UDC 协议设计

文件版本：V1.0.1

目录

1. 引言

1.1. 关于 UDC

UDC 是有人公司为自己数据传输终端产品设计的协议，其目的在于帮助用户结合我们提供的二次开发包，更容易使用我们的产品，并能很容易的进行二次开发，本文档给出 UDC 协议的详细定义。

1.2. 定义

UDC(USR Data Center Protocol)有人数据中心协议，基于 UDP 或 TCP，提供登录、心跳、数据传输、退出等机制。

2. 动态链接库(udc.dll)的使用

2.1. 动态链接库简介

动态链接库（udc.dll）提供了与有人产品通讯所需要的全部 API 函数，包括服务的启动、数据发送、数据接收、关闭服务等。

数据中心软件可以基于动态链接库（udc.dll）所提供的接口完成业务数据处理，其所需要完成的主要功能如下：

1. 调用 API 启动服务和停止服务；
2. 调用 API 接收数据和向 GM3P 发送数据，并且对数据作进一步处理；

2.2. API 接口定义

动态链接库（udc.dll）提供了下列 API 接口：

API 函数	功能	参数	返回值
SetWorkMode	设定服务中心网络事件工作模式	int nWorkMode，取值如下： 0: 阻塞模式,可应用于无窗口环境下的编程,适用于支持线程的开发环境; 1: 非阻塞模式,可应用于无窗口环境下的编程,亦可适用于不支持线程的开发环境; 2: 消息驱动模式,适用于窗口环境下的消息编程,兼容早	工作模式值:(0~2)

		期 版 本.必须在调用函数 start_gprs_server 之前调用。	
start_gprs_server	用于启动底层服务，服务启动后才能和DTU进行通讯。启动该服务后，在服务中心网络事件工作模式为 2 时,主窗口中要有响应消息的函数，以便和底层服务通讯。	1、HWND该参数用于指定接收消息的窗口,底层服务软件接收到 DTU 数据后，会向该窗口发送消息,一般指定为 DC 的主窗口,该窗口中要有响应消息的函数,所发送的消息由下一个参数指定。(当服务中心网络事件工作模式为 0 或 1 时,可以不传递真实的窗口句柄)输入参数。 2、unsigned int wMsg，有 DTU 数据到来后，会向上一个参数指定的窗口发送该消息。(当服务中心网络事件工作模式为 0 或 1 时,可以不传递真实的窗口消息)输入参数。 3、int ServerPort服务端口，指定服务的端口.输入参数。 4、char * mess，返回一些启动的信息，如果不需要，可设置为 NULL。输出参数。	0：成功； －1：失败，失败后可查看参数 mess 信息。
stop_gprs_server	停止服务	1、Char *mess，返回服务启动过程中的一些信息,如果不需要，可设置为 NULL。输出参数	0：成功； －1：失败；
do_read_proc	读数据，底层服务接收到 DTU 发送的数据后，会向启动服务函数中指定的 DC 的窗口发送消息，该窗口中的消息响应函数应立即调用读数据函数将 DTU 发送过来的数据读出。	1、data_record *recdPtr，data_record 为存放 DTU 信息和数据的结构（见后）读函数执行成功后，返回的数据存放到该参数指向的结构中。输出参数 2、char * mess，存放读数据过程中的信息，如果不需要,可设为 NULL。输出参数 3、BOOL reply,是否对收到的数据包进行应答.用户在发布程序时要把第三个参数设置为 false,调试过程中可以设置成 true。输入参数	0：成功 －1：失败
do_send_user_data	向指定 ID 的 DTU 发送数据。	1、unsigned char * userid，userid 为存放 11 位 DTU 识别码并以‘\0’为结束符的字符串（或数组）用于存放发送数据	0：成功 －1：失败

		的目的 DTU。输入参数。 2、unsigned char * data，指向要发送给 DTU 的数据。输入参数。 3、unsigned int len：发送的数据的长度。输入参数。 4、char * mess，处理过程中返回的信息,不需要可以设置为 NULL。输出参数。	
do_close_one_user	关闭 DTU 用户列表中指定 ID 的 DTU 终端,主要用来维护 DTU 用户列表。	1、unsigned char * userid，要关闭的 DTU 终端的 11 位 DTU 识别码，以‘\0’结束。输入参数 2、char * mess，处理过程中返回的信息,不需要可以设置为 NULL。输出参数	0：成功 －1：失败
do_close_one_user2	关闭 DTU 用户列表中指定 ID 的 DTU 终端,同时发送下线指令给远端 DTU。	1、unsigned char * userid，要关闭的 DTU 终端的 11 位 DTU 识别码，以‘\0’结束。输入参数 2、char * mess，处理过程中返回的信息,不需要可以设置为 NULL。输出参数	0：成功 －1：失败
do_close_all_user	关闭所有的在线 DTU 终端,一般在停止服务前执行该 API。没有发送下线指令给远端 DTU。	1、char * mess，处理过程中返回的信息，不需要可以设置为 NULL。输出参数	0：成功 －1：失败
do_close_all_user2	关闭所有的在线 DTU 终端,一般在停止服务前执行该 API。同时发送下线指令给远端 DTU。	1、char * mess，处理过程中返回的信息，不需要可以设置为 NULL。输出参数	0：成功 －1：失败
get_max_user_amount	获得当前支持的最大 DTU 数量，即 DTU 用户列表的长度。	无	最大 DTU 数量
set_max_user_amount	设置系统支持的最大 DTU 数量，即 DTU 用户列表的长度。	无	最大 DTU 数量
get_user_at	获得 DTU 用户列表中某个位置的 DTU 信息，其参数指定该 DTU 在 DTU 用户列表中的位置;该函数一般在 DC 轮询 DTU 列表时，用于查看某一 DTU 信息。	1、unsigned int pos，某一 DTU 在 DTU 用户列表中的位置。输入参数 2、user_info * infoPtr，user_info 是用于保存 DTU 用户信息的结构（见后），该指针指向的结构保存返回指定位置的 DTU 信息。输出参数	0：成功 －1：失败
get_user_info	根据 DTU 识别码查询某个 DTU 用户的信息。	1、unsigned char * userid，DTU 用户识别码,11 位，并以’\0’结束。输入参数	0：成功 －1：失败

		2、user_info * infoPtr，user_info 是用于保存 DTU 用户信息的结构（见后），该指针指向的结构保存返回指定 DTU 的信息。输出参数	
get_online_user_amount	获得当前在线 DTU 数量。	无	在线 DTU 数量
DeleteAllUser	删除 DTU 用户表中所有的 DTU 用户。	无	无
AddOneUser	向 DTU 用户列表中添加一个 DTU 用户，该操作同 DTU 登录等效；	user_info * pUserInfo，DTU 用户信息结构。输入参数	0：成功 －1：失败
SetCustomIP	指定一个 IP 给服务使用。用于解决多 IP 问题，如果只有一个 IP，则不需要使用。	Unsigned long ulIPAddr 指定使用的 IP，4 字节的网络字节序长整型。必须在调用函数 start_gprs_server 之前调用	无

2.3. 数据结构定义

上述 API 所涉及到的数据结构定义如下：

1、**DTU** 注册信息接口结构

```
typedef struct GPRS_USER_INFO
{
    char    m_userid[12];          //DTU 身份识别码
    uint32    m_sin_addr;          //DTU 进入 Internet 的代理主机 IP 地址
    uint16    m_sin_port;          //DTU 进入 Internet 的代理主机 IP 端口
    uint32    m_local_addr;        //DTU 在移动网内 IP 地址
    uint16    m_local_port;        //DTU 在移动网内 IP 端口
    char    m_logon_date[20];      //DTU 登录时间，字符串格式
    char    m_update_time[20];     //DTU 包更新时间，DC 接收到该
                                   //DTU 最近一个包的时间，使用前四字节，
                                   //time_t 类型，后 16 字节未使用 uint8 m_status;
                                   //DTU 状态, 1 在线 ，0 不在线
}user_info;
```

2、数据包接口结构

```
typedef struct GPRS_DATA_RECORD{
    char m_userid[12];                      // DTU 身份识别码
    char m_recv_date[20];                   //接收到数据包的时间
    char m_data_buf[MAX_RECEIVE_BUF];       //存储接收到的数据
```

```
    uint16    m_data_len;                    //接收到的数据包长度
    uint8     m_data_type;                   //接收到的数据包类型 1 收到心跳包,2 收到退出包,
                                             //3 收到登录包，9 收到终端发上来的数据，
}data_record;
```

其中，数据类型和宏变量定义如下：

```
typedef   unsigned char uint8;
typedef   unsigned short uint16;
typedef   unsigned int uint32;
#define MAX_RECEIVE_BUF    1024
```

2.4. API 接口的调用方法

动态库调用包括隐式调用和显式调用两种，显式调用是调用系统函数 LoadLibary 和 Freelibrary 动态导入动态库，再调用 GetProcAddress 获得函数地址，隐式调用是将动态库文件包括进项目中编译，系统启动时自动将动态库导入程序空间，如果使用多个动态库，其中某个出问题，则整个应用程序都无法启动，所以，建议采用显式调用方式。

显式调用方式时，上述 API 的调用方法如下：

✧ **Load** 动态库：

调用windows API函数LoadLibrary装载动态库，如下：

```
#define MYMESS WM_USER+0x23   //定义用户消息
char mess[512];
   HMODULE hDllMudule;   //指向动态库的句柄
   Int (* start_gprs_server)(HWND,unsigned int,int,char *);  //定义一个指向函数的地 址的指针
   hDllModule=LoadLibrary(“udc.dll”);
   If (hDllModule!=NULL) //判断调用是否成功
   {
      //从动态库中取函数地址 start_gprs_server=GetProcAddress(hDllModule,” start_gprs_server”);
      if (start!=NULL)   //判断是否取到该函数地址
          {
              //可以先设定工作模式
              //SetWorkMode(1) //0;1;2
              if ((* start_gprs_server)(this->m_hWnd,MYMESS,5002,mess)==0) MessageBox(“启动成功”);
              else
          }
   }
```

在程序开始时需要调用动态库，程序运行完毕后，要释放动态库，调用 windows API 函数 FreeLibrary可释放动态库：

```
FreeLibrary(hDllModule);            //TRUE-success  FALSE-failed
```

程序中LoadLibrary次数必须和FreeLibrary相同，每调用一次LoadLibrary，相应的应该 调用一次FreeLibrary，保

证每次调用后都会释放。

✧ 启动服务：

首先从动态库中取到该函数地址，取到地址后，就可以执行该函数，如下：

```
Int (* start_gprs_server)(HWND,unsigned int,int,char *);  //定义一个指向函数的地址的 指针
start_gprs_server= \
(int (*)(HWND,int,int,char *))GetProcAddress(hDllModule,” start_gprs_server”);
if (start_gprs_server!=NULL)
{
    (*start_gprs_server)( 窗口，消息值，服务端口，返回信息);
}
```

✧ 停止服务：

```
int (*stop_gprs_server)(char * mess);
stop_gprs_server=(int (*)(char *))GetProcAddress(hDllModule,” stop_gprs_server”);
If (stop_gprs_server!=NULL)
    (*stop_gprs_server)(mess);
```

✧ 读数据：

```
int do_read_proc(data_record *,char *);
do_read_proc= \
 (int (*)(data_record *,char *))GetProcAddress(hDllModule,” do_read_proc”);
if (do_read_proc!=Null)
 if ((*do_read_proc)(&dr,mess)==0)  //dr 为 data_record 型结构
 {
    //处理结构 dr 中的数据
 }
```

✧ 发送数据：

```
int do_send_user_data(char *,char *,int,char *);
do_send_user_data= \
(int (*)(char *,char *,int,char *))GetProcAddress(hDllModule,” do_send_user_data”);
if (do_send_user_data!=NULL)
{
 (*do_send_user_data)(userid,src,len,mess);
}
```

✧ 用户列表：

底层服务维护一张用户列表，记录当前在线用户的信息，DC 如果想知道底层用户列表，需要 调用提供的 API 函数：get_user_at、get_user_info

```
for (int I=0;I<(*get_max_user_amount)();I++)
{
      (*get_user_at)(I,&ur);
      //1、处理记录用户信息记录 ur 中的信息；

      //2、计算用户最后一次登录时间和当前时间的差值，如果大于一定的时间
      //间隔（2 分钟），可认为该用户已经下线了（没法注销数据包），关闭该用户；
      //后面会介绍处理方法；
}
```

2.5. 数据中心软件的一般框架

因为 windows 是基于消息驱动的，底层服务接收到远端 DTU 的数据后，会向启动服务时作为参数 传递给启动函数的窗口发送一个消息，因此，数据中心程序的框架中应该有一个响应消息的函数；

- ✧ VC中定义消息响应函数：
  - a) 在窗口定义类中定义消息响应函数：afx_msg void ResponseMsgRoutine(void);
  - b) 在窗口实现CPP文档中,在消息声明部分添加:ON_MESSAGE(MYMESS,ResponseRoutine)
  - c) 在窗口实现CPP文档中，实现函数ResponseRoutine函数；
- ✧ Delphi中定义消息响应函数：
  - a) 在窗口类定义中，添加procedure ResponseRoutine;message MYMESS;
  - b) 实现过程ReponseRoutine；
- ✧ VB中定义消息响应函数：
  - a) 定义函数

```
Public Function NewWindowProc(ByVal hw As Long, ByVal uMsg As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long
    Dim lpPrevWndProc As Long

    If uMsg = (MYMESS) Then

        '调用do_read_proc函数处理结果
        result1 = do_read_proc(rvdata, mess1(1), False) If result1 = 0 Then
       If result1 = 0 Then
                    '处理结果
        End If
    Else
```

```
        '将消息传递给原来的处理函数，这一行代码是必须的，否则其它消息无法处理
        NewWindowProc = CallWindowProc(oldwindow, hw, uMsg, wParam, lParam)
        End If
    End Function
```

b) 系统启动服务时，调用windows API函数SetWindowLong：

```
SetWindowLong(hwnd, GWL_WNDPROC, AddressOf NewWindowProc)
```

VB中处理起来比较麻烦，可参考demo。

数据中心程序中除了上面所述的接收例程外，还要有维护用户列表的例程，以判断某个 DTU 是否超时，该例程可采用定时器，每隔一定时间轮询一次用户列表，并计算每个用户最后一次登录时间和当前时间的差值以判断是否超时，刷新用户列表。

用户可根据需要在数据中心程序中添加进一步处理数据的功能函数，包括写数据库、数据分析等。

✧ 如何处理 DTU 超时

GPRS_USER_INFO 结构中成员 m_update_time 表示该 DTU 用户最后一次与 DC 通讯的时间（如果 DTU 在线，会每隔一定时间发送一个注册数据包，一般为 40s），轮询时，判断最后一次通讯时间和当前时间的差值，如果超过指定时间（一般设为比 DTU 在线报告时间间隔略大一些），则认为该 DTU 已经不在线，关闭该 DTU。

3. 联系方式

公 司：济南有人物联网技术有限公司

地 址：济南市历下区茂岭山三号路中欧校友产业大厦 12、13 层有人物联网

网 址：http://www.usr.cn

客户支持中心：http://im.usr.cn

邮 箱：sales@usr.cn

电 话：4000-255-652 或 0531-66592361

有人定位：可靠的智慧工业物联网伙伴

有人愿景：成为工业物联网领域的生态型企业

有人使命：连接价值 价值连接

价 值 观：天道酬勤 厚德载物 共同成长 积极感恩

产品理念：可靠 易用 价格合理

企业文化：联网的事情找有人

4. 免责声明

5. 更新历史

2017-03-03 版本 V1.0.0 创立
2023-06-30 版本 V1.0.1 修改公司地址信息