Overview

The MC protocol is a proprietary character based serial (RS232) protocol developed by Mitsubishi that allows a variety of automation devices such as Programmable Logic Controllers (PLCs) and Human Machine Interfaces (HMIs) to communicate with each other.

The MC protocol operates in Half duplex mode (for two-way alternate communication). Half duplex configuration allows a multi-drop topology with one master and one or more slaves.

This application note (and associated example code) details how a Mint controller can operate as a master on a MC protocol network (i.e. effectively ‘emulate’ a Mitsubishi PLC) and read/write data to a Mitsubishi PLC such as the FX or the A2SHCPU (in the case of the latter a communication module such as the A1SJ71C24-R2 is required).

A MC interpreter has been written in the Mint language. This demonstrates the power and flexibility of the Mint programming language.

The sample Mint program allows a multi-tasking Mint controller to read/write the 'D' PLC data area (integer words) using the Mitsubishi block read (WR) and block write (WW) commands. A block of contiguous data can be read from the PLC (it is recommended that a maximum of 10 PLC registers are read as a single block). The sample Mint program provides the necessary code to write to a single PLC register.

MC Protocol

Cabling

To ensure reliable communication between the Mint controller and the Mitsubishi PLC connect the interfacing RS232 cable as shown below:

<table>
<thead>
<tr>
<th>Mint Controller</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
Data Format

The Mint interpreter offers support for slave (e.g. PLC) devices with communication settings of:

- 8 data bits, 1 stop bit, No Parity

Baud rates are limited to those supported by the Mint controller (and of course must match the baud rate setting made on the slave device).

The Mint Startup block defines the controller’s serial port setting (the example program illustrates a setting of 19200 baud on terminal channel 1 – e.g. the serial port on a Nextmove ESB controller)...  

`SERIALBAUD(_TERM1) = 19200`

The Mint interpreter offers support for Type 4 MC Protocol messaging as standard (i.e. Carriage Return and Line Feed characters are appended to the messages and a checksum character is included).

The user should refer to the appropriate Mitsubishi manuals to determine how to setup the correct messaging formats/baud rates etc... for the particular PLC in use. As an example the A1S communications module uses a rotary switch to set the Message Type (set to 4 for Type 4 messaging) and a series of dip switches to setup baud rate, parity, etc. The FX range of PLC controllers have an internal control register which must be manipulated at bit level to setup the appropriate communication format.

Debugging

The variable bDebug has been defined to allow some simple debugging. If this is set to _TRUE, various messages will be displayed via terminal channel 2 (e.g. the USB connection on a NextMove ESB) during the execution of the interpreter. For Mint controllers that only support a single terminal channel (e.g. MintDrive II which only has a single serial interface) the example program should be edited to remove references to terminal channel 2 (#2).
Examples

The sample code included with this Application Note provides the core code for the MC protocol block read and block write commands. Included below are some example code snippets showing how the read and write subroutines may be called from a Mint program.

Read Example (reading a block of 5 words from the PLC)

Dim nReadBlock1(0 to 4) As Integer

    nCommsResult = doPLCRead (0,298,5,nReadBlock1)

    'If the read has been successful then copy the data to local Mint variables...
    If nCommsResult = _nErrNone Then
        nPosTarget   = nReadBlock1(0)  'D298
        nSRamp       = nReadBlock1(1)  'D299
        nSlewSpeed   = nReadBlock1(2)  'D300
        nAccel       = nReadBlock1(3)  'D301
        nDecel       = nReadBlock1(4)  'D302
    End If

Write Example (writing the value of AxisError.0 to D303 on the PLC)

    nCommsResult = doPLCWrite (0,303,Int(AxisError.0))

    'If the read has failed inform the user...
    If nCommsResult <> _nErrNone Then ?#2 "Failed to Write AxisError"