STM32 Course Report

Cao Zhengyang
Department of Electronics and Electrical Engineering
12110623@mail.sustech.edu.cn

I. Lecture 1

A. Summary

The introductory lecture, led by Instructor Wang Xiaojing, offered a comprehensive overview of STM32 microcontrollers. Key topics included the significance of STM32 as a 32-bit microcontroller from STMicroelectronics, an introduction to the C language, and the use of Keil5 (MDK) for programming. Practical aspects covered lighting an LED through both register and firmware library programming.

The lecture emphasized the importance of foundational learning materials, including datasheets, reference manuals, and schematic diagrams. Students were encouraged to explore these resources on the STMicroelectronics website.

The homework assignment involves software installation, conducting a LED lighting experiment, and modifying the code to light a different LED (PC13). The lecture concluded by highlighting the critical role of the startup file as the project entrance.

In summary, the lecture provided a well-rounded introduction to STM32, blending theory with practical demonstrations and hands-on assignments to ensure a solid understanding among students.

B. LED lighting using gpio pc13

```
// Enable GPIOC clock and set Pin-
//13 as push-pull output (10MHz)
*(unsigned int*)0x40021018 |= (1 << 4);
*(unsigned int*)0x40011004 = 0xFF0FFFFF;
*(unsigned int*)0x40011004 |= 0x00100000;

// Turn on the LED by clearing the
// corresponding bit in ODR
*(unsigned int*)0x4001100C &= ~(1 << 13);

// To turn off the LED, comment the
// line above and uncomment the line below
// *(unsigned int*)0x4001100C |= (1 << 13);
```

II. Lecture 2

A. Firmware library programming

```
// Main firmware library
#include <stm32f103_hal.h>
// GPIO library
#include <stm32f103_gpio.h>

// Enable clock for GPIOC
RCC->APB2ENR = (1 << 4);

// Configure GPIO Pin 13 as pull-up input

GPIO_InitTypeDef  GPIO_InitStruct;
// Specify Pin 13
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_13;
// Input mode with pull-up
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;

// Apply the configuration using the
// initialized structure
GPIO_Init(GPIOC, & GPIO_InitStruct);

// Reset the Pin 13
GPIO_ResetBits(GPIOC, GPIO_Pin_13);
```

B. Register Programming

```
// Configure PC13 as a pull-up input
// Set PC13's mode bit as input
(*((unsigned int *)0x400210181)) |= (1 << 4);
// Clear PC13 output data ensure initially low
(*((unsigned int *)0x40011004)) &= ~(0xFF0FFFFF);
// Set PC13's pull-up bit
(*((unsigned int *)0x400110041)) |= (0x00800000);
// Clear PC13's pull-up bit in low power mode
(*((unsigned int *)0x4001100c)) &= (~(1 << 13));
```

III. Lecture 3

A. GPIO_Init break down

The ‘GPIO_Init‘ function, which is used to configure the mode and speed of the specified GPIO pins. It takes two parameters:

- **GPIOx:** The GPIO port to be configured.
- **GPIO_InitStruct:** A pointer to a `GPIO_InitTypeDef` structure, which contains the configuration parameters for the GPIO pins.

The function performs the following steps:

1) **Parameter Checks:** It checks that the provided parameters are valid, ensuring that the GPIO port and pin configurations are within the valid range. This is done using the ‘IS_GPIO_ALL_PERIPH‘ and ‘IS_GPIO_PIN‘ macros.
2) **GPIO Mode Configuration:** It extracts the GPIO mode bits from the ‘GPIO_InitStruct‘ structure and combines them with the output speed bits (if applicable). This is done using the ‘currentmode‘ variable.

3) **GPIO CRL and CRH Configuration:** It iterates through the lower and upper 8 pins of the GPIO port separately, using the ‘CRL‘ and ‘CRH‘ registers. For each pin, it determines the corresponding mode bits using the ‘pinpos‘ variable. It then sets or clears the mode bits in the appropriate register based on the ‘currentmode‘ value.
4) **Pull-up/Pull-down Resistance:** If the GPIO mode is ‘GPIO_Mode_IPD‘ (pull-down) or ‘GPIO_Mode_IPU‘ (pull-up), it sets the corresponding bits in the ‘BRR‘ or ‘BSRR‘ registers to set or clear the output data register.
5) Write Back Registers: Finally, it updates the ‘CRL‘ and ‘CRH‘ registers with the modified ‘tmpreg‘ values.

The flow chart is as follows:

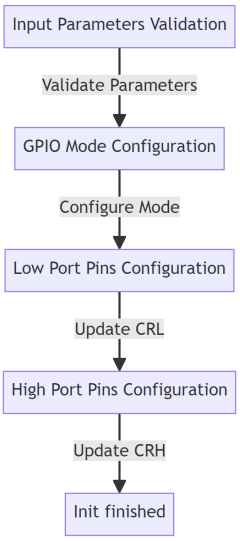

Fig. 1. Flow chart of ”GPIO_Init”

B. File tree

Using the command line code, we can get a clear look about the file structure inside a folder:

```
$D:\Download\Template_FWLib> tree /a /f
```

Using LLM to turn the tree output into mermaid code, then render the file structure plot as follow:

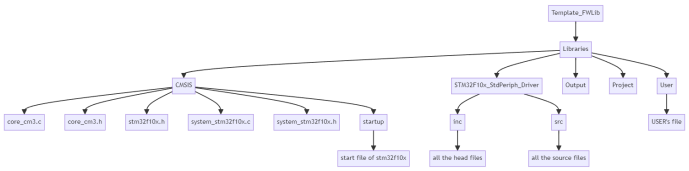

Fig. 2. File structure