**Downloading the Integrated Development Environment (IDE)**

1. OpenSTM32 System Workbench for STM32: www.openstm32.org

**Creating a New Project**

1. Launch the System Workbench for STM32.
2. The *Eclipse Launcher* dialog box will prompt you to enter a *Workspace*. Enter the directory where your projects will reside. It is best to keep the directory path short.
3. The Eclipse welcome screen will appear. Dismiss it by clicking on the *Workbench* icon in the upper right.
4. Click on *File -> New -> C Project*.
5. For the *Project name*, enter *Session07*.  
   The *Use default location* checkbox should be checked.  
   For the *Project Type*, under *Executable* select *Ac6 STM32 MCU Project*.  
   For *Toolchains*, the only choice should be Ac6 STM32 MCU GCC.  
   The *Show project types and toolchains ...* checkbox should be checked.  
   Click the *Next* button.
6. Under *Configurations* both *Debug* and *Release* should be checked.  
   Click on the *Next* button.
7. Click on the *Board* tab.  
   For the *Series* select *STM32L4*.  
   For the Board select *B-L475E-IOT01A1*.  
   Click on the *Next* button.
8. In the first section, select *Hardware Abstraction Layer (Cube HAL).*  
   If the second section does not say *Firmware 'STM32Cube\_FW\_L4\_V1.14.0' has been found*, click on *Download target firmware*.  
   In the third section, select *Add low level drivers in the project*. Below that select *As static external libraries*.  
   In the section *Additional drivers*, nothing should be selected.  
   In the section *Additional utilities and third-party utilities*, *FreeRTOS* should be selected.  
   Click on the *Finish* button.
9. In the *Project Explorer* window on the left, select the *Session07* project.  
   Click on *Project -> Build Project*. The progress of the build can be observed in the *Console* window at the bottom.
10. Make sure the Discovery board is plugged into your computer. Clock on *Run -> Run* and then select *Ac6 STM32 C/C++ Application* to download the firmware image into the STM32L475E microcontroller. The LED near the USB micro-B connector on the board will blink between red and green as the firmware image is downloaded.
11. At this point, there should be no activity on the Discovery board, as you have just loaded code that does nothing! Now we need to add some code to the empty main() function.
12. In the *Project Explorer* window, under the *Session07* project, expand *src* and right-click on *main.c* and delete it. You will be provided with a zipfile called *session07.zip* which contains the files *main.c* and *main.h*, shown below. Copy *main.c* into the *src* directory and main.h into the *inc* directory. Refresh the project, then build it and run it.
13. Experiment with changing LED\_TOGGLE\_DELAY and building and running the code.

//

// COMP-GENG 422 - Tom Lupfer

//

// Session 7 - Thread creation

//

#include "main.h"

#define LED\_TOGGLE\_DELAY 500

#define LED\_TASK\_STACK\_SIZE 1000

#define LED\_TASK\_PRIORITY 1

void LedBlinkTask(void \*pvParameters);

static void SystemClock\_Config(void);

int main(void)

{

// Initialize the hardware and the system clocks

HAL\_Init();

SystemClock\_Config();

// Initialize LED2

BSP\_LED\_Init(*LED2*);

// Create the LED blink task

xTaskCreate(LedBlinkTask, // Task function

"LedBlinkTask", // Task name, for debugging

LED\_TASK\_STACK\_SIZE, // Stack size

NULL, // Task parameter

LED\_TASK\_PRIORITY, // Task priority

NULL); // Task handle

// Start the scheduler, which never exits

vTaskStartScheduler();

}

// Task to blink LED2

void LedBlinkTask(void \*pvParameters)

{

while (TRUE)

{

BSP\_LED\_Toggle(*LED2*);

vTaskDelay(LED\_TOGGLE\_DELAY);

}

}

//

// System Clock Configuration

//

// System Clock source = PLL (MSI)

// SYSCLK(Hz) = 80000000

// HCLK(Hz) = 80000000

// AHB Prescaler = 1

// APB1 Prescaler = 1

// APB2 Prescaler = 1

// MSI Frequency(Hz) = 4000000

// PLL\_M = 1

// PLL\_N = 40

// PLL\_R = 2

// PLL\_P = 7

// PLL\_Q = 4

// Flash Latency(WS) = 4

//

static void SystemClock\_Config(void)

{

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_OscInitTypeDef RCC\_OscInitStruct;

// MSI is enabled after System reset, activate PLL with MSI as source

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_MSI;

RCC\_OscInitStruct.MSIState = RCC\_MSI\_ON;

RCC\_OscInitStruct.MSIClockRange = RCC\_MSIRANGE\_6;

RCC\_OscInitStruct.MSICalibrationValue = RCC\_MSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_MSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 40;

RCC\_OscInitStruct.PLL.PLLR = 2;

RCC\_OscInitStruct.PLL.PLLP = 7;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if(HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

while (TRUE) // hang if there is an initialization error

;

}

// Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2 clock dividers

RCC\_ClkInitStruct.ClockType = (RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_HCLK |

RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2);

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if(HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_4) != *HAL\_OK*)

{

while (TRUE) // hang if there is an initialization error

;

}

}

//

// COMP-GENG 422 - Tom Lupfer

//

// General header file

//

#ifndef MAIN\_H\_

#define MAIN\_H\_

#include "stm32l4xx\_hal.h"

#include "stm32l475e\_iot01.h"

#include "freertos.h"

#include "task.h"

#define FALSE 0

#define TRUE 1

typedef int Bool; // Boolean

#endif // MAIN\_H\_

**Quiz Prep Notes**

**HAL (Hardware Abstraction Layer)**: layer of programming that allows a computer OS to interact with a hardware device at a general or abstract level rather than hardware level (called from OS kernel or device driver)

**BSP (Board Support Package)**: layer of software containing hardware-specific drivers to allow an OS (usually RTOS) to function in a certain hardware environment

**IDE Stats Explanation**

**Text**: The code and constant data that ends up in flash memory

**Data**: The initialized data (that might be associated with the global/extern variable

**BSS**: Block Started By Symbol ~ all the uninitialized data (ends up in RAM), so better save space!