SP798 - 41

Basic interpreter

Westinghouse Elektrotechniek en
Instrumentatie B.V.
Hof van Zaenden 65
1508-XD Zaandam

./.

TABLE OF CONTENTS

INTRODUCTION	page 3
INSTALLATION	page 3
COMMANDS	page 4
EDITING and LINEFORMATS	page 5
OUTPUT STATEMENT	page 5
INPUT STATEMENT	page 6
VARIABLES	page 7
CONTROL STATEMENTS	page 8
ASSIGNMENT STATEMENT	page 9
ARITHMETIC OPERATORS	page 9
RELATIONSHIP TESTS	page 10
FUNCTIONS	page 10
SHORT FORM VOCABULARY	page 11
ERROR MESSAGES	page 12
I/O CONNECTIONS	page 13
SAVING PROGRAMS	page 14
LOADING PROGRAMS	page 14
SAVING DATA	page 15
LOADING DATA	page 15
AND/OR EXAMPLES	page 16
RELEASE NOTES	**page 17**
LADDER PROGRAM	APPENDIX I

./.

INTRODUCTION

Various dedicated software packages in the NCMZ-798 module make it possible to communicate with Westinghouse PC-700 or PC-900 at RS232 level. Most of these programs have a fixed task, and offer no possibilities to extend or change the modules function.
NCM-BASIC was developped 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 modules 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 mathematic
- easy access to PC holding registers
- large program storage area
- one string variable
- high speed of execution

Westinghouse Elektrotechniek en Instrumentatie B.V. cannot assume any responsibility for any applications or uses of this program, nor for any errors or the consequent results thereof.

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 the Appendix. 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 page 13 of this manual.

COMMANDS

NEW	deletes all lines and data
LIST	lists the program as follows:
LIST	lists the entire program
LIST X	lists line with linenumber X
LIST X-Y	lists all lines between X and Y
SIZE	prints two decimal numbers 1 - number of bytes, used by program 2 - number of free bytes Array strorage 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: 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 for instance does PRINT #7 ring a bell?
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 seperated 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)
   example: DIM X(5,10),Y(C+30) defines two arrays
   the maximum subscript size is 255
   no minus or zero subscript allowed
   subscripts can be an expression: LET X(B,C)=3

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 terminals keyboard. All relationship tests and arithmatic 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 ...
```

./.

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 (any key pressed?)
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
CLOSE	closes port B

control statements

FOR .. TO .. NEXT	program loop between FOR and 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

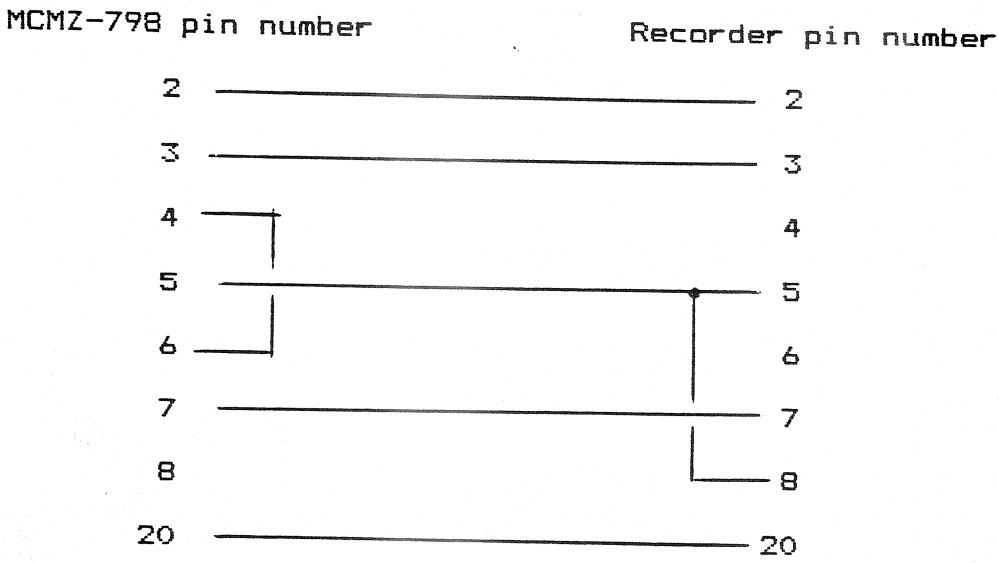
2. error codes

   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

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



./.

SAVING A PROGRAM ON TAPE

1. Connect the recorder to port B according to page 13

2a. Rewind tape, press WRITE TAPE, wait for tapemovement 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 (tapemovement 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 (tapemovement stops)

NOTE

a. = STR-LINK II instructions
b. = Sanyo instructions

./.

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 portB
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 portA
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 portB
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 300
170 GOTO 100
200 REM      SUBROUTINE TO SET A BIT
210 R(1)=R(1)∧B(A)
220 PRINT "BIT ";A;" is set by orr-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
```

./.

RELAEASE NOTES

september 1984

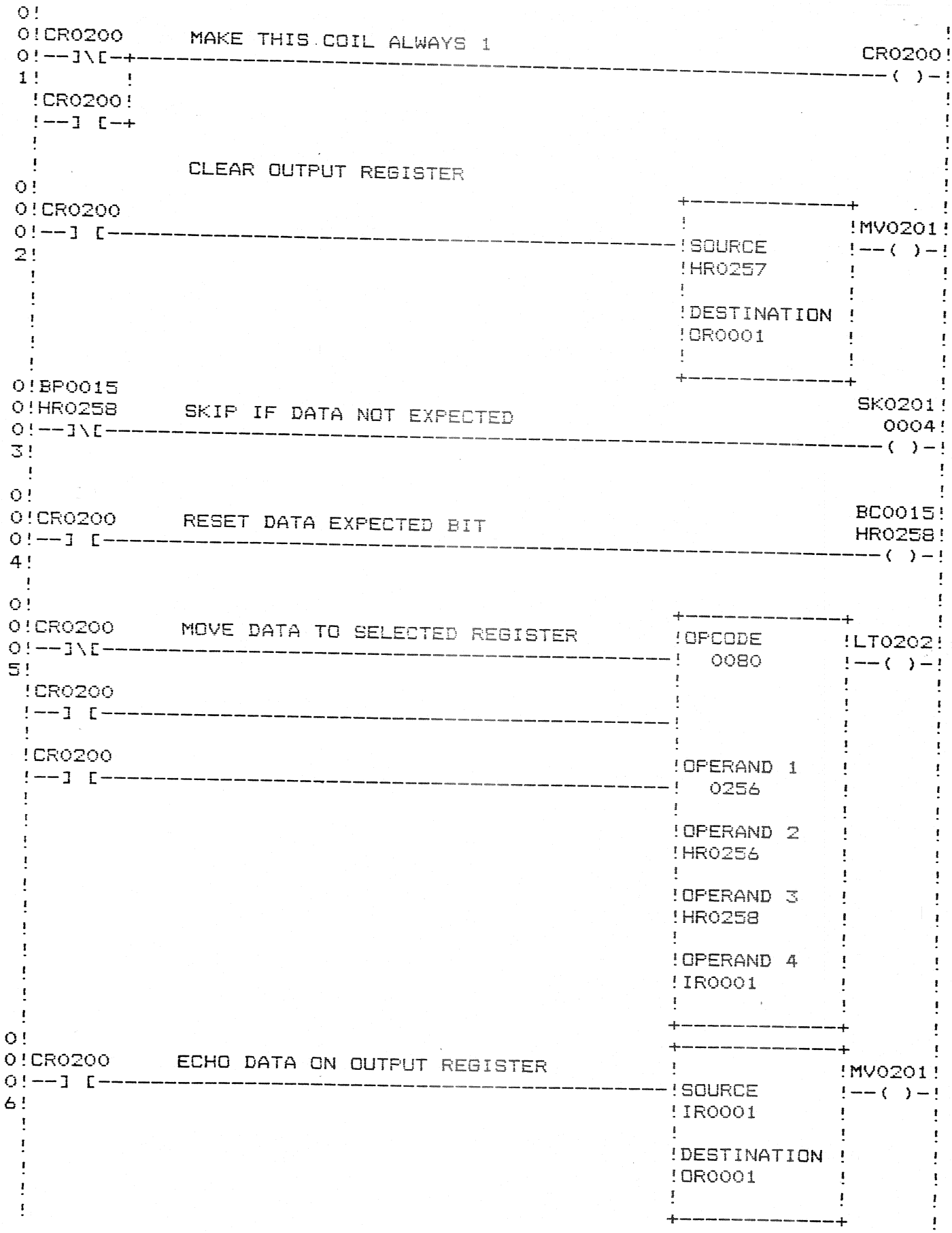
The basic interpreter is extended with an automatic restart after power down. Two commands control the automatic restart: **AUTO ON** and **AUTO OFF**.
These commands are to be used in a statement, as in
0025 AUTO ON

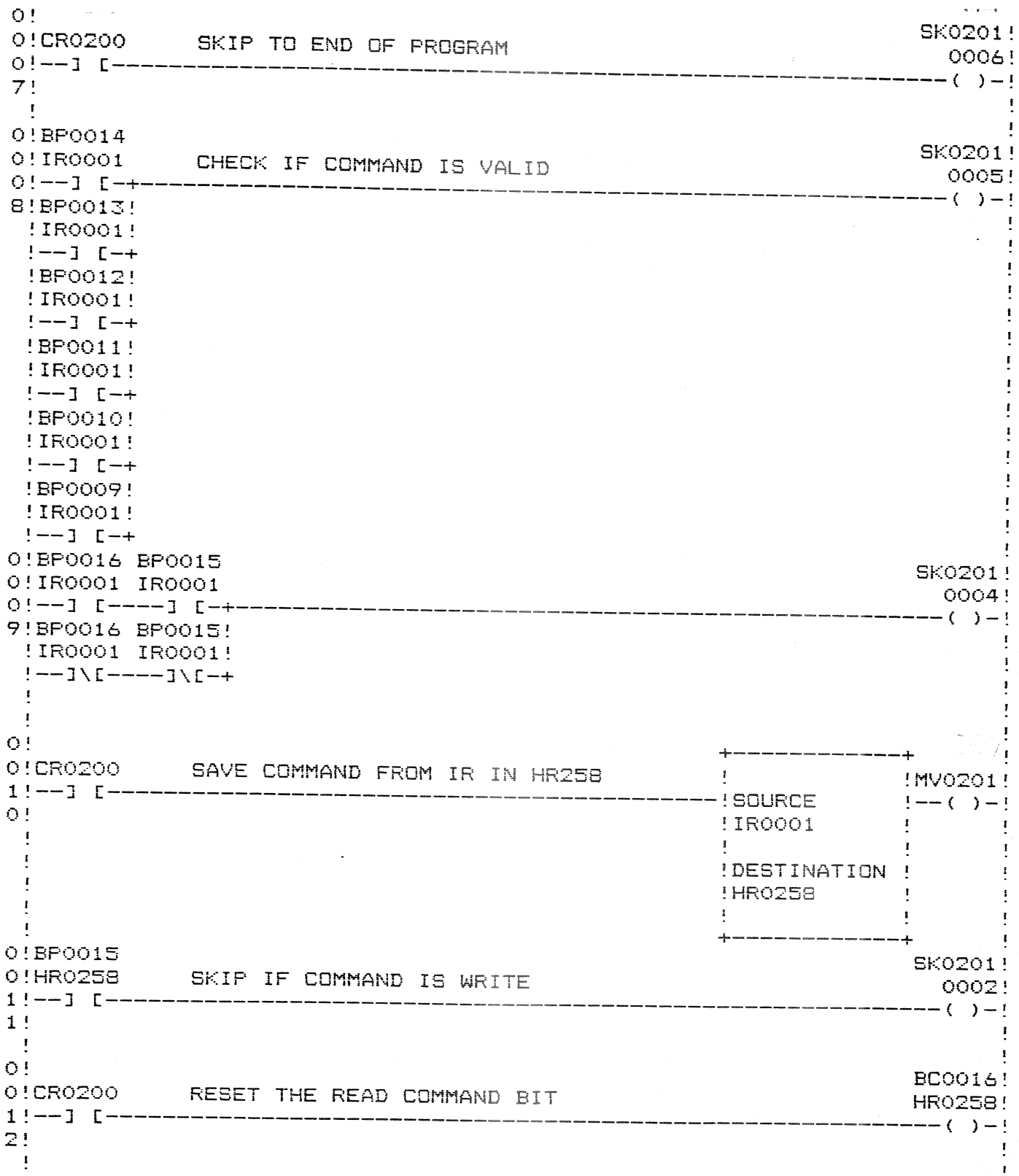
Once the AUTO ON statement is executed, the module is forced into "run" after a power failure. Normally the run command clears all variables. In case of the automatic run however, variables declared by a DIM statement are preserved. All other variables are cleared.

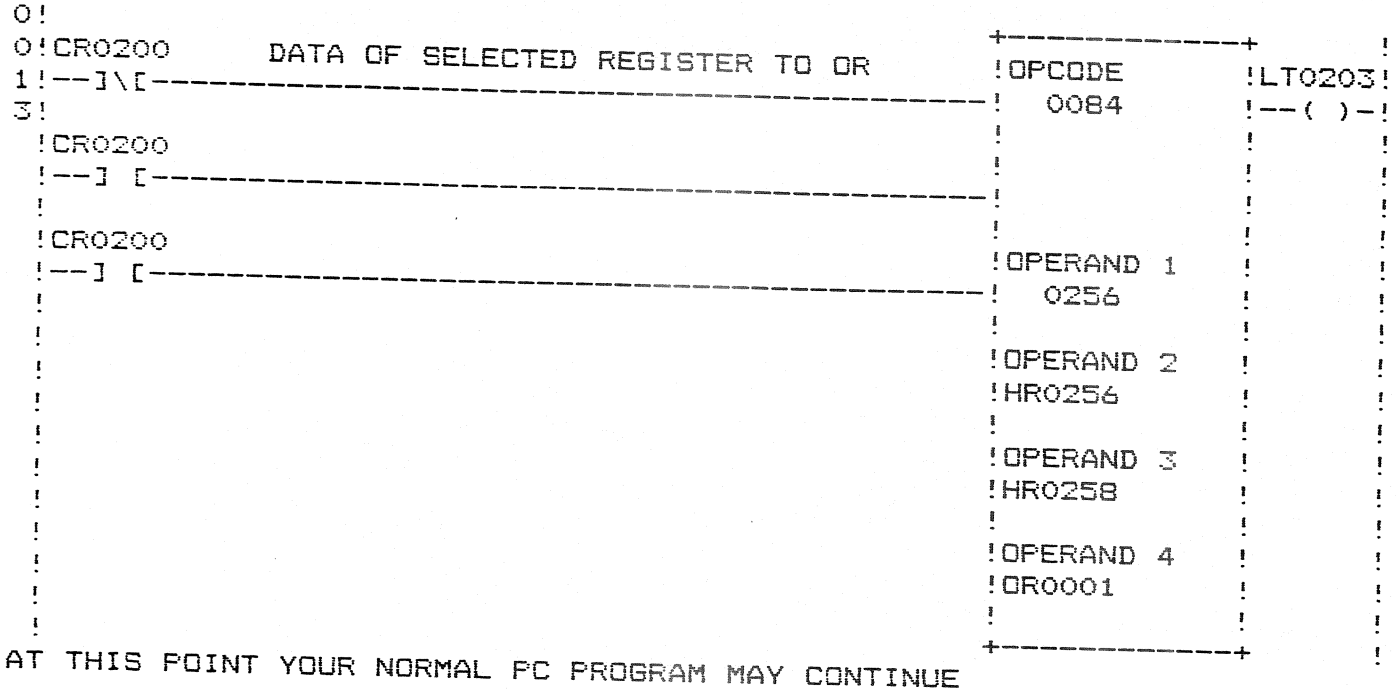
PROGRAM EXAMPLE

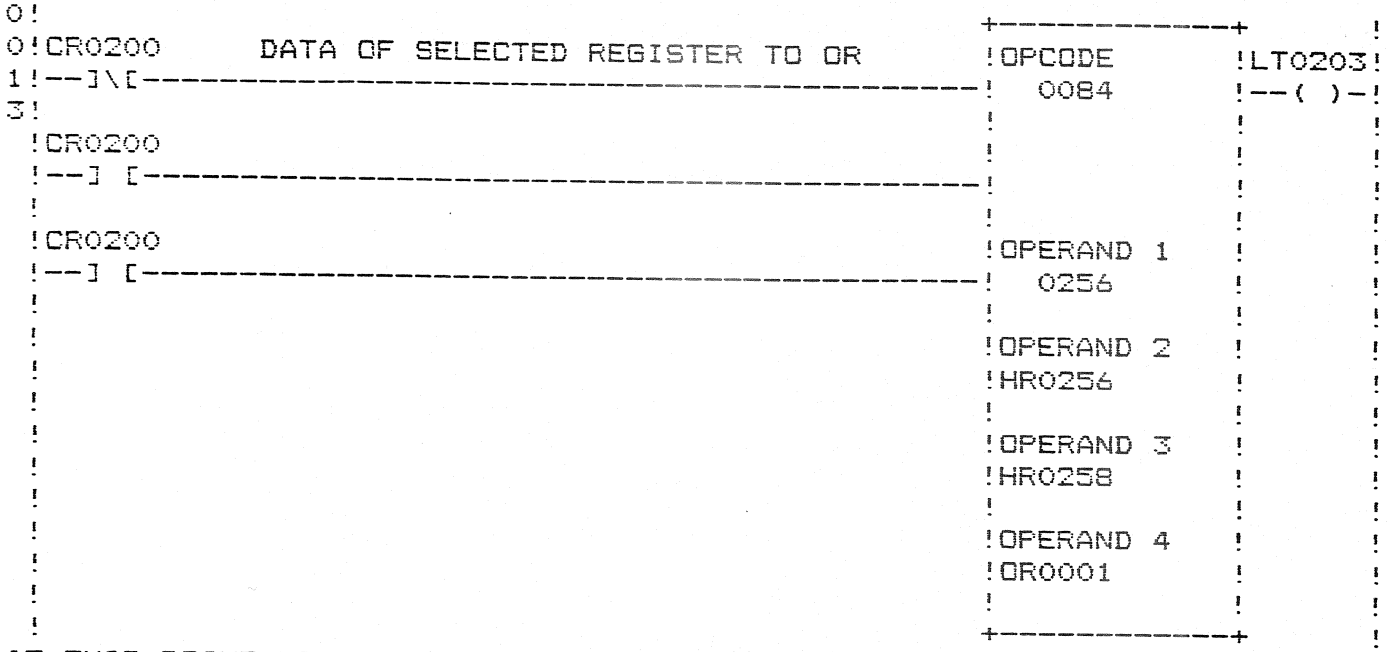
```
0001 AUTO ON
0005 DIM A(10)
0010 IF A(10)=0 GOTO 20
0015 PRINT "STARTUP AFTER POWER FAILURE"
0020 A(10)=5
0025 PRINT "PROGRAM IS RUNNING"
0030 FOR B=1 TO 400
0035 NEXT B
0040 GOTO 25
```

./.


MAKE THIS COIL ALWAYS 1
CR0200
CLEAR OUTPUT REGISTER
SOURCE
HR0257
DESTINATION
OR0001
MV0201
SKIP IF DATA NOT EXPECTED
SK0201
0004
RESET DATA EXPECTED BIT
BC0015
HR0258
MOVE DATA TO SELECTED REGISTER
OPCODE
0080
OPERAND 1
0256
OPERAND 2
HR0256
OPERAND 3
HR0258
OPERAND 4
IR0001
LT0202
ECHO DATA ON OUTPUT REGISTER
SOURCE
IR0001
DESTINATION
OR0001
MV0201


CR0200 SKIP TO END OF PROGRAM
SK0201 0006
BP0014
IR0001 CHECK IF COMMAND IS VALID
SK0201 0005
BP0013
IR0001
BP0012
IR0001
BP0011
IR0001
BP0010
IR0001
BP0009
IR0001
BP0016 BP0015
IR0001 IR0001
SK0201 0004
BP0016 BP0015
IR0001 IR0001
CR0200 SAVE COMMAND FROM IR IN HR258
SOURCE
IR0001
DESTINATION
HR0258
MV0201
BP0015
HR0258 SKIP IF COMMAND IS WRITE
SK0201 0002
CR0200 RESET THE READ COMMAND BIT
BC0016 HR0258


CR0200
DATA OF SELECTED REGISTER TO OR
CR0200
CR0200
OPCODE
0084
OPERAND 1
0256
OPERAND 2
HR0256
OPERAND 3
HR0258
OPERAND 4
OR0001
LT0203
AT THIS POINT YOUR NORMAL PC PROGRAM MAY CONTINUE



AT THIS POINT YOUR NORMAL PC PROGRAM MAY CONTINUE