PROJECT 2: FUNCTION GENERATOR

By Lily Goldman, Giovanni Ramirez Angel

May 9, 2022

California Polytechnic State University – San Luis Obispo

CPE 329 – 05: Microcontroller-Based Systems Design

Dr. Paul Hummel

Table of Contents:

List of Figures

List of Tables

1 Behavioral Description

The function generator will use a microcontroller, DAC, and user input from the user to develop a square waveform, sinusoidal, waveform, sawtooth(ramp) waveform, and a triangle waveform. 5 keys on the keypad can change the frequency to either 100Hz, 200Hz, 300Hz, 400Hz, or 500Hz. When the square waveform is selected, the user can increase and decrease the duty cycle by 10% all the way from 10% duty cycle to 90%. The function generator will default to a 100Hz square wave with a 50% duty cycle.

2 System Specifications

The device operates on specifications in Table 1 and the keypad interface is elaborated in Table 2.

Table 1: Function Generator specification

Input Supply Voltage (USB)	3.3V
f_{max}	500Hz
f_{min}	100Hz
Max Duty Cyle	90%
Waveforms	Square, Sawtooth, Sine and Triangle
DC Offset	1.5V
$V_{out(max)}$	3.0V
$V_{out(min)}$	0V
Resolution	119,217.9 samples/sec
Sample amount	1080 samples
DAC Resolution	12 bits

Table 2: Keypad values and their uses

Functions	*Key*	*Description*
Frequencies	1	100Hz
	2	200Hz
	3	300Hz
	4	400Hz
	5	500Hz
Waveform	6	Sinusoid
	7	Triangle
	8	Sawtooth
	9	Square
*Duty Cycle**	#	10% duty cycle increase
	*	10% duty cycle decrease
	0	Set to a 50% duty cycle

*Capability only for the square waveform

3 Resolution Calculation

A max resolution of the waveform generator can be found using the time to output a single value to the DAC. For a more accurate estimate of the max resolution, the time to enter and execute the ISR is found in the fourth row in Table 3. To find the time to enter the ISR, the amount of clock cycles was found in a previous experimentation to be 22. The equation below was used to find the values for row 2 of Table 3.

Into ISR Duration: 22 * 1/fclk

Table 3: Duration of writeDAC() function within an ISR

	4MHz	24MHz	32MHz
Into ISR	5.5us	1.375us	0.688us
Execute ISR	61us	10.16us	7.7us
Into and execute ISR	66.5us	11.54us	8.388us

Chosen frequency for function generator is 32MHz. Resolution is found by taking the inverse of the duration of the chosen frequency.

Resolution = (1sec) / (duration of Into and execute ISR)

= (1sec) / 8.388us

= **119,217.9 samples/sec**

Using the lowest frequency option for the waveforms, the amount of samples/period. A sample rate for all waveforms can then be found.

Sample Amount = (Period of 100Hz) / (1/Resolution)

= 100ms / (1/119,217.9 samples/sec)

= **1192 samples**

The sample value needs to be divisible by 5, 4, 3, and 2 so all the frequency variations can easily be implemented in the code. The value that is the closest common multiple is 1080. Each shape array will have a count of 1080 samples and will interrupt every 296 clock cycles. 296 is found from taking the difference of the 1192 sample amount and the chosen one, finding the extra duration that would cause and then adding it to 8.388us. There are 296 counts at 32MHz in 9.26us.

4 System Schematic

The function generator uses the STM32L476RG board to generate a SPI interface with the external DAC as seen in Figure 1. The keypad data is read by the board and then the serial data is written to the DAC.

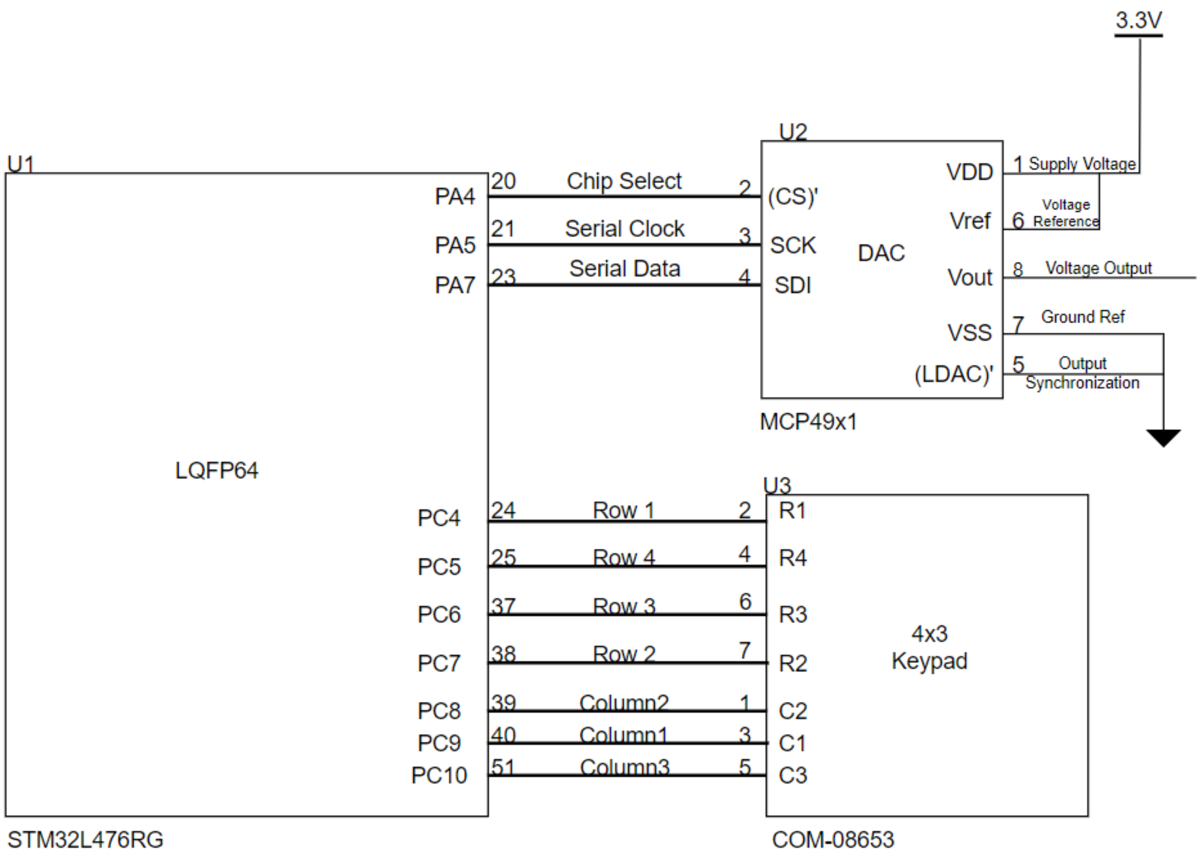

Figure 1: Function Generator System Schematic

5 Software Architecture

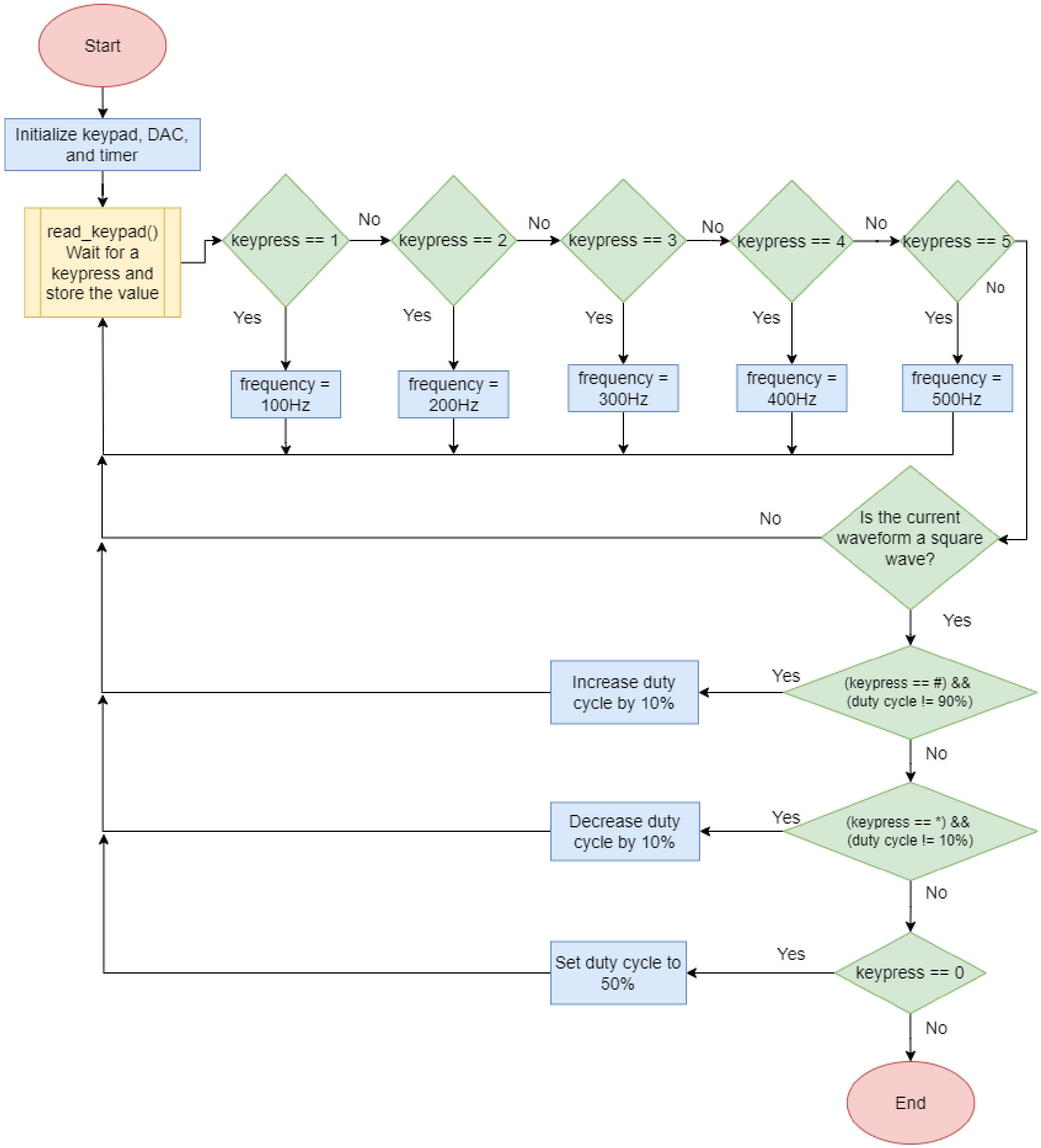

Figure 1. Main Loop Flowchart

The software is built using a series of if statements to allow for the fastest implementation of the program. An FSM was considered to organize the waveforms in their own states, but elements such as the interrupt count and the frequency value affect each waveform. These variables will be updated in the main function and then used in the ISR. This was done to reduce the amount of if statements and logic in the ISR and overall increase its speed. The main function displayed in Figure 1 is used to update the count, the frequency, and the duty cycle global variables.

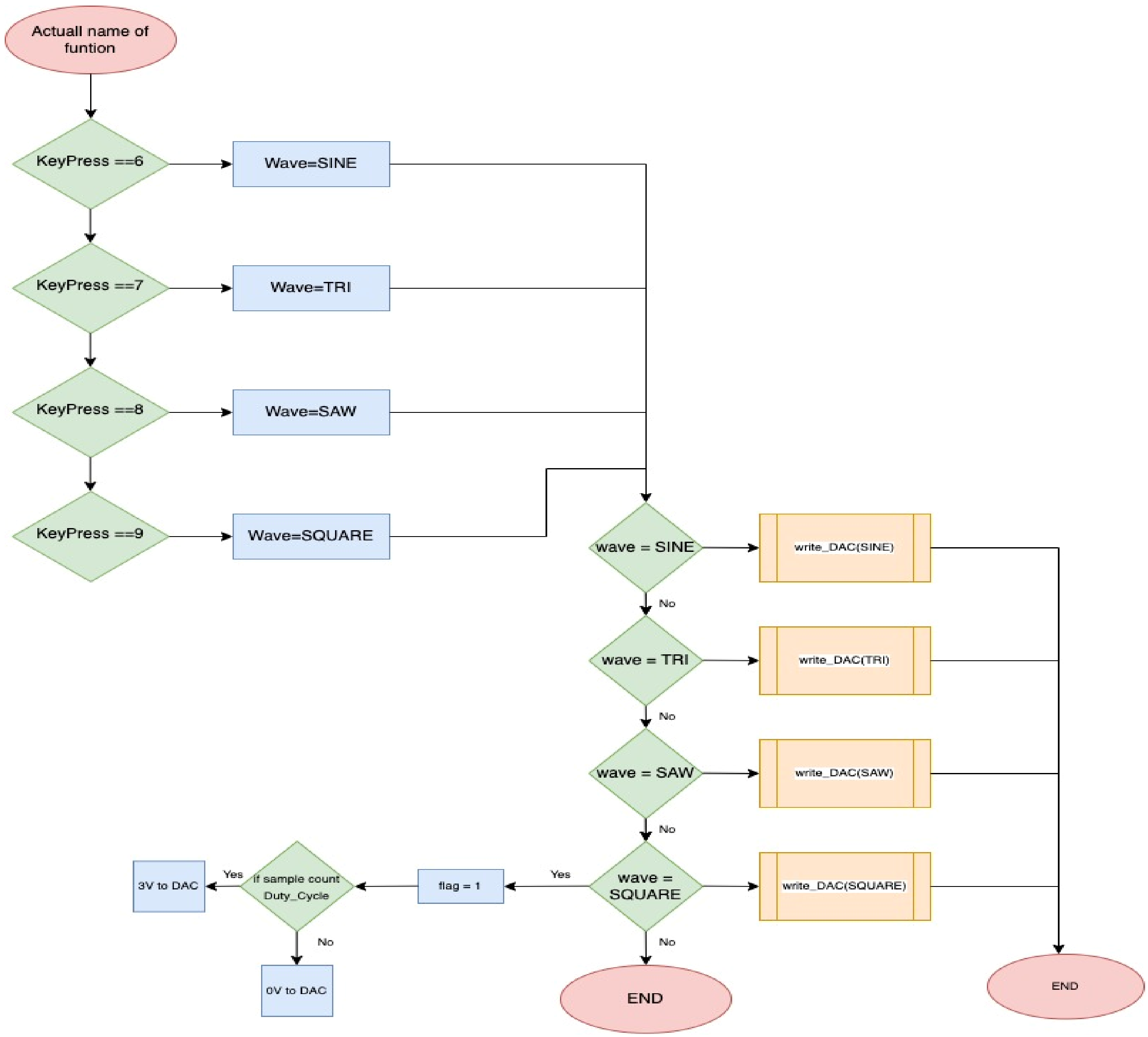

Figure 2. ISR Flowchart

Checking which wave is being selected by checking button keypress. 6 = Sine wave, 7 = Triangle, 8 = Saw wave, 9 = Square wave (seen in Figure 2). Utilized write DAC function to output selected wave form point. If Square wave also checked flag to see if sample count is the same as Duty cycle. If true writes 3V to DAC if no 0V to DAC.

6 Appendix

```
/*
*  main.c
*
*  Created on: May 7, 2022
*      Author: Lily Goldman and Giovanni Ramirez Angel
*      Brief: Emulates a wave generator. Using the external keypad, the
*      keys specify the waveform, the frequency, and the duty cycle for
*      the square wave. The waveforms are capable of sawtooth, sinusoid
*      triangle, and square. Each wave can switch between
*      frequencies of 100Hz, 200Hz, 300Hz, 400Hz, and 500Hz. Values are
*      written to the external DAC and output onto an oscilloscope.
*
*/
#include "main.h"
#include "DAC.h"
#include "keypad.h"
#include "shapes.h"

//Prototypes
void timer_init(void);
void SystemClock_Config(void);

//Global Variables
uint8_t frequency = 1;  //start the square wave at 100Hz
uint32_t CNT_tracker = 0; //internal count that the ISR increases
uint16_t keyPress = 9; //starts the display as a square wave
uint32_t Duty_Cycle = CNT_half; //50% duty cycle
uint8_t square_flag = 1; //tells the ISR to display the square wave and
allow for duty cycle changes

int main(void) {

        //initializations
        HAL_Init();
        SystemClock_Config();
        keypad_init();
        configureDAC();
        timer_init();

        while (1) {
                //waiting for a key to be pressed
                while (read_keypad() == 0);
                keyPress = read_keypad();
                CNT_tracker = 0; //restarts the count
                //releasing the key for a new value
                while ((read_keypad()) != 0);

                //frequency cases; pins: 1 - 5
                if (keyPress == 1){          //100Hz
                        frequency = 1;
```

```
		}
		else if (keyPress == 2){	//200Hz
			frequency = 2;
		}
		else if (keyPress == 3){	//300Hz
			frequency = 3;
		}
		else if (keyPress == 4){	//400Hz
			frequency = 4;
		}
		else if (keyPress == 5){	//500Hz
			frequency = 5;
		}

	//Duty cycle; pins: pound, star, and 0
	//The duty cycle is built using the 1080 samples as a base
	//The the Duty_Cycle variable starts as half the amount of
	//samples and only writes high to the dac that amount of
	//samples

		if (square_flag == 1){

			//prevents the duty cycle from going over 90%
			//increases the duty cycle by 10%
			if ((keyPress == POUND_KEY)
				&& (Duty_Cycle != CNT_max)){

				Duty_Cycle += CNT_ten_percent;
				for(int i = 0; i < 500;  i ++);
				//key de-bouncer
			}

			//prevents the duty cycle from going bellow 10%
			//decreases the duty cycle by 10%

			else if ((keyPress == STAR_KEY)
				&& (Duty_Cycle != CNT_ten_percent)){

				Duty_Cycle -= CNT_ten_percent;
				for(int i = 0; i < 500;  i ++);
				 //key de-bouncer
			}
			//set the duty cycle to 50%
			else if (keyPress == ZERO_KEY){
				Duty_Cycle = CNT_half;
			}
		}

	}
}

//ISR Functions:
//1. updates the count and resets it when it gets to the sample amount
```

```
//2. reads the keypress value from the while loop in main to assign the
wave variable with the correct waveform
//3. reads the previously set wave variable to decide which waveform the
DAC will output
void TIM2_IRQHandler(void) {
	//waveform definitions
	static uint8_t SINE = 1;
	static uint8_t TRI = 2;
	static uint8_t SAW = 3;
	static uint8_t SQUARE = 4;
	static uint8_t wave = 0;

	//reset the sample amount to 0 if the max value of 1080 is hit
	if (CNT_tracker > CNT) {
		CNT_tracker = 0;
	}

	//Sets the wave variable once keys 6-9 are pressed
	//Solves the issue of the keypress variable equaling 0 when it
	//isn't pressed.
	//The wave value is necessary so the waveform does not switch to
	//default square wave
	if (keyPress == 6){
		wave = SINE;
	}
	else if (keyPress == 7){
		wave = TRI;
	}
	else if (keyPress == 8){
		wave = SAW;
	}
	else if ((keyPress == 9) | (square_flag == 1)){
		//*(square_flag == 1)* this allows for the duty cycle to be
		//changed when the function generator is turned on

		wave = SQUARE;
	}

	//writes the defined wave array to the DAC
	//waveforms: Sine, Triangle, Sawtooth, Square
	if (wave == SINE) {
		writeDAC(sine[CNT_tracker]); //predefined array in shapes.h
	}
	else if(wave == TRI) {
		writeDAC(tri[CNT_tracker]); //predefined array in shapes.h
	}
	else if(wave == SAW) {
		writeDAC(saw[CNT_tracker]); //predefined array in shapes.h
	}
	else if (wave == SQUARE){
		square_flag = 1;
		if (CNT_tracker < Duty_Cycle){
		//creates the high portion of the square wave
			writeDAC(Vmax); //3V DAC output
```

```c
		}
		else{
			writeDAC(0); //0V DAC output
		}
	}

	//updates the square flag if a different wave is selected
	if (wave != SQUARE){
		square_flag = 0;
	}
	//the frequency controls the count, the greater the frequency, the
	//samples that are skipped in the count

	CNT_tracker += frequency;
	//clears the interrupt flags
	TIM2->SR &= ~(TIM_SR_UIF | TIM_SR_CC1IF);
	//updates the register after 296 clock cycles
	TIM2->CCR1 += interuptCNT;
}

void timer_init(void) {
	// Configure Timer TIM2

	//enable TIM2 clock
	RCC->APB1ENR1 |= (RCC_APB1ENR1_TIM2EN);

	// enable interrupts on UEV (update event)
	TIM2->DIER |= (TIM_DIER_UIE | TIM_DIER_CC1IE);

	// clear interrupt flag
	TIM2->SR &= ~(TIM_SR_UIF | TIM_SR_CC1IF);

	// set capture compare register
	TIM2->CCR1 = interuptCNT - 1;

	// enable TIM2 ISR in NVIC
	NVIC->ISER[0] = (1 << (TIM2_IRQn & 0x1F));

	// start timer
	TIM2->CR1 |= (TIM_CR1_CEN);

	// enable interrupts globally
	__enable_irq();
}
/**
 * @brief System Clock Configuration
 * @retval None
 */
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_10;
	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_1) !=
HAL_OK) {
		Error_Handler();
	}
}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void) {
	/* USER CODE BEGIN Error_Handler_Debug */
	/* User can add his own implementation to report the HAL error
return
 state */
	__disable_irq();
	while (1) {
	}
	/* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
```

```
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
	/* USER CODE BEGIN 6 */
	/* User can add his own implementation to report the file name and
line
number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
	/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/
```

```c
/*
 * shapes.h
 *
 *  Created on: May 7, 2022
 *      Author: Lily Goldman
 */

#ifndef SRC_SHAPES_H_
#define SRC_SHAPES_H_

//ISR
#define interuptCNT 296

//Duty Cycle
#define CNT 1080
#define CNT_half CNT/2
#define CNT_ten_percent 108
#define CNT_max 972

//DAC
#define Vmax 3600

//Keypad Values
#define POUND_KEY 12
#define STAR_KEY 10
#define ZERO_KEY 11

//Waveform Arrays
//generated in Matlab
int sine[1080] = { 1861, 1872, 1883, 1894, 1905, 1915, 1926, 1937, 1948,
1959,1970, 1980, 1991, 2002, 2013, 2024, 2034, 2045, 2056, 2067, 2077,
2088,2099, 2110, 2120, 2131, 2142, 2153, 2163, 2174, 2185, 2195, 2206,
2217,2227, 2238, 2248, 2259, 2270, 2280, 2291, 2301, 2312, 2322, 2333,
2343,2354, 2364, 2375, 2385, 2395, 2406, 2416, 2426, 2437, 2447, 2457,
2467,2478, 2488, 2498, 2508, 2518, 2529, 2539, 2549, 2559, 2569, 2579,
2589,2599, 2609, 2619, 2628, 2638, 2648, 2658, 2668, 2677, 2687, 2697,
2707,2716, 2726, 2735, 2745, 2754, 2764, 2773, 2783, 2792, 2802, 2811,
2820,2829, 2839, 2848, 2857, 2866, 2875, 2884, 2893, 2902, 2911, 2920,
2929,2938, 2947, 2956, 2964, 2973, 2982, 2990, 2999, 3008, 3016, 3025,
3033,3041, 3050, 3058, 3066, 3075, 3083, 3091, 3099, 3107, 3115, 3123,
3131,3139, 3147, 3155, 3162, 3170, 3178, 3185, 3193, 3201, 3208, 3216,
3223,3230, 3238, 3245, 3252, 3259, 3266, 3274, 3281, 3288, 3294, 3301,
3308,3315, 3322, 3328, 3335, 3342, 3348, 3355, 3361, 3368, 3374, 3380,
3386,3393, 3399, 3405, 3411, 3417, 3423, 3429, 3434, 3440, 3446, 3452,
3457,3463, 3468, 3474, 3479, 3484, 3490, 3495, 3500, 3505, 3510, 3515,
3520,3525, 3530, 3535, 3539, 3544, 3549, 3553, 3558, 3562, 3567, 3571,
3575,3579, 3583, 3588, 3592, 3596, 3599, 3603, 3607, 3611, 3615, 3618,
3622,3625, 3629, 3632, 3635, 3639, 3642, 3645, 3648, 3651, 3654, 3657,
3660,3662, 3665, 3668, 3670, 3673, 3675, 3678, 3680, 3682, 3685, 3687,
3689,3691, 3693, 3695, 3697, 3698, 3700, 3702, 3703, 3705, 3706, 3708,
3709,3711, 3712, 3713, 3714, 3715, 3716, 3717, 3718, 3719, 3719, 3720,
3721,3721, 3722, 3722, 3722, 3723, 3723, 3723, 3723, 3723, 3723, 3723,
3723,3723, 3722, 3722, 3722, 3721, 3721, 3720, 3719, 3719, 3718, 3717,
3716,3715, 3714, 3713, 3712, 3711, 3709, 3708, 3706, 3705, 3703, 3702,
```

3700,3698, 3697, 3695, 3693, 3691, 3689, 3687, 3685, 3682, 3680, 3678,
3675,3673, 3670, 3668, 3665, 3662, 3660, 3657, 3654, 3651, 3648, 3645,
3642,3639, 3635, 3632, 3629, 3625, 3622, 3618, 3615, 3611, 3607, 3603,
3599,3596, 3592, 3588, 3583, 3579, 3575, 3571, 3567, 3562, 3558, 3553,
3549,3544, 3539, 3535, 3530, 3525, 3520, 3515, 3510, 3505, 3500, 3495,
3490,3484, 3479, 3474, 3468, 3463, 3457, 3452, 3446, 3440, 3434, 3429,
3423,3417, 3411, 3405, 3399, 3393, 3386, 3380, 3374, 3368, 3361, 3355,
3348,3342, 3335, 3328, 3322, 3315, 3308, 3301, 3294, 3288, 3281, 3274,
3266,3259, 3252, 3245, 3238, 3230, 3223, 3216, 3208, 3201, 3193, 3185,
3178,3170, 3162, 3155, 3147, 3139, 3131, 3123, 3115, 3107, 3099, 3091,
3083,3075, 3066, 3058, 3050, 3041, 3033, 3025, 3016, 3008, 2999, 2990,
2982,2973, 2964, 2956, 2947, 2938, 2929, 2920, 2911, 2902, 2893, 2884,
2875,2866, 2857, 2848, 2839, 2829, 2820, 2811, 2802, 2792, 2783, 2773,
2764,2754, 2745, 2735, 2726, 2716, 2707, 2697, 2687, 2677, 2668, 2658,
2648,2638, 2628, 2619, 2609, 2599, 2589, 2579, 2569, 2559, 2549, 2539,
2529,2518, 2508, 2498, 2488, 2478, 2467, 2457, 2447, 2437, 2426, 2416,
2406,2395, 2385, 2375, 2364, 2354, 2343, 2333, 2322, 2312, 2301, 2291,
2280,2270, 2259, 2248, 2238, 2227, 2217, 2206, 2195, 2185, 2174, 2163,
2153,2142, 2131, 2120, 2110, 2099, 2088, 2077, 2067, 2056, 2045, 2034,
2024,2013, 2002, 1991, 1980, 1970, 1959, 1948, 1937, 1926, 1915, 1905,
1894,1883, 1872, 1861, 1850, 1840, 1829, 1818, 1807, 1796, 1786, 1775,
1764,1753, 1742, 1731, 1721, 1710, 1699, 1688, 1677, 1667, 1656, 1645,
1634,1624, 1613, 1602, 1591, 1581, 1570, 1559, 1549, 1538, 1527, 1517,
1506,1495, 1485, 1474, 1464, 1453, 1443, 1432, 1421, 1411, 1400, 1390,
1379,1369, 1359, 1348, 1338, 1327, 1317, 1307, 1296, 1286, 1276, 1265,
1255,1245, 1235, 1225, 1214, 1204, 1194, 1184, 1174, 1164, 1154, 1144,
1134,1124, 1114, 1104, 1094, 1084, 1074, 1065, 1055, 1045, 1035, 1026,
1016,1006, 997, 987, 978, 968, 959, 949, 940, 930, 921, 912, 902, 893
884,875, 866, 856, 847, 838, 829, 820, 811, 802, 793, 785, 776, 767,
758,750, 741, 732, 724, 715, 707, 698, 690, 681, 673, 665, 656, 648,
640,632, 624, 616, 607, 600, 592, 584, 576, 568, 560, 552, 545, 537,
530,522, 515, 507, 500, 492, 485, 478, 470, 463, 456, 449, 442, 435,
428,421, 414, 408, 401, 394, 388, 381, 374, 368, 361, 355, 349, 342, 336,
330, 324, 318, 312, 306, 300, 294, 288, 282, 277, 271, 265, 260, 254,
249, 244, 238, 233, 228, 223, 217, 212, 207, 202, 198, 193, 188, 183,
179, 174, 169, 165, 160, 156, 152, 148, 143, 139, 135, 131, 127, 123,
119, 116, 112, 108, 104, 101, 97, 94, 91, 87, 84, 81, 78, 75, 72, 69,
66, 63, 60, 57, 55, 52, 50, 47, 45, 42, 40, 38, 36, 34, 32, 30, 28, 26,
24, 22, 21, 19, 18, 16, 15, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 3, 2,
2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 24, 26, 28, 30, 32, 34,
36, 38, 40, 42, 45, 47, 50, 52, 55, 57, 60, 63, 66, 69, 72, 75, 78, 81,
84, 87, 91, 94, 97, 101, 104, 108, 112, 116, 119, 123, 127, 131, 135,
139, 143, 148, 152, 156, 160, 165, 169, 174, 179, 183, 188, 193, 198,
202, 207, 212, 217, 223, 228, 233, 238, 244, 249, 254, 260, 265, 271,
277, 282, 288, 294, 300, 306, 312, 318, 324, 330, 336, 342, 349, 355,361,
368, 374, 381, 388, 394, 401, 408, 414, 421, 428, 435, 442, 449, 456,
463, 470, 478, 485, 492, 500, 507, 515, 522, 530, 537, 545, 552,560, 568,
576, 584, 592, 600, 607, 616, 624, 632, 640, 648, 656, 665, 673, 681, 690,
698, 707, 715, 724, 732, 741, 750, 758, 767, 776, 785,793, 802, 811, 820,
829, 838, 847, 856, 866, 875, 884, 893, 902, 912,921, 930, 940, 949, 959,
968, 978, 987, 997, 1006, 1016, 1026, 1035,1045, 1055, 1065, 1074, 1084,
1094, 1104, 1114, 1124, 1134, 1144, 1154,1164, 1174, 1184, 1194, 1204,
1214, 1225, 1235, 1245, 1255, 1265, 1276,1286, 1296, 1307, 1317, 1327,

```
1338, 1348, 1359, 1369, 1379, 1390, 1400,1411, 1421, 1432, 1443, 1453,
1464, 1474, 1485, 1495, 1506, 1517, 1527,1538, 1549, 1559, 1570, 1581,
1591, 1602, 1613, 1624, 1634, 1645, 1656,1667, 1677, 1688, 1699, 1710,
1721, 1731, 1742, 1753, 1764, 1775, 1786, 1796, 1807, 1818, 1829, 1840,
1850, 1861 };

int saw[1080] = { 0, 3, 6, 10, 13, 17, 20, 24, 27, 31, 34, 37, 41, 44, 48,
51,55, 58, 62, 65, 68, 72, 75, 79, 82, 86, 89, 93, 96, 99, 103, 106,
110,113, 117, 120, 124, 127, 131, 134, 137, 141, 144, 148, 151, 155,
158,162, 165, 168, 172, 175, 179, 182, 186, 189, 193, 196, 199, 203,
206,210, 213, 217, 220, 224, 227, 231, 234, 237, 241, 244, 248, 251,
255,258, 262, 265, 268, 272, 275, 279, 282, 286, 289, 293, 296, 299,
303,306, 310, 313, 317, 320, 324, 327, 330, 334, 337, 341, 344, 348, 351,
355, 358, 362, 365, 368, 372, 375, 379, 382, 386, 389, 393, 396, 399,
403, 406, 410, 413, 417, 420, 424, 427, 430, 434, 437, 441, 444, 448,
451, 455, 458, 462, 465, 468, 472, 475, 479, 482, 486, 489, 493, 496,
499, 503, 506, 510, 513, 517, 520, 524, 527, 530, 534, 537, 541, 544,
548, 551, 555, 558, 561, 565, 568, 572, 575, 579, 582, 586, 589, 593,
596, 599, 603, 606, 610, 613, 617, 620, 624, 627, 630, 634, 637, 641,
644, 648, 651, 655, 658, 661, 665, 668, 672, 675, 679, 682, 686, 689,
693, 696, 699, 703, 706, 710, 713, 717, 720, 724, 727, 730, 734, 737,
741, 744, 748, 751, 755, 758, 761, 765, 768, 772, 775, 779, 782, 786,
789, 792, 796, 799, 803, 806, 810, 813, 817, 820, 824, 827, 830, 834,
837, 841, 844, 848, 851, 855, 858, 861, 865, 868, 872, 875, 879, 882,
886, 889, 892, 896, 899, 903, 906, 910, 913, 917, 920, 924, 927, 930,
934, 937, 941, 944, 948, 951, 955, 958, 961, 965, 968, 972, 975, 979,
982, 986, 989, 992, 996, 999, 1003, 1006, 1010, 1013, 1017, 1020, 1024,
1027, 1030, 1034, 1037, 1041, 1044, 1048, 1051, 1055, 1058, 1061, 1065,
1068, 1072, 1075, 1079, 1082, 1086, 1089, 1092, 1096, 1099, 1103, 1106,
1110, 1113, 1117, 1120, 1123, 1127, 1130, 1134, 1137, 1141, 1144, 1148,
1151, 1155, 1158, 1161, 1165, 1168, 1172, 1175, 1179, 1182, 1186, 1189,
1192, 1196, 1199, 1203, 1206, 1210, 1213, 1217, 1220, 1223, 1227, 1230,
1234, 1237, 1241, 1244, 1248, 1251, 1255, 1258, 1261, 1265, 1268, 1272,
1275, 1279, 1282, 1286, 1289, 1292, 1296, 1299, 1303, 1306, 1310, 1313,
1317, 1320, 1323, 1327, 1330, 1334, 1337, 1341, 1344, 1348, 1351, 1354,
1358, 1361, 1365, 1368, 1372, 1375, 1379, 1382, 1386, 1389, 1392, 1396,
1399, 1403, 1406, 1410, 1413, 1417, 1420, 1423, 1427, 1430, 1434, 1437,
1441, 1444, 1448, 1451, 1454, 1458, 1461, 1465, 1468, 1472, 1475, 1479,
1482, 1486, 1489, 1492, 1496, 1499, 1503, 1506, 1510, 1513, 1517, 1520,
1523, 1527, 1530, 1534, 1537, 1541, 1544, 1548, 1551, 1554, 1558, 1561,
1565, 1568, 1572, 1575, 1579, 1582, 1585, 1589, 1592, 1596, 1599, 1603,
1606, 1610, 1613, 1617, 1620, 1623, 1627, 1630, 1634, 1637, 1641, 1644,
1648, 1651, 1654, 1658, 1661, 1665, 1668, 1672, 1675, 1679, 1682, 1685,
1689, 1692, 1696, 1699, 1703, 1706, 1710, 1713, 1717, 1720, 1723, 1727,
1730, 1734, 1737, 1741, 1744, 1748, 1751, 1754, 1758, 1761, 1765, 1768,
1772, 1775, 1779, 1782, 1785, 1789, 1792, 1796, 1799, 1803, 1806, 1810,
1813, 1816, 1820, 1823, 1827, 1830, 1834, 1837, 1841, 1844, 1848, 1851,
1854, 1858, 1861, 1865, 1868, 1872, 1875, 1879, 1882, 1885, 1889, 1892,
1896, 1899, 1903, 1906, 1910, 1913, 1916, 1920, 1923, 1927, 1930, 1934,
1937, 1941, 1944, 1948, 1951, 1954, 1958, 1961, 1965, 1968, 1972, 1975,
1979, 1982, 1985, 1989, 1992, 1996, 1999, 2003, 2006, 2010, 2013, 2016,
2020, 2023, 2027, 2030, 2034, 2037, 2041, 2044, 2048, 2051, 2054, 2058,
2061, 2065, 2068, 2072, 2075, 2079, 2082, 2085, 2089, 2092, 2096, 2099,
2103, 2106, 2110, 2113, 2116, 2120, 2123, 2127, 2130, 2134, 2137, 2141,
```

```
2144, 2147, 2151, 2154, 2158, 2161, 2165, 2168, 2172, 2175, 2179, 2182,
2185, 2189, 2192, 2196, 2199, 2203, 2206, 2210, 2213, 2216, 2220, 2223,
2227, 2230, 2234, 2237, 2241, 2244, 2247, 2251, 2254, 2258, 2261, 2265,
2268, 2272, 2275, 2279, 2282, 2285, 2289, 2292, 2296, 2299, 2303, 2306,
2310, 2313, 2316, 2320, 2323, 2327, 2330, 2334, 2337, 2341, 2344, 2347,
2351, 2354, 2358, 2361, 2365, 2368, 2372, 2375, 2378, 2382, 2385, 2389,
2392, 2396, 2399, 2403, 2406, 2410, 2413, 2416, 2420, 2423, 2427, 2430,
2434, 2437, 2441, 2444, 2447, 2451, 2454, 2458, 2461, 2465, 2468, 2472,
2475, 2478, 2482, 2485, 2489, 2492, 2496, 2499, 2503, 2506, 2510, 2513,
2516, 2520, 2523, 2527, 2530, 2534, 2537, 2541, 2544, 2547, 2551, 2554,
2558, 2561, 2565, 2568, 2572, 2575, 2578, 2582, 2585, 2589, 2592, 2596,
2599, 2603, 2606, 2609, 2613, 2616, 2620, 2623, 2627, 2630, 2634,
2637,2641, 2644, 2647, 2651, 2654, 2658, 2661, 2665, 2668, 2672, 2675,
2678,2682, 2685, 2689, 2692, 2696, 2699, 2703, 2706, 2709, 2713, 2716,
2720,2723, 2727, 2730, 2734, 2737, 2741, 2744, 2747, 2751, 2754, 2758,
2761,2765, 2768, 2772, 2775, 2778, 2782, 2785, 2789, 2792, 2796, 2799,
2803,2806, 2809, 2813, 2816, 2820, 2823, 2827, 2830, 2834, 2837, 2840,
2844,2847, 2851, 2854, 2858, 2861, 2865, 2868, 2872, 2875, 2878, 2882,
2885,2889, 2892, 2896, 2899, 2903, 2906, 2909, 2913, 2916, 2920, 2923,
2927,2930, 2934, 2937, 2940, 2944, 2947, 2951, 2954, 2958, 2961, 2965,
2968,2972, 2975, 2978, 2982, 2985, 2989, 2992, 2996, 2999, 3003, 3006,
3009,3013, 3016, 3020, 3023, 3027, 3030, 3034, 3037, 3040, 3044, 3047,
3051,3054, 3058, 3061, 3065, 3068, 3072, 3075, 3078, 3082, 3085, 3089,
3092,3096, 3099, 3103, 3106, 3109, 3113, 3116, 3120, 3123, 3127, 3130,
3134,3137, 3140, 3144, 3147, 3151, 3154, 3158, 3161, 3165, 3168, 3171,
3175,3178, 3182, 3185, 3189, 3192, 3196, 3199, 3203, 3206, 3209, 3213,
3216,3220, 3223, 3227, 3230, 3234, 3237, 3240, 3244, 3247, 3251, 3254,
3258,3261, 3265, 3268, 3271, 3275, 3278, 3282, 3285, 3289, 3292, 3296,
3299,3303, 3306, 3309, 3313, 3316, 3320, 3323, 3327, 3330, 3334, 3337,
3340,3344, 3347, 3351, 3354, 3358, 3361, 3365, 3368, 3371, 3375, 3378,
3382,3385, 3389, 3392, 3396, 3399, 3402, 3406, 3409, 3413, 3416, 3420,
3423,3427, 3430, 3434, 3437, 3440, 3444, 3447, 3451, 3454, 3458, 3461,
3465,3468, 3471, 3475, 3478, 3482, 3485, 3489, 3492, 3496, 3499, 3502,
3506,3509, 3513, 3516, 3520, 3523, 3527, 3530, 3534, 3537, 3540, 3544,
3547,3551, 3554, 3558, 3561, 3565, 3568, 3571, 3575, 3578, 3582, 3585,
3589,3592, 3596, 3599, 3602, 3606, 3609, 3613, 3616, 3620, 3623, 3627,
3630,3633, 3637, 3640, 3644, 3647, 3651, 3654, 3658, 3661, 3665, 3668,
3671,3675, 3678, 3682, 3685, 3689, 3692, 3696, 3699, 3702, 3706, 3709,
3713,3716, 3720, 3723 };

int tri[1080] = { 0, 6, 13, 20, 27, 34, 41, 48, 55, 62, 68, 75, 82, 89,
96, 103,110, 117, 124, 131, 137, 144, 151, 158, 165, 172, 179, 186, 193,
199,206, 213, 220, 227, 234, 241, 248, 255, 262, 268, 275, 282, 289, 296,
303, 310, 317, 324, 330, 337, 344, 351, 358, 365, 372, 379, 386, 393,
399, 406, 413, 420, 427, 434, 441, 448, 455, 462, 468, 475, 482, 489,
496, 503, 510, 517, 524, 530, 537, 544, 551, 558, 565, 572, 579, 586,
593, 599, 606, 613, 620, 627, 634, 641, 648, 655, 661, 668, 675, 682,
689, 696, 703, 710, 717, 724, 730, 737, 744, 751, 758, 765, 772, 779,
786, 792, 799, 806, 813, 820, 827, 834, 841, 848, 855, 861, 868, 875,
882, 889, 896, 903, 910, 917, 924, 930, 937, 944, 951, 958, 965, 972,
979, 986, 992, 999, 1006, 1013, 1020, 1027, 1034, 1041, 1048, 1055,
1061, 1068, 1075, 1082, 1089, 1096, 1103, 1110, 1117, 1123, 1130, 1137,
1144, 1151, 1158, 1165, 1172, 1179, 1186, 1192, 1199, 1206, 1213, 1220,
1227, 1234, 1241, 1248, 1255, 1261, 1268, 1275, 1282, 1289, 1296, 1303,
```

1310, 1317, 1323, 1330, 1337, 1344, 1351, 1358, 1365, 1372, 1379, 1386,
1392, 1399, 1406, 1413, 1420, 1427, 1434, 1441, 1448, 1454, 1461, 1468,
1475, 1482, 1489, 1496, 1503, 1510, 1517, 1523, 1530, 1537, 1544, 1551,
1558, 1565, 1572, 1579, 1585, 1592, 1599, 1606, 1613, 1620, 1627, 1634,
1641, 1648, 1654, 1661, 1668, 1675, 1682, 1689, 1696, 1703, 1710, 1717,
1723, 1730, 1737, 1744, 1751, 1758, 1765, 1772, 1779, 1785, 1792, 1799,
1806, 1813, 1820, 1827, 1834, 1841, 1848, 1854, 1861, 1868, 1875, 1882,
1889, 1896, 1903, 1910, 1916, 1923, 1930, 1937, 1944, 1951, 1958, 1965,
1972, 1979, 1985, 1992, 1999, 2006, 2013, 2020, 2027, 2034, 2041, 2048,
2054, 2061, 2068, 2075, 2082, 2089, 2096, 2103, 2110, 2116, 2123, 2130,
2137, 2144, 2151, 2158, 2165, 2172, 2179, 2185, 2192, 2199, 2206, 2213,
2220, 2227, 2234, 2241, 2247, 2254, 2261, 2268, 2275, 2282, 2289, 2296,
2303, 2310, 2316, 2323, 2330, 2337, 2344, 2351, 2358, 2365, 2372, 2378,
2385, 2392, 2399, 2406, 2413, 2420, 2427, 2434, 2441, 2447, 2454, 2461,
2468, 2475, 2482, 2489, 2496, 2503, 2510, 2516, 2523, 2530, 2537, 2544,
2551, 2558, 2565, 2572, 2578, 2585, 2592, 2599, 2606, 2613, 2620, 2627,
2634, 2641, 2647, 2654, 2661, 2668, 2675, 2682, 2689, 2696, 2703, 2709,
2716, 2723, 2730, 2737, 2744, 2751, 2758, 2765, 2772, 2778, 2785, 2792,
2799, 2806, 2813, 2820, 2827, 2834, 2840, 2847, 2854, 2861, 2868, 2875,
2882, 2889, 2896, 2903, 2909, 2916, 2923, 2930, 2937, 2944, 2951, 2958,
2965, 2972, 2978, 2985, 2992, 2999, 3006, 3013, 3020, 3027, 3034, 3040,
3047, 3054, 3061, 3068, 3075, 3082, 3089, 3096, 3103, 3109, 3116, 3123,
3130, 3137, 3144, 3151, 3158, 3165, 3171, 3178, 3185, 3192, 3199, 3206,
3213, 3220, 3227, 3234, 3240, 3247, 3254, 3261, 3268, 3275, 3282, 3289,
3296, 3303, 3309, 3316, 3323, 3330, 3337, 3344, 3351, 3358, 3365,
3371,3378, 3385, 3392, 3399, 3406, 3413, 3420, 3427, 3434, 3440, 3447,
3454,3461, 3468, 3475, 3482, 3489, 3496, 3502, 3509, 3516, 3523, 3530,
3537,3544, 3551, 3558, 3565, 3571, 3578, 3585, 3592, 3599, 3606, 3613,
3620,3627, 3633, 3640, 3647, 3654, 3661, 3668, 3675, 3682, 3689, 3696,
3702,3709, 3716, 3716, 3709, 3702, 3696, 3689, 3682, 3675, 3668, 3661,
3654,3647, 3640, 3633, 3627, 3620, 3613, 3606, 3599, 3592, 3585, 3578,
3571,3565, 3558, 3551, 3544, 3537, 3530, 3523, 3516, 3509, 3502, 3496,
3489,3482, 3475, 3468, 3461, 3454, 3447, 3440, 3434, 3427, 3420, 3413,
3406,3399, 3392, 3385, 3378, 3371, 3365, 3358, 3351, 3344, 3337, 3330,
3323,3316, 3309, 3303, 3296, 3289, 3282, 3275, 3268, 3261, 3254, 3247,
3240,3234, 3227, 3220, 3213, 3206, 3199, 3192, 3185, 3178, 3171, 3165,
3158,3151, 3144, 3137, 3130, 3123, 3116, 3109, 3103, 3096, 3089, 3082,
3075,3068, 3061, 3054, 3047, 3040, 3034, 3027, 3020, 3013, 3006, 2999,
2992,2985, 2978, 2972, 2965, 2958, 2951, 2944, 2937, 2930, 2923, 2916,
2909,2903, 2896, 2889, 2882, 2875, 2868, 2861, 2854, 2847, 2840, 2834,
2827,2820, 2813, 2806, 2799, 2792, 2785, 2778, 2772, 2765, 2758, 2751,
2744,2737, 2730, 2723, 2716, 2709, 2703, 2696, 2689, 2682, 2675, 2668,
2661,2654, 2647, 2641, 2634, 2627, 2620, 2613, 2606, 2599, 2592, 2585,
2578,2572, 2565, 2558, 2551, 2544, 2537, 2530, 2523, 2516, 2510, 2503,
2496,2489, 2482, 2475, 2468, 2461, 2454, 2447, 2441, 2434, 2427, 2420,
2413,2406, 2399, 2392, 2385, 2378, 2372, 2365, 2358, 2351, 2344, 2337,
2330,2323, 2316, 2310, 2303, 2296, 2289, 2282, 2275, 2268, 2261, 2254,
2247,2241, 2234, 2227, 2220, 2213, 2206, 2199, 2192, 2185, 2179, 2172,
2165,2158, 2151, 2144, 2137, 2130, 2123, 2116, 2110, 2103, 2096, 2089,
2082,2075, 2068, 2061, 2054, 2048, 2041, 2034, 2027, 2020, 2013, 2006,
1999,1992, 1985, 1979, 1972, 1965, 1958, 1951, 1944, 1937, 1930, 1923,
1916,1910, 1903, 1896, 1889, 1882, 1875, 1868, 1861, 1854, 1848, 1841,
1834,1827, 1820, 1813, 1806, 1799, 1792, 1785, 1779, 1772, 1765, 1758,
1751,1744, 1737, 1730, 1723, 1717, 1710, 1703, 1696, 1689, 1682, 1675,

```
1668,1661, 1654, 1648, 1641, 1634, 1627, 1620, 1613, 1606, 1599, 1592,
1585,1579, 1572, 1565, 1558, 1551, 1544, 1537, 1530, 1523, 1517, 1510,
1503,1496, 1489, 1482, 1475, 1468, 1461, 1454, 1448, 1441, 1434, 1427,
1420,1413, 1406, 1399, 1392, 1386, 1379, 1372, 1365, 1358, 1351, 1344,
1337,1330, 1323, 1317, 1310, 1303, 1296, 1289, 1282, 1275, 1268, 1261,
1255,1248, 1241, 1234, 1227, 1220, 1213, 1206, 1199, 1192, 1186, 1179,
1172,1165, 1158, 1151, 1144, 1137, 1130, 1123, 1117, 1110, 1103, 1096,
1089,1082, 1075, 1068, 1061, 1055, 1048, 1041, 1034, 1027, 1020, 1013,
1006,999, 992, 986, 979, 972, 965, 958, 951, 944, 937, 930, 924, 917,
910,903, 896, 889, 882, 875, 868, 861, 855, 848, 841, 834, 827, 820, 813,
806, 799, 792, 786, 779, 772, 765, 758, 751, 744, 737, 730, 724, 717,
710, 703, 696, 689, 682, 675, 668, 661, 655, 648, 641, 634, 627, 620,
613, 606, 599, 593, 586, 579, 572, 565, 558, 551, 544, 537, 530, 524,
517, 510, 503, 496, 489, 482, 475, 468, 462, 455, 448, 441, 434, 427,
420, 413, 406, 399, 393, 386, 379, 372, 365, 358, 351, 344, 337, 330,
324, 317, 310, 303, 296, 289, 282, 275, 268, 262, 255, 248, 241, 234,
227, 220, 213, 206, 199, 193, 186, 179, 172, 165, 158, 151, 144, 137,
131, 124, 117, 110, 103, 96, 89, 82, 75, 68, 62, 55, 48, 41, 34, 27, 20,
13, 6, 0, 0 };

#endif /* SRC_SHAPES_H_ */
```

```
/*
 * DAC.h
 *
 *  Created on: Apr 27, 2022
 *      Author: Lily Goldman
 */

#ifndef SRC_DAC_H_
#define SRC_DAC_H_

#define vREF 3300
#define resolution 4096

void configureDAC(void);
void writeDAC(uint16_t data);
uint16_t DAC_volt_conv(uint16_t inputDAC);


#endif /* SRC_DAC_H_ */
```

```
/*
 * DAC.c
 *
 * Created on: Apr 27, 2022

Author: Lily Goldman
Brief: Initializes the external dac to the SPI on the STM board.
Writes to the SPI data register when it is empty. Uses the conversion from
the DAC data sheet to allow for the proper analog output voltage.

*/
#include "main.h"
#include "DAC.h"

void configureDAC(void){
	//PA4 - Chip Select(CS)
	//PA5 - Serial Clock(SCK)
	//PA7 - Serial Data(SDI)
	RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN);

	//GPIO output configuration
	GPIOA->MODER &= ~(GPIO_MODER_MODE4
			  | GPIO_MODER_MODE5
			  | GPIO_MODER_MODE7 );

	GPIOA->MODER |= (GPIO_MODER_MODE4_1
			  | GPIO_MODER_MODE5_1
			  | GPIO_MODER_MODE7_1);

	GPIOA->OTYPER &= ~(GPIO_OTYPER_OT4
			  | GPIO_OTYPER_OT5
			  | GPIO_OTYPER_OT7);

	GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD4
			  | GPIO_PUPDR_PUPD5
			  | GPIO_PUPDR_PUPD7);

	GPIOA->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED4
			  | GPIO_OSPEEDR_OSPEED6
			  | GPIO_OSPEEDR_OSPEED7);

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

	//enable the SPI in transmit only + Pulse mode
	RCC->APB2ENR |= (RCC_APB2ENR_SPI1EN);

	SPI1 -> CR1 = (SPI_CR1_MSTR);

	SPI1 -> CR2 = (SPI_CR2_DS | SPI_CR2_NSSP | SPI_CR2_SSOE);

	SPI1 -> CR1 |= (SPI_CR1_SPE);
```

```
}

void writeDAC(uint16_t data){
	//Writes if TxFIFO empty and not busy
	while((SPI1->SR & SPI_SR_TXE)){
		//sends the 12 bit data with the 3 in front
		SPI1 -> DR = (0x3000 | data);
	}
}



uint16_t DAC_volt_conv(uint16_t inputDAC){
	uint16_t vOUT;
	//the hex equation to convert the voltages for the DAC
	vOUT = (inputDAC * resolution)/ vREF;
	return vOUT;
}
```

```c
/*
* keypad.h
*
*  Created on: Apr 5, 2022
*      Author: Lily Goldman
*/

#ifndef SRC_KEYPAD_H_
#define SRC_KEYPAD_H_

//Prototypes
void keypad_init(void);
int read_keypad(void);


#define Column2 GPIO_ODR_OD8   //Pin1_Column2
#define Column1 GPIO_ODR_OD9 //Pin3_Column1
#define Column3 GPIO_ODR_OD10 //Pin5_Column3

#define col_ODR_mask (Column2 | Column1 | Column3)

#define Row1 GPIO_IDR_ID4 //Pin2_Row1
#define Row4 GPIO_IDR_ID5 //Pin4_Row4
#define Row3 GPIO_IDR_ID6 //Pin6_Row3
#define Row2 GPIO_IDR_ID7 //Pin7_Row2


#define row_IDR_mask (Row1 | Row4 | Row3 | Row2)

#define col_length 3




#endif /* SRC_KEYPAD_H_ */
```

```c
/*
* keypad.c
*
*  Created on: Apr 5, 2022
*      Author: Lily Goldman
*      Brief: Initializes external keypad. Reads pins 0-9, #, and * and
*       outputs them as a value from 1 to 12
*/
#include "keypad.h"
#include "main.h"
#include <math.h>

int Button_Array[12] = {1,2,3,4,5,6,7,8,9,10,11,12};

void keypad_init(void){
RCC->AHB2ENR |=  (RCC_AHB2ENR_GPIOCEN);

//Row initialization
GPIOC->MODER &= ~(GPIO_MODER_MODE4
               | GPIO_MODER_MODE5
               | GPIO_MODER_MODE6
               | GPIO_MODER_MODE7); //00 input MODER

GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD4
               | GPIO_PUPDR_PUPD5
               | GPIO_PUPDR_PUPD6
               | GPIO_PUPDR_PUPD7);

GPIOC->PUPDR |= (GPIO_PUPDR_PUPD4_1
               | GPIO_PUPDR_PUPD5_1
               | GPIO_PUPDR_PUPD6_1
               | GPIO_PUPDR_PUPD7_1); //Pull Down resistors


//Column initialization
GPIOC->MODER &= ~(GPIO_MODER_MODE8
               | GPIO_MODER_MODE9
               | GPIO_MODER_MODE10);

GPIOC->MODER |= (GPIO_MODER_MODE8_0
               | GPIO_MODER_MODE9_0
               | GPIO_MODER_MODE10_0);  //01 output MODER

GPIOC->OTYPER &= ~(GPIO_OTYPER_OT8 | GPIO_OTYPER_OT9 | GPIO_OTYPER_OT10);

GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD8
               | GPIO_PUPDR_PUPD9
               | GPIO_PUPDR_PUPD10); //Push Pull

//set columns high
GPIOC->ODR &= ~(col_ODR_mask);
GPIOC->ODR |= (col_ODR_mask);
}
```

```c
int read_keypad(void){
//read the IDR
uint32_t rows = GPIOC->IDR;
rows = (rows & row_IDR_mask);
uint8_t row_index = 0;

//check if the keypad is read
if (rows != 0){
	for(uint8_t col = 0; col <= (col_length - 1); col++){
		//set one column high at a time
		GPIOC->ODR &= ~(col_ODR_mask);
		GPIOC->ODR |= (1<<(col + GPIO_ODR_OD8_Pos));

		//find the row that is high
		rows = GPIOC->IDR;
		rows = (rows & row_IDR_mask);
		rows = rows >> 4;

		//find which row is high
		if (rows != 0){
			if (rows == 1)
				row_index = 0;
			if (rows == 2)
				row_index = 1;
			if (rows == 4)
				row_index = 2;
			if (rows == 8)
				row_index = 3;

			//return the value of the key
			uint8_t index = (row_index * 3) + col;
			return index + 1;
				}
			}
		}


else
	GPIOC->ODR &= ~(col_ODR_mask);
	GPIOC->ODR |= (col_ODR_mask);
	return 0;
}
```

References

[1] Andrew, “COM-08653, “COM-08653 Datasheet. [Online]. Available: https://cdn.sparkfun.com/datasheets/Components/General/SparkfunCOM-08653_Datasheet.pdf.[Accessed: 06-May-2022]. [Online]

[2] Microchip, “MCP4901/4911/4921” 8/10/12-Bit Voltage Output Digital-to-Analog Converter with SPI Interface, April 2010. [Online]

[3] STMicroelectronics, “RM0351 Reference Manual” STM32L47xxx, STM32L48xxx, STM32L49xxx and STM32L4Axxx advance Arm® - based 32-bit MCUs reference manual, May. 2015 [Revised Jun. 2021]. [Online]

[4] STMicroelectronics, “UM1724 User Manual” STM32 Nucleo-64 boards (MB1136) user manual, Feb. 2014[Revised Aug.2020]. [Online]