1.1 Ziel des Projektes

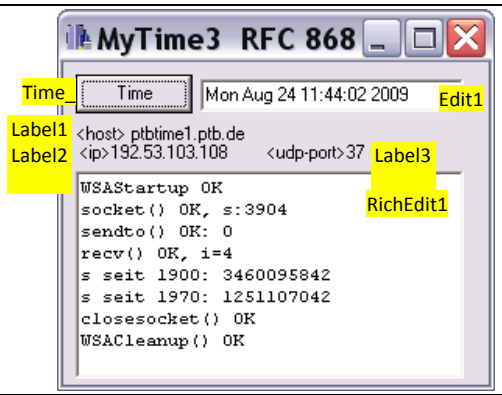
- Grundlegender Umgang mit dem Socket-API.
- Programmierung einer Client-Anwendung, die vom Zeitserver ptbtime1.ptb.de (IP: 192.53.103.108), UDP-Port 37) die Zeit abfragt und diese in lesbarer Form darstellt.
- Es soll das Transportprotokoll UDP verwendet werden. Alles andere wie MyTime1/2

1.2 Grundlagen

Auszug aus RFC 868 Time Protocol

```
When used via UDP the time service works as follows:

   S: Listen on port 37 (45 octal).

   U: Send an empty datagram to port 37.

   S: Receive the empty datagram.

   S: Send a datagram containing the time as a 32 bit binary number.

   U: Receive the time datagram.

   The server listens for a datagram on port 37.  When a datagram
   arrives, the server returns a datagram containing the 32-bit time
   value.  If the server is unable to determine the time at its site, it
   should discard the arriving datagram and make no reply.
```

1.3 Realisierung des Projektes MyTime3_RFC868_WINSOCK_DGRAM

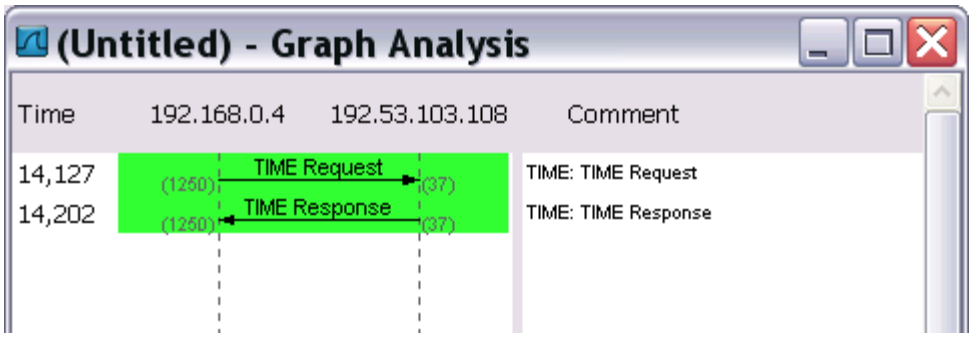


- Erzeugen Sie ein neues Projekt " **MyTime3_RFC868_WINSOCK_DGRAM**" im gleichnamigen Order.
- Editieren Sie die Oberfläche entsprechend der nebenstehenden Abbildung.
- Erzeugen Sie Schritt für Schritt den Programm-code, entsprechend dem Beispiel-Code.
- Nutzen Sie die IP-Adresse: 192.53.103.108, DGRAM-Port 37

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <winsock.h>//erforderlich
#include <time.h>   //erforderlich
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
            : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Time_Click(TObject *Sender)
{
      RichEdit1->Clear();
//===(1) Am API anmelden; nur bei WINSOCK
//Funktion: WSAStartup(); Variable: WSADATA wsaData;
      WSADATA wsaData;
      if (WSAStartup(0x101,&wsaData))
       {RichEdit1->Lines->Add("WSAStartup ERR");goto ende;}
      else
       RichEdit1->Lines->Add("WSAStartup OK");

//===(2) Socket errichten
//Funktion: socket(); Merke: s =====
      int s; //Socket-Descriptor
      s=socket(AF_INET,SOCK_DGRAM,0);
      if (s==0)
       {RichEdit1->Lines->Add("socket() ERR"); goto ende;}
      else
       RichEdit1->Lines->Add("socket() OK, s:"+IntToStr(s));

//===(3) Hostadresse (IP, Port) eintragen, wohin die Verbindung gehen soll
//Variable: SOCKADDR_IN sAddr ===
      SOCKADDR_IN sAddr;
      sAddr.sin_family=AF_INET;
      sAddr.sin_port=htons(37);
      sAddr.sin_addr.S_un.S_addr=inet_addr("192.53.103.108"); //

//===(4) Leeres Datum senden
//Funktion: sendto()  ===
       if (int slen=sendto(s,buf, strlen(buf),0,(struct sockaddr*) &sAddr,sizeof(sAddr))==0)
       {RichEdit1->Lines->Add("sendto() ERR"); goto ende;}
      else
       RichEdit1->Lines->Add("sendto() OK: "+IntToStr(slen));

//===(5) Daten aus Socket lesen
//Funktion: recv();  ===
      int i; time_t t;
    i=recv(s, (char *) &t,4,0);
      if (i!=4)
       {RichEdit1->Lines->Add("recv() ERR"); goto ende;}
      else
       {RichEdit1->Lines->Add("recv() OK, i="+IntToStr(i));
        RichEdit1->Lines->Add("s seit 1900: "+(AnsiString)ntohl(t) );
        t=ntohl(t)-2208988800;
        RichEdit1->Lines->Add("s seit 1970: "+(AnsiString)t);
        Edit1->Text=((AnsiString)asctime(localtime(&t))).SubString(0,24);
       }
//===(6) Verbindung beenden
//Funktion: closesocket();  ===
       if (closesocket(s)!=0)
        {RichEdit1->Lines->Add("closesocket() ERR"); goto ende;}
       else
        RichEdit1->Lines->Add("closesocket() OK");
//===(7) Am API abmelden; nur bei WINSOCK
//Funktion: WSACleanup();  ===
      if (WSACleanup()!=0)
        {RichEdit1->Lines->Add("WSACleanup() ERR"); goto ende;}
       else
        RichEdit1->Lines->Add("WSACleanup() OK");
ende: {}
}
//---------------------------------------------------------------------------
```

1.4 Wireshark-Trace



UDP-Time-Request: ein 4-Byte-Behälter wird gesendet	
0000 **00 1f 3f ad d7 1e 00 18 39 15 f4 fb 08 00 45 00**	**MAC**
0010 **00 20 0d 8e 00 00 80 11 44 f1 c0 a8 00 04 c0 35**	**IP**
0020 **67 6c 04 e2 00 25 00 0c 4d ba *61 62 63 64***	**UDP *Inhalt***

00 25
→37

UDP-Time-Response: 4-Byte-Time	
0000 **00 18 39 15 f4 fb 00 1f 3f ad d7 1e 08 00 45 00**	**MAC**
0010 **00 20 19 aa 40 00 34 11 44 d5 c0 35 67 6c c0 a8**	**IP**
0020 **00 04 00 25 04 e2 00 0c 67 f5 *ce 3c dc 4e***	**UDP *Timestamp***

00 25
→37