

GSM library for GTM-201

User Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 2011 by ICP DAS Co., LTD. All rights reserved worldwide.

Trademark

The names used for identification only may be registered trademarks of their respective companies.

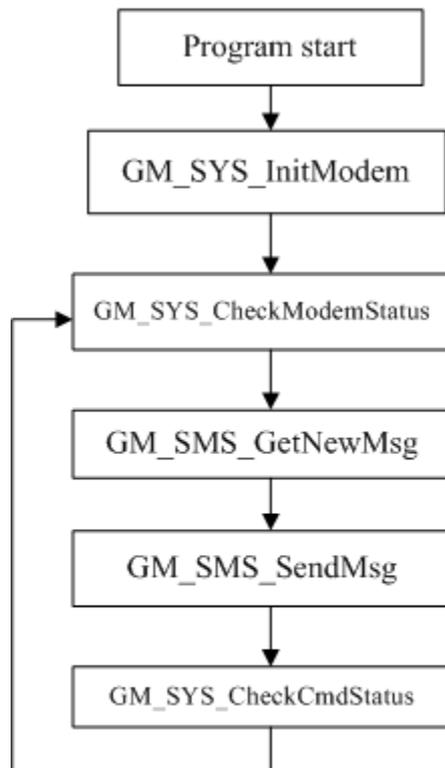
Table of Contents

CHAPTER 1 INTRODUCTION.....	3
1.1 DESIGN FLOWCHART	3
CHAPTER 2 GSM LIBRARY	4
2.1 DATA STRUCTURE DEFINE.....	4
2.2 FUNCTION	5
--SYSTEM Function--	5
2.2.1 GM_SYS_GetLibVersion	5
2.2.2 GM_SYS_GetLibDate.....	6
2.2.3 GM_SYS_InitModem.....	7
2.2.4 GM_SYS_CloseModem	8
2.2.5 GM_SYS_CheckModemStatus.....	9
2.2.6 GM_SYS_CheckCmdStatus	10
2.2.7 GM_SYS_CheckSignal	11
2.2.8 GM_SYS_CheckReg	12
--SMS Function--	13
2.2.9 GM_SMS_SendMsg.....	13
2.2.10 GM_SMS_GetNewMsg.....	14
--GPS Function--	15
2.2.11 GM_GPS_Set_type.....	15
2.2.12 GM_GPS_Get_data	16

Chapter 1 Introduction

1.1 Design Flowchart

SMS Design Flowchar



Chapter 2 GSM Library

2.1 Data structure define

There are some data structure that is useful when you program with GSM library.

VC

SMS:

```
//-- structure for sending/reading SMS
```

```
typedef struct STRENCODE_MSG{
    char phoneNumber[30];    //phone number
    char time[20];          //sms_time_stamp
    char msg[161];          //message's content
    unsigned char dataLen; //Message's length,
                           // Max length: 7-bit=160 words, UCS2=70 words(140 bytes)
    char mode;              //encode style: 0=GSM_7BIT, 8=GSM_UCS2(uni-code)
} strEncode_Msg;
```

SYSTEM:

```
//-- structure for setting system parameters
```

```
typedef struct SYS_PROFILE
{
    char PINCode[5]; //The pin code of SIM card, ex: "0000"
    int modemPort;  //modem port number.
    int hardware;   //hardware type. 0: GTM-201
}SYSProfile;
```

C#

SMS:

```
//-- class for sending/reading SMS
```

```
public class Encode_Msg
{
    public string phoneNumber;    //phone number
    public string time;          //sms_time_stamp
    public string msg;           //message's content
    public int mode;             //encode style: 0=GSM_7BIT, 8=GSM_UCS2(uni-code)
}
```

2.2 Function

--SYSTEM Function--

2.2.1 GM_SYS_GetLibVersion

VC
int GM_SYS_LibVersion(void);
C#
public static string GM_SYS_GetLibVersion()

Prototype:

Description:

Get Lib. version

Parameter:

no

Return:

VC
Version format = A.BC
C#
“A.BC”

Prototype:

2.2.2 GM_SYS_GetLibDate

Prototype:

VC
void GM_SYS_GetLibDate(char* libDate);
C#
public static string GM_SYS_GetLibDate()

Description:

Get the library's date

Parameter:

The library's date, format="Jul 21 2010"

Return:

no

2.2.3 GM_SYS_InitModem

Prototype:

VC
<code>int GM_SYS_InitModem(SYSProfile sysProfile);</code>
C#
<code>public static int GM_SYS_InitModem(string PinNum, int modem)</code>

Description:

Initialize Modem

*must use GM_SYS_CheckModemStatus() to check modem status later

Parameter:

VC
sysProfile: set system profile
C#
PinNum: PIN code
modem: port number of the modem

Return:

GM_NOERROR: success
GM_COMERROR: comport error
GM_INITERROR: init fail error

2.2.4 GM_SYS_CloseModem

Prototype:

VC
<code>int GM_SYS_CloseModem(int mode);</code>
C#
<code>public static int GM_SYS_CloseModem(int mode)</code>

Description:

Close the modem

*Please call GM_SYS_InitModem() to wake up modem after using GM_SYS_CloseModem(1) to shut down the modem.

Parameter:

mode: 0: close modem, but maintain it power on
1: close modem and set it power off (only for MiniOS7)

Return:

GM_NOERROR: no error
GM_CMDERROR: command error

2.2.5 GM_SYS_CheckModemStatus

Prototype:

VC
<code>int GM_SYS_CheckModemStatus(void);</code>
C#
<code>public static int GM_SYS_CheckModemStatus();</code>

Description:

Check modem status, and suggest you check it in your loop every time

Parameter:

no

Return:

GM_NOERROR: modem register success, can service

GM_NOREG: modem not registered, can't service

2.2.6 GM_SYS_CheckCmdStatus

Prototype:

VC
<code>int GM_SYS_CheckCmdStatus(void);</code>
C#
<code>public static int GM_SYS_CheckCmdStatus();</code>

Description:

Get the status of the command you sent

Parameter:

no

Return:

GM_BUSY:	modem busy, you can't send other command
GM_NOERROR:	success
GM_TIMEOUT:	time out
GM_CMDERROR :	command error
Other:	please refer to error codes of GSM.h

2.2.7 GM_SYS_CheckSignal

Prototype:

VC
int GM_SYS_CheckSignal(void);
C#
public static int GM_SYS_CheckSignal();

Description:

Check signal quality

Parameter:

no

Return:

Signal:	signal quality
0	-113 dBm or less
1	-111 dBm
2...30	-109... -53 dBm
31	-51 dBm or greater

2.2.8 GM_SYS_CheckReg

Prototype:

VC
<code>int GM_SYS_CheckReg(void);</code>
C#
<code>public static int GM_SYS_CheckReg();</code>

Description:

Check register

Parameter:

no

Return:

Register flag

- 0: not registered
- 1: registered, home network
- 2: not registered, and searching...
- 3: registration denied
- 4: unknown
- 5: registered, roaming

--SMS Function--**2.2.9 GM_SMS_SendMsg****Prototype:**

VC
<code>int GM_SMS_SendMsg(strEncode_Msg* strMsg);</code>
C#
<code>public static int GM_SMS_SendMsg(Encode_Msg strMsg)</code>

Description:

Send a message

* must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

strMsg: the message

Return: None

GM_NOERROR no error

GM_NOREG: not registered, or can't service

GM_BUSY: modem busy

2.2.10 GM_SMS_GetNewMsg

Prototype:

VC
<code>int GM_SMS_GetNewMsg(strEncode_Msg* msg);</code>
C#
<code>public static int GM_SMS_GetNewMsg(ref Encode_Msg strMsg)</code>

Description:

Get a new sms message

Parameter:

msg: new sms message

Return:

0: no new message

1: new message coming

--GPS Function--**2.2.11 GM_GPS_Set_type****Prototype:**

VC
<code>int GM_GPS_Set_type(int type);</code>
C#
<code>public static extern int GM_GPS_Set_type(int type);</code>

Description:

Set the type of the "GSP data" that get by the 3G modem.

Parameter:

type: The type of the GPS Data.

1: GGA

2: RMC

3: GSA

4: VTG

5: GNS

Return: None

Command sort number.

2.2.12 GM_GPS_Get_data

Prototype:

VC
<code>void GM_GPS_Get_data(LPSTR szResult);</code>
C#
<code>public static extern void GM_GPS_Get_data(byte[] ByteResult);</code>

Description:

Get a new GPS data

Parameter:

VC
LPSTR szResult: GPS receive string array
C#
<code>byte[] ByteResult</code> : GPS receive byte array Use the "PACNET.MISC.WideString" functino to transform the Byte array to string

Return:

no

Version Record

Version	By	Date	Description
1.0.0	Malo	2011/04/18	release
1.0.1	Tunglu	2017/06/30	release