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:

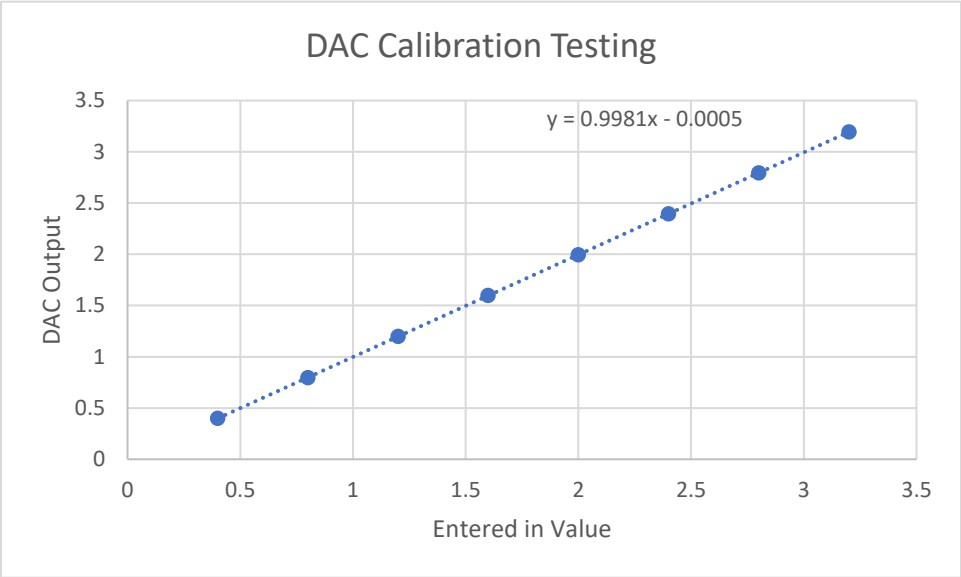
Offset = 1.136mV

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

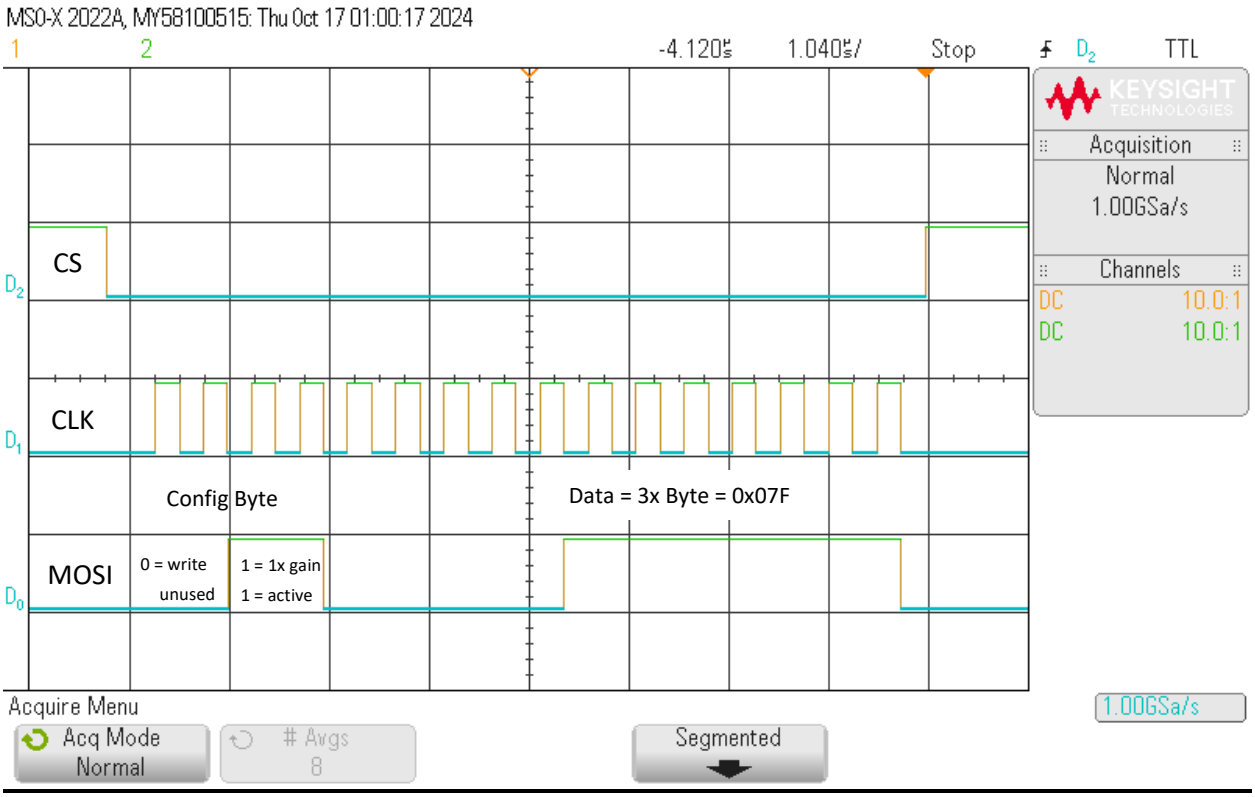
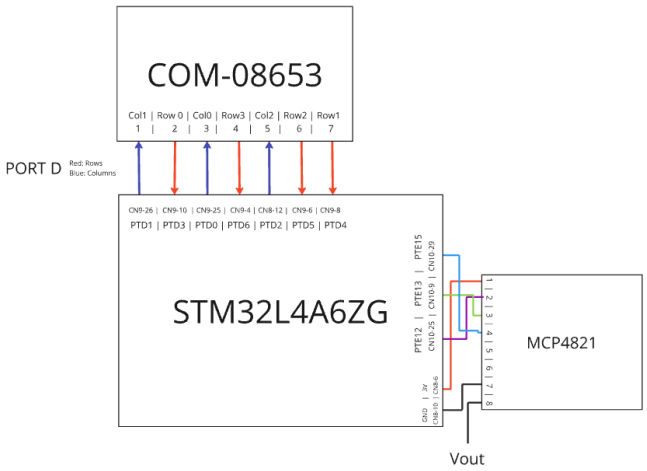
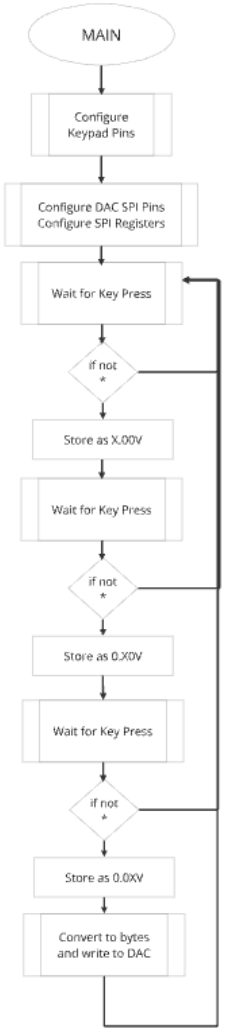


Figure A5(c): DAC and Keypad Wiring Diagram



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_ */
```

<u>Formatted Source Code keypad.c:</u>

```
/**
  ******************************************************************************
  * @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
}
```