Раздел 2
Программирование прерываний

Теоретическая часть

Прерывания — это вызовы определенных функций, генерируемые, главным образом, аппаратной частью микроконтроллера. В результате прерывания выполнение программы останавливается, и происходит переход к соответствующей подпрограмме обработки прерывания.

Прерывания бывают внутренними и внешними. Источниками внутреннего прерывания являются встроенные модули микроконтроллера (например, таймер/счетчик или сторожевой таймер). Внешние прерывания вызываются сбросом (сигнал на выводе RESET) или сигналами на выводах INT0 и INT1.

Язык С позволяет работать с различными видами прерываний микроконтроллера. Процедура обработки прерывания представляется в виде одной из функции: INTERRUPT( ) или SIGNAL( ). Различие состоит в том, что при выполнении функции INTERRUPT( ) разрешены прерывания и при появлении более приоритетного прерывания работа может быть приостановлена, т.е. возможны вложенные прерывания. При выполнении SIGNAL( ) прерывания запрещены.

Формат обработчика:

```
INTERRUPT(Идентификатор прерывания)

{
    // Программа обработки
}

или


SIGNAL(Идентификатор прерывания)

{
    // Программа обработки
}
```

Некоторые идентификаторы прерываний приведены в 7 пункте приложения.

Использование внешних прерываний

Для разрешения внешних прерываний используется регистр GICR.

INT1	INT0	-	-	-	-	IVSEL	IVCE

Рис. 1. Формат регистра GICR

INT1 – Разрешено прерывание с входа INT1
INT0 – Разрешено прерывание с входа INT0

IVSEL и IVCE управляют положением таблицы прерываний. В них необходимо записать 0.

Характер сигналов на выводах INTO/INTl, вызывающих прерывание, определяется с помощью разрядов регистра управления MCUCR:

-	-	-	-	ISC11	ISC10	ISC01	ISC00

Рис. 2. Формат регистра MCUCR

Таблица 1. Выбор способа активизации прерывания по входу INTO

Разряд ISC01	Разряд 1SC00	Описание
0	0	Прерывание вызывается по уровню лог. 0 на входе INTO
1	0	Прерывание вызывается по срезу сигнала INTO
1	1	Прерывание вызывается по фронту сигнала INTO

Таблица 2. Выбор способа активизации прерывания по входу INT1

Разряд ISC11	Разряд ISC10	Описание
0	0	Прерывание вызывается по уровню лог. 0 на входе INT1
1	0	Прерывание вызывается по срезу сигнала INT1
1	1	Прерывание вызывается по фронту сигнала INT1

Прерывания от таймера

Таймеры-счетчики могут использоваться для различных целей: измерения временных интервалов, подсчета внешних событий, генерации сигнала с широтно-импульсной модуляцией.

В рамках данной лабораторной работы остановимся лишь на возможности отсчета временных интервалов с помощью таймера T0.

Для включения данного режима необходимо ввести в регистр управления TCCR0 число 0b01001001.

В этом режиме таймер работает, как суммирующий счетчик, увеличивая своё значение каждый такт. Как только этот счетчик (регистр TCNT0) достигнет значения, записанного в регистре OCR0, произойдет прерывание.

Максимальное число, которое может быть записано в регистр OCR0 равно 255. Для того чтобы получить большую задержку необходимо понизить частоту работы таймера. Для этого используется делитель частоты. Его режим работы определяется тремя младшими битами регистра управления TCCR0.

бит 2	**бит1**	**бит0**	**Коэффициент деления предделителя**
0	0	0	Таймер-счетчик остановлен
0	0	1	1
0	1	0	8
0	1	1	64

1	0	0	256
1	0	1	1024
1	1	0	Тактовый сигнал подается с внешнего вывода T0. Счет осуществляется по срезу.
1	1	1	Тактовый сигнал подается с внешнего вывода T0. Счет осуществляется по фронту.

Для того, чтобы разрешить прерывание по совпадению от таймера T0, необходимо записать в регистр разрешения прерываний таймеров TIMSK число 0b00000001.

Практическая часть

1. Программа обработки прерывания

1.1. С помощью утилиты MFILE **создайте файл параметров**.

1.2. **Введите следующую программу**

```
#include <avr/io.h>            //Подключение библиотеки ввода-вывода
#include <avr/interrupt.h>  //Для доступа к функции равзрешения прерываний sei()
#include <avr/signal.h>        //Для доступа к функции SIGNAL

char n;

SIGNAL(SIG_INTERRUPT0)
{
n=PORTA;                        //Считываем предыдущее значение порта A
if (n==0) n=0xFF; else n=0;   //Инвертируем это значение
PORTA=n;                        //Записываем полученное значение обратно в порт
}


void main(void)                    //Начало основной программы
{
GICR=0b01000000;                //Разрешаем прерывание со входа INT0
MCUCR=0b00000011;             //Прерывание будет по фронту
DDRA=0xFF;                            //Программируем порт A на вывод
sei();                          //Глобальное разрешение прерываний
}
```

1.3. **Откомпилируйте программу**

1.4. **Откройте hex-файл в AVR-Studio**

1.5. Выполните основную часть программы**.** Для этого **нажмите** (замедленное автоматическое выполнение) на панели инструментов. Как только программа «зависнет» на одном месте, **нажмите** (пауза).

1.6. В окне Workspase проверьте правильность выполнения основной части программы:

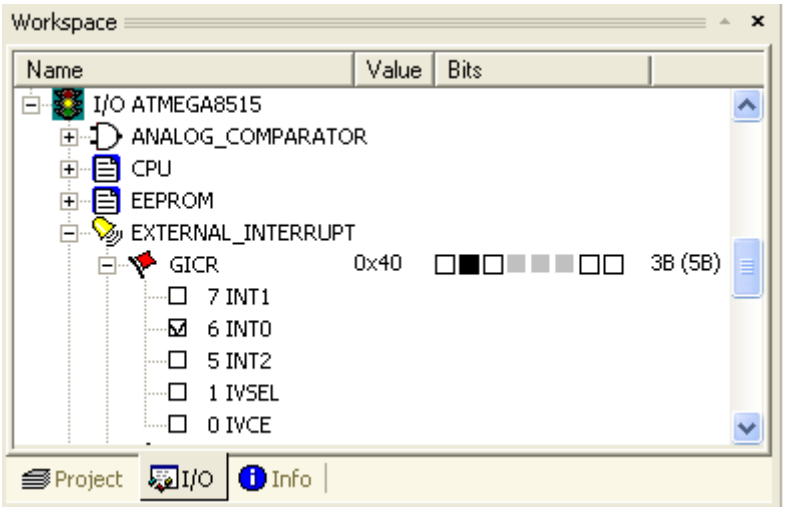

Рис. 3. Регистр GICR

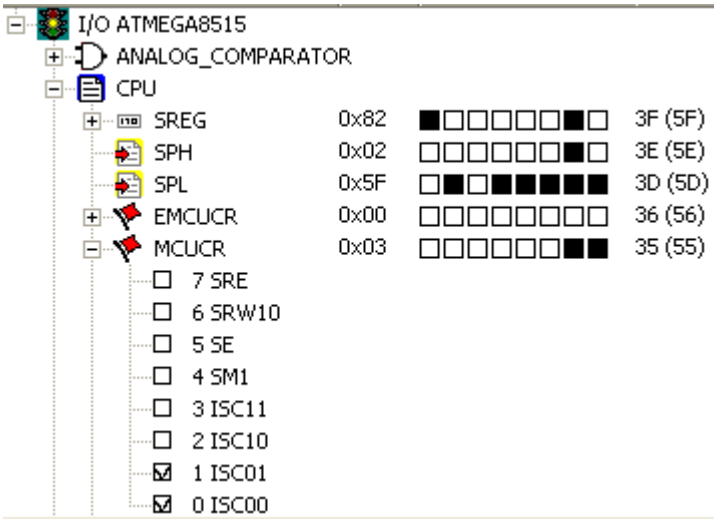

Рис. 4. Регистр MCUCR

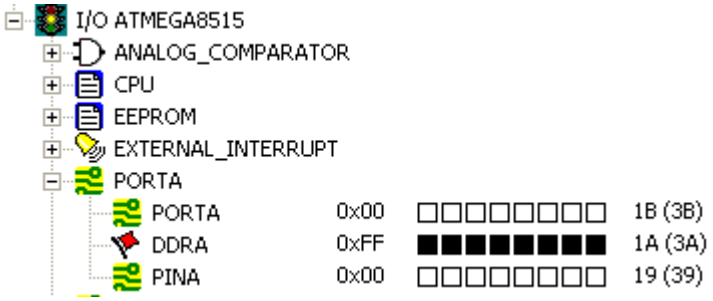

Рис. 5. Регистр DDRA

1.7. Сэмулируйте прерывание INT0. Для этого с помощью мыши **введите единичное значение во второй бит регистра PIND** (Нумерация справа налево, начиная с нулевого).

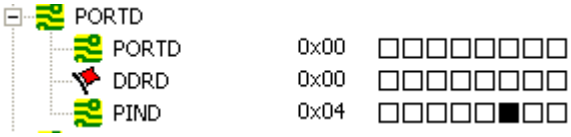

Рис. 6. Подача сигнала на вход INT0 (PIND2)

1.8. Выполните программу обработки прерывания. Для этого **нажмите** . Как только программа «зависнет» на одном месте, **нажмите** (пауза).

1.9. **Проверьте результаты работы**: программа должна была провести инверсию регистра PORTA.

Рис. 7. Регистр PORTA после выполнения подпрограммы.

1.10. **Проверьте, что при вводе нулевого значения во второй бит PIND прерывания не происходит.**

1.11. **Вновь введите единицу во второй бит PIND и проследите, чтобы регистр PORTD вновь перешел в нулевое состояние.**

2. Использование внешних прерываний.

2.1. **Напишите программу, которая по прерыванию INT0 прибавляла бы к содержимому порта A единицу, а по прерыванию INT1 вычитала бы единицу** (вход внешнего прерывания INT1 соответствует третьему биту регистра PIND). **Проверьте правильность выполнения программы.**

2.2. **Напишите программу, которая по прерыванию INT0 разрешала бы прерывание INT1 и запрещала прерывание INT0, а по прерыванию INT1 наоборот: разрешала INT0 и запрещала INT1. Прерывания должны генерироваться по срезу. Проверьте правильность выполнения программы**

3. Прерывания от таймера

3.1. Рассмотрим программу, использующую прерывания от таймера. Пусть каждые 100 тактов необходимо сложить числа введенные с портов A и B и результат вывести в порт C.

3.1.1. Введите следующую программу

```
#include <avr/io.h>
#include <avr/interrupt.h>    //Для доступа к функции sei()
#include <avr/signal.h>  //Для доступа к макросу SIGNAL

SIGNAL(SIG_OUTPUT_COMPARE0)  //Подпрограмма обработки прерывания по
{                                  //совпадению с регистром сравнения таймера T0
PORTC=PINA+PINB;
}
 int main (viod)
{
DDRC=0xFF;
TCCR0=0b00001001;
TIMSK=0b00000001;
OCR0=100;
sei();
}
```

3.1.2. **Проверьте правильность выполнения программы.**

3.2. **Напишите программу, которая каждые 250 микросекунд меняла бы содержимое порта A на противоположное (с нуля на 0xFF и наоборот)**.

Частота работы микроконтроллера=8МГЦ, значит длительность одного такта 0.25мкс. Для выполнения задания необходимо воспользоваться предделителем.

Контрольные вопросы

1. В чём отличие внутренних и внешних прерываний?
2. Чем отличаются функции INTERRUPT( ) или SIGNAL( ) ?
3. Зачем нужны регистры MCURC и GICR?
4. Что означает функция sei() ?