Benchohra, Sereen
Tang, Elvis
CPE329
10/27/2021

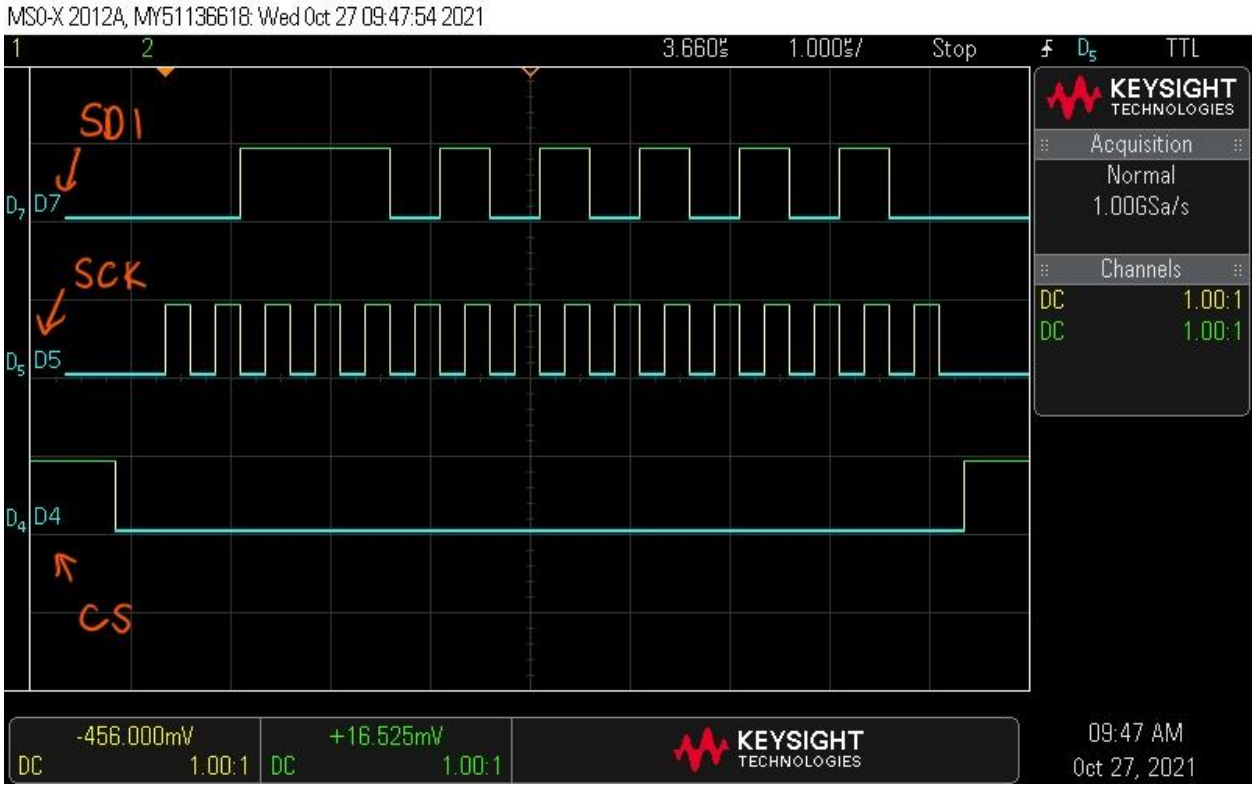
A5 - SPI/ DAC

Demo:

Demoed to TA in class

Transmission to the DAC including CS, SCLK, and SDI

The Data passed in SDI is (0x3000 | 0xAAA)




Code:

---

main.c

---

```
#include "main.h"
#include "keypad.h"
#include "DAC.h"
```

```
void SystemClock_Config(void);

#define SHIFTLEFT 10
#define MAX_VOLTS 3300

int main(void)
{

  HAL_Init();

  SystemClock_Config();

  uint8_t key;
  int keyLen, input;

  keypad_init();      // setup gpio pins for keypad
  DAC_init();     // initializes the DAC

  while(1)
  {
        input = 0;
        keyLen = 0;

        while(keyLen < 3) // loops until you have 3 digits
        {
              key = read_keypad();

              if(key != NO_KEY) //  if numerical key, append to attempt
              {
                    input = (input * SHIFTLEFT) + key; // shifts the
previous digit by a place and insers the new digit
                    while(read_keypad() != NO_KEY);
                    keyLen++;
              }
        }
        input *= SHIFTLEFT; //shifts the input into milliVolts
        // if input is greater than 3300 milliVolts then set in
        if(input > MAX_VOLTS)
              input = MAX_VOLTS;
        DAC_volt_conv(input); //converts and outputs the voltage
  }

}
```

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};


  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
      Error_Handler();
  }

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
      Error_Handler();
  }

  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
      Error_Handler();
  }
}


void Error_Handler(void)
{

  __disable_irq();
```

```
  while (1)
  {
  }

}

#ifdef  USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif /* USE_FULL_ASSERT */
```

---

Keypad.h

---

```
/*
 * keypad.h
 *
 *  Created on: Sep 29, 2021
 *      Author: Sereen
 */

#ifndef SRC_KEYPAD_H_
#define SRC_KEYPAD_H_

void keypad_init(void);
uint8_t read_keypad(void);

#define STAR   10
#define POUND  11
#define NO_KEY 0xFF

// Ports PC4 - PC7
#define ROW1 GPIO_IDR_ID4
#define ROW2 GPIO_IDR_ID5
#define ROW3 GPIO_IDR_ID6
#define ROW4 GPIO_IDR_ID7

#define ROW_PORT GPIOC
#define COL_PORT GPIOC
```

```
#define ROW_PORT_IDR (ROW_PORT->IDR)
#define COL_PORT_ODR (COL_PORT->ODR)

// change to PC 0, 1 , 2
#define COL1 GPIO_ODR_OD0
#define COL2 GPIO_ODR_OD1
#define COL3 GPIO_ODR_OD2

#define COL_MASK (COL1|COL2|COL3)
#define ROW_MASK (ROW1|ROW2|ROW3|ROW4)


#endif /* SRC_KEYPAD_H_ */
```

---

Keypad.c

---

```
#include "main.h"
#include "keypad.h"

void keypad_init(void)
{
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOBEN); // enable GPIOB clock on bus
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOCEN); // enable GPIOC clock on bus

   // clear GPIOB PA4-PA7 (also sets them for input
	ROW_PORT->MODER &= ~( GPIO_MODER_MODE4
	                    | GPIO_MODER_MODE5
	                    | GPIO_MODER_MODE6
	                    | GPIO_MODER_MODE7);
	// clear MODE PC5 - PC7  bits for keypad and use as columns
	COL_PORT->MODER &= ~(GPIO_MODER_MODE0
	                    | GPIO_MODER_MODE1
	                    | GPIO_MODER_MODE2);

	// set PC5-PC7 as outputs for columns
	COL_PORT->MODER |= ( (1 << GPIO_MODER_MODE0_Pos)
	                   | (1 << GPIO_MODER_MODE1_Pos)
	                   | (1 << GPIO_MODER_MODE2_Pos) );

	// enable pulldown resistor for rows on PA4-7
```

```
	// clear pupdr
	ROW_PORT->PUPDR &=  ~(    GPIO_PUPDR_PUPD4_1
	                        | GPIO_PUPDR_PUPD5_1
	                        | GPIO_PUPDR_PUPD6_1
	                        | GPIO_PUPDR_PUPD7_1);


	ROW_PORT->PUPDR |=  (    GPIO_PUPDR_PUPD4_1
	                        | GPIO_PUPDR_PUPD5_1
	                        | GPIO_PUPDR_PUPD6_1
	                        | GPIO_PUPDR_PUPD7_1);

	// enable push-pull for columns on PC5-PC7
	// (check later if there is problems)
	COL_PORT->OTYPER &= ~(  GPIO_OTYPER_OT0
	                        | GPIO_OTYPER_OT1
	                        | GPIO_OTYPER_OT2);


	// set slow speed for columns (PC5-PC7)

	COL_PORT->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0
	                                        | GPIO_OSPEEDR_OSPEED1
	                                        | GPIO_OSPEEDR_OSPEED2); // PA0
slow speed

	// set columns to high
	COL_PORT_ODR |= COL_MASK;

}


uint8_t calculate_key(uint16_t col, uint8_t row)
{
	uint8_t key;
	uint8_t rows;

	rows = row >> 4; // right shift rows 4 places for easiers calc

	if(rows == 4)
		rows = 3;
	if(rows == 8)
```

```
            rows = 4;
      // calculate key based on col
      switch(col)
      {
            case 0:
                  key = (3 * rows) - 2;
                  break;
            case 1:
                  key = (3 * rows) - 1;
                  break;
            case 2:
                  key = (3 * rows);
                  break;
      }


      if(key == 10)   // leave as 10 so it can output to LED (change with
LCD)
            key  = '*'; // leave it in ASCII for later use
      if(key == 11)
            key = 0;
      if(key == 12)  // leave as 12 so so it can output to LED (change with
LCD)
            key = '#';

      return key;
}

void check_columns(uint8_t cur_col)
{
      COL_PORT_ODR  &= ~(COL_MASK); // turn all colums off
      switch(cur_col)  // set a col high depending on the row
      {
            case 0:
                  COL_PORT_ODR |= COL1;
                  break;
            case 1:
                  COL_PORT_ODR |= COL2;
                  break;
            case 2:
                  COL_PORT_ODR |= COL3;
                  break;
```

```
	}
}


uint8_t read_keypad(void)
{
	uint8_t rows;
	uint8_t cur_col ;
	uint8_t key;
	// Read the rows PB4-PB7 only
	rows = ROW_PORT_IDR & ROW_MASK ;

	if(rows == 0) // check to see if all the rows are low (is so return
NO KEY)
		return NO_KEY;

	for(cur_col = 0; cur_col < 3; cur_col++)
	{

		// set current columns high others low
		check_columns(cur_col);
		// read the rows
	    rows = ROW_PORT_IDR & ROW_MASK;

		if(rows != 0)
		{
			// calculate button from row and col
			key = calculate_key(cur_col, rows);
			// set columns to high
			COL_PORT_ODR |= COL_MASK;
			return key;
		}

	}

	// set columns to high
	COL_PORT_ODR |= COL_MASK;

	return NO_KEY;


}
```

---

DAC.c

---

```
/*
 * DAC.c
 *
 *  Created on: Oct 27, 2021
 *      Author: Sereen
 */

#include "main.h"
#include "DAC.h"


void DAC_GPIO_config()
{
	// PA4 = SPI1_CS, PA5 = SPI1_SCK, PA7 = SPI1_MOSI
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);    // IO port A clock enable

	// Clear MODE bits for PA4-PA7
	DAC_PORT->MODER &= ~(GPIO_MODER_MODE4
			          | GPIO_MODER_MODE5
			          | GPIO_MODER_MODE7);

	DAC_PORT->MODER |= (GPIO_MODER_MODE4_1
			          |GPIO_MODER_MODE5_1
			          | GPIO_MODER_MODE7_1);  // alternate function
mode


	DAC_PORT->OTYPER &= ~(GPIO_OTYPER_OT4
				        | GPIO_OTYPER_OT5
				        | GPIO_OTYPER_OT7);  // Push-pull output

	DAC_PORT->PUPDR &= ~(GPIO_PUPDR_PUPD4
					   | GPIO_PUPDR_PUPD5
					   | GPIO_PUPDR_PUPD7);   // no pu/pd
resistor

	DAC_PORT->OSPEEDR |= (GPIO_OSPEEDR_OSPEED4
					    | GPIO_OSPEEDR_OSPEED5
					    | GPIO_OSPEEDR_OSPEED7);      // high
```

```
speed

	DAC_PORT->AFR[0] |= (5 << GPIO_AFRL_AFSEL4_Pos
	                    | 5 << GPIO_AFRL_AFSEL5_Pos
	                    | 5 << GPIO_AFRL_AFSEL7_Pos);
}


void DAC_init(void)
{

	DAC_GPIO_config();

	RCC->APB2ENR |= (RCC_APB2ENR_SPI1EN);     //enable SP1

	SPI1 -> CR1 = (SPI_CR1_MSTR); // enable Master bit

	SPI1 ->CR2 = ( SPI_CR2_DS  // enable 16 bit DS mode
	                    | SPI_CR2_NSSP); // NSSP mode

	SPI1 -> CR1 |= (SPI_CR1_SPE); // Enable SPI
}


void DAC_write(uint16_t data)
{
	while(!(SPI1->SR & SPI_SR_TXE));    // Check to Make Sure Buffer is
Empty
	SPI1->DR  = (SHDN | GAIN| data);        // enable HIGH and SHDN bits
	while(!(SPI1->SR & SPI_SR_RXNE));   // Wait for RXIFG to be Set
(RXBUF Empty)

}

// converts milliVolts taken in to be converted into count for DAC
void DAC_volt_conv(uint16_t mVolt)
{
    int DAC_count;
    DAC_count = (mVolt * DAC_RES)/ (VREF * 1000);
    DAC_write(DAC_count);
}
```

---

DAC.h

---

```
/*
 * DAC.h
 *
 *  Created on: Oct 27, 2021
 *      Author: Sereen
 */

#ifndef SRC_DAC_H_
#define SRC_DAC_H_

void DAC_init(void);
void DAC_write(uint16_t);
void DAC_volt_conv(uint16_t);

#define SHDN 0x2000     // Bit 12
#define GAIN 0x1000            // Bit 13
#define DAC_PORT GPIOA
#define DAC_RES 4095
#define VREF 3.3

#endif /* SRC_DAC_H_ */
```