SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

Real-Time Embedded Systems - ECEN 121
Lab 2

Writing and debugging in the STM32cubeMX environment

Dylan Thornburg and Kai Hoshide

Prelab:

This week's prelab can be done by project groups.

The STM32cubeMX development environment allows for automated generation of Keil projects. We are going to use those tools. Manuals have been identified for this assignment on Camino and examples have been demonstrated in class. There is an introductory document explaining how to configure a project on the ECC machines. That document is called *Using the STM32cubeMX environment on ECC Lab Computers* and is in the Lab Handouts directory. There is a document that explains copying STM32cubeMX projects called *Copying a CubeMX Project* that is also in the Lab Handouts directory. Read them both in advance of lab.

Review sections 3.11.2, 3.12, and 31 of *STM32L4 HAL and low-layer drivers* (in the Readings directory). In your prelab,:

1. identify the parameters that need to be provided to the function HAL_GPIO_TogglePin()
Parameters:
GPIOx   Selects the GPIO peripheral to be pin toggled where x is the letter (b, e, etc.)
GPIO_PinSpecifies the pins to be toggled.

2. Write an example call to this function using constants from the stm32l476xx.h file.
HAL_GPIO_TogglePin (GPIOE, ((uint16_t)0x0100));
There was no GPIO_PIN_8 constant in the file so I just used the actual number.

In lab:

You are now going to:

1) Write a simple LED-blinking app
2) Copy and/or branch the project and reconfigure it in STM32cubeMX

3) Make sure your app still works
4) Answer some questions about STM32cubeMX

This will be followed up by another project using the LED strip.

Step 1: Make some blinky lights.

Create a project called MX First Project using the directions in *Using the STM32cubeMX environment on ECC Lab Computers*.

One an MX project is created, it is very important that you only change or add code in the locations that are marked for user modification. Otherwise, any time you go back to the STM32cubeMX tool and change the configuration, it will overwrite all of your code.

You may place code in 2 places:

1) You may add additional .c, .s, or .h files to the project. They belong in the Src and Inc directories, respectively. Keil will usually put them in the right place. They should be preserved if you rerun STM32cubeMX. In any case they will not be deleted and can be added back in.
2) You can add code in files like main.c only in the spots between comments that are allocated for user code. It is easy to get confused and not put the code in the right place – be careful.

In main.c, find the main() routine. Within that main() routine, there is an infinite loop:

```
while (1) {

 }
```

Near that loop there are two comments:

```
/* USER CODE BEGIN WHILE */

/* USER CODE END WHILE */
```

Note that the placement is weird. The while statement is between these two comments, but the closing brace is outside of them. You can place code inside of this infinite loop, but it must be after the while statement and before the second comment.

First include a call to HAL_GPIO_TogglePin() that is configured to toggle the red LED. You can research the HAL_GPIO_TogglePin() call in the STM32L4 HAL and low-layer drivers manual in Chapter 31. Best practice is not to use the specific port name and pin like you did in the prelab, but to use mnemonics generated by STM32cubeMX. These mnemonic parameters to provide to the HAL_GPIO_TogglePin() are determined by the configuration in STM32cubeMX, particularly by the way GPIO pins were named. These names are defined in main.h.

Go back to the STM32cubeMX and look for the names assigned to the red and green LED pins. (The part in brackets does not count – it is a comment). It may help to look at the *STM32L476-DISCO Schematics* in the ARM References directory. Find the pin names and search main.h for the relevant parameters to use with HAL_GPIO_TogglePin().

Now write your call to HAL_GPIO_TogglePin() to toggle the red LED.

If you build and download this project then reset your board, the red LED should turn on.

Next, add the HAL_Delay() function to the loop so that the light blinks on for ½ second and off for ½ second. HAL_Delay() is in the *STM32L4 HAL and low-layer drivers manual* in Chapter 6. Test this.

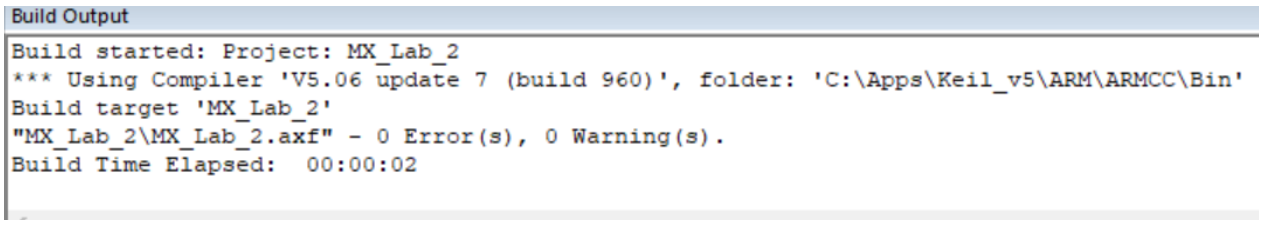
Next add another call to HAL_GPIO_TogglePin() that is configured to toggle the green LED after the HAL_Delay() function and another ½ second HAL_Delay() function after the green LED toggle. Get this working and demo it. This is DEMO 1. Turn in your main.c file and a screen shot of a successful build in your report.

CODE:

```
while (1)
 {
                HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
                HAL_Delay(500);
                HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
                HAL_Delay(500);
  /* USER CODE END WHILE */
  MX_USB_HOST_Process();

  /* USER CODE BEGIN 3 */
 }
```

SCREENSHOTS:



```
Build started: Project: MX_Lab_2
*** Using Compiler 'V5.06 update 7 (build 960)', folder: 'C:\Apps\Keil_v5\ARM\ARMCC\Bin'
Build target 'MX_Lab_2'
"MX_Lab_2\MX_Lab_2.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:02
```

Step 2: Optimize your chip configuration

There is a document that explains copying STM32cubeMX projects called *Copying a CubeMX Project* that is also in the Lab Handouts directory. Use either procedure described in that document to make a copy of MX First Project and call it MX Second Project.

Once this is all done, double click on the new MX project file to open it. It can take a while.

You probably noticed that there were a lot of peripherals being configured and initialized in MX First Project. In fact, if you look at your board, the green LED next to the second USB port will be on since that port is running and powered up. You don't need all that stuff, so we are going to try to simplify the configuration to reduce complexity.

Go to the Pinout and Configuration tab on STM32cubeMX.

First we are going to disable USB. Open the Middleware tab on the very left and select USB-HOST. In the drop-down menu, select Disable.

Next, the QuadSPI Flash. Under the Connectivity tab, select the QuadSPI (in green). Select Disable in the drop-down menu.

In the same Connectivity tab, disable I2C1, I2C2, SPI2, USART2, and

USB_OTG_FS. Under the Multimedia tab, disable LCD and SAI1.

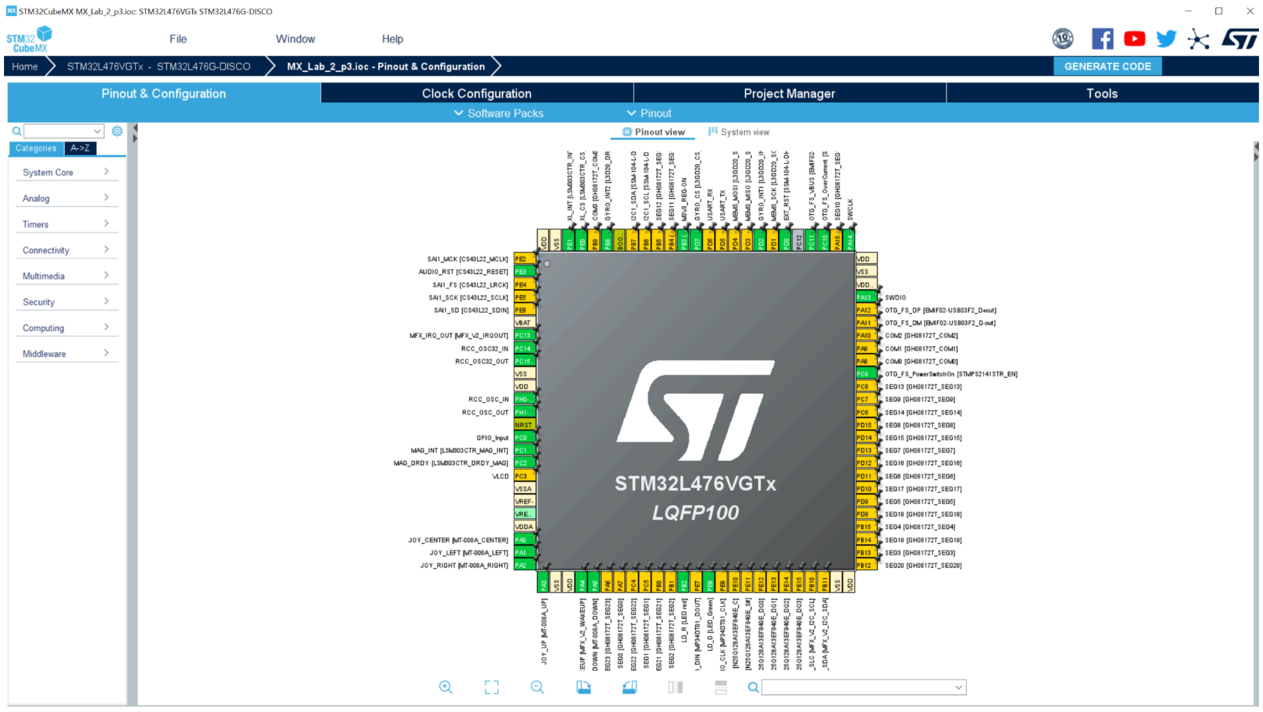
Save the project file and re-run Generate Code. Then open the project when asked. You should be able to build, download, and run that code just like before. If it does not work, you probably put some of your code in the wrong spot. Fix it in both projects. Turn in your main.c file and a screen shot of a successful build in your report.

Step 3: Re-using the configuration

It can be a little bit simpler to just copy a configuration setup and not the entire project.

In the MX directory where your projects are, create a folder MX Third Project next to MX First Project and MX Second Project. Now go to the MX Second Project folder and copy the MX configuration file MX Second Project. Paste it into MX Third Project and rename it MX Third Project.

Now open that project in STM32cubeMX. Generate code. You should get an empty project that has all the pins and peripherals configured as in MX Second Project. Turn in a screen shot of your STM32cubeMX Pinout and Configuration page.



3

Now write any program that has the following characteristics:

1. It has at least one global variable
2. It has at least one #define
3. It has at least one local variable
4. It has at least some code that runs before the infinite loop in main()
5. It has at least some code that runs in the infinite loop in main().

Write your program, compile it, and test it.

Now, quit Keil, go back to STM32cubeMX, and regenerate code. Open the Keil project and rebuild.

Test your code again. If it no longer works, you put some code in the wrong place. Fix it. **This is**

Demo 2

Turn in main.c

CODE ENDING WITH MAIN:

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file          : main.c
 * @brief         : Main program body
 ******************************************************************************
```

```
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define r 4
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
RTC_HandleTypeDef hrtc;

/* USER CODE BEGIN PV */
int global = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_RTC_Init(void);
```

```
/* USER CODE BEGIN PFP */


/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
        int local = 0;
        local++;
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
			HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
			HAL_Delay(500);
			HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
			HAL_Delay(500);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```