Wyatt Tack
EE 329-01 F'24
2024-Sept-26

EE 329 A1

This code is designed to, with the use of a logic analyzer of oscilloscope, measure execution timings through the visual use of an LED, and the used tools for quantitative measurements. The code is can be changed through the data type referenced in the typedef for var_type (to measure timing vs data width) as well as the commented out instructions in the TestFunction subroutine. This program was helpful for acquainting with instruction timing, as well as loading values into registers, for both GPIO purposes as well as initializing said GPIO pins.

Link to YouTube Presentation:

https://youtu.be/Fy1MBK32LfE

Obtained Data:

1. Delay per loop iteration average: *5.4 uS per for loop iteration*
   a. Loop 1: 2x50000 counts for each cycle → 507.8 ms
   b. Loop 2: 2x70000 counts for each cycle → 781.3 ms
   c. Loop 3: 2x85000 counts for each cycle → 948.6 ms

Table A1(a): LED circuit calculations & measurements for R = 560 Ω

LED drive circuit	GPIO Vout	LED Vf	LED current
calculations	3.3-0.4 = 2.9V	1.8V	2mA
measurements	3.13V	1.87V	1.953mA

Obtained Data (ctd):

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

Execution Time	uint8_t	int32_t	int64_t	float	double
function call latency	3.503 us	3.503 us	4.007 us	3.503 us	4.508 us
test_var = num	1.749 us	1.749 us	2.501 us	1.749 us	2.501 us
test_var = num + 1	2.250 us	2.250 us	3.002 us	2.752 us	25.04 us
test_var = num * 3	2.750 us	2.751 us	4.505 us	2.750 us	19.53 us
test_var = num / 3	3.001 us	3.253 us	28.54 us	6.006 us	28.28 us
test_var = num * num	2.500 us	2.250 us	6.006 us	2.501 us	19.53 us
test_var = num % 10	4.252 us	4.755 us	27.79 us	N/A	N/A
test_var = pow(num, 3)	N/A	2.236 ms	2.304 ms	2.228 ms	2.219 ms
test_var = sqrt(num)	N/A	360.3 us	425.6 us	349.8 us	333.9 us
test_var = sin(num)	N/A	619.2 us	685.2 us	612.2 us	602.6 us

Formatted Source Code:

```
---------------------------------------------------------------------------------------
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) 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 */

/* 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 -----------------------------------------------------------*/

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

Formatted Source Code:

```
---------------------------------------------------------------------------------------
main.h
---------------------------------------------------------------------------------------


/*
*               Lab A1 for EE329
*               Instructor: John Pennevene
*
*               Written by: Wyatt Tack
*
*               Adapted from EE329 Lab Manual
*               for part a) of A1
*               https://youtu.be/Fy1MBK32LfE

*/
// file ExecutionTime.c sample code for A1
#include "main.h"
#include <math.h>
void SystemClock_Config(void);
typedef uint8_t var_type;       //definition for var_type
var_type TestFunction(var_type num);
void main(void)  {
                var_type main_var;
                HAL_Init();
                SystemClock_Config();
  // configure GPIO pins PC0-3 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);
  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 PC0-3 to 0
                 // Loop 15 count 3 times
for (int smallLoop = 0; smallLoop < 3; smallLoop++){
                for(int loopCount = 0; loopCount <= 16; loopCount++){
                                //if(main_var >= 15) main_var = 0;
                                //main_var++;  // added to eliminate not used warning
                GPIOC->ODR = (loopCount);
                for(uint32_t delaycnt = 0; delaycnt < 40000; delaycnt++);
                }
                }
  while (1)    {   // infinite loop to avoid program exit
                  // time the test function call using PC0
                  GPIOC->BSRR = (GPIO_PIN_0);             // turn on PC0
                  main_var = TestFunction(15);            // call test function
                  GPIOC->BRR = (GPIO_PIN_0);              // turn off PC0
  }
}
var_type TestFunction(var_type num) {
  var_type test_var;                                                           // local variable
  GPIOC->BSRR = (GPIO_PIN_1);             // turn on PC1
                  test_var = num;                                                                      // instruction to test
//              test_var = num + 1;
//              test_var = num * 3;
//              test_var = num / 3;
//              test_var = num * num;
//              test_var = num % 10;
//              test_var = pow(num, 3);
//              test_var = sqrt(num);
//              test_var = sin(num);
  GPIOC->BRR = (GPIO_PIN_1);              // turn off PC1
  return test_var;
}
//Below is system configurations
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  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_0) != HAL_OK)
  {
    Error_Handler();
  }
}
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }
}
#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{
}
#endif /* USE_FULL_ASSERT */
```