Лекция 9: “Периферийные устройства МК: универсальный асинхронный приемопередатчик. Компиляция и сборка исполнимого модуля”

Гончаров Олег Игоревич

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

2013

Универсальный асинхронный приемопередатчик

Аппаратная поддержка приема и передачи данных по цифровым интерфейсам.

Минимальный набор функций:

- Синхронизация с входным потоком данных.
- Декодирование/кодирование кадров: формирование флагов кадра (ошибочный кадр, ошибка четности)
- Буферизация (обычно один байт), детектирование переполнения.
- Формирование прерываний: прием/передача кадра завершена, входной

Последовательный интерфейс: данные передаются побитово.
Формат кадра:

1 стартовый бит	5-9 бит данных	[1 бит четности]	1-2 стоповых бита

Для успешного обмена: формат кадров совпадает, расхождение скорости передачи не превышает $2 - 6$ %.

Универсальный асинхронный приемопередатчик

Передатчик

1 стартовый бит	5-9 бит данных	[1 бит четности]	1-2 стоповых бита

- ноль — высокий уровень, единица — низкий,
- стартовые биты — единица, стоповые — ноль.

Функционирование:

- Общий делитель, определяющий длительность бита (на 1 бит обычно приходится 16 тактов).
- **Передатчик**
  1. Копирование данных из буферного регистра в сдвиговый регистр, прерывание *UART data register empty* $\rightsquigarrow$.
  2. Формирование кадра: по тактовыми импульсами делителя передается стартовый бит, сдвигаются биты данных из сдвигового регистра, формируется бит четности, передаются стоповые биты.
  3. Прерывание *transmission complete* $\rightsquigarrow$.

Универсальный асинхронный приемопередатчик

Приемник

1 стартовый бит	5-9 бит данных	[1 бит четности]	1-2 стоповых бита

- ноль — высокий уровень, единица — низкий,
- стартовые биты — единица, стоповые — ноль.

Функционирование:

- Общий делитель, определяющий длительность бита (на 1 бит обычно приходится 16 тактов).
- **Приемник**
  1. Ожидание падающего фронта (стартовый бит) — синхронизация с началом кадра с точностью до такта.
  2. Декодирование кадра: значение очередного бита считывается в предполагаемой средине соответствующего временного интервала. Биты данных — в сдвиговый регистр.
  3. Проверка целостности кадра: установка флагов ошибки формата кадра, ошибки четности.
  4. Копирование данных в буфер передатчика, если его содержимое не прочитано, установка флага переполнения.
  5. Прерывание *receiving complete* $\rightsquigarrow$.

Универсальный асинхронный приемопередатчик

Дополнительные возможности:

- синхронный режим — синхронизация по внешнему сигналу,
- аппаратное управление потоком: сигналы CTS, RTS позволяют избежать потери кадров,
- режим межпроцессорной коммуникации: прерывания вызывают только адресные кадры специальной структуры,
- поддержка различны иных режимов работы: LIN, SPI, Infrared Serial, one wire (полудуплекс),
- DMA.

Порядок работы:

1. настроить скорость передачи (делитель частоты): 9600, 19200, 38400, 57600, 115200,
2. настроить формат кадра,
3. включить приемник и/или передатчик,
4. по прерывания или в цикле: изъятие данных из буфера приемника, запись данных в буфер передатчика.

Другие модули цифровых интерфейсов

Могут быть встроены модули: I2C, CAN, USB.

- Основные функции: синхронизация (если требуется), декодирование, буферизация, прерывания.
- Дополнительные функции в зависимости от протокола.

Поддержка прямого доступа к памяти (DMA).
Для канала DMA назначаются

- указатели источника и приемника,
- событие, по которому происходит копирование данных из источника в приемник,
- режим работы: как сдвигаются по завершению копирования указатели,
- прерывания (передан определенный объем данных).

AVR: Пример использования UART

Без использования прерываний

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

void uart0_init() {
    /* Set baudrate */
#define BAUD 57600
    /* Calculate UART BaudRate Register value */
#include <util/setbaud.h>
    UBRR0 = UBRR_VALUE;
    UCSR0A = 0;
#ifdef USE_2X
    UCSR0A |= _BV(U2X0);
#endif
    /* Set frame format */
    UCSR0C = _BV(USBS0) | _BV(UCSZ00) | _BV(UCSZ01);
    /* Enable receiver and transmitter */
    UCSR0B = _BV(TXEN0) | _BV(RXEN0);
}

int8_t uart0_rx_overun = 0;
static FILE uart0 = FDEV_SETUP_STREAM(uart0_putchar, uart0_getchar,
    _FDEV_SETUP_RW);
```

AVR: Пример использования UART

Без использования прерываний

```
static int uart0_putchar(char c, FILE *stream) {
    /* wait until transmitter buffer is empty */
    loop_until_bit_is_set(UCSR0A, UDRE0);
    UDR0 = c;
    return 0;
}
static int uart0_getchar(FILE *stream) {
    int c;
    /* wait until there is data in receiver buffer */
    loop_until_bit_is_set(UCSR0A, RXC0);
    uart0_rx_overun |= UCSR0A & _BV(DOR0);
    if (UCSR0A & (_BV(FE0)|_BV(UPE0))) {
        c = UDR0;
        return _FDEV_ERR;
    }
    else return UDR0;
}
#include <avr/pgmspace.h>
int main() {
    uart0_init(); stdout = &uart0; stdin = &uart0;
    puts_P(PSTR("Hello, world!"));
}
```

Создание образа памяти

Особенности МК:

- Отсутствие динамического загрузчика:
  - программа в неизменяемой памяти,
  - нет динамически загружаемых модулей,
  - все адреса известны заранее.
- Требуется самостоятельно выполнить подготовительные действия:
  - установка режимов работы МК,
  - инициализация данных в ОЗУ.

Основные этапы

1. работа компилятора →объектные модули,
2. сборка →собранный “исполнимый” модуль,
3. преобразование в формат загрузчика →образ памяти,
4. загрузка в память.

Компилирование

Вход:

- исходные файлы: *.h, *.c;
- заголовки библиотек: avr/io.h, stdio.h;
- инициализация: startup.c;
- аппартно-зависимые объявления:

```
#define MCU ATMEGA1280
#define F_CPU 16000000UL
```

компилятор: avr-gcc -DF_CPU=16000000UL -mmcpu=amega1280 -I. -I$INCDIRS main.c -o main.o

Создание образа памяти

объявления	исходные файлы	инициализация	заголовочные файлы библиотек
F_CPU=16000000UL	main.h main.c	startup.c	avr/io.h stdio.h

компилятор

объектные файлы	скрипт компоновщика	библиотеки
*.o	ldscript.ld	libc.a

сборщик

```
avr-gcc -mmcpu=amega1280 -L$LDDIRS main.o main.elf
```

создание образа памяти

```
avr-objcopy -O ihex main.elf main.hex
```

загрузка в память устройства

```
avrdude -p atmega1280 -P /dev/ttyUSB0 -c avrisp2 -U falsh:w:main.hex
```