Overview

DF1 is a proprietary character based serial (RS232/RS422) protocol developed by Rockwell Automation that allows a variety of automation devices such as Programmable Logic Controllers (PLCs) and Human Machine Interfaces (HMIs) to communicate with each other.

DF1 is able to operate in two modes;
- Full duplex (for two-way simultaneous communication)
- Half duplex (for two-way alternate communication)

Half duplex configuration is a multi-drop protocol for one master (the PLC in this case) and one or more slaves (Mint controllers). DF1 allows up to 254 slave devices to be connected on a single network.

This application note (and associated example code) details how a Mint controller can 'emulate' an Allen Bradley PLC and communicate via DF1 in half duplex mode.

A DF1 interpreter has been written in the Mint language. This not only demonstrates the power and flexibility of the Mint programming language, but also allows DF1 functions to be tailored to your application. The Mint-based interpreter offers support for half duplex operation with the Mint controller operating as a slave device.

The sample Mint program allows a DF1 master (e.g. SLC500 series PLC) to read/write the Mint Comms array (with the Comms array either appearing to the PLC as an SLC500 'N7' integer data area or a 'F8' floating point data area).

The sample code should allow connection of the supported Mint controllers to Allen Bradley SLC, MicroLogix and ControlLogix processors (connection to a SLC 5/03 PLC is illustrated as an example).

Important Note:

The DF1 protocol is such that the ASCII characters [DLE], [EOT] and [BEL] (Hex code 10, 04 and 07 respectively) are likely to be transmitted by the DF1 master. These same characters are used by Baldor communication protocols (BBP, HCP and HCP2) as a 'start of message' indicator. It is therefore necessary to 'disable' all Baldor protocol operation on the Mint controller's serial port. The example Mint program includes the following code in the Startup block to achieve this:

```
COMMSMODE = _cmHCP_DISABLE + _cmBBP_DISABLE
```

Disabling BBP will prevent communication with the controller from Workbench via the serial port. Workbench can still communicate with controllers provided with USB. To re-enable Workbench communications via the serial port when using MintDrive™ after disabling BBP either:
• Activate the ‘Prevent Auto Run’ dip switch settings (either at power up or whilst the program is running) – i.e. Switches 5 and 9 ON and switch 8 OFF

• Activate the ‘Set Factory Defaults’ dip switch settings at power up – i.e. Switches 1 to 4 ON and switch 8 OFF

DF1 Half Duplex

Cabling

For the purposes of this application note a NextMoveESB Mint controller was connected to the serial port (Channel 0) of an Allen Bradley SLC 5/03 PLC via RS232. The pinout for this lead was as detailed below:

<table>
<thead>
<tr>
<th>Mint Controller</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Data Format

The Mint interpreter offers support for DF1 master 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 DF1 master 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)...

```plaintext
SERIALBAUD(_TERM1) = 19200
```

Using Rockwell’s RSLogix 500 PLC programming software our SLC 5/03 PLC’s serial port was configured (via the ‘Channel Configuration’ dialogue) as shown below:
Note that the interpreter is coded to utilize a Block Check Character (BCC) checksum (as opposed to Cyclic Redundancy Check - CRC). Also ensure that the Mint controller’s node address does not conflict with the setting for the PLC (our PLC was setup as node 1 and our Mint controller, a NextMove ESB, was setup as either node 2 or node 16, depending on the test at the time).

**NOTE:** For operation with MintDrive™ it is necessary to set both the ACK Timeout and Reply Msg. Timeout values to 50 (i.e. 1000ms).

**Message Framing**

Half duplex DF1 is character oriented and uses combinations of the ASCII control characters listed below to apply specific meanings to messages:

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Hex Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SOH]</td>
<td>0x01</td>
</tr>
<tr>
<td>[STX]</td>
<td>0x02</td>
</tr>
<tr>
<td>[ETX]</td>
<td>0x03</td>
</tr>
<tr>
<td>[EOT]</td>
<td>0x04</td>
</tr>
<tr>
<td>[ENQ]</td>
<td>0x05</td>
</tr>
<tr>
<td>[ACK]</td>
<td>0x06</td>
</tr>
<tr>
<td>[DLE]</td>
<td>0x10</td>
</tr>
<tr>
<td>[NAK]</td>
<td>0x15</td>
</tr>
</tbody>
</table>
The combinations of these characters are termed ‘symbols’ and are defined as follows (Note: BCC indicates the calculated Block Check Character):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DLE][SOH]</td>
<td>Symbol indicating start of master message</td>
</tr>
<tr>
<td>[DLE][STX]</td>
<td>Symbol separating multi-drop header from the data</td>
</tr>
<tr>
<td>[DLE][ETX]</td>
<td>Sender symbol that terminates a message</td>
</tr>
<tr>
<td>[DLE][ACK]</td>
<td>Response symbol to indicate a message has been received successfully</td>
</tr>
<tr>
<td>[DLE][ENQ]</td>
<td>Symbol indicating poll request by master</td>
</tr>
<tr>
<td>[DLE][EOT]</td>
<td>Response symbol issued by slave in response to a poll when there is no message to send as a reply</td>
</tr>
</tbody>
</table>

A typical set of DF1 message frames to execute a specific command (e.g. Read a PLC data area) are shown below (where STN=node address of slave and xxxx represents the data specific to a particular command, known as the Application Layer message packet – this is detailed later).

```
Master                                                Slave
Command  [DLE][SOH] STN [DLE][STX]xxxx[DLE][ETX] BCC  
          ↓                                               
          [DLE][ACK]                                          Acknowledge
Poll     [DLE][ENQ] STN BCC                                 
          ↓                                               
          [DLE][STX]xxxx[DLE][ETX] BCC                       Poll Response
Acknowledge                                                                                           
          ↓                                               
          [DLE][ACK]                                          
```

Note: If the STN (Node address) of the slave (Mint controller) is 16 (10 hex or [DLE] as an ASCII character) then the message must be ‘DLE stuffed’ (i.e. [DLE] is transmitted as [DLE][DLE]).

**Application Layer Message Packet**

DF1 devices send and receive application layer messages using the following message packet format:

```
Command | DST | SRC | CMD | STS | TNS | Command specific data packet |
---------|-----|-----|-----|-----|-----|----------------------------|
          |     |     |     |     |     | FNC | ADDR | SIZE | DATA |
Reply (read) | SRC | DST | CMD | STS | TNS | Command specific data packet |
Reply (write) | SRC | SRC | CMD | STS | TNS |
```

Bytes are shown from left to right in the order they are transmitted across the network.

The table below details the legend for the message packets shown above:
Field | Contents
---|---
DST | Destination node for the message
SRC | Source node of the message
CMD | Command code
STS | Status code
TNS | Transaction number (2 bytes)
FNC | Function code
ADDR | Address relating to function
SIZE | Number of bytes to be transferred
DATA | Data values being transferred. The number of bytes being dependant on the command or function being executed

**Destination Node (DST)**
This byte (character) defines which node should process the received message. e.g. the character 2 would represent node 50 (decimal).

**Source Node (SRC)**
This byte (character) defines which node has sent the message. e.g. the character 4 would represent node 52 (decimal).

**Command and Function Codes (CMD and FNC)**
These bytes work together to define the activity that is to be performed by the command message at the destination node.

The contents of the Command byte are defined as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Command or Reply</td>
<td>0</td>
<td>0</td>
<td>Command code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the PLC sends a command to the Mint controller bit 6 of this byte is set to 0. When the Mint controller replies to a command (i.e. sends the poll response) bit 6 is set to 1.

The Mint DF1 interpreter only supports receipt of Command Code 0xF hex (15 decimal or [SI] as a character). When sending a poll response the Mint interpreter will send 0x4F hex in this byte (79 decimal or O as a character).

As already outlined, in addition to the Command byte the master device will also place a character into the Function byte to indicate which specific function the slave should execute. The Mint DF1 interpreter supports only two of the many available DF1 function codes:

- 0xA2 hex – Protected typed logical read with three address fields
- 0xAA hex – Protected typed logical write with three address fields
Support for these functions allows the Mint controller to appear to be a SLC 500 series PLC thereby allowing a SLC/MicroLogix/ControlLogix DF1 master to read or write data from/to the Mint controller. The three address fields (bytes) relating to this function are detailed in a later section of this Application Note.

**Status Code (STS)**

This byte is always set to 0 in this implementation of DF1.

**Transaction Number (TNS)**

The two TNS (transaction) bytes contain a unique 16-bit transaction identifier. This number is generated by the DF1 master device (PLC in this case) by maintaining a 16-bit counter. The counter is incremented each time the master creates a new message, and the counter values is stored in the two TNS bytes of the new message. The low byte (least significant bits) of the TNS value is transmitted before the high byte (most significant bits).

The DF1 slave device (Mint controller in this case) copies the TNS field of the command message into the TNS field of the corresponding reply (poll response) message.

The Mint interpreter checks that the transaction number has changed from the previously processed message (i.e. duplicate messages are ignored).

**Address (ADDR)**

As the interpreter supports function codes 0xA2 and 0xAA it expects to receive three address fields (bytes or characters).

These three fields are:

- File Number (i.e. 0x07 hex to represent N7, 0x08 to represent F8)
- File Type (i.e. 0x89 hex to indicate integer, 0x8A to indicate float)
- Element Number (e.g. 0x1A hex to represent word/comms location 26)

The three address fields (bytes) define the Data table number the command is addressing (e.g. 0x7 hex for an N7 integer area), the data type (i.e. integer - 0x89 hex; float - 0x8A hex) and the sub element number within the data table (e.g. 0x1C hex for element 28 decimal).

**Size**

If reading or writing Integer data (range +/- 32767) the size field is always set to 2 times the number of data elements.

If reading or writing Floating point data (range +/-3.402824 x 10^{38} to +/-1.1754944 x 10^{-38}) the size field is always set to four times the number of data elements.

**NOTE:** The PLC imposes a restriction on the maximum number of data elements transferred in a single DF1 message;

- Integer – 103
- Float – 51
Data

Data is encoded in one of two forms (depending on whether the message is reading/writing integer or floating point data).

Integer data is represented via two bytes/characters, the least significant byte being transmitted first. Negative numbers are represented in a ‘two’s complement’ style.

Floating point data is represented in IEEE binary floating point (32 bit) format and hence requires four bytes/characters – this doesn’t match Mint’s internal representation of floating point numbers so the interpreter includes functions to translate between each format.

In either case (integer or float) the DF1 protocol dictates that a [DLE] character must be represented by [DLE][DLE] (i.e. data must be DLE stuffed in the same way as the Station field).

BCC Checksum

The Block Check Character (BCC) for a command message is calculated by adding together (using modulo 256 arithmetic) the STN byte and all of the bytes in the application layer message packet. This produces a result in the range 0 – 0xFF. Next the two’s complement of this result is produced (i.e. the result is subtracted from 0xFF and 1 is added to this result. Again modulo 256 is used to produce the final checksum in the range 0 – 0xFF).

The BCC for a poll request message is calculated by taking the two’s complement of the STN character.

The BCC for a poll response message is calculated by adding together (using modulo 256 arithmetic) all of the bytes in the application layer message packet. This produces a result in the range 0 – 0xFF. Next the two’s complement of this result is produced.

Note: In all cases the BCC algorithm will treat the character sequence [DLE][DLE] as a single [DLE] character (i.e. it ignores the stuffed DLE).

Further Information

For more detailed information about the DF1 protocol examine Allen Bradley publication 1770-6.5.16 (DF1 Protocol and Command Set Reference Manual).

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.

Controller Specifics

The sample Mint code that accompanies this Application Note is specifically targeted at a NextMove ESB controller. Please refer to comments within the code for minor keyword differences when using a NextMove e100 controller (e.g. MISCERRORDISABLE may not be supported depending upon firmware revision used) or MintDrive™.
If using NextMove e100 then firmware revision 5518 onwards is a pre-requisite, for all other NextMove controllers supporting USB firmware revision 5350 onwards is a pre-requisite.

If using a MintDrive™ then firmware revision 5351 onwards is a pre-requisite.

**Timing**

Operating at 19200 baud with only the DF1Slave task running on a Nextmove ESB, timing for the read and write functions (with the Channel Configuration described earlier) was found to be as follows:

- Read of 1 Integer / 1 Float Comms location  ~ 145 ms
- Read of 99 Integer / 51 Float Comms locations  ~ 350 ms
- Write of 1 Integer / 1 Float Comms location  ~ 145 ms
- Write of 99 Integer / 51 Float Comms locations  ~ 340 ms

Some of this time is a result of delays at the PLC end (e.g. the delay between the PLC receiving [DLE][ACK] and sending the Poll Request). The time delay at the PLC end can be adjusted if necessary by modifying the settings of the Reply Message Timeout on the RSLogix500 Channel Configuration dialogue (shown earlier).

If it is found that intermittent errors are occurring with the PLC message then it may be because the Mint controller is not responding in a timely fashion (e.g. if the Mint controller is executing a large number of tasks).

In this case either:

- the ‘ACK Timeout (x20ms)’ and ‘Reply Msg. Timeout (x20ms)’ values on the ‘Channel 0 – System’ configuration dialogue in RSLogix should be increased until reliable operation is achieved

and/or

- the TASKPRIORITY of the DF1Slave task should be increased (e.g. TaskPriority DF1Slave,50 can be added to the main parent task)

As MintDrive™ executes Mint at a slower rate than NextMove controllers the PLC ‘Ack’ and ‘Reply Msg’ Timeouts should be initially set to 50 (i.e. 1000ms).

There are a number of other options which will allow Mint to execute faster on a MintDrive™:

- Turn off analogue input processing \( \text{ADCMODE.0 = 4, ADMODE.1 = 4} \)
- Turn off auxiliary encoder processing \( \text{AUXENCODERMODE.0 = 16} \)
- Set seven segment LED to User Mode \( \text{LED = -2} \)
- Turn off Pulse and Direction input processing \( \text{PULSEDIRMODE = 20} \)
PLC Ladder Logic

In our application example we used the SLC MSG instruction to read and write data from/to our Mint controller. The illustrations below detail how each MSG instruction was configured:

**Floating Point Read**

This example reads 51 floating point values from Mint controller Node 16 starting from Comms(1) and stores them in the PLC’s F8 data area (starting at F8:1)

**Floating Point Write**

This example writes a single floating point value from PLC word F8:4 to Comms(7) on Mint controller Node 2
Integer Read

This example reads 12 integer values from Mint controller Node 16 starting from Comms(3) and stores them in the PLC’s N10 data area (starting at N10:10).

Integer Write

This example writes twenty three integer values starting from PLC word N10:20 to Comms(9) onwards on Mint controller Node 16.

NOTE: To avoid conflict with the control words used by the message instructions (N7:0 to N7:13 and N7:20 to N7:33) a N10 integer data area was created on the PLC and used for these examples.