VE373 Microprocessor Based System Design

Final Project Report

PIC32 Based Multifunctional Electronic Piano

Prepared by

Wang, Qian 5113709012

Wang, Liyu 5113709018

Li, Yanrong 5113709246

Undergraduate students of ECE

UM-SJTU Joint Institute

Prepared for

Dr. Zheng, Gang

Instructor of VE373

UM-SJTU Joint Institute

Mr. Jia, Lingxiao

Teaching assistant of VE373

UM-SJTU Joint Institute

Miss Zheng, Yue

Teaching assistant of VE373

UM-SJTU Joint Institute

Date of Submission

August 9, 2014

Contents

1. Introduction

Many of us have played some musical instruments at our early age. When we play the instruments, the fascinating rhythm flowing from our fingers attracts us to continue digging out the deepest secrets behind, and kindles our fervor to become a true master of every piece of melody. However, current musical instruments don't provide anything else but sounds. This limited function becomes an obstacle in our way pursuing an outstanding comprehension to music as well as proficiency playing skills. Therefore, we here present an updated version of musical instrument – **multifunctional electronic piano.** We wish this electronic piano may help a potential musician to make better progress.

The basic function that this piano provides is, of course, **broadcasting sounds**. This main part consists of a keyboard and a speaker. The keyboard is exactly the same as a true piano. When we press a key, the signal change will be transferred to a PIC32 MCU. This MCU is set particularly for signal detecting because we have no enough ports to support digital input on one PIC32. Then the PIC32 I will tell PIC32 II which key is pressed. This will trigger an interrupt and PIC32 will output a series of PWM wave. Each sound has a corresponding specifically designed PWM wave based on the waveform of the sound. After that, the PWM wave will be sent to a filtering current, an amplifying circuit and the speaker successively. Finally, the sound will be broadcast by the speaker.

The first feature that this piano provides is an **LED spectrum display.** As we know, each sound has its spectral expression, and there are usually subtle spectral relationships between several tones whose combination is sweet-sounding. To reveal the secret behind this, we introduce a spectrum display. This display mainly consists of an LED array and a motor. When we play the piano, the motor will rotate at a constant angular velocity, and the LED will display the spectrum of the sound by turning on and off. Since there is only one line of LED, the rotation must be very fast so that human brain will be wrongly impressed that there is an LED array continuously distributed in the space. It seems that an LED spectrum display is common, but **there are actually two innovations in this device:** First, all the current LED rotating display products cannot update its display contents dynamically, while ours can display changing patterns based on real-time transmission. Second, current spectrum display can only show the spectrum of a stored music, while our mechanism allows it to show anything that we are playing immediately.

The second feature of this piano is that it can record what one played and **print it onto a staff on a PC.** To implement this function, we exploit UART to communicate with PC. Every time when we press a key, the MCU will transmit messages to PC, which will transform it into a musical note on a staff. The position of the note is determined by the time we press the key. For example, if we press two keys almost at the same time, then two notes will appear on the staff very adjacently. Finally, we can get a staff with all the notes we played during the whole process. This feature may help people to find out if there are any mistakes during their play and to correct them.

The whole system is based on two PIC32MX795F512L microcontrollers. With the aid of this microchip, we wish we can implement all the functions described above, and we believe this artwork will give every player a brand new joy.

2. Overview

This is a function block diagram of our system.

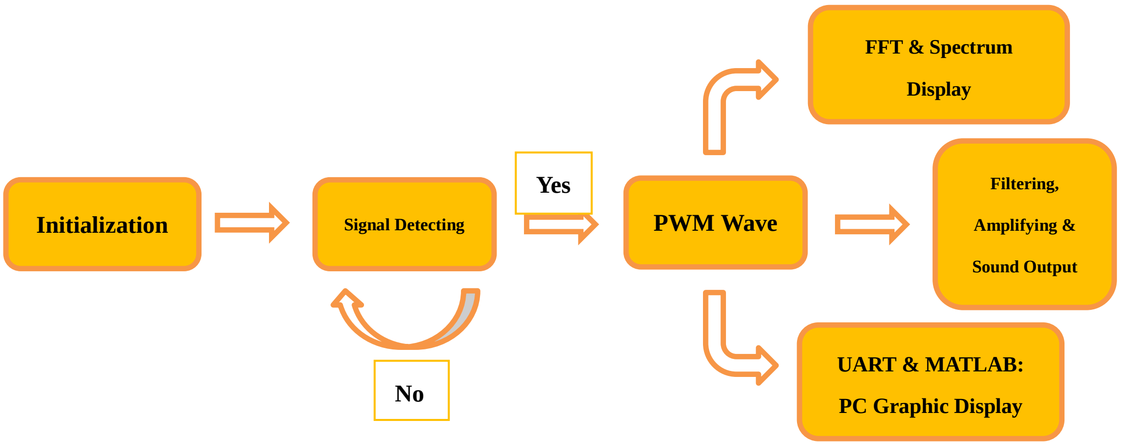

Initialization
Signal Detecting
Yes
No
PWM Wave
FFT & Spectrum Display
Filtering, Amplifying & Sound Output
UART & MATLAB: PC Graphic Display

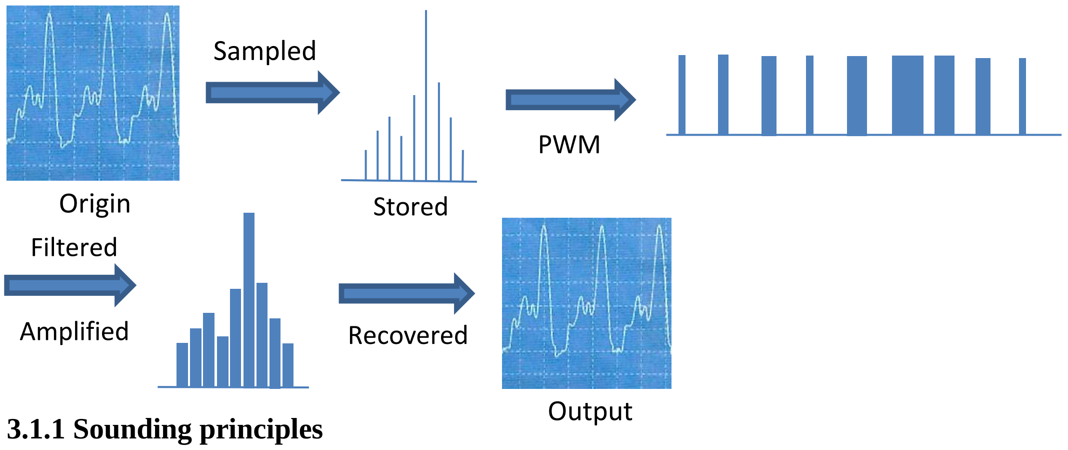
Fig 1. Function block diagram of the system.

3. Sounding

The core function of a piano is sounding. To produce standard sound without frequency shift, first we need to get clear how a sound is produced and reveal the secret behind different tones and timbres.

3.1 Sounding Process

Generally, the sounding process is like following:



3.1.1 Sounding principles

The sound wave is essentially the propagation of mechanical vibration. It is an analog signal proportional to the displacement of a mass point deviating from the equilibrium position. Usually for a musical instrument, this kind of vibration has certain frequency/period. If the vibration is totally random, then that will be noise. A musical instrument has three main factors: volume, timbre and pitch. Once these three factors are determined, a unique sound is determined.

Volume

Volume denotes the loudness of a sound. It is determined by the amplitude of the vibration. When we use a loudspeaker to sound, the amplitude of the vibration is actually proportional to the voltage swing of the audio signal.

Timbre

Different instrument have different timbres, and the timbres are determined by the waveform of the sound. Figure 2 shows some waveforms. By repeating those waveforms in a certain frequency, we can produce musical sounds.

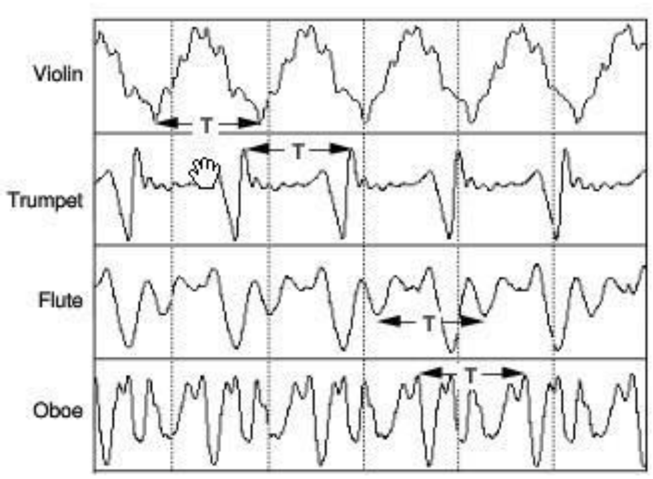
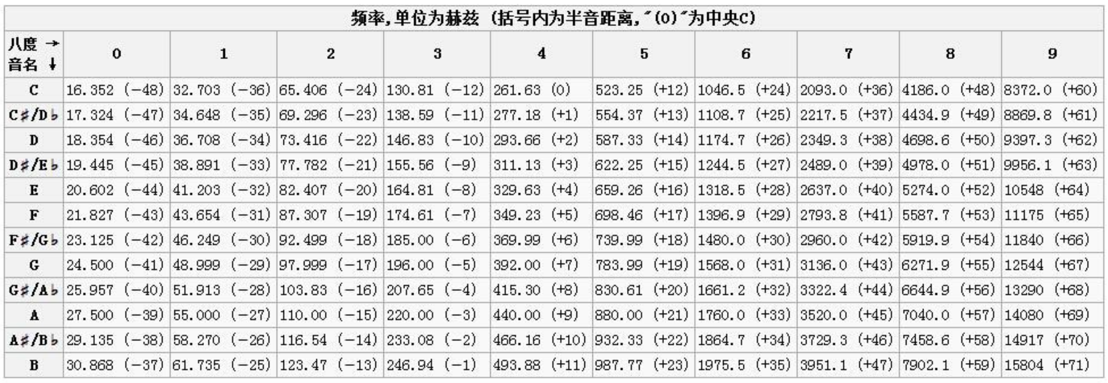


Fig 2. Waveform of different musical instruments (adopted from www.guokr.com)

Pitch

From the lowest bound (20Hz) to the highest bound (20000Hz), the pitch of the sound wave changes from low to sharp. Among those various pitches, the 440Hz one is stipulated as the standard pitch, and assigned name as A4. Based on A4, people also produce C4, C#4 D4, D#4 E4, F4, F#4, G4, G#4, A#4, B4 and combine them as a group octave. To broaden the sound range, we set multiple octaves by doubling or halving the frequency of the sound. For example, the frequency of A3 is half of that of A4. Here we show a pitch-frequency converting table:



频率，单位为赫兹 (括号内为半音距离，"(0)"为中央C)										
八度 → 音名 ↓	0	1	2	3	4	5	6	7	8	9
C	16.352 (−48)	32.703 (−36)	65.406 (−24)	130.81 (−12)	261.63 (0)	523.25 (+12)	1046.5 (+24)	2093.0 (+36)	4186.0 (+48)	8372.0 (+60)
C♯/D♭	17.324 (−47)	34.648 (−35)	69.296 (−23)	138.59 (−11)	277.18 (+1)	554.37 (+13)	1108.7 (+25)	2217.5 (+37)	4434.9 (+49)	8869.8 (+61)
D	18.354 (−46)	36.708 (−34)	73.416 (−22)	146.83 (−10)	293.66 (+2)	587.33 (+14)	1174.7 (+26)	2349.3 (+38)	4698.6 (+50)	9397.3 (+62)
D♯/E♭	19.445 (−45)	38.891 (−33)	77.782 (−21)	155.56 (−9)	311.13 (+3)	622.25 (+15)	1244.5 (+27)	2489.0 (+39)	4978.0 (+51)	9956.1 (+63)
E	20.602 (−44)	41.203 (−32)	82.407 (−20)	164.81 (−8)	329.63 (+4)	659.26 (+16)	1318.5 (+28)	2637.0 (+40)	5274.0 (+52)	10548 (+64)
F	21.827 (−43)	43.654 (−31)	87.307 (−19)	174.61 (−7)	349.23 (+5)	698.46 (+17)	1396.9 (+29)	2793.8 (+41)	5587.7 (+53)	11175 (+65)
F♯/G♭	23.125 (−42)	46.249 (−30)	92.499 (−18)	185.00 (−6)	369.99 (+6)	739.99 (+18)	1480.0 (+30)	2960.0 (+42)	5919.9 (+54)	11840 (+66)
G	24.500 (−41)	48.999 (−29)	97.999 (−17)	196.00 (−5)	392.00 (+7)	783.99 (+19)	1568.0 (+31)	3136.0 (+43)	6271.9 (+55)	12544 (+67)
G♯/A♭	25.957 (−40)	51.913 (−28)	103.83 (−16)	207.65 (−4)	415.30 (+8)	830.61 (+20)	1661.2 (+32)	3322.4 (+44)	6644.9 (+56)	13290 (+68)
A	27.500 (−39)	55.000 (−27)	110.00 (−15)	220.00 (−3)	440.00 (+9)	880.00 (+21)	1760.0 (+33)	3520.0 (+45)	7040.0 (+57)	14080 (+69)
A♯/B♭	29.135 (−38)	58.270 (−26)	116.54 (−14)	233.08 (−2)	466.16 (+10)	932.33 (+22)	1864.7 (+34)	3729.3 (+46)	7458.6 (+58)	14917 (+70)
B	30.868 (−37)	61.735 (−25)	123.47 (−13)	246.94 (−1)	493.88 (+11)	987.77 (+23)	1975.5 (+35)	3951.1 (+47)	7902.1 (+59)	15804 (+71)

Fig 3. Pitch – frequency converting table. (This table is adopted from www.elecfans.com)

3.1.2 Wave sampling

To simulate a piano sound, we need to generate the sound wave with the same frequency and waveform. Therefore, the first step is sampling. The standard sampling rate to audible sound is 44100Hz (this is derived by Nyquist sampling principle: $f = 2f_0$, while the upper bound frequency of audible sound is around 20000Hz). However, the memory of PIC32 MCU is too small to store all the sounds we need, so we only sample one waveform from each sound, and broadcast it repeatedly with certain frequency, and we can get the sound.

We use GuitarPro6 software to produce piano sound from C2 to B5 (four octaves in total). The sounds are saved as .wav files. Then we use MATLAB to sample them and choose one period of waveform. Here we take C4 as an example:

Opening c4.wav with MATLAB we got waveform shown in Fig 4, and we choose one period of it.

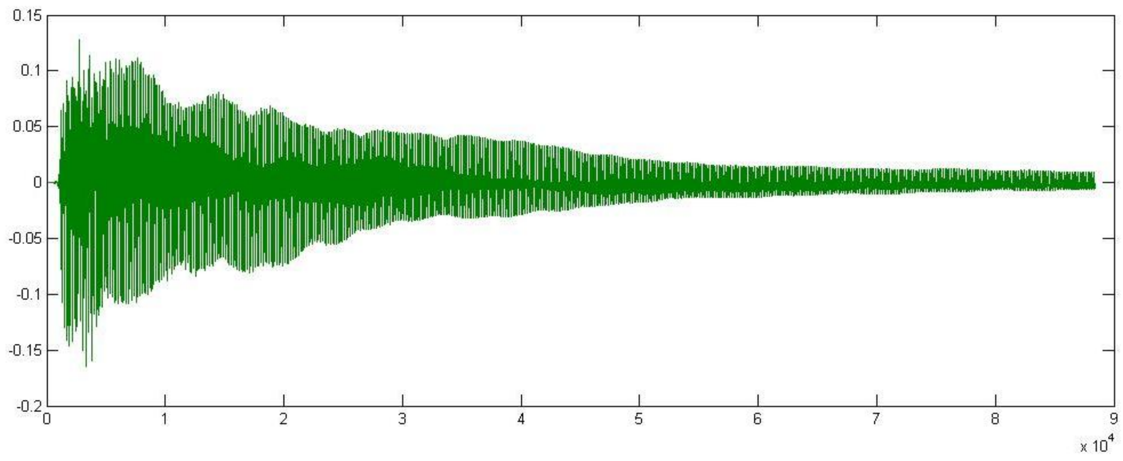


Fig 4(a) The waveform of a standard C4 sound lasting for 2 seconds.

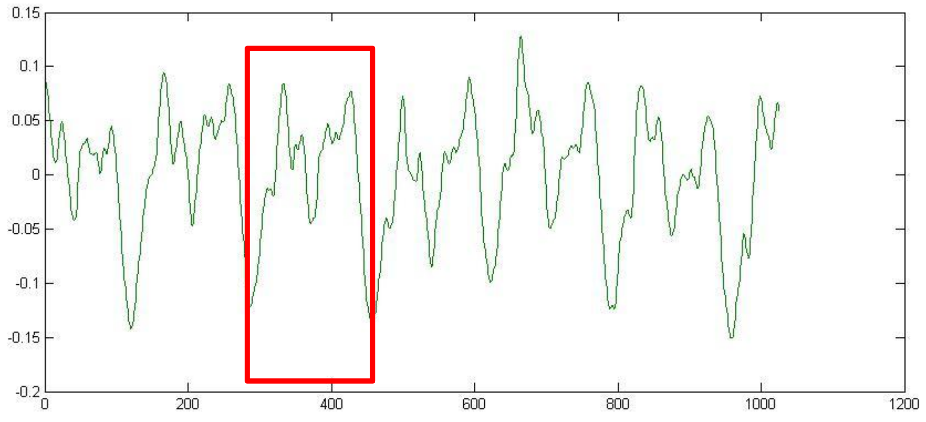


Fig 4(b) Locally magnified waveform of a C4 sound. The waveform in the red rectangle denotes the waveform in a period.

The displacement will be (sampling rate = 44100Hz, 169 samples in total):

```
const int Table25[159] = {-44, -44, -43, -41, -39, -37, -35, -33, -31, -30, -29, -
28, -27, -25, -24, -23, -22, -21, -20, -18, -17, -14, -12, -9, -5, -2, 1, 4, 7, 11,
 14, 17, 20, 23, 24, 25, 25, 24, 22, 18, 14, 10, 6, 3, 0, -1, -2, -3, -5, -6, -8, -
8, -8, -7, -4, -1, 3, 6, 8, 10, 10, 11, 11, 11, 11, 11, 12, 12, 11, 10, 8, 5, 1, -2
, -5, -7, -8, -7, -7, -7, -7, -8, -8, -9, -9, -8, -6, -4, -2, 0, 2, 3, 4, 5, 6, 8,
10, 12, 15, 18, 20, 22, 23, 24, 25, 26, 27, 29, 31, 32, 34, 35, 36, 37, 37, 37, 38,
 38, 39, 40, 40, 40, 39, 37, 33, 29, 24, 19, 14, 9, 4, -1, -4, -7, -9, -11, -12, -
14, -15, -16, -17, -19, -20, -21, -22, -23, -25, -26, -28, -31, -33, -36, -39, -41,
 -43, -45, -45, -45, -45};
```

This array will be recorded in the memory, and we just need to visit the array with frequency 44100Hz repeatedly, the sound C4 will be recovered. Moreover, to simulate the gradual weakness of the sound, we multiply an exponentially decay coefficient. Similarly, for all the other 47 sounds, we do the same work.

3.1.3 PWM Generating

The system should express the amplitude of the sampling points by the duty cycle of PWM wave, so we set the PWM interrupt period as 1/44100 second. Since the amplitude may be both positive and negative, we set PR/2 as zero output, and add the amplitude on PR/2 to be non-zero output.

```
OC2CON = 0;
OC2RS = 0;
OC2R = 0;
OC2CONbits.OCM = 6; //PWM, timer2;
OC2CONbits.ON = 1;
```

…

```
A = FIFO.TableN[FIFO.count];
DecayLevel = FIFO.duration >> 10;
D = Decay[DecayLevel];
sum = ((A*D) >> 10);
```

…

```
OC2RS = PR2/2 + sum;
```

3.1.4 Filtering circuit

Only adjusting PWM wave is not enough, this is because every square pulse contains infinite frequency components, and if we want the duty cycle to be proportional to the amplitude, then what we want is only the DC component. Any none-zero order harmonious wave should be filtered. So we build a simple second order low-pass filter circuit, and we also show the filter curve:

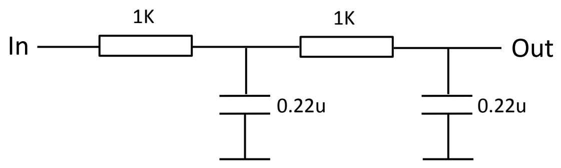


Fig 5 Second order low pass filter schematic

Of course this filter is not ideal, but according to the test result, it can approximately fulfill our demands.

3.1.5 Amplifying circuit

The PWM pulse provides at most 3.3V of voltage swing, which is not enough to drive a loudspeaker, so we have to build an amplifying circuit. Based on available device, we choose LM386 from our lab to be the core operational amplifier, and designed the peripheral circuits. The circuit schematic is shown in Figure 6.

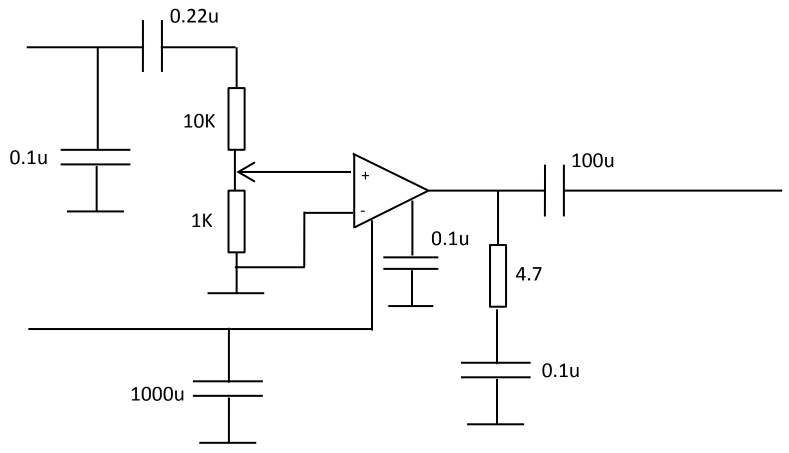


Fig 6. Amplifier schematic

Then is the test of the filter and amplifier. The PWM wave duty cycle is changing from 0~10 repeatedly, and ideally it will produce a triangle wave after it passes through the circuit. From the oscilloscope in figure 8 we see that the circuit performs really well.

According to the test, a 0.6V average voltage swing input is able to generate a loud enough output. Given that PR is set as 227, we set the peak value of the pulse as +-45.

Finally, we can send the output to the loudspeaker.

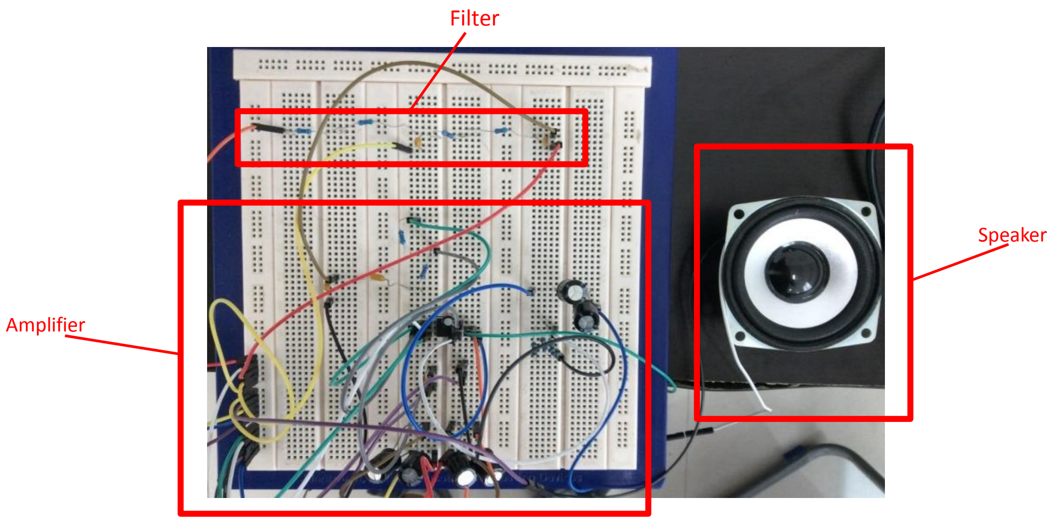


Fig 7. Circuit Implementation

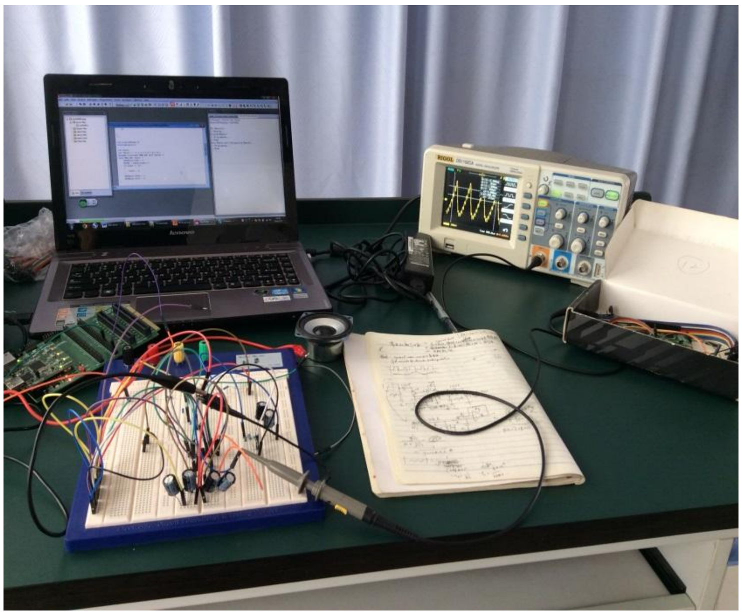


Fig 8. Test of circuits.

3.2 Keyboard

The keyboard consists of five main parts: base, button, rubber, pull-down resistor and key set. Figure 9 gives a vision from the bottom to the top layer of the whole keyboard.

The base of the keyboard is built with Acrylic plate. Upon it we lay a bread board for sensor circuit. Then we set a switch array. The switches control the connections between the high voltage (3.3V) line and the input signal line. To make sure the keys can bounce back after it is pressed, we add a piece of elastic rubber (this is also used in a true electronic piano). Finally, we lay the key set on the top layer.

However, that is not enough. When the button is released, the input of the MCU is actually “pending”, that is, it doesn’t connect to anything. So the input will be randomly switches between 0 and 1. This kind of “pending” is really a disaster, so we add a 2K ohm pull-down (grounded) resister to provide a low level when there is no signal.

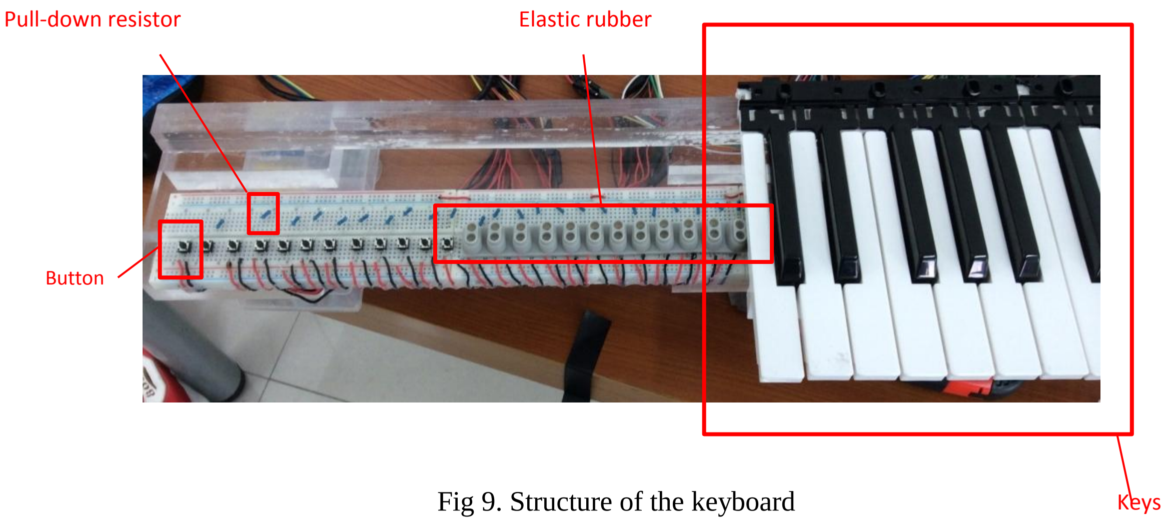


Fig 9. Structure of the keyboard

3.3 Signal transmission and processing

Up to this, the whole keyboard is finished. Every time a key is pressed, the button below that key is also pressed, the 3.3V high voltage is transmitted to the input port and the first PIC32 MCU will judge which key is pressed based on the input pattern. Then it transmits six bits of 0 and 1 representing 0~48 as well as a change notice signal to the second MCU. The second MCU receives CN, performs a CN interrupt and read the input 6 bits. By then it knows which key is just pressed. Finally what it needs to do is to visit the corresponding array periodically send PWM wave.

```
if(PORTCbits.RC13 == 1)
{

    checkinQ = 0;
    readnum = PORTDbits.RD2+PORTDbits.RD3*2+PORTDbits.RD4*4+PORTDbits.RD5*8+
    PORTDbits.RD6*16+PORTDbits.RD7*32;
```

When playing a sound, there are two software counters kept. The first one is used for visiting the certain position in the array. The second one is a global counter, used to track how long the sound has been played. This counter helps determine the decay coefficient, and stop broadcasting the sound if the time is longer than 2.5 second.

```
FIFO.count++;
if(FIFO.count == FIFO.TableSize)
{
    FIFO.count = 0;
}
FIFO.duration++;
if(FIFO.duration == TIMEOUT)
{
    dequeue();
}
```

Exploiting the global counter, we also implement jitter elimination in this part. Every time a CN is raised, the program will check whether this sound is identical with the same sound playing. If it is and the playing time has not exceeded 0.1 second yet, the system will regard this as a jitter, and the newly coming one won't replace the current playing one. If two identical consecutive playing has time interval > 0.1 second, then we think they are pressed deliberately and replay the new sound (including visit the first element in the array and reset the two counters).

```
if((readnum == 1 && FIFO.num == 0)||(readnum == FIFO.num))
    {
        checkinQ = 1;
        if(FIFO.duration > 4000)
        {
            FIFO.ifabort = 1;
            if(readnum == 1)
            {
                enqueue(0);
                send(0);    //send # of key to PC by UART;
            }
            else
            {
                enqueue(readnum);
                send(readnum);
            }
        }
    }
```

4. LED Spectrum Display

4.1 Objective

Every note has its corresponding frequency spectrum. It is a very common used function in the music player to show the spectrum by a histogram like graph. We also want to visualize the tone when we play the electronic piano just like the figure showed below.

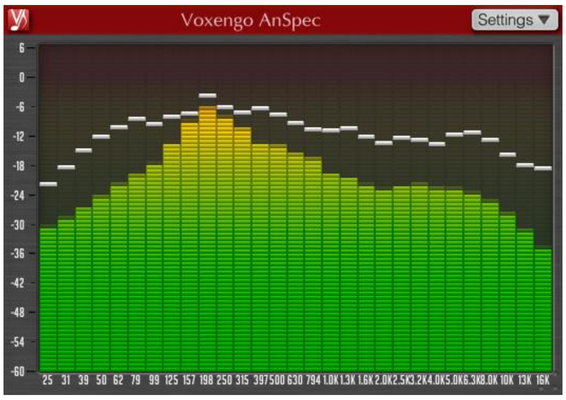

Voxengo AnSpec
Settings
6
0
-6
-12
-18
-24
-30
-36
-42
-48
-54
-60
25 31 39 50 62 79 99 125 157 198 250 315 397 500 630 794 1.0K 1.3K 1.6K 2.0K 2.5K 3.2K 4.0K 5.0K 6.3K 8.0K 10K 13K 16K

Fig 10. Spectrum histogram in music player (Picture derived from rekkerd.org)

In order to implement this function, LED lights will be used and controlled by the PIC32 board according to different tones. Due to the limitation of the output pins in the board for LED Spectrum Display, we let LED rotate very fast. At different position, it displays different frequency components. According to the visual staying phenomenon, every light will produce a line and a line of lights can show a spectrum graph if the lights are controlled correctly.

4.2 Program Implementation

In our programming code, we pre-store the number of illuminating LEDs at each time interval for every note in a two-dimensional array. This number is related to the Fast Fourier Transform (FFT) result of each sound we pressed closely. We can first calculate the FFT of each note sample by MATLAB, and store the absolute value of each FFT result as an indicator for the energy at each stage. The sound samples are produced previously as the format of wave file.

```
y = audioread(fileName);
y = y(:,1);
f = fft(y);
f = abs(f);
f = f(1:50);
```

Because most of the energy of a sound concentrates a low frequency area, for each note, we only focus on the first 50 absolute results of the FFT, from which the number of illuminating LEDs for the note i can be determined by

```
delta = (maxF - minF) / 8;
light((k-1)*12+j,:) = min(8,ceil((f - minF) / delta));
```

To use the obtained data conveniently in the C code, we use the file operation in MATLAB to temporarily store this two-dimensional array in a txt file. The resulted constant two-dimensional array representing the number of lighting LEDs at each time interval for every note is in the appendix. We use this result to calculate the signal of corresponding LED lights which should be turn on as the output of the PIC32 board.

4.3 Mechanical structure

The structure of LED spectrum display is divided into three parts, the supporting frame part, the rotating part and the LED lights part.

The supporting frame part is composed of two circles, which are made of acrylic boards. There are four T-shaped stanchions to hold the lower circle board and support the upper circle board. The A-B glue is used to fix the joint parts. The following figure shows the details of the circle boards we design and the unit of the size in the figure is millimeter.

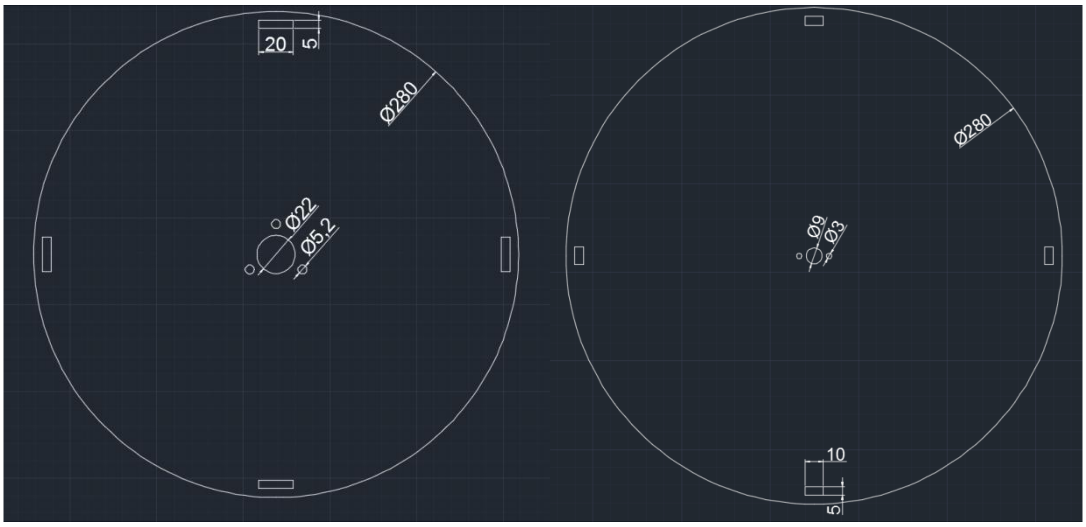


Fig 11. The Upper & Lower Circle Board

The stanchions are specially designed with two T-shaped extruding parts. These two parts are related with the size of the rectangles in the circle board and they must be matched well in order to assemble them together. The following figure shows the details of the stanchions.

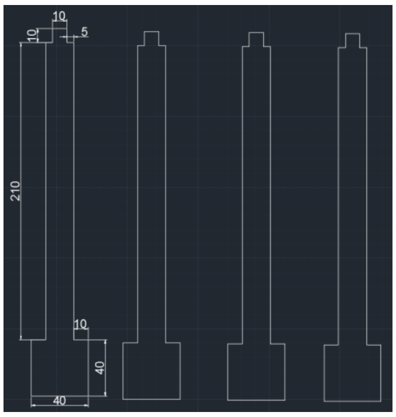


Fig 12. The Stanchions Part

The rotating part is the most complicated one in this structure. There are two magic components in this part, one is the conductive ring and the other is the“8”-shaped coupling. When the LED board is rotating, we must let the wires rotate together with the board. If we just simply connect wires with the LEDs, the wires will surely twine together. The conductive ring is designed for such situation. The stator and the rotor are combined with something like the electric brush in the motor so that the wires are fixed in one side and are able to rotate in the other side.

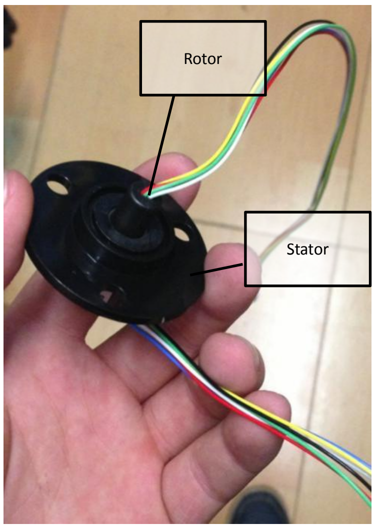


Fig 13. The Conductive Ring

In order to let the rotor work, we need link to motor to it. The ordinary motor coupling cannot be used since we cannot deal with the wires. The “8”-shaped coupling solves this problem perfectly. One side can be fixed on the rotor part of the conductive ring and the wires can come out from the gap. The other side is connected to the motor through a steel axle.

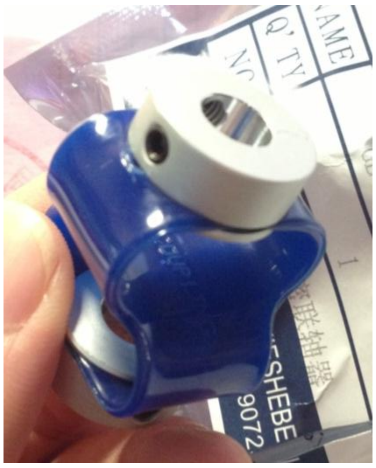


Fig 14. The “8”-Shaped Coupling

The conductive ring is fixed in the lower circle board and the motor is fixed in the upper one. These two components are combined together with the coupling and the steel axle. In the middle of the axle, the three-directions axle sleeve is used to link with another steel axle vertical to the major one. Hot melt adhesive is used to make sure that the joint part is in high strength.

The last part is the LED board. It is a simple part and I dig 10 holes in the board, two for self-tapping screw to fix the board and the remaining eight holes for LEDs.

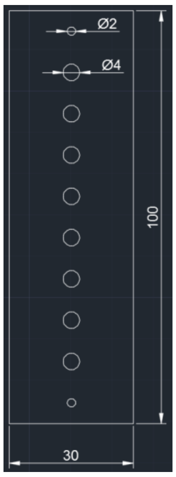


Fig 15. The LED array

After finishing the three parts above, the full structure of the display has be built up. The following figure give an over view of the structure. The 12 volt is required for the motor and 3.3 volt for the LEDs.

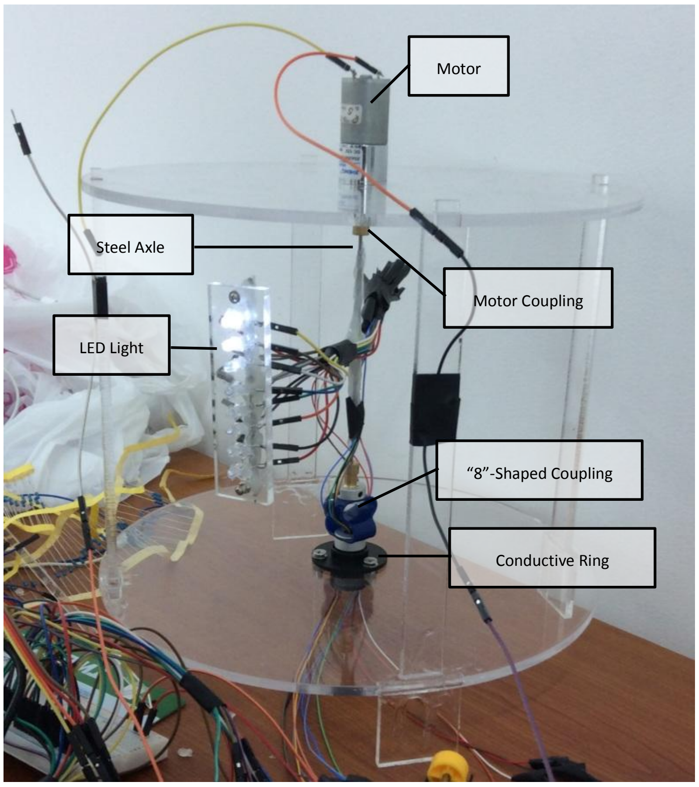


Fig 16. The Full Structure of LED Spectrum Display

4.4 Test

We turned on the LED Spectrum Display with required voltage and play the piano to let the LEDs change. Showing from the following figure, the display really gives us a graph of spectrum as what we have described in the previous part. Through a series of figure, we can see that the display changes its graph with the performer playing the piano.

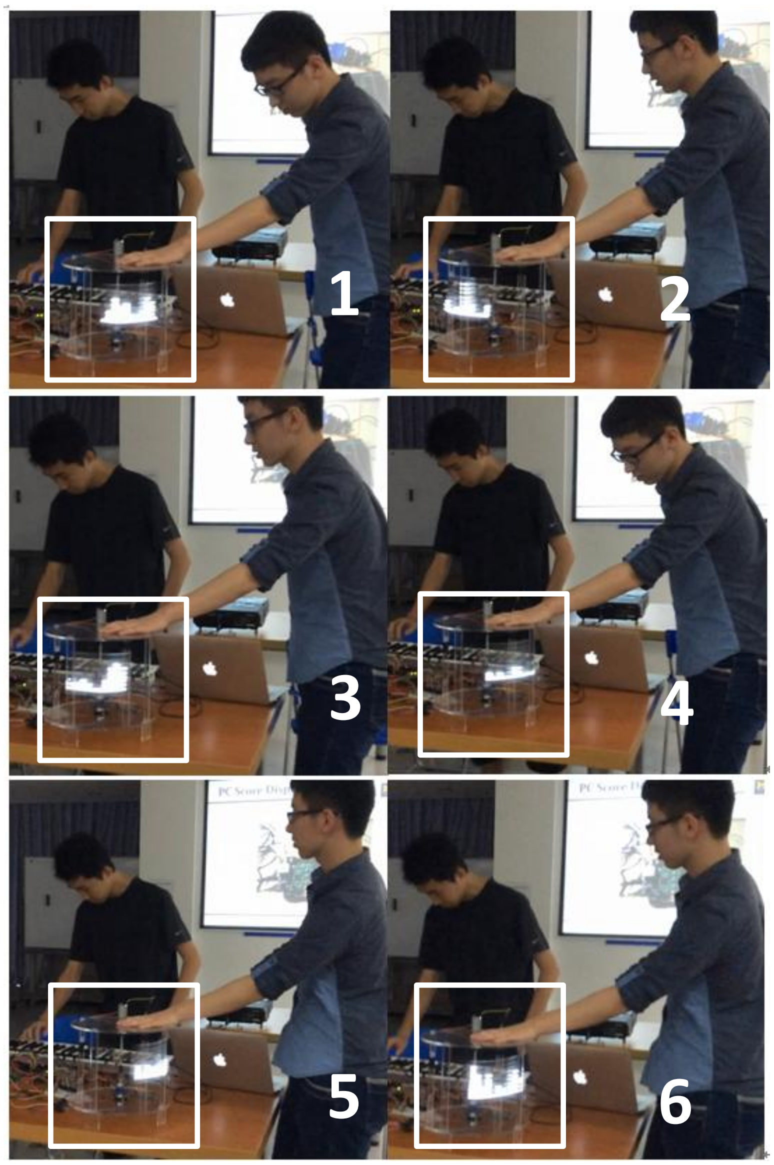


Fig 17. The Testing of the LED Spectrum Display

5. PC Score Display

5.1 Objective

A png file will be generated for every thirty-two music notes. The C2 and C3 octaves are recorded in the bass and The C4 and C5 octaves are recorded in the treble. The purpose to show the score on the screen and to store the playing result into several png files is that the piano beginner can double-check whether he or she has pressed the correct key.

5.2 PIC32 – PC Communication

The signal from the keyboard will be captured by the PC through an RS232 – TTL converter, which is connected to the PIC32 board. The reason why we use an RS232-TTL converter is that we cannot communicate between PIC32 board and PC directly through the USB cable, as what we did in the lab. Therefore, we decide to use an RS232-TTL converter as a communication medium between the PIC32 board and PC.

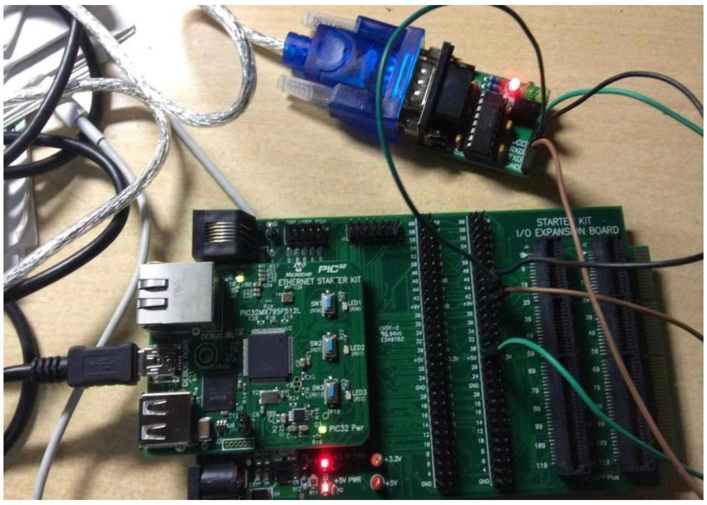


Fig 18. RS232 – TTL converter

An advantage of this implementation is that MATLAB can easily read the signal sent by the RS232-TTL converter. The received result can be applied to produce the score immediately in the same software.

```
s = serial('COM3');
fopen(s);
s.baudrate = 9600;
tmp = fread(s,1)
fclose(s);
```

The value s we received is always a one-byte integer between 0 and 47. Here, a number between 0 and 23 represents a note in bass, and a number between 24 and 47 represents a note in treble. We can temporarily store each number received in two arrays, indicating the notes sequence for bass and treble. For the convenience, when we plot the score, we subtract 24 from this number if it is between 24 and 47.

5.3 Music Score Generation

After receiving the music notes, it is note difficult for us to plot the corresponding note to the screen through:

```
plot(posX(i),posYH(high(i) + 1),'-k.','MarkerSize',30);
plot(posX(i),posYL(low(i) + 1),'-k.','MarkerSize',30);
```

Here, i is a number between 1 and 48. The array posX means the x-coordinate for the i-th note of the 48 notes in one png; posYH and posYL are the y-coordinate in the treble or the bass for the i-th note of the 48 notes in one png.

We should also correctly denote the sharp symbol on the score. An array flag is used to indicate the corresponding number to a sharp note. If we find that a sharp symbol is needed, we can print it to a place near the note.

```
if (flag(high(i) + 1) == 1)
    text(posX(i) + 0.01, posYH(high(i) + 1) + 0.04,'#');
end

if (flag(low(i) + 1) == 1)
    text(posX(i) + 0.01, posYL(low(i) + 1) + 0.04,'#');
end
```

5.4 PC Score Display Test

To test this part separately, we use the user input to simulate the signal from the RS232-TTL converter. One of the resulted png format score is shown as following:

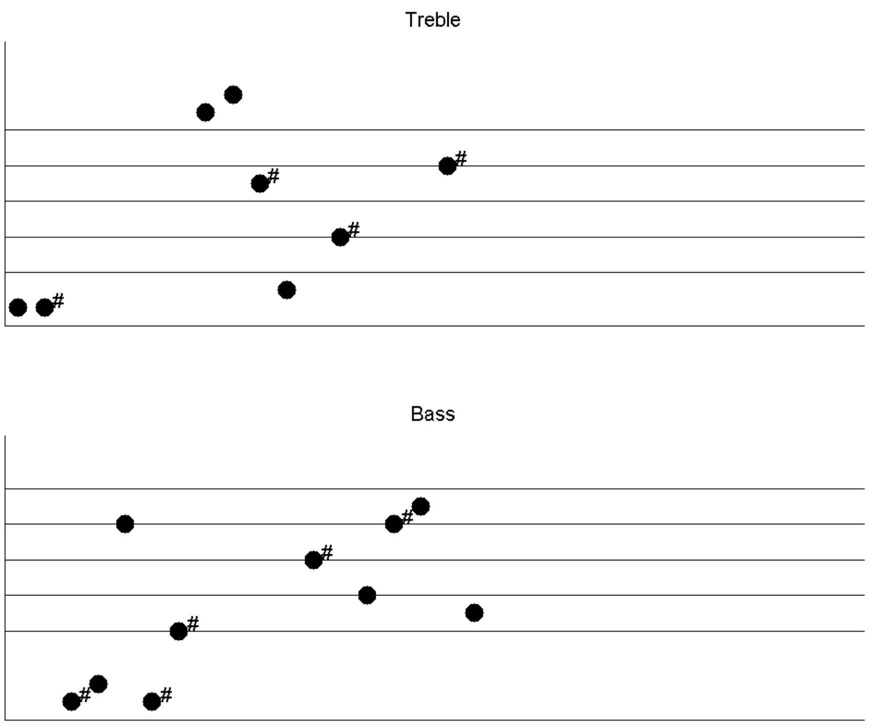


Fig 18. Resulted score figure

6. Performance Evaluation

We not only test our prototype with separate parts, but also perform an integral testing. We assemble every component into a whole system and perform a complete song to test whether it works well. After that, we can find that, in spite of some tiny faults, the timbre mostly resembles the piano, the spectrum shown can change with the rhythm synchronously, and the png files for the music score can be generated by PC immediately after the performance is finished. That is to say, we can conclude that, although there still exist some weak points, our prototype performs well in general.

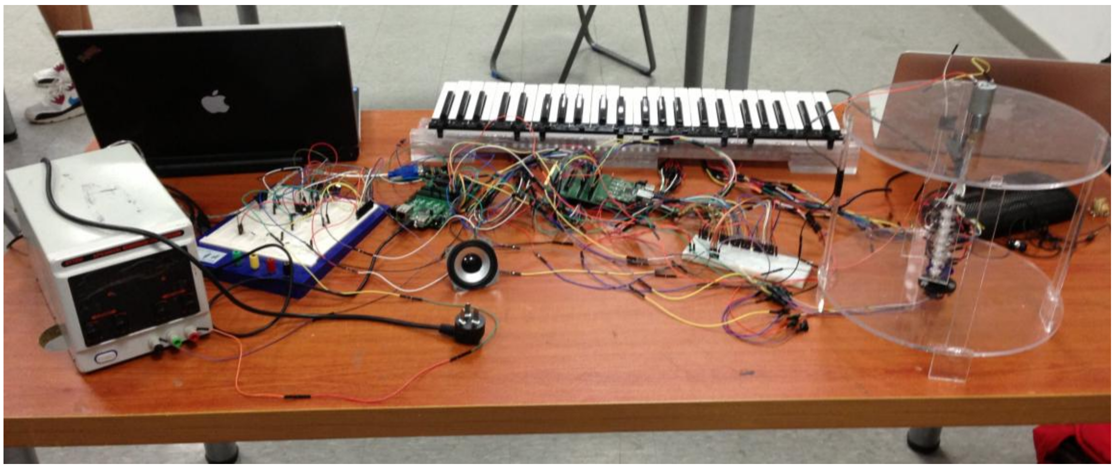


Fig.19 The final system

7. Weakness Discussion

Owing to the lack of time and the hardware limitation, there are still some weak points in our prototype for future improvement. In this section we discuss some weaknesses of our design.

7.1 Single sound limit

Our electronic piano can only play one single note at the same instant. Actually, at first, we aimed at implementing a multi-sound electronic piano, which may produce multiple sounds at the same time, and we even implement the codes. However, the CPU operational capability limits this function: it cannot execute too many instructions at a very short time. As we have explained before, the output waveform is originated from the PWM wave we generated, and that means if two or more sounds are produced at the same time, we need to superposition their waveform, that is, to add their PWM pulse width. Therefore, more sounds mean more time consumption. According to our test, if more than one sound is produced, then the pitch will suddenly become very low, and the decay time for the superposition sound becomes much longer than 2.5 second. We deduce that this phenomenon is caused by an elongated PWM interrupt handler. When one PWM interrupt is generated, CPU enters the handler and will keep running the handler instructions. However, if the CPU doesn’t finish it before the next PWM interrupt comes, the next one has to be pending and waiting, which in fact leads to lower output frequency, and more time to decay. Although we have considered this problem while coding and tried our best to reduce the handler cycle, such as avoiding expensive arithmetic (division, modulus or power), enhancing the SYSCLK to 80MHz, etc., it still needs too much time to broadcast more than one sound. Hence, we think that this problem is caused by a low CPU performance.

7.2 Pitch limit

Although we have prepared both data and hardware for all 48 sounds (four octaves), we finally do not produce them all, because we do not have enough ports for input. During our experiments, we found some I/O ports are actually limited in function. For example, RG8 and RG9 cannot be used as input. Some of them are limited by the system, and some of them just malfunction due to aging of the board. So finally we only produced 31 sounds: C3, D3, E3, F3, G3, A3, B3, and all notes in the fourth and fifth octaves. If given more I/O ports, we can implement even more octaves, and build a full piano.

7.3 Single timbre limit

Our piano can only resemble one timbre – piano timbre, while a true electronic piano can produce many different timbres. This problem is caused by limited time and memory. Facing a tight schedule, we cannot repeat the time-consuming sampling-recording process for too many times, but each timbre needs several dozens of this process. We do not have so much time to do it. The other reason is the memory limit. By simple calculation, we have around 10000 sampling points for all the 48 notes, and each sample is recorded as an integer (4 byte). For 128K RAM, we have at most 3 timbres. However, the timber conversion is really very simple: we just need to read different array set from the memory. So if given more time and a larger memory, this function can also be implemented.

7.4 Timbre distortion

The low frequency sounds our design resembles true piano sound very much, while the high frequency part are not quite like the true piano timbre. This is caused by a sampling number decrement. Higher frequency means shorter waveform period, and thus less samples based on a 44100Hz sampling rate. For example, for each waveform of C4, we have 169 samples, while for each waveform of C5 we only have 84 samples. So we cannot 100 percent recover the sound. Another reason is that true piano wave actually contains two stages: attack stage and maintain/decay stage. During the maintain/decay stage the sound is just a series of repeated waveform, while during the attack stage the waveform is very random, which is hard to simulate. So we can only produce a series of distorted sounds.

8. Conclusion

In this project, we successfully design and implement a multifunctional electrical piano, which can not only generate the sound, but also display the spectrum and show the music score on the PC screen. This system is implemented on the PIC32 board, with timer, change notice, UART serial communication, and PWM. Apart from the knowledge learnt in this course, we also integrate the knowledge from other courses to make the system come true. For example, we use the knowledge about physics to analyze the relation between the sound of notes and waveform; we use the knowledge about circuit design to build up the filtering circuit; we use the knowledge about FFT to derive the state of LEDs; we also use the knowledge about MATLAB to calculate the FFT and to produce the music score. To conclude, our project successfully integrate different aspects of knowledge to deliver an innovative electrical piano with multiple useful functions.

Appendix

Appendix A: graphScore.m

```
cnt = 0;
clef = 0;
```

```
maxLen = 32;
high = -ones(1,maxLen);
low = -ones(1,maxLen);
x = 0:0.01:1;
y = ones(1,length(x));
posX = linspace(1.0/maxLen/2,1 - 1.0/maxLen/2,maxLen);
flag = [0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0];
posYL = [-0.3 -0.3 -0.2 -0.2 -0.1 0 0 0.1 0.1 0.2 0.2 0.3 0.4 0.4 0.5 0.5
0.6 0.7 0.7 0.8 0.8 0.9 0.9 1.0];
posYH = posYL + 0.2;
while (1)
    % s = serial('COM3');
    % fopen(s);
    % s.baudrate = 9600;
    % tmp = fread(s,1)
    % fclose(s);
    tmp = input('An integer between 0 and 47: ');
    cnt = cnt + 1;
    if (length(tmp) == 0)
        clf;
        subplot(2,1,1);
        hold on;
        plot(x,y*0.9,'k');
        plot(x,y*0.7,'k');
        plot(x,y*0.5,'k');
        plot(x,y*0.3,'k');
        plot(x,y*0.1,'k');
        for i = 1:maxLen
            if (high(i) >= 0)
                plot(posX(i),posYH(high(i) + 1),'-k.','MarkerSize',30);
                if (flag(high(i) + 1) == 1)
                    text(posX(i) + 0.01, posYH(high(i) + 1) + 0.04,'#');
                end
            end
        end
        axis([0 1 -0.2 1.4]);
        set(gca,'XTick',[],'YTick',[]);
        title('Treble');
        subplot(2,1,2);
        hold on;
        plot(x,y*0.9,'k');
        plot(x,y*0.7,'k');
        plot(x,y*0.5,'k');
        plot(x,y*0.3,'k');
        plot(x,y*0.1,'k');
        for i = 1:maxLen
            if (low(i) >= 0)
                plot(posX(i),posYL(low(i) + 1),'-k.','MarkerSize',30);
                if (flag(low(i) + 1) == 1)
                    text(posX(i) + 0.01, posYL(low(i) + 1) + 0.04,'#');
                end
            end
        end
        axis([0 1 -0.4 1.2]);
        set(gca,'XTick',[],'YTick',[]);
        title('Bass');
        saveas(gcf,strcat('pic',clef+48),'png');
        clef = clef + 1;
        break;
    elseif (tmp < 24)
        low(cnt) = tmp;
    elseif (tmp >= 24)
        high(cnt) = tmp - 24;
    end
    if (cnt == maxLen)
        clf;
        subplot(2,1,1);
        hold on;
        plot(x,y*0.9,'k');
        plot(x,y*0.7,'k');
        plot(x,y*0.5,'k');
        plot(x,y*0.3,'k');
        plot(x,y*0.1,'k');
        for i = 1:maxLen
            if (high(i) >= 0)
                plot(posX(i),posYH(high(i) + 1),'-k.','MarkerSize',30);
                if (flag(high(i) + 1) == 1)
                    text(posX(i) + 0.01, posYH(high(i) + 1) + 0.04,'#');
                end
            end
        end
        axis([0 1 -0.2 1.4]);
        set(gca,'XTick',[],'YTick',[]);
        title('Treble');
        subplot(2,1,2);
        hold on;
        plot(x,y*0.9,'k');
        plot(x,y*0.7,'k');
        plot(x,y*0.5,'k');
        plot(x,y*0.3,'k');
        plot(x,y*0.1,'k');
        for i = 1:maxLen
            if (low(i) >= 0)
                plot(posX(i),posYL(low(i) + 1),'-k.','MarkerSize',30);
                if (flag(low(i) + 1) == 1)
                    text(posX(i) + 0.01, posYL(low(i) + 1) + 0.04,'#');
                end
            end
        end
        axis([0 1 -0.4 1.2]);
        set(gca,'XTick',[],'YTick',[]);
        title('Bass');
        cnt = 0;
        high = -ones(1,maxLen);
        low = -ones(1,maxLen);
        saveas(gcf,strcat('pic',clef+48),'png');
        clef = clef + 1;
    end
end
```

Appendix B: process.m

```
strA = ['c','c','d','d','e','f','f','g','g','a','a','b'];
flag = [0,1,0,1,0,0,1,0,1,0,1,0];
light = zeros(48,50);
for k = 1:4
    strC = k + 49;
    for j = 1:12
        if (flag(j) == 0)
            fileName = strcat(strA(j),strC,'.wav');
        else
            fileName = strcat(strA(j),'#',strC,'.wav');
        end
        y = audioread(fileName);
        y = y(:,1);
        f = fft(y);
        f = abs(f);
        f = f(1:50);
        minF = min(f);
        maxF = max(f);
        delta = (maxF - minF) / 8;
        light((k-1)*12+j,:) = min(8,ceil((f - minF) / delta));
        fprintf('{');
```

```
            for i = 1:49
                fprintf('%d,',light((k-1)*12+j,i));
            end
            fprintf('%d},\n',light((k-1)*12+j,50));
        end
    end
```

Appendix C: fft.h

```
#ifndef FFT_H
#define FFT_H

const int lightPos[48][50] = {
{4,1,1,1,2,0,1,1,2,1,1,1,1,1,1,1,2,2,1,2,2,2,1,1,1,2,1,2,3,1,2,2,2,1,
3,3,2,3,3,2,2,3,4,5,8,5,4,5,4,2},
{0,4,4,5,3,3,4,4,3,2,4,5,5,5,5,7,5,6,4,4,5,4,7,6,4,6,5,5,5,5,5,5,6,7,
6,7,5,7,5,7,7,5,7,7,7,7,5,6,5,8},
{0,7,6,6,7,6,6,6,6,6,6,6,6,6,6,6,6,7,7,7,6,6,7,7,7,7,7,6,7,7,7,7,7,7,
7,7,7,7,7,8,8,7,8,8,8,8,8,7,8,8},
{8,5,1,0,2,2,1,1,1,2,2,2,2,2,2,2,2,3,3,3,2,3,4,3,3,4,4,4,4,5,4,4,4,5,
5,5,5,5,5,5,6,6,6,6,6,6,7,7,7,8},
{0,8,6,6,8,7,6,6,6,7,7,6,6,6,6,6,7,7,6,6,6,7,6,7,7,7,7,7,7,7,7,7,7,7,
7,8,7,7,8,7,8,7,7,8,8,8,7,8,8,8},
{8,4,1,2,3,0,1,2,2,1,3,3,3,3,2,3,1,3,1,2,3,3,3,3,3,3,3,3,4,4,4,4,4,5,
5,4,4,5,5,6,5,4,4,5,6,4,7,7,6,4},
{7,3,0,2,1,2,1,2,2,2,3,3,3,4,3,2,2,3,3,4,4,4,3,3,4,4,5,4,4,4,5,5,5,6,
6,6,6,6,6,6,6,6,6,5,6,7,7,7,8,8},
{8,2,0,3,2,2,2,2,2,2,2,2,3,3,4,4,2,3,2,3,3,4,3,3,3,3,3,4,4,4,3,4,4,4,
4,4,4,5,4,4,6,6,5,6,5,5,2,4,5,5},
{0,1,1,2,1,2,2,2,2,2,2,2,1,2,3,2,2,2,2,2,2,2,2,3,3,3,3,3,3,4,3,3,4,4,
4,4,5,5,5,6,6,6,6,7,7,7,8,8,8,8},
{3,2,2,2,2,1,2,2,0,2,2,2,2,1,2,2,3,3,3,4,3,4,3,4,3,4,4,3,4,5,4,5,5,4,
5,6,6,5,5,6,6,6,6,6,6,8,8,7,8,8},
{0,2,2,3,2,1,2,3,4,4,4,3,2,4,3,3,3,4,3,3,4,5,3,5,4,5,6,5,5,5,3,5,4,5,
5,5,7,7,6,5,6,7,7,6,7,8,7,8,8,8},
{3,0,3,5,5,5,4,5,4,4,4,4,5,5,5,5,4,5,4,5,4,5,5,4,5,5,5,5,6,5,5,5,4,4,
4,5,5,5,4,5,5,6,8,7,6,8,7,6,6,7},
{0,4,6,6,5,5,6,5,6,6,6,5,5,6,6,5,5,6,5,6,6,6,5,6,7,6,5,6,6,5,6,6,5,6,
6,7,5,6,6,6,5,5,5,5,6,7,7,7,8,8},
{8,4,1,1,1,1,1,1,2,1,1,1,3,2,0,1,1,1,2,2,3,3,1,2,1,2,2,1,1,2,1,2,2,2,
1,2,3,2,3,4,3,3,4,4,4,4,2,1,1,1},
{4,0,7,7,6,6,6,6,6,5,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,7,6,6,7,7,6,
6,6,6,6,7,7,8,8,6,7,7,7,8,7,8,8},
{0,8,7,3,4,2,1,2,2,2,2,3,3,3,4,4,4,4,4,4,4,4,4,4,5,4,5,5,5,5,5,4,6,6,
6,7,6,6,7,7,7,7,8,6,5,7,7,6,8,8},
{0,6,5,2,3,2,2,2,1,2,2,2,2,2,3,2,2,2,2,2,3,2,2,3,2,3,3,3,3,4,4,3,4,4,
3,3,3,4,5,5,4,4,5,6,6,7,8,7,5,6},
{7,8,3,1,1,1,1,0,1,1,2,3,2,1,2,2,2,2,2,2,3,2,2,2,2,3,3,3,3,4,4,4,4,4,
4,4,5,4,4,4,3,3,5,5,5,5,5,5,5,5},
{0,8,6,5,5,5,5,5,5,5,5,5,5,5,5,5,5,6,5,5,5,5,5,5,5,5,5,5,5,5,6,5,6,6,
6,5,5,6,5,6,5,6,6,5,5,5,5,5,5,5},
{0,8,6,5,5,5,5,5,6,5,5,6,5,5,5,5,5,5,5,5,5,5,5,5,6,6,6,5,5,5,5,6,6,5,
6,5,5,5,5,5,6,6,6,5,6,6,5,5,5,6},
{8,7,1,1,4,2,1,2,0,2,1,2,2,2,2,2,2,2,1,2,2,1,2,2,1,2,2,2,2,1,3,2,2,2,
1,1,1,1,1,2,2,3,2,2,3,3,2,1,1,3},
{8,8,0,2,5,2,2,1,2,1,3,2,3,2,1,1,1,2,2,2,3,3,2,2,3,3,3,2,2,2,3,2,3,2,
1,2,1,1,2,2,2,2,2,1,3,3,3,4,5,4},
{0,8,2,4,3,3,4,4,5,3,5,4,3,5,4,4,5,3,4,4,3,5,4,4,4,5,5,4,4,5,4,3,4,3,
5,4,3,4,4,3,4,4,4,4,3,3,4,5,5,5},
{8,0,2,1,3,1,3,3,2,2,2,2,2,2,3,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,2,
1,2,1,2,1,1,1,2,2,2,2,3,2,2,1,1},
{0,8,2,3,3,3,2,3,2,2,2,2,2,2,3,2,2,3,3,2,2,1,2,2,3,2,2,2,2,2,2,2,1,2,
2,3,3,2,2,3,4,3,5,4,1,1,2,3,1,2},
{0,8,6,6,7,7,7,6,6,6,6,6,7,6,6,7,6,6,7,6,6,6,6,6,7,7,6,6,7,6,6,6,6,6,
6,6,6,7,7,6,7,6,7,6,7,7,7,5,6,6},
{8,3,7,2,7,7,6,4,2,2,2,2,2,1,2,2,3,4,4,4,3,2,3,2,3,3,3,3,3,3,2,2,2,1,
4,2,4,2,3,2,1,1,4,2,3,3,2,3,1,0},
{8,4,6,1,6,4,5,5,4,2,1,1,1,2,1,2,3,2,3,3,3,2,1,1,2,2,2,1,2,3,2,3,2,2,
2,1,3,2,3,2,3,1,1,1,4,3,3,0,3,3},
{8,3,5,1,6,3,5,8,4,3,1,2,2,1,2,2,2,2,2,3,2,3,1,3,2,3,2,3,2,2,3,3,3,2,
2,2,2,1,3,2,2,1,2,2,2,0,1,4,2,4},
{2,1,0,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,4,4,4,4,4,5,5,5,6,6,6,6,6,6,7,7,
7,7,8,8,8,8,8,8,8,8,8,8,8,8,8,8},
{8,5,1,3,5,7,5,5,3,4,3,4,4,5,6,6,8,6,6,6,5,4,4,4,5,5,4,4,5,5,5,7,5,6,
6,4,3,2,2,2,2,1,0,2,8,7,3,2,6,8},
{8,3,1,0,3,3,2,3,3,5,4,5,5,4,3,4,3,5,2,2,4,4,4,5,5,5,4,4,3,3,2,3,4,5,
4,5,5,4,3,2,3,3,2,2,3,1,1,2,2,3},
{0,8,6,3,2,2,3,3,4,4,3,3,4,3,3,3,3,3,4,3,3,4,4,4,5,4,4,4,4,4,4,4,4,4,
4,4,5,4,4,4,3,2,1,2,3,2,4,4,3,2},
{6,8,8,6,4,3,3,2,2,2,2,2,2,1,1,1,1,1,1,1,2,2,1,1,2,2,1,2,1,1,1,1,2,2,
2,1,1,2,2,2,1,0,1,2,4,2,1,3,3,2},
{7,2,3,3,0,2,4,3,5,5,4,4,4,3,4,4,4,4,3,3,3,4,3,3,4,4,4,4,5,4,3,3,5,4,
2,3,3,3,3,4,4,5,4,5,6,7,8,6,4,4},
{4,2,8,8,8,8,2,3,1,2,2,1,1,2,2,3,2,1,2,3,1,1,2,2,1,2,2,2,2,0,2,1,2,1,
2,2,2,2,1,1,3,4,3,4,4,3,3,3,2,4},
{1,3,5,2,1,8,6,4,4,4,2,2,3,3,2,2,2,1,2,2,2,4,3,3,2,3,3,3,3,2,3,4,2,4,
3,3,3,3,4,3,4,3,5,4,2,0,1,2,1,1},
{3,2,7,8,2,8,8,5,6,4,3,3,2,2,2,2,1,2,2,1,2,1,2,1,1,2,2,1,1,2,1,2,2,1,
2,2,1,2,3,2,2,0,1,2,1,3,6,5,4,4},
{1,4,2,2,4,1,1,1,2,1,1,3,3,2,1,1,1,1,3,1,1,1,3,2,2,3,4,5,5,2,2,2,2,2,
1,2,0,3,5,2,2,3,3,1,4,2,8,3,4,6},
{2,4,2,3,3,1,2,2,0,2,2,2,2,4,3,3,2,1,1,1,1,1,1,2,1,1,2,3,4,4,6,4,2,3,
1,2,1,2,1,4,4,4,2,1,3,2,1,5,7,8},
{2,3,3,3,4,4,3,2,2,3,2,1,0,3,6,7,6,6,1,2,2,3,2,1,3,5,1,3,1,4,8,4,8,6,
3,6,1,3,2,1,3,6,4,5,2,2,3,6,5,2},
{1,2,2,1,1,3,7,3,4,6,8,7,5,3,1,1,2,3,2,2,1,4,2,3,1,1,1,1,2,2,2,2,1,0,
1,1,1,1,1,2,3,4,3,3,2,2,2,4,5,8},
{0,2,1,2,1,1,6,3,5,4,2,8,3,4,2,1,1,2,2,1,1,1,2,1,3,1,1,1,1,1,1,2,1,1,
2,1,1,1,2,1,1,3,3,4,3,2,1,1,1,3},
{0,2,1,1,1,1,3,5,2,4,3,3,8,4,4,2,2,1,1,2,1,2,1,1,3,3,1,1,1,1,1,2,1,2,
1,1,2,1,1,1,2,1,1,2,2,3,3,2,1,1},
{1,3,1,1,1,1,0,1,2,4,8,3,2,6,8,4,4,2,2,1,1,1,1,1,1,1,2,1,1,1,1,2,1,1,
1,1,2,1,2,1,2,2,2,1,1,1,1,1,1,3},
{1,2,1,1,1,1,1,2,2,3,1,8,3,2,7,7,4,4,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,2,
1,1,1,1,1,1,2,1,1,2,0,1,1,1,1,1},
{1,1,2,2,1,2,2,1,3,1,4,4,6,5,4,8,5,5,3,2,1,1,1,1,1,1,1,1,1,1,1,1,0,1,
1,2,1,1,1,1,2,1,1,1,1,3,2,2,1,1},
{1,3,1,2,3,2,3,3,3,2,3,5,6,4,3,3,3,4,5,5,4,5,3,3,2,0,1,3,3,2,1,2,2,3,
3,2,1,2,4,5,4,5,8,5,4,5,4,3,4,2}};
#endif
```

Appendix D: sound.h

```
#ifndef SOUND_H
#define SOUND_H
#define FFTINPUTSIZE 1024
#define FFTOUTPUTSIZE 50
#define FFTPERIOD 31250 //0.2s*10MHz/64(PreScaler = 1:64)
#define LEDPERIOD 625   //FFTPERIOD/FFTOUTPUTSIZE(PreScaler = 1:64)
#define PWMPERIOD 227   //1s/44100Hz*10MHz
#define TIMEOUT 110250
#define A0 1
#define A1 2
#define A2 3
#define A3 4
#define A4 5
#define A5 6
#define A6 7
const int Table47[45] = {-47, -45, -44, -42, -39, -37, -36, -34, -31, -29,
-26, -22, -20, -14, -12, -4, 0, 5, 11, 14, 19, 23, 26, 30, 32, 33, 34,
30, 29, 22, 19, 14, 7, 3, -4, -9, -12, -18, -23, -27, -33, -36, -39, -41,
-41};
const int Table46[47] = {-31, -30, -27, -23, -18, -12, -7, -2, 4, 11, 18,
24, 29, 34, 38, 42, 44, 45, 45, 43, 42, 39, 35, 30, 24, 17, 10, 2, -5,
-10, -14, -14, -14, -13, -13, -13, -14, -15, -17, -19, -20, -22, -24, -27,
-31, -33, -35};
const int Table45[50] = {-26, -26, -25, -22, -20, -18, -18, -18, -18, -19,
-20, -21, -21, -21, -19, -16, -12, -7, -2, 5, 12, 19, 26, 32, 37, 41, 44,
45, 45, 42, 39, 36, 33, 29, 26, 21, 17, 11, 6, 0, -5, -10, -14, -17, -19,
-21, -23, -25, -27, -27};
const int Table44[53] = {-27, -26, -26, -25, -23, -22, -21, -19, -17, -14,
-12, -10, -8, -7, -6, -5, -4, -4, -3, -1, 3, 7, 11, 15, 20, 25, 30, 35,
40, 42, 45, 45, 45, 43, 40, 36, 31, 25, 19, 13, 6, 0, -6, -11, -14, -17,
-19, -21, -23, -24, -26, -26, -26};
const int Table43[56] = {-38, -35, -31, -25, -18, -11, -7, -6, -7, -7,
-6, -5, -4, -2, 0, 3, 7, 7, 4, -3, -9, -15, -17, -16, -11, -3, 5, 11, 15,
20, 24, 31, 37, 41, 45, 45, 44, 38, 32, 25, 20, 16, 12, 5, -2, -7, -7,
-8, -8, -12, -16, -21, -28, -33, -39, -41};
const int Table42[60] = {-45, -45, -41, -34, -28, -22, -20, -19, -21, -24,
-25, -25, -22, -21, -19, -17, -13, -8, -1, 5, 9, 10, 6, 1, -3, -6, -6,
-5, -2, 0, 2, 0, -3, -6, -5, -3, 4, 10, 17, 22, 26, 30, 33, 37, 37, 37,
35, 33, 31, 27, 24, 20, 13, 6, -6, -15, -26, -35, -40, -41};
const int Table41[63] = {-45, -42, -38, -32, -27, -22, -19, -16, -12, -8,
-5, -3, -2, 0, -1, 0, -2, -1, 0, 1, 2, 2, 1, -1, -1, 0, 2, 6, 9, 11, 8,
4, -2, -7, -10, -8, -1, 8, 16, 22, 26, 29, 33, 35, 36, 34, 31, 26, 21,
16, 12, 10, 8, 6, 4, -1, -7, -16, -25, -33, -38, -40, -40};
const int Table40[67] = {-45, -44, -42, -42, -41, -35, -22, -7, 9, 23,
33, 37, 32, 22, 9, -1, -11, -20, -25, -23, -16, -7, 1, 4, 1, -7, -17, -26,
-32, -32, -26, -16, -5, 5, 12, 16, 14, 5, -6, -15, -15, -10, 2, 16, 32,
43, 42, 32, 15, 0, -7, -5, 3, 13, 23, 31, 37, 38, 32, 22, 11, 1, -10, -17,
-23, -27, -33};
const int Table39[71] = {-45, -42, -34, -24, -14, -5, 2, 7, 8, 6, 0, -4,
-5, -3, -2, -2, -2, 0, 2, 2, 3, 3, 6, 10, 15, 19, 19, 18, 11, 3, -6, -13,
-15, -11, -4, 3, 5, 4, 1, -2, -8, -15, -22, -29, -33, -35, -33, -28, -20,
-14, -12, -14, -20, -23, -22, -18, -12, -4, 4, 13, 21, 26, 24, 19, 8, -3,
-17, -29, -40, -45, -45};
const int Table38[75] = {-42, -38, -30, -23, -15, -8, -2, 1, 1, 0, -4,
-9, -15, -20, -22, -20, -12, -1, 8, 15, 16, 20, 22, 27, 29, 28, 25, 20,
13, 1, -13, -26, -33, -34, -32, -31, -30, -28, -23, -14, -3, 8, 15, 21,
24, 28, 31, 33, 34, 36, 41, 45, 45, 37, 24, 10, -1, -7, -12, -12, -6, 5,
19, 32, 40, 44, 43, 36, 21, 2, -15, -28, -35, -37, -38};
const int Table37[80] = {-45, -44, -42, -39, -37, -34, -31, -28, -25, -23,
-19, -16, -12, -7, -3, 1, 5, 7, 9, 9, 11, 11, 12, 12, 14, 16, 19, 23, 27,
31, 33, 33, 33, 29, 26, 22, 19, 18, 17, 18, 18, 19, 18, 17, 15, 15, 15,
17, 20, 22, 24, 24, 22, 18, 14, 9, 6, 3, 0, -3, -6, -10, -14, -20, -25,
-31, -34, -36, -36, -36, -34, -34, -34, -35, -36, -38, -40, -42, -42, -42};
const int Table36[84] = {-45, -44, -44, -41, -40, -37, -35, -34, -34, -34,
-35, -36, -36, -34, -31, -26, -21, -16, -12, -8, -5, -3, 0, 2, 7, 12, 18,
23, 28, 30, 31, 30, 29, 27, 28, 29, 31, 34, 37, 39, 40, 40, 39, 38, 37,
36, 37, 38, 39, 38, 38, 35, 31, 27, 23, 21, 19, 20, 21, 23, 26, 28, 30,
30, 29, 27, 23, 19, 13, 9, 3, -1, -6, -11, -15, -20, -23, -27, -31, -35,
-37, -40, -41, -42};
const int Table35[89] = {-43, -42, -41, -38, -36, -33, -30, -28, -26, -25,
-24, -22, -21, -19, -16, -14, -12, -11, -10, -10, -9, -8, -7, -4, 0, 4,
9, 14, 17, 20, 21, 21, 21, 22, 22, 23, 25, 26, 28, 28, 28, 27, 25, 23,
21, 21, 21, 22, 24, 26, 26, 27, 26, 24, 22, 19, 17, 15, 14, 14, 14, 14,
14, 14, 13, 12, 10, 8, 6, 3, 0, -2, -4, -6, -7, -8, -9, -10, -12, -14,
-18, -22, -26, -32, -36, -40, -43, -44, -45};
const int Table34[95] = {-45, -44, -42, -40, -38, -36, -34, -31, -28, -26,
-23, -20, -17, -14, -11, -8, -4, -1, 2, 5, 8, 11, 13, 15, 17, 18, 20, 22,
25, 27, 30, 31, 32, 33, 34, 34, 34, 34, 33, 32, 31, 31, 30, 29, 28, 28,
27, 26, 26, 25, 25, 25, 24, 23, 23, 22, 21, 20, 18, 16, 13, 11, 8, 5, 3,
1, -1, -2, -4, -6, -7, -9, -11, -13, -15, -17, -19, -21, -22, -24, -25,
-27, -28, -30, -31, -33, -35, -36, -38, -40, -42, -44, -45, -45, -45};
const int Table33[100] = {-45, -45, -44, -43, -42, -40, -38, -36, -33,
-30, -27, -23, -18, -14, -10, -6, -2, 1, 3, 6, 8, 11, 13, 16, 18, 21, 24,
26, 28, 30, 32, 34, 35, 36, 37, 37, 37, 37, 38, 39, 39, 39, 39, 38, 38,
37, 36, 35, 33, 31, 29, 28, 26, 26, 25, 25, 24, 24, 23, 23, 22, 20, 18,
15, 13, 10, 8, 5, 2, 0, -3, -5, -7, -10, -12, -13, -15, -17, -19, -20,
-22, -23, -24, -26, -27, -29, -31, -33, -35, -37, -39, -41, -42, -42, -43,
-43, -44, -44, -45, -45};
const int Table32[106] = {-45, -45, -44, -43, -43, -42, -42, -41, -40,
-38, -36, -34, -32, -30, -27, -24, -21, -18, -14, -11, -8, -5, -1, 2, 5,
9, 12, 15, 18, 21, 23, 26, 28, 30, 31, 32, 33, 33, 34, 35, 37, 38, 39,
40, 41, 42, 42, 42, 42, 42, 41, 40, 39, 37, 36, 35, 34, 33, 31, 30, 29,
28, 27, 25, 24, 23, 21, 19, 17, 16, 14, 13, 11, 10, 9, 7, 5, 3, 1, -1,
-4, -7, -11, -14, -17, -19, -21, -22, -23, -25, -26, -28, -28, -29, -30,
-31, -32, -34, -36, -38, -39, -41, -42, -43, -43, -44};
const int Table31[113] = {-43, -43, -42, -41, -40, -39, -39, -39, -38,
-37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -26, -23, -21, -18, -15,
-13, -11, -9, -8, -6, -5, -3, -1, 1, 4, 6, 9, 12, 14, 16, 18, 20, 22, 24,
26, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 39, 40, 41, 41, 42, 42, 43,
44, 45, 45, 44, 43, 41, 39, 38, 36, 35, 33, 32, 31, 30, 29, 28, 26, 24,
22, 19, 17, 14, 11, 7, 4, 1, -2, -5, -8, -10, -13, -16, -20, -23, -26,
-29, -32, -34, -36, -38, -39, -39, -40, -41, -42, -43, -43, -44, -44, -44,
-44, -44};
const int Table30[119] = {-44, -44, -43, -43, -42, -41, -40, -39, -38,
-36, -36, -35, -34, -34, -33, -33, -33, -32, -31, -29, -26, -24, -21, -20,
-19, -17, -16, -14, -12, -10, -8, -6, -4, -2, 1, 3, 5, 8, 10, 12, 14, 16,
17, 19, 20, 22, 23, 25, 27, 29, 31, 33, 35, 36, 37, 39, 40, 41, 41, 41,
42, 42, 42, 42, 43, 43, 43, 43, 43, 43, 43, 43, 42, 41, 40, 39, 38, 36,
34, 33, 32, 30, 29, 27, 25, 22, 18, 15, 13, 11, 9, 7, 6, 3, 1, -2, -5,
-8, -12, -15, -19, -23, -26, -29, -31, -34, -35, -37, -39, -40, -40, -41,
-42, -43, -44, -45, -45, -45, -45};
const int Table29[126] = {-45, -45, -45, -44, -43, -42, -41, -40, -39,
-39, -38, -37, -35, -34, -32, -31, -31, -30, -30, -29, -29, -28, -27, -25,
-24, -22, -20, -18, -17, -15, -14, -12, -10, -9, -7, -6, -4, -3, -1, 1,
4, 6, 9, 11, 13, 15, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28, 30, 31, 33,
34, 35, 36, 37, 37, 38, 38, 38, 39, 39, 40, 40, 40, 40, 40, 40, 40, 40,
41, 40, 39, 38, 36, 34, 33, 31, 31, 30, 29, 28, 27, 25, 24, 22, 20, 17,
14, 11, 9, 6, 4, 2, 0, -3, -5, -7, -10, -12, -14, -17, -19, -21, -24, -27,
-29, -31, -33, -35, -36, -38, -40, -42, -43, -43, -43, -43, -43};
const int Table28[134] = {-39, -39, -37, -35, -31, -28, -25, -23, -21,
-21, -20, -21, -21, -23, -25, -27, -30, -31, -31, -30, -28, -24, -20, -15,
-10, -5, -1, 2, 3, 3, 3, 2, 2, 3, 5, 6, 8, 10, 13, 16, 19, 22, 24, 25,
26, 24, 22, 19, 15, 12, 9, 6, 3, 0, -3, -7, -10, -12, -14, -16, -18, -19,
-21, -22, -22, -21, -19, -18, -17, -16, -16, -16, -15, -15, -15, -14, -13,
-12, -9, -4, 1, 8, 15, 21, 27, 31, 33, 34, 34, 33, 33, 34, 35, 37, 40,
42, 44, 44, 45, 44, 43, 41, 40, 39, 37, 35, 33, 29, 25, 21, 17, 13, 11,
10, 9, 8, 6, 3, -1, -4, -8, -10, -12, -15, -18, -22, -26, -29, -32, -35,
-37, -38, -38, -38};
const int Table27[142] = {-45, -44, -42, -38, -35, -32, -30, -29, -29,
-30, -30, -29, -28, -27, -25, -23, -21, -19, -17, -14, -11, -9, -6, -5,
-4, -4, -5, -7, -8, -10, -11, -12, -12, -10, -6, 0, 6, 12, 17, 21, 24,
26, 27, 27, 28, 27, 27, 25, 24, 21, 18, 15, 12, 10, 8, 6, 4, 1, -1, -3,
-4, -5, -6, -7, -9, -11, -15, -19, -24, -28, -31, -34, -34, -34, -32, -29,
-25, -20, -14, -9, -4, 0, 4, 8, 11, 14, 16, 17, 17, 16, 16, 16, 17, 20,
22, 27, 31, 36, 40, 43, 45, 45, 45, 44, 43, 42, 40, 38, 35, 33, 31, 29,
27, 26, 24, 21, 18, 14, 11, 9, 7, 5, 2, -2, -5, -8, -10, -12, -14, -16,
-18, -21, -25, -30, -34, -38, -41, -43, -43, -43, -43, -43};
const int Table26[150] =
{-45,-45,-44,-41,-37,-34,-32,-30,-30,-30,-31,-30,-30,-29,-27,-25,-24,
-22,-20,-18,-16,-13,-11,-8,-6,-5,-5,-5,-6,-7,-8,-10,-11,-12,-12,-10,
-7,-2,3,9,14,18,21,24,26,27,27,28,27,27,26,25,22,19,16,13,11,9,7,6,3,
1,-1,-3,-4,-5,-5,-7,-8,-11,-14,-18,-22,-26,-30,-32,-33,-34,-33,-31,-2
8,-24,-19,-14,-9,
-5,0,3,7,10,13,15,17,17,17,17,16,17,18,21,23,27,31,36,40,43,45,45,45,
45,44,43,42,40,37,35,33,31,29,28,26,24,22,19,15,12,10,8,6,3,0,-3,-6,-
8,-11,-13,-14,-16,-18,-20,-24,-29,-33,-37,-40,-42,-43,-43,-43,-43};
const int Table25[159] = {-44, -44, -43, -41, -39, -37, -35, -33, -31,
-30, -29, -28, -27, -25, -24, -23, -22, -21, -20, -18, -17, -14, -12, -9,
-5, -2, 1, 4, 7, 11, 14, 17, 20, 23, 24, 25, 25, 24, 22, 18, 14, 10, 6,
3, 0, -1, -2, -3, -5, -6, -8, -8, -8, -7, -4, -1, 3, 6, 8, 10, 10, 11,
11, 11, 11, 11, 12, 12, 11, 10, 8, 5, 1, -2, -5, -7, -8, -7, -7, -7, -7,
-8, -8, -9, -9, -8, -6, -4, -2, 0, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 20,
22, 23, 24, 25, 26, 27, 29, 31, 32, 34, 35, 36, 37, 37, 37, 38, 38, 39,
40, 40, 40, 39, 37, 33, 29, 24, 19, 14, 9, 4, -1, -4, -7, -9, -11, -12,
-14, -15, -16, -17, -19, -20, -21, -22, -23, -25, -26, -28, -31, -33, -36,
-39, -41, -43, -45, -45, -45, -45};
const int Table24[169] =
{-45,-45,-44,-43,-41,-39,-36,-34,-31,-29,-27,-26,-25,-25,-24,-23,
22,-21,-19,-17,-16,-13,-11,-8,-6,-3,0,3,6,10,13,16,19,21,22,23,24,24,
23,22,21,8,15,11,7,2,-1,-4,-6,-7,-8,-7,-6,-5,-4,-3,-2,-1,1,2,4,7,10,1
3,15,16,15,14,13,11,10,9,8,7,6,5,3,0,-2,-5,-8,-9,-10,-10,-10,-8,-7,-5
,-4,-3,-2,-2,-2,-1,1,3,6,8,11,13,14,15,16,17,18,21,23,26,28,30,32,32,
33,33,34,35,36,37,39,40,41,42,43,43,42,41,40,39,38,37,35,33,29,25,20,
15,9,4,-1,-5,-8,-11,-13,-14,-15,-16,-16,-17,-18,-19,-20,-21,-23,-25,-
26,-28,-30,-31,-33,-35,-36,-39,-41,-43,-44,-45,-45,-45,-45};
const int Table23[179] = {-45, -45, -44, -43, -41, -39, -37, -35, -33,
-31, -29, -27, -24, -21, -18, -15, -13, -11, -10, -10, -9, -9, -7, -6,
-4, -1, 1, 3, 6, 8, 9, 11, 12, 14, 15, 17, 18, 20, 22, 23, 24, 24, 23,
20, 17, 13, 10, 7, 4, 1, 0, -3, -4, -6, -7, -8, -9, -8, -7, -6, -5, -3,
-1, 2, 4, 7, 10, 12, 14, 16, 16, 17, 16, 15, 13, 11, 9, 7, 5, 3, 1, -1,
-2, -3, -4, -5, -6, -7, -8, -9, -9, -8, -7, -6, -4, -3, -1, 1, 2, 4, 6,
9, 11, 14, 16, 17, 19, 20, 21, 22, 22, 24, 25, 27, 29, 31, 32, 33, 34,
35, 35, 36, 36, 36, 36, 37, 38, 38, 39, 39, 38, 36, 35, 33, 31, 30, 28,
27, 25, 23, 21, 18, 15, 10, 5, 0, -4, -9, -13, -16, -18, -20, -22, -24,
-26, -28, -29, -31, -32, -32, -32, -32, -31, -30, -29, -28, -29, -30, -32,
-35, -37, -39, -40, -41, -42, -42, -42, -43, -43};
const int Table22[189] = {-45, -44, -42, -40, -39, -38, -37, -36, -35,
-35, -34, -34, -33, -33, -32, -31, -30, -30, -29, -30, -29, -28, -26, -23,
-19, -16, -13, -11, -9, -8, -8, -8, -8, -8, -8, -7, -6, -4, -2, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, -1, -1, -3, -4, -5, -5, -5, -4, -2, 0, 3, 5, 8, 11,
13, 14, 14, 14, 12, 11, 10, 8, 7, 5, 3, 0, -2, -5, -8, -11, -13, -13, -13,
-12, -11, -9, -7, -4, -1, 2, 4, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 21,
24, 27, 30, 33, 36, 38, 40, 42, 42, 42, 41, 39, 38, 36, 36, 35, 35, 34,
32, 29, 27, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 26, 26, 25, 24, 24,
23, 24, 25, 26, 26, 25, 23, 22, 21, 20, 19, 19, 18, 16, 14, 12, 11, 10,
9, 8, 5, 3, 1, -1, -3, -5, -7, -10, -13, -17, -20, -22, -24, -24, -25,
-24, -23, -23, -22, -23, -24, -26, -28, -31, -33, -36, -38, -39, -39, -39,
-39, -40};
const int Table21[200] = {-45, -44, -42, -40, -38, -37, -36, -36, -37,
-37, -39, -39, -40, -40, -39, -39, -37, -36, -34, -32, -30, -28, -27, -26,
-24, -23, -21, -19, -17, -16, -14, -13, -13, -12, -12, -12, -10, -8, -6,
-4, -2, -1, 1, 3, 4, 4, 3, 1, -1, -2, -3, -4, -4, -4, -5, -5, -5, -4, -3,
-2, 0, 1, 3, 5, 7, 10, 12, 14, 16, 17, 16, 15, 14, 12, 11, 10, 8, 5, 2,
-1, -4, -7, -9, -10, -11, -10, -9, -9, -8, -6, -5, -2, 0, 1, 2, 2, 2, 3,
4, 7, 9, 11, 12, 14, 16, 18, 21, 24, 27, 31, 34, 36, 37, 39, 40, 41, 41,
41, 40, 39, 38, 37, 36, 36, 35, 35, 34, 34, 33, 31, 31, 30, 29, 28, 28,
27, 27, 27, 27, 27, 27, 26, 26, 26, 25, 25, 25, 26, 27, 27, 27, 26, 24,
22, 21, 19, 18, 18, 17, 17, 17, 17, 17, 17, 15, 14, 11, 8, 4, 1, -3, -6,
-8, -10, -11, -13, -14, -16, -17, -19, -20, -20, -20, -20, -20, -20, -22,
-23, -26, -28, -31, -34, -37, -39, -41, -41, -41, -41, -41, -41};
const int Table20[212] = {-45, -44, -42, -40, -38, -37, -35, -34, -33,
-32, -30, -30, -29, -30, -30, -31, -31, -30, -28, -26, -24, -23, -21, -20,
-20, -19, -17, -14, -12, -9, -7, -5, -4, -4, -3, -3, -3, -4, -4, -4, -4,
-4, -3, -1, 1, 3, 4, 4, 4, 3, 1, 0, -1, -1, -2, -2, -2, -2, -2, -1, 0,
1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 15, 16, 15, 13, 11, 8, 6, 4, 2, 0, -2,
-4, -6, -7, -8, -9, -9, -9, -10, -10, -9, -8, -6, -4, -1, 2, 6, 9, 12,
14, 15, 16, 17, 17, 18, 19, 20, 22, 23, 25, 26, 28, 30, 32, 34, 36, 37,
39, 39, 40, 40, 40, 40, 39, 38, 36, 34, 32, 30, 29, 27, 26, 26, 26, 25,
25, 24, 24, 24, 24, 24, 24, 24, 23, 24, 24, 24, 24, 24, 23, 22, 21, 20,
20, 19, 19, 19, 20, 20, 19, 18, 16, 14, 13, 11, 10, 9, 8, 6, 4, 2, 1, 0,
0, -2, -3, -6, -9, -11, -13, -15, -17, -18, -20, -22, -24, -26, -28, -29,
-29, -30, -29, -29, -29, -30, -30, -30, -31, -31, -32, -33, -35, -37, -40,
-43, -44, -45, -45, -45, -45, -45};
const int Table19[225] = {-37, -37, -37, -36, -36, -35, -34, -33, -32,
-31, -30, -29, -28, -27, -26, -24, -23, -22, -21, -20, -20, -19, -19, -19,
-18, -18, -17, -16, -15, -13, -11, -9, -7, -4, -2, 0, 1, 3, 4, 5, 5, 6,
6, 6, 5, 5, 5, 5, 5, 4, 4, 4, 4, 3, 3, 3, 2, 2, 1, 0, 0, -1, -1, -1, -1,
-1, -1, -1, -1, 0, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0,
0, -1, -1, -2, -2, -3, -4, -4, -5, -5, -6, -6, -6, -6, -6, -5, -4, -3,
-2, -1, 1, 2, 3, 4, 5, 6, 7, 7, 8, 9, 9, 10, 10, 11, 11, 11, 12, 13, 14,
14, 15, 16, 16, 17, 17, 18, 18, 19, 20, 21, 22, 23, 25, 26, 27, 29, 30,
32, 34, 35, 37, 39, 40, 42, 43, 44, 45, 45, 45, 45, 44, 43, 41, 40, 38,
36, 35, 33, 32, 31, 31, 30, 29, 28, 27, 26, 25, 24, 22, 21, 20, 19, 17,
16, 14, 12, 10, 8, 6, 4, 1, -1, -4, -7, -10, -13, -15, -18, -20, -22, -24,
-25, -27, -28, -29, -29, -29, -29, -29, -28, -28, -27, -26, -26, -25, -25,
-26, -27, -28, -29, -30, -31, -33, -34, -35, -36, -37, -37, -38, -38};
const int Table18[238] = {-39, -39, -39, -38, -38, -36, -35, -33, -32,
-30, -29, -27, -26, -25, -24, -22, -22, -21, -20, -20, -19, -19, -18, -17,
-16, -15, -14, -13, -11, -10, -9, -8, -6, -5, -3, -2, 0, 2, 4, 5, 7, 8,
8, 9, 9, 8, 8, 7, 6, 5, 4, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 0,
0, 0, -1, -1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, -1, 0,
-1, -1, -1, -1, -1, -1, -1, -2, -2, -2, -2, -3, -3, -3, -3, -3, -2, -1,
0, 1, 2, 4, 5, 6, 8, 9, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15,
16, 16, 16, 17, 18, 18, 19, 20, 21, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 33, 34, 35, 36, 38, 39, 41, 42, 43, 44, 45, 45, 45, 45, 44, 43,
42, 41, 40, 39, 37, 36, 34, 33, 31, 29, 27, 26, 24, 23, 22, 21, 21, 20,
19, 18, 17, 16, 15, 13, 11, 9, 7, 5, 2, 0, -3, -5, -7, -9, -12, -14, -16,
-18, -20, -22, -24, -26, -28, -30, -31, -32, -32, -33, -32, -32, -32, -31,
-31, -30, -30, -29, -29, -29, -28, -29, -29, -29, -30, -31, -32, -33, -34,
-36, -36, -38, -38, -39, -39, -40, -40};
```

```
const int Table17[253] = {-42, -41, -41, -39, -38, -36, -34, -32, -30,
-28, -26, -25, -23, -22, -20, -19, -18, -17, -16, -15, -15, -14, -13, -13,
-12, -11, -10, -10, -9, -8, -6, -5, -4, -2, 0, 1, 3, 5, 6, 8, 9, 9, 10,
10, 10, 10, 10, 9, 9, 8, 8, 7, 6, 5, 4, 3, 2, 1, 0, 0, -1, -1, -1, -1,
0, 0, 1, 1, 2, 2, 3, 3, 3, 3, 2, 2, 1, 1, 1, 0, 0, 0, 0, 1, 1, 2, 2, 3,
3, 3, 2, 2, 1, 1, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 1, 1,
2, 3, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 18, 19, 19,
19, 20, 20, 20, 20, 20, 20, 20, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30,
31, 32, 33, 34, 35, 36, 36, 37, 38, 39, 40, 41, 42, 43, 43, 44, 45, 45,
45, 45, 45, 44, 43, 42, 41, 40, 38, 36, 35, 33, 31, 29, 27, 25, 23, 21,
19, 17, 16, 14, 13, 12, 11, 10, 9, 8, 7, 5, 4, 2, 1, -1, -3, -5, -8, -10,
-12, -15, -17, -20, -22, -25, -27, -29, -31, -33, -34, -35, -36, -37, -38,
-38, -38, -38, -38, -38, -38, -37, -37, -37, -37, -36, -36, -36, -36, -35,
-34, -34, -33, -33, -34, -34, -35, -36, -37, -38, -39, -40, -41, -42, -42,
-42, -42, -42, -42, -42};
const int Table16[268] = {-45, -45, -44, -43, -42, -40, -38, -36, -33,
-31, -29, -27, -27, -26, -25, -24, -23, -22, -21, -20, -18, -17, -15, -15,
-15, -15, -16, -16, -17, -17, -18, -18, -17, -17, -17, -17, -17, -18, -19,
-19, -19, -19, -19, -19, -19, -18, -17, -16, -15, -14, -13, -11, -9, -7,
-6, -5, -4, -4, -4, -5, -6, -6, -7, -7, -8, -8, -8, -8, -7, -5, -4, -3,
-3, -3, -4, -5, -6, -7, -7, -7, -7, -7, -6, -5, -3, -2, -2, -2, -2, -3,
-5, -6, -7, -8, -9, -10, -12, -13, -15, -17, -19, -20, -21, -21, -22, -22,
-22, -22, -21, -19, -18, -17, -16, -14, -12, -8, -4, 1, 5, 9, 12, 14, 17,
19, 21, 22, 22, 21, 20, 19, 19, 19, 19, 20, 20, 21, 21, 23, 25, 27, 30,
32, 33, 34, 35, 35, 35, 35, 34, 34, 32, 31, 30, 29, 28, 28, 28, 28, 27,
25, 24, 23, 23, 24, 26, 28, 29, 31, 33, 35, 36, 37, 38, 38, 37, 35, 34,
32, 30, 28, 26, 25, 23, 21, 19, 17, 15, 15, 14, 13, 12, 11, 11, 11, 13,
14, 15, 14, 13, 12, 11, 11, 11, 11, 10, 9, 7, 6, 4, 2, 0, -2, -4, -5, -6,
-7, -6, -5, -4, -3, -1, 0, 1, 2, 2, 3, 4, 5, 7, 8, 9, 9, 9, 8, 8, 7, 5,
4, 3, 3, 3, 4, 5, 5, 5, 5, 4, 3, 1, -1, -4, -6, -9, -12, -15, -19, -22,
-26, -29, -32, -35, -38, -40, -42, -43, -44, -44, -44};
const int Table15[283] = {-42, -42, -41, -40, -38, -37, -35, -33, -31,
-28, -26, -22, -19, -15, -13, -10, -9, -9, -8, -8, -8, -9, -10, -11, -12,
-12, -13, -13, -14, -15, -16, -17, -19, -20, -21, -21, -20, -19, -18, -17,
-15, -14, -13, -13, -13, -12, -12, -12, -12, -13, -13, -12, -11, -10, -9,
-8, -7, -6, -5, -5, -4, -3, -2, -2, -1, -1, -1, -2, -2, -4, -5, -6, -6,
-7, -8, -8, -8, -7, -6, -5, -5, -4, -4, -4, -4, -4, -6, -7, -7, -8, -7,
-7, -7, -6, -6, -6, -6, -7, -8, -10, -11, -12, -13, -14, -15, -15, -15,
-14, -14, -13, -13, -12, -11, -10, -9, -7, -5, -2, 1, 4, 7, 10, 12, 14,
15, 17, 18, 20, 23, 26, 29, 32, 34, 36, 37, 38, 37, 36, 35, 32, 30, 27,
26, 25, 25, 26, 26, 27, 27, 28, 28, 29, 29, 29, 30, 30, 30, 31, 32, 32,
33, 32, 32, 31, 31, 31, 31, 30, 28, 27, 27, 26, 25, 23, 21, 20, 19, 19,
20, 21, 21, 22, 24, 27, 29, 31, 33, 32, 31, 28, 26, 23, 20, 17, 14, 11,
8, 6, 4, 2, 1, 0, -1, -1, -1, 0, 0, 0, 1, 2, 4, 5, 6, 6, 6, 5, 4, 3, 2,
2, 1, 0, -1, -2, -2, -3, -3, -4, -5, -6, -7, -7, -8, -7, -6, -3, -1, 1,
3, 5, 6, 8, 9, 9, 10, 10, 9, 8, 7, 5, 4, 3, 2, 1, -1, -3, -5, -8, -10,
-12, -15, -17, -19, -20, -22, -23, -25, -26, -28, -29, -30, -31, -32, -33,
-34, -36, -37, -38, -39, -40, -42, -44, -45};
const int Table14[300] = {-43, -43, -41, -39, -37, -35, -32, -29, -26,
-23, -20, -18, -17, -16, -15, -14, -13, -13, -12, -10, -9, -7, -6, -6,
-6, -6, -8, -10, -12, -14, -16, -19, -21, -23, -24, -25, -25, -24, -23,
-22, -21, -20, -19, -17, -15, -14, -13, -11, -10, -9, -8, -7, -7, -7, -7,
-8, -8, -8, -8, -8, -7, -7, -6, -5, -4, -4, -3, -3, -3, -2, -2, -1, -1,
-1, -2, -3, -3, -4, -5, -5, -6, -8, -9, -10, -10, -10, -10, -10, -9, -7,
-5, -4, -3, -3, -3, -4, -5, -6, -7, -8, -9, -9, -10, -11, -11, -10, -10,
-10, -10, -10, -11, -11, -11, -11, -9, -8, -7, -6, -5, -3, -2, 0, 2, 4,
7, 9, 12, 16, 20, 23, 27, 30, 32, 34, 36, 37, 38, 38, 38, 37, 37, 38, 39,
39, 40, 40, 39, 39, 39, 38, 38, 36, 34, 32, 30, 28, 27, 26, 26, 27, 27,
28, 29, 30, 32, 33, 34, 35, 36, 36, 37, 37, 37, 36, 35, 34, 33, 31, 29,
26, 24, 22, 21, 20, 20, 21, 22, 23, 25, 26, 26, 26, 25, 24, 23, 21, 20,
20, 19, 19, 20, 20, 20, 20, 20, 19, 17, 14, 10, 7, 3, -1, -4, -7, -8, -9,
-10, -10, -9, -8, -7, -5, -3, -1, 1, 3, 5, 7, 9, 9, 8, 6, 4, 2, -1, -3,
-5, -7, -8, -9, -9, -8, -7, -6, -5, -4, -3, -2, -1, -1, 0, 0, 1, 2, 3,
4, 6, 7, 8, 8, 8, 8, 7, 6, 4, 2, 0, -2, -4, -7, -10, -13, -16, -19, -22,
-25, -27, -30, -32, -34, -35, -35, -35, -35, -34, -34, -35, -36, -37, -38,
-40, -42, -43, -44, -45, -45, -45, -45, -45, -45};
const int Table13[318] = {-37, -37, -36, -34, -32, -31, -29, -27, -25,
-23, -21, -20, -19, -19, -18, -18, -18, -18, -18, -18, -19, -19, -20, -21,
-21, -21, -20, -17, -15, -11, -6, -2, 1, 4, 5, 6, 5, 3, 2, 2, 2, 3, 5,
7, 8, 10, 12, 14, 15, 15, 15, 15, 15, 14, 14, 14, 15, 17, 19, 21, 24, 26,
28, 31, 33, 34, 36, 37, 37, 38, 39, 39, 38, 37, 36, 34, 32, 30, 29, 29,
29, 31, 32, 34, 36, 38, 39, 40, 40, 39, 39, 39, 39, 40, 41, 42, 43, 44,
45, 45, 45, 44, 42, 40, 38, 35, 34, 33, 33, 34, 36, 37, 38, 38, 37, 35,
32, 28, 25, 21, 18, 16, 14, 13, 12, 11, 10, 8, 6, 5, 3, 1, 0, -1, -2, -2,
-2, -1, 0, 2, 3, 5, 5, 6, 6, 5, 3, 1, -2, -5, -9, -11, -14, -17, -20, -23,
-25, -27, -28, -29, -29, -28, -27, -25, -24, -22, -20, -18, -18, -18, -19,
-21, -24, -26, -28, -30, -32, -33, -33, -33, -31, -30, -28, -26, -25, -23,
-23, -22, -23, -23, -24, -24, -25, -26, -26, -27, -27, -27, -27, -28, -28,
-29, -29, -29, -29, -29, -28, -28, -27, -26, -25, -24, -24, -23, -22, -21,
-19, -17, -15, -13, -10, -8, -6, -4, -3, -2, -2, -2, -3, -3, -2, -2, 0,
2, 5, 8, 11, 14, 16, 18, 18, 19, 19, 19, 19, 19, 18, 17, 16, 15, 14, 13,
12, 12, 11, 11, 11, 11, 12, 13, 14, 15, 16, 17, 18, 18, 18, 17, 16, 14,
13, 12, 11, 11, 11, 11, 11, 10, 8, 6, 5, 3, 2, 1, 0, -2, -3, -5, -6, -8,
-11, -13, -16, -19, -21, -23, -24, -25, -26, -27, -27, -27, -27, -27, -28,
-29, -30, -32, -34, -35, -36, -36, -36, -36, -36, -36, -36, -36};
const int Table12[337] = {-40, -40, -39, -37, -35, -32, -30, -27, -25,
-24, -24, -24, -26, -27, -29, -31, -33, -34, -35, -36, -36, -35, -35, -34,
-33, -32, -31, -31, -30, -29, -29, -28, -27, -26, -25, -23, -22, -20, -18,
-16, -14, -12, -10, -8, -7, -5, -3, -2, 0, 1, 3, 4, 5, 5, 6, 6, 7, 8, 9,
10, 12, 14, 16, 19, 21, 23, 24, 25, 25, 24, 23, 21, 19, 17, 15, 13, 12,
11, 11, 11, 12, 13, 14, 15, 15, 15, 15, 16, 16, 17, 18, 19, 19, 19, 19,
18, 16, 14, 12, 10, 8, 6, 4, 3, 2, 2, 1, 0, -1, -3, -4, -6, -7, -9, -11,
-14, -17, -19, -22, -24, -25, -26, -27, -29, -30, -32, -34, -35, -36, -37,
-37, -37, -37, -36, -36, -35, -35, -35, -36, -37, -39, -40, -40, -40, -39,
-37, -35, -34, -32, -30, -28, -25, -23, -22, -21, -20, -19, -19, -19, -18,
-18, -18, -19, -19, -20, -20, -21, -21, -21, -21, -19, -17, -13, -9, -5,
0, 3, 6, 8, 8, 7, 6, 5, 4, 4, 5, 7, 8, 10, 12, 14, 16, 17, 18, 19, 19,
19, 19, 18, 18, 18, 19, 20, 22, 24, 27, 29, 32, 34, 36, 38, 40, 41, 42,
43, 44, 45, 45, 45, 44, 43, 41, 38, 36, 35, 34, 34, 35, 36, 38, 39, 41,
43, 44, 45, 45, 45, 44, 44, 44, 45, 45, 46, 48, 48, 49, 50, 50, 50, 49,
48, 45, 43, 40, 38, 37, 37, 37, 39, 40, 41, 41, 41, 40, 38, 35, 31, 27,
23, 20, 17, 15, 14, 13, 12, 10, 9, 7, 5, 3, 1, -1, -2, -3, -4, -5, -5,
-4, -2, -1, 1, 2, 3, 3, 4, 3, 2, 0, -2, -6, -9, -12, -15, -18, -22, -25,
-28, -30, -32, -34, -35, -35, -35, -34, -33, -31, -29, -27, -25, -24, -23,
-23, -24, -26, -28, -31, -33, -36, -37, -39, -39, -39, -39, -39, -39, -39,
-39};
const int Table11[357] = {-37, -36, -35, -34, -33, -31, -30, -28, -27,
-27, -26, -26, -27, -27, -27, -28, -28, -29, -29, -29, -29, -29, -30, -30,
-30, -31, -31, -31, -31, -30, -30, -30, -29, -28, -28, -27, -26, -25, -24,
-24, -23, -22, -20, -19, -17, -15, -13, -10, -8, -6, -5, -3, -2, -2, -2,
-2, -2, -2, -1, -1, 1, 2, 5, 7, 10, 13, 15, 17, 18, 19, 20, 20, 20, 20,
21, 20, 20, 19, 18, 17, 16, 16, 15, 15, 14, 14, 13, 13, 13, 14, 15, 16,
16, 17, 18, 19, 20, 20, 20, 20, 19, 18, 16, 15, 14, 14, 13, 13, 13, 13,
13, 12, 10, 9, 7, 6, 5, 4, 3, 2, 0, -1, -2, -4, -5, -7, -10, -12, -14,
-17, -19, -20, -21, -22, -23, -24, -24, -25, -25, -25, -25, -25, -26, -27,
-28, -30, -31, -33, -33, -33, -33, -33, -32, -32, -32, -32, -31, -31, -30,
-29, -27, -25, -23, -20, -18, -16, -14, -14, -13, -14, -15, -17, -19, -20,
-22, -23, -23, -22, -21, -19, -17, -14, -11, -9, -7, -5, -3, -3, -2, -2,
-2, -2, -1, 0, 1, 2, 4, 6, 8, 9, 10, 11, 12, 12, 12, 13, 13, 14, 14, 15,
15, 15, 15, 15, 15, 15, 16, 17, 18, 20, 23, 25, 28, 31, 33, 35, 37, 39,
39, 39, 39, 38, 37, 36, 34, 33, 33, 33, 33, 34, 34, 35, 36, 36, 36, 35,
34, 33, 32, 32, 33, 34, 35, 38, 40, 42, 43, 44, 45, 45, 44, 42, 41, 39,
38, 37, 36, 36, 36, 36, 37, 37, 38, 38, 39, 38, 38, 37, 36, 35, 33, 31,
30, 28, 27, 25, 24, 23, 21, 19, 18, 16, 13, 11, 9, 6, 4, 1, -1, -2, -4,
-4, -4, -4, -4, -4, -3, -2, -2, -1, 0, 1, 1, 1, 1, 1, 0, -1, -3, -5, -7,
-10, -13, -17, -20, -23, -26, -28, -30, -31, -31, -31, -30, -29, -27, -26,
-25, -24, -23, -24, -25, -26, -27, -28, -30, -31, -32, -33, -34, -34, -34,
-34, -34, -34, -34, -34, -34};
const int Table10[378] = {-45, -45, -44, -42, -40, -38, -36, -35, -35,
-34, -34, -33, -32, -32, -32, -32, -32, -31, -30, -29, -28, -27, -26, -26,
-25, -24, -24, -23, -21, -20, -20, -19, -19, -20, -20, -21, -21, -21, -20,
-20, -19, -17, -16, -15, -14, -14, -15, -15, -16, -17, -18, -19, -20, -20,
-20, -20, -20, -20, -20, -21, -22, -22, -22, -21, -21, -20, -19, -18, -17,
-17, -17, -18, -20, -21, -22, -23, -24, -24, -25, -25, -25, -26, -27, -28,
-28, -28, -28, -28, -27, -26, -24, -23, -21, -20, -19, -19, -19, -19, -19,
-19, -18, -17, -16, -15, -15, -14, -14, -14, -15, -15, -16, -16, -16, -15,
-14, -14, -14, -13, -13, -12, -11, -10, -8, -5, -2, 1, 4, 7, 10, 12, 14,
15, 15, 15, 15, 16, 16, 17, 17, 17, 17, 17, 17, 18, 19, 20, 21, 22, 22,
23, 23, 23, 24, 24, 23, 23, 24, 24, 25, 25, 26, 26, 26, 25, 24, 23, 23,
22, 22, 22, 22, 22, 22, 22, 23, 24, 25, 26, 27, 28, 28, 29, 29, 29, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 27, 26, 24, 23, 23, 23, 23, 23,
23, 24, 25, 26, 27, 28, 27, 27, 26, 25, 25, 24, 25, 26, 27, 28, 28, 29,
29, 29, 29, 30, 30, 31, 31, 31, 31, 30, 30, 30, 30, 29, 29, 29, 30, 30,
31, 32, 32, 32, 33, 33, 33, 34, 34, 34, 34, 35, 35, 36, 36, 37, 37, 36,
35, 34, 33, 32, 32, 31, 29, 28, 26, 24, 22, 19, 17, 16, 15, 15, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 14, 13, 13, 12,
11, 9, 8, 8, 7, 6, 6, 5, 5, 4, 2, 0, -2, -3, -4, -5, -5, -6, -6, -5, -4,
-3, -1, -1, -1, -1, -1, -1, -2, -3, -5, -6, -7, -8, -9, -10, -10, -10,
-10, -9, -10, -10, -11, -12, -13, -13, -13, -14, -14, -15, -16, -18, -19,
-21, -23, -25, -27, -28, -29, -29, -30, -30, -30, -29, -29, -28, -28, -28,
-29, -29, -30, -31, -33, -34, -36, -38, -39, -41, -42, -43, -44, -44, -44,
-44, -44};
const int Table9[401] = {-45, -44, -43, -41, -40, -37, -36, -35, -35, -36,
-38, -39, -40, -41, -41, -42, -42, -41, -40, -39, -38, -37, -36, -35, -35,
-34, -33, -32, -31, -29, -27, -24, -22, -20, -19, -18, -19, -19, -20, -20,
-20, -20, -20, -19, -19, -19, -18, -18, -17, -16, -16, -15, -15, -15, -15,
-15, -15, -15, -15, -14, -14, -14, -13, -13, -13, -12, -12, -12, -13, -14,
-14, -15, -16, -16, -17, -19, -20, -21, -22, -22, -22, -22, -21, -20, -19,
-18, -17, -16, -15, -14, -14, -14, -14, -16, -17, -18, -20, -21, -23, -24,
-25, -26, -27, -27, -28, -27, -27, -25, -24, -23, -21, -19, -17, -14, -12,
-11, -10, -9, -9, -8, -8, -8, -8, -9, -9, -10, -11, -11, -12, -12, -13,
-13, -13, -12, -11, -9, -7, -6, -4, -2, 0, 2, 4, 6, 8, 11, 13, 15, 16,
17, 18, 19, 21, 22, 23, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 24,
24, 24, 25, 26, 27, 28, 29, 29, 29, 30, 30, 30, 31, 31, 30, 30, 29, 28,
27, 26, 25, 24, 23, 23, 24, 25, 27, 28, 30, 31, 31, 31, 31, 30, 30, 29,
29, 29, 29, 30, 30, 30, 30, 30, 30, 29, 29, 28, 28, 27, 27, 28, 28, 28,
28, 28, 27, 26, 25, 24, 24, 24, 24, 25, 25, 26, 27, 27, 26, 25, 23, 22,
22, 22, 22, 24, 25, 27, 29, 31, 32, 32, 32, 31, 29, 28, 28, 28, 28, 28,
29, 29, 29, 29, 29, 29, 28, 27, 27, 27, 27, 27, 28, 28, 29, 30, 31, 31,
32, 32, 33, 33, 34, 34, 34, 33, 32, 30, 29, 27, 25, 23, 21, 19, 18, 17,
17, 16, 16, 15, 14, 13, 12, 11, 10, 9, 8, 8, 7, 6, 6, 6, 6, 6, 7, 7, 7,
7, 6, 5, 5, 5, 5, 5, 5, 5, 4, 4, 3, 3, 2, 1, 0, -1, -2, -3, -4, -6, -6,
-7, -8, -9, -10, -10, -11, -12, -13, -13, -14, -14, -14, -13, -12, -10,
-9, -7, -6, -6, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16, -18,
-19, -21, -23, -25, -26, -27, -28, -28, -27, -26, -24, -23, -22, -22, -22,
-24, -27, -30, -32, -34, -36, -36, -37, -37, -37, -37, -37, -37, -37, -38,
-39, -39, -40, -39};
const int Table8[425] = {-45, -45, -45, -44, -43, -42, -40, -38, -37, -35,
-34, -33, -32, -32, -33, -33, -34, -35, -36, -36, -37, -37, -37, -36, -36,
-34, -33, -31, -29, -27, -25, -23, -21, -19, -17, -15, -14, -14, -14, -14,
-14, -14, -14, -13, -13, -13, -13, -14, -15, -15, -16, -16, -16, -15, -14,
-14, -13, -12, -11, -10, -9, -8, -7, -7, -7, -8, -8, -9, -9, -10, -10,
-11, -11, -12, -12, -12, -12, -12, -13, -13, -14, -15, -16, -17, -17, -18,
-18, -18, -17, -17, -16, -15, -14, -13, -11, -10, -9, -9, -9, -10, -12,
-13, -15, -16, -18, -19, -20, -21, -22, -22, -23, -23, -23, -23, -22, -20,
-18, -16, -13, -11, -8, -6, -4, -2, -1, 0, 0, 0, -1, -2, -3, -3, -4, -5,
-6, -6, -6, -7, -7, -7, -7, -6, -5, -4, -3, -1, 1, 4, 7, 10, 13, 16, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 27, 27, 28, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 30, 30, 30, 31, 31, 31, 31, 32, 32, 31, 31, 31, 31, 30,
30, 31, 31, 32, 32, 32, 32, 31, 29, 28, 27, 26, 25, 25, 26, 27, 28, 29,
30, 31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 31, 30, 29, 28, 28, 27,
26, 26, 25, 25, 25, 25, 26, 27, 28, 29, 29, 29, 28, 26, 25, 23, 22, 21,
21, 21, 22, 23, 23, 23, 23, 23, 22, 22, 22, 22, 22, 22, 23, 24, 25, 26,
27, 28, 28, 28, 28, 28, 27, 27, 27, 27, 27, 27, 26, 26, 25, 24, 23, 22,
21, 21, 21, 21, 22, 23, 25, 27, 28, 29, 29, 29, 28, 28, 27, 26, 26, 25,
25, 26, 26, 26, 25, 24, 22, 20, 18, 16, 14, 12, 10, 8, 7, 6, 6, 6, 5, 5,
5, 5, 5, 5, 5, 5, 4, 4, 3, 2, 1, 0, -1, -2, -2, -2, -2, -2, -1, -1, -1,
-1, -1, -1, -1, -2, -2, -2, -3, -3, -4, -4, -5, -6, -7, -8, -10, -11, -13,
-14, -15, -15, -16, -17, -17, -18, -18, -19, -19, -19, -19, -18, -17, -17,
-16, -14, -13, -12, -11, -11, -11, -11, -12, -13, -14, -15, -16, -17, -18,
-20, -22, -24, -26, -27, -29, -30, -31, -32, -32, -33, -33, -33, -32, -30,
-29, -28, -27, -28, -28, -29, -30, -31, -33, -34, -35, -36, -37, -38, -40,
-41, -42, -44, -44, -45, -45, -45, -43, -42};
const int Table7[450] = {-45, -44, -43, -41, -39, -37, -34, -32, -29, -28,
-26, -26, -26, -27, -27, -28, -29, -29, -30, -30, -31, -31, -31, -32, -32,
-33, -33, -34, -35, -35, -35, -35, -35, -34, -33, -31, -29, -27, -24, -22,
-20, -18, -16, -15, -13, -11, -10, -8, -7, -7, -6, -6, -7, -8, -8, -9,
-9, -9, -9, -8, -7, -5, -3, -1, 0, 2, 3, 3, 3, 2, 1, -1, -2, -3, -4, -5,
-5, -4, -4, -3, -2, -1, 0, 1, 2, 3, 4, 4, 5, 6, 7, 9, 11, 13, 15, 17, 19,
21, 23, 25, 28, 30, 33, 35, 37, 39, 40, 40, 39, 38, 36, 33, 30, 27, 24,
21, 18, 16, 13, 12, 10, 8, 7, 6, 4, 3, 1, 0, -2, -4, -6, -8, -10, -12,
-13, -15, -16, -17, -17, -17, -18, -17, -17, -17, -17, -17, -17, -17, -17,
-18, -18, -19, -20, -20, -21, -21, -21, -21, -20, -19, -18, -17, -16, -15,
-15, -15, -16, -16, -16, -17, -16, -16, -14, -13, -11, -9, -7, -4, -2,
0, 2, 3, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8, 9, 10, 10, 11, 11, 12, 13, 14, 15,
15, 15, 15, 15, 14, 14, 13, 13, 12, 12, 12, 12, 13, 14, 15, 17, 18, 19,
20, 20, 20, 19, 18, 16, 15, 13, 12, 12, 11, 11, 12, 12, 13, 14, 16, 17,
18, 20, 20, 21, 21, 20, 20, 19, 18, 17, 16, 15, 15, 14, 13, 13, 12, 11,
11, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 10, 10, 10, 9, 8, 7, 5, 3, 2, 0, -1, -2, -2, -3, -3, -3, -3, -2, -1,
1, 3, 6, 8, 11, 13, 15, 16, 17, 18, 19, 20, 22, 24, 25, 27, 28, 30, 31,
32, 32, 32, 32, 32, 31, 30, 29, 28, 28, 28, 29, 30, 31, 33, 34, 36, 38,
39, 40, 41, 41, 40, 39, 38, 36, 34, 32, 31, 29, 28, 26, 25, 24, 22, 21,
20, 20, 20, 20, 20, 21, 21, 22, 23, 23, 24, 24, 25, 25, 25, 25, 26, 25,
24, 23, 21, 19, 16, 12, 8, 4, -1, -6, -11, -15, -19, -22, -24, -26, -27,
-27, -27, -26, -25, -25, -24, -23, -22, -22, -21, -20, -19, -18, -17, -16,
-15, -15, -15, -15, -15, -16, -17, -18, -19, -20, -21, -22, -23, -23, -24,
-25, -26, -26, -27, -28, -29, -29, -29, -29, -28, -27, -26, -24, -23, -23,
-22, -22, -23, -24, -26, -29, -32, -34, -37, -39, -40, -41, -42, -42, -42,
-42, -42, -42, -42, -42, -42, -42};
const int Table6[477] = {-45, -45, -44, -44, -42, -41, -40, -38, -37, -35,
-34, -32, -31, -30, -28, -28, -27, -27, -27, -27, -28, -29, -31, -32, -33,
-34, -35, -36, -37, -37, -37, -36, -35, -33, -32, -30, -29, -27, -26, -25,
-24, -23, -22, -21, -19, -18, -16, -14, -12, -10, -8, -6, -4, -4, -3, -3,
-3, -4, -4, -5, -6, -6, -6, -6, -5, -5, -4, -3, -3, -2, -1, 0, 0, 1, 2,
3, 3, 3, 3, 3, 3, 3, 2, 2, 1, 0, 0, 0, 0, 1, 2, 4, 6, 9, 11, 14, 16, 18,
20, 22, 23, 23, 24, 25, 26, 27, 28, 30, 32, 33, 35, 36, 37, 38, 38, 38,
37, 36, 34, 32, 29, 27, 24, 21, 18, 15, 12, 10, 7, 5, 2, 0, -2, -3, -4,
-5, -6, -6, -6, -6, -6, -7, -7, -8, -9, -11, -12, -14, -15, -16, -17, -17,
-17, -17, -16, -15, -14, -13, -13, -12, -12, -13, -13, -13, -14, -14, -15,
-15, -15, -15, -15, -15, -14, -14, -13, -12, -11, -9, -8, -7, -6, -5, -4,
-3, -3, -2, -1, 0, 0, 1, 2, 4, 5, 6, 8, 10, 11, 13, 15, 16, 16, 17, 17,
17, 16, 16, 15, 14, 14, 13, 13, 14, 15, 16, 17, 18, 19, 20, 20, 20, 20,
19, 18, 18, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 19, 19, 20, 20, 20,
20, 20, 19, 19, 18, 18, 17, 16, 15, 15, 15, 15, 15, 15, 16, 17, 18, 19,
20, 21, 22, 22, 22, 21, 20, 18, 16, 15, 13, 11, 10, 9, 8, 8, 8, 8, 9, 10,
10, 11, 11, 11, 11, 11, 11, 11, 11, 10, 10, 10, 10, 9, 9, 8, 7, 6, 5, 5,
4, 3, 3, 2, 1, 0, -1, -2, -2, -2, -1, 0, 0, 1, 2, 3, 4, 6, 8, 10, 12, 14,
17, 19, 21, 23, 25, 26, 26, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28,
28, 29, 29, 29, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 30, 31,
31, 32, 32, 32, 31, 30, 28, 26, 24, 21, 18, 16, 14, 13, 12, 12, 12, 12,
12, 12, 13, 14, 15, 15, 16, 17, 17, 17, 17, 17, 16, 15, 13, 12, 11, 9,
7, 5, 3, 0, -3, -6, -9, -13, -16, -20, -23, -26, -29, -31, -33, -34, -35,
-36, -35, -35, -33, -32, -30, -28, -26, -24, -22, -20, -18, -17, -16, -16,
-17, -18, -19, -19, -20, -21, -22, -22, -23, -23, -23, -24, -25, -26, -26,
-27, -28, -29, -29, -30, -31, -31, -31, -31, -31, -30, -29, -29, -28, -27,
-27, -27, -28, -28, -28, -29, -29, -30, -31, -32, -33, -35, -36, -37, -39,
-40, -41, -42, -43, -43, -43, -43, -43, -43};
const int Table5[505] = {-45, -45, -44, -44, -42, -41, -40, -38, -37, -36,
-34, -33, -31, -30, -29, -27, -26, -26, -25, -25, -26, -26, -27, -28, -29,
-30, -32, -33, -34, -34, -35, -35, -34, -33, -32, -31, -29, -28, -26, -24,
-23, -22, -21, -21, -20, -18, -17, -15, -14, -12, -10, -8, -6, -4, -2,
-1, 0, 1, 1, 1, 1, 0, 0, -1, -1, -2, -2, -1, -1, 0, 1, 1, 2, 3, 3, 4, 5,
6, 6, 7, 8, 8, 8, 8, 8, 7, 7, 7, 6, 6, 5, 5, 5, 5, 5, 6, 8, 10, 12, 14,
17, 19, 21, 23, 25, 26, 27, 28, 28, 29, 30, 31, 32, 34, 35, 37, 38, 39,
41, 42, 42, 42, 42, 41, 40, 38, 36, 34, 31, 29, 26, 23, 20, 17, 15, 12,
10, 7, 5, 3, 1, 0, -1, -2, -3, -3, -3, -4, -4, -4, -5, -6, -7, -8, -9,
-11, -12, -13, -14, -15, -15, -15, -15, -14, -14, -13, -12, -11, -11, -11,
-11, -11, -12, -12, -13, -13, -13, -14, -14, -14, -14, -14, -13, -13, -12,
-11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -2, -1, 0, 0, 1, 2, 3, 4, 5,
6, 8, 9, 11, 12, 14, 15, 16, 17, 17, 17, 17, 17, 16, 16, 15, 14, 14, 14,
14, 15, 16, 17, 18, 19, 20, 20, 20, 20, 20, 19, 19, 18, 18, 17, 17, 17,
17, 17, 17, 17, 18, 18, 18, 19, 19, 19, 20, 20, 20, 20, 19, 19, 18, 18,
17, 16, 15, 15, 14, 14, 14, 14, 14, 15, 16, 17, 18, 19, 20, 21, 21, 21,
21, 20, 18, 17, 15, 13, 12, 10, 9, 8, 7, 7, 7, 7, 7, 8, 8, 9, 9, 9, 10,
10, 9, 9, 9, 9, 9, 8, 8, 8, 8, 7, 6, 6, 5, 4, 3, 3, 2, 1, 1, 0, -1, -2,
-3, -4, -4, -4, -4, -3, -2, -2, -1, 0, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18,
20, 22, 23, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26,
26, 26, 26, 26, 26, 25, 25, 25, 25, 25, 25, 25, 26, 26, 27, 27, 28, 28,
29, 29, 29, 29, 28, 26, 24, 22, 20, 17, 15, 13, 11, 10, 9, 8, 8, 8, 9,
9, 9, 10, 11, 11, 12, 13, 13, 14, 14, 13, 13, 12, 11, 10, 9, 7, 6, 4, 3,
0, -2, -5, -8, -11, -14, -17, -20, -23, -26, -29, -32, -34, -36, -38, -39,
-39, -39, -39, -38, -36, -35, -33, -31, -29, -27, -25, -23, -21, -20, -20,
-20, -20, -20, -21, -22, -23, -24, -24, -25, -25, -26, -26, -26, -27, -27,
-28, -29, -30, -30, -31, -31, -32, -33, -33, -33, -33, -33, -32, -32, -31,
-30, -29, -29, -29, -29, -29, -29, -29, -30, -30, -31, -32, -32, -33, -34,
-36, -37, -38, -39, -40, -41, -42, -43, -43, -43, -43, -43, -43};
const int Table4[535] = {-45, -45, -44, -44, -43, -42, -42, -41, -41, -41,
-40, -40, -40, -40, -39, -38, -37, -36, -35, -34, -33, -32, -32, -31, -31,
-31, -32, -33, -34, -35, -36, -36, -36, -36, -36, -35, -35, -34, -33, -32,
-31, -29, -28, -27, -25, -24, -22, -21, -20, -19, -18, -17, -17, -15, -14,
-13, -11, -9, -7, -6, -4, -2, -1, 0, 1, 2, 2, 3, 2, 2, 2, 1, 1, 1, 1, 1,
1, 1, 2, 2, 3, 4, 4, 5, 5, 5, 5, 5, 4, 3, 2, 1, 0, -1, -1, -1, -1, 0, 0,
1, 2, 3, 4, 5, 5, 5, 6, 6, 6, 6, 6, 7, 8, 8, 9, 10, 10, 11, 12, 13, 13,
14, 15, 16, 17, 18, 18, 18, 18, 17, 17, 16, 15, 13, 12, 11, 10, 9, 8, 7,
6, 6, 5, 4, 3, 3, 2, 2, 2, 3, 4, 6, 7, 9, 11, 12, 14, 15, 15, 16, 16, 17,
17, 16, 16, 16, 15, 14, 13, 12, 11, 10, 9, 9, 10, 10, 11, 11, 11, 10, 9,
7, 4, 2, -1, -4, -6, -9, -10, -12, -13, -14, -15, -16, -16, -17, -17, -17,
-17, -18, -18, -19, -19, -19, -19, -19, -19, -18, -17, -16, -15, -13, -11,
-10, -8, -7, -6, -5, -5, -5, -5, -5, -5, -6, -6, -6, -6, -5, -5, -4, -2,
-1, 0, 1, 3, 4, 5, 6, 7, 8, 9, 9, 10, 11, 11, 11, 11, 11, 11, 12, 12, 12,
13, 14, 15, 15, 15, 15, 15, 15, 14, 14, 14, 15, 15, 16, 16, 17, 17, 17,
16, 16, 16, 15, 14, 14, 13, 12, 12, 11, 10, 8, 7, 5, 4, 3, 2, 1, 1, 1,
2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 1, 0, -1, -1, -2, -2, -1, -1, -1, 0, 0,
1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 16, 18, 19, 21, 22, 24, 25, 27, 28, 29,
30, 30, 29, 28, 27, 25, 23, 22, 20, 19, 19, 18, 18, 18, 18, 18, 18, 19,
19, 19, 20, 20, 20, 21, 21, 21, 21, 21, 20, 20, 20, 20, 20, 20, 19, 19,
18, 17, 16, 15, 15, 14, 14, 14, 13, 13, 13, 13, 13, 14, 14, 15, 16, 16,
16, 16, 16, 15, 14, 13, 12, 11, 11, 10, 10, 9, 9, 9, 9, 9, 9, 10, 10, 11,
11, 12, 12, 12, 12, 12, 12, 12, 11, 11, 11, 11, 11, 12, 12, 13, 14, 15,
15, 16, 16, 17, 17, 17, 17, 17, 16, 16, 15, 14, 14, 14, 14, 14, 13, 13,
12, 11, 10, 8, 7, 5, 3, 1, 0, -2, -3, -4, -5, -5, -6, -7, -8, -8, -8, -8,
-8, -7, -6, -6, -5, -4, -3, -2, -2, -1, 0, 0, 1, 1, 1, 0, -1, -2, -4, -5,
-7, -9, -10, -12, -13, -14, -15, -16, -16, -16, -16, -15, -14, -14, -13,
-12, -12, -12, -12, -13, -13, -14, -15, -16, -17, -18, -20, -21, -23, -25,
-26, -27, -28, -29, -29, -29, -30, -30, -31, -32, -34, -35, -37, -39, -40,
-42, -43, -44, -45, -45, -45, -45, -45};
const int Table3[567] = {-44, -44, -44, -44, -44, -43, -43, -43, -42, -42,
-41, -41, -40, -40, -39, -39, -38, -37, -37, -36, -35, -35, -34, -34, -33,
-33, -32, -31, -31, -30, -29, -28, -27, -27, -27, -27, -28, -29, -30, -31,
-32, -32, -32, -31, -30, -29, -28, -26, -24, -23, -21, -19, -17, -15, -13,
-12, -10, -9, -8, -7, -7, -6, -6, -5, -5, -4, -2, -1, 1, 3, 5, 7, 8, 9,
10, 10, 10, 10, 9, 8, 7, 7, 6, 6, 6, 6, 7, 7, 8, 8, 9, 9, 9, 9, 9, 8, 8,
7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9, 10, 10, 10, 10, 10,
10, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 20, 20, 20, 20, 19, 19, 19,
19, 18, 18, 18, 17, 17, 16, 16, 15, 15, 14, 13, 13, 12, 11, 10, 10, 8,
7, 6, 5, 4, 3, 3, 3, 3, 4, 5, 7, 10, 12, 15, 17, 19, 20, 20, 20, 20, 19,
18, 17, 16, 15, 14, 14, 13, 12, 11, 11, 10, 9, 8, 7, 6, 5, 4, 4, 3, 1,
0, -2, -4, -5, -7, -9, -10, -12, -13, -14, -15, -16, -17, -18, -18, -19,
-20, -21, -21, -22, -22, -22, -21, -20, -19, -18, -17, -16, -14, -13, -12,
-10, -10, -9, -8, -8, -7, -6, -6, -5, -4, -4, -4, -3, -3, -3, -3, -3, -3,
-2, -2, -1, 0, 1, 3, 4, 5, 5, 6, 7, 8, 8, 9, 10, 10, 11, 12, 13, 14, 14,
15, 15, 15, 15, 15, 15, 15, 15, 15, 14, 14, 14, 13, 13, 13, 13, 13, 13,
13, 14, 15, 16, 16, 17, 17, 16, 16, 15, 14, 12, 11, 10, 8, 7, 6, 5, 4,
3, 3, 2, 2, 2, 2, 2, 2, 2, 1, 1, 0, 0, 0, -1, -1, -1, -1, -1, -1, 0, 0,
1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 8, 10, 12, 15, 18, 21,
24, 26, 28, 29, 29, 30, 30, 29, 28, 28, 27, 26, 25, 24, 23, 22, 21, 20,
19, 19, 18, 18, 18, 18, 18, 18, 18, 18, 17, 17, 16, 16, 16, 16, 16, 16,
17, 18, 19, 20, 21, 21, 21, 21, 20, 19, 18, 16, 15, 13, 12, 11, 10, 10,
9, 9, 9, 9, 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 14, 13, 12, 11, 10,
8, 7, 7, 6, 6, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9,
9, 10, 10, 11, 12, 12, 13, 13, 14, 14, 14, 13, 13, 12, 11, 11, 10, 10,
10, 10, 10, 10, 11, 10, 10, 10, 9, 9, 8, 6, 5, 3, 1, -1, -3, -5, -7, -8,
-9, -10, -10, -10, -11, -11, -10, -10, -10, -10, -10, -10, -9, -9, -9,
-9, -9, -9, -9, -9, -8, -7, -6, -6, -5, -4, -3, -3, -4, -4, -5, -7, -8,
-10, -11, -13, -15, -17, -19, -20, -21, -23, -23, -23, -23, -22, -21, -20,
-18, -17, -16, -15, -15, -15, -16, -17, -19, -21, -23, -25, -26, -27, -28,
-29, -30, -30, -31, -32, -33, -34, -35, -36, -37, -38, -38, -39, -40, -41,
-42, -43, -44, -44, -45, -45};
const int Table2[601] = {-45, -44, -44, -43, -42, -41, -39, -38, -36, -35,
-33, -32, -31, -29, -28, -27, -26, -25, -24, -23, -22, -21, -21, -20, -20,
-20, -20, -19, -19, -18, -18, -18, -17, -17, -18, -18, -18, -18, -18, -18,
-18, -17, -16, -14, -12, -11, -9, -7, -5, -4, -2, -1, 0, 1, 2, 3, 3, 4,
5, 5, 6, 7, 7, 8, 9, 10, 11, 12, 14, 15, 17, 19, 21, 22, 23, 23, 23, 22,
22, 20, 19, 17, 16, 14, 13, 12, 11, 11, 11, 11, 11, 11, 12, 13, 13, 14,
14, 14, 14, 14, 14, 14, 13, 13, 13, 12, 12, 12, 12, 12, 12, 12, 12, 12,
13, 14, 14, 15, 16, 17, 18, 18, 18, 18, 18, 18, 17, 17, 17, 17, 17, 18,
19, 19, 20, 21, 21, 22, 22, 22, 21, 21, 20, 20, 19, 18, 17, 17, 16, 15,
14, 13, 12, 12, 11, 10, 10, 9, 9, 8, 8, 8, 7, 6, 6, 5, 4, 4, 3, 3, 3, 4,
4, 5, 7, 8, 9, 11, 12, 13, 14, 14, 14, 14, 14, 13, 12, 11, 9, 8, 7, 6,
5, 4, 3, 2, 1, 0, -1, -2, -4, -6, -8, -9, -11, -13, -16, -18, -20, -22,
-23, -25, -26, -27, -27, -27, -27, -27, -26, -24, -23, -21, -20, -19, -18,
-18, -17, -17, -17, -17, -16, -16, -15, -15, -14, -14, -13, -13, -13, -12,
-11, -10, -9, -7, -5, -3, -1, 0, 2, 3, 4, 5, 6, 6, 6, 6, 7, 7, 7, 7, 7,
7, 7, 7, 7, 8, 8, 8, 9, 10, 10, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15,
16, 16, 17, 17, 17, 17, 17, 17, 16, 16, 15, 14, 14, 13, 11, 10, 9, 8, 8,
8, 8, 8, 8, 9, 9, 9, 10, 10, 9, 9, 8, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1,
0, 0, 0, -1, -1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 6, 7,
9, 10, 12, 13, 15, 16, 17, 19, 20, 20, 21, 22, 22, 22, 22, 22, 22, 22,
22, 22, 23, 23, 24, 25, 26, 27, 28, 30, 30, 31, 31, 31, 30, 29, 28, 26,
25, 23, 21, 20, 19, 17, 17, 16, 16, 16, 16, 16, 15, 15, 14, 13, 12, 12,
11, 10, 10, 9, 9, 10, 10, 10, 11, 12, 13, 13, 14, 15, 15, 16, 16, 16, 16,
15, 14, 13, 11, 10, 8, 6, 5, 3, 2, 2, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5,
5, 6, 6, 6, 6, 6, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 7, 8, 8, 9, 9, 9, 9, 8, 7, 5, 4, 3,
1, 0, -1, -2, -3, -4, -5, -6, -7, -8, -8, -9, -9, -10, -10, -10, -11, -11,
-12, -12, -13, -14, -15, -16, -17, -17, -18, -18, -19, -19, -19, -18, -18,
-17, -16, -16, -15, -14, -14, -13, -13, -13, -13, -14, -13, -13, -13, -13,
-12, -12, -12, -12, -12, -13, -14, -15, -16, -17, -18, -19, -20, -22, -23,
-24, -26, -27, -28, -29, -30, -31, -31, -31, -30, -30, -29, -29, -29, -28,
-28, -28, -29, -29, -30, -30, -31, -32, -32, -33, -34, -34, -34, -35, -35,
-36, -36, -36, -37, -38, -38, -39, -40, -41, -42, -42, -43, -44, -44, -45,
-45, -45, -45, -45};
const int Table1[636] = {-45, -45, -45, -44, -44, -43, -42, -41, -40, -39,
-38, -37, -36, -35, -34, -34, -33, -33, -32, -31, -30, -29, -28, -27, -26,
-26, -25, -24, -23, -23, -22, -22, -22, -22, -23, -23, -23, -23, -23, -24,
-23, -23, -23, -22, -21, -20, -19, -17, -15, -14, -12, -10, -8, -6, -5,
-3, -2, -1, -1, -1, -2, -2, -2, -3, -4, -5, -5, -5, -6, -6, -5, -5, -4,
-4, -3, -2, -1, 0, 1, 2, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 2, 2, 1, 1, 0, 0,
-1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 3, 3, 4,
4, 4, 5, 5, 5, 5, 5, 6, 6, 7, 7, 8, 8, 10, 10, 12, 13, 15, 16, 18, 20,
22, 23, 25, 27, 28, 30, 31, 31, 32, 32, 33, 32, 32, 32, 31, 31, 30, 30,
30, 29, 29, 29, 29, 30, 30, 30, 31, 31, 31, 31, 32, 31, 31, 31, 30, 30,
29, 28, 27, 26, 25, 25, 24, 24, 23, 23, 23, 23, 24, 24, 24, 25, 25, 25,
25, 25, 25, 25, 24, 23, 23, 23, 22, 22, 22, 22, 22, 21, 22, 22, 22, 21,
21, 21, 20, 19, 19, 17, 16, 15, 14, 12, 10, 9, 7, 6, 5, 4, 3, 2, 1, 0,
0, -1, -1, -1, 0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 0, 0, 0, -1, -1, -1, -1, -1, 0, 0,
0, 0, 0, 0, 0, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -11, -12,
-12, -12, -12, -11, -11, -11, -10, -10, -10, -9, -9, -9, -9, -10, -10,
-11, -12, -14, -15, -16, -18, -19, -20, -22, -23, -23, -24, -24, -24, -25,
-25, -25, -25, -25, -25, -26, -26, -27, -27, -28, -28, -29, -29, -30, -30,
-30, -30, -29, -29, -28, -27, -27, -26, -25, -24, -23, -23, -22, -22, -21,
-21, -21, -21, -20, -20, -20, -20, -20, -20, -20, -20, -20, -20, -20, -20,
-20, -20, -21, -21, -21, -22, -22, -23, -24, -24, -25, -26, -27, -28, -29,
-29, -29, -29, -29, -29, -29, -29, -28, -28, -28, -27, -27, -27, -26, -26,
-26, -26, -26, -26, -26, -26, -25, -24, -23, -22, -21, -20, -18, -17, -16,
-14, -13, -12, -12, -11, -11, -11, -10, -11, -11, -11, -11, -11, -11, -10,
-10, -9, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 3, 4, 4, 4, 5,
5, 6, 6, 7, 8, 8, 9, 10, 11, 12, 12, 13, 14, 14, 15, 15, 15, 15, 15, 15,
15, 15, 16, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 29, 30,
30, 30, 30, 30, 29, 29, 28, 28, 27, 26, 26, 25, 24, 24, 23, 23, 22, 22,
22, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 23, 23, 24, 25, 26, 26,
28, 28, 29, 30, 31, 31, 31, 31, 31, 31, 31, 30, 30, 30, 29, 29, 29, 30,
30, 30, 30, 31, 31, 32, 32, 33, 33, 34, 34, 34, 34, 35, 35, 35, 35, 35,
34, 34, 34, 33, 33, 32, 31, 30, 29, 29, 27, 26, 25, 24, 23, 21, 20, 18,
17, 15, 13, 11, 9, 7, 5, 4, 3, 1, 1, 0, -1, -2, -2, -3, -4, -4, -5, -6,
-7, -8, -10, -11, -12, -14, -15, -17, -19, -21, -22, -24, -26, -27, -29,
-30, -31, -32, -33, -34, -34, -35, -35, -36, -36, -37, -38, -38, -39, -40,
-41, -41, -42, -43, -44, -44, -45, -45, -45};
const int Table0[674] = {-45, -45, -45, -44, -43, -41, -40, -38, -36, -34,
-33, -31, -30, -29, -28, -28, -27, -27, -27, -27, -27, -27, -27, -27, -27,
-26, -25, -24, -24, -23, -22, -21, -19, -18, -16, -15, -14, -13, -12, -12,
-12, -12, -13, -13, -13, -13, -14, -14, -15, -15, -15, -15, -15, -14, -13,
-12, -10, -9, -8, -6, -4, -2, -1, 1, 3, 4, 5, 6, 6, 6, 5, 5, 5, 4, 3, 3,
2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 5, 5,
6, 6, 6, 6, 6, 6, 6, 5, 4, 4, 3, 2, 2, 1, 0, -1, -2, -2, -3, -3, -3, -3,
-2, -1, -1, 1, 2, 4, 5, 7, 9, 10, 11, 12, 13, 13, 13, 14, 13, 13, 13, 13,
12, 12, 11, 11, 12, 12, 14, 15, 17, 19, 21, 24, 26, 28, 30, 31, 33, 34,
34, 35, 35, 34, 33, 32, 31, 30, 29, 27, 26, 24, 23, 22, 21, 21, 21, 21,
22, 22, 23, 23, 24, 25, 25, 26, 26, 26, 25, 24, 23, 22, 21, 20, 19, 18,
17, 17, 17, 16, 16, 16, 17, 17, 17, 17, 18, 18, 18, 19, 19, 19, 19, 18,
18, 18, 17, 16, 15, 14, 13, 13, 12, 12, 12, 12, 12, 12, 11, 11, 10, 9,
8, 7, 6, 5, 3, 2, 0, -1, -2, -3, -4, -5, -5, -6, -6, -6, -6, -6, -6, -6,
-6, -6, -6, -5, -6, -6, -6, -6, -6, -6, -6, -5, -5, -4, -4, -3, -2, -1,
0, 1, 2, 3, 3, 3, 3, 3, 2, 2, 1, 0, 0, -2, -3, -5, -6, -7, -8, -9, -9,
-9, -9, -9, -8, -7, -7, -6, -6, -5, -5, -4, -4, -4, -4, -5, -5, -6, -6,
-7, -8, -9, -10, -12, -13, -15, -16, -16, -16, -16, -15, -15, -14, -13,
-13, -12, -12, -12, -11, -11, -11, -12, -12, -13, -14, -15, -15, -16, -17,
-18, -19, -19, -20, -21, -22, -22, -22, -22, -22, -23, -23, -24, -24, -25,
-26, -27, -27, -27, -27, -28, -28, -28, -27, -26, -25, -24, -23, -21, -20,
-19, -19, -18, -17, -16, -16, -15, -14, -14, -13, -13, -14, -14, -15, -15,
-15, -15, -15, -15, -14, -14, -14, -14, -13, -13, -13, -13, -13, -14, -15,
-15, -16, -18, -18, -19, -20, -21, -22, -23, -24, -24, -25, -24, -24, -23,
-22, -21, -20, -19, -18, -17, -17, -16, -16, -16, -16, -16, -17, -17, -18,
-18, -18, -17, -17, -16, -15, -14, -12, -11, -10, -9, -8, -6, -5, -4, -3,
-2, -1, -1, 0, 0, 0, 0, 0, -1, -1, -2, -2, -3, -2, -2, -2, -1, 0, 1, 2,
2, 3, 4, 5, 6, 7, 7, 8, 8, 8, 8, 8, 8, 8, 9, 9, 10, 11, 12, 13, 14, 15,
16, 17, 17, 18, 19, 19, 19, 19, 19, 18, 18, 18, 18, 18, 17, 17, 17, 17,
18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 31, 31, 31, 31, 30,
29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 20, 20, 19, 19, 19, 19, 19, 19,
18, 17, 17, 16, 16, 16, 16, 16, 16, 16, 17, 17, 18, 19, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 31, 31, 31, 31, 30, 29, 28, 27, 26, 25, 24,
24, 23, 23, 23, 23, 24, 24, 25, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27,
27, 26, 25, 24, 22, 21, 20, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10,
9, 7, 6, 4, 2, 1, -1, -3, -5, -7, -8, -9, -11, -12, -13, -13, -14, -14,
-14, -15, -14, -15, -15, -15, -15, -16, -16, -17, -18, -19, -21, -22, -24,
-26, -28, -30, -31, -33, -34, -35, -36, -37, -38, -39, -40, -40, -40, -40,
-40, -40, -40, -40, -40, -41, -41, -41, -42, -42, -42, -42, -42};
const int* Table[48] =
{Table0,Table1,Table2,Table3,Table4,Table5,Table6,Table7,Table8,Table
9,Table10,Table11,Table12,Table13,Table14,Table15,Table16,Table17,Tab
le18,Table19,Table20,Table21,Table22,Table23,Table24,Table25,Table26,
Table27,Table28,Table29,Table30,Table31,Table32,Table33,Table34,Table
35,Table36,Table37,Table38,Table39,Table40,Table41,Table42,Table43,Ta
ble44,Table45,Table46,Table47};
const int Size[48] =
{674,636,601,567,535,505,477,450,425,401,378,357,337,318,300,283,268,
253,238,225,212,200,189,179,169,159,150,142,134,126,119,113,106,100,9
5,89,84,80,75,71,67,63,60,56,53,50,47,45};
const int Decay[108] = {1024, 987, 951, 916, 883, 851, 820, 790, 761, 734,
707, 681, 657, 633, 610, 588, 566, 546, 526, 507, 488, 470, 453, 437, 421,
406, 391, 377, 363, 350, 337, 325, 313, 302, 291, 280, 270, 260, 251, 242,
233, 224, 216, 208, 201, 193, 186, 180, 173, 167, 161, 155, 149, 144, 139,
134, 129, 124, 120, 115, 111, 107, 103, 99, 96, 92, 89, 86, 83, 80, 77,
74, 71, 69, 66, 64, 61, 59, 57, 55, 53, 51, 49, 47, 46, 44, 42, 41, 39,
38, 37, 35, 34, 33, 32, 30, 29, 28, 27, 26, 25, 24, 23, 23, 22, 21, 20,
19};
#endif
```

Appendix E: revised.c

```
#include <p32xxxx.h>
#include <plib.h>
```

```
#include "sound.h"
#include "fft.h"

/*SYSCLK = 80MHz, PBCLK = 10MHz*/

struct note{
    int num; //num = {0,2,3,...47};
    int count; //index in TableNum
    int duration; //duration increases by 44100 every 1 second, up to
2.5*44100;
    int* TableN;
    int ifabort;    //if ifabort == 1, then duration increases but no sound
    int TableSize;
};

int FFTinput[FFTINPUTSIZE];
int FFToutput[FFTOUTPUTSIZE];


int timedomainCNT;  //0~FFTINPUTSIZE-1;
int spectrumCNT;       //0~FFTOTPUTSIZE-1;

struct note FIFO;

int readnum;
int checkinQ;

#pragma interrupt readNum_ISR ipl7 vector 26   // interrupt vector CN
#pragma interrupt PWM_ISR ipl6 vector 8
#pragma interrupt LED_ISR ipl4 vector 12
#pragma interrupt FFT_ISR ipl5 vector 16

void readNum_ISR();
void PWM_ISR();
void LED_ISR();
void FFT_ISR();
void init_CN();
void init_T2();
void init_T3();
void init_T4();
void init_LED();
void init_PWM();
void init_UART();
void init_FIFO();
void enqueue(int Num);
void dequeue();
void send(int Num);

int main(){
    readnum = 0;
    checkinQ = 0;
    timedomainCNT = 0;
    spectrumCNT = 0;
    INTCONbits.MVEC = 1;
    asm("di");
    init_CN();
    init_T2();
    init_T3();
    init_T4();
    init_LED();
    init_PWM();
    init_UART();
    init_FIFO();
    asm("ei");
    while(1);
    return 0;
}
void readNum_ISR(){
    int i;
    IEC1bits.CNIE = 0;
    if(PORTCbits.RC13 == 1)
    {
        checkinQ = 0;
        readnum =
PORTDbits.RD2+PORTDbits.RD3*2+PORTDbits.RD4*4+PORTDbits.RD5*8+PORTDbi
ts.RD6*16+PORTDbits.RD7*32;
        if((readnum == 1 && FIFO.num == 0)||(readnum == FIFO.num))
            {
                checkinQ = 1;
                if(FIFO.duration > 4000)
                {
                    FIFO.ifabort = 1;
                    if(readnum == 1)
                    {
                        enqueue(0);
                        send(0);    //send # of key to PC by UART;
                    }
                    else
                    {
                        enqueue(readnum);
                        send(readnum);
                    }
                }
            }
            else if (readnum == 1)
            {
                enqueue(0);
                send(0);
            }
            else
            {
                enqueue(readnum);
                send(readnum);
            }
        }
        IFS1bits.CNIF = 0;
        IEC1bits.CNIE = 1;
    }
    void PWM_ISR() {
        IEC0bits.T2IE = 0;
        int sum, A, D, DecayLevel;
        sum = 0;
        DecayLevel = 0;
        if(FIFO.num >= 0)
        {
            A = FIFO.TableN[FIFO.count];
            DecayLevel = FIFO.duration >> 10;
            D = Decay[DecayLevel];
            sum = ((A*D) >> 10);
            FIFO.count++;
            if(FIFO.count == FIFO.TableSize)
            {
                FIFO.count = 0;
            }
            FIFO.duration++;
            if(FIFO.duration == TIMEOUT)
            {
                dequeue();
            }
        }
        if(timedomainCNT < 1024)
        {
```

```
        FFTinput[timedomainCNT++] = sum;
    }
    OC2RS = PR2/2 + sum;
    IFS0bits.T2IF = 0;
    IEC0bits.T2IE = 1;
}
void LED_ISR(){
    IEC0bits.T3IE = 0;
    if(spectrumCNT < FFTOUTPUTSIZE)
    {
        if(FFToutput[spectrumCNT] <= A0)
        {
            PORTE = 1;
        }
        else if(FFToutput[spectrumCNT] > A0 && FFToutput[spectrumCNT] <=
A1)
        {
            PORTE = 3;
        }
        else if(FFToutput[spectrumCNT] > A1 && FFToutput[spectrumCNT] <=
A2)
        {
            PORTE = 7;
        }
        else if(FFToutput[spectrumCNT] > A2 && FFToutput[spectrumCNT] <=
A3)
        {
            PORTE = 15;
        }
        else if(FFToutput[spectrumCNT] > A3 && FFToutput[spectrumCNT] <=
A4)
        {
            PORTE = 31;
        }
        else if(FFToutput[spectrumCNT] > A4 && FFToutput[spectrumCNT] <=
A5)
        {
            PORTE = 63;
        }
        else if(FFToutput[spectrumCNT] > A5 && FFToutput[spectrumCNT] <=
A6)
        {
            PORTE = 127;
        }
        else if(FFToutput[spectrumCNT] > A6)
        {
            PORTE = 255;
        }
    }
    spectrumCNT++;
    IFS0bits.T3IF = 0;
    IEC0bits.T3IE = 1;
}
void FFT_ISR(){
    IEC0bits.T4IE = 0;
    int i;
    for (i = 0; i < FFTOUTPUTSIZE; i++)
        FFToutput[i] = lightPos[readnum][i];
    spectrumCNT = 0;    //Return spectrum pointer to start.
    timedomainCNT = 0;  //Return time domain record to start.
    IFS0bits.T4IF = 0;
    IEC0bits.T4IE = 1;
}
void init_CN(){
    TRISC = 0x2000; //CN1(RC13) is input;
    TRISD = 0xFC;   //RD2~7 are inputs;
    PORTC = 0;
    CNENbits.CNEN1 = 1;
    CNPUEbits.CNPUE1 = 1;
    IEC1bits.CNIE = 1;
    IFS1bits.CNIF = 0;
    IPC6bits.CNIP = 7;
    IPC6bits.CNIS = 3;
    CNCONbits.ON = 1;
}
void init_T2(){
    T2CON = 0;
    TMR2 = 0;
    PR2 = PWMPERIOD;
    T2CONbits.T32 = 0;
    IEC0bits.T2IE = 1;
    IFS0bits.T2IF = 0;
    IPC2bits.T2IP = 6;
    IPC2bits.T2IS = 3;
    T2CONbits.ON = 1;
}
void init_T3(){
    T3CON = 0;
    TMR3 = 0;
    PR3 = LEDPERIOD;
    T3CONbits.TCKPS = 6;    //1:64
    IEC0bits.T3IE = 1;
    IFS0bits.T3IF = 0;
    IPC3bits.T3IP = 4;
    IPC3bits.T3IS = 3;
    T3CONbits.ON = 1;
}
void init_T4(){
    T4CON = 0;
    TMR4 = 0;
    PR4 = FFTPERIOD;
    T4CONbits.T32 = 0;
    T4CONbits.TCKPS = 6;    //1:64
    IEC0bits.T4IE = 1;
    IFS0bits.T4IF = 0;
    IPC4bits.T4IP = 5;
    IPC4bits.T4IS = 3;
    T4CONbits.ON = 1;
}
void init_LED(){
    TRISE = 0xff00; //RE0-RE7 are output;
    PORTE = 0;
}
void init_PWM(){
        //OC2/RD1 is output;
    OC2CON = 0;
    OC2RS = 0;
    OC2R = 0;
    OC2CONbits.OCM = 6; //PWM, timer2;
    OC2CONbits.ON = 1;
}
void init_UART(){
    U1AMODEbits.UEN = 0;
    U1AMODEbits.BRGH = 1;
    U1ABRG = 259; //baud rate 9600.
    U1AMODEbits.PDSEL = 0;
    U1AMODEbits.STSEL = 0;
    U1ASTAbits.UTXEN = 1;
    U1AMODEbits.ON = 1;
}
void init_FIFO(){
    FIFO.num = -1;
    FIFO.count = 0;
    FIFO.duration = 0;
    FIFO.ifabort = 0;
}
void enqueue(int Num){
    FIFO.num = Num;
    FIFO.count = 0;
    FIFO.duration = 0;
    FIFO.TableN = (int*) Table[Num];
    FIFO.TableSize = Size[Num];
    FIFO.ifabort = 0;
}
void dequeue(){
    init_FIFO();
}
void send(int Num){
    U1ATXREG = Num;
}
```