

AnyDATA 短消息使用手册

[DTE 与 MODEM 之间基于 AT 的 SMS 协议]

Module	DTSS/DTGS Module
Subject	AT based SMS protocol between DTE and Modem
ssued Date	April. 20, 2004
Version	1.4
Written by	AnyDATA (Shanghai) Co., Ltd.



AnyDATA.NET

AnyTime AnyPlace Any Wireless Data Solutions™

中国公司联系方式

AnyDATA (Shanghai) Co., Ltd.

上海浦东新区张江高科技园区松涛路

489 号 B 区 1F

e-Mail : info_china@uct.net

Korea Contact

AnyDATA.NET

Woori Bank Bldg. 6th Fl., 1-12

Byulyang-dong Kwachon,

Kyunggi-do, Korea

e-Mail : info@anydata.net

目 录

1 引言	4
1.1 目的	4
1.2 版本	4
1.3 参考文献	4
1.4 缩略语	4
1.5 约定	5
2 短消息收发过程.....	6
2.1 短消息发送过程	6
2.1.1 步骤分解	6
2.2 短消息接收过程	9
2.2.1 步骤分解	9
3 短消息协议.....	10
3.1 相关 AT 命令	10
3.1.1 查询当前协议	10
3.1.2 设置短消息协议	10
3.2 简单短消息发送格式	11
3.3 简单短消息接收格式	12
3.4 扩展短消息协议	13
3.4.1 概述	13
3.4.2 目的电话号码	14
3.4.3 回叫电话号码 (源电话号码)	14
3.4.4 电信服务识别号	15
3.4.5 用户数据区	15

3.4.6 编码类型	16
3.4.7 时间戳	17
3.5 扩展短消息发送格式	18
3.6 扩展短消息接收格式	19
4 新增短信相关 AT 命令	21
4.1 AT+SMSP	22
4.2 AT+SMSP?	23
4.3 AT+FSET=<param>	24
4.4 AT+ MSGF=<param>	24
4.5 AT*SKT*MOREQ	24
4.6 AT*SKT*READMT	26
4.7 AT*SKT*DELMT	27
4.8 AT*SKT*PING	28
5. 短信编码相关问题	30
5.1 乱码问题	30
5.2 乱码原因	30
5.3 乱码解决方法	31
5.4 Unicode 码的转换	32
5.5 关于 Uni2GB.txt 文件的说明	34

1 引言

1.1 目的

本文档描述了 AnyDATA 公司的 CDMA 无线 MODEM 的短消息接收/发送过程及数据协议。此协议是针对 DTE 与 MODEM 之间的接口。

AnyDATA Modem 提供了两种不同的 SMS 接收/发送协议。一种为简单协议, 适合于一些简单的应用, 所有的模块都支持此协议。一种为扩展协议, 收发的格式比较复杂。提供了一些额外的参数及控制手段, 适合于一些比较复杂的应用环境。只有新版软件的模块支持此协议。

1.2 版本

版本	日期	作者	简述
1.0	2000/10/20		初始版本
1.1	2002/11/11		增加了扩展短消息协议的说明
1.3	2003/07/03		增加了短消息的编码类型, 修订发送流程
1.4	2004/04/20		增加了短消息表

1.3 参考文献

- [1] Short Message Services for Spread Spectrum System TIA/EIA/IS-637-A Qualcomm
- [2] Administration of Parameter Value Assignments for CDMA2000 Spread Spectrum Standards, Release A. TSB-58-C Qualcomm
- [3] Cellular Radio telecommunications Intersystem Operation TIA/EIA-41-D Qualcomm
- [4] CDMA Data Terminal –AT Command User Manual AT01-V10 AnyDATA, Inc.
- [5] AnyDATA 短消息内容乱码解决方法 AnyDATA(Shanghai) Co., Ltd.

1.4 缩略语

CDMA	Code Division Multiple Access	码分多址 CDMA
RUIM	Removable User identity module	CDMA 用户卡
DTE	Data Terminal Equipment	数据终端设备(一般为 PC 机)
DCE	Data Communication Equipment	数据通信设备(一般为 MODEM)

SMS	Short Message Service	短消息
SMSC	Short Message Service Center	短消息中心
BS	Base Station	基站
L2	Layer 2	层 2, 指协议的层次
L3	Layer 3	层 3, 指协议的层次
PIN	Personal Identification Number	个人身份号码
SKT	SK Telecomm	韩国 SK 电信公司
TI	Teleservice Identifier	电信服务识别号

1.5 约定

1 在以下的描述中, 所有的 AT 命令及返回值都以斜体字表示, 但图例中的 AT 命令不在此范围内。

例:

AT+CAD?

+CAD:1

2 对于模块设置的默认值以黑体字表示

例:

电话服务识别号可选值如下:

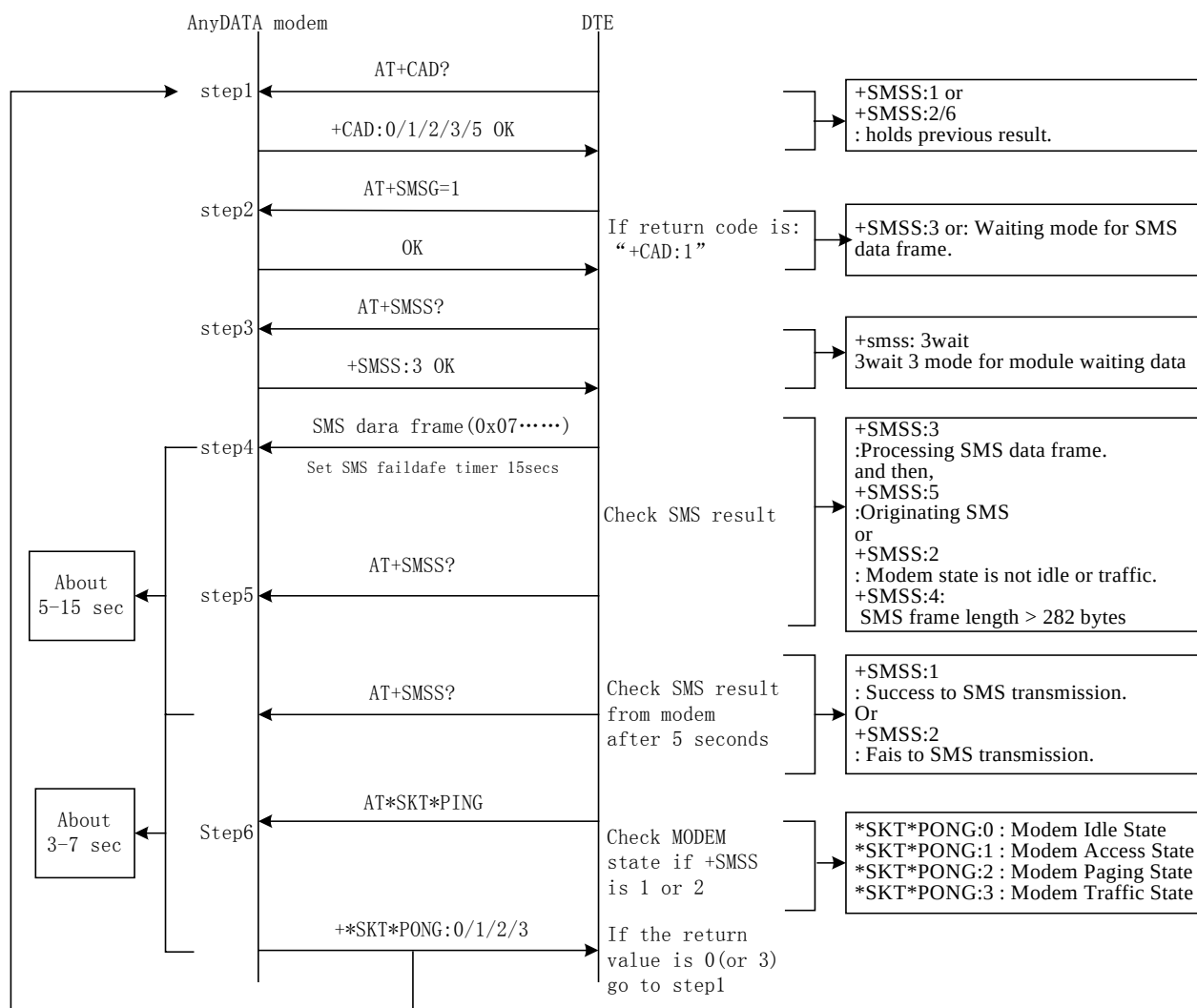
TI	描述
0x1001	寻呼信息
0x1002	SMS (短消息)
0x1003	语音短消息
其它值保留以备将来使用. 更多内容参见文献 [2]	

表明上例中默认值为 0x1002

2 短消息收发过程

2.1 短消息发送过程

下图是 AnyDATA 模块发送短消息的过程



短消息发送过程

2.1.1 步骤分解

- AnyDATA 模块提供两种方式来检测当前模块的状态（是否有 CDMA 网络可以使用）。一种使用 AT 命令检测，另一种通过对 60 针连接线中的一个特别针脚进行检测。

- AT 命令检测法

模块上电启动后, 使用 `AT+CAD?` 命令检测模块的状态

此命令的返回格式为

`+CAD:0/1/2..5`

OK

如果返回结果是 1, 说明模块已经处于就绪状态, 可以进行进一步的工作。如果返回结果为 0, 则可能模块还是初始化中, 可以等几秒钟后再试, 如果一直处于这种情况, 可能在本地区没有信号或者天线没有安装好。请检测一下是否有以上的问题。

- 特定针脚检测法

可以检测模块 60 针接口的第 16 针。如果此针为低电平, 则说明模块已经处于就绪状态, 可以进行进一步的工作, 如果是高电平, 则可能模块还是初始化中, 可以等几秒钟后再试, 如果一直处于这种情况, 可能在本地区没有信号或者天线没有安装好。请检测一下是否有以上的问题。

※ 在此状态下 `AT+SMSS?` 的可能返回值如下:

1. `SMSS:1` 表明上一个 SMS 发送成功
2. `SMSS:2` 表明上一个 SMS 发送失败

2. 发送 `AT+SMG=1`

返回结果 `+SMG:1`, 如果有返回, 可以进入下一步发送数据包, 否则结果发送过程, 发送失败, 稍后重试。

※ 在此状态下 `AT+SMSS?` 的可能返回值如下:

1. `SMSS:3` 表明模块等待 DTE 发送短消息数据包。关于数据包的格式在下面的章节中描述。

3. 发送 `AT+SMS?` 查询模块状态

直到返回结果 `SMSS: 3`, 可以进入下一步发送数据包, 否则发送失败, 稍后重试。

4. 发送特定格式的数据包, 此数据包的格式将在下面的章节中描述

5. 等 5-15 秒钟后, 发送 `AT+SMSS?` 查询发送的情况

※ 在此状态下 `AT+SMSS?` 的可能返回值如下:

1. `SMSS:3` 表明模块等待 DTE 发送数据包, 此状态一般会在 DTE 发完数据后一秒内结束。
2. `SMSS:2` 表明模块状态不在 IDLE 及业务状态, 需要重新检测模块的状态, 发送失败, 稍后回到第一步重试。
3. `SMSS:4` 表明模块收到的数据包格式有错误, 或者长度不对。
4. `SMSS:5` 表明模块正在发送数据中, 可能处于以下两种可能情况: 模块正在向基站/SMSC 发送数据, 或数据已经发送完毕, 正在等待 BS/SMSC 的 L2 或 L3 响应消息。在这种状

态下可以等待 5-15 秒后再次通过 *AT+SMSS?* 查询状态。

6. 模块收到 SMSC 的 L3 应答

※ 在此状态下 *AT+SMSS?* 的可能返回值如下:

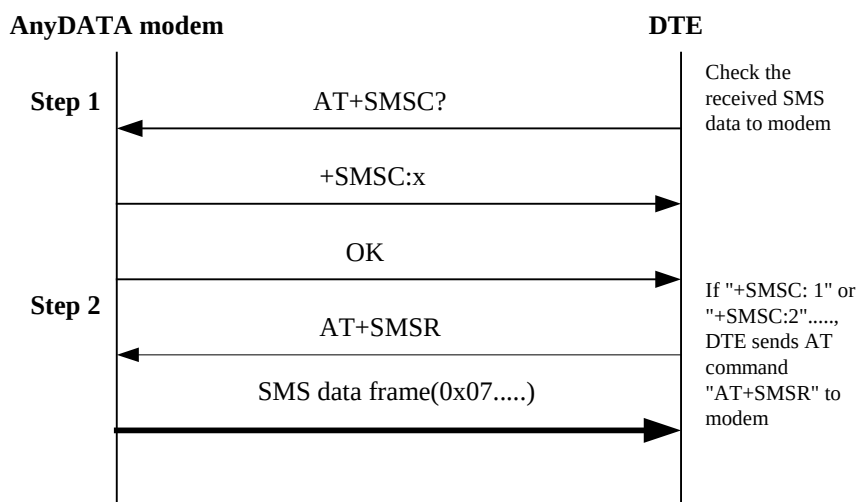
1. *SMSS:1* 表明短消息已经成功的从 SMSC 发送到了目的地
2. *SMSS:2* 表明从 SMSC 到目的地的发送失败, 但是从源到 SMSC 的发送是成功的。

7. 可选步骤

如果 *AT+SMSS?* 返回的结果是 1, 2 或者 4, 则 DTE 可以发送下一条 SMS。

2.2 短消息接收过程

下图是 AnyDATA 模块接收短消息的过程



短消息接收过程

2.2.1 步骤分解

1. 发送 **AT+SMSC?** 查询当前模块中有没有短消息

模块的返回结果为

+SMSC:0/1/2/3 ... 23

OK

如果返回结果不等于 0，说明有短消息，可以进入下一步读取短消息。

在新的软件版本中，可以控制当收到短消息的时候，从串口送出 **SMSR** 字串提示。具体内容参见下面章节“新增 AT 命令”。

注：DTS-800 最大可存储 23 条短消息，一条语音短消息。

2. 发送 **AT+SMSR** 读取短消息，返回的格式在下面的章节有详细的描述。

3 短消息协议

AnyDATA Modem 提供了两种不同的 SMS 接收/发送协议。一种为简单协议，适合于一些简单的应用，所有的模块都支持此协议。一种为扩展协议，收发的格式比较复杂。提供了一些额外的参数及控制手段，适合于一些比较复杂的应用环境。只有新版软件的模块支持此协议。

扩展短消息协议是为了适应越来越多的短消息应用而开发的一种全新协议，相比原来的协议，可扩充性及灵活性得到了很大的提高，可以很容易的支持以后的 MMS 等新的短消息业务类型。

所有的 AnyDATA 模块都支持简单短消息协议，所以对于已有应用，根本不用担心新的协议引入会导致原有的应用不能用。在以后 AnyDATA 所有新推出的模块中都会提供新协议的支持。

3.1 相关 AT 命令

这两种模式的可以通过 AT 命令进行查询及切换。涉及到的 AT 命令如下

3.1.1 查询当前协议

AT+FSET? 查询当前的状态，示例：

```
AT+FSET?  
+FSET:0xC  
OK
```

返回结果与 0x4 进行逻辑与，如果结果不为 0，则表明模块采用扩展短消息协议，否则采用简单短消息格式。当然，老的软件版本不支持 AT+FSET 命令，所以会返回 ERROR，在这种情况下只能采用简单短消息协议。

在上例中返回结果为 0xC，与 0x4 逻辑与的结果为 1，所以模块当前使用扩展短消息协议。

3.1.2 设置短消息协议

AT+FSET=x 设置当前所采用的协议，注意，在设置之前，要先通过 AT+FSET? 读出当前的值，与 0x4 逻辑或后再通过此命令设置进行。示例：

1. 先读出当前值

```
AT+FSET?  
+FSET:0xB  
OK
```

2. 计算新值

$0xB | 0x4 = 0xF$

3. 设置新值

```
AT+FSET=F  
+FSET:0xF  
OK
```



注意，在设置命令中，输入的数据为十六进制数值，但有前面不要加 0x 前缀，如果加了 0x 前缀，模块会认为数值不正确。

`AT+FSET=x` 命令执行完毕后立即生效，并不需要重新启动之类的操作，设置的值将永久保存，直到下一次设置为止。

3.2 简单短消息发送格式

当采用简单短消息协议时，发送 SMS 时的数据格式如下：

```
typedef struct {
    BYTE company_id;
    BYTE len;
    BYTE dest[13];
    BYTE callback[13];
    BYTE data[MAX_TX_SMS_DATA+1];
} sms_send_type;
```

- company_id 为公司标识, 固定为 0x07

- len 为消息的长度，包括 len, dest, callback 及 data 四个字段的长度，计算方法为：

$$\text{len} = 1 + 13 + 13 + \text{有效 SMS 字节数}$$

1 为 len 字段本身的长度

第一个 13 为目标电话号码字段的长度

第二个 13 为回叫电话号码字段的长度

- dest 为目标电话号码，长度为 13 位，如果号码不足 13 位，最后几位用 0x20，也就是空格填充，例：

```
0x31 0x33 0x33 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x20 0x20  "13312345678"
```

- callback 为回叫号码，长度为 13 位，不足时在最后填空格，这个字段在**国内实际上没有用处，一般不用填，直接填空格就行了**

-data 为 SMS 的内容，消息体的最大长度与各地网络有关，一般最大限制在 140 个字节左右，这是受七号信令中的 MTP 包的大小限制的。MODEM 本身目前不支持短信分包业务。过大的包需要应用层进行分解。

示例：发送给 13312345678 这个号码一串字符“abcd”，消息格式如下

```
0x07 0x1F 0x31 0x33 0x33 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x20 0x20 0x20 0x20 0x20 0x20
0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x61 0x62 0x63 0x64
```

红色的为电话号码 13312345678

绿色的为回叫号码，在这里全部填空格(0x20)

黑色的为发送的短信内容，“abcd”

发送时的注意事项

在发送 SMS 时, 如果从 PC 机到模块的发送速度过快, 由于 MODEM 内部缓冲有限, 处理速度不够快, 容易丢失部分数据, 造成的后果是数据不完整或发送错误数据, 特别是在发送比较大的 SMS 时很容易发生。简单的解决方法是在往串口输送数据时每发送若干字节后停顿一小会。建议每发送 10 个字节后停顿 20ms

3.3 简单短消息接收格式

当采用简单短消息协议时, 接收到的 SMS 数据格式如下:

```
typedef struct {  
    BYTE company_id;  
    BYTE len;  
    BYTE dest[13];  
    BYTE data[MAX_RX_SMS_DATA+1];  
} sms_read_type;
```

- company_id 为公司标识, 固定为 0x07
- len 为消息的长度, 包括 len, dest 及 data 三个字段的长度, 计算方法为:
len = 1 + 13 + 有效 SMS 字节数
1 为 len 字段本身的长度
13 为 SMS 发送源电话电码字段的长度
- dest 为 SMS 发送方的电话号码, 长度为 13 位, 如果号码不足 13 位, 最后用是 0x0 填充, 例:
0x31 0x33 0x33 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 “13312345678”
- data 为 SMS 的内容。
- 如果开启了显示短信时间的功能, 则在内容字段中前 12 个字节为日期及时间
YYYYMMDDHHMM
例如: 200207061234 表示 2002 年 7 月 6 日, 12:34

示例: 收到从号码 13312345678 发来的 SMS, 内容为'abcd', 消息格式如下:

0x07 0x12 0x31 0x33 0x33 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 0x61 0x62 0x63 0x64

如果打开显示时间信息功能, 消息格式如下:

0x07 0x12 0x31 0x33 0x33 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 0x32 0x30 0x30 0x32 0x30
0x39 0x30 0x31 0x31 0x32 0x33 0x34 0x61 0x62 0x63 0x64

红色的为电话号码 13312345678

兰色的为短信发送时间 2002-09-01 12:34

黑色的为接收的到的短信内容 “abcd”

3.4 扩展短消息协议

3.4.1 概述

扩展短消息协议不再采用简单短消息协议那样固定字段，固定长度的形式。而采用了更为灵活的动态字段及长度的方式。所以在处理上将会上简单协议有很大的不同。

此格式如下：

Start	Arg_no	Arg[1]			Arg[2]	Arg[N] (Data)		
		Arg_ID	Arg_Len	Arg_Data		Arg_ID	Arg_Len	Arg_Data
0x07	N		M1				MN	
1 byte	1 byte	2 bytes	1 byte	M1 bytes		2 bytes	1 byte	MN bytes

每个字段的含义如下：

- Start: 起始字节，固定为 0x7，与简单协议的 company_id 一样。
- Arg_no: 参数包个数，指下面描述的参数块的数目
- Arg#: 短消息数据包中的参数块，参数块包含短消息中有效内容，如主叫地址，短消息时间，短消息内容等信息。每一个参数块由三部分组成：Arg_ID 参数块 ID，Arg_Len 参数块的长度，Arg_Data 参数内容。

※ Arg_ID: 参数块 ID，用来识别不同的参数块，此字段占用两个字节，可使用的 ID 参见下表。

※ Arg_Len: 参数块的长度，此字段标识其后的 Arg_Data 字段的长度（字节数）。不包括前面的 Arg_ID 及它自己的长度。

※ Arg_Data: 参数内容，此字段的长度等于 Arg_Len 字段的值。

Arg_ID	参数名称	发送 S/接收 R	参数值
0x0001	目的电话号码	S(只有发送端使用)	最大 32 位, ASCII 码形式
0x0002	回叫电话号码	S/R(收/发)	最大 32 位, ASCII 码形式
0x0004	电信服务号(TI)	S(只有发送端使用)	2 字节,十六进制值
0x0008	短消息内容	S/R(收/发)	最大 255 个字节, 8-bit ASCII 值, 其最大可使用长度视运营商提供的网络支持情况而定。一般情况下最大可使用字节为 140 个字节。
0x0010	编码类型	R(只有接收端使用)	2 位, ASCII 码形式
0x0020	时间戳	R(只有接收端使用)	12 位, ASCII 码形式
其它的值 0x0040 ~ 0x8000 为保留值, 以备将来使用			

参数块 ID 的值

3.4.2 目的电话号码

此参数块只有发送方使用

此参数块的内容是短消息接收方的电话号码, 最大支持 32 位电话号码, 但在实际使用中视运营商及不同的国家的情况而定, 在中国一般使用 11 位电话号码。

示例: 13312345678

Arg_ID		Arg_Len	Arg_Data										
0x0001		11	'1'	'3'	'3'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'
0x00	0x01	0x0B	0x31	0x33	0x33	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38
2 bytes		1 byte	0x0B = 11 digits										

3.4.3 回叫电话号码 (源电话号码)

此参数块可选

此参数块的内容是短消息发送方的电话号码, 最大支持 32 位电话号码, 但在实际使用中视运营商及不同的国家的情况而定, 在中国一般使用 11 位电话号码。

如果发送方使用此字段, 则模块将使用在此字段中指字的电话号码作为源号码发送过去。并且对方模块接收时也显示此号码。

一般此号码就是发送方真正的电话号码, 但是 DTE 也可以使用任意其它值。

注 1: 如果对方是手机, 一般会忽略此字段, 只显示真正的电话号码;

注 2: 如果此字段为空, 则对方将显示真正的发送方电话号码。

示例: 13531234567

Arg_ID		Arg_Len	Arg_Data										
0x0002		11	'1'	'3'	'5'	'3'	'1'	'2'	'3'	'4'	'5'	'6'	'7'
0x00	0x02	0x0B	0x31	0x33	0x35	0x33	0x31	0x32	0x33	0x34	0x35	0x36	0x37
2 bytes		1 byte	0x0B = 11 digits										

3.4.4 电信服务识别号

此参数块只有发送方使用, 此参数块可选

此参数块用来指示所使用的服务类型, 短消息服务中心根据此字段来处理收到的信息。更详细信息请参考文献[2]

此参数块的值采用十六进制数
可使用的值为:

TI	描述
0x1001	寻呼信息
0x1002	SMS (短消息)
0x1003	语音短消息
其它值保留以备将来使用, 更多内容参见文献 [2]	

示例 : SMS TI(0x1002)

Arg_ID		Arg_Len	Arg_Data	
0x0004		2	0x1002	
0x00	0x04	0x02	0x10	0x02
2 bytes		1 byte	0x02 = 2 bytes	

3.4.5 用户数据区

此参数块接收/发送方都使用

此参数块包含了用户真正到发送给对方的数据信息, 其最大长度受运营商提供的设备的支持能力而定, 所以不同的运营商可能提供的最大允许长度会有所不同。在中国, 一般最大允许长度为 140 个字节。

示例: "SMS Data Frame2"

Arg_ID		Arg_Len	Arg_Data									
0x0008		15	'S'	'M'	'S'	' '	'D'	'a'	't'	'a'	' '	'F'
0x00	0x08	0x0F	0x53	0x4d	0x53	0x20	0x44	0x61	0x74	0x61	0x20	0x46
			'r'	'a'	'm'	'e'	'2'					
			0x72	0x61	0x6d	0x65	0x32					
2 bytes		1 byte	0x0F = 15 bytes									

3.4.6 编码类型

此参数块只有接收方使用, 此参数块可选

此参数块用来指示短消息的编码类型。用户可以根据此字段来处理收到的信息。
 此参数块的值采用十六进制数
 可使用的值为:

编码类型(ASCII 码值)	描述
"00"	Octet
"01"	Extended Protocol Message
"02"	ASCII
"03"	IA5
"04"	UNICODE
"16"	KSC5601
其它值保留以备将来使用, 更多内容参见文献 [2]	

示例: ASCII 类型

Arg_ID		Arg_Len	Arg_Data	
0x0010		2	'0'	'2'
0x00	0x10	0x02	0x30	0x32
2 bytes		1 byte	0x02 = 2 bytes	

3.4.7 时间戳

此参数块只有接收方使用，此参数块可选

此字段是短消息服务中心收到短消息的时间。

此字段有 12 个字节，排列顺序如下：

年	月	日	时	分
4 bytes	2 bytes	2 bytes	2 bytes	2 bytes

示例： 2002/11/04, 18:19

Arg_ID		Arg_Len	Arg_Data									
0x0020		12	'2'	'0'	'0'	'2'	'1'	'1'	'0'	'4'	'1'	'8'
0x00	0x20	0x0C	0x32	0x30	0x30	0x32	0x31	0x31	0x30	0x34	0x31	0x38
			'1'	'9'								
			0x31	0x39								
2 bytes		1 byte	0x0C = 12 bytes									

3.5 扩展短消息发送格式

当采用扩展短消息协议时，发送短消息的数据格式示例如下：

- 目的电话号码 011-123-4567
- 回叫电话号码 019-777-1111,
- 短消息发送缓冲区名称 sms_buf[]
- 需要发送的短消息长度为 100 个字节.
- 发送的短消息内容为

“ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
 VWXYZabcdefghijklmnopqrstuvwxyz”

SMS data frame	sms_buf[]
起始位	sms_buf[0] = 0x07
参数块个数	sms_buf[1] = 0x03
	sms_buf[2] = ','
目的电话号码	Arg_ID1 => sms_buf[3] = 0x00, sms_buf[4] = 0x01, Arg_no1 => sms_buf[5] = 0x0A Arg_Data1 => sms_buf[6] = 0x30, sms_buf[7] = 0x31, sms_buf[8] = 0x31, sms_buf[9] = 0x31, sms_buf[10] = 0x32, sms_buf[11] = 0x33, sms_buf[12] = 0x34, sms_buf[13] = 0x35, sms_buf[14] = 0x36, sms_buf[15] = 0x37
	sms_buf[16] = ','
回叫电话号码	Arg_ID2 => sms_buf[17] = 0x00, sms_buf[18] = 0x02, Arg_no2 => sms_buf[19] = 0x0A Arg_Data2 => sms_buf[20] = 0x30, sms_buf[21] = 0x31, sms_buf[22] = 0x39, sms_buf[23] = 0x37, sms_buf[24] = 0x37, sms_buf[25] = 0x37, sms_buf[26] = 0x31, sms_buf[27] = 0x31, sms_buf[28] = 0x31, sms_buf[29] = 0x31
	sms_buf[30] = ','
用户数据区	Arg_ID3 => sms_buf[31] = 0x00, sms_buf[32] = 0x08, Arg_no3 => sms_buf[33] = 0x64, Arg_Data3 => sms_buf[34] = 0x41('A'), sms_buf[35] = 0x42('B'), sms_buf[36] = 0x43('C')... sms_buf[131] = 0x74('t') sms_buf[132] = 0x75('u'), sms_buf[133] = 0x76('v')

3.6 扩展短消息接收格式

当采用扩展短消息协议时，接收短消息的数据格式示例如下：

- 回叫电话号码/源电话号码为 019-777-1111,
- 短消息中心收到短消息的时间为 2002/11/04, 08:02 PM
- 短消息接收缓冲区名称为 sms_buf[]
- 接收到的短消息内容长度为 100 个字节.

- 接收到的短消息内容为

“ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
VWXYZabcdefghijklmnopqrstuv”

SMS data frame	sms_buf[]
起始位	sms_buf[0] = 0x07
参数块个数	sms_buf[1] = 0x04
	sms_buf[2] = ','
回叫电话号码	Arg_ID1 => sms_buf[3] = 0x00, sms_buf[4] = 0x02, Arg_no1 => sms_buf[5] = 0x0A Arg_Data1 => sms_buf[6] = 0x30, sms_buf[7] = 0x31, sms_buf[8] = 0x31, sms_buf[9] = 0x37, sms_buf[10] = 0x37, sms_buf[11] = 0x37, sms_buf[12] = 0x31, sms_buf[13] = 0x31, sms_buf[14] = 0x31, sms_buf[15] = 0x31
	sms_buf[16] = ','
时间戳	Arg_ID2 => sms_buf[17] = 0x00, sms_buf[18] = 0x20, Arg_no2 => sms_buf[19] = 0x0C Arg_Data2 => sms_buf[20] = 0x32, sms_buf[21] = 0x30, sms_buf[22] = 0x30, sms_buf[23] = 0x32, sms_buf[24] = 0x31, sms_buf[25] = 0x31, sms_buf[26] = 0x30, sms_buf[27] = 0x34, sms_buf[28] = 0x32, sms_buf[29] = 0x30, sms_buf[30] = 0x30, sms_buf[31] = 0x32
	sms_buf[32] = ','
编码类型	Arg_ID3 => sms_buf[33] = 0x00, sms_buf[34] = 0x10, Arg_no3 => sms_buf[35] = 0x02 Arg_Data3 => sms_buf[36] = 0x30, sms_buf[37] = 0x32
	sms_buf[38] = ','
用户数据区	Arg_ID4 => sms_buf[39] = 0x00, sms_buf[40] = 0x08, Arg_no4 => sms_buf[41] = 0x64, Arg_Data4 => sms_buf[42] = 0x41('A'), sms_buf[43] = 0x42('B'), sms_buf[44] = 0x43('C')... sms_buf[139] = 0x74('t') sms_buf[140] = 0x75('u'), sms_buf[141] = 0x76('v')

注: 如果 DTE 使用 AT+SMSR 命令接收短消息, 模块会在消息命令执行完成后删除模块中的短消息。

4 新增短信相关 AT 命令

命令语法	返回值	功能说明
<i>Interface 1</i>		
AT+SMSP=<param> <param>: 2CD --- ASCII 编码 285 --- UNICODE 编码 2C5 --- OCTET 编码	OK	设置 SMS 发送编码格式
AT+SMSG=<val> <val>: 0 --- 取消 SMS 发送模式 1 --- 设置串口为 SMS 发送模式	OK	设置或取消 SMS 发送模式
AT+SMSD	OK	删除所有短消息
AT+SMSR	<str> OK	读出并删除最近一条未读短消息
AT+SMSA?	+SMSA: <count> OK	查询可用的地址
AT+SMSO?	+SMSO: <count> OK	查询已读短消息的条数
AT+SMSC?	+SMSC: <count> OK	查未读短消息的条数
AT+SMSS?	+SMSS: <value> <value>: 1 --- 上一次 SMS 发送成功 2 --- 上一次 SMS 发送失败 3 --- 模块等待 DTE 发送数据包 4 --- 数据包格式有错误 5 --- 模块正在发送数据中	查询模块发送短消息状态
AT+RSMS=<x> <x>: 保存短消息的地址索引	<str> OK	读出指定地址短消息, 读出后并不删除此 SMS, 只是标记为已读。
AT+DSMS=<x> <x>: 保存短消息的地址索引	OK	删除指定地址短消息

Interface II		
AT*SKT*MOREQ=<x>,<y>,<z>,<u>,<v>, <x>: 序号 <y>: 目的电话号码 <z>: 回叫电话号码 <u>: 电话服务识别号 4098 <v>: 消息内容	*SKT*MOREQ:<no> *SKT*MOACK:<no>,<result> <no>: 序号 <result>: 结果	SKT 的发送短信命令
AT*SKT*READMT=<x> <x>: 序号	*SKT*READMT:<x>,<y>,<z>,<u> <v> <x>: 序号 <y>: 时间戳 <z>: 回叫电话号码 <u>: 电话服务识别号 4098 <v>: 消息内容	SKT 的读取短信命令
AT*SKT*DELMT=<x> <x>: 序号	*SKT*READMT:<result> <result>: 结果	SKT 的删除短信命令

4.1 AT+SMSP

功能:

设置短信相关的参数

参数:

此参数使用十六进制值

可使用的参数值如下:

参数值	描述
0xCD	采用 ASCII 码格式编码短消息内容, 适合发送 32-127 之间的可见 ASCII 字符
0x85	采用 Unicode 码格式对发送的短信进行编码, 适合发送中文或其它不可见字符
0xC5	采用 Octet 码格式编码短消息内容, 适合发送二进制数据 (版本 10 新增)
0x200	是否开启收到短消息指示
0x10	是否在短消息中包含时间戳
其它值保留以备将来使用.	

注 1: 前两个值为互斥, 两者只能选其一, 但每一个都可以与其它值互相组合, 所以在设置之前, 最好通过 AT+SMSP? 命令读出当前值, 与要设置、清除的位进行相应操作后再设置回去 ;

注 2: 0x10, 是否显示时间戳, 只针对简单短消息协议和 SKT 接收 SMS 的命令, 对扩展短消息协议, 此参数无效;

注 3: 在输入命令时, 参数值为十六进制, 但不包括前面的 0x 前缀;

注 4: 此命令执行完后, 立即生效, 并保留此值知道下一次修改; 版本 10 以前的版本需要模块复位后才能生效。

结果:

OK

示例 1:

AT+SMSP=CD

OK

AT+RESET

此例中, 模块采用 ASCII 编码形式。不开启来短信指示及时间戳。

示例 2:

AT+SMSP=295

OK

AT+RESET

此例中, 模块采用 Unicode 编码形式。开启来短信指示及时间戳。295 = 0x200 | 0x85 | 0x10

在开启来短信指示功能后, 在收到短消息时, 模块会输入以下字符串:

SMR

4.2 AT+SMSP?

功能:

查询当前短信设置参数, 参数的具体含义参见命令 *AT+SMSP*

参数:

无

结果:

+SMSP:0xXXX

OK

XXX 的当前的参数值, 以十六进制表示, 返回具体的位数视当前设置值而定, 最大可能为 4 位, 最小 1 位。前导 0 不会返回。参数的具体含义参见命令 *AT+SMSP*

注意:

无

示例:

AT+SMSP?

+SMSP:0x2DD

OK

此例表明，当前模块采用 ASCII 编码模式，并且开启了来短信指示及时间戳功能。

4.3 AT+FSET=<param>

功能:

特殊功能设置命令，用于设置模块的一些特殊功能开关，此命令中的参数并不一定所有的模块都适用。

注意:

具体请参见《AnyDATA CDMA modem AT CMD-vx.x》PDF 文档。

4.4 AT+ MSGF=<param>

功能:

显示提示信息设置命令，用于设置显示模块的状态信息。此命令中的参数并不一定所有的模块都适用。

注意:

具体请参见《AnyDATA CDMA modem AT CMD-vx.x》PDF 文档。

4.5 AT*SKT*MOREQ

功能:

SKT 的发送短信命令，SKT 是韩国 SK 电信公司的简称，它制定了一系列自己的规范。

参数:

短消息序号,目的电话号码,回叫电话号码,4098,消息内容

序号:

0- 255 之间的数, 用于区分两条不同的短消息

必选;

目的电话号码:

要发往的目的地的用户的号码

必选;

回叫电话号码:

发送信息的电话号码或任意其它值

当此值为空时, 使用发送方的电话本电话号码为准, 否则将使用用户填写的号码为准。

可选;

TI: 电话服务识别号, 必选

可选值如下:

TI	描述
4097	寻呼信息
4098	SMS (短消息)
4099	语音短消息
其它值保留以备将来使用, 更多内容参见文献[2]	

消息内容:

要发送的内容, 只能使用 0x20-0xFF 之间的数据, 对于其它数据不能保证一定能够发送成功。

结果:

发送前返回

*SKT*MOREQ:序号

发送后返回

*SKT*MOREQ:序号

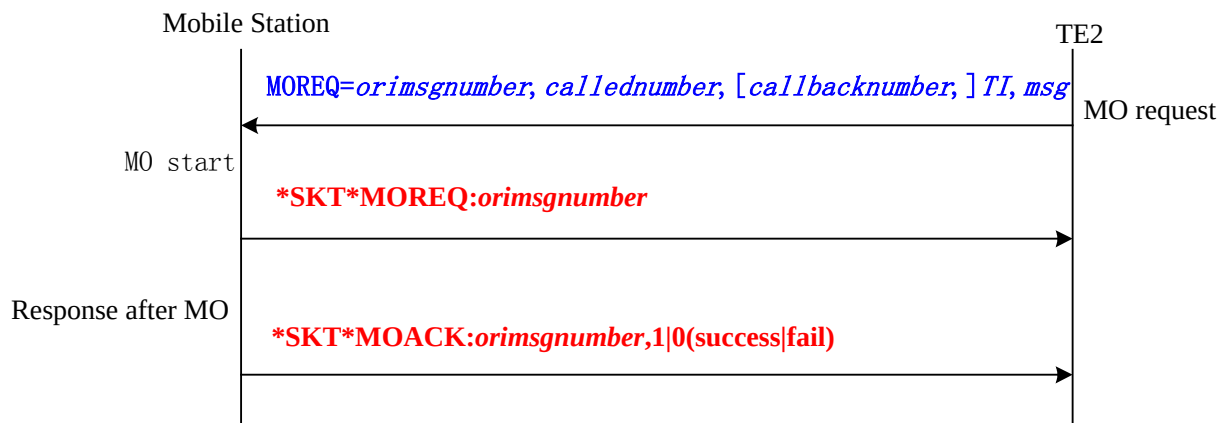
*SKT*MOACK:序号,结果

结果为 0 表示失败, 结果为 1 表示成功

*SKT*MOACK:0,0 序号为 0 的短信发送失败

*SKT*MOACK:2,1 序号为 2 的短信发送成功

参考下图:



注意:

对于小于0x20的字符,有很多字符不能使用,如0x07 0x0D 0x0A 0x08 ... ,所以在使用时请不要使用小于0x20的数据。不能保证能够发送成功。

示例:

```
AT*SKT*MOREQ=0,13311001111,,4098,abcd
```

```
*SKT*MOREQ:0
```

```
*SKT*MOACK:0,1
```

此例表示序号为1的短消息发送成功,发送的目的为13311001111

4.6 AT*SKT*READMT

功能:

从模块中读取短消息

参数:

recvmsgnumber :

短消息序号,如果非零读取指定序号的短消息;

如果是零,则返回最后一条短消息,最后一条指的是最近收到的;

结果:

```
*SKT*READMT:recvmsgnumber, rcvtime, [callback], TI, msg
```

rcvmsgnumber: 短消息的序号

rcvtime: 接收到的时间, 指的是SMSC收到的时间, 不是模块收到的时间, 一般早于模块收到的时间, 时间格式参见AT*SKT*MOREQ

callback: 回叫号码, 即发送此SMS的号码

TI: 电信服务识别号, 参见AT*SKT*MOREQ的详细说明

Msg: 接收到的短消息内容

注意:

1. 用此命令接收短消息, 模块并不会将短消息从模块中删除, 需要发命令删除, 命令参见AT*SKT*DELMT
2. 当收到另一条短消息时, 原先的序号会发生变化
3. 序号从0开始, 最新收到的为0, 序号越大, 收到越早
4. 不建议使用此命令接收SMS, 其它接收方法参见文献[10]

示例1:

*AT*SKT*READMT=2*

**SKT*READMT:2, 20020903081324, 13312345678, 4098, CDMASMS*

此例接收序号为2的短消息, 内容为 "CDMASMS", 短消息的时间为2002-9-3 8:13:24, 来自13312345678, 此例指定了序号

示例2:

*AT*SKT*READMT=0*

**SKT*READMT:0, 20020518162345, 13312345678, 4098, AnyDATA*

此例接收序号为0的短消息, 内容为 "AnyDATA", 短消息的时间为2002-5-18 16:23:45, 来自13312345678, 此例序号为零, 接收最近一条短消息。

4.7 AT*SKT*DELMT

功能:

删除一条短消息

参数:

rcvmsgnumber

短消息序号, 如果非零删除指定序号的短消息;

如果是零, 则删除最后一条短消息, 最后一条指的是最近收到的;

结果:

**SKT*DELMT:X*

X为1表示删除成功，为0表示删除失败

注意:

1. 因为收到新的SMS时序号会发生变化，所以删除时可能会删除错误的短消息
2. 序号从0开始，最新收到的为0，序号越大，收到越早

示例1:

*AT*SKT*DELMT=3*

*AT*SKT*DELMT:0*

此例表示删除序号为3的短消息失败，可能的原因是没有这条短消息。

示例2:

*AT*SKT*DELMT=0*

*AT*SKT*DELMT:0*

此例表示删除最后一条短消息成功，此例指定短消息序号为零，表示最后一条。

4.8 AT*SKT*PING

功能:

读取模块的状态。

参数:

结果:

**SKT*PONG:X*

- 0: 模块进入IDLE状态
- 1: 模块进入ACCESS状态
- 2: 模块进入PAGING状态
- 3: 模块进入TRAFFICE状态

注意:

示例1:



AnyDATA

AnyDATA (Shanghai) Co., Ltd

上海浦东张江高科技园区

松涛路 489 号 B 区 1F

邮编: 201203

电话: (021) 5080-4828

传真: (021) 5080-3828

网站: <http://www.anydata.net.cn>

电子邮件: info_china@uct.net

*AT*SKT*PING*

*AT*SKT*PONG:0*

此例表示模块进入IDLE状态。

5. 短信编码相关问题

5.1 乱码问题

现在模块与手机之间收发SMS存在以下几种乱码的情况

- 5.1 模块发英文SMS到手机，手机显示乱码；
- 5.2 模块发中文SMS到手机，手机显示乱码；
- 5.3 手机发中文SMS到模块，模块显示乱码；
- 5.4 手机发英文SMS到模块，一般模块能正常显示，也有部分手机发出的SMS，模块显示出的内容中会多出一些0x0来，其中0x0是不可见的字符，现象就是每一个字符之间有一个空格。

5.2 乱码原因

显示乱码归根到底是由于系统之间编码格式的不一致引起的，市面上绝大多数手机（包括GSM和CDMA）中汉字使用的编码为双字节的Unicode码，而我们的模块往往接在PC机上使用，PC机的Windows操作系统使用的GB码，这两种编码是不一样的。这是造成乱码的主要原因，我们可以发现，模块与模块之间发送SMS，不论中文还是英文都不存在乱码的问题。手机之间也是一样的。

在SMS的规范中，对SMS的打包格式定义了几种不同的打包格式，我们的模块目前支持以下两种方式：

1. ASCII码方式，这种方式只支持ASCII码值小于128的字符，也就是普通的字母以及数字，中文字符一般由两字节组成，而且每一个字节都大于128，所以不能发送中文或字节内容大于128有短消息。

优点： 发送英文SMS到手机，一般的手机都能正确接收并显示；

对发送内容的字节数没有要求，可以奇数字节，也可以偶数字节。

缺点： 不能发送码值大于128的字节，也就是高位为1的字节，发送时高位会被网络截掉；

可以通过执行以下命令切换到这种模式

```
AT+SMSP=CD
```

```
OK
```

可以用AT+SMSP?来查询当前的模式

2. Unicode码方式，这种方式下可以发送从0x00到0xFF之间所有的字符。由于我们绝大多数用户把模块用于模块间的数据通信，并不是用于模块与手机之间通信，而且要求模块能够透明的传送0~255之间所有的字符。所以我们模块出厂时的默认值为这种模式。

优点： 可以发送0~255之间所有的字符；

缺点: 直接发送英文/中文SMS到手机, 手机将显示乱码;

由于 Unicode 码为双字节编码, 所以要求发送的内容必须为偶数个字节, 奇数个字节的情况下部分网络会拒绝, 部分网络会丢弃最后一个字节。

可以通过执行以下命令切换到这种模式:

```
AT+SMSP=85
```

```
OK
```

可以用 AT+SMSP?来查询当前的模式

3. Octet 码方式, 这种方式下可以发送从0x00到0xFF之间所有的字符。对发送内容的字节数没有要求, 可以奇数字节, 也可以偶数字节。所以这种模式最适于用于模块间的数据通信。

优点: 可以发送0~255之间所有的字符; 对发送内容的字节数没有要求, 可以奇数字节, 也可以偶数字节。

缺点: 大多数手机不识别这种编码, 将显示乱码;

可以通过执行以下命令切换到这种模式:

```
AT+SMSP=C5
```

```
OK
```

可以用AT+SMSP?来查询当前的模式

5.3 乱码解决方法

根据以上分析, 对于上述的几种情况我们可以用以下方法解决

1. 模块发英文SMS到手机, 手机显示乱码;

当模块采用ASCII编码方式时不存在这种情况, 所以可以采用以下方式:

a. 把模块的编码方式改为ASCII码方式;

b. 如果一定要用Unicode 码方式, 发送前把SMS的内容转换成Unicode码后再发送给模块。

2. 模块发中文SMS到手机, 手机显示乱码;

解决方法: 把模块的编码方式改为Unicode码方式, 发送前把SMS的内容转换成Unicode码后再发送给模块。

3. 手机发中文SMS到模块, 模块显示乱码;

解决方法: 将模块收到的SMS内容转换成GB码再显示。注: 模块的编码方式可以任意, 不会影响接收的SMS内容。

4. 手机发英文SMS到模块, 一般模块能正常显示, 也有部分手机发出的SMS, 模块显示出的内容中会多出一些0x0来, 其中0x0是不可见的字符, 现象就是每一个字符之间有一个空格。

部分手机在发送纯英文的SMS时, 会将编码方式设置为ASCII方式以节省消息长度, 但也有部

分手机不论发送中文还是英文SMS，编码方式固定为Unicode码方式，对于第一种处理方式的手机发出的SMS，模块显示是没有问题的，第二种处理方式的手机则会多出一半的字符来，这些字符全是0x0，这是因为对于英文及数字，ASCII码与Unicode码的区别是，Unicode码把一个ASCII码扩展成两个字节，并把第一个字节填0x0。解决方法是把收到的SMS转换成ASCII码再显示就可以了。

5. 特殊说明：

在最新的短消息协议中，已经增加了收到的短消息编码格式的指示字段，用户可以在收到短消息时通过对此字段的判断来决定采取什么样的措施。

同时用户可以下载<http://www.anydata.net.cn>上的有关测试程序，这个测试程序可以支持中文/英文短消息的接收与发送。并可以将收到的Unicode码转换成GB码并显示出来。

5.4 Unicode 码的转换

对于小于128的ASCII字符，与Unicode码的区别是，Unicode码把一个ASCII码扩展成两个字节，并把第一个字节填0x0。例：

	ASCII	Unicode
'A'	0x41	0x00, 0x41
'Z'	0x5A	0x00, 0x5A
'a'	0x61	0x00, 0x61
'z'	0x7A	0x00, 0x7A
'0'	0x30	0x00, 0x30
'1'	0x31	0x00, 0x31
'9'	0x39	0x00, 0x39

ASCII→Unicode，把每一个字切扩展成两个字节，第一个字节填0x0，注意这样转换后长度增加一倍，需要调整消息体中长度字段的值。

Unicode→ASCII，每两个字节中把第一个0x00的字节删除

对于GB码与Unicode之间没有简单的转换关系，只能通过查表来解决了，也就是用一张Unicode<->GB的表格来查找，把转换的字符进行一一替换。例：

	GB	Unicode
'中'	0xD6, 0xD0	0x4E, 0x2D
'国'	0xB9, 0xFA	0x56, 0xFD

在PC机上我们输入'中'字，它的值是 0xD6, 0xD0，查表得它的Unicode码为0x4E, 0x2D，用 0x4E, 0x2D替换前面的0xD6, 0xD0传给模块即可。

GB与Unicode之间的码表可以在<http://www.anydata.net.cn>上下载，根据自己的程序需要改造一下就可以了。

GB→Unicode转换的示例代码(C/C++)：

```
typedef unsigned short WORD;
```



```
typedef unsigned char BYTE;
#define LOBYTE(w) ((BYTE)(w))
#define HIBYTE(w) ((BYTE)((WORD)(w) >> 8) & 0xFF)
#define MAKEWORD( l,h) ((WORD)((BYTE)(l)) | ((WORD)((BYTE)(h))) << 8))

void _GetUnicode( const WORD wIn, WORD *wOut ) /* 查表函数, 将单个GB码转换成Unicode码*/
{
    /* 在此假设码表已经构造完成 */
    /* TBL_LEN 码表的长度, Tbl 为码表, Citem为表中元素的结构 */
    Citem *lpItem;
    *wOut = 0xFFFF;

    /* 在这里用最简单的顺序查找, 可以优化 */
    for ( int j = 0; j < TBL_LEN; j ++ )
    {
        lpItem = &Tbl[j];
        if ( lpItem->wGB == wIn )
        {
            *wOut = lpItem->wUnicode;
            return;
        }
    }
}

void GB2Unicode( const char *lpIn, char *lpOut, int *outLen ) /* 转换函数, 将字符串转换成Unicode
串*/
{
    WORD wIn, wOut;

    *outLen = 0;
    while ( *lpIn )
    {
        if ( (*lpIn) < 0x7F ) /* 普通ASCII字符 */
        {
            *lpOut = 0x00;
            lpOut ++;
            *lpOut = *lpIn;
            lpOut ++;
            lpIn ++;
        }
    }
}
```

```
}  
else  
{  
    wIn = MAKEWORD( In[0], In[1]); /* 把两个字节组成一个WORD */  
    _GetUnicode( wIn, &wOut ); /* 查表, 并把结果放在lpOut中返回 */  
    *lpOut = HIBYTE( wOut ); /* 把高位放在首字节*/  
    lpOut ++;  
    *lpOut = LOBYTE( wOut ); /* 低位放在第二字节*/  
    lpOut ++;  
    lpIn += 2;  
}  
outLen += 2;  
}  
}
```

5.5 关于 Uni2GB.txt 文件的说明

这是一张Unicode码到GB码之间的转换码表, 表的第一列为Unicode码, U-xx或U-xxxx为ASCII/GB码, 关于其它部分可以参考它的说明。

你可以根据程序的需要提取出相应的列, 按一定的顺序组成其它文件以便程序中使用。