Project 45

bluetooth soundbox

Kyle Rosenthal | CPE 329-05-Gerfen | 6/12/2017 | Spring 2017

## Link to Video

# Purpose

# System Requirements

# System Specifications

|  |  |  |
| --- | --- | --- |
| Component | Spec | Value |
| MSP432 | Model | MSP432P401R |
|  | Frequency | 24 MHz |
|  | Interrupts | Enabled |
|  | Input Power | 5 V |
| UART | Baud Rate | 9600 |
|  | UART Data Bits | 8 |
|  | UART Mode | 8N1 |

*Table 1 – System Specifications.*

# System Architecture

Figure 1 – System Architecture Diagram

Figure 2 – System Software Diagram

# Component Design

All setting are default settings of MSP432P401R Launchpad running at 3 MHz with interrupts enabled. The five

* main.c
  + main()
    - calls all the setup functions
* uart.c
  + Setup\_UART()
    - Configures the UART
    - Enables interrupt
  + UART0TX(char)
    - Sends a single char
  + UART\_Strign(char \*, int)
    - Sends a string
  + IRQHandler()
    - Sends commands to adjust wave parameters based on input character.
* timer.c
  + Timer\_Setup()
    - Enables the timer and turns on the interrupt.
  + IRQ\_Handler()
    - Calls Sound\_Go from sound.h

TODO sound.h dac.h

## Schematic

Figure 3 – Schematic Diagram

# Bill of Materials

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Item # | Part # | Supplier | Quantity | Price Ea | Total Price $ |
| MSP432 Launchpad | 1 | MSP432P401R | Digikey | 1 | 13.03 | 13.03 |
| Jumper cables | 2 | 0 | Amazon | 3 | .01 | .03 |
| Wire | 3 |  | Amazon |  |  |  |
| 10k Resitor | 4 |  | Amazon |  | .1 |  |
| 50k Ohm Potentiometer | 5 |  | Amazon |  | .25 |  |
| Breadboard | 6 | 352 | Pololu | 1 | 3.97 | 3.97 |
| 1000 uF Capacitor | 7 | 0 | IEEE |  | .25 |  |
| 100 uF Capacitor | 8 | 0 |  |  | .25 |  |
| 10 uF Capacitor | 9 | 0 |  |  | .25 |  |
| 1 uF Capacitor | 10 | 0 |  |  | .25 |  |
| 1 nF Capacitor | 11 | 0 |  |  | .25 |  |
| AA Battery Housing | 12 |  |  |  |  |  |
| AA Battery | 13 |  | Amazon |  |  |  |
| 5V Rechargeable Battery | 14 |  |  |  |  |  |
| Enclosure | 15 |  |  |  |  |  |
| LM386 Amp |  |  | IEEE | 1 | 2.25 | 2.25 |
| 4921 SPI DAC |  | 4921 SPI DAC | DigiKey | 1 | 1.69 | 1.69 |
| Total |  |  |  |  |  | 17.03 |
|  |  |  |  |  |  |  |

*Table 2 – Bill of Materials.*

# System Integration

# Conclusion

# Appendices

## Referances

* CPE 329 - Project 45 – Final Project - S2017
* [MSP432 - Technical Reference Manual File](http://www.ti.com/general/docs/litabsmultiplefilelist.tsp?literatureNumber=slau356f)
* [MCP4921 – Spec Sheet](http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf)
* [LM386 - DataSheet](http://www.ti.com/lit/ds/symlink/lm386.pdf)
* Schematic created with: http://www.draw.io/

## Code

### main.c

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

//

// MSP432 main.c template - Empty main

//

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

**#include** "msp.h"

**#include** "uart.h"

**#include** "dac.h"

**#include** "sound.h"

**#include** "math.h"

**#include** "timer.h"

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

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

// 24 mhz

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0 = CS\_CTL0\_DCORSEL\_4;

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

Setup\_UART();

Setup\_DAC();

Timer\_Setup();

\_\_enable\_irq();

Sound\_Init(0, 21, 3, 18);

**while** (1) {

}

}

### uart.h

/\*

\* uart.h

\*

\* Created on: May 8, 2017

\* Author: kmrosent

\*/

#ifndef UART\_H\_

#define UART\_H\_

int statusFlag;

int val;

void Setup\_UART();

unsigned char UART0Rx(void);

unsigned char UART0Tx(unsigned char c);

void UART\_String(char \* str, int len, int newline);

void EUSCIA0\_IRQHandler(void);

#endif /\* UART\_H\_ \*/

### uart.c

/\*

\* uart.c

\*

\* Created on: May 8, 2017

\* Author: kmrosent

\*/

#include "uart.h"

#include "multi.h"

#include "msp.h"

/\*\* sets up the uart \*/

void Setup\_UART() {

val = 0;

statusFlag = 0;

EUSCI\_A0->CTLW0 |= EUSCI\_A\_CTLW0\_SWRST;

EUSCI\_A0->MCTLW = 0;

EUSCI\_A0->CTLW0 = EUSCI\_A\_CTLW0\_SSEL1 + EUSCI\_A\_CTLW0\_SWRST;

EUSCI\_A0->BRW = 26;

P1SEL0 |= (BIT2 + BIT3);

P1SEL1 &= ~(BIT2 + BIT3);

EUSCI\_A0->CTLW0 &= ~EUSCI\_A\_CTLW0\_SWRST;

EUSCI\_A0->IFG |= EUSCI\_A\_IFG\_RXIFG;

EUSCI\_A0->IE |= EUSCI\_A\_IE\_RXIE;

//NVIC\_SetPriority(EUSCIA0\_IRQn, 4);

NVIC\_EnableIRQ(EUSCIA0\_IRQn);

}

/\* read a character from UART0 \*/

unsigned char UART0Rx(void) {

char c;

while(!(EUSCI\_A0->IFG & 0x01)) ;

c = EUSCI\_A0->RXBUF;

return c;

}

/\* write a character to UART \*/

unsigned char UART0Tx(unsigned char c) {

while(!(EUSCI\_A0->IFG&0x02)) ;

EUSCI\_A0->TXBUF = c;

return c;

}

/\* write a string to UART \*/

void UART\_String(char \* str, int len, int newline) {

int i;

for (i = 0; i < len; i++) {

UART0Tx(str[i]);

}

if (newline == 1) {

UART0Tx('\033');

UART0Tx('E');

}

}

/\*\*

\* Handle interrupt and if ‘return’ calls Store()

\*/

void EUSCIA0\_IRQHandler(void) {

**static** **int** wave\_num = 0;

**char** c = EUSCI\_A0->RXBUF;

**switch** (c) {

**case** 'a':

wave\_num = (wave\_num + 1) % 3;

**break**;

**case** 'b':

Update\_Period(wave\_num, 1);

**break**;

**case** 'c':

Update\_Period(wave\_num, -1);

**break**;

**case** 'd':

Update\_Freq(wave\_num, 1);

**break**;

**case** 'e':

Update\_Freq(wave\_num, 1);

**break**;

}

//**while**(!(EUSCI\_A0->IFG & 0x02)) {}

//EUSCI\_A0->TXBUF = c;

}

### timer.h

/\*

\* timer.h

\*

\* Created on: May 15, 2017

\* Author: kmrosent

\*/

#ifndef TIMER\_H\_

#define TIMER\_H\_

void Timer\_Setup();

void TA0\_0\_IRQHandler(void);

#endif /\* TIMER\_H\_ \*/

### timer.c

/\*

\* timer.c

\*

\* Created on: May 15, 2017

\* Author: kmrosent

\*/

#include "timer.h"

#include "adc.h"

#include "msp.h"

/\*\* setup and enables timer \*/

void Timer\_Setup() {

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

TIMER\_A0->CCR[0] = 750;

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_UP |

TIMER\_A\_CTL\_ID\_\_1;

NVIC\_EnableIRQ(TA0\_0\_IRQn);

}

/\*\* requests another sample at 4000 samples/sec rate \*/

void TA0\_0\_IRQHandler(void) {

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

Sound\_Go();

}

### sound.h

/\*

\* sound.h

\*

\* Created on: Jun 2, 2017

\* Author: kmrosent

\*/

**#ifndef** SOUND\_H\_

**#define** SOUND\_H\_

**#define** SET\_WAVE(num,freq,period) wave[num][1] = freq; wave[num][2] = period;

**#define** NEXT\_WAVE(num) wave[num][0] = ((wave[num][0] + wave[num][1] ) % (256 \* wave[num][2]))

**int** wave[4][3];

**void** **Sound\_Init**(**int**,**int**,**int**,**int**);

**void** **Update\_Wave**(**int** waveNum, **int** freq, **int** period);

**void** **Update\_Period**(**int** waveNum, **int** period);

**void** **Update\_Freq**(**int** waveNum, **int** freq);

**void** **Sound\_Go**();

### sound.c

/\*

\* sound.c

\*

\* Created on: Jun 7, 2017

\* Author: kmrosent

\*/

**#include** "sound.h"

**#include** "dac.h"

**int** sinwaveX[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** sawX[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};

**void** **Sound\_Init**(**int** x1, **int** x2, **int** x3, **int** x4) {

wave[0][0] = x1;

wave[1][0] = x2;

wave[2][0] = x3;

wave[3][0] = x4;

wave[0][1] = 1;

wave[1][1] = 2;

wave[2][1] = 3;

wave[3][1] = 4;

wave[0][2] = 12;

wave[1][2] = 14;

wave[2][2] = 16;

wave[3][2] = 8;

}

**void** **Update\_Period**(**int** waveNum, **int** period){

Update\_Wave(waveNum, 0, period);

}

**void** **Update\_Freq**(**int** waveNum, **int** freq) {

Update\_Wave(waveNum, freq, 0);

}

**void** **Update\_Wave**(**int** waveNum, **int** freq, **int** period) {

**if** (waveNum >=0 && waveNum <4) {

**if** (wave[waveNum][1] + freq > 0 && wave[waveNum][1] + freq <= 128)

wave[waveNum][1] += freq;

**if** (wave[waveNum][2] + period > 0 && wave[waveNum][2] + period <= 32)

wave[waveNum][2] += period;

}

}

**void** **Sound\_Go**() {

**int** i = 0;

//i++;

//Drive\_DAC(sinwaveX[i%256]);

NEXT\_WAVE(0);

NEXT\_WAVE(1);

**if** (wave[0][0] % 3 == 0) { NEXT\_WAVE(2); }

NEXT\_WAVE(3);

Drive\_DAC( sinwaveX[wave[0][0]%256] / (2 + wave[0][0]/256) +

sinwaveX[wave[1][0]%256] / (2 + wave[1][0]/256) +

sinwaveX[wave[2][0]%256] / (2 + wave[2][0]/256));

}