   Wyatt Tack

Jaden Tran  
   EE 329-01 F'24

Group

   2024-Oct-11

**EE 329 A5**

This code is designed to use the SPI alternate function registers to connect and write to the SPI peripheral. Many chips are designed to communicate with fast speeds, with low connection counts inter-chip, so a standard protocol is needed. SPI uses a clock and one or more data lines to write to and into a chip, while selecting that chip with another wire. We used SPI to write to a digital analog converter (DAC), which only needed 3 connections: Select, clock, and data. Using this connection, we have an accurate controllable DAC that we are able to write to, and thus measure it’s analog performance. The code provided takes in a keypad input and outputs data to the DAC through the SPI register, thus outputting the voltage typed in in terms of 10s of millivolts selection (if 123 is input, 1.23 V will be output).

**Link to YouTube Presentation:**

<https://youtu.be/ifJxHws0IVM>

**Calculations:**

Equation A4.a(a): DAC Performace Calculations:

**Offset** = 1.136mV

**INL** = 3.00V – 2.9906V = 9.4mV

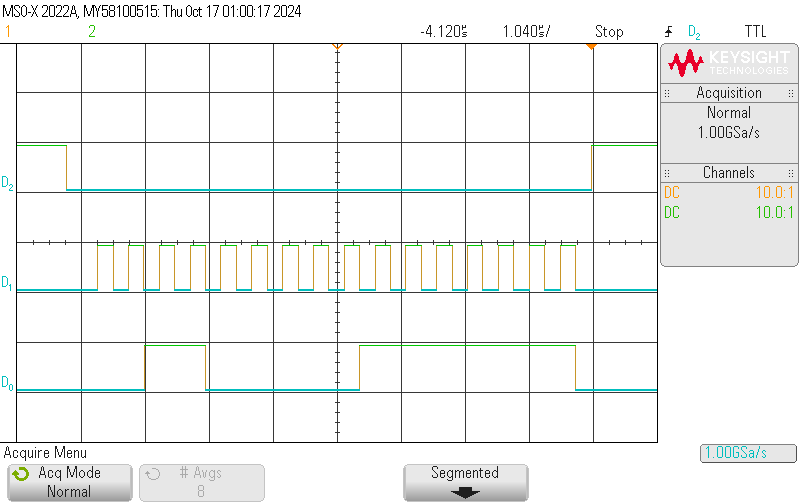
|  |  |  |  |
| --- | --- | --- | --- |
| Hex Value | Expected change = 0.5 = 2.048/4096 |  |  |
| 0x26c | 0.30934 |  |  |
| 0x26d | 0.30988 |  |  |
| **DNL:** | 0.54 | mV |  |
|  | 1.08 | lsb |  |
| 0x4d8 | 0.61831 |  |  |
| 0x4d9 | 0.61882 |  |  |
| **DNL:** | 0.51 | mV |  |
|  | 1.02 | lsb |  |
| 0x9b1 | 1.2384 |  |  |
| 0x9b2 | 1.2389 |  |  |
| **DNL:** | 0.5 | mV |  |
|  | 1 | lsb |  |

**Obtained Data:**

Figure A5.a(a): DAC Uncalibrated Output Trend

**Captures:**

Figure A5(b): DAC SPI Transmission

****

Data = 3x Byte = 0x07F

Config Byte

1 = active

1 = 1x gain

unused

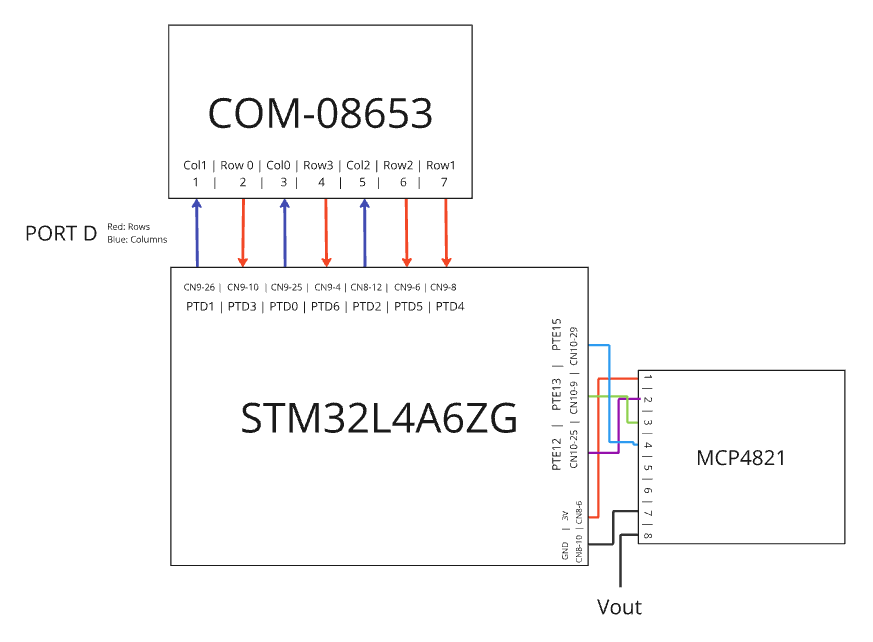
0 = write

MOSI

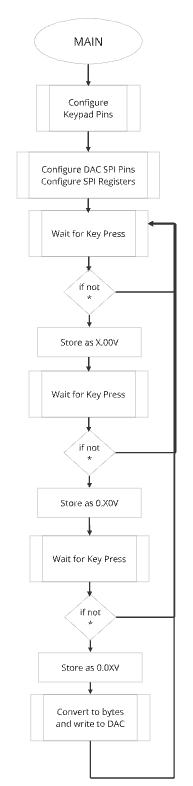
CLK

CS

Figure A5(c): DAC and Keypad Wiring Diagram

****

**Psuedocode Flow Chart:**

****

**Formatted Source Code main.h:**

/\*\*

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

\* @file : main.h

\* project : EE329 Lab A5

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

\* date : 10/12/2024

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

void DAC\_init(void);

void DAC\_write(uint16\_t mvolts);

uint16\_t DAC\_volt\_conv(uint16\_t mvolts);

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

#include "stm32l4xx\_hal.h"

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

void SystemClock\_Config(void);

void Error\_Handler(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Formatted Source Code main.c:**

/\*\*

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

\* @file : main.c

\* project : EE329 Lab A5

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

\* date : 10/12/2024

\* 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 "spi.h"

#include "delay.h"

#include "keypad.h"

int main(void)

{

//Initialize clock, keypad, spi config

HAL\_Init();

SystemClock\_Config();

SysTick\_Init();

Keypad\_Config();

DAC\_init();

uint16\_t mvolt, volt, mvolt100, mvolt10;

while (1){

//poll for mv digits or reset

volt = Key\_Poll();

if (volt == 0xA) continue;

mvolt100 = Key\_Poll();

if (mvolt100 == 0xA) continue;

mvolt10 = Key\_Poll();

if (mvolt10 == 0xA) continue;

//set up total mV count and write

mvolt = (volt\*1000) + (mvolt100\*100) + (mvolt10\*10);

DAC\_write(mvolt);

}

}

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

void DAC\_init(void){

SPI\_init();

}

void DAC\_write(uint16\_t mvolts){

uint16\_t data = DAC\_volt\_conv(mvolts);

SPI1->DR = data;

while (!(SPI1->SR & SPI\_SR\_TXE));

while (SPI1->SR & SPI\_SR\_BSY);

}

uint16\_t DAC\_volt\_conv(uint16\_t mvolts){

//determines if mvolts requested is above/below 1/2 threshold

mvolts = mvolts\*10000/9981 + 1; //trendline from calibration

if(mvolts < 2048){

uint16\_t data = (mvolts \* 4095)/2048;

return (data | (0x3 << 12));

}

else if(mvolts <= 3300 ){

uint16\_t data = (mvolts \* 4095)/(2\*2048);

return (data | (0x1 << 12));

}

else{

return (0x1FFF);

}

//uint16\_t data = (mvolts \* 4095)/3300;

//return (data | (0x3 << 12));

}

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

// 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 SPI.h:**

/\*\*

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

\* @file : spi.h

\* project : EE329 Lab A4

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

\* date : 10/13/2024

\* firmware : ST-Link V1

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

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

\*

\* main header for defines for spi.h

\*

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

\*/

#ifndef INC\_SPI\_H\_

#define INC\_SPI\_H\_

#include "stm32l4xx\_hal.h"

#define SPIPORT GPIOE

#define SPI\_RCC RCC\_AHB2ENR\_GPIOEEN

#define SPI\_MODER (GPIO\_MODER\_MODE12 | GPIO\_MODER\_MODE13 | GPIO\_MODER\_MODE14 | GPIO\_MODER\_MODE15)

#define SPI\_MODER\_1 (GPIO\_MODER\_MODE12\_1 | GPIO\_MODER\_MODE13\_1 | GPIO\_MODER\_MODE14\_1 | GPIO\_MODER\_MODE15\_1)

#define SPI\_OTYPER (GPIO\_OTYPER\_OT12 | GPIO\_OTYPER\_OT13 | GPIO\_OTYPER\_OT14 | GPIO\_OTYPER\_OT15)

#define SPI\_PUPDR (GPIO\_PUPDR\_PUPD12 | GPIO\_PUPDR\_PUPD13 | GPIO\_PUPDR\_PUPD14 | GPIO\_PUPDR\_PUPD15)

#define SPI\_OSPEEDR ((3 << GPIO\_OSPEEDR\_OSPEED12\_Pos) | (3 << GPIO\_OSPEEDR\_OSPEED13\_Pos)| (3 << GPIO\_OSPEEDR\_OSPEED14\_Pos) | (3 << GPIO\_OSPEEDR\_OSPEED15\_Pos))

#define SPI\_AFRCLEAR ((0x000F << GPIO\_AFRH\_AFSEL12\_Pos) | (0x000F << GPIO\_AFRH\_AFSEL13\_Pos)| (0x000F << GPIO\_AFRH\_AFSEL14\_Pos) | (0x000F << GPIO\_AFRH\_AFSEL15\_Pos))

#define SPI\_AFRSET ((0x0005 << GPIO\_AFRH\_AFSEL12\_Pos) | (0x0005 << GPIO\_AFRH\_AFSEL13\_Pos) | (0x0005 << GPIO\_AFRH\_AFSEL14\_Pos) | (0x0005 << GPIO\_AFRH\_AFSEL15\_Pos))

void SPI\_init( void );

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

**Formatted Source Code SPI.c:**

/\*\*

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

\* @file : spi.c

\* project : EE329 Lab A4

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

\* date : 10/13/2024

\* firmware : ST-Link V1

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

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

\*

\* Functions for SPI module, set up as SPI1 through GPIOA.

\* PTE-15 -> MOSI

\* PTE-14 -> MISO

\* PTE-13 -> SCK

\* PTE-12 -> NSS

\*

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

\*/

#include "spi.h"

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

void SPI\_init( void ) {

// setup for SPI at GPIOA and SPI1, adapted from EE329 Lab Manual

// enable clock for GPIOA & SPI1

RCC->AHB2ENR |= (SPI\_RCC); // GPIOA: DAC NSS/SCK/SDO

RCC->APB2ENR |= (RCC\_APB2ENR\_SPI1EN); // SPI1 port

// configure AF select, push pull, no pu/pd/ fast mode

SPIPORT->MODER &= ~(SPI\_MODER);

SPIPORT->MODER |= (SPI\_MODER\_1);

//SPIPORT->OTYPER &= ~(SPI\_OTYPER);

//SPIPORT->PUPDR &= ~(SPI\_PUPDR);

SPIPORT->OSPEEDR |= (SPI\_OSPEEDR);

// configure AFR for SPI1 function (clear nibble, set 5)

SPIPORT->AFR[1] &= ~(SPI\_AFRCLEAR);

SPIPORT->AFR[1] |= (SPI\_AFRSET);

// SPI config as specified @ STM32L4 RM0351 rev.9 p.1459

// build control registers CR1 & CR2 for SPI control of peripheral DAC

// assumes no active SPI xmits & no recv data in process (BSY=0)

// CR1 (reset value = 0x0000)

SPI1->CR1 &= ~( SPI\_CR1\_SPE ); // disable SPI for config

SPI1->CR1 &= ~( SPI\_CR1\_RXONLY ); // recv-only OFF

SPI1->CR1 &= ~( SPI\_CR1\_LSBFIRST ); // data bit order MSb:LSb

SPI1->CR1 &= ~( SPI\_CR1\_CPOL | SPI\_CR1\_CPHA ); // SCLK polarity:phase = 0:0

SPI1->CR1 |= SPI\_CR1\_MSTR; // MCU is SPI controller

// CR2 (reset value = 0x0700 : 8b data)

SPI1->CR2 &= ~( SPI\_CR2\_TXEIE | SPI\_CR2\_RXNEIE ); // disable FIFO intrpts

SPI1->CR2 &= ~( SPI\_CR2\_FRF); // Moto frame format

SPI1->CR2 |= SPI\_CR2\_NSSP; // auto-generate NSS pulse

SPI1->CR2 |= SPI\_CR2\_DS; // 16-bit data

SPI1->CR2 |= SPI\_CR2\_SSOE; // enable SS output

// CR1

SPI1->CR1 |= SPI\_CR1\_SPE; // re-enable SPI for ops

}

**Formatted Source Code keypad.h:**

/\*\*

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

\* @file : keypad.h

\* project : EE329 Lab A3

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

\* date : 9/27/2024

\* firmware : ST-Link V1

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

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

\*

\* main header for defines for keypad.h

\*

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

\*/

#ifndef INC\_KEYPAD\_H\_

#define INC\_KEYPAD\_H\_

#include "stm32l4xx\_hal.h"

//-------------------------------- KEYPAD Defines ------------------------------

#define COL\_PORT GPIOD

#define COL\_PINS (GPIO\_PIN\_0 | GPIO\_PIN\_1 | GPIO\_PIN\_2)

#define ROW\_PORT GPIOD

#define ROW\_PINS (GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_6)

#define SETTLE 1900 //defined as time for 20ms loop instruction (\*5.5us)

#define BIT0\_COL GPIO\_PIN\_0 //defined as first bit for columns

#define BIT0\_ROW GPIO\_PIN\_3 //defined as first bit for rows

#define NUM\_COLS 3

#define NUM\_ROWS 4

#define NO\_KEYPRESS 0x0

#define KEY\_ZERO 11

#define CODE\_ZERO 0xF

//--------------------------------- function prototypes ------------------------

void Keypad\_Config(void);

int Keypad\_IsAnyKeyPressed(void);

int Keypad\_WhichKeyIsPressed(void);

uint8\_t Key\_Poll(void);

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

**Formatted Source Code keypad.c:**

/\*\*

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

\* @file : keypad.c

\* project : EE329 Lab A3

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

\* date : 10/7/2024

\* firmware : ST-Link V1

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

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

\*

\* Keypad pulling function source file provided on behalf of the EE329 lab

\* manual. Adapted from EE329 lab manual. Currently attached to GPIO PORT D

\*

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

\*/

#include "keypad.h"

// ------------------------------------- modified from excerpt from keypad.c ---

void Keypad\_Config(void){//must be manually changed if separate GPIO port is used

//set for port D as current config

//Port clock initialize

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

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

COL\_PORT->MODER &= ~(GPIO\_MODER\_MODE0 | GPIO\_MODER\_MODE1

| GPIO\_MODER\_MODE2);

COL\_PORT->MODER |= (GPIO\_MODER\_MODE0\_0 | GPIO\_MODER\_MODE1\_0

| GPIO\_MODER\_MODE2\_0);

COL\_PORT->OTYPER &= ~(GPIO\_OTYPER\_OT0 | GPIO\_OTYPER\_OT1

| GPIO\_OTYPER\_OT2 | GPIO\_OTYPER\_OT3);

COL\_PORT->PUPDR &= ~(GPIO\_PUPDR\_PUPD0 | GPIO\_PUPDR\_PUPD1

| GPIO\_PUPDR\_PUPD2 | GPIO\_PUPDR\_PUPD3);

COL\_PORT->OSPEEDR |= ((GPIO\_OSPEEDR\_OSPEED0) |

(GPIO\_OSPEEDR\_OSPEED1) |

(GPIO\_OSPEEDR\_OSPEED2));

//Row pin initialize - Input, pull down

ROW\_PORT->MODER &= ~(GPIO\_MODER\_MODE3 | GPIO\_MODER\_MODE4

| GPIO\_MODER\_MODE5 | GPIO\_MODER\_MODE6);

ROW\_PORT->PUPDR &= ~(GPIO\_PUPDR\_PUPD3 | GPIO\_PUPDR\_PUPD4

| GPIO\_PUPDR\_PUPD5 | GPIO\_PUPDR\_PUPD6);

ROW\_PORT->PUPDR |= (GPIO\_PUPDR\_PUPD3\_1 | GPIO\_PUPDR\_PUPD4\_1

| GPIO\_PUPDR\_PUPD5\_1 | GPIO\_PUPDR\_PUPD6\_1);

}

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

int Keypad\_IsAnyKeyPressed(void) {

// drive all COLUMNS HI; see if any ROWS are HI

// return true if a key is pressed, false if not

COL\_PORT->BSRR = COL\_PINS; // set all columns HI

for ( uint16\_t idx=0; idx<SETTLE; idx++ ) // let it settle

;

if ((ROW\_PORT->IDR & ROW\_PINS) != 0 ) { // got a keypress!

for ( uint16\_t idx=0; idx < SETTLE; idx++ ){

if ((ROW\_PORT->IDR & ROW\_PINS) == 0 ) return( 0 );

} // if key held for 20ms then return 1 (debounce)

return( 1 );

}

else

return( 0 ); // nope.

}

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

int Keypad\_WhichKeyIsPressed(void) {

// detect and encode a pressed key at {row,col}

// assumes a previous call to Keypad\_IsAnyKeyPressed() returned TRUE

// verifies the Keypad\_IsAnyKeyPressed() result (no debounce here),

// determines which key is pressed and returns the encoded key ID

int8\_t iRow=0, iCol=0, iKey=0; // keypad row & col index, key ID result

int8\_t bGotKey = 0; // bool for keypress, 0 = no press

COL\_PORT->BSRR = COL\_PINS; // set all columns HI

for ( iRow = 0; iRow < NUM\_ROWS; iRow++ ) { // check all ROWS

if ( ROW\_PORT->IDR & (BIT0\_ROW << iRow) ) { // keypress in iRow!!

COL\_PORT->BRR = ( COL\_PINS ); // set all cols LO

for ( iCol = 0; iCol < NUM\_COLS; iCol++ ) { // 1 col at a time

COL\_PORT->BSRR = ( BIT0\_COL << (iCol) ); // set this col HI

if ( ROW\_PORT->IDR & (BIT0\_ROW << iRow) ) { // keypress in iCol!!

bGotKey = 1;

break; // exit for iCol loop

}

}

if ( bGotKey )

break;

}

}

// encode {iRow,iCol} into LED word : row 1-3 : numeric, ‘1’-’9’

// row 4 : ‘\*’=10, ‘0’=15, ‘#’=12

// no press: send NO\_KEYPRESS

if ( bGotKey ) {

iKey = ( iRow \* NUM\_COLS ) + iCol + 1; // handle numeric keys ...

if ( iKey == KEY\_ZERO ) // works for ‘\*’, ‘#’ too

iKey = CODE\_ZERO;

return( iKey ); // return encoded keypress

}

return( NO\_KEYPRESS ); // unable to verify keypress

}

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

uint8\_t Key\_Poll(void) {

// polls key and returns value once key is inputed and let go

uint8\_t currentKeyValue;

while(!Keypad\_IsAnyKeyPressed());

if (Keypad\_WhichKeyIsPressed() > 0 && Keypad\_WhichKeyIsPressed() < 16){

currentKeyValue = Keypad\_WhichKeyIsPressed();

if (currentKeyValue == 0xF ) currentKeyValue = 0; //zero position

}

else

currentKeyValue = (0xF); //error flag

while(Keypad\_IsAnyKeyPressed());

return currentKeyValue;

//wait till key is let go

}