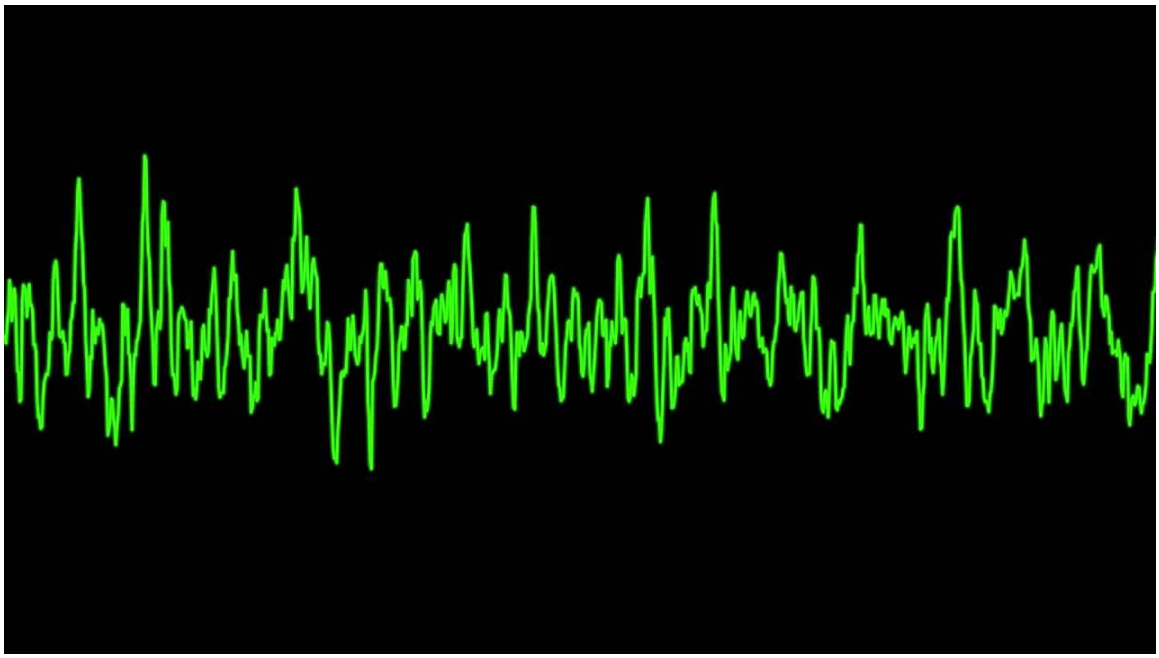

Project 2

FUNCTION GENORATOR

Kyle Rosenthal | CPE 329-05-Gerfen | 4/17/2017 | Spring 2017

Purpose

The purpose of this project was to develop a better understanding of timers and interrupts, off chip communication through SPI, and digital to analog conversion. Though the designing and building of a waveform generator. In addition the project included though supplemental research, a better understand of pseudo random number generators and how they can be used to simulate noise. The waveform generator parts will also refresh the understanding of getting keypad input as well as setting up a state machine.

LINK TO VIDEO

https://www.youtube.com/watch?v=U7773u7U4y4

System Requirements

- System shall have a keypad.
    - The '1' button shall decrease frequency by 100 Hz to a minimum of 100 Hz.
    - The '2' button shall set frequency to 300 Hz.
    - The '3' button shall increase frequency by 100 Hz to a maximum of 500 Hz.
    - The '5' button shall set the waveform mode to noise.
    - The '6' button shall cycle the pulse multiplier between 0 and 9.
    - The '7' button shall set the waveform mode to square wave.
    - The '8' button shall set the waveform mode to sine wave.
    - The '9' button shall set the waveform mode to saw wave.
    - The '*' button shall decrease duty cycle by 10% to a minimum of 10%.
    - The '0' button shall set duty cycle to 50%.
    - The '#' button shall increase duty cycle by 10% to a maximum of 90%.
- System shall have a DAC outputting waveforms as specified by the keypad.
    - The DAC shall output the wave as set by the keypad.
    - The DAC shall adjust frequency as set by the keypad.
    - The DAC shall output pseudo-noise when in noise waveform mode.
    - The DAC shall adjust the duty cycle of the square wave as set by the keypad.
    - The DAC shall pause each delay each cycle of a wave by the period multiplied by the pulse multiplier as set by the keypad.
- Shall operate at 24 MHz

System Specifications

Component	Spec	Value
MSP432	Model	MSP432P401R
	Frequency	24 MHz
	Interrupts	Enabled
	Input Power	5 V
4921 SPI DAC	Model	MCP4921
	V[DD]	3.3 V
	V[ref]	3.3 V
	V[SS]	0 V
	LDAC	0 V
	Tested Output Impedance	1M ohm
Keypad	Operation Mode	Pull down
	Buttons	12

WAVEFORM ACCURACIES

	Frequency				
	100 Hz	200 Hz	300 Hz	400 Hz	500 Hz
Square	99.88	199.7	298.3	399.4	491.1
Sine	99.88	199.75	298.3	399.4	491.1
Saw	99.86	199.75	298.4	399.4	491.1
	Duty Cycle	Measured		Vpp	
Square	10%	9.76		2.87	
	20%	19.9			
	30%	29.68		Measured with 1M Ohm output impedance	
	40%	39.83			
	50%	49.99			
	60%	59.75			
	70%	69.91			
	80%	79.68			
	90%	89.84			

System Architecture

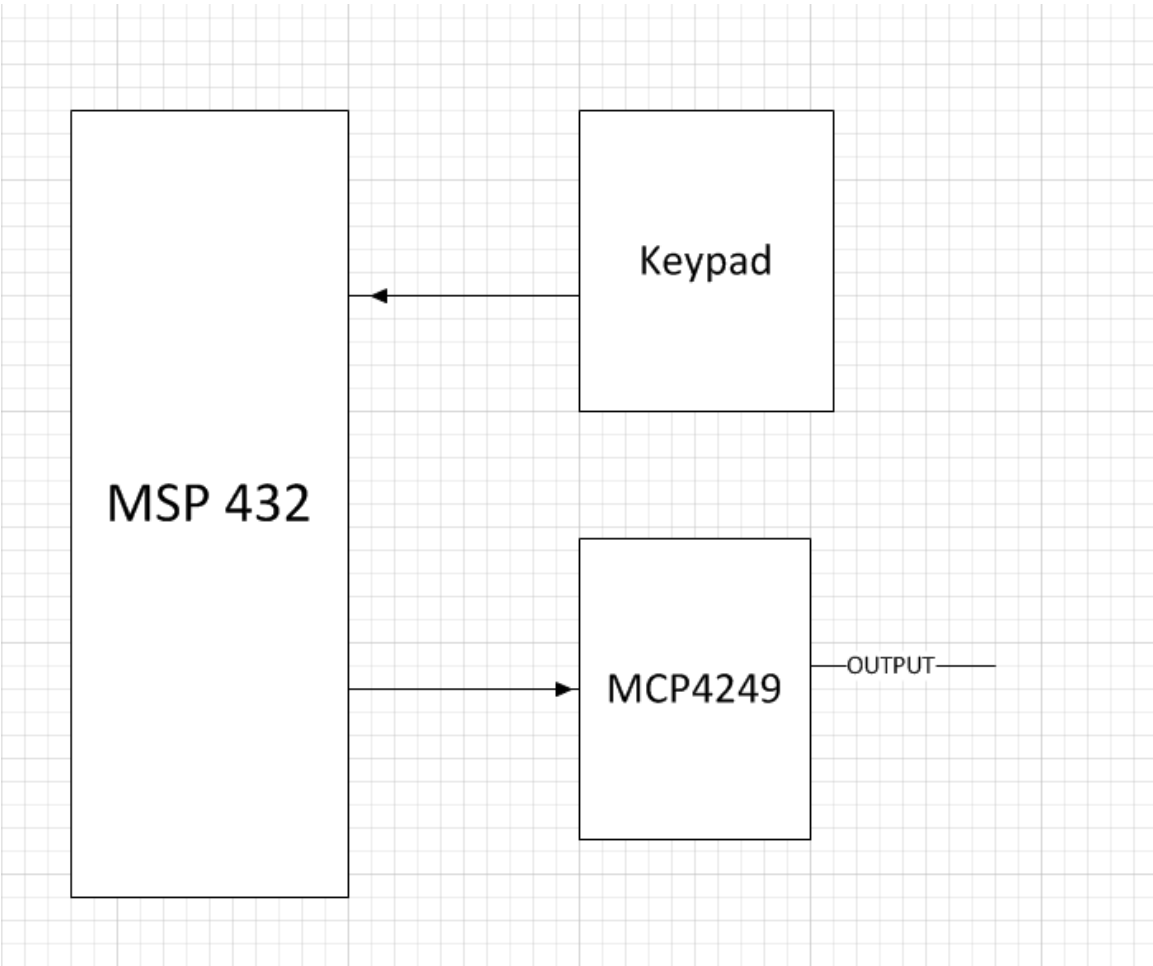

Figure 1 – System Architecture Diagram

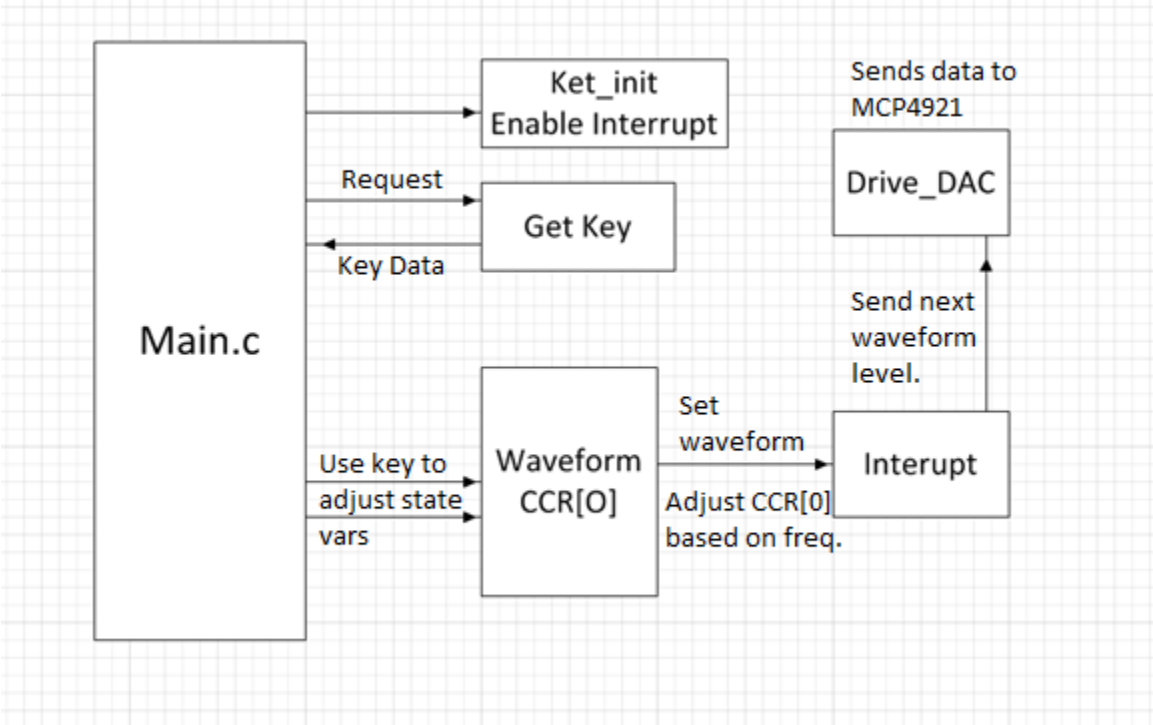

Figure 2 – System Software Diagram

Component Design

All setting are default settings of MSP432P401R Launchpad running at 24 MHz with interrupts enabled.

- void main()
  - Set DCO to 24 MHz
  - Setup I/O pins
  - Enable interrupts
  - Get keypad state and adjust state
- void TA0_0_IRQHandler(void)
  - Sets waveform to next item in wave array
  - If in noise mode randomly generates a new noise value
  - Reset interrupt
- void Drive_DAC(void)
  - Provided function
  - Removed delay – did not affect wave and wasted cpu cycles.
  - Sends DAC a value to output
- delay() and delayMs(int)
  - Generate short delays that would be more expensive to set up a interrupt for.

- Key Get_key()
    - Polls the keypad.
- Key_init()
    - Sets pins for keypad as pull down inputs.

SCHEMATIC

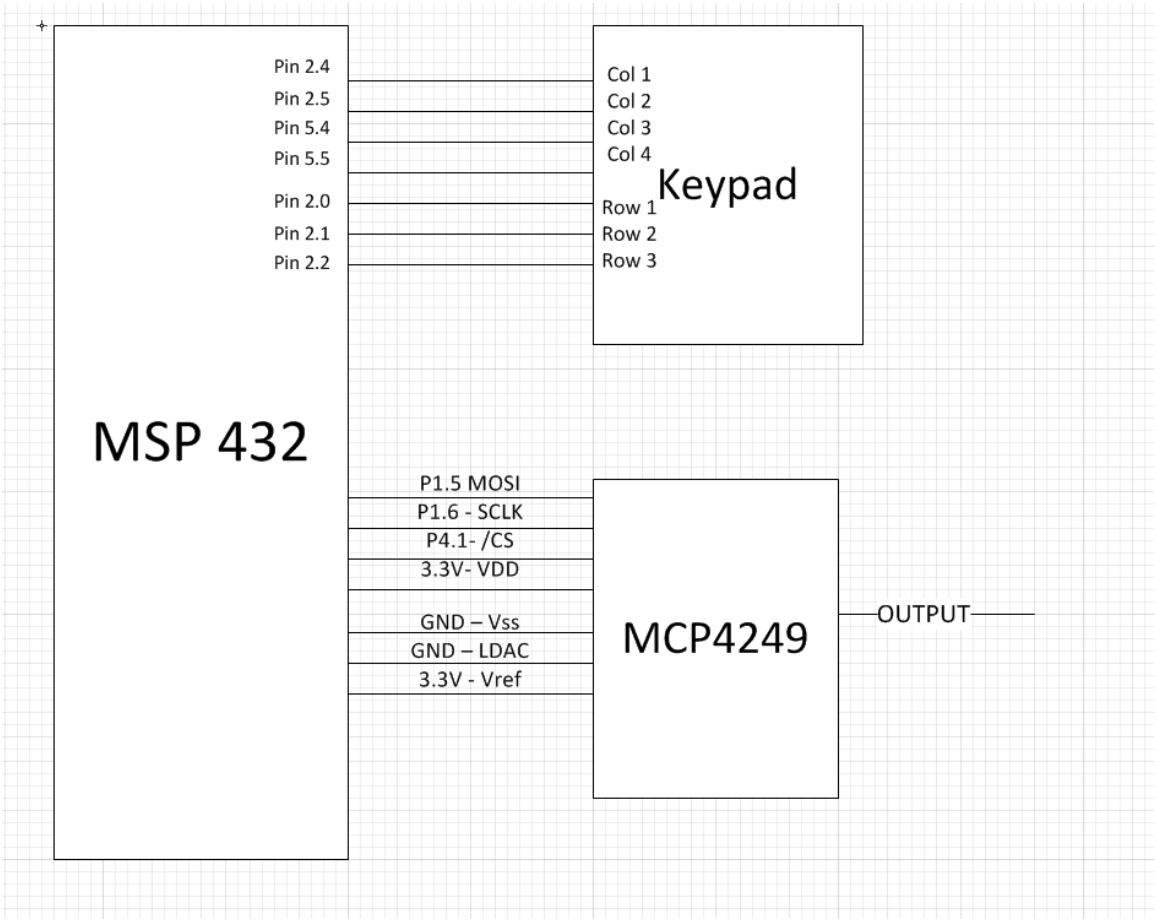

Figure 3 – Schematic Diagram

Bill of Materials

	Item #	Part #	Supplier	Quantity	Price Ea	Total Price $
MSP432 Launchpad	1	MSP432P401R	Digikey	1	13.03	
7 pin, 12 button keypad	2		Digikey	1	3.55	3.55
Jumper cables	3	0	Amazon	20	.01	.20
4921 SPI DAC	4	MCP4921	Digikey	1	1.69	1.69
Breadboard	5	352	Pololu	1	3.97	3.97
Total						9.41

System Integration

The system was designed in two parts the keypad with its associated state machine, and the timer interrupt system that controlled the DAC. These parts interacted through one waveform variable pointing to the correct waveform, and the keypad state machine adjusting the CCR[0] values. The timer was chosen to operate in 'UP' mode, as the CCR[0] would not need to change on the interrupt that was and save a few cycles there, allowing greater resolution. A challenge approached when designing the system was the somewhat hard to predict outcome when the CCR[0] values were set too low and the CPU would be unable to fully exit the interrupt, as the next one was already queued up.

When setting up the built in parameters the scope was set up in such a way as to measure all the required values automatically so the values could quickly be adjusted to the proper numbers. See Figure 4 for an example of this.

The noise generation was another interesting challenge because it needed to be computationally cheap in order to keep the same resolution and not block out the keypad. A few of in house algorithms were tested before deciding on the linear congruential generator, a very fast generator used by many software solutions.

The pulse multiplier had some initial difficulties as the additional code threw off the timing for the 500 Hz interrupt and was locking the system. The code had to be refactored to be more optimized.

Answers to Questions

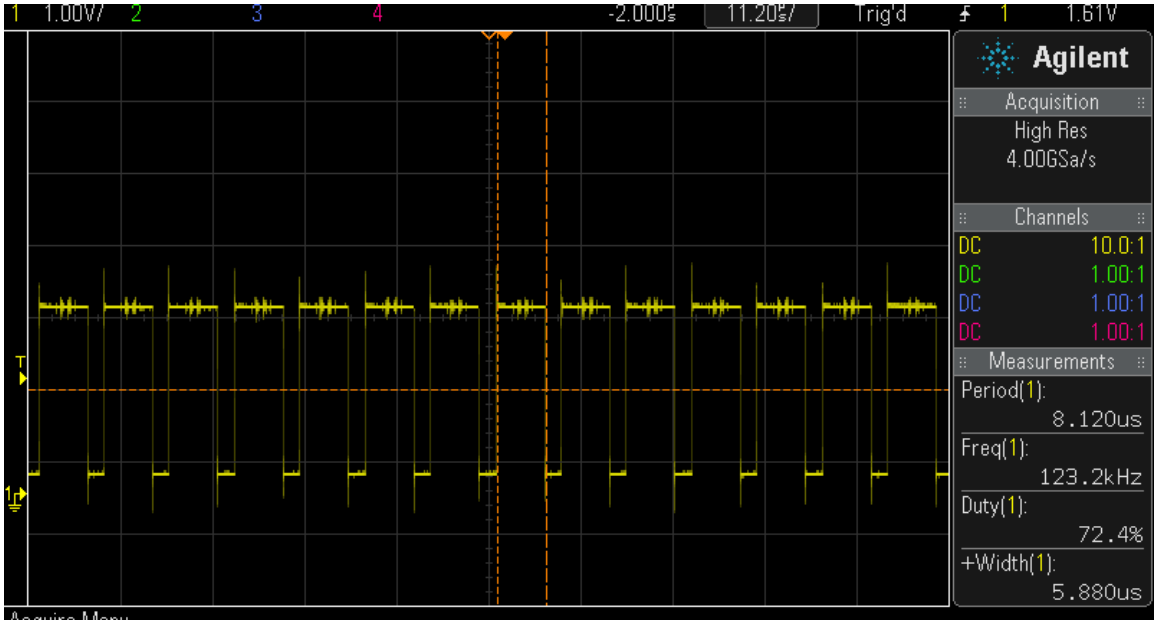

Figure 4 – 500 MHz wave interrupt timing.

Figure 4 shows how the 500 MHz wave was optimized to utilize the maximum amount of resolution by setting the CCR[0] value to just 47. Drive_DAC's delay was removed to help achieve this, as well as any computation of the waves themselves. The MCP4921 Spec sheet lists a maximum operating frequency of 20 MHz, and because it takes more than 2 cycles per interrupt there is no effectively maximum bound for the MSP432 running at 24 MHz to send DAC level to the MCP4921

Conclusion

In conclusion, a function waveform generator was design, built, and programming using the MSP432, MCP4921, and a keypad. This project stressed the importance of timing and for embedded systems using slower clocks than modern PC, every cycle can matter. Through careful setup of the interrupt function we were able to get 256 points of resolution per period. This results in a very smoothing looking waveform on the scope and comfortably could be used as a sine wave. The additional feature of pulsing allows systems to react to a wave then settle down internal circuits to close to initial conditions, a feature that could be very useful in industry. Noise simulation allows noise to be injected into a system, ensuring that the system can safely handle random data, even if that data is within safe voltage parameters. The design could be improved by shifting to 48 MHz, although that would introduce more complexities as this could operate faster than the DAC could shift between levels. Additionally looking at the spec sheet of the MCP4921 it described the LDAC pin and how it could be used to double buffer inputs. Further research may

reveal applicable uses to the waveform generator, perhaps by sending multiple levels per interrupt and further increasing the resolution. Another advancement this project could benefit from would be the precomputation on the fly of the predefined levels for the waves. While this would increase the latency between each type of waveform, it would allow a greater flexibility in the properties of the waves.

In building this project it is recommended that one first gets a good understanding of how low one can set the CCR[0] values in order to maximize resolution. This means learning what pieces of code take more or less cycles to complete. In addition, take the spec sheets with a grain of salt, as sometimes one can go above or below the recommended parameters to get better results. This is seen in the code below when the Drive_DAC delay was removed. Finally, when adding additionally functionality, take into account the careful timing that the interrupts use and be wary that added code in the wrong location could throw that off.

Appendices

REFERANCES

- CPE 329 - Project 2 – Function Genorator v0.02 - S2017
- MSP432 - Technical Reference Manual File
- MCP4921 – Spec Sheet
- Schematic created with: http://www.schematics.com/

CODE

main.c

```
//******************************************************************************
//
// MSP432 main.c - Project 2
//
//******************************************************************************

#include "msp.h"
#include "proj2.h"
#include <stdlib.h>

void Drive_DAC(unsigned int level);

volatile unsigned int TempDAC_Value = 0;

int *waveform;
int freq = FREQ_100_Hz; // in 100Hz
int fArr[6] = {470/2, 470/4, 470/6, 470/8, 470/10, 4700}; //CCRO values
int pulseMode;
int noise[256];
```

```
void main(void)
    {

    WDTCTL = WDTPW | WDTHOLD;           // Stop watchdog timer

    // DCO = 24 MHz, SMCLK and MCLK = DCO
    CS->KEY = CS_KEY_VAL;
    CS->CTL0 = 0;
    CS->CTL0 = CS_CTL0_DCORSEL_4;   // DCO = 24 MHz
    CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3;
    CS->KEY = 0;

    // Configure port bits for SPI
    P4->DIR |= BIT1;                    // Will use BIT4 to activate /CE on the DAC
    P1SEL0 |= BIT6 + BIT5;              // Configure P1.6 and P1.5 for UCB0SIMO and UCB0CLK
    P1SEL1 &= ~(BIT6 + BIT5);           //

    // SPI Setup
    EUSCI_B0->CTLW0 |= EUSCI_B_CTLW0_SWRST;   // Put eUSCI state machine in reset

    EUSCI_B0->CTLW0 = EUSCI_B_CTLW0_SWRST |   // Remain eUSCI state machine in reset
                      EUSCI_B_CTLW0_MST   |   // Set as SPI master
                      EUSCI_B_CTLW0_SYNC  |   // Set as synchronous mode
                      EUSCI_B_CTLW0_CKPL  |   // Set clock polarity high
                      EUSCI_B_CTLW0_MSB;      // MSB first

    EUSCI_B0->CTLW0 |= EUSCI_B_CTLW0_SSEL__SMCLK; // SMCLK
    EUSCI_B0->BRW = 0x01;                         // divide by 16, clock = fBRCLK/(UCBRx)
    EUSCI_B0->CTLW0 &= ~EUSCI_B_CTLW0_SWRST;      // Initialize USCI state machine, SPI
                                                  // now waiting for something to
                                                  // be placed in TXBUF

    EUSCI_B0->IFG |= EUSCI_B_IFG_TXIFG;  // Clear TXIFG flag

    //setup interupt
    TIMER_A0->CCTL[0] = TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
    TIMER_A0->CCR[0] = 470;

    TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK | // SMCLK, continuous mode
            TIMER_A_CTL_MC__UP |
            TIMER_A_CTL_ID__4;

    //SCB->SCR |= SCB_SCR_SLEEPONEXIT_Msk;    // Enable sleep on exit from ISR

    // Enable global interrupt
    __enable_irq();

    NVIC->ISER[0] = 1 << ((TA0_0_IRQn) & 31);

    P6->DIR |= BIT0;

    waveform = duty[3];
    freq = FREQ_100_Hz;
    TIMER_A0->CCR[0] = fArr[freq];
    Key key = NONE;
    Key_init();
    int dutyNum = 4;
    while (1) {
        while (key != NONE) key = Get_key();
        while (key == NONE) key = Get_key();
        switch (key) {
            case K1:
                freq--;
                if (freq < FREQ_100_Hz) freq = FREQ_100_Hz;
                if (waveform == noise) freq = FREQ_100_Hz;
                TIMER_A0->CCR[0] = fArr[freq];
                break;
```

```
            case K2:
                freq = FREQ_300_Hz;
                if (waveform == noise) freq = FREQ_100_Hz;
                TIMER_A0->CCR[0] = fArr[freq];
                break;
            case K3:
                freq++;
                if (freq > FREQ_500_Hz) {
                    freq = FREQ_500_Hz;
                }
                if (waveform == noise) freq = FREQ_100_Hz;
                TIMER_A0->CCR[0] = fArr[freq];
                break;
            case K4:
                pulseMode = 0;
                break;
            case K5:
                TIMER_A0->CCR[0] = fArr[0];
                waveform = noise;
                break;
            case K6:
                if (waveform != noise)
                    pulseMode = (pulseMode + 1) % 10;
                break;
            case K7:
                dutyNum = 4;
                waveform = duty[4];
                pulseMode = 0;
                break;
            case K8:
                pulseMode = 0;
                waveform = sin;
                break;
            case K9:
                pulseMode = 0;
                waveform = saw;
                break;
            case K_STAR:
                if (waveform != saw && waveform != sin && waveform != noise)
                {
                    dutyNum = dutyNum + 1;
                    if (dutyNum > 8) dutyNum = 8;
                    waveform = duty[dutyNum];
                }
                break;
            case K0:
                if (waveform != saw && waveform != sin && waveform != noise)
                {
                    dutyNum = 4;
                    waveform = duty[4];
                }
                break;
            case K_POUND:
                if (waveform != saw && waveform != sin)
                {
                    dutyNum = dutyNum - 1;
                    if (dutyNum < 0) dutyNum = 0;
                    waveform = duty[dutyNum];
                }
                break;
        }


    }
}

void TA0_0_IRQHandler(void) {
```

```
    static char step = 0;
    static int pulseWait = 0;
    static int rand = 35;
    //P6->OUT |= BIT0;
    TIMER_A0->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
    if (waveform != noise) {
        Drive_DAC(waveform[step]);
        if (step == 0 && pulseWait < (256 * pulseMode))  {
                pulseWait++;
        } else {
            step = step + 1;
            pulseWait = 0;
        }
    } else {
        //rand ^= rand << 2; //alt PRNG
        //rand ^= rand >> 7;
        //rand ^= rand << 5;
        rand = (4321*rand + 420);
        Drive_DAC(rand % 1919);
    }
    //P6->OUT &= ~BIT0;
}


void Drive_DAC(unsigned int level){
  unsigned int DAC_Word = 0;
  //int i;

  DAC_Word = (0x1000) | (level & 0x0FFF);   // 0x1000 sets DAC for Write
                                            // to DAC, Gain = 2, /SHDN = 1
                                            // and put 12-bit level value
                                            // in low 12 bits.
  P4->OUT &= ~BIT1;                                  // Clear P4.1 (drive /CS low on DAC)
                                                     // Using a port output to do this for now
  EUSCI_B0->TXBUF = (unsigned char) (DAC_Word >> 8);  // Shift upper byte of DAC_Word
                                                     // 8-bits to right
  while (!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));              // USCI_A0 TX buffer ready?
  EUSCI_B0->TXBUF = (unsigned char) (DAC_Word & 0x00FF);     // Transmit lower byte to DAC
  while (!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));      // Poll the TX flag to wait for completion
  //for(i = 200; i > 0; i--);                          // Delay 200 16 MHz SMCLK periods
                                                    //to ensure TX is complete by SIMO
  P4->OUT |= BIT1;                                  // Set P4.1   (drive /CS high on DAC)
  return;
}

void delay(){}

void delayMs(int n) {
    int i, j;
    for (j = 0; j < n; j++)
        for (i = 750; i > 0; i--);      /* Delay */
}

Key Get_key(void) {
    Key retMe = NONE;

    P2->OUT = 0xF7;

    //row 1
    P2->DIR = 0x00;
    P2->DIR |= 0xF0;
    P2->OUT &= ~0x10;
    delay();
    unsigned char in = P2->IN & 0x07;
    P2->OUT |=0x10;
    if (in == 0x6)
        retMe = K3;
```

```
    if (in == 0x5)
        retMe = K2;
    if (in == 0x3)
        retMe = K1;

    if (retMe != NONE) return retMe;
    // row 2


    P2->OUT &= ~0x20;
    delay();
    in = P2->IN & 0x07;
    P2->OUT |=0x20;
    if (in == 0x6)
        retMe = K6;
    if (in == 0x5)
        retMe = K5;
    if (in == 0x3)
        retMe = K4;

    //row 3

    P5->DIR = 0x00;
    P5->DIR |= 0x10;
    P5->OUT &= ~0x10;
    delay();
    in = P2->IN & 0x07;
    P5->OUT |=0x10;
    if (in == 0x6)
        retMe = K9;
    if (in == 0x5)
        retMe = K8;
    if (in == 0x3)
        retMe = K7;

    //row 4
    P5->DIR = 0x00;
    P5->DIR |= 0x20;
    P5->OUT &= ~0x20;
    delay();
    in = P2->IN & 0x07;
    P5->OUT |=0x20;
    if (in == 0x6)
        retMe = K_POUND;
    if (in == 0x5)
        retMe = K0;
    if (in == 0x3)
        retMe = K_STAR;


    P2->OUT = 0xF7;
    P2->DIR = 0x00;


    delayMs(1);
    return retMe;
}

/* P2(0:2) is cols / input; P2(4:7) is rows/output */
void Key_init(void) {
    P2->DIR = 0x00;
    P2->REN = 0x70;
    //P6->DIR &= ~BIT5;
    //P6->REN |= BIT5;

}
```

proj2.h

```
/*
 * proj2.h
 *
 *  Created on: Apr 28, 2017
 *      Author: kmrosent
 */

#ifndef PROJ2_H_
#define PROJ2_H_


#define RS 1     /* P4.0 mask */
#define RW 2     /* P4.1 mask */
#define EN 4     /* P4.2 mask */

#define CYCLES_PER_LOOP 11
#define SETUP_CYCLES 5
#define FREQ_32_KHz        32768
#define FREQ_1_5_MHz     1500000
#define FREQ_3_MHz       3000000
#define FREQ_6_MHz       6000000
#define FREQ_12_MHz     12000000
#define FREQ_24_MHz     24000000
#define FREQ_48_MHz     48000000

#define FREQ_100_Hz     0
#define FREQ_200_Hz     1
#define FREQ_300_Hz     2
#define FREQ_400_Hz     3
#define FREQ_500_Hz     4


typedef unsigned long ul;

void delayMs(int n);
void delay_ms(ul ms);
void delay_ns(ul ns);

void Key_init(void);
typedef enum { K1 = 0, K2, K3, K4, K5, K6, K7, K8, K9, K_STAR, K0, K_POUND,
NONE } Key;
Key Get_key(void);


//pregenorated waves

//sin wave precalulated with 128 steps
int sin[256] =
{960,984,1007,1031,1054,1078,1101,1124,1147,1170,1193,1216,1239,
                1261,1283,1305,1327,1349,1370,1392,1413,1433,1454,1474,1493,
                1513,1532,1551,1569,1587,1605,1622,1639,1655,1671,1687,1702,
                1717,1731,1745,1758,1771,1783,1795,1807,1817,1828,1838,1847,
```

```
                    1856,1864,1872,1879,1885,1891,1897,1902,1906,1910,1913,1915,
                    1917,1919,1920,1920,1920,1919,1917,1915,1913,1910,1906,1902,
                    1897,1891,1885,1879,1872,1864,1856,1847,1838,1828,1817,1807,
                    1795,1783,1771,1758,1745,1731,1717,1702,1687,1671,1655,1639,
                    1622,1605,1587,1569,1551,1532,1513,1493,1474,1454,1433,1413,
                    1392,1370,1349,1327,1305,1283,1261,1239,1216,1193,1170,1147,
                    1124,1101,1078,1054,1031,1007,984,960,936,913,889,866,842,
                    819,796,773,750,727,704,681,659,637,615,593,571,550,528,507,
                    487,466,446,427,407,388,369,351,333,315,298,281,265,249,233,
                    218,203,189,175,162,149,137,125,113,103,92,82,73,64,56,48,41,
                    35,29,23,18,14,10,7,5,3,1,0,0,0,1,3,5,7,10,14,18,23,29,35,41,
                    48,56,64,73,82,92,103,113,125,137,149,162,175,189,203,218,233,
                    249,265,281,298,315,333,351,369,388,407,427,446,466,487,507,
                    528,550,571,593,615,637,659,681,704,727,750,773,796,819,842,
                    866,889,913,936};

//saw wave precalulated with 128 steps
int saw[256] = {0,7,15,22,30,37,45,52,60,67,75,82,90,97,105,112,120,127,135,
                    142,150,157,165,172,180,187,195,202,210,217,225,232,240,247,
                    255,262,270,277,285,292,300,307,315,322,330,337,345,352,360,
                    367,375,382,390,397,405,412,420,427,435,442,450,457,465,472,
                    480,487,495,502,510,517,525,532,540,547,555,562,570,577,585,
                    592,600,607,615,622,630,637,645,652,660,667,675,682,690,697,
                    705,712,720,727,735,742,750,757,765,772,780,787,795,802,810,
                    817,825,832,840,847,855,862,870,877,885,892,900,907,915,922,
                    930,937,945,952,960,967,974,982,989,997,1004,1012,1019,1027,
                    1034,1042,1049,1057,1064,1072,1079,1087,1094,1102,1109,1117,
                    1124,1132,1139,1147,1154,1162,1169,1177,1184,1192,1199,1207,
                    1214,1222,1229,1237,1244,1252,1259,1267,1274,1282,1289,1297,
                    1304,1312,1319,1327,1334,1342,1349,1357,1364,1372,1379,1387,
                    1394,1402,1409,1417,1424,1432,1439,1447,1454,1462,1469,1477,
                    1484,1492,1499,1507,1514,1522,1529,1537,1544,1552,1559,1567,
                    1574,1582,1589,1597,1604,1612,1619,1627,1634,1642,1649,1657,
                    1664,1672,1679,1687,1694,1702,1709,1717,1724,1732,1739,1747,
                    1754,1762,1769,1777,1784,1792,1799,1807,1814,1822,1829,1837,
                    1844,1852,1859,1867,1874,1882,1889,1897,1904,1912};

//pregen'd duty cycles
int duty[9][256] = {
 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,1919,1919,1919,1919,1919,1919,1919,
```

```
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919 },

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},
```

```
 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
  1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,1919,
 1919,1919,1919,1919,1919,1919,1919,1919,1919}};


#endif /* PROJ2_H_ */
```