   Wyatt Tack

Joshua Lin  
   EE 329-01 F'24

Group

   2024-Oct-11

**EE 329 A4**

This code is designed to validate the use of Timers, with multiple sections using/measuring outputs due to the onboard timers. The STM32L4 has multiple on board clocks, the one we were using was a 32-Bit timer TIM2, and its mid way capture register CC1. Using this timer we measured interrupt time, the smallest frequency countable on the interrupt, and then applied it into a reaction timer system. The reaction timer worked as waiting for an input button, prompting the user to get ready, randomly turning on an LED, and logging the decimal seconds taken to turn on the LED.

**Link to YouTube Presentation:**

<https://youtu.be/eCX4bWneXJQ>

**Calculations:**

Equation A4.a(a): Frequency Calculations:

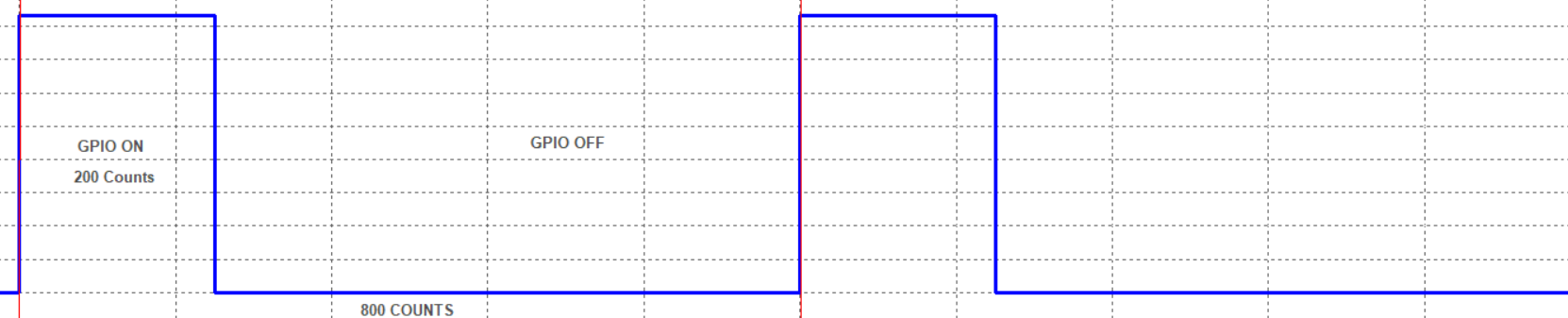
Counts of 4MHz clock for 5KHz wave:

Period = - 1 🡪 **Period = 799**

25% DC = - 1🡪 **CC1 Counts = 199**

**Captures:**

Figure A4.a(a): Drawn 4MHz and 5KHz clock sketches (Not to Scale)



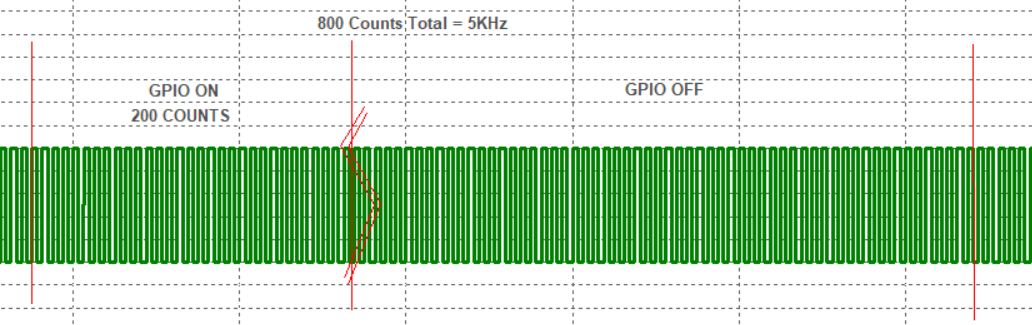


Figure A4.a(b): Oscilloscope 5KHz 25%DC Capture:

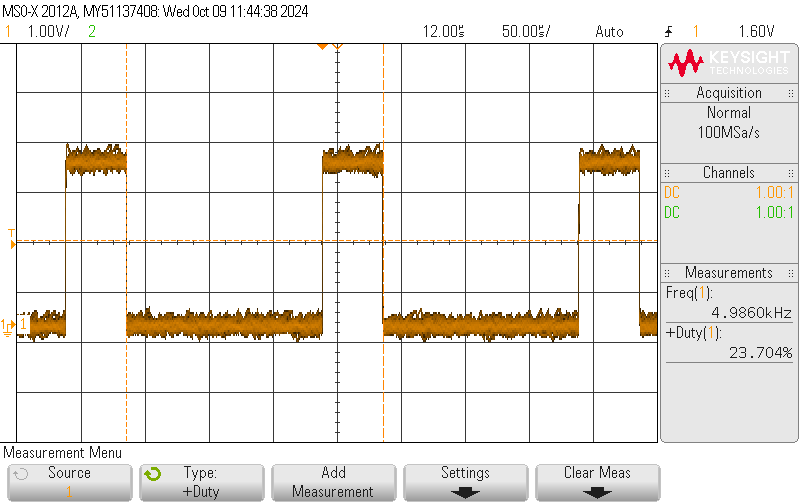


Figure A4.b(a): Oscilloscope MCO ISR Length Capture:

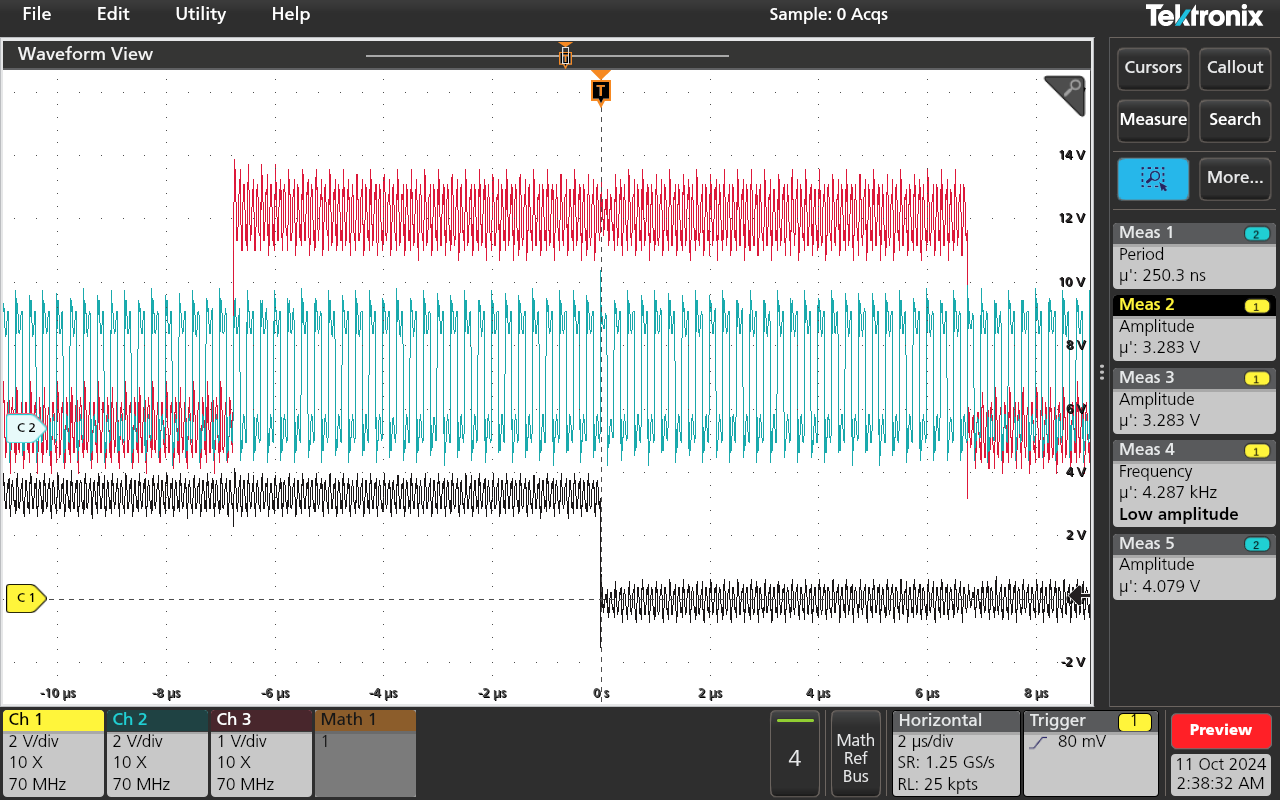


Figure A4.c(a): Oscilloscope CCR1 Smallest Output Capture:



**Obtained Data:**

Table A4.b(a): ISR MCO instruction length

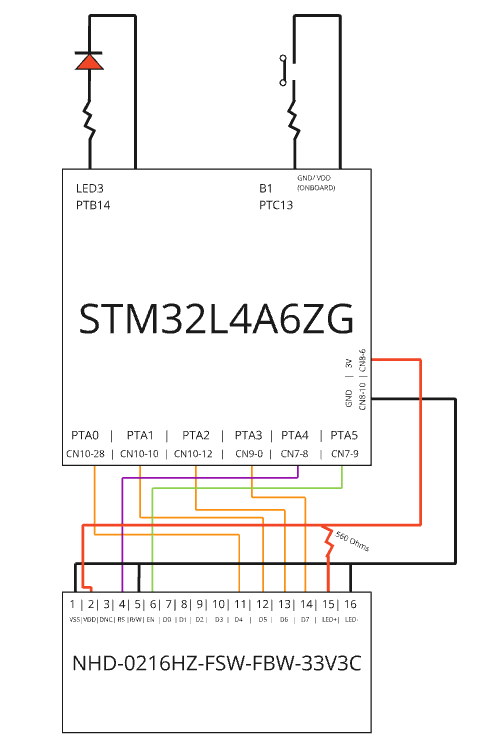
|  |  |
| --- | --- |
| **MCO CYCLES PER ISR** | **ISR TIME** |
| 55 Cycles per ISR | 13.538 uS |

Table A4.c(a): CCR1 Smallest Value

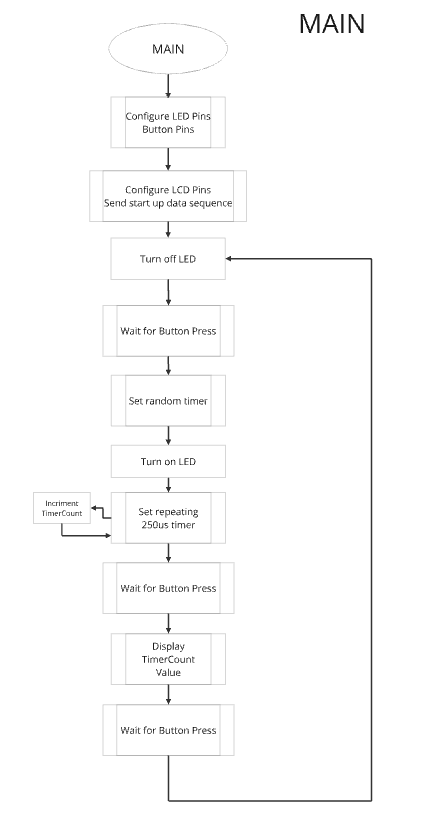
|  |  |
| --- | --- |
| **SMALLEST CC1 COUNT** | **SMALLEST ISR TIME** |
| 65 Cycles per ISR | 13.597 uS |

\*smallest CC1 count is at/close to cycles per ISR count\*

Figure A4(a): OnBoard LED/Button and 16x2 LCD Wiring

****

**Psuedocode Flow Chart:**

****

**Formatted Source Code part B main.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.h

\* project : EE329 Lab A4

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/9/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for C and stm32 headers/hal

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef \_\_MAIN\_H

#define \_\_MAIN\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

/\* Created defines and function prototypes -----------------------------------\*/

#define PERIOD (0xFFFFFFFF);

#define DUTY\_CYCLE (399);

void Led\_Config(void);

void setup\_TIM2( int iDutyCycle );

void TIM2\_IRQHandler(void);

void setup\_MCO\_CLK(void);

/\* Includes ------------------------------------------------------------------\*/

#include "stm32l4xx\_hal.h"

/\* Exported functions prototypes ---------------------------------------------\*/

void SystemClock\_Config(void);

void Error\_Handler(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Formatted Source Code part B main.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* project : EE329 Lab A4

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/9/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Device uses exclusively interupts to produce 5KHz 50%DC from internal Tim2

\* at 4MHz. Uses GPIOC pin 0 to output 5KHz 50%DC wave, GPIOC Pin 1 to signal

\* entering and leaving ISR (every 400 cycles of 4MHz), and GPIOA pin 8 to

\* match 4MHz clock.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "main.h"

int main(void)

{

//Initialize clock and delay configs

HAL\_Init();

SystemClock\_Config();

Led\_Config();

setup\_MCO\_CLK();

setup\_TIM2 (DUTY\_CYCLE); //400 ticks for each flip

// infinite loop to avoid break - program done in ISRs

while (1){;}

}

//-------------------------- ISR -----------------------------------------

void TIM2\_IRQHandler(void) {

GPIOC->BSRR |= (GPIO\_PIN\_1); //toggle in ISR

if (TIM2->SR & TIM\_SR\_CC1IF) { // triggered by CCR1 event ...

TIM2->SR &= ~(TIM\_SR\_CC1IF); // manage the flag

GPIOC->ODR ^= (GPIO\_PIN\_0); // Flip GPIO each 400 ticks

TIM2->CCR1 = TIM2->CNT + DUTY\_CYCLE // reset counter

}

if (TIM2->SR & TIM\_SR\_UIF) { // triggered by ARR event ...

TIM2->SR &= ~(TIM\_SR\_UIF); // manage the flag

}

GPIOC->BRR |= (GPIO\_PIN\_1); //toggle out ISR

}

//---------------------------------------------------------------------------

void setup\_TIM2( int iDutyCycle ) {

RCC->APB1ENR1 |= RCC\_APB1ENR1\_TIM2EN; // enable clock for TIM2

TIM2->DIER |= (TIM\_DIER\_CC1IE | TIM\_DIER\_UIE); // enable event gen, rcv CCR1

TIM2->ARR = PERIOD; // ARR = T = counts @4MHz

TIM2->CCR1 = iDutyCycle; // ticks for duty cycle

TIM2->SR &= ~(TIM\_SR\_CC1IF | TIM\_SR\_UIF); // clr IRQ flag in status reg

NVIC->ISER[0] |= (1 << (TIM2\_IRQn & 0x1F)); // set NVIC interrupt: 0x1F

\_\_enable\_irq(); // global IRQ enable

TIM2->CR1 |= TIM\_CR1\_CEN; // start TIM2 CR1

}

void setup\_MCO\_CLK(void) {

// Enable MCO, select MSI (4 MHz source)

RCC->CFGR = ((RCC->CFGR & ~(RCC\_CFGR\_MCOSEL)) | (RCC\_CFGR\_MCOSEL\_0));

// Configure MCO output on PA8

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOAEN);

GPIOA->MODER &= ~(GPIO\_MODER\_MODE8); // clear MODER bits

GPIOA->MODER |= (GPIO\_MODER\_MODE8\_1); // set alternate function mode

GPIOA->OTYPER &= ~(GPIO\_OTYPER\_OT8); // Push-pull output

GPIOA->PUPDR &= ~(GPIO\_PUPDR\_PUPD8); // no resistor

GPIOA->OSPEEDR |= (GPIO\_OSPEEDR\_OSPEED8); // high speed

GPIOA->AFR[1] &= ~(GPIO\_AFRH\_AFSEL8); // select MCO function

}

//---------------------------------------------------------------------------

void Led\_Config(void)

{ // 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);

GPIOC->MODER |= (GPIO\_MODER\_MODE0\_0 | GPIO\_MODER\_MODE1\_0);

GPIOC->OTYPER &= ~(GPIO\_OTYPER\_OT0 | GPIO\_OTYPER\_OT1);

GPIOC->PUPDR &= ~(GPIO\_PUPDR\_PUPD0 | GPIO\_PUPDR\_PUPD1);

GPIOC->OSPEEDR |= ((3 << GPIO\_OSPEEDR\_OSPEED0\_Pos) | (3 << GPIO\_OSPEEDR\_OSPEED1\_Pos));

GPIOC->BRR |= (GPIO\_PIN\_0 | GPIO\_PIN\_1);

}

// System

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();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

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();

}

/\*\* 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\_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

**Formatted Source Code part D main.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.h

\* project : EE329 Lab A4

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/9/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for C and stm32 headers/hal

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef \_\_MAIN\_H

#define \_\_MAIN\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

/\* Created defines and function prototypes -----------------------------------\*/

#define BUTTON\_PORT GPIOC

#define LED\_PORT GPIOB

#define BUTTON\_PRESS ((((~(GPIOC->IDR)) & GPIO\_PIN\_13) >> 13))

void Button\_Config(void);

void Button\_Press(void);

void Write\_Time(int timeTaken);

void Led\_Config(void);

void setup\_TIM2( int iDutyCycle );

void set\_TIM2( int period );

void TIM2\_IRQHandler(void);

void setup\_MCO\_CLK(void);

void setup\_RNG(void);

/\* Includes ------------------------------------------------------------------\*/

#include "stm32l4xx\_hal.h"

/\* Exported functions prototypes ---------------------------------------------\*/

void SystemClock\_Config(void);

void Error\_Handler(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Formatted Source Code part D main.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* project : EE329 Lab A4

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/9/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Device uses interrupts to act as a reaction timer game. User is prompted to

\* press button, and then press again once the LED lights after a randomly

\* generated set of seconds. Once pressed, interrupts are used to monitor

\* reaction speed.

\* LED defined at GPIOB PTB14 (on board LED3 RED)

\* Button defined at GPIOC PTC13 (on board B1)

\* LCD defined at GPIOA PTA0-3 (4 bit data), PTA4-5 (RS and EN)

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "main.h"

#include "lcd.h"

#include "delay.h"

uint32\_t usReaction = 0;

uint8\_t myflag = 0;

int main(void)

{

//Initialize clock and delay/LCD configs

HAL\_Init();

SystemClock\_Config();

SysTick\_Init();

Led\_Config();

Button\_Config();

LCD\_init();

setup\_RNG();

setup\_TIM2(0xFFFFFFFF);

//-------------------------- LCD VARIABLES -----

//Key Points/openings on LCD

uint8\_t origin[2] = {0,0};

uint8\_t line2[2] = {0,1};

uint8\_t openingLine1[16] = "EE 329 A4 REACT ";

uint8\_t openingLine2[16] = "PUSH SW TO TRIG ";

uint8\_t openingLine3[16] = " Ready? ";

uint8\_t openingLine4[16] = " Too Fast Dude ";

uint32\_t timeTaken = 0;

uint8\_t breakFlag;

//----------------------- MAIN EXECUTION -----

while (1){

//Opening Sequence

myflag = 0;

breakFlag = 0;

LED\_PORT->BRR |= (GPIO\_PIN\_14);

LCD\_set\_cursor(origin);

LCD\_write\_string(openingLine1);

LCD\_set\_cursor(line2);

LCD\_write\_string(openingLine2);

//Button Pushed, get ready

Button\_Press();

LCD\_set\_cursor(line2);

LCD\_write\_string(openingLine3);

//Set random timer

set\_TIM2 (((RNG->DR)% 0x1312D00) + 0x7A1200); //Set timer for random 0-8Seconds

while (myflag == 0){

if(!BUTTON\_PRESS){ //if pressed too early

breakFlag = 1;

LCD\_set\_cursor(line2);

LCD\_write\_string(openingLine4);

while(!BUTTON\_PRESS); //once released break and skip rest of code

break;

}

}

if(!breakFlag){ //end sequence only if button wasn't early

//Random Time Finished

LED\_PORT->BSRR |= (GPIO\_PIN\_14); //Light On

usReaction = 0;

set\_TIM2 (999); //Set timer for counting

while (BUTTON\_PRESS) if(usReaction > 10000000) break; //wait for 10S or keypress

timeTaken = usReaction; //log reaction time

while (!BUTTON\_PRESS); //wait for button release

Write\_Time(timeTaken);

}

//Wait for button to replay

Button\_Press();

}

}

//---------------------------------------------------------------------------

void Write\_Time(int timeTaken){

//Takes in time to press button in uS and parses it to data, then writes

//results onto LCD screen

//Parse total time into small times

uint8\_t Sec = timeTaken/1000000;

uint8\_t MSecHun = timeTaken/100000 - 10\*Sec;

uint8\_t MSecTen = timeTaken/10000 - 100\*Sec - 10\*MSecHun;

uint8\_t MSecOne = timeTaken/1000 - 1000\*Sec - 100\*MSecHun - 10\*MSecTen;

//Write Screen

uint8\_t line2[2] = {0,1};

uint8\_t endLine1[16] = "TIME = N.MMM s ";

LCD\_set\_cursor(line2);

LCD\_write\_string(endLine1);

//Write Data

uint8\_t clockSec[2] = {7,1};

uint8\_t clockMSecHun[2] = {9,1};

uint8\_t clockMSecTen[2] = {10,1};

uint8\_t clockMSecOne[2] = {11,1};

uint8\_t clockEnd[2] = {15,1};

LCD\_set\_cursor(clockSec);

LCD\_write\_char(Sec + 0x30);

LCD\_set\_cursor(clockMSecHun);

LCD\_write\_char(MSecHun + 0x30);

LCD\_set\_cursor(clockMSecTen);

LCD\_write\_char(MSecTen + 0x30);

LCD\_set\_cursor(clockMSecOne);

LCD\_write\_char(MSecOne + 0x30);

LCD\_set\_cursor(clockEnd);

LCD\_write\_char(' ');

uint8\_t endLine2[16] = " Too Slow Dude ";

if(Sec>=10){

LCD\_set\_cursor(line2);

LCD\_write\_string(endLine2);

}

}

//-------------------------- ISR -----------------------------------------

void TIM2\_IRQHandler(void) {

if (TIM2->SR & TIM\_SR\_CC1IF) { // triggered by CCR1 event ...

TIM2->SR &= ~(TIM\_SR\_CC1IF); // manage the flag

//myflag = 1; // Flip GPIO each 400 ticks

//usReaction = usReaction + 250;

//TIM2->EGR |= TIM\_EGR\_UG;

}

if (TIM2->SR & TIM\_SR\_UIF) { // triggered by ARR event ...

TIM2->SR &= ~(TIM\_SR\_UIF); // manage the flag

usReaction = usReaction + 250; //incriment count

myflag = 1; //set flag for main

myflag = 1;

}

}

//---------------------------------------------------------------------------

void setup\_RNG(void) {

RCC->AHB2ENR |= RCC\_AHB2ENR\_RNGEN; // enable clock for RNG

RCC->CRRCR |= (RCC\_CRRCR\_HSI48ON);

RNG->CR |= (RNG\_CR\_RNGEN | RNG\_CR\_IE);

RNG->SR |= (RNG\_SR\_DRDY); // enable RNG

}

void set\_TIM2( int period ) {

RCC->APB1ENR1 |= RCC\_APB1ENR1\_TIM2EN; // enable clock for TIM2

TIM2->DIER |= (TIM\_DIER\_CC1IE | TIM\_DIER\_UIE); // enable event gen, rcv CCR1

TIM2->ARR = period; // ticks for ARR Period

TIM2->SR &= ~(TIM\_SR\_CC1IF | TIM\_SR\_UIF); // clr IRQ flag in status reg

NVIC->ISER[0] |= (1 << (TIM2\_IRQn & 0x1F)); // set NVIC interrupt: 0x1F

\_\_enable\_irq(); // global IRQ enable

TIM2->CR1 |= TIM\_CR1\_CEN; // start TIM2 CR1

}

void setup\_TIM2( int period ) {

RCC->APB1ENR1 |= RCC\_APB1ENR1\_TIM2EN; // enable clock for TIM2

TIM2->DIER |= (TIM\_DIER\_CC1IE | TIM\_DIER\_UIE); // enable event gen, rcv CCR1

TIM2->ARR = period;

//PERIOD; // ARR = T = counts @4MHz

//TIM2->CCR1 = iDutyCycle; // ticks for duty cycle

TIM2->SR &= ~(TIM\_SR\_CC1IF | TIM\_SR\_UIF); // clr IRQ flag in status reg

NVIC->ISER[0] |= (1 << (TIM2\_IRQn & 0x1F)); // set NVIC interrupt: 0x1F

\_\_enable\_irq(); // global IRQ enable

TIM2->CR1 |= TIM\_CR1\_CEN; // start TIM2 CR1

}

/\*

void setup\_MCO\_CLK(void) {

// Enable MCO, select MSI (4 MHz source)

RCC->CFGR = ((RCC->CFGR & ~(RCC\_CFGR\_MCOSEL)) | (RCC\_CFGR\_MCOSEL\_0));

// Configure MCO output on PA8

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOAEN);

GPIOA->MODER &= ~(GPIO\_MODER\_MODE8); // clear MODER bits

GPIOA->MODER |= (GPIO\_MODER\_MODE8\_1); // set alternate function mode

GPIOA->OTYPER &= ~(GPIO\_OTYPER\_OT8); // Push-pull output

GPIOA->PUPDR &= ~(GPIO\_PUPDR\_PUPD8); // no resistor

GPIOA->OSPEEDR |= (GPIO\_OSPEEDR\_OSPEED8); // high speed

GPIOA->AFR[1] &= ~(GPIO\_AFRH\_AFSEL8); // select MCO function

}

\*/

//--------------------------- GPIO -----------------------------------------

void Button\_Press(void){

while (BUTTON\_PRESS)delay\_us(100);

delay\_us(100);

while (!BUTTON\_PRESS)delay\_us(100);

delay\_us(100);

}

void Button\_Config(void)

{ // configure GPIO pin PC13 for:

// Button pin initialize - Input, on board PU

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOCEN);

BUTTON\_PORT->MODER &= ~(GPIO\_MODER\_MODE13);

BUTTON\_PORT->PUPDR &= ~(GPIO\_PUPDR\_PUPD13);

}

void Led\_Config(void)

{ // configure GPIO pin PB14 for:

// output mode, push-pull, no pull up or pull down, high speed

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOBEN);

LED\_PORT->MODER &= ~(GPIO\_MODER\_MODE14);

LED\_PORT->MODER |= (GPIO\_MODER\_MODE14\_0);

LED\_PORT->OTYPER &= ~(GPIO\_OTYPER\_OT14);

LED\_PORT->PUPDR &= ~(GPIO\_PUPDR\_PUPD14);

LED\_PORT->OSPEEDR |= (3 << GPIO\_OSPEEDR\_OSPEED14\_Pos);

LED\_PORT->BRR |= (GPIO\_PIN\_14);

}

//--------------------------------- SYSTEM -----------------------------------

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();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

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();

}

/\*\* 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\_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

**Formatted Source Code delay.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : delay.h

\* project : EE329 Lab A3

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/1/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for delay.h

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#ifndef** INC\_DELAY\_H\_

**#define** INC\_DELAY\_H\_

**#include** "stm32l4xx\_hal.h"

**void** **SysTick\_Init**(**void**);

**void** **delay\_us**(**const** uint32\_t time\_us);

**#endif** /\* INC\_DELAY\_H\_ \*/

**Formatted Source Code delay.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : delay.c

\* project : EE329 Lab A3

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/1/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Functions for using SysTick clock for software delays. Provided

\* on behalf of EE329 lab manual.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#include** "delay.h"

// --------------------------------------------------- delay.c w/o #includes ---

// configure SysTick timer for use with delay\_us().

// warning: breaks HAL\_delay() by disabling interrupts for shorter delay timing.

**void** **SysTick\_Init**(**void**) {

SysTick->CTRL |= (SysTick\_CTRL\_ENABLE\_Msk | // enable SysTick Timer

SysTick\_CTRL\_CLKSOURCE\_Msk); // select CPU clock

SysTick->CTRL &= ~(SysTick\_CTRL\_TICKINT\_Msk); // disable interrupt

}

// delay in microseconds using SysTick timer to count CPU clock cycles

// do not call with 0 : error, maximum delay.

// careful calling with small nums : results in longer delays than specified:

// e.g. @4MHz, delay\_us(1) = 10=15 us delay.

**void** **delay\_us**(**const** uint32\_t time\_us) {

// set the counts for the specified delay

SysTick->LOAD = (uint32\_t)((time\_us \* (SystemCoreClock / 1000000)) - 1);

SysTick->VAL = 0; // clear timer count

SysTick->CTRL &= ~(SysTick\_CTRL\_COUNTFLAG\_Msk); // clear count flag

**while** (!(SysTick->CTRL & SysTick\_CTRL\_COUNTFLAG\_Msk)); // wait for flag

}

**Formatted Source Code lcd.h:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : lcd.h

\* project : EE329 Lab A3

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/1/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main header for defines for lcd.h

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#ifndef** INC\_LCD\_H\_

**#define** INC\_LCD\_H\_

**#include** "delay.h"

**#include** "stm32l4xx\_hal.h"

**#define** LCD\_MODER (0x03FFF)

**#define** LCD\_MODER\_0 (0x01555)

**#define** LCD\_OTYPER (0x07F)

**#define** LCD\_PUPDR (0x03FFF)

**#define** LCD\_OSPEEDR (0x03FFF)

**#define** LCD\_PORT GPIOA

**#define** LCD\_EN GPIO\_PIN\_5 //Pin 5

**#define** LCD\_RS GPIO\_PIN\_4 //Pin 4

**#define** LCD\_DATA\_BITS (GPIO\_PIN\_0 | GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3)

//Pins 11-14

//pull down R/W (set as only outputs)

**void** **LCD\_init**( **void** );

**void** **LCD\_pulse\_ENA**( **void** );

**void** **LCD\_4b\_command**( uint8\_t command );

**void** **LCD\_command**( uint8\_t command );

**void** **LCD\_write\_char**( uint8\_t letter );

**void** **LCD\_set\_cursor**( uint8\_t position[2]);

**void** **LCD\_write\_string**( uint8\_t writeData[] );

**#endif** /\* INC\_LCD\_H\_ \*/

**Formatted Source Code lcd.c:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : lcd.c

\* project : EE329 Lab A3

\* author : Wyatt Tack (wwt) - wtack@calpoly.edu

\* date : 10/1/2024

\* firmware : ST-Link V1

\* @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Functions for interfacing and communicating to LCD display through

\* nibble mode. Provided on behalf of EE329 lab manual. Configured to

\* be wired through GPIO PORT A Pin0-3 as D4-D7; Pin4 as RS; Pin5 as EN

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#include** "lcd.h"

// ------------------------------------------------------ excerpt from lcd.c ---

**void** **LCD\_init**( **void** ) {

//Port clock initialize

RCC->AHB2ENR |= (RCC\_AHB2ENR\_GPIOAEN);

//LCD pin initialize - Push Pull, no PU/PD, high speed

LCD\_PORT->MODER &= ~(LCD\_MODER);

LCD\_PORT->MODER |= (LCD\_MODER\_0);

LCD\_PORT->OTYPER &= ~(LCD\_OTYPER);

LCD\_PORT->PUPDR &= ~(LCD\_PUPDR);

LCD\_PORT->OSPEEDR |= (LCD\_OSPEEDR);

delay\_us( 80000 ); // power-up wait 80 ms

LCD\_PORT->ODR &= ~(LCD\_RS); // clear RS bit

**for** ( **int** idx = 0; idx < 3; idx++ ) { // wake up 1,2,3: DATA = 0011 XXXX

LCD\_4b\_command( 0x30 );// HI 4b of 8b cmd, low nibble = X

delay\_us( 5000 );

}

LCD\_4b\_command( 0x20 ); // fcn set #4: 4b cmd set 4b mode - next 0x28:2-line

delay\_us(3000);

LCD\_command( 0x28 ); // fcn set 4b mode and 2x28 line

delay\_us( 300 );

LCD\_command( 0x10 ); //Set cursor

delay\_us( 300 );

LCD\_command( 0x0F ); // display, cursor, blink on

delay\_us( 300 );

LCD\_command(0x06); //Entry mode

delay\_us( 300 );

LCD\_command(0x01); //clear display

delay\_us( 300 );

LCD\_command(0x80); // set cursor home

delay\_us( 300 );

}

**void** **LCD\_pulse\_ENA**( **void** ) {

// ENAble line sends command on falling edge

// set to restore default then clear to trigger

LCD\_PORT->ODR |= ( LCD\_EN ); // ENABLE = HI

delay\_us( 25 ); // TDDR > 320 ns

LCD\_PORT->ODR &= ~( LCD\_EN ); // ENABLE = LOW

delay\_us( 20 ); // low values flakey, see A3:p.1

}

**void** **LCD\_4b\_command**( uint8\_t command ) {

// LCD command using high nibble only - used for 'wake-up' 0x30 commands

LCD\_PORT->ODR &= ~( LCD\_DATA\_BITS ); // clear DATA bits

LCD\_PORT->ODR |= ( command >> 4 ); // DATA = command

delay\_us( 15 );

LCD\_pulse\_ENA( );

}

**void** **LCD\_command**( uint8\_t command ) {

// send command to LCD in 4-bit instruction mode

// HIGH nibble then LOW nibble, timing sensitive

LCD\_PORT->ODR &= ~( LCD\_DATA\_BITS ); // isolate cmd bits

LCD\_PORT->ODR |= ( (command>>4) & LCD\_DATA\_BITS ); // HIGH shifted low

delay\_us( 15 );

LCD\_pulse\_ENA( ); // latch HIGH NIBBLE

LCD\_PORT->ODR &= ~( LCD\_DATA\_BITS ); // isolate cmd bits

LCD\_PORT->ODR |= ( command & LCD\_DATA\_BITS ); // LOW nibble

delay\_us( 15 );

LCD\_pulse\_ENA( ); // latch LOW NIBBLE

}

**void** **LCD\_write\_char**( uint8\_t letter ) {

// calls LCD\_command() w/char data; assumes all ctrl bits set LO in LCD\_init()

LCD\_PORT->ODR |= (LCD\_RS); // RS = HI for data to address

delay\_us( 15 );

LCD\_command( letter ); // character to print

LCD\_PORT->ODR &= ~(LCD\_RS); // RS = LO

}

**void** **LCD\_set\_cursor**( uint8\_t position[2]) {

// calls LCD\_command to change cursor position

//position formatted as [row,col] (zero indexed)

//sets ddram address for cursor set

LCD\_PORT->ODR &= ~(LCD\_RS);

uint8\_t ddRamAdd = (40 \* position[1]);

ddRamAdd |= (0x80);

LCD\_command(ddRamAdd);

delay\_us( 500 );

**for** (**int** col = 0; col < position[0]; col++){

LCD\_command( 0x14 );

delay\_us( 300 );

}

//set address as RS\_Low, data as [0x80 | address]

//address defined as 0x00-0x0F, 0x40-0x4F

}

**void** **LCD\_write\_string**( uint8\_t writeData[]) {

// calls LCD\_write\_char in row long for loop

**for** (uint8\_t indexCol = 0; indexCol < 16; indexCol++ ){

LCD\_write\_char(writeData[indexCol]);

delay\_us(60);

}

}