EEE3074W
Embedded Systems

LAB 1: *DISCOVERING* THE STM32 DISCOVERY

STUDENT NUMBER 1:

………

STUDENT NUMBER 2:

………

This lab will serve as an introduction to the STM32F4 Discovery board. This is the prescribed board for the course. It will also expose them to the development environment Atollic TrueSTUDIO for ARM Lite (also based on the Eclipse IDE).

INTRODUCTION

Each pair of students will need to demonstrate the ability to write a simple C embedded program for a STM32 Discovery Board, compile the code and burn it to flash, stop/start execution of the program on the target platform, and ability to display and change data at certain memory addresses

MOTIVATION

The purpose of this lab is to introduce you to the STM32 Discovery and the Atollic TrueSTUDIO IDE. You will be using these extensively through the labs this year, as well as during the project.

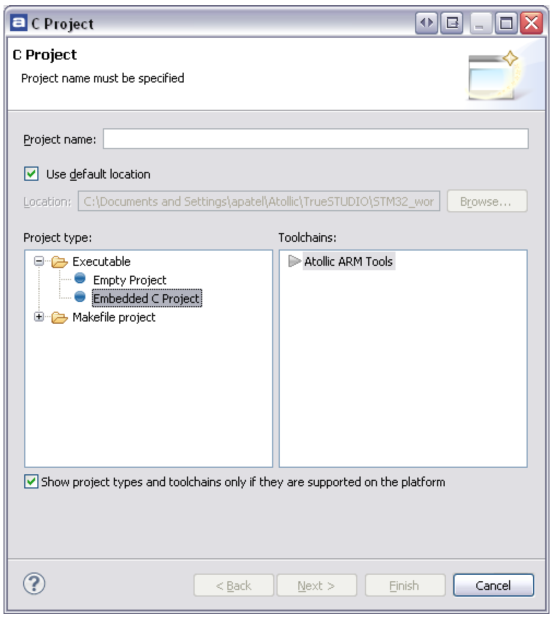
STARTING OUT

Start by getting the Windows XP virtual machine running.

Next, start Atollic by clicking the desktop icon. Atollic TrueSTUDIO for STMicroelectronics STM32 Lite 2.2.0 Select the default workspace.

We will now create a new project.

Once Atollic is started, click File->New->C Project. The following screen should appear:


C Project

C Project

Project name must be specified

Project name:

☑ Use default location

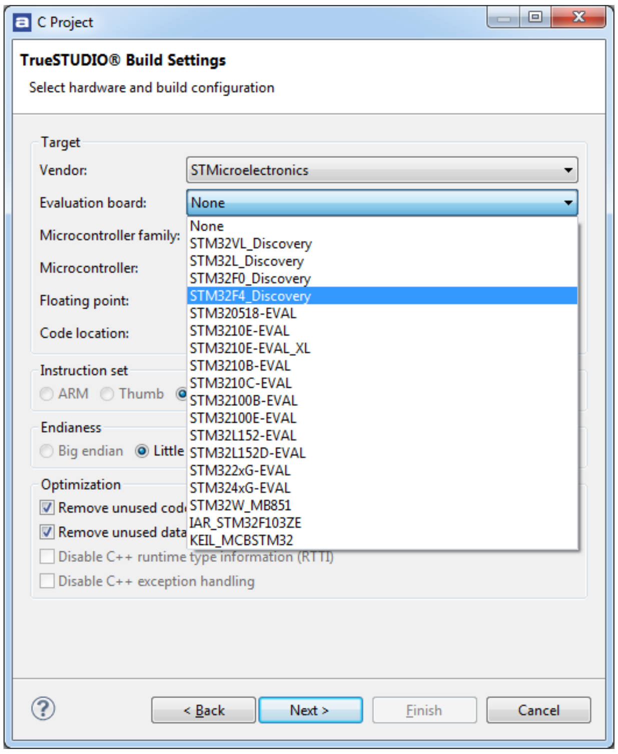
Location: C:\Documents and Settings\apatel\Atollic\TrueSTUDIO\STM32_wor

Project type: Executable, Empty Project, Embedded C Project, Makefile project

Toolchains: Atollic ARM Tools

☑ Show project types and toolchains only if they are supported on the platform

Call your project "Lab1" and make sure "Embedded C Project" is selected. Then click next. The following screen should appear:


C Project

TrueSTUDIO® Build Settings

Select hardware and build configuration

Target

Vendor: STMicroelectronics

Evaluation board: None

Microcontroller family:

Microcontroller:

Floating point:

Code location:

None
STM32VL_Discovery
STM32L_Discovery
STM32F0_Discovery
STM32F4_Discovery
STM320518-EVAL
STM3210E-EVAL
STM3210E-EVAL_XL
STM3210B-EVAL
STM3210C-EVAL
STM32100B-EVAL
STM32100E-EVAL
STM32L152-EVAL
STM32L152D-EVAL
STM322xG-EVAL
STM324xG-EVAL
STM32W_MB851
IAR_STM32F103ZE
KEIL_MCBSTM32

Instruction set: ARM, Thumb

Endianess: Big endian, Little

Optimization

☑ Remove unused cod

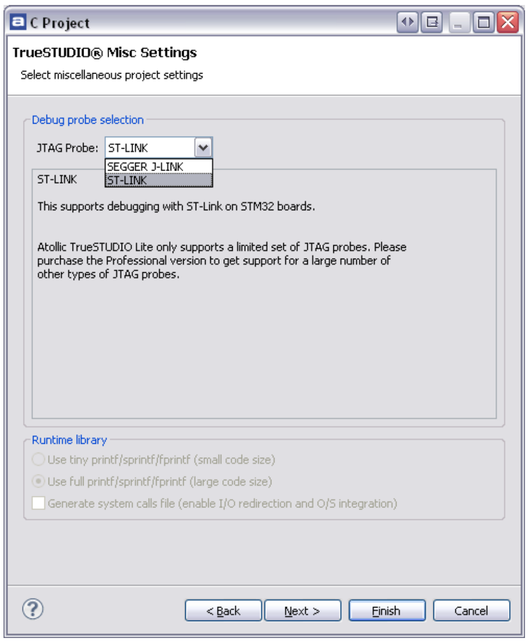
☑ Remove unused data

☐ Disable C++ runtime type information (RTTI)

☐ Disable C++ exception handling

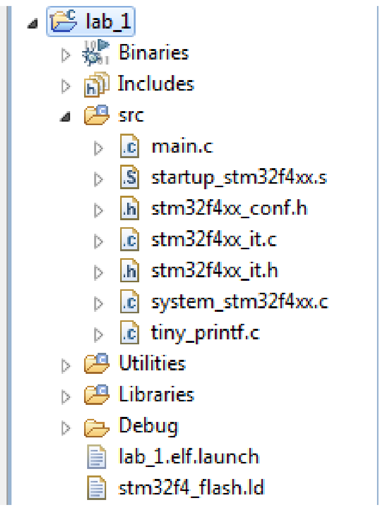
Click evaluation board, and select 'STM32F4_Discovery. Click next.

In the next screen, under JTAG probe, select 'St-Link' as seen below:


C Project
TrueSTUDIO® Misc Settings
Select miscellaneous project settings
Debug probe selection
JTAG Probe: ST-LINK
SEGGER J-LINK
ST-LINK
ST-LINK
This supports debugging with ST-Link on STM32 boards.
Atollic TrueSTUDIO Lite only supports a limited set of JTAG probes. Please purchase the Professional version to get support for a large number of other types of JTAG probes.
Runtime library
Use tiny printf/sprintf/fprintf (small code size)
Use full printf/sprintf/fprintf (large code size)
Generate system calls file (enable I/O redirection and O/S integration)
< Back
Next >
Finish
Cancel

Finally, click 'Next' and then 'Finish'. Your project has now been created.

In the Project Explorer, under 'lab1', navigate to the 'src' folder and open the 'main.c' file.

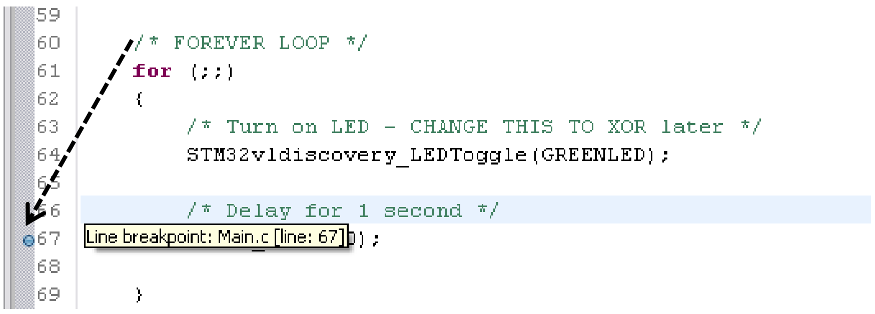


This file should contain some generic demo code, which is a bit messy. Open the main.c file provided on Vula, and replace its contents with those. The file should now be a bit leaner.

DEBUGGING AND BUILDING – ECSA ELO B2

We will now demonstrate the power of Atollic (and actually Eclipse) by doing some debugging. Inside the infinite loop, add a breakpoint next to the delay function call. A breakpoint will halt the program while it is executed and allows one to debug the software.

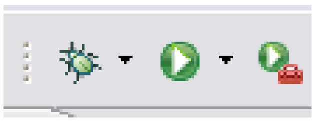
To add a breakpoint, double click next to the number line. This should add a small blue circle to the left of the number line:



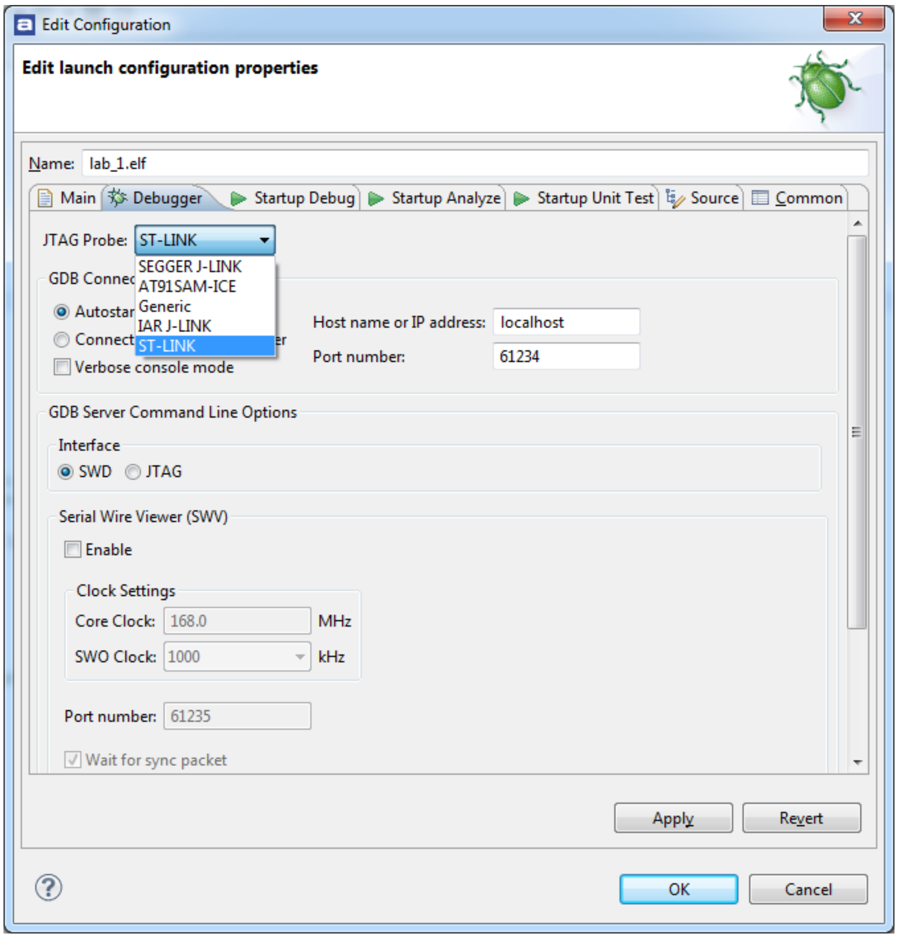
```
    /* FOREVER LOOP */
    for (;;)
    {
        /* Turn on LED - CHANGE THIS TO XOR later */
        STM32vldiscovery_LEDToggle(GREENLED);

        /* Delay for 1 second */
Line breakpoint: Main.c [line: 67]0);

    }
```

Now, build the project by right-clicking on the project in the Project Explorer and selecting 'Build'.

Let's debug the application. Click the green bug icon in the top of Atollic:



The debug configuration window should open up. Click the Debugger tab, and make sure that 'ST-Link' is selected under JTAG Probe, then click debug.


Edit Configuration

Edit launch configuration properties

Name: lab_1.elf

Main | Debugger | Startup Debug | Startup Analyze | Startup Unit Test | Source | Common

JTAG Probe: ST-LINK

SEGGER J-LINK
AT91SAM-ICE
Generic
IAR J-LINK
ST-LINK

GDB Connec

(●) Autostar
( ) Connect
[ ] Verbose console mode

Host name or IP address: localhost
Port number: 61234

GDB Server Command Line Options

Interface
(●) SWD ( ) JTAG

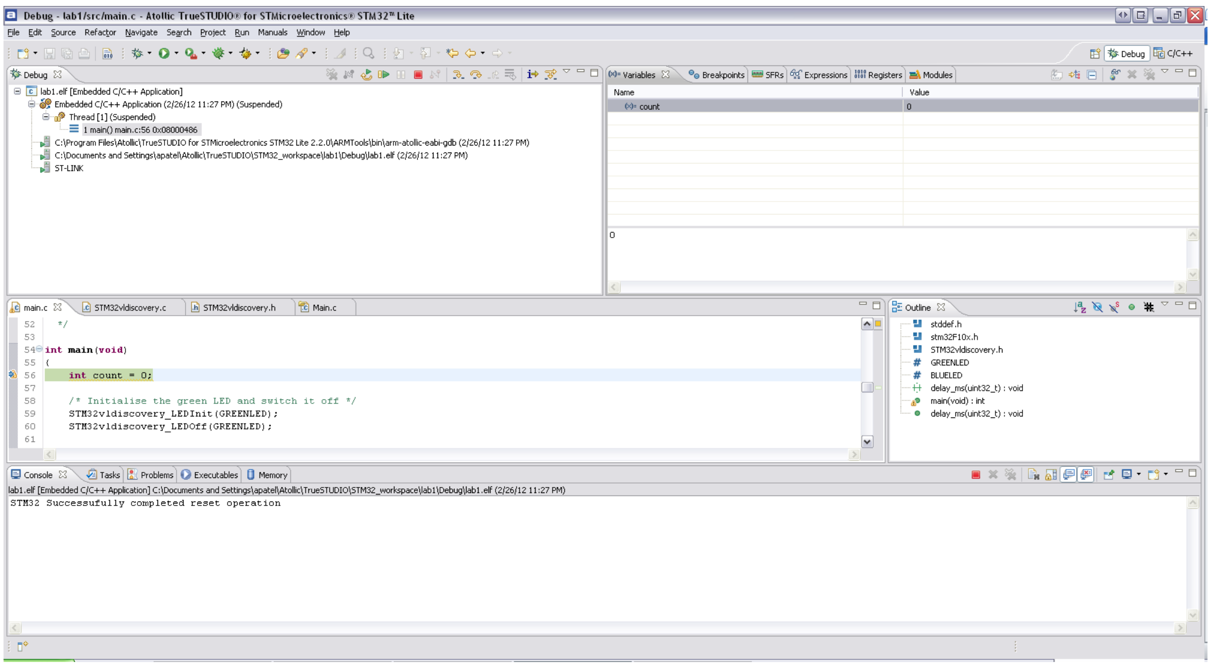
Serial Wire Viewer (SWV)
[ ] Enable

Clock Settings
Core Clock: 168.0 MHz
SWO Clock: 1000 kHz

Port number: 61235

[✓] Wait for sync packet

This should open the debug perspective. You should the following screen:

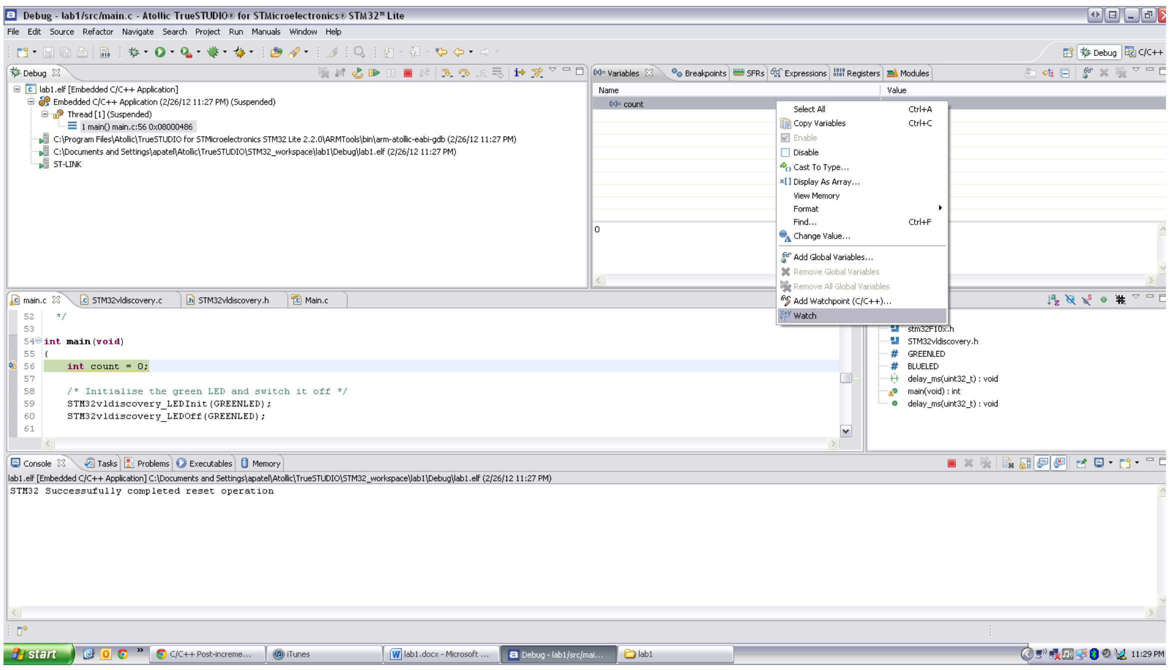


```
lab1.elf [Embedded C/C++ Application]
  Embedded C/C++ Application (2/26/12 11:27 PM) (Suspended)
    Thread [1] (Suspended)
      1 main() main.c:56 0x08000486
  C:\Program Files\Atollic\TrueSTUDIO for STMicroelectronics STM32 Lite 2.2.0\ARMTools\bin\arm-atollic-eabi-gdb (2/26/12 11:27 PM)
  C:\Documents and Settings\apatel\Atollic\TrueSTUDIO\STM32_workspace\lab1\Debug\lab1.elf (2/26/12 11:27 PM)
  ST-LINK
```

```c
 */

int main(void)
{
    int count = 0;

    /* Initialise the green LED and switch it off */
    STM32vldiscovery_LEDInit(GREENLED);
    STM32vldiscovery_LEDOff(GREENLED);
```

```
lab1.elf [Embedded C/C++ Application] C:\Documents and Settings\apatel\Atollic\TrueSTUDIO\STM32_workspace\lab1\Debug\lab1.elf (2/26/12 11:27 PM)
STM32 Successfully completed reset operation
```

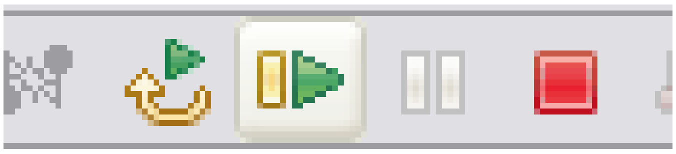
To watch the value of 'count' being incremented, we need to 'watch' the variable. We do this by right clicking on count under the 'variables' tab and clicking 'watch':



```c
 */

int main(void)
{
    int count = 0;

    /* Initialise the green LED and switch it off */
    STM32vldiscovery_LEDInit(GREENLED);
    STM32vldiscovery_LEDOff(GREENLED);
```

```
lab1.elf [Embedded C/C++ Application] C:\Documents and Settings\apatel\Atollic\TrueSTUDIO\STM32_workspace\lab1\Debug\lab1.elf (2/26/12 11:27 PM)
STM32 Successfully completed reset operation
```

This will ensure that we will be able to see the value of 'count' change during program execution. Let us now run the program. Click the 'resume' button.



The program will run, but will stop at the breakpoint. In the variables window, you should see the value of 'count' has changed to 5. Clicking the resume button, will cause the program to run again, until reaching the break point. Click it several times, until the value of 'count' reaches 25 and demonstrate this to the tutor.

Demonstrated (tick):	Signature of tutor/TA:

EMBEDDED SOFTWARE

TASK 1

It's now time to start coding! First thing to do, is close the debugging perspective by clicking the stop button.

Next, remove the break point by double clicking on the blue circle. Click debug again, the code will be compiled and the debug perspective will be open. Click the 'reset' (Black) button on the Discovery board. The green LED should be flashing at ~1HZ.

Modify the code to make the blue led flash oppositely. In other words, change the code to enable the blue led to turn on when the green led is off and vice versa. The frequency of flashing should also be 1Hz.

Demonstrated (tick):	Signature of tutor/TA:

TASK 2

Now that you've demonstrated this, go back to the main.c. Click on the 'STM32vldiscovery_LEDInit' function and press F3. This should jump you to a few other functions which might be useful for the following task.

Jump back to your Main.c by pressing alt+left cursor. Now, modify the task 1 code to enable the flashing rate to increase by 10Hz on each button press by the user. The rate should wrap at 50Hz.

Things to consider:

- Explore ‘STM32vldiscovery.h’, the library functions will make your life easier
- How you are going to handle switch debouncing?

Demonstrated (tick):	Signature of tutor/TA: