EET3350: Keil uVision Demo Lab

PRE-LAB ASSIGNMENT TO BE SUBMITTED IN BLACKBOARD PRIOR TO THE LAB:
If your lab is Monday submit prior to lab 2.20PM Monday. If your lab is Wednesday submit prior lab start on Wednesday.
Read these lab instructions, the Ebook chapter 4, 5 and the C power point.
1. What is the data type and range of uint32_t? [2.5pt]
2. What do you expect this C code to do? While (1) { Body( ) } [2.5pt] ;

OBJECTIVES:

1) To become familiar with the Keil uVision IDE, its functionality and GUI
2) To run an example project/file in Keil u vision IDE
3) To create a new project from an existing project
4) To debug a project
5) To use the simulator
6) To use the disassembly window to better understand C program execution in assembly code.
7) Learn basic about GPIO input and output configuration

PROCEDURE:

Open Keil u Vision from the start menu. Handouts are posted on BlackBoard.

I. Keil u Vision Demo Functionality

µVision is a windows-based software development platform that combines a robust and modern editor with a project manager and make facility tool. It integrates all the tools needed to develop embedded applications including a C/C++ compiler, macro assembler, linker/locator, and a HEX file generator. µVision helps expedite the development process of embedded applications.

The **µVision IDE and Debugger** is the central part of the Keil development toolchain and has numerous features that help the programmer to develop embedded applications quickly and successfully. The Keil tools are easy to use, and are guaranteed to help you achieve your design goals in a timely manner.

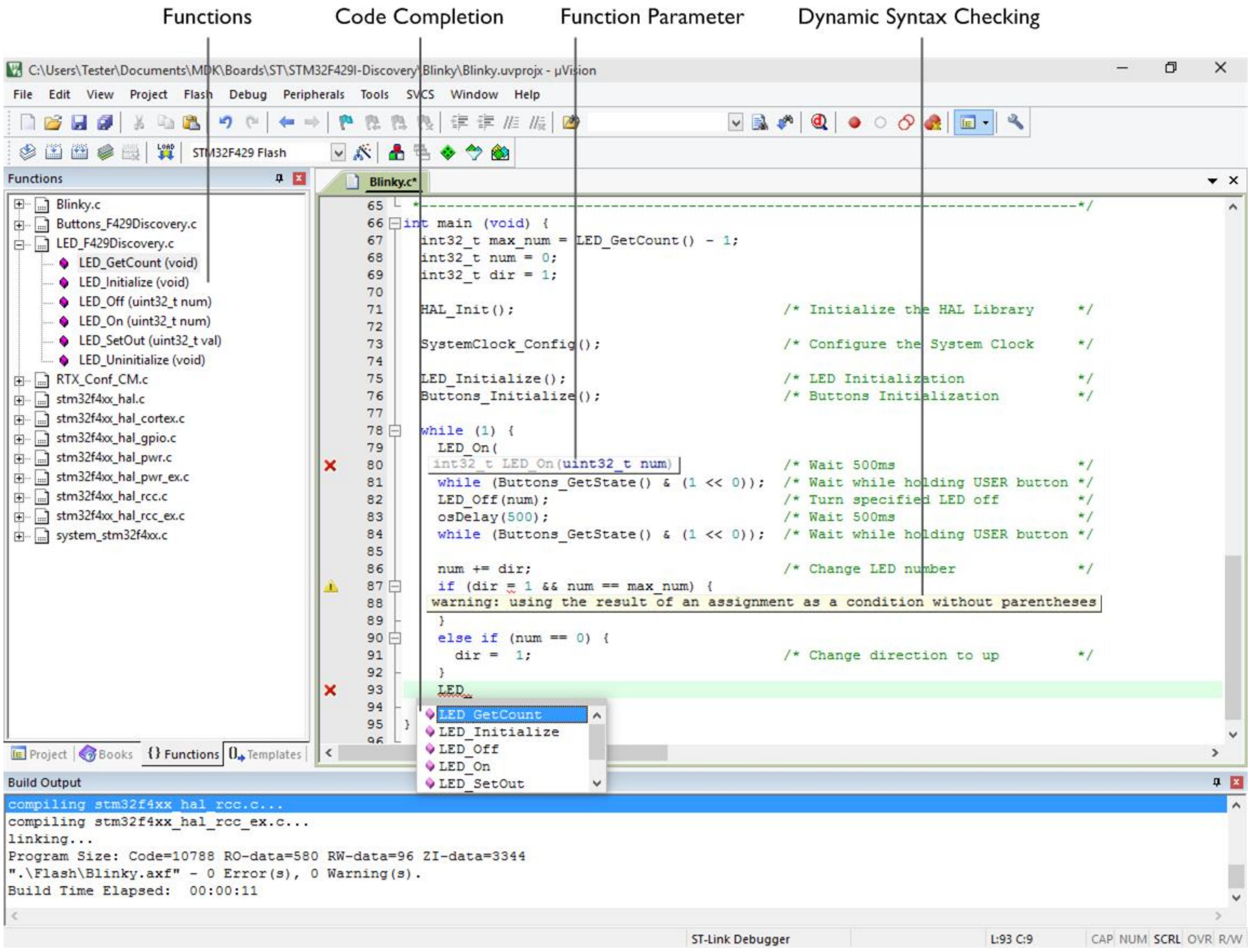
µVision offers a **Build Mode** for creating applications and a **Debug Mode** for debugging applications. Applications can be debugged with the integrated µVision **Simulator** or directly on hardware, for example ULINK Debug and Trace Adapters. Developers can also use other AGDI adapters or external third-party tools to analyze applications.

The µVision GUI provides **menus** for selecting commands and **toolbars** with command buttons. The Status Bar, at the bottom of the window, displays information and messages about the current µVision command. Windows can be relocated and docked to another physical screen. The window layout is saved for each project automatically and restored the next time the project is used. You can restore the default layout using the menu **Window – Reset View to Defaults**.

µVision has two operating modes, the Build Mode for creating applications and the Debug Mode for analyzing applications, which offers additional Windows and Dialogs.

µVision Editor

The integrated µVision Editor includes all standard features of a modern source code editor and is also available during debugging. Color syntax highlighting, text indentation, and source outlining are optimized for C/C++.



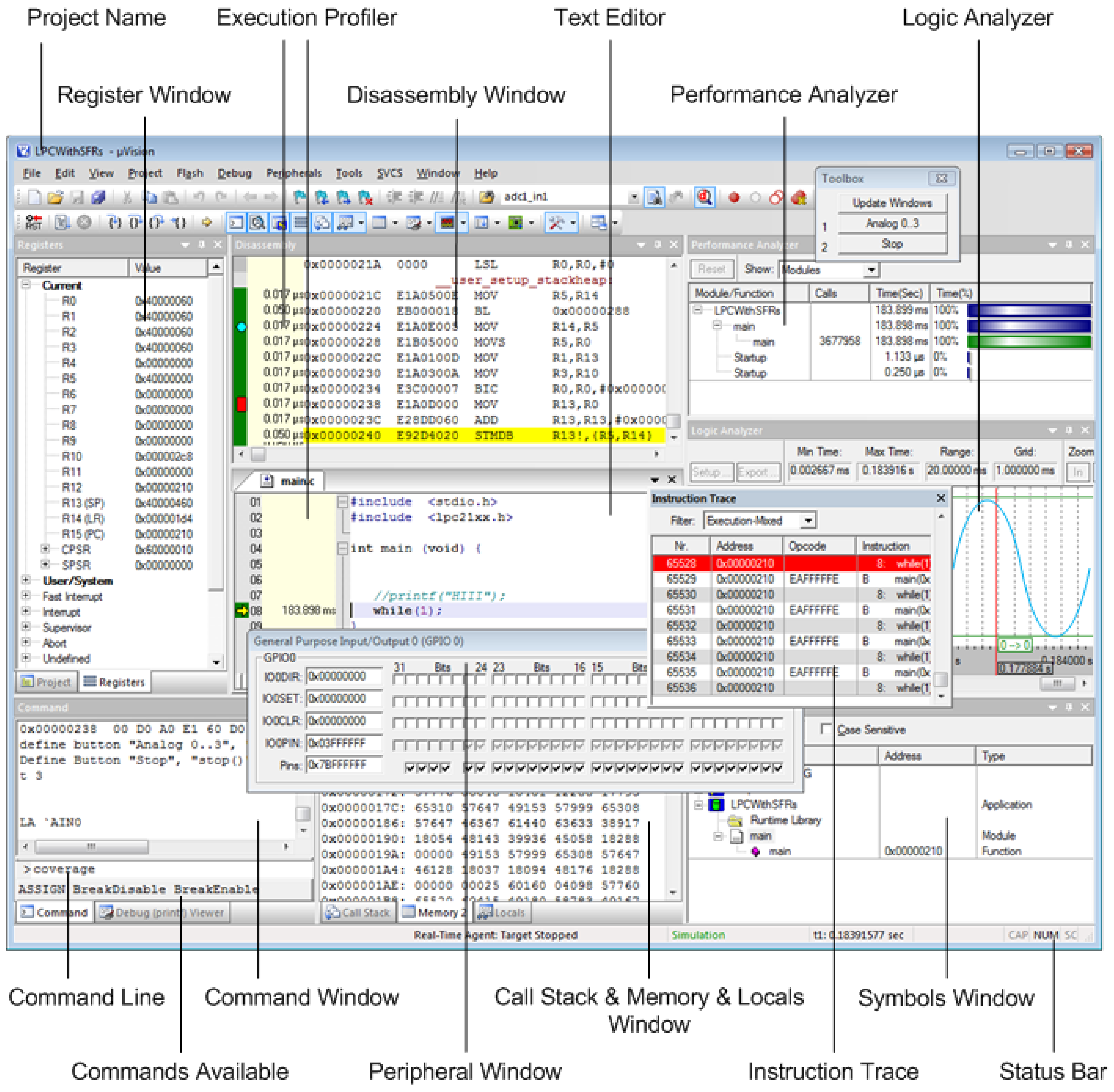
```
 *---------------------------------------------------------------------------*/
int main (void) {
  int32_t max_num = LED_GetCount () - 1;
  int32_t num = 0;
  int32_t dir = 1;

  HAL_Init();                                 /* Initialize the HAL Library     */

  SystemClock_Config();                       /* Configure the System Clock     */

  LED_Initialize();                           /* LED Initialization             */
  Buttons_Initialize();                       /* Buttons Initialization         */

  while (1) {
    LED_On(
   int32_t LED_On(uint32_t num)               /* Wait 500ms                     */
    while (Buttons_GetState() & (1 << 0));    /* Wait while holding USER button */
    LED_Off(num);                             /* Turn specified LED off         */
    osDelay(500);                             /* Wait 500ms                     */
    while (Buttons_GetState() & (1 << 0));    /* Wait while holding USER button */

    num += dir;                               /* Change LED number              */
    if (dir = 1 && num == max_num) {
   warning: using the result of an assignment as a condition without parentheses
    }
    else if (num == 0) {
      dir =  1;                               /* Change direction to up         */
    }
    LED_
  }
```

```
compiling stm32f4xx_hal_rcc.c...
compiling stm32f4xx_hal_rcc_ex.c...
linking...
Program Size: Code=10788 RO-data=580 RW-data=96 ZI-data=3344
".\Flash\Blinky.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:11
```

- The **Functions** window gives fast access to the functions in each C/C++ source code module.
- The **Code Completion** list and **Function Parameter** information helps you to keep track of symbols, functions, and parameters.
- **Dynamic Syntax Checking** validates the program syntax while you are typing and provides real-time alerts to potential code violations before compilation.

µVision Debug Mode

The picture shows some of the key µVision windows in **Debug Mode**.



```
0x0000021A  0000      LSL     R0,R0,#0
                      __user_setup_stackheap:
0x0000021C  E1A0500E  MOV     R5,R14
0x00000220  EB000018  BL      0x00000288
0x00000224  E1A0E005  MOV     R14,R5
0x00000228  E1B05000  MOVS    R5,R0
0x0000022C  E1A0100D  MOV     R1,R13
0x00000230  E1A0300A  MOV     R3,R10
0x00000234  E3C00007  BIC     R0,R0,#0x00000007
0x00000238  E1A0D000  MOV     R13,R0
0x0000023C  E28DD060  ADD     R13,R13,#0x0000
0x00000240  E92D4020  STMDB   R13!,{R5,R14}
```

```
#include  <stdio.h>
#include  <lpc21xx.h>

int main (void) {

    //printf("HIII");
    while(1);
```

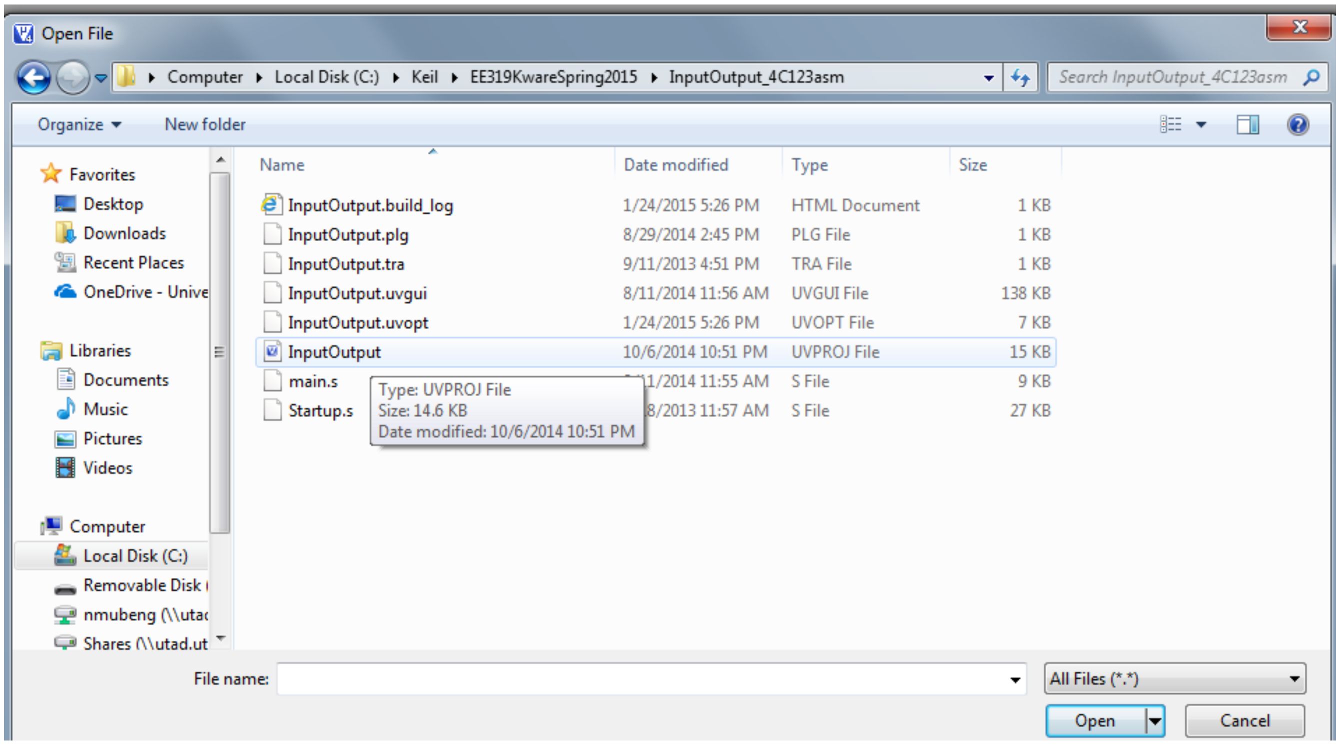
```
0x00000238  00 D0 A0 E1 60 D0
define button "Analog 0..3",
Define Button "Stop", "stop()
t 3

LA `AIN0
```

```
>coverage
ASSIGN BreakDisable BreakEnable
```

Real-Time Agent: Target Stopped   Simulation   t1: 0.18391577 sec

<u>II. How to run an example code in Keil u Vision simulator</u>

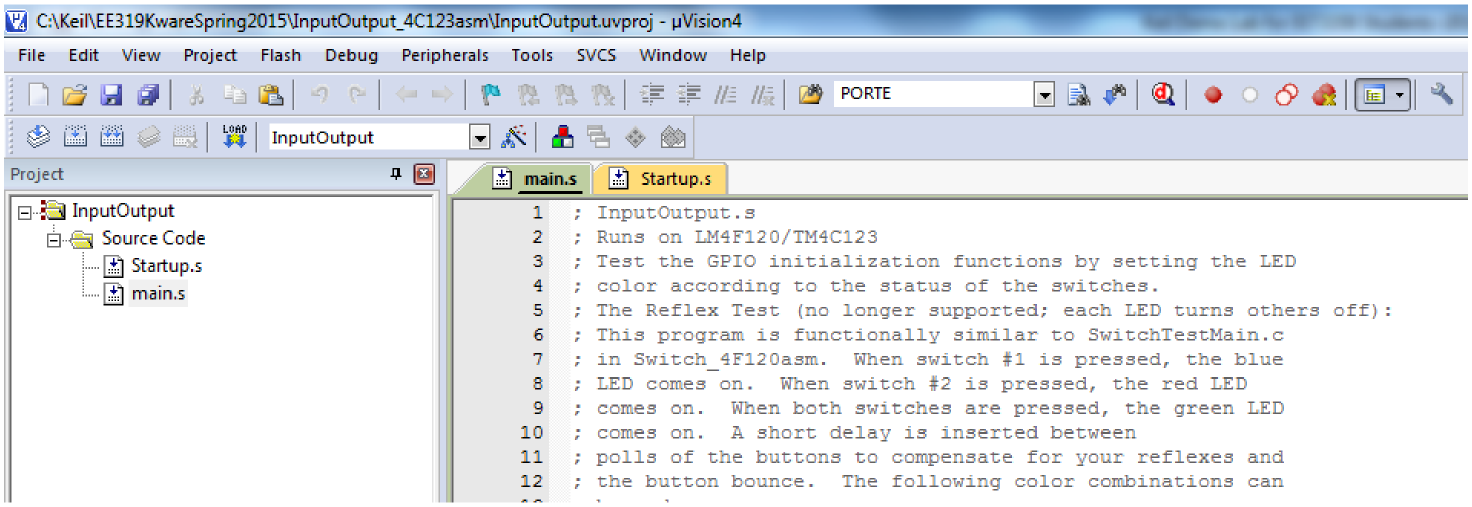
Example code are in c:/Keil directory. To open the example code, from the main Build Mode window, File→open→EE319KwareSpring2015→ **InputOutput_4C123asm→InputOutput.uvprj**
Show the windows, build, simulate, Watch window, single step



Name	Date modified	Type	Size
InputOutput.build_log	1/24/2015 5:26 PM	HTML Document	1 KB
InputOutput.plg	8/29/2014 2:45 PM	PLG File	1 KB
InputOutput.tra	9/11/2013 4:51 PM	TRA File	1 KB
InputOutput.uvgui	8/11/2014 11:56 AM	UVGUI File	138 KB
InputOutput.uvopt	1/24/2015 5:26 PM	UVOPT File	7 KB
InputOutput	10/6/2014 10:51 PM	UVPROJ File	15 KB
main.s	…1/2014 11:55 AM	S File	9 KB
Startup.s	…8/2013 11:57 AM	S File	27 KB

Type: UVPROJ File
Size: 14.6 KB
Date modified: 10/6/2014 10:51 PM

The project is opened in the Build Mode window as shown below.


C:\Keil\EE319KwareSpring2015\InputOutput_4C123asm\InputOutput.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PORTE
InputOutput
Project
InputOutput
Source Code
Startup.s
main.s
main.s
Startup.s

```
; InputOutput.s
; Runs on LM4F120/TM4C123
; Test the GPIO initialization functions by setting the LED
; color according to the status of the switches.
; The Reflex Test (no longer supported; each LED turns others off):
; This program is functionally similar to SwitchTestMain.c
; in Switch_4F120asm.  When switch #1 is pressed, the blue
; LED comes on.  When switch #2 is pressed, the red LED
; comes on.  When both switches are pressed, the green LED
; comes on.  A short delay is inserted between
; polls of the buttons to compensate for your reflexes and
; the button bounce.  The following color combinations can
```

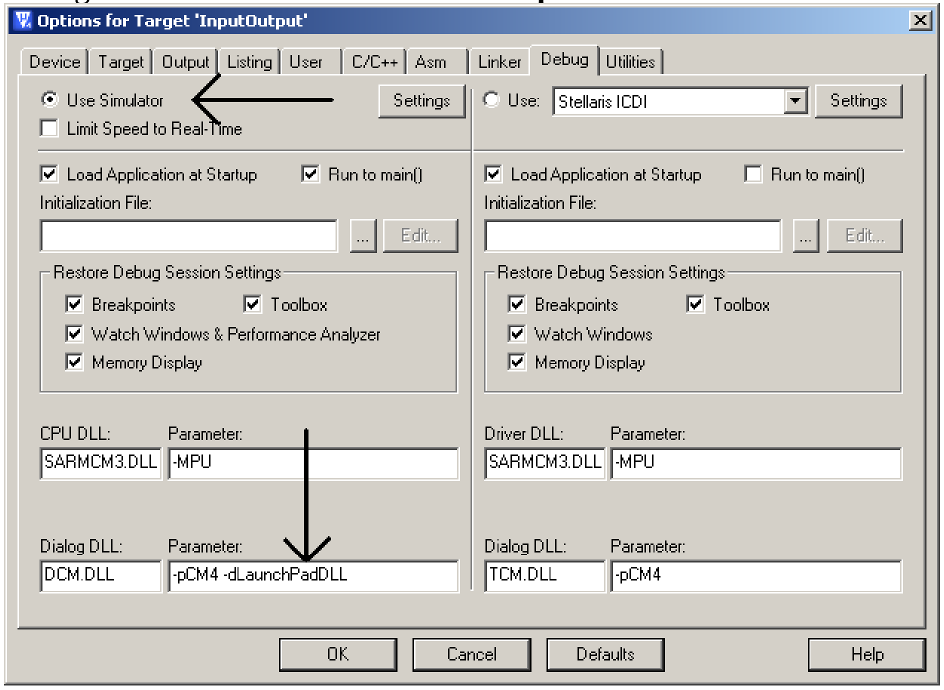
Deliverable: Get a screenshot of the Build Mode Window showing your project name.

Do not modify the startup.s file. The main.s file contains the code with a complete description shown as comments in this format **;abc…xyz**

1) Look at Project→ Options for target→ Target tab crystal should be 16 MHz, debugger is simulator.

1a) The installer should have put the file LaunchPadDLL.dll into Keil\ARM\BIN folder . Open the **InputOutput_4C123asm** project, and make this option

Dialog DLL **DCM.DLL** Parameter **-pCM4 -dLaunchPadDLL**



Options for Target 'InputOutput'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Settings
Edit...
OK
Cancel
Defaults
Help

[● Use Simulator] ○ Use: Stellaris ICDI
☐ Limit Speed to Real-Time

✓ Load Application at Startup	✓ Load Application at Startup
✓ Run to main()	☐ Run to main()
Initialization File:	Initialization File:
Restore Debug Session Settings	Restore Debug Session Settings
✓ Breakpoints	✓ Breakpoints
✓ Toolbox	✓ Toolbox
✓ Watch Windows & Performance Analyzer	✓ Watch Windows
✓ Memory Display	✓ Memory Display
CPU DLL: SARMCM3.DLL	Driver DLL: SARMCM3.DLL
Parameter: -MPU	Parameter: -MPU
Dialog DLL: DCM.DLL	Dialog DLL: TCM.DLL
Parameter: -pCM4 -dLaunchPadDLL	Parameter: -pCM4

2) Click OK→ Rebuild → Evaluation mode warning Click OK→ Debug.

Deliverable: Get screenshot of the Debug Mode windows showing the project name.

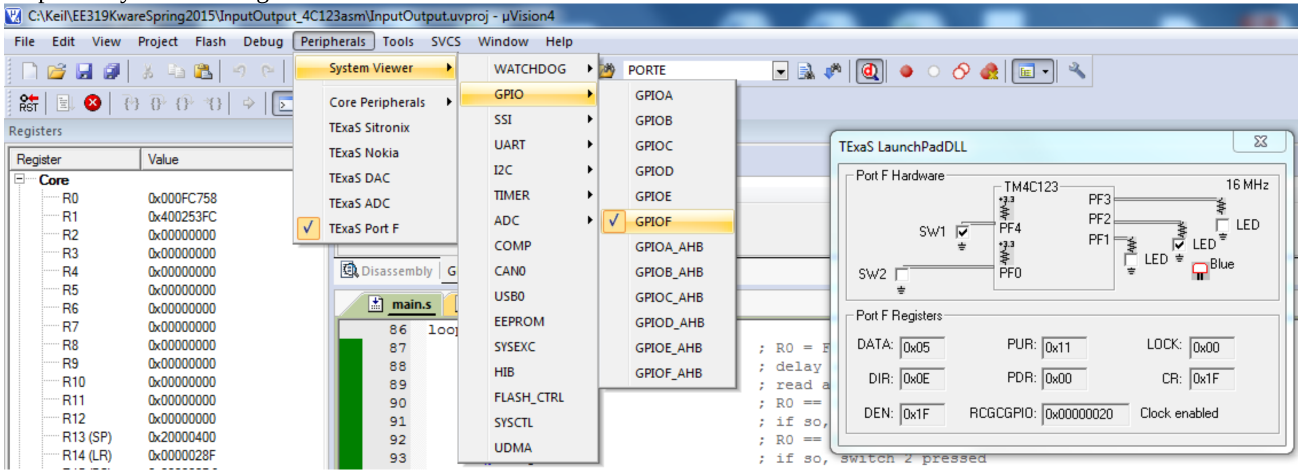
3) In the Debug Mode Window, click Run.

4) Go to Peripherals→Texas Port F. The Texas Launchpad shows the simulation of the Launchpad. Click and unlick the switches (SW1 and SW2) to see what color LED will light up.

Question: What is this assembly program doing?

You can also view the specific registers for a General Purpose IO port (GPIO). To open PortF, Open Peripherals→System Viewer→GPIOF. Look at PortF bits 4,3,2,1,0 in debugger.

Question: Will this software ever turn on all three LED colors (PF3,2,1 = 111)?



C:\Keil\EE319KwareSpring2015\InputOutput_4C123asm\InputOutput.uvproj - µVision4

Register	Value
Core	
R0	0x000FC758
R1	0x400253FC
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0x0000028F

```
loo
; R0 = F
; delay
; read a
; R0 ==
; if so,
; R0 ==
; if so, switch 2 pressed
```

Port F Registers

DATA:	0x05	PUR:	0x11	LOCK:	0x00
DIR:	0x0E	PDR:	0x00	CR:	0x1F
DEN:	0x1F	RCGCGPIO:	0x00000020	Clock enabled	

5) Experiment with single stepping features

Reset → Single step until LED comes on

Deliverable: Record a screenshot with the LED on and the name of the file.

5) Experiment with step over features

Reset → Single step over until LED flashes

6) Experiment with break point features

Reset → Click on any line and insert a breakpoint

Run (notice it stops)

Run (notice it stops)

Question: What happens when you run the code?

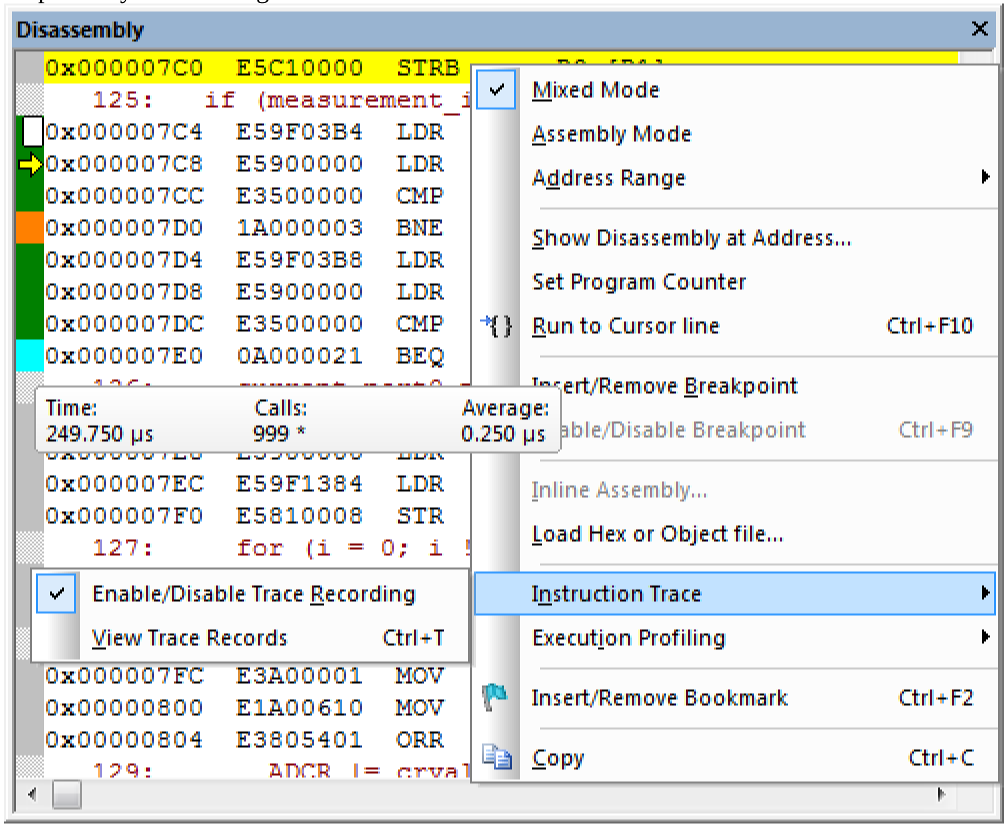
III. How to create a project from an existing project

1) Copy **InputOutput_4C123c**. and create a new project called **InputOutput_4C123_lastname_c .** Follow the directions in the handout titled "Porting a project".
Open the .c file and insert your full name as a comment in c **; First and last name**
Deliverable: Get a screenshot of your name as a comment and the compiling window (0 errors)

2) Rebuild → Evaluation mode warning Click OK→ Debug. In the Debug Mode Window, click Run.
The Texasl Launchpad shows the simulation of the Launchpad. Click and unlick the switches (SW1 and SW2).
Question: What is this .c program doing?

3) Open the Disassembly Window.
The **Disassembly Window** shows the program execution in assembly code, or, intermixed with the source code (device dependent). When the Disassembly Window is the active window, then all debug-stepping commands work on assembly level.
Open the window with the menu **View — Disassembly Window**.

- Enable the trace history with **View - Trace - Enable Trace Recording**.
- Use the Context Menu to access commands or set options quickly.
- Set Breakpoints by clicking into the left (colored) margin.


Disassembly
Mixed Mode
Assembly Mode
Address Range
Show Disassembly at Address...
Set Program Counter
Run to Cursor line Ctrl+F10
Insert/Remove Breakpoint
able/Disable Breakpoint Ctrl+F9
Inline Assembly...
Load Hex or Object file...
Instruction Trace
Execution Profiling
Insert/Remove Bookmark Ctrl+F2
Copy Ctrl+C
Enable/Disable Trace Recording
View Trace Records Ctrl+T

```
0x000007C0  E5C10000  STRB
    125:    if (measurement_i
0x000007C4  E59F03B4  LDR
0x000007C8  E5900000  LDR
0x000007CC  E3500000  CMP
0x000007D0  1A000003  BNE
0x000007D4  E59F03B8  LDR
0x000007D8  E5900000  LDR
0x000007DC  E3500000  CMP
0x000007E0  0A000021  BEQ
0x000007EC  E59F1384  LDR
0x000007F0  E5810008  STR
    127:        for (i = 0; i !
0x000007FC  E3A00001  MOV
0x00000800  E1A00610  MOV
0x00000804  E3805401  ORR
    129:           ADCR |= crval
```

Time:	Calls:	Average:
249.750 µs	999 *	0.250 µs

The code execution markers identify:

▨**No Code** - Lines with no code are marked with a light gray block.
▨**Unexecuted Code** - Unexecuted lines (instructions) are marked with a dark gray block.
🟩**Executed Code** - Fully-executed lines (instructions) are marked with a green block. A green block on a branch instruction indicates that both the true and false conditions have been tested.
🟦**Branch Condition True** - A cyan (blue) block indicates that only this condition of a branch was true and therefore always taken.
🟧**Branch Condition False** - An orange block indicates that this condition of a branch was never true and therefore never taken.
Deliverable: Get a screenshot of the disassembly window.

Compare the assembly code from the disassembly window to the assembly code from Step II.
Question: Are they different? If yes, explain how? In your own words, analyze each line of the code and explain how the c program is executed in assembly code.
Question: What is the 1st line that is 1) fully executed code 2) branch condition true 3) branch condition false.
Question: How are the programs from Stesp II. and III different? How are they the same? Which one is easier to understand and why?

Deliverable: Fill in the table below

SW1	SW2	*II. LED color(assembly file)*	*III. LED color(.c file)*
0	0		
0	1		
1	0		
1	1		