

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 Р3410

*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

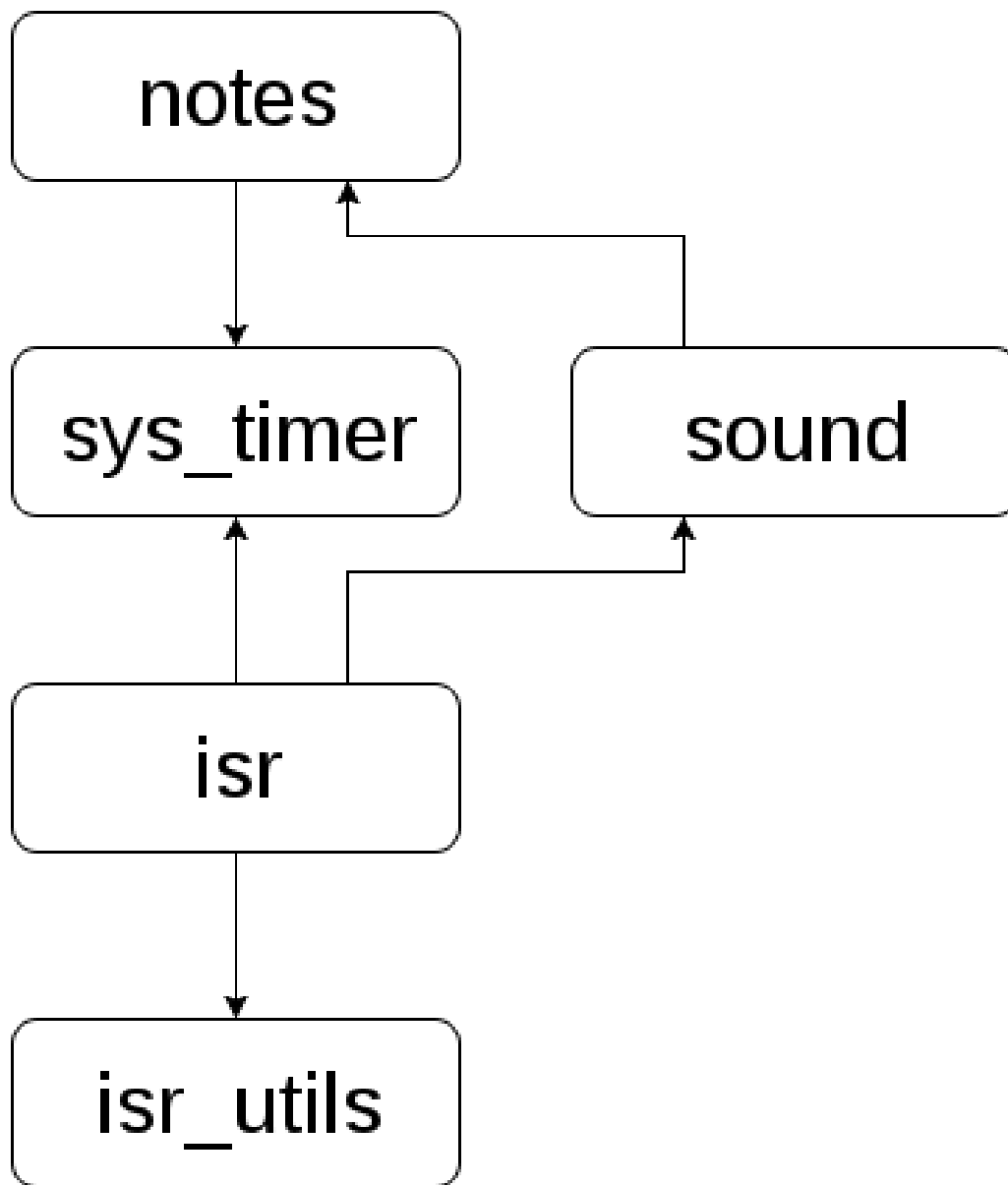


Рис. 1: Модель взаимодействия модулей

# Листинги

## isr.h

```
1  #pragma once
2
3  // Set interrupts vectors
4  void set_vectors( );
```

## isr.c

```
1  #include "isr_utils.h"
2  #include "isr.h"
3  #include "sys_timer.h"
4  #include "aduc812.h"
5  #include "max.h"
6  #include "sound.h"
7  #include "led.h"
8
9  // INT1 counter
10 unsigned char k;
11
12 // Toggler for ENA in T0 ISR
13 char t;
14
15 char g_note_h;
16 char g_note_l;
17
18 unsigned long systime = 0;
19
20 // Handler for INT1 interrupt, controls LED output
21 void INT1_ISR( void ) __interrupt {
22     k++;
23     leds( k );
24 }
25
26 // Handler for timer1 interrupts, interrupts every millisecond
27 void T1_ISR ( void ) __interrupt {
28     systime++;
29
30     TH1 = MS_H;
31     TL1 = MS_L;
32 }
33
34 // Handler for timer0 interrupts, generates sound of set frequency
35 void T0_ISR( void ) __interrupt {
36
37     if( t ) {
38         write_max(ENA, VOL1);
39         t = 0;
40     } else {
41         write_max(ENA, VOL0);
42         t = 1;
43     }
44     TH0 = g_note_h;
45     TL0 = g_note_l;
46 }
```

```

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

```
1  #pragma once
2
3  #define C_NOTE_0      16.4
4  #define C_SHARP_NOTE_0 17.3
5  #define D_NOTE_0      18.4
6  #define D_SHARP_NOTE_0 19.5
7  #define E_NOTE_0      20.6
8  #define F_NOTE_0      21.8
9  #define F_SHARP_NOTE_0 23.1
10 #define G_NOTE_0      24.5
11 #define G_SHARP_NOTE_0 26
12 #define A_NOTE_0      27.5
13 #define A_SHARP_NOTE_0 29.1
14 #define B_NOTE_0      30.9
15
16 #define C_NOTE_3      130.81
17 #define C_SHARP_NOTE_3 138.59
18 #define D_NOTE_3      146.83
19 #define D_SHARP_NOTE_3 155.56
20 #define E_NOTE_3      164.81
21 #define F_NOTE_3      174.61
22 #define F_SHARP_NOTE_3 185.00
23 #define G_NOTE_3      196.00
24 #define G_SHARP_NOTE_3 207.65
25 #define A_NOTE_3      220.00
26 #define A_SHARP_NOTE_3 233.08
27 #define B_NOTE_3      246.94
28
29 #define C_NOTE_4      261.63
30 #define C_SHARP_NOTE_4 277.18
31 #define D_NOTE_4      293.66
32 #define D_SHARP_NOTE_4 311.13
33 #define E_NOTE_4      329.63
34 #define F_NOTE_4      349.23
35 #define F_SHARP_NOTE_4 369.99
36 #define G_NOTE_4      392.00
37 #define G_SHARP_NOTE_4 415.30
38 #define A_NOTE_4      440.00
39 #define A_SHARP_NOTE_4 466.16
40 #define B_NOTE_4      493.88
41
42 #define C_NOTE_5      523.25
43 #define C_SHARP_NOTE_5 554.37
44 #define D_NOTE_5      587.33
45 #define D_SHARP_NOTE_5 622.25
46 #define E_NOTE_5      659.25
47 #define F_NOTE_5      698.46
48 #define F_SHARP_NOTE_5 739.99
49 #define G_NOTE_5      783.99
50 #define G_SHARP_NOTE_5 830.61
51 #define A_NOTE_5      880
52 #define A_SHARP_NOTE_5 932.33
53 #define B_NOTE_5      987.77
54
```

```

55 short note( int hz );
56 char note_h( int hz );
57 char note_l( int hz );

```

#### notes.c

```

1  #include "notes.h"
2  #include "sys_timer.h"
3
4  short note( int hz ) {
5      // hz is multiplied by 2 cause we handle a note in two half periods
6      return 0xFFFF - MCLKIN / ( ( hz * 2 ) ) / 12;
7  }
8
9  char note_h( int hz ) {
10     return ( note( hz ) >> 8 ) & 0xFF;
11 }
12
13 char note_l( int hz ) {
14     return ( note( hz ) & 0xFF );
15 }

```

#### sys\_timer.h

```

1  #pragma once
2
3  // For init set
4  #define A_H 0xFB
5  #define A_L 0x8B
6
7  // Define main clock speed
8  #define MCLKIN 11059200
9
10 // Calculate timer ticks to count a millisecond
11 #define MS 0xFFFF - ( MCLKIN / 12 ) / 1000
12
13 // Break ms into two 8 bit values
14 #define MS_H ( MS >> 8 ) & 0x00FF
15 #define MS_L MS & 0x00FF
16
17 // Time in ms since timer1 start
18 extern unsigned long systime;
19
20 // Get current systime
21 unsigned long get_ms_counter( void );
22
23 // Get time difference
24 unsigned long d_time_ms( unsigned long t2 );
25
26 // Set delay in ms
27 void delay_ms( unsigned long ms );
28
29 // Init timers
30 void init_timers( );

```

## sys\_timer.c

```
1  #include "sys_timer.h"
2  #include "notes.h"
3  #include "aduc812.h"
4
5  unsigned long get_ms_counter( void ) {
6      unsigned long res;
7      ET1 = 0;
8      res = systime;
9      ET1 = 1;
10     return res;
11 }
12
13 unsigned long d_time_ms( unsigned long t2 ) {
14     unsigned long t1 = get_ms_counter();
15     return t1 - t2;
16 }
17
18 void delay_ms( unsigned long ms ) {
19     unsigned long t1 = get_ms_counter();
20     while ( 1 ) {
21         if ( d_time_ms( t1 ) > ms ) break;
22     }
23 }
24
25 void init_timers( ) {
26
27     // Set T0 and T1 to timer mode
28     TMOD = 0x11;
29
30     // Set INT1 edge-sensitive
31     IT1 = 1;
32
33     // Timers priorities
34     PT1 = 0;
35     PT0 = 1;
36
37     // Start values for T0
38     TH0 = A_H;
39     TL0 = A_L;
40
41     // Start values for T1
42     TH1 = MS_H;
43     TL1 = MS_L;
44
45     // Start timers T0 and T1
46     TR1 = 1;
47     TR0 = 1;
48
49     // Enable timers T0 and T1
50     ET1 = 1;
51     ET0 = 1;
52
53     // Enable external interrupt 1
54     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.