GSM library for G-4500, µPAC-5000, iPAC-8000 and µPAC-7186E embedded controller

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

Trademark

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

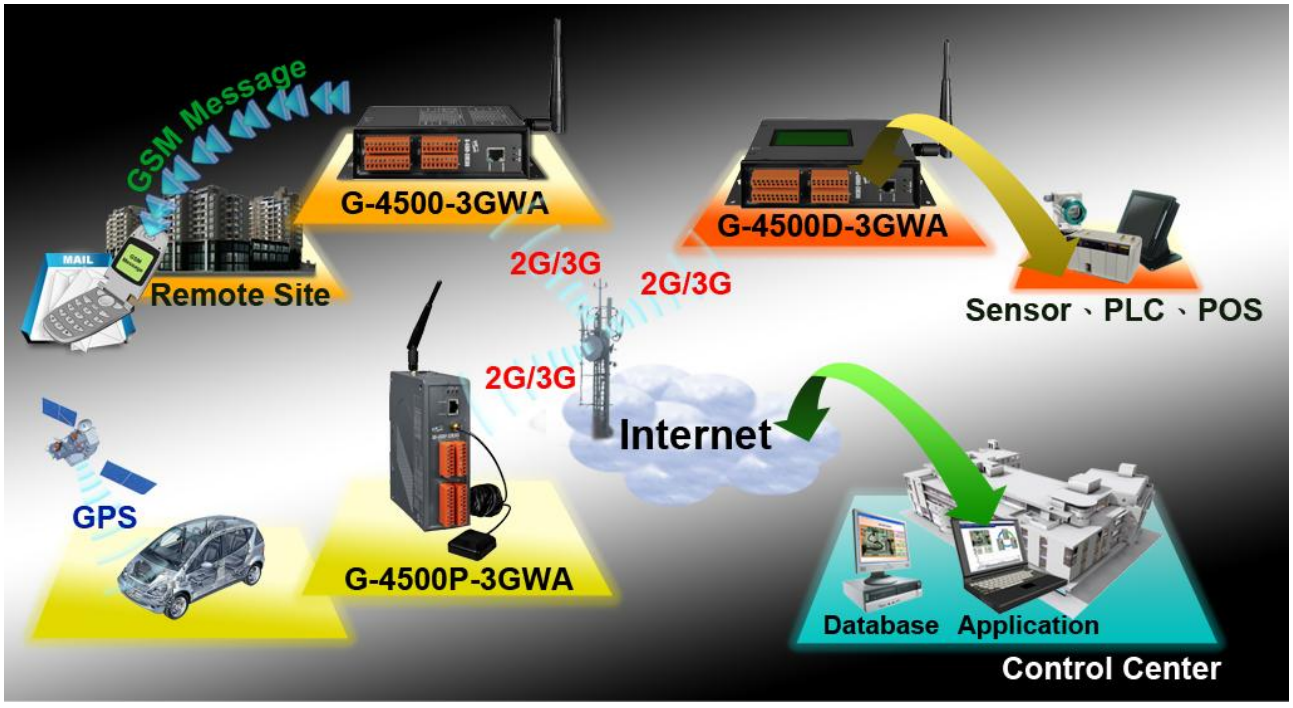
Table of Contents

Chapter 1 Introduction

1.1 2G/3G and PAC embedded controller

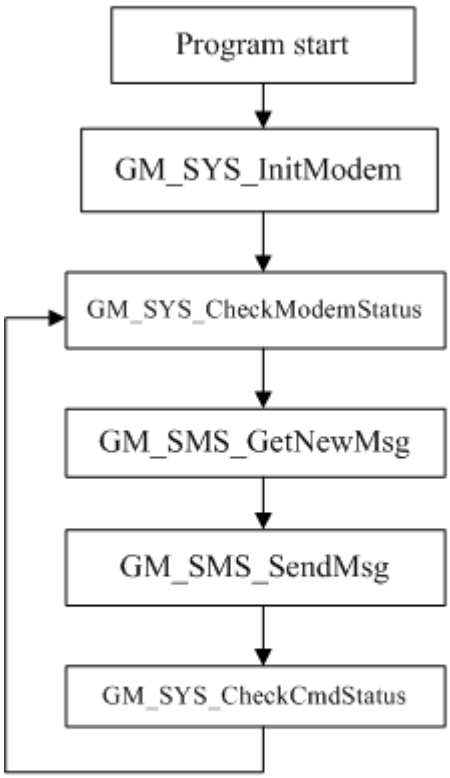
The 2G/3G is a service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service. 2G/3G is NOT related to GPS (the Global Positioning System), a similar acronym that is often used in mobile contexts. 2G/3G facilitates instant connections whereby information can be sent or received immediately as the need arises, subject to radio coverage. No dial-up modem connection is necessary. This is why 2G/3G users are sometimes referred to be as being "always connected". Immediacy is one of the advantages of 2G/3G (and SMS) when compared to Circuit Switched Data. High immediacy is a very important feature for time critical applications.

ICP DAS provides the 2G/3G library for PAC embedded controller. The library is an easy way to applying the 2G/3G service in the embedded controller. Otherwise, ICP DAS supports many IO modules and GPS modules for users. Therefore, there are many application architectures to apply in the system. Or users can integrate other controller system with 2G/3G library. The follows is a standard application architecture.

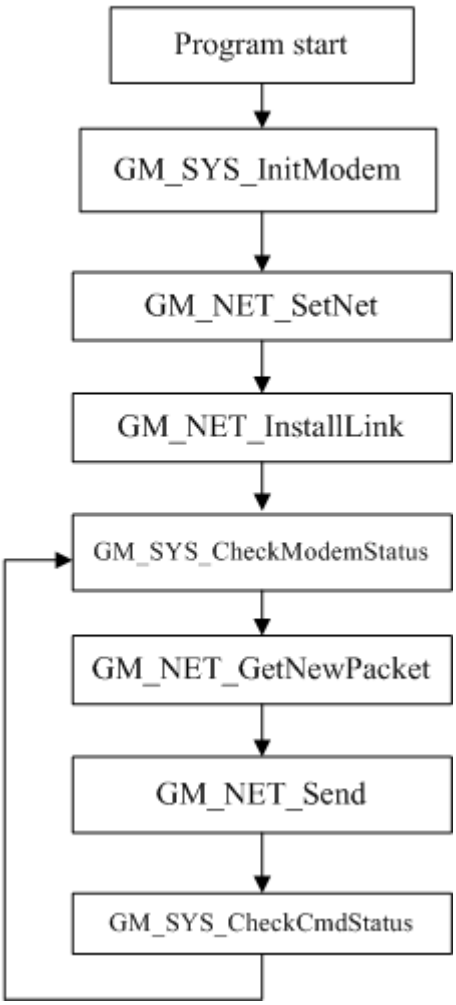


1.2 Design Flowchart

SMS Design Flowchar



GPRS 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.

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;
```

GPRS:

```
//-- structure for reading GPRS sockets
typedef struct GPRSDATA{
    char data[1500]            //data
    int dataLen;               //data length
    char IP[16];               //IP of the server，000.000.000.000 and '0x00', total 16 byte
    unsigned int port;         //TCP/UDP port of the server
    char link;                 //data of link[n]
} GPRSData;

//-- structure for setting network
typedef struct NET_PROFILE
{
    char APN[60];     //APN for network provided by your cellular provider
    char user[32];    //username for network provided by your cellular provider
    char pw[32];      //password for network provided by your cellular provider
    char DnsServerIP[16];  //The most basic task of DNS is to translate hostnames
                           //such as www.icpdas.com to IP address such as 96.9.41.131.
} NetProfile;
```

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: other hardware,
                      //1: G-4500, 2: uPAC-5000, 3: iP-8000
}SYSProfile;
```

2.2 Function

Before using:

1. If you use multi-client, 2 client link to the same IP server, your port must set different, or Library will couldn't recognize new packet from which link. (3G series support multi-clients, 2G series support single-client)
2. Server couldn’t send packet data to the client over GPRS very fast. Suggest you send packet data no more than 1 packet / 1 second.
3. The Library will use a Timer. If you also use LCD function, you will remain Timer 1, 2 to use.

2.2.1 GM_SYS_GetLibVersion

Prototype:

int GM_SYS_LibVersion(void);

Description:

Get Lib. version

Parameter:

no

Return:

version format = A.BC

2.2.2 GM_SYS_GetLibDate

Prototype:

void GM_SYS_GetLibDate(char* libDate);

Description:

Get lib. date

Parameter: None

a string of lib. date, format="Jul 21 2010"

Return:

no

2.2.3 GM_SYS_InitModem

Prototype:

int GM_SYS_InitModem(SYSProfile sysProfile);

Description:

Initialize Modem, and turn it on

*must use GM_SYS_CheckModemStatus() to check modem status later

Parameter:

sysProfile: set system profile

Return:

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

2.2.4 GM_SYS_CloseModem

Prototype:

int GM_SYS_CloseModem(int mode);

Description:

Close the modem

(1) use it before you want to finish the program

(2) or your case must save the power of the battery during device working

after GM_SYS_CloseModem(1), you must use GM_SYS_InitModem() to wake up modem

Parameter:

mode:	0:	close modem, but maintain it power on
	1:	close modem and set it power off

Return:

GM_NOERROR:	no error
GM_CMDERROR:	command error

2.2.5 GM_SYS_CheckModemStatus

Prototype:

int GM_SYS_CheckModemStatus(void);

Description:

check modem status, 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:

int GM_SYS_CheckCmdStatus(void);

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:

int GM_SYS_CheckSignal(void);

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:

int GM_SYS_CheckReg(void);

Description:

Check register

Parameter:

no

Return:

register flag, the value will fill here, when get value from modem

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:

int GM_SMS_SendMsg(strEncode_Msg* strMsg);

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:

int GM_SMS_GetNewMsg(strEncode_Msg* msg);

Description:

Get new sms message

Parameter:

msg: new sms message

Return:

0: no new message

1: new message coming

--3G / GPRS data Transmission Function--

2.2.11 GM_NET_SetNet

Prototype:

int GM_NET_SetNet(NetProfile netProfile);

Description:

Set Net profile data

Parameter:

netProfile: Net profile data

Return:

GM_NOERROR	no error
GM_ERROR	error

2.2.12 GM_NET_InstallLink

Prototype:

int GM_NET_InstallLink(int n, int tcp, char* serverIP, unsigned int serverPort);

Description:

Built TCP/UDP link

Parameter:

n	link number (0~9), 3G:0~9, 2G:0
tcp	client type, tcp=1 for TCP client ; tcp=0 for UDP client
serverIP	IP or Domain name of the server, ex: "61.111.222.333", "test.com.tw"
serverPort	TCP/UDP Port of the server (1~65535), ex: 1234

Return:

GM_NOERROR	correct parameter to install TCP/UDP link
GM_CMDERROR	command error

2.2.13 GM_NET_CloseNet

Prototype:

int GM_NET_CloseNet(void);

Description:

Close NetWork

Parameter:

no

Return:

GM_NOERROR	no error
GM_CMDERROR	command error
GM_BUSY:	modem busy

2.2.14 GM_NET_GetIP

Prototype:

void GM_NET_GetIP(char* ipaddr);

Description:

Get local IP

Parameter:

ipaddr: IP string, format: char ipaddr[16];

Return:

no

2.2.15 GM_NET_CloseLink

Prototype:

int GM_NET_CloseLink(int n);

Description:

Close client link[n]

Parameter:

n: client link[n], 3G:0~9, 2G:0

Return:

GM_NOERROR	no error
GM_CMDERROR	command error
GM_BUSY:	modem busy

2.2.16 GM_NET_GetLinkStatus

Prototype:

int GM_NET_GetLinkStatus(int n);

Description:

get status of Link[n]

Parameter:

n: link[n], 3G:0~9, 2G:0

Return:

status: 0=not link, 1=linked

2.2.17 GM_NET_Send

Prototype:

int GM_NET_Send(char link, char* data, int dataLen);

Description:

Send a packet

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

Parameter:

link:	link number, 3G:0~9, 2G:0
data:	data
dataLen:	data length, Max.=1000

Return:

GM_NOERROR	no error
GM_CMDERROR	command error
GM_BUSY:	modem busy

2.2.18 GM_NET_GetNewPacket

Prototype:

int GM_NET_GetNewPacket(GPRSData* gprsData);

Description:

Get the new packet

Parameter:

gprsData: new data packet

Return:

0: no new packet

1: new packet coming

Version Record

Version	By	Date	Description
1.0.0	Malo	2011/03/04	release