Function Generator

Mark Nakamae

Cal Poly SLO

April 2020

**Behavior Description**

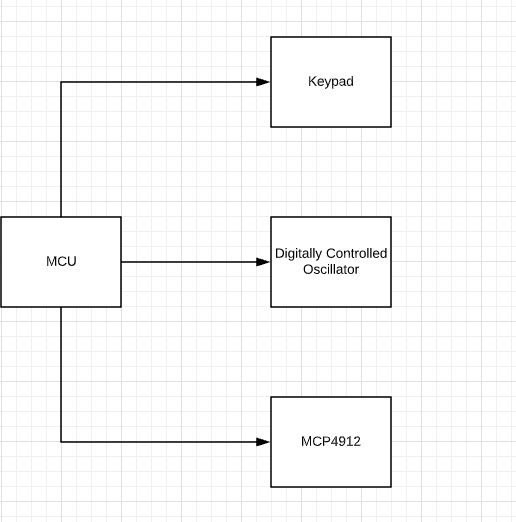
This function generator upon startup outputs a square wave at 100 Hz with a 50% duty cycle. Using the keypad buttons 6 - 9, the user can respectively output a triangle, square, sinusoidal, or sawtooth wave. The user is also able to set the output frequency to 100 Hz, 200 Hz, 300 Hz, 400 Hz, or 500Hz with the 1 - 5 buttons on the keypad. Additionally, the duty cycle for a square wave can be adjusted from 10% to 90% in intervals of 10 using the \* and # buttons to, respectively, decrease and increase the duty cycle. On the other hand, pressing 0 resets the duty cycle to 50%.

**System Specification**

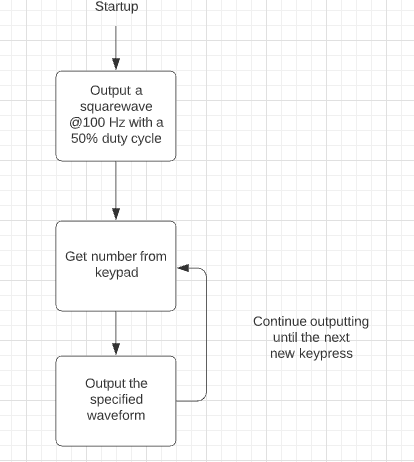
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Power Supply Voltage** | **Clock Frequency** | **Size of keypad**  **(cm)** | **Size of MSP432**  **(cm)** | **Response time of keypad** | **Size of MCP4912**  **(cm)** | **Resolution**  **(Hz)** |
| 3.3 V | 3 MHz | 6.6 x 7.6 | 5.7 x 9.2 | 1 second | 1.75 x 0.5 | 102564 |

**Table 1:** System specifications

**System Architecture**

****

**Figure 1, Black Box Diagram:** This is a high-level schematic of all the components involved in thefunction generator.

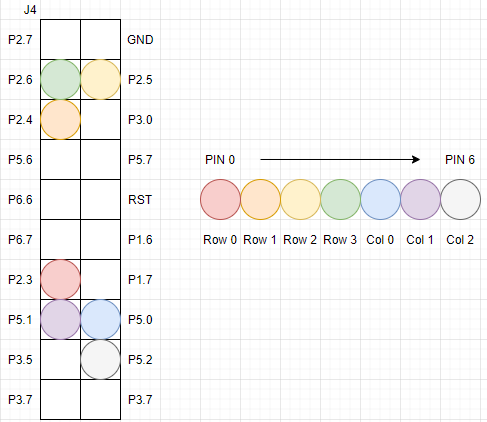
****

**Figure 2, Flow Chart:** This is a flow chart of how the program functions.

**System Schematic**

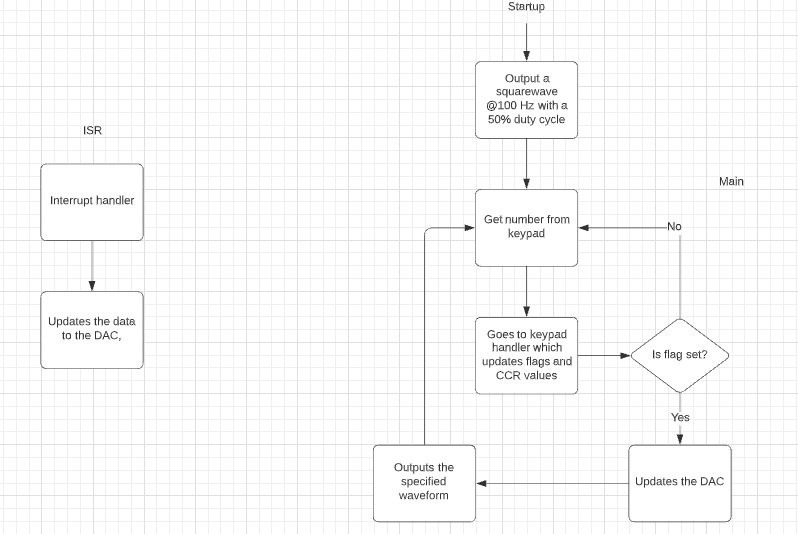
****

**Figure 3, Low Level Schematic:** This shows the MCP low level diagram



**Figure 4, Keypad Pinnings:** This diagram labels the pins connecting the keypad to the MSP432P401R board. Pin 0 on the keypad corresponds to row 0, and pin 6 corresponds to column 2. Note that J2 and J4 on the board is used for all GPIO related to the keypad.

**Software Architecture**

****

**Figure 5, Software flowchart**: shows the Main and ISR routines

**Bill of Materials**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ITEM NUMBER** | **PART NUMBER** | **ITEM** | **SUPPLIER** | **QUANTITY** | **PRICE EACH** | **PRICE EXTENDED** |
| 1 | 296-39653-ND | LaunchPad MSP432P401R Board | Texas Instruments | 1 | $23.59 | $23.59 |
| 2 | 1568-1511-ND | Jumper Wire M/F 6” 20 Pieces | SparkFun Electronics | 1 | $1.95 | $1.95 |
| 3 | 1528-1136-ND | Switch Keypad 12 Key Non-ILLUM | Adafruit Industries LLC | 1 | $3.95 | $3.95 |
| 4 | MCP4912-E/P-ND | IC DAC 10BIT V-OUT 14DIP | Microchip Technology | 1 | $1.83 | $1.83 |
| 5 | EL-CP-003 | 830 Point Solderless Breadboard 3PCS | ELEGOO | 1 | $8.99 | $8.99 |
| TOTAL | | | | | | **$40.31** |

**Table 2:** Bill of Materials

**Ethical Implications**

**Education:** Function generators are an essential piece of equipment in college labs. Having a precise function generator allows students to take the knowledge they learn in lecture and confirm these concepts in labs. Also having a function generator that can produce a variety of output waves and at different frequencies allows more experiments to be conducted and saves the school time and money instead of having to procure extra equipment. Utilizing a reliable function generator is another essential aspect. When debugging hardware, it is reassuring to know that the equipment the student uses works properly over many times. This also makes debugging a simpler task and easier to pinpoint the flaws in the student’s circuits.

**Environmental:** Function generators are a fairly common piece of lab equipment, and are often used in industry. Since they are so widely used, it’s important to create a sustainable model that does not induce a heavy strain on the environment. Some substances used in PCB production are toxic to humans directly, but have no lasting negative impacts on the environment. Additionally, mass producing these boards is also financially sustainable, and are fairly cheap to supply to both universities and industrial users.

**Societal:** In modern society, there is electronics involved in all sorts of activities from transactions to playing video games. These electronics all need to be tested and one vital piece of equipment is the function generator. With this tool, the various portions of the electronic systems can be tested and debugged. The function generators that are sold by companies sell for $1,433 which is much more than the $40.31 that this system costs. This function generator is affordable for the common student to have in their rooms and not need to travel to campus to use a function generator. With the PWM option of this function generator, you can also use this to drive leds in a dorm room and provide cool lighting.

**Appendices**

main.c

#include "msp.h"

#include "keypad.h"

#include "spi.h"

#include "sine.h"

#include "delay.h"

#define SQUARE 0

#define SINE 1

#define SAW 2

#define TANGLE 3

#define sq500 6022 // To get these values, took the DCO speed and divided by the desired

#define sq400 7529 //wave speed

#define sq300 10038

#define sq200 15057

#define sq100 30114

#define dut500 602

#define dut400 753

#define dut300 1004

#define dut200 1506

#define dut100 3011

#define sine100 234

#define tri100 116 // tangle wave

**volatile** **static** uint8\_t state = SQUARE; // output a square wave upon starting up

**volatile** **static** uint8\_t flag = 1; // when flag is set, output to the SPI

**volatile** **static** uint16\_t data = 0; // data is the value output to the SPI

**volatile** **static** uint16\_t dutybase = dut100;

**volatile** **static** uint8\_t freq = 1; // output 100 hz upon starting up

**volatile** **static** uint8\_t duty = 5; // output 50% duty cycle upon starting up

// set up the interrupts used for the waves

**void** timer\_init(**void**)

{

// setup TIMER\_A0

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG; // clear interrupt

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCR[0] = sq100; // set CCR0 count

TIMER\_A0->CCR[1] = dutybase \* duty;

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | TIMER\_A\_CTL\_MC\_\_UP; // SMCLK, up mode

NVIC->ISER[0] = 1 << ((TA0\_0\_IRQn) & 31)| (1 << ((TA0\_N\_IRQn) & 31)); // set NVIC interrupt

}

// take in input from keypad to set function generator output

**void** readin(uint8\_t key)

{

**switch**(key)

{

**case** 11: // reset duty cycle to 50%

duty = 5;

TIMER\_A0->CCR[1] = dutybase \* duty;

**break**;

**case** 1: // set freq to 100 hz

freq = 1;

**if** (state == SQUARE)

{

TIMER\_A0->CCR[0] = sq100;

dutybase = dut100;

TIMER\_A0->CCR[1] = dutybase \* duty ;

}

**else** **if** (state == TANGLE)

{

TIMER\_A0->CCR[0] = tri100;

}

**else**

{

TIMER\_A0->CCR[0] = sine100;

}

**break**;

**case** 2: // spiset freq to 200 hz

freq = 2;

**if** (state == SQUARE)

{

TIMER\_A0->CCR[0] = sq200;

dutybase = dut200;

TIMER\_A0->CCR[1] = dutybase \* duty ;

}

**else** **if** (state == TANGLE)

{

TIMER\_A0->CCR[0] = tri100;

}

**else**

{

TIMER\_A0->CCR[0] = sine100;

}

**break**;

**case** 3: // set freq to 300 hz

freq = 3;

**if** (state == SQUARE)

{

TIMER\_A0->CCR[0] = sq300;

dutybase = dut300;

TIMER\_A0->CCR[1] = dutybase \* duty ;

}

**else** **if** (state == TANGLE)

{

TIMER\_A0->CCR[0] = tri100;

}

**else**

{

TIMER\_A0->CCR[0] = sine100;

}

**break**;

**case** 4: // set freq to 400 hz

freq = 4;

**if** (state == SQUARE)

{

TIMER\_A0->CCR[0] = sq400;

dutybase = dut400;

TIMER\_A0->CCR[1] = dutybase \* duty ;

}

**else** **if** (state == TANGLE)

{

TIMER\_A0->CCR[0] = tri100;

}

**else**

{

TIMER\_A0->CCR[0] = sine100;

}

**break**;

**case** 5: // set freq to 500 hz

freq = 5;

**if** (state == SQUARE)

{

TIMER\_A0->CCR[0] = sq500;

dutybase = dut500;

TIMER\_A0->CCR[1] = dutybase \* duty ;

}

**else** **if** (state == TANGLE)

{

TIMER\_A0->CCR[0] = tri100;

}

**else**

{

TIMER\_A0->CCR[0] = sine100;

}

**break**;

**case** 6:

state = 3;

set\_DCO(FREQ\_24MHz);

TIMER\_A0->CCR[0] = tri100;

TIMER\_A0->R = 0;

**break**;

**case** 7: // output a square wave

state = 0;

set\_DCO(FREQ\_3MHz);

**switch**(freq) // change the period and duty cycle

{

**case** 1:

TIMER\_A0->CCR[0] = sq100;

dutybase = dut100;

**break**;

**case** 2:

TIMER\_A0->CCR[0] = sq200;

dutybase = dut200;

**break**;

**case** 3:

TIMER\_A0->CCR[0] = sq300;

dutybase = dut300;

**break**;

**case** 4:

TIMER\_A0->CCR[0] = sq400;

dutybase = dut400;

**break**;

**case** 5:

TIMER\_A0->CCR[0] = sq500;

dutybase = dut500;

**break**;

**default**:

**break**;

}

TIMER\_A0->CCR[1] = dutybase \* duty;

**break**;

**case** 8: // output a sine wave

state = 1;

set\_DCO(FREQ\_24MHz);

TIMER\_A0->CCR[0] = sine100;

TIMER\_A0->R = 0;

**break**;

**case** 9: // output a sawtooth wave

state = 2;

set\_DCO(FREQ\_24MHz);

TIMER\_A0->CCR[0] = sine100;

TIMER\_A0->R = 0;

**break**;

**case** 10: // decrease duty cycle by 10%

**if** (duty > 1)

{

duty--;

}

TIMER\_A0->CCR[1] = dutybase \* duty ;

**break**;

**case** 12: // increase duty cycle by 10%

**if** (duty < 9)

{

duty++;

}

TIMER\_A0->CCR[1] = dutybase \* duty ;

**break**;

**default**:

**break**;

}

}

**void** main(**void**)

{

uint8\_t press;

set\_DCO(FREQ\_3MHz);

dac\_init();

timer\_init();

keypad\_init();

\_\_enable\_irq(); // Enable global interrupt

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

**while**(1)

{

**if** (flag)

{

spi\_transmit(data);

flag = 0;

}

press = keypad\_Getkey();

**if** (press !=0xFF)

{

readin(press);

delay\_ms(500);// debounce the keypad

}

}

}

// handler used to output variable voltages for each wave

**void** TA0\_0\_IRQHandler(**void**)

{

**volatile** **static** uint16\_t ind = 0;

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG; // Clear the CCR0 interrupt

**if** (state == SINE)

{

**if** (ind >= 1023)

{

ind = 0;

}

data = sin[ind];

ind += freq;

}

**else** **if** (state == SAW)

{

**if** (ind >= 1023)

{

ind = 0;

}

data = sa[ind];

ind += freq;

}

**else** **if** (state == TANGLE)

{

**if** (ind >= 2047)

{

ind = 0;

}

data = tri[ind];

ind += freq;

}

**else**

{

data = 0x3FF;

}

flag = 1;

}

// handler used to output a low on a square wave

**void** TA0\_N\_IRQHandler(**void**)

{

TIMER\_A0->CCTL[1] &= ~TIMER\_A\_CCTLN\_CCIFG; // Clear the CCR1 interrupt

data = 0;

flag = 1;

}

sine.h

/\*

\* sine.h

\*

\* Created on: May 13, 2020

\* Author: nicks

\*/

#ifndef SINE\_H\_

#define SINE\_H\_

// lookup table to output a sine wave

**int** sin[1024] = { 512, 515, 518, 521, 524, 527, 530, 533,

537, 540, 543, 546, 549, 552, 555, 559,

562, 565, 568, 571, 574, 577, 580, 584,

587, 590, 593, 596, 599, 602, 605, 608,

611, 614, 618, 621, 624, 627, 630, 633,

636, 639, 642, 645, 648, 651, 654, 657,

660, 663, 666, 669, 672, 675, 678, 681,

684, 687, 690, 693, 696, 699, 702, 705,

707, 710, 713, 716, 719, 722, 725, 728,

730, 733, 736, 739, 742, 745, 747, 750,

753, 756, 758, 761, 764, 767, 769, 772,

775, 777, 780, 783, 785, 788, 791, 793,

796, 799, 801, 804, 806, 809, 811, 814,

816, 819, 822, 824, 826, 829, 831, 834,

836, 839, 841, 844, 846, 848, 851, 853,

855, 858, 860, 862, 865, 867, 869, 871,

874, 876, 878, 880, 882, 884, 887, 889,

891, 893, 895, 897, 899, 901, 903, 905,

907, 909, 911, 913, 915, 917, 919, 921,

923, 925, 926, 928, 930, 932, 934, 935,

937, 939, 941, 942, 944, 946, 947, 949,

951, 952, 954, 955, 957, 959, 960, 962,

963, 965, 966, 967, 969, 970, 972, 973,

974, 976, 977, 978, 980, 981, 982, 983,

985, 986, 987, 988, 989, 990, 991, 993,

994, 995, 996, 997, 998, 999, 1000, 1001,

1001, 1002, 1003, 1004, 1005, 1006, 1007, 1007,

1008, 1009, 1010, 1010, 1011, 1012, 1012, 1013,

1014, 1014, 1015, 1015, 1016, 1016, 1017, 1017,

1018, 1018, 1019, 1019, 1020, 1020, 1020, 1021,

1021, 1021, 1022, 1022, 1022, 1022, 1023, 1023,

1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023,

1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023,

1023, 1023, 1023, 1022, 1022, 1022, 1022, 1021,

1021, 1021, 1020, 1020, 1020, 1019, 1019, 1018,

1018, 1017, 1017, 1016, 1016, 1015, 1015, 1014,

1014, 1013, 1012, 1012, 1011, 1010, 1010, 1009,

1008, 1007, 1007, 1006, 1005, 1004, 1003, 1002,

1001, 1001, 1000, 999, 998, 997, 996, 995,

994, 993, 991, 990, 989, 988, 987, 986,

985, 983, 982, 981, 980, 978, 977, 976,

974, 973, 972, 970, 969, 967, 966, 965,

963, 962, 960, 959, 957, 955, 954, 952,

951, 949, 947, 946, 944, 942, 941, 939,

937, 935, 934, 932, 930, 928, 926, 925,

923, 921, 919, 917, 915, 913, 911, 909,

907, 905, 903, 901, 899, 897, 895, 893,

891, 889, 887, 884, 882, 880, 878, 876,

874, 871, 869, 867, 865, 862, 860, 858,

855, 853, 851, 848, 846, 844, 841, 839,

836, 834, 831, 829, 826, 824, 822, 819,

816, 814, 811, 809, 806, 804, 801, 799,

796, 793, 791, 788, 785, 783, 780, 777,

775, 772, 769, 767, 764, 761, 758, 756,

753, 750, 747, 745, 742, 739, 736, 733,

730, 728, 725, 722, 719, 716, 713, 710,

707, 705, 702, 699, 696, 693, 690, 687,

684, 681, 678, 675, 672, 669, 666, 663,

660, 657, 654, 651, 648, 645, 642, 639,

636, 633, 630, 627, 624, 621, 618, 614,

611, 608, 605, 602, 599, 596, 593, 590,

587, 584, 580, 577, 574, 571, 568, 565,

562, 559, 555, 552, 549, 546, 543, 540,

537, 533, 530, 527, 524, 521, 518, 515,

512, 508, 505, 502, 499, 496, 493, 490,

486, 483, 480, 477, 474, 471, 468, 464,

461, 458, 455, 452, 449, 446, 443, 439,

436, 433, 430, 427, 424, 421, 418, 415,

412, 409, 405, 402, 399, 396, 393, 390,

387, 384, 381, 378, 375, 372, 369, 366,

363, 360, 357, 354, 351, 348, 345, 342,

339, 336, 333, 330, 327, 324, 321, 318,

316, 313, 310, 307, 304, 301, 298, 295,

293, 290, 287, 284, 281, 278, 276, 273,

270, 267, 265, 262, 259, 256, 254, 251,

248, 246, 243, 240, 238, 235, 232, 230,

227, 224, 222, 219, 217, 214, 212, 209,

207, 204, 201, 199, 197, 194, 192, 189,

187, 184, 182, 179, 177, 175, 172, 170,

168, 165, 163, 161, 158, 156, 154, 152,

149, 147, 145, 143, 141, 139, 136, 134,

132, 130, 128, 126, 124, 122, 120, 118,

116, 114, 112, 110, 108, 106, 104, 102,

100, 98, 97, 95, 93, 91, 89, 88,

86, 84, 82, 81, 79, 77, 76, 74,

72, 71, 69, 68, 66, 64, 63, 61,

60, 58, 57, 56, 54, 53, 51, 50,

49, 47, 46, 45, 43, 42, 41, 40,

38, 37, 36, 35, 34, 33, 32, 30,

29, 28, 27, 26, 25, 24, 23, 22,

22, 21, 20, 19, 18, 17, 16, 16,

15, 14, 13, 13, 12, 11, 11, 10,

9, 9, 8, 8, 7, 7, 6, 6,

5, 5, 4, 4, 3, 3, 3, 2,

2, 2, 1, 1, 1, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 1, 1, 1, 2,

2, 2, 3, 3, 3, 4, 4, 5,

5, 6, 6, 7, 7, 8, 8, 9,

9, 10, 11, 11, 12, 13, 13, 14,

15, 16, 16, 17, 18, 19, 20, 21,

22, 22, 23, 24, 25, 26, 27, 28,

29, 30, 32, 33, 34, 35, 36, 37,

38, 40, 41, 42, 43, 45, 46, 47,

49, 50, 51, 53, 54, 56, 57, 58,

60, 61, 63, 64, 66, 68, 69, 71,

72, 74, 76, 77, 79, 81, 82, 84,

86, 88, 89, 91, 93, 95, 97, 98,

100, 102, 104, 106, 108, 110, 112, 114,

116, 118, 120, 122, 124, 126, 128, 130,

132, 134, 136, 139, 141, 143, 145, 147,

149, 152, 154, 156, 158, 161, 163, 165,

168, 170, 172, 175, 177, 179, 182, 184,

187, 189, 192, 194, 197, 199, 201, 204,

207, 209, 212, 214, 217, 219, 222, 224,

227, 230, 232, 235, 238, 240, 243, 246,

248, 251, 254, 256, 259, 262, 265, 267,

270, 273, 276, 278, 281, 284, 287, 290,

293, 295, 298, 301, 304, 307, 310, 313,

316, 318, 321, 324, 327, 330, 333, 336,

339, 342, 345, 348, 351, 354, 357, 360,

363, 366, 369, 372, 375, 378, 381, 384,

387, 390, 393, 396, 399, 402, 405, 409,

412, 415, 418, 421, 424, 427, 430, 433,

436, 439, 443, 446, 449, 452, 455, 458,

461, 464, 468, 471, 474, 477, 480, 483,

486, 490, 493, 496, 499, 502, 505, 508};

// lookup table to output a triangle wave

**int** tri[2048] = { 0, 1, 2, 3, 4, 5, 6, 7,

8, 9, 10, 11, 12, 13, 14, 15,

16, 17, 18, 19, 20, 21, 22, 23,

24, 25, 26, 27, 28, 29, 30, 31,

32, 33, 34, 35, 36, 37, 38, 39,

40, 41, 42, 43, 44, 45, 46, 47,

48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63,

64, 65, 66, 67, 68, 69, 70, 71,

72, 73, 74, 75, 76, 77, 78, 79,

80, 81, 82, 83, 84, 85, 86, 87,

88, 89, 90, 91, 92, 93, 94, 95,

96, 97, 98, 99, 100, 101, 102, 103,

104, 105, 106, 107, 108, 109, 110, 111,

112, 113, 114, 115, 116, 117, 118, 119,

120, 121, 122, 123, 124, 125, 126, 127,

128, 129, 130, 131, 132, 133, 134, 135,

136, 137, 138, 139, 140, 141, 142, 143,

144, 145, 146, 147, 148, 149, 150, 151,

152, 153, 154, 155, 156, 157, 158, 159,

160, 161, 162, 163, 164, 165, 166, 167,

168, 169, 170, 171, 172, 173, 174, 175,

176, 177, 178, 179, 180, 181, 182, 183,

184, 185, 186, 187, 188, 189, 190, 191,

192, 193, 194, 195, 196, 197, 198, 199,

200, 201, 202, 203, 204, 205, 206, 207,

208, 209, 210, 211, 212, 213, 214, 215,

216, 217, 218, 219, 220, 221, 222, 223,

224, 225, 226, 227, 228, 229, 230, 231,

232, 233, 234, 235, 236, 237, 238, 239,

240, 241, 242, 243, 244, 245, 246, 247,

248, 249, 250, 251, 252, 253, 254, 255,

256, 257, 258, 259, 260, 261, 262, 263,

264, 265, 266, 267, 268, 269, 270, 271,

272, 273, 274, 275, 276, 277, 278, 279,

280, 281, 282, 283, 284, 285, 286, 287,

288, 289, 290, 291, 292, 293, 294, 295,

296, 297, 298, 299, 300, 301, 302, 303,

304, 305, 306, 307, 308, 309, 310, 311,

312, 313, 314, 315, 316, 317, 318, 319,

320, 321, 322, 323, 324, 325, 326, 327,

328, 329, 330, 331, 332, 333, 334, 335,

336, 337, 338, 339, 340, 341, 342, 343,

344, 345, 346, 347, 348, 349, 350, 351,

352, 353, 354, 355, 356, 357, 358, 359,

360, 361, 362, 363, 364, 365, 366, 367,

368, 369, 370, 371, 372, 373, 374, 375,

376, 377, 378, 379, 380, 381, 382, 383,

384, 385, 386, 387, 388, 389, 390, 391,

392, 393, 394, 395, 396, 397, 398, 399,

400, 401, 402, 403, 404, 405, 406, 407,

408, 409, 410, 411, 412, 413, 414, 415,

416, 417, 418, 419, 420, 421, 422, 423,

424, 425, 426, 427, 428, 429, 430, 431,

432, 433, 434, 435, 436, 437, 438, 439,

440, 441, 442, 443, 444, 445, 446, 447,

448, 449, 450, 451, 452, 453, 454, 455,

456, 457, 458, 459, 460, 461, 462, 463,

464, 465, 466, 467, 468, 469, 470, 471,

472, 473, 474, 475, 476, 477, 478, 479,

480, 481, 482, 483, 484, 485, 486, 487,

488, 489, 490, 491, 492, 493, 494, 495,

496, 497, 498, 499, 500, 501, 502, 503,

504, 505, 506, 507, 508, 509, 510, 511,

512, 513, 514, 515, 516, 517, 518, 519,

520, 521, 522, 523, 524, 525, 526, 527,

528, 529, 530, 531, 532, 533, 534, 535,

536, 537, 538, 539, 540, 541, 542, 543,

544, 545, 546, 547, 548, 549, 550, 551,

552, 553, 554, 555, 556, 557, 558, 559,

560, 561, 562, 563, 564, 565, 566, 567,

568, 569, 570, 571, 572, 573, 574, 575,

576, 577, 578, 579, 580, 581, 582, 583,

584, 585, 586, 587, 588, 589, 590, 591,

592, 593, 594, 595, 596, 597, 598, 599,

600, 601, 602, 603, 604, 605, 606, 607,

608, 609, 610, 611, 612, 613, 614, 615,

616, 617, 618, 619, 620, 621, 622, 623,

624, 625, 626, 627, 628, 629, 630, 631,

632, 633, 634, 635, 636, 637, 638, 639,

640, 641, 642, 643, 644, 645, 646, 647,

648, 649, 650, 651, 652, 653, 654, 655,

656, 657, 658, 659, 660, 661, 662, 663,

664, 665, 666, 667, 668, 669, 670, 671,

672, 673, 674, 675, 676, 677, 678, 679,

680, 681, 682, 683, 684, 685, 686, 687,

688, 689, 690, 691, 692, 693, 694, 695,

696, 697, 698, 699, 700, 701, 702, 703,

704, 705, 706, 707, 708, 709, 710, 711,

712, 713, 714, 715, 716, 717, 718, 719,

720, 721, 722, 723, 724, 725, 726, 727,

728, 729, 730, 731, 732, 733, 734, 735,

736, 737, 738, 739, 740, 741, 742, 743,

744, 745, 746, 747, 748, 749, 750, 751,

752, 753, 754, 755, 756, 757, 758, 759,

760, 761, 762, 763, 764, 765, 766, 767,

768, 769, 770, 771, 772, 773, 774, 775,

776, 777, 778, 779, 780, 781, 782, 783,

784, 785, 786, 787, 788, 789, 790, 791,

792, 793, 794, 795, 796, 797, 798, 799,

800, 801, 802, 803, 804, 805, 806, 807,

808, 809, 810, 811, 812, 813, 814, 815,

816, 817, 818, 819, 820, 821, 822, 823,

824, 825, 826, 827, 828, 829, 830, 831,

832, 833, 834, 835, 836, 837, 838, 839,

840, 841, 842, 843, 844, 845, 846, 847,

848, 849, 850, 851, 852, 853, 854, 855,

856, 857, 858, 859, 860, 861, 862, 863,

864, 865, 866, 867, 868, 869, 870, 871,

872, 873, 874, 875, 876, 877, 878, 879,

880, 881, 882, 883, 884, 885, 886, 887,

888, 889, 890, 891, 892, 893, 894, 895,

896, 897, 898, 899, 900, 901, 902, 903,

904, 905, 906, 907, 908, 909, 910, 911,

912, 913, 914, 915, 916, 917, 918, 919,

920, 921, 922, 923, 924, 925, 926, 927,

928, 929, 930, 931, 932, 933, 934, 935,

936, 937, 938, 939, 940, 941, 942, 943,

944, 945, 946, 947, 948, 949, 950, 951,

952, 953, 954, 955, 956, 957, 958, 959,

960, 961, 962, 963, 964, 965, 966, 967,

968, 969, 970, 971, 972, 973, 974, 975,

976, 977, 978, 979, 980, 981, 982, 983,

984, 985, 986, 987, 988, 989, 990, 991,

992, 993, 994, 995, 996, 997, 998, 999,

1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007,

1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015,

1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023,

1023, 1023, 1022, 1021, 1020, 1019, 1018, 1017,

1016, 1015, 1014, 1013, 1012, 1011, 1010, 1009,

1008, 1007, 1006, 1005, 1004, 1003, 1002, 1001,

1000, 999, 998, 997, 996, 995, 994, 993,

992, 991, 990, 989, 988, 987, 986, 985,

984, 983, 982, 981, 980, 979, 978, 977,

976, 975, 974, 973, 972, 971, 970, 969,

968, 967, 966, 965, 964, 963, 962, 961,

960, 959, 958, 957, 956, 955, 954, 953,

952, 951, 950, 949, 948, 947, 946, 945,

944, 943, 942, 941, 940, 939, 938, 937,

936, 935, 934, 933, 932, 931, 930, 929,

928, 927, 926, 925, 924, 923, 922, 921,

920, 919, 918, 917, 916, 915, 914, 913,

912, 911, 910, 909, 908, 907, 906, 905,

904, 903, 902, 901, 900, 899, 898, 897,

896, 895, 894, 893, 892, 891, 890, 889,

888, 887, 886, 885, 884, 883, 882, 881,

880, 879, 878, 877, 876, 875, 874, 873,

872, 871, 870, 869, 868, 867, 866, 865,

864, 863, 862, 861, 860, 859, 858, 857,

856, 855, 854, 853, 852, 851, 850, 849,

848, 847, 846, 845, 844, 843, 842, 841,

840, 839, 838, 837, 836, 835, 834, 833,

832, 831, 830, 829, 828, 827, 826, 825,

824, 823, 822, 821, 820, 819, 818, 817,

816, 815, 814, 813, 812, 811, 810, 809,

808, 807, 806, 805, 804, 803, 802, 801,

800, 799, 798, 797, 796, 795, 794, 793,

792, 791, 790, 789, 788, 787, 786, 785,

784, 783, 782, 781, 780, 779, 778, 777,

776, 775, 774, 773, 772, 771, 770, 769,

768, 767, 766, 765, 764, 763, 762, 761,

760, 759, 758, 757, 756, 755, 754, 753,

752, 751, 750, 749, 748, 747, 746, 745,

744, 743, 742, 741, 740, 739, 738, 737,

736, 735, 734, 733, 732, 731, 730, 729,

728, 727, 726, 725, 724, 723, 722, 721,

720, 719, 718, 717, 716, 715, 714, 713,

712, 711, 710, 709, 708, 707, 706, 705,

704, 703, 702, 701, 700, 699, 698, 697,

696, 695, 694, 693, 692, 691, 690, 689,

688, 687, 686, 685, 684, 683, 682, 681,

680, 679, 678, 677, 676, 675, 674, 673,

672, 671, 670, 669, 668, 667, 666, 665,

664, 663, 662, 661, 660, 659, 658, 657,

656, 655, 654, 653, 652, 651, 650, 649,

648, 647, 646, 645, 644, 643, 642, 641,

640, 639, 638, 637, 636, 635, 634, 633,

632, 631, 630, 629, 628, 627, 626, 625,

624, 623, 622, 621, 620, 619, 618, 617,

616, 615, 614, 613, 612, 611, 610, 609,

608, 607, 606, 605, 604, 603, 602, 601,

600, 599, 598, 597, 596, 595, 594, 593,

592, 591, 590, 589, 588, 587, 586, 585,

584, 583, 582, 581, 580, 579, 578, 577,

576, 575, 574, 573, 572, 571, 570, 569,

568, 567, 566, 565, 564, 563, 562, 561,

560, 559, 558, 557, 556, 555, 554, 553,

552, 551, 550, 549, 548, 547, 546, 545,

544, 543, 542, 541, 540, 539, 538, 537,

536, 535, 534, 533, 532, 531, 530, 529,

528, 527, 526, 525, 524, 523, 522, 521,

520, 519, 518, 517, 516, 515, 514, 513,

512, 511, 510, 509, 508, 507, 506, 505,

504, 503, 502, 501, 500, 499, 498, 497,

496, 495, 494, 493, 492, 491, 490, 489,

488, 487, 486, 485, 484, 483, 482, 481,

480, 479, 478, 477, 476, 475, 474, 473,

472, 471, 470, 469, 468, 467, 466, 465,

464, 463, 462, 461, 460, 459, 458, 457,

456, 455, 454, 453, 452, 451, 450, 449,

448, 447, 446, 445, 444, 443, 442, 441,

440, 439, 438, 437, 436, 435, 434, 433,

432, 431, 430, 429, 428, 427, 426, 425,

424, 423, 422, 421, 420, 419, 418, 417,

416, 415, 414, 413, 412, 411, 410, 409,

408, 407, 406, 405, 404, 403, 402, 401,

400, 399, 398, 397, 396, 395, 394, 393,

392, 391, 390, 389, 388, 387, 386, 385,

384, 383, 382, 381, 380, 379, 378, 377,

376, 375, 374, 373, 372, 371, 370, 369,

368, 367, 366, 365, 364, 363, 362, 361,

360, 359, 358, 357, 356, 355, 354, 353,

352, 351, 350, 349, 348, 347, 346, 345,

344, 343, 342, 341, 340, 339, 338, 337,

336, 335, 334, 333, 332, 331, 330, 329,

328, 327, 326, 325, 324, 323, 322, 321,

320, 319, 318, 317, 316, 315, 314, 313,

312, 311, 310, 309, 308, 307, 306, 305,

304, 303, 302, 301, 300, 299, 298, 297,

296, 295, 294, 293, 292, 291, 290, 289,

288, 287, 286, 285, 284, 283, 282, 281,

280, 279, 278, 277, 276, 275, 274, 273,

272, 271, 270, 269, 268, 267, 266, 265,

264, 263, 262, 261, 260, 259, 258, 257,

256, 255, 254, 253, 252, 251, 250, 249,

248, 247, 246, 245, 244, 243, 242, 241,

240, 239, 238, 237, 236, 235, 234, 233,

232, 231, 230, 229, 228, 227, 226, 225,

224, 223, 222, 221, 220, 219, 218, 217,

216, 215, 214, 213, 212, 211, 210, 209,

208, 207, 206, 205, 204, 203, 202, 201,

200, 199, 198, 197, 196, 195, 194, 193,

192, 191, 190, 189, 188, 187, 186, 185,

184, 183, 182, 181, 180, 179, 178, 177,

176, 175, 174, 173, 172, 171, 170, 169,

168, 167, 166, 165, 164, 163, 162, 161,

160, 159, 158, 157, 156, 155, 154, 153,

152, 151, 150, 149, 148, 147, 146, 145,

144, 143, 142, 141, 140, 139, 138, 137,

136, 135, 134, 133, 132, 131, 130, 129,

128, 127, 126, 125, 124, 123, 122, 121,

120, 119, 118, 117, 116, 115, 114, 113,

112, 111, 110, 109, 108, 107, 106, 105,

104, 103, 102, 101, 100, 99, 98, 97,

96, 95, 94, 93, 92, 91, 90, 89,

88, 87, 86, 85, 84, 83, 82, 81,

80, 79, 78, 77, 76, 75, 74, 73,

72, 71, 70, 69, 68, 67, 66, 65,

64, 63, 62, 61, 60, 59, 58, 57,

56, 55, 54, 53, 52, 51, 50, 49,

48, 47, 46, 45, 44, 43, 42, 41,

40, 39, 38, 37, 36, 35, 34, 33,

32, 31, 30, 29, 28, 27, 26, 25,

24, 23, 22, 21, 20, 19, 18, 17,

16, 15, 14, 13, 12, 11, 10, 9,

8, 7, 6, 5, 4, 3, 2, 1};

// lookup table to output a sawtoothwave

**int** sa[2048] = { 0, 1, 2, 3, 4, 5, 6, 7,

8, 9, 10, 11, 12, 13, 14, 15,

16, 17, 18, 19, 20, 21, 22, 23,

24, 25, 26, 27, 28, 29, 30, 31,

32, 33, 34, 35, 36, 37, 38, 39,

40, 41, 42, 43, 44, 45, 46, 47,

48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63,

64, 65, 66, 67, 68, 69, 70, 71,

72, 73, 74, 75, 76, 77, 78, 79,

80, 81, 82, 83, 84, 85, 86, 87,

88, 89, 90, 91, 92, 93, 94, 95,

96, 97, 98, 99, 100, 101, 102, 103,

104, 105, 106, 107, 108, 109, 110, 111,

112, 113, 114, 115, 116, 117, 118, 119,

120, 121, 122, 123, 124, 125, 126, 127,

128, 129, 130, 131, 132, 133, 134, 135,

136, 137, 138, 139, 140, 141, 142, 143,

144, 145, 146, 147, 148, 149, 150, 151,

152, 153, 154, 155, 156, 157, 158, 159,

160, 161, 162, 163, 164, 165, 166, 167,

168, 169, 170, 171, 172, 173, 174, 175,

176, 177, 178, 179, 180, 181, 182, 183,

184, 185, 186, 187, 188, 189, 190, 191,

192, 193, 194, 195, 196, 197, 198, 199,

200, 201, 202, 203, 204, 205, 206, 207,

208, 209, 210, 211, 212, 213, 214, 215,

216, 217, 218, 219, 220, 221, 222, 223,

224, 225, 226, 227, 228, 229, 230, 231,

232, 233, 234, 235, 236, 237, 238, 239,

240, 241, 242, 243, 244, 245, 246, 247,

248, 249, 250, 251, 252, 253, 254, 255,

256, 257, 258, 259, 260, 261, 262, 263,

264, 265, 266, 267, 268, 269, 270, 271,

272, 273, 274, 275, 276, 277, 278, 279,

280, 281, 282, 283, 284, 285, 286, 287,

288, 289, 290, 291, 292, 293, 294, 295,

296, 297, 298, 299, 300, 301, 302, 303,

304, 305, 306, 307, 308, 309, 310, 311,

312, 313, 314, 315, 316, 317, 318, 319,

320, 321, 322, 323, 324, 325, 326, 327,

328, 329, 330, 331, 332, 333, 334, 335,

336, 337, 338, 339, 340, 341, 342, 343,

344, 345, 346, 347, 348, 349, 350, 351,

352, 353, 354, 355, 356, 357, 358, 359,

360, 361, 362, 363, 364, 365, 366, 367,

368, 369, 370, 371, 372, 373, 374, 375,

376, 377, 378, 379, 380, 381, 382, 383,

384, 385, 386, 387, 388, 389, 390, 391,

392, 393, 394, 395, 396, 397, 398, 399,

400, 401, 402, 403, 404, 405, 406, 407,

408, 409, 410, 411, 412, 413, 414, 415,

416, 417, 418, 419, 420, 421, 422, 423,

424, 425, 426, 427, 428, 429, 430, 431,

432, 433, 434, 435, 436, 437, 438, 439,

440, 441, 442, 443, 444, 445, 446, 447,

448, 449, 450, 451, 452, 453, 454, 455,

456, 457, 458, 459, 460, 461, 462, 463,

464, 465, 466, 467, 468, 469, 470, 471,

472, 473, 474, 475, 476, 477, 478, 479,

480, 481, 482, 483, 484, 485, 486, 487,

488, 489, 490, 491, 492, 493, 494, 495,

496, 497, 498, 499, 500, 501, 502, 503,

504, 505, 506, 507, 508, 509, 510, 511,

512, 513, 514, 515, 516, 517, 518, 519,

520, 521, 522, 523, 524, 525, 526, 527,

528, 529, 530, 531, 532, 533, 534, 535,

536, 537, 538, 539, 540, 541, 542, 543,

544, 545, 546, 547, 548, 549, 550, 551,

552, 553, 554, 555, 556, 557, 558, 559,

560, 561, 562, 563, 564, 565, 566, 567,

568, 569, 570, 571, 572, 573, 574, 575,

576, 577, 578, 579, 580, 581, 582, 583,

584, 585, 586, 587, 588, 589, 590, 591,

592, 593, 594, 595, 596, 597, 598, 599,

600, 601, 602, 603, 604, 605, 606, 607,

608, 609, 610, 611, 612, 613, 614, 615,

616, 617, 618, 619, 620, 621, 622, 623,

624, 625, 626, 627, 628, 629, 630, 631,

632, 633, 634, 635, 636, 637, 638, 639,

640, 641, 642, 643, 644, 645, 646, 647,

648, 649, 650, 651, 652, 653, 654, 655,

656, 657, 658, 659, 660, 661, 662, 663,

664, 665, 666, 667, 668, 669, 670, 671,

672, 673, 674, 675, 676, 677, 678, 679,

680, 681, 682, 683, 684, 685, 686, 687,

688, 689, 690, 691, 692, 693, 694, 695,

696, 697, 698, 699, 700, 701, 702, 703,

704, 705, 706, 707, 708, 709, 710, 711,

712, 713, 714, 715, 716, 717, 718, 719,

720, 721, 722, 723, 724, 725, 726, 727,

728, 729, 730, 731, 732, 733, 734, 735,

736, 737, 738, 739, 740, 741, 742, 743,

744, 745, 746, 747, 748, 749, 750, 751,

752, 753, 754, 755, 756, 757, 758, 759,

760, 761, 762, 763, 764, 765, 766, 767,

768, 769, 770, 771, 772, 773, 774, 775,

776, 777, 778, 779, 780, 781, 782, 783,

784, 785, 786, 787, 788, 789, 790, 791,

792, 793, 794, 795, 796, 797, 798, 799,

800, 801, 802, 803, 804, 805, 806, 807,

808, 809, 810, 811, 812, 813, 814, 815,

816, 817, 818, 819, 820, 821, 822, 823,

824, 825, 826, 827, 828, 829, 830, 831,

832, 833, 834, 835, 836, 837, 838, 839,

840, 841, 842, 843, 844, 845, 846, 847,

848, 849, 850, 851, 852, 853, 854, 855,

856, 857, 858, 859, 860, 861, 862, 863,

864, 865, 866, 867, 868, 869, 870, 871,

872, 873, 874, 875, 876, 877, 878, 879,

880, 881, 882, 883, 884, 885, 886, 887,

888, 889, 890, 891, 892, 893, 894, 895,

896, 897, 898, 899, 900, 901, 902, 903,

904, 905, 906, 907, 908, 909, 910, 911,

912, 913, 914, 915, 916, 917, 918, 919,

920, 921, 922, 923, 924, 925, 926, 927,

928, 929, 930, 931, 932, 933, 934, 935,

936, 937, 938, 939, 940, 941, 942, 943,

944, 945, 946, 947, 948, 949, 950, 951,

952, 953, 954, 955, 956, 957, 958, 959,

960, 961, 962, 963, 964, 965, 966, 967,

968, 969, 970, 971, 972, 973, 974, 975,

976, 977, 978, 979, 980, 981, 982, 983,

984, 985, 986, 987, 988, 989, 990, 991,

992, 993, 994, 995, 996, 997, 998, 999,

1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007,

1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015,

1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023};

#endif /\* SINE\_H\_ \*/

keypad.h

/\*

\* keypad.h

\*

\* Created on: Apr 26, 2020

\* Author: nicks

\*/

#ifndef KEYPAD\_H\_

#define KEYPAD\_H\_

#include <stdint.h>

#define COL\_0 BIT0

#define COL\_1 BIT1

#define COL\_2 BIT2

#define ROW\_0 BIT3

#define ROW\_1 BIT4

#define ROW\_2 BIT5

#define ROW\_3 BIT6

**void** keypad\_init(**void**);

uint8\_t keypad\_Getkey(**void**);

#endif /\* KEYPAD\_H\_ \*/

keypad.c

#include "keypad.h"

#include "msp.h"

#include "delay.h"

**void** keypad\_init(**void**)

{

P2->SEL0 &= ~(ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3); // Sets P5.0-P5.2 to GPIO (COL)

P2->SEL1 &= ~(ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3);

P2->DIR &= ~(ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3); // Sets inputs with a pulldown resistor

P2->REN |= (ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3);

P2->OUT &= ~(ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3);

P5->SEL0 &= ~(COL\_0 | COL\_1 | COL\_2); // Sets P2.3-P2.6 to GPIO (ROW)

P5->SEL1 &= ~(COL\_0 | COL\_1 | COL\_2);

P5->DIR |= (COL\_0 | COL\_1 | COL\_2); // Sets P2.3-P2.6 to outputs

}

/\*

\* This is a non-blocking function to read the keypad.

\* If a key is pressed, it returns that key value 0-9. \* is 10, # is 12

\* If no key is pressed, it returns 0xFF

\* Port 4.0 - 4.3 are used as inputs and connected to the rows. Pull-down

\* resistors are enabled so when no key is pressed, these pins are pulled low

\*

\* The Port 4.4 - 4.6 are used as outputs that drives the keypad columns.

\* First all columns are driven high and the input pins are read. If no key is

\* pressed, they will read zero because of the pull-down resistors. If no key

\* is pressed, return 0xFF. If the value is non-zero, determine which key is

\* being pressed.

\* To determine which key is being pressed, the program proceeds to drive one

\* column high at a time and read the input pins (rows). Knowing which row is

\* high and which column is active, the program can decide which key is pressed

\*

\*/

uint8\_t keypad\_Getkey(**void**)

{

uint8\_t row, col, key;

/\* check to see any key pressed \*/

P5->OUT |= (COL\_0 | COL\_1 | COL\_2); // drive all column pins high

\_delay\_cycles(25); // wait for signals to settle

row = P2->IN & (ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3); // read all row pins

**if** (row == 0) // if all rows are low, no key pressed

**return** 0xFF;

/\* If a key is pressed, it gets here to find out which key.

\* It activates one column at a time and reads the input to see

\* which row is active. \*/

**for** (col = 0; col < 3; col++)

{

// zero cols

P5->OUT &= ~(COL\_0 | COL\_1 | COL\_2);

// shift a 1 into the correct column depending on which to turn on

P5->OUT |= (COL\_0 << col);

\_delay\_cycles(25); // wait for signals to settle

row = P2->IN & (ROW\_0 | ROW\_1 | ROW\_2 | ROW\_3); // mask only the row pins

**if** (row != 0) **break**; // if the input is non-zero, key detected

}

P5->OUT &= ~(COL\_0 | COL\_1 | COL\_2); // drive all columns low

**if** (col == 3) **return** 0xFF; // if we get here, no key was detected

// rows are read in binary, so powers of 2 (1,2,4,8)

**if** (row == 8) row = 1;

**if** (row == 16) row = 2;

**if** (row == 32) row = 3;

**if** (row == 64) row = 4; //

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* IF MULTIPLE KEYS IN A COLUMN ARE PRESSED THIS WILL BE INCORRECT \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// calculate the key value based on the row and columns where detected

**if** (col == 0) key = row\*3 - 2;

**if** (col == 1) key = row\*3 - 1;

**if** (col == 2) key = row\*3;

//if (key == 11) key = 0; // fix for 0 key

**return** key;

}

spi.h

/\*

\* spi.h

\*

\* Created on: May 8, 2020

\* Author: nicks

\*/

#ifndef SPI\_H\_

#define SPI\_H\_

#include "msp.h"

#define SPI\_CS\_PIN BIT6

#define CONTROL\_BITS (0b01110000)

#define DAC\_DATA\_MASK (0x0FFF)

**void** dac\_init();

**void** spi\_transmit(**const** uint16\_t data);

**void** cs\_low();

**void** cs\_high();

#endif /\* SPI\_H\_ \*/

spi.c

/\*

\* SPI.c

\*

\* Created on: May 8, 2020

\* Author: nicks

\*/

#include "spi.h"

**void** dac\_init(**void**)

{

P3-> SEL0 &= ~(SPI\_CS\_PIN);

P3-> SEL1 &= ~(SPI\_CS\_PIN);

P3 ->DIR |= (SPI\_CS\_PIN);

cs\_high();

P1->SEL0 |= BIT5 | BIT6 | BIT7; // Set P1.5, P1.6, and P1.7 as

// SPI pins functionality

P2->DIR |= BIT0 | BIT1 | BIT2; // set as output for LED

EUSCI\_B0->CTLW0 |= EUSCI\_B\_CTLW0\_SWRST; // Put eUSCI state machine in reset

EUSCI\_B0->CTLW0 = EUSCI\_B\_CTLW0\_SWRST | // keep eUSCI in reset

EUSCI\_B\_CTLW0\_MST | // Set as SPI master

// EUSCI\_B\_CTLW0\_CKPH |

EUSCI\_B\_CTLW0\_SYNC | // Set as synchronous mode

EUSCI\_B\_CTLW0\_CKPL | // Set clock polarity high

EUSCI\_B\_CTLW0\_SSEL\_\_SMCLK | // SMCLK

EUSCI\_B\_CTLW0\_MSB; // MSB first

EUSCI\_B0->BRW = 0x02; // no div - fBitClock = fBRCLK/(UCBRx)

EUSCI\_B0->CTLW0 &= ~EUSCI\_B\_CTLW0\_SWRST; // Initialize USCI state machine

EUSCI\_B0->IE |= EUSCI\_B\_IE\_RXIE; // Enable RX interrupt

}

**void** cs\_high()

{

P3 -> OUT |= SPI\_CS\_PIN;

}

**void** cs\_low()

{

P3 -> OUT &= ~SPI\_CS\_PIN;

}

**void** spi\_transmit( **const** uint16\_t val)

{

cs\_low();

**while**(!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG));

EUSCI\_B0->TXBUF = (CONTROL\_BITS | ((val & 0x3C0) >> 6));

**while**(!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG));

EUSCI\_B0->TXBUF = (val & 0xFF) << 2;

**while**(!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_RXIFG));

cs\_high();

}

**References**

**MCP4921 datasheet:** [**http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf**](http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf)

**MSP432 Reference Manual:** <https://canvas.calpoly.edu/courses/16731/modules/items/53061>

**Video Links:**

[Basic Functionality](https://drive.google.com/open?id=1Tk9FRlco7E0SFSkpEmyhhUQOJveE_AlV)

[Extra Functionality](https://drive.google.com/open?id=1Tjz02zH1x_WN-oVObda-nAZaUPtjJzyL)