



NextMove PCI ✓ NextMove BX ✓ MintDrive II ✓ Flex+Drive II ✓

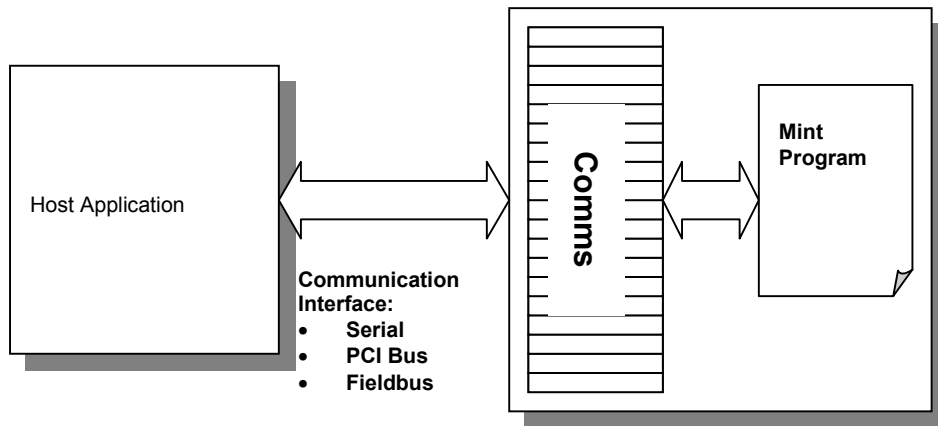
Related Applications or Terminology

- Computer to Controller communications
- Host to Controller communications
- Data transfer from host computer

Relevant Keywords
Comms() array
Event CommsX

Overview

The Mint Comms array provides an efficient and safe way to transfer data bi-directionally between an executing Mint program and a host application (PC or PLC). 99 data elements are available for use within any Mint program, called the comms array. These can be used to transfer commands or recipes for new machine configurations. With MintMT events, it is even possible for the Mint application to be interrupted when new data is posted in comms locations 1 through to 5.



Using the Comms Array, a Host Application can communication with multiple Mint devices over a fieldbus or RS485 network.

The following simple example will jog the motor at a speed defined by comms location 1 which is transmitted to the controller from the host. At the same time, the host can read the position of the motor by reading comms location 2.

```
comms (1) = 0
Loop
  JOG.0 = comms (1)
  comms (2) = POS.0
End Loop
```

The Mint Host Communications Protocol is based on ANSI x3.28 allowing is to be easily integrated into host computers or PLCs.



Use of the Commas Array

Comms looks like an array variable of dimension 99. Comms differs from the normal Mint array variables in that there is no need to define it with a Dim statement since it is automatically defined.

Use of Comms is very simple. For instance, consider the following wire winding application, where a drum is traversed between two points defined by a host computer:

Example:

```
comms(3) = 0 'Initialize the value first
Loop
  If POS>COMMS(2) Then Follow = -COMMS(3)
  If POS<COMMS(1) Then Follow = COMMS(3)
  COMMS(4) = POS
End Loop
```

The program is a simple continuous loop. The drum will be driven back and forth between point specified by Comms(2) and Comms(1) at a speed proportional to the rotation of the drum (measured using an encoder on the drum). The following ratio is specified by the variable Comms(3). These values can be changed at any time by sending data packets from the host computer and are automatically made available to the Mint program.

Example:

```
comms(1) = 0
Loop
  command = comms(1)
  If command <> 0 Then
    IF command = 1 THEN SP.1 = comms(2):MOVEA.1 = comms(3):GO.1:PAUSE IDLE.1
    IF command = 2 THEN SP.1 = comms(2):MOVER.1 = comms(3):GO.1:PAUSE IDLE.1
    IF command = 3 THEN HM.1 = comms(4):PAUSE IDLE.1
  End If
  comms(1) = 0 : 'Send Ack back to host
End Loop
```

In this example, the host will transmit a command in *comms(1)*. The data for the move type is held in indexes 2 to 4. When the move is complete, the command is set to zero. This acts as an acknowledgement to the host.

Similarly, the host can read the position of the drum by reading the value of Comms(4), which is updated once per loop (approximately every 10mS). The communications routines ensure that valid data is available to both MINT and host computer.

Events

When a Comms location 1 through to 5 are updated by the host, this will result in a Mint event being executed. Mint code can be executed immediately when the location is updated.

```
Event Comms1
  JOG.0 = comms(1)
End Event
```

In this example, the motor will jog at a speed defined by Comms(1).



Read and Writing Data from a Host

In order to read from a location on a controller or write to a location, a datapacket is sent from the host. This is made up of ASCII characters, including control characters. In order to send a number, this is transmitted as a string, for example "1234.5". All control characters are shown in the shaded boxes.

Writing Data

The host sends the following data packet to write information to the card.

Reset:	EOT (04H)
Card ID:	ASCII character 0-9;A-F; set by axis switch on card
Card ID:	Card ID repeated
Start:	STX (02H)
Comms address digit 1:	ASCII character 0-9; location in Comms array M.S. digit
Comms address digit 2:	ASCII character 0-9; location in Comms array L.S. digit
Data field:	Up to 60 characters, decimal format, comma separated, for example: "1234,45.6,-23.4"
End:	ETX (03H)
Checksum:	1 byte, XOR of everything after (but not including) STX, up to and including ETX

Note: M.S.: most significant; L.S.: least significant

Note that the card ID and the comms address are both ASCII values. To write to comms address 12, the user would send the ASCII character 1, followed by the ASCII character 2.

The card will respond with an ACK (06H) or a NAK (015H), within 50mS of receipt of checksum. A NAK will be sent if the data field contains an invalid string, or the checksum is incorrect. The card ID is set by the 4 way address switch on the card.

The data may be comma separated where each data value will load into subsequent comms addresses. For example, if the data string "10,20,30" is sent to comms address 10, address 10 will contain 10, address 11 will contain 20 and address 12 will contain 30. Note that the total data length field must not exceed 60 characters.

Reading Data

The host sends the following data packet to read data:

Reset:	EOT (04H)
Card ID:	ASCII character 0-9;A-F; set by axis switch on card
Card ID:	Card ID repeated
Start:	STX (02H)
Comms address digit 1:	ASCII character 0-9; location in COMMS array M.S. digit
Comms address digit 2:	ASCII character 0-9; location in COMMS array L.S. digit
End:	ENQ (05H)



The card will reply with the following data packet:

Start:	STX (02H)
Comms address digit 1:	0-9; location in COMMS array M.S. digit
Comms address digit 2:	0-9; location in COMMS array L.S. digit
Data field:	Up to 60 characters, decimal format
End:	ETX (03H)
Checksum:	1 byte, XOR of everything after (but not including) STX, up to and including ETX

The reply will be sent within 50mS of receipt of the checksum. A NAK will be sent if the comms address is invalid.