Форум по AVR

- не работает программа из примера про пролистывания меню
- sinaprog не работает
- Пароль к архивам на сайте
- Пароль
- HDD и прерывания - доработка программы из статьи /node/220

Все записи

Главная страница » Блоги » i.orfanidi's blog

Отправка SMS с микроконтроллера

SMS | Микроконтроллер

Для отправки смс через мобильный телефон, применяя микроконтроллер есть несколько проблем:
1. Конвертирование исходного сообщения в готовое для отправки
2. Сам процесс передачи подготовленного сообщения в мобильный телефон.

Для работы с мобильным телефоном или **GSM модулем** необходимо использовать набор команд, эти команды называются **AT команды**. Синтаксис от телефона к телефону не сильно меняется, по сему можно ориентироваться что код будет работать на большинстве мобильных **GSM телефонов**.

1)Это набор макросов, они просты и реализуют работу AT командами модуля SIM300, которые понимают большинство GSM модулей и мобильных телефонов.

```
#define AT      for(uint8_t i=0; i<sizeof(at); i++)     usart0_write(at[i]);    //Trasmit AT.
#define ATE0    for(uint8_t i=0; i<sizeof(echo_off); i++)       usart0_write(echo_off[i]);
#define ATV1    for(uint8_t i=0; i<sizeof(atv_on); i++) usart0_write(atv_on[i]);
#define ATV0    for(uint8_t i=0; i<sizeof(atv_off); i++)        usart0_write(atv_off[i]);
#define ATD     for(uint8_t n=0; n<sizeof(atd); n++) usart0_write(atd[n]);

#define NUMBER0_UCS2    for(uint16_t n=ADDR_EEP_NUM0; n<(12+ADDR_EEP_NUM0);
n++) number0_unicode(n);
#define NUMBER1_UCS2    for(uint16_t n=ADDR_EEP_NUM1; n<(12+ADDR_EEP_NUM1);
n++) number1_unicode(n);
#define NUMBER2_UCS2    for(uint16_t n=ADDR_EEP_NUM2; n<(12+ADDR_EEP_NUM2);
n++) number2_unicode(n);
#define NUMBER3_UCS2    for(uint16_t n=ADDR_EEP_NUM3; n<(12+ADDR_EEP_NUM3);
n++) number3_unicode(n);
#define ENTER_ATD       for(uint8_t n=0; n<sizeof(atd_enter);
n++) usart0_write(atd_enter[n]);
#define SMS             for(uint8_t n=0; n<sizeof(sms_send);
n++) usart0_write(sms_send[n]);
#define CGMF1   for(uint8_t n=0; n<sizeof(format_text);
n++) usart0_write(format_text[n]);
#define ENTER_SMS       for(uint8_t n=0; n<sizeof(sms_enter);
n++) usart0_write(sms_enter[n]);

#define NUMBER0_GSM     for(uint16_t n=ADDR_EEP_NUM0;
n<(sizeof(number0)+ADDR_EEP_NUM0); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER1_GSM     for(uint16_t n=ADDR_EEP_NUM1;
n<(sizeof(number1)+ADDR_EEP_NUM1); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER2_GSM     for(uint16_t n=ADDR_EEP_NUM2;
n<(sizeof(number2)+ADDR_EEP_NUM2); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER3_GSM     for(uint16_t n=ADDR_EEP_NUM3;
n<(sizeof(number3)+ADDR_EEP_NUM3); n++) usart0_write(eeprom_read_byte(n));

#define END_SMS         for(uint8_t n=0; n<sizeof(sms_end);
n++) usart0_write(sms_end[n]); //Ctrl+Z
#define CSCS_UCS2       for(uint8_t n=0; n<sizeof(te_format_ucs2);
n++) usart0_write(te_format_ucs2[n]);

const char at[]={'A', 'T', 0x0D, 0x0A};
const char echo_off[]={'A','T','E', '0', 0x0D, 0x0A};
const char atv_off[]={'A','T','V', '0', 0x0D, 0x0A};
const char atv_on[]={'A','T','V', '1', 0x0D, 0x0A};
const char format_text[]={'A','T','+','C','M','G','F','=','1', 0x0D, 0x0A};

const char atd_enter[]={';', 0x0D, 0x0A};
const char sms_enter[]={'"', 0x0D, 0x0A};

const char sms_end[]={0x1A};
const char atd[4]={'A','T','D'};

const char sms_send[]={'A','T','+','C','M','G','S','=','"'};

const char te_format_ucs2[]={'A','T','+','C','S','C','S','=','"','U','C','S','2','"', 0x0D, 0x0A};
```

Здесь все просто, берем символы и отправляем по usart, за исключением, что усарт сделан как кольцевой буфер. Вообще это отдельная тема, которую надо затронуть, но о нем не сейчас, на чипеенабле есть статья и реализация его. Правда, эта функция моя, но все однотипно.
usart0_write('A'); – отправляет символ A в усарт, соответственно usart0_write(at[i]);- отправляет i-тый символ массива at.
Data=usart0_read(); - читает один байт данных из усарта.
len=usart0_rx_len(); - читает в пременную len количество принятых байт, если таковых нет функция вернет 0. Соответственно после отправки данных ответ мы ждем проверяя функцией usart0_rx_len();
если что то пришло, то возвращается количество принятых байт.
И две важные функции очистки буферов передатчика и приемника
usart0_clear_rx_buffer();
usart0_clear_tx_buffer();

Размеры этих буферов задаются в **USART.h** файле.
Также там задается формат посылки, скорость, четность и стоповые биты.
Про усарт все.

2) Переменные необходимые для отправки

```
char number0[]EEMEM={'+','7','1','1','1','1','1','1','1','1','1','1'};
char number1[]EEMEM={'+','7','2','2','2','2','2','2','2','2','2','2'};
char number2[]EEMEM={'+','7','3','3','3','3','3','3','3','3','3','3'};
```

Навига
- Авто
- Жур
  - Ж Ра
  - Ж Ра
- Ката
- data Ката
- Шаб прог
- Игры iPho
- Фор
- Блог

Раздел
- Data
- Жур
- Жур Ради
- Реги
- Фор
- Шаб прог
- Конт
- Ката
- Инст

Вход в

Имя по

Па

- Р

Пользо online
Сейчас *пользов гостей.*

```
char number3[]EEMEM={'+','7','4','4','4','4','4','4','4','4','4','4'};

char text_sms0[]EEMEM="АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯабвгдежзийклмнопрстуфхцчшщьыэюя000000";
// Текст нашей 1 смски.
char text_sms1[]EEMEM="ВБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯабвгдежзийклмнопрстуфхцчшщьыэюя111111";
// Текст нашей 2 смски.
char text_sms2[]EEMEM="СБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯабвгдежзийклмнопрстуфхцчшщьыэюя222222";
// Текст нашей 3 смски.
char text_sms3[]EEMEM="DБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯабвгдежзийклмнопрстуфхцчшщьыэюя333333";
// Текст нашей 4 смски.
```

Их тоже необходимо преобразовать в юникод. Номер преобразуется и отправляется в функции **numberx_unicode(n);** где n-соответствующая цифра в **ASCII** коде. x- номер телефонного номера из еепром Таким образом что бы преобразовать и передать весь номер нулевргo тел. нужно написать:

```
for(uint16_t n=ADDR_EEP_NUM0; n<(sizeof(number0)+ADDR_EEP_NUM0); n++) number0_unicode(n);
что и делает макрос NUMBER0_UCS2.
```

3) Преобразование и отправка номера телефона в функции происходит так:

```
void number0_unicode(uint16_t num)
{
        usart0_clear_tx_buffer();       //Clear Tx buffer.
        usart0_clear_rx_buffer();       //Clear Rx buffer.


if(eeprom_read_byte(num)=='+')
//Если в начале номера не цифра, а + (+7951xxxxxxx-например).
        {
                char plus[]="002B";
                for(uint8_t t1=0; t1<sizeof(plus); t1++) usart0_write(plus[t1]);
        }
                else
                {
                char not_plus[]="003";
                for(uint8_t t1=0; t1<sizeof(not_plus); t1++) usart0_write(not_plus[t1]);

                usart0_write(eeprom_read_byte(num));
                }
        _delay_ms(100);

}
```

Здесь все просто за исключением знака +, его символ в юникоде 002B. Номера читаются из еепром.
Я рассчитываю, что ты знаешь о юникоде, что там отправляется один символ 4 байтами.

4) Преобразование и отправка непосредственно смс раскладывается в два этапа. Первое это преобразование символа в юникод и второе это преобразование уже юникода в hex символ.
Первый этап реализуется в функции text_unicode(uint16_t word);
В программе пишем:

```
for(uint16_t i=0; i<len; i++) text_ucs2[i]=text_unicode(text_sms[i]);
```

где len соответственно длина нашей строки смс

text_ucs2[len]-это дву байтный массив строки смс кодированный юникодом.
Сама функция проста:

```
int text_unicode(uint16_t word)
{
        if(word>0x00BF)
        {
                word+=0x0350;
        }
return word;
}
```

Почему прибавляем 0x0350, так кириллица начинается с **0x0410** номера, а символ «А» это номер 0x0C. 0x0C+0x0350=0x0410, далее все символы идут по порядку, кроме буквы Ё, она к сожалению осталась в не удел, а жаль.

5) Второй этап, это преобразование юникода в хекс юникод.
Делаем это в функции text_unicode_hex(uint16_t word); делаем и отправляем один символ, а на деле отправляется 4 байта.

```
        for(uint16_t i=0; i<len; i++) text_unicode_hex(text_ucs2[i]);
```

Теперь что в самой функции:

```
void text_unicode_hex(uint16_t word)
{
        char data_tx[4];
        uint8_t byte;

                byte=word>>12;
                byte=byte+0x30;
                data_tx[0]=byte;

                byte=word>>8;
                byte&=0x0F;
                byte=byte+0x30;
                data_tx[1]=byte;

                byte=word>>4;
                byte&=0x0F;
                byte=byte+0x30;
                data_tx[2]=byte;

                byte=word;
                byte&=0x0F;
```

```
                if(byte<0x0A)
                {
                        byte=byte+0x30;
                        data_tx[3]=byte;
                }
                        else
                        {
                        byte=byte+0x37;
                        data_tx[3]=byte;
                        }

        for(uint8_t tx=0; tx<4; tx++) usart0_write(data_tx[tx]);
}
```

Здесь мы передаем символы из массива юникода символов смс **text_ucs2[i]** полученных ранее из функции **text_unicode(text_sms[i]);**

Далее просто преобразуем их в строку data_tx[tx] и отправляем в порт усарт.

6) Теперь о всей программе расскажу.
Настраиваем наш модуль на скорость, у меня почему то работает только на 115200.

```
//Функция at_ok() вернет 0 в случае правильной настройки модуля и 1
//если не получилась и нужно попробовать ещё.
while(at_ok())
        {
        at_ok();        //Шлем OK

        ATV0;   //Отключаем словесный формат ответа.
//Теперь вместо OK будет проходить 0(в ASCII конечно)
        _delay_ms(1000); // Ждем.
        ATE0;   //Отключаем эхо чтобы сами себя не слушали

        _delay_ms(1000); //И опять ждем.
        }


CGMF1;  //Устанавливаем режим PDU.

while(ok());    //Ждем OK.

CSCS_UCS2;      //Посылаем команду AT+CSCS="UCS2"
                //Она настраивает на юникод.
while(ok());    //Ждем OK.

//Далее можно все загнать в отдельную функцию

//Здесь первый параметр номер телефона, а второй текст смски.//
//И номеров и текстов смс четыре начиная с нуля.//

int sms_tx(uint16_t num_tel, uint16_t num_sms)
{
        usart0_clear_tx_buffer(); //Clear Tx buffer.
        usart0_clear_rx_buffer(); //Clear Rx buffer.

        CMGF1;  //Макрос устанавливает режим PDU.

        uint8_t status=1;
        while(status)
        {
                status=ok();
                if(status=='4') return 0;
        }

        SMS; //Говорим модему, что хотим послать SMS.

        _delay_ms(100);
        if(num_tel==0) NUMBER0_UCS2; //На номер #0.
        if(num_tel==1) NUMBER1_UCS2; //На номер #1.
        if(num_tel==2) NUMBER2_UCS2; //На номер #2.
        if(num_tel==3) NUMBER3_UCS2; //На номер #3.

        ENTER_SMS; //После этой команды вводим текст смски.

        while(!(usart0_rx_len())); //Ждем приглашение от модуля.
        while(usart0_read()!='>'); //Как только прийдет ">" можно будет вводить текст смс.

        usart0_clear_tx_buffer(); //Clear Tx buffer.

        if(num_sms==0)
        {
                uint8_t len=0;

                for(uint16_t i=ADDR_EEP_SMS0; i<(0x47+ADDR_EEP_SMS0); i++)
                {
                if(eeprom_read_byte(i)) len++;
                }

                char text_sms[len];
                uint8_t ii=0;
                for(uint16_t i=ADDR_EEP_SMS0; i<(len+ADDR_EEP_SMS0); i++)
                {
                text_sms[ii]=eeprom_read_byte(i);
                ii++;
                }

                uint16_t text_ucs2[len];

                for(uint16_t i=0; i<len; i++) text_ucs2[i]=text_unicode(text_sms[i]);
                for(uint16_t i=0; i<len; i++) text_unicode_hex(text_ucs2[i]);
        }

        if(num_sms==1)
```

```
	{
		uint8_t len=0;

		for(uint16_t i=ADDR_EEP_SMS1; i<(0x47+ADDR_EEP_SMS1); i++)
		{
		if(eeprom_read_byte(i)) len++;
		}

		char text_sms[len];
		uint8_t ii=0;
		for(uint16_t i=ADDR_EEP_SMS1; i<(len+ADDR_EEP_SMS1); i++)
		{
		text_sms[ii]=eeprom_read_byte(i);
		ii++;
		}

		uint16_t text_ucs2[len];

		for(uint16_t i=0; i<len; i++) text_ucs2[i]=text_unicode(text_sms[i]);
		for(uint16_t i=0; i<len; i++) text_unicode_hex(text_ucs2[i]);

	}

	if(num_sms==2)
	{
		uint8_t len=0;

		for(uint16_t i=ADDR_EEP_SMS2; i<(0x47+ADDR_EEP_SMS2); i++)
		{
		if(eeprom_read_byte(i)) len++;
		}

		char text_sms[len];
		uint8_t ii=0;
		for(uint16_t i=ADDR_EEP_SMS2; i<(len+ADDR_EEP_SMS2); i++)
		{
		text_sms[ii]=eeprom_read_byte(i);
		ii++;
		}

		uint16_t text_ucs2[len];

		for(uint16_t i=0; i<len; i++) text_ucs2[i]=text_unicode(text_sms[i]);
		for(uint16_t i=0; i<len; i++) text_unicode_hex(text_ucs2[i]);
	}

	if(num_sms==3)
	{
		uint8_t len=0;

		for(uint16_t i=ADDR_EEP_SMS3; i<(0x47+ADDR_EEP_SMS3); i++)
		{
		if(eeprom_read_byte(i)) len++;
		}

		char text_sms[len];
		uint8_t ii=0;
		for(uint16_t i=ADDR_EEP_SMS3; i<(len+ADDR_EEP_SMS3); i++)
		{
		text_sms[ii]=eeprom_read_byte(i);
		ii++;
		}

		uint16_t text_ucs2[len];

		for(uint16_t i=0; i<len; i++) text_ucs2[i]=text_unicode(text_sms[i]);
		for(uint16_t i=0; i<len; i++) text_unicode_hex(text_ucs2[i]);
	}

	END_SMS; //Макрос завершающий отправку смс.

	while(ok());
	_delay_ms(100); //Ждем ОК.

return 1;
}
```

В главном цикле можно написать проверку какого либо условия например для сигнализации проверить пин.

```
while(1)
{
	if(!(PIN_KEY & (1<<KEY0)))
	{
//Здесь нужно отключить всякие таймеры и внешние прерывания, если они были до
//этого. !!! Заметь не запретить прерывания а просто исключить те вещи которые их
//могут создать.//
sms_tx(0,0);	//Шлем смс на первый номер.
	for(uint8_t p=0; p<5; p++) _delay_ms(1000);
sms_tx(0,1);	//Шлем смс на второй номер.
	for(uint8_t p=0; p<5; p++) _delay_ms(1000);
sms_tx(0,2);	//Шлем смс на третий номер.
	for(uint8_t p=0; p<5; p++) _delay_ms(1000);
sms_tx(0,3);	//Шлем смс на четвертый номер.
	for(uint8_t p=0; p<5; p++) _delay_ms(1000);

	for(uint8_t p=0; p<10; p++) _delay_ms(1000);

	usart0_clear_rx_buffer();
	usart0_clear_tx_buffer();

	//А здесь можно опять включить таймеры, и др. вещи создающею прерывания//
	}
```

По поводу номеров, их можно подправить непосредственно в еепроме, если цифр в номере больше или меньше, то надо уже править эти макросы:

по смс

```
#define NUMBER0_UCS2    for(uint16_t n=ADDR_EEP_NUM0; n<(12+ADDR_EEP_NUM0);
n++) number0_unicode(n);
#define NUMBER1_UCS2    for(uint16_t n=ADDR_EEP_NUM1; n<(12+ADDR_EEP_NUM1);
n++) number1_unicode(n);
#define NUMBER2_UCS2    for(uint16_t n=ADDR_EEP_NUM2; n<(12+ADDR_EEP_NUM2);
n++) number2_unicode(n);
#define NUMBER3_UCS2    for(uint16_t n=ADDR_EEP_NUM3; n<(12+ADDR_EEP_NUM3);
n++) number3_unicode(n);
```

по дозвону

```
#define NUMBER0_GSM     for(uint16_t n=ADDR_EEP_NUM0; n<(12+ADDR_EEP_NUM0); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER1_GSM     for(uint16_t n=ADDR_EEP_NUM1; n<(12+ADDR_EEP_NUM1); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER2_GSM     for(uint16_t n=ADDR_EEP_NUM2; n<(12+ADDR_EEP_NUM2); n++) usart0_write(eeprom_read_byte(n));
#define NUMBER3_GSM     for(uint16_t n=ADDR_EEP_NUM3; n<(12+ADDR_EEP_NUM3); n++) usart0_write(eeprom_read_byte(n));
```

Цифра 12 это количество цифр в мобильном номере.

Адреса в тел. номеров и текстов смс написаны в этих дефайнах:

```
        #define ADDR_EEP_NUM0 0x0008
        #define ADDR_EEP_NUM1 0x0014
        #define ADDR_EEP_NUM2 0x0020
        #define ADDR_EEP_NUM3 0x002C

        ////////////////////////////////////////////////

        #define ADDR_EEP_SMS0 0x0038
        #define ADDR_EEP_SMS1 0x007F
        #define ADDR_EEP_SMS2 0x00C6
        #define ADDR_EEP_SMS3 0x010D
```

Их тоже нужно править, если вдруг сократишь или увеличишь количество цифр в номере, но там ничего сложного нет, будут вопросы пиши.

Текст смс должен быть строго 70 символов (0x010D-0x00C6=71 но последний символ не учитывается), недостающие символы необходимо заменить пробелами. Можно было заморочится на автоматическое вычисление размера смс, но адреса в еепроме будут плавать и дефайнами уже не обойтись, нужен будет алгоритм, а это мне не нужно проще пробелами замостить смс до 70 символа.

Так же к статье прилагается файл проекта с исходниками, принцип работы проекта:
программа сканирует один пин порта микроконтроллера и если происходит сработка то шлет четыре разных смс на четыре разных номера.

Скачать файл проекта

» **блог i.orfanidi'a** | **Войдите** или **зарегистрируйтесь**, чтобы получить возможность отправлять комментарии