Overview

The Ethernet ports on e100 controllers are often used for either:

- Connection of Ethernet Powerlink (as part of a system comprising a NextMove e100 and e100 drives)

OR

- Connection of a host PC (where the PC may be issuing Mint commands directly to the e100 controller via the Mint ActiveX control)

However, there is another way in which the Ethernet ports may be used to interface with third party products and that is by using these ports as a Mint ‘terminal’ channel (i.e. raw ASCII data can be received and transmitted via the Ethernet port).

Application Note AN00159 already illustrates this principle, by implementing a Modbus RTU interpreter via the Raw Ethernet terminal (using INKEY(_TERM4) to receive data bytes and PRINT #_TERM4 to transmit data). _TERM4 is the Mint pre-defined constant for the TCP/IP Raw terminal channel.

By default the Mint Raw Ethernet Terminal uses Port number 5002, but this can be changed using the Mint keyword TERMINALETHERNETPORT.

e.g. TERMINALETHERNETPORT(_TERM4) = 502

Note: Do not change the port to 5000. This is reserved for the Mint ICM Ethernet Port (the port used by Workbench and the Mint ActiveX control to send/receive Mint commands).

Possible uses for the Raw Ethernet terminal are:

- Implementation of protocol interpreters for standard Ethernet protocols (such as Modbus TCP, GE-SNPX etc..)

- Implementation of custom communication interfaces (e.g. a PLC with Ethernet communication abilities may be able to transmit/receive simple messages via Ethernet and a decoder written in Mint on the e100 controller can act on these). This can provide a very simple and cost effective means of interfacing a PLC with a Mint e100 controller.

- An input channel for data from a host program such as the HPGL interpreter (see http://www.baldormotion.com/support/SupportMe/productsupport.asp?ID=HPGL) or any other program that might typically send data via a serial buffer of some form

For this application note we will develop the idea of implementing a simple custom communication interface to illustrate the use of Raw Ethernet on e100 products. The application note assumes the use of a MicroFlex e100 or MotiFlex e100 drive. For convenience we will connect this to a PC and use Visual Basic 2010 Express and the Microsoft ‘Winsock’ control to
transmit and receive Raw Ethernet data. In practice a PC running under Windows would use the Mint ActiveX control to communicate with a Mint product but we’ll use the PC to ‘simulate’ a non-Windows third party device such as a PLC or a Linux PC for example.

The VB application uses the Mint Controller ActiveX build 5628 so please be sure to download and install Mint Machine Center version 5628 (or later) if not already installed....


**Custom Protocol Definition**

**Data Format**

For this application note we’ve invented a very simple ASCII style protocol as detailed below:

Host/Client sends:

<table>
<thead>
<tr>
<th>Start Byte (STX)</th>
<th>Function Bytes</th>
<th>Read/Write Byte</th>
<th>Data Bytes (write only)</th>
<th>End Byte (ETX)</th>
</tr>
</thead>
</table>

E100 controller replies to a valid write request with:

<table>
<thead>
<tr>
<th>Start Byte (STX)</th>
<th>Function Bytes</th>
<th>Write Byte</th>
<th>Result Byte (ACK)</th>
<th>End Byte (ETX)</th>
</tr>
</thead>
</table>

E100 controller replies to a valid read request with:

<table>
<thead>
<tr>
<th>Start Byte (STX)</th>
<th>Function Bytes</th>
<th>Read Byte</th>
<th>Result Byte (ACK)</th>
<th>Data Bytes</th>
<th>End Byte (ETX)</th>
</tr>
</thead>
</table>

E100 controller replies to an invalid read/write request with:

<table>
<thead>
<tr>
<th>Start Byte (STX)</th>
<th>Function Bytes</th>
<th>Read/Write Byte</th>
<th>Result Byte (NAK)</th>
<th>End Byte (ETX)</th>
</tr>
</thead>
</table>

To keep the example as simple as possible we haven’t included a checksum in the message frame.
Start Byte

The message frame starts with the ASCII character [STX] – 02 hex. This is true for both the host/client message and the e100 controller response.

The Mint program on the e100 controller can therefore continually read data from the Raw Ethernet terminal using \texttt{x = INKEY(_TERM4)} until it detects the [STX] character/byte.

Even though the Mint Host Comms Protocol (see Application Note AN00110) also starts with the [STX] character there is no need to disable this protocol as the controller does not intercept any characters received via the Raw Ethernet terminal to check if they are part of any native protocols.

Function Bytes

These bytes are used to tell the controller what type of function is to be performed (or to tell the host/client what function the response relates to). For this simple example we have defined just a small set of functions:

- MVR – relative move (write only)
- SPD – read/write speed
- POS – read/write position

The data is case sensitive (e.g. the host must send “MVR” to issue a relative move and not “mvr”). The function bytes apply to both the host/client message and the e100 controller response.

Read/Write Byte

The host/client sets this byte to ‘=‘ for a write or ‘?’ for a read

Data Bytes

The host/client and e100 controller sends these bytes in a standard unpacked decimal fashion. So for example, “148” or “-233.56”. If the host/client is reading data then the Data Bytes field is not included in the message frame.

End Byte

The message frame is terminated with the [ETX] character/byte (03 hex). The Mint program and host program are therefore able to sit in a loop collecting data until they detect the [ETX] character, at which point they can then process the whole message to see what action is required.

Result Byte

The e100 controller replies with [ACK] if a command from the host/client was successfully decoded and implemented or [NAK] if the command was invalid (e.g. unrecognized function type).
Example message transactions:

- `[STX]MVR=12.3[ETX]` Host tells controller to move relative 12.3 user units
- `[STX]MVR=[ACK][ETX]` Controller accepts relative move command
- `[STX]MVR=[NAK][ETX]` Reply if controller rejects move command
- `[STX]SPD?[ETX]` Host reads value of speed from controller
- `[STX]SPD?[ACK]400[ETX]` Controller returns value of 400 for current SPEED setting
- `[STX]POS=77.2[ETX]` Host tells controller to set Position of axis to 77.2
- `[STX]POS=[ACK][ETX]` Controller accepts position write

Example Mint Program

Included with this application note is a sample Mint program that interprets our custom protocol defined above.

The core of this code is simply the receipt of characters via the Raw Ethernet Terminal...

```
nByteIn = INKEY(_TERM4)
```

...and the transmission of a complete string back to the host via the Raw Ethernet Terminal...

```
Print #_TERM4 sReply,
```

The comma on the end of the Print statement suppresses the Carriage Return that would normally be included automatically on the end of transmitted characters. This program can be run on any e100 controller including MicroFlex and MotiFlex e100 drives providing a firmware version of 5623 or later is installed (i.e. Mint Lite must be supported by the drive).

The Raw Ethernet port on the e100 controller has a port number of 5002 by default. This can be changed using the TERMINALETHERNETPORT command if necessary...

```
e.g. TERMINALETHERNETPORT(_TERM4) = 1000.
```

For our example we have left it set to the default value of 5002 and we have setup the Microsoft Winsock control in our VB example to use this port number.
Example VB2010 Program

If you don’t have VB2010 Express then this can be downloaded from the Microsoft website for free (http://msdn.microsoft.com/en-us/vbasic/default.aspx). This application note includes all the source code for the example host application.

The screenshot below shows the runtime form:

Connect your PC’s Ethernet adaptor to your e100 controller (ensure the PC’s Ethernet adaptor is on the 192.168.100.x subnet).

Set the EPL node address of the drive/controller you are connecting to (in the range 1-239) via the up/down control (this should match the EPL address set on the front of the drive via the rotary hex switches) and the Ethernet Port number in the text box below (5002 is the default port number for the Ethernet Raw terminal channel on an e100 controller).

Now click on the Connect button – the status label should change to ‘Connected’ if all is correct.

If you can’t connect to the controller check your PC’s network adaptor settings via Window’s Control Panel.

Once connected Speed and Position can be set or read and move relatives can be issued using the buttons provided. For a move to be implemented the e100 drive needs to be configured such that its CONTROLREFSOURCE is set to ‘Direct’.