numa·logic

# NCM
# Basic Manual

## Catalog No. NLAM-B201

**Westinghouse Electric Corporation**
Automation Division
Programmable Controls Department
200 Beta Drive
Pittsburgh, PA 15238

**WARNING**

**THIS EQUIPMENT HAS NOT BEEN TESTED TO SHOW COMPLIANCE WITH NEW FCC RULES (47 CFR, PART 15) DESIGNED TO LIMIT INTERFERENCE TO RADIO AND TV RECEPTION. OPERATION OF THIS EQUIPMENT IN A RESIDENTIAL AREA IS LIKELY TO CAUSE UNACCEPTABLE INTERFERENCE TO RADIO COMMUNICATION, REQUIRING THE OPERATOR TO TAKE WHATEVER STEPS ARE NECESSARY TO CORRECT THE INTERFERENCE.**

Since the equipment explained in this manual has a variety of uses, the user and those responsible for applying this equipment must satisfy themselves as to the acceptability of each application and use of the equipment. Under no circumstances will Westinghouse Electric Corporation be responsible or liable for any damage, including indirect or consequential losses resulting from the use, misuse, or application of this equipment.

The text, illustrations, charts, and examples included in this manual are intended solely to explain the use and application of the NCMZ-798 Module. Due to the many variables associated with specific uses or applications, Westinghouse Electric Corporation **cannot** assume responsibility or liability for actual use based upon the data provided in this manual.

No patent liability is assumed by Westinghouse Electric Corporation with respect to the use of circuits, information, equipment, or software described in this manual.



This manual is printed in the U.S.A. and is subject to change without notice.



The NCMZ-798 Module is designed and manufactured by:

Westinghouse Electrotechniek en
Instrumentatie B.V.
1500 EB Zaandam (The Netherlands)

# FOREWORD

Several documents have been bound together to form this manual. Each of these can be identified by its individual document number, as follows:

- NLAM-B201
  NCM Basic Manual

- SP798-11
  128 Alarmtext Monitor with 40 Character Messages

- SP798-61
  Radial Communication Network

Each of these documents has a separate table of contents, and is page-numbered separately.

Additional information on software for the NCMZ-798 Communications Module can be found in other documents. Please contact your Westinghouse representative for details.

# NCM BASIC MANUAL

TABLE OF CONTENTS

## INTRODUCTION

In the past various dedicated software packages in the NCMZ-798 module made it possible to communicate with a Westinghouse PC-700 or PC-900 at RS-232 level. Each of these programs had a fixed task, and offered no possibilities to extend or change the modules' function.

NCM-BASIC was developed to cover a wide variety of applications, using a terminal and/or a printer connected to a PC-700/900 system. Now, the user himself can define the module's function with the simple BASIC language.

The tape-loading capabilities, using the standard tape-loaders NLTL-783 or ZE-601, allow for rapidly reprogramming the module and make the module interchangeable between different control applications.

NCM-BASIC characteristics:

- Integer mathematics
- Easy access to PC holding registers -- large program storage area
- One string variable
- High speed of execution

## INSTALLATION

To make this BASIC fully operational, a small ladder program is required in the PC. Due to the use of REGISTER to TABLE and TABLE to REGISTER functions, the PC must be an advanced PC-900 or PC-700. The ladder program is given in Appendix A. It supports the exchange of data between module and PC for a 255 wide register-table via output-register 1 and input-register 1.

For hardware installation refer to the NCMZ-798 instruction leaflet, Catalog No. NCMZ-798.

Port A must be connected to a terminal. This port is factory set at 4800 BAUD, data format is 7 bits, even parity, and 1 stop bit. Port B may be connected to either a printer or a tape recorder. This port is factory set at 1200 BAUD. For I/O cabling, see the applicable section of this manual.

## COMMANDS

| | |
|---|---|
| NEW | Deletes all lines and data |
| LIST | Lists the program as follows:<br>LIST Lists the entire program<br>LIST X Lists line with line number X<br>LIST X-Y Lists all lines between X and Y |
| SIZE | Prints two decimal numbers<br>1 - number of bytes, used by program<br>2 - number of free bytes<br>Array storage is not included until the program has been run |
| RUN | Executes the program |
| BREAK (key) | Terminates the program execution and returns to the prompt mode |
| SAVE | Saves the program on tape. A STR-LINK II or SANYO recorder must be connected to the module port B |
| LOAD | Loads the program from tape |
| PORTA | Redirects the output to port A |
| PORTB | Redirects the output to port B |
| OPEN | Redirects the input from port B; no echoing (used for data load) |
| CLOSE | Redirects the input from port A; echo set on again |
| LEDON | Turns on the module status LED |
| LEDOFF | Turns off the module status LED |

All commands above may be used within a BASIC statement. We strongly discourage the use of statements such as: 10 IF A=3 THEN NEW

## EDITING and LINE-FORMATS

1. Line numbers must be in range 1 to 32767
2. Lines are appended and/or inserted
3. A line number followed immediately by C/R deletes the line
4. "Control-X" deletes the line if entered before C/R
5. "Control-H" or Backspace deletes last character
6. Blanks are immaterial
7. The system prompts with a #
8. Multiple statement lines are not permitted (as in A=3:B=4)

OUTPUT STATEMENT

| | |
|---|---|
| PRINT | Prints a return and linefeed |
| PRINT A | Prints the value of variable A |
| PRINT #A | Prints the low order byte of variable A |
| PRINT "STRING" | Prints the string between quotes |
| PRINT A,B | Prints two values with zone spacing (the comma is used for zone spacing) |
| PRINT A;B | Prints two values without zone spacing (the semicolon suppresses zone spacing) |

A semicolon at the end of a print line suppresses return and linefeed.

## INPUT STATEMENT

1. On execution of an input statement, the system prompts with a question mark: ?

2. The input list may define either one or more variables, or the string variable ($)

3. If the input list defines more input than is entered, an additional ? is prompted

4. Numbers inputted must be separated by a comma

5. The string variable $ can be up to 19 characters long

6. Entry of numbers out of the range +/- 32767 causes an error

7. If the first character of an inputted number is alphabetic (by mistake) the system prompts with RE-ENTER

8. The input statement cannot be used in the # (immediate execution) mode

9. Examples

```
INPUT X
INPUT X,Y,Z     (enter 3 numbers)
INPUT $         (enter a string)
```

## VARIABLES

1. 26 variable names A,B,C.....Z may be used

2. All variables are integer, and range from -32767 to +32767

3. Variables may be subscripted (one or two dimensions)
   The maximum subscript size is 255; no minus or zero subscript allowed; subscripts can be an expression: LET X(B,C)=3
   Example: DIM X(5,10),Y(C+30) defines two arrays

4. PC holding registers are referred to as R(expr). The variable R cannot and may not be defined in a DIM statement. The result of (expr) must range from 1 to 255
   Examples A=R(8) "moves" HR8 to A
   R(2+3)=B "moves" B to HR5

5. A special variable KS (keystroke) contains the value of the last key that was pressed on the terminal's keyboard. All relationship tests and arithmetic operations are permitted on KS. The variable must be set to zero before testing it.

```
Example   5  KS=0
          10  IF  KS=0  GOTO  10   (wait  for  a  key)
          15 GOSUB  3000
          20 .....
          3000 REM SERVE OPERATORS REQUEST
          3005 .....
          3010 RETURN
```

6. A special variable named $ (string) may be used to enter a string. The only arithmetic available for string is:

```
IF $="STRING" THEN ...
```

   The string variable (called "$") can be printed by the PRINT statement,as in:

```
 5 INPUT $
10 PRINT A,B,$
```

   The individual characters of the string may be considered as an integer variable with a range from 0 to 255. The syntax is $(expr). The string may be up to 30 characters long. The end of a string is marked by the value 30. No range-check is performed on the value of $(expr), so be carefull that $(expr) will not exceed the range in FOR NEXT loops. If you attempt to write a value over 255 to $(expr), the value will be truncated to 8 bits.

Examples:

```
5 DIM B(30)
10 INPUT $
12 REM COPY STRING INTO ARRAY B
15 A=1
19 REM value 30 marks end of string
20 IF $(A)=30 GOTO 100
25 B(A)=$(A)
30 A=A+1
35 GOTO 20
100 REM now REVERSE the string
110 FOR C=1 TO A-1
120 $(C)=B(A-C)
130 NEXT C
140 PRINT $
145 REM a difficult way to print a string
150 A=1
160 IF $(A)=30 GOTO 200
170 PRINT#$(A);
180 A=A+1
190 GOTO 160
200 REM
```

## CONTROL STATEMENTS

1. Computed GOTO — GOTO (expr)
2. Computed GOSUB — GOSUB (expr)
3. RETURN — Must be preceeded by GOSUB
4. FOR (var) = (expr) TO (expr)
5. NEXT (var) — Must be preceeded by FOR, adds 1 to the variable
6. FOR/NEXT loops and GOSUB's can be nested up to a depth of 8
7. END — Stops program execution
8. PAUSE — Holds program execution until a return is entered
9. EXAMPLES

```
5 GOTO 10
10 GOTO A*B
100 GOSUB 4
110 FOR A= 1 TO 3
120 PRINT A
130 NEXT A
135 PAUSE
140 END
```

## ASSIGNMENT STATEMENT and ARITHMETIC OPERATORS

1. The general format of an assignment statement is:

   ```
   LET var = expr
   ```

2. LET may be omitted

3. The arithmetic operators are:

   ```
   *   multiply
   /   divide
   .   logical and
   ^   logical or  (^= $5E)
   +   add
   -   subtract
   ```

4. The logical operators have the same priority as * and /

5. Examples:

   ```
   LET X = Y+B*(C-5)
   X=X.32767
   R(5)=R(5)^R(6)
   ```

6. Error detection on divide by zero

   Over/underflow at multiply and divide

## RELATIONSHIP TESTS

1. General format is:

   IF expr relationship expr statement

2. Valid relationship characters:

   | | |
   |---|---|
   | > | greater than |
   | < | less than |
   | = | equal |
   | >= | greater than or equal |
   | <= | less than or equal |
   | <> | not equal |
   | >< | not equal |

3. Examples

```
            IF X=Y GOTO 30
            IF X>Y IF A>B LET C=0
            IF R(A)>R(B) GOTO R(R(D))
Specials:   IF KS <> 0 THEN RETURN
            IF $="ABCD" PRINT "FIRST 4 OF ALPHABET"
```

## FUNCTIONS

1. TAB(expr) Used in a print statement, positions cursor at location specified by (expr)
   If the cursor is beyond specified location, printing starts at current location
   Example: PRINT TAB(15);X,Y;TAB(49);"REMARK"

2. RND Generates a random number
   Examples: X=RND (between +/- 32767)
   Y=RND.32767 (positive numbers only)

## SHORT FORM BASIC VOCABULARY

### Normal commands

| | |
|---|---|
| RUN | Runs program |
| LIST | Lists program |
| NEW | Clears memory |
| SIZE | Shows available memory |
| END | Stops execution |
| PAUSE | Holds execution until return entered |
| REM | Remark |
| DIM | Defines array |

### Special commands

| | |
|---|---|
| PORTA | Output to Port A |
| PORTB | Output to Port B |
| SAVE | Saves program on tape |
| LOAD | Read program from tape |
| LEDON | Turns LED on |
| LEDOFF | Turns LED off |
| OPEN | Opens Port B for data input from tape |
| CLOSC | closes Port B |

### Control statements

| | |
|---|---|
| FOR .. TO .. | Program loop between FOR and NEXT |
| NEXT | |
| GOTO | Program jump to line number |
| GOSUB | Subroutine jump |
| RETURN | Return from subroutine |
| IF | Condition test |
| THEN | Leadin for statement if condition was true |
| PRINT | Output statement |
| INPUT | Input statement |
| LET | Assignment statement (may be omitted) |

### Functions

| | |
|---|---|
| TAB(10) | Used in print statement to position cursor |
| RND | Random generator |

### Arithmetic operators

| | |
|---|---|
| + | Add |
| - | Subtract |
| * | Multiply |
| / | Divide |
| . | Logical and |
| ^ | Logical or |

### Variables

| | |
|---|---|
| A...Z | Integer variables |
| $ | String variable |
| KS | Keystroke |
| RND | Random variable |
| R(expr) | PC variable (holding register) |

## ERROR MESSAGES

1. General format is:

   ERROR # number IN LINE linenumber

   If linenumber = 00000, error occurred in direct execution mode

2. Error codes

   1. Input line over 72 characters
   2. Numeric overflow
   3. Illegal character or variable
   4. No ending " in print statement
   5. Dimensioning error
   6. Illegal arithmetic
   7. Line number not found
   8. Divide by zero attempted
   9. Excessive subroutine nesting (>8)
   10. RETURN without GOSUB
   11. Illegal variable
   12. Unrecognizable statement
   13. Parenthesis error
   14. Memory full
   15. Subscript error
   16. Excessive loops nesting (>8)
   17. NEXT without FOR
   18. Illegal string arithmetic

## AUTOMATIC RESTART

The basic interpreter is extended with an automatic restart after power down. Two commands control the automatic restart: AUTO ON and AUTO OFF. These commands are to be used in a statement, as in 0025 AUTO ON

Once the AUTO ON statement is executed, the module is forced into "run" after a power failure. Normally, the run command clears all variables. In case of the automatic run however, variables declared by a DIM statement are preserved. All other variables are cleared.

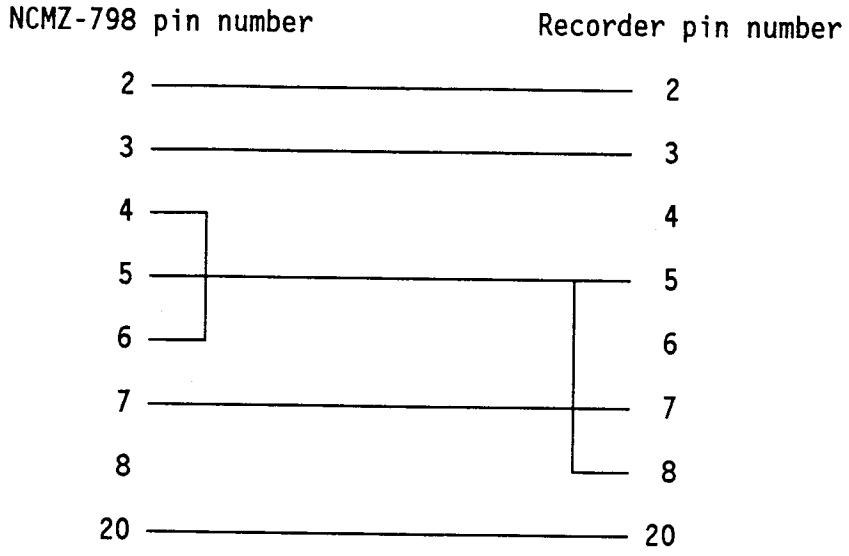
Example:

```
0001 AUTO ON
0005 DIM A(10)
0010 IF A(10)=0 GOTO 20
0015 PRINT "STARTUP AFTER POWER FAILURE"
0020 A(10)=5
0025 PRINT "PROGRAM RUNNING"
0030 FOR B=1 TO 400
0035 NEXT B
0040 GOTO 25
```

## I/O CABLE CONNECTIONS

Both port A and port B are wired for Module-to-DTE communication (see page 5 of the communication module description, catalog No. NCMZ-798). In this way, most terminals and printers can be connected directly to the module. Proper motor control for the SANYO or STR-LINK II cassette recorder requires some cross-wiring, as described below.

Sanyo and STR-LINK II connection



NOTE

In the following instructions, some steps are labeled a. and b., where

a. = STR-LINK II instructions

b. = Sanyo instructions

SAVING A PROGRAM ON TAPE

1. Connect the recorder to port B

2a. Rewind tape, press WRITE TAPE, wait for tape movement to stop

2b. Rewind tape, press PLAY+REC, MODE switch on DATA

3. Enter SAVE on your terminal

4. Wait for READY message on terminal (tape movement stops)

LOADING A PROGRAM FROM TAPE

1a. Press REWIND, press READ TAPE

1b. Rewind tape, press PLAY, MODE switch on DATA

2. Enter LOAD on your terminal

3. Wait for READY message on terminal (tape movement stops)

## SAVING DATA ON TAPE

Data can be saved on tape under control of a BASIC program. A program example is given, to illustrate the actions to be taken.

```
01 DIM X(100)                          Define array
09 REM SUBROUTINE TO WRITE DATA TO TAPE
10 PORTB                               Set output to port B
20 FOR C= 1 TO 1000                    Delay to create a trail
30 NEXT C
40 FOR C= 1 TO 100                     100 array elements to tape
50 PRINT X(C)
60 FOR D=1 TO 150                      Delay between elements
70 NEXT D
80 NEXT C
90 PORTA                               Set output to port A
99 RETURN
```

## LOADING DATA FROM TAPE

This program example reads in the array X

```
199 REM SUBROUTINE TO READ ARRAY FROM TAPE
200 OPEN                               Set input from port B
210 FOR C = 1 TO 100                   Read 100 elements
220 INPUT X(C)
230 NEXT C
240 CLOSE                              Input from port A again
250 RETURN
```

## AND / OR FUNCTION PROGRAM EXAMPLE

The AND and OR operations allow you to manipulate single bits of the variable.

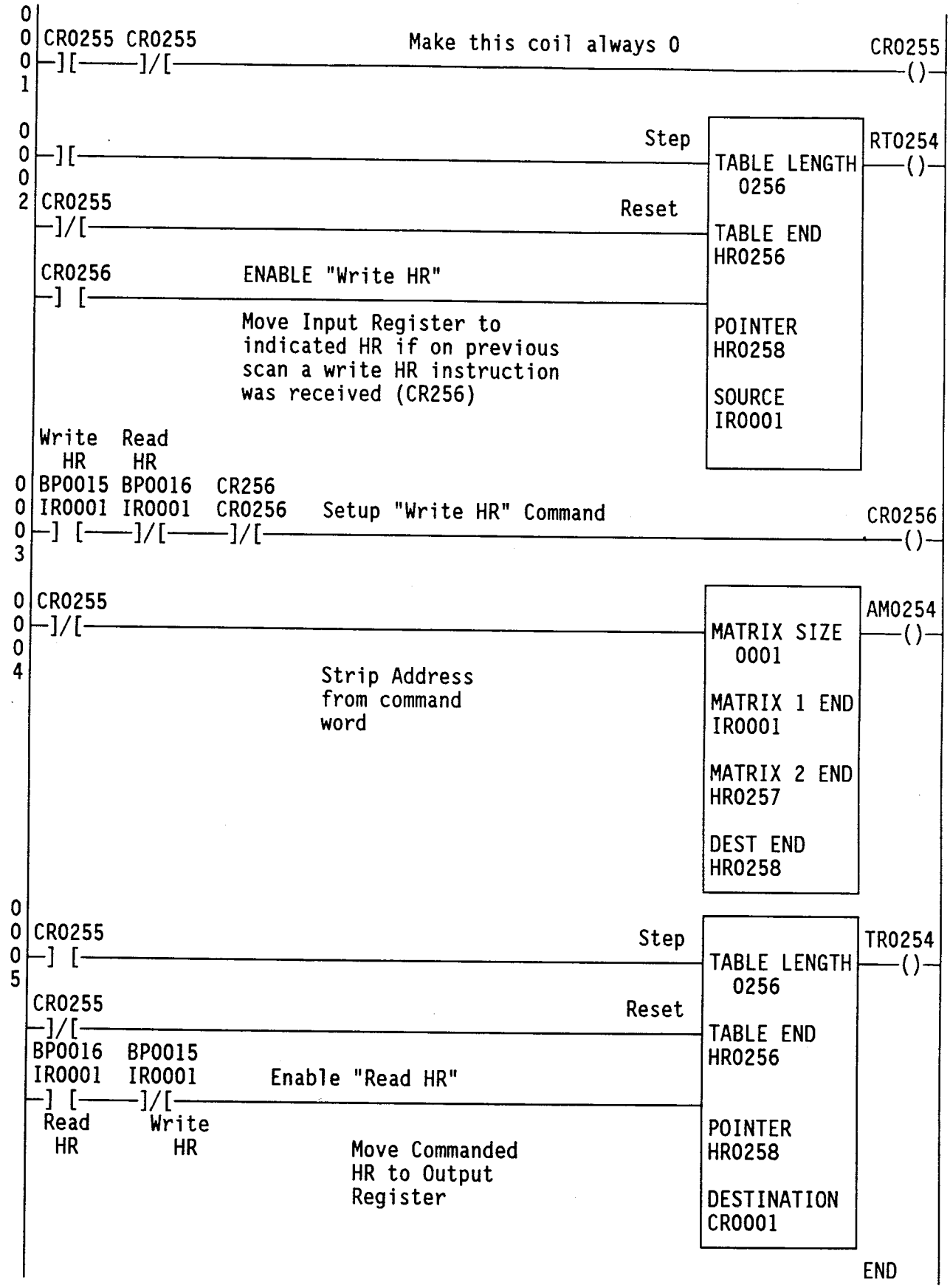
The value of a variable is represented by 16 bits two's complement, for example:

```
    -1    is   1111 1111 1111 1111
+32767    is   0111 1111 1111 1111
-32767    is   1000 0000 0000 0001
   +1     is   0000 0000 0000 0001
```

The program below demonstrates the use of the AND and OR operators. With the CRT program loader in "MONITOR MODE", showing HR 1 in binary format, you can easily check what happens.

```
010 PRINT "program to demonstrate AND and OR"
020 REM LINES 30-100 FILL ARRAY WITH SINGLE BIT INTEGERS
030 DIM B(16)
040 A=1
050 FOR C=1 TO 15
060 B(C)=A
065 IF C=15 GOTO 80
070 A=A*2
080 NEXT C
090 B(16)=-1-32767
100 PRINT "enter bitnumber to be changed ";
110 INPUT A
120 IF A>0 IF A<17 GOTO 130
125 GOTO 100
130 PRINT "enter SET or CLEAR ";
140 INPUT $
150 IF $ = "SET" GOSUB 200
160 IF $ = "CLEAR" GOSUB #00
170 GOTO 100
200 REM       SUBROUTINE TO SET A BIT
210 R(1)=R(1)^B(A)
220 PRINT "BIT ";A;" is set by or-ing R(1) with ";B(A)
230 RETURN
300 REM       SUBROUTINE TO CLEAR A BIT
310 R(1)=R(1). (-1-B(A))
320 PRINT "BIT ";A;" is cleared by and-ing R(1) with ";-1-B(A)
330 RETURN
```

# APPENDIX A: LADDER DIAGRAM


CR0255 CR0255
Make this coil always 0
CR0255
Step
RT0254
TABLE LENGTH
0256
Reset
TABLE END
HR0256
CR0256
ENABLE "Write HR"
POINTER
HR0258
Move Input Register to
indicated HR if on previous
scan a write HR instruction
was received (CR256)
SOURCE
IR0001
Write HR
Read HR
BP0015 BP0016 CR256
IR0001 IR0001 CR0256
Setup "Write HR" Command
CR0256
CR0255
AM0254
MATRIX SIZE
0001
MATRIX 1 END
IR0001
Strip Address
from command
word
MATRIX 2 END
HR0257
DEST END
HR0258
CR0255
Step
TR0254
TABLE LENGTH
0256
CR0255
Reset
TABLE END
HR0256
BP0016 BP0015
IR0001 IR0001
Enable "Read HR"
Read HR
Write HR
POINTER
HR0258
Move Commanded
HR to Output
Register
DESTINATION
CR0001
END

# APPENDIX B: INSERTING LADDER IN HPPC FOR NCM BASIC

The application of the NCMZ-798 in a system using the HPPC as the processor requires special procedures be followed. This discussion will attempt to explain the procedure.

The problem in this module's operation with the HPPC is caused by the non-synchronous nature of the HPPC's I/O operation. The NCMZ-798 requires synchronous operation of its I/O, like that found in the PC-700 and PC-900. In addition the time between I/O cycles for the NCMZ-798 must be in excess of 6 milliseconds.

The ladder diagram for the HPPC is shown in figure 2. Note the use of UPDATE IMMEDIATE I/O functions. The address of the NCMZ-798 module can only be addressed by the UPDATE IMMEDIATE function for this module to work correctly.

To achieve this, the following procedure is recommended:

A. The use of the module must be planned when laying out the system I/O. A defined break in I/O assignments must be planned in the SIM in which the module will be mounted. This break is in the assignment of I/O groups and includes discrete I/O as well as register I/O. The concern is that the I/O bus signal -"GSEL"- is only used for the NCMZ-798. It is difficult to cover the possibilities of this assignment because each SIM starts numbering its own assigned I/O with the lowest number as if it started as GSEL1 (IG0001, IR0001, OG0001, OR0001) even though the group could be any group (even IG0002). An example of SIM assignment is shown in Figure 1.

   A SIM is capapable of generating 4 "GSEL's" if enough I/O is configured to it. Each GSEL will address 4 groups of discrete I/O (IG & OG) and 8 I/O registers (IR & OR). As I/O is assigned to a SIM, the first four groups of discrete are assigned to GSEL1, the next four groups of discrete are assigned to GSEL2, etc. As register I/O is assigned to a SIM, the GSEL assignment is separately calculated. The first 8 I/O registers are assigned to GSEL1, the second 8 I/O registers are assigned to GSEL2, etc. If any I/O is unassigned it will be addressed to the SIM having the appropriate space in its address structure. Addressing of unassigned I/O occurs only through UPDATE IMMEDIATE I/O functions. The NCMZ-798 must reside in unassigned I/O space and neither discrete nor register I/O can be assigned which would cause the GSEL associated with the NCMZ-798 to be generated.

B. Prior to entering the ladder diagram, the system is configured with I/O assigned for everything including the NCMZ-798. (see Figure 1)

C. The ladder diagram entered, specifically the ladder for the NCMZ-798, may include any added ladder desired.

D. Save the ladder on disk.

E. Reconfigure the system but do not assign the I/O of the NCMZ-798.

F. Reload the ladder from disk but do so using the processor configuration.

The ladder now in the memory of the HPPC will contain the necessary program for communicating with the NCMZ-798 module. Add any additional ladder to obtain the complete system operation. When the final ladder program is saved, it will be with this new processor configuration.

It should also be noted that when this module was designed, it was designed to work with the PC-700 and the PC-900. In these processors, HR0001 is memory address zero. In the HPPC, HR0001 can be any address as found in the parameter table. This difference in concept causes a reguest for HR0001 from the NCMZ-798 to yield the contents of HR0002. Therefore reguests for specific holding registers must calculate the number as the register desired minus one.

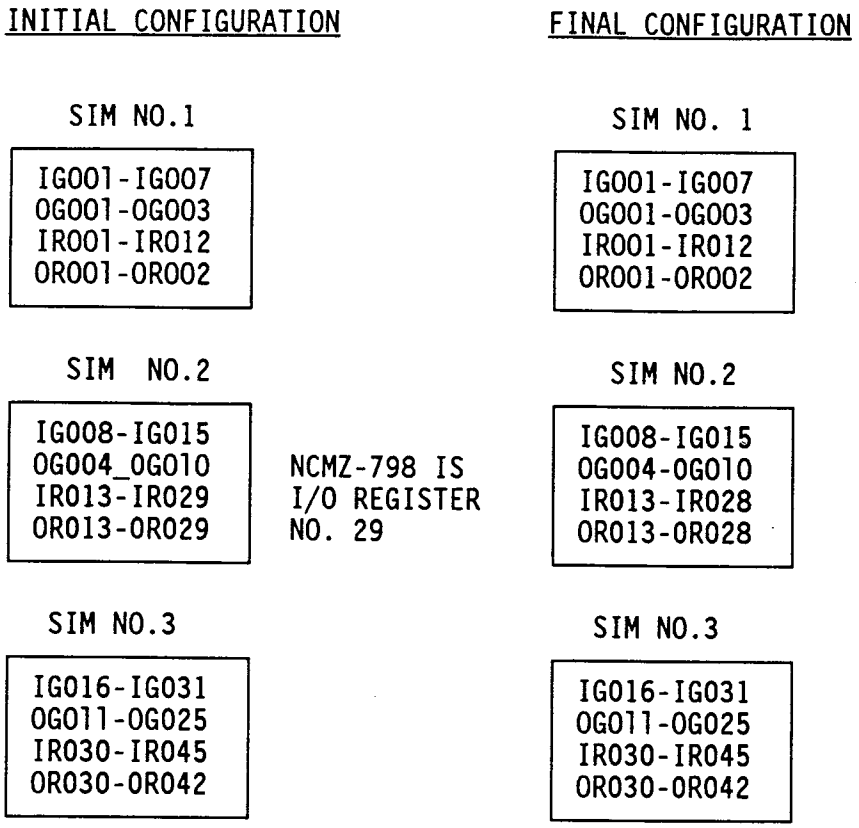
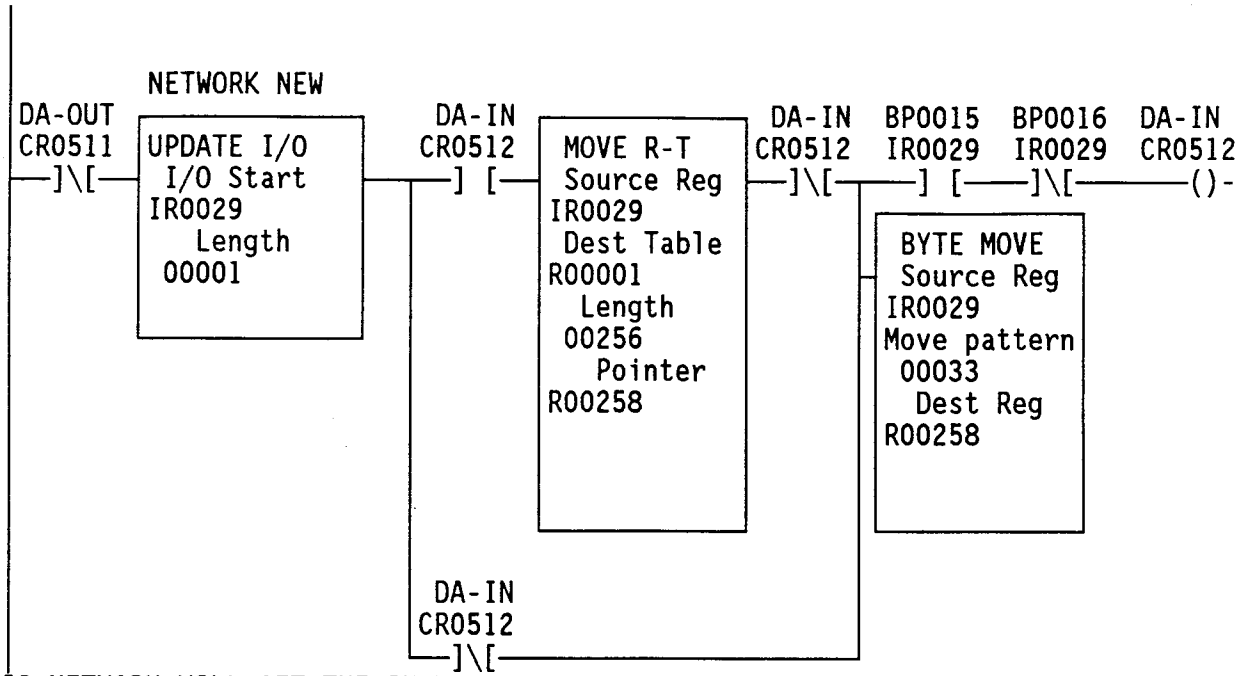
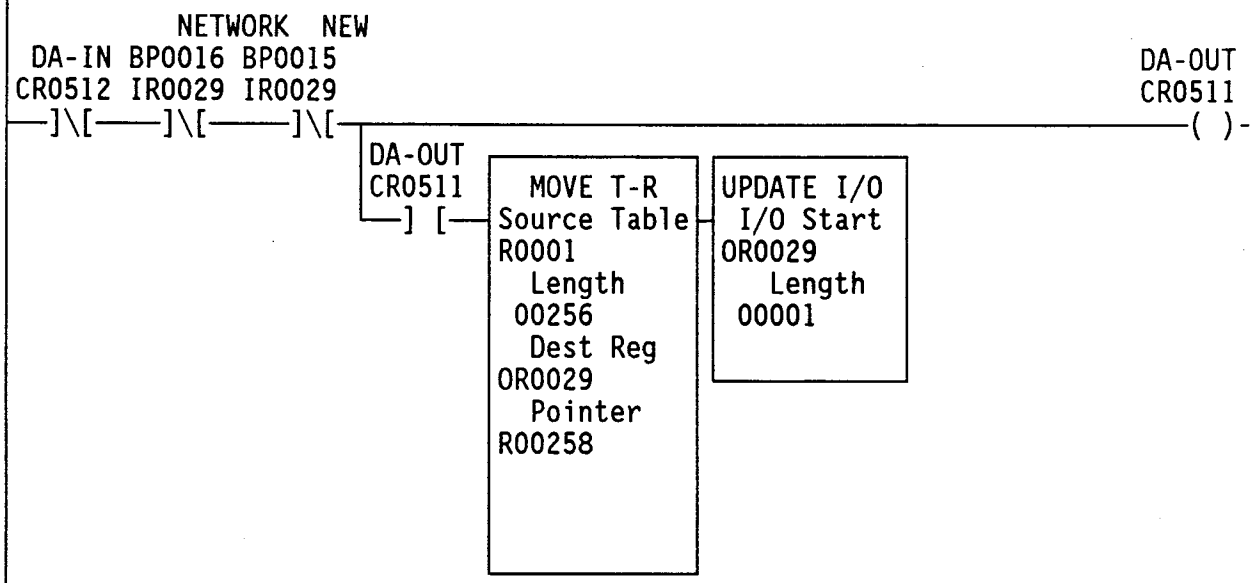

INITIAL CONFIGURATION
FINAL CONFIGURATION
SIM NO.1
IG001-IG007
0G001-0G003
IR001-IR012
0R001-0R002
SIM NO. 1
IG001-IG007
0G001-0G003
IR001-IR012
0R001-0R002
SIM NO.2
IG008-IG015
0G004_0G010
IR013-IR029
0R013-0R029
NCMZ-798 IS
I/O REGISTER
NO. 29
SIM NO.2
IG008-IG015
0G004-0G010
IR013-IR028
0R013-0R028
SIM NO.3
IG016-IG031
0G011-0G025
IR030-IR045
0R030-0R042
SIM NO.3
IG016-IG031
0G011-0G025
IR030-IR045
0R030-0R042

Figure 1.




THIS NETWORK WILL GET THE INPUT FROM ASSIGNED INPUT REGISTER AND TEST FOR INPUT IF INPUT IS REQUIRED IT OCCURS ON THE NEXT PROGRAM PASS - THE BYTE MOVE FUNCTION SEPARATES THE ADDRESS FOR INPUT OR OUTPUT FROM THE COMMAND WORD FROM THE NCMZ-798




THIS NETWORK PERFORMS THE OUTPUT FUNCTION WHEN REQUESTED BY THE NCMZ-798 - CR511 SYNCHRONIZES THE OUTPUT FUNCTION TO OCCUR ON THE NEXT PROGRAM PASS - THE ADDRESS TO BE OUTPUT COMES FROM THE PREVIOUS NETWORK.

End

# APPENDIX C. PROGRAMMING THE NCMZ-798 CARD WITH THE IBM PERSONAL COMPUTER

This appendix will show how the NCMZ-798 communication card can be programmed using the IBM (or compatible) Personal Computer with some related software. This method is potentially far superior to the current method of programming via a "dumb terminal" and a STR-LINK digital cassette recorder due to the fact that a program can be written Off-line and stored on floppy disk versus digital tape.

Previously, in order to store and load programs from the NCMZ-798 card, a cassette recorder connected to PORT B along with the commands SAVE and LOAD allowed the card to upload or download the files. This method is documented earlier in this manual.

An alternative for program development and storage is using an IBM personal computer in conjunction with a software package called "PC-TALK". This package was written primarily for use with modems but it works nicely with this application also. The user will develop his program using Wordstar or Edlin then use PC-TALK to load it into the NCMZ-798 card.

First, the user must develop the program. This can be accomplished On-line using PC-TALK or Off-line using Wordstar. If the user is writing the program On-line, he must simply start-up PC-TALK, connect the computer to the NCMZ-798 card with a straight-thru cable, and strike the Return key to get a system prompt. The "#" sign is considered the system prompt for the NCMZ-798 card. Refer to the NCMZ-798 BASIC Manual for programming instructions. At this point the IBM (or other computer) is acting as a "dumb terminal". Also note that the cable connection must be made through a serial port on the IBM identified as COM1 or COM2. You can not use the AST CC-232 card or the parallel port for this communication channel. Refer to the PC-TALK instruction manual for configuration instructions.

If the User intends to develop his program Off-line, he must develop an ASCII file using Wordstar or Edlin (from PC-DOS or MS-DOS). Using Wordstar, use the Non-Document option when creating the file, in the same way as an AUTOEXEC.BAT file is created. The first line of the program should be left-justified whereas every other line MUST be offset by 2 spaces. An example program is entered below.

```
00010 DIM A(32),B(32)
  00020 FOR I=1 TO 32
  00030 A(I)=I
  00040 B(I)=-I
  00050 PRINT A(I),B(I)
  00060 NEXT I
  00090 END
```

When a program is copied from the NCMZ-798 card and read by Wordstar or Edlin, it looks like the following:

```
00010 DIM A(32),B(32)
^@^@00020 FOR I=1 TO 32
^@^@00030 A(I)=I
ETC.
```

If you make any changes, simply leave 2 blank spaces as discussed and ignore the characters you see above. These characters represent a NULL space for the NCMZ-798 card. You can also ignore the leading zeros for the line numbers.

To save a program on floppy disk, use the SAVE command of the NCMZ-798 command set. At the system prompt (#), type SAVE. Then use the Alt-R command of PC-TALK which will request the file name and open the computer to accept the file. Then connect the cable to port B of the NCMZ-798 card (since that is where the programs are loaded from). After the file is transmitted, close the file by typing Alt-R again. Obviously, if you had an RS-232 switch box it would greatly simplify this task, but it is not necessary. Switching the cable back to port A will get the system prompt back.

To load a program, use the Alt-T command and the LOAD command from the NCMZ-798. With the cable connected to port A, first type LOAD, switch the cable to port B, then type Alt-T from the PC-TALK command structure. After the file is transmitted, reconnect the cable to port A to find the system prompt.

NOTE: Required hardware includes:
IBM Personal Computer
Serial Port (this is available in many configurations including 9-pin or 25-pin D-shell. The 25-pin type can use an NLC-3PL (CRT cable)).