Лабораторная работа №3. Внешние прерывания

Микроконтроллер имеет на кристалле большое количество разнообразных периферийных устройств, и все они могут работать параллельно и независимо друг от друга, а также параллельно выполняемой программе. Каждое периферийное устройство может вызывать прерывание по наступлению определенного события. Прерывание будет происходить только в том случае, если оно разрешено. Разрешение прерываний устанавливается для каждого устройства отдельно.

Для управления механизмом прерываний используется контроллер векторных прерываний (VIC). Он имеет 32 входа запроса прерываний, а также содержит в своём составе 16 слотов прерываний. Источники прерываний программно могут быть разделены на 2 категории: FIQ и IRQ.

* FIQ (Fast Interrupt request) - запросы с наивысшим приоритетом. Если белее чем один запрос прерывания отнесен к категории FIQ, контроллер прерываний объединяет запросы и посылает сигнал процессору ARM. Наибольшая скорость работы достигается, если к категории FIQ отнесен один запрос прерывания.
* Векторные IRQ (Interrupt ReQuest) - запросы со средним приоритетом. Любой из 32 запросов может быть назначен на любой из 16 слотов векторных IRQ.

В случае возникновения прерываний от нескольких слотов одновременно, VIC обработает их по очереди в соответствии с уровнем приоритетности – первым прерывание от слота 1 и далее в порядке возрастания их порядковых номеров.

Основные регистры VIC

Регистр VICIntEnClear (Interrupt Enable Clear Register). Этот регистр позволяет отключить соответствующие прерывания. Доступен только для записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | 1: запись 1 очищает соответствующий бит в регистре разрешения прерываний  0: запись 0 оставляет соответствующий бит в VICIntEnable без изменений |

Регистр VICIntSelect (Interrupt Select Register). Этот регистр задает к какой категории относится запрос прерывания (FIQ или IRQ). Доступен для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | 1: соответствующий запрос прерывания относится к категории FIQ  0: соответствующий запрос прерывания относится к категории IRQ |

Регистры VICVectAddr0-15 (Vector Address Registers). Эти регистры содержат адреса обработчиков прерываний для векторных IRQ. Доступны для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | Если одно или более прерываний разрешены, отнесены к категории IRQ и подключены к включенному векторному слоту IRQ, значение этого регистра с наивысшим приоритетом будет возвращено, когда обработчик прерывания обратится к регистру VICVectAddr. |

Регистры VICVectCntl0-15 (Vector Control Registers). Эти регистры настраивают 16 слотов векторных IRQ. Доступны для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 4:0 | Номер запроса прерывания назначенный этому слоту IRQ. Хорошим стилем программирования считается, когда программа не назначает один и тот же запрос прерывания больше чем одному включенному слоту векторного IRQ. Но если это происходит, то используется слот с меньшим номером. |
| 5 | 1: этот слот векторного IRQ включен и может вызывать адрес обработчика  прерывания, если соответствующее прерывание разрешено, отнесено к категории IRQ и подключено. |

Регистр VICIntEnable (Interrupt Enable Register). Этот регистр содержит информацию о разрешенных запросах прерываний. Доступен для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | При чтении этого регистра, единицы указывают, какие запросы прерываний разрешены. При записи в этот регистр, единицы разрешают соответствующие запросы прерываний. Запись нулей не запрещает прерывания. Для запрещения прерываний следует использовать регистр VICIntEnClear. |

Регистр VICVectAddr (Vector Address Register). Этот регистр предназначен для получения обработчиком прерывания адреса функции обработки конкретного прерывания. Доступен для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | Если любой запрос прерывания, подключенный к слоту векторного IRQ, разрешен,  чтение из этого регистра вернет адрес обработчика прерывания наиболее  приоритетного запроса (с меньшим номером слота). В противном случае будет возвращен адрес обработчика прерываний по умолчанию (VICDefVectAddr). Запись в этот регистр не устанавливает значение, которое будет прочитано в следующий раз. Запись в этот регистр должна находиться в конце обработчика прерываний для обновления очередности прерываний. |

Регистр VICDefVectAddr (Default Vector Address Register). Этот регистр содержит адрес обработчика прерываний для не векторных IRQ. Доступен для чтения и записи.

|  |  |
| --- | --- |
| Разряды | Назначение |
| 31:0 | Когда обработчик прерывания считывает регистр VICVectAddr, и ни один из слотов прерываний не выбирается, то возвращается значение этого регистра. |

Для успешной работы с прерываниями необходимо вначале инициализировать все требуемые регистры. Это можно осуществить с помощью функции:

void InitVIC(void)

{

DWORD i = 0;

DWORD \*vect\_addr, \*vect\_cntl;

VICINTENCLEAR = 0xFFFFFFFF; //Инициализация VIC

VICADDRESS = 0;

VICINTSELECT = 0;

for ( i = 0; i < VIC\_SIZE; i++ ) //Начальная установка в 0 векторов

{

vect\_addr = (DWORD \*)(VIC\_BASE\_ADDR + VECT\_ADDR\_INDEX + i \* 4);

vect\_cntl = (DWORD \*)(VIC\_BASE\_ADDR + VECT\_CNTL\_INDEX + i \* 4);

\*vect\_addr = 0x0;

\*vect\_cntl = 0xF;

}

}

Программа должна включать в себя также следующую строку:

#include "irq.h"

При этом в программу будет включен файл irq.h, в котором определены значения констант, используемых в функции инициализации.

Адрес функции-обработчика прерывания от источника N должен быть записан в регистр VICVectAddrN. В функцию InstallIRQ этот адрес передаётся параметром \*HandlerAddr, где HandlerAddr – имя функции-обработчика. Функция InstallIRQ вместо имён регистров VICVectAddr и VICVectPriority использует их косвенную адресацию относительно базового адреса VIC:

void InstallIRQ( DWORD IntNumber, void \*HandlerAddr, DWORD Priority )

{

DWORD \*vect\_addr;

DWORD \*vect\_cntl;

VICINTENCLEAR = 1 << IntNumber; //Запрет прерывания на время настройки

// Поиск первого свободного VIC адреса

vect\_addr = (DWORD \*)(VIC\_BASE\_ADDR + VECT\_ADDR\_INDEX + IntNumber \* 4);

vect\_cntl = (DWORD \*)(VIC\_BASE\_ADDR + VECT\_CNTL\_INDEX + IntNumber \* 4);

//Установка вектора прерывания

\*vect\_addr = (DWORD)HandlerAddr;

\*vect\_cntl = Priority;

//Разрешение прерывания

VICINTENABLE = 1 << IntNumber;

}

Благодаря этому инициализацию VIC для любого прерывания можно провести простым изменением передаваемых функции параметров.

Для разрешения и запрета прерывания от любого из 32 источников, необходимо записать единичный бит в соответствующую позицию регистров VICIntEnable и VICIntEnClr. Запись нулевых битов в указанные регистры не оказывает никакого действия, благодаря этому вместо операции чтение-модификация-записть можно использовать простую запись в регистры:

VICIntEnClr = 1 << IntNumber; // Запретить прерывание.

VICIntEnable = 1 << IntNumber; // Разрешить прерывание.

После возникновения прерывания от слота адрес обработчика назначенного ему источника из регистра VICVectAddrN копируется в общий регистр адреса прерывания VIC – VICAddress, содержимое которого для перехода на обработчик загружается в программный счётчик PC микроконтроллера.

После завершения программы обслуживания прерываний очистка флага прерываний на уровне периферии отразится в соответствующих битах регистров VIC (VICRawlntr, VICFIQStatus и VICIRQStatus). Кроме того, прежде, чем следующее прерывание может быть обработано, необходимо выполнить запись в регистр VICVectAddr до возвращения из программы обработки прерывания. Эта запись, очистит соответствующий флаг прерывания во внутренней аппаратуре приоритетов прерываний.

Для обслуживания внешних прерываний, например от кнопок, необходимо их инициализировать с помощью функции:

void InitExternalInterrupts()

{

IO0INTENF = 0x0020; //Прерывание по срезу - кнопка на порту P0.5

//Внешнее прерывание устанавливается на EINT3

InstallIRQ( EINT3\_INT, (void \*)ExternalInterruptsHandler3, 0x02 );

}

Также в программу необходимо добавить саму программу обслуживания прерываний:

\_\_irq \_\_nested \_\_arm void ExternalInterruptsHandler3()

{

if(IO0INTSTATF & 0x0020) //Если прерывание вызвано нажатием на первую кнопку

{

FIO2PIN = 0xFF; //Включить все светодиоды

}

IO0INTCLR = 0xFFFFFFFF;//Очистка прерываний от GPIO PORT0

VICADDRESS = 0;

}

Лабораторная работа включает в себя несколько фронтальных заданий:

1. Написать программу проверки состояния кнопок с использованием прерываний

Для этого функцию main прототипа проекта необходимо модифицировать следующим образом:

int main (void)

{

SCS |= 0x20; // Разрешить генератор с кварцевым резонатором 12МГц

while( !(SCS & 0x40) ); //Подождать стабилизации частоты генератора

// Настроить порт

SCS |= 0x01; //Разрешить быстрый ввод/вывод

FIO2DIR = 0x00FF; // Биты 0-7 порта 2 на вывод для управления светодиодами

FIO2MASK = 0; //Все разряды порта 2 работают в быстром режиме

FIO2PIN = 0x00; // Выключить светодиоды

InitVIC(); //Инициализация системы прерываний

InitExternalInterrupts(); //Инициализация внешних прерываний

while (1)//Loop forever

{

//Ничего не делаем - все происходит в функции обработки прерывания

}

return 0;

}

В случае нажатия первой кнопки произойдет внешнее прерывание и, соответственно, будет вызвана функция ExternalInterruptsHandler3(). В данном случае будет обработано прерывание только от первой кнопки. После проверки работоспособности данного примера необходимо модифицировать программу таким образом, чтобы производилась проверка состояния всех кнопок BTN1-BTN4 с отображением на светодиодах (аналогично предыдущей лабораторной работе). При нажатии на первую кнопку должен загораться первый светодиод, на вторую – второй и т.д.

1. Модифицировать программу обработку прерываний от кнопок таким образом, чтобы первый светодиод включался при нажатии на кнопку 1 и выключался при нажатии на кнопку 2
2. Модифицировать программу обработку прерываний от кнопок таким образом, чтобы визуализировать количество срабатываний кнопок при однократном нажатии (проверка наличия эффекта дребезга контактов). Для этого необходимо добавить в программу переменную «счетчик дребезга», которую необходимо увеличивать на единицу в каждом прерывании и выводить полученное число на светодиоды в двоичном формате. Для обнуления «счетчика дребезга» необходимо использовать одну из кнопок, например первую.