# EE2004 Microcomputer System

# Mini-Project Title

|  |  |
| --- | --- |
| Team Member Name | ID: |
| TSOI Hing Chung Sunny | 55216557 |
| Tse Sin Man | 55231745 |

# Instructor: Leung Chi Sing Andrew

# Tutorial Session: TA4

# Tutorial Tutor: Chan Kwok Leung

# Semester B

# Date

24/4/2019

**Project Title: Digital Piano**

**Project Description**

It is to describe your project What it is supposed to do.

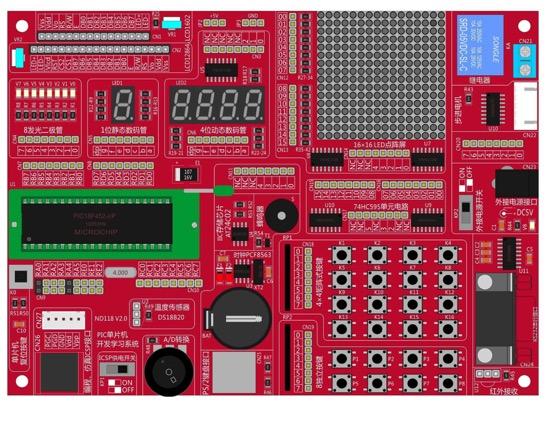
In this project, we have to design a digital piano with MPLAB and PIC18F4520 Microcontroller. The digital piano will produce 7 different sound (C, D, E, F…) and a melody (Little Star) by the buzzer when the 8 input buttons are pressed. Our digital piano system can also display the number of the button which is pressed by the user through 7 segment display. To generate different frequencies of the sound, we use time delay function and timer.

**System Architecture**

Draw the system architecture of your project and describe the functionality of each components

Hardware:

1. ND118-4520 development kit equipped with PIC18F4520 Microcontroller (fig. 1)



We used the following components:

* 8 input buttons (as input device)
* 1 7-seg display (as number output)
* Buzzer (as sound output)

1. Microchip’s PICkit3 In-Circuit Debugger/Programmer (fig. 2)

一張含有 室內 的圖片

自動產生的描述

It is used to program the PIC18F4520 microcontroller

1. USB cables

Software:

1. MPLAB IDE V8

Connection:

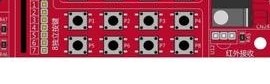
**I/O Description**

It is to define the desirable input and output

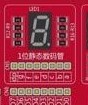
In this section, the I/O description will be discussed. In our project, ND118-4520 development kit equipped with PIC18F4520 Microcontroller is used. There are lots of components in the microcontroller, but in the project only few components are used beside the PIC18F4520. There are 8-button, 7-segment LED display and buzzer.

For the input, the 8-button is used for distinguishing what button and the system can then be processing the corresponding function (seen figure 3).

For the output, there are mainly two components used which are 7-segment LED(seen figure 4) and buzzer(seen figure 5). The 7-segment LED is aimed to display different number due to different buttons are pressed. The buzzer is the most important component in this project since it is responsible for playing different tone.

****

**fig 3**

****

**fig 4**

****

**fig 5**

**Program Description**

Description of your program structure with aid of flow chart if any

We discover that different sound can be played by modifying the frequency of the buzzer. Hence, we search the Internet and find 7 tonal scales with corresponding frequency. Table 1 illustrates the corresponding frequency of different sounds. We designed a delay function to generate a different frequency. We also designed a timer to generate a time delay. After that, the buzzer can play the sound according to the frequency created by the timer. After button 8 is pressed, the song “little star” will be played. Therefore, we created a subroutine “sound\_of\_song” which stores the corresponding values of different notes of “little star”. To avoid the speed of the song become too fast, we made another timer subroutine.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Button | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Pitch | C5 | D5 | E5 | F5 | G5 | A5 | B5 |
| Frequency(Hz) | 523.25 | 587.33 | 659.25 | 698.46 | 783.99 | 880.00 | 987.77 |

Table 1: the corresponding frequency of different sounds

Structure of our code:

First and foremost, the code will go to the Main to set and initialize the input and output ports. In the program, there are nine variables are used in subroutine which are DELAY\_H, DELAY\_L, input, led, con\_long, con\_loop, con\_tone, con\_song and con\_stop. DELAY\_H and DELAY\_L are used in the main program for detecting whether the button is pressed or not. input shows which button is pressed and will be used in the subroutine later on. led is used to control the display on 7-segment LED. The rest of the variables are used in the subroutine as loop control variable.

Then the program will go to the loop part, which will keep scanning key. In the looping part, it will keep on scanning key. If the button (1 to 6) is pressed, the corresponding 7-seg display is turned on by file register “led”. After that, the program will jump to the ‘sound’ routine. This routine uses a delay function to implement sound output. If the button 8 is pressed, the program will jump to the “sound\_of\_timer” routine. This routine uses a timer to implement sound output. If the button 7 is pressed, the program will jump to “sound\_of\_song” subroutine.

In the later paragraph, each of the subroutine will be explained their operating principles. There are total three function to generate sound or melody through buzzer, two delay functions and one supporting subroutine “find\_low”.

First of all, the “sound” subroutine is a function that would generate 6 different tone for corresponding buttons. There is a variable con\_long is used. con\_long is the loop control variable which not only allow the loop to process and also control how long will the tone last. According to pseudo code below, the value of con\_long is set to 40h. This value had been tested and the tone outputted can have the best with con\_long’ s value of 40H. After setting up the value for the loop control variable, bit 5 of PORTB is set to 1 to turn on buzzer and it will be set to 0 after the delay subroutine “delay\_buzzer” is call.”find\_low” is aimed to provide different delay time for each button. Thus it would check input before setting the value of con\_tone.

After processing the delay function twice, the con\_loop will be decreased by 1 and con\_loop will be checked whether it is 0 or not. If not, the loop is continuous. Else the loop is broke, the subroutine is finished and return to the main function.

Pseudo code of “sound”:

set value of con\_long to 40h

there turn on the buzzer (PORTB, 2)

call delay

toggle PORTB, 5 to create square wave

call delay

decrease con\_long (create a loop)

check if con\_long is zero or not if not keep loop, else return

Pseudo code of “delay\_buzzer”:

set value of con\_loop as 04h

call find\_low and set con\_tone

decreased con\_loop by 1

check whether con\_loop is 0 or not

if so end of loop and return, else continous

Pseudo code of “find\_low”:

check input for detecting which button is pressed

put the corresponding value in WREG and return

Second, the “sound\_of\_timer” is similar with the “sound”. These two subroutines are playing different sound by generating square wave through buzzer. However, “sound\_of\_timer” does not need to call another delay function. It used the timer to generating square wave. While the timer is still counting, the buzzer is being toggle and a square wave will be generated.

pseudo code of “sound\_of\_timer”:

set 1 to PORTB,5 (turn on buzzer)

set up the timer

initialize the timer

setting up the value for TMR0H and TMR0L

toggle PORTB, 5 (generating square wave)

start turn on the timer

time\_back

check whether the timer is finish

if so, turn on timer and return

else continuous and back the time\_back

Finally, the “sound\_of\_song” is aimed to play a song through buzzer. In the beginning, our group wanted to use array to implement a song. However, with the limitation of time and programming skills. We do not implement song with array instead the tone is set inside “sound\_of\_song”. Then, “sound” is called to play the tone. In order to create a beat, there is another delay subroutine “delay\_song” is used. This delay function would generate a fixed value of delay which allow the song is easier to recognize with a timer interrupt.

Pseudo code of “sound\_of\_song”:

set value of input with corresponding value

call sound

call delay\_song

until the end of the song

Pseudo code of “delay\_song”:

set con\_stop as 40H

set con\_song as ffh <-(stop)

set up timer interrupt

initialize the timer <-( again)

turn on timer

check whether is stopped or not

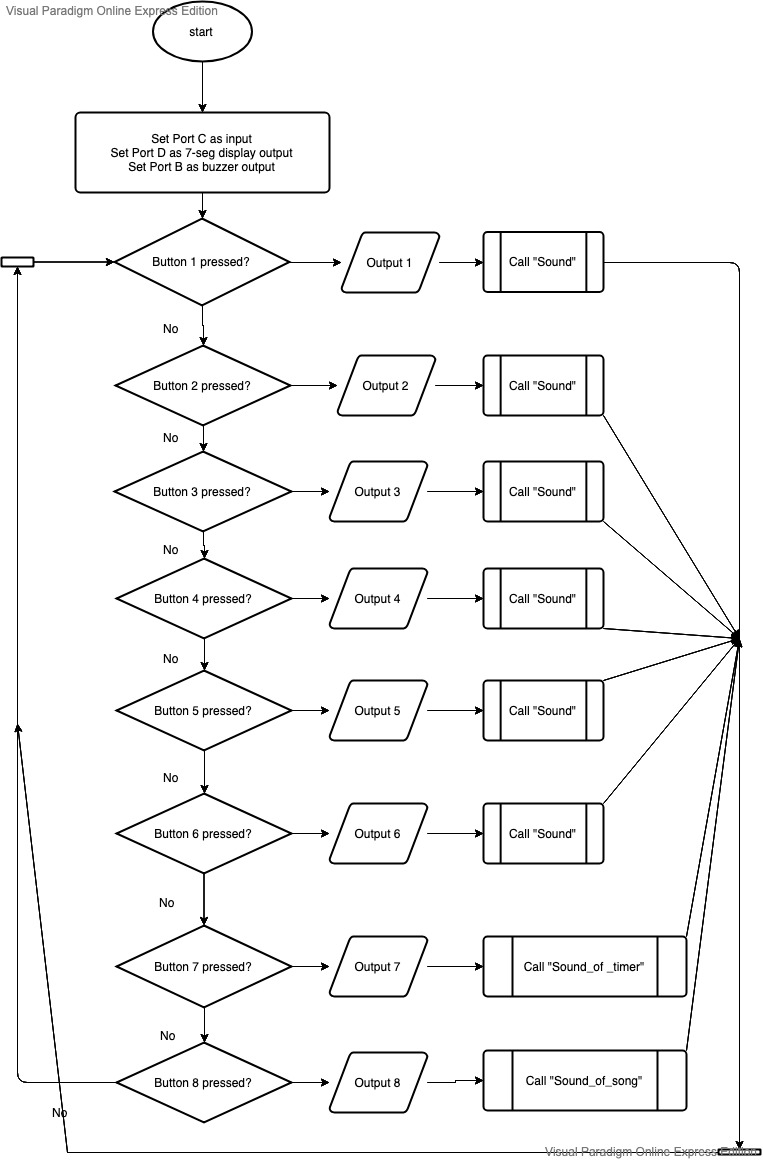
if so, decf con\_song and if con\_stop not equal to 0 bra again

else if con\_stop equal to 0, con\_stop decf

if con\_stop = 0, return

else bra stop

Flow Chart:



**Appendix**

Source Code

LIST P=18F4520

#include <P18F4520.INC>

CONFIG OSC = XT

CONFIG WDT = OFF

CONFIG LVP =OFF

CBLOCK 0X000

DELAY\_H

DELAY\_L

input

led

con\_long

con\_loop

con\_tone

con\_song

con\_stop

ENDC

org 0x0000

goto main

org 0x0100

main: movlw 0x00

movwf input

clrf TRISD

clrf PORTD

bcf TRISB,2

bcf PORTB,2

setf TRISC

SETF PORTC

;P8 係第一粒

call display

Loop call read\_press

call display\_1

goto Loop

display: movlw d'0'

movwf input

movwf PORTD

return

display\_1: movff led, PORTD

return

read\_press:

wait\_press: movf PORTC, w

xorlw 0xff

BNZ next\_1

bra wait\_press

next\_1: btfsc PORTC, 0 ;user timer to generate delay

bra next\_2

movlw b'11111110'

movwf input

movlw b'01111111'

movwf led

call sound\_of\_timer

bra exit

next\_2: btfsc PORTC, 1

bra next\_3

movlw b'11111101'

movwf input

movlw b'00000111'

movwf led

call sound\_of\_song

bra exit

next\_3: btfsc PORTC, 2

bra next\_4

movlw b'11111011'

movwf input

movlw b'01111101'

movwf led

call sound

bra exit

next\_4: btfsc PORTC, 3

bra next\_5

movlw b'11110111'

movwf input

movlw b'01101101'

movwf led

call sound

bra exit

next\_5: btfsc PORTC, 4

bra next\_6

movlw b'11101111'

movwf input

movlw b'01100110'

movwf led

call sound

bra exit

next\_6: btfsc PORTC, 5

bra next\_7

movlw b'11011111'

movwf input

movlw b'01001111'

movwf led

call sound

bra exit

next\_7: btfsc PORTC, 6

bra next\_8

movlw b'10111111'

movwf input

movlw b'01011011'

movwf led

call sound

bra exit

next\_8: movlw b'01111111'

movwf input

movlw b'00000110'

movwf led

call sound

;call sound\_of\_song

exit: call delay

wait\_up: movf PORTC

xorlw 0xff

BNZ wait\_up

call delay

return

delay: MOVLW 0X7F

MOVWF DELAY\_H

LOP\_1: MOVLW 0

MOVWF DELAY\_L

LOP\_2: DECF DELAY\_L,F

BNZ LOP\_2

DECF DELAY\_H,F

BNZ LOP\_1

return

sound: movlw 0x40

movwf con\_long

there bsf PORTB,2

call delay\_buzzer

bcf PORTB,2

call delay\_buzzer

decf con\_long

bnz there

return

delay\_buzzer:

movlw 0x04

movwf con\_loop

a1 call find\_low

movwf con\_tone

hj decf con\_tone

bnz hj

decf con\_loop

bnz a1

return

find\_low:

btfss input, 0

retlw 0x2a

btfss input, 1

retlw 0x3a

btfss input, 2

retlw 0x4a

btfss input, 3

retlw 0x5a

btfss input, 4

retlw 0x6a

btfss input,5

retlw 0x7a

btfss input, 6

retlw 0x8a

btfss input, 7

retlw 0x9a

sound\_of\_timer:

bsf PORTB,2

movlw 0x08

movwf T0CON

BCF INTCON, TMR0IF

movlw 0x01

movlw TMR0H

movlw 0x00

movlw TMR0L

BTG PORTB,2

start bsf T0CON,TMR0ON

time\_back

BTFSS INTCON, TMR0IF

bra time\_back

BCF T0CON, TMR0ON

bcf PORTB,2

return

sound\_of\_song:

movlw b'01111111' ;1

movwf input

call sound

call delay\_song

movlw b'01111111' ;1

movwf input

call sound

call delay\_song

movlw b'11110111' ;5

movwf input

call sound

call delay\_song

movlw b'11110111' ;5

movwf input

call sound

call delay\_song

movlw b'11111011' ;6

movwf input

call sound

call delay\_song

movlw b'11111011' ;6

movwf input

call sound

call delay\_song

movlw b'11011111' ;5

movwf input

call sound

call delay\_song

call delay\_song

movlw b'11101111' ;4

movwf input

call sound

call delay\_song

movlw b'11101111' ;4

movwf input

call sound

call delay\_song

movlw b'11011111' ;3

movwf input

call sound

call delay\_song

movlw b'11011111' ;3

movwf input

call sound

call delay\_song

movlw b'10111111' ;2

movwf input

call sound

call delay\_song

movlw b'10111111' ;2

movwf input

call sound

call delay\_song

movlw b'01111111' ;1

movwf input

call sound

return

delay\_song:

movlw 0x40

movwf con\_stop

stop movlw 0xff

movwf con\_song

movlw 0x30

movwf T1CON

song\_again

movlw 0xff

movwf TMR1H

MOVLW 0XFF

MOVWF TMR1L

BSF T1CON, TMR1ON

song\_here

btfss PIR1, TMR1IF

BRA song\_here

BCF T1CON, TMR1ON

decf con\_song

bnz song\_again

decf con\_stop

bnz stop

return

END