Прерывания в AVR

Пшеничников Артём P3107, лаба 6

Внешние аппаратные прерывания

- Ядро не занимается опросом пина и не тратит на это время, но как только напряжение сменяется – микроконтроллер получает сигнал
- Используется для детектирования кратковременных событий, импульсов, подсчёта их количества, обработки кнопок или других устройств ввода в нагруженном коде
- Возможность вывода МК из режима энергосбережения
- С помощью PCINT можно подключить прерывания на любом пине. Также есть INT, они проще в настройке, но их гораздо меньше
- Для прерывания нужна функция-обработчик
- В функции крайне не желательно совершать тяжёлые расчёты и долгие действия, это будет тормозить работу МК
- 4 режима RISING, FALLING, CHANGE, LOW
- Отключение прерываний ломает функции времени, шим и т. д.

МК / номер прерывания	INT 0	INT 1	INT 2	INT 3	INT 4	INT 5
ATmega 328/168 (Nano, UNO, Mini)	D2	D3	-	-	-	-
ATmega 32U4 (Leonardo, Micro)	D3	D2	D0	D1	D7	-
ATmega 2560 (Mega)	D2	D3	D21	D20	D19	D18

Прерывания по таймеру

- Можно использовать для генерации сигналов, измерения времени, параллельного выполнения задач, выполнения задач со строго заданной частотой
- Таймер – считает такты МК и выдаёт сигнал на ядро/дёргает пинами
- Два режима работы таймера, Overflow и Compare Match
- Можно использовать предделитель для изменения частоты таймера

```
TCCR0B |= (1 << CS02) | (1 << CS00); // Делитель 1024 (CS02=1, CS01=0, CS00=1)
```

- Обработчик прерывания таймера

```
ISR(TIMER0_COMPA_vect) {
    PORTB ^= (1 << PB5); // Инвертировать PB5
}
```

Таблица таймеров ATmega328p

Таймер	Разрядность	Частоты	Периоды	Выходы	Пин Arduino	Пин МК
Timer0	8 бит	61 Гц.. 1 МГц	16 384.. 1 мкс	CHANNEL_A	D6	PD6
				CHANNEL_B	D5	PD5
Timer1	16 бит	0.24 Гц.. 1 МГц	4 200 000.. 1 мкс	CHANNEL_A	D9	PB1
				CHANNEL_B	D10	PB2
Timer2	8 бит	61 Гц.. 1 МГц	16 384.. 1 мкс	CHANNEL_A	D11	PB3
				CHANNEL_B	D3	PD3

Прерывания АЦП

- Генерируется по завершении преобразования АЦП

```
ADCSRA |= (1 << ADIE);  // Разрешить прерывание АЦП
ADCSRA |= (1 << ADSC);  // Запустить преобразование
sei();
```

```
ISR(ADC_vect) {
    uint16_t adc_value = ADC; // Прочитать результат
}
```

Прерывания аналогового компаратора

- Срабатывает, когда напряжение на AIN0 превышает AIN1 (опорное)

```
ACSR |= (1 << ACIE);  // Разрешить прерывание компаратора
sei();
```

```
ISR(ANALOG_COMP_vect) {
    /* Код при срабатывании */
}
```

Прерывания UART

- USART_RX_vect, USART_TX_vect, USART_UDRE_vect

```
UCSR0B |= (1 << RXCIE0);  // Разрешить прерывание приема
sei();
```

```
ISR(USART_RX_vect) {
    char data = UDR0;  // Прочитать принятый байт
}
```

Прерывания TWI (I2C)

- Используется для обработки событий I2C (начало/стоп, передача/прием)

```
TWCR |= (1 << TWIE);  // Разрешить прерывание I2C
sei();
```

```
ISR(TWI_vect) {
    // Обработка событий I2C
}
```

Прерывания SPI

- Срабатывает после завершения передачи/приема данных по SPI

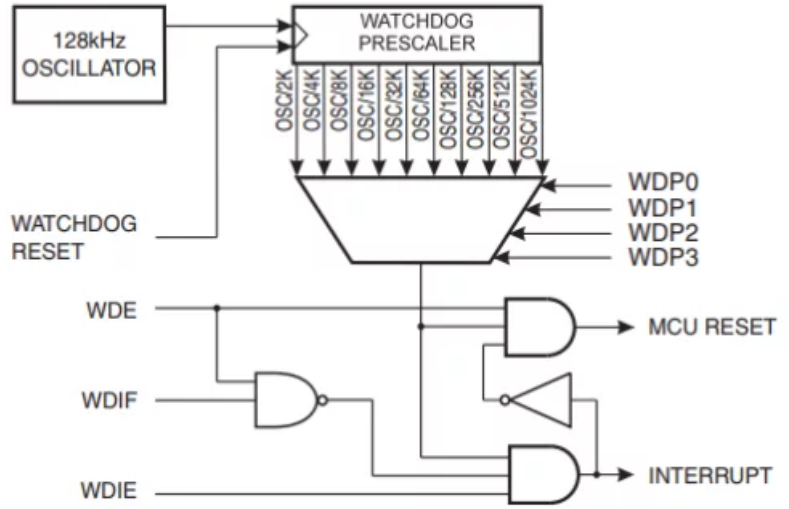
```
SPCR |= (1 << SPIE);  // Разрешить прерывание SPI
sei();
```

```
ISR(SPI_STC_vect) {
    uint8_t data = SPDR;  // Прочитать данные
}
```

Сторожевой таймер (watchdog)

- Позволяет перезагрузить МК при зависании вследствие программной или аппаратной ошибки
- Для сброса таймаута нужно внутри программы периодически вызывать инструкцию WDR
- Есть 10 предделителей частоты, WDT_PRESCALER_2-1024, 16 ms – 8 s соотв.
- Частота имеет достаточно высокий разброс с завода, +/- 10%
- При переполнении таймера может вызываться прерывание, функция обработчик, или инициироваться сброс МК (самый простой способ самоперезагрузки)
- Есть комбинированный режим, сначала вызов функции и перенастройка на перезагрузку по таймауту, если не получилось исправить программную ошибку, в следующий таймаут произойдет перезагрузка

Figure 10-7. Watchdog Timer



Приоритет прерываний

- Прерывания могут быть вложенными (включить sei(); в обработчике)
- Из неглубокого сна МК может вывести что угодно (прерываний довольно много, интерфейсы, ацп и т. д.)
- В AVR нет аппаратного приоритета, он задаётся положением вектора в таблице прерываний

1. RESET
2. INT0
3. IN1
4. TIMER2_COMPA
5. TIMER2_COMPB
6. ...