TCP server and Modbus TCP library for G-4500-2G Series

User's Manual

Version 1.00

Important Notices

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.

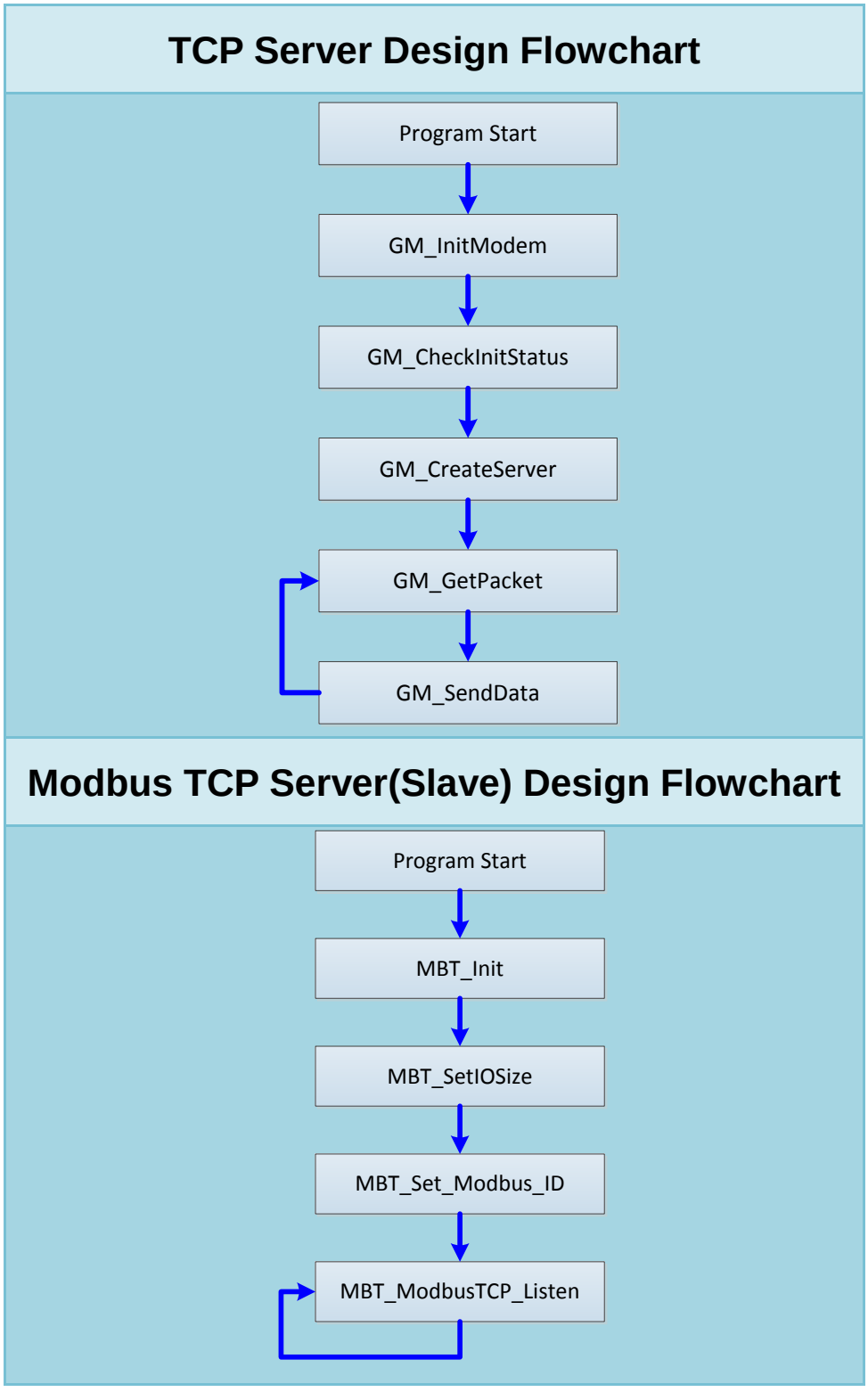
Copyright

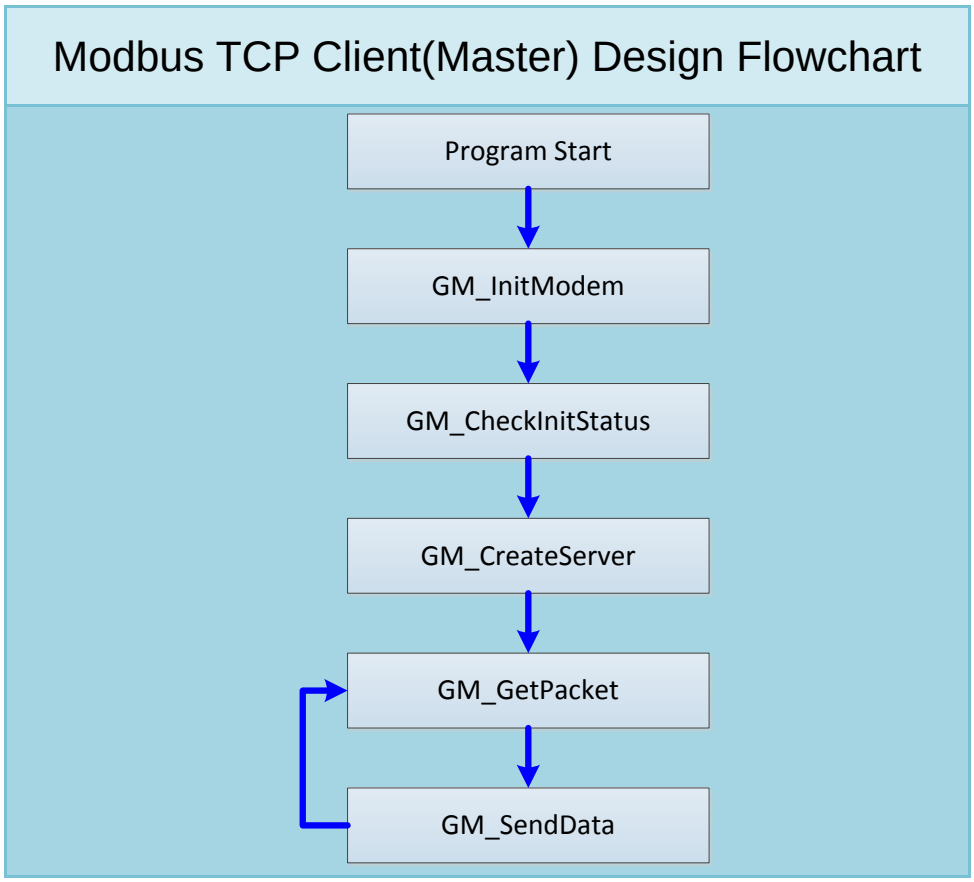
Trademark

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

CHAPTER 1 INTRODUCTION

1.1 Design Flowchart




Modbus TCP Client(Master) Design Flowchart
Program Start
GM_InitModem
GM_CheckInitStatus
GM_CreateServer
GM_GetPacket
GM_SendData

CHAPTER 2 TCP SERVER LIBRARY

2.1 Data structure define

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

```
//-- structure for setting network
typedef struct _SYS_PROFILE_
{
    char WAPN[30];        //APN for network provided by your cellular provider
    char WName[30];       //username for network provided by your cellular provider
    char WPW[30];         // password for network provided by your cellular provider
    char WPINCODE[5];     // PIN code for SIM card ex: "0000"
} SYS_PROFILE;

//-- structure for Client status
typedef struct _CLIENT_
{
    int IsConnected;      // Client status of connect, 1:connect,0:disconnect
    char MyIP[16];        // Client IP ex:”192.168.0.1”
    char MyProtocol[4];   // TCP or UDP
    int Port;             // Client port
    char State[20];       // Close or connected
}Client_Info;

//-- structure for Server status
typedef struct _SERVER_
{
    int IsListing;              // Server status
    char SIP[16];               // Server IP
    long SPort;                 // Server port
    Client_Info SMyClient[7];   // Client status
}Server_Info;
```

```
//-- structure for reading GPRS sockets
typedef struct _GPRS_DATA_
{
    char RecvData[1600];          // data
    char Client_IP[16];           // client IP, ex:"192.168.0.1"
    unsigned int Client_Socket;   // Socket number
    unsigned int RecvDataLen;     // data length
}_GPRS_DATA;
```

2.2 Function

Notice:

1. The TCP Server library needs OS7_COM.lib and G-4500.lib please include it.
2. The Server need public IP.
3. The GSM library needs the Timer that installed by "InstallUserTimer()". Please don't collide with it.

Function list	
Function	Description
GM_InitModem	Initialize Modem
GM_CheckInitStatus	Check initial status
GM_CreateServer	Create server
GM_CheckModemSignal	Get modem signal
GM_CheckModemRegister	Check modem register
GM_GetServerInfo	Get TCP server status
GM_SendData	Send data to client
GM_GetPacket	Receive data from client
GM_CloseClient	Close socket
GM_CloseServer	Close TCP Server
GM_ResetModem	Restart Modem
GM_CloseLib	Close Modem

GM_InitModem

Prototype:

void GM_InitModem(SYS_PROFILE SysProfile);

Description:

Initialize Modem

Parameter:

SYS_PROFILE: system profile

Return:

No

GM_CheckInitStatus

Prototype:

int GM_CheckInitStatus(void);

Description:

Check modem status

Parameter:

No

Return:

0: initial is not finish

1: initial is finish

GM_CreateServer

Prototype:

int GM_CreateServer(char *LocalPort);

Description:

Create server

Parameter:

LocalPort: TCP port of server, ex:”65535”, maximum is 65535

Return:

1: No error
0: command processing
-1: command error
-2: modem is not initial
-3: Port more then 65535

GM_CheckModemSignal

Prototype:

int GM_CheckModemSignal(int *Signal);

Description:

Check signal quality

Parameter:

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

Return:

1: No error

0: command processing

-1: command error

-2: modem is not initial

GM_CheckModemRegister

Prototype:

int GM_CheckModemRegister(int *GM_CREG);

Description:

Check modem register

Parameter:

GM_CREG: Register flag,

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

Return:

1: No error
0: command processing
-1: command error
-2: modem is not initial

GM_GetServerInfo

Prototype:

int GM_GetServerInfo(Server_Info *ServerInfo);

Description:

Get TCP server status

Parameter:

ServerInfo: Get server information.

Return:

1: No error

0: command processing

-1: command error

-2: modem is not initial

GM_SendData

Prototype:

int GM_SendData(char Client, char *SendData, int SendDataLen);

Description:

Send data to client

Parameter:

Client:	Client number, 0~6
SendData:	Data
SendDataLen:	Data length, Max=1000

Return:

1: No error
0: command processing
-1: command error
-2: modem is not initial
-3: Client error
-4: data length > 1000
-5: Client is not conected

GM_GetPacket

Prototype:

int GM_GetPacket(_GPRS_DATA *gprsData);

Description:

Receive data from client

Parameter:

gprsData: data packet

Return:

1: has data in buffer

-1: no data in buffer

-2: modem is not initial

GM_CloseClient

Prototype:

int GM_CloseClient(int SocketNum);

Description:

Close socket

Parameter:

SocketNum: Socket number, 0~6.

Return:

1: No error

0: command processing

-1: command error

-2: modem is not initial

-3: client number error

GM_CloseServer

Prototype:

int GM_CloseServer(void);

Description:

Close TCP server

Parameter:

No

Return:

1: No error

0: command processing

-1: command error

-2: modem is not initial

GM_ResetModem

Prototype:

int GM_ResetModem(void);

Description:

Restart modem

Parameter:

No

Return:

1: No error

-2: modem is not initial

GM_CloseLib

Prototype:

void GM_CloseLib(void);

Description:

Close modem

Parameter:

No

Return:

No

CHAPTER 3 MODUBS TCP MASTER LIBRARY

3.1 Function

Function list	
Function	Description
InitGPRSModbus	Give the 4 array pointers to InitModbus
GPRSModbusTCP_Master2Slave	Get DI/DO/AI/DO data
GPRSModbusTCP_Receive	Assignment data to DI/DO/AI/AO memory

InitGPRSModbus

Prototype:

Void InitGPRSModbus(unsigned char *iPointer_DI, unsigned char iPointer_DO,
Int iPoint_AI, int iPoint_AO)

Description:

Give the 4 array point to InitModbus

Parameter:

DI_Address: DI array

DO_Address: DO array

AI_Address: AI array

AO_Address: AO array

Return:

No

GPRSModbusTCP_Master2Slave

Prototype:

int GPRSModbusTCP_Mater2Slave(char *ModCMD, unsigned char cNetID,
unsigned char cFunction, int
iLocalMemoryBaseAddress, int
iRemoteMemoryBaseAddress, int iIOCount);

Description:

Get DI/DO/AI/DO data

Parameter:

ModCMD: Assembler Modbus TCP command

cNetID: 0~0xFF, the Net ID(station Number) of destination Modbus/Slave device

cFunction: Modbus function code

iLocalMemoryBaseAddress: internal register base address that you want to deal

iRemoteMemoryBaseAddress: register base address of device that you want to deal

iIOCount: count of coils or registers that you want to deal

Return:

0: No Error

>80: Exception Code

GPRSModbusTCP_Receive

Prototype:

int GPRSModbusTCP_Receive(char *cInBuf, int iLength);

Description:

Assignment data to DI/DO/AI/AO memory

Parameter:

cInBuf: Response data from Modbus TCP slave

iLength: Response length form Modbus TCP slave

Return:

0: No Error

>80: Exception Code

CHAPTER 4 MODBUS TCP SLAVE LIBRARY

4.1 Function

Notice:

1. The Max input data is 1024 Bytes.

Function list	
Function	Description
MBT_GetVer	Get library version
MBT_Init	Give the 4 array pointers to InitModbus
MBT_SetIOSize	Set Memory size
MBT_Set_ModbusID	Set local unit identifier
MBT_ModbusTCP_Listen	Assembler Modbus TCP command and disassembler Modbus TCP data

MBT_GetVer

Prototype:

Void MBT_GetVer(char *VerBuf);

Description:

Get Lib. version

Parameter:

A String of library, format = "2013/10/17 Ver 1.0"

Return:

No

MBT_Init

Prototype:

Void MBT_Init(unsigned char *DI_Address, unsigned char *DO_Address,
unsigned short *AI_Address, unsigned short *AO_Address);

Description:

Give the 4 array pointers to InitModbus.

Parameter:

DI_Address: DI array
DO_Address: DO array
AI_Address: AI array
AO_Address: AO array

Return:

No

MBT_SetIOSize

Prototype:

Void MBT_SetIOSize(int DI_Size, int DO_Size, int AI_Size, int AO_Size);

Description:

Set Memory size.

Parameter:

DI_Size: DI Memory size

DO_Size: DO Memory size

AI_Size: AI Memory size

AO_Size: AO Memory size

Return:

No

MBT_Set_ModbusID

Prototype:

Void MBT_Set_ModbusID(unsigned char SetID);

Description:

Set local unit identifier.

Parameter:

SetID: 0~255

MBT_ModbusTCP_Listen

Prototype:

MBT_ModbusTCP_Listen(unsigned char *RecvData, unsigned char *SendData);

Description:

Assembler Modbus TCP command and disassembler Modbus TCP data

Parameter:

RecvData: Modbus TCP master data

SendData: Modbus TCP slave data

Version Record

Version	By	Date	Description
1.00	Kane	2013/12/31	Release