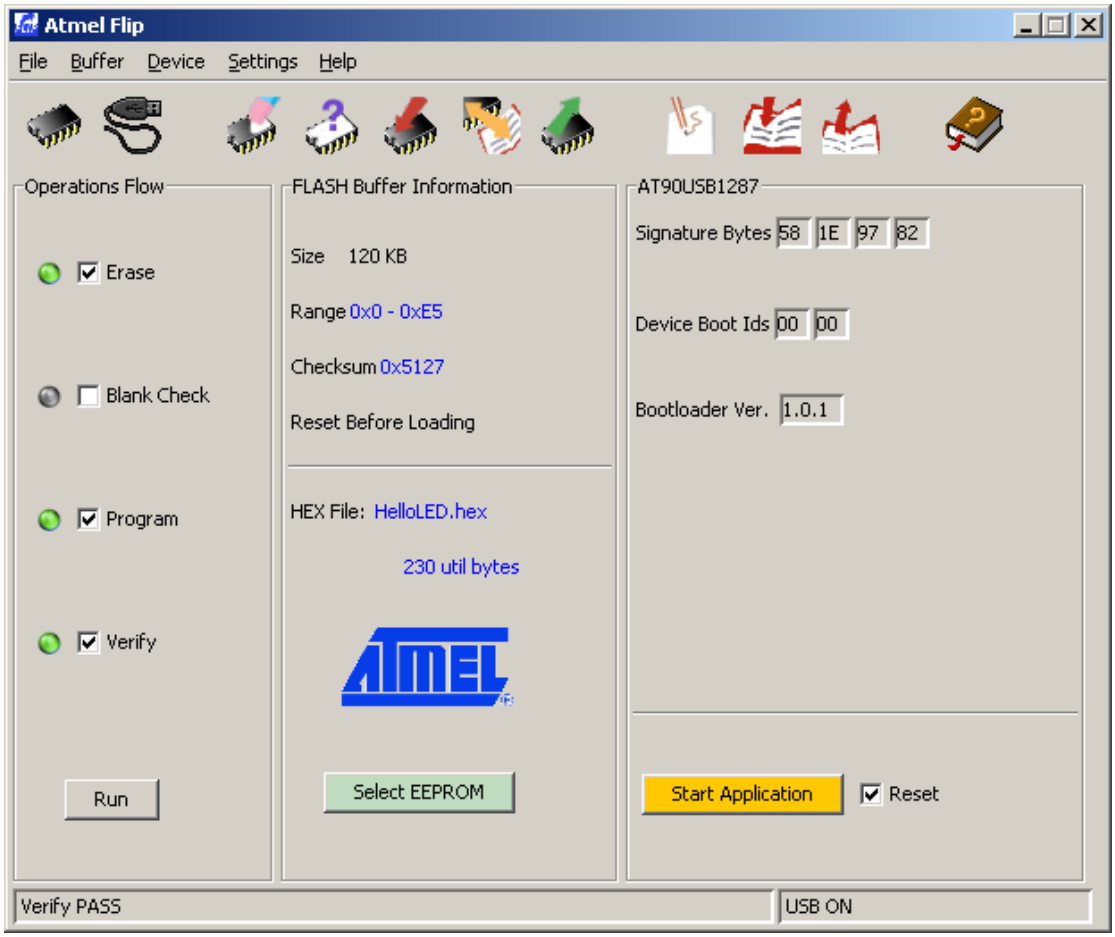
Select File / Load HEX File or since WE WANT TO BE SPEEDY hit CTRL L to load your .hex file which is the machine code for the micro, and select the location where the HEX file is stored. On mine it was:



A file with the name of your Project should appear such as HelloLED.hex.
Now we have to establish a communications channel with the microcontroller so we can upload the program to the microcontroller memory. Ensure that your microcontroller is connected to your computer's USB port. In your kit there should be a USB connector. Ensure the USB connected is connected to the microcontroller in the following location to allow communications and provide power to the microcontroller. Depress the RST and HWB buttons at the same time on the microcontroller and then FIRST release the RST button then SECOND release the HWB button. If you don't know which button is which, look at the white lettering next to the button.

Click on Settings / Communications / USB but SINCE WE WANT TO BE SPEEDY hit CTRL U to open communications to the microcontroller, and the click on RUN (no speedy shortcut for this). You will see a bar indicating that the programming is being uploaded. If you have problems, please ask your TA for assistance. There have been some issues with the USB drivers in the past. Then to activate the program depress the RST button.

WARNING: In the future, be aware that the program will execute as soon as you press the RST button. Take this into consideration when using sensors and ESPECIALLY actuators.



NOTE: Each time you modify your program, save it, and then rebuild it you have to go through the 'UPLOAD YOUR PROGRAM TO THE MICROCONTROLLER' routine.

d) PROGRAM OUTPUT

You should see some LEDs on the microcontroller light up. Note that these LEDs are bi-colored and attached to PORTD. Therefore you can use these LEDs for output or debugging. You have successfully completed your first program. **Congratulations.**

4. Knight Rider Display

```
#include <util/delay_basic.h>


void delaynus(int n)      %%% delay microsecond
{
 int    k;
 for(k=0;k<n;k++)
   _delay_loop_1(1);
}
```

```
void delaynms(int n)    %%%delay millisecond
{
  int k;
  for(k=0;k<n;k++)
    delaynus(1000);
}
```

5. STILL HAVING PROBLEMS

If you have problems, attempt to remove all the software you installed, reboot the computer and start the software installation process again. Ensure your USBKey has been removed. If you have problems again, please contact the TAs. Don't worry, if you have problems. Unfortunately the first few times you use ANY hardware/software programs that involve communications there are ALWAYS problems (aka challenges). After several years of this, you will feel strange if you don't get errors the first time. If possible, please bring your computer (box only) to the first lab and let us know in advance you are doing this. We'll help you get it setup. Once again, it is advised to use WinXP. We are not supporting Windows Vista at this time.

References:

1, Kelly Sakaki, "MILESTONE 1: INTRO TO THE MCU", 2009.

2, Alexander Hoole, " AT90USBKey – Basic Programming Configuration for MS Windows Environments", 2006.

MILESTONE 1: ASSIGNMENT

Instructions: Complete the following 2 questions ANYTIME before the end of the lab session and submit to your teaching assistant.

Question1: Modify the line that changes PORT D, execute your program and fill out all values of Table 1. Refer to Section 10 of the datasheet (doc7593) when you do the assignment.

Table 1: Onboard Diode Decoding Table

Binary # to PORT D	Hexadecimal Equivalent	Decimal Equivalent	D5 Color	D2 Color
0b00011100				
	0x10			
		32		
0b00110001				
	0x40			
		25		
0b10101010				
	0x70			
		128		
0b01010101				
	0xA0			
0b11000000				

Question 2: How might the color codes be useful in debugging real time applications?

Lab Session:

Name 1:_____ Student ID: _____

Name 2:_____ Student ID: _____