   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*
   1. Loop 1: 2x50000 counts for each cycle 🡪 507.8 ms
   2. Loop 2: 2x70000 counts for each cycle 🡪 781.3 ms
   3. 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 \*/