```
/* Cytec Matrix Test Program for LAN                */
/* This program uses Microsoft's WS2_32 Library         */
/* and winsock2.h. These are available in the           */
/* Microsoft SDKs and can be downloaded from            */
/* Microsoft's Developer Network                        */
/* https://msdn.microsoft.com/en-us/default.aspx        */

#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>   /* for exit() */

int init_LAN(int sock);
int point_ops(int sock,int cmd, int mod, int rly);
int matrix_clear(int sock);
void DieWithError(char *errorMessage);

#define MAX_MOD 4
#define MAX_RLY 12

int main(int argc, char *argv[])
{
        int sock;
        char *servIP = "10.0.0.144";  /*Default IP Address*?
        struct sockaddr_in servAddr; /* IP address */
        unsigned short servPort = 8080;  /* Port */
        int mod, rly,status;

        if (argc == 3)
        {
                servIP = argv[1];
                servPort = atoi(argv[2]);
        }


        WSADATA wsaData;        /* Structure for WinSock setup communication */
        WSAStartup(0x202, &wsaData);     /* Load Winsock 2.2 DLL */

        /* Create a reliable, stream socket using TCP */
        if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))<0)
                DieWithError("socket() failed");

        /* Construct the server address structure */
        memset(&servAddr, 0, sizeof(servAddr));     /* Zero out structure */
        servAddr.sin_family = AF_INET;          /* Internet address family */
        servAddr.sin_addr.s_addr = inet_addr(servIP);   /* Server IP address */
        servAddr.sin_port  = htons(servPort); /* Server port */

        /* Establish the connection to the server */
        if (connect(sock, (struct sockaddr *) &servAddr, sizeof(servAddr))<0)
                        DieWithError("connect() failed");

        /* Initialize Device using init_LAN Function */
        init_LAN(sock);

        /* Send Clear Command to Device with matrix_clear Function*/
        if ((status = matrix_clear(sock)) != 48)
                printf("Error clearing device/n");

        /* Simple looping through switchpoints */
        for (mod=0; mod<MAX_MOD; mod++)    {
                for (rly=0; rly<MAX_RLY; rly++)  {
                        if (((status = point_ops(sock,'L',mod,rly))) !=49)
                                printf("Error point %d %d %d not closed\n",mod,rly);
```

```
                        else
                                printf("Latched point %d %d\n",mod, rly);

                        if (((status = point_ops(sock,'U',mod,rly))) !=48)
                                printf("Error point %d %d %d not open\n",mod,rly);
                        else
                                printf("Unlatched point %d %d\n",mod, rly);
                }
           }
        }
        closesocket(sock);
        WSACleanup();  /* Cleanup Winsock */
        return 0;
}
/*-------------Initialize--------------------------*/

int init_LAN(int sock)
{
        char rcvString[40];     /* Buffer for device response */
        int rcvStringLen;      /* Length of device response */
        void DieWithError(char *errorMessage);
        /* Initialize Device */
        if ((send(sock, "E0 73;V0 73;TCPANSWERBACK 1\n", 28, 0)) != 28)
        {
                DieWithError("send() failed");
        }
        Sleep(1000);      /* Wait for Response from Device */
        if ((rcvStringLen = recv(sock, rcvString, 9, 0)) < 9)
        {
                DieWithError("recv() failed or connection closed prematurely");
        }
        rcvString[rcvStringLen] = '\0';
        return 0;
}

/*-----------------Clear Matrix----------------------*/

int matrix_clear(int sock)
{

        char rcvString[8];     /* Buffer for device response */
        int rcvStringLen;      /* Length of device response */

        if ((send(sock,"C\n",2,0)) != 2)
                DieWithError("send() failed");

        Sleep(200);   /* Wait for Response

        /* Receive Response from Device */
        if ((rcvStringLen = recv(sock, rcvString, 10, 0)) <= 0)
                DieWithError("recv() failed or connection closed prematurely");
        rcvString[rcvStringLen] = '\0';
        int status = rcvString[0] & 0x3f;

        return status;
}
/*-----------Switchpoint Operation-----------------  */

int point_ops(int sock,int cmd, int mod, int rly)
{
        char cmd_str[40];          /* Formatted command string */
        char rcvString[8];     /* Buffer for device response */
        int rcvStringLen;      /* Length of device response */
```

```
        /* Format String */
        sprintf(cmd_str,"%c%d %d\n",cmd,mod,rly);

        /* Send Command to Device */
        if ((send(sock,cmd_str,strlen(cmd_str),0)) != strlen(cmd_str))
                        DieWithError("send() failed");

        Sleep(200);   /* Wait for Response

        /* Receive Response from Device */
        if ((rcvStringLen = recv(sock, rcvString, 10, 0)) <= 0)
                DieWithError("recv() failed or connection closed prematurely");
        rcvString[rcvStringLen] = '\0';
        int status = rcvString[0] & 0x3f;

        return status;
}

/*------------Error Handling Function---------------------*/

 void DieWithError(char *errorMessage)
{
        fprintf(stderr,"%s: %d\n", errorMessage, WSAGetLastError());
        getchar();
        exit(1);
}
```