Kelvin Shi, Brayden Daly

Lab A5

EE 329-05

2025 May 15

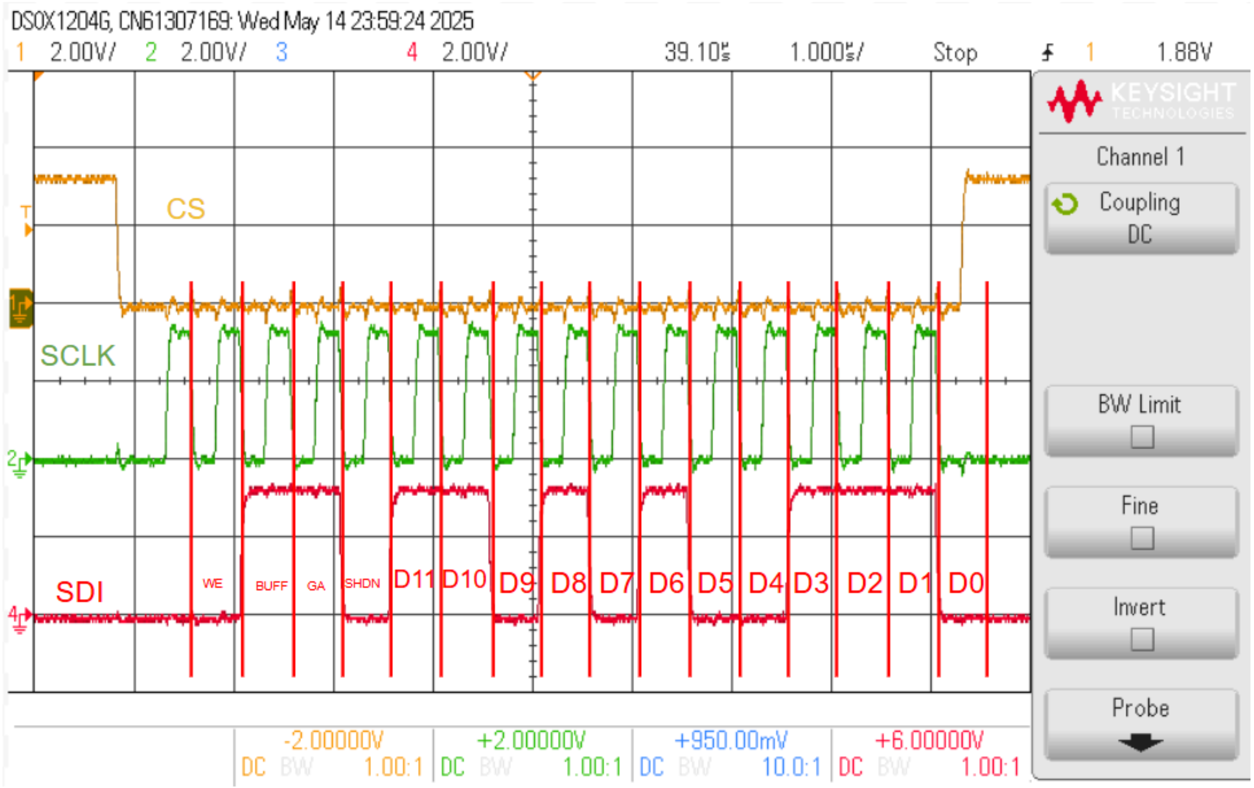
A5 - SPI Digital to Analog Converter

Introduction:

In this lab, we programmed the STM32 to output an analog voltage specified by the voltage we typed into the Keypad. When we press the ‘*’ key, we also reset the program so we can enter a new value. Our code worked as expected and our DAC outputted the expected voltage with +/- 1 mV precision.

YouTube demo: https://youtu.be/99yqVaKjOoM?si=oid_6PWGlq095TLo

Annotated Logic Analyzer Image:



WE → Write enable (0 means we are writing to DAC)

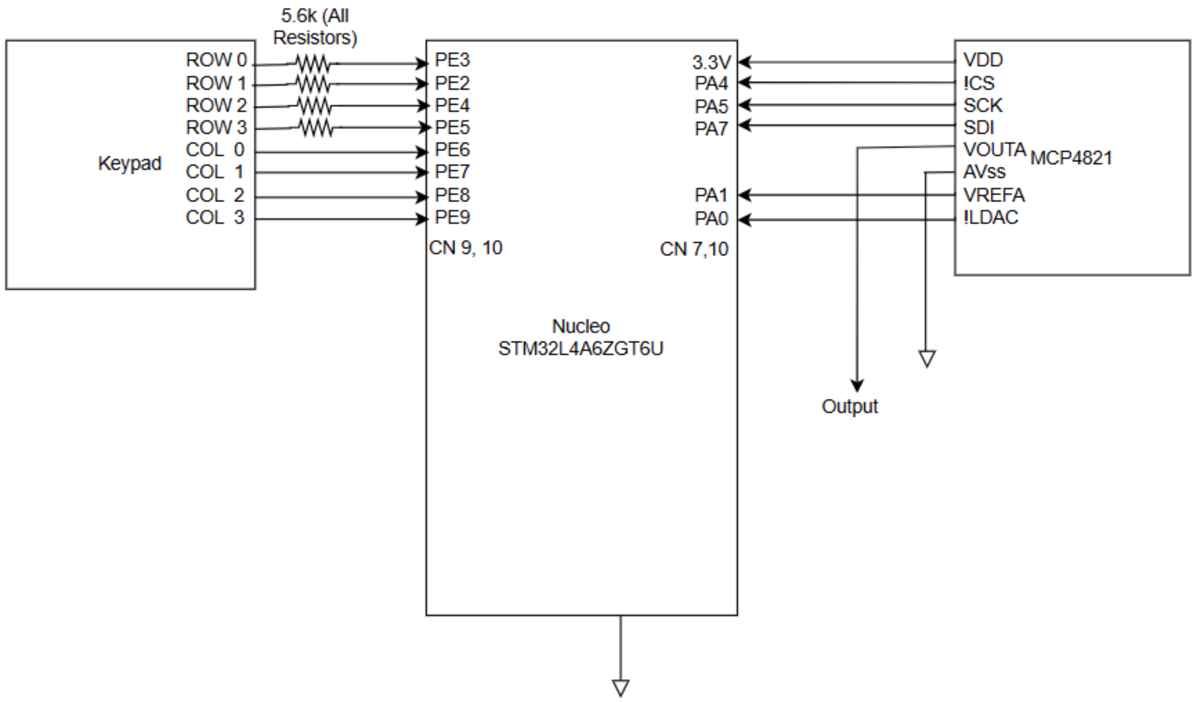
BUFF → Buffer Bit

GA → Gain value for passing 2.048V threshold of DAC

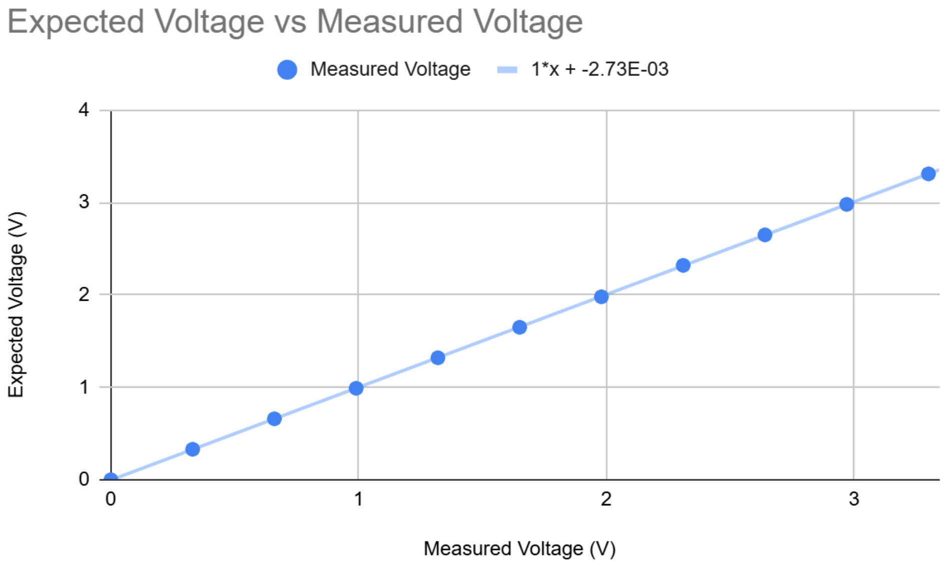
SHDN → Shutdown bit

D0 - D11 → Data Bits 0 - 11 sent to DAC

Wiring Diagram:



DAC Calibration:



Expected Voltage	Measured Voltage	Digital Output	Difference	DNL
0	0	0	0	0
0.33	0.33	409.5	0.33	0
0.66	0.66	819	0.33	0
0.99	0.99	1228.5	0.33	0
1.32	1.32	1638	0.33	0
1.65	1.65	2047.5	0.33	0
1.98	1.98	2457	0.33	0.01
2.31	2.32	2866.5	0.34	0
2.64	2.65	3276	0.33	0
2.97	2.98	3685.5	0.33	0
3.3	3.31	4095	0.33	0
		INL		0.01

We did not need to calibrate since our output voltages were accurate to our input values.

<u>MCU-DAC Performance</u>

Digital Output	Measured	DNL (V)	LSB
D0x000	1 mV		

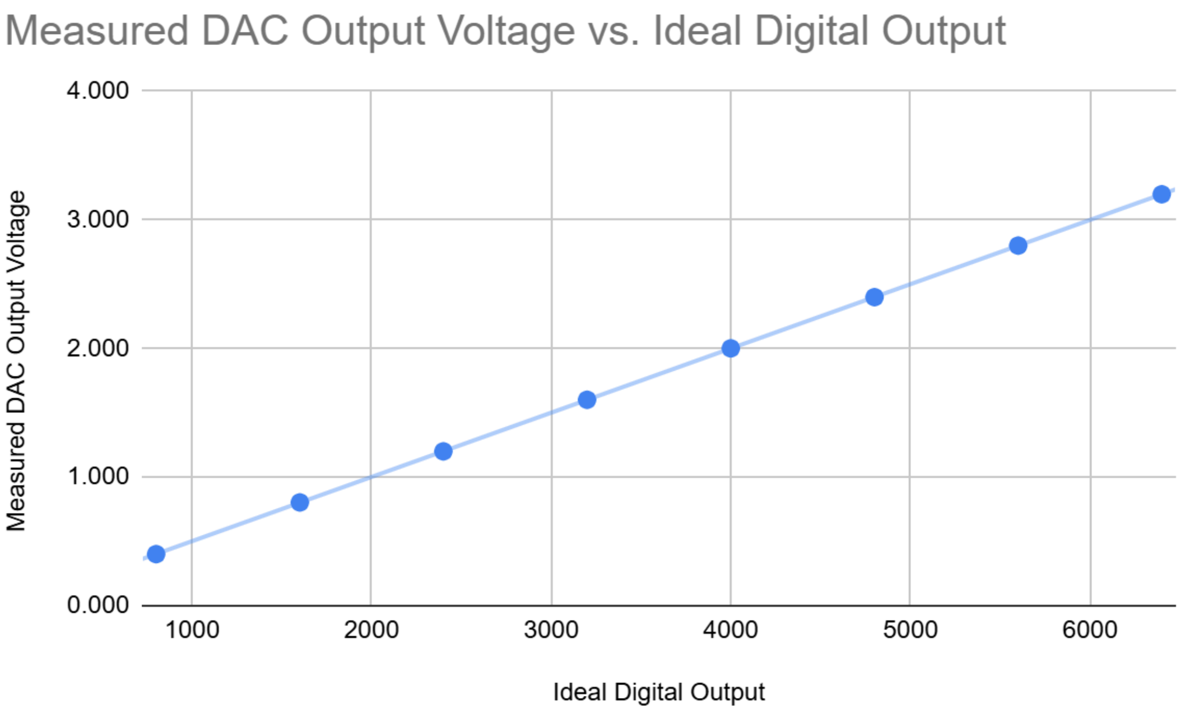
D0xBB7	3.01 V		
		delta [V]	
0x26C	0.312	0.001	0.1
0x26D	0.313		
0x4D8	0.624	0.001	0.1
0x4D9	0.625		
0x9B1	1.247	0.001	0.1
0x9B2	1.248		

Using the following equation, the ideal digital DAC word is calculated. This is compared to the actual measured value of the DAC.

$$D_N = \frac{V_{out} D_{max}}{V_{REF}}$$

Ideal Digital Output:	Ideal DAC Output Voltage:	Measured DAC Output Voltage:
800	0.4	0.401
1600	0.8	0.803
2399	1.2	1.200
3199	1.6	1.601
3999	2	2.001
4799	2.4	2.399
5599	2.8	2.800
6398	3.2	3.199

NOTE: Since the DAC changes from a gain of 1 to 2 when the input voltage is greater than the reference voltage, our table does show values that exceed the ideal DAC 12 bit values



Appendix

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

```
/*
*******************************************************************************
* EE 329 A5 MAIN SUPPORT
*******************************************************************************
* @file           : main.h
* @brief          : main support functions and definitions
* project         : EE 329 S'25 - Assignment A5
* authors         : Kelvin Shi - kshi04@calpoly.edu
* version         : 1.0
* date            : 2025/05/15
* compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
*******************************************************************************
**/
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
```

```
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32l4xx_hal.h"
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
uint16_t getUserInput(void);
void SystemClock_Config(void);
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

```
-------------------------------------------------------------------------
main.c
-------------------------------------------------------------------------
```

```
/******************************************************************************
* EE 329 A5 USER DEFINED DAC
*******************************************************************************
* @file           : main.c
* @brief          : Covnert the user input to an actual voltage via the DAC
* project         : EE 329 S'25 - Assignment A5
* authors         : Kelvin Shi - kshi04@calpoly.edu
* version         : 1.0
* date            : 2025/05/15
* compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
******************************************************************************
*/
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "DAC.h"
#include "keypad.h"
/* -----------------------------------------------------------------------------
* function : int main(void)
* INs      : void
* OUTs     : int - status value
* action   : The entry point of the program
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
int main(void){
 // Reset of all peripherals, Initializes the Flash interface and the Systick.
 HAL_Init();
```

```
    /* Configure the system clock */
    SystemClock_Config();
    DAC_init();
    keypad_Config();
    uint16_t value;
    /* Infinite loop */
    while (1){
        value = getUserInput();
        DAC_write(DAC_volt_conv(value));
        DAC_update();
    }
}
/* -----------------------------------------------------------------------------
* function : uint16_t getUserInput(void)
* INs      : void
* OUTs     : uint16_t - returns the user input from a keypad to a 16 bit word
* action   : Enable the SPI peripheral and configure it for communication
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
uint16_t getUserInput(){
    uint8_t first, second, third;
    // get first valid digit
    do{
        first = keypad_getInput();
        if (first == 0xE){
            // restart process
            return getUserInput();
        }
    }
    while(first > 0x9);
    // get second valid digit
    do{
        second = keypad_getInput();
        if (second == 0xE){
            // restart process
            return getUserInput();
        }
    }
    while(second > 0x9);
    // get third valid digit
    do{
        third = keypad_getInput();
        if (third == 0xE){
            // restart process
            return getUserInput();
        }
    }
    while(third > 0x9);
    // computations of voltage in mV
    float voltage;
    voltage = first*1000 + second*100 + third*10;
    if (voltage > VOLTAGE_RAIL){
        voltage = VOLTAGE_RAIL;
    }
    return voltage;
}
```

----

main.h

```
    /*
    *******************************************************************************
    * EE 329 A5 SPI/DAC SUPPORT
    *******************************************************************************
    * @file           : DAC.h
    * @brief          : SPI/DAC support functions and definitions
    * project         : EE 329 S'25 - Assignment A5
    * authors         : Kelvin Shi - kshi04@calpoly.edu
    * version         : 1.0
    * date            : 2025/05/15
    * compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
    * target          : NUCLEO-L4A6ZG
    * clocks          : 4 MHz MSI to AHB2
    * @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
    *******************************************************************************
    **/
    #ifndef INC_SPI_H_
    #define INC_SPI_H_
    #endif /* INC_SPI_H_ */
    #include "stm32l4xx_hal.h"
    #define VOLTAGE_RAIL 3300
    #define CONTROL_BITS_G1 0x3000
    #define CONTROL_BITS_G2 0x1000
    #define VOLTAGE_REF 2048
    void SPI_GPIO_setup(void);
    void SPI_init(void);
    void DAC_init(void);
    uint16_t DAC_volt_conv(uint16_t voltage);
    void DAC_GPIO_setup(void);
    void DAC_update(void);
    void DAC_write(uint16_t data);
```

DAC.c

```
------------------------------------------------------------------------------------------------
/*******************************************************************************
* EE 329 A5 DAC/SPI FUNCTIONS
*******************************************************************************
* @file           : DAC.c
* @brief          : SPI and DAC configuration
* project         : EE 329 S'25 - Assignment A5
* authors         : Kelvin Shi - kshi04@calpoly.edu
* version         : 1.0
* date            : 2025/05/15
* compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
*******************************************************************************
*/
#include "DAC.h"
/* -----------------------------------------------------------------------------
* function : void DAC_init(void)
* INs      : none
* OUTs     : void
* action   : Initialize the SPI and DAC GPIO pins; wake up DAC device
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
void DAC_init(){
   SPI_GPIO_setup();
   DAC_GPIO_setup();
   SPI_init();
   GPIOA->BSRR = (GPIO_PIN_1 | GPIO_PIN_0); // no shutdown
}
/* -----------------------------------------------------------------------------
* function : uint16_t DAC_volt_conv(uint16_t)
* INs      : uint16_t - the voltage to send to the DAC in mV
* OUTs     : uint16_t - the 12 bit word to be sent to the DAC
* action   : Convert the user defined input voltage to the correct level for
*            the DAC to process
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
uint16_t DAC_volt_conv(uint16_t voltage){
   // cap the maximum voltage to the voltage rail
   if (voltage > VOLTAGE_RAIL){
      voltage = VOLTAGE_RAIL;
   }
   if (voltage < 0){
        voltage = 0;
    }
    // calculate the correct step
    float step = ((float) VOLTAGE_REF)/4095;
    return voltage/step;
}
/* -----------------------------------------------------------------------------
* function : void DAC_write(uint16_t)
* INs      : uint16_t - the 12 bit word to send to the DAC
* OUTs     : void
* action   : Using the SPI, send over the 12 bit data to the DAC and adjust for
*           the gain/reference voltage threshold
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
void DAC_write(uint16_t data){
    GPIOA->BSRR = GPIO_PIN_0;
    uint16_t command;
    // check if the data is over the internal reference voltage
    if (data > 4095U){
        // adjust for double gain
        command = CONTROL_BITS_G2;
        data /= 2;
    }
    else
        command = CONTROL_BITS_G1;
    // only mask the last 12 bits
    command |= (data & 0x0FFF);
    // ensure that the transmission buffer is cleared before sending
    while (!(SPI1->SR & 0x02));
    SPI1->DR = command;
}
/* -----------------------------------------------------------------------------
* function : void SPI_init(void)
* INs      : void
* OUTs     : void
* action   : Enable the SPI peripheral and configure it for communication
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
void SPI_init( void ) {
  // SPI config as specified @ STM32L4 RM0351 rev.9 p.1459
  // called by or with DAC_init()
  // 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 (MOSI/MISO active)
  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 enable
  SPI1->CR1 |=   SPI_CR1_SPE;                 // re-enable SPI for ops
}
/* ----------------------------------------------------------------------------
* function : void SPI_GPIO_setup(void)
* INs      : void
* OUTs     : void
* action   : Configure the GPIOA ports for SPI alternate function use.
*           NOTE: does not use POCI since DAC is simplex communication
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/05/15
* -------------------------------------------------------------------------- */
void SPI_GPIO_setup(){
   // enable clock for GPIOA & SPI1
   RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);          // GPIOA: DAC NSS/SCK/SDO
   RCC->APB2ENR |= (RCC_APB2ENR_SPI1EN);           // SPI1 port
   // Configure SPI GPIO pins as AF mode, push-pull, and high speed
   GPIOA->MODER &= ~(GPIO_MODER_MODE4 | GPIO_MODER_MODE5 | GPIO_MODER_MODE7);
   GPIOA->MODER |= ((2 << 8) | (2 << 10) | (2 << 14));
   GPIOA->OTYPER &= ~(GPIO_OTYPER_OT4 | GPIO_OTYPER_OT5 | GPIO_OTYPER_OT7);
   GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED4_Pos | GPIO_OSPEEDR_OSPEED5_Pos
          | GPIO_OSPEEDR_OSPEED7_Pos);
   GPIOA->OSPEEDR |= ((3 << GPIO_OSPEEDR_OSPEED4_Pos)
          | (3 << GPIO_OSPEEDR_OSPEED5_Pos) | (3 << GPIO_OSPEEDR_OSPEED7_Pos));
   // SPI NSS AF configuration
   GPIOA->AFR[0] &= ~((0x000F << GPIO_AFRL_AFSEL4_Pos));
   GPIOA->AFR[0] |=  ((0x0005 << GPIO_AFRL_AFSEL4_Pos));
   // SPI SCK AF configuration
   GPIOA->AFR[0] &= ~((0x000F << GPIO_AFRL_AFSEL5_Pos));
   GPIOA->AFR[0] |=  ((0x0005 << GPIO_AFRL_AFSEL5_Pos));
   // SPI MOSI AF configuration
   GPIOA->AFR[0] &= ~((0x000F << GPIO_AFRL_AFSEL7_Pos));
   GPIOA->AFR[0] |=  ((0x0005 << GPIO_AFRL_AFSEL7_Pos));
}
/* ----------------------------------------------------------------------------
* function : void DAC_GPIO_setup(void)
* INs       : void
* OUTs      : void
* action    : Configures the GPIP pins that control the DAC's "SHUTDOWN" and
*            "LDAC"
* authors   : Kelvin Shi - kshi04@calpoly.edu
* version   : 1.0
* date      : 2025/05/15
* -------------------------------------------------------------------------- */
void DAC_GPIO_setup(){
   // Configure GPIO has output mode, push-pull, no PUPD, high speed
   GPIOA->MODER &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1);
   GPIOA->MODER |= ((1 << 0) | (1 << 2));
   GPIOA->OTYPER &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1);
   GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1);
   GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0_Pos | GPIO_OSPEEDR_OSPEED1_Pos);
   GPIOA->OSPEEDR |= ((3 << GPIO_OSPEEDR_OSPEED0_Pos)
         | (3 << GPIO_OSPEEDR_OSPEED1_Pos));
}
/* ---------------------------------------------------------------------------
* function : void DAC_update(void)
* INs       : void
* OUTs      : void
* action    : Toggles the LDAC pin to update the DAC to reflect new input data
* authors   : Kelvin Shi - kshi04@calpoly.edu
* version   : 1.0
* date      : 2025/05/15
* -------------------------------------------------------------------------- */
void DAC_update(){
   // toggle the LDAC pin with a small software delay
   GPIOA->BRR = GPIO_PIN_0;
   for (int i = 0; i < 5; i++);
   GPIOA->BSRR = GPIO_PIN_0;
}
```

----

keypad.h

```
/*
******************************************************************************
* EE 329 A2 KEYPAD SUPPORT
******************************************************************************
* @file           : keypad.h
* @brief          : keypad support functions and defintions
* project         : EE 329 S'25 - Assignment A2
* authors         : Kelvin Shi - kshi04@calpoly.edu
* version         : 1.0
* date            : 2025/04/14
* compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
********************************************************************************
**/
/* Define to prevent recursive inclusion --------------------------------------*/
#ifndef INC_KEYPAD_H_
#define INC_KEYPAD_H_
#endif /* INC_KEYPAD_H_ */
/* Includes -------------------------------------------------------------------*/
#include "stm32l4xx_hal.h"
/* Defined Variable -----------------------------------------------------------*/
#define ROW0_PIN 0x0004 //PE2
#define ROW1_PIN 0x0008 //PE3
#define ROW2_PIN 0x0010 //PE4
#define ROW3_PIN 0x0020 //PE5
#define COL1_PIN 0x0040 //PE6
#define COL2_PIN 0x0080 //PE7
#define COL3_PIN 0x0100 //PE8
#define COL4_PIN 0x0200 //PE9
#define ALL_ROWS 0x03c
#define ALL_COLS 0x3c0
#define NUM_ROWS 4
#define NUM_COLS 4
#define NO_KEYPRESS -1
#define FALSE 0
#define TRUE 1
#define NUM_ROWS 4
#define NUM_COLS 4
#define TIME_SPACE 40
/* Exported functions prototypes ---------------------------------------------*/
void keypad_Config(void);
uint8_t keypad_IsAnyKeyPressed(void);
uint8_t keypad_WhichKeyIsPressed(void);
uint8_t keypad_Debounce(void);
uint8_t keypad_getInput(void);
```

--

keypad.c

--

```
/*******************************************************************************
* EE 329 A2 KEYPAD FUNCTIONS
********************************************************************************
* @file           : keypad.c
* @brief          : keypad configuration, debouncer, and key identification
* project         : EE 329 S'25 - Assignment A2
* authors         : Kelvin Shi - kshi04@calpoly.edu
```

```c
* version         : 1.0
* date            : 2025/04/14
* compiler        : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
*******************************************************************************
* KEYPAD WIRING 4 ROWS 4 COLS (pinout NUCLEO-L4A6ZG = L496ZG)
*      peripheral – Nucleo I/O
*
* GRID LAYOUT:
*        COL1 COL2 COL3 COL4
* Row 1: "1" , "2" , "3", "A"
* Row 1: "4" , "5" , "6", "B"
* Row 1: "7" , "8" , "9", "C"
* Row 1: "*" , "0" , "#", "D"
*
* PINOUT:
* COL1 - D59 - PE6 - OUTPUT
* COL2 - D41 - PE7 - OUTPUT
* COL3 - D42 - OE8 - OUTPUT
* COL4 - D6  - PE9 - OUTPUT
* ROW1 - D60 - PE3 - INPUT - PDR
* ROW2 - D56 - PE2 - INPUT - PDR
* ROW3 - D56 - PE4 - INPUT - PDR
* ROW4 - D58 - PE5 - INPUT - PDR
*
* ADDITIONAL INFORMATION
* "*" - 0xE
* "#" - 0xF
*******************************************************************************
*/
#include "keypad.h"
/* -----------------------------------------------------------------------------
* function : void keypad_Config(void)
* INs      : none
* OUTs     : none
* action   : Initialize the STM32 ports to interface with the keypad. enables
*           GPIO E ports, set rows as the inputs, columns as the outputs,
*           high speed, push-pull outputs, and pull down resistor for inputs
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/04/23
* -------------------------------------------------------------------------- */
void keypad_Config(void) {
    RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOEEN);
    GPIOE->MODER &= ~(GPIO_MODER_MODE2 | GPIO_MODER_MODE3
           | GPIO_MODER_MODE4 | GPIO_MODER_MODE5);
```

```
   GPIOE->MODER &= ~(GPIO_MODER_MODE6 | GPIO_MODER_MODE7
          | GPIO_MODER_MODE8 | GPIO_MODER_MODE9);
   GPIOE->MODER |= (GPIO_MODER_MODE6_0 | GPIO_MODER_MODE7_0
          | GPIO_MODER_MODE8_0 | GPIO_MODER_MODE9_0);
   GPIOE->OTYPER &= ~(GPIO_OTYPER_OT6 | GPIO_OTYPER_OT7 | GPIO_OTYPER_OT8
          | GPIO_OTYPER_OT9);
   GPIOE->OSPEEDR |= ~((3 << GPIO_OSPEEDR_OSPEED6_Pos)
   | (3 << GPIO_OSPEEDR_OSPEED7_Pos) | (3 << GPIO_OSPEEDR_OSPEED8_Pos)
          | (3 << GPIO_OSPEEDR_OSPEED9_Pos));
   GPIOE->PUPDR &= ~(GPIO_PUPDR_PUPD6 | GPIO_PUPDR_PUPD7
          | GPIO_PUPDR_PUPD8 | GPIO_PUPDR_PUPD9);
   GPIOE->PUPDR |= ((2 << GPIO_PUPDR_PUPD2_Pos) | (2 << GPIO_PUPDR_PUPD3_Pos)
          | (2 << GPIO_PUPDR_PUPD4_Pos) | (2 << GPIO_PUPDR_PUPD5_Pos));
}
/* -----------------------------------------------------------------------------
* function : uint8_t keypad_WhichKeyIsPressed(void)
* INs      : none
* OUTs     : int - the value associated with the keypad press
* action   : verifies a keypad press is present and determines which key
*           is pressed. Calculate the value pressed and return the value
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.1
* date     : 2025/04/23
* REVISION HISTORY
* 1.1 2025/04/23 Kelvin Shi  refactored key decode to use array
* ------------------------------------------------------------------------- */
uint8_t keypad_WhichKeyIsPressed(void) {
   int8_t iRow = 0, iCol = 0; // keypad row/col index
   int8_t bGotKey = 0;                  // bool for keypress: 0 = no press
   for (iRow = 0; iRow < NUM_ROWS; iRow++) {   // check all ROWS
      if (GPIOE->IDR & (1 << (2 + iRow))) {   // keypress in iRow!
         GPIOE->BRR = (ALL_COLS);             // set all cols LOW
         for (iCol = 0; iCol < NUM_COLS; iCol++) {   // 1 col at a time
            GPIOE->BSRR = (1 << (6 + iCol));         // set this col HI
            if ( GPIOE->IDR & (1 << (2 + iRow))) {   // keypress in iCol!
               bGotKey = 1;
               break;                               // exit for iCol loop
            }
         }
         if (bGotKey)
            break;
      }
   }
   // create LUT and index based on row/col index
   if (bGotKey) {
      uint8_t lookup[16] = {1,2,3,10,4,5,6,11,7,8,9,12,14,0,15,13};
      return lookup[NUM_ROWS*iRow + iCol];
   }
   return (NO_KEYPRESS);                     // unable to verify keypress
}
/* --------------------------------------------------------------------------
* function : uint8_t keypad_IsAnyKeyPressed(void)
* INs      : none
* OUTs     : int - a status value, 1 corresponding to key press detected and
*            0 corresponding to no key press detected
* action   : check to see if there is a keypress detected. return the status
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/04/14
* ------------------------------------------------------------------------ */
uint8_t keypad_IsAnyKeyPressed(void) {
   GPIOE->BSRR = ALL_COLS;                  // set all columns HI
   if ((GPIOE->IDR & (ALL_ROWS)) != 0)         // got a keypress!
      return (TRUE);
   else
      return (FALSE);                         // nope keypress
}
/* --------------------------------------------------------------------------
* function : uint8_t keypad_Debounce(void)
* INs      : none
* OUTs     : int - a verified status value, 1 corresponding to key press detected
*            and 0 corresponding to no key press detected
* action   : verifies that keypress is actually received by performing 4 checks
*            spaced 100 us apart. return true if verified keypress; else false
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/04/14
* ------------------------------------------------------------------------ */
//will check on 4 instances spaced 100 us apart
uint8_t keypad_Debounce(void) {
   if (keypad_IsAnyKeyPressed()) {
      for (int i = 0; i < TIME_SPACE; i++)
         ;
      if (keypad_IsAnyKeyPressed()) {
         for (int i = 0; i < TIME_SPACE; i++)
            ;
         if (keypad_IsAnyKeyPressed()) {
            for (int i = 0; i < TIME_SPACE; i++)
               ;
            if (keypad_IsAnyKeyPressed())
               return TRUE;
         }
      }
   }
   return FALSE;
}
```

```c
/* ---------------------------------------------------------------------------
* function : uint8_t keypad_getInput(void)
* INs      : none
* OUTs     : uint8_t - a confirmed value of the button pressed on the keypad
* action   : block until user input from the keypad is received. return the
*           value of the keypad pressed
* authors  : Kelvin Shi - kshi04@calpoly.edu
* version  : 1.0
* date     : 2025/04/22
* ------------------------------------------------------------------------ */
uint8_t keypad_getInput(void) {
    while (1) {
        if (keypad_IsAnyKeyPressed()) {
            if (keypad_Debounce()) {
                uint8_t key = keypad_WhichKeyIsPressed();
                while (keypad_IsAnyKeyPressed());     //wait for press release
                return key;
            }
        }
    }
}
```