Appendix A

Program Process.c

```
/*********************************************************************

        This application will send X, Y, Z, and W end points to the Mx4 card
        using the C/C++ DLL, MX495.DLL.  The functions mainly used are
        monitor_var, change_var, and var.

        The algorithm is as follows,

         1. Everytime Process() is called, var34 on the Mx4 card is
                checked.  If var34 = 1, then we exit the Process() procedure.
                If var34 = 0, then we continue on...

         2. At this point, var34 = 0.  Now we send the new end points for
                X, Y, Z, and W to the Mx4 card.  That is we set var22 = X end point
                var27 = Y end point, and var28 = Z end point.

         3. We set var34 = 1 to notify the DSPL that we have sent the
                new end points.

*********************************************************************/

#include <windows.h>
#include "mx4wpl.h"
#include "Process.h"

void Process(HWND hwnd)
{
        static double dX = 0 ;                  // X target position
        static double dY = 0 ;                  // Y target position
        static double dZ = 0 ;                  // Z target position
        static double dW = 0 ;                  // W target position
        static int iIndex = 0 ;                 // Index into points

        // Hard coded end points, these could come from a file instead
        static double dPts[20] = {0,1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1};


        // Set the new end points
        dX = dPts[iIndex] * 1000.0 ;
        dY = dPts[iIndex] * 1000.0 + 250.0;
        dZ = dPts[iIndex] * 1000.0 + 500.0;
        dW = dPts[iIndex] * 1000.0 + 750.0;

        // Set axis Z to 100000 to test if the cubic rate is changing
        if(iIndex == 5)
                dZ = 100000 ;

        // Set axis Z to 10000 to test if the cubic rate is changing
        if(iIndex == 15)
                dZ = 10000 ;

        // Check if Flag = 0, NOTICE: This requires that var39 is being
        // updated to VARIABLE viewing window #1
        if(var(1) == 1.0)
                return ;
```

```
        // Change the variables to the new end points
        begin_RTC();
                change_var(22, dX);
                change_var(27, dY);
                change_var(28, dZ);
                change_var(15, dW);
        end_RTC();

        // Flag the DSPL that vars have been changed
        change_var(34, 1.0);

        // Get the new index point into the endpoints table
        iIndex = (iIndex + 1) % 20;
}

/*************************************************************************/

// Header file for Processing The Handshaking of points

void Process(HWND hwnd);

/*************************************************************************/
/*********************************************************************
```

Program Target.c

```
        This application will send X, Y, Z, and W end points to the Mx4 card
        using the C/C++ DLL, MX4WPL.DLL.  The functions mainly used are
        monitor_var, change_var, and var.

        The algorithm for this program (without the window handling)
        is as follows,

         1.     Every TIMER ms (see the #define below) the procedure Process()
                is called.

        The algorithm for Process() is as follows,

         1. Everytime Process() is called, var34 on the Mx4 card is
                checked.  If var34 = 1, then we exit the Process() procedure.
                If var34 = 0, then we continue on...

         2. At this point, var34 = 0.  Now we send the new end points for
                X, Y, and Z to the Mx4 card.  That is we set var22 = X end point
                var27 = Y end point, and var28 = Z end point.

         3. We set var34 = 1 to notify the DSPL that we have sent the
                new end points.

*************************************************************************/


#include <windows.h>
#include <string.h>
#include "mx4wpl.h"
#include "Process.h"

// Global definitions
#define ID_START_BUTTON 100
#define ID_STOP_BUTTON  101
#define ID_CLOSE_BUTTON 102

// Timer in milliseconds
#define TIMER 50

// Global handles
```

```
HWND hposition;
HWND herror;
HWND hvelocity;

// Function prototypes
long FAR PASCAL TargetWndProc( HWND hwnd, UINT message,
                                                    WPARAM wparam, LPARAM lparam );


/*****************************************************************

  WinMain

        This is the main windows procedure.  Processes the message loop.

*****************************************************************/
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                                    LPSTR lpCmdLine, int nCmdShow)
{
        WNDCLASS wc;                                    // Window Class
        HWND hwnd;                                      // Handle to the main window
        MSG msg;                                        // The message
        static char buffer[20];                         // For checking the signature

        if (!hPrevInstance){
                wc.style = NULL;
       wc.lpfnWndProc = TargetWndProc;
                wc.cbClsExtra = 0;
                wc.cbWndExtra = 0;
                wc.hInstance = hInstance;
       wc.hIcon = LoadIcon( hInstance, "Target");
                wc.hCursor = LoadCursor(NULL, IDC_ARROW);
                wc.hbrBackground = (HBRUSH) (COLOR_BTNFACE+1);
                wc.lpszMenuName =  NULL;
       wc.lpszClassName = "TargetWndClass";

                // Register the class
                if (!RegisterClass(&wc))
                        return FALSE;
        }

        // Verify that the Mx4 or DM4 was found at the address in the DSPCG.INI file
        if (_fstrncmp( signature( buffer ), "MX4", 3 )!= 0){
                if (_fstrncmp( signature( buffer ), "DM4", 3 )!= 0){
                        MessageBox( NULL, "Mx4 Not Found", "", MB_OK );
                        return NULL;
                }
        }

        // Set up the position and time units for the DLL
        time_unit(1);
        position_unit(1);

        // Create the windows
        hwnd = CreateWindow("TargetWndClass","Target", WS_SYSMENU | WS_OVERLAPPED,
                   CW_USEDEFAULT, CW_USEDEFAULT, 125, 180, NULL,NULL, hInstance, NULL );

        CreateWindow( "button", "Start", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
                                  10, 10, 100, 35, hwnd, ID_START_BUTTON, hInstance, 0L );

        CreateWindow( "button", "Stop", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
                                  10, 60, 100, 35, hwnd, ID_STOP_BUTTON, hInstance, 0L );

        CreateWindow( "button", "Close", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
                                  10, 110, 100, 35, hwnd, ID_CLOSE_BUTTON, Instance, 0L );
```

```
        // Show and update the windows
        ShowWindow(hwnd, nCmdShow);
        UpdateWindow(hwnd);

        // Process the messages
        while (GetMessage(&msg,NULL,NULL,NULL)){
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        return (msg.wParam);
}

/*********************************************************************

  TargetWndProc

        Handles the messages.

*********************************************************************/
long FAR PASCAL TargetWndProc( HWND hwnd, UINT message,
                                                        WPARAM wparam, LPARAM lparam )
{
        switch( message ){

                case WM_COMMAND:

                        switch ( wparam ){

                           case ID_START_BUTTON:

                                 // Send the monitor var RTC
                                 monitor_var(1, 34);       // Flag variable

                                 // Start the timer
                                 SetTimer( hwnd, 1, TIMER, NULL );
                                 break;

                           case ID_STOP_BUTTON:

                                 // Kill the timer
                                 KillTimer( hwnd, 1 );
                                 break;

                           case ID_CLOSE_BUTTON:

                                 // Send the close message
                                 SendMessage( hwnd, WM_CLOSE, 0, 0L );
                                 break;
                        }
                        break;

                case WM_TIMER:

                        // Process the handshaking
                        Process(hwnd) ;
                        break;

                case WM_DESTROY:
                        PostQuitMessage(0);
                        break;

                default:
                        return DefWindowProc(hwnd, message, wparam, lparam);
        }
        return NULL;
}
```

This page intentionally blank.