Appendix A Editor

BASIC programs usually are created and maintained by using an editor. Programs can be created Online (connected to the 1799) or Offline (developed in an ASCII text editor).

> NOTE
>
> When developing programs offline, ensure that the editor does not insert any non-printable formatting characters. The file created for downloading to the module must be pure ASCII.

The 1799 module includes a built in editor for online program development.

The editor is line oriented. All statements have a line-number, followed by a blank at the start of each line, The editor is invoked by typing the BASIC command 'EDIT'.

The system will reply with:

```
Westinghouse text editor VI.1
._
```

and from now on until you return to BASIC command level, the system prompt will be '.'

The editor command set includes all the necessary commands for rapidly changing and writing a program, such as AUTO line increment, global search and replace, moving, duplicating and lines deleting records.

A command summary is given on your screen, if you type: **H** (help screen).

This part of the manual describes the commands in detail in alphabetic order.

A.1 General command syntax

All editor commands consist of a single character followed by a range specification. In some cases, the range specification may be omitted, and then the range defaults to i-32767.

Range specifications have the format:

F-L, F- or -L or N, in which	F= first line number.
	L= last line number.
	N= any single line number.

The commands will only effect the lines in the specified range. Spaces in the range specification are ignored.

A.2 Append

Syntax : A [range]/string[/]

Append adds a string to the end of a line or a group of lines.

A.3 Block mode

Syntax : B range.

Block changes the default range specification values.

Initially, the default range specification is set from O-32767 and consequently, if you omit the range specification in a command, it will default to this value.

By using the block mode command, you can use global change and find commands in a specific part of the program, without the need of specifying ‘range’ over and over again.

A.4 Change

Syntax : C [range]/string l/string 2/

Change changes the first occurrence of string 1 into string 2 within a line or a group of lines.

The delimiter "/" may be replaced by any other character.

This command is the normal line edit command.

A.6 Duplicate

Syntax : D range, dest [,inc]

Duplicate makes a copy of a line or a group of lines elsewhere in the program.

Range	: specifies the lines to be duplicated.
dest	: specifies the destination line number.
inc	: specifies the line-increment, to be used.

If the destination line will overlap the existing program, a *destination conflict* error message will appear on the screen and no records will be duplicated.

A.6 Erase

Syntax : E range.

Erase removes a line or group of lines from your program.

A single line number and carriage return will also erase that line.

A.7 Find

Syntax : F [range]/string 1[/]

Find finds the occurrences of string 1 in the program area, specified by range, and prints the occurrences on the screen.

The delimiter "/" may be replaced by any other character.

A.0 insert

Syntax : I F [,inc]

The insert command invokes the insertion mode. Line numbers are automatically generated, starting from F (first line number) and will increment by inc. If increment is omitted, it will default to IO.

Insertion is terminated, if only a carriage return is typed.

Warning
In the insertion mode, it is possible to overwrite lines, that were previously programmed.

A.9 List

Syntax : L [range]

Lists the lines specified by range, on your screen.

A.10 Move

Syntax : M range, dest [,inc]

Move	: moves a line or group of lines elsewhere in your program.
Range	: specifies the lines to be moved.
dest	: specifies the destination line-number
inc	: specifies the line-increment.

As with duplicate, a *destination conflict* error may occur, and no records will be moved.

A.11 Quit

Syntax : Q.

Quit will return you to the BASIC command level. The prompt, #, indicates that BASIC is at the Command level. You cannot execute a program from the EDIT mode.

A.12 Help

Syntax : H.

Help gives an editor command overview on your screen.

** editor command summary**

Range has the general format F-L.
Other formats : F, -L.

F	= first line number.
L	= last line number.
dest	= destination line number.
inc	= line increment.

A [range]/string	append.
B range	block mode.
C [range]/string l/string 2 [/]	change.
D range, dest[,inc]	duplicate.
E range	erase.
F [range]/string 1	find.
I F [,inc]	insert.
L [range]	list.
M range, dest[,inc]	move.
Q	quit.

Appendix B Memory Map

	CPU Address	RAM Address	PROM
SYSTEM MEMORY	$0000	0000	----
	$10FF	$10FF	
USER RAM MEMORY	91100	$1100	----
	$79FF	$79FF	
SYSTEM STACK	$7A00	$7A00	----
	$7FFF	$7FFF	----
EPROM BACK-UP MEMORY (MAX. 16K)	88000	-----	0000
			$3FFF
PLC-IO TWO-PORT MEMORY	$C000	SC000	----
FREE FOR USER	$C100	$C100	----
NOT USED	$C400	SC400	----
DUART PORT C + D	SC600	$C600	----
DUART PORT A + B	SC700	$C700	----
BASIC EDITOR BASIC INTERPRETER	SC800	----	$4800
			$7FFF

For more information on PROM storage, refer to Appendix I (page 91).

$0A-$0B	LINE POINTER
$7F	"UNSIGNED" FLAG
$DA-$F4	LOCAL VARIABLES STORAGE
$D8	TASK RUN-TIMER
$D9	MAXIMUM RUNTIME FOR TASK
$F4	SUSPEND TIMER
SF6	01h - PC1200 I/O Bus, 00h - PC1 100 I/O Bus

For each additional TASK these locations must be increased by $100. For instance POKE ($17F,1) allows for unsigned calculations in the first additional task.

$856,$857	POINTER TO END OF BASIC PROGRAM
$3F7-$7F7	PORT INPUT & OUTPUT BUFFERS
$85A	AUTO-RESTART FLAG
$85C	I/O COMMUNICATION REGISTER BASE ADDRESS
$85D,$85E	PORT A INTERRUPT VECTOR
$85F,$860	PORT B INTERRUPT VECTOR
$861,$862	PORT C INTERRUPT VECTOR
$863,$864	PORT D INTERRUPT VECTOR
$865,$866	EXTRA INTERRUPT VECTOR
$867-$8B5	GLOBAL VARIABLE STORAGE
$8B5-$10EF	STRING VARIABLE STORAGE
$1 0F8	SYSTEM TIMER (TM)
$C800	START EDITOR

The 68681 UART registers are memory mapped into the 32K RAM space that is accessed by the 6809. Each channel of the UART is assigned 8 bytes of this read/write memory. Each byte of this memory has two uses. Reading from these addresses is used to retrieve data and UART status. Writing to this memory reprograms the UART and sends data out the serial port. There are two 68681 UARTs mounted on the 1799. One UART handles PORT A and PORT B, while the other handles PORT C and PORT D. Since the ports are grouped together, certain programming functions on one port could affect the other (38.4K baud selection for example).

$C600	BASE ADDRESS PORT C
$C608	BASE ADDRESS PORT D
$C700	BASE ADDRESS PORT B
$C708	BASE ADDRESS PORT A
$FFFC-$FFFD	NMI VECTOR
$FFFE-$FFFF	6809 RESTART VECTOR

MC68681 Register to Memory Map Cross-Reference

Address	Value retrieved when location read from	Action taken when location written
Port c		
$C600	Mode Register Port C	Mode Register Port C
$C601	Status Register Port C	Clock Select Register C
$C602	Do not Access	Command Register Port C
$C603	Receiver Buffer Port C	Transmitter Buffer Port C
$C604	Input Port Change Reg	Auxiliary Control Register
$C605	Interrupt Status Reg.	Interrupt Mask Register
$C606	Counter Mode MSB C	Counter/Timer Upper Register C
$C607	Counter Mode LSB C	Counter/Timer Lower Register C
Port D		
$C608	Mode Register Port D	Mode Register Port D
$C609	Status Register Port D	Clock Select Register D
$C60A	Do not Access	Command Register Port D
$C60B	Receiver Buffer Port D	Transmitter Buffer Port D
$C60C	Interrupt Vector Register	Interrupt Vector Register
$C60D	Input Port (Unlatched)	Output Port Configuration Register
$C60E	Start-Counter command	Output Port - Bit Set Command
$C60F	Stop-Counter command	Output Port - Bit Clear Command
Port B		
$C700	Mode Register Port B	Mode Register Port B
$C701	Status Register Port B	Clock Select Register B
$C702	Do not Access	Command Register Port B
$C703	Receiver Buffer Port B	Transmitter Buffer Port B
$C704	Input Port Change Reg	Auxiliary Control Register
$C705	Interrupt Status Reg.	Interrupt Mask Register
$C706	Counter Mode MSB	Counter/Timer Upper Register B
$C707	Counter Mode LSB	Counter/Timer Lower Register B
Port A		
$C708	Mode Register Port A	Mode Register Port A
$C709	Status Register Port A	Clock Select Register A
$C70A	Do not Access	Command Register Port A
$C70B	Receiver Buffer Port A	Transmitter Buffer Port A
$C70C	Interrupt Vector Register	Interrupt Vector Register
$C70D	Input Port (Unlatched)	Output Port Configuration Register
$C70E	Start-Counter command	Output Port - Bit Set Command
$C70F	Stop-Counter command	Output Port - Bit Clear Command

Examples of MC66661 Programming from BASIC

POKE ($C70E,2)	SETS RTS PORT A **HIGH**[4]
POKE ($C70F,2)	SETS RTS PORT A **LOW**
POKE ($C70E, 1)	SETS RTS PORT B **HIGH**
POKE ($C70F, 1)	SETS RTS PORT B **LOW**
POKE ($C60E,1)	SETS RTS PORT C **HIGH**
POKE ($C60F, 1)	SETS RTS PORT C **LOW**
POKE ($C60E,2)	ENABLES PORT D TO TRANSMIT[5]
POKE ($C60F,2)	ENABLES PORT D TO RECEIVE
POKE ($C60E,4)	SETS SYNC PORT D **HIGH**
POKE ($C60F,4)	SETS SYNC PORT D **LOW**
PEEK ($C70D).1 (2)=0	CTS PORT B(A) **HIGH** (Note: inverted logic)
PEEK ($C60D).1(2)=0	CTS PORT C(D) **HIGH** (Note: inverted logic)
POKE ($C604,$70)	SELECTS ALTERNATE BAUD PORT C,D[6]
POKE ($C704,$70)	SELECTS ALTERNATE BAUD PORT A,B
Example:	
POKE ($C604,$70)	PORT C(D) 19200,N,8,1:REM SETS PORT C(D) TO 38K BAUD

[4] The RTS lead must be high before any 1799 statement that transmits data is to be used. Depending on the PROM supplied with the module, that may be the default. To verify RTS is high as a default, either monitor the state of the RTS lead with an RS-232 breakout box (and suitable cable to convert the non-standard 9 pin serial port to something the breakout box will understand [see page 87 for cabling diagrams] or alternately measure the voltage between pin 8 and pin 5 of the 9 pin serial port. If pin 8 measures approximately +10 volts DC (as referenced to pin 5 being common), then RTS is high.

[5] The RS-485 port must be "enabled for Transmit" prior to sending data with the PRINT statement. Do not "enable for Receive" immediately after executing the PRINT statement. Rather pause for 20 mS using the SUSPEND 1 statement. Enabling'the RS-485 port for receiving causes the RS-485 transmitter to switch off and any characters buffered by the 68681 UART will not have an opportunity to be transmitted.

[6] When the "Alternate Baud Rate" bit is set high, the 19.2K BPS is replaced with 38.41: BPS. Note that setting this bit high modifies both ports connected to the same 68681 UART. Therefore 19.2K BPS will be replaced on both ports. Ports A and B share a UART as do Ports C and D.

Appendix C Save/Load

The following is **a** step by step instruction, to save and load data to and from a STR-LINK II or Sanyo cassette recorder.

Cabling of port B and recorder is described below.

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.

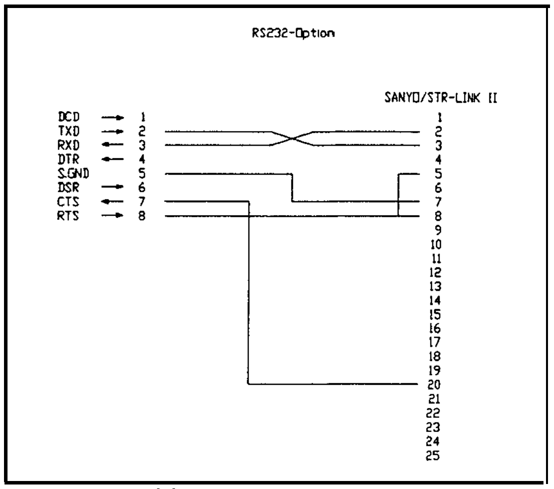
la. Rewind tape, press READ TAPE.

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

2. Enter LOAD on your terminal.

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

NOTE :a. = STR-LINK II instructions;
b. = Sanyo instructions.



TR-Link Cable

Appendix D Data Logging with Magnetic Tape Interface (Cassette Recorder)

The following is an example program, to show how data can be saved to and loaded from a cassette recorder. The OPEN and CLOSE statements operate the same way as in SP798-41 (old communication module for PC700/900). For ports other than PORT **B,** you can use the PORT X S and PORT X E statements instead. In that case, the switching of RTS control line must be provided by the user program (see appendix B for PEEK and POKES).

SAVING DATA ON TAPE

```
 01 DIM X(100)                      define array
 09 REM subroutine to write data to tape
 10 PORT B 1200,N,8,1               set output to port B
 20 SUSPEND 1000                    delay to create a trail
 40 FOR C=l TO 100                  100 array elements to tape
 50 PRINT X(C)
 60 SUSPEND 10                      delay between elements
 80 NEXT C
 90 PORT A                          set output to port A
100 RETURN
```

LOADING DATA FROM TAPE

```
199  REM subroutine to read array from tape
200  PORT B 1200,N,8,1
205  OPEN                           set input from port B
210  FOR C=l TO 100                 read 100 elements
220  INPUT X(C)
230  NEXT C
240  CLOSE                          input from port A again
250  RETURN
```

Appendix E Programming Example

The new functions (introduced with BASIC V1.4) BREAD, BWRIT and SYNC largely supercede the PLC read/write code shown. This code may be useful, however, if additional PLC opcodes are required. BREAD (Opcode 05) and BWRIT (07) don't require any additional code in order to read and write PLC memory.

```
00005 REM ALLOCATE MEMORY FOR ARRAYS
00010 DIM R(1 I),T(I 1)
00020 TASK 2000
00030 TASK 3000
00050 GOT0 1000
00100 REM SUBROUTINE CLEAR SCREEN
00110 FOR LI=1 TO 54:PRINT:NEXT LI:RETURN
01000 REM TASKS 1 : ASK FOR HR AND DATA AND WRITES TO PC VIA I/O BUS
01010 LH=30:REM CURSOR HOME CODE
01015 GOSUB 100
01020 PRINT #LH;TAB(20);"NCMZ 1799 DEMONSTRATION PROGRAM'
01030 PRINT
01040 PRINT 'This program demonstrates some of the possibilities of the"
01050 PRINT "NCMZ 1799 communication module.'
01060 PRINT 'The following TASKS are programmed :"
01070 PRINT
01080 PRINT "TASK 1 : On this screen (PORTA) a HR number and data is"
01090 PRINT 'asked and send to the PC via the I/O bus.'
01100 PRINT "TASK 2 : On the terminal at PORT B values of HR's are "
01110 PRINT 'displayed from an array which is filled at TASK 3.'
01120 PRINT 'TASK 3 : Read's HR values via PORT C and the 6 byte protocol'
01130 PRINT 'and writes the values to array R
01200 GOSUB 1530
01210 PRINT TAB(10);"ENTER HR NUMBER <10-20> ";:INPUT LR
01220 IF LR>9 IF LR<21 GOT0 1240
01230 GOT0 1260
01240 PRINT TAB(1 0);"HR(";LR;") = ";
01245 PRHEX HR(LR);:INPUT "    new value : ",D
01250 HR(LR) =D
01260 GOSUB 1500
01270 GOT0 1210
01500 REM SUBROUTINE CLEAR LINE 18-22
01510 PRINT #LH;:FOR LI=1 TO 18:PRINT :NEXT LI
01520 FOR LI=1 TO 5:PRINT TAB(70):NEXT LI
01530 REM SET CURSOR AT LINE 18
01540 PRINT #LH;:FOR LI=1 TO 18:PRINT :NEXT LI
01550 RETURN
02000 REM TASK 2
02010 PORT B 9600,N,8,1
02020 LH=30:REM HOME CODE
02030 GOSUB 100
02050 PRINT #LH;
```

```
02060 PRINT TAB(35);"TASK 2
02070 PRINT
02080 PRINT TAB(10);"This TASK displays the values of HR10 to HR20 continuously'
02090 PRINT TAB(10);"from array R which is filled by TASK 3.'
02100 PRINT #LH;
021 10 FOR LI=1 TO 7:PRINT:NEXT LI
02120 FOR LI=1 TO 11
02130 PRINT TAB(5);"HR(";LI+9;") = ";:PRHEX R(LI);" "
02140 NEXT LI
02150 GOT0 2100
03000 REM TASK3
03020 REM 'This program communicates with a programmable controller "
03030 REM "(PC1 100) via PORT C ."
03070 REM 'Connect the PC to PORT C.'
03130 REM INITIALIZE PORT C AND SWITCH THE CTS CONTROL OF
03135 REM SYNCHRONIZE THE COMMUNICATION WITH THE PC
03140 F=3:PORT C 9600,O,8,1:POKE($C600,$07):GOSUB 3600: F=O
03210 REM initialize string to send
03220 Z$(0)=6: Z$(1)=0: Z$(3)=0: Z$(4)=0: Z$(5)=0
03250 FOR LI=1 TO 11:REM READ REGISTERS
03255 SUSPEND 1:REM wait 20 mSec.
03260 Z$(2)=LI+8:REM set address
03350 REM COMPUTE THE CHECKSUM
03360 Z$(6) =Z$(2)
03400 REM SEND THE 6 BYTES TO THE PC AND WAIT 120 mSEC
03410 PRINT Z$;:SUSPEND 6
03440 REM IF NOT 6 BYTE ANSWER THEN SYNCHRONIZE AGAIN
03450 IF LOC<>6 GOSUB 3600:GOT0 3570
03480 REM GET ANSWER FROM PC
03490 FOR LC=1 TO 6:GET T(LC):NEXT LC
03500 REM CHECK THE CHECKSUM
03510 IF T(6) < > ((T(1) +T(2) +T(3) +T(4) +T(5)).$FF) THEN GOT0 3570
03550 POKE($27F,1):REM SET UNSIGNED CALCULATIONS
03560 R(LI)=256*T(5)+T(4)
03570 NEXT LI
03580 GOT0 3250
03600 REM SUBROUTINE SYNCHRONIZE
03605 LF=F: F=3
03610 PRINT #0;:SUSPEND 6:IF LOC=O GOT0 3610
03615 F=LF
03620 GET C$,LOC
03630 RETURN
```

Appendix F Ladder Program for HR Access

NCMZ 1799 - PC1 100 COMMUNICATION PROGRAM - BAS1100.LDR

```
!CR0200                                       CR0200!
!-] [--+--------------------------------------( )-!
!      !                                           !
!CR0200!                                           !
!-]\[--+                                           !
!                                                  !
!                                                  !
!                                                  !
```

Dummy Coil. Always on

```
!BP0016                                            !
!IR0002                       +------------+MB0201!
!-] [--+----------------------!            !--( )-!
!      !                      !SOURCE      !      !
!BP0015!                      !IR0002      !      !
!IR0002!                      !            !      !
!-] [--+                      !DESTINATION !      !
!                             !HR0258      !      !
!                             !            !      !
!                             !XFER CODE   !      !
!                             ! 0001       !      !
!                             !            !      !
!                             +------------+      !
!                                                  !
!                                                  !
!                                                  !
```

ReadtheControl/Pointer register (IR0002). Mask off the upper 8 bits. The resulting value isthepointerto atableof registers.

```
!CR0200                       +------------+GE0202!
!-] [-------------------------!            !--( )-!
!                             !OPERAND 1   !      !
!                             !IR0002      !      !
!                             !            !      !
!                             !OPERAND 2   !      !
!                             ! 0001       !      !
!                             !            !      !
!                             +------------+      !
!                                                  !
!                                                  !
!BP0015                                            !
! IR0002                      +------------+MV0203!
!-] [--+ ----   --------------!            !--( )-!
!      !                      !SOURCE      !      !
!BP0016!                      ! IR0002     !      !
!IR0002!                      !            !      !
!-] [--+                      !DESTINATION !      !
!      !                      !OR0002      !      !
!GE0202!                      !            !      !
!-]\[--+                      +------------+      !
!                                                  !
```

Check if the Control/Pointer Register(IR0002) is not equal to zero.

If WRITE bit (BP16), READ bit (BP15) or pointer is not zero, echo the Command/Pointer register back to the NCMZ module.

NCMZ 1799 - PC1 100 COMMUNICATION PROGRAM - BAS1 1 OO.LDR

```
!                                                        !
!BP0016                                                  !
!IR0002                          +------------+IM0204!
!-] [----------------------------!            !--( )-!
!                                !TABLE LENGTH!      !
!                                ! 0256       !      !
!                                !            !      !
!                                !TABLE END   !      !
!                                !HR0256      !      !
!                                !            !      !
!                                !POINTER     !      !
!                                !HR0258      !      !
!                                !            !      !
!                                !SOURCE      !      !
!                                !IR0001      !      !
!                                !            !      !
!                                +------------+      !
!                                                    !
!                                                    !
!                                                    !
!                                                    !
!CR0200                          +------------+TR0205!
!-]\[----------------------------!            !--( )-!
!                                !TABLE LENGTH!      !
!CR0200                          ! 0256       !      !
!-] [----------------------------!            !      !
!                                !TABLE END   !      !
!BP0016                          !HR0256      !      !
!IR0002                          !            !      !
!-] [--+-------------------------!POINTER     !      !
!      !                         !HR0258      !      !
!BP0015!                         !            !      !
!IR0002!                         !DESTINATION !      !
!-] [--+                         !OR0001      !      !
!                                +------------+      !
!                                                    !
```

If the WRITE bit (BP16) is on, transfer the Data Register (IR0001) to the HR table in the PLC.

If the READ bit (BP15) or the WRITE bit (BP16) is one, copy the data pointed to by HR0258 into OR0001. This data is then read by the NCMZ *module.*

End of BAS1 1 OO.LDR

NCMZ 1799 - PC1200 COMMUNICATION PROGRAM - BAS1200.LDR

```
!CR0200                                          CR0200!
!-] [--+-----------------------------------------( )-!
!      !                                             !
!CR0200!                                             !
!-]\[--+                                             !
                                                     !
!                                                    !
!                                                    !
```

Dumny Coil. Always on

```
!BP0016                                              !
!IR0001                      +------------+MB0201!
!-] [--+------------    ----------------!       !--( )-!
!      !                     !SOURCE      !          !
!BP0015!                     !IR0001      !          !
!IR0001!                     !            !          !
!-] [--+                     !DESTINATION !          !
!                            !HR0258      !          !
!                            !            !          !
!                            !XFER CODE   !          !
!                            ! 0001       !          !
!                            !            !          !
!                            +------------+          !
!                                                    !
!                                                    !
!                                                    !
```

Read the Control/Pointer register (IROOOI). Mask off the upper 8 bits. The resulting value isthepointertoatable of registers.

```
!CR0200                      +------------+GE0202!
!-] [ ------------    ----------------!       !--( )-!
!                            !OPERAND 1   !          !
!                            ! IROOOI     !          !
!                            !            !          !
!                            !OPERAND 2   !          !
!                            ! 0001       !          !
!                            !            !          !
!                            +------------+          !
!                                                    !
!                                                    !
```

Check if the Control/Pointer Register(IR0001) is not equal to zero.

```
!BP0015                                              !
!IR0001                      +------------+MV0203!
!-] [--+ ----------------------------!       !--( )-!
!      !                     !SOURCE      !          !
!BP0016!                     !IR0001      !          !
!IR0001!                     !            !          !
!-] [--+                     !DESTINATION !          !
!      !                     !OR0001      !          !
!GE0202!                     !            !          !
!-]\[--+                     +------------+          !
!                                                    !
```

If WRITE bit (BP16), READ bit (BP15) or pointer is not zero, echo the Command/Pointer register back to the NCMZ module.

NCMZ 1799 - PC1200 COMMUNICATION PROGRAM - BAS1200.LDR

```
!                                                     !
!BP0016                                               !
!IR0001                            +------------+IM0204!
!-] [------------------------------!            !--( )-!
!                                  !TABLE LENGTH!      !
!                                  ! 0256       !      !
!                                  !            !      !
!                                  !TABLE END   !      !
!                                  !HR0256      !      !
!                                  !            !      !
!                                  !POINTER     !      !
!                                  !HR0258      !      !
!                                  !            !      !
!                                  !SOURCE      !      !
!                                  ! IR0002     !      !
!                                  !            !      !
!                                  +------------+      !
!                                                     !
!                                                     !
!                                                     !
!                                                     !
```

If the WRITE bit (BP16) is on, transfer the Data Register (IR0002) to the HR table in the PLC.

```
!CR0200                            +------------+TR0205!
!-]\[------------------------------!            !--( )-!
!                                  !TABLE LENGTH!      !
!CR0200                            ! 0256       !      !
!-] [------------------------------!            !      !
!                                  !TABLE END   !      !
!BP0016                            !HR0256      !      !
!IR0001                            !            !      !
!-] [--+---------------------------!POINTER     !      !
!      !                           !HR0258      !      !
!BP0015!                           !            !      !
!IR0001!                           !DESTINATION !      !
!-] [--+                           !OR0002      !      !
!                                  +------------+      !
!                                                     !
```

If the READ bit (BP15) or the WRITE bit (BP16) is one, copy the data pointed to by HR0258 into OR0002. This data is then read by the NCMZ module.

End of BAS1200.LDR

Be sure and include in your BASIC program:

```
xxxx    POKE ($F6,1) :REM enable PC1200 HR access from BASIC
```