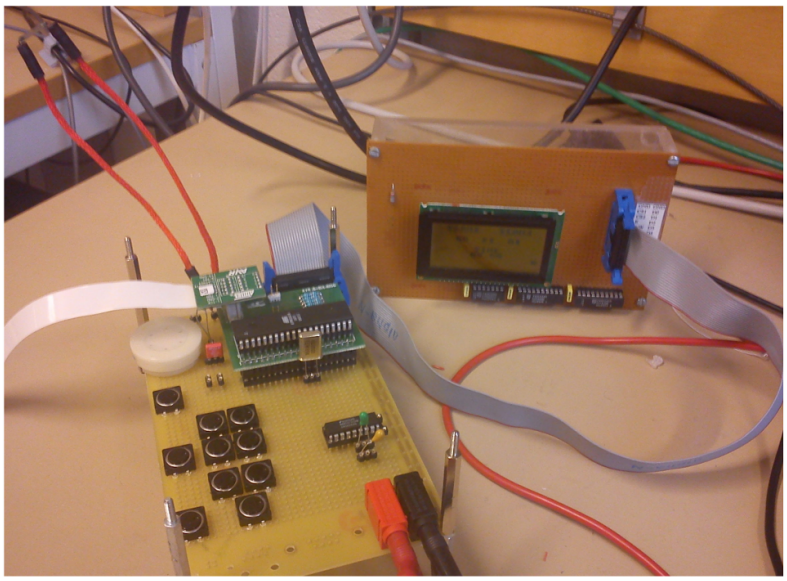
HockeyMate
Projektarbete i EITF40 - Digitala Projekt
Institutionen för elektro- och informationsteknik

Grupp 7
Niclas Thuning, 860215-3994, et06nt3@student.lth.se
Carl Cristian Arlock, 840306-3954, dt07ca7@student.lth.se
Handledare: Bertil Lindvall



Abstract

This report concludes the project named HockeyMate. The students planned, designed, built and programmed a jumbotron [1] made for table icehockey games. The project was successful and produced a working prototype.

Innehåll

1 Inledning

I detta projekt valde deltagarna att bygga en jumbotron till bordshockeyspel. Jumbotron är ett registrerat varumärke ägt av Sony, men ordet används ofta som benämning för produkttypen [1]. En jumbotron är en skärm som visar information som till exempel kvarvarande tid och mål för sportevenmanget där den används.

När folk spelar bordshockey kan det ibland vara svårt att hålla reda på mål, tid och periodnummer på ett relativt exakt sätt. Deltagarna i projektet kom på idén att bygga ett system med tillhörande skärm som kan hålla reda på alla detaljer medan spelarna kan ägna sig åt spelandet. Systemet skall kunna känna av när ett mål görs, räkna ner tid och visa all relevant data på en skärm. Dessutom skall systemet kunna påkalla spelarnas uppmärksamhet med en högtalare.

Det finns redan elektroniska hjälpmedel, men dessa har inte alla funktioner som kanske efterfrågas. Det finns applikationer för ändamålet till telefoner och surfplattor, men alla har inte tillgång till något sådant.

2 Problemställning

2.1 Kravspecifikation

Dessa krav sammanställdes vid start av projektet.

Det skall konstrueras en elektronisk enhet som håller reda på allt som har med bordshockeyspelet att göra. Den skall kunna visa kvarvarande periodtid, periodnummer och mål. Den skall även automatiskt känna av när ett mål görs. Dessutom skall det finnas en uppsättning knappar för inställningar, start och liknande. Det skall även finnas en högtalare så att enheten kan påkalla uppmärksamhet när det blir mål eller en period tar slut. Detta alternativ skall vara enkelt, stilrent och billigt jämfört med konkurrenterna.

2.1.1 Krav

- Skärm skall uppdateras inom rimlig tid
- Skärm skall visa aktuell tid
- Skärm skall visa antal mål per lag
- Skärm skall visa periodnummer
- När det blir mål skall spelet stannas och måste återupptas manuellt. Med stannas innebär tiden pausas och inga mål kan bli gjorda.
- När det blir mål skall högtalaren spela ett ljud
- När tiden tar slut skall högtalaren spela ett ljud
- Knappar som skall ingå:
  - Start/stop
  - Reset för hela enheten, där allt nollställs

- Öka antalet minuter
- Minska antalet minuter
- Mål lag 1 öka
- Mål lag 1 minska
- Mål lag 2 öka
- Mål lag 2 minska
- Öka periodnummer
- Högtalare av/på

2.2 Sammanställning

Ett enkelt utbyggbart system som kan hantera de funktioner som efterfrågas bör kretsa kring någon typ av AVR-processor som kan programmeras efter behag. Det behövs en bra bunt pinnar för in- och utdata, avbrottsrutiner samt räknare. För att kunna visa all relevant data på ett snyggt sätt behövs en LCD-skärm med ett lagom antal pixlar. Fördelen mot en sjusegmentsdisplay är att systemet lätt kan byggas ut, tydligheten ökar och det blir mer en jumbotron.

Knapp	Antal	Funktion
Tid	2	Ställa in tid
Räknare	1	Av- och påknapp för räknaren
Mål	4	Manuell inställning av mål
Period	1	Periodnummerinställning
Reset	1	Återställning av systemet
Högtalare	1	Högtalare av och på

Till knapparna behövs en avkodare, så att inte varje knapp använder varsin pinne på AVR:en.

För att systemet ska kunna känna av mål behövs två sensorer. Dessa behöver inte vara avancerade.

Högtalaren behöver ingen särskild funktion, mer än en PWM-signal för enkla melodier. Kanske behövs det någon typ av filter också.

3 Utförande

Efter sammanställning av krav växte en design fram. Systemet kretsar kring en AVR, närmare bestämt en ATMega16 från Atmel. Denna mikrokontroller har en 8-bitars processor, 16 KB flashminne, 1 KB ramminne, stöd för upp till 16 Mhz klockfrekvens och 32 in- och utpinnar. För att räknaren ska bli mer exakt föreslog handledaren en 16 Mhz-kristall direkt.

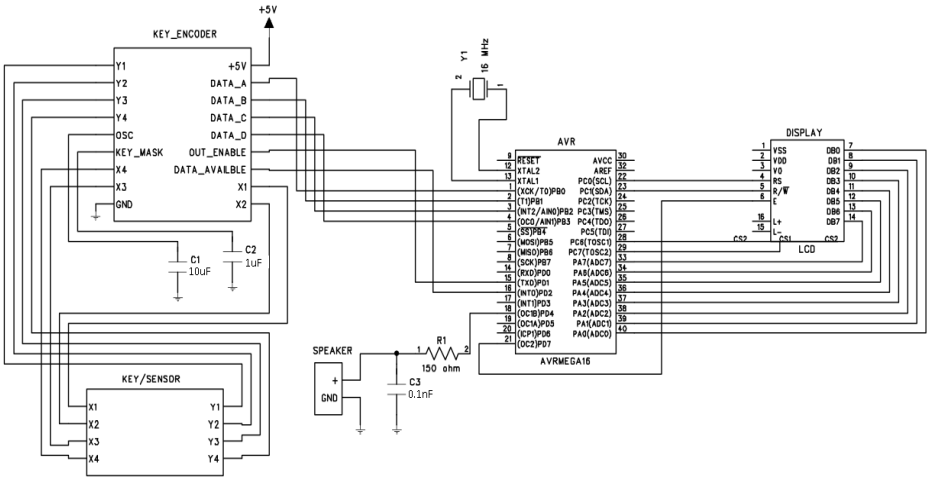
Knappavkodaren valdes till en MM74C922 16-Key Encoder från National Semiconductor. Avkodaren har stöd för upp till sexton knappar vilket är mer än nog för systemet, som enligt sammanställningen endast behöver tio. Avkodaren skickar en fyrabitars bitkod samt en extra bit för signal till AVR:en. Totalt behöver avkodaren alltså fem pinnar. Avkodaren har dessutom inbyggt skydd från multipla knapptryckningar med hjälp av kondensatorer. Till detta ändamål valdes två kondensatorer på 1μF och 10μF efter databladets rekommendation.

Som knappar valdes enkla brytare med fyra in- och utpinnar varav två pinnar per knapp används till en början.

Högtalaren valdes till en lättdriven liten peizohögtalare. Till denna designades ett lågpassfilter, för att bli av med oönskat ljud. Lågpassfiltret består av en kondensator på 0.1nF och ett motstånd på 150ohm. Då hamnar brytfrekvensen på 10 kHz. Högtalaren kopplades till AVR:ens PWM-utgång.

Sensorerna är ett par magnetsensorer som ger kortslutning vid tillräckligt starkt magnetfält. Idéen är att ha en magnet i pucken som kortsluter sensorn och denna i sin tur skickar en signal till AVR:en. Ett par lysdioder kopplas på som likriktare och indikatorer för mål. Efter en tids fundering togs lysdioderna bort och magnetsensorerna kopplades direkt till knappavkodaren, då de fungerar som knappar. Denna lösning var mycket enklare och frigjorde två pinnar samt ett avbrott.

Den grafiska skärmen valdes till en 128 gånger 64 pixlars LCD av typen GDM-12864C. Skärmen har ett eget ramminne.

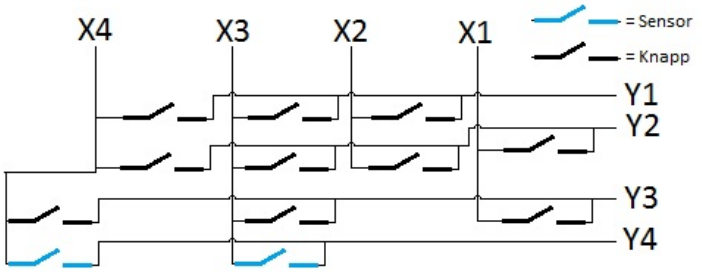


Figur 1: Blockschema

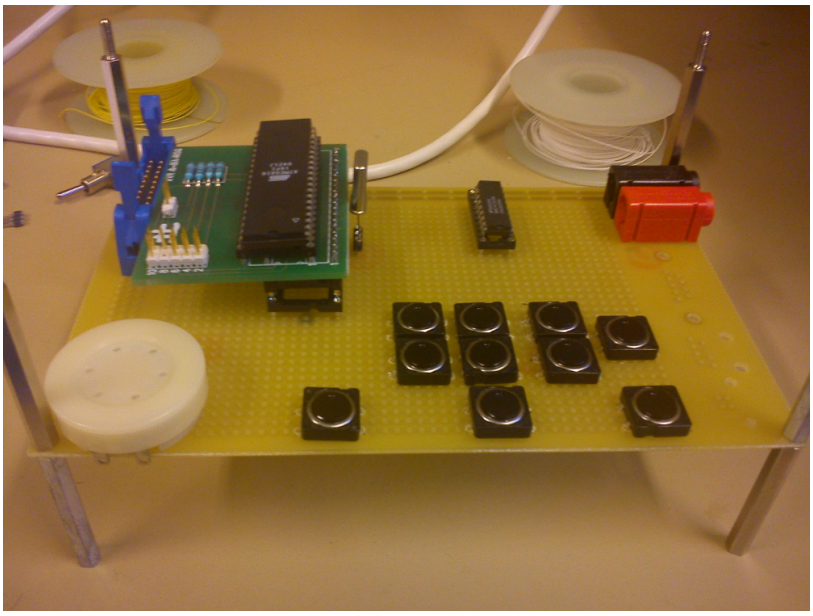
I figur 1 är alla anslutningar förutom knapparna och sensorerna utmarkerade. I figur 2 är knapparna och sensorerna avbildade. `X`- och `Y`-variablerna i figur 1 på `KEY/SENSOR`-komponenten matchar de i figur 2.

Komponenterna lödades på plats och sladdar lödades och virades. Knappar och högtalare limmades då dessa inte gick att löda fast.

När allt var på plats påbörjades kodandet. AVR:en programmerades i AVR Studio 4 där möjligheten finns att programmera i C som sedan omvandlas till assembler för AVR:en. Den första uppgiften var att få någon typ av indikation från systemet. Efter lite eftertänksamhet började deltagarna programmera skärmen. Data från tidigare projekt låg kvar i skärmens minne vilket underlättade när kod för att rensa skärmen skrevs.



Figur 2: Knappar och sensorer



Figur 3: Utplacerade komponenter

För att skriva ut text på skärmen genererades en matris med bitkoder. Åtta gånger åtta pixlar per bokstav är standard för programmet. Det går självklart att ha vilka dimensioner som helst på texten, men dessa valdes för att få plats med allt på skärmen och samtidigt kunna skriva ut mer utifall att behov finns. En generell funktion för att skriva ut symboler på angiven plats skrevs. När denna var färdig och fungerande kunde systemet testas med lätthet.

Den data som är konstant, som till exempel matrisen för bokstäver, lagras i flashminnet. Annars kan ramminnet lätt fyllas.

Ett stort problem vid programmeringen var optimeringen och kompilatorns egenheter. Detta löstes med globala variabler. Vid ett tillfälle fungerade det inte alls att skicka med en temporär variabel till en funktion, då kompilatorn tog bort den direkt. Det löstes med en global variabel som ändras innan varje anrop till funktionen.

När systemet gav återkoppling på skärmen kunde knapparna testas. Knapparna fungerade från första början, så när som på bitkoderna som behövde justeras i vissa fall.

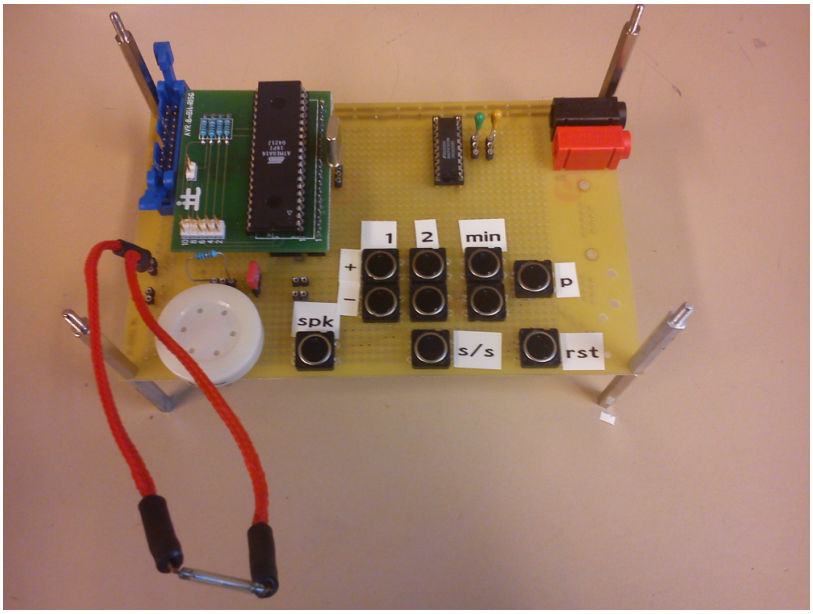
Den åttabitarsräknare som används konfigurerades och skickar signal via avbrott varje gång räknaren slår över. En variabel ökas då och huvudprogrammet läser av denna. Efter lite finjustering räknar programmet nu ner med ganska bra precision.

En allmän funktion för att skicka signal via PWM-utgången skrevs. Till denna kan man skicka in tempo, ackord och ljudvolym som sedan omvandlas till PWM-signal och spelas upp i högtalaren.

För att inte målsensorerna skall registrera flera mål samtidigt sattes en begränsning på max ett mål per sekund.

4 Resultat

Färdig prototyp med alla funktioner som efterfrågades i kravspecifikationen visas i figur 4.



Figur 4: Färdig design

Skärmens design visas i figur 5. `TEAM1` och `TEAM2` visar hur många mål respektive lag har. `P` anger vilken period som gäller. `TIME` anger hur mycket tid som är kvar. Den lilla ikonen nere till vänster skiftar mellan en högtalare och ett `X` anger om ljudet är aktiverat eller ej.

4.1 Genomgång av systemet

När systemet startas visar skärmen alla grundinställningar. Antal mål och tid är noll, periodnummer är ett. För att starta en match måste tiden ändras. De

Figur 5: Grafiskt gränssnitt

två knapparna under etiketten `min` ändrar antalet minuter upp och ner. För att sedan starta tiden trycker man på `s/s`. När antalet minuter når noll och perioden inte är tre kommer en melodi spelas och texten `TIME UP` skrivs ut på skärmen. När tiden går ut i period tre kommer det istället att stå `GAME OVER`. För att ändra periodnummer trycker man på `p`, värdet ändras mellan går från ett till tre och sedan tillbaka till ett igen. Alla knappar pausar tiden sånär som på den märkt `spk`. Denna knapp stänger av och sätter på högtalaren. De två långa röda kablarna med en magnetsensor på mitten känner av om det blir mål. I denna prototyp är sensorn satt på detta vis av praktiska skäl. När magneten kommer när nog spelas ett ljud från högtalaren och `GOAL` skrivs ut på skärmen under det lag som gjort mål. Knappen `rst` återställer hela systemet till grundkonfiguration.

I skrivande stund spelas en känd titelmelodi upp när tiden går ut. Detta kan ändras till vad som, men måste bestämmas vid kompilering av programmet.

Referenser

[1] http://en.wikipedia.org/wiki/Jumbotron

A Appendix

Programkod med kommentarer följer. Programmet är skrivet på engelska och således är kommentarerna också på engelska.

hockeymate.c

```
#include <avr/io.h>
#include <stdio.h>
#include <stdbool.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <math.h>
#include <stdlib.h>
#include "text.h" //where to put all the precoded characters
#include "notes.h" //where to put all the precoded notes

void send_e();
void draw_symbol(uint8_t x, uint8_t y, uint8_t cs);
void timer_ss();
void clear_display();
void draw_lcd();
void show_goal(int team);
void button_handler();
void timer_stop();
int return_number(int in);

#define A 0
#define B 1
#define C 2
#define D 3
#define E 4
#define F 5
#define G 6
#define H 7
#define I 8
#define J 9
#define K 10
#define L 11
#define M 12
#define N 13
#define O 14
#define P 15
#define Q 16
#define R 17
#define S 18
#define T 19
#define U 20
#define V 21
#define W 22
#define X 23
#define Y 24
#define Z 25
#define EXCL 26
#define FNUTT 27
#define BRAED 28
#define COMMA 29
#define PUNCT 30
#define ZERO 31
#define ONE 32
#define TWO 33
#define THREE 34
#define FOUR 35
#define FIVE 36
#define SIX 37
#define SEVEN 38
#define EIGHT 39
#define NINE 40
#define COLON 41
#define LESS 42
#define EQ 43
```

```
#define GREAT 44
#define QUESTION 45
#define HALFCOLON 46
#define HALFCOLON2 49
#define HALFIL 47
#define HALFIR 48
#define CS1 1
#define CS2 2
#define CS3 3
#define DEFAULT_VOLUME 100
#define EMPTY 50
#define SPEAKER 51

/*
 * pa0 - lcd d0 - out
 * pa1 - lcd d1 - out
 * pa2 - lcd d2 - out
 * pa3 - lcd d3 - out
 * pa4 - lcd d4 - out
 * pa5 - lcd d5 - out
 * pa6 - lcd d6 - out
 * pa7 - lcd d7 - out
 *
 * pb0 - buttonc da - in
 * pb1 - buttonc db - in
 * pb2 - buttonc dc - in
 * pb3 - buttonc dd - in
 *
 * pc0 - lcd r/w - out
 * pc1 - lcd d/i(rs) - out
 * pc6 - lcd cs1 - out
 * pc7 - lcd cs2 - out
 *
 * pd0 - lcd rst
 * pd1 - button out enable - out
 * pd2 - button interrupt - in
 * pd3 - sensor interurpt - in
 * pd4 - speaker pwm - out
 * pd5 - g1 sensor - in
 * pd6 - g2 sensor - in
 * pd7 - lcd e - out
 *
 *
 */

// Star Wars
const int starwars[] =
{
  Ais2,8, Ais2,8, P,16, F3,8, F3,8, P,16, Dis3,16, P,16, D3,16, P,16, C3,16, P,16, Ais3,8,
  Ais3,8, P,16, F3,8, P,16, Dis3,16, P,16, D3,16, P,16, C3,16, P,16, Ais3,8, Ais3,8, P,16,
  F3,8, P,16, Dis3,16, P,16, D3,16, P,16, Dis3,16, P,16, C3,8, C3,8,
  MUSIC_END
};
// Fur Elise
const int furelise[] =
{
    e4, 8, d4x, 8, e4, 8, d4x, 8, e4, 8, b3, 8, d4, 8, c4, 8, a3,8, p, 8,
    c3, 8, e3, 8, a3, 8,  b3, 4, p, 8, e3, 8, g3x, 8, b3, 8, c4, 4, p, 8, e3, 8,
    e3, 8, d4x, 8, e4, 8, d4x, 8, e4, 8, b3, 8, d4, 8, c4, 8, a3, 8, p, 8, c3, 8,
    e3, 8, a3, 8, b3, 4, p, 8, e3, 8, c4, 8, b3, 8, a3, 4,
    MUSIC_END
};

// Beatles, Hey Jude
const int Jude[] = {
  G2,8, E2,8, P,16, E2,16, E2,16, G2,16, A2,16, D2,8, P,16, D2,16, E2,16, F2,8,
  C3,8, C3,16, C3,16, H2,16, G2,16, A2,16, G2,16, F2,16, E2,8, P,16, G2,16,
  A2,16, A2,8, A2,16, D3,16, C3,16, H2,16, H2,16, C3,16, A2,16, G2,8, P,16,
  C2,16, D2,16, E2,16, A2,16, A2,16, G2,8,
  MUSIC_END
};

//Goal sound
const int goal[] = {
```

```
  G2,8, G2,8,G2,8,G2,8,
  MUSIC_END
};


const int octave[] = {c4, 8, d4, 8, e4, 8, f4, 8, g4, 8, a4, 8, h4, 8, c5, 8, MUSIC_END};

volatile int seconds;
volatile int counter;
volatile int timer_running;
volatile uint8_t g1;
volatile uint8_t g2;
volatile uint8_t period;
volatile uint8_t goal1;
volatile uint8_t goal2;
volatile uint8_t ctw; //the compiler is borked, this is an ugly solution.
volatile uint8_t sjab;
volatile uint8_t volume;
volatile int gdebounce;
volatile uint8_t flag;

void PlayMusic( const int* pMusicNotes /** Pointer to table containing music data */,
        uint8_t tempo /** paying tempo from 0 to 100. Higher value = slower playback*/ )
{
  int duration;
  int note;
  int i;
  uint16_t delay = tempo * 1000*2;

  while( *pMusicNotes )
  {
    note = *pMusicNotes;
    pMusicNotes++;

    duration = *pMusicNotes;
    pMusicNotes++;

    if( p == note )
    {
      //pause, do not generate any sound
      OCR1B = 0;
    }
    else
    {
      //not a pause, generate tone
      OCR1B = volume;

      //set frequency
      ICR1H = (note >> 8);
      ICR1L = note;
    }

    //wait duration
    for(i=0;i<32-duration;i++)
    {
      _delay_loop_2( delay );
    }

  }


  //turn off any sound
  OCR1B = 0;
}


/*
*Write to lcd
*/
void send_e(){
  PORTD|= (1<<PD7);
  _delay_us (100); //100us delay between e and !e, needed for the lcd to catch up.
  PORTD&= ~(1<<PD7);
  _delay_us (100); //100us delay between !e and e, needed for the lcd to catch up.
```

```
}


/*
* Start and stop the timer
*/
void timer_ss(){
  timer_running = !timer_running;
  if(timer_running){
    TIMSK |= (1<<TOIE0); //enable timer
  }else{
    TIMSK &= ~(1<<TOIE0); //disable timer
  }
}

/*
* Clear the display
*/
void clr_display(){
  uint8_t x;
  uint8_t y;
  PORTC&=~(1<<PC7); //set cs2 = 0
  PORTC&=~(1<<PC6); //set cs1 = 1
  PORTA = 0b01000000; //set y
  send_e();
  for(x = 0; x < 8; x++){
    PORTA = 0b10111000 + x; //set x
    send_e();
    PORTC|=(1<<PC1); //rs = 1
    PORTA = 0b00000000; //data to write
    for( y = 0; y < 64; y++){
      send_e();
    }
    PORTC&=~(1<<PC1); //rs = 0
    send_e();
  }
}

/*
* Initiate the display, to enable it and so on.
*/
void init_lcd(){
  cli(); //disable interrupts
  PORTD|=(1<<PD0); //sending rst = 1 to lcd. 0 = reset, 1 = normal operation.
  PORTC&=~(1<<PC0); //r/w to lcd
  PORTC&=~(1<<PC1); //rs to lcd
  PORTA = 0b00111111; //display on to lcd
  send_e();//write to lcd
  sei(); //re-enable interrupts
}

/*
* Draw the GUI
*/
void draw_lcd(){
  init_lcd();
  clr_display();
  uint8_t i;
  PORTC&=~(1<<PC7); //set cs2 = 1
  PORTC&=~(1<<PC6); //set cs1 = 1
  PORTA = 0b01001100; //set y
  send_e();
  PORTA = 0b10111001; //set x
  send_e();
  PORTA = 0b00000001; //data to write
  PORTC|=(1<<PC1); //rs = 1
  for(i = 0; i < 40; i++){
    send_e();
  }

  PORTC&=~(1<<PC1); //rs = 0

  PORTC&=~(1<<PC1); //rs = 0
  ctw = T; //the compiler is borked. this is one ugly solution.
```

```
  draw_symbol(0, 12, CS3);
  draw_symbol(4, 48, CS1);
  ctw = E;
  draw_symbol(0, 20, CS3);
  draw_symbol(4, 8, CS2);
  ctw = A;
  draw_symbol(0, 28, CS3);
  ctw = M;
  draw_symbol(0, 36, CS3);
  draw_symbol(4, 0, CS2);
  ctw = ONE;
  draw_symbol(0, 44, CS1);
  draw_symbol(2, 0, CS2);
  ctw = TWO;
  draw_symbol(0, 44, CS2);
  ctw = P;
  draw_symbol(2, 56, CS1);

  ctw = I;
  draw_symbol(4, 56, CS1);
  ctw = HALFCOLON;
  draw_symbol(5, 60, CS1);

  ctw = ZERO;
  draw_symbol(5, 44, CS1);
  draw_symbol(5, 52, CS1);

  draw_symbol(5, 4, CS2);
  draw_symbol(5, 12, CS2);

  draw_symbol(2, 24, CS3);
  draw_symbol(2, 32, CS3);

  ctw = SPEAKER;
  draw_symbol(6, 56, CS2);

}

/*
* Draw symbol from the matrix.
*/
void draw_symbol(uint8_t x, uint8_t y, uint8_t cs){
  volatile uint8_t i;
  volatile uint8_t j;
  if(cs == 2){
    PORTC&=~(1<<PC7); //set cs2 = 0
    PORTC|=(1<<PC6); //set cs1 = 1
  }else if(cs == 1){
    PORTC&=~(1<<PC6); //set cs1 = 0
    PORTC|=(1<<PC7); //set cs2 = 1
  }else{
    PORTC&=~(1<<PC7); //set cs2 = 1
    PORTC&=~(1<<PC6); //set cs1 = 1
  }
  PORTA = 0b01000000 + y;
  send_e();
  PORTA = 0b10111000 + x;
  send_e();
  PORTC|=(1<<PC1); //rs = 1
  for(i = 0; i < 8; i++){
    sjab = 0b00000000;
    for(j = 0; j < 8; j++){
      if((7 - i - j) >= 1){
        sjab |= ((pgm_read_byte(&fs[ctw][j]) & (0b10000000 >> i)) >> (7-i-j));
      }else{
        sjab |= ((pgm_read_byte(&fs[ctw][j]) & (0b10000000 >> i)) << abs(7-i-j));
      }
    }
    PORTA = sjab;
    send_e();
  }
  PORTC&=~(1<<PC1); //rs = 0
}
```

```
/*
* Takes care of all the buttons and sensors
*/
void button_handler(int button_type){
  switch(button_type){
    case 0b11111010: timer_ss(); break; //ss button - OK
    case 0b11110011: timer_stop();
            if((g1 >= 0) && (g1 < 99)){ //g1+ button - OK
              g1++;
            }else if(g1 >= 99){
              g1 = 0;
            } break;
    case 0b11110111: timer_stop();
            if((g1 > 0) && (g1 < 100)){ //g1- button - OK
              g1--;
            }else if(g1 <= 0){
              g1 = 99;
            } break;
    case 0b11110010: timer_stop();
            if((g2 >= 0) && (g2 < 99)){ //g2+ button - OK
              g2++;
            }else if(g2 >= 99){
              g2 = 0;
            } break;
    case 0b11110110: timer_stop();
            if((g2 > 0) && (g2 < 100)){ //g2- button - OK
              g2--;
            }else if(g2 <= 0){
              g2 = 99;
            }
            break;
    case 0b11110001: timer_stop();
            if(seconds < 5940){ //minplus button - OK
              seconds += 60;
            }
            break;
    case 0b11110101: timer_stop();
            if(seconds > 60){ //minmin button - OK
                seconds -= 60;
              }else{
                seconds = 0;
              }
              break;
    case 0b11110100: timer_stop();
          if(period < 3){ //period button - OK
             period++;
           }else{
             period = 1;
           }
           break;
    case 0b11111000: TCNT0 = 0; //RESET OK! //ccw
         g1 = g2 = counter = seconds = gdebounce = flag = 0;
         timer_running = false;
         period = 1;
         draw_lcd();
         timer_stop();
         break; //reset
    case 0b11111011: if(volume == 100){   //volume button. sets speaker on/off
              volume = 0;
              ctw = X;
              draw_symbol(6, 56, CS2);
            }else{
                volume = 100;
                ctw = SPEAKER;
                draw_symbol(6, 56, CS2);
                }
                break;
    case 0b11111110: timer_stop();
            if(gdebounce != seconds){
              g1++;
              goal1 = 1;
              gdebounce = seconds;
            }
            break;
```

```
435     case 0b11111111: timer_stop();
436             if(gdebounce != seconds){
437               g2++;
438               goal2 = 1;
439               gdebounce = seconds;
440             }
441             break;
442     default: break;
443   }
444 }
445
446 /*
447 * Converts input number to constant
448 */
449 int return_number(int in){
450   volatile uint8_t rv;
451   switch(in){
452     case 0: rv = ZERO; break;
453     case 1: rv = ONE; break;
454     case 2: rv = TWO; break;
455     case 3: rv = THREE; break;
456     case 4: rv = FOUR; break;
457     case 5: rv = FIVE; break;
458     case 6: rv = SIX; break;
459     case 7: rv = SEVEN; break;
460     case 8: rv = EIGHT; break;
461     case 9: rv = NINE; break;
462     default: break;
463   }
464   return rv;
465 }
466
467 /*
468 * Update the lcd with current data
469 */
470 void update_lcd(){
471   //goals
472   if(g1 >= 10){
473     ctw = return_number(g1/10);
474     draw_symbol(2, 24, CS1);
475     ctw = return_number(g1%10);
476     draw_symbol(2, 32, CS1);
477   } else{
478     ctw = ZERO;
479     draw_symbol(2, 24, CS1);
480     ctw = return_number(g1);
481     draw_symbol(2, 32, CS1);
482   }
483   if(g2 >= 10){
484     ctw = return_number(g2/10);
485     draw_symbol(2, 24, CS2);
486     ctw = return_number(g2%10);
487     draw_symbol(2, 32, CS2);
488   } else{
489     ctw = ZERO;
490     draw_symbol(2, 24, CS2);
491     ctw = return_number(g2);
492     draw_symbol(2, 32, CS2);
493   }
494   //time
495   volatile uint8_t min;
496   min = seconds/60;
497   if(seconds > 59){
498     if(seconds > 599){
499       ctw = return_number(min/10);
500       draw_symbol(5, 44, CS1);
501       ctw = return_number(min%10);
502       draw_symbol(5, 52, CS1);
503       ctw = return_number((seconds-(min*60))/10);
504       draw_symbol(5, 4, CS2);
505       ctw = return_number((seconds-(min*60))%10);
506       draw_symbol(5, 12, CS2);
507     }else{
508       ctw = ZERO;
```

```
509       draw_symbol(5, 44, CS1);
510       ctw = return_number(min);
511       draw_symbol(5, 52,  CS1);
512       ctw = return_number((seconds-(min*60))/10);
513       draw_symbol(5, 4, CS2);
514       ctw = return_number((seconds-(min*60))%10);
515       draw_symbol(5, 12, CS2);
516     }
517   } else {
518       ctw = ZERO;
519       draw_symbol(5, 44, CS1);
520       ctw = ZERO;
521       draw_symbol(5, 52, CS1);
522       ctw = return_number(seconds/10);
523       draw_symbol(5, 4, CS2);
524       ctw = return_number(seconds%10);
525       draw_symbol(5, 12, CS2);
526   }
527   //period
528   ctw = return_number(period);
529   draw_symbol(2, 0, CS2);
530 }
531
532 /*
533 * shows time up and game over
534 */
535 void show_end(){
536   flag = 1;
537   if(period == 3){
538     ctw = G;
539     draw_symbol(7, 30, CS1);
540     ctw = A;
541     draw_symbol(7, 38, CS1);
542     ctw = M;
543     draw_symbol(7, 46, CS1);
544     ctw = E;
545     draw_symbol(7, 54, CS1);
546     ctw = O;
547     draw_symbol(7, 2, CS2);
548     ctw = V;
549     draw_symbol(7, 10, CS2);
550     ctw = E;
551     draw_symbol(7, 18, CS2);
552     ctw = R;
553     draw_symbol(7, 26, CS2);
554   }else{
555     ctw = T;
556     draw_symbol(7, 38, CS1);
557     ctw = I;
558     draw_symbol(7, 46, CS1);
559     ctw = M;
560     draw_symbol(7, 54, CS1);
561     ctw = E;
562     draw_symbol(7, 0, CS2);
563     ctw = U;
564     draw_symbol(7, 12, CS2);
565     ctw = P;
566     draw_symbol(7, 20, CS2);
567   }
568 }
569
570 /*
571 * Scrolls goal over the screen
572 */
573 void show_goal(int team){
574   uint8_t i;
575   for(i = 1; i < 14; i++){
576     _delay_ms (2000); //speed of the scrolling text
577     if( i < 9){
578       ctw = L;
579       draw_symbol(7, (i*8)-8, team);
580     }
581     if(i > 1 && i < 10){
582       ctw = A;
```

```
583       draw_symbol(7, (i*8)-16,team);
584     }
585     if(i > 2 && i < 11){
586       ctw = 0;
587       draw_symbol(7, (i*8)-24,team);
588     }
589     if(i > 3 && i < 12){
590       ctw = G;
591       draw_symbol(7, (i*8)-32,team);
592     }
593     if(i > 4 && i < 13){
594       ctw = EMPTY;
595       draw_symbol(7, (i*8)-40,team);
596     }
597   }
598 }
599
600 /*
601 * Plays sound
602 */
603 void play_goal_sound(){
604 PlayMusic( goal,8);
605 }
606
607 /*
608 * Plays sound
609 */
610 void play_end_sound(){
611 PlayMusic( starwars, 17 );
612 }
613
614 /*
615 * Function for stopping the timer. Mainly used when there's a goal or a button press
616 */
617 void timer_stop(){
618   TIMSK &= ~(1<<TOIE0); //disable timer
619   timer_running = 0;
620 }
621
622 /*
623 * Interrupt routine for the buttons.
624 */
625 ISR(INT0_vect){ //button interrupt
626   PORTD|=(1<<PD1);
627   button_handler(PINB);
628   PORTD&=~(1<<PD1);
629 }
630 /*
631 * Interrupt routine for the sensors. Not used anymore and should be deleted.
632 */
633 ISR(INT1_vect){ //goal interrupt
634   TIMSK &= ~(1<<TOIE0); //disable timer
635   timer_running = 0;
636   if((PIND & (1 << PD5)) == 0){
637     g1++;
638       goal1 = 1;
639   }else if(PIND & (1 << PD6)){
640     g2++;
641       goal2 = 1;
642   }
643 }
644
645 /*
646 * Timer interrupt routine
647 */
648 ISR(TIMER0_OVF_vect){ //timer interrupt
649   counter++;
650 }
651
652 /*
653 * Initiate the PWM system
654 */
655 void pwm_init(){
656   // Configure OC1B pin as output
```

```
657
658   DDRD |= _BV(DDD4); //OC1B as output
659
660   // timer1 configuration (for PWM)
661     TCCR1A |= _BV(COM1B1);  // Clear OC1A/OC1B on compare match
662
663
664   TCCR1B |= _BV(WGM13)  //mode 8, PWM, Phase and Frequency Correct (TOP value is ICR1)
665         |_BV(CS11);    //prescaler(8)
666 }
667
668
669
670 int main(void){
671
672
673
674
675   DDRA = 0xFF; //set all pins for output
676   DDRB = 0x00; //set all pins as input
677   DDRC = 0b11000011; //set 0,1,6,7 as output, rest as input
678   DDRD = 0b10010011; //set different in/out
679
680   PORTB = 0xFF; //enable pullup
681
682   PORTD|=(1<<PD2);
683   PORTD&= ~(1<<PD3); //&= ~
684   PORTD&= ~(1<<PD5);
685   PORTD&= ~(1<<PD6);
686
687   GICR |= 1<<INT0;
688   GICR |= 1<<INT1;
689
690   MCUCR |= 0x0F;  // set trigger for INT0 and INT1
691
692   volume = 100;
693   pwm_init();
694   seconds = 0;
695   g1 = 0;
696   g2 = 0;
697   counter = 0;
698   period = 1;
699   timer_running = 0;
700   goal1 = 0;
701   goal2 = 0;
702   gdebounce = 0;
703   flag = 0;
704   draw_lcd();
705
706 //prescaler for timer
707   TCCR0 |= (1<<CS02) | (1<<CS00);
708
709   TCNT0 = 0; //start value of counter
710
711   sei();
712   while(1){
713     if(goal1){
714
715       play_goal_sound();
716
717       show_goal(1);
718
719       goal1 = 0;
720     }
721     if(goal2){
722
723       play_goal_sound();
724
725       show_goal(2);
726
727       goal2 = 0;
728     }
729     if(timer_running && flag){
730         ctw = EMPTY;
```

```
        uint8_t i;
        for(i = 1; i < 9; i++){
          draw_symbol(7, i*8, CS3);
        }
        flag = 0;
  }

    if(counter >= 61){
      counter = 0;
      if(seconds <= 0){
        TIMSK &= ~(1<<TOIE0); //disable timer
        timer_running = 0;
        gdebounce = 0;
        show_end();
        play_end_sound();
      } else{
        seconds--;
      }
    }
    cli();
    update_lcd();
    sei();
  }

}
```

text.h

```
#include <avr/pgmspace.h>
const uint8_t fs[52][8] PROGMEM ={
{0x38, 0x7C, 0xE6, 0xE6, 0xFE, 0xE6, 0xE6, 0x00},// Char 001 (A)
{0xFC, 0x76, 0x76, 0x7C, 0x76, 0x76, 0xFC, 0x00},// Char 002 (B)
{0x7C, 0xE6, 0xE0, 0xE0, 0xE0, 0xE6, 0x7C, 0x00},// Char 003 (C)
{0xF8, 0x74, 0x72, 0x72, 0x72, 0x74, 0xF8, 0x00},// Char 004 (D)
{0xFE, 0x72, 0x70, 0x78, 0x70, 0x72, 0xFE, 0x00},// Char 005 (E)
{0xFE, 0x72, 0x70, 0x78, 0x70, 0x70, 0xF8, 0x00},// Char 006 (F)
{0x7C, 0xE6, 0xE0, 0xEE, 0xE6, 0xE6, 0x7C, 0x00},// Char 007 (G)
{0xE6, 0xE6, 0xE6, 0xFE, 0xE6, 0xE6, 0xE6, 0x00},// Char 008 (H)
{0x7C, 0x38, 0x38, 0x38, 0x38, 0x38, 0x7C, 0x00},// Char 009 (I)
{0x1E, 0x0E, 0x0E, 0x0E, 0x0E, 0xCE, 0x7C, 0x00},// Char 010 (J)
{0xE6, 0xE6, 0xEC, 0xF8, 0xEC, 0xE6, 0xE6, 0x00},// Char 011 (K)
{0xF0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE2, 0xFE, 0x00},// Char 012 (L)
{0xC6, 0xEE, 0xFE, 0xD6, 0xC6, 0xC6, 0xC6, 0x00},// Char 013 (M)
{0xE6, 0xF6, 0xFE, 0xEE, 0xE6, 0xE6, 0xE6, 0x00},// Char 014 (N)
{0x7C, 0xE6, 0xE6, 0xE6, 0xE6, 0xE6, 0x7C, 0x00},// Char 015 (O)
{0xFC, 0x76, 0x76, 0x76, 0x7C, 0x70, 0xF8, 0x00},// Char 016 (P)
{0x7C, 0xE6, 0xE6, 0xE6, 0xE6, 0xF6, 0x7C, 0x0C},// Char 017 (Q)
{0xFC, 0x72, 0x72, 0x72, 0x7C, 0x76, 0xF6, 0x00},// Char 018 (R)
{0x7C, 0xEE, 0xE0, 0x7C, 0x06, 0xEE, 0x7C, 0x00},// Char 019 (S)
{0xFE, 0xFE, 0xBA, 0x38, 0x38, 0x38, 0x7C, 0x00},// Char 020 (T)
{0xE6, 0xE6, 0xE6, 0xE6, 0xE6, 0xE6, 0x7C, 0x00},// Char 021 (U)
{0xE6, 0xE6, 0xE6, 0xE6, 0x66, 0x3C, 0x18, 0x00},// Char 022 (V)
{0xC6, 0xC6, 0xC6, 0xD6, 0xFE, 0xEE, 0xC6, 0x00},// Char 023 (W)
{0xE6, 0xE6, 0xE6, 0x7C, 0x7C, 0xE6, 0xE6, 0x00},// Char 024 (X)
{0xEE, 0xE6, 0xE6, 0x7E, 0x06, 0xE6, 0x3C, 0x00},// Char 025 (Y)
{0xFE, 0xFC, 0x18, 0x30, 0x60, 0xFE, 0xFE, 0x00},// Char 026 (Z)
{0x18, 0x3C, 0x3C, 0x3C, 0x18, 0x00, 0x18, 0x00},// Char 027 (!)
{0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00},// Char 028 (")
{0x6C, 0x6C, 0xFE, 0x6C, 0xFE, 0x6C, 0x6C, 0x00},// Char 029 (#)
{0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x30, 0x00},// Char 030 (,)
{0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00},// Char 031 (.)
{0x7C, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0x7C, 0x00},// Char 032 (0)
{0x3C, 0x7C, 0x3C, 0x3C, 0x3C, 0x3C, 0x3C, 0x00},// Char 033 (1)
{0x7C, 0xCE, 0x0E, 0x1C, 0x38, 0x7E, 0xFE, 0x00},// Char 034 (2)
{0x7C, 0xCE, 0x0E, 0x1C, 0x0E, 0xCE, 0x7C, 0x00},// Char 035 (3)
{0x1C, 0x3C, 0x6C, 0xCC, 0xFE, 0xFE, 0x0C, 0x00},// Char 036 (4)
{0xFE, 0xE2, 0xE0, 0xFC, 0x06, 0xC6, 0x7C, 0x00},// Char 037 (5)
{0x3C, 0x66, 0xE0, 0xFC, 0xE6, 0x66, 0x3C, 0x00},// Char 038 (6)
{0xFE, 0xFE, 0x0E, 0x0E, 0x1C, 0x1C, 0x1C, 0x00},// Char 039 (7)
{0x7C, 0xEE, 0xC6, 0x7C, 0xC6, 0xEE, 0x7C, 0x00},// Char 040 (8)
{0x7C, 0xC6, 0xC6, 0x7E, 0x0E, 0xCE, 0x7C, 0x00},// Char 041 (9)
{0x00, 0x00, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00},// Char 042 (:)
{0x0E, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0E, 0x00},// Char 043 (<)
```

```
{0x00, 0x00, 0x7E, 0x00, 0x7E, 0x00, 0x00, 0x00},// Char 044 (=)
{0x70, 0x18, 0x0C, 0x06, 0x0C, 0x18, 0x70, 0x00},// Char 045 (>)
{0x3C, 0x66, 0x06, 0x0C, 0x18, 0x00, 0x18, 0x00},// Char 046 (?)
{0x00, 0x00, 0x10, 0x10, 0x00, 0x10, 0x10, 0x00},// Char 047 halv :1
{0x70, 0x30, 0x30, 0x30, 0x30, 0x30, 0x70, 0x00},// Char 048 halv I1
{0x0C, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0C, 0x00},// Char 049 halv I2
{0x00, 0x00, 0x10, 0x10, 0x00, 0x10, 0x10, 0x00},// Char 050 halv :2
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, //Char 051 EMPTY
{0x04, 0xDD, 0xFD, 0xFD, 0xFD, 0xDD, 0x04, 0x00} //Char 052 speaker
};
```

notes.h

```
/*
ICR values for AVR micro controllers to play sound using PWM.
System clock:   8000000 [Hz]
Prescaler:      8
PWM mode:       phase/frequency correct 16 bit
*/

#ifndef __PWM_NOTES_H__
#define __PWM_NOTES_H__

/* Pause */

#define P1 1
#define PAUSE P1
#define p P1

/* end of notes table */
#define MUSIC_END 0

/* There are 12 notes in octave + variations (lower case, other names)  */

/* Octave #2 */

#define A2   4545   /* PWM: 110.01 Hz, note freq: 110.00 Hz, error 0.01% */
#define a2   4545   /* PWM: 110.01 Hz, note freq: 110.00 Hz, error 0.01% */
#define A2b  4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define a2b  4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define A2x  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define a2x  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define Ais2 4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define ais2 4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define B2   4050   /* PWM: 123.46 Hz, note freq: 123.47 Hz, error 0.01% */
#define b2   4050   /* PWM: 123.46 Hz, note freq: 123.47 Hz, error 0.01% */
#define B2b  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define b2b  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define C2   7645   /* PWM: 65.40 Hz, note freq: 65.41 Hz, error 0.01% */
#define c2   7645   /* PWM: 65.40 Hz, note freq: 65.41 Hz, error 0.01% */
#define C2x  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define c2x  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define Cis2 7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define cis2 7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define D2   6810   /* PWM: 73.42 Hz, note freq: 73.42 Hz, error 0.01% */
#define d2   6810   /* PWM: 73.42 Hz, note freq: 73.42 Hz, error 0.01% */
#define D2b  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define d2b  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define D2x  6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define d2x  6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define Dis2 6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define dis2 6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define E2   6067   /* PWM: 82.41 Hz, note freq: 82.41 Hz, error 0.01% */
#define e2   6067   /* PWM: 82.41 Hz, note freq: 82.41 Hz, error 0.01% */
#define E2b  6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define e2b  6428   /* PWM: 77.78 Hz, note freq: 77.78 Hz, error 0.00% */
#define F2   5727   /* PWM: 87.31 Hz, note freq: 87.31 Hz, error 0.00% */
#define f2   5727   /* PWM: 87.31 Hz, note freq: 87.31 Hz, error 0.00% */
#define F2x  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define f2x  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define Fis2 5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define fis2 5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define G2   5102   /* PWM: 98.00 Hz, note freq: 98.00 Hz, error 0.00% */
```

```
#define g22   5102  /* PWM: 98.00 Hz, note freq: 98.00 Hz, error 0.00% */
#define G2b  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define g2b  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define G2x  4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define g2x  4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define Gis2 4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define gis2 4816   /* PWM: 103.82 Hz, note freq: 103.83 Hz, error 0.01% */
#define H2   4050   /* PWM: 123.46 Hz, note freq: 123.47 Hz, error 0.01% */
#define h2   4050   /* PWM: 123.46 Hz, note freq: 123.47 Hz, error 0.01% */
#define H2b  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define h2b  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define bH2  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define bh2  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define xA2  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define xa2  4290   /* PWM: 116.55 Hz, note freq: 116.54 Hz, error 0.01% */
#define xC2  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define xc2  7215   /* PWM: 69.30 Hz, note freq: 69.30 Hz, error 0.01% */
#define xF2  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define xf2  5405   /* PWM: 92.51 Hz, note freq: 92.50 Hz, error 0.01% */
#define xG2  5102   /* PWM: 98.00 Hz, note freq: 98.00 Hz, error 0.00% */
#define xg2  5102   /* PWM: 98.00 Hz, note freq: 98.00 Hz, error 0.00% */

/* Octave #3 */

#define A3   2273   /* PWM: 219.97 Hz, note freq: 220.00 Hz, error 0.01% */
#define a3   2273   /* PWM: 219.97 Hz, note freq: 220.00 Hz, error 0.01% */
#define A3b  2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define a3b  2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define A3x  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define a3x  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define Ais3 2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define ais3 2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define B3   2025   /* PWM: 246.91 Hz, note freq: 246.94 Hz, error 0.01% */
#define b3   2025   /* PWM: 246.91 Hz, note freq: 246.94 Hz, error 0.01% */
#define B3b  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define b3b  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define C3   3822   /* PWM: 130.82 Hz, note freq: 130.81 Hz, error 0.01% */
#define c3   3822   /* PWM: 130.82 Hz, note freq: 130.81 Hz, error 0.01% */
#define C3x  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define c3x  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define Cis3 3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define cis3 3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define D3   3405   /* PWM: 146.84 Hz, note freq: 146.83 Hz, error 0.01% */
#define d3   3405   /* PWM: 146.84 Hz, note freq: 146.83 Hz, error 0.01% */
#define D3b  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define d3b  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define D3x  3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define d3x  3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define Dis3 3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define dis3 3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define E3   3034   /* PWM: 164.80 Hz, note freq: 164.81 Hz, error 0.01% */
#define e3   3034   /* PWM: 164.80 Hz, note freq: 164.81 Hz, error 0.01% */
#define E3b  3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define e3b  3214   /* PWM: 155.57 Hz, note freq: 155.56 Hz, error 0.00% */
#define F3   2863   /* PWM: 174.64 Hz, note freq: 174.61 Hz, error 0.02% */
#define f3   2863   /* PWM: 174.64 Hz, note freq: 174.61 Hz, error 0.02% */
#define F3x  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define f3x  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define Fis3 2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define fis3 2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define G3   2551   /* PWM: 196.00 Hz, note freq: 196.00 Hz, error 0.00% */
#define g3   2551   /* PWM: 196.00 Hz, note freq: 196.00 Hz, error 0.00% */
#define G3b  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define g3b  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define G3x  2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define g3x  2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define Gis3 2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define gis3 2408   /* PWM: 207.64 Hz, note freq: 207.65 Hz, error 0.01% */
#define H3   2025   /* PWM: 246.91 Hz, note freq: 246.94 Hz, error 0.01% */
#define h3   2025   /* PWM: 246.91 Hz, note freq: 246.94 Hz, error 0.01% */
#define H3b  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define h3b  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define bH3  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define bh3  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
```

```
#define xA3  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define xa3  2145   /* PWM: 233.10 Hz, note freq: 233.08 Hz, error 0.01% */
#define xC3  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define xc3  3608   /* PWM: 138.58 Hz, note freq: 138.59 Hz, error 0.01% */
#define xF3  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define xf3  2703   /* PWM: 184.98 Hz, note freq: 185.00 Hz, error 0.01% */
#define xG3  2551   /* PWM: 196.00 Hz, note freq: 196.00 Hz, error 0.00% */
#define xg3  2551   /* PWM: 196.00 Hz, note freq: 196.00 Hz, error 0.00% */

/* Octave #4 */

#define A4   1136   /* PWM: 440.14 Hz, note freq: 440.00 Hz, error 0.03% */
#define a4   1136   /* PWM: 440.14 Hz, note freq: 440.00 Hz, error 0.03% */
#define A4b  1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define a4b  1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define A4x  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define a4x  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define Ais4 1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define ais4 1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define B4   1012   /* PWM: 494.07 Hz, note freq: 493.88 Hz, error 0.04% */
#define b4   1012   /* PWM: 494.07 Hz, note freq: 493.88 Hz, error 0.04% */
#define B4b  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define b4b  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define C4   1911   /* PWM: 261.64 Hz, note freq: 261.63 Hz, error 0.01% */
#define c4   1911   /* PWM: 261.64 Hz, note freq: 261.63 Hz, error 0.01% */
#define C4x  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define c4x  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define Cis4 1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define cis4 1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define D4   1703   /* PWM: 293.60 Hz, note freq: 293.66 Hz, error 0.02% */
#define d4   1703   /* PWM: 293.60 Hz, note freq: 293.66 Hz, error 0.02% */
#define D4b  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define d4b  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define D4x  1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define d4x  1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define Dis4 1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define dis4 1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define E4   1517   /* PWM: 329.60 Hz, note freq: 329.63 Hz, error 0.01% */
#define e4   1517   /* PWM: 329.60 Hz, note freq: 329.63 Hz, error 0.01% */
#define E4b  1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define e4b  1607   /* PWM: 311.14 Hz, note freq: 311.13 Hz, error 0.00% */
#define F4   1432   /* PWM: 349.16 Hz, note freq: 349.23 Hz, error 0.02% */
#define f4   1432   /* PWM: 349.16 Hz, note freq: 349.23 Hz, error 0.02% */
#define F4x  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define f4x  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define Fis4 1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define fis4 1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define G4   1276   /* PWM: 391.85 Hz, note freq: 392.00 Hz, error 0.04% */
#define g4   1276   /* PWM: 391.85 Hz, note freq: 392.00 Hz, error 0.04% */
#define G4b  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define g4b  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define G4x  1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define g4x  1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define Gis4 1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define gis4 1204   /* PWM: 415.28 Hz, note freq: 415.30 Hz, error 0.01% */
#define H4   1012   /* PWM: 494.07 Hz, note freq: 493.88 Hz, error 0.04% */
#define h4   1012   /* PWM: 494.07 Hz, note freq: 493.88 Hz, error 0.04% */
#define H4b  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define h4b  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define bH4  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define bh4  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define xA4  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define xa4  1073   /* PWM: 465.98 Hz, note freq: 466.16 Hz, error 0.04% */
#define xC4  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define xc4  1804   /* PWM: 277.16 Hz, note freq: 277.18 Hz, error 0.01% */
#define xF4  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define xf4  1351   /* PWM: 370.10 Hz, note freq: 369.99 Hz, error 0.03% */
#define xG4  1276   /* PWM: 391.85 Hz, note freq: 392.00 Hz, error 0.04% */
#define xg4  1276   /* PWM: 391.85 Hz, note freq: 392.00 Hz, error 0.04% */

/* Octave #5 */

#define A5   568    /* PWM: 880.28 Hz, note freq: 880.00 Hz, error 0.03% */
#define a5   568    /* PWM: 880.28 Hz, note freq: 880.00 Hz, error 0.03% */
```

```
209 #define A5b  602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
210 #define a5b  602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
211 #define A5x  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
212 #define a5x  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
213 #define Ais5 536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
214 #define ais5 536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
215 #define B5   506    /* PWM: 988.14 Hz, note freq: 987.77 Hz, error 0.04% */
216 #define b5   506    /* PWM: 988.14 Hz, note freq: 987.77 Hz, error 0.04% */
217 #define B5b  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
218 #define b5b  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
219 #define C5   956    /* PWM: 523.01 Hz, note freq: 523.25 Hz, error 0.05% */
220 #define c5   956    /* PWM: 523.01 Hz, note freq: 523.25 Hz, error 0.05% */
221 #define C5x  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
222 #define c5x  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
223 #define Cis5 902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
224 #define cis5 902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
225 #define D5   851    /* PWM: 587.54 Hz, note freq: 587.33 Hz, error 0.04% */
226 #define d5   851    /* PWM: 587.54 Hz, note freq: 587.33 Hz, error 0.04% */
227 #define D5b  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
228 #define d5b  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
229 #define D5x  804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
230 #define d5x  804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
231 #define Dis5 804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
232 #define dis5 804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
233 #define E5   758    /* PWM: 659.63 Hz, note freq: 659.26 Hz, error 0.06% */
234 #define e5   758    /* PWM: 659.63 Hz, note freq: 659.26 Hz, error 0.06% */
235 #define E5b  804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
236 #define e5b  804    /* PWM: 621.89 Hz, note freq: 622.25 Hz, error 0.06% */
237 #define F5   716    /* PWM: 698.32 Hz, note freq: 698.46 Hz, error 0.02% */
238 #define f5   716    /* PWM: 698.32 Hz, note freq: 698.46 Hz, error 0.02% */
239 #define F5x  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
240 #define f5x  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
241 #define Fis5 676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
242 #define fis5 676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
243 #define G5   638    /* PWM: 783.70 Hz, note freq: 783.99 Hz, error 0.04% */
244 #define g5   638    /* PWM: 783.70 Hz, note freq: 783.99 Hz, error 0.04% */
245 #define G5b  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
246 #define g5b  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
247 #define G5x  602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
248 #define g5x  602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
249 #define Gis5 602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
250 #define gis5 602    /* PWM: 830.56 Hz, note freq: 830.61 Hz, error 0.01% */
251 #define H5   506    /* PWM: 988.14 Hz, note freq: 987.77 Hz, error 0.04% */
252 #define h5   506    /* PWM: 988.14 Hz, note freq: 987.77 Hz, error 0.04% */
253 #define H5b  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
254 #define h5b  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
255 #define bH5  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
256 #define bh5  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
257 #define xA5  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
258 #define xa5  536    /* PWM: 932.84 Hz, note freq: 932.33 Hz, error 0.05% */
259 #define xC5  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
260 #define xc5  902    /* PWM: 554.32 Hz, note freq: 554.37 Hz, error 0.01% */
261 #define xF5  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
262 #define xf5  676    /* PWM: 739.64 Hz, note freq: 739.99 Hz, error 0.05% */
263 #define xG5  638    /* PWM: 783.70 Hz, note freq: 783.99 Hz, error 0.04% */
264 #define xg5  638    /* PWM: 783.70 Hz, note freq: 783.99 Hz, error 0.04% */
265
266 /* Octave #6 */
267
268 #define A6   284    /* PWM: 1760.56 Hz, note freq: 1760.00 Hz, error 0.03% */
269 #define a6   284    /* PWM: 1760.56 Hz, note freq: 1760.00 Hz, error 0.03% */
270 #define A6b  301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
271 #define a6b  301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
272 #define A6x  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
273 #define a6x  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
274 #define Ais6 268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
275 #define ais6 268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
276 #define B6   253    /* PWM: 1976.28 Hz, note freq: 1975.53 Hz, error 0.04% */
277 #define b6   253    /* PWM: 1976.28 Hz, note freq: 1975.53 Hz, error 0.04% */
278 #define B6b  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
279 #define b6b  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
280 #define C6   478    /* PWM: 1046.03 Hz, note freq: 1046.50 Hz, error 0.05% */
281 #define c6   478    /* PWM: 1046.03 Hz, note freq: 1046.50 Hz, error 0.05% */
282 #define C6x  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
```

```
#define c6x  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define Cis6 451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define cis6 451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define D6   426    /* PWM: 1173.71 Hz, note freq: 1174.66 Hz, error 0.08% */
#define d6   426    /* PWM: 1173.71 Hz, note freq: 1174.66 Hz, error 0.08% */
#define D6b  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define d6b  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define D6x  402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define d6x  402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define Dis6 402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define dis6 402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define E6   379    /* PWM: 1319.26 Hz, note freq: 1318.51 Hz, error 0.06% */
#define e6   379    /* PWM: 1319.26 Hz, note freq: 1318.51 Hz, error 0.06% */
#define E6b  402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define e6b  402    /* PWM: 1243.78 Hz, note freq: 1244.51 Hz, error 0.06% */
#define F6   358    /* PWM: 1396.65 Hz, note freq: 1396.91 Hz, error 0.02% */
#define f6   358    /* PWM: 1396.65 Hz, note freq: 1396.91 Hz, error 0.02% */
#define F6x  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define f6x  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define Fis6 338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define fis6 338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define G6   319    /* PWM: 1567.40 Hz, note freq: 1567.98 Hz, error 0.04% */
#define g6   319    /* PWM: 1567.40 Hz, note freq: 1567.98 Hz, error 0.04% */
#define G6b  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define g6b  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define G6x  301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
#define g6x  301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
#define Gis6 301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
#define gis6 301    /* PWM: 1661.13 Hz, note freq: 1661.22 Hz, error 0.01% */
#define H6   253    /* PWM: 1976.28 Hz, note freq: 1975.53 Hz, error 0.04% */
#define h6   253    /* PWM: 1976.28 Hz, note freq: 1975.53 Hz, error 0.04% */
#define H6b  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define h6b  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define bH6  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define bh6  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define xA6  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define xa6  268    /* PWM: 1865.67 Hz, note freq: 1864.66 Hz, error 0.05% */
#define xC6  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define xc6  451    /* PWM: 1108.65 Hz, note freq: 1108.73 Hz, error 0.01% */
#define xF6  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define xf6  338    /* PWM: 1479.29 Hz, note freq: 1479.98 Hz, error 0.05% */
#define xG6  319    /* PWM: 1567.40 Hz, note freq: 1567.98 Hz, error 0.04% */
#define xg6  319    /* PWM: 1567.40 Hz, note freq: 1567.98 Hz, error 0.04% */

/* Octave #7 */

#define A7   142    /* PWM: 3521.13 Hz, note freq: 3520.00 Hz, error 0.03% */
#define a7   142    /* PWM: 3521.13 Hz, note freq: 3520.00 Hz, error 0.03% */
#define A7b  150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define a7b  150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define A7x  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define a7x  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define Ais7 134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define ais7 134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define B7   127    /* PWM: 3937.01 Hz, note freq: 3951.07 Hz, error 0.36% */
#define b7   127    /* PWM: 3937.01 Hz, note freq: 3951.07 Hz, error 0.36% */
#define B7b  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define b7b  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define C7   239    /* PWM: 2092.05 Hz, note freq: 2093.00 Hz, error 0.05% */
#define c7   239    /* PWM: 2092.05 Hz, note freq: 2093.00 Hz, error 0.05% */
#define C7x  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define c7x  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define Cis7 225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define cis7 225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define D7   213    /* PWM: 2347.42 Hz, note freq: 2349.32 Hz, error 0.08% */
#define d7   213    /* PWM: 2347.42 Hz, note freq: 2349.32 Hz, error 0.08% */
#define D7b  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define d7b  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define D7x  201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define d7x  201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define Dis7 201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define dis7 201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define E7   190    /* PWM: 2631.58 Hz, note freq: 2637.02 Hz, error 0.21% */
#define e7   190    /* PWM: 2631.58 Hz, note freq: 2637.02 Hz, error 0.21% */
```

```
#define E7b  201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define e7b  201    /* PWM: 2487.56 Hz, note freq: 2489.02 Hz, error 0.06% */
#define F7   179    /* PWM: 2793.30 Hz, note freq: 2793.83 Hz, error 0.02% */
#define f7   179    /* PWM: 2793.30 Hz, note freq: 2793.83 Hz, error 0.02% */
#define F7x  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define f7x  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define Fis7 169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define fis7 169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define G7   159    /* PWM: 3144.65 Hz, note freq: 3135.96 Hz, error 0.28% */
#define g7   159    /* PWM: 3144.65 Hz, note freq: 3135.96 Hz, error 0.28% */
#define G7b  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define g7b  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define G7x  150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define g7x  150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define Gis7 150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define gis7 150    /* PWM: 3333.33 Hz, note freq: 3322.44 Hz, error 0.33% */
#define H7   127    /* PWM: 3937.01 Hz, note freq: 3951.07 Hz, error 0.36% */
#define h7   127    /* PWM: 3937.01 Hz, note freq: 3951.07 Hz, error 0.36% */
#define H7b  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define h7b  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define bH7  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define bh7  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define xA7  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define xa7  134    /* PWM: 3731.34 Hz, note freq: 3729.31 Hz, error 0.05% */
#define xC7  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define xc7  225    /* PWM: 2222.22 Hz, note freq: 2217.46 Hz, error 0.21% */
#define xF7  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define xf7  169    /* PWM: 2958.58 Hz, note freq: 2959.96 Hz, error 0.05% */
#define xG7  159    /* PWM: 3144.65 Hz, note freq: 3135.96 Hz, error 0.28% */
#define xg7  159    /* PWM: 3144.65 Hz, note freq: 3135.96 Hz, error 0.28% */

#endif /* __PWM_NOTES_H__ */
```