Professor: John Penvenne

Written by:
Brayden Daly

EE 329-05

Spring 2025

MWF 1pm-3pm

Project A1

2025-Apr-9

Introduction:

I programmed the STM32 Dev board to turn on 4 LEDs and count from 1-15 using binary. Then I infinitely called Test Function so I could test the latency and function call time with different information. The project is completely working. The only bugs I encountered included bad connectivity with the breadboard and an incorrect clock rate, which I accidentally prescaled while trying to set up a timer. I learned to debug my projects with LEDs to work step by step.

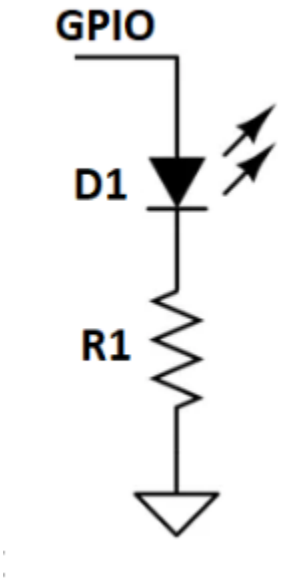

Figure 1: Pull-Down Resistor with LED

Table A1(a): LED circuit calculations & measurements for R =558 Ω

LED drive circuit	GPIO Vout	LED Vf	LED current
calculations	3.3	2.5V	1.1 mA
measurements	**3.3V**	**2.61V**	**1.1mA**

My measurements were close to the calculated values. The measured forward voltage across my LED was larger than the calculated LED forward voltage.

Table A1(b): Instruction execution timing for different data types

Execution Time	uint8_t	int32_t	int64_t	float	double
function call latency	3.52 us	3.52 us	4.26 us	3.52 us	4.52 us
test_var = num	1.76 us	1.76 us	2.5 us	1.76 us	2.5 us
test_var = num + 1	2.28 us	2.28 us	3.04 us	2.28 us	25 us
test_var = num * 3	2.72 us	2.72 us	4.48 us	2.74 us	19.4 us
test_var = num / 3	2.94 us	2.94 us	27 us	6 us	28 us
test_var = num * num	2.5 us	2.28 us	6.1 us	2.5 us	19 us
test_var = num % 10	4.3 us	4.3 us	27.1 us	N/A	N/A
test_var = pow(num, 3)	N/A	2.29 ms	2.24 ms	2.19 ms	2.19 ms
test_var = sqrt(num)	N/A	350 us	396 us	342 us	332 us
test_var = sin(num)	N/A	598 us	656 us	598 us	594 us

Table A1 (b) lists the latency times for each data type used in the Test Function call as well as all of the times it take to run the instruction in the Test Function with different data types. The float and the int32_t take the same amount of time because a float is 4 bytes (32 bits). Similarly, the double is 64, so it should be similar times to the int64_t data type. The pow() and sqrt() functions take much longer to run than num or num + 1 because they are more complex, and they are calling other functions which require more overhead.

Youtube Video (729,027 KB; 04/09/2025): https://youtu.be/h8oplWXZCFo

Appendix:

---

main.h

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.h
 * @brief          : Header for main.c file.
 *                   This file contains the common defines of the application.
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 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 */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32l4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
```

```
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define LD3_Pin GPIO_PIN_14
#define LD3_GPIO_Port GPIOB
#define USB_OverCurrent_Pin GPIO_PIN_5
#define USB_OverCurrent_GPIO_Port GPIOG
#define USB_PowerSwitchOn_Pin GPIO_PIN_6
#define USB_PowerSwitchOn_GPIO_Port GPIOG
#define STLK_RX_Pin GPIO_PIN_7
#define STLK_RX_GPIO_Port GPIOG
#define STLK_TX_Pin GPIO_PIN_8
#define STLK_TX_GPIO_Port GPIOG
#define USB_SOF_Pin GPIO_PIN_8
#define USB_SOF_GPIO_Port GPIOA
#define USB_VBUS_Pin GPIO_PIN_9
#define USB_VBUS_GPIO_Port GPIOA
#define USB_ID_Pin GPIO_PIN_10
#define USB_ID_GPIO_Port GPIOA
#define USB_DM_Pin GPIO_PIN_11
#define USB_DM_GPIO_Port GPIOA
#define USB_DP_Pin GPIO_PIN_12
#define USB_DP_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
#define LD2_Pin GPIO_PIN_7
#define LD2_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

---

main.c

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 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"
#include <math.h>
typedef double var_type;
void SystemClock_Config(void);
var_type TestFunction(var_type num);
UART_HandleTypeDef hlpuart1;
PCD_HandleTypeDef hpcd_USB_OTG_FS;
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_LPUART1_UART_Init(void);
static void MX_USB_OTG_FS_PCD_Init(void);
/**
 * @brief  The application entry point.
 * @retval int
 */
/* ---------------------------------------------------------------------------
* function : DelayMs( )
* INs      : uint32_t delay (amount of delay)
* OUTs     : None (void)
* action   : Sets a for loop iterating to the delay input number to
*                        : p
* authors  : Brayden Daly - bdaly01@calpoly.edu
* version  : 1
* date     : 04092025
* ------------------------------------------------------------------------- */
void DelayMs(uint32_t delay)
{
        for (uint32_t i = 0; i < delay * 20; i++)
        {
                continue;
        }
}
/* ---------------------------------------------------------------------------
* function : main( )
* INs      : none
* OUTs     : None (void)
* action   : count 1-15 in binary on LEDs then infinitely
*                        :run Test Function
* authors  : Brayden Daly - bdaly01@calpoly.edu
* version  : 1
* date     : 04092025
* ------------------------------------------------------------------------- */
void main(void)
{
        int counter = 0; //Dummy counter variable to exit led counting loop
        HAL_Init();
        SystemClock_Config();
        // configure GPIO pins PC0, PC1 for:
```

```
   // output mode, push-pull, no pull up or pull down, high speed
   RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
   GPIOC->MODER &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2
                | GPIO_MODER_MODE3);
   GPIOC->MODER |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0 | GPIO_MODER_MODE2_0
                | GPIO_MODER_MODE3_0);
   GPIOC->OTYPER &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1 | GPIO_OTYPER_OT2
                | GPIO_OTYPER_OT3);
   GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2
                | GPIO_PUPDR_PUPD3);
   // Set very high output speed for PC0, PC1, PC2, and PC3
   GPIOC->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED0_Pos) |
                       (3 << GPIO_OSPEEDR_OSPEED1_Pos) |
                       (3 << GPIO_OSPEEDR_OSPEED2_Pos) |
                       (3 << GPIO_OSPEEDR_OSPEED3_Pos));
   GPIOC->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2
                | GPIO_PIN_3);                                          //preset PC0, PC1, PC2, PC3 to
0
   while (1)                                                                    //infinite loop to
avoid program exit
   {
         if (counter < 3)                                           //run led counting twice
         {
                 for (int i = 1; i < 16; i++)             //count 1 to 15
                 {
                                 uint32_t tempvar_for_leds = i;   //set a variable to keep track of
                                 if (tempvar_for_leds >= 8)              //check if MSB is 8 or more
                                 {
                                         GPIOC->BSRR = (GPIO_PIN_3); //turn on PC3
                                         tempvar_for_leds -= 8;          //subtract 8 for remaining
leds
                                         DelayMs(1000);                          //delay to see lights
easier
                                 }
                                 if (tempvar_for_leds >= 4)              //check if MSB is 4 or more
                                 {
                                         GPIOC->BSRR = (GPIO_PIN_2); //turn on PC2
                                         tempvar_for_leds -= 4;          //subtract 4
                                         DelayMs(1000);                          //delay
                                 }
                                 if (tempvar_for_leds >= 2)              //check is MSB is 2 or more
                                 {
                                         GPIOC->BSRR = (GPIO_PIN_1); //turn on PC1
                                         tempvar_for_leds -= 2;          //subtract 2 from tempvar
                                         DelayMs(1000);                          //delay
                                 }
                                 if (tempvar_for_leds >= 1)              //check is MSB is 1 or more
                                 {
                                         GPIOC->BSRR = (GPIO_PIN_0); // turn on PC0
                                         tempvar_for_leds -= 1;      //subtract 1 from tempvar
                                         DelayMs(1000);                          //delay
                                 }
                                 GPIOC->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2
                                                 | GPIO_PIN_3); // turn off all LEDs
                                 DelayMs(1000);                                  //delay
                         }
                 counter ++;                                                     //increment counter
         }
```

```
         // time the test function call using PC0
                GPIOC->BSRR = (GPIO_PIN_0);             // turn on PC0
                var_type main_var = TestFunction(15);   // call test function
                GPIOC->BRR = (GPIO_PIN_0);              // turn off PC0
        main_var++;  // added to eliminate not used warning
   }
}
/* ---------------------------------------------------------------------------
* function : TestFunction( )
* INs      : var_type num
* OUTs     : None (void)
* action   : drives gpio pin high and low and runs the instruction
* authors  : Brayden Daly - bdaly01@calpoly.edu
* version  : 1
* date     : 04092025
* ------------------------------------------------------------------------- */
var_type TestFunction(var_type num)
{
 var_type test_var;                             // local variable
 GPIOC->BSRR = (GPIO_PIN_1);                    // turn on PC1
 test_var = num + 1;                            //instruction
 GPIOC->BRR = (GPIO_PIN_1);                     // turn off PC1
 return test_var;                               //return test_var
}
/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 /** Configure the main internal regulator output voltage
 */
 if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
 {
   Error_Handler();
 }
 /** Configure LSE Drive Capability
 */
 HAL_PWR_EnableBkUpAccess();
 __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
 /** Initializes the RCC Oscillators according to the specified parameters
 * in the RCC_OscInitTypeDef structure.
 */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSE|RCC_OSCILLATORTYPE_MSI;
 RCC_OscInitStruct.LSEState = RCC_LSE_ON;
 RCC_OscInitStruct.MSIState = RCC_MSI_ON;
 RCC_OscInitStruct.MSICalibrationValue = 0;
 RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
 RCC_OscInitStruct.PLL.PLLM = 1;
 RCC_OscInitStruct.PLL.PLLN = 71;
 RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
 RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
 RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV6;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
```

```
 {
   Error_Handler();
 }
 /** Initializes the CPU, AHB and APB buses clocks
 */
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
 {
   Error_Handler();
 }
 /** Enable MSI Auto calibration
 */
 HAL_RCCEx_EnableMSIPLLMode();
}
/**
 * @brief LPUART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_LPUART1_UART_Init(void)
{
 /* USER CODE BEGIN LPUART1_Init 0 */
 /* USER CODE END LPUART1_Init 0 */
 /* USER CODE BEGIN LPUART1_Init 1 */
 /* USER CODE END LPUART1_Init 1 */
 hlpuart1.Instance = LPUART1;
 hlpuart1.Init.BaudRate = 209700;
 hlpuart1.Init.WordLength = UART_WORDLENGTH_7B;
 hlpuart1.Init.StopBits = UART_STOPBITS_1;
 hlpuart1.Init.Parity = UART_PARITY_NONE;
 hlpuart1.Init.Mode = UART_MODE_TX_RX;
 hlpuart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
 hlpuart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
 hlpuart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
 if (HAL_UART_Init(&hlpuart1) != HAL_OK)
 {
   Error_Handler();
 }
 /* USER CODE BEGIN LPUART1_Init 2 */
 /* USER CODE END LPUART1_Init 2 */
}
/**
 * @brief USB_OTG_FS Initialization Function
 * @param None
 * @retval None
 */
static void MX_USB_OTG_FS_PCD_Init(void)
{
 /* USER CODE BEGIN USB_OTG_FS_Init 0 */
 /* USER CODE END USB_OTG_FS_Init 0 */
 /* USER CODE BEGIN USB_OTG_FS_Init 1 */
 /* USER CODE END USB_OTG_FS_Init 1 */
 hpcd_USB_OTG_FS.Instance = USB_OTG_FS;
```

```
 hpcd_USB_OTG_FS.Init.dev_endpoints = 6;
 hpcd_USB_OTG_FS.Init.speed = PCD_SPEED_FULL;
 hpcd_USB_OTG_FS.Init.phy_itface = PCD_PHY_EMBEDDED;
 hpcd_USB_OTG_FS.Init.Sof_enable = ENABLE;
 hpcd_USB_OTG_FS.Init.low_power_enable = DISABLE;
 hpcd_USB_OTG_FS.Init.lpm_enable = DISABLE;
 hpcd_USB_OTG_FS.Init.battery_charging_enable = ENABLE;
 hpcd_USB_OTG_FS.Init.use_dedicated_ep1 = DISABLE;
 hpcd_USB_OTG_FS.Init.vbus_sensing_enable = ENABLE;
 if (HAL_PCD_Init(&hpcd_USB_OTG_FS) != HAL_OK)
 {
   Error_Handler();
 }
 /* USER CODE BEGIN USB_OTG_FS_Init 2 */
 /* USER CODE END USB_OTG_FS_Init 2 */
}
/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStruct = {0};
 /* USER CODE BEGIN MX_GPIO_Init_1 */
 /* USER CODE END MX_GPIO_Init_1 */
 /* GPIO Ports Clock Enable */
 __HAL_RCC_GPIOC_CLK_ENABLE();
 __HAL_RCC_GPIOH_CLK_ENABLE();
 __HAL_RCC_GPIOB_CLK_ENABLE();
 __HAL_RCC_GPIOG_CLK_ENABLE();
 HAL_PWREx_EnableVddIO2();
 __HAL_RCC_GPIOA_CLK_ENABLE();
 /*Configure GPIO pin Output Level */
 HAL_GPIO_WritePin(GPIOB, LD3_Pin|LD2_Pin, GPIO_PIN_RESET);
 /*Configure GPIO pin Output Level */
 HAL_GPIO_WritePin(USB_PowerSwitchOn_GPIO_Port, USB_PowerSwitchOn_Pin,
                   GPIO_PIN_RESET);
 /*Configure GPIO pin : B1_Pin */
 GPIO_InitStruct.Pin = B1_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
 /*Configure GPIO pins : LD3_Pin LD2_Pin */
 GPIO_InitStruct.Pin = LD3_Pin|LD2_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
 HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
 /*Configure GPIO pin : USB_OverCurrent_Pin */
 GPIO_InitStruct.Pin = USB_OverCurrent_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 HAL_GPIO_Init(USB_OverCurrent_GPIO_Port,
                   &GPIO_InitStruct);
 /*Configure GPIO pin : USB_PowerSwitchOn_Pin */
 GPIO_InitStruct.Pin = USB_PowerSwitchOn_Pin;
 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```

```
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
 HAL_GPIO_Init(USB_PowerSwitchOn_GPIO_Port,
                 &GPIO_InitStruct);
 /* USER CODE BEGIN MX_GPIO_Init_2 */
 /* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
 /* USER CODE BEGIN Error_Handler_Debug */
 /* User can add his own implementation to report the HAL error return state */
 __disable_irq();
 while (1)
 {
 }
 /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
 /* USER CODE BEGIN 6 */
 /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
 /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```