Writing a Connection Routine Basic instructions

When you execute the dspInit routine, you can specify a pointer to a routine that you provide (*user routine*). When an unsolicited connection event occurs, the **AppleTalk DSP (ADSP)** sets a flag in the connection control block (**CCB**) and calls the user routine. The user routine must clear the flag to acknowledge that it has read the flag field, and then can respond to the event in any manner you deem appropriate. The **CCB** flags are described in **Connection Control Block**. The four following types of unsolicited connection events set flags in the **CCB**:

- **ADSP** has been informed by the remote connection end that the remote connection end is about to close the connection. An appropriate response might be to store a flag indicating that the connection end is about to close. When your application regains control, it can then display a dialog box informing the user of this event and asking whether the application should attempt to reconnect later.

- **ADSP** has determined that the remote connection end is not responding and so has closed the connection. Your user routine can attempt to open a new connection immediately. Alternatively, you can store a flag indicating that the connection has closed, and when your application regains control, it can display a dialog box asking the user whether to attempt to reconnect.

- **ADSP** has received an attention message from the remote connection end. Depending on what you are using the attention-message mechanism for, you might want to read the attention code in the *attnCode* field of the **CCB** and the attention message pointed to by the *attnPtr* field of the **CCB**.

- **ADSP** has received a forward reset command from the remote client end, has discarded all **ADSP** data not yet delivered, including the data in the receive queue of the local client end, and has resynchronized the connection. Your response to this event depends on the purpose for which you are using the forward reset mechanism. You might want to resend the last data you have sent or inform the user of the event.

When **ADSP** calls your user routine, the CPU is in interrupt-processing mode and register A1 contains a pointer to the **CCB** of the connection end that generated the event. You can examine the *userFlags* field of the **CCB** to determine what event caused the interrupt, and you can examine the *state* field of the **CCB** to determine the current state of the connection.

Because the CPU is set to interrupt-processing mode, your user routine must preserve all registers other than A0, A1, D0, D1, and D2. Your routine must not make any direct or indirect calls to the Memory Manager and can't depend on handles to unlocked blocks being valid. If you want to use any of your application's global variables, you must save the contents of the A5 register before using the variables, and you must restore the A5 register before your routine terminates.

If you want to execute a routine each time an unsolicited connection event occurs but the interrupt environment is too restrictive, you can specify a NIL pointer to the user routine and periodically poll the *userFlags* field of the **CCB**.

> **Warning:** When an unsolicited connection event occurs, you must clear the bit in the *userFlags* field to 0 or the connection will hang. To ensure that you do not lose any attention messages, you must read any attention messages into an internal buffer before you clear the bit in the *userFlags* field.

The code example below is the user routine called by the listing **Using ADSP to establish and use a connection in DSPParamBlock**. When this routine is called, it first checks the **CCB** to determine the source of the interrupt and then clears the bit in the *userFlags* field of the **CCB**. If the routine has received an attention message, the user routine reads the message into an internal buffer before it clears the flag bit. The definitions of procedures PushA5, GetMyTRCCBA5, and PopA5 are shown in following listing for your convenience. In a complete application these procedures would be defined in the calling routine

Using ADSP to establish and use a connection in DSPParamBlock.

```
// Listing  An ADSP user routine
// Assuming inclusion of <MacHeaders>

#include <AppleTalk.h>
#include <ADSP.h>

typedef struct myTRCCB {
    long     myA5;
    TRCCB u;
    } myTRCCB;

pascal void myConnectionEvtUserRoutine (void);
void TellUserItsClosed (void);
void TellUserItsBroken (void);
void TellUserItsReset (void);
void CopyAttnMsg (unsigned char *attnPtr, short attnSize,
    unsigned char * myAttnDataPtr);

void PushA5 (void)    // moves current value of A5 onto stack
    = {0x2F0D};       // MOVE.L A5,-(SP)

void GetMyTRCCBA5 (void)    // retrieves A5 from the head of the TRCCB
                            // (pointed to by A1) and sticks it in A5.
    = {0x2A69};       // MOVE.L -4(A1), A5

void PopA5 (void)// restores A5 from stack
    = {0x2A5F};   // MOVE.L (SP)+, A5

myTRCCB  gDspCCB;
Boolean     gReceivedAnEvent;
unsigned short gMyAttnCode;
unsigned char  *gMyAttnDataPtr;
unsigned char  gTempFlag;
unsigned char  gTempCFlag;

pascal void myConnectionEvtUserRoutine ()
{
```

```
    // The connection received an unexpected connection event.  Find out
    //  what kind and process accordingly.

    PushA5();               // save the current A5
    GetMyTRCCBA5();             // set up A5 to point to your
                                //application's  global variables

    if ( gDspCCB.u.userFlags & eClosed )
        TellUserItsClosed();
    if ( gDspCCB.u.userFlags & eTearDown )
        TellUserItsBroken();
    if ( gDspCCB.u.userFlags & eFwdReset )
        TellUserItsReset();
    if ( gDspCCB.u.userFlags & eAttention ) {     // the event is an attention
                                    //message
        gMyAttnCode = gDspCCB.u.attnCode;   // Get the attention code.
        CopyAttnMsg(gDspCCB.u.attnPtr, gDspCCB.u.attnSize, gMyAttnDataPtr);

                                    // copy the attention
                                    //  message into our buffer
        gTempFlag = gDspCCB.u.userFlags;
        gTempCFlag = eAttention;
        gTempFlag = (gTempFlag) & (~gTempCFlag);    // clear the flag
        gDspCCB.u.userFlags = gTempFlag;

        // do something with the message

        gReceivedAnEvent = TRUE;
    }
    PopA5 (); // restore the current A5
}
```