花生壳嵌入式程序代码段（仅作参考）

//预定义参数

```
#define COMMAND_AUTH        "auth router\r\n"
#define COMMAND_REGI        "regi a"
#define COMMAND_CNFM        "cnfm\r\n"
#define COMMAND_QUIT        "quit\r\n"
#define UDP_OPCODE_UPDATE               10
#define UDP_OPCODE_UPDATE_OK            50
#define UDP_OPCODE_UPDATE_ERROR         1000

#define UDP_OPCODE_LOGOUT               11
#define UDP_OPCODE_LOGOUT_RESPONSE      51
#define UDP_OPCODE_LOGOUT_ERROR         1001

#define KEEPALIVE_PACKET_LEN   20
```

//发送的 UDP 数据包结构

```
struct DATA_KEEPALIVE
{
        long lChatID;
        long lOpCode;
        long lID;
        long lSum;
        long lReserved;
};
```

//收到的 UDP 数据包结构

```
struct DATA_KEEPALIVE_EXT
{
	DATA_KEEPALIVE keepalive;
	long ip;
};

//开始UDP心跳包的发送，请注意szTcpConnectAddress必须要与TCP的连接地址一致
BOOL CTcpThread::BeginKeepAlive()
{
	USES_CDEBUG;
	OUTPUT_DEBUGA(_T("CTcpThread::BeginKeepAlive()\n"));
	if (!bTcpUpdateSuccessed) return FALSE;
	if (!m_udpsocket) return FALSE;
	if (!m_udpsocket->Connect(szTcpConnectAddress,serverport,&nAddressIndex)) return FALSE;

	//register for the data-waiting thread
	m_udpsocket->RegisterThreadRead(this->m_nThreadID,WM_ON_UDP_DATA);

	tmLastResponse = CTime::GetCurrentTime();
	bQuitTimer = false;
	AfxBeginThread(ThreadMyTimer,(void*)&(this->m_nThreadID));
	return TRUE;
	//return SendKeepAlive();
}
```

//发送 UDP 包函数

(opCode 在相关文档中有说明)

```
BOOL CTcpThread::SendKeepAlive(int opCode)
{
	if (!bTcpUpdateSuccessed) return FALSE;
	if (!m_udpsocket) return FALSE;

	USES_CDEBUG;
	OUTPUT_DEBUGA(_T("CTcpThread::SendKeepAlive() %d\n"),opCode);

	DATA_KEEPALIVE data;
	ZeroMemory(&data,sizeof(data));
```

```
        data.lChatID = nChatID;
        data.lID = nStartID;
        data.lOpCode = opCode;
        data.lSum = 0 - (data.lID + data.lOpCode);
        data.lReserved = 0;

        CBlowfish bf;
        bf.SetKey((BYTE*)szChallenge,nChallengeLen);
        char p1[KEEPALIVE_PACKET_LEN],p2[KEEPALIVE_PACKET_LEN];
        memcpy(p1,&data,KEEPALIVE_PACKET_LEN);
        memcpy(p2,&data,KEEPALIVE_PACKET_LEN);
        bf.EnCode(p1+4,p2+4,KEEPALIVE_PACKET_LEN-4);
        m_udpsocket->Send(p2,KEEPALIVE_PACKET_LEN,0);
        //RecvKeepaliveResponse();
        return TRUE;
}
```

//注销登陆时发送的 UDP 数据包

```
SendKeepAlive(UDP_OPCODE_LOGOUT);
```

//收到 UDP 数据的处理

```
BOOL CTcpThread::RecvKeepaliveResponse()
{
        if (!bTcpUpdateSuccessed) return FALSE;
        if (!m_udpsocket) return FALSE;

        //prevent the thread to be suspended while waiting for data
        if (m_udpsocket->DataReadable(0)<=0)
        {
                //USES_CDEBUG;
                //OUTPUT_DEBUGA(_T("CTcpThread::RecvKeepaliveResponse() NO DATA!\n"));
                return FALSE;
        }
        //DATA_KEEPALIVE data;
        //if (m_udpsocket->Receive(&data,sizeof(DATA_KEEPALIVE),0)<=0) return FALSE;
        DATA_KEEPALIVE_EXT rdata;
```

```
    if (m_udpsocket->Receive(&rdata,sizeof(DATA_KEEPALIVE_EXT),0)<=0) return FALSE;
    DATA_KEEPALIVE data = rdata.keepalive;

    CBlowfish bf;
    bf.SetKey((BYTE*)szChallenge,nChallengeLen);
    char p1[KEEPALIVE_PACKET_LEN],p2[KEEPALIVE_PACKET_LEN];
    memcpy(p1,&data,KEEPALIVE_PACKET_LEN);
    memcpy(p2,&data,KEEPALIVE_PACKET_LEN);
    bf.DeCode(p1+4,p2+4,KEEPALIVE_PACKET_LEN-4);
    memcpy(&data,p2,KEEPALIVE_PACKET_LEN);
    nStartID = data.lID + 1;

    USES_CDEBUG;
    OUTPUT_DEBUGA(_T("CTcpThread::RecvKeepaliveResponse() Data comes, OPCODE:%d\n"),data.lOpCode);
    if (data.lID - nLastResponseID > 3 && nLastResponseID != -1)
    {
        ::PostMessage(theApp.m_hWndController,WM_DOMAIN_UPDATEMSG,errorRetrying,0);
        OnRetry(0,0); //reupdate
    }
    if (data.lOpCode == 
UDP_OPCODE_UPDATE_ERROR) ::PostMessage(theApp.m_hWndController,WM_DOMAIN_UPDATEMSG,errorKeepAliv
eError,0);
    if (data.lOpCode == 
UDP_OPCODE_UPDATE_OK) ::PostMessage(theApp.m_hWndController,WM_DOMAIN_UPDATEMSG,okKeepAliveRecve
d,rdata.ip);
    ///////////////////////////////////////////////////////////////////////
    //calc time from last response
    CTimeSpan span = CTime::GetCurrentTime() - tmLastResponse;
    if (span.GetMinutes() > 
3) ::PostMessage(theApp.m_hWndController,WM_DOMAIN_UPDATEMSG,okKeepAliveRecved,0);

    nLastResponseID = data.lID;
    tmLastResponse = CTime::GetCurrentTime();
    return TRUE;
}
```

//TCP 更新过程

```
int CTcpThread::ExecuteUpdate()
{
    char buffer[128];
    char out_buffer[256];
   char username[128] = "";
    char key[128] = "";
    char out_key[256];
    char sendbuffer[256];
    char sendbuffer_pre[256];

   char domains[512][128];
   char regicommand[255];
    int i,len, totaldomains;
    long challengetime = 0;
    long clientinfo = 0x10013511;                    //厂商编号(1001)以及客户端版本信息(3511)
    long new_challengekey = 0x11111111;              //厂商认证信息
    if (!m_tcpsocket) return errorConnectFailed;

    USES_CDEBUG;
    OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Connecting %s.\n"),theApp.szHost);

    if (!m_tcpsocket->Connect(theApp.szHost,serverport,&nAddressIndex,szTcpConnectAddress))
    {
           OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate errorConnectFailed.\n"));
           nAddressIndex++;
           return errorConnectFailed;
    }
    ////////////////////////////////////////////////////////////////////////
    //Recv server hello string
    bzero(buffer, 128);
    len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
    if (len <=0 )
    {
           OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server hello string failed.\n"));
           m_tcpsocket->Close();
           nAddressIndex++;
           return errorConnectFailed;
    }

    OUTPUT_DEBUGA(_T("SEND AUTH REQUEST COMMAND..."));
    m_tcpsocket->Send(COMMAND_AUTH,sizeof(COMMAND_AUTH),0);
   OUTPUT_DEBUGA(_T("OK.\n"));

    ////////////////////////////////////////////////////////////////////////
    //Recv server key string
    bzero(buffer, 128);
    len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
    if (len <=0 )
    {
           OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server key string failed.\n"));
           m_tcpsocket->Close();
           return errorConnectFailed;
    }
   OUTPUT_DEBUGA(_T("SERVER SIDE KEY \"%s\" RECEIVED.\n"),buffer);



    //////////////////////////////////////////////////////////////////////
    //////////////////////////////////////////////////////////////////////
    //////////////////////////////////////////////////////////////////////
    //Generate encoded auth string
    strcpy(key,(char*)theApp.szUserPWD);
   len = CMailCoder::base64_decode(buffer+4, strlen(buffer)-4, out_buffer);
    nChallengeLen = len;
    //save challenge string "+4" skips "334 "
    memcpy(szChallenge,out_buffer,len);
    challengetime = *((long*)(szChallenge + 6));
    challengetime |= ~new_challengekey;
    int nMoveBits = challengetime % 30;
    long challengetime_new = (challengetime << (32 - nMoveBits)) | ((challengetime >> nMoveBits) & ~(0xffffffff << (32 - 
   nMoveBits)));
    //OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate challenge string:%s\n"),szChallenge);
    //////////////////////////////////////////////////////////////////////
    hmac_md5_1((unsigned char*)key, strlen(key), (unsigned char *)out_buffer,len,(unsigned char *)out_key);
    bzero(&sendbuffer_pre,sizeof(sendbuffer_pre));
    strcpy(sendbuffer_pre,(char*)theApp.szUserID);
   strcat(sendbuffer_pre," ");

    int j,totallen,pre_len;
    pre_len = strlen(sendbuffer_pre);
    totallen = pre_len + 16 + 4 + 4;
    memcpy(sendbuffer_pre+pre_len,&challengetime_new,4);
    memcpy(sendbuffer_pre+pre_len+4,&clientinfo,4);

    for (i=pre_len+8,j=0;i<pre_len+16+8;i++,j++)
    {
           sendbuffer_pre[i] = out_key[j];
    }

    bzero(sendbuffer,256);
    len = CMailCoder::base64_encode(sendbuffer_pre,totallen,sendbuffer);
   strcat(sendbuffer,"\r\n");
```

```
//Generate ok.
 ///////////////////////////////////////////////////////////////////////
 ///////////////////////////////////////////////////////////////////////
 ///////////////////////////////////////////////////////////////////////

 ///////////////////////////////////////////////////////////////////////
 //send auth data
 OUTPUT_DEBUGA(_T("SEND AUTH DATA..."));
 m_tcpsocket->Send(sendbuffer,strlen(sendbuffer),0);
OUTPUT_DEBUGA(_T("OK\n"));

 bzero(buffer, 128);
 len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
 buffer[3] = 0;

 if (len <=0 )
 {
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server auth response failed.\n"));
        m_tcpsocket->Close();
        //modified skyvense 2005/10/08, for server db conn lost bug
        //return errorAuthFailed;
        return errorConnectFailed;
 }
 if (strcmp(buffer,"250")!=0)
 {
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate auth failed.\n"));
        m_tcpsocket->Close();
        return errorAuthFailed;
 }

 ///////////////////////////////////////////////////////////////////////
 //list domains
 for (i=0,totaldomains=0;i<512;i++)
{
      bzero(domains[i],128);
      m_tcpsocket->ReadOneLine(domains[i],128);
      OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate domain \"%s\"\n"),domains[i]);
      totaldomains++;
      if (domains[i][0] == '.') break;
}
 if (totaldomains<=0)
 {
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Domain List Failed.\n"));
        m_tcpsocket->Close();
        return errorDomainListFailed;
 }

 ::SendMessage(theApp.m_hWndController,WM_DOMAIN_UPDATEMSG,okDomainListed,(long)domains);
 /////////////////////////////////////////////////////////////////////
 //send domain regi commands list
 for (i=0;i<512;i++)
{
      if (domains[i][0] == '.') break;
        bzero(regicommand,128);
      strcpy(regicommand, COMMAND_REGI);
      strcat(regicommand, " ");
      strcat(regicommand, domains[i]);
      strcat(regicommand, "\r\n");
      printf("%s",regicommand);
      m_tcpsocket->Send(regicommand,strlen(regicommand),0);
}

 /////////////////////////////////////////////////////////////////////
 //send confirm
 OUTPUT_DEBUGA(_T("SEND CNFM DATA..."));
m_tcpsocket->Send(COMMAND_CNFM,strlen(COMMAND_CNFM),0);
OUTPUT_DEBUGA(_T("OK\n"));

 for (i=0;i<totaldomains-1;i++)
{
        bzero(buffer, 128);
        len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
        if (len <= 0)
        {
                OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server confirm response failed.\n"));
                m_tcpsocket->Close();
                return errorDomainRegisterFailed;
        }
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate %s\n"),buffer);
}

 bzero(buffer, 128);
 len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
 if (len <= 0)
 {
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server confirmed chatID response failed.\n"));
        m_tcpsocket->Close();
        return errorDomainRegisterFailed;
```

```
}
OUTPUT_DEBUGA(_T("%s\n"),buffer);

//////////////////////////////////////////////////////////////////////////
//find chatid & startid
char *chatid = buffer + 4;
char *startid = NULL;

for (i=4;i<strlen(buffer);i++)
{
        if (buffer[i] == ' ')
        {
                buffer[i] = 0;
                startid = buffer + i + 1;
                break;
        }
}
nChatID = atoi(chatid);
if (startid) nStartID = atoi(startid);
OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate nChatID:%d, nStartID:%d\n"),nChatID,nStartID);
//////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//good bye!
OUTPUT_DEBUGA(_T("SEND QUIT COMMAND..."));
m_tcpsocket->Send(COMMAND_QUIT,sizeof(COMMAND_QUIT),0);
OUTPUT_DEBUGA(_T("OK.\n"));

bzero(buffer, 128);
len = m_tcpsocket->ReadOneLine(buffer,sizeof(buffer));
if (len <= 0)
{
        OUTPUT_DEBUGA(_T("CTcpThread::ExecuteUpdate Recv server goodbye response failed.\n"));
        m_tcpsocket->Close();
        delete m_tcpsocket;
        m_tcpsocket = NULL;
        return okDomainsRegistered;
}
OUTPUT_DEBUGA(_T("%s\n"),buffer);
if (m_tcpsocket)
{
        m_tcpsocket->Close();
        delete m_tcpsocket;
        m_tcpsocket = NULL;
        }
        return okDomainsRegistered;
}
```