Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики
(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

Кафедра вычислительной техники

Лабораторная работа №2
по дисциплине «Информационно-управляющие системы»
«Таймеры. Система прерываний»
Вариант 6

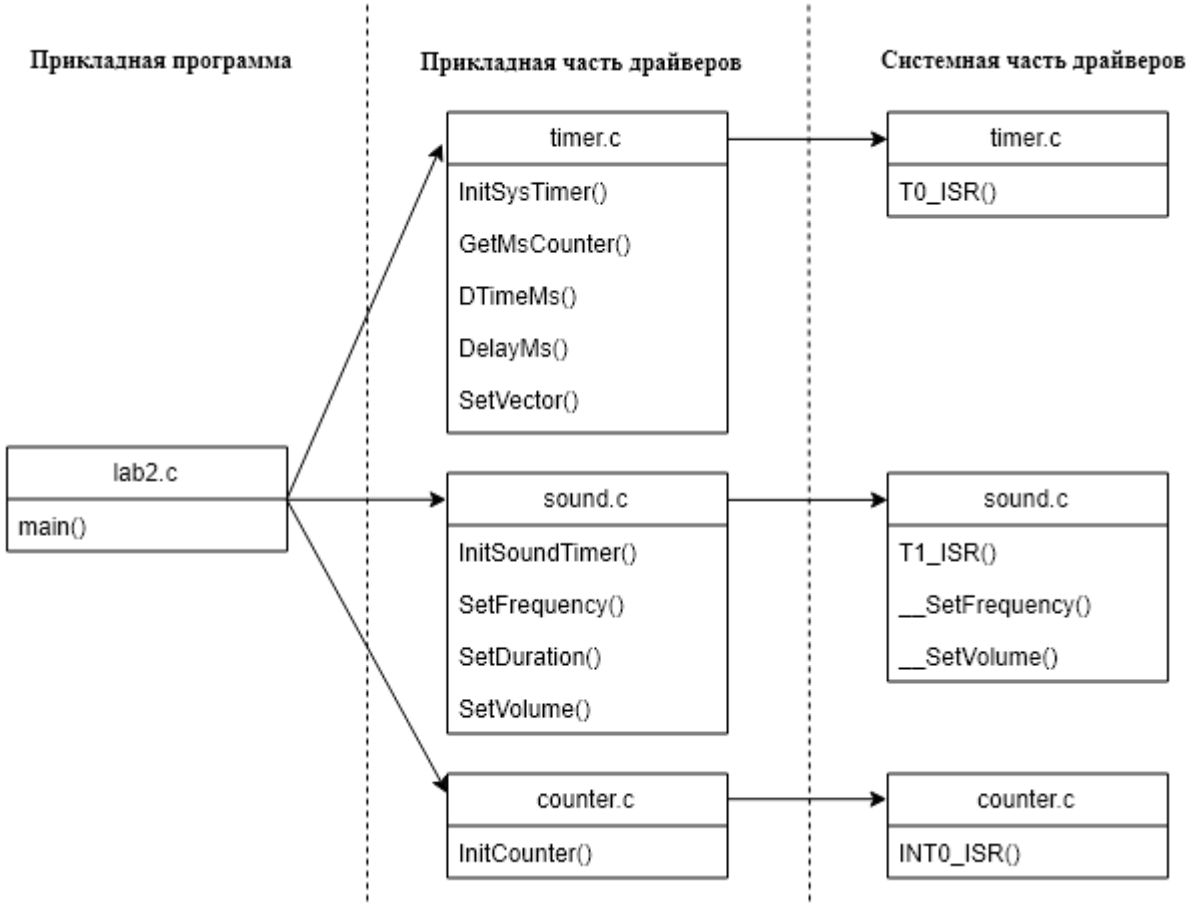
Выполнили
студенты 4 курса,
группы P3400
Абыков Айдар Альбертович
Бурангулов Аскар Азаматович
Сапожников Борис Константинович
Руководитель:
Ключев Аркадий Олегович

Санкт-Петербург — 2018

1. Задание

Контроллер SDK-1.1 циклически проигрывает восходящую гамму нот первой октавы (длительность каждой ноты – 0,5 секунды) и на линейку светодиодов выводит количество замыканий входа INT0. В результате выполнения работы должны быть разработаны драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.

2. Модель взаимодействия прикладной программы, прикладной части драйверов и системной части драйверов.



3. Исходный текст программы с комментариями.

timer.c

```
#include "aduc812.h"
#include "max.h"

void SetVector(unsigned char xdata * Address, void * Vector);
void T0_ISR(void) __interrupt(0);

unsigned long __systime = 0;

//Инициализация системного таймера
void InitSysTimer(void)
{
	TR0 = 0x0;//Выключение таймера 0
	TMOD = TMOD & 0b11111101 | 0b0000001;
	// Выбор режима работы - 16-разрядный таймер
	TH0 = 0xFC; //Настройка таймера на частоту 1000 Гц
	TL0 = 0x66;
	TR0 = 0x1; //Включение таймера 0
```

```
	SetVector(0x200B, (void *)T0_ISR);
	//Установка вектора прерываний в пользовательской таблице
	ET0 = 1; //Разрешение прерываний от таймера 0
	EA = 1; //Разрешение всех прерываний
}

//Обработчик прерывания от таймера 0
void T0_ISR(void) __interrupt(0) {
	__systime++; // Время в милисекундах
	TH0 = 0xFC; //Настройка таймера на частоту 1000 Гц
	TL0 = 0x66;
}

//Получение текущей метки времени в миллисекундах
unsigned long GetMsCounter(void) {
	unsigned long res;
	ET0 = 0;
	res = __systime;
	ET0 = 1;
	return res;
}

//Подсчет количества миллисекунд, прошедших с временной метки t0 до текущего времени.
unsigned long DTimeMs(unsigned long t0) {
	unsigned long t1 = (unsigned long)GetMsCounter();
	return t1 - t0;
}

//Задержка на t миллисекунд
void DelayMs(unsigned long t) {
	unsigned long t1 = (unsigned long)GetMsCounter();
	while (1) {
		if (DTimeMs(t1) > t) break;
	}
}

//Установка вектора прерывания в пользовательской таблице
void SetVector(unsigned char xdata * Address, void * Vector) {
	unsigned char xdata * TmpVector; // Временная переменная
	// Первым байтом по указанному адресу записывается
	// код команды передачи управления ljmp, равный 0x02
	*Address = 0x02;
	// Далее записывается адрес перехода Vector
	TmpVector = (unsigned char xdata *) (Address + 1);
	*TmpVector = (unsigned char)((unsigned short)Vector >> 8);
	++TmpVector;
	*TmpVector = (unsigned char)Vector;
	// Таким образом, по адресу Address теперь
	// располагается инструкция ljmp Vector
}
```

sound.c

```
#include <stdint.h>
#include "aduc812.h"
#include "timer.h"
#include "max.h"
#include "led.h"

#define round(x) ((int)((x)+0.5))
#define F_OSC 11059200
#define COUNTS_MAX 65536
#define MN 12
#define OFF 0
#define ON 1

unsigned long start_time_stamp;
unsigned long duration;
uint8_t volume = 7; //значение по умолчанию
```

```
float frequency;
unsigned char on_off = ON;

void T1_ISR(void) __interrupt(1);
void __SetFrequency();
void __SetVolume(unsigned char on_off);

//Установка длительности звучания (в милисекундах)
void SetDuration(unsigned long d) {
	duration = d;
	__SetFrequency(); //Настройка таймера на заданную частоту
	__SetVolume(ON); //Включение звукоизлучателя
	TR1 = 0x1; //Включение таймера 0
	start_time_stamp = GetMsCounter(); //Установить временную метку
}

//Установка частоты
void SetFrequency(float f) {
	frequency = f;
}

//Установка громкости
void SetVolume(unsigned char v) {
	volume = v > 7 ? 7 : v;
}

//Инициализация звукового таймера
void InitSoundTimer(void) {
	InitSysTimer();
	TR1 = 0x0; //Выключение таймера 1
	TMOD = TMOD & 0b10011111 | 0b00010000;
	// Выбор режима работы - 16-разрядный таймер
	SetVector(0x201B, (void *)T1_ISR);
	//Установка вектора прерываний в пользовательской таблице
	ET1 = 1; //Разрешение прерываний от таймера 1
	EA = 1; //Разрешение всех прерываний
}

//Обработчик прерывания от таймера 1
void T1_ISR(void) __interrupt(1) {
	if (DTimeMs(start_time_stamp) < duration) {
		__SetFrequency();
		__SetVolume(on_off);
	}
	else {
		__SetVolume(OFF);
		TR1 = 0x0;
	}
}

//Внутренняя функция для установки уровня громкости
void __SetVolume(unsigned char __on_off) {
	write_max(ENA, __on_off ? 0b0100000 | (volume>7 ? 7 : volume) << 2 : 0b0100000); //0b001XXX00
	on_off = ++__on_off % 2;
}

//Внутренняя функция установки регистров TH1 и TL1
void __SetFrequency() {
	uint16_t freq = COUNTS_MAX - (int)round(F_OSC / MN / (frequency * 2));
	//Настраиваем на частоту в два раза больше заданной,
	//поскольку трубется прерываться два раза в период
	//чтобы включить и выключить звукоизлучатель
	TH1 = (freq >> 8) & 0xFF;
	TL1 = freq & 0xFF;
}
```

counter.c

```
#include "aduc812.h"
#include "led.h"
#include "max.h"
#include "sound.h"
#include "timer.h"


unsigned char int_counter = 0;

void INT0_ISR(void) __interrupt(2);

// Инициализация счетчика
void InitCounter(void) {
	SetVector(0x2003, (void *)INT0_ISR);
	//Установка вектора прерываний в пользовательской таблице
	IT0 = 0x1; //Определяем тип запроса прерывания: по спаду
	write_max(ENA, 0b0100000);
	EX0 = 1; //Разрешение внешнего прерывания 0
	EA = 1; //Разрешение всех прерываний
}

// Обработчик внешнего прерывания
void INT0_ISR(void) __interrupt(2) {
	leds(int_counter++);
}
```

lab2.c

```
#include "sound.h"
#include "timer.h"
#include "counter.h"

float notes[] = { 261.63, 293.67, 329.63, 349.22, 391.99, 440.00, 493.88 };

// Главная функция
void main(void) {
	unsigned char cur_note = 0x00;

	InitSysTimer();
	InitSoundTimer();
	InitCounter();

	while (1)
	{
		SetFrequency(notes[cur_note]);
		SetDuration(500);
		cur_note = (++cur_note) % 7;
		DelayMs(500);
	}
}
```

4. Основные результаты.

В результате выполнения работы были разработаны прикладная программа, соответсвующая заданию и драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.