Simple C++

A general code example for all Matrix Orbital Displays

Application Note

Revision 1.0

Serial Communication

Interfacing to any Matrix Orbital display begins with the correct selection of the serial port and associated attributes. In C++ a serial port object can be defined as a file using the port name, read and write designations, as well as a host of other settings.

```
port = CreateFile("COM9", GENERIC_READ | GENERIC_WRITE,
0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

Once defined, serial protocol specific attributes can be defined. Most Matrix Orbital displays will run at a default speed of 19200 baud, with 8 bits of data, no parity bit, and 1 stop bit. Again, especially with older displays, flow control should be turned off, but please consult your display manual to confirm.

```
GetCommState(port, &port_attributes);
port_attributes.BaudRate = CBR_19200;
port_attributes.ByteSize = 8;
port_attributes.Parity = NOPARITY;
port_attributes.StopBits = ONESTOPBIT;
port_attributes.fRtsControl = RTS_CONTROL_DISABLE;
SetCommState(port, &port_attributes);
```

Finally, if information is to be read from the serial port it is important to set the timeout value. This will ensure that the program will not wait infinitely for a byte to be read should an error occur.

```
timeouts.ReadIntervalTimeout = 500;
SetCommTimeouts(port, &timeouts);
```

Once a port is selected and its attributes set, text can easily be transmitted to an attached display.

Text

Once a serial port file is created and configured, text can be written to it as a series of bytes. In C++ a byte is defined as a char, adding the unsigned stipulation allows the extended table to be used.

```
unsigned char message[] = "Hello World!";
WriteFile(my_port, &message, sizeof(message), &bytes_written, NULL)
```

Once defined, an array can be written to the display by specifying the port, referencing the array, indicating the number of bytes to send, and providing a place to record the number of bytes written.

Commands

Commands are transmitted in very much the same way as text. However, as they are often unprintable characters, they are usually defined within an array of bytes as decimal or hexadecimal values.

```
unsigned char command[] = {254, 88};
WriteFile(port, &command, sizeof(command), &bytes, NULL);
```

Again, the values are written to the serial port in the same way as a series of text characters.

On Reading

The easiest way to read key presses from a Matrix Orbital display is to use polling mode. First, turn off the auto transmission of key presses so they stay in the ten key buffer rather than being transmitted immediately to the serial port.

```
unsigned char command[] = {254, 79};
WriteFile(port, &command, sizeof(command), &bytes, NULL);
```

Next, ensuring a key has been pressed, issue the command to poll the display. This will result in the key press being transmitted to the host.

```
unsigned char command[] = {254, 38};
if(WriteFile(port, &command, sizeof(command), &bytes, NULL))
```

Finally, read a single byte from the serial port file. If there are additional keys in the buffer, the most significant bit of the value will be a one, if there are no further keys in the buffer it will be zero.

```
        return(ReadFile(port, key, sizeof(key), &bytes, NULL));
else
        return(false);
```

In Closing

Once all transactions with an attached display are complete using the serial port file, it should be closed before exiting the program.

```
CloseHandle(port);
```

With this all input and output aspects of a Matrix Orbital display have been explored in the C++ programming environment. These concepts provide a scratchpad from which an epic tome of applications may be adapted for your Matrix Orbital display.

Contact

Sales
Phone: 403.229.2737
Email: sales@matrixorbital.ca

Support
Phone: 403.204.3750
Email: support@matrixorbital.ca

Online
Purchasing: www.matrixorbital.com
Support: www.matrixorbital.ca