Anhang D (Programmbeispiele)

Im folgenden Abschnitt wird Schritt für Schritt erläutert, wie der EB200, aus der Sicht eines Steuerrechners , zu programmieren ist. Die Beispiele sind in der Programmiersprache C erstellt und können als Grundlage für eigene Steuerprogramme dienen. Sie basieren auf Microsoft Windows Sockets. Die Portierung auf andere Betriebssysteme sollte mit geringem Aufwand möglich sein, da ausschließlich die vom Berkley Institut definierten Socketaufrufe verwendet werden.

Diese Kapitel beschränkt sich ausschließlich auf die Programmierung des EB200. Grundlagen der Socketprogrammierung bzw. Netzwerktechnik können hier nicht erläutert werden. Hierzu existiert genügend einschlägige Literatur, wie z.B. TCP/IP Illustrated, Volume1 von W.Richard Stevens.

1. Verbindungsaufbau

Bevor der EB200 ferngesteuert werden kann, muß die PPP Verbindung, wie im Anhang A erläutert, vorhanden sein (entfällt bei Verwendung der LAN Option). Zur Herstellung einer TCP-Verbindung muß die IP Adresse des TCP Servers und dessen Servicenummer (auch Portnummer genannt) bekannt sein. Diese Informationen sind im Menü SETUP - REMOTE des EB200 einstellbar. Für die PPP Verbindung ist die Defaultadresse 192.0.0.2 mit der Portnummer 5555. Dem Hostrechner wird bei Aufbau der PPP Verbindung eine IP Adresse zugewiesen, die sich aus der EB200 IP Adresse + 1 ergibt.

Beispiel:

EB200 IP Adresse: 192.0.0.2 -> Host IP Adresse: 192.0.0.3

Diese zugewiesene IP Adresse gilt nur für den PPP Kanal. Eine eingebaute Netzwerkkarte hat eine davon unabhängige IP Adresse.

Im Fall einer LAN Verbindung zum EB200 müssen die IP Netzwerknummern (einschließlich Subnetznummern) des Hostrechners und des EB200 gleich sein. Hierzu einige Beispiele:

IP Hostrechner	Subnetmask Hostrechner	IP EB200	Subnetmask EB200	Netzklasse
89.10.6.53	255.0.0.0	89.17.11.23	255.0.0.0	Class A
89.10.6.53	255.255.0.0	89.10.11.23	255.255.0.0	Class A
89.10.6.53	255.255.255.0	89.10.6.23	255.255.255.0	Class A
132.2.3.4	255.255.0.0	132.2.20.21	255.255.0.0	Class B
132.2.3.4	255.255.255.0	132.2.3.21	255.255.255.0	Class B
192.3.4.1	255.255.255.0	192.3.4.2	255.255.255.0	Class C

Um den EB200 von außerhalb des lokalen Subnetzes zu steuern, muß dem EB200 ein entsprechendes Gateway bekannt gemacht werden. Dazu kann die entsprechende Gateway IP Adresse ebenfalls im Menü SETUP - REMOTE eingestellt werden (gilt nur für LAN Option; für PPP wird immer der PPP Partner als Gateway verwendet).

Grundsätzlich gilt: **Jede IP Adresse muß eindeutig sein!** Eine mehrfach verwendete IP Adresse führt zu unvorhersehbaren Ergebnissen und kann ein Netzwerk völlig zum Erliegen bringen. Im Normalfall werden deshalb die IP Adressen zentral vom Netzwerkadministrator vergeben. Er weiß welche IP Adressen bereits vergeben worden sind und wie die übrigen Netzwerkeinstellungen (Subnetmask und Gateway) vorzunehmen sind.

Im folgenden Programmbeispiel wird eine Socketverbindung zu einem EB200 aufgenommen, dessen IP Adresse 192.0.0.2 und dessen Portnummer 5555 beträgt.

```
struct sockaddr_in  addr;
int err;
SOCKET nSocketID;

/* create a new socket descriptor */
nSocketID = socket(AF_INET, SOCK_STREAM, 0);
if (nSocketID != -1)
{
        /* we have got a valid socket descriptor.
        now setup a connection request to EB200 */
        memset(&addr, 0, sizeof(addr));
        addr.sin_family      = AF_INET;
        /* fill out IP-Address */
        addr.sin_addr.s_addr = inet_addr("192.0.0.2");
        addr.sin_port        = htons(5555);

        /* now do the connection */
        err = connect(nSocketID, (struct sockaddr *)&addr, sizeof(addr));
        if (!err)
        {
                /* Connection has been accepted by EB200.
                Now do some initialisations */

                /* Disable nagle algorithm to get better realtime responses */
                int i=1;
                setsockopt(nSocketID, IPPROTO_TCP, TCP_NODELAY, (char*)&i, sizeof(i));
                /* Do something with EB200 */
        }
```

Der Programmteil nach dem CONNECT-Aufruf dient zur Verbesserung der Reaktionszeit. Hierzu wird der sogenannte Nagle Algorithmus abgeschaltet. Dieser sorgt normalerweise dafür, daß kleinere Datenpakete zu einem größeren Paket zusammengefaßt werden, um den Datendurchsatz zu steigern. Dieses Verhalten kann in einer Fernsteuerapplikation zu unerwünschten Wartezeiten führen, da zumeist immer ein Wechselspiel zwischen Kommandos und Abfragen existiert.

2. Initialisierung des Gerätes

Das Gerät sollte zu Beginn in einen definierten Zustand gebracht werden. Der Befehl *CLS löscht dazu das Status Reporting System, der Befehl *RST lädt alle Einstellparameter mit Defaultwerten.

3. Senden von Geräteeinstellbefehlen

Im folgenden Beispiel werden Empfangsfrequenz, Bandbreite und Demodulationsart eingestellt.

```
send(nSocketID, "FREQ 98.5E6\n", 12, 0);
send(nSocketID, "BAND 150 khz\n", 13, 0);
send(nSocketID, "DEM FM\n", 7, 0);
```

4. Auslesen von Geräteeinstellungen

Die im obigen Beispiel 3 eingestellten Parameter werden hier wieder ausgelesen. Dazu werden drei Abfragekommandos in einen SCPI-String gesendet. Anschließend wird die Antwort eingelesen und ausgedruckt.

```
char cBuffer[100];    /* Buffer for device response */
int len;
send(nSocketID, "FREQ?;:BAND?;:DEM?\n", 19, 0);
len = recv(nSocketID, cBuffer, sizeof(cBuffer)-1, 0);
cBuffer[len] = 0;
printf("frequency;bandwidth;demodulation: %s\n", cBuffer);
```

Beim Einlesen von Geräteantworten muß darauf geachtet werden, daß der RECV-Aufruf auch mit kleineren Paketen als erwartet zurückkommen kann. In diesem Fall muß ein erneuter Aufruf dieser Funktion den Rest der Daten einlesen. Als Kriterium kann hierzu das Schlußzeichen (linefeed) verwendet werden. Das obige Beispiel wird demnach folgendermaßen erweitert:

```
char cBuffer[100];    /* Buffer for device response */
int len;
int totallen = 0;
send(nSocketID, "FREQ?;:BAND?;:DEM?\n", 19, 0);
do
{
        len = recv(nSocketID, &cBuffer[totallen], sizeof(cBuffer)-1-totallen, 0);
        totallen += len;
} while (cBuffer[totallen-1] != '\n');
cBuffer[totallen] = 0;
printf("frequency;bandwidth;demodulation: %s\n", cBuffer);
```

5. Bearbeiten von SRQs

SRQs dienen zur Meldung von asynchronen Ereignissen (Fehlermeldungen, Ergebnisse etc.). Bei IEEE488 Systemen (IEC-625, IECBUS) existiert zu diesem Zweck eine Hardware-Leitung zwischen Gerät und Controller. Der EB200 simuliert dieses Verhalten, in dem er über den Socket den String &SRQ<CR><LF> schickt. Dieser String kann völlig asynchron zu einer Geräteantwort versendet werden. Der Empfänger (Hostrechner) muß also damit rechnen, daß innerhalb eines Antwortstrings diese SRQ-Meldung auftauchen kann. Zu diesem Zweck erweitern wir das obige Beispiel folgendermaßen:

```
int bSrq = 0;
char *pSRQ;
do
{
        len = recv(nSocketID, &cBuffer[totallen], sizeof(cBuffer)-1-totallen, 0);
        totallen += len;
} while (cBuffer[totallen-1] != '\n');
cBuffer[totallen] = 0;
/* Look for SRQ message in string */
do
{
        pSRQ = strstr(cBuffer, "&SRQ\r\n");
        if (pSRQ != NULL)
        {
                /* SRQ message encountered */
                bSrq = 1;
                /* delete SRQ message from received string  */
                memmove(pSRQ, pSRQ+6, strlen(pSRQ)-5);
        }
} while (pSRQ != NULL);
```

Nachdem der String bis zum Schlußzeichen eingelesen wurde, wird der String auf SRQs untersucht. Da bei der Simulation der Hardwareleitung nur der Flankenwechsel 0->1 gemeldet wird, darf keine SRQ-Meldung verloren gehen, da sonst eine SRQ-getriebene Kommunikation zum Stillstand kommt.

Das Auftreten eines SRQs wird deshalb im Flag bSrq abgelegt. Dieses Flag muß anschließend entsprechend weiterverarbeitet werden. Hierzu wird dem Gerät ein Serial Poll (&POL) gesendet. Das Gerät antwortet daraufhin mit &xyz<CR><LF>. xyz steht hierbei für den Wert des Statusbytes aus dem Status Reporting Systems. In ihm ist die Ursache des SRQs kodiert.

6. Beispielprogramm TCP/IP

Auf der mitgelieferten Diskette (Utility Disk) ist ein kleines Programm zur Steuerung des EB200 vorhanden (CExample.c). Es kann als Grundlage für eigene Programme verwendet werden. Das Beispiel ist in ANSI C geschrieben und wurde mit Visual C 5.0 getestet. Bei der Konfiguration des Projekts in Visual C muß die Library “wsock32.lib“ dazu “gelinked“ werden. Ein weiteres Beispiel ist in der Programmiersprache JAVA erstellt und ist ebenfalls als Source-Code auf der Diskette zu finden (eb200.java und eb200example.java).

Beide Programme initialisieren einen Suchlauf von 118 MHz bis 136 MHz mit 25 kHz Schrittweite. Anschließend wird der Suchlauf gestartet und alle Meßergebnisse am Bildschirm angezeigt. Um den Scanlauf beobachten zu können, wird die sich ändernde Frequenz ebenfalls angezeigt.

7. Beispielprogramm UDP

Der EB200 kann bei entsprechender Konfiguration Datagramme (UDP-Daten) versenden. Siehe hierzu Anhang F.

Auf der mitgelieferten Utilities Diskette ist ein kleines UDP-Programm (UDPExample.exe) und die zugehörigen C-Sourcen enthalten. Dieses Programm zeigt, wie der EB200 für das Senden von Datagrammen zu konfigurieren ist. Das Programm nimmt in jeder Betriebsart (CW, FSCAN, MSCAN, DSCAN, FASTLEVCW, LIST, IFPAN, AUDIO) Datagramme vom EB200 entgegen, wertet die Daten aus und zeigt laufend Statusinformationen an.

Wenn bei den Aufrufparametern des UDPExample ein Audiomode ungleich 0 (siehe hierzu auch die Tabelle zur Befehlsbeschreibung `SYSTem:AUDio:REMote:MODe`) gewählt wird, dann wird die NF über die Fernsteuerschnittstelle im gewählten Datenformat übertragen und anschließend über die Soundkarte des PC wiedergegeben. Mit der RS232-Fernsteuerschnittstelle können die NF-Daten nur bei höheren Baudraten und im Audiomode 12 oder 13 ohne Lücken übertragen werden. Mit der LAN-Fernsteuerschnittstelle können die NF-Daten in jedem Datenformat bzw. Audiomode ohne Lücken übertragen werden. Diese Applikation kann parallel zu jeder Fernsteuerapplikation gestartet werden.