Университет ИТМО

**Лабораторная работа №2 по дисциплине**

**«Системы ввода/вывода и**

**периферийные устройства»**

Выполнили:

студенты 4-го курса

группы P3415

Припадчев Артём

Кунцова Анастасия

Григорьева Екатерина

Санкт-Петербург

2015

**Задание**

Разработать и реализовать драйвер системного таймера микроконтроллера ADuC812. Написать тестовую программу с использованием разработанного драйвера по алгоритму, соответствующему варианту задания.

**Вариант 8.**

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 (линия внешнего прерывания INT1 на рис.5). В результате выполнения работы должны быть разработаны драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.

**Таймеры-счетчики**

Таймер позволяет производить отсчет временных интервалов заданной продолжительности. Принцип действия таймера основан на двоичном счетчике с возможностью предварительной записи исходного значения. После каждого такта синхросигнала счетчик прибавляет или отнимает единицу от имеющегося у него значения. При достижении нуля (т.е. при переполнении), счетчик вырабатывает активный уровень на выходе. Как правило, выходной сигнал таймера заводят на вход запроса прерывания микропроцессора или контроллера прерываний.

В большинстве современных встроенных систем таймеры используются в качестве основы для организации системы разделения времени на базе переключателя задач. В данном случае таймер используется в паре с механизмом прерываний.

Микроконтроллер ADuC812 имеет три программируемых 16-битных таймера/счетчика: таймер 0, таймер 1, таймер 2. Каждый таймер состоит из двух 8-битных регистров THX и TLX. Все три таймера могут быть настроены на работу в режимах «таймер» или «счетчик».

В режиме «таймер» регистр инкрементируется каждый машинный цикл, т.е. можно рассматривать это как подсчет машинных циклов. Так как машинный цикл состоит из 12 перепадов напряжения на тактовом входе микроконтроллера, частота инкрементирования таймера в 12 раз меньше тактовой частоты микроконтроллера (соответствующего кварцевого резонатора).

В режиме «счетчик» регистр инкрементируется по перепаду из «1» в «0» внешнего входного сигнала, подаваемого на вывод микроконтроллера T0, T1 или T2. Когда опрос показывает высокий уровень в текущем машинном цикле и низкий уровень в следующем, счетчик увеличивается на 1. Таким образом, на распознавание периода требуются два машинных цикла, максимальная частота подсчета входных сигналов равна 1/24 частоты кварцевого резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входной сигнал должен удерживать значение «1», как минимум, в течение одного машинного цикла микроконтроллера.

Таким образом, отличие этих двух режимов работы таймера заключается в типе источника инкрементирования. В остальном функционирование в режимах счетчика и таймера аналогично.

Для управления звуковым излучателем используется регистр ПЛИС ENA (адрес 080004h). 2-4 биты регистра ENA управляют величиной напряжения на динамике, т.е. позволяют задавать громкость звука: чем больше «единиц» выставлено в этих битах, тем громче звук. Частота звука задается частотой смены нулей и единиц в управляющих битах регистра ENA.

В микроконтроллере ADuC812 девять источников прерываний с двумя уровнями приоритетов. Когда происходит прерывание, значение программного счетчика помещается в стек, а в сам счетчик загружается адрес соответствующего вектора прерывания. Прерывания ADuC812 имеют вектора в диапазоне 0003h-0043h, которые попадают в область младших адресов памяти программ. Это пространство соответствует 8Кб (0000h- 2000h) FLASH-памяти.

В стенде SDK-1.1 пользователь не имеет возможности записи во FLASH-память (запись программ осуществляется во внешнюю память программ и данных), следовательно, не может подставить свои процедуры обработки прерываний (точнее, команды перехода к процедурам) по адресам, соответствующим векторам прерываний. Проблема использования прерываний в пользовательских программах решается следующим образом:

1. По адресам (0003h-0043h) векторов прерываний во FLASH-памяти SDK-1.1 располагаются команды переходов на вектора пользовательской таблицы, размещенной в адресах 2003h-2043h.

2. По адресам векторов пользовательской таблицы пользователем указываются команды переходов на процедуры обработки прерываний.

**Описание работы**

Данная лабораторная работа посвящена изучению таймера и системы прерываний микроконтроллера ADuC812. Основными функциями системного таймера являются: измерение интервалов времени и выполнение периодических задач. В данной работе с помощью таймеров требуется управлять светодиодными индикаторами (динамическая индикация) или звуковым излучателем (проигрывание мелодии), входящими в состав контроллера SDK-1.1.

Драйвер системного таймера должен состоять из двух частей: основной части, находящейся в обработчике прерывания и прикладной части – API-функций, используемых в программе:



Кроме того, могут быть реализованы функции работы с таймером в режиме «счетчик» (например, чтение счетчика).

Драйвер светодиодных индикаторов/звукового излучателя (зависит от варианта задания) должен быть реализован по тому же принципу, что и драйвер системного таймера. А именно: в обработчике прерывания от таймера должна выполняется сама динамическая индикация/проигрывание мелодии – примеры периодических задач, а через API-функции осуществляется настройка отображения анимации/звука и управление этими процессами.

**Пояснения для вариантов заданий с использованием звукового излучателя**

 Требуется написать драйвер звукового излучателя, позволяющий задавать частоту (в герцах) и длительность звука (в миллисекундах).

 Для управления звуковым излучателем используется регистр ПЛИС ENA (адрес 080004h). 2-4 биты регистра ENA управляют величиной напряжения на динамике, т.е. позволяют задавать громкость звука: чем больше «единиц» выставлено в этих битах, тем громче звук. Частота звука задается частотой смены нулей и единиц в управляющих битах регистра ENA. Ниже в таблице приведены частоты нот первой октавы.

|  |  |
| --- | --- |
| **Нота** | **Частота, Гц** |
| До | 261,63 |
| Ре | 293,67 |
| Ми | 329,63 |
| Фа | 349,22 |
| Соль | 391,99 |
| Ля | 440,00 |
| Си | 493,88 |

**Исходный код**

/\*--------------------------------------------------------------

Файл: new.c

Описание: Синтез звуков из второй октавы при помощи

 внутреннего таймера, а также регистрирование

 внешних прерываний и вывод их количество на диодах.

--------------------------------------------------------------\*/

#include "aduc812.h"

#include "max.h"

#include "led.h"

/\*\*----------------------------------------------------------------------------

 delay

-------------------------------------------------------------------------------

Задержка выполнения на некоторое время

Вход: ms - переменная указывающая время задержки

Выход: нет

Результат: нет

Описание: Производится задержка исполнения программы

 на время работы циклов.

-----------------------------------------------------------------------------\*/

void delay ( unsigned long ms )

{

 volatile unsigned long i, j;

 for( j = 0; j < ms; j++ )

 {

 for( i = 0; i < 50; i++ );

 }

}

int i = 0; //Счётчик для регулирования времени исполнения ноты

char t = 0; //Флаг для переключения регистра ENA

unsigned char k = 0; //Счётчик внешних прерываний

/\*\*----------------------------------------------------------------------------

 T1\_ISR

-------------------------------------------------------------------------------

Обработчик внешнего прерывания.

Вход: нет

Выход: нет

Результат: нет

Описание: Счётчик внешний прерываний

-----------------------------------------------------------------------------\*/

void T1\_ISR( void ) \_\_interrupt ( 1 )

{

 k++;

}

/\*\*----------------------------------------------------------------------------

 T0\_ISR

-------------------------------------------------------------------------------

Обработчик прерывания по таймеру 0

Вход: нет

Выход: нет

Результат: нет

Описание: Устанавливается определённое значение в регистр прерываний

 для конфигурации частоты прерываний (нужно для выработки

 звука нужной частоты).

-----------------------------------------------------------------------------\*/

void T0\_ISR( void ) \_\_interrupt ( 2 )

{

 if( i <= 500 && t == 0 )

 {

 //Нота Си - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xFC;

 TL0 = 0x17;

 }

 else if ( i <= 500 && t == 1 )

 {

 //Нота Си - верхняя часть

 write\_max(ENA, 0b0111100);

 t=0;

 i++;

 TH0 = 0xFC;

 TL0 = 0x17;

 }

 else if( i > 500 && i <= 948 && t==0 )

 {

 //Нота Ля - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xFB;

 TL0 = 0x8B;

 }

 else if( i > 500 && i <= 948 && t == 1 )

 {

 //Нота Ля - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xFB;

 TL0 = 0x8B;

 }

 else if( i > 948 && i <= 1338 && t == 0 )

 {

 //Нота Соль - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xFA;

 TL0 = 0xFF;

 }

 else if( i > 948 && i <= 1338 && t== 1 )

 {

 //Нота Соль - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xFA;

 TL0 = 0xFF;

 }

 else if( i > 1338 && i <= 1687 && t==0 )

 {

 //Нота Фа - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xFA;

 TL0 = 0x69;

 }

 else if( i > 1338 && i <= 1687 && t==1 )

 {

 //Нота Фа - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xFA;

 TL0 = 0x69;

 }

 else if( i > 1687 && i <= 2016 && t==0 )

 {

 //Нота Ми - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xFA;

 TL0 = 0x0F;

 }

 else if( i > 1687 && i <= 2016 && t==1 )

 {

 //Нота Ми - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xFA;

 TL0 = 0x0F;

 }

 else if( i > 2016 && i <= 2310 && t==0 )

 {

 //Нота Ре - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xF9;

 TL0 = 0x5B;

 }

 else if( i > 2016 && i <= 2310 && t==1 )

 {

 //Нота Ре - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xF9;

 TL0 = 0x5B;

 }

 else if( i > 2310 && i <= 2571 && t==0 )

 {

 //Нота До - нижняя часть

 write\_max(ENA, 0b0100000);

 t = 1;

 i++;

 TH0 = 0xF8;

 TL0 = 0x89;

 }

 else if( i > 2310 && i <= 2571 && t==1 )

 {

 //Нота До - верхняя часть

 write\_max(ENA, 0b0111100);

 t = 0;

 i++;

 TH0 = 0xF8;

 TL0 = 0x89;

 }

 else

 {

 //Начать проигрывать мелодию заново

 i = 0;

 t = 0;

 }

}

/\*\*----------------------------------------------------------------------------

 SetVector

-------------------------------------------------------------------------------

Установка пользовательского вектора прерывания.

Вход:

 Vector – адрес обработчика прерывания,

 Address – вектор пользовательской таблицы прерываний.

Выход: нет

Результат: нет

Описание: Функция, устанавливающая вектор прерывания в

 пользовательской таблие прерываний

-----------------------------------------------------------------------------\*/

void SetVector(unsigned char \_\_xdata \* Address, void \* Vector)

{

 unsigned char \_\_xdata \* TmpVector;

 // Первым байтом по указанному адресу записывается // код команды передачи управления ljmp, равный 02h

 \*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

}

void main( void )

{

 // Инициализация работы таймера-0

 TH0 = 0xFC;

 TL0 = 0x17;

 TMOD = 0x01;

 TCON = 0x11;

 // Установка вектора в пользовательской таблице

 SetVector( 0x200B, (void \*)T0\_ISR );

 SetVector( 0x2003, (void \*)T1\_ISR );

 // Разрешение прерываний от таймеров

 EA = 1;

 ET0 = 1;

 EX0 = 1;

 while( 1 )

 {

 leds( k );

 }

}

**Вывод**

В ходе выполнения данной лабораторной работы реализован драйвер системного таймера микроконтроллера ADuC812. Также написана тестовая программа с использованием разработанного драйвера по алгоритму.