GSM_U2 Library

for G-4511, G-4513 series

User's Manual V1.0.1

High Quality, Industrial Data Acquisition, and Control Products

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.

Contact us

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email：service@icpdas.com

Table of Contents

1. Introduction

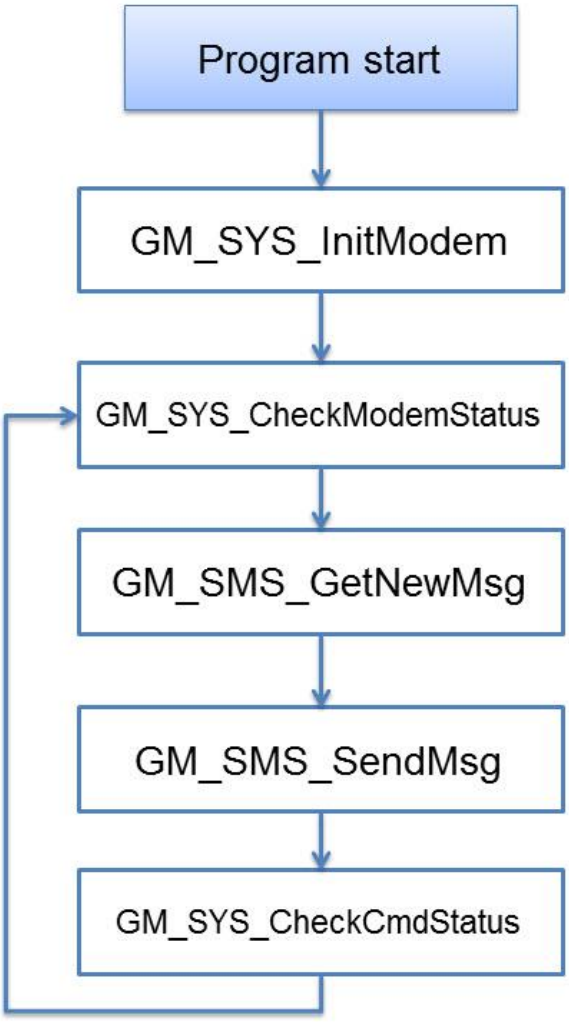
1.1 2G/3G and PAC Embedded Controller

2G/3G is a service that allows information to be sent and received across a mobile telephone network. It supports CSD (Circuit Switched Data), SMS (Short Message Service) and GPRS (General Packet Radio Service). GPRS is NOT related to GPS (the Global Positioning System), a similar acronym that is often used in mobile contexts. 2G/3G offers instant connections whereby information can be sent or received immediately as the need arises, subject to radio coverage. 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.

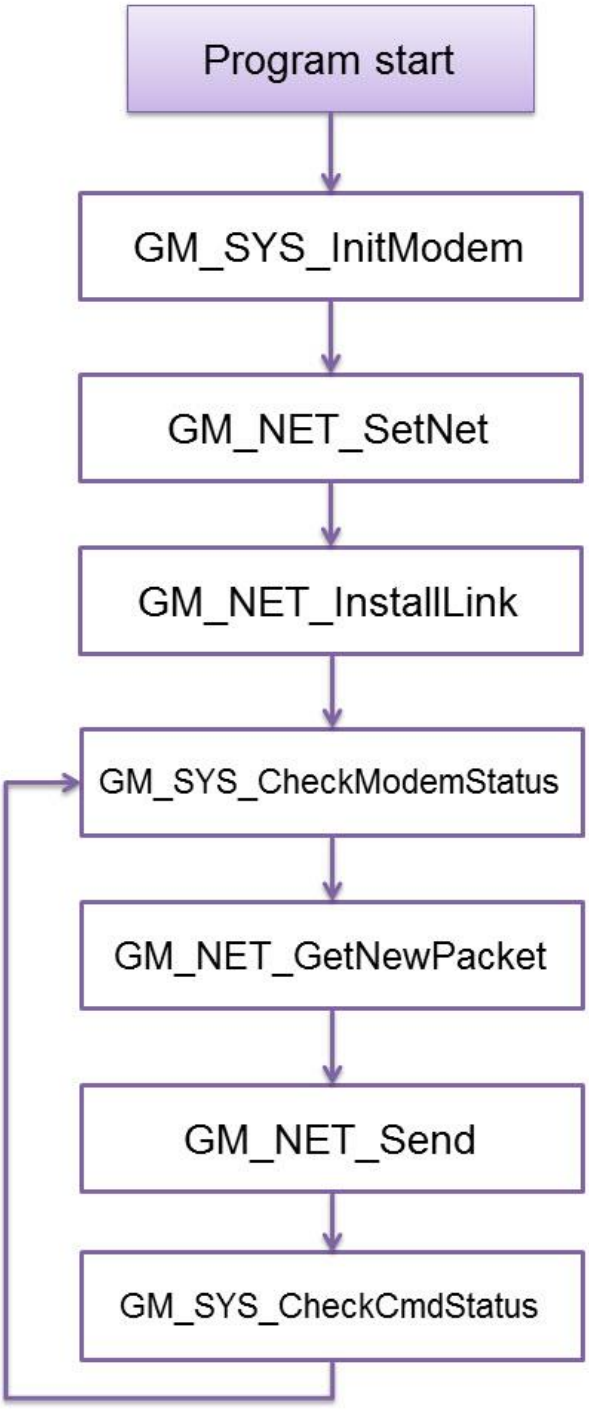
ICP DAS provides the 2G/3G library for PAC embedded controller. The library is an easy way to applying the 2G/3G service in the embedded controller. Otherwise, ICP DAS supports many IO modules and GPS modules for users. Therefore, there are many application architectures to apply in the system. Or users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

1.2 Design Flowchart

SMS Design Flowchart



GPRS Design Flowchart



2. GSM_U2 Library

2.1 Data Structure Define

There are some data structure that is useful when you program with GSM_U2 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 fromIP[16];       //IP of the server，000.000.000.000 and '0x00', total 16 byte
    unsigned int port;     //TCP/UDP port of the server
    int 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: G-4511、G-4513 series, 1: G-4500
}SYSProfile;
```

Tips & Warnings

1. The GSM_U2 library needs OS7_COM.lib. Please include it.
2. The speed of GPRS is less than 1 packet / 1 second.
3. The GSM_U2 library needs the Timer that installed by "InstallUserTimer()". Please don't collide with it.

2.2 SYSTEM Function

Function definition	Description
GM_SYS_GetLibVersion	Get Library version
GM_SYS_GetLibDate	Get Library date
GM_SYS_InitModem	Initialize Modem
GM_SYS_CloseModem	Close the modem
GM_SYS_CheckModemStatus	Check modem status, and suggest you check it in your loop every time
GM_SYS_CheckCmdStatus	Get the status of the command you sent
GM_SYS_CheckSignal	Check signal quality
GM_SYS_CheckReg	Check register
GM_SYS_EnableNITZ	Enable NITZ function
GM_SYS_NITZUpdateRTC	Update the RTC of the System by NITZ
GM_SYS_CheckNITZ	Check the status of NITZ

2.2.1 GM_SYS_GetLibVersion

Get library version.

Syntax

```
int GM_SYS_GetLibVersion(void);
```

Parameters

None

Return values

Version format = A.BC

2.2.2 GM_SYS_GetLibDate

Get library date.

Syntax

```
void GM_SYS_GetLibDate(
    char* libDate
);
```

Parameters

libDate
a string of lib. date, format="Jul 21 2014"

Return values

None

2.2.3 GM_SYS_InitModem

Initialize Modem.

**must use GM_SYS_CheckModemStatus() to check modem status later

Syntax

```
int GM_SYS_InitModem(
    SYSProfile sysProfile
);
```

Parameters

sysProfile
set system profile

Return values

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

2.2.4 GM_SYS_CloseModem

Close the modem.

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

Syntax

```
int GM_SYS_CloseModem(
    int mode
);
```

Parameters

mode

0：close modem, but maintain it power on

1：close modem and set it power off

Return values

GM_NOERROR：no error

GM_CMDERROR：command error

2.2.5 GM_SYS_CheckModemStatus

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

Syntax

```
int GM_SYS_CheckModemStatus(void);
```

Parameters

None

Return values

GM_NOERROR：modem register success, can service
GM_NOREG：modem not registered, can't service

2.2.6 GM_SYS_CheckCmdStatus

Get the status of the command you sent.

Syntax

```
int GM_SYS_CheckCmdStatus(void);
```

Parameters

None

Return values

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_U2.h

2.2.7 GM_SYS_CheckSignal

Check signal quality.

Syntax

```
int GM_SYS_CheckSignal(void);
```

Parameters

None

Return values

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

Check register.

Syntax

```
int GM_SYS_CheckReg(void);
```

Parameters

None

Return values

Register flag

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

2.2.9 GM_SYS_EnableNITZ

Enable NITZ function.

**NITZ function can auto-adjust RTC of the system at the moment of the modem registering to GSM system.
**Please call "GM_SYS_NITZUpdateRTC" to update RTC after GM_SYS_EnableNITZ(1).

Syntax

```
void GM_SYS_EnableNITZ(
    int nitz
);
```

Parameters

nitz
0：disable
1：enable

Return values

None

2.2.10 GM_SYS_NITZUpdateRTC

Update the RTC of the System by NITZ.

**Notice: this function will disable all 3G/GSM function about 1~2 minutes.
**Please use this function after you stop all 3G/GSM function

Syntax

```
void GM_SYS_NITZUpdateRTC(void);
```

Parameters

None

Return values

None

2.2.11 GM_SYS_CheckNITZ

Check the status of NITZ.

Syntax

```
void GM_SYS_CheckNITZ(void);
```

Parameters

None

Return values

0：fail to update RTC
1：success
2：updating

2.3 SMS Function

Function definition	Description
GM_SMS_SendMsg	Send a message
GM_SMS_GetNewMsg	Get a new sms message

2.3.1 GM_SMS_SendMsg

Send a message.

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

Syntax

```
int GM_SMS_SendMsg(
    strEncode_Msg* strMsg
);
```

Parameters

strMsg
the message that will be sent.

Return values

GM_NOERROR：no error
GM_NOREG：not registered, or can't service
GM_BUSY：modem busy

2.3.2 GM_SMS_GetNewMsg

Get a new sms message.

Syntax

```
int GM_SMS_GetNewMsg(
    strEncode_Msg* msg
);
```

Parameters

msg

new sms message

Return values

0：no new message

1：new message coming

2.4 3G / GPRS Data Transmission Function

Function definition	Description
GM_NET_SetNet	Set Net profile data
GM_NET_InstallLink	Built TCP/UDP link
GM_NET_CloseNet	Close Network
GM_NET_GetIP	Get local IP
GM_NET_CloseLink	Close client link[n]
GM_NET_GetLinkStatus	Get status of Link[n]
GM_NET_Send	Send a packet
GM_NET_GetNewPacket	Get the new packet

2.4.1 GM_NET_SetNet

Set Net profile data.

Syntax

```
int GM_NET_SetNet(
    NetProfile netProfile
);
```

Parameters

netProfile
Net profile data

Return values

GM_NOERROR：no error
GM_CMDERROR：command error

2.4.2 GM_NET_InstallLink

Built TCP/UDP link.

Syntax

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

Parameters

n

link number (0~6)
3G (G-4513 series)：0~6
2G (G-4511 series)：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

2.4.3 GM_NET_CloseNet

Close Network.

Syntax

```
int GM_NET_CloseNet(void);
```

Parameters

None

Return values

GM_NOERROR：no error
GM_CMDERROR：command error
GM_BUSY：modem busy

2.4.4 GM_NET_GetIP

Get local IP.

Syntax

```
void GM_NET_GetIP(
    char* ipaddr
);
```

Parameters

ipaddr

IP string, format: char ipaddr[16];

Return values

None

2.4.5 GM_NET_CloseLink

Close client link[n].

Syntax

```
int GM_NET_CloseLink(
    int n
);
```

Parameters

n

3G (G-4513 series)：0~6

2G (G-4511 series)：0

Return values

GM_NOERROR：no error

GM_CMDERROR：command error

GM_BUSY：modem busy

2.4.6 GM_NET_GetLinkStatus

Get status of Link[n].

Syntax

```
int GM_NET_GetLinkStatus(
    int n
);
```

Parameters

n

3G (G-4513 series)：0~6

2G (G-4511 series)：0

Return values

0：not link

1：linked

2.4.7 GM_NET_Send

Send a packet.
**must use "GM_SYS_CheckCmdStatus()" to check status later

Syntax

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

Parameters

link
link number
3G (G-4513 series)：0~6
2G (G-4511 series)：0

data
data that will be sent

dataLen
data length, Max.=1000

Return values

GM_NOERROR：no error
GM_CMDERROR：command error
GM_BUSY：modem busy

2.4.8 GM_NET_GetNewPacket

Get the new packet.

Syntax

```
int GM_NET_GetNewPacket(
    GPRSData* gprsData
);
```

Parameters

gprsData
new data packet

Return values

0：no new packet
1：new packet coming

2.5 APIs and Demo References

For example, send and receive sms message

```
#include <conio.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include "../lib/G4500.h"
#include "../lib/GSM_U2.h"
#include "../lib/OS7_COM.h"
#include "../lib/MCU2LIB.h"

int main(void)
{
	int iAction=1, quit=1;
	int i, j, tmp;
	int Result=0;
	strEncode_Msg RecMsg, SendMsg;
	char sendNumber[20];
	int send_n;
	int sendStatus = 0;
	SYSProfile sysProfile;

	//-- init
	InitLib();

	/*---- init modem----*/
	strcpy(sysProfile.PINCode, "0000"); /*The pin code of SIM card, ex: "0000"*/
	sysProfile.modemPort = 4;   /*modem port number. G-4500 = 4, uP-5000 = 11*/
	sysProfile.hardware = 0;       /*hardware type. 1: G-4500, 2: uPAC-5000, 0: Other*/
	GM_SYS_SetPowerFunction(powerFunction); /*set power-control function*/


	if((Result = GM_SYS_InitModem(sysProfile)) == GM_NOERROR)
		Print("init_modem success!!\r\n");
```

```
else
{
    Print("init_modem fail!! return value is %d\r\n", Result);
    return 1;
}

/*--check Could the modem service? --*/
while(GM_SYS_CheckModemStatus() != GM_NOERROR)
{
    Print("wait modem register...\r\n");
    DelayMs(1000);
}
Print("modem registered!!\r\n");

while(iAction!=0)
{
    iAction=0;
    quit=0;
    Print("1) Send ASCII messages\r\n");
    Print("2) check signal quality\r\n");
    Print("3) check registered?\r\n");
    Print("0) Quits demo program\r\n");
    Print("Choose an option and press [Enter]: ");
    Scanf("%d", &iAction);

    switch(iAction)
    {
    case 1:     /*Send ASCII messages*/
        Print("Send ASCII messages\r\n");

        Print("Please Input Phone Number =");
        Scanf("%s", sendNumber);

        Print("How many message do you want to send?\r\n");
        Scanf("%d", &send_n);
        i = 0;
        Print("== start to send sms, please press <ESC> to exit ==\r\n");
        while(1)
        {
            /* press "ESC" to exit */
```

```
                    if(Kbhit())
                    {
                        tmp = Getch();
                        if( tmp == 27 || tmp == 'q')
                            break;
                    }

/*--(1) check that could the modem service, if it can't, skip operating the modem below --*/
                    if(GM_SYS_CheckModemStatus() != GM_NOERROR)
                        continue;

                    /*--(2) send messages, when the modem can service--*/
                    if(i<send_n)
                    {
                        switch((sendStatus=GM_SYS_CheckCmdStatus()))
                        {
                        case GM_READY:
                            Print("sending message(%d)....\r\n", i);

                            strcpy(SendMsg.phoneNumber, sendNumber);
                            SendMsg.mode = GSM_7BIT;
                            sprintf(SendMsg.msg, "GSM_Test(%2d)", i);
                            SendMsg.dataLen = strlen(SendMsg.msg);
                            GM_SMS_SendMsg(SendMsg);
                            break;
                        case GM_NOERROR:
                            Print("send success!!\r\n");
                            i++;
                            break;
                        case GM_BUSY:  //sending, and waiting reply
                            break;
                        default:
                            Print("send error, and skip this one, error code=%d\r\n",
                            sendStatus);
                            i++;
                            break;
                        }
                    }
                    /*--(3) if any sms message come in, print it --*/
                    if( GM_SMS_GetNewMsg(&RecMsg) != 0)
```

```
                {
                    printMsg(RecMsg);
                }
            }
            break;

        case 2://check signal quality
            Result = GM_SYS_CheckSignal();
            Print("signal value = %d\r\n", Result);
            break;

        case 3://check register value
            Result = GM_SYS_CheckReg();
            Print("register value = %d    (0:no register, 1:registered, 2:registering)\r\n",
            Result);
            break;

        case 0:
        default:
            quit=1;
            break;

        }//end switch()

        if(!quit)
        {
            Print("Press any key to continue...\r\n");
            Getch();
        }
    }   /*end while(1)*/

    /*must close before program ending to release you resource*/
    /*-- Close the modem, 0:not turn off modem, 1:turn off modem*/
    GM_SYS_CloseModem(0);

    Print("Please press ENTER to exit...\r\n");
    Getch();
    return 0;
}
```

For example, TCP client Demo

```
#include <conio.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include "../lib/G4500.h"
#include "../lib/GSM_U2.h"
#include "../lib/OS7_COM.h"
#include "../lib/MCU2LIB.h"

int main(void)
{
   int iAction=1, quit=1;
   int i, j,tmp;
   int Result=0;
   int send_n;
   NetProfile netProfile;
   SYSProfile sysProfile;
   GPRSData gprsData;
   char serverIP[16];
   int serverPort;
   long socket_n;
   int netSendStatus = 0;
   char myIP[16];

   InitLib();

   /*---- init modem ----*/
   strcpy(sysProfile.PINCode, "0000");     /*The pin code of SIM card, ex: "0000"*/
   sysProfile.modemPort = 4;     /*modem port number. G-4500 = 4, uP-5000 = 11*/
   sysProfile.hardware = 0;        /*hardware type. 1: G-4500, 2: uPAC-5000, 0: Other*/
   GM_SYS_SetPowerFunction(powerFunction); /*set power-control function*/

   if( (Result = GM_SYS_InitModem(sysProfile)) == GM_NOERROR)
         Print("init_modem success!!\r\n");
   else{
         Print("init_modem fail!! return value is %d\r\n", Result);
         return 1;
```

```
}

/*-- check Could the modem service? --*/
while(GM_SYS_CheckModemStatus() != GM_NOERROR)
{
    Print("wait modem register...\r\n");
    DelayMs(1000);
}
Print("modem registered!!\r\n");

while(iAction!=0)
{
    iAction=0;
    quit=0;
    Print("1) TCP client demo\r\n");
    Print("0) Quits demo program\r\n");
    Print("Choose an option and press [Enter]: ");
    Scanf("%d", &iAction);
    Print("\r\n");

    switch(iAction)
    {
      case 1:   //TCP client demo
      Print("TCP client demo start\r\n");

        /* set Network profile */
        /* APN for network provided by your cellular provider*/
        strcpy(netProfile.APN, "INTERNET");
        /*username for network provided by your cellular provider */
        strcpy(netProfile.pw, "guest");
        /*password for network provided by your cellular provider */
        strcpy(netProfile.user, "guest");
        /* The most basic task of DNS is to translate hostnames such as
        www.icpdas.com to IP address such as 96.9.41.131 */.
        strcpy(netProfile.DnsServerIP, "");      /*empty string = system default value*/
        GM_NET_SetNet(netProfile);

        /*set ip, port of server */
        Print("please input server IP:(ex: 74.125.227.48)\r\n");
        Scanf("%s", serverIP);
```

```
        Print("please input server Port:(ex: 80)\r\n");
        Scanf("%d", &serverPort);

        /*--(1) install link[0], GM_NET_InstallLink(0, 0, serverIP, serverPort) for UDP--*/
        GM_NET_InstallLink(0, 1, serverIP, serverPort);
        Print("linking...\r\n");
        socket_n = 0;    //count for the packets

        while (1)
        {
            /* press "ESC" to exit */
            if(Kbhit())
            {
                tmp = Getch();
                if( tmp == 27 || tmp == 'q')
                    break;
            }

/*-- (2) check that could the modem service, if it can't, skip operating the modem below --*/
            if(GM_SYS_CheckModemStatus() != GM_NOERROR)
                continue;

            if(GM_NET_GetLinkStatus(0)!=1)
                continue;
            else
                GM_NET_GetIP(myIP);

            /*--(3) send the data to server, and when LinkStatus[1]=1 --*/
            switch((netSendStatus=GM_SYS_CheckCmdStatus()))
            {
              case GM_READY:
                Print("sending package[%8ld]..., myIP = %s\r\n", socket_n, myIP);
                gprsData.link = 0;
                sprintf(gprsData.data, "-<%8ld>-TCP send test!!", socket_n);

                gprsData.dataLen = strlen(gprsData.data);
                if(GM_NET_Send(gprsData.link,gprsData.data,gprsData.dataLen)!=G
                M_NOERROR)
                    Print("can't send package[%8ld]\r\n");
                break;
```

```
                    case GM_NOERROR:
                        Print("send success!!\r\n");
                        socket_n++;
                        break;
                    case GM_BUSY:  /*sending, and waiting reply*/
                        break;
                    default:
                        Print("send error, and re-send again, error code=%d\r\n",
                        netSendStatus);
                        break;
            }

            /*-- (4) if any new data packet come in, print it --*/
                if(GM_NET_GetNewPacket(&gprsData) != NULL)
                {
                    Print("\n== new data packet come in\r\n");
                    printPacket(gprsData);
                }
        }
        GM_NET_CloseLink(0); /*--Close client link[n], 3G:0~6, 2G:0 --*/
        DelayMs(1000);
        GM_NET_CloseNet();
        DelayMs(1000);

        break;
    }

}
```

3. Revision History

Revision	Date	Author	Description
1.0.0	2014/09/03	William	Release version
1.0.1	2015/02/16	William	Modify the description of GM_SYS_CheckReg.