简易电子琴的设计和实现

1. 功能设计

播放功能：可以播放两首不同乐曲，分别是清华大学校歌以及偏爱；

弹奏功能：可以弹奏3个八度音程。

使用说明：

主界面：L3、L4、L5、L6都亮着，代表可以选择下面两个功能:按下K1选择弹奏功能；按下K2选择播放功能。

**弹奏功能：**

1、进入弹奏功能时L5、L6灭；

2、分别按下K1—K7可以发出音符1-7的音调；

3、按下K8键可以选择不同的音域；

4、L3和L4两灯的光亮表示不同的音域：

L3亮、L4灭——低音

L3灭、L4亮——中音

L3亮、L4亮——高音

5、同时按下K2和K8两个按钮可以退出弹奏功能，返回最初的功能选择界面。

**播放功能：**

1、进入播放功能时L3、L4灭，L5、L6亮，表示可以选择曲目；

2、按下K1表示选择第一首歌即清华校歌，此时L5亮，L6灭；按下K2表示选择第二首歌即偏爱，此时L6亮，L5灭；

3、曲目播放完毕会回到选择界面，4个灯全亮。

4、播放过程中，可以按K8中止播放，返回选择界面。

1. 电子琴硬件构成示意图

一張含有 文字, 圖表, 行, 設計 的圖片

自動產生的描述

1. 已调试完成的功能

播放功能：可以播放两首不同乐曲，分别是清华大学校歌以及偏爱；

弹奏功能：可以弹奏3个八度音程。

1. 难点问题及解决方法
2. 歌曲的乐谱不能直接找到，需要自己慢慢调整

最后花了大量时间调整和在同学的帮助下成功把两道歌的乐谱录入单片机

1. 单片机的引脚有限，在实现弹奏功能时对引脚的分配要合理

利用一个引脚多个功能的做法，成功解决引脚不足问题

1. Debug过程中，两个功能间的跳转常常失败

最后通过节点和分析语句的处理逻辑后，花费大量时间终于完整实现两个功能

1. 收获
2. 在一周的时间内快速掌握了CCS的基本开发
3. 理解单片机中一些底层硬件的大概原理
4. 动手调试能力有了很大的提升

代码

**#include** "msp430.h"

**#define** Rhythm 1.9 //Rhythm用來調整歌的節奏

**const** **char** tune1[207] = { 22, 65, 67, 37, 7, 64, 34, 22, 35, 35, 39, 38, 68, 37,

67, 7, 82, 67, 68, 67, 7, 52, 63, 63, 35, 64, 4, 22,

22, 65, 67, 37, 37, 68, 69, 39, 67, 69, 9, 52, 69, 69,

68, 11, 41, 67, 37, 68, 67, 7, 22, 69, 70, 69, 70, 39,

67, 67, 67, 65, 68, 8, 22, 22, 40, 40, 40, 69, 69, 38,

68, 69, 9, 52, 69, 69, 69, 12, 38, 69, 69, 9, 52, 65,

65, 35, 36, 67, 39, 68, 8, 8, 9, 9, 44, 44, 14, 67,

74, 44, 74, 22, 67, 44, 73, 72, 72, 71, 39, 9, 69, 68,

67, 65, 67, 38, 69, 9, 64, 67, 38, 69, 22, 65, 67, 38,

69, 9, 9, 22, 44, 44, 44, 73, 72, 72, 43, 74, 52, 67,

67, 44, 74, 73, 72, 41, 69, 9, 69, 68, 67, 65, 9, 69,

68, 67, 65, 9, 69, 68, 67, 65, 65, 38, 38, 8, 52, 44,

74, 73, 72, 74, 74, 71, 12, 42, 22, 71, 70, 69, 70,

70, 68, 69, 68, 8, 8, 73, 73, 72, 73, 44, 44, 79, 9,

42, 42, 71, 71, 39, 68, 8, 8, 22, 22, 22, 22 };

**const** **char** tune2[115] = { 7, 37, 39, 11, 11, 12, 44, 42, 11, 11, 9, 9, 41, 39,

7, 5, 37, 38, 11, 11, 12, 12, 42, 44, 11, 9, 8, 39,

38, 7, 8, 11, 41, 40, 11, 12, 12, 43, 42, 11, 14, 44,

44, 12, 14, 11, 41, 42, 11, 11, 12, 42, 42, 11, 9, 8,

38, 39, 11, 11, 7, 37, 37, 7, 9, 8, 39, 38, 7, 7, 12,

42, 42, 11, 9, 8, 39, 38, 7, 7, 14, 14, 14, 22, 12,

12, 12, 22, 11, 11, 12, 11, 8, 9, 11, 11, 14, 14, 14,

22, 12, 12, 12, 22, 11, 11, 12, 11, 8, 39, 38, 7, 7,

22, 22 };

//tune1和tune2存放了兩首歌的樂譜

**const** **char** melody1[21] = { 250, 222, 198, 187, 166, 148, 132, 124, 111, 98, 93,

83, 73, 65, 62, 55, 49, 46, 41, 36, 32 };

**const** **char** melody2[21] = { 25, 22, 19, 18, 16, 14, 13, 12, 11, 9, 9, 8, 7, 6, 6,

5, 4, 4, 4, 3, 3 };

//melody1和melody2存放了7個音調

**void** **initializePins**(); //初始化管腳

**void** **initializeInterrupts**(); //初始化中斷管腳

**void** **int\_light**(); //控制四个燈的狀態

**void** **ctrl\_light1\_1**();

**void** **ctrl\_light1\_2**();

**void** **ctrl\_light1\_3**();

**void** **light2**();

**void** **crtl\_light2\_1**();

**void** **crtl\_light2\_2**();

**unsigned** pauseif = 0; //結束功能

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

{

WDTCTL = WDTPW + WDTHOLD;

initializePins(); //初始化引脚

initializeInterrupts();

**unsigned** **int** pitchRange = 0; //pitchRange用來存放音域，0為低音，1為中音，2為高音

**unsigned** **int** note = 22; //note用來記錄按下的音調()

**unsigned** **int** i;

**unsigned** **int** tone;

**unsigned** **int** save;

**while** (1)

{

int\_light();

pauseif = 0; //pauseif为1时回到功能选择界面

**while** ((P1IN & BIT0) != 0 & (P1IN & BIT1) != 0)

;

**if** ((P1IN & BIT0) == 0)

{ //按下K1，進入功能1

**\_\_delay\_cycles**(500000); //延時，避免一次按鍵多次回應

**while** (pauseif == 0)

{

note = 22;

**if** ((P1IN & BIT7) == 0)

{

pitchRange++;

**\_\_delay\_cycles**(500000);

**if** (pitchRange == 3)

pitchRange = 0;

}

**if** (pitchRange == 0) //不同音調下的亮燈

ctrl\_light1\_1();

**else** **if** (pitchRange == 1)

ctrl\_light1\_2();

**else** **if** (pitchRange == 2)

ctrl\_light1\_3();

**if** ((P1IN & BIT0) == 0) //劃分音調

note = 7 \* pitchRange;

**else** **if** ((P1IN & BIT1) == 0)

note = 7 \* pitchRange + 1;

**else** **if** ((P1IN & BIT2) == 0)

note = 7 \* pitchRange + 2;

**else** **if** ((P1IN & BIT3) == 0)

note = 7 \* pitchRange + 3;

**else** **if** ((P1IN & BIT4) == 0)

note = 7 \* pitchRange + 4;

**else** **if** ((P1IN & BIT5) == 0)

note = 7 \* pitchRange + 5;

**else** **if** ((P1IN & BIT6) == 0)

note = 7 \* pitchRange + 6;

**if** (note != 22) //輸出音調

{

TA1CCR0 = melody1[note];

TA1CCR1 = melody2[note];

**\_\_delay\_cycles**(400000 \* Rhythm);

TA1CCR0 = 32767;

TA1CCR1 = 32767;

}

**else** **if** (note == 22)

{

TA1CCR0 = 32767;

TA1CCR1 = 32767;

}

}

} //功能1結束

**else** **if** ((P1IN & BIT1) == 0)

{ //进入播放功能

light2();

**\_\_delay\_cycles**(500000);

**while** ((P1IN & BIT0) != 0 & (P1IN & BIT1) != 0)

; //等待按键选择

**if** ((P1IN & BIT0) == 0)

{ //选择第一首曲子

save = 115; //save存储了这首曲子的音符总数

crtl\_light2\_1();

}

**else** **if** ((P1IN & BIT1) == 0)

{ //选择第二首曲子

save = 207;

crtl\_light2\_2();

}

i = 0;

**while** ((i < save) & (pauseif == 0))

{

**if** (save == 207)

tone = tune1[i];

**else** **if** (save == 115)

tone = tune2[i];

**if** (tone == 22)

{

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(400000 \* Rhythm);

}

**else** **if** (tone == 52)

{

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(200000 \* Rhythm);

}

**else** **if** (tone == 82)

{

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(100000 \* Rhythm);

}

**else** **if** (tone >= 30 & tone < 60)

{

TA1CCR0 = melody1[tone - 30];

TA1CCR1 = melody2[tone - 30];

**\_\_delay\_cycles**(180000 \* Rhythm);

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(20000 \* Rhythm);

}

**else** **if** (tone >= 60)

{

TA1CCR0 = melody1[tone - 60];

TA1CCR1 = melody2[tone - 60];

**\_\_delay\_cycles**(80000 \* Rhythm);

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(20000 \* Rhythm);

}

**else**

{

TA1CCR0 = melody1[tone];

TA1CCR1 = melody2[tone];

**\_\_delay\_cycles**(370000 \* Rhythm);

TA1CCR0 = 32767;

TA1CCR1 = 32767;

**\_\_delay\_cycles**(30000 \* Rhythm);

}

i++;

}

} //功能2结束

}

}

**void** **initializePins**(**void**)

{

P2SEL |= BIT1;

P2SEL2 &= ~BIT1;

P2DIR |= BIT1;

P2DIR |= BIT2 + BIT3 + BIT4 + BIT5;

P2OUT |= BIT2 + BIT3 + BIT4 + BIT5;

P1OUT = 0xff;

P1REN = 0xff;

TA1CTL |= TASSEL0;

TA1CCR0 = 250;

TA1CCTL1 = OUTMOD\_2;

TA1CCR1 = 25;

TA1CTL |= TACLR + MC0;

TA1CCR0 = 32767;

TA1CCR1 = 32767;

}

**void** **initializeInterrupts**()

{

\_DINT();

P1IES |= BIT7;

P1IFG &= ~BIT7;

P1IE |= BIT7;

\_EINT();

}

**void** **int\_light**()

{

P2OUT &= ~(BIT2 + BIT3 + BIT4 + BIT5);

}

**void** **ctrl\_light1\_1**()

{

P2OUT &= ~BIT2;

P2OUT |= BIT3 + BIT4 + BIT5;

}

**void** **ctrl\_light1\_2**()

{

P2OUT &= ~BIT3;

P2OUT |= BIT2 + BIT4 + BIT5;

}

**void** **ctrl\_light1\_3**()

{

P2OUT &= ~(BIT2 + BIT3);

P2OUT |= BIT4 + BIT5;

}

**void** **light2**()

{

P2OUT &= ~(BIT4 + BIT5);

P2OUT |= BIT2 + BIT3;

}

**void** **crtl\_light2\_1**()

{

P2OUT &= ~BIT4;

P2OUT |= BIT2 + BIT3 + BIT5;

}

**void** **crtl\_light2\_2**()

{

P2OUT &= ~BIT5;

P2OUT |= BIT2 + BIT3 + BIT4;

}

**#pragma** vector=PORT1\_VECTOR

**\_\_interrupt** **void** **pause1**()

{

**if** ((P1IFG & BIT7) != 0)

{

P1IFG &= ~BIT7;

**if** ((P2OUT & BIT2) != 0 & (P2OUT & BIT3) != 0)

pauseif = 1;

**else** **if** ((P1IN & BIT1) == 0)

pauseif = 1;

}

}