EE 224 **February 8, 2018**
ARM Nucleo Board

Features

The ARM Nucleo board is shown in Figure 1.1. The board has the following features:

- STM32F446RET6 microcontroller with the following features:
  - LWFP64 package
  - Floating point unit with frequencies up to 180 MHz.
  - 512 kB of Flash memory plus 128 KB of SRAM
  - Dual mode Quad SPI interface
  - 1.7 V to 3.6 V application supply and I/Os
  - 4-to-26 MHz crystal oscillator Internal 16 MHz factory-trimmed RC (1% accuracy)
  - Internal 32 kHz RC with calibration
  - Three 12-bit, 2.4 MSPS ADC: 16 channels
  - Two 12-bit D/A converters
  - General-purpose DMA
  - 14 timers
  - SWD & JTAG interfaces
  - Board power supply: through USB bus or from an external 5V supply voltage
  - External application power supply: 3V and 5V
  - ST MEMS motion sensor, 3-axis digital output accelerometer
  - ST MEMS audio sensor
  - Audio DAC with integrated class D speaker driver
  - Eight LEDs (4 for user)
  - Two pushbuttons (user and reset)
  - USB micro-AB connector for connection to a mouse or other USB peripheral.
  - USB mini-B connector for programming directly from the Keil µVison 5 development system.
  - 50 GPIO ports with interrupt capability (most are 5 V tolerant)
  - Four I^2C interfaces
  - Four USARTs and two UARTs
  - Four SPIs
  - Two SAI (serial audio interface)
  - Two CAN (2.0B Active)
  - SDIO interface
  - Consumer electronics control (CEC) I/F
  - Advanced connectivity
  - 8- to 14-bit parallel camera interface up at 54 Mbytes/s

The STM32F446RE is an ARM Cortex M4 processor. The M4 is identical to the M3 but it has an additional DSP engine added on.

The ARM Cortex M3 and M4 processors are RISC machines with a 3 stage pipe (Fetch, Decode, and Execute) see Figure 1.2. The M3 and M4 processors execute the Thumb2 instruction set which consists of 50, 16-bit instructions and 6, 32-bit instructions.

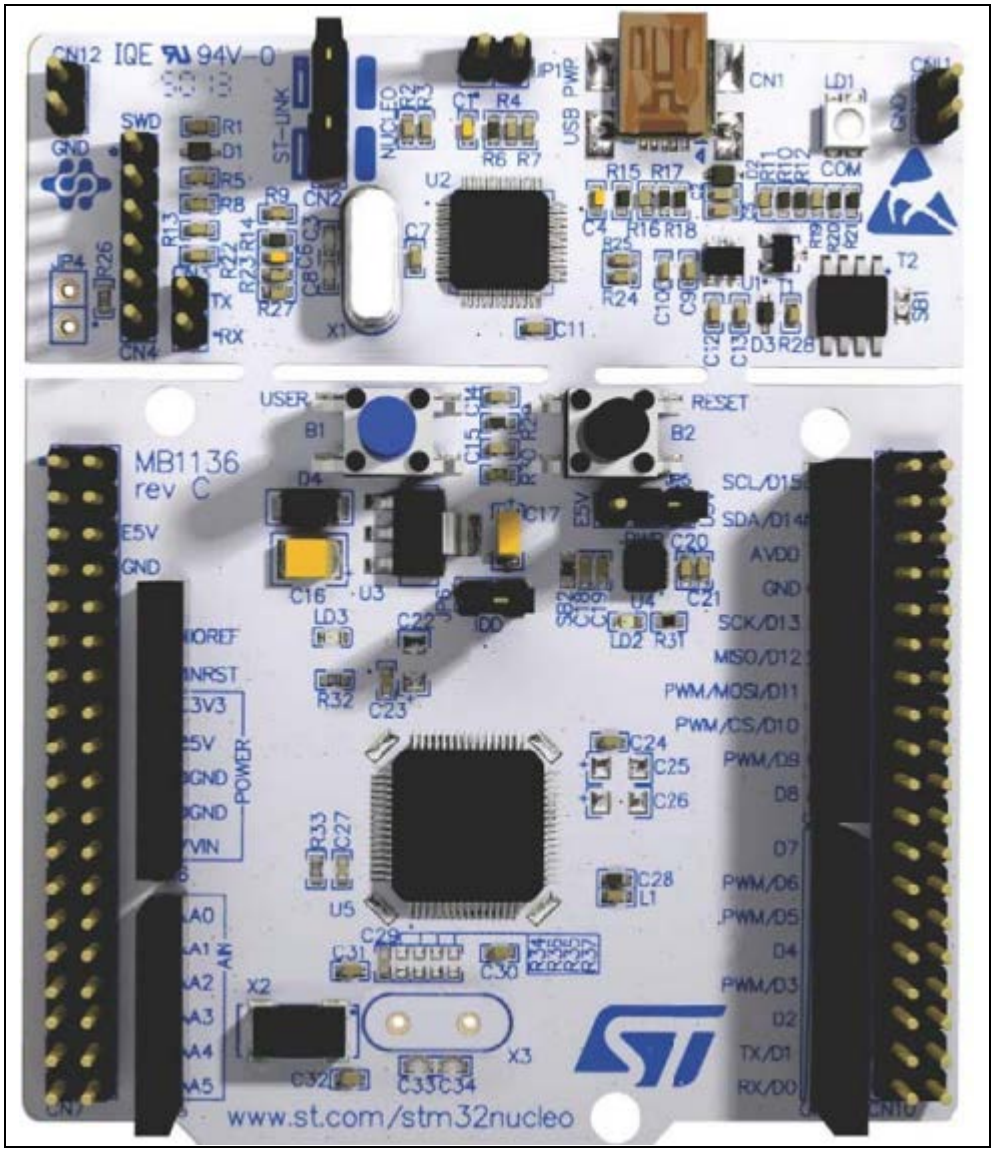


Figure 1.1
The ARM Nucleo board. The USB port at the top of the board (mini-B) is the programming port.[3]

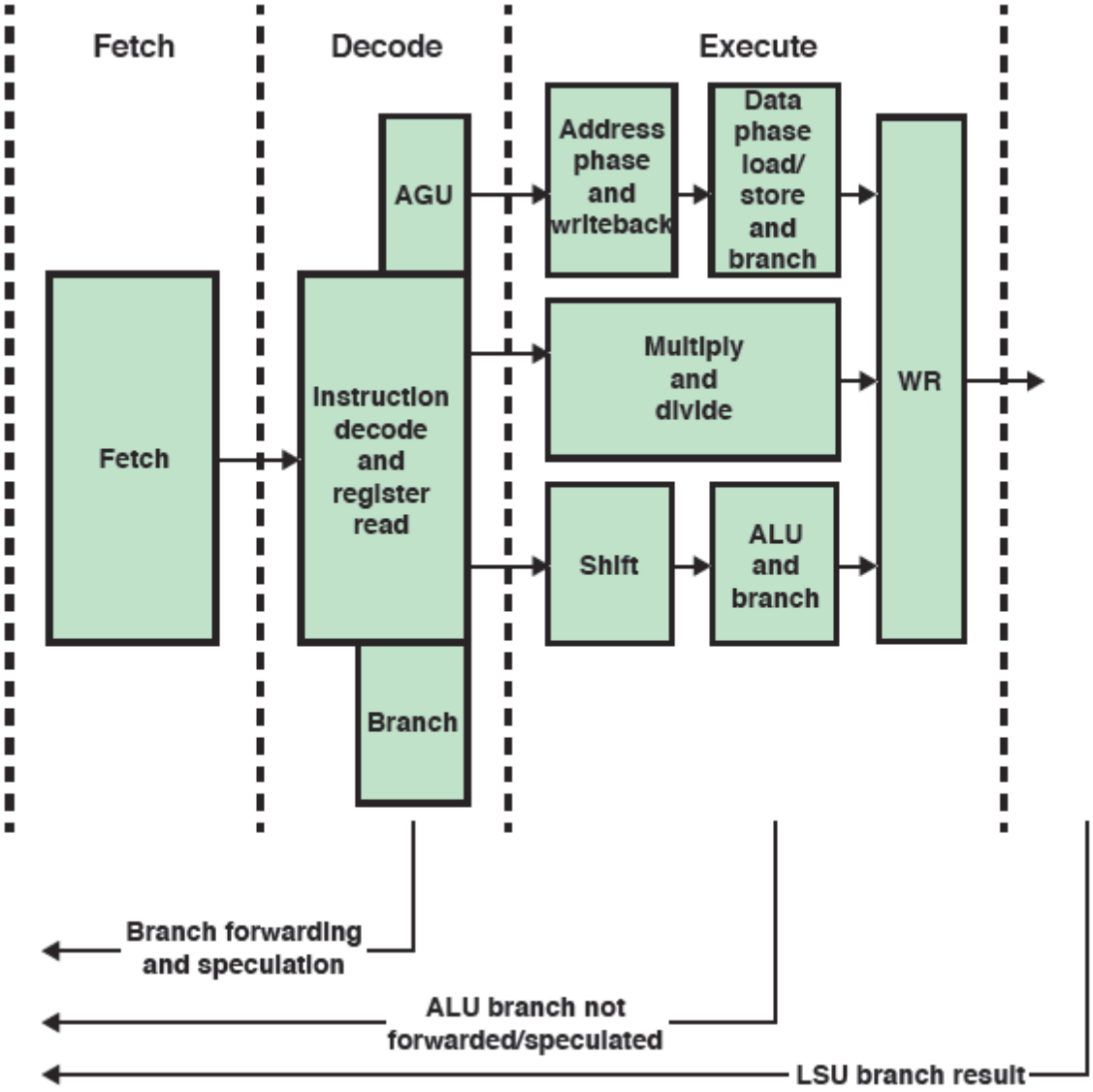


Figure 1.2
The ARM Cortex M3, M4 pipeline. [4]

Section 2 Keil Development System

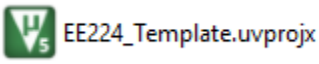
Keil µVision 5 Setup

The free version of the Keil Development system for the ARM board is available from http://www.keil.com/ Under *Software Downloads* click on *Product Downloads* → *MDK-ARM v5* and fill out the registration form. Keil will download the compiler and send you a license file via email.

After this is installed download the following file from the website:
https://csserver.evansville.edu/~blandfor/EE224/EE224Template.zip

Put EE224Template.zip into a folder where you will store your Nucleo Board software. Unzip the folder (Right click on it and select *Extract All*). After unzippoing the folder you will find two new folders: EE224Template and EE224TemplateEample

Open EE224TemplateExample and double click on the project file which is called EE224_Template.uvprojx.



This will open Keil and load the project. You will see something like that shown in Figure 2.1.

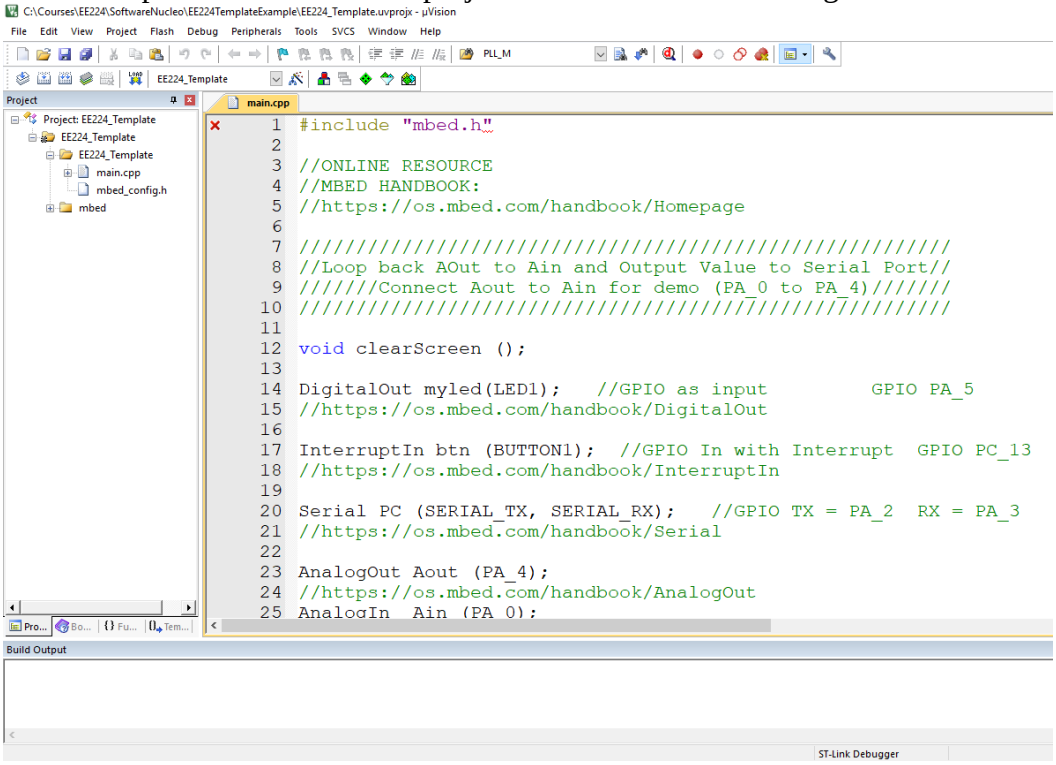


Figure 2.1
Keil µVision 5 with the EE224TemplateExample project loaded.

Your project may have a red x on the line for #include "mbed.h". Ignore this error.

If this is the first time that the Nucleo board has been used on the computer you are using, the system may open a *package loader*. If it does, allow it to load the packages it needs. (If you have trouble with this see your instructor or Jeff Cron.)

Click on *Project* → *Build Target*
At the bottom of your screen you will see a window that contains messages from the compiler. It should look something like that shown in Figure 2.2.

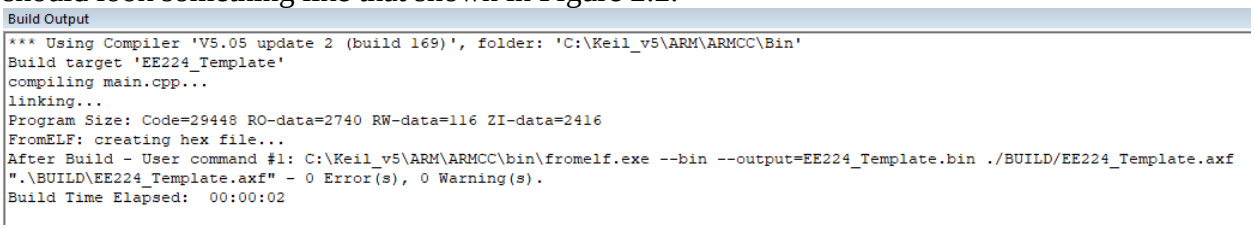


Figure 2.2
Build output window. This compile was successful – note that is says "0 Error(s), 0 Warning(s)"

If you have been leading a virtuous life you will get a message that says there are no errors and no warnings. If this is not the case see your instructor for help.

Assuming the compile was successful you can now connect your Nucleo board to your computer and load the compiled program into the flash memory. Connect the USB cable to your Nucleo board to the mini B port (top in Figure 1.1). Connect the other end of the USB cable to your computer. This will cause your computer to load the proper driver for the board and that may take a minute or two. You will need to be connected to the internet. In some cases this driver may need to be loaded "by hand". See Jeff Cron.

Assuming that all is well with the connection you can download the program to the flash memory on the board. To do this click on *Debug* → *Start/Stop Debug Session* or click on the red *Debug* button on the toolbar. 

You should see a blue progress bar for a few seconds on the bottom of your screen. This will be followed by the notice in Figure 2.3.

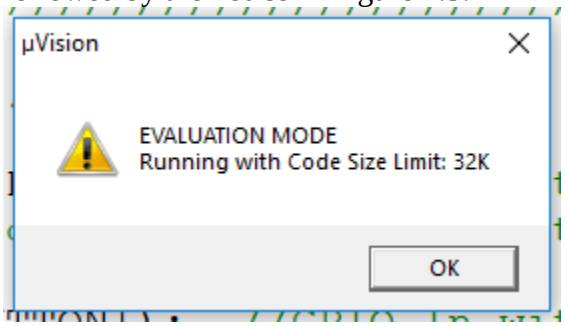


Figure 2.3
Evaluation Mode notice. This is not the professional edition.

Click on OK to get the debug screen shown in Figure 2.4.

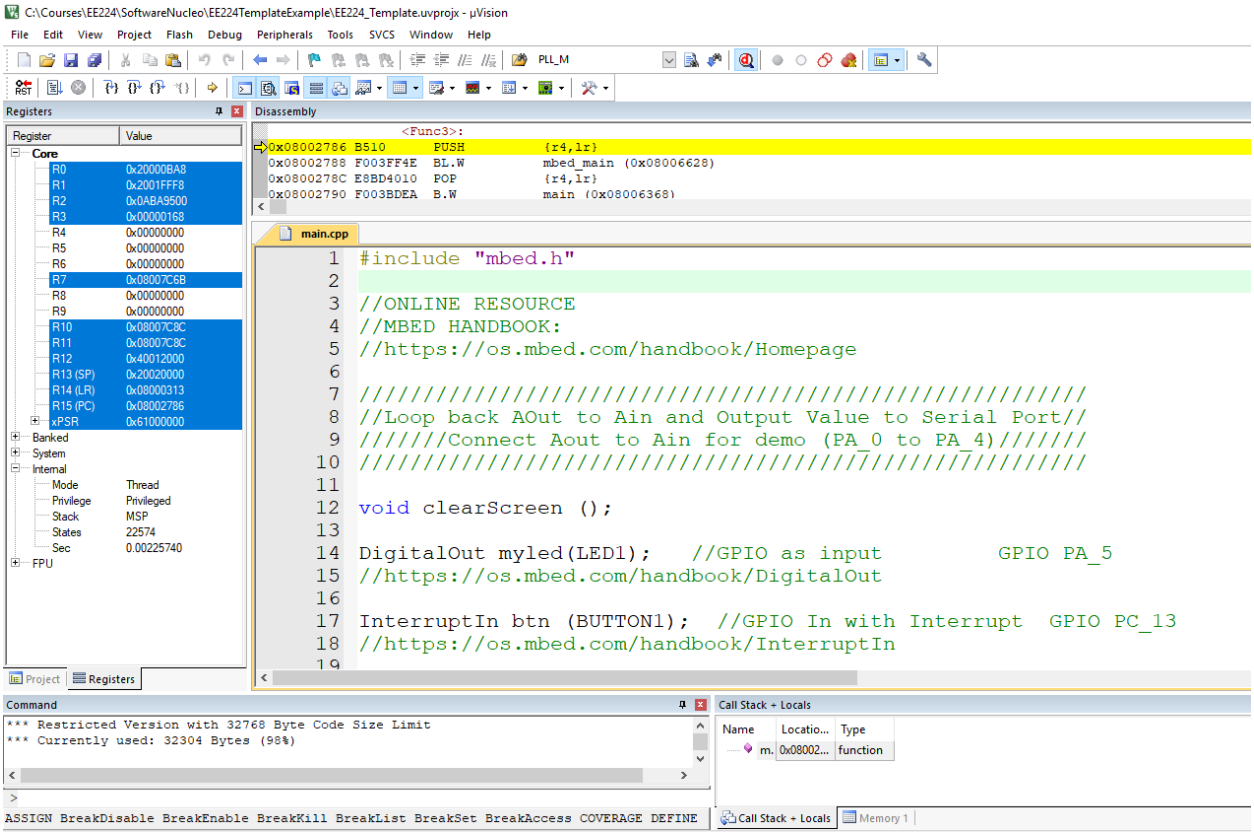


Figure 2.4
The debug screen.

In the debug screen click on *Debug* → *Run*, or Push the F5 function key or click on the run button on the toolbar 

The green LED on the Nucleo board should blink about once a second. You can stop debugging by clicking on *Debug* → *Start/Stop Debugging* or by clicking on the red debug button on the toolbar. 

The program will continue to run on the board. In fact, you can unplug your board and at some point later plug it in again and the program will begin running again.

Creating Your Own Project Assignments
To make it easier to set up and run your own projects you can use the following procedure:
1. Download this file from the website and unzip it:
After this is installed download the following file from the website:
https://csserver.evansville.edu/~blandfor/EE224/EE224Template.zip
2. In the unzipped folder select and copy all of the files in *EE224Template.*
3. Create a new folder for your assignment. Suppose it is called Asn01. Open this new folder and paste the copied files into the Asn01 folder.

4. In the Asn01 folder double click on *EE224_Template.uvprojx*. This will open Keil 5 to an empty main program like that shown in Figure 2.5.

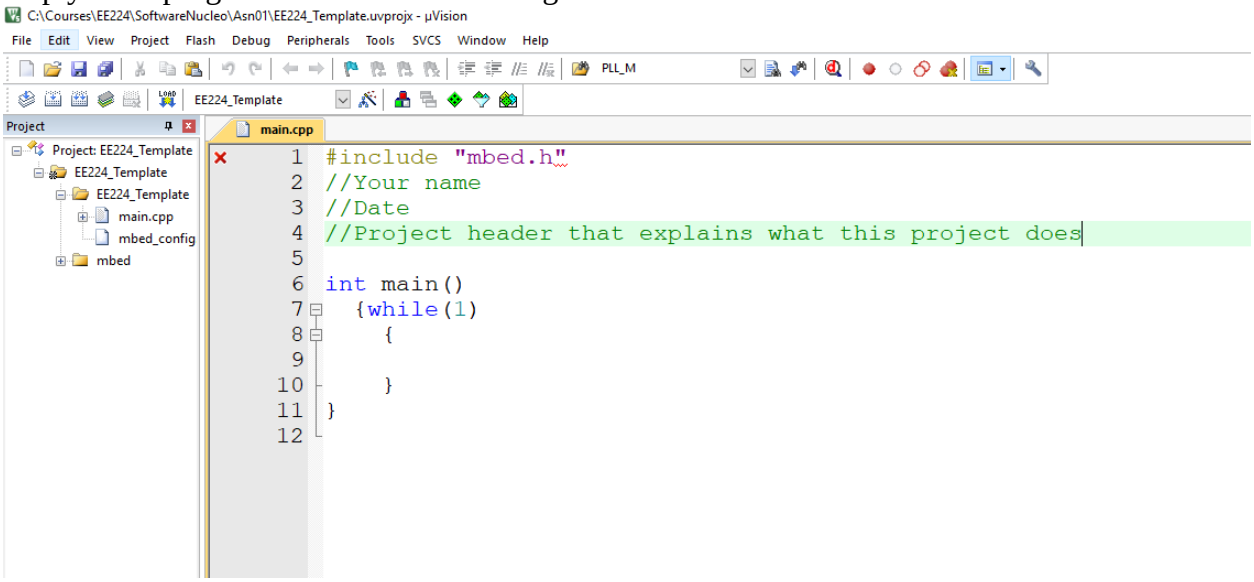


Figure 2.5
An empty main program in the EE224Template file in the Asn01 folder.

Enter your own code for the main program, build and load the code into your board.

If you have trouble loading your project into your board try the following:
A) Click on *Project → Options for EE224_Template* to open the screen shown in Figure 2.6

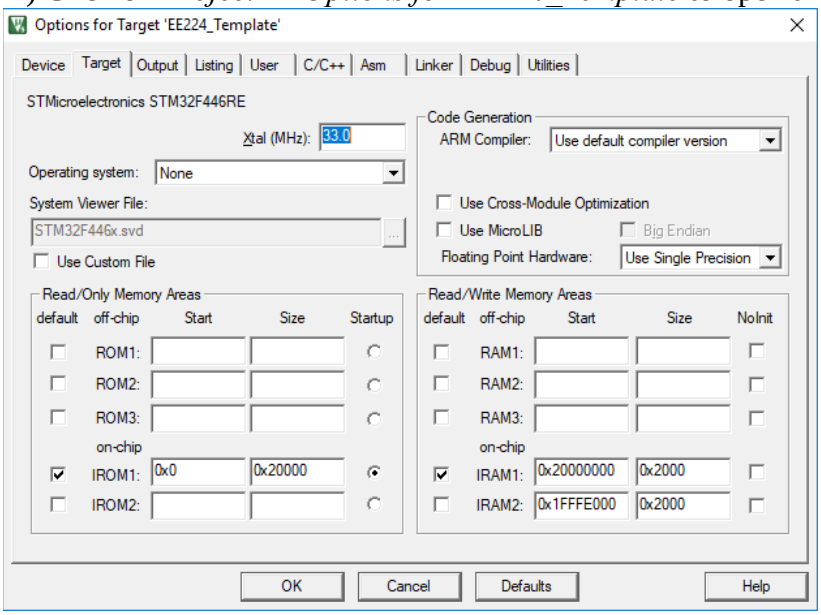


Figure 2.6
Options screen for the project. Select the Debug tab.

B) Select the *Debug tab* and click on *Settings* (top right) to get the screen shown in Figure 2.7

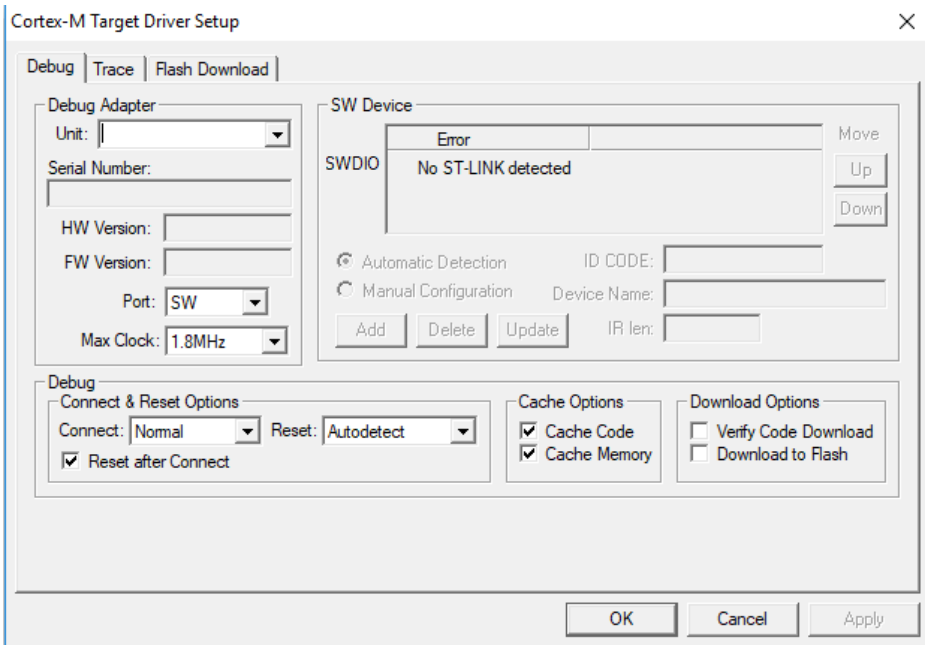


Figure 2.7
The Settings option under the Debug tab. Make sure the Port is set to SW.

The Port option on the Settings window must be set to SW. If it is not, change it from the pull-down menu.

Using the Simulator
The simulator is all software based so it needs to know in advance what memory you are using. This is done by creating an *initialization* file which typically has a *.ini* extension but is otherwise a simple text file. After you have compiled your program successfully, click on *file → new*. The µVision environment will open a new file for you. Enter a mapping function given by

```
MAP 0x40000000, 0x40028000 READ WRITE
```

Save the file with the name *Debug.ini*. See Figure 2.8.

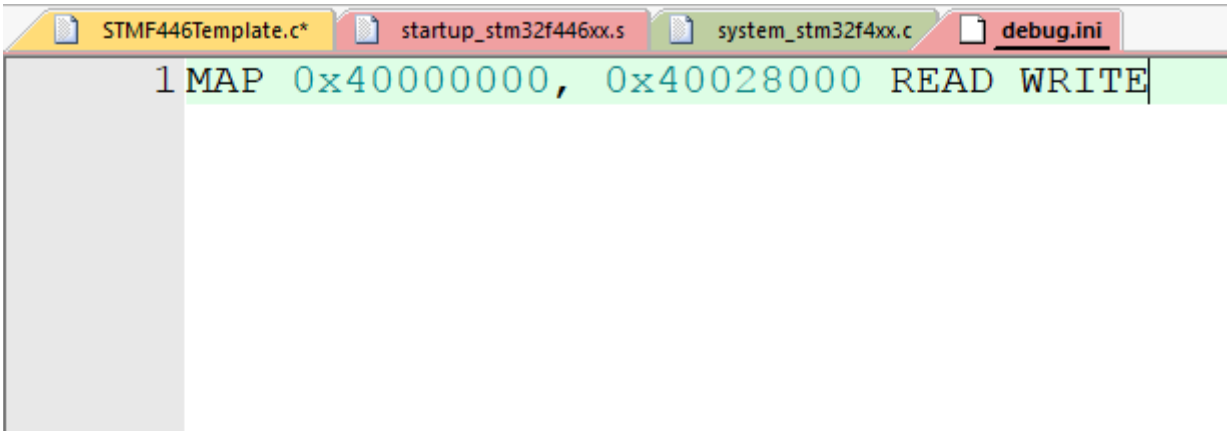


Figure 2.8
The Debug.ini file. This allows the simulator to read and write to simulated memory at 0x40000000 to 0x40028000 which covers most of the memory area where the processor maps its I/O ports.

After creating *Debug.ini* and saving it, right click on Target1 in the *project* window and select *Options for Target 1*. In the screen that appears select the *Debug* tab as shown in Figure 2.7.

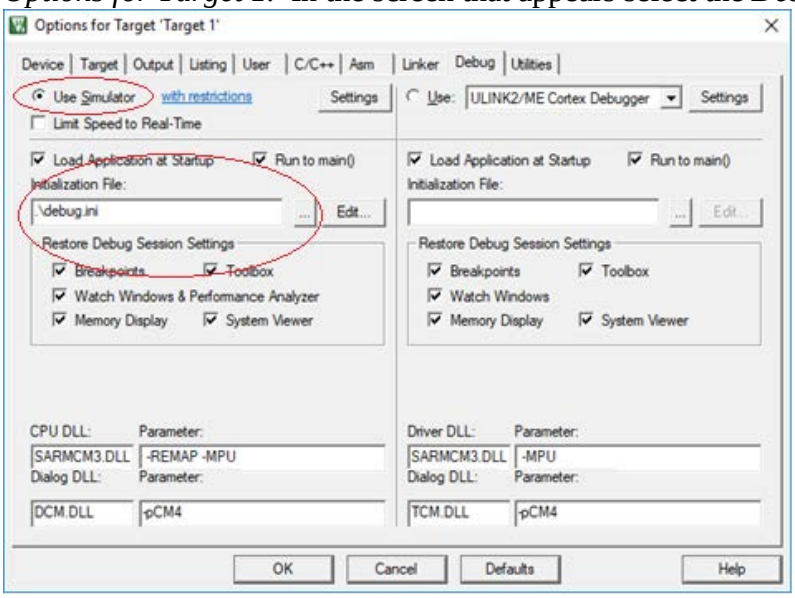


Figure 2.9
Right click on Target 1 in the Project window and select *Options for Target 1*. Select the *Debug* tab to get this screen. Click on the "Use Simulator" radio button and browse to the debug.ini file for the Initialization File.

In Figure 2.9 select "Use Simulator" and on the left side panel click on the box with three dots (…) in the *Initialization file*. This will allow you to browse out to locate the *Debug.ini* file you created earlier. When you select Debug.ini the Initialization file window will look like that shown in Figure 2.9.

Click on OK and we are ready to do our simulation. Click on *Debug* → *Start/Stop Debug Session* or click on the red button at the top of the screen:

When you start the debugger you will get a message saying that you are in the "Evaluation mode" with a code size limit of 32K. If you are using the professional version of µVision 5 this message will not appear. For this class the student version is adequate for all assignments. Figure 2.10 shows the debugger screen for this project.

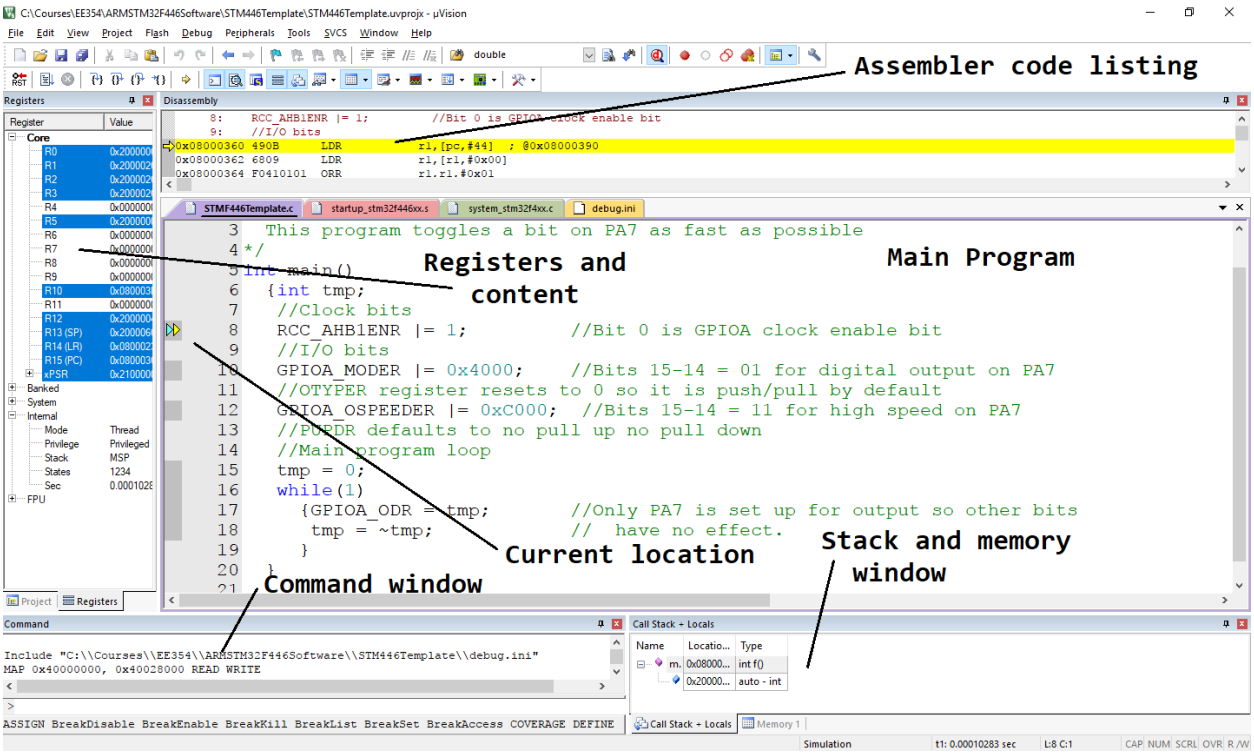


Figure 2.10
The initial debugger screen. If your debugger screen does not have all of these components you can use the view menu to get them.

The program that we are debugging uses port A for output. This is called GPIOA (General Purpose I/O port A). Click on *Peripherals* → *System Viewer* → *GPIOA* to show the registers in GPIO port A. See Figure 2.11

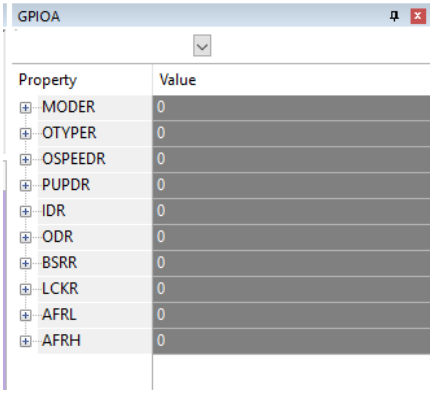


Figure 2.11
The registers in GPIOA as seen in the debugger. Click on the + sign to expand the register to see the bits.

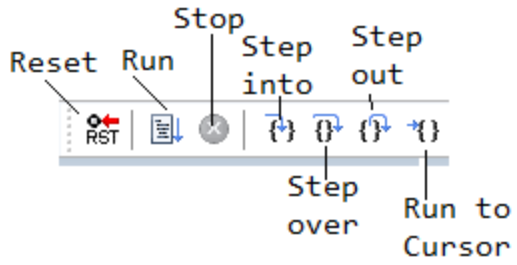


Figure 2.12
Menu buttons for controlling the debugger. These are located on the Debug Toolbar at the top of the screen. If these are not present on your screen you can find them on the view menu.

To run the program and do the simulation click on the *Run* button on the Debug Toolbar. (See Figure 2.12). For this program you will see the *Output Data Register* (ODR) on GPIOA change from FFFFFFFF to 00000000 since the program writes all ones and all zeros to the port continuously. Notice also that at the bottom of the Register window the number of simulated seconds is also changing. (This time will not be accurately simulated unless you have correctly set the crystal speed before you entered the debugger. To do this right click on Target 1 in the *Project* windows and select *Options for Target 1*. Choose the *Target* tab and set the crystal speed to match that on your board.)

You can stop the program at any point by pushing the stop button on the Debug Toolbar. You can also single step through your program using the *Step into* button.

You can stop debugging and go back to the main program screen by clicking on *Debug* → *Start/Stop Debugging*.

If while debugging you find an error it is tempting to change the code in the debugger to fix the problem. Such changes will not be effective. If you make any changes to the code they will not go into effect until you have again recompiled the program.