Лекция 8: “Периферийные устройства МК: ввод/вывод общего назначения, внешние прерывания”

Гончаров Олег Игоревич

Факультет вычислительной математики и кибернетики,
Московский государственный университет имени М.В. Ломоносова

2013

Периферийные устройства

Общий алгоритм работы с периферией

Выбор режима работы и взаимодействия с периферией осуществляется посредством чтения/записи в соответствующие порты. Некоторые библиотеки используют структуры и вызовы.

Общий алгоритм:

1. Прочитать документацию (!): описание работы модуля, документацию библиотек.
2. Инициализация:
   - выбор режима работы,
   - настройка режима работы выводов МК,
   - настройка прерываний,
   - включение устройства.
3. Взаимодействие с устройством: чтение/запись данных в регистры или биты.

Порты ввода/вывода общего назначения

Вывод и ввод квантованных по уровню сигналов.

Большинство выводов МК могут работать как входы и выходы для квантованных сигналов с двумя уровнями: логический ноль (низкий уровень напряжения) и логическая единица (высокий уровень)

Электрическая модель (типовые характеристики)

- В простейшем приближении выход являются переключателем: логической единице соответствует напряжение питания V_{cc}, нулю — нулевой уровень.
- В более сложной модели вместо переключателя транзисторы, поэтому логической единице соответствует $V_{cc} - V_D$, а нулю — V_D, где $V_D \approx 0.6$ В — напряжение диодного перехода.
- Входы оборудованы триггерами Шмидта, и схемами синхронизации, дающими задержку на один или несколько тактов. Высокий уровень — напряжение выше $0.6 V_{cc}$, низкий — меньше $0.2 V_{cc}$.
- Можно подключать подтягивающие резисторы: подтягивающий вверх (pull up) — на отключенном входе по умолчанию будет “1”, вниз — “0”.

AVR: пример работы с портами ввода/вывода

Программная модель

- Выводы объединены в порты GPIO, портам соответствуют регистры (порты ввода/вывода).
- Часть регистров используется для выбора режима: переключения ввод/вывод, подключения подтягивающих резисторов и т.д.
- Каждому выводу соответствует бит в регистре.

```
#include <avr/io.h>

/*#define _BV(x) (1<<(x)) */
/* init: configure pin 1 as output */
DDRA |= _BV(PA1);
/* drive pin 1 to high level */
PORTA |= _BV(PA1);
...
/* drive pin 1 to low level */
PORTA &= ~_BV(PA1);
```

```
#include <avr/io.h>

//#define _BV(x) (1<<(x))
/* init: configure pin 1 as input */
DDRA &= ~_BV(PA1);
/* init: enable pin pull-up */
PORTA |= _BV(PA1);
...
/* read pin state */
if (PINA & _BV(PA1)) { ... }
```

Внешние прерывания

Под внешними событиями обычно подразумевается изменения уровня на одном из выводов МК.

Модель функционирования

- Прерывание происходит с некоторой задержкой (обычно несколько циклов) после события.
- Задержка может зависеть от режима обработки сигнала: наличия/отсутствия шумоподавления.
- Прерывание может быть связано с одним выводом, либо с группой выводов. В последнем случае прерывание инициализируется любым из выводов.
- Может быть возможность выбрать тип события: падающий фронт, нарастающий фронт.
- Для некоторых МК (AVR) прерывания происходят даже, если вывод настроен как выход — программный способ вызова прерываний.

AVR: установка обработчика внешних прерываний

У МК AVR два типа внешних прерываний:

- INT0, INT1... жестко связаны с определенными выводами, можно настраивать тип события.
- PCINT0, PCINT1... связаны с группой выводов.

```
#include <avr/io.h>
#include <avr/interrupt.h>
/*interrupt handler*/
ISR(INT1_VECT) {
    /*code here*/
}
int main(void) {
    /*init external interrupt*/
    /*falling edges trigger INT1*/
    EICRA &= ~_BV(ISC11);
    /*clear INT1 flag*/
    EIFR &= ~_BV(INT1);
    /*enable INT1*/
    EIMSK |= _BV(INT1);
    /*global enable interrupts*/
    sei();
}
```

```
#include <avr/io.h>
#include <avr/interrupt.h>
/*interrupt handler*/
ISR(PCINT0_VECT, ISR_NOBLOCK) {
    /*code here*/
}
int main(void) {
    /*init external interrupt*/
    /*enable PCINT0 interrupt
        triggering by pin PB0*/
    PCMSK0 |= _BV(PCINT0);
    /*clear PCINT0 flag*/
    PCIFR &= ~_BV(PCIF0);
    /*enable INT1*/
    PCIER |= _BV(PCIE0);
    /*global enable interrupts*/
    sei();
}
```