PDU 短信格式解析

1.短信格式的编码主要是有两种一个是 text 格式的，明文不用解码一看就懂，一个是 PDU 格式的，国内的都是的 PDU 格式的。以下分析的都是 PDU 格式数据包。

2.PDU 格式的数据包，发送的数据包和接收的数据包数据段是不一样的。相同是一，他们都是把数据变成字符 0123456789ABCDEF。二，都是 2 个字符算一个数，除了电码号码其的数都是 16 进制的。

3.接收数据包，内容为：“123,测试,test。”,来自电话号 18701634332
串口 AT 回复：
+CMTI: "MT",37
再发送指令 读取短信
AT+CMGR=37
串口 AT 回复：
+CMGR: 0,,51
0891683108100005F0040D91688107614333F2000801408211544423180031003200330002C6D4B8BD5002C007400650073007300743002
OK

中间蓝色的就是接收的数据包了。

字符	描述
08	SMSC 短信中心号码段的长度，不包含 08 这两个字符,但包含 91
91	Type-of-address of the SMSC. 短信中心号的类型一般 0x91 是国际型，0xA1 是国内型，遇到国际型的解析的时候就在前面加个+号，大家一般都是这么干的
68 31 08 10 00 05 F0	这个是短信中心号码，每两字符互换就是了 8613080100500，最后一位 F 是表示中心号码的结束符，这个 F 只有在电话号码是奇数位的时候才有。
04	First octet of this SMS-DELIVER message.意思很多可以不管。
0D	发信人的号码长度 0xD = 13 位，不包括 0D 91.
91	Type-of-address 类型和上面一样 0x91 国际型
68 81 07 61 43 33 F2	发信人号码和上面一样解析 8618701634332 因为是国际型的一般显示为+8618701634332
00	TP-PID. Protocol identifier.协议头，可以不管

<table>
<tr><td>08</td><td>TP-DCS Data coding scheme,这个很重要，描述了下面的短信内容是以下三种格式中的那一种 1.8 位 ASCII 编码 2.7 位压缩编码 3.Unicode 16 位编码

当最上面的 4-7 位为 =>00xx 时主要上中间的下面的第 3 和第 4 位，也就是 bit2 和 bit3。这里是 0x08----->b1000 表示 Unicode 16 位编码,国内一般就这个。有些手机会把纯英文的用 7 位编码发送.

<table><tr><td>Bit 3</td><td>Bit 2</td><td>编码格式</td></tr><tr><td>0</td><td>0</td><td>7 位压缩编码</td></tr><tr><td>0</td><td>1</td><td>8 位 ASCII 编码</td></tr><tr><td>1</td><td>0</td><td>Unicode 16 位编码</td></tr><tr><td>1</td><td>1</td><td>保留</td></tr></table></td></tr>
<tr><td>01 40 82 11 54 44 23</td><td>这个就是发信时间 Time stamp (semi-octets)，按字节互换解析，基下时间是已经按时区转好了的。时间就是:2010-4-28 11:45:44

<table><tr><td>数据</td><td>字符个数</td><td>Description</td></tr><tr><td>01</td><td>2</td><td>年份后两位 2010 年</td></tr><tr><td>40</td><td>2</td><td>月份 4 月</td></tr><tr><td>82</td><td>2</td><td>日期 28 日</td></tr><tr><td>11</td><td>2</td><td>小时 11 点</td></tr><tr><td>54</td><td>2</td><td>分 45 分</td></tr><tr><td>44</td><td>2</td><td>秒 44 秒</td></tr><tr><td>23</td><td>2</td><td>GMT. 时区 32/4 =8 小时</td></tr></table></td></tr>
<tr><td>18</td><td>短信内容字符的长度对于 Unicode 编码，字符为 12 个
1.Unicode 的编码的数据来说 0x18 = 24 表是一共有 24/2 =12 个 Unicode 字符"123,测试,test。"，实际的数据是 24*2=48 个 ASCII 字符组成
2.对于 7Bit 的为说一共是 24 个字符，实际的数据是 24*7/8 =21 (向上取整) 21 * 2 =42;个 ASCII 字符
3.对于 8 位的编码的来说也是 24 个字符。实际的数据是 48 个 ASCII 字符组成</td></tr>
<tr><td>003100320033002C6D4B8BD5002C00740065007300743002</td><td>短信数据 0031 这 4 个 ASCII 字符表示 Unicode 编码 0x0031,
如果是 7 位的编码是用 7 位表示字符，但是用 8 位</td></tr>
</table>

	来存贮。很复杂下面是 e 文的例子 Message "hellohello" 。

3.发送 PDU 格式消息

例如：向手机 18701634332 发送 内容为：“123,测试,test。”的短信

首先发送 AT 指令
AT+CMGS=38
串口 AT 回复：
>
发送以下 ASCII 字符串就是 PDU 格式的短信数据包
0031000BA18107614333F20008A718003100320033002C6D4B8BD5002C00740065007300743002
再发送
0x1a(Ctrl-z 的 ASCII 码，这个也可以和上面的数据写在一起发送)

串口 AT 回复：
+CMGS: 221
OK

字符	描述
00	SMSC 短信中心地址,写成 0 表示从电话内部取中心号码 ,如果要自己写就按照上面的发送短信的 SMSC 段来写。分长度，类型，号码，三个段。
31	First octet of the SMS-SUBMIT message. 可以不管直接写为 01
00	也可以不管，写为 00
0B	收信人的电话号长度手机 11 位，如果是发给飞信的会大于 11 位
A1	Type-of-Address. 表示国内型
8107614333F2	手机号码 18701634332
00	TP-PID. Protocol identifier 也在不管写为 00
08	TP-DCS. Data coding scheme.采用 Unicode 编码，支持汉字。
A7	TP-Validity-Period. 设置短信有效期 VP 4 天有效
18	数据的长度 0x18=24
003100320033002C6D4B8BD5002C00740065007300743002	Unicode 编码的数据内容为，"123,测试,test。"，和上面收的一样

注意：
1.在写发送指令的时候 38 就是 PDU 包除了 SMSC 段之后的长度是 38*2=76 个 ASCII 字符
AT+CMGS=38//表示后面有 76 个 ASCII 字符
2.这里的 38 就是不包含前面的上表中黄色的 00 这两个字符。

3.超长短信
本来超长短信是有规定格式的，不过在中国短信延时比较严重，显得没有什么意义了。超长短信的也是符合上面的格式的，只是要短信的内容段加了固定的头。表示这是条超长短信.
("050003") // 6-bit codec header
("060804") // 7-bit codec header

以下是的代码是直接将超长短信分为两条短信，没有做结合。在西门子 MC75i 上测试过

```
/************************************************
* 文件名称：SMS.cpp
* 摘　　要：用于短信数据的编解码
* 当前版本：v1.0
* 作　　者：胡丰凯 hufengkai@foxmail.com
* 创建日期：2010 年 4 月 21 日
* 修改纪录：
*
*
************************************************/
//-------------------------------------------------------------------
--------
//--Include file
#include "sms.h"
#include "Debug.h"
#define LONG_SMS_HEAD_6BIT ("050003") // 6-bit codec header
#define LONG_SMS_HEAD_7BIT ("060804") // 7-bit codec header

//-------------------------------------------------------------------
--------
//--静态函数
static bool ReadByte(string& s_buf,int start_pos,unsigned char &out);
static bool ReadString(string& s_buf,int start_pos,int len ,string &out);
static bool Gsm7bitDecoding(string& s_context);
static bool Gsm8bitDecoding(string& s_context);
static bool TimeDecode(string & s_in);
static bool UCS2Decoding(string& s_context);
static bool UCS2Encoding(string& s_context_out,wstring& s_context_in);
static void ByteToString(int i_value,char *buffer,int buf_len);
```

```
//-------------------------------------------------------------------
--------
//--函数
/*++
isLongSms
     判断是否是长短信
参数:
const char * pUD: 短信头地址
返回:
短信头
没有为0
--*/
static int isLongSms(const char * pUD)
{

int ret;
if (0==strncmp(LONG_SMS_HEAD_6BIT, pUD, strlen(LONG_SMS_HEAD_6BIT)))
{
     ret = 6 * 2;
}
else if (0==strncmp(LONG_SMS_HEAD_7BIT, pUD,
strlen(LONG_SMS_HEAD_7BIT)))
{
     ret = 7 * 2;
}
else {
     ret = 0;
}

return ret;
}
/*++
ReadByte
读16进制的数据
参数:
[IN]string& s_buf 输入缓冲
[IN]int start_pos 读字节启始位置
[OUT]unsigned char &out 输出
返回:
成功为TRUE 否则false
--*/
static bool ReadByte(string& s_buf,int start_pos,unsigned char &out)
{
string s_byte = s_buf.substr(start_pos,2);
```

```
int len = s_byte.length();
if(len != 2)
    return false;
out = (unsigned char)strtol(s_byte.c_str(),NULL,16);
return true;
}
/*++
ReadString
读错位相读字符，主要电话号码
参数:
[IN]string& s_buf 输入缓冲
[IN]int start_pos 读字节启始位置
[IN]int len 要读取字符串的长度
[OUT]string &out 输出
返回:
成功为 TRUE 否则 false
--*/
static bool ReadString(string& s_buf,int start_pos,int len ,string &out)
{
string s_string = s_buf.substr(start_pos,len);
int r_len = s_string.length();
int i;
out = s_string;
if(len != r_len)
    return false;
for(i=0;i<len;i++)
{
    if(0== (i%2))
     {
       out[i+1] = s_string[i];
     }
    else
     {
       out[i-1] = s_string[i];
     }
}
if(out[len-1] == 'F')
{
     out = out.substr(0,len-1);
}
return true;
}
/*++
TimeDecode
```

```
原来格式
100423133554 转后
> 2010-04-23 13:35:54
参数：
[IN,OUT]string & s_in 输入缓冲,对时间
返回：
成功为 TRUE 否则 false
--*/
static bool TimeDecode(string & s_in)
{
string s_time = "20";
const string s_1 = "-";
const string s_2 = " ";
const string s_3 = ":";
s_time += s_in;
s_time.insert(4,s_1);
s_time.insert(7,s_1);
s_time.insert(10,s_2);
s_time.insert(13,s_3);
s_time.insert(16,s_3);
s_in = s_time.substr(0,s_time.length()-2);
return true;
}
/*++
DecodeSMS
解码原始的 PDU 数据包.
参数：
[IN]string s_in_pdu 原始 PDU 数据包
[OUT]string& number 发信人号码
[OUT]string& s_time 发信时间
[OUT]string& s_context 发信内容
返回：
成功为 TRUE 否则 false
--*/
bool DecodeSMS(string s_in_pdu, string& s_number,string& s_time,string&
s_context)
{
CPDUReceive l_pdu;
int len =s_in_pdu.length();
bool res = false;
unsigned char smsc_len = 0;
unsigned char address_len = 0;
int offset = 0;
int encode_type = 0;
```

```
//-----------------------------------
//--读 SMSC 服务中心号码
res = ReadByte(s_in_pdu,0,smsc_len);
if(!res)
    return false;
res = ReadByte(s_in_pdu,2,l_pdu.i_smsc_type);
if(!res)
     return false;
len = (smsc_len - 1)*2;
res = ReadString(s_in_pdu,4,len,l_pdu.s_smsc);
if(!res)
    return false;
if(l_pdu.i_smsc_type == 0x91)//国际型
{
     l_pdu.s_smsc ="+" +l_pdu.s_smsc;
}

//-----------------------------------
//--读 FO 第一个字节
offset = (smsc_len + 1)*2;
res = ReadByte(s_in_pdu,offset,l_pdu.i_fo);
if(!res)
    return false;
//-----------------------------------
//--读发件电话号长度
offset += 2;
res = ReadByte(s_in_pdu,offset,address_len);
if(!res)
    return false;
//-----------------------------------
//--读发件电话号类型
offset += 2;
res = ReadByte(s_in_pdu,offset,l_pdu.i_address_type);
if(!res)
    return false;
//-----------------------------------
//--读电话号码
offset += 2;
if(0!=(address_len%2))
     len = address_len+1;
else
     len = address_len;
res = ReadString(s_in_pdu,offset,len,l_pdu.s_dest_address);
```

```
if(!res)
     return false;
if(l_pdu.i_address_type == 0x91)//国际型
{
     l_pdu.s_dest_address = "+" + l_pdu.s_dest_address;
}
//-----------------------------------
//--协议表示
offset += len;
res = ReadByte(s_in_pdu,offset,l_pdu.i_pid);
if(!res)
    return false;
//-----------------------------------
//--数据编码标准
offset += 2;
res = ReadByte(s_in_pdu,offset,l_pdu.i_dsc);
if(!res)
    return false;
//-----------------------------------
//--发信时间
offset += 2;
res = ReadString(s_in_pdu,offset,14,l_pdu.s_time);
if(!res)
    return false;

//-----------------------------------
//--短信长度
offset += 14;
res = ReadByte(s_in_pdu,offset,l_pdu.i_context_len);
if(!res)
    return false;

//-----------------------------------
//--内容
offset += 2;
l_pdu.s_context = s_in_pdu.substr(offset,s_in_pdu.length()-offset);

TimeDecode(l_pdu.s_time);
s_number = l_pdu.s_dest_address;
s_time = l_pdu.s_time;
s_context = l_pdu.s_context;
if(0 != (l_pdu.i_dsc & 0x60))
    return false;
```

```
encode_type = (l_pdu.i_dsc>>2)&0x3;
offset = isLongSms(s_context.c_str());
if(offset > 0)
{
     s_context = s_context.substr(offset,s_context.length()-offset);
}
switch(encode_type)
{
    case 0:// 7bit 编码
       Gsm7bitDecoding(s_context);
      break;
    case 1://8 bit
       Gsm8bitDecoding(s_context);
      break;
    case 2:// 16 bit 编码
       UCS2Decoding(s_context);
      break;
}
return true;
}

/*++
Gsm7bitDecoding
7bit 包的解码
参数:
[IN,OUT]string& s_context 短信内容
返回:
成功为 TRUE 否则 false
--*/
static bool Gsm7bitDecoding(string& s_context)
{
unsigned char src[MAX_UD];
char dst[MAX_7UD];
char c_temp[3];
int len = 0,i=0;
       int srcLength = 0;
       int dstLength = 0;
       int a, b, k;
string s_out;
memset(src,0,sizeof(src));
memset(dst,0,sizeof(dst));
len = s_context.length();
if(len > MAX_7UD || len < 1)
    return false;
```

```
//--------------------------------------
//--转为数据
for(i = 0 ;i<len/2;i ++)
{
     ReadByte(s_context,i*2,src[i]);
}

srcLength = len/2;
dstLength = srcLength * 8/7;
       for (a = 0, b = 0; b < srcLength ; a++, b++)
       {
     k = a % 8;
    if (a > 0)
     {
       dst[a] = (unsigned char)(((src[b] << k) & 0x7f) | (src[b - 1] >>
8 - k));
     }
     else
     {
       dst[a] = (unsigned char)(src[b] & 0x7f);
     }
     if (k == 7 && a > 0)
     {
       dst[++a] = (unsigned char)(src[b] & 0x7f);
     }
       }
s_out = dst;
s_context = s_out;
return true;
}
void ByteToString(int i_value,char *buffer,int buf_len)
{
const char s_num[17] ="0123456789ABCDEF";
memset(buffer,0,buf_len);
buffer[0] = s_num[i_value/16];
buffer[1] = s_num[i_value%16];


}
/*++
UCS2Encoding
Unicode 数据编码
参数:
[OUT]string& s_context 短信内容
```

```
返回:
成功为 TRUE 否则 false
--*/
static bool UCS2Encoding(string& s_context_out,wstring& s_context_in)
{
     // 高低字节对调，输出
int i = 0,len = 0,temp;
char buffer[32];
len = s_context_in.length();

if(len > MAX_UD/2)
    return false;
for( i=0; i<len; i++)
{
     temp = (s_context_in[i] >> 8); // 先输出高位字节
     ByteToString(temp,buffer,32);
     s_context_out += buffer;
     temp = (s_context_in[i] & 0xff);    // 后输出低位字节
     ByteToString(temp,buffer,32);
     s_context_out += buffer;
}
return true;

}
/*++
UCS2Decoding
Unicode 数据解码
参数:
[IN,OUT]string& s_context 短信内容
返回:
成功为 TRUE 否则 false
--*/
static bool UCS2Decoding(string& s_context)
{
int nSrcLength,i;    // UNICODE 宽字符数目
wchar_t wchar[MAX_UD]; // UNICODE 串缓冲区
unsigned char src[MAX_UD];
unsigned char * pSrc = NULL;
wstring ws_out;
nSrcLength = s_context.length();
memset(wchar,0,sizeof(wchar));
memset(src,0,sizeof(src));

if( nSrcLength > (MAX_UD*2) || nSrcLength< 1)
```

```
    return false;
//--------------------------------------
//--转为数据
for(i = 0 ;i<nSrcLength/2;i ++)
{
     ReadByte(s_context,i*2,src[i]);
}
pSrc = (unsigned char * )src;
// 高低字节对调，拼成 UNICODE
for(int i=0; i<nSrcLength/2; i++)
{
     wchar[i] = *pSrc++ << 8; // 先高位字节
     wchar[i] |= *pSrc++;    // 后低位字节
}
ws_out = wchar;
s_context = ws2s(ws_out);
return true;
}
/*++
Gsm8bitDecoding
8 位 数据解码
参数:
[IN,OUT]string& s_context 短信内容
返回:
成功为 TRUE 否则 false
--*/
static bool Gsm8bitDecoding(string& s_context)
{
return true;
}
/*++
FormatNumber
调整个 num 为反序
参数:
[IN,OUT]string &s_num 电话号码
返回:
成功为 TRUE 否则 false
--*/
static void FormatNumber(string &s_num)
{
int len = s_num.length();
int i = 0;
string s_org ;
if(len %2 != 0)
```

```
{
      s_num += "F";
      len ++;
}
s_org = s_num;
for(i = 0;i<len;i++)
{
if(i%2 == 0)
        s_num[i] = s_org[i+1];
      else
        s_num[i] = s_org[i-1];
}

}
void ByteToTwoString(int data,char *buffer,int buf_len)
{
memset(buffer,0,buf_len);
if(data < 16)
{
      buffer[0]='0';
      itoa(data,buffer+1,16);
}
else
{
      itoa(data,buffer,16);
}
}
/*++
EncodeSMS
解码原始的 PDU 数据包.
参数:
[OUT]string &s_in_pdu 原始 PDU 数据包
[IN]string &smsc 短信中心号码
[IN]string& number 发信人号码
[IN]wstring& s_context 发信内容
返回:
0 表示出错否则返回要发送的长度
--*/
int EncodeSMS(string &s_in_pdu,string &smsc, string& s_number,wstring&
s_context)
{
const string s_FOTP = "3100";//F0,前段 TP-Message-Reference
const string s_International_type = "91";
const string s_national_type = "A1";
```

```
const string s_context_type ="0008A7";//pid 0, dsc 08 ---Unicode 模式 VP
4 天有效
const string s_end_flag = "\x01a";//Ctrl-z 为 0x1a
int len =0,smcc_len = 0;
char buffer[32];
string s_temp;
string s_ucs2_context;
string::size_type pos = string::npos;
wstring s_out;
int res = 0;
//-------------------------------------
//-SMCS 段
pos = smsc.find('+');
if(string::npos != pos)//国际型
{
     s_temp = smsc.substr(1,smsc.length()-1);
     FormatNumber(s_temp);
     smcc_len = s_temp.length();
    if(smcc_len> 255)
      return 0;
     memset(buffer,0,sizeof(buffer));
     len = smcc_len/2 +1;
     ByteToString(len,buffer,32);
     s_in_pdu = buffer+s_International_type + s_temp;
}
else
{
     smcc_len = smsc.length();
    if(smcc_len !=0 )
     {
      if(smcc_len> 255)
        return 0;
       memset(buffer,0,sizeof(buffer));
       len = smcc_len/2 +1;
       ByteToString(len,buffer,32);
       FormatNumber(smsc);
       s_in_pdu = buffer+s_national_type + smsc;
     }
    else
     {
       s_in_pdu = "00";//从电话中读取中心号码
       len = 0;
     }
}
```

```
smcc_len = len+1;
s_in_pdu += s_FOTP;
//-------------------------------------
//--收件人电话段
pos = s_number.find('+');
if(string::npos != pos)//国际型
{
     s_number = s_number.substr(1,s_number.length()-1);
     len = s_number.length();
    if(len> 255)
      return 0;
     ByteToString(len,buffer,32);
     FormatNumber(s_number);
     s_in_pdu = s_in_pdu + buffer+s_International_type + s_number;
}
else
{
     len = s_number.length();
    if(len> 255)
      return 0;
     ByteToString(len,buffer,32);
     FormatNumber(s_number);
     s_in_pdu = s_in_pdu + buffer+s_national_type + s_number;
}

//-------------------------------------
//--短信设置段
s_in_pdu += s_context_type;
UCS2Encoding(s_ucs2_context,s_context);
len = s_ucs2_context.length()/2;
if(len> 255)
    return 0;
ByteToString(len,buffer,32);
s_in_pdu = s_in_pdu + buffer + s_ucs2_context;
res = s_in_pdu.length()/2- smcc_len ;
s_out = s2ws(s_in_pdu);
TRACE(L"\n\n");
TRACE(s_out.c_str());
s_in_pdu += s_end_flag;
return res;
}
```