Appendix G Serial Port Pinouts

1799 Loader Cable (25 pin COM1)

NLC-11CM MALE

1799 male			IBM PC (DB-25) ~~female~~	
RxD	3	2	TxD	Use this cable to load programs from your IBM PC to the 1799 if your PC has a 25 pin serial port.
TxD	2	3	RxD	
DSR	4			
Gnd	5	7	Gnd	
DTR	6	6	DSR	Notice that the 1799 transmits data on pins 2, 6 and 8. The 1799 receives data on pins 3 and 7. Pin 4 and 9 are not connected to anything inside the 1799.
		20	DTR	
CTS	7	4	RTS	
RTS	8	5	CTS	

1799 Loader Cable (9 pin COM1)

NLC-10CM

1799 MALE			IBM PC (DB-9)	
RxD	3	3	TxD	Use this cable to load programs from your IBM PC to the 1799 if your PC has a 9 pin serial port.
TxD	2	2	RxD	
DSR	4			
		5	Gnd	
DTR	6	6	DSR	
Gnd	5	4	DTR	
CTS	7	7	RTS	
RTS	8	8	CTS	

1799 to Modem Cable

NLC-15CM

1799 male			Modem (DB-25) male	
RxD	3	3	TxD	Use this cable to connect a modem to the 1799. The modem is assumed to have a 25 pin port.
TxD	2	2	RxD	
DSR	4			
		7	Gnd	
DTR	6	6	DSR	This cable scheme can be used with full duplex dial up phone line modems, or half duplex leased line modems.
Gnd	5	20	DTR	
CTS	7	5	CTS	
RTS	8	4	RTS	

1799 to 799 25 pin port

NLC-19CM

1799 male			NCMZ 799 (DB-25) male	
RxD	3	3	TxD	Use this cable to connect between a 9 pin 1799 port and a 25 pin 799 port.
TxD	2	2	RxD	
DSR	4			
Gnd	5	7	Gnd	
DTR	6	6	DSR OUT	
		20	DTR IN	
CTS	7	4	RTS IN	
RTS	8	5	CTS OUT	

1799 to 1799 9 pin to 9 pin cable

NLC-18CM

1799 (DB-9)			1799 (DB-9)	
RxD	3	2	TxD	Use this cable to connect two 1799 9 pin ports to each other. This cable is also used to connect a 9 pin 1799 port to a 9 pin 799 port.
TxD	2	3	RxD	
DSR	4			
Gnd	5	5	Gnd	
DTR	6	6	DTR	
CTS	7	7	CTS	
RTS	8	8	RTS	

1799 to Printer Cable NLC-21PT

1799 male			Printer (DB-25) male	
RxD	3	2	TxD	**Use this cable to connect a modem to the 1799. The modem is assumed to have a 25 pin port.**
TxD	2	3	RxD	
DSR	4			
Gnd	5	7	Gnd	
DTR	6	6	DSR	**This cable scheme can be used with full duplex dial up phone line modems, or half duplex leased Line modems.**
		20	DTR (busy)	
CTS	7	5	CTS	

Some printers use Pin 11 as the busy lead.

Depending on the application, other cables are possible. Refer to Section 2.3, page 15 for information on these other cables.

Appendix H Real Time Clock

To interface with the real time clock, four additional Basic statements are available:

DATE	Prints the date in the format: day MM-DD-YY (day: MON, TUE,...)
TIME	Prints the time in the format: HH:MM:SS
SETDATE	Allows the user to adjust the date. The current date is printed and a new date is asked. If no input is given, date will remain unchanged.
SETTIME	Allows the user to adjust the time. The current time is printed and a new time is asked. If no input is given, time will remain unchanged.
TIM$	string variable containing the time in the format HH:MM:SS.
DAT$	a string variable containing the date in the format MM-DD-YY.

Both string variables can be used in expressions like:

```
10 PRINT TIM$
20 A$ = TIM$+" "+DAT$
30 B$ = MID$(DAT$,4,2)
```

The values of the real time clock can also be accessed by the PEEK and POKE statements. To prevent reading data in transition, some precautions must be taken. Setting the write bit 'locks" the data in the clock registers, thus preventing reading of data in transition. Setting this bit high only affects this buffer and does not disturb the actual clock. Do not accidentally set the KS (POKE $7FB,$80) bit. This will result in excessive current draw and will shorten battery life.

- set bit 6 of address $7F8 : POKE($7F8,$40)
- read value of clock : PEEK($7F9) through PEEK($7FF)
- clear bit 6 of address $7F8 : POKE($7F8,0)

Sequence for writing data to the clock is:

- set bit 7 of address $7F8 : POKE($7F8,$80)
- write value to clock : POKE($7F9,x) through POKE($7FF,xX)
- clear bit 7 of address $7F8 : POKE($7F8,0)

The next table shows the addresses of the RTC and their contents. Note that all data is in Binary Coded Decimal format. All bits marked 0 may not be altered. (If bit marked 0 is set, the clock will not run).

	D7	D6	D5	D4	D3	D2	D1	DO	
$7FF	-	-	-	-	-	-	-	-	YEAR 00-99
$7FE	0	0	0	-	-	-	-	-	MONTH 01-12
$7FD	0	0	-	-	-	-	-	-	DATE 0131
$7FC	0	0	0	0	-	-	-	-	DAY 01-07
$7FB	KS	0	-	-	-	-	-	-	HOUR 00-23
$7FA	0	-	-	-	-	-	-	-	MINUTES 00-59
$7F9	ST	-	-	-	-	-	-	-	SECONDS 00-59
$7F8	W	R	0	0	0	0	0	0	CONTROL R/W

If the clock is accidentally stopped. Clear all bits in each of these 8 bytes to zero. Then use the SETTIME and SETDATE commands to reset the clock. If this still doesn't work, follow these steps:

1. Set the W bit to 1. Set the ST bit to 0. Set the KS bit to 1.
2. Reset the W bit to 0. Wait 2 seconds. Set the W bit to 1. Reset the KS bit to 0.
3. Use SETTIME and SETDATE to set time. Verify that the W bit is cleared to 0.

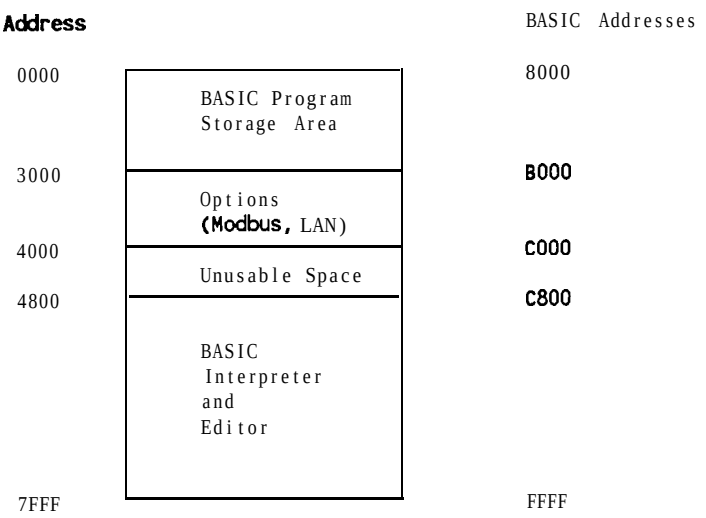
Appendix I PROM Modifications

1.0 PROM storage in the NCMZ1799

The NCMZ-1799 module uses one 27C256 EPROM. This PROM serves two purposes:

- Store the user program
- Store the Executive Firmware

Storing the user program in PROM provides the advantage of "hardcoding' an application into the module, In effect, this converts the general purpose BASIC module into an application specific module.



All addresses in the PROM are offset (as referenced by a BASIC program) by 8000h. Note that the 1799 address decode logic routes any request for addresses C000 through C7FF to the AMD2130 Dual Port RAM[8] rather than the EPROM, therefore these addresses are not available for use in the EPROM. They can be used to store data (project number, special configuration data) that can only be read by a PROM burner.

If an option package is installed, the full 16K (0000-3FFF) is not available for BASIC program storage, In these cases, 12K (0000-2FFF) is available.

[8] The Dual Port RAM is used to store the most recent I/O update. A separate DMA-like circuit eavesdrops on the I/O bus, copying all I/O points to memory. Likewise, when data is placed into Dual Port RAM by the CPU, this information is written to the I/O bus as the module's address is presented on the I/Cl address lines AO-A7.

1.2 Saving Programs to PROM

Once a program has tested satisfactorily, it may be desirable to burn this program into EPROM. The BASIC program in RAM can be downloaded to a PROM burner. The memory is transmitted from the 1799 module's port B to a serial device (PROM burner, terminal). The serial data follows the Motorola Exorciser (S record, block size $10000, offset 0, start address 0) format.

The maximum size of the BASIC program is 16K (no options) or 12K (with options; Modbus, LAN, port share). Note that you may write a BASIC program that uses up to 26K of RAM. This means that it is possible to write a program that is larger than what can be accommodated in EPROM.

The SIZE statement is helpful in determining the size of the program. However, the SIZE statement shows the total amount of memory used by your program plus memory used by dynamic array variables.

Since an array variable is created at runtime, space does not need to be set aside for this in a PROM, To determine the amount of memory used by a BASIC program without dynamic array memory, follow these steps:

1. Save your BASIC program to disk
2. NEW out the 1799
3. Reload your program from disk
4. From the "#" prompt, type SIZE. Do not *type* RUN *first.*

The following example shows a typical program, plus the intermediate steps required to convert this to a PROM.

BASIC Program

```
00060 DIM A(100),B(100)
00100 PORT D 9600,O,8,2
00110 PORT A 9600,O,8,2
00120 PORT C 9600,N,8,1
00150 EXEC $8000
00200 SUSPEND 100
00206 PORT D
00210 EXEC $BEOO :REM Claim serial port
00220 BREAD 1,64,A(1),E
00230 IF E=O THEN LEDON:SUSPEND 5O:LEDOFF
00240 EXEC $BF00:REM Release serial port
00250 EXEC $BE00
00260 BWRIT 1,64,B(1),E
00270 MEC $BFOO
00280 GOT0 210
```

To convert this program to PROM, connect a cable from Port B of the 1799 module to either a PROM burner or computer. If you are using a computer, load up a terminal program (such as UPDWLOAD.EXE) so you can save the data sent from the 1799 module. Configure both ends of the serial connection to the proper baud rate. Port B of the 1799 defaults to:

9600 baud, 8 data bits, no parity, 1 stop bit

Type from the READY prompt:

```
READY
#DWLOAD          (underlined text is entered by user)
```

The following data will be sent out of port B to the PROM burner (computer).

Result of DWLOAD command (captured in PROM burner)

```
S00B00004E45574241532020F4
S12100001100120100321090C20412831303029 2C42283130302900006 40F8E204420C4
S121001E393630302C4F2C382C3200006E0F8E20412039363030 2C4F2C382C3200001C
S121003C780F8E204320393630302C4E2C382C3100009608962024423030300000C8EE
S121005A06A7203130300000CD048E20440000D21D9620244245303020 3AA120436CE9
S121007861696D2073657269616C20706F72740000DC0EAE20312C36342C412831293C
S12100962C450000E6188120453D30205448454E204C45444F4E3AA72035303A920073
S12100B400F01E962024424630303AA12052656C656173652073657269616C20706FFF
S12100D272740000FA089620244245303000010 40EAF20312C36342C422831292C4559
S11800F000010E089620244246303000011806822032313000CA
S9030000FC
```

After loading the data into the PROM burner, you may wish to examine the individual bytes. If your PROM burner[9] can show a full page of PROM data, your display may look something like this:

Convert Motorola S record to binary (EPROM Programmer display)

```
0000 11 00 12 01 00 32 10 9C 20 41 28 31 30 30 29 2C  .....2.. A(100),
0010 42 28 31 30 30 29 00 00 64 OF 8E 20 44 20 39 36  B(100)..d.. D 96
0020 30 30 2C 4F 2C 38 2C 32 00 00 6E OF 8E 20 41 20  00,0,8,2..n.. A
0030 39 36 30 30 2C 4F 2C 38 2C 32  00 00 78 OF 8E 2 0  9600,0,8,2..x..
0040 43 20 39 36 30 30 2C 4E 2C 38 2C 31 00 00 96 08  C 9600,N,8,1....
0050 96 20 24 42 30 30 30 00 00 C8 06 A7 20 31 30 30  . $8000. . . . . 100
0060 00 00 CD 04 8E 20 44 00 00 D2 1D 96 20 24 42 45  .....D.....$BE
0070 30 30 20 3A Al 20 43 6C 61 69 6D 20 73 65 72 69  00 : . Claim seri
008061 6C 20 70 6F 72 74 00 00 DC OEAE2031 2C 36  alport..... 1,6
0090 34 2C 41 28 31 29 2C 45 00 00 E6 18 81 20 45 3D  4,A(1),E.. . . . E=
OOAO 30 20 54 48 45 4E 20 4C 45 44 4F 4E 3A A7 20 35  0 THEN LEDON:. 5
00B0 30 3A 92 00 00 F0 1E 96 20 24 42463030 3AAl O:..... $BFOO:.
OOCO 20 52 65 6C 65 61 73 65 20 73 65 72 69 61 6C 20  Release serial
OODO 70 6F 72 74 00 00 FA 08 96 20 24 42 45 30 30 00  port. . . . . $BE00.
OOEO 01 04 OE AF 20 31 2C 36 34 2C 42 28 31 29 2C 45  .... 1,64,B(1),E
OOFO 00 01 OE 08 96 20 24 42 46 30 30 00 01 18 06 82  .....$BF00.....
0100 20 32 31 30 00 00 00 00 00 00 00 00 00 00 00 00  210.............
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

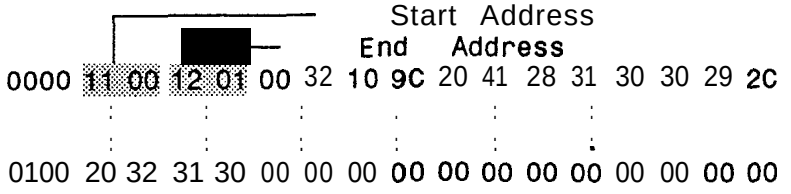
Merge this program code into the 27C256 EPROM, starting at offset 0000.

---

[9] Needham Electronics IEPROM Programmer is a half size PC XT card with an external Zero Insertion Force (ZIF) socket mounted at the end of a three foot ribbon cable. Hardware and software retail (1989 prices) for under $150. Call 916-924-8037 for more information.

Multiple programs can be stored in this PROM. These programs are accessed using the PROGRAM [n] statement. Refer to the NCMZ-1799 programming manual for more information on using the PROGRAM statement.

If additional programs were stored in this PROM, they must reside in contiguous memory locations. The program statement is able to find the subsequent programs by looking at the first four (4) bytes of each program, starting with the first. By noting where each program ends, the module can calculates where the next program would start. If the value found at the memory location just following the previous program block is zero, then the module assumes that no program is stored there.



The program starts at 0000h and ends at 0103h, for a total of 103h (259 decimal) bytes. Remember that the 1799 module RAM starts at $1100, not 0000, therefore the actual start and end addresses are $1100 and $1201 ($1203-2 bytes that are the start location... these are not necessary for operation and are therefore not loaded into RAM).

You may have noticed that some of the ASCII characters stored in the PROM were readable, but the BASIC keywords (PRINT, EXEC, etc.) were not? Why not?

The reason is, BASIC replaces each keyword with a single byte. This byte is called a token, There is a cross reference between each keyword and a token.

Each BASIC line ends with a $00 (null).

You can create a cross reference between each of the keywords and a token by creating a simple (nonsense) program that has a different keyword on each line. When this program is DWLOAD'ed, you will be able to determine what token the interpreter uses for that keyword.

> Note
>
> The tokens used for a particular version (V1.5, etc.) of the interpreter are always the same, However, certain keywords do have different tokens in other versions.

1.3 Special Locations in EPROM

An EPROM can be modified, thus customizing the module to a particular task.

Disable Port A Break Detect

V1.5	$7471 = $27	->	$12	(VALID FOR 799)
	$7472 = $04	->	$12	

Warning: During program testing it may be desirable to break out of the executing code and modify or display certain variables. If <BREAK> detection is disabled, there will be no way to do this, This feature can also be used to 'lock out' the ability to either view or make changes to the program.

Select RTS Low after Reset

Clearing the corresponding bit will cause the associated Request to Send (RTS) lead to be low whenever the card stops (watchdog trip, END statement encountered, error in program, BREAK detected on Port A [if enabled]).

v1.5	$7312 = $03	- bit 0 = Port B RTS - bit 1 = Port A RTS
	$7316 = $01	- bit 0 = Port C RTS - bit 1 = Port D RTS

Port Sharina Timeout

This value sets the timeout value on the port sharing software. The default value is 200 mS. This value can be increased to support longer scan time times. For example, when searching through a long program, a PLC will require more than one scan to return an answer, If the search requires more than 200 mS, the port sharing software will timeout.

Modbus 3.3	$3AE6 = $OA	= 20 mS timer ticks
LAN 2.3	$3AEA = $0A	= 20 mS timer ticks (VALID FOR 799)

BREAD / BWRIT Timeout oeriod

The BREAD and BWRIT functions will wait a certain amount of time for a response from the PLC. This time period is stored in the PROM as a 16 bit number representing the number of 20 mS timer ticks to wait. The default timeout period is $0046 (1.40 seconds). You can increase this timeout period (to, for example, 6 seconds) by modifying the following two locations.

VALID FOR 799

V1.5	$7D6F = $00	->	$01
	$7D70 = $46	->	$2C ($012C = 6 seconds) ($01F4 = 10 seconds) (12FE - 15 SECONDS)

1.4 PROM checksums

PROM cleared to "FF"

	RTS High Port	BASIC V1.5 Only	Modbus 3.3C	Modbus 3.3D
Break Allowed	A,B,C	8EA8	20E5	3992
	A	8EA6	20E3	3990
Break Disabled	A,B,C	8EA1	20DE	3988
	A	8E9F	20DC	3989

PROM cleared to "00"

	RTS High Port	BASIC 1.5 Options	Modbus 3.3C	Modbus 3.3D
Break Allowed	A,B,C	D6A8	58E5	7192
	A	D6A6	58E3	7190
Break Disabled	A,B,C	D6A1	58DE	718B
	A	D69F	58DC	7189

NOTE

When upgrading a module to a later firmware version, pay particular attention if you are also burning your program on the same EPROM. Always follow this step by step procedure:

1. Save your program to disk
2. Install the new firmware, but DO NOT burn the program onto the same PROM yet.
3. NEW out the old program from the 1799 RAM.
4. Load the disk file back into the 1799.
5. DWLOAD the program to the PROM burner.
6. Merge into the PROM burner the balance of the EPROM code (BASIC, Modbus, etc.)
7. Burn the finished PROM.

It is extremely important that the file DWLOAD'ed to the PROM burner was NOT the old program still in RAM, but rather a 'fresh' copy loaded from a disk file. This must be done because the old code in RAM will be 'retokenized' by the new interpreter firmware. It is likely that some of the old statements will have changed to new tokens and mean something different under this new interpreter.

Appendix J Troubleshooting

1. Status 1 and 2 LED's are flashing back and forth

The module watchdog timer has tripped. Push reset. If the program has an AUTO ON statement, it may restart. However, if the program had been corrupted, then reloading may be necessary. Verify that the proper PROM is installed. Verify that the user program is calling the correct $EXEC address. Verify that the module ground strap on the bottom mounting screw is securely mounted. Verify that the I/O rack is grounded. Verify that the devices connected to the W-232 ports are isolated.

2. Terminal won't talk to Module on Port A

Verify that the computer terminal COM port is selected correctly. NC799COM.EXE only operates on the IBM PC COM1 port. Verify that the cable and software operates by noting if the combination will communicate with another similar module. If the RTS lead of PORT A was forced low (due to a PROM modification), you may have to force the CTS line (pin 7 of the DB-9 connector on the front of the module) high. Do this by disconnecting any wire that may be connected to pin 7 and instead jumper pin 7 up to pin 6. Pin 6 is always in a 'high" condition. Check the external power supply (5.0 VDC to 5.1 VDC).

The terminal will not be able to communicate with the module if the module has disabled its serial ports, either through a programming statement or through a system error. To recover, repeatedly press the Reset Pushbutton while press the <BREAK> (CTRL-C using NC799COM.EXE) key sequence on your terminal. It is important to isolate if the module RS-232 circuitry is operational or if the module is simply ignoring data being received from the serial port.

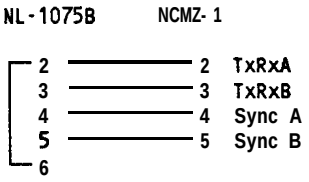
3. Status LED 2 (bottom) flashing slowly

Slow flash indicates that a communications error has occurred between the 1799 module and the PC1xxx PLC. Verify that HR0005 contains a value between 1 and 255. Verify that the bit rate and data format for the PLC port match those of the 1799 module. If using a Configure Port (CP) in the PC1200, verify that the CP coil is energized and that the Port is either 1 or 2. (If using Version "D" of the Modbus firmware) Verify that the RS-485 cable is secure and properly wired. Verify that either power is applied to terminal 11 and 13 or that terminal 11 is jumpered to terminal 17 and that terminal 13 is jumpered to terminal 19. Connect a breakout box to any of the RS-232 channels. Verify that DSR and DCD illuminate. Verify that the PClxxx RS-485 port has all of its termination resistors closed. Verify that the 1799 module has all of its termination resistors open, except the middle jumper on each bank (TxRx bank plus Sync bank).

If Version "D" of the Modbus firmware (RS-485 communications with host PLC), verify that:

PC1 200 Verify that the 16th bii of the CP configuration register is set high. This causes the PC1200 to ignore the state of the CTS line.

PC1 100 Verify that the CP function is programmed with PORT 2 specified. This must be done even if the default baud and communication format of 9600, odd, 8 and 2 is used. The reason this must be done is that the CP function will override the CTS detection on the B port. Otherwise, you will need to jumper CTS (Pin 5 of the Port B RS-232 connector) to Pin 4 (RTS).



4. Status LED 2 (bottom) flashing very slowly

Very slow flash indicates a communications error on the Modbus network. This is only an advisory message since a subsequent valid message will be answered, in turn illuminating the LED steady. If the module does not respond to polling and LED 2 continues to flash very slowly (once every two seconds), then confirm that the network is indeed noise free and that messages are arriving uncorrupted. Verify that the bit rate and data format (especially parity) match.

5. One or more of the ports on the module don't work.

Warning: The communication ports, while optically isolated from the rest of the module (and PLC) they are not isolated from each other. Confirm that the devices plugged into the front of the module have isolated serial ports. If not verify that all devices are connected to the same source and polarity (L1/L2/Ground) of power. If these guidelines are not followed, the module output circuitry can be damaged. The chips are, however, socketed. Replace as follows:

RS-232 ports A,B and C	Driver Chip - MC145406P
RS-485	Driver Chip - 75176

6. Module 'hangs' while communicating over the backplane

Verify that the handshake program (BAS1100.LDR or BAS1200.LDR) is loaded. Verify that the POKE($F6,x) function is properly programmed. Verify that the DIP switches are set properly, If the communications over the backplane is intermittent, then all of the preceding are correct. Communications will not even occur without each of the preceding items being correct. Verify that no other module in the I/O rack is addressed to the same address (IR's and OR's) that the 1799 module is using. Verify that PC1200 users have a revision "SW1926-1" board (silk screened on PC board).

7. Port sharing software causes 'REFRAMING' errors.

Verify that the external power supply voltage is 5 VDC. If using BREAD or BWRIT statements over the same serial port to the PLC that the port sharing software is using, then be sure that the CLAIM (EXEC $BEOO) and RELEASE (EXEC $BFOO) statements are being used. Verify that the BREAD and BWRIT commands are directed to the proper serial port via the PORT command.