**Лабораторная работа 2**

**Обработка внешних прерываний в микроконтроллерах AVR**

*Цель работы:*

- изучение системы прерываний микроконтроллеров AVR,

- освоение системы команд микроконтроллеров AVR,

- ознакомление с работой стека при вызове подпрограмм и обработчиков прерываний,

- программирование внешних прерываний.

**Введение**

Контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств микроконтроллера (таймеров, портов последовательного ввода-вывода, аналогового компаратора и др.). Все прерывания являются маскируемыми. Адреса и маски прерываний указаны в табл.1. В ней использованы обозначения:

- INT0, INT1 – сигналы внешних прерываний, поступающие по линиям порта PD2, PD3. Маски внешних прерываний представлены разрядами INT0 и INT1 (6-й и 7-й разряды соответственно регистра GIMSK микроконтроллера AT90S8515 или регистра GICR микроконтроллера ATmega8515). Сигналы внешних прерываний устанавливают в «1» флаги прерываний INTF0 и INTF1 (6-й и 7-й разряды соответственно регистра GIFR);

- прерывания от таймеров T1, T0 имеют адреса c $003 по $007. Маскирование прерываний от таймеров осуществляется битами регистра TIMSK;

- последующие адреса прерываний для запросов от последовательных каналов ввода – вывода SPI, UART и аналогового компаратора;

- остальные прерывания (INT2, T/C0 COMP и др.), помеченные знаком «\*», поддерживаются только микроконтроллером ATmega8515.

Из табл. 1 видно, что таблица векторов прерываний моделей ATx8515 располагается в начальной области памяти программ, начиная с адреса $0001. Размер таблицы зависит от типа модели. Следует также иметь в виду, что микроконтроллеры семейства Mega допускают перемещение таблицы векторов из начальной области памяти программ в начало области загрузчика, которая располагается в конце программной памяти. Для этого используют два младших бита регистра GICR микроконтроллеров ATmega8515: IVSEL (1-й разряд) и IVCE (0-й разряд). Если IVSEL=0, таблица векторов располагается в начале памяти программ, в противном случае, при IVSEL=1, – в начале области загрузчика. Конкретное значение начального адреса области загрузчика определяется конфигурационными ячейками BOOTSZ1 и BOOTSZ2. Бит IVCE используется для разрешения изменения бита IVSEL.

*Таблица 2.1.* **Адреса и маски прерываний ATx8515**

|  |  |  |  |
| --- | --- | --- | --- |
| Запрос | Адрес | Маска | Флаг |
| RESET | 000 | – | – |
| INT0 | 001 | GIMSK.6/  GICR.6 | GIFR.6 |
| INT1 | 002 | GIMSK.7/  GICR.7 | GIFR.7 |
| T/C1 CAPT | 003 | TIMSK.3 | TIFR.3 |
| T/C1 COMPA | 004 | TIMSK.6 | TIFR.6 |
| T/C1 COMPB | 005 | TIMSK.5 | TIFR.5 |
| T/C1 OVF | 006 | TIMSK.7 | TIFR.7 |
| T/C0 OVF | 007 | TIMSK.1 | TIFR.1 |
| SPI STC | 008 | SPCR.7 | SPSR.7 |
| UART RXC | 009 | UCR.7/  UCRB.7 | USR.7/  UСSRA.7 |
| UART DRE | 00A | UCR.5/  UCSRB.5 | USR.5/  UСSRA.5 |
| UART TXC | 00B | UCR.6/  UCSRB.6 | USR.6/  UСSRA.6 |
| ANA COMP | 00C | ACSR.3 | ACSR.4 |
| INT2\* | 00D | GICR.5 | GIFR.5 |
| T/C0 COMP\* | 00E | TIMSK.0 | TIFR.0 |

Положение вектора в таблице прерываний определяет приоритет соответствующего прерывания. Запрос с меньшим адресом имеет более высокий приоритет. Флаг общего прерывания I расположен в регистре состояния микроконтроллера SREG (бит 7). При I=1 и единичном значении маски запроса прерывание данного типа разрешено. При поступлении запроса устанавливается флаг прерывания в одном из регистров ввода – вывода, который может вызвать аппаратное прерывание. Состояние флага может быть также опрошено программой.

Обработка прерывания начинается после завершения текущей команды, для чего может понадобиться несколько тактов в зависимости от типа выполняемой команды. При обработке прерывания разряд I в регистре SREG сбрасывается в состояние 0, запрещая обработку всех остальных запросов. В стеке сохраняется адрес возврата и выполняется переход по вектору прерывания на первую команду обработчика прерывания. При выходе из прерывающей программы разряд I вновь устанавливается в состояние 1, разрешая обработку прерываний. Программа, выполняемая при пуске микроконтроллера и использующая вектор запроса RESET, не зависит от состояния разряда I. Для обработки прерываний она должна выполнить команду разрешения прерывания SEI, устанавливающую флаг I в состояние 1.

*Обработка внешних прерываний*

В качестве входов внешних прерываний используются входы портов с альтернативной функцией. Это входы PD2, PD3 для прерываний INT0, INT1 и вход PE0 для прерывания INT2 в микроконтроллере ATmega8515. Запросы внешних прерываний INT0, INT1 могут быть представлены низким уровнем сигнала прерывания (L), переходом от высокого уровня сигнала к низкому (HL - по отрицательному фронту), переходом от низкого уровня сигнала к высокому (LH - по положительному фронту), запрос INT2 только переходами (LH) и (HL). В зависимости от типа запроса в регистре управления микроконтроллера MCUCR необходимо установить биты ISCx0 и ISCx1 согласно табл.2 для каждого из прерываний INTx (x=0,1) и определить бит ISC2 в регистре EMCUCR для прерывания INT2. При ISC2=0 прерывание осуществляется по отрицательному фронту, при ISC2=1 – по положительному фронту.

*Таблица 2.2.* **Таблица выбора типа запроса**

|  |  |  |
| --- | --- | --- |
| ISCx1 | ISCx0 | Тип запроса |
| 0 | 0 | L |
| 0 | 1 | - |
| 1 | 0 | HL |
| 1 | 1 | LH |

Далее подготовим и исследуем ряд проектов, представляющих контроллеры для управления светодиодными элементами.

*Проект 1.* Контроллер с программным опросом состояния кнопок.

В качестве примера рассмотрим проект, в котором управление включением/выключением светодиода осуществляется с помощью двух кнопок, подключенных к линиям порта PA0, PA1 (рис.1, а). Кнопка SW0 при замыкании включает светодиод LED на 1 с, кнопка SW1 - на 2 с.

|  |  |
| --- | --- |
| 88 |  |

а) б)

**Рис. 1.** Контроллер (а) и схема алгоритма работы (б)

Исходная программа соответствует алгоритму последовательной проверки состояния кнопок (рис. 1, б) и в случае замыкания одной из них включает светодиод на заданное время. После выполнения временной задержки выполняется проверка возврата кнопки в состояние разомкнуто. Рассчитав параметры циклов для подпрограммы задержки delay1 при частоте работы микроконтроллера 3,69 МГц, добавляем ее в программу.

**Программа 2.1.**

;Соединения на плате STK500: SW0-PA0, SW1-PA1, LED0-PB0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m8515def.inc" ;файл определений для ATmega8515

.def temp = r16 ;временный регистр

.equ led = 0 ;0-й бит порта PB

.equ sw0 = 0 ;0-й бит порта PA

.equ sw1 = 1 ;1-й бит порта PA

.org $000

rjmp INIT ;обработка сброса

;\*\*\*Инициализация МК\*\*\*

INIT: ldi temp,$5F ;установка

out SPL,temp ; указателя стека

ldi temp,$02 ; на последнюю

out SPH,temp ; ячейку ОЗУ

ser temp ;инициализация выводов

out DDRB,temp ; порта PB на вывод

out PORTB,temp ;погасить LED

clr temp ;инициализация

out DDRA,temp ; порта PA на ввод

ldi temp,0b00000011 ;включение ‘подтягивающих’

out PORTA,temp ; резисторов порта PA

test\_sw0: sbiс PINA,sw0 ;проверка состояния

rjmp test\_sw1 ; кнопки sw0

cbi PORTB, led

rcall delay1

sbi PORTB,led

wait\_0: sbis PINA,sw0

rjmp wait\_0

test\_sw1: sbiс PINA,sw1 ;проверка состояния

rjmp test\_sw0 ; кнопки sw1

cbi PORTB,led

rcall delay2

sbi PORTB,led

wait\_1: sbis PINA,sw1

rjmp wait\_1

rjmp test\_sw0

delay1: ; подпрограмма 1 с

ret

delay2: ; подпрограмма 2 с

rcall delay1

rcall delay1

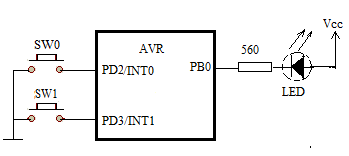
ret

*Задание 1.* Запустив AVR Studio, проверить работу программы в шаговом режиме. С целью ускорения отладки сократить время задержек до минимума. Проконтролировать работу стека при вызове подпрограмм delay1, delay2.

Убедившись в правильности работы программы восстановить параметры подпрограмм задержки и заново откомпилировать программу. Загрузить программу в память микроконтроллера и проверить её работу на плате.

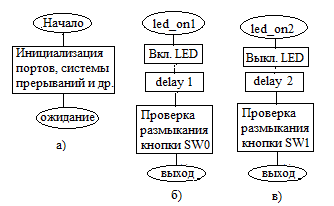
*Проект 2.* Контроллер с двумя внешними запросами прерываний.

Для исследования механизма действия внешних запросов прерываний создадим проект, подключив кнопки SW0, SW1 к входам прерываний INT0, INT1 порта PD (PD2, PD3) микроконтроллера ATmega8515 (рис. 2).



**Рис. 2.** Контроллер с двумя прерываниями

Вместо программной проверки состояний кнопок используем двухуровневую систему внешних прерываний микроконтроллера, на каждый уровень которой поступает запрос от кнопки. Схемы алгоритмов основной программы и обработки прерываний приведены на рис. 3.



**Рис. 3.** Схемы алгоритмов основной программы (а) и прерываний (б, в)

*Задание 2.* Вносим изменения и дополнения в исходный текст программы 2.1, касающиеся обработки прерываний. На этапе инициализации указываются область стека для сохранения адресов возврата, при необходимости адреса векторов прерываний и сами векторы, маска прерываний, общее разрешение прерываний. Завершаем инициализацию переводом процессора в фоновый режим ожидания:

;Соединения на плате STK500: SW0-PD2, SW1-PD3, LED0-PB0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m8515def.inc" ;файл определений для ATmega8515

.def temp = r16 ;временный регистр

.equ led = 0 ;0-о бит порта PB

.equ sw0 = 2 ;2-й бит порта PD

.equ sw1 = 3 ;3-й бит порта PD

.org $000

;\*\*\*Таблица векторов прерываний, начиная с адреса $000\*\*\*

rjmp INIT ;обработка сброса

rjmp led\_on1 ;на обработку запроса INT0

rjmp led\_on2 ;на обработку запроса INT1

;\*\*\*Инициализация SP, портов, регистра маски\*\*\*

INIT: ldi temp,$5F ;установка

out SPL,temp ; указателя стека

ldi temp,$02 ; на последнюю

out SPH,temp ; ячейку ОЗУ

ser temp ;инициализация выводов

out DDRB,temp ; порта PB на вывод

out PORTB,temp ;погасить СД

clr temp ;инициализация

out DDRD,temp ; порта PD на ввод

ldi temp,0b00001100 ;включение ‘подтягивающих’

out PORTD,temp ; резисторов порта PD

ldi temp,((1<<INT0)|(1<<INT1));разрешение прерываний

out GICR,temp ; в 6,7 битах регистра маски GICR

ldi temp,0 ;обработка прерываний

out MCUCR,temp ; по низкому уровню

sei ;глобальное разрешение прерываний

loop: nop ;режим ожиданий

rjmp loop

При программировании прерывающих процедур наряду с переменными основной программы, изменяемыми согласно назначению обработчиков прерываний, восстанавливаются свойства датчиков прерываний, в данном случае единичное состояние кнопки после её отпускания:

led\_on1:

cbi PORTB,led

rcall delay1

sbi PORTB,led

wait\_0: sbis pind,sw0

rjmp wait\_0

reti

led\_on2:

cbi PORTB,led

rcall delay2

sbi PORTB,led

wait\_1: sbis pind,sw1

rjmp wait\_1

reti

delay1: nop

;для подпрограммы задержки 1 c

ret

delay2: ;подпрограмма задержки 2 c

rcall delay1

rcall delay1

ret

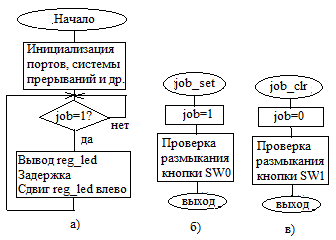
Собираем программу. Рассчитав параметры подпрограмм задержки delay1, delay2, добавляем их в программу.

После компиляции выполняем отладку в шаговом режиме, контролируя переходы к таблице векторов прерываний, состояния программного счетчика, указателя стека и соответствующей области стека. Отметить адреса, сохраняемые в стеке, и объяснить их появление в стеке.

Отлаженный код загружаем в память микроконтроллера и проверяем работу проекта на плате.

В качестве второго примера для самостоятельного исследования механизма действия внешних прерываний рассмотрим проект для управления схемой индикации.

За основу алгоритма управления, представленного на рис. 4, а, примем процедуру, использующую циклический сдвиг управляющего слова в регистре индикации (reg\_led), под действием которого осуществляется движение включенного светодиода на линейке индикации. Время свечения светодиода на каждой позиции индикатора определяется временем задержки delay. Алгоритмом предусмотрен режим Работа/Останов по запросам от внешних кнопок SW0 (Работа), SW1 (Останов), подключенных к входам прерываний INT1 (PD3) и INT2 (PE0).



**Рис. 4.** Схемы алгоритмов основной программы (а) и прерываний (б, в)

*Задание 3.* Подготовим программу соответствующую заданному алгоритму работы. При инициализации помимо общих директив устанавливаем исходный управляющий код в регистре индикации, нулевой разряд которого инициирует зажигание светодиода, настраиваем на вывод порт микроконтроллера и указатель стека.

В цикле алгоритма на каждой итерации выполняем вывод в порт микроконтроллера управляющего слова, временную задержку, затем циклический сдвиг влево управляющего слова.

Программные коды основной программы и обработки прерываний дополняем операциями, связанными с обработкой внешних запросов от кнопок, учитывая заданные входы прерываний (INT1 - линия порта PD3, адрес вектора прерываний $002, маска – бит 7 в регистре GICR; INT2 - линия порта PE0, адрес вектора прерываний $00D, маска – бит 5 в регистре GICR).

Управление работой индикатора в режиме Работа/Останов можно осуществить, проверяя в начале каждой итерации флаг job, устанавливаемый/сбрасываемый с помощью кнопок Работа и Останов.

loop\_job: sbrs job, 0

rjmp loop\_job

; выполнение рабочих операций

…

Значение этого флага (1 или 0) определяется с помощью отдельных прерывающих процедур (job\_set, job\_clr). Первая запускается по запросу прерывания от кнопки Работа, вторая – от кнопки Останов. При программировании прерывающих процедур наряду с определением переменной job восстанавливаются свойства датчиков запросов прерываний, в данном случае единичное состояние кнопки после её отпускания:

job\_set: ldi job, 1

…

reti

job\_clr: clr job

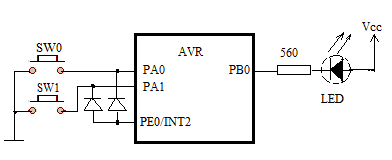
…

reti

Собираем программу, добавляя подпрограмму задержки delay.

После компиляции выполнить пошаговую отладку программы. Отлаженную программу загрузить в память микроконтроллера и проверить работу программы на плате.

*Проект 3.* Контроллер с одним общим входом для запросов прерываний (рис. 5).



**Рис. 5.** Контроллер с общим входом запросов прерываний

Объединим два запроса прерываний от кнопок с помощью диодной сборки для передачи на вход прерывания микроконтроллера INT2. Обработка прерывания в этом случае начинается с программной идентификации источника запроса прерывания. Для решения этой задачи необходимо дополнительно подключить каждую кнопку к выводам, например, порта PA по примеру схемы на рис.1. После определения источника запроса выполняются действия, связанные с идентифицированным запросом (при нажатии кнопки SW0 включается светодиод на 1 с, кнопки SW1 - на 2 с).

*Задание 4.* Составить программу согласно описанному алгоритму работы. Отладить работу программы в пошаговом режиме в среде AVR Studio.

Собрать схему для моделирования в среде ISIS Proteus. Создаём проект, выбрав папку для проекта и указав имя проекта.

Выбрав на панели инструментов слева поле *Component Mode* в окне *Pick Devices* в категории *Microprocessor* выбираем используемый микроконтроллер (например, ATmega8515) и вводим его в окно редактора схемы. Добавляем периферийные элементы: кнопки (Button) из библиотеки компонентов, резистор (Res), светодиод (Led). Присоединяем выводы кнопки: один - к выводу порта PA, второй – к общей шине (*Terminals Mode, GROUND*), выбрав ее из списка *Terminals*. Присоединяем светодиод к шине питания (*Terminals, POWER*).

С помощью команды *Edit Source Code* контекстно-зависимого меню (щелкнув правой кнопкой мыши на обозначении микроконтроллера) добавляем в проект файл с расширением \*.asm, предварительно подтвердив тип микроконтроллера и используемый транслятор (AVRASM). Выполняем компиляцию с помощью команды основного меню *Build Project.*

В меню *Debug* запускаем проект на выполнение (*Start*) и затем способ выполнения программы (пошаговый или прогон). Проверить работу программы, поочередно нажимая кнопки sw0 и sw1 и наблюдая состояние светодиода.

*Задание 5.* Собрать схему на плате STK500. Загрузить программу в память микроконтроллера и проверить работу программы на макете.

**Оформление отчета**

Отчет должен содержать:

а) схемы контроллеров, алгоритмов основной программы, обработчиков прерываний, тексты отлаженных программ всех проектов, результаты наблюдений при тестировании программ;

б) скриншоты стековой области памяти данных (Data), указателя стека, программного счетчика при вызове подпрограммы задержки и обработчиков прерываний с объяснением наблюдаемых результатов;

в) схему проекта («скриншот» из окна редактора Proteus) с текущим состоянием светодиода в режиме исполнения программы.

г) выводы по работе.

Требования при защите: уметь отвечать на вопросы по обработке прерываний, текстам приведенных в отчете программ, по работе стека, отвечать на контрольные вопросы.

**Контрольные вопросы**

1. Какие линияи портов микроконтроллера ATMega8515 используются для внешних прерываний? По каким адресам в памяти программ располагаются векторы внешних прерываний? Как задают адреса векторов при написании программы?

2. Охарактеризуйте каждое из внешних прерываний, указав его технические параметры?

3. Какие команды и регистры нужно задействовать для управления внешними прерываниями?

4. Какое из 3-x внешнее прерывание имеет более высокий приоритет?

5. Как работает диодная схема в случае подключения обоих кнопок к одному входу прерывания? Каким логическим элементом можно заменить диоды?

**Задачи для самостоятельного программирования**

**при подготовке к рубежному контролю**

1. Написать и промоделировать (в среде Proteus или VMLab) программу, которая при нажатии кнопки SWi, подключенной к выводу порта Px, кратковременно включает светодиод LEDj на 40 мс.

2. Написать и промоделировать программу, которая при первом нажатии кнопки SWi включает, а при повторном нажатии выключает светодиод LEDj.

3. Промоделировать работу выходного порта микроконтроллера:

а) в режиме двоичного счета, б) в режиме двоично-десятичного счета,

в) в режиме обратного двоичного счета, г) в режиме обратного двоично-десятичного счета.

4. Вывести на семисегментный индикатор последовательность десятичных цифр 0, 1, 2,..9. Проверить работу программы в среде Proteus или VMLab.

5. Вывести на семисегментный индикатор номер нажатой кнопки.

6. Вывести на светодиоды LED0 – LED3 двоичный код номера нажатой кнопки. Проверить работу программы в симуляторе. В окне редактора присоединить к выходному порту микроконтроллера 4-входовой семисегментный индикатор с встроенным преобразователем 4-разрядного кода в семиразрядный. Наблюдать работу схемы, поочередно нажимая каждую из восьми кнопок.

7. Подобрать необходимую частоту регенерации, при которой последовательно включаемые светодиоды группы из 4 светодиодов (в позициях цифры 4 одного семисегментного индикатора) будут восприниматься глазом так же, как по команде одновременного включения этих светодиодов. Проверить работу программы с помощью симулятора Proteus или VMLab.

8.Написать и промоделировать в Proteus программу “Кодового Замка”, которая после ввода 4-х разрядного двоичного PIN-кода с помощью кнопок SW0 - SW3 осуществляет сравнение с заданным PIN-кодом (номером студента в журнале группы) и включает один светодиод при правильном вводе. При 3-х неправильных попытках введения PIN-кода кнопочная клавиатура должна быть заблокирована, а все светодиоды включены.