Университет ИТМО

Факультет программной инженерии
и компьютерной техники

Кафедра вычислительной техники

Информационно-управляющие системы
Лабораторная работа №2
Вариант №8

Выполнили:
Милосердов А. О.
Калугин Ф. И.
Группа P3410

Преподаватель:
Ключев А. О.

Санкт-Петербург
2017 г.

Описание задания

Разработать и реализовать драйвер системного таймера микроконтроллера и драйвер звукового излучателя, позволяющий задавать частоту для ADuC812. Написать тестовую программу с использованием разработанного драйвера по алгоритму: контроллер SDK-1.1 циклически проигрывает нисходящую гамму нот второй октавы (длительность каждой ноты – 0,5 секунды) и на линейку светодиодов выводит количество замыканий входа INT1. В результате выполнения работы должны быть разработаны драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.

Модель взаимодействия

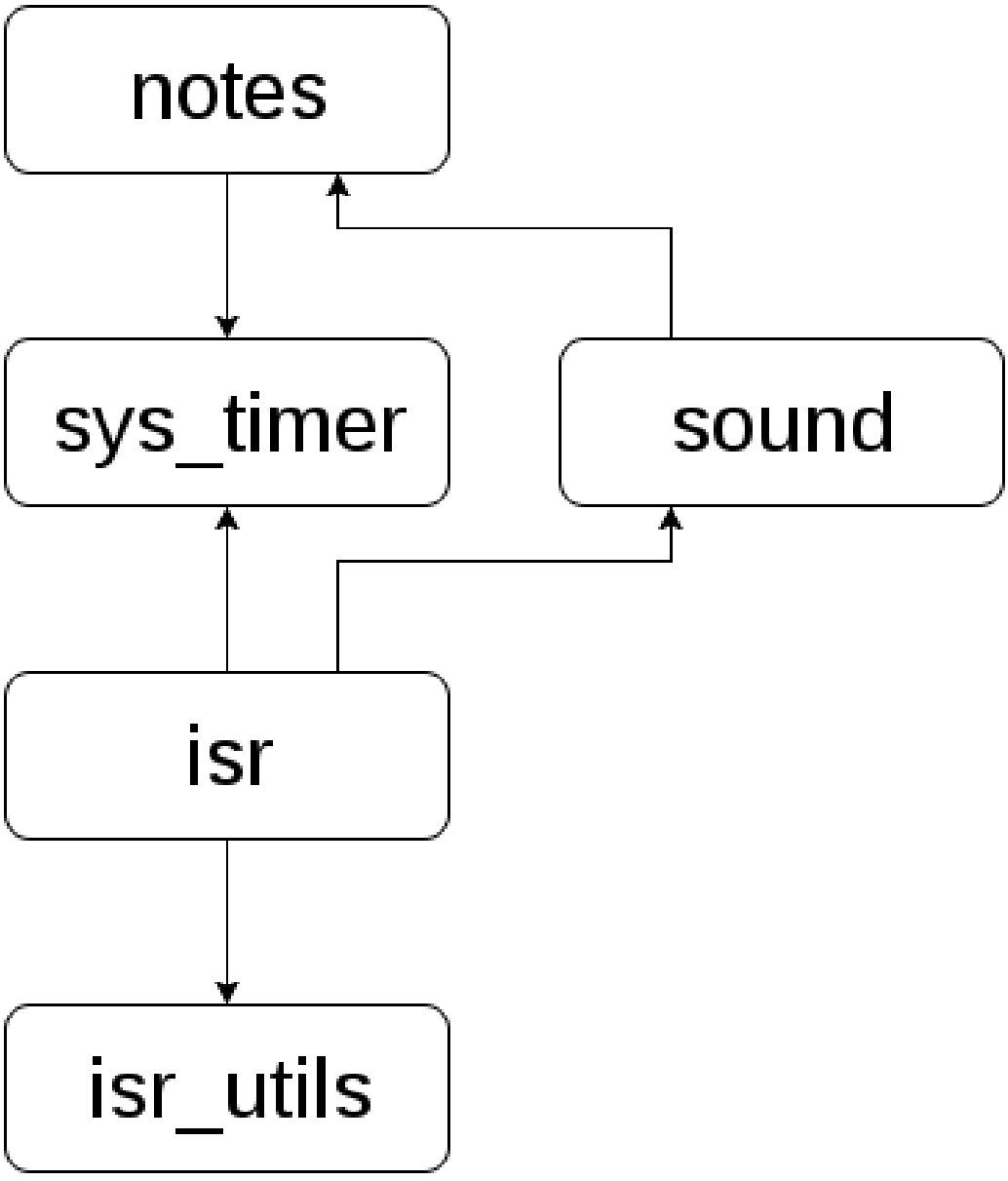

Рис. 1: Модель взаимодействия модулей

Листинги

isr.h

```
#pragma once

// Set interrupts vectors
void set_vectors( );
```

isr.c

```
#include "isr_utils.h"
#include "isr.h"
#include "sys_timer.h"
#include "aduc812.h"
#include "max.h"
#include "sound.h"
#include "led.h"

// INT1 counter
unsigned char k;

// Toggler for ENA in T0 ISR
char t;

char g_note_h;
char g_note_l;

unsigned long systime = 0;

// Handler for INT1 interrupt, controls LED output
void INT1_ISR( void ) __interrupt {
        k++;
        leds( k );
}

// Handler for timer1 interrupts, interrupts every millisecond
void T1_ISR ( void ) __interrupt {
  systime++;

  TH1 = MS_H;
  TL1 = MS_L;
}

// Handler for timer0 interrupts, generates sound of set frequency
void T0_ISR( void ) __interrupt {

  if( t ) {
    write_max(ENA, VOL1);
    t = 0;
  } else {
    write_max(ENA, VOL0);
    t = 1;
  }
  TH0 = g_note_h;
  TL0 = g_note_l;
}
```

```
47
48  void set_vectors( ) {
49    set_vector( 0x200B, (void *)T0_ISR );
50    set_vector( 0x2013, (void *)INT1_ISR );
51    set_vector( 0x201B, (void *)T1_ISR );
52  }
```

isr_utils.h

```
1   #pragma once
2
3   // Set user interrupts handlers
4   void set_vector( unsigned char __xdata * Address, void * Vector );
```

isr_utils.c

```
1   #include "isr_utils.h"
2
3   void set_vector( unsigned char __xdata * Address, void * Vector ) {
4     unsigned char __xdata * TmpVector;
5
6     *Address = 0x02;
7
8     TmpVector = (unsigned char __xdata *) (Address + 1);
9     *TmpVector = (unsigned char) ((unsigned short)Vector >> 8);
10    ++TmpVector;
11    *TmpVector = (unsigned char) Vector;
12
13  }
```

sound.h

```
1   #pragma once
2
3   // Volume levels
4   #define VOL0 0
5   #define VOL1 0b00011100
6
7   // Note frequency data
8   extern char g_note_h;
9   extern char g_note_l;
10
11  // Set sound frequency
12  void set_note( int hz );
```

sound.c

```
1   #include "sound.h"
2   #include "notes.h"
3
4   void set_note( int hz ) {
5     g_note_h = note_h( hz );
6     g_note_l = note_l( hz );
7   }
```

notes.h

```
#pragma once

#define C_NOTE_0        16.4
#define C_SHARP_NOTE_0  17.3
#define D_NOTE_0        18.4
#define D_SHARP_NOTE_0  19.5
#define E_NOTE_0        20.6
#define F_NOTE_0        21.8
#define F_SHARP_NOTE_0  23.1
#define G_NOTE_0        24.5
#define G_SHARP_NOTE_0  26
#define A_NOTE_0        27.5
#define A_SHARP_NOTE_0  29.1
#define B_NOTE_0        30.9

#define C_NOTE_3        130.81
#define C_SHARP_NOTE_3  138.59
#define D_NOTE_3        146.83
#define D_SHARP_NOTE_3  155.56
#define E_NOTE_3        164.81
#define F_NOTE_3        174.61
#define F_SHARP_NOTE_3         185.00
#define G_NOTE_3        196.00
#define G_SHARP_NOTE_3         207.65
#define A_NOTE_3               220.00
#define A_SHARP_NOTE_3         233.08
#define B_NOTE_3               246.94

#define C_NOTE_4        261.63
#define C_SHARP_NOTE_4  277.18
#define D_NOTE_4        293.66
#define D_SHARP_NOTE_4  311.13
#define E_NOTE_4        329.63
#define F_NOTE_4        349.23
#define F_SHARP_NOTE_4         369.99
#define G_NOTE_4        392.00
#define G_SHARP_NOTE_4         415.30
#define A_NOTE_4               440.00
#define A_SHARP_NOTE_4         466.16
#define B_NOTE_4               493.88

#define C_NOTE_5        523.25
#define C_SHARP_NOTE_5  554.37
#define D_NOTE_5        587.33
#define D_SHARP_NOTE_5  622.25
#define E_NOTE_5        659.25
#define F_NOTE_5        698.46
#define F_SHARP_NOTE_5         739.99
#define G_NOTE_5        783.99
#define G_SHARP_NOTE_5         830.61
#define A_NOTE_5               880
#define A_SHARP_NOTE_5         932.33
#define B_NOTE_5               987.77

```

```
short note( int hz );
char note_h( int hz );
char note_l( int hz );
```

notes.c

```
#include "notes.h"
#include "sys_timer.h"

short note( int hz ) {
  // hz is multiplied by 2 cause we handle a note in two half periods
  return 0xFFFF - MCLKIN / ( ( hz * 2 )  ) / 12;
}

char note_h( int hz ) {
  return ( note( hz ) >> 8 ) & 0xFF;
}

char note_l( int hz ) {
  return ( note( hz ) & 0xFF );
}
```

sys_timer.h

```
#pragma once

// For init set
#define A_H 0xFB
#define A_L 0x8B

// Define main clock speed
#define MCLKIN 11059200

// Calculate timer ticks to count a millisecond
#define MS 0xFFFF - ( MCLKIN / 12 ) / 1000

// Break ms into two 8 bit values
#define MS_H ( MS >> 8 ) & 0x00FF
#define MS_L MS & 0x00FF

// Time in ms since timer1 start
extern unsigned long systime;

// Get current systime
unsigned long get_ms_counter( void );

// Get time difference
unsigned long d_time_ms( unsigned long t2 );

// Set delay in ms
void delay_ms( unsigned long ms );

// Init timers
void init_timers( );
```

sys_timer.c

```
#include "sys_timer.h"
#include "notes.h"
#include "aduc812.h"

unsigned long get_ms_counter( void ) {
  unsigned long res;
  ET1 = 0;
  res = systime;
  ET1 = 1;
  return res;
}

unsigned long d_time_ms( unsigned long t2 ) {
  unsigned long t1 = get_ms_counter();
  return t1 - t2;
}

void delay_ms( unsigned long ms ) {
  unsigned long t1 = get_ms_counter();
  while ( 1 ) {
    if ( d_time_ms( t1 ) > ms ) break;
  }
}

void init_timers( ) {

  // Set T0 and T1 to timer mode
  TMOD = 0x11;

  // Set INT1 edge-sensitive
  IT1 = 1;

  // Timers priorities
  PT1 = 0;
  PT0 = 1;

  // Start values for T0
  TH0 = A_H;
  TL0 = A_L;

  // Start values for T1
  TH1 = MS_H;
  TL1 = MS_L;

  // Start timers T0 and T1
  TR1 = 1;
  TR0 = 1;

  // Enable timers T0 and T1
  ET1 = 1;
  ET0 = 1;

  // Enable external interrupt 1
  EX1 = 1;
```

```
55
56    // Allow interrupts
57    EA = 1;
58  }
```

main.c

```
1   #include "sys_timer.h"
2   #include "isr.h"
3   #include "sound.h"
4   #include "notes.h"
5   #include "MasterOfPuppets.h"
6   #define DELAY_MS 500
7
8   void main( void ) {
9     init_timers( );
10    set_vectors( );
11
12    while( 1 ) {
13
14      set_note( B_NOTE_4 );
15      delay_ms( DELAY_MS );
16
17      set_note( A_NOTE_4 );
18      delay_ms( DELAY_MS );
19
20      set_note( G_NOTE_4 );
21      delay_ms( DELAY_MS );
22
23      set_note( F_NOTE_4  );
24      delay_ms( DELAY_MS );
25
26      set_note( E_NOTE_4 );
27      delay_ms( DELAY_MS );
28
29      set_note( D_NOTE_4 );
30      delay_ms( DELAY_MS );
31
32      set_note( C_NOTE_4 );
33      delay_ms( DELAY_MS );
34
35    }
36  }
```

Вывод

В результате выполнения лабораторной работы были разработаны и реализованы драйвер системного таймера микроконтроллера и драйвер звукового излучателя, позволяющий задавать частоту для ADuC812.