Chapter 3 Getting Started

Switch off power to your PLC system.

Place the module in an I/O rack, and connect the 5 Vdc external power supply to terminals 11 (+5VDC) and 13 (gnd). Do **not install or remove the module in a powered I/O rack. The module will draw approximately 350 mA from the CPU and 500 mA from the external power supply. If the CPU cannot provide that much power, connect the power supply In the "Alternate 2' method as shown on page 14. The Alternate 2 method draws all power from the remote power supply.**

To start BASIC, connect a terminal[3] to port A. This port defaults to 9600 baud, 8 data-bits, no parity, 1 stop bit.

Warning

The serial ports on the NCMZ-1799 module are not optically isolated from each other. Confirm that the serial ports of the devices that are connected to the 1799 module are either common to each other or isolated from any common. You cannot plug serial devices into this module that have different grounds (common mode references).

The RS-485 port is optically isolated from the three RS-232 serial ports, Refer to the schematic diagrams (Appendix R) for more information.

Power up the module by switching on the PLC and the external power supply. If everything is OK, you should read

```
READY
#
```

If you don't get the READY prompt, press the RESET BUTTON on the front of the module. Verify that the bottom LED flashes on briefly. Verii that the external power supply voltage is 5 ± .1 VDC. Also try sending a few <BREAK> characters to the module (NC799COM.EXE uses the <CTRL>-C to send a <BREAK>). Verify that the correct cable is connecting between the 1799 and your computer. Refer to the Troubleshooting section, page 97.

Enter all commands and BASIC statements in UPPER CASE (CAPITAL letters).

Type **NEW** to clear the data and program memory. From this point on, you can enter the BASIC statements.

The reset-button permits restarting the module, without removing power from the CPU or module.

At start-up the status LED flashes on briefly. If for any reason the basic interpreter loses control, a watchdog error will occur after 100 msec. In this case both status LED's will flash

[3] A terminal program NC799COM.EXE is available from Westinghouse

alternately with a frequency of 5 Hz and the interpreter is stopped to prevent any undefined actions.

A watchdog error can be reset by pressing the reset button. Refer to Appendix J for more information on troubleshooting a watchdog trip.

Chapter 4 Entering a Program

The "#" prompt indicates that BASIC is at command level.

If you type LIST (followed by a carriage return), this command will be executed immediately and you will see a list of the BASIC program on your screen.

At the command level, you can delete and/or append lines.

A line number followed by carriage return deletes a line.

A line number followed by some statements adds a line to your program or replaces **a** line, if the line number already existed.

For editing in the command mode, the following rules apply:

Line numbers must be in the range 1 - 32767;
Control X deletes a line, if entered before carriage return;
Control H, backspace or DEL deletes the last character;
Control A repeats the previous input; handy for making a minor change to a newly entered line
Multiple statements in a line must be separated by a colon (:).
Spaces are usually optional

More. advanced editing can be achieved by entering the edit mode : type EDIT.
The prompt is now ".".

For a full description of the editor features, see **Appendix A** of this manual.

4.1 Code Samples

4.1.1 Sending 'Printable' ASCII Characters out Serial Port (with CR LF)

Send these characters out the B serial port >A4000${}^{C}_{R}{}^{L}_{F}$. Note ${}^{C}_{R}$ stands for CARRIAGE RETURN (hex 13), ${}^{L}_{F}$ stands for LINE FEED (hex OA).

```
100    PORT B 9600,N,8,1      :REM 9600 baud, 8 data bits, no parity and 1 stop bit

: (other code) :
       : :
500    PORT B                 :REM reselect PORT B (assume other port was previously used)
510    PRINT '>A4000          :REM send characters out the serial port
520    SUSPEND 1              :REM pause 20 milliseconds before switching to another PORT
```

A CARRIAGE RETURN and LINE FEED are automatically appended to a PRINT statement, unless you specifically inhibit this (see example 4.1.2).

4.1.2 Sending "Printable' and 'Non-Printable' ASCII Characters out Serial Port (with CR LF)

Send these characters out the serial port ${}^{S}_{X}{}^{D}_{L}$Message${}^{E}_{X}{}^{E}_{T}$. ${}^{S}_{X}$ stands for START OF TEXT (hex 02). ${}^{D}_{L}$ stands for DATA LINK ESCAPE (hex 10). ${}^{E}_{X}$ stands for END OF TEXT (hex 03). ${}^{E}_{T}$ stands for END OF TRANSMISSION (hex 04).

```
       : :
500    PORT B                 :REM reselect PORT B
510    PRINT#2;#$10;"Message";#3;#4;
520    SUSPEND 1              :REM pause 20 milliseconds before switching to another PORT
```

Notice the ending semicolon (;). This will suppress the CARRIAGE RETURN and LINE FEED characters that are normally appended to the end of a PRINT statement.

The dollar sign ($) means to interpret the following character as a hexadecimal value rather than decimal.

Notice that the individual characters are separated by semicolons (;). This character is used by BASIC to separate different portions of the PRINT statement. Refer to the PRINT description (page 59) for more information.

Notice the SUSPEND statement. The purpose of this statement is to allow the data transmitter chip (UART) to clear all of its characters before a new command is given to the chip. This statement is not necessary so long as other code is placed immediately after a PRINT statement, but before a PORT statement.

4.1.3 Immediate Mode Programming

A nice feature of BASIC is that you can execute many of the statements directly from the command line. A command line starts with the pound (#) sign.

Example

```
#PRINT 'This is a test'
```

This prints out the following message to the 'A' serial port.

```
This is a test

READY
```

Example

```
#PRINT (56+4)*3
```

Calculate and print this statement

```
180

READY
#
```

Example

```
#PORT C:GET A$,LOC:PORT A
```

Read block of data from the 'C' serial port.

Example

```
#PORT C:PRINT "Test Message":SUSPEND 2:PORT A
```

Send message out the 'C' serial port. Wait before switching back to the 'A' serial port.

Example

```
#PORT D:X=CE:PORT A:PRINT X
```

Read the communication status of PORT D, and PRINT this value out PORT A

Chapter 5 BASIC Variables

All variables (except array variables) of this BASIC interpreter have a fixed address.

On one hand, this limits the variable names that may be used. On the other hand it eliminates the need for variable-stacking and therefore increases the speed of execution.

The following variable-names may be used:

a - 26 numeric variables	A, B, C . . . Z;
b - 26 string variables	A$, B$ . . . . Z$;
c - 26 local variables	LA, LB .... LZ;
d - PC related variables	OR, IR, OG, IG, HR;
e - Special variables	TM, CE, KS, RND, LOC, TIM$, DAT$

The interpreter distinguishes between **local** and **global** variables, Because of the multitasking facilities of the interpreter, you must carefully select which variable should be used.

- Global variables are used between tasks, to share information.

- Local variables can be used in simple subroutines, that take the same BASIC code, but have to result in values that are unique to the task, in which they are called.

- String variables are always global, so if you assign a value to it, it can effect all tasks that reference that variable.

Global - The numeric variables range from -32768 to +32767.
- The variables may be subscripted (one or two dimensions).
- The maximum subscript size is 255.
- No zero or minus subscript is allowed.
- Subscripts may be an expression.

String - The string variables are 80 characters wide.
- Each character (byte) can be individually addressed, as in A$(1) and in A$(X).
- A$(0) always holds the length of the string.

Local - Local variables range from 0 to 255 and cannot be subscripted.

PC - The PC related variables OR, IR, OG and IG must be subscripted with a value or expression that ranges from 1 to 32. The reference number is the same as in your PC ladder program. Holding registers (HR) must be subscripted with a value or expression that ranges from 1 to 4096. (Current ladder programs only support 256 HR's).

All PC variables are global variables.

```
Examples :   OR(1) = IR(6)
             HR(5) = $00FF.HR(5).
```

5. BASIC VARIABLES (cont.)

Special variables.

TM is global and may be read or written to. It contains the system clock, and its value is increased by one every 20 msec.

CE is a local variable (communication error) and gives the status of the associated port, once it is referred to.
Valid error bits are :
bit $40 framing error
bit $20 parity error
bit $10 overrun error.

```
100   PORT A
110   IF CE<>0 THEN GOSUB 1000       :REM Test for PORT A comm error
120   PORT B
130   IF CE<>0 THEN GOSUB 2000       :REM Test for PORT B comm error
```

KS is a local variable (keystroke) and contains the last character, received by the associated port.

```
100   PORT A
110   X=KS        :REM Read last character rcv'd on PORT A
120   PORT B
130   Y=KS        :REM Read last character rcv'd on PORT B
```

RND is a global variable (pseudo-random).

LOC is a local variable and contains the number of characters in the receive buffer. It can be used to test if any character was received. In combination with the GET statement LOC can be used to empty each receive buffer. Precede LOC with a PORT statement in order to select the proper serial port.

```
100   PORT B
110   IF LOC > 0 THEN GOT0 1000:REM Test if PORT B has data
200   PORT C
210   IF LOC > 0 THEN GOT0 2000:REM Test if PORT C has data
```

TIM$ a global string variable that contains **the** real time in the form HH:MM:SS

DAT$ a global string variable that contains **the** date in the form MM-DD-YY. The built-in real time clock automatically handles 28, 29, 30 and 31 day months plus leap-year. Daylight Savings Time must be programmed. Refer to Appendix H.

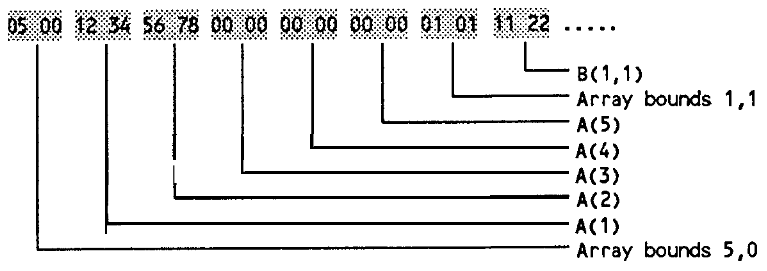
Array Variables are stored at the end of the BASIC program. A pointer to the first array variable DIMentioned can be found by locating the end of the BASIC program. The first byte of the first array is two bytes after that location.

```
50     DIM A(5),B(2,2)
60     A(1)=$1234                        :REM load a dummy value
100    POKE($7F,1)                       :REM unsigned math
110    X=256*PEEK($856)+PEEK($857)       :REM point to end of BASIC program
120    x=x+2                             :REM point to first byte of first array
130    V=256*PEEK(X)+PEEK(X+1)           :REM convert this first value to integer
140    PRHEX V                           :REM print this

#RUN                (test this program)

1234                (prints the first value of the array)

READY
#
```

What is actually stored in memory is the data in the array, plus the dimensions of the array.

```
50     DIM A(5),B(2,2)
60     A(1)=$1234
70     A(2)=$5678
80     B(1,1)=$1122
```

Find the value of X (256*PEEK($856)+PEEK($857). This points to a block of memory that looks something like this:



Chapter 6 BASIC Commands

The following is a list of all commands in BASIC. Most of the commands can be used in a statement as well, but generally this is not very useful. Detailed information about each command can be found in chapter 8.

COMMAND	RESULT
BREAK (key)	Terminates program execution and returns to the prompt mode. (This can be disabled; see Appendix I.
DWLOAD	Down-loads a program in memory to a standard PROM programmer (like Data I/O) under Motorola EXORciser format.
EDIT	Invokes the line-editor (see appendix A).
LIST	Lists program lines on the terminal (port A).
LLIST	Lists program lines on the printer (port B). (L)LIST lists the entire program; (L)LIST X lists line with number X; (L)LIST X-Y lists all lines from X to Y.
LOAD	Loads program from tape (port B).
NEW	Erases the program and data in memory.
PROGRAM	Loads a program from the optional EPROM into memory. If PROGRAM is followed by a number N, the Nth additional program in EPROM will be loaded into memory.
RUN	Starts execution of the program in memory.
SAVE	Saves a program on tape (to port B).
SIZE	Prints two decimal numbers, 1 - number of bytes, used by program: 2 - number of free bytes. Array storage is not included in these numbers, until the program has been run.
UPLOAD	Uploads the program into memory from a standard PROM programmer, with use of the Motorola EXORciser format.