Annexe D (Exemples de programmation)

Le chapitre suivant explique la façon de programmer l'EB 200 à partir d'un contrôleur. Les exemples sont donnés en langage C et peuvent servir de base pour des programmes de commande. Ils reposent sur les descripteurs de fichiers de Microsoft Windows. Il est possible de les importer facilement dans d'autres systèmes (l'Institut Berkeley a défini tous les appels des descripteurs de fichiers).

Ce chapitre ne traite que de la programmation de l'EB 200. Les bases de la programmation des descripteurs de fichiers et de mise en œuvre du réseau n'y figurent pas. Pour plus de renseignements sur ces sujets, se reporter à la documentation concernée (ex. : TCP/IP Illustrated, volume 1, de W. Richard Stevens.

1. Etablissement d'une connexion

Avant de télécommander l'EB 200, il faut que la connexion PPP soit établie (voir Annexe A), à moins que l'option LAN ne soit utilisée. Pour effectuer une connexion TCP, il faut connaître l'adresse IP du serveur TCP ainsi que son numéro (également connu comme numéro de port). Ils peuvent être configurés dans le menu SETUP – REMOTE de l'EB 200. L'adresse par défaut de la connexion PPP est 192.0.0.2 avec le numéro de port 5555. Lorsque la connexion PPP est établie, une adresse IP, composée de l'adresse IP de l'EB 200 + 1, est affectée à l'ordinateur hôte.

Exemple :

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

(adresse IP de l'EB200 : 192.0.0.2 -> adresse IP de l'ordinateur hôte : 192.0.0.3)

L'adresse IP assignée n'est valable que pour la voie PPP. L'adresse IP d'une carte de réseau intégrée est différente. Si l'EB 200 est connecté à un réseau LAN, les numéros de réseau IP (y compris les numéros de sous-réseaux) de l'ordinateur hôte et de l'EB 200 doivent être les mêmes. Voici quelques exemples :

Ordinateur hôte IP	Masque de sous-réseau de l'ordinateur hôte	IP EB200	masque de sous-réseau de l'EB200	Classe du réseau
89.10.6.53	255.0.0.0	89.17.11.23	255.0.0.0	Classe A
89.10.6.53	255.255.0.0	89.10.11.23	255.255.0.0	Classe A
89.10.6.53	255.255.255.0	89.10.6.23	255.255.255.0	Classe A
132.2.3.4	255.255.0.0	132.2.20.21	255.255.0.0	Classe B
132.2.3.4	255.255.255.0	132.2.3.21	255.255.255.0	Classe B
192.3.4.1	255.255.255.0	192.3.4.2	255.255.255.0	Classe C

Pour commander l'EB 200 en dehors du sous-réseau local, il faut lui indiquer une passerelle. L'adresse IP de la passerelle concernée peut être configurée dans le menu SETUP – REMOTE (uniquement pour l'option LAN ; pour ce qui est de PPP, le partenaire PPP sert toujours de passerelle).

Il faut toujours respecter les points suivants : chaque adresse IP doit être bien précise. Si une adresse IP sert à plusieurs composants, les conséquences sont alors imprévisibles et peuvent mettre en panne le réseau dans sa totalité. C'est pour cette raison que l'administrateur réseau attribue les adresses IP de chacun des composants. Il connaît les adresses IP qui ont déjà été affectées et la façon d'effectuer les configurations de réseau restantes (masque de sous-réseau et passerelle).

L'exemple de programmation suivant décrit la façon d'effectuer une connexion de descripteurs de fichiers vers l'EB 200 avec l'adresse IP 192.0.0.2 et le numéro de port 5555.

```
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 */
        }
```

La routine qui suit l'appel de connexion sert à améliorer le temps de réponse (en désactivant l'algorithme de Nagle). En principe, cet algorithme regroupe les petits paquets de données en paquets plus grands, ce qui conduit à un débit plus élevé de données. Cependant, dans les applications de télécommande, cela peut provoquer des retards indésirables dus à l'interaction entre les commandes et les interrogations.

2. Initialisation de l’appareil

Tout d'abord, l'appareil doit avoir un statut défini. L’instruction *CLS efface le "status reporting system", alors que l’instruction *RST charge les valeurs par défaut pour tous les paramètres de configuration.

3. Transmission des instructions de configuration de l’appareil

L'exemple suivant illustre la configuration de la fréquence de réception, de la largeur de bande et du type de démodulation.

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

4. Lecture des configurations de l’appareil

Les paramètres configurés des exemples cités dans le paragraphe 3 sont de nouveau rappelés par l'envoi de trois instructions d'interrogation dans la chaîne de caractères SCPI. La réponse est ensuite lue et imprimée.

```
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);
```

A la lecture, il se peut que les paquets de réponse aux appels reçus soient plus petits que prévu. Dans ce cas, il faut rappeler cette fonction pour lire les données restantes. Le délimiteur de fin (remplissage de la ligne) sert de critère. L'exemple ci-dessus est étendu comme suit :

```
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. Traitement des SRQs

Les SRQs servent à signaler les événements asynchrones (messages d'erreur, résultats, etc…). C'est pour cela que les systèmes IEEE488 (bus IEC-625, IEC/IEEE) sont équipés d'une ligne hardware pour connecter l'appareil avec le contrôleur. L'EB 200 simule ces activités en envoyant la chaîne de caractères &SRQ<CR><LF> via les descripteurs de fichiers. Il est possible d'envoyer cette chaîne de façon totalement asynchrone à une réponse d'instrument. Il faut donc considérer que l'ordinateur hôte peut recevoir des messages SRQ au milieu d'une chaîne de caractères de réponse. L'exemple suivant peut être étendu comme suit :

```
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);
```

Dès que la chaîne de caractères a été lue en incluant le délimiteur de fin, les SRQs sont recherchés. Si la ligne Hardware est simulée, seule la modification des fronts de 0->1 est signalée, ce qui signifie que les messages SRQ ne doivent pas être perdus, sinon toute communication SRQ est interrompue.

Dès son apparition, l'instruction SRQ est enregistrée sous l'indicateur bSrq, qui nécessite un traitement approfondi. C'est pour cela qu'un "serial poll" (&POL) est transmis à l'appareil, qui répond par &xyz<CR><LF>, représentant la valeur de l'octet d'état du "status reporting system", et qui contient la cause du SRQ sous forme codée.

6. Exemple de programme TCP/IP

Sur la disquette fournie avec l'appareil (Utility Disk), figure un petit programme de commande de l'EB200 (CExample.c). Il peut servir de base à l'utilisateur pour écrire ses propres programmes. Cet exemple est écrit en ANSI C et a été testé avec Visual C 5.0. La configuration du programme en Visual C exige d'ajouter à l'édition de liens la bibliothèque "wsock32.lib". Un autre exemple est donné en langage de programmation JAVA et se trouve également en code source sur la disquette (eb200.java und eb200example.java).

Les deux programmes initialisent un balayage en fréquence de 118 MHz à 136 MHz par pas de 25 kHz. Le balayage est ensuite lancé, avec affichage de tous les résultats à l'écran. Pour pouvoir observer le balayage, la variation de la fréquence est également affichée.

7. Exemple de programme UDP

L'EB200, configuré en conséquence, peut envoyer des paquets de données ou "datagrammes". Voir à ce sujet l'annexe F.

Sur la disquette fournie avec l'appareil (Utility Disk), figure un petit programme UDP (UDPExample.exe) et les sources en C associées. Ce programme montre comment configurer l'EB200 pour l'envoi de datagrammes. Le programme reçoit dans n'importe quel mode (CW, FSCAN, MSCAN, DSCAN, FASTLEVCW, LIST, IFPAN, AUDIO) des datagrammes de l'EB200, analyse les données et donne en permanence des informations d'état.

Lorsque dans les paramètres d'appel du programme UDPExample, on choisit un mode audio différent de 0 (voir aussi à ce sujet le tableau décrivant l'instruction `SYSTem:AUDio:REMote:MODe`), la BF est transmise au format sélectionné via l'interface de télécommande et peut être restituée par la carte son du PC. Avec l'interface de télécommande RS232, les données BF ne peuvent être transmises sans décrochages qu'à des vitesses de transmission élevées et qu'en mode audio 12 ou 13. Avec l'interface de télécommande LAN, les données BF peuvent être transmises sans décrochages dans n'importe quels format et mode audio. Cette application peut être lancée parallèlement à toute application de télécommande.