# Лабораторная работа №1. Порты ввода/вывода общего назначения. Управление контроллером обработки векторов прерываний

**Цель работы**: научиться управлять работой портов ввода/вывода общего назначения, а также изучить основные возможности контроллера обработки прерываний.

## Краткие теоретические сведения

Контроллер векторов прерываний (VIC) обслуживает 32 входа запросов пре­рываний, которые могут назначаться для обслуживания в трех категориях:

– быстрый запрос прерывания (FIQ);

– векторизированный запрос прерывания (IRQ);

– невекторизированный запрос прерывания (IRQ).

Такие назначения могут ­программироваться. Поэтому приоритеты прерываний от различных периферий­ных устройств могут динамически назначаться и корректироваться в процессе выполнения программы пользова­теля.

Быстрый запрос прерывания (FIQ) имеет самый высокий приоритет. Если в данный момент времени поступил более чем один запрос FIQ, то контроллер векторов прерываний подает на ARM-ядро сигналы запросов FIQ, объ­единенные в соответствии с логической функцией ИЛИ. Самое малое из воз­можных времен ожидания обслуживания для FIQ достигается, если только один запрос прерывания классифицирован контроллером как FIQ, в этом случае сервисная подпрограмма обслуживания запускается для вза­имодействия непосредственно с устройством, запросившим прерывание. Если в текущий момент времени имеется более одного запроса, классифицирован­ного контроллером как FIQ, сервисная программа сначала считает из кон­троллера векторов прерываний слово, идентифицирующее источники, за­просившие прерывания.

Векторизированные запросы прерываний имеют средний приоритет. Только 16 из 32 возможных запросов могут назначаться контроллером в эту катего­рию. Любой из 32 запросов может быть назначен на любой из 16 слотов векто­ризированных запросов прерываний, среди которых слот 0 имеет самый высокий приоритет, а слот 15 – самый низкий.

Невекторизированные запросы прерываний обладают самым низким приоритетом.

Контроллер векторов прерываний объединяет в соответствии с функцией логическое ИЛИ сигналы от всех векторизованных и невекторизованных запросов прерываний (IRQ) для их подачи на ядро. Сервисная подпрограмма об­служивания запроса прерывания (IRQ) читает регистр из VIC и производит пе­реход на требуемый вектор. Если поступает любой из векторизованных запросов прерывания, то VIC выделяет для него адрес с самым высоким приоритетом, со­ответствующий сервисной подпрограмме обслуживания запросов прерываний. В ином случае, VIC по умолчанию выделяет для запроса адрес подподпрограммы, ко­торая обслуживает все невекторизованные запросы прерываний. Эта выделяе­мая по умолчанию подпрограмма читает другой регистр VIC, чтобы определить, какие запросы прерываний в данный момент активны.

## Порядок выполнения работы

* + 1. Изучите до начала выполнения лабораторной работы основные возможности использования портов ввода/вывода общего назначения, а также работу контроллера прерываний.
    2. Изучите примеры программирования работы портов ввода/вывода общего назначения, а также контроллера обработки прерываний, приведенные в разделе 2 данных методических указаний.
    3. Разработайте схему алгоритма управления светодиодным индикатором с учетом установленного варианта задания по таблице 4.1.
    4. Запустите интегральную среду разработки Keil uVision.
    5. Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.

Таблица 1.1 – Варианты заданий

|  |  |
| --- | --- |
| № варианта | Формулировка задания |
| 1 | Реализовать алгоритм управления светодиодным индикатором при помощи джойстика SW2. При нажатии джойстика влево или вправо некоторый светодиод перемещается в выбранном направлении, при нажатии вверх или вниз – остаётся на прежней позиции. Перемещение светодиода происходит по срабатыванию прерывания от таймера (частота выбирается произвольно). |

* + 1. Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.
    2. Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением \*.hex, находящийся в папке проекта, в стенд.
    3. Визуально оцените правильность функционирования разработанного алгоритма.

## Результат выполнения работы

#include "lpc21xx.h"

#define LEFT 1

#define RIGHT 2

#define MASK 0x0000FF00

int flag = 0;

//обработчик прерывания от Timer0

\_\_irq void Timer0ISR(void) {

flag = 1;

T0IR = 0x01;

VICVectAddr = 0;

return;

}

char new\_led\_place(int direction, char place){

if (direction == LEFT){

if (place == 1) {

return 0x80;

} else {

return place >>= 1;

}

} else if (direction == RIGHT) {

if (place == 0x80) {

return 1;

} else {

return place <<= 1;

}

}

return 1;

}

void InitTimer0(void) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Инициализация VIC \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

VICDefVectAddr = (unsigned int) &Timer0ISR;

VICIntEnable = 0x10; //Channel#4 is the Timer0

VICIntSelect = 0x00; //all interrupts are IRQs

T0MR0 = 18000000; //Timer match (~ 0.4 second)

T0MCR = 0x03; //Interrupt on Match0, reset timer on match

T0PC = 0x01; // Prescaler to 2

T0TC = 0x00; // reset Timer counter

T0TCR = 0x01; // enable Timer

return;

}

int delay(int \_slp) {

int i, j = 0;

for (i = 0; i < \_slp \* 1000; i++)

j = i + 1;

return j;

}

int main(void) {

int direction = LEFT;

char place = 1;

IODIR0 &= ~(MASK);

IOSET0 = place;

InitTimer0();

while(1) {

if (flag){

IODIR0 |= MASK;

place = new\_led\_place(direction, place);

IOSET0 = place << 16;

IODIR0 &= ~(MASK);

flag = 0;

InitTimer0();

}

switch ((IOPIN0 & (0x1F<<16)) >> 16) {

case 0x1D: direction = LEFT; break;

case 0x0F: direction = RIGHT; break;

case 0x17: break;

case 0x1B: break;

default: break;

}

delay(20);

}

}

## 1.4 Особенности IDE Keil uVision выявленные в ходе выполнения лабораторной работы

В ходе выполнения лабораторной работы никаких новых особенностей работы IDE Keil uVision выявлено небыло.

**Выводы**

В ходе лабораторной работы была изучена система портов ввода/вывода микроконтроллера LPC2148, также была изучена система прерываний и контроллер прерываний VIC данного микроконтроллера. Были рассмотрены общие характеристики микроконтроллерного ядра ARM7TDMI-S. Было написано приложение, которое управляет линейкой светодиодов с помощью таймера и джойстика.