Степанов А.Б., Суворов К.А.

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ В РАДИОТЕХНИЧЕСКИХ СИСТЕМАХ

РЕАЛИЗАЦИЯ АЛГОРИТМОВ НА ЭЛЕМЕНТНОЙ БАЗЕ СО СВЕРХНИЗКИМ ЭНЕРГОПОТРЕБЛЕНИЕМ

Лабораторный практикум по дисциплине

СПбГУТ

Лабораторная работа №2
Работа с прерываниями микроконтроллера

Цель работы: реализовать систему управления светодиодами с использованием прерываний микроконтроллера.

2.1. Краткая теоретическая справка

Прерывания используются в цифровых сигнальных процессорах и микроконтроллерах. Рассмотрим процедуру выполнения прерывания (рис. 4) [2]:

1. Выполнение цифровым сигнальным процессором (микроконтроллером) основной (исполняемой) программы до некоторой команды *n*.

2. Поступление запроса на прерывание.

3. Сохранение текущего состояния процессора (микроконтроллера) в стеке, освобождение регистров.

4. Выполнение подпрограммы обслуживания прерывания, соответствующей запросу на прерывание.

5. В случае наличия в подпрограмме обслуживания прерывания команды возврата к выполнению основной программы, из стека происходит изъятие информации о состоянии процессора (микроконтроллера) до поступления запроса на прерывание.

6. Продолжение выполнения основной программы устройства, начиная с команды *n*+1.

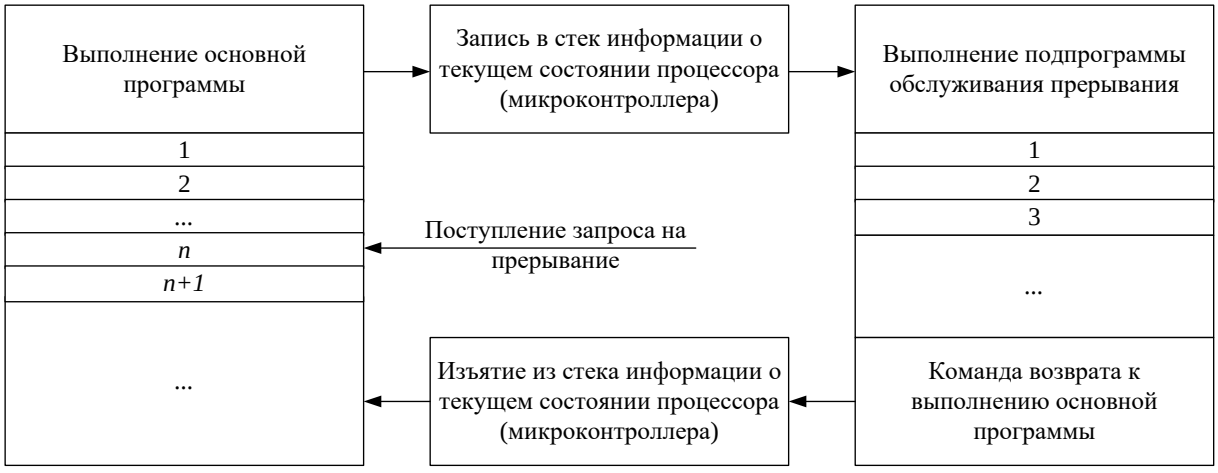

Рис. 4. Процедура выполнения прерывания

Описанный принцип выполнения прерываний соответствует простому прерыванию. Также выделяют вложенные прерывания, которые описывают алгоритм работы процессора (микроконтроллера) при обслуживании запроса на прерывание и одновременном поступлении нового запроса на

прерывание с более высоким приоритетом. В этом случае прерывается обслуживание первого запроса на прерывание, информация о состоянии процессора (микроконтроллера) записывается в стек, выполняется подпрограмма обслуживания прерывания с более высоким приоритетом. В случае наличия команды возврата происходит изъятие информации из стека и продолжается выполнение подпрограммы обслуживания прерывания первого запроса.

Более подробную информацию о прерываниях процессора (микроконтроллера) можно получить в [2].

2.2. Задание на лабораторную работу

Выполнение лабораторной работы осуществляется в следующем порядке:

1. Подключите отладочную плату MSP-EXP430G2 к компьютеру.
2. Запустите Code Composer Studio: «Start → All Programs → Texas Instruments → Code Composer Studio».
3. Создайте новый проект: «File → New → CCS Project».
4. В поле «Project name» введите имя проекта, нажмите «Next», установите тип проекта: «Project Type → MSP430». Нажмите «Next» два раза для перехода к странице настроек проекта – «Project Settings». В поле «Device Variant» выберите необходимое устройство из списка – семейство и модель микроконтроллера, например, MSP430G2452. Модель микроконтроллера можно уточнить, обратившись к маркировке микросхемы. Далее нажмите «Finish».
5. Создайте новый исходный файл: «File → New → Source File». Введите имя файла на английском языке и добавьте расширение *.c.
6. В окне редактирования исходного кода введите текст программы [1]:

```
#include <msp430g2452.h>
unsigned int blink = 0;  //данная переменная отв. за режим раб.
void main(void)
{
WDTCTL = WDTPW + WDTHOLD;       //остановка сторожевого таймера
P1DIR |= (BIT0 + BIT6);
/* установка портов P1.0 и P1.6 в качестве выходных */
P1OUT &= ~(BIT0 + BIT6);               //выключение светодиодов
P1IE |= BIT3;   //разрешение прерываний для порта P1.3 (кнопка)
__enable_interrupt();  //ком. глобального разрешения прерываний
for (;;)
/* Основной цикл. Работа в одном из режимов мигания. */
{
if (blink == 0)
/* режим 0 – стартовый режим. Светодиоды отключены. */
```

```
{
P1OUT &= ~(BIT0 + BIT6);
}
if(blink == 1)                                          //режим 1
{
P1OUT ^= (BIT0 + BIT6);
__delay_cycles(100000);
}
if(blink == 2)                                          //режим 2
{
P1OUT ^= BIT0;
__delay_cycles(50000);
}
if(blink == 3)                                          //режим 3
{
P1OUT ^= BIT6;
__delay_cycles(25000);
}
if(blink == 4)                                          //режим 4
{
P1OUT ^= BIT0;
__delay_cycles(100000);
P1OUT ^= BIT6;
__delay_cycles(100000);
}
if(blink == 5)                                          //режим 5
{
P1OUT ^= BIT0;
__delay_cycles(15000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(500000);
P1OUT ^= BIT6;
__delay_cycles(15000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(500000);
}
if(blink == 6)                                          //режим 6
{
P1OUT ^= BIT6;
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(50000);
P1OUT ^= BIT6;
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(50000);
P1OUT ^= BIT6;
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(400000);
P1OUT ^= BIT0;
```

```
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(50000);
P1OUT ^= BIT0;
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(50000);
P1OUT ^= BIT0;
__delay_cycles(25000);
P1OUT &= ~(BIT0 + BIT6);
__delay_cycles(400000);
}
}
}
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)           //обработчик прерывания
{
if (blink < 6)                         //все ли режимы пройдены?
{
blink++;
/* нет, инкремент переменной (переход к следующему режиму) */
}
else blink = 0;     //да, сброс переменной (переход к режиму 0)
P1IFG &= ~BIT3;                    //сброс флага прерываний P1.3
P1OUT &= ~(BIT0 + BIT6);               //отключение светодиодов
}
```

7. Запустите отладчик: «Target → Debug Active Project» или нажмите кнопку  . При появлении сообщения об ошибках в программном коде их необходимо исправить. При отсутствии ошибок отладчик автоматически произведет форматирование памяти микроконтроллера и запишет в нее новый исполняемый код.
8. Запустите исполняемую программу: «Target → Run» или во вкладке «Debug» нажмите кнопку  . Убедитесь в работе отладочной платы. Встроенные светодиоды P1.0 и P1.6 должны мигать в зависимости от выбранного режима. Выполните нажатие пользовательской кнопки, контролируя изменение режима работы светодиодов.
9. Приостановите выполнение исполняемой программы – нажмите: «Target → Halt» или кнопку  .
10. Для остановки выполнения исполняемой программы нажмите: «Target → Terminate All» или кнопку  .
11. Для выхода из Code Composer Studio нажмите: «File → Exit».

2.3. Требования к отчету

Отчет по лабораторной работе оформляется согласно принятым требованиям и включает:

1. Титульный лист с указанием: названия университета и кафедры, номера и названия лабораторной работы, ФИО обучающихся, номера их учебной группы, ФИО и должности преподавателя. Внизу страницы указывается город и год.

2. Раздел «Выполнение лабораторной работы» содержит: цель работы, основные этапы создания проекта с рисунками и комментариями. При технической возможности получения фотографий работы отладочной платы в различных режимах, они также размещаются в отчете.

3. Рисунок, поясняющий принцип выполнения прерываний.

4. Выводы и ответы на контрольные вопросы.

5. Список литературы.

2.4. Контрольные вопросы

1. Опишите принцип выполнения прерывания.
2. Какие прерывания называются долгими?
3. Какие прерывания называются быстрыми?
4. Какие прерывания называются вложенными?
5. Что такое векторы прерываний?
6. Для чего используются приоритеты прерываний?
7. С какой целью применяется маскирование прерываний?
8. Что такое флаги прерываний?

2.5. Список литературы

1. Официальный сайт компании Texas Instruments [Электронный ресурс]. URL: http://www.ti.com (дата обращения 01.09.2019).

2. *Солонина, А. И.* Алгоритмы и процессоры цифровой обработки сигналов / А. И. Солонина, Д. А. Улахович, Л. А. Яковлев. – СПб. : БХВ–Петербург, 2002. – 464 с.

Лабораторная работа №3
Реализация системы воспроизведения тоновых мелодий

Цель работы: реализовать систему воспроизведения тоновых мелодий на основе микроконтроллера семейства MSP430G2xxx.

3.1. Краткая теоретическая справка

Данная лабораторная работа посвящена исследованию возможности применения микроконтроллеров для воспроизведения тоновых мелодий.

В основе алгоритма работы системы воспроизведения тоновых мелодий используется поочередное воспроизведение нот, заданных в виде соответствующих частот. При этом также может задаваться длительность воспроизведения каждой ноты и амплитуда сигнала. Все это делает возможным воспроизведение простых мелодий. Для создания более сложных систем воспроизведения тоновых мелодий, как правило, требуется наличие