AN NSC800 DEVELOPMENT SYSTEM

by

DWIGHT WALLACE GORDON

B. A., Susquehanna University, 1981

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:

Major Professor

Table of Contents

* Format: ADC bits/Multiply bits/weight and constant bits.

List of Tables

List of Figures

List of Tables and Figures

INTRODUCTION

After the introduction of the Widrow LMS algorithm[13] by B. Widrow in 1975 researchers in signal processing searched for a microprocessor with the high speed necessary to implement such an algorithm in real time. Also, it was desired that the microprocessor have a sufficiently low power consumption that it could be run from a battery for extended periods of time. The NSC800, developed by National Semiconductor, with its P^2CMOS technology provided a possible end to this search.

Working under this assumption, and with the encouragement of SANDIA National Laboratories (Albuquerque, New Mexico), Mac Cody, then a Master's student at Kansas State University, developed a dedicated system for this implementation[1]. (A block diagram of this system can be found in Figure 1.) The hardware was based on the NSC800 and other P^2CMOS support devices available at that time. Where P^2CMOS technology was unavailable, CMOS was substituted until the necessary technology came to support such devices*. The board incorporated a twelve bit CMOS analog to digital converter (ADC), the National Semiconductor AD1210HD. Completing the analog interface was an eight bit digital to analog converter, the Analog Devices AD7524.

The Widrow Adaptive LMS algorithm[1,2,12,13] requires that a number of multiplies be performed in each stage of the iteration loop. Although it was one of the most powerful eight bit instruction sets available at that time, the Z80 instruction set

* Those components still unavailable in P^2CMOS are still CMOS.

NSC800 Board Hardware Block Diagram

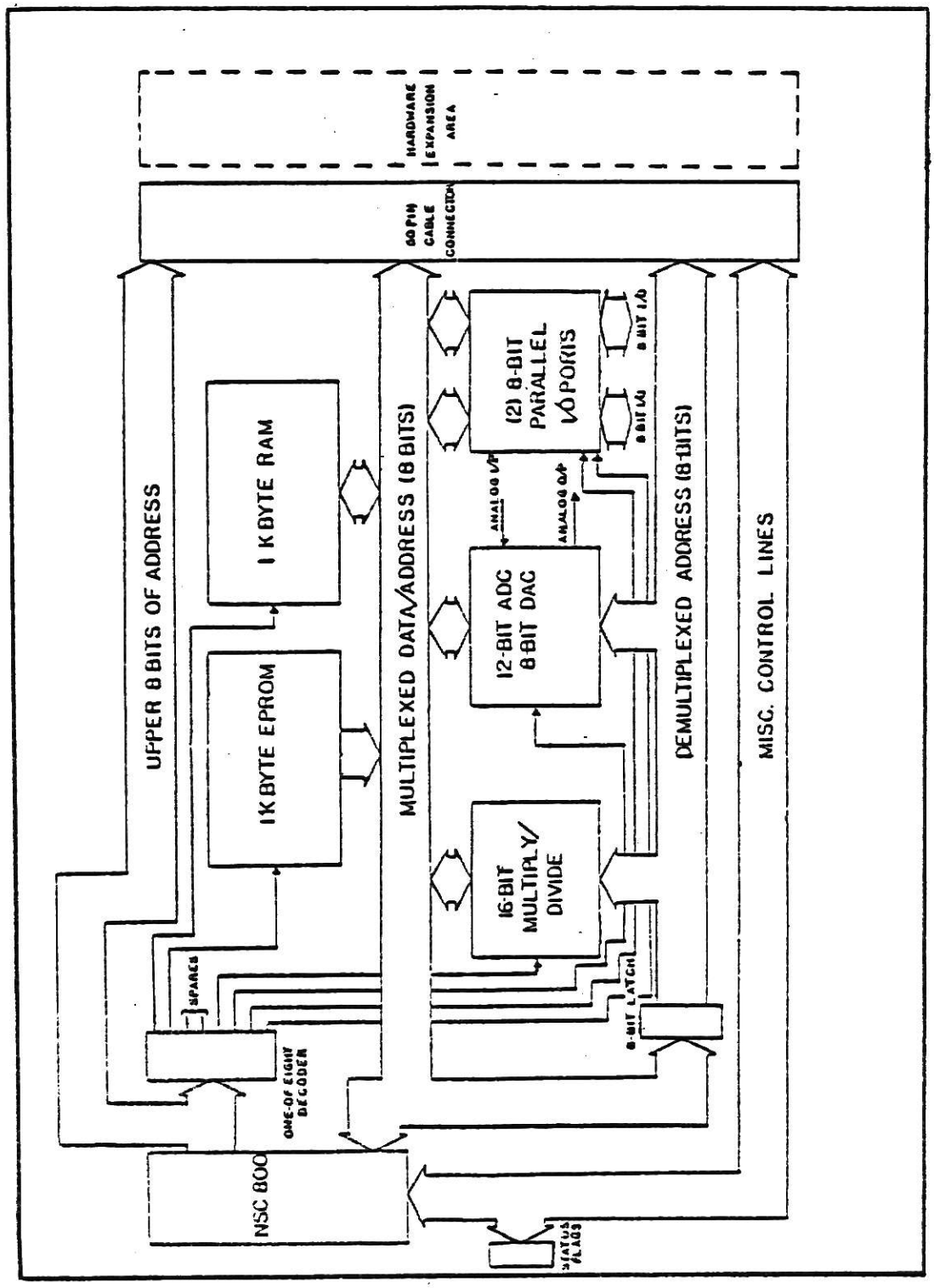


Figure 1.

employed by the NSC800, did not provide for a hardware multiply. A software multiply consumed too much processor time and slowed the algorithm down intolerably[1]. It had been decided that a pair of RCA CDP1855 multiply/divide units (MDUs) be included on the board to make up for this deficiency.

The software developed by Cody was a slight alteration of that developed by Donovan Nickel[2] in 1979. Modifications were based upon the system dependent locations of RAM and EPROM characteristic of every microcomputer installation. Also, Cody incorporated the use of the MDUs in his application. Due to a lack of time on the part of Cody, the software was never optimized beyond that done by Nickel. Cody also implemented corresponding versions of an adaptive lattice structure[1,2].

The completed system, after much time in the debugging stage, successfully performed its adaptive task[1]. Each program was placed manually in separate EPROMs and tested on the board. Execution was initialized by a power-on reset included on the board (in the case of a "lost" program, a separate reset switch was provided). Testing of the programs on the NSC800 board itself was a go/nogo procedure. Either the program worked or it didn't. If it didn't, the programmer had to modify the assembly listing, recode to object code (making all of the necessary addressing modifications), and manually reprogram another EPROM. Turnaround time was long (order of magnitude: hours).

It was this author's original intention to provide timing documentation for the NSC800 microprocessor itself. In order to

simplify this task it was decided that a number of short program loops needed to be constructed in order to isolate a repetitive occurrence of a given signal. One method of achieving this goal was that of Cody: manually programming each short program into separate EPROMs. This author found this to be unacceptable in terms of future expansion. It was decided that a monitor for the NSC800 be developed in order to interface it to the ADM-3A, thereby allowing user-direct modification of memory and starting of processor execution in said memory. Such a monitor and interface was constructed. In the process, an interface between the NorthStar Horizon and the PC/M Model 660 CMOS EPROM Programmer was designed so that an object listing obtained from a ZASM* or ASM* assembly of a Z80 or 8080 source listing could be directly programmed without hand input of the object code (see Appendix A). This was utilized to program the NSC800 board monitor EPROMs.

After working with the Tektronix 7603 Oscilloscope with the 7D01 Logic Analyzer and further work with the newly published specifications for the NSC800[11] it was decided that this author's timing specifications were unacceptable. The diagnostic equipment performed erratically. Also, the author had not taken into account the tristate nature of most of the system bus. On a two state analyzer such as the 7D01 it required a significant amount of additional circuitry in order to do an accurate timing analysis. However, work continued interfacing the NSC800 to the NorthStar Horizon.

The NorthStar Horizon did not employ the secondary channels

* ZASM and ASM are CP/M assemblers available at Kansas State Univ.

utilized by the ADM-3A and the NSC800 monitor. After much work with handshaking and file I/O on the CP/M operating system of the NorthStar Horizon, the interface program was completed. This program allowed the user to operate the NSC800 board monitor through the ports of the NorthStar Horizon. Upon detection of special commands the NorthStar Horizon would initiate an object code transfer from the NorthStar Horizon to the memory of the NSC800 board, thereby achieving downloading from the assemblers available on the NorthStar system.

In order to simplify some of the debugging tasks involved in the use of the NSC800 board, and to acquaint an unfamiliar user with the NSC800 monitor functions, an emulation of the NSC800 monitor was installed on the NorthStar Horizon (see Appendix E). This monitor allows the user to perform many of the same tasks allowed on the NSC800 board with the exception of the use of the ADC, DAC, and MDUs. Download capability was achieved through the LINKing* and LOADing* the monitor program and the user program under the auspices of the CP/M operating system.

Finally, all of the above were utilized to implement a version of the Widrow LMS algorithm on the NSC800 board (see Appendix D for program listings). The emulator was used to help debug the software on the NorthStar Horizon. Once this software was debugged, a version of the program was downloaded to the NSC800 board and tested and debugged there. This second set of testing and debugging was of the ADC, DAC, MDUs, and location dependent addresses.

* LINK and LOAD are file utility programs available at K. State.

ADM-3A <=> NSC800 BOARD INTERFACE

Motivation

A well-designed monitor allows dynamic program input and testing. In a dedicated system all work is done and programmed in EPROM. With a monitor all of this work was done in RAM.

Originally the author planned to do timing measurements of the NSC800 itself. This task was simplified by writing a short program loop. For example:

```
0400  D3 00     AGAIN: OUT 00H
0402  C3 00 04         JP  AGAIN
```

might have been used to test the I/O strobe width. To program this on a system with a monitor is trivial. A system without requires the erasing and programming of an EPROM.

A monitor allowed the author to perform rather complicated tasks with a single command. In the monitor implemented, the author could set a single breakpoint, execute the program until it reached this point, observe the machine status/registers at this point, and continue execution at the next sequential instruction (either with or without setting another breakpoint). To implement this type of provision on a user program was impractical.

The serial communication provided by the NSC800 monitor was more or less standard. It, with some minor modifications, can be adapted to interface any other relatively "dumb" device to the NSC800 board. This was partially demonstrated through the NSC800 <=> NorthStar Horizon interface presented later.

Design Criteria

Once it had been decided to interface the NSC800 board with the ADM-3A, design criteria were established. The communication, due to hardware limitations of the ADM-3A, is asynchronous serial. Even though the ADM-3A could use either current loop or RS232C, it was decided that the communication link would be RS232C. Since the majority of asynchronous serial devices to which the NSC800 might be connected are RS232C, this decision allows the NSC800 to communicate with more devices with fewer modifications than a similar current loop configuration allows.

The rate of data transfer with the ADM-3A was chosen to be the fastest that the equipment allowed. This is limited by the UART employed and by the transmission clock frequency. Since hard copy of the monitor might be required, and the only printer available to the author at that time was a Decwriter II (with a maximum communication rate of 300 Baud), it was decided that the UART be programmed such that it might have multiple, but fixed, rates of transfer. These included the fastest standard that the ADM-3A and NSC800 would allow, and the 300 Baud required of the Decwriter II.

In order to evaluate the performance of the NSC800 in its signal processing application, it was required that the processor run at a number of fixed clock rates (ie. 1.0, 2.0, 2.5, and, when it is available, 4.0 megahertz). This caused a problem with timing. None of these frequencies divided down to standard baud

rates with any ease. If they had, no easy way could be found to obtain the baud rate frequencies from all of the aforementioned operating frequencies. This made the use of a hardware UART without a separate baud rate clock (independent of the system clock) difficult. In addition to this problem, at the time that this monitor was designed, no timing specifications were available for the NSC800 cpu. It was impossible to predict the access and timing requirements necessary to interface a hardware UART onto the system bus.

For these reasons the author chose to write a software UART, using one of the available pre-existing input and output ports. The cpu read some bits of the input port to determine the system clock rate and communication rate from a number of user-accessable switches or jumpers. The data was sent or received by "twiddling bits" on the input and output lines. This software UART would keep the board power consumption to a tolerable level because a minimum number of extra IC packages needed to be installed onto the existing system in order to support such a UART.

All control signals originating or terminating at the communication port were designated as active low. An inverter package was included on the interface board in order to invert the necessary lines and mate them with the ADM-3A standard. {Author's note: Insufficient time was available to modify existing software in order to eliminate this hardware overhead.} This convention allowed the author to change wires on the interface board in order to force the control signals to be compatible with those of the

ADM-3A. This was seen as a better alternative than reprogramming the monitor EPROM if a control level was in error. The documentation in the ADM-3A manual[10] did not state whether a control signal was active high or low. It said that the signal went "active"!

The monitor needed to be installed on existing EPROMs. This allowed only 1024 bytes of memory for the program (two 6653 EPROMs). Early versions of the monitor allowed only minimal commands and nearly filled this space. As the system was further refined, the program was compacted and additional functions were added. (As it presently stands in version 1.91, the monitor still has over 48 unused bytes at the top of the EPROM for expansion!)

One severe design limitation involved in the code space compaction was that response speed suffered. Because most of the use of the monitor would be with a relatively slow human operator, response speed was not seen to be a critical design factor. This response speed was a big difficulty when the relatively fast NorthStar Horizon was interfaced to the NSC800 with a not-completely responsive handshake. First attempts at communication caused loss of data if the NSC800 were not operating at a sufficiently high system clock frequency and data transmission rate. In order to combat this problem the NorthStar Horizon interface program had delay loops added in order to make it seem slower, with respect to response time, than the NSC800 board (see Appendix C). Also, communication links with the NorthStar Horizon were kept at 2400 baud (the fastest available from the NSC800

UART). This kept NSC800 processor time (on a data transmit or receive) to a minimum.

Monitor Functions

The method of writing the program was carefully planned. The first and most important task was that of serial communication. The input and output routines (RECEIV and XMITER respectively) were written and carefully tested. In fact, XMITER was the first routine written. A small loop which repeatedly transmitted a single ASCII character was written to utilize XMITER until the routine was completely debugged. The debugging process relied heavily on the 7D01 Logic Analyzer.

Coding the communication routines as subroutines allowed the rest of the program to become rather modular with respect to data transmission and reception. If a routine needed a single ASCII character as data, it need only have executed "CALL RECEIV" and the character was returned in accumulator A. Similarly, if a routine needed to transmit a single ASCII character, it need only have loaded the character into accumulator A and executed a "CALL XMITER." It was later discovered that character transmission occurred in most routines. A single byte subroutine call (RST 28H) was reserved for the XMITER routine for code compaction purposes.

Functions were added to the monitor in order to make it useful for the user. These included viewing and modifying a single byte of memory (MEMCHG), starting program execution at a

user-specified location (GOPROG), viewing the machine status/registers (REGLOK), copying a block of data from a source location to a destination location for a given number of bytes (COPY), viewing a block of memory from a given starting address to a given ending address (MEMVUE), setting a single breakpoint in order that program execution be interrupted and sent to a predetermined location in the monitor where the machine status was saved and control returned to the keyboard (BRKPT), and restarting program execution on the next instruction after an interrupted program had reached a breakpoint (RESTAR). These commands are more fully explaned in Appendix B: NSC800 Monitor Version 1.91 User's Manual.

Hardware Description

The hardware portion of the interface consists of a small interface board (see Figure 2). On the board is a 16-pin dip socket into which a 16-pin jumper is plugged in order to connect the interface board to a similar 16-pin dip socket on the NSC800 cpu board. An inverter package is provided as per the previous explanation. A pair of RS232C <=> CMOS interface chips and the supporting voltage supply network mates the two data representation standards (ie. CMOS and RS232).

A DB25 connector provides the link with the ADM-3A. This is not a standard three, five or seven wire RS232C connection. This configuration was chosen in order to efficiently utilize the secondary channels available on the ADM-3A. If a device that does

not have these secondary channels is interfaced directly to the NSC800 interface board, operation is not guaranteed! Also, care was taken to choose the appropriate connector cables between the NSC800 and the ADM-3A. More information may be found in Appendix B.

Hardware Block Diagram

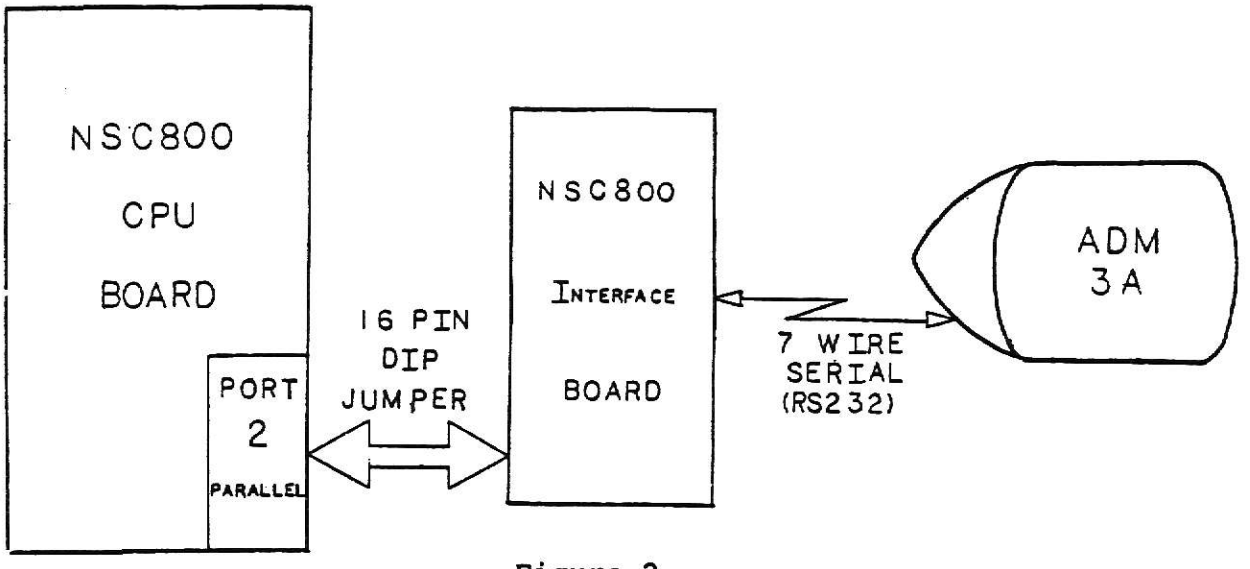


Figure 2.

Software Description

The primary function of the monitor software is that of bookkeeping. It initializes necessary registers and communicates a minimal set of prompts with the operator in order to inform him of the machine status. Commands issued by the operator are interpreted by the monitor and the appropriate action taken. (Such actions are clearly described in Appendix B.)

Initialization Routine

This section of code [0000-0006, 00E5-00F0] initializes the stack pointer to usable RAM, enables incoming interrupts, forces the communication interface lines to the safe state, holds them there long enough for things to settle, and transmits the wake-up banner message (see BANNER in the program listing). Note that the stack pointer is not initialized to the top of RAM. This area between STACK and RAMEND-1 is reserved for monitor functions and should not be used by the uninformed user. The delay inserted after the initialization of the communication lines is provided for the benefit of the ADM-3A. Without this delay the first few characters transmitted by the monitor upon wake-up are garbage. This is due to the fact that the ADM-3A (on a system reset) doesn't know where the first start bit of the first character is. Upon successful completion of the above tasks the monitor transmits the banner message to the ADM-3A in confirmation of the fact that the monitor and communication link (at least in the transmit mode) works.

Register Bookkeeping

The code contained in this section [00F1-0104] provides for the tidy maintenance of the machine status, and breakpoint address and code. This section of code is executed each time a pseudo software-interrupt ('FF') is encountered in program execution. This command may be due to a breakpoint entered by the user, or from a lost program's attempt to execute empty address space.

Since the code 'FF' is nothing more than a fast subroutine call, the program counter has been pushed onto the stack. The monitor assumes that the execution of the 'FF' was due to a user-supplied breakpoint. The address pushed onto the stack is that of the next sequential instruction. The breakpoint code, 'FF,' had been substituted for the program opcode. It is necessary to decrement the pushed program counter [00F1-00F3] so that a return will be executed to the correct location, and to remove the old breakpoint code (ie. 'FF'), replace it with the old opcode, and remove the pointer [0102-0104].

The remaining portion of code in this section [00F4-0101] saves the machine status onto the stack. All registers with the exception of the 'R' and 'I' registers are pushed onto the stack. This allows a terminated program's registers to be examined with or without alteration. In fact, the program is such that it can be restarted with the exact register configuration as was present prior to its interruption.

Command Input and Execution: Command Input Mode.

A properly executing monitor now displays a cursor [0105-0108], '>', at this time as a signal that the monitor is ready to accept a command instruction [0109-010B]. The monitor then tries to match the entered command to one of those present in the command table [010C-0121]. If a match is made with any but the last command, the address of the desired routine is obtained and program execution begins at that point [011B-011F]. If the input

command is the last command in the table (the carriage return), the command input process repeats itself [0120-0121]. If no match is found, an error message ('???') is displayed and the ADM-3A bell sounds [0122-0127]. The process then continues with a command input.

Description of Individual Routines

MEMCHG allows the user to view a byte of memory and, if so desired, change its contents. The routine accepts a four+ hex digit address* followed by a carriage return [0128-012A]. If for any reason the address supplied is invalid then the routine is terminated, an error message displayed ('???'), the bell sounded, and command returned to the command input routine [012B-012D].

If the supplied address is a valid address, then the address is displayed (followed by a '>'), and the present contents of said address [0131-0142]. The routine then accepts data [0143-0147]. Any key on the keyboard is accepted, but invalid hex digits cause a termination of the routine in an error state (ie. error termination) [0153-0155]. The exceptions to this are the line feed (this means 'keep the present contents the same and advance onto the next sequential memory location') [014F-0152, 015E-0163], the 'S' (this means 'stop the routine and continue with the command input') [014A-014E], or the carriage return preceded by two valid hex digits [0148-0149, 0156-015D or 0156-0163]. This last condition causes the routine to try to write this data to the supplied memory address [0159]. If the write is successful then

* Only the last four digits are retained as the address.

the routine continues at the next sequential memory address. If not (such as a write to EPROM), the routine error terminates.

GOPROG [0164-017F] causes a four^{+} hex digit address to be input. Program execution begins at the specified address. All registers which had been saved are recovered prior to the start of program execution at the specified address. The address is terminated with a carriage return. An invalid starting address causes the routine to error terminate.

REGLOK displays the saved machine status. All registers which had been previously pushed onto the stack are displayed. Registers are displayed in groups of four nibbles at a time. Each pass through the loop [01CA-01E0] causes one of these four-nibble values to be displayed. The stack pointer is obtained in the first statement of the routine [01BD-01C0] and is displayed in the first time around the loop [01CA-01E0]. The value displayed is the current location of the stack pointer. It points to the last filled system stack location.

COPY implements a block copy. The four hex nibble source address is input as the "From? " address [01E1-01E7]. The four hex nibble destination address is input as the "To? " address [01EE-01F4]. The four hex digit length of the string is input as the "Length? " [01FB-0201]. If any of the above are invalid the routine error terminates [01E8-01E9 or 01F5-01F6 or 0202-0203, 020F-0213]. Each operand is stored temporarily on the stack [01ED, 01FA]. The short trailing routine [020F-0213] cleans-up the stack if an error termination occurs.

RESTAR performs the same function as GOPROG except the address at which the program starts execution is implicitly known on the stack. This permits this routine to implement a RET as a jump to the program. This routine is primarily used to restart a program which has been interrupted by a breakpoint.

MEMVUE allows the user to view a block of memory. The starting address of the block is input as zero or more hex nibbles [0223-022B]. This is the only routine that assumes an address of less than four hex nibbles is valid. The program segment assumes leading zeros. The starting address is terminated with a comma [022C-022D]. Any other terminal is invalid, and, along with an input of an invalid hex character, causes an error termination of the routine [022E-0230].

The routine continues with the input of four^{+} hex digits representing the ending address of the view block [0234-0236]. This address terminates with a carriage return. All other invalid hex entries cause the routine to abort with an error condition [0237-0239].

Output is formatted such that the starting address of a line appears as four hex digits on the right of the screen [023F-0247]. These are followed by a space [0248-024A]. The data bytes are arranged as two hex nibbles and are separated by a space [0248-0264]. The first line of data continues until an address with an 'F' as the least significant nibble is output [025E-0262]. The program then outputs a carriage return/line feed and begins a new line (starting with the starting address of that line) [0261-0262,

023F-0247]. This process continues until the last data byte is output, at which time the routine returns to the command input section [0255-025D].

BRKPT allows the user to remove any pending breakpoints [0265-0267] and enter a new breakpoint [0268-027D]. The latter is done by inputting a four^{+} valid hex digit as the breakpoint address [0268-026A]. This terminates with a carriage return. (Any other non-hex digit input causes the routine to terminate in an error state [026B-026D].) This address is stored in a specific location [BREAKA:=07F8-07F9]. The routine fetches the opcode at the breakpoint address [0271] and stores it at a specific location [CODE:=07F7]. A pseudo software interrupt, 'FF', is inserted at the breakpoint address [0278-027A]. Command returns to the command input section [027B-027D].

Subroutine Descriptions

{Author's note:

Each subroutine is documented in the assembly listing as to its register expectations. If specific information with regard to registers utilized and disturbed is required, consult the assembly listing for more information.

Each subroutine, unlike the in-line code, may be freely used by the programmer in order to ease the programming responsibility. This is especially true with regard to input and output. It is not necessary for the user to rewrite the I/O routines.}

OUTPUT transmits the block of ASCII data starting from the

address specified by the HL register until bit seven of the data is found set. The data byte which has bit seven set is sent prior to its being tested. This routine is used throughout the monitor to display messages to the user. (It may be invoked through the execution of a RST 30H ('F7') command.)

REMOVE removes a pending breakpoint by restoring the old opcode to its original location [0287-028D] and forcing the breakpoint pointer (BREAKA) to point to page zero [028E-028F]. All of page zero in this monitor is occupied by EPROM. A write to EPROM, that which occurs when the RESET point is entered, does not affect the system.

OUT4, OUT4A, and OUT2 are complicated routines to output to the terminal either two (OUT2) or four (OUT4 and OUT4A) nibbles from either the HL register pair (OUT4), the memory locations FIX and FIX+1 (OUT4A), or memory location FIX+1 (OUT2). The routine incorportates a hex to ASCII conversion [029C-02A8]. This routine is useful for displaying a register, register pair, or four nibble data input via the HILOW4 subroutine or any other routine which leaves the display data in memory locations FIX and FIX+1.

HILOW4 and HILOW2 allow the user to input either four or two hex nibbles and store them in FIX and FIX+1, or FIX+1. If HILOW2 is used, the E register contains four minus the number of characters desired as input. In the case of its traditional utilization, this value is two. HILOW4 does the initializing for the user [02AE-02AF].

The routines request a character [02B1-02B3]. This character

is tested to see if it is the terminal carriage return [02B4-02B7]. If a match is made, the routine outputs a carriage return and line feed, and checks to see if enough characters were input [02B6-02B7, 02D4-02DC]. If an insufficient number of characters is input or an invalid input occurs then the routine exits with the carry flag set [02DB-02DC, 02DE-02DF].

In order to achieve an ASCII to hex conversion, a table lookup is performed [02B8-02C4]. If no match is made, the routine error terminates [02C0-02C2, 02C5-02C6, 02DE-02DF]. When a match is found, the ASCII to hex conversion is performed [02C7-02C8], and the nibble is rotated into the buffer specified by FIX and FIX+1 [02C9-02D1]. Execution continues with a character input [02D2-02D3, 02B1].

LOOKUP performs a table lookup to determine the timing constant for data transmission [02E1-02F0]. It also initializes a counter for the purpose of the UART timing loops [02F4-02F6]. The speed bits are wired to bits one thru four (LSB:=0) of the second parallel port. This port is read [02E1-02E3] and the data bits are masked [02E4]. The effective address of the valid table location is calculated through the addition of the offset provided by the speed bits from PORT2 and the start of the lookup table provided by CONST [02E6-02EC]. The effective address is used to load the table value into the HL register and the memory locations SCRTCH and SCRTCH+1 [02ED-02F3]. Finally, the routine initializes the DE register to a value used in the UART I/O routines [04F4-04F6].

XMITER outputs a single seven bit ASCII character located in accumulator A upon entry. This routine causes extensive damage to any data located in the alternate register set. This and RECEIV are the only monitor routines that do so.

The NSC800 first makes a request to send [02FA-02FE]. It waits until the ADM-3A responds with a clear to send [02FF-0305]. The NSC800 obtains the timing constants required for the data transmission [0306-0308]. Data, starting with a start bit, is serially output on D0 [030F-0323]. Three stop bits are output at the end of the character [0324-0341]. The subroutine leaves the communication bits in the *safe*, 'FF,' state [0342-0343]. (For more information see Figure B-4.)

RECEIV is responsible for the handshaking and reception necessary to obtain data. The ADM-3A makes a request to send to the NSC800 [0348-034C]. If this is not done, the monitor program gets hung-up in an infinite loop.

The monitor prepares the necessary constants for later use [034F-0357]. The first delay utilized is one half a bit length [0353-0354]. All other delays are one bit length.

The NSC800 tells the ADM-3A that it is ready to receive a data character and waits for the start bit [0358-0364]. This is another place where the program can get hung-up in an infinite loop if problems arise in the system. After the start bit is detected, the program delays one half a bit length and tests for the continuation of the start bit [0365-0371]. This allows the monitor to detect a false start on the data transmission line.

The data line is sampled at intervals of one bit length from the successful detection of the middle of the start bit [0374-0387]. Data is shifted into accumulator A. This routine contains a number of dummy statements which effect a time delay in order to mate the timing to that of XMITER.

Two stop bits are expected [0388-03A0]. Should either one or both of these stop bits be in error, the routine outputs the appropriate error message [03A6-03BF] (see Figure B-4).

Half duplex operation of the ADM-3A is expected. The concluding section of the routine echoes a received character to the ADM-3A for verification [03C3-03C6]. This is a desirable result in that a nonresponsive keyboard on the ADM-3A can be most often diagnosed as a lost program on the NSC800 board.

UART Timing Parameters

An integral part of the software UART is the timing parameters stored in memory. Much care has been taken to provide for the most accurate balance of sample times in transmission (XMITER) and reception (RECEIV) of serial data. The maximum data transmission and reception rates are limited by the uncertainty in the input test loops in the XMITER and RECIEV routines, and by the maximum serial error permitted by either the ADM-3A or the NSC800. The latter error has usually been defined as one half bit length over a data word. When the former error is greater than or equal to the latter error the software UART fails. Another error occurs in the timing delay loop allowed in the software. This loop can

only be adjused in groups of 21 T cycles. At fast rates of transfer this is insufficient precision to allow a reliable transfer.

Within the above restrictions the formula for the determination of the timing table constants is:

$$T = 65539.619 - F / (21 * B)$$

where T = Decimal number in the table (+/- 1),

B = Baud rate of data transfer (bits / second),

and F = System clock frequency (hertz).

{The above equation may be used to alter the system clock rate and still maintain data transfer. The user must place the hex value of this constant in the appropriate location of the lookup table which starts at location CONST. It may also be desirable to place this table somewhere in RAM so that dynamic alteration of its values may happened in real time. In this circumstance it is necessary to initialize the RAM table in the initialization section of the monitor so that the monitor always wakes up in some predetermined state.}

ADM-3A Configuration

The internal configuration of the ADM-3A must be as follows:

Cursor control	:	ON
Local	:	ON
103	:	OFF
202	:	OFF
Code	:	OFF (Secondary)
EXT	:	OFF
EOT	:	OFF

Table 1.

The external configuration of the ADM-3A must be as follows:

RS232		
HDX		
Bit 8	:	1
Parity	:	INH
Stop	:	2
Data	:	8

Table 2.

NSC800 Board <=> NorthStar Horizon Interface

After the monitor was written and debugged it was a natural progression to interface the NSC800 board to the NorthStar Horizon through the second serial port on the NorthStar. The NorthStar Horizon has the necessary mass storage to hold the lengthy assembly listings necessary for any serious program development on the NSC800 system. The CP/M operating system supports many useful programs for program development. These include the editors ED, EDIT and WordStar. Software debugging of 8080 programs implemented on the NSC800 system can be done through the Dynamic Debugging Tool (DDT) utility. The NorthStar assemblers, ZASM (a Z80 assembler) and ASM (an 8080 assembler), provide paper tape punch format object code that is easily downloaded to the NSC800 through this interface. Later an emulation of the NSC800 monitor was coded so that the NorthStar can be utilized in much the same fashion that the NSC800 monitor is (see Appendix E).

Hardware Description

Due to the careful design of the NSC800 monitor no additional integrated circuits are required to interface the NSC800 to the NorthStar Horizon. The only piece of special hardware is a specially designed RS232 connector. Since the 8251 UART employed in the NorthStar architecture does not support the secondary channels utilized in the NSC800 <=> ADM-3A interface, it is necessary to route these handshake lines to appropriate points on

the NorthStar connector. If the lines are left dangling (as was done on the first draft of the interface cable) inconsistant operation occurs.

The only other pieces of hardware required by the interface are jumper configurations internal to the NorthStar. These jumpers control the baud rate of data transfer, and the location of the handshake lines on the DB25 connector located on the back on the NorthStar Horizon. The latter configuration was specified by Dr. M.S.P. Lucas as being compatible with the Multiprogrammer Interface designed here at Kansas State. NorthStars are utilized by the Kansas State University Department of Electrical Engineering for various projects dealing with the said Multiprogrammer. At any time the author's NorthStar may be required for such a purpose. It is easier to keep that connector than to keep changing it when the need arises.

NSC800 <=> NorthStar Horizon Patch Cable

NSC800 pin number	NorthStar Horizon pin number
1	1
2	3
3	2
4	5
11	6
12	20

Table 3.

NorthStar Horizon Second Serial Header Configuration

1-11	2-16
3-15	4-13
5-12	6-NC
7-8	9-10
12-NC	

Table 4.

Software Description

NORTH.ASM is an 8080A-based program to interface the NSC800 board monitor (version 1.8+) to the second serial port of the NorthStar Horizon. The program allows the user to specify an input object file (type: xxxxxxxx.HEX or xxxxxxxx.OBJ) as an input file for downloading to the NSC800. Bidirectional communication is established between the terminal on the NorthStar and the NSC800 board.

All characters input from the keyboard are sent to the NSC800 except for certain special characters. The ESCape (^[) resets the beginning of file pointer and inputs the specified input file to the NSC800 via its monitor memory change ('M') command. CNTRL-C (^C) causes the interface program to terminate and control to return to the CP/M operating system. If the interface routine is in the midst of a download, this command is sometimes ignored.

This program was written in 8080 assembly code because of the difficulty with which file I/O is accomplished under CP/M. The code for obtaining a byte of code from a specified file is based

upon that used in the DUMP.ASM utility provided with CP/M. Rather than convert this code to Z80 mnemonics it was easier for the author to copy the code and add to it.

The first ten bytes of code [0100-0109] do the bookkeeping for the routine. The stack pointer is loaded into the HL register [0100-0103] and stored in a specific location for later use [0104-0106]. The last address stored on the system stack before it executes a user program is the return point to the operating system. The program initializes its stack pointer to spare RAM [0107-0109].

The system initialization continues. Both serial ports are reconfigured for program use [010A-0122]. After reconfiguring, the UART lines are allowed to settle so that devices on the lines obtain clean data transmission [0123-0125]. {This initialization will cause some problems with other programs which utilize the second serial port. It is suggested that the reset button on the NorthStar be engaged after each use of this program.}

An operational program is signalled by a sign-on message at this time [0126-012E]. This message provides documentation as to the purpose and commands available on this program. This is the only time that this is available to the user.

Under normal operating conditions the program requires that an input file be specified for downloading purposes. A check is made to see if the file open is performed without error [012F-0131]. If no error is detected then the program continues normally [0132-0136]. The file pointer is initialized and the

program enters the input polling routine [015A-0165].

Should an error occur, the operator has the option of continuing operation or terminating the program [0137-0159, 0166...]. This section provides an appropriate prompt for the user in order that he know that a file open error has occurred and that the program may or may not be terminated at this time. If the appropriate response to the question is received, the program terminates.

The input polling routine gives equal priority to both the terminal and the NSC800 [0166-0178]. The status of the UART for each device is checked to determine if the input buffer is full. A service routine for each device was written for each of these occurrences (Terminal: IN0, NSC800: IN1). Any characters sent by the NSC800 are delivered to the terminal without alteration [0190-019C]. Characters sent by the terminal are tested against certain control characters and if a match is made, the appropriate action is taken [0179-18F].

CINE tries 255 times to read a character from the NSC800 UART buffer [019D-01BA]. Between each try is a time delay [01A4-01A6]. This routine is designed so that if a transmission error occurs, the interface program will not hang-up in an infinite loop waiting for a transmission which will never occur. When data is received it is sent to the terminal [01AF-01BA].

XMIT is a general purpose output routine for data transfer. Data is placed in accumulator B and the number of the output device is placed in accumulator A. The routine determines if the

appropriate output buffer is empty [01BB-01C0] before outputting the data. Transmission of data to the terminal is as simple as loading the appropriate output buffer with the data [01C6-01C9]. Utilization of this type of routine affords the option of designing a complicated transmission routine to send data to the NSC800 [01CA-01E7]. The data is manually transferred to the NSC800 with handshaking by the routine. {Note: This is a place where the program might get hung-up in an infinite loop. Also, because of its nonresponsive nature, data may be lost here.}

DELAYS provides a delay for various routines through the use of nested delay loops [01E8-01F8]. This routine is used to make the NorthStar appear slower than the NSC800 with respect to data transfer. With respect to the not-completely responsive handshaking utilized in this interface, this is necessary.

OST checks the status of the specified output buffer to see if the buffer is full [01F9-020D]. This routine, like XMIT, utilizes accumulator A as an index to the required device. If the value of accumulator is one then the specified device is the NSC800. For any other value of accumulator A, the specified device is the terminal.

NUM determines the number of data bytes on a line of paper tape format file. The length in bytes is always the next two bytes immediately following a colon, ":", in the object file [020E-0215]. {Note: If the specified input file is not of type xxxxxxxx.HEX or xxxxxxxx.OBJ then this search will cause extreme difficulty with respect to data transfer and possible failure of

the NorthStar system.} This length is the ASCII-hex length with a value in the most significant nibble of zero to nine inclusive. This double byte ASCII value is converted into a valid hex length for later use [0216-022E].

Data transfer is achieved through the utilization of the NorthStar as a fast and accurate operator of the NSC800 monitor. The NorthStar emulates a user in the modify memory mode of operation. CMD provides this emulation. The format employed is mated to that of the NSC800 monitor, ie. M<starting address><cr> [0230-0256]. Two bytes used by the paper tape format but not by the author are also skipped in order to leave the file pointer at the first data byte of the paper tape format line [0259-025E]. The routine also provides for the reception of the lengthy NSC800 monitor response to the memory modify command [025F-0269].

OUTT outputs the number of bytes of data specified by the contents of accumulator B from the input file to the NSC800 monitor [026C-025B]. This routine is used to obtain the transfer of a record of data. No error checking is done by the routine. If an error occurs in the transfer and the NSC800 monitor detects same by outputting an error code, the routine has no method of error recovery. {Note: If the bell sounds for any reason other than an end of transfer, it is due to an error and the user should type a carriage return to terminate the data transfer.} The routine does, however, wait for and obtain the NSC800 response to each line of a command.

OUTPUT is used for cursor positioning. It outputs a line

feed and a carriage return to the terminal [02B6-02C4]. It is not used very much by later versions of the program. The routine was mostly used in early versions of the program when user prompts were few.

The main routines for file data transfer are XMFILE and DONEMG. XMFILE sets up the input file [02CE-02D0] and flags an error if necessary [02D1-02D5]. Termination of a data transfer (at an end of record) in the middle of a transfer is done by hitting any key on the terminal keyboard [02D6-02E7]. Transfer of records of data continues until a length of zero on a record is obtained by NUM [02E8-02F9].

FINIS is the routine to terminate the program and restore status to that expected by CP/M [02FA-0301]. The operating system's stack pointer is restored. Said stack pointer contains the address of the operating system as the last address pushed onto the stack.

ERR prints a string of ASCII characters to the terminal [02C5-02CD]. The appropriate CP/M macro is utilized. Termination of the ASCII string is done through the detection of a dollar sign, '$'.

GNB reads a byte of the specified input file. It is taken from the DUMP.ASM utility of the CP/M operating system. {The author does not profess to completely understand how this routine works. A minimal knowledge of its operation was obtained from the comments included in the code.} The routine tries a file read through the appropriate CP/M macro [0308-0312]. If the read

fails, the appropriate error message is displayed on the terminal and the routine exits with the carry flag set [0313-0328]. Data in CP/M is mapped from the disk space to an 80H buffer in real memory. The real memory address is obtained through a complex calculation involving the disk address [0329-0334]. Bit seven is always reset on a file read (this implies that the file must be ASCII data) [0335-0337].

SETUP [0339-0345] and DISKR [0346-0354] are two utility routines provided for file setup and reading respectively. Both utilize CP/M macros in order to achieved the desired result. They are first presented in the CP/M utility DUMP.ASM.

Design Limitations

Data transmission from the NSC800 to the NorthStar is a not-completely responsive transmission. Design of a completely responsive interface is beyond the scope of this work. In order to eliminate any potential problems, time delays were added to the NorthStar software in order to make it seem slower than the NSC800. Fast transmission rates to the terminal are employed in order that the XMIT routine to the terminal will not hang-up the NorthStar processor for a long period of time. This condition will cause an overrun on the second serial input port.

CP/M is a difficult operating system for file I/O. Documentation provided by the manufacturer[8] is somewhat confusing and presents few examples. For this reason no uploading is designed into the system. Downloading is achieved only through

the use of preexisting routines obtained from the DUMP.ASM utility. If a high level language had been available on the operating system, it would have made the file I/O much neater and have provided extended capabilities for the system.

Application Example: Widrow LMS Algorithm

The tapped-delay line (or transversal filter) as shown in Figure 3 is utilized for various signal processing applications including that of intruder detection. The filter's purpose is that of input signal decorrelation. The weights use previous values of the input signal to predict the current input signal via a linear approximation. The difference between the expected and the actual signal is the output error. A closed loop configuration is utilized where the output error is fed back to the processing stage for weight modification. This modification technique uses a least mean squares technique (LMS) for output error magnitude minimization.

In an intruder detection application the weights adapt to the statistics of the input signal and use these to linearly predict the output. The filter adapts to input noise of sufficiently low frequencies without great variations in error output. (Most noise sources in a real application are of this nature due to the low-pass filter effect of the Earth.) An intruder passing close to the input transducer produces a brief change in the statistics of the input signal. This brief change tends to cause a relatively large fluctuation in the output error signal. If this output signal is monitored (either in hardware by a threshold detector, or by a human operator) an intruder might be detected by large variations in the output, Q, after the filter has a chance to adjust to the input signal's statistics. (Figure 4 shows a flowchart of the utilized algorithm.)

A practical system for intruder detection is portable and has a relatively large input bandwidth. Portability means the system is able to be run with a battery. A CMOS microprocessor system fills this criterion but, up to a few years ago, CMOS technology was too slow to provide sufficient bandwidth to satisfy the second criterion. The P^2CMOS technology of the NSC800 is a viable candidate to satisfy both of the above criteria.

The board developed by Cody[1] was an attempt to satisfy the first criterion. It was this author's intention to utilize Cody's board to satisfy the second--high speed. With this intention in mind this author modified and time-compacted programs developed by Nickel[2] and Cody[1]. This author also performed tests to determine board power consumption as a function of system clock frequency while the system was running WIDROW16.DWG with a sine wave input modulated by a sawtooth wave (with a period of approximately one second). The range of the resulting system input waveform was between approximately 0.2 Hertz and 100. Hertz. Resulting data points exhibited only short-term stability (ie. the program would not necessarily run under the specified conditions for long periods of time without failing). (For more information see Appendix G.)

Wiener solution convergence of the LMS algorithm is dependent upon the input and data manipulation precision employed within the algorithm. Nowhere could a study be found that discussed the necessary precision for an application of this nature. Lower precision data manipulation yields larger filter bandwidth, but

implies slower Wiener convergence.

For these reasons two versions of the Cody and Nickel Widrow programs were coded and tested on the author's development system. The first employs eight bits of ADC input and eight by eight multiplies in all equations requiring a multiply. The second employs twelve bits of ADC input and sixteen by sixteen multiplies. Both retain sixteen bits for each generated error and weight. Timing characteristics of these two versions are presented in Tables 5 and 6 below.

One on the limiting factors in the time per iteration is the time required to do a signed multiply. In the original version of this algorithm[1,2] the multiplies were performed in software. Each software multiply (in a subroutine format) required approximately 147.25 microseconds at 4.0 MHz system clock [1: 50, Table 4.1]. The hardware version utilized by Cody (in a subroutine) required 70.5 microseconds [1: 51, Table 4.2]. This author ignored traditional methods of subroutine usage and forced the routine to in-line code. Such techniques allow the author to perform each eight by eight signed multiply on the NSC800 board in 41.75 microseconds (see Appendix F for program listings of said multiply algorithms and assumptions). {Author's note: All of the above time specifications are for eight by eight bit signed multiplies on the NSC800 board. Similar comparisons may be made for sixteen by sixteen bit signed multiplies.}

Software Timing Analysis

Program Name	WIDLNG8.DWG	WIDROW16.DWG
Precision [1]	8/8/16	12/16/16
Timings [2]		
Data Acquisition	113	139
Compute G(M)	4779	5723
Compute E(M)	134	148
Update B(M,K)	6671	6811
Block Move	1453	1453
Form/Output Q(M)	366	389
Total Time [2]	13516	14663

Table 5.

Notes:

[1] Format is input bits/multiply bits/retained data bits.

[2] Units of worst case clock ("T") cycles per iteration.

Execution Speeds *

System Clock	Min. Vcc	Program Name	
		WIDLNG8.DWG	WIDROW16.DWG
1.0 MHz.	5.0 Volts	73.99 sps	68.20 sps
2.0 MHz.	5.0 Volts	147.97 sps	136.40 sps
2.5 MHz.	5.0 Volts	184.97 sps	170.50 sps
2.7 MHz.	5.0 Volts	199.76 sps	184.14 sps
3.6 MHz.	6.0 Volts	266.35 sps	245.52 sps
4.1 MHz.	7.0 Volts	303.34 sps	279.62 sps

Table 6.

* Note: Execution Speeds can be formed by dividing the System Clock Frequency by the "T" cycles per iteration found in the last column of Table 5 above. Execution Speeds are based on a worst case set of conditions. As of 7-1-82 the 4.0 MHz. (at five volts) version of the NSC800 was unavailable to the author. National Semiconductor claims[11: 4-9, Figure 4-10] that the current version of the NSC800 will operate at 4.0 MHz and 10 volts supply. The samples obtained by the author would not operate on Cody's board above 7.0 volts Vcc. Certain samples would not operate above 6.0 Volts. This difference has lead the author to believe that the CPU was the cause of the failure. This is by no means conclusive evidence that the CPU is the cause. The available processors would run for limited periods of time at all speeds and voltages specified in the above table.

Transversal Filter

$$G(M) = \sum_{K=1}^{16} B(K,M) * F(M-K) \qquad E(M) = G(M) - F(M)$$

$$B(K,M+1) = \mu * B(K,M) + v * E(M) * F(M-K)$$

$$Q(M) = \left\{ \sum_{M=1}^{16} E(M-16) \right\}^2$$

Figure 3.

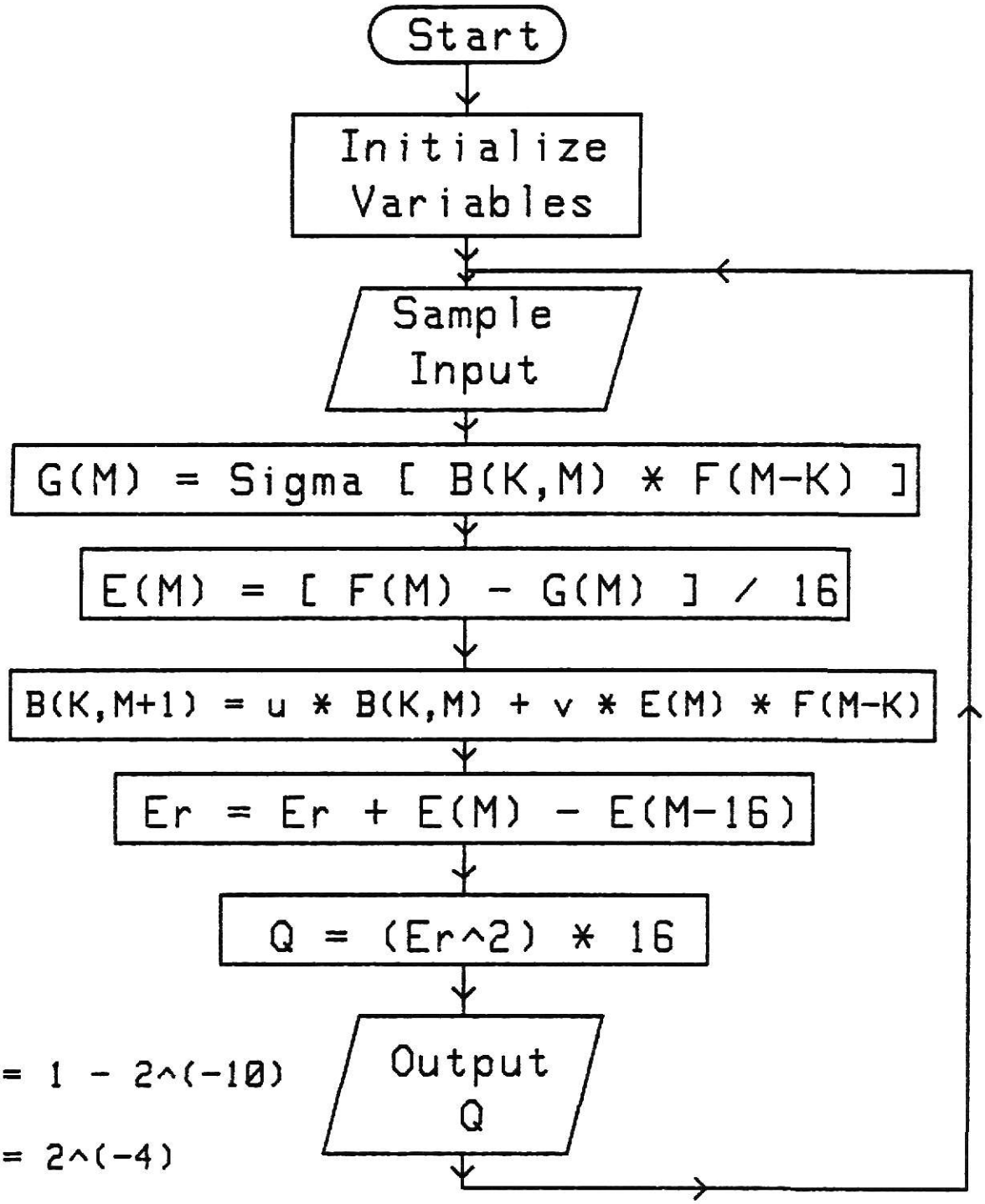

Start
Initialize Variables
Sample Input
G(M) = Sigma [B(K,M) * F(M-K)]
E(M) = [F(M) - G(M)] / 16
B(K,M+1) = u * B(K,M) + v * E(M) * F(M-K)
Er = Er + E(M) - E(M-16)
Q = (Er^2) * 16
Output Q
u = 1 - 2^(-10)
v = 2^(-4)
0 < K < 17

Figure 4.

Conclusions

In spite of the inconvenience of not having upload capabilities, the NSC800 development system dramatically reduced turnaround time when writing software for the user. Cody had only been able to obtain 194 sps at 4.0 MHz [1: Table 4.2] with an 8/8/16 format and a hardware multiply. This author was able to obtain 295.95 sps under the same operating conditions. This represents a 52.6% increase in filter bandwidth!

Writing this software on the NorthStar Horizon with its associated assemblers and editors encouraged the author to write assembly code with comments more so than hand assembly would. The ability to automatically program the memory of the NSC800 board not only eased the user's job, but increased reliability (especially with long programs).

The system allowed the author to dramatically reduce the cycle time required for the Widrow LMS algorithm. Because of the rapid turnaround time and the disk facilities available on the NorthStar, the author was encouraged to experiment with time compaction techniques that would normally have been overlooked due to programmer time limitations on the Cody[1] system.

In order to make the system more user-oriented it is suggested that the not-completely responsive handshake on the NSC800 <=> NorthStar data link be upgraded to a completely responsive link. This will allow the user to use a 300 baud terminal with the NorthStar. Since these are readily available at Kansas State University this would allow the NSC800 board to be a

good tool for student use (provided that the equipment could be mounted in a suitable container).

Acknowledgements

The author would like to thank SANDIA National Laboratories for their support of this project. A special word of thanks is extended to Richard Wayne without whose special talents the author would have been unable to obtain some of the integrated circuits utilized on the NSC800 system. The author would like to extend thanks to Dr. D.H. Lenhert, John Bartholomew, and Daniel Schowengerdt for their help with the "little" and the "not so little" problems that the author encountered during his work with the NSC800 system. Appreciation is extended to Dr. James H. Tracey and Dr. Kenneth Conrow for their work as committee members. Finally, the author would like to thank Dr. M.S.P. Lucas who served as the author's major advisor and provided a great deal of assistance.

References

1. M. Cody, An Evaluation of the NSC800 8-Bit Microprocessor for Digital Signal Processing Applications, A Master's Report, Kansas State Universsity, 1981.

2. D. Nickel, An Evaluation of Various Microprocessor Implementations of an Adaptive Digital Predictor for Intrusion Detection, A Master's Report, Kansas State University, 1979.

3. Texas Instruments, The TTL Data Book for Design Engineers, Second Edition, Dallas, Texas, 1976.

4. National Semiconductor, Interface Databook, Santa Clara, California, 1978.

5. RCA, CDP1800-Series IC Products, Somerville, New Jersey, 1982.

6. Pacific Cyber/Metrix, Inc., Operating Manual for Model 660 EPROM Programmer, Second Printing, San Ramon, California, Document 660-01, May 1978.

7. North Star, HORIZON Computer System Manual, Revision 1, Berkeley, California, 1978.

8. North Star, HORIZON System Software Manual Addendum, Revision 2.1, Berleley, California, July 1980.

9. Lifeboat Associates, Manual Z80 Development Package, New York, New York, 1980.

10. Lear Siegler, Inc., ADM-3A Interactive Display Terminal Operator's Handbook, 1975.

11. National Semiconductor, NSC800 Microprocessor Family Handbook, 1981.

12. Nasir Ahmed and R. J. Fogler, On an Adaptive Lattice Predictor and a Related Application, Kansas State University.

13. B. Widrow, et al., Stationary and Nonstationary Learning Characteristics of the Adaptive LMS Filter, Proc. IEEE, Volume 64, Page 1151, August 1976.

APPENDIX A: NorthStar Horizon <=> Model 660 CMOS EPROM Programmer Interface

The author desired easy programming of long programs located in object format on the NorthStar Horizon disk. Hand input of data was slow and subject to human error. Such data had to be separated into nibble format for input. Programming time was much longer than the author could allow.

Hardware Block Diagram

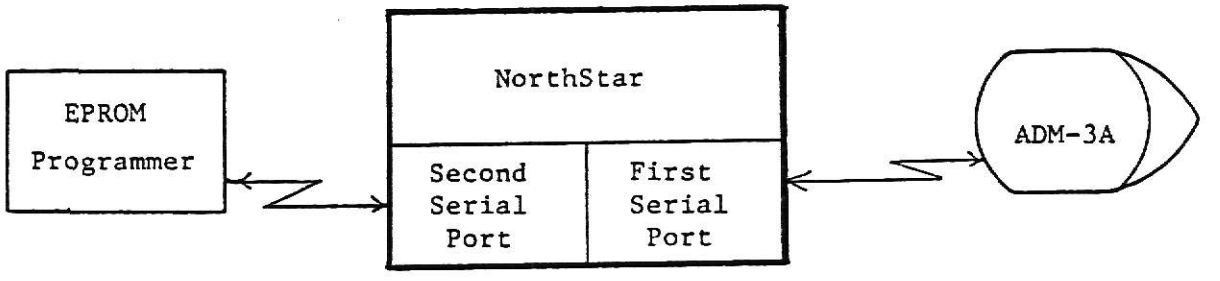


Figure A-1.

Hardware Configuration

Minimal hardware is required for this interface. The only additional piece of hardware is the connector cable from the second serial port of the NorthStar to the EPROM programmer. Internal jumpers of the NorthStar are specified, along with the switch settings on the bottom of the PC/M Model 660 CMOS EPROM Programmer.

DB25 Connector Configuration

NorthStar	Programmer
1	1
2	3
3	2
7	7

Table A-1.

Northstar Horizon Second Serial Header Configuration

1-11	2-16
3-15	4-13
5-12	6-NC
7-8	9-10
12-NC	

Table A-2.

PC/M Model 660 CMOS EPROM Programmer Switch Settings

S1:	ON (6653)
S2:	OFF (terminal)
S3:	OFF (master PROM)
S4:	Don't care
S5:	ON => 9600 Baud
S6:	ON => 2400 Baud*
S7:	ON => 300 Baud
S6 & S7:	ON => 110 Baud

Table A-3.

* Utilized by the test system.

Software Description

PROMP4.ASM, in terms of software, was designed from NORTH.ASM described in the text. Most of the software description can be taken from that of NORTH.ASM. Only the differences are highlighted in this appendix.

Unlike NORTH.ASM, PROMP4.ASM has two methods of loading the data buffer. A ^L (CNTRL-L) loads the specified file's lower nibbles into the E-P buffer. A ^H (CNTRL-H or backspace) loads the high nibbles. This distinction is necessary due to the nibble nature of 6653 EPROMs.

At any time the user may request a file reset by typing a ^R (CNTRL-R). This was more useful in earlier versions where a file reset was required each time either the low or high bytes were to be programmed. In the newer revisions this is unnecessary.

PROMP4.ASM requires that an input file be specified when the program is started. If immediate mode is exclusively used, a dummy file must still be specified. Such a file is PROMP4.COM.

An interruption, ie. any key closure, during a file transfer terminates the program. Recovery is accomplished by restarting the program from the beginning. This is done to somewhat protect the EPROM from possible damage due to the appearance of erroneous characters usually accompanied by an error termination.
{Author's note: For more information on commands available in the immediate mode of operation of the E-P, see its manual[6].}

Wire and connect all headers and connectors as specified in the hardware description and Tables A-1, A-2, and A-3. The parts necessary for the connector specified under Table A-1 are a male to male DB25 connection with four wires. (Remember to connect the terminal on the RIGHT serial port and the E-P on the LEFT serial port of the NorthStar.) The header from Table A-2 is a sixteen pin DIP header with wires soldered as specified. The switches are located on the bottom of the programmer.

Write, edit, and assemble a source program under the CP/M operating system. An object file of type xxxxxxxx.OBJ* or xxxxxxxx.HEX* must be created (or an appropriate dummy file substituted therefor). {It is assumed that the user can accomplish this without further instructions from this source, and that the user is relatively familiar with the CP/M operating system and its utilities.} Make sure that PROMP4.COM is on the specified system disk by typing "DIR PROMP4.*<cr>". To start the program type "PROMP4 <object program name>.<extension><cr>". The program should respond with "NorthStar <=> PC/M Model 660 CMOS EPROM Programmer Interface Program" and a command summary. The terminal is now connected to the CMOS EPROM programmer in the immediate mode of operation. If this should fail to occur, check all connections (including power on all equipment) and try again.

Downloading Notes

1. All addressing is absolute. Addresses greater than 03FFH will yield unpredictable results. Modify the source code appropriately.

* "xxxxxxxx" may be any user-selected CP/M file name.

2. Downloading may be terminated by typing any key during the downloading process. The program aborts under this condition.

Special Commands:

^C Return to CP/M operating system.

^H Load the high order nibbles of the source object code into the E-P buffer for later programming.

^L Load the low order nibbles of the source object code into the E-P buffer for later programming.

^R Reset the source input file.

E Erase check the EPROM in the socket of the E-P.

Format: "E<cr>".

R Read master EPROM into local buffer.

Format: "R<cr>".

V Verify that EPROM matches buffer contents and print out any locations that do not.

Format: "V<cr>".

C Complement buffer contents.

Format: "C<cr>".

S Set operation parameters (see manual[6]).

D Set carriage return delay (see manual[6]).

Format: "D mm <cr>" where mm is the delay in milliseconds that the E-P will wait before sending a character after outputting a carriage return.

P Program the EPROM with the buffer contents and verify that the specified EPROM's contents after the programming is the

same as that of the buffer.

Format: "P<cr>".

I Insert data into buffer (see manual[6]).

Format: "I aaa <cr>" where aaa is the address of the insert location. Data input should be a single hex nibble terminated with a carriage return. Termination of the insert mode is achieved through the use of a double carriage return.

T Type buffer contents from the specified starting address to the specifed ending address (see manual[6]).

Format: "T sss,eee <cr>" where sss is the starting location of the view block and eee is the ending address of the view block.

M Move the specified data block (see manual[6]).

Format: "M sss,eee,ddd <cr>" where sss is the starting address of the source location, eee is the ending address of the source location, and ddd is the destination address of the first byte of the block.

Possible Modification Suggestions

1. Adding to or Changing of the Command set may be achieved by alteration of the IN0 routine. The routine performs a sequential search of commands in this section of code. It is easy to include an extra CPI with the new command, and a JZ NEWCODE where NEWCODE is the new piece of in-line code. At the conclusion of NEWCODE the routine may either jump to CIN for acceptance of additional commands or jump to FINIS where

the program will terminate.

2. Modifying the program to work with 6654s is not so easily achieved. This requires that either the routine FILEH or FILEL and its associated jump in IN0 be eliminated. {For this example I will assume that FILEL must be modified.} Additional changes need to be made in the FILEL routine itself. These include removing the CALL GNB in codes 028F-0291H (this may be accomplished by inserting NOPs). The following section of code must be inserted after 0239H:

```
CDDF02*     CALL  GNB   ;GET LOW NIBBLE
47          MOV   B,A   ;TRANSMIT TO E-P
3E01        MVI   A,1   ;
CD8501*     CALL  XMIT  ;
```

The CALL GNB in memory locations 023A-023CH must be removed in order to complete the modification.

3. A suggested alteration, but one which will not be discussed, is that of adding a test for an E-P error condition in the middle of a file transfer and an exit from the automatic file transfer if this condition should occur.

* These addresses may relocate due to the above changes.

Sample Run*

```
{Author's note:  All user inputs appear underlined.}
[reset]
CP/M2 on North Star disk
Double density - Quad capacity
32K vers 2.20
Copyright (C) 1980 Lifeboat Associates

A>dir a:promp4.*[cr]
A: PROMP4   ASM : PROMP4   COM
A>dir b:test.*[cr]
B: TEST     BAK : TEST     OBJ : TEST         : TEST     PRN
A>promp4 b:test.obj[cr]

NorthStar <=> PC/M Model 660 CMOS EPROM Programmer Interface Program.

Command Summary:
   ^C  Return to CP/M.  System Reboot.
   ^H  Type out of the current input file
       and program HIGH nibbles into E-P
       buffer.
   ^L  Type out of the current input file
       and program LOW nibbles into E-P
       buffer.
   ^R  Reset the current input file.

[cr]                                    {Input by user.}
.r                                      {Input by user: Read empty socket.}
OK
.t 000,02F[cr]                          {Input by user: Type buffer contents.}
000 > F F F F F F F F
008 > F F F F F F F F
010 > F F F F F F F F
018 > F F F F F F F F
020 > F F F F F F F F
028 > F F F F F F F F
.[^H]I0000                              {Input by user: Load High nibbles}
000 > D,2,0,0,F,2,0,0,D,7,0,F,4,0,D,4,0,F,5,0,D,5,0,F,6,0,D,6,0,D,2,F,
020 >
OK
.I0020
020 > 2,2,0,0,1,0,0,0,0,0,
02A

Done filling the EPROM buffer with file data.
.t 000,037                              {Input by user: Type buffer contents.}
000 > D 2 0 0 F 2 0 0
008 > D 7 0 F 4 0 D 4
010 > 0 F 5 0 D 5 0 F
018 > 6 0 D 6 0 D 2 F
020 > 2 2 0 0 1 0 0 0
028 > 0 0 F F F F F F
030 > F F F F F F F F
```

* Nonprintable characters appear in brackets, eg. [cr].

```
.p[cr]                                  {Input by user: Program EPROM.}
000 > F D                                  {Note: Socket is empty and errors
001 > F 2                                   detected and printed by the E-P.}
002 > F 0
003 > F 0
005 > F 2
006 > F 0
007 > F 0
008 > F D
009 > F 7
00A > F 0
00C > F 4
00D > F 0
00E > F D
00F > F 4
010 > F 0
012 > F 5
013 > F 0
014 > F D
015 > F 5
016 > F 0
018 > F 6
019 > F 0
01A > F D
01B > F 6
01C > F 0
01D > F D
01E > F 2
020 > F 2
021 > F 2
022 > F 0
023 > F 0
024 > F 1
025 > F 0
026 > F 0
027 > F 0
028 > F 0
029 > F 0
.[^L]I0000                              {Input by user: Load Low nibbles.}
000> D,1,0,0,D,1,1,0,D,E,0,D,6,0,D,E,1,D,6,1,D,E,2,D,6,2,D,E,3,D,3,D,
020 >
OK
.I0020
020 > 3,1,0,0,1,1,0,1,2,0,
02A >

Done filling the EPROM buffer with file data.
.E                                      {Input by user: Erase check empty socket}
OK
.[^C]

A>

{END OF SAMPLE RUN.}
```

Test Program Print Listing

```
                                        SD SYSTEMS Z80 ASSEMBLER PAGE 0001
 ADDR  CODE      STMT SOURCE STATEMENT

                 0001 ;       Dwight W. Gordon                6-15-82
                 0002 ;       Kansas State University
                 0003 ;       Department of Electrical Engineering
                 0004 ;       Seaton Hall
                 0005 ;       Manhattan, Kansas 66506
                 0006 ;
                 0007 ;       This program is a simple test of the programming
                 0008 ; mode of the PROMP4.ASM program.
                 0009 ;
                 0010         PSECT   ABS
                 0012         NAME    TEST
 >0000           0013         ORG     0000H
 0000  DD210000  0014         LD      IX,0000H
 0004  FD210100  0015         LD      IY,0001H
 0008  DD7E00    0016         LD      A,(IX)
 000B  FD4600    0017         LD      B,(IY)
 000E  DD4E01    0018         LD      C,(IX+1)
 0011  FD5601    0019         LD      D,(IY+1)
 0014  DD5E02    0020         LD      E,(IX+2)
 0017  FD6602    0021         LD      H,(IY+2)
 001A  DD6E03    0022         LD      L,(IX+3)
 001D  DD23      0023         INC     IX
 001F  FD23      0024         INC     IY
 0021  210000    0025         LD      HL,0000H
 0024  110100    0026         LD      DE,0001H
 0027  010200    0027         LD      BC,0002H
                 0028         END

ERRORS=0000
```

TEST Program Object Listing

```
$TEST  05003F
:20000000DD210000FD210100DD7E00FD4600DD4E01FD5601DD5E02FD6602DD6E03DD23FDB8
:0A00200023210000110100010200 7D
:00000001FF
```

PROMP4.ASM Print Listing (ASM assembly)

```
;*************************************************************
;                                                            *
;       DWIGHT W. GORDON        10-16-81                     *
;                            REV. 5-18-82, 7-14-82           *
;                                                            *
;       KANSAS STATE UNIVERSITY                              *
;       DEPARTMENT OF ELECTRICAL ENGINEERING                 *
;       SEATON HALL                                          *
;       MANHATTAN, KS 66506                                  *
;============================================================*
;       PROMP4 IS AN INTERFACE PROGRAM BETWEEN THE CMOS      *
; EPROM PROGRAMMER (MODEL PC/M 660) AND THE NORTHSTAR        *
; CP/M VERSION OF THE ZASM (LIFEBOAT) ASSEMBLER.             *
;       THE PROGRAM IS DESIGNED SO THAT A SPECIFIED INPUT    *
; FILE IS OUTPUTTED TO THE PROGRAMMER IN NIBBLE FORMAT       *
; USING THE INSERT MODE OF PROGRAMMER OPERATION.  ALSO,      *
; THE USER MAY OPERATE THE PROGRAMMER IN THE STANDARD        *
; TERMINAL OPERATIONS MODE, ISSUING ONE COMMAND AT A TIME.   *
; SEE PROGRAMMER USER'S MANUAL FOR FURTHER INFORMATION.      *
;       WITH THE ABOVE RESTRICTIONS IN EFFECT, THE           *
; PROGRAMMER MUST BE CONFIGURED FOR TERMINAL OPERATION,      *
; THE BAUD RATE OF THE SECOND SERIAL PORT OF THE NS, 6603    *
; OPERATION, MASTER PROM, AND HEX OR BINARY FORMAT.          *
;       TO EXECUTE THE PROGRAM THE USER MUST SPECIFY A       *
; SOURCE FILE (EVEN IF ONE IS NOT GOING TO BE USED IN A      *
; RUN).  COMMANDS TYPED INTO THE NS TERMINAL ARE ECHOED TO   *
; AND FROM THE PROGRAMMER, EXCEPT FOR CERTAIN SPECIAL        *
; CONTROL CHARACTERS:  ^C, ^H (BACKSPACE), ^L, AND ^R.       *
; THERE FUNCTIONS ARE AS FOLLOWS:                            *
;       ^C      RETURN TO CP/M.  SYSTEM REBOOT.              *
;       ^H     *TYPE OUT OF THE CURRENT INPUT FILE: THE HIGH*
;              NIBBLES TO THE BUFFER OF THE PROGRAMMER.      *
;       ^L     *TYPE OUT OF THE CURRENT INPUT FILE: THE LOW  *
;              NIBBLES TO THE BUFFER OF THE PROGRAMMER.      *
;       ^R      RESET THE CURRENT INPUT FILE (USED WHEN BOTH*
;              HIGH AND LOW NIBBLES ARE TO BE PROGRAMMED     *
;              WITHOUT LEAVING THE PROGRAM).                 *
;                                                            *
; * NOTE:  COMPLETION IS SIGNALED BY THE SOUND OF A BELL     *
;          AND THE DONE MESSAGE ON THE SCREEN.               *
;       5-18-82 Revisions have made the file reset auto-     *
;          matic each time a ^L or ^H is used.               *
;          Also, the programmer may now terminate the        *
;          program in the ^L or ^H mode by simply pressing   *
;          any key on the keyboard.                          *
;       7-14-82 Program test conditions were terminal at     *
;          9600 Baud and E-P at 2400 Baud                    *
;     THIS PROGRAM WAS CONSTRUCTED BY THE CP/M FILE DUMP     *
; PROGRAM.  IT STILL CONTAINS SOME OF THE ATTRIBUTES OF      *
; SAID PROGRAM.  SOME OF THE FILE I-O DOCUMENTATION CAN BE   *
; OBTAINED FROM THE CP/M MANUALS:                            *
;     CP/M 2.0 INTERFACE GUIDE                               *
```

```
                ;     DIGITAL RESEARCH  (C) 1979                              *
                ;     PO BOX 579, PACIFIC GROVE, CA 93950. (408)-649-3896     *
                ;     PAGES 34-37                                             *
                ;*************************************************************
0100                          ORG     100H
0005 =          BDOS    EQU     0005H   ;DOS ENTRY POINT
0001 =          CONS    EQU     1       ;READ CONSOLE
0002 =          TYPEF   EQU     2       ;TYPE FUNCTION
0009 =          PRINTF  EQU     9       ;BUFFER PRINT ENTRY
000B =          BRKF    EQU     11      ;BREAK KEY FUNCTION (+ IF CHAR READY)
000F =          OPENF   EQU     15      ;FILE OPEN
0014 =          READF   EQU     20      ;READ FUNCTION
                ;
005C =          FCB     EQU     5CH     ;FILE CONTROL BLOCK ADDRESS
0080 =          BUFF    EQU     80H     ;INPUT DISK BUFFER ADDRESS
                ;
                ;       NON GRAPHIC CHARACTERS
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
                ;
                ;       FILE CONTROL BLOCK DEFINITIONS
005C =          FCBDN   EQU     FCB+0   ;DISK NAME
005D =          FCBFN   EQU     FCB+1   ;FILE NAME
0065 =          FCBFT   EQU     FCB+9   ;DISK FILE TYPE (3 CHARACTERS)
0068 =          FCBRL   EQU     FCB+12  ;FILE'S CURRENT REEL NUMBER
006B =          FCBRC   EQU     FCB+15  ;FILE'S RECORD COUNT (0 TO 128)
007C =          FCBCR   EQU     FCB+32  ;CURRENT (NEXT) RECORD Num. (0 TO 127)
007D =          FCBLN   EQU     FCB+33  ;FCB LENGTH
                ;
                ;       SET UP STACK
0100 210000             LXI     H,0
0103 39                 DAD     SP
                ;       ENTRY STACK POINTER IN HL FROM THE CCP
0104 221A06             SHLD    OLDSP
                ;       SET SP TO LOCAL STACK AREA (RESTORED AT FINIS)
0107 313C06             LXI     SP,STKTOP
010A 112C03             LXI     D,SIGNON        ;OUTPUT SIGN ON MESSAGE
010D CDD902             CALL    ERR
                ;       READ AND PRINT SUCCESSIVE BUFFERS
                ;       ALSO, ENTRY POINT FOR FILE RESET: ^R
0110 CD1003     RESETR  CALL    SETUP   ;SET UP INPUT FILE
0113 FEFF               CPI     255     ;255 IF FILE NOT PRESENT
0115 C22101             JNZ     OPENOK  ;SKIP IF OPEN IS OK
                ;
                ;       FILE NOT THERE, GIVE ERROR MESSAGE AND RETURN
0118 115D05             LXI     D,OPNMSG
011B CDD902             CALL    ERR
011E C3D102             JMP     FINIS   ;TO RETURN
                ;
                OPENOK: ;OPEN OPERATION OK, SET BUFFER INDEX TO END
0121 3E80               MVI     A,80H
0123 321806             STA     IBP     ;SET BUFFER POINTER TO 80H
```

```
                        ;
                        ;*********************************************************
                        ;       CIN IS THE INPUT POLLING ROUTINE USED TO SCAN
                        ; THE STATUS OF THE TWO INPUT DEVICES: TERMINAL AND E-P
                        ;*********************************************************
                        ;
0126 DB03       CIN     IN      3       ;CHECK STATUS OF TERMINAL
0128 2F                 CMA
0129 E602               ANI     2
012B CA3901             JZ      IN0     ; IF BUFFER FULL => IN0
012E DB05               IN      5       ;CHECK STATUS OF PROGRAMMER
0130 2F                 CMA
0131 E602               ANI     2
0133 CA5A01             JZ      IN1     ; IF BUFFER FULL => IN1
0136 C32601             JMP     CIN     ;    ELSE CYCLE AGAIN
                        ;
                        ;       END OF POLLING CYCLE
                        ;
                        ;*****************************************************************
                        ;       INPUT FROM TERMINAL AND CHECK FOR SPECIAL COMMAND,
                        ; ELSE SEND COMMAND TO E-P AND WAIT FOR ECHO BACK.
                        ;*****************************************************************
                        ;
0139 DB02       IN0     IN      2       ;GET DATA FROM TERMINAL
013B E67F               ANI     7FH     ;MASK OFF PARITY
013D FE03               CPI     3       ;CHECK FOR EXIT COMMAND ^C
013F CAD102             JZ      FINIS   ;  IF EXIT => FINIS
0142 FE0C               CPI     0CH     ;CHECK FOR PROGRAM LOW NIBBLE Cmd. ^L
0144 CA8002             JZ      FILEL   ;  IF PROGRAM LOW => FILEL
0147 FE08               CPI     08      ;CHECK FOR PROGRAM HIGH NIBBLE Cmd. ^H
0149 CAA702             JZ      FILEH   ;  IF PROGRAM HIGH => FILEH
014C FE12               CPI     12H     ;CHECK FOR INPUT FILE RESET COMMAND ^R
014E CA1001             JZ      RESETR  ;  IF INPUT FILE RESET => RESETR
0151 47                 MOV     B,A     ;MOVE DATA TO ACC-B
0152 3E01               MVI     A,01    ;SPECIFY OUTPUT DEVICE 1 (EPROM PROG.)
0154 CD8501             CALL    XMIT    ;XMIT COMMAND TO E-P (WAIT FOR ECHO)
0157 C32601             JMP     CIN     ;GET NEXT COMMAND
                        ;
                        ;       INPUT DATA FROM E-P AND ECHO TO TERMINAL
                        ;
015A DB04       IN1     IN      4       ;GET PROGRAMMER DATA
015C E67F               ANI     7FH     ;MASK AS BEFORE
015E 47                 MOV     B,A     ;DATA TO ACC-B
015F 3E00               MVI     A,0     ;OUTPUT TO DEVICE 0 (TERMINAL)
0161 CD8501             CALL    XMIT
0164 C32601             JMP     CIN     ;GET NEXT COMMAND
                        ;
                        ;       CINE WAITS FOR A SPECIFIC PERIOD OF TIME FOR INPUT
                        ; FROM THE E-P.
                        ;
0167 C5         CINE    PUSH    B       ;SAVE STATUS
0168 0690               MVI     B,90H   ;NUMBER OF TIMES E-P STATUS CHECKED
016A 05         CINER   DCR     B
```

```
016B CA8301             JZ      OUTCIN  ;IF NO DATA WITHIN THE SPECIFIED
                ;                       PERIOD OF TIME => EXIT
016E CD9B01             CALL    DELAYS  ;DELAY FOR A WHILE
0171 DB05               IN      5       ;CHECK E-P INPUT STATUS
0173 2F                 CMA
0174 E602               ANI     2
0176 C26A01             JNZ     CINER   ;NO DATA => TRY AGAIN
0179 DB04       INLE    IN      4       ;  ELSE GET DATA
017B E67F               ANI     7FH     ;MASK AS BEFORE
017D 47                 MOV     B,A     ;OUTPUT TO TERMINAL
017E 3E00               MVI     A,0
0180 CD8501             CALL    XMIT
0183 C1         OUTCIN  POP     B       ;RECOVER STATUS
0184 C9                 RET
                ;
                ;       XMIT TAKES THE DATA IN ACC-B AND OUTPUTS IT TO THE
                ; DEVICE SPECIFIED BY ACC-A (0=DEFAULT:TERMINAL;1=E-P).
                ;
0185 CDA401     XMIT    CALL    OST     ;WAIT FOR DEVICE BUFFER EMPTY
0188 C28501             JNZ     XMIT
018B FE01               CPI     1       ;E-P => COUTT1
018D CA9401             JZ      COUTT1
0190 78         COUTT0  MOV     A,B     ;DEFAULT  => TERMINAL
0191 D302               OUT     2
0193 C9                 RET
0194 78         COUTT1  MOV     A,B     ;E-P OUTPUT
0195 D304               OUT     4
0197 CD6701             CALL    CINE    ;ECHO TO TERMINAL DATA SENT TO E-P
019A C9                 RET
                ;
                ;       DELAY FOR A WHILE
                ;
019B C5         DELAYS  PUSH    B       ;SAVE PREVIOUS ENVIRONMENT
019C 0680               MVI     B,80H   ;LOAD DELAY CONSTANT
019E 05         DELAYL  DCR     B       ;DELAY!
019F C29E01             JNZ     DELAYL
01A2 C1                 POP     B       ;RESTORE PREVIOUS ENVIRONMENT
01A3 C9                 RET
                ;
                ;       CHECK IF OUTPUT BUFFER FOR SPECIFIED OUTPUT DEVICE
                ; IS EMPTY.  ZERO BIT SET WHEN BUFFER IS EMPTY.
                ;
01A4 FE01       OST     CPI     1
01A6 CAB101             JZ      OST1    ;E-P => OST1
01A9 DB03       OST0    IN      3       ;ELSE TERMINAL
01AB 2F                 CMA
01AC E601               ANI     1       ;SET ZERO FLAG IF BUFFER FULL
01AE 3E00               MVI     A,0     ;RESTORE ACC-A TO POINT TO TERMINAL
01B0 C9                 RET
01B1 DB05       OST1    IN      5       ;CHECK E-P
01B3 2F                 CMA
01B4 E601               ANI     1       ;SET ZERO FLAG IF BUFFER FULL
01B6 3E01               MVI     A,1     ;RESTORE ACC-A TO POINT TO E-P
01B8 C9                 RET
```

```
                    ;
                    ;       GET STARTING ADDRESS OF LINE OF INPUT DATA.
                    ;
01B9 CDDF02     NUM     CALL    GNB     ;GNB = GET NEXT BYTE
01BC FE3A               CPI     ':'     ;SEARCH FOR A ':'
01BE C2B901             JNZ     NUM
01C1 CDDF02             CALL    GNB     ;NEXT TWO ASCII CHARACTERS ARE
                    ;                    LENGTH OF LINE OF DATA
01C4 07                 RLC             ;CONVERT ASCII TO BINARY
01C5 07                 RLC
01C6 07                 RLC             ;FIRST DIGIT IS A 0, 1, or 2 ONLY!
01C7 07                 RLC
01C8 E6F0               ANI     0F0H    ;MASK OFF LOWER NIBBLE
01CA 47                 MOV     B,A     ;PUT UPPER NIBBLE IN ACC-B
01CB C5                 PUSH    B
01CC CDDF02             CALL    GNB     ;GET LSB (HEX/ASCII)
01CF C1                 POP     B
01D0 D630               SUI     '0'     ;CONVERT TO VALID HEX NIBBLE
01D2 FE0A               CPI     10
01D4 DAD901             JC      P101
01D7 D607               SUI     7
01D9 80         P101    ADD     B       ;ADD VALUE OF MS-NIBBLE
01DA 47                 MOV     B,A     ;MOVE RESULT TO ACC-B
01DB C9                 RET
                    ;
                    ;       OUTPUTS AN 'I' FOLLOWED BY THE FOUR DIGIT ADDRESS
                    ; OF THE NEXT DATA LINE OF THE INPUT FILE.
                    ;
01DC C5         CMD     PUSH    B       ;SAVE NUMBER OF BYTES ON LINE
                    ;                    OUTPUT COMMAND STRING 'Iaddr<cr>'
01DD CD6701             CALL    CINE    ;GET ANY RESIDUAL CHARACTERS BEING
01E0 CD6701             CALL    CINE    ;  SEND BY THE E-P
01E3 CD6701             CALL    CINE
01E6 3E01               MVI     A,1     ;TRANSMIT AN 'I'
01E8 0649               MVI     B,'I'
01EA CD8501             CALL    XMIT
01ED CD9B01             CALL    DELAYS  ;DELAY FOR TIME
01F0 CD9B01             CALL    DELAYS
01F3 CD6701             CALL    CINE    ;GET DATA BACK FROM E-P
01F6 CD9B01             CALL    DELAYS
01F9 CD6701             CALL    CINE
01FC 0604               MVI     B,4
01FE C5         CMD1    PUSH    B       ;OUTPUT STARTING ADDRESS
01FF CD6701             CALL    CINE    ;  FOUR ASCII CHARACTERS LONG
0202 CDDF02             CALL    GNB     ;  NO CHECKING FOR VALID PROGRAMMING
0205 47                 MOV     B,A     ;  ADDRESS (MUST BE 0000-03FF).
0206 3E01               MVI     A,1
0208 CD8501             CALL    XMIT
020B C1                 POP     B
020C 05                 DCR     B
020D C2FE01             JNZ     CMD1
0210 060D               MVI     B,CR    ;OUTPUT CR
0212 3E01               MVI     A,1
0214 CD8501             CALL    XMIT
```

```
0217 CDDF02           CALL    GNB     ;ADVANCE BUFFER POINTER TO DATA START
021A CDDF02           CALL    GNB     ;  PAST TWO ZASM-DEFINED CHARACTERS
021D 0607             MVI     B,7
021F C5       ECHOI   PUSH    B       ;GET RESPONSE FROM E-P
0220 CD6701           CALL    CINE
0223 C1               POP     B
0224 05               DCR     B
0225 C21F02           JNZ     ECHOI
0228 C1               POP     B       ;RECOVER LENGTH
0229 C9               RET
                ;
                ;       OUTPUT A SINGLE LINE OF DATA TO E-P FROM INPUT FILE
                ;
022A CD6701   OUTT    CALL    CINE    ;GET RESIDUAL TRANSMISSION FROM E-P
022D CD6701           CALL    CINE
0230 C5       OUTIN   PUSH    B       ;SAVE NUMBER OF BYTES
0231 CDDF02           CALL    GNB     ;GET DATA BYTE
0234 47               MOV     B,A     ;TRANSMIT TO E-P
0235 3E01             MVI     A,1
0237 CD8501           CALL    XMIT
023A CDDF02           CALL    GNB     ;SKIP ONE NIBBLE
023D 062C             MVI     B,','   ;TRANSMIT ',' TO E-P
023F 3E01             MVI     A,1
0241 CD8501           CALL    XMIT
0244 C1               POP     B
0245 05               DCR     B       ;ANY MORE BYTES ON THIS LINE?
0246 C23002           JNZ     OUTIN   ; +=> OUTIN
0249 060D             MVI     B,CR    ;SIGNAL END OF LINE <cr>,<cr>
024B 3E01             MVI     A,1
024D CD8501           CALL    XMIT
0250 0608             MVI     B,8     ;GET RESPONSE FROM E-P
0252 C5       ECHO01  PUSH    B
0253 CD6701           CALL    CINE
0256 C1               POP     B
0257 05               DCR     B
0258 C25202           JNZ     ECHO01
025B 060D             MVI     B,CR    ;SIGNAL END OF INSERT COMMAND
025D 3E01             MVI     A,1     ;  BY TRANSMISSION OF A CR
025F CD8501           CALL    XMIT
0262 0607             MVI     B,7
0264 C5       ECHOO   PUSH    B       ;GET E-P RESPONSE
0265 CD9B01           CALL    DELAYS  ;  BY DELAYING AND GETTING DATA
0268 C1               POP     B
0269 05               DCR     B
026A C26402           JNZ     ECHOO
026D CD6701           CALL    CINE
0270 C9               RET
                ;
                ;       OUTPUT A LF,CR TO TERMINAL
                ;
0271 060A     CRLF0   MVI     B,LF
0273 3E00             MVI     A,0
0275 CD8501           CALL    XMIT
0278 060D             MVI     B,CR
```

```
027A 3E00              MVI     A,0
027C CD8501            CALL    XMIT
027F C9                RET
                ;
                ;       ROUTINE TO TRANSFER LOWER NIBBLES OF INPUT FILE
                ; TO STORAGE BUFFER OF E-P.
                ;
0280 CD1003     FILEL  CALL    SETUP   ;REINITIALIZE FILE POINTER
0283 CDB901     FILEL1 CALL    NUM     ;GET LENGTH AND PLACE IN ACC-B
0286 3EFF              MVI     A,0FFH
0288 A0                ANA     B       ;MOVE LENGTH TO ACC-A; IF LENGTH=0
0289 CA9B02            JZ      DONEMG  ; PRINT THE DONEMessaGe
028C CDDC01            CALL    CMD     ; ELSE PRINT INSERT COMMAND
028F CDDF02            CALL    GNB     ;SKIP THE NEXT BYTE (POINT TO FIRST
                ;                      LOWER NIBBLE
0292 CD2A02            CALL    OUTT    ;OUTPUT A LINE
0295 CDBF02            CALL    BREAK   ;CHECK FOR BREAK
0298 C38302            JMP     FILEL1  ;ELSE DO IT UNTIL DONE
                ;
                ;       MESSAGE SENT TO TERMINAL UPON COMPLETION OF A
                ; DATA TRANSFER.
                ;
029B CD7102     DONEMG: CALL    CRLF0   ;OUTPUT A CR,LF TO TERMINAL
029E 11DF05            LXI     D,DONMSG ;START OF MESSAGE
02A1 CDD902            CALL    ERR     ;OUTPUT
02A4 C32601            JMP     CIN     ;RETURN TO INPUT POLLING
                ;
                ;       ROUTINE TO TRANSFER UPPER NIBBLES OF INPUT FILE
                ; TO STORAGE BUFFER OF E-P.
                ;
02A7 CD1003     FILEH  CALL    SETUP   ;REINITIALIZE FILE POINTER
02AA CDB901     FILEH1 CALL    NUM     ;GET LENGTH
02AD 3EFF              MVI     A,0FFH
02AF A0                ANA     B
02B0 CA9B02            JZ      DONEMG  ;IF DONE THEN EXIT
02B3 CDDC01            CALL    CMD     ;OUT INSERT COMMAND
02B6 CD2A02            CALL    OUTT    ;OUT A LINE
02B9 CDBF02            CALL    BREAK   ;CHECK FOR A BREAK
02BC C3AA02            JMP     FILEH1  ;ELSE DO UNTIL DONE
                ;
                ;***********************************************
                ;
                BREAK:  ;BREAK DETECTION ROUTINE (REALLY, ANY KEYBOARD
                        ; CLOSURE WILL DO)
02BF F5                 PUSH    PSW
02C0 DB03               IN      3       ;GET 8251 STATUS
02C2 E602               ANI     02H
02C4 C2C902             JNZ     FINIS1  ;EXIT WHEN CHARACTER DETECTED
02C7 F1                 POP     PSW
02C8 C9                 RET
```

```
                        ;
                        ;***********************************************
                        ;
02C9 DB02               FINIS1: IN      2       ;REMOVE DUMMY CHARACTER FROM BUFFER
02CB 11A104                     LXI     D,INTRPR ;OUTPUT INTeRruPt eRror MESSAGE
02CE CDD902                     CALL    ERR
                        ;
                        FINIS:
                        ;       END OF SECESSION, RETURN TO CCP
                        ;       (NOTE THAT A JMP TO 0000H REBOOTS)
02D1 CD7102                     CALL    CRLF0
02D4 2A1A06                     LHLD    OLDSP
02D7 F9                         SPHL
                        ;       STACK POINTER CONTAINS CCP'S STACK LOCATION
02D8 C9                         RET             ;TO THE CCP
                        ;
                        ;
                        ;       SUBROUTINES
                        ;
                        ;***************************************************
                        ERR:    ;PRINT ERROR OR ANY MESSAGE
                        ;  D,E ADDRESSES MESSAGE ENDING WITH '$'
                        ;
02D9 0E09                       MVI     C,PRINTF        ;PRINT BUFFER FUNCTION
02DB CD0500                     CALL    BDOS
02DE C9                         RET
                        ;
                        ;***************************************************
02DF 3A1806             GNB:    LDA     IBP     ;GET NEXT INPUT FILE BYTE
02E2 FE80                       CPI     80H
02E4 C20003                     JNZ     G0      ;IF RAM BUFFER NOT EMPTY THEN G0
02E7 CD1D03                     CALL    DISKR
02EA B7                         ORA     A       ;ZERO VALUE IF READ OK
02EB CA0003                     JZ      G0      ;FOR ANOTHER BYTE
02EE F5                         PUSH    PSW     ;PRINT THE ERROR MESSAGE
02EF E67F                       ANI     07FH    ;  ON FILE READ ERROR (EOF)
02F1 47                         MOV     B,A
02F2 3E00                       MVI     A,0
02F4 CD8501                     CALL    XMIT
02F7 113D05                     LXI     D,FREADR ;File READ eRror message start
02FA CDD902                     CALL    ERR     ;OUTPUT
02FD F1                         POP     PSW
02FE 37                         STC             ;SET CARRY => ERROR ON FILE READ
02FF C9                         RET
0300 5F                 G0:     MOV     E,A     ;LS BYTE OF BUFFER INDEX
0301 1600                       MVI     D,0     ;DOUBLE PRECISION INDEX TO DE
0303 3C                         INR     A       ;INDEX=INDEX+1
0304 321806                     STA     IBP     ;BACK TO MEMORY
0307 218000                     LXI     H,BUFF
030A 19                         DAD     D
030B 7E                         MOV     A,M
030C B7                         ORA     A       ;RESET CARRY BIT
030D E67F                       ANI     7FH     ;  => SUCCESSFUL FILE READ
030F C9                         RET
```

```
0310 AF          SETUP:  XRA     A       ;ZERO TO ACCUM
0311 327C00              STA     FCBCR   ;CLEAR CURRENT RECORD
0314 115C00              LXI     D,FCB
0317 0E0F                MVI     C,OPENF ;OPEN INPUT FILE
0319 CD0500              CALL    BDOS
031C C9                  RET
031D E5D5C5      DISKR:  PUSH H! PUSH D! PUSH B  ;DISK READ FUNCTION
0320 115C00              LXI     D,FCB
0323 0E14                MVI     C,READF
0325 CD0500              CALL    BDOS
0328 C1D1E1              POP B! POP D! POP H
032B C9                  RET
032C 0D0A4E6F72SIGNON    DB  CR,LF,'NorthStar <=> PC/M Model 660 CMOS EPROM '
0356 50726F6772          DB  'Programmer Interface Program.',CR,LF,LF
0376 436F6D6D61          DB  'Command Summary:',CR,LF
0388 2020205E43          DB  '   ^C  Return to CP/M.  System Reboot.',CR,LF
03B0 2020205E48          DB  '   ^H  Type out of the current input file',CR,LF
03DB 2020202020          DB  '       and program HIGH nibbles into E-P',CR,LF
0405 2020202020          DB  '       buffer.',CR,LF
0415 2020205E4C          DB  '   ^L  Type out of the current input file',CR,LF
0440 2020202020          DB  '       and program LOW nibbles into E-P',CR,LF
0469 2020202020          DB  '       buffer.',CR,LF
0479 2020205E52          DB  '   ^R  Reset the current input file.',CR,LF,LF,'$'
04A1 0D0A0707    INTRPR: DB    CR,LF,07H,07H
04A5 2A2A2A2A2A          DB  '**********************************',CR,LF
04CA 2A2A2A2A2A          DB  '*******FILE INPUT INTERRUPTED******',CR,LF,07H
04F0 2A2A2A2A2A          DB  '**********PROGRAM ABORTED**********',CR,LF,07H
0516 2A2A2A2A2A          DB  '**********************************',CR,LF,07H,'$'
053D 0D070A2A2AFREADR    DB CR,07H,LF,'*****FILE READ ERROR*****',07H,CR,LF,'$'
055D 0D0A4E4F20OPNMSG:   DB CR,LF,'NO INPUT FILE PRESENT ON DISK',LF,CR
057E 466F722049          DB  'For IMMEDIATE mode operation ONLY, specify a '
05AB 64756D6D79          DB  'dummy source file',CR,LF
05BE 7768656E20          DB  'when you restart this program.',LF,CR,'$'
05DF 0D0A070707DONMSG:   DB CR,LF,07H,07H,07H,07H,07H,'Done filling the EPROM '
05FD 6275666665          DB  'buffer with file data.',CR,LF,07H,07H,'$'
0618             IBP:    DS      2
061A             OLDSP:  DS      2
061C                     DS      32
                 STKTOP:
                 ;
063C                     END
```

PROMP4.ASM Object Listing

```
:10010000210000039221A06313C06112C03CDD902F8
:10011000CD1003FEFFC22101115D05CDD902C3D16F
:10012000023E80321806DB032FE602CA3901DB05E6
:100130002FE602CA5A01C32601DB02E67FFE03CA8C
:10014000D102FE0CCA8002FE08CAA702FE12CA1023
:1001500001473E01CD8501C32601DB04E67F473E12
:1001600000CD8501C32601C5069005CA8301CD9B3C
:1001700001DB052FE602C26A01DB04E67F473E0091
:10018000CD8501C1C9CDA401C28501FE01CA94017A
:1001900078D302C978D304CD6701C9C5068005C2EA
:1001A0009E01C1C9FE01CAB101DB032FE6013E0079
:1001B000C9DB052FE6013E01C9CDDF02FE3AC2B917
:1001C00001CDDF0207070707E6F047C5CDDF02C113
:1001D000D630FE0ADAD901D6078047C9C5CD6701F6
:1001E000CD6701CD67013E010649CD8501CD9B015B
:1001F000CD9B01CD6701CD9B01CD67010604C5CD27
:100200006701CDDF02473E01CD8501C105C2FE0178
:10021000060D3E01CD8501CDDF02CDDF020607C50B
:10022000CD6701C105C21F02C1C9CD6701CD6701FC
:10023000C5CDDF02473E01CD8501CDDF02062C3E54
:1002400001CD8501C105C23002060D3E01CD8501FB
:100250000608C5CD6701C105C25202060D3E01CD9B
:10026000850106 07C5CD9B01C105C26402CD6701AA
:10027000C9060A3E00CD8501060D3E00CD8501C9A7
:10028000CD1003CDB9013EFFA0CA9B02CDDC01CD4C
:10029000DF02CD2A02CDBF02C38302CD710211DF7E
:1002A00005CDD902C32601CD1003CDB9013EFFA073
:1002B000CA9B02CDDC01CD2A02CDBF02C3AA02F542
:1002C000DB03E602C2C902F1C9DB0211A104CDD9E8
:1002D00002CD71022A1A06F9C90E09CD0500C93AE4
:1002E0001806FE80C20003CD1D03B7CA0003F5E661
:1002F0007F473E00CD8501113D05CDD902F137C9BB
:100300005F16003C32180621800019 7EB7E67FC9CF
:10031000AF327C00115C000E0FCD0500C9E5D5C5DC
:10032000115C000E14CD0500C1D1E1C90D0A4E6F5C
:1003300072746853746172203C3D3E2050432F4DCF
:10034000204D6F64656C2036363020434D4F53206E
:100350004550524F4D2050726F6772616D6D6572DE
:1003600020496E7465726661636520 50726F6772B2
:10037000616D2E0D0A0A436F6D6D616E64205375B9
:100380006D6D6172793A0D0A2020205E4320205263
:1003900065747572E20746F2043502F4D2E20208F
:1003A000053797374656D205265626F6F742E0D0AF8
:1003B0002020205E48202054797065206F757420BD
:1003C0006F6620746865206375727265E7420694B
:1003D0006E7075742066696C650D0A2020202020DF
:1003E0002020616E642070726F6772616D204849D1
:1003F0004748206E6962626C657320696E746F2075
:10040000452D500D0A20202020202020206275666690
:1004100065722E0D0A2020205E4C202054797065D4
:10042000206F7574206F66207468652063757272 22
:10043000656E7420696E7075742066696C650D0A4E
```

PROMP4.ASM Object Listing (cont.)

```
:10044000202020202020616E642070726F67724F
:10045000616D204C4F57206E6962626C6573206934
:100460006E746F20452D500D0A2020202020202062
:10047000627566666572E0D0A2020205E5220206D
:10048000526573657420746865206375727265E59
:1004900074206696E7075742066696C652E0D0A0A89
:1004A000240D0A07072A2A2A2A2A2A2A2A2A2A35
:1004B0002A2A2A2A2A2A2A2A2A2A2A2A2A2A2A9C
:1004C0002A2A2A2A2A2A2A0D0A2A2A2A2A2AC9
:1004D0002A46494C4520494E50555420494E5445D2
:1004E000525255550544544 2A2A2A2A2A0D0A07CC
:1004F0002A2A2A2A2A2A2A2A2A2A50524F4752418D
:100500004D2041424F525445442A2A2A2A2A2A57
:100510002A2A0D0A072A2A2A2A2A2A2A2A2A9B
:100520002A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2B
:100530002A2A2A2A2A2A2A2A0D0A07240D070AE1
:100540002A2A2A2A46494C4520524541442045 18
:1005500052524F522A2A2A2A070D0A240D0A4EDD
:100560004F20494E5055542046494C452050524545
:1005700053454E54204F4E204449534B0A0D466F6D
:10058000720494D4D454449415445206D6F646585
:10059000206F7065726174696F6E204F4E4C592CDC
:1005A000207370656369667920612064756D6D796B
:1005B00020736F7572636520666696C650D0A7768D4
:1005C000656E20796F7520726573746172742074 22
:1005D000686973207 0726F6772616D2E0A0D240D49
:1005E0000A0707070707446F6E652066696C6C6928
:1005F0006E67207468652045 50524F4D20627566C5
:10060000666572207769746820666696C652064612C
:0806100074612E0D0A07072496
:0000000000
```

APPENDIX B: ADM-3A <=> NSC800 Board Interface

Assembly Instructions and Wiring List

1. Insert wire wrap sockets and posts into specified locations on the NSC800 board and interface board. Locations may be obtained from the component layouts, Figures B-1a and B-1b.

2. Using 30 gauge Kynar wire wrap wire (or any other suitable substitute), wrap the connections as specified in Tables B-1 and B-2, and Figures B-1a, B-1b and B-2.

Note: If sufficient wire wrap space is available on the NSC800 cpu board then J8, J9, the 16-pin dip connector, and the 2.5" x 5" wire wrap board may be eliminated. The remaining components may be located on the NSC800 board wherever space permits. These should be wired according the the interface schematics (Figure B-2.).

Component Layouts

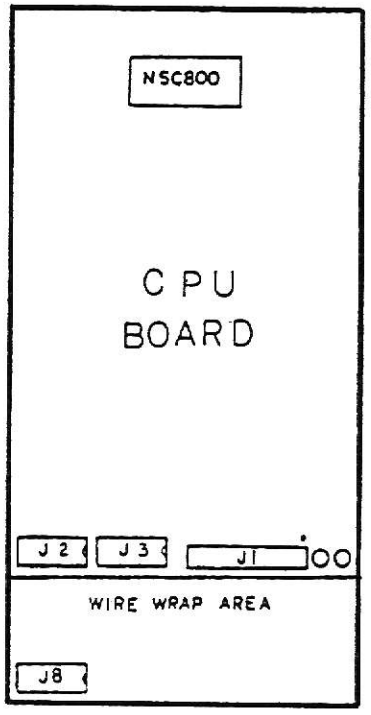


Figure B-1a.

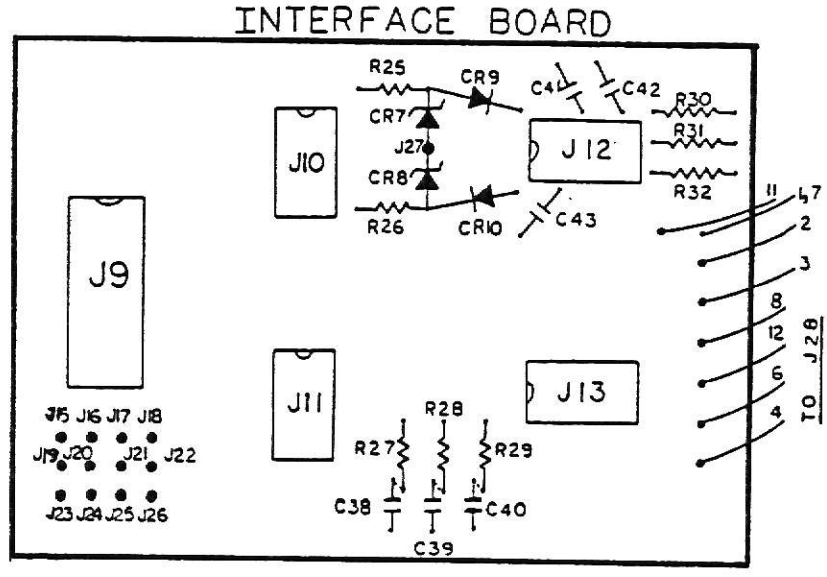


Figure B-1b.

NSC800 Board Wiring List

From	To
J8 PIN 1	J8 PIN 16
J8 PIN 1	J1 PIN 1
J8 PIN 2	J1 PIN 4
J8 PIN 3	J1 PIN 49
J8 PIN 3	J1 PIN 50
J8 PIN 4	J3 PIN 11
J8 PIN 5	J3 PIN 12
J8 PIN 6	J3 PIN 16
J8 PIN 7	J3 PIN 2
J8 PIN 8	J3 PIN 3
J8 PIN 9	J3 PIN 4
J8 PIN 10	J3 PIN 5
J8 PIN 11	J3 PIN 8
J8 PIN 12	J3 PIN 7
J8 PIN 13	J3 PIN 1
J8 PIN 15	J3 PIN 3
J8 PIN 16	J1 PIN 2
Cut trace at pin 28 of U1	

Table B-1.

Interfact Board Wiring List

From	To
J15	J9 PIN 3, J16, J17, J18
J19	J9 PIN 7, J11 PIN 10
J20	J9 PIN 8, J11 PIN 11
J21	J9 PIN 9, J11 PIN 12
J22	J9 PIN 10, J11 PIN 13
J23	J9 PIN 1, J24, J25, J26
J11 PIN 1	J9 PIN 11
J11 PIN 2	J9 PIN 12
J11 PIN 3	J9 PIN 13
J11 PIN 4	J9 PIN 6
J11 PIN 5	J9 PIN 5
J11 PIN 6	J9 PIN 4
J11 PIN 7	J9 PIN 1
J11 PIN 14	J9 PIN 3
J10 PIN 1	J9 PIN 4
J10 PIN 2	J12 PIN 13, J12 PIN 12
J10 PIN 7	J9 PIN 1
J10 PIN 10	J9 PIN 11
J10 PIN 11	J13 PIN 6
J10 PIN 14	J9 PIN 3
J13 PIN 3	J9 PIN 11
J13 PIN 7	J9 PIN 1
J13 PIN 8	J9 PIN 13
J13 PIN 14	J9 PIN 3

Interfact Board Wiring List (cont.)

J14 PIN 2	J9 PIN 6
J14 PIN 7	J9 PIN 1
J14 PIN 8	J9 PIN 5, J14 PIN 9
J14 PIN 12	J14 PIN 13
R25a	J9 PIN 2
R25b	CR7 CATHODE
CR7 CATHODE	CR9 ANODE
CR7 ANODE	J27
CR9 CATHODE	J12 PIN 14
J27	CR8 CATHODE, J9 PIN 1
CR9 ANODE	R26b, CR10 CATHODE
CR10 ANODE	J12 PIN 1
J27	C41a, C42a, C43a
C41b	J12 PIN 8, R30a
C42b	J12 PIN 11, R31a
C43b	J12 PIN 3, R32a
R30b	J9 PIN 1, R31b, R32b
R28a	J9 PIN 3, R29a, R30a
R28b	J13 PIN 8, C38a
R29b	J13 PIN 5, C39a
R30b	J13 PIN 2, C40a
C38b	J9 PIN 1, C39b, C40b
J28 PIN 1	J9 PIN 1, J28 PIN 7
J28 PIN 2	J13 PIN 10
J28 PIN 3	J12 PIN 3

Interfact Board Wiring List (cont.)

J28 PIN 4	J13 PIN 1
J28 PIN 8	J12 PIN 11
J28 PIN 11	J13 PIN 4
J28 PIN 12	J12 PIN 8

Table B-2.

Interface Parts List

Two 16-pin wire wrap sockets (J8 and J9)

Four 14 pin wire wrap sockets (J10-J13)

Thirteen double-ended wire wrap pins (J15-J27)

One DB25S connector (J28)

One 74PC04 hex inverter (U28)

One 1488 quad RS232 driver (U29)

One 1489 quad RS232 receiver (U30)

One 180 ohm, 1/4 watt resistor (R25)

One 300 ohm, 1/4 watt resistor (R26)

Three 13K ohm, 1/4 watt resistors (R27-R29)

Three 2K ohm, 1/4 watt resistors (R30-R32)

Two 10 volt zener diodes (CR7, CR8)

Two 20 PIV diodes (CR9, CR10)

Three 10p farad ceramic disk capacitors (C38-C40)

Three 330p farad ceramic disk capacitors (C41-C43)

One 2.5" x 5" circuit board suitable for wire wrapping

One 16 pin dip ribbon connector.

Table B-3.

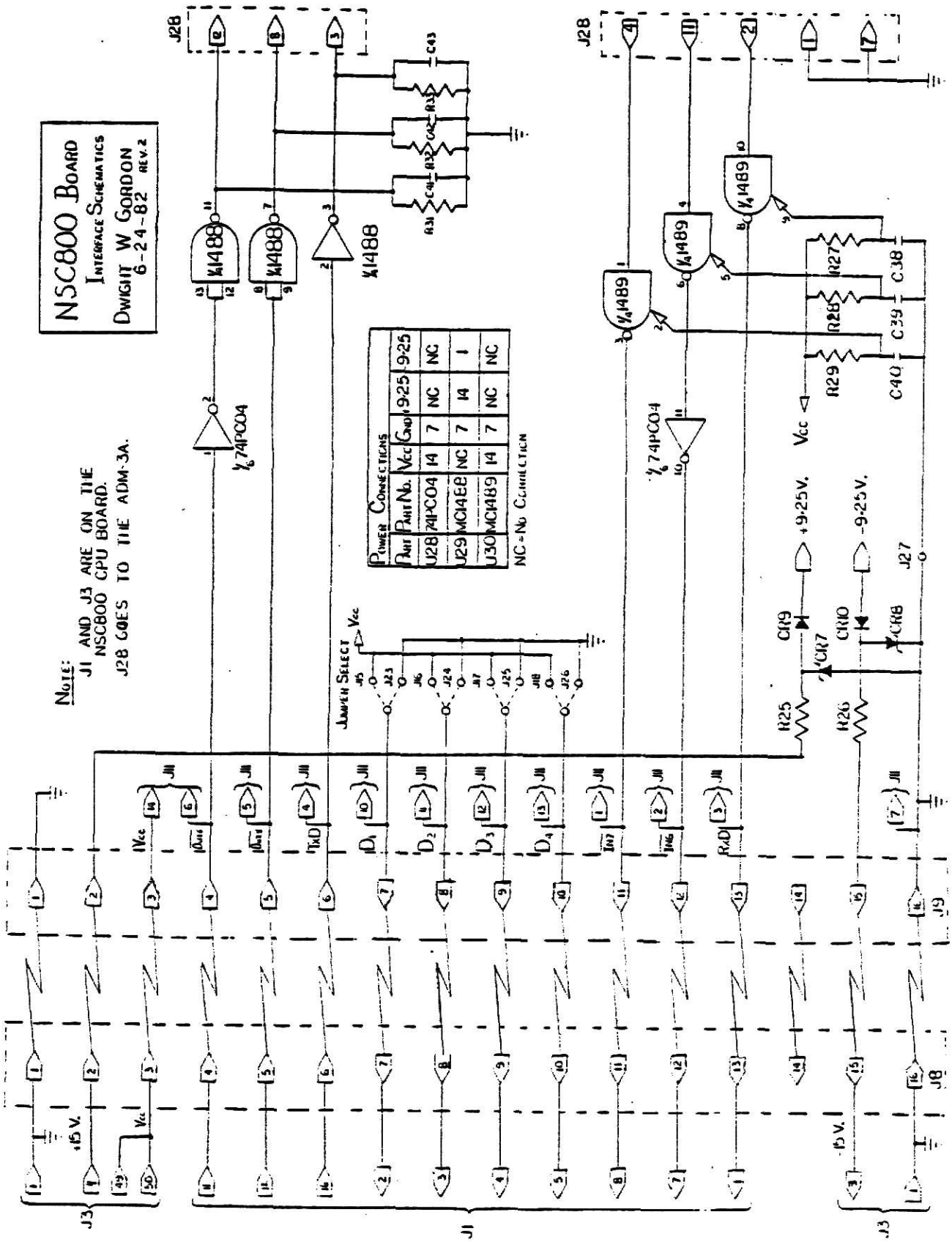

NSC800 Board
Interface Schematics
Dwight W Gordon
6-24-82 Rev 2
Note:
J1 and J3 are on the NSC800 CPU board.
J28 goes to the ADM-3A.
Power Connections
Part | Part No. | Vcc | Gnd | +9.25 | -9.25
U28 | 74PC04 | 14 | 7 | NC | NC
U29 | MC1488 | NC | 7 | 14 | 1
U30 | MC1489 | 14 | 7 | NC | NC
NC = No Connection
Jumper Select
+9.25V.
-9.25V.
+5 V.
Vcc
J1
J3
J27
J28

Figure B-2.

1. Plug in the ADM-3A and turn on the power by flipping the switch on the back of the terminal.

2. Make sure that the appropriate monitor EPROMs are located in the sockets on the cpu board. Take special care to make sure that the high nibble EPROM is in the high nibble socket.

3. Use a 16 pin dip ribbon connector to connect the cpu board to the interface board. As designed, pin 1 of J8 is the closest to the fifty pin connector, and pin 1 of J9 is farthest from all components on the interface board.

4. Set the configuration jumpers on the interface board according to the NSC800 system clock frequency, baud rate, and Tables B-5a and B-5b. D1 is located to the right of D4 when looking at the interface board with the ribbon connector on the left.

UART Configuration

D2	D1	Clock Frequency
0	0	1.0 MHz.
0	0	2.0 MHz.
1	0	2.5 MHz.
1	1	4.0 MHz.

Table B-4a.

D4	D3	Baud Rate
0	0	300
0	1	600
1	0	1200
1	1	2400

Table B-4b.

5. Set the switches internal and external to the ADM-3A according to Table 1 and Table 2 (see page 22). The baud rate selected should correspond to that chosen in 4. above.

6. Connect the ADM-3A (modem) to the DB25 connector on the interface board of the NSC800 by the appropriate connector. If an

incorrect connector is used, data will be lost. Consult the schematics (Figure B-2) for necessary wires on the connector.

7. Apply ground to pins 1 and 2 of J1, +5 volts to pins 49 and 50 of J1, +15 volts to pin 4 of J1, and -15 volts to pin 3 of J1.

8. Press the reset button on the NSC800 board. The monitor should wake up with the banner:

```
NSC800 Monitor Version 1.91
Kansas State University
>
```

{The black mark in the above shows the location of the cursor.} You are now in control of the NSC800 monitor. Any of the monitor commands may now be issued. For more information on the available commands, see the command explanation below.

Command Summary

M View and alter memory starting at the user specified address.
Format: "Mxxxx<cr>" where xxxx is a four nibble address.

G Start program execution at the specified address.
Format: "Gxxxx<cr>" where xxxx is a four nibble address.

V View a block of memory between the specified bounds.
Format: "Vxxxx,yyyy<cr>" where xxxx is a four nibble starting address and yyyy is a four nibble ending address

R View the current machine registers.
Format: "R".

C Copy a block of memory over the specified bounds.
Format: "C". All questions must be answered with a four

nibble answer.

B Set a single user breakpoint.

Format: "Bxxxx<cr>" where xxxx is the breakpoint address.

N Restart a program after it was interrupted by a breakpoint.

Format: "N".

Detailed Command Description

Memory Modify, 'M,' allows the user to view and alter any Read/Write (eg. RAM) location in the memory space. Write only locations (eg. DAC) may be written. Read only locations (eg. ADC) may be read.

To invoke the mode the user need only type 'Mxxxx<cr>'. 'xxxx' is at least a four nibble address. If more than four valid hex digits are entered prior to the carriage return, only the last four digits will be kept as the address. If less than four valid hex digits or an invalid hex digit is input, the monitor will respond with '???' and the terminal bell.

The monitor is now in the memory modify mode. The monitor responds with the specified address and the present contents of said address. The user must now type two or more valid hex digits to be entered into the specified location. If less than two valid hex digits or an invalid hex digit (with two exceptions: <lf> and 'S'), the routine will error terminate with '???' and the sound of the terminal bell. (Termination means that command returns to the command mode signalled with the command prompt: '>'.)

If at any time (even after valid hex digits have been

entered) a line feed, <lf>, is entered then the monitor will assume that the original content of the memory location is correct, and the routine continues with the next sequencial memory location. If at any time an 'S' is entered, all previous entries for the displayed memory location are ignored and the routine terminates and returns to the command mode.

When the carriage return is entered the last two entries are interpreted and entered into the specified memory location. A verify is done to make sure that the location has been correctly altered. If the verify fails then the routine error terminates with '???' and the sound of the terminal bell. If the verify passes, the routine displays the next sequencial memory address and its content for modification.

Starting program execution may be accomplished through the use of the Go command. The format for this command is 'Gxxxx<cr>' where xxxx is the address at which program execution should start. All registers displayed (see Register view command) are popped from the stack (the stack pointer is appropriately altered) prior to the start of program execution. This allows the user to specify the machine status (through the use of the Register view command to find the stack pointer, and the Memory modify command to alter the associated memory locations) at the beginning of program execution (see Table B-5.).

A number of inputs will cause the routine to error terminate. Input of other than a carriage return or a valid hex digit will cause the monitor to respond with '???' and sound the terminal

bell. Input of less than four valid hex digits prior to that of the carriage return also causes the error termination. After an error termination, command is returned to the monitor. (The command input prompt, '>', is displayed.)

Viewing a block of memory is done with the View command. It's format is 'Vzzzz,xxxx<cr>' where zzzz is the starting address of the view block and xxxx is the last address of the view block. zzzz may contain zero or more hex nibbles. Zero's are padded at the beginning of this address for unspecified digits. For example, either 'V0000,1234<cr>', 'V000,1234<cr>', 'V00,1234<cr>', 'V0,1234<cr>', or 'V,1234<cr>' will cause the memory from 0000H to 1234H to be displayed.

The second address, xxxx, must be no less than four valid hex nibbles. If more than four are entered only the last four are kept. If less than four or an invalid hex digit is entered, then the routine will respond with '???' and the sound of the terminal bell. Command is returned to the command input section of the monitor.

The output format of the view block is designed for user readability. The starting address of a line is displayed on the left followed by the data bytes (in nibble format) separated by a space. The monitor will attempt to place addresses with a zero as the least significant nibble as the starting address of a line of output. See sample run for an example of this output.

Currently executable machine status may be observed with the Register view command. The command is executed upon detection of

'R' when in the command mode. All registers except I and R are available. The current value of the stack pointer is the last filled stack location. All registers displayed are currently on the stack. The value of the stack pointer increases by 20D (the number of register bytes on the stack) when a Go command is executed. When a Next command is executed the value will increase by 22D (the program counter is also removed from the stack).

This command is useful when used with the Memory modify command in order to change the machine status. The registers are on the stack in the order specified (recall that the Z80 uses low-byte/high-byte ordering for 16 bit values in memory). Table B-5 contains the location of each of the displayed registers relative to the displayed value of the stack pointer.

NSC800 Register Locations (relative to the Stack Pointer)

Register	Location	Register	Location
A'	SP+ 1	A	SP+ 9
B'	SP+ 3	B	SP+11
C'	SP+ 2	C	SP+10
D'	SP+ 5	D	SP+13
E'	SP+ 4	E	SP+12
H'	SP+ 7	H	SP+15
L'	SP+ 6	L	SP+14
F'	SP+ 0	F	SP+ 8
IX	SP+16	PC	SP+20
IY	SP+18		

Table B-5.

Copying of a block of memory may be done with the Copy command. This routine will allow the user to copy a block of a predetermined length from a source location to a destination. This is accomplished through the use of a single byte sequential move.

Input format is 'C'. The monitor will respond with 'From? '. The user must now input a four^{+} valid hex nibble address terminated with a carriage return. At which time the monitor will return with 'To? '. Again, the user must input a four^{+} valid hex nibble address terminated with a carriage return. Finally, the monitor will output 'Length? '. The user must now input a four valid hex nibble length terminated with a carriage return. If any of the above numbers are incorrectly entered, the routine will error terminate with '???' and the sound of the terminal bell. Command is returned to the command mode at the completion of the copy, or at the detection of an invalid entry. The copy does no verify.

Care must be taken in order to copy text forward where part of the destination block lies over the source block. This must be done in two steps. First the block must be copied to a free portion of RAM. The relocated block may now be moved to the final destination address. This two step procedure is necessary because the routine will destroy part of the source block when an interleaved copy is performed.

The copy routine may also be used to fill a block of memory with a specified value. This, too, is a two step procedure. It

is first necessary to initialize the first address of the block to the value through the use of the Memory modify command. Copy may now be used with the following format for entry:

'From' Address = Starting address of the block,

'To' Address = Starting address of the block + 1,

and 'Length' = Physical length of the block - 1.

The NSC800 monitor allows the user to specify a single breakpoint with the Breakpoint command. This command will also allow the user to remove any previously set breakpoint. (This is normally done automatically during program execution when a breakpoint is reached thereby allowing the user to "single step" manually through a program by setting in turn sequential breakpoints at each opcode location). A breakpoint must only be placed at the first byte of any multibyte intruction. Incorrect operation will occur if a user tries to place a breakpoint at other than the first byte.

Input is initiated by typing 'Bxxxx<cr>' where xxxx is the address at which the breakpoint is to reside. xxxx must be a at least four valid hex digits. If more than four hex digits are entered, the last four will be retained as the address. If an incorrect address in any form is input, the routine will terminate with '???' and the terminal bell. Command returns to the command input mode.

The error termination can be utilized to remove a previously set breakpoint by typing 'Bq' where q is any invalid hex digit. The routine removes any old breakpoints immediately after the 'B'

is received. The new breakpoint is set only after the successful input of the breakpoint address.

A user who wishes to set a breakpoint in a program need only enter the program and then enter the breakpoint. Program execution should be started the first time through the use of the Go command. Program control will jump to the command mode of the monitor when the program reaches the breakpoint. The breakpoint is automatically removed by the monitor. The user may now examine the machine status through the use of the Register display command, or do anything normally available in the command mode of the monitor (this includes setting of another breakpoint).

Restarting the program after it has been interrupted due to a breakpoint is done with the Next command. Command format is 'N'. The command causes the machine status and program counter to be removed from the stack. Program execution continues with the address shown in a Register display done prior to the Next command.

User Program Termination

There are three common methods of terminating a user program. The first two are explicit jumps to points within the monitor. The third is an implicit jump to monitor utilizing the reserved restart command which the monitor thinks is a breakpoint.

Jumps to the monitor are most commonly done to the beginning of memory (0000H) or the NMI reset location (0066H). The former causes the loss of many of the machine's registers, but will cause

a reinitialization of the stack pointer. A jump to the NMI location causes all of the machine status (except the R and I registers) to be pushed onto the stack. If the stack pointer had been altered by the user program and no longer pointed to enough RAM to support the system monitor, the monitor will probably fail.

User programs may also be terminated with the use of the restart instruction reserved for breakpointing. Due to the nature of the algorithm, an 'FF' inserted at any location in a program will cause a jump to monitor where all of the machine registers will be saved. Attempts to execute the Next command ('N') will not allow the program to execute any instructions beyond the location containing the 'FF.'

Customization Notes

1. Relocation of the monitor on a system with different EPROM, communication port (ie. PORT2), and RAM locations is easy on the assembly level. RAMST must contain the first location of contiguous RAM. RAMEND contains the top of RAM sufficient for scratchpad and stack usage. PORT2 contains the parallel port address used for communication. The above three parameters may be changed by changing their equates at the beginning of the MONIT (version 1.91) assembly listing, and reassembling (and loading) of the new version of the monitor.

2. If the restart locations are not available then each instance of a use must be changed to the associated CALL instruction. The Breakpoint insert routine and supporting sections of code (eg. the

three instructions at RESET) will be inoperable.

3. The cursor may be changed by changing the associated equate at the beginning of the MONIT (version 1.91) assembly listing, and reassembly (and loading) of the new version of the monitor.

4. UART modifications (in terms of system clock frequency and baud rate) may be done with the use of the following equation:

$$T = 65539.619 - F / (21 * B)$$

where T = Decimal number in the CONST table (+/-1),

B = Baud rate of data transfer (bits / second),

and F = System clock frequency (hertz).

With the use of the above equation the system clock rate may be changed and data transfer maintained.

The hex value obtained in the above equation must be loaded into the appropriate location in the CONST table in the assembly listing (see Tables B-5a and B-5b) for the present configuration. The location corresponding to a D4-D3-D2-D1 value of "d" is Lc'n = CONST + 2 * d. For example, the value d = 1101B = 0DH corresponds to a location (with CONST = 007DH) of 007D + 2 * D = 0097H.

5. If a system has a hardware UART and the user wishes to incorporate this into the NSC800 monitor few additions to the monitor need be made. {This modification has been done for the NSC800 monitor emulator: see Appendix E.} The programmer must write a routine to transmit a single character in accumulator A without disturbing any of the standard register set. Name this routine XMITER and add it to the assembly listing of the NSC800 monitor. Write a routine to get a single character and place it

in accumulator A without disturbing any of the standard register set. At the end of this second routine include the following lines of code:

```
PUSH  AF
CALL  XMITER
POP   AF.
```

Name this second routine RECEIV and append it to the NSC800 monitor assembly listing.

Much of the old interface routines need to be deleted. The old XMITER, RECEIV, and LOOKUP should be deleted. Also, the table CONST need not be retained.

6. Use of the existing software UART with an incomplete (or not-completely responsive) handshake is easily done. An example of this is the NSC800 <=> NorthStar Horizon interface. The reception of data by the NorthStar is not completely responsive.

For a not completely responsive transmission by the NSC800 connect J28 pin 11 to -9.25 Volts. For a not completely responsive transmission by the terminal connect J28 pin 4 to +9.25 Volts. For both conditions to occur at the same time, make both connections. {Warning: Less than completely responsive data transmissions by either the NSC800 or the terminal may result in lost data.}

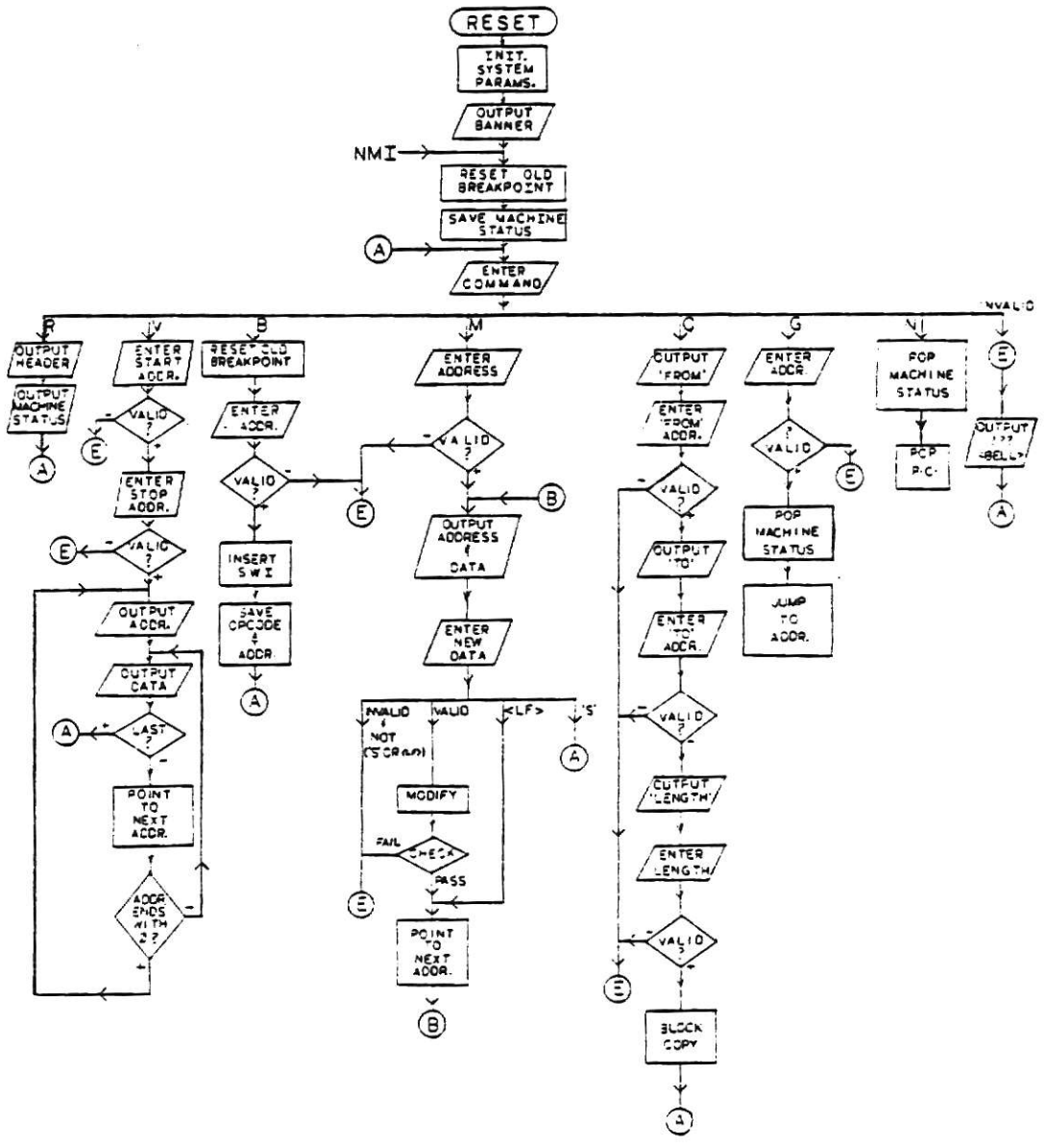
The connections described cause the NSC800 to respond to its own expectations. When the NSC800 tells the terminal that it is ready to receive or send data, the above changes force the terminal to tell the NSC800 that it will send or receive data.

Such a condition may lead to processor deadlock.

Lost Processor

If at any time the NSC800 either halts (red status LED on the cpu board is lighted) or appears to be lost, the user is advised to press the reset button on the processor board. This will cause the monitor to reinitialize and the banner to be displayed. If this is an unacceptable alternative, the monitor has been designed so that a NMI will perform exactly as if a software interruption ('FF') had occurred. All registers (including the program counter) will be stored on the stack. Hardware to implement the NMI has not been included with this report. More information on how to implement a NMI can be found in the specification sheets on the NSC800[11].

NSC800 MONITOR FLOWCHART

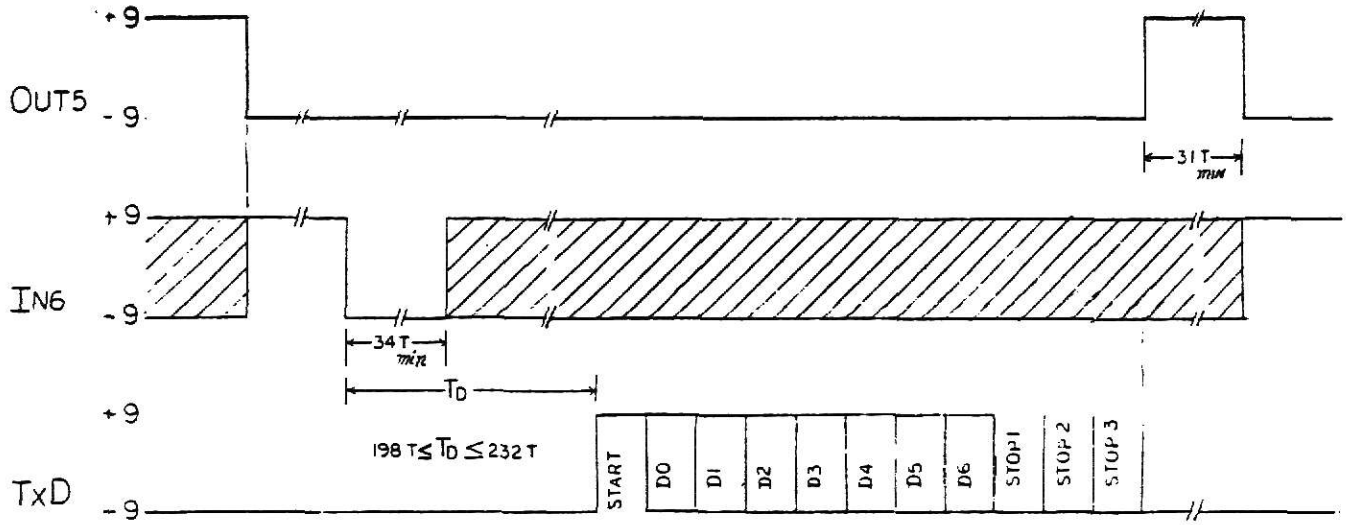


NOTE: A BREAKPOINT WILL START MONITOR EXECUTION AT THE NMI ENTRANCE POINT.

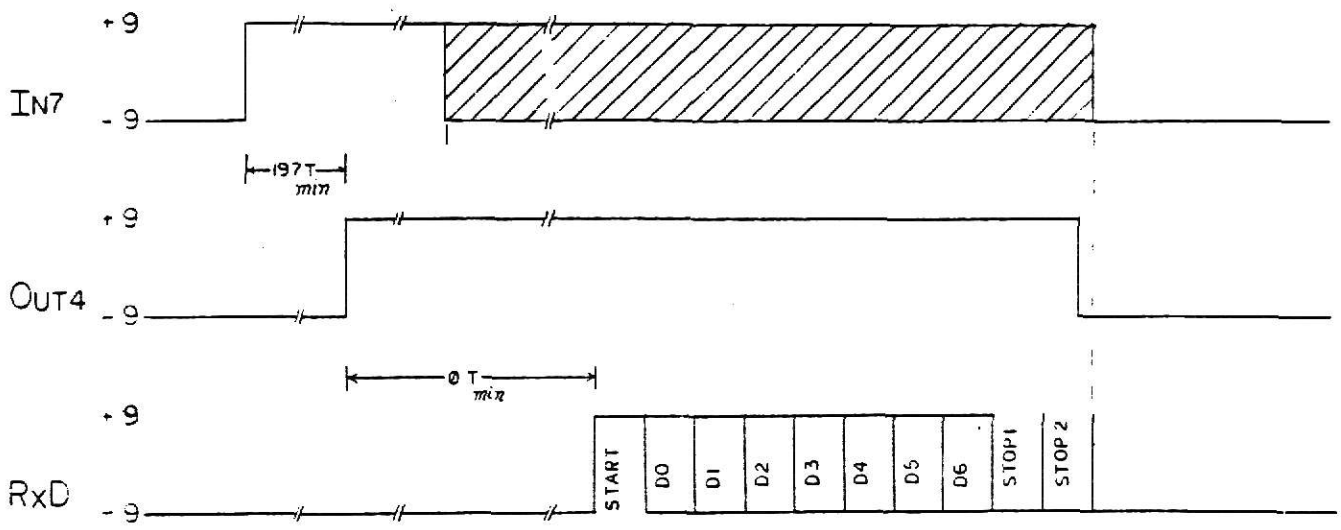
FIGURE B-3.

SOFTWARE UART
TIMING DIAGRAMS

TRANSMIT



RECEIVE



NOTE: ▨ = DON'T CARE. T=(SYSCLK.)$^{-1}$

FIGURE B-4.

Sample Run*

```
{Author's note:  All user inputs appear underlined.}
[reset]
NSC800 Monitor Version 1.91
Kansas State University

>r???[beep]                                {Lower case letters are invalid.}
>R                                         {Upper case works: register view.}
SP   A'F' B'C' D'E' H'L' A F  B C  D E  H L  IX   IY   PC
07E3 0200 00FF 0001 0000 1A4D 0000 0001 00E4 5555 5555 FFFD
>V0003,0025[cr]                            {View a block of memory}

0003 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF C3 18 04 0A 0D BE FF FF
0020 C3 20 04 FF FF FF
>V003,0025[cr]                             {View a block of memory}

0003 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF C3 18 04 0A 0D BE FF FF
0020 C3 20 04 FF FF FF
>V03,0025[cr]                              {View a block of memory}

0003 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF C3 18 04 0A 0D BE FF FF
0020 C3 20 04 FF FF FF
>V3,0025[cr]                               {View a block of memory}

0003 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF C3 18 04 0A 0D BE FF FF
0020 C3 20 04 FF FF FF
>V,0025[cr]                                {View a block of memory}

0000 31 F7 07 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF C3 18 04 0A 0D BE FF FF
0020 C3 20 04 FF FF FF
>V0000,025[cr]???[beep]                    {Insufficient ending address => error.}
>M0400[cr]                                 {Modify memory starting at 0400H.}
0400>FD 3E[cr]                  {LD A,}    {User input}
0401>FF [lf]                    {0FFH}     {<lf> => leave contents undisturbed.}
0402>FF 0123456789ABCDEFC3[cr]  {JP}       {Only the last two nibbles are kept.}
0403>FD 00[cr]                  {0066 = RESET}
0404>FF 66[cr]
0405>FF S                                  {Stop Memory modify mode.}
>V0402,0402[cr]                            {Verify that only the last two remain.}

0402 C3
```

* Nonprintable characters appear in brackets, eg. [cr].

```
>B0400[cr]                              {Set a breakpoint at 0400H.}

>G0400[cr]                              {Start the program at 0400H.}

>R                                      {Check registers of halted program.}
SP   A'F' B'C' D'E' H'L' A F  B C  D E  H L  IX   IY   PC
07E1 0200 00FF 0001 0000 1A4D 0000 0001 00E4 5555 5555 0400
>B0402[cr]                              {Set the next break point.}

>N                                      {Restart with the Next instruction.}
>R                                      {Check registers of halted program.}
SP   A'F' B'C' D'E' H'L' A F  B C  D E  H L  IX   IY   PC
07E1 0200 00FF 0001 0000 1A4D 0000 0001 00E4 5555 5555 0402
>N                                      {Restart with the Next instruction.}
>V410,041F[cr]                          {View a block of memory.}

0410 FF FD FF FF FF FF FD FF FD FF FF FF FD FD FF FF
>C                                      {Copy a block of memory.}
From? 0000[cr]                          {Source address.}
To? 0410[cr]                            {Destination address.}
Length? 0005[cr]                        {String length.}

>V0410,040F[cr]                         {View the copied area.}

0410 31 F7 07 FB C3 FF FD FF FD FF FF FF FD FD FF FF
>C                                      {Demonstrate errors on Copy input.}
From? 123[cr]
???[beep]
>C
From? 1234[cr]
To? 123[cr]
???[beep]
>C
From? 1234[cr]
To? 1234[cr]
Length? 123[cr]
???[beep]
>Q???[beep]                             {Invalid input.}
>G0123456789ABCDEF0000[cr]              {Cold start the monitor.}
NSC800 Monitor Version 1.91
Kansas State University

>D???[beep]                             {Invalid input.}
>W???[beep]                             {Invalid input.}
>

{End of Sample Run.}
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0001         NAME    MONIT91
                0002         PSECT   ABS
                0004 ;*********************************************************
                0005 ;*  NSC800 <=> ADM-3A SYSTEM MONITOR                     *
                0006 ;*                                                       *
                0007 ;*  PROGRAMMER:   Dwight W. Gordon          4-21-82      *
                0008 ;*                Kansas State University                *
                0009 ;*                Department of Electrical Engineering   *
                0010 ;*                Seaton Hall                            *
                0011 ;*                Manhattan, Kansas 66506                *
                0012 ;*********************************************************
                0013 ;*     This program has been designed as a minimal       *
                0014 ;* monitor for the newly developed NSC800 uP.            *
                0015 ;*     This program has been somewhat optimized, but     *
                0016 ;* use it at your own risk.                              *
                0017 ;*********************************************************
                0018 ;
                0019 ;VARIABLES USED BY THIS PROGRAM:
                0020 ;
>1C00           0021 PORT2  EQU   1C00H       ;INPUT/OUTPUT PORT
>0400           0022 RAMST  EQU   0400H       ;STARTING LOCATION OF SYSTEM RAM
>0800           0023 RAMEND EQU   0800H       ;END OF SYSTEM RAM + 1
>07FD           0024 SCRTCH EQU   RAMEND-3    ;AREA FOR ASYNC COMM'CATION CONST
>07FB           0025 FIX    EQU   SCRTCH-2    ;SCRATCHPAD MEMORY FOR MONITOR
>07FA           0026 USED   EQU   FIX-1       ;USED BY Go ROUTINE
>07F8           0027 BREAKA EQU   USED-2      ;BREAKPOINT ADDRESS
>07F7           0028 CODE   EQU   BREAKA-1    ;BREAKPOINT CODE ADDRESS
>07F7           0029 STACK  EQU   CODE        ;MONITOR BOTTOM OF STACK
>003E           0030 CURSOR EQU   3EH         ;">"
>0008           0031 CMDLEN EQU   08H         ;NUMBER OF COMMANDS ACCEPTED
>000A           0032 LF     EQU   0AH         ;LINE FEED
>000D           0033 CR     EQU   0DH         ;CARRIAGE RETURN
>0007           0034 BELL   EQU   07H         ;BELL
>0020           0035 SPACE  EQU   20H         ;ASCII SPACE
>008D           0036 CREND  EQU   CR+80H      ;END OF OUTPUT CR DELIMITER
>00BE           0037 CUREND EQU   CURSOR+80H  ;END OF OUTPUT CURSOR DELIMITER
>0087           0038 BELEND EQU   BELL+80H    ;END OF OUTPUT BELL DELIMITER
>008A           0039 LFEND  EQU   LF+80H      ;END OF OUTPUT LF DELIMITER
>00A0           0040 SPCEND EQU   SPACE+80H   ;END OF OUTPUT SPACE DELIMITER
                0041 ;*********************************************************
                0042 ;*********************************************************
>0000           0043        ORG   0000H
0000  31F707     0044        LD    SP,STACK    ;INITIALIZE MONITOR STACK
0003  FB         0045        EI                ;ENABLE INTERRUPTS
0004  C3E500     0046        JP    MAIN        ;BEGIN EXECUTION OF MAIN PROGRAM
>0008           0047        ORG   0008H
0008  C30804     0048        JP    $+RAMST     ;CHANGE RESET & INTERRUPT RESTART
>0010           0049        ORG   0010H       ; LOCATIONS TO RAM BY ADDING JUMP
0010  C31004     0050        JP    $+RAMST     ; OFFSET OF RAMST TO RESPECTIVE
>0018           0051        ORG   0018H       ; RESTART LOCATION
0018  C31804     0052        JP    $+RAMST
>0020           0053        ORG   0020H
```

```
0020  C32004    0054        JP    $+RAMST
>002C             0055        ORG   002CH
002C  C32C04    0056        JP    $+RAMST
>0034             0057        ORG   0034H
0034  C33404    0058        JP    $+RAMST
>003C             0059        ORG   003CH
003C  C33C04    0060        JP    $+RAMST
>0028             0061        ORG   0028H     ;RESERVED FOR XMITER ROUTINE
0028  C3F802    0062        JP    XMITER
>0030             0063        ORG   0030H     ;RESERVED FOR OUTPUT ROUTINE
0030  C37E02    0064        JP    OUTPUT
>0038             0065        ORG   0038H     ;RESERVED FOR BREAKPOINTING
0038  C3F100    0066        JP    RESET
                  0067 ;
                  0068 ;#########################################################
                  0069 ;      ERROR MESSAGE
                  0070 ;
>000B             0071        ORG   000BH
000B  3F3F3F    0072 ERRMES DEFM  '???'
000E  87        0073        DEFB  BELEND
                  0074 ;
                  0075 ;#########################################################
                  0076 ;      MESSAGE USED BY COPY ROUTINE
                  0077 ;
>0013             0078        ORG   0013H
0013  546F3F    0079 TO     DEFM  'To?'
0016  A0        0080        DEFB  SPCEND
                  0081 ;
                  0082 ;#########################################################
                  0083 ;      NEWLINE DATA WITH CURSOR PROMPT
                  0084 ;
>001B             0085        ORG   001BH
001B  0A0DBE    0086 LFCRCU DEFB  LF,CR,CUREND
                  0087 ;
                  0088 ;*********************************************************
                  0089 ;          ASCII <=> BINARY LOOKUP TABLE                *
                  0090 ;*********************************************************
>0040             0091        ORG   0040H
0040  30313233  0092 ASCII  DEFM  '0123456789ABCDEF'
      34353637
      38394142
      43444546
                  0093 ;
                  0094 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                  0095 ;      DATA TRANSFER ERROR MESSAGE
                  0096 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
0050  3130      0097 ERRTEN DEFM  '10'
0052  2D53544F  0098 ERRTIN DEFM  '-STOP BIT EXPECTED'
      50204249
      54204558
      50454354
      4544

0064  87        0099        DEFB  BELEND
```

```
 ADDR  CODE      STMT SOURCE STATEMENT

                 0100 ;
                 0101 ;#########################################################
                 0102 ;################## NMI RESET OFFSET #####################
                 0103 ;#########################################################
>0066             0104        ORG   0066H
0066  C3F100      0105        JP    RESET
                 0106 ;
                 0107 ;#########################################################
                 0108 ;      MORE DATA FOR COPY ROUTINE
                 0109 ;
0069  0A0D       0110 FROM   DEFB  LF,CR
006B  46726F6D   0111        DEFM  'From?'
      3F
0070  A0         0112        DEFB  SPCEND
0071  4C656E67   0113 LENGTH DEFM  'Length?'
      74683F
0078  A0         0114        DEFB  SPCEND
                 0115 ;
                 0116 ;*********************************************************
                 0117 ;*   ASYNCHRONOUS COMMUNICATION LOOKUP TABLE CONSTANTS   *
                 0118 ;*=======================================================*
                 0119 ;*     D2 D1  MEANING:   CLOCK FREQUENCY (SYSCLK)        *
                 0120 ;*     -- --  -------------------------------------------*
                 0121 ;*     0  0    1.0 MHz.                                  *
                 0122 ;*     0  1    2.0 MHz.                                  *
                 0123 ;*     1  0    2.5 MHz.                                  *
                 0124 ;*     1  1    4.0 MHz.                                  *
                 0125 ;*=======================================================*
                 0126 ;*     D4 D3  MEANING:   BAUD RATE OF TRANSFER (I/O)     *
                 0127 ;*     -- --  -------------------------------------------*
                 0128 ;*     0  0     300 Baud                                 *
                 0129 ;*     0  1     600 Baud                                 *
                 0130 ;*     1  0    1200 Baud                                 *
                 0131 ;*     1  1    2400 Baud                                 *
                 0132 ;*:::::::::::::::::::::::::::::::::::::::::::::::::::::::*
                 0133 ;* EQUATION FOR DETERMINING THE CONSTANTS=>               *
                 0134 ;*                                                        *
                 0135 ;*       T = 65539.619 - F / (21 * B)                     *
                 0136 ;*                                                        *
                 0137 ;*   T = DECIMAL NUMBER IN THE FOLLOWING TABLE (+/- 1).   *
                 0138 ;*   B = BAUD RATE OF DATA TRANSFER (BITS/SECOND).        *
                 0139 ;*   F = CLOCK FREQUENCY (HERTZ).                         *
                 0140 ;*********************************************************
0079  65FFC6FE   0141 CONST  DEFW  65381D,65222D,65143D,64905D
      77FE89FD
0081  B4FF65FF   0142        DEFW  65460D,65381D,65341D,65222D
      3DFFC6FE
0089  DCFFB4FF   0143        DEFW  65500D,65460D,65440D,65381D
      A0FF65FF
0091  F0FFDCFF   0144        DEFW  65520D,65500D,65490D,65460D
      D2FFB4FF
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0145 ;
                0146 ;*******************************************************
                0147 ;*     COMMAND DECODING AND LOOKUP ADDRESS TABLE        *
                0148 ;*******************************************************
0099  4D        0149 CMDTBL DEFM  'M'         ;view and alter Memory
009A  2801      0150        DEFW MEMCHG
009C  47        0151        DEFM  'G'         ;Go a user defined program
009D  6401      0152        DEFW GOPROG
009F  56        0153        DEFM  'V'         ;View a block of memory
00A0  2302      0154        DEFW MEMVUE
00A2  52        0155        DEFM  'R'         ;view existing Register values
00A3  BD01      0156        DEFW REGLOK
00A5  43        0157        DEFM  'C'         ;Copy a block of data
00A6  E101      0158        DEFW  COPY
00A8  4E        0159        DEFM  'N'         ;Next command from software itpt.
00A9  1402      0160        DEFW  RESTAR
00AB  42        0161        DEFM  'B'         ;set a single Breakpoint
00AC  6502      0162        DEFW  BRKPT
00AE  0D        0163        DEFB  CR          ;CR as terminal (NOP)
                0164 ;
                0165 ;************WAKEUP BANNER MESSAGE***********************
00AF  4E534338  0166 BANNER DEFM  'NSC800 Monitor Version 1.91'
      3030204D
      6F6E6974
      6F722056
      65727369
      6F6E2031
      2E3931
00CA  0A0D      0167        DEFB  LF,CR
00CC  4B616E73  0168        DEFM  'Kansas State University'
      61732053
      74617465
      20556E69
      76657273
      697479
00E3  0A8D      0169 NEWLIN DEFB  LF,CREND
                0170 ;
                0171 ;*******************************************************
                0172 ;*******************************************************
                0173 ;**     BEGINNING OF THE MAIN PROGRAM                   *
                0174 ;**-------------------------------------------------------*
                0175 ;**  WARNING:  NMI will cause a restart at this location*
                0176 ;**   This has been done in order that a hardware device*
                0177 ;**   might be interfaced between the ADM-3A and the NSC*
                0178 ;**     800 as a method of retrieving "lost" programs   *
                0179 ;**   without distroying any of their registers.        *
                0180 ;*******************************************************
                0181 ;*******************************************************
                0182 ;*  WARNING:  Routines invoked from the main program are
                0183 ;*     NOT subroutines and may NOT be used as such by the
                0184 ;*     user.  There are, however, many subroutines which
                0185 ;*     may be called by the user.  Care should be taken
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0186 ;*     to insure that registers disturbed by a called
                0187 ;*     subroutine, which contain information prior to the
                0188 ;*     call, are pushed in the user main program prior to
                0189 ;*     calling the desired subroutine.
                0190 ;*
                0191 ;
00E5  3EFF      0192 MAIN   LD    A,0FFH      ;INITIALIZE I/O PORT LINES
00E7  47        0193        LD    B,A         ;ALLOW TIME FOR LINES TO SETTLE
00E8  10FE      0194 WAKEUP DJNZ  WAKEUP
00EA  CD8702    0195        CALL  REMOVE      ;REMOVE PENDING BREAKPOINTS
00ED  21AF00    0196        LD    HL,BANNER   ;OUTPUT BANNER MESSAGE
00F0  F7        0197        RST   30H
00F1  E3        0198 RESET  EX    (SP),HL     ;CLEAN UP BREAKPOINT POINTER
00F2  2B        0199        DEC   HL
00F3  E3        0200        EX    (SP),HL
00F4  FDE5      0201        PUSH  IY          ;PUSH MAIN REGISTER SET
00F6  DDE5      0202        PUSH  IX
00F8  E5        0203        PUSH  HL
00F9  D5        0204        PUSH  DE
00FA  C5        0205        PUSH  BC
00FB  F5        0206        PUSH  AF
00FC  08        0207        EX    AF,AF'      ;GET ALTERNATE REGISTER SET
00FD  D9        0208        EXX               ;  AND PUSH SAME
00FE  E5        0209        PUSH  HL
00FF  D5        0210        PUSH  DE
0100  C5        0211        PUSH  BC
0101  F5        0212        PUSH  AF
0102  CD8702    0213        CALL  REMOVE      ;CLEAN UP BREAKPOINT
0105  211B00    0214 OK     LD    HL,LFCRCU   ;OUTPUT CURSOR
0108  F7        0215        RST   30H
0109  CD4803    0216        CALL  RECEIV      ;GET COMMAND
010C  219700    0217        LD    HL,CMDTBL-2   ;GET POINTER TO CMD TABLE-2
010F  010800    0218        LD    BC,CMDLEN     ;GET LENGTH OF SAME
0112  23        0219 NXTCMD INC   HL          ;POINT TO COMMAND NAME
0113  23        0220        INC   HL
0114  EDA1      0221        CPI               ;COMPARE ENTERED-TABLE COMMANDS
0116  E22001    0222        JP    PO,LASTCD   ;LAST CMD? +=> CR & -> LASTCommanD
0119  20F7      0223        JR    NZ,NXTCMD   ;MATCH? -=> NeXTCoMmanD
011B  5E        0224        LD    E,(HL)      ;ELSE: POINT TO LOCATION OF GO
011C  23        0225        INC   HL          ; AND START EXECUTION OF SAME
011D  56        0226        LD    D,(HL)
011E  EB        0227        EX    DE,HL
011F  E9        0228        JP    (HL)
0120  28E3      0229 LASTCD JR    Z,OK        ;IF A CR => GO TO OK
                0230 ;*************************************************************
                0231 ;      EXECUTION OF THE FOLLOWING MODULE WILL ONLY OCCUR
                0232 ;      UPON DETECTION OF AN INVALID COMMAND
                0233 ;*************************************************************
0122  210B00    0234 NOCMD  LD    HL,ERRMES   ;START OF ERROR MESSAGE
0125  F7        0235        RST   30H
0126  18DD      0236        JR    OK          ;RETURN TO INPUT MODE
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0237 ;
                0238 ;*****************************************************
                0239 ;      MEMCHG ALLOWS THE USER TO VIEW A BLOCK OF MEMORY
                0240 ; AND ALTER ITS CONTENTS.  DETECTION OF A "S" TERMINATES
                0241 ; THE ROUTINE.  DETECTION OF A LF CAUSES THE POINTER TO
                0242 ; ADVANCE TO THE NEXT MEMORY LOCATION WITHOUT DISTURBING
                0243 ; THE CONTENTS OF THE PRESENTLY INDEXED LOCATION.
                0244 ;*****************************************************
0128  CDAE02    0245 MEMCHG CALL  HILOW4     ;GET STARTING ADDRESS
012B  DA2201    0246        JP    C,NOCMD    ;ERROR => NOCoMmanD
012E  2AFB07    0247        LD    HL,(FIX)   ;GET STARTING ADDRESS
0131  CD9302    0248 NXTMEM CALL  OUT4       ;DISPLAY ADDRESS
0134  3E3E      0249        LD    A,CURSOR   ;OUT CURSOR
0136  EF        0250        RST   28H
0137  7E        0251        LD    A,(HL)     ;GET MEMORY CONTENTS
0138  32FC07    0252        LD    (FIX+1),A  ;PLACE IN STORAGE
013B  0602      0253        LD    B,02H      ;TWO NIBBLES OUTPUTTED
013D  CD9802    0254        CALL  OUT2
0140  3E20      0255        LD    A,SPACE    ;SPACE BETWEEN OLD AND NEW VALUES
0142  EF        0256        RST   28H
0143  1E02      0257        LD    E,02H      ;TWO NIBBLES TO BE INPUT
0145  CDB002    0258        CALL  HILOW2     ;GET TWO NIBBLES
0148  300C      0259        JR    NC,PNOCMD  ;NO ERROR => PNOCMD
014A  FE53      0260        CP    53H        ;ON DETECTION OF AN "S" => STOP
014C  CA0501    0261        JP    Z,OK
014F  FE0A      0262        CP    LF         ;ON DETECTION OF A LF => CONT.
0151  280B      0263        JR    Z,LFINPT
0153  C32201    0264        JP    NOCMD      ;  ELSE ERROR
0156  3AFB07    0265 PNOCMD LD    A,(FIX)    ;GET DATA
0159  77        0266        LD    (HL),A     ;STORE
015A  BE        0267        CP    (HL)       ;CHECK
015B  C22201    0268        JP    NZ,NOCMD   ;NOCHECK => ERROR
015E  23        0269 LFINPT INC   HL         ;POINT TO NEXT
015F  3E0D      0270        LD    A,CR       ;OUTPUT CR
0161  EF        0271        RST   28H
0162  18CD      0272        JR    NXTMEM     ;CONTINUE
                0273 ;
                0274 ;*****************************************************
                0275 ;      THE FOLLOWING ROUTINE WILL INPUT A FOUR NIBBLE
                0276 ; ADDRESS AND START PROGRAM EXECUTION AT SAID ADDRESS
                0277 ; UPON DETECTION OF A CR.
                0278 ;*****************************************************
0164  CDAE02    0279 GOPROG CALL  HILOW4     ;GET ADDRESS
0167  DA2201    0280        JP    C,NOCMD    ;ERROR => NOCoMmanD
016A  3EC3      0281        LD    A,0C3H     ;Z80 JUMP COMMAND
016C  32FA07    0282        LD    (FIX-1),A  ;SET UP JUMP COMMAND TO SPECIFIED
                0283 ;                         ADDRESS
016F  F1        0284        POP   AF         ;POP ALT REG SET
0170  C1        0285        POP   BC
0171  D1        0286        POP   DE
0172  E1        0287        POP   HL
0173  08        0288        EX    AF,AF      ;GET MAIN REG SET
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0174  D9        0289        EXX
0175  F1        0290        POP   AF          ;POP SAME
0176  C1        0291        POP   BC
0177  D1        0292        POP   DE
0178  E1        0293        POP   HL
0179  DDE1      0294        POP   IX
017B  FDE1      0295        POP   IY
017D  C3FA07    0296        JP    FIX-1       ;AFFECTS A JUMP TO THE LOCATION
                0297 ;                          SPECIFIED BY FIX AND FIX+1.
                0298 ;
                0299 ;*********************************************************
                0300 ;      REGLOK WILL DISPLAY THE PUSHED REGISTER SET
                0301 ;*********************************************************
                0302 ;---------------------------------------------------------
                0303 ;                          DATA FOR THE ROUTINE
                0304 ;---------------------------------------------------------
0180  0A0D      0305 REGHED DEFB  LF,CR         ;LF,CR
0182  53502020  0306        DEFM  'SP   A''F'' B''C'' D''E'' H''L'' '
      20412746
      27204227
      43272044
      27452720
      48274C27
      20
019B  41204620  0307        DEFM  'A F  B C  D E  H L'
      20422043
      20204420
      45202048
      204C
01AD  20204958  0308        DEFM  '  IX   IY   PC'
      20202049
      59202020
      5043
01BB  0A8D      0309        DEFB  LF,CREND      ;LF, CR, DELIMITER
                0310 ;//////////////////////////////////////////////////////
01BD  ED73FB07  0311 REGLOK LD    (FIX),SP      ;SP TO TEMP STORAGE
01C1  218001    0312        LD    HL,REGHED     ;START OF REGISTER HEADER
01C4  F7        0313        RST   30H           ;OUTPUT HEADER
01C5  2AFB07    0314        LD    HL,(FIX)      ;INITIALIZE POINTER TO PUSHED REGS
01C8  060C      0315        LD    B,0CH         ;NUMBER OF REGISTERS
01CA  C5        0316 REGAGN PUSH  BC            ;SAVE COUNTER
01CB  CD9602    0317        CALL  OUT4A         ;OUTPUT FOUR NIBBLES
01CE  3E20      0318        LD    A,SPACE       ;SPACE
01D0  EF        0319        RST   28H
01D1  7E        0320        LD    A,(HL)        ;MOVE DOUBLE WORD TO FIX
01D2  32FB07    0321        LD    (FIX),A
01D5  23        0322        INC   HL
01D6  7E        0323        LD    A,(HL)
01D7  32FC07    0324        LD    (FIX+1),A
01DA  23        0325        INC   HL            ;POINT TO NEXT  REG PAIR
01DB  C1        0326        POP   BC            ;RECOVER COUNTER
01DC  10EC      0327        DJNZ  REGAGN        ;MORE => REGAGN
```

```
ADDR  CODE      STMT SOURCE STATEMENT

01DE  C30501    0328        JP    OK          ;ELSE RETURN
                0329 ;
                0330 ;*********************************************************
                0331 ;       BLOCK COPY SUBPROGRAM COPIES A BLOCK OF CODE
                0332 ; FROM THE 'FROM' ADDRESS TO THE 'TO' ADDRESS FOR A
                0333 ; TOTAL OF 'LENGTH' (HEX) BYTES OF DATA.  NO VERIFICATION
                0334 ; IS DONE.
                0335 ;*********************************************************
01E1  216900    0336 COPY   LD       HL,FROM
01E4  F7        0337        RST      30H         ;OUTPUT ROUTINE 'FROM'
01E5  CDAE02    0338        CALL     HILOW4      ;GET STARTING ADDRESS
01E8  3827      0339        JR       C,NOCMD0    ;ERROR => NOCoMmanD0
01EA  2AFB07    0340        LD       HL,(FIX)    ;GET AND SAVE DATA ON STACK
01ED  E5        0341        PUSH     HL
01EE  211300    0342        LD       HL,TO
01F1  F7        0343        RST      30H         ;OUTPUT ROUTINE 'TO'
01F2  CDAE02    0344        CALL     HILOW4      ;GET DESTINATION ADDRESS
01F5  3819      0345        JR       C,NOCMD1    ;ERROR => NOCoMmanD1
01F7  2AFB07    0346        LD       HL,(FIX)    ;GET AND SAVE DEST. ON STACK
01FA  E5        0347        PUSH     HL
01FB  217100    0348        LD       HL,LENGTH
01FE  F7        0349        RST      30H         ;OUTPUT ROUTINE 'LENGTH'
01FF  CDAE02    0350        CALL     HILOW4      ;GET LENGTH
0202  380B      0351        JR       C,NOCMD2    ;ERROR => NOCoMmanD2
0204  ED4BFB07  0352        LD       BC,(FIX)    ;GET LENGTH
0208  D1        0353        POP      DE          ;DESTINATION
0209  E1        0354        POP      HL          ;START
020A  EDB0      0355        LDIR                 ;MOVE
020C  C30501    0356        JP       OK          ;RETURN TO MAIN ROUTINE
                0357 ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                0358 ;       ERROR HANDLING ROUTINE TO AVOID CREEPING STACK
                0359 ;
020F  E1        0360 NOCMD2 POP      HL          ;POP STACK TO ADJUST STACK PTR.
0210  E1        0361 NOCMD1 POP      HL          ; AND JUMP TO NOCoMmanD
0211  C32201    0362 NOCMD0 JP       NOCMD
                0363 ;
                0364 ;*********************************************************
                0365 ;*      SOFTWARE INTERRUPT RESTART ROUTINE PULLS ALL     *
                0366 ;* REGISTERS FROM STACK (INCLUDING PROGRAM COUNTER).     *
                0367 ;*********************************************************
0214  F1        0368 RESTAR POP  AF
0215  C1        0369        POP  BC
0216  D1        0370        POP  DE
0217  E1        0371        POP  HL
0218  08        0372        EX   AF,AF'
0219  D9        0373        EXX
021A  F1        0374        POP  AF
021B  C1        0375        POP  BC
021C  D1        0376        POP  DE
021D  E1        0377        POP  HL
021E  DDE1      0378        POP  IX
0220  FDE1      0379        POP  IY
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0222  C9        0380        RET
                0381 ;
                0382 ;*****************************************************
                0383 ;      MEMVUE WILL CAUSE A STARTING ADDRESS TO BE INPUT,
                0384 ; FOLLOWED BY A COMMA (','), AND CONCLUDED BY AN ENDING
                0385 ; ADDRESS.  THE ROUTINE WILL OUTPUT THE CONTENTS OF MEMORY
                0386 ; IN BYTE FORM FROM THE BEGINNING OF THE BLOCK TO THE END
                0387 ; OF THE BLOCK.
                0388 ;     AFTER THE FIRST CHARACTER IS OUTPUT, THE ROUTINE
                0389 ; SEARCHES FOR AN ADDRESS WITH A ZERO AS THE LS NIBBLE,
                0390 ; AND A NEW LINE WITH THE NEXT ADDRESS TO BE OUTPUT IS
                0391 ; OUTPUTTED.
                0392 ;
                0393 ; SAMPLE OUTPUT:
                0394 ;>V000E,0012<CR>  {NOTE: THIS LINE WAS INPUT FROM ADM-3A)
                0395 ;                    {BLANK LINE}
                0396 ; 000E 01 03
                0397 ; 0010 3C 66 44
                0398 ; >
                0399 ;
                0400 ;*****************************************************
0223  210000    0401 MEMVUE LD    HL,0000H    ;INITIALIZE BEGINNING POINTER
0226  22FB07    0402        LD    (FIX),HL
0229  CDAE02    0403        CALL  HILOW4      ;GET STARTING ADDRESS
022C  FE2C      0404        CP    ','         ;HAS A COMMA BEEN SENT
022E  C22201    0405        JP    NZ,NOCMD    ; - => ERROR
0231  2AFB07    0406        LD    HL,(FIX)    ;SAVE STARTING ADDRESS
0234  CDAE02    0407        CALL  HILOW4      ;GET ENDING ADDRESS
0237  DA2201    0408        JP    C,NOCMD     ;ERROR? + => NOCoMmanD
023A  ED5BFB07  0409        LD    DE,(FIX)    ;GET ENDING ADDRESS
023E  13        0410        INC   DE          ;POINT TO END+1
023F  E5        0411 I0     PUSH  HL          ;SAVE CURRENT ADDRESS
0240  21E300    0412        LD    HL,NEWLIN   ;OUTPUT A NEW LINE
0243  F7        0413        RST   30H
0244  E1        0414        POP   HL          ;RECOVER CURRENT LINE
0245  CD9302    0415        CALL  OUT4        ;OUTPUT ADDRESS
0248  3E20      0416 I1     LD    A,' '       ;OUTPUT A SPACE
024A  EF        0417        RST   28H
024B  7E        0418        LD    A,(HL)      ;GET DATA
024C  32FC07    0419        LD    (FIX+1),A   ;STORE IN BUFFER
024F  0602      0420        LD    B,02H       ;INITIALIZE COUNTER
0251  CD9802    0421        CALL  OUT2        ;OUTPUT BYTE
0254  23        0422        INC   HL          ;POINT TO NEXT CHAR
0255  7D        0423        LD    A,L         ;LOW BYTE TO ACC
0256  BB        0424        CP    E           ;16 BIT COMPARE
0257  2005      0425        JR    NZ,NEXTDT   ;NOT SAME => CONTINUE
0259  7C        0426        LD    A,H         ;
025A  BA        0427        CP    D           ;
025B  CA0501    0428        JP    Z,OK        ;SAME => RETURN TO MAIN PROG.
025E  7D        0429 NEXTDT LD    A,L         ;LAST INPUT FOR LINE?
025F  E60F      0430        AND   0FH         ;LAST PER LINE? + => Z SET
0261  28DC      0431        JR    Z,I0        ; Z => I0
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0263  18E3      0432        JR    I1          ;ELSE NEXT
                0433 ;*******************************************************
                0434 ;*      THE FOLLOWING ROUTINE WILL CLEAR ANY PENDING
                0435 ;* BREAKPOINTS BY FORCING THEM TO THE EPROM ON PAGE ZERO,
                0436 ;* AND THEN SET ANOTHER BREAKPOINT.  TO CLEAR A PENDING
                0437 ;* BREAKPOINT TYPE THE COMMAND 'B' FOLLOWED BY ANY INVALID
                0438 ;* CHARACTER (NON HEX).  THE COMPUTER WILL RESPOND WITH
                0439 ;* THREE QUESTION MARKS AND A BEEP.
                0440 ;*******************************************************
0265  CD8702    0441 BRKPT  CALL  REMOVE      ;REMOVE PENDING BREAKPOINT
0268  CDAE02    0442        CALL  HILOW4      ;GET AN ADDRESS
026B  DA2201    0443        JP    C,NOCMD     ;ERROR => NOCoMmanD
026E  2AFB07    0444        LD    HL,(FIX)    ;GET ADDRESS
0271  7E        0445        LD    A,(HL)      ;GET CODE
0272  22F807    0446        LD    (BREAKA),HL ;STORE ADDRESS
0275  32F707    0447        LD    (CODE),A    ; AND CODE
0278  3EFF      0448        LD    A,0FFH      ;BREAKPOINT COMMAND
027A  77        0449        LD    (HL),A      ;SUBSTITUTE
027B  C30501    0450        JP    OK
                0451 ;
                0452 ;*******************************************************
                0453 ;      OUTPUT WILL TAKE THE ASCII DATA STARTING AT THE
                0454 ; INPUTTED VALUE OF THE HL REGISTER AND OUTPUT THIS DATA
                0455 ; TO THE TERMINAL.  TERMINATION WILL OCCUR WHEN THE
                0456 ; INDEXED DATA CONTAINS THE MOST SIG BIT (D7) SET.
                0457 ;*******************************************************
                0458 ;*                                                     *
                0459 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0460 ;*     INPUT- - - - - - D D  - - - - - - - - - -  *
                0461 ;*    OUTPUT- C U U U U C C  C  C  C  C  C  C  C  U  U  *
                0462 ;*                                                     *
                0463 ;*      FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'     *
                0464 ;*     INPUT- - - - - - - - - - - - - - - - - -     *
                0465 ;*    OUTPUT- C C C 0 C  C  0 1  C C C C C  C  C C     *
                0466 ;*                                                     *
                0467 ;*    MEMORY:       07FA 07FB 07FC 07FD 07FE          *
                0468 ;*     INPUT-        --   --   --   --   --           *
                0469 ;*    OUTPUT-        UU   UU   UU   CC   CC           *
                0470 ;*                                                     *
                0471 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0472 ;*                       -=DON'T CARE.                 *
                0473 ;*                                                     *
                0474 ;*******************************************************
027E  7E        0475 OUTPUT LD    A,(HL)
027F  F5        0476        PUSH  AF
0280  EF        0477        RST   28H
0281  F1        0478        POP   AF
0282  17        0479        RLA
0283  D8        0480        RET   C
0284  23        0481        INC   HL
0285  18F7      0482        JR    OUTPUT
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0483 ;
                0484 ;*********************************************************
                0485 ;*     REMOVE A BREAKPOINT BY FORCING THE CURRENT BREAKPT
                0486 ;* TO BE IN THE EPROM ON PAGE ZERO.
                0487 ;*********************************************************
                0488 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0489 ;*     INPUT- - - - - - - -  - - - - - - - - - -  *
                0490 ;*    OUTPUT- C U U U U C C  U  U  U  U  U  U  U  U  U  *
                0491 ;*                                                       *
                0492 ;*      FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'      *
                0493 ;*     INPUT- - - - - - - - - -  - - - - - - - - -       *
                0494 ;*    OUTPUT- C C C C C  C  C C  U U U U U  U  U U       *
                0495 ;*                                                       *
                0496 ;*    MEMORY: 07F7 07F8 07F9                             *
                0497 ;*     INPUT-  DD   DD   DD                              *
                0498 ;*    OUTPUT-  CC   UU   CC                              *
                0499 ;*                                                       *
                0500 ;*********************************************************
0287  2AF807      0501 REMOVE LD    HL,(BREAKA) ;CLEAR ANY PENDING BREAKPOINT
028A  3AF707      0502        LD    A,(CODE)
028D  77          0503        LD    (HL),A
028E  AF          0504        XOR   A           ;POINT BREAKPOINT TO EPROM
028F  32F907      0505        LD    (BREAKA+1),A
0292  C9          0506        RET
                0507 ;
                0508 ;*********************************************************
                0509 ;      THIS ROUTINE, IF ENTERED FROM THE FIRST STATEMENT
                0510 ; WILL CAUSE THE FOUR NIBBLES LOCATED IN THE HL REGISTER
                0511 ; TO BE ENCODED TO ASCII AND DISPLAYED AT THE TERMINAL.
                0512 ;     IF THE ROUTINE IS ENTERED AT THE SECOND+ STATEMENT,
                0513 ; THEN THE VALUE OF ACC-B (B=<4) IS USED TO DETERMINE
                0514 ; THE NUMBER OF OUTPUTTED NIBBLES AND THE DATA IS TAKEN
                0515 ; DIRECTLY FROM LOCATION FIX+1.
                0516 ;      ENTERING AT OUT4A IMPLIES THAT DATA IS IN (FIX) AS
                0517 ; OPPOSED TO HL UPON ENTRY.
                0518 ;*********************************************************
                0519 ;* OUT4:                                                 *
                0520 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0521 ;*     INPUT- - - - - - D D  - - - - - - - - - -  *
                0522 ;*    OUTPUT- C C U U U U U  C  C  C  C  C  C  C  U  U  *
                0523 ;*                                                       *
                0524 ;*      FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'      *
                0525 ;*     INPUT- - - - - - - - - -  - - - - - - - - -       *
                0526 ;*    OUTPUT- C 0 C C C  C  C C  C C C C C  C  C C       *
                0527 ;*                                                       *
                0528 ;*    MEMORY: 07FA 07FB 07FC 07FD 07FE                   *
                0529 ;*     INPUT-  --   --   --   --   --                    *
                0530 ;*    OUTPUT-  UU   CC   CC   CC   CC                    *
                0531 ;*                                                       *
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0532 ;* OUT4A: {NOTE CHANGES TO THE ABOVE}                    *
                0533 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0534 ;*     INPUT- - - - - - - - - - - - - - - - - - - -  *
                0535 ;*                                                      *
                0536 ;*    MEMORY: 07FA 07FB 07FC 07FD 07FE                  *
                0537 ;*     INPUT-  --   DD   DD   --   --                   *
                0538 ;*                                                      *
                0539 ;* OUT2:  {NOTE CHANGES TO OUT4A ABOVE}                 *
                0540 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0541 ;*     INPUT- - D - - - - - - - - - - - - - - - - -  *
                0542 ;*                                                      *
                0543 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0544 ;*                      -=DON'T CARE.                   *
                0545 ;*                                                      *
                0546 ;********************************************************
0293  22FB07    0547 OUT4   LD    (FIX),HL   ;DATA TO FIX
0296  0604      0548 OUT4A  LD    B,04H      ;NUMBER OF NIBBLES
0298  E5        0549 OUT2   PUSH  HL         ;SAVE MEMORY POINTER
0299  21FB07    0550 OUTCHR LD    HL,FIX     ;LOW DATA BYTE TO HL
029C  3E30      0551        LD    A,30H      ;ASCII ZERO
029E  ED6F      0552        RLD              ;NIBBLE ROTATE LOW BYTE
02A0  23        0553        INC   HL         ;POINT TO HIGH BYTE
02A1  ED6F      0554        RLD              ;NIBBLE ROTATE HIGH BYTE
02A3  FE3A      0555        CP    3AH        ;IS NUMBER GREATER THAN "9?"
02A5  3802      0556        JR    C,OUTDTA   ;NO => ADD NO CORRECTION
02A7  C607      0557        ADD   A,07H      ;ELSE ADD A CORRECTION TERM
02A9  EF        0558 OUTDTA RST   28H        ;OUT DATA
02AA  10ED      0559        DJNZ  OUTCHR     ;MORE? + => OUTCHR
02AC  E1        0560        POP   HL         ;RESTORE MEMORY POINTER
02AD  C9        0561        RET
                0562 ;
                0563 ;********************************************************
                0564 ;      THE FOLLOWING ROUTINE WILL OBTAIN A FOUR NIBBLE
                0565 ; ADDRESS FROM DATA INPUT FROM THE TERMINAL (IF ENTERED
                0566 ; FROM THE FIRST STATEMENT).  IF THE ROUTINE IS ENTERED
                0567 ; FROM THE SECOND STATEMENT WITH AN 'E' VALUE OF 02H,
                0568 ; THEN THE ROUTINE WILL COMPUTE ONLY A TWO NIBBLE HEX
                0569 ; VALUE.
                0570 ;      THE FOUR NIBBLE ADDRESS WILL BE STORED IN THE
                0571 ; LOCATION FIX (LSB) AND FIX+1 (MSB). THE TWO NIBBLE
                0572 ; VALUE WILL BE LOCATED IN LOCATION FIX+1.
                0573 ;      THE ROUTINE WILL KEEP ONLY THE LAST FOUR (TWO)
                0574 ; NIBBLES WHICH THE USER INPUTS FROM THE TERMINAL
                0575 ;      INSUFFICIENT NUMBER OF INPUT CHARACTERS, OR AN
                0576 ; INVALID HEX INPUT CAUSES THE ROUTINE TO TERMINATE WITH
                0577 ; THE CARRY BIT SET, ELSE THE CARRY BIT IS RESET.
                0578 ;      INPUT IS TERMINATED BY INPUTTING AN INVALID HEX
                0579 ; DIGIT (ERROR) OR A CR.
                0580 ;       THE LAST CHARACTER WHICH WAS INPUT IS LEFT IN THE
                0581 ; ACC-A UPON EXIT WITH AN ERROR DUE TO AN INVALID CHR.
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0582 ;*********************************************************
                0583 ;* HILOW4:                                               *
                0584 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0585 ;*      INPUT- - - - - - - -  - - - - - - - - - -  *
                0586 ;*     OUTPUT- D C C C C U U  C  C  C  C  C  C  C  U  U  *
                0587 ;*                                                       *
                0588 ;*       FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'     *
                0589 ;*      INPUT- - - - - -  -  - -  - - - - - -  -  - -     *
                0590 ;*     OUTPUT- C C C C C  C  C D  C C C C C  C  C C     *
                0591 ;*                                                       *
                0592 ;*     MEMORY: 07FA 07FB 07FC 07FD 07FE                  *
                0593 ;*      INPUT-  --   --   --   --   --                   *
                0594 ;*     OUTPUT-  UU   DD   DD   CC   CC                   *
                0595 ;*                                                       *
                0596 ;* HILOW2:  {NOTE CHANGES TO THE ABOVE}                 *
                0597 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0598 ;*      INPUT- - - - - D - -  - - - - - - - - - -  *
                0599 ;*                                                       *
                0600 ;*     MEMORY: 07FA 07FB 07FC 07FD 07FE                  *
                0601 ;*     OUTPUT-  UU   CC   DD   CC   CC                   *
                0602 ;*                                                       *
                0603 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0604 ;*                      -=DON'T CARE.                    *
                0605 ;*                                                       *
                0606 ;*********************************************************
02AE  1E00      0607 HILOW4 LD    E,00H       ;INITIALIZE FOR 4 NIBBLES
02B0  E5        0608 HILOW2 PUSH  HL          ;SAVE MEMORY POINTER
02B1  CD4803    0609 HILORP CALL  RECEIV      ;GET NIBBLE
02B4  FE0D      0610        CP    CR          ;END?
02B6  281C      0611        JR    Z,ENDHGH    ; + => ENDHGH
02B8  214000    0612        LD    HL,ASCII    ;START OF ASCII LOOKUP
02BB  011000    0613        LD    BC,10H      ;LENGTH OF SAME
02BE  EDA1      0614 NXTHGH CPI               ;LAST VALUE THAT OF TABLE?
02C0  E2C502    0615        JP    PO,OUTAHX   ;LAST ENTRY => OUTAHX
02C3  20F9      0616        JR    NZ,NXTHGH   ;VALUE? - => NXTHGH
                0617 ;EXECUTION OF THE FOLLOWING STATEMENT OCCURS IF EITHER
                0618 ; A MATCH IS FOUND (IN WHICH CASE THE JUMP IS REDUNDANTLY
                0619 ; FALSE), OR IF END OF THE TABLE IS FOUND (IN WHICH CASE
                0620 ; THE JUMP TESTS TO SEE IF THE INPUTTED CHARACTER IS "F"
                0621 ; OR IF IT IS NOT IN THE TABLE--OUTHER).
02C5  2017      0622 OUTAHX JR    NZ,OUTHER   ;IS THIS THE VALUE? - => OUTHER
02C7  2D        0623        DEC   L           ;ADJUST POINTER
02C8  7D        0624        LD    A,L         ;COMPUTE LOOKUP ADDRESS
02C9  21FB07    0625        LD    HL,FIX      ;POINTER TO LOW NIBBLE
02CC  ED6F      0626        RLD               ;ROTATE IN NIBBLE
02CE  23        0627        INC   HL          ;POINT TO HIGH BYTE
02CF  ED6F      0628        RLD               ;ROTATE IN NIBBLE
02D1  1C        0629        INC   E           ;ADJUST COUNTER
02D2  18DD      0630        JR    HILORP      ;CONTINUE INPUT
02D4  21E300    0631 ENDHGH LD    HL,NEWLIN   ;OUTPUT LF,CR
02D7  F7        0632        RST   30H
02D8  3EFC      0633        LD    A,0FCH      ;2'S COMP OF 0004H
```

```
ADDR  CODE      STMT SOURCE STATEMENT

02DA  83        0634        ADD   A,E         ;SEE IF ENOUGH NIBBLES INPUT
02DB  3F        0635        CCF               ;ADJUST CARRY FOR ERROR STATE
02DC  E1        0636        POP   HL          ;RECOVER MEMORY POINTER
02DD  C9        0637        RET
02DE  37        0638 OUTHER SCF               ;SET CARRY FOR ERROR
02DF  E1        0639        POP   HL          ;RECOVER MEMORY POINTER
02E0  C9        0640        RET
                0641 ;
                0642 ;*************************************************************
                0643 ;      SUBROUTINE TO CALCULATE LOOKUP TABLE CONSTANT AND
                0644 ;  PLACE IN "(SCRTCH)."
                0645 ;*************************************************************
                0646 ;*                                                           *
                0647 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0648 ;*      INPUT- - - - - - - -  - - - - - - - - - -  *
                0649 ;*     OUTPUT- C C C D D D D  U  U  U  U  U  U  U  U  U  *
                0650 ;*                                                           *
                0651 ;*       FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'     *
                0652 ;*      INPUT- - - - - - - - - -  - - - - - - - - -      *
                0653 ;*     OUTPUT- C C C C C  C  C C  U U U U U  U  U U      *
                0654 ;*                                                           *
                0655 ;*     MEMORY: 07FA 07FB 07FC 07FD 07FE                  *
                0656 ;*      INPUT-  --   --   --   --   --                    *
                0657 ;*     OUTPUT-  UU   UU   UU   DD   DD                   *
                0658 ;*                                                           *
                0659 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0660 ;*                      -=DON'T CARE.                        *
                0661 ;*                                                           *
                0662 ;*************************************************************
02E1  3A001C    0663 LOOKUP LD    A,(PORT2)  ;GET SPEED BIT DATA (D1-D4)
02E4  E61E      0664        AND   1EH        ;MASK SAME
02E6  2600      0665        LD    H,00H      ;CALCULATE OFFSET AND LOOKUP TIMER
02E8  6F        0666        LD    L,A        ; CONSTANT FROM TABLE STARTING
02E9  017900    0667        LD    BC,CONST   ; AT CONST.
02EC  09        0668        ADD   HL,BC
02ED  5E        0669        LD    E,(HL)
02EE  23        0670        INC   HL
02EF  56        0671        LD    D,(HL)
02F0  EB        0672        EX    DE,HL      ;CONSTANT IN HL
02F1  22FD07    0673        LD    (SCRTCH),HL  ;AND ALSO IN SCRTCH
02F4  110100    0674        LD    DE,0001    ;CONSTANT FOR TIME DELAY
02F7  C9        0675        RET
                0676 ;
                0677 ;*************************************************************
                0678 ;      XMITER WILL OUTPUT A SINGLE ASCII CHARACTER
                0679 ; LOCATED IN ACC-A UPON ENTRY.
                0680 ;*************************************************************
                0681 ;*                                                           *
                0682 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0683 ;*      INPUT- D - - - - - -  - - - - - - - - - -  *
                0684 ;*     OUTPUT- C U U U U U U  C  C  C  C  C  C  C  U  U  *
                0685 ;*                                                           *
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0686 ;*      FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'     *
                0687 ;*     INPUT- - - - - -  - - -  - - - - - -  - - -      *
                0688 ;*    OUTPUT- C C C C C  C C C  C C C C C C  C C C      *
                0689 ;*                                                      *
                0690 ;*    MEMORY: 07FA 07FB 07FC 07FD 07FE                  *
                0691 ;*     INPUT-  --   --   --   --   --                   *
                0692 ;*    OUTPUT-  UU   UU   UU   CC   CC                   *
                0693 ;*                                                      *
                0694 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0695 ;*                      -=DON'T CARE.                   *
                0696 ;*                                                      *
                0697 ;****************************************************
02F8  08        0698 XMITER EX    AF,AF'      ;DATA TO A'
02F9  D9        0699        EXX               ;SAVE ENVIRONMENT
02FA  3EDF      0700        LD    A,0DFH      ;uP REQUEST TO SEND
02FC  32001C    0701        LD    (PORT2),A
02FF  3A001C    0702 CTSZRO LD    A,(PORT2)   ;GET STATUS BITS
0302  17        0703        RLA               ;CTS TO CARRY
0303  17        0704        RLA
0304  38F9      0705        JR    C,CTSZRO    ;ADM-3A READY? - => CTSZRO
0306  CDE102    0706        CALL  LOOKUP      ;GET ASYNC TRANSFER CONSTANT
0309  08        0707        EX    AF,AF'      ;GET DATA
030A  CB27      0708        SLA   A           ;0 START BIT
030C  4F        0709        LD    C,A         ;DATA TO C
030D  0608      0710        LD    B,08H       ;BITS/BYTE
030F  2AFD07    0711 DATAOT LD    HL,(SCRTCH) ;GET TIMER CONSTANT
0312  00        0712        NOP               ;TIME DELAY
0313  00        0713        NOP
0314  79        0714        LD    A,C         ;DATA TO A
0315  E6DF      0715        AND   0DFH        ;MASK
0317  F6DE      0716        OR    0DEH        ;MASK
0319  32001C    0717        LD    (PORT2),A   ;OUT DATA
031C  19        0718 LOOP1  ADD   HL,DE       ;DELAY ONE BIT LENGTH
031D  D21C03    0719        JP    NC,LOOP1
0320  CB39      0720        SRL   C           ;NEXT DATA BIT TO C-D0
0322  10EB      0721        DJNZ  DATAOT      ;MORE BITS? +=> DATAOT
0324  0603      0722        LD    B,03H       ;GUARANTEED 3 STOP BITS
0326  0EFF      0723        LD    C,0FFH      ;STOP BITS
0328  2AFD07    0724 STOPOT LD    HL,(SCRTCH)  ;GET TIMER CONSTANT
032B  79        0725        LD    A,C         ;GET STOP BITS
032C  E6DF      0726        AND   0DFH        ;MASK
032E  F6DE      0727        OR    0DEH        ;MASK
0330  32001C    0728        LD    (PORT2),A   ;OUT DATA
0333  00        0729        NOP               ;TIME DELAY
0334  00        0730        NOP
0335  19        0731 LOOP2  ADD   HL,DE       ;DELAY ONE BIT
0336  D23503    0732        JP    NC,LOOP2
0339  CB29      0733        SRA   C           ;NEXT STOP BIT
033B  10EB      0734        DJNZ  STOPOT      ;MORE BITS? +=> STOPOT
033D  0602      0735        LD    B,02H       ;SET UP FOR TIMING DELAY
033F  10FE      0736 ROUND  DJNZ  ROUND       ;DELAY 21 T
0341  00        0737        NOP               ;TIMING DELAY
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0342  79        0738         LD    A,C         ;RESET ALL FUNCTIONS (C=0FFH)
0343  32001C    0739         LD    (PORT2),A
0346  D9        0740         EXX               ;RESTORE ENVIRONMENT
0347  C9        0741         RET
                0742 ;
                0743 ;*******************************************************
                0744 ;       RECEIV INPUTS ONE ASCII CHARACTER INTO ACC-A AND
                0745 ; ECHOS BACK THE DATA TO THE ADM-3A
                0746 ;*******************************************************
                0747 ;*                                                     *
                0748 ;* REGISTERS: A B C D E H L  A' B' C' D' E' H' L' IX IY *
                0749 ;*      INPUT- - - - - - - -  - - - - - - - - - -  *
                0750 ;*     OUTPUT- D U U U U U U  C  C  C  C  C  C  C  U  U  *
                0751 ;*                                                     *
                0752 ;*       FLAG: S Z ? H ? P/V N C  S'Z'?'H'?'P/V'N'C'    *
                0753 ;*      INPUT- - - - - - - - - -  - - - - - - - -      *
                0754 ;*     OUTPUT- C C C C C  C  C C  C C C C C  C  C C     *
                0755 ;*                                                     *
                0756 ;*     MEMORY: 07FA 07FB 07FC 07FD 07FE                *
                0757 ;*      INPUT-  --   --   --   --   --                 *
                0758 ;*     OUTPUT-  UU   UU   UU   CC   CC                 *
                0759 ;*                                                     *
                0760 ;* C=CHANGED (NO DATA). U=UNCHANGED. D=DATA (IN OR OUT) *
                0761 ;*                      -=DON'T CARE.                  *
                0762 ;*                                                     *
                0763 ;*******************************************************
0348  3A001C    0764 RECEIV LD    A,(PORT2)   ;CHECK FOR RTS FROM ADM-3A
034B  07        0765        RLCA
034C  38FA      0766        JR    C,RECEIV    ;NO RTS=> RECEIV.
034E  D9        0767        EXX               ;PRESERVE ENVIRONMENT
034F  CDE102    0768 TRYREC CALL  LOOKUP      ;GET TIMER CONSTANTS
0352  37        0769        SCF               ;HALF TIMER FOR FIRST DELAY
0353  CB1C      0770        RR    H
0355  CB1D      0771        RR    L
0357  23        0772        INC   HL          ;USED FOR TIMING PURPOSES
0358  3EEF      0773        LD    A,0EFH      ;NSC800 READY TO RECEIVE (D4=0)
035A  0607      0774        LD    B,07H       ;DATA BITS/CHARACTER
035C  32001C    0775        LD    (PORT2),A   ;TELL ADM-3A THAT 800 IS READY
035F  3A001C    0776 RTOP   LD    A,(PORT2)   ;SEARCH FOR START BIT
0362  1F        0777        RRA
0363  38FA      0778        JR    C,RTOP      ;NO START BIT => RTOP
0365  23        0779        INC   HL          ;TIMING DELAY
0366  2B        0780        DEC   HL          ;TIMING DELAY
0367  00        0781        NOP
0368  19        0782 LOOP3  ADD   HL,DE       ;DELAY 1/2 BIT
0369  D26803    0783        JP    NC,LOOP3
036C  3A001C    0784        LD    A,(PORT2)   ;CHECK FOR FALSE START
036F  1F        0785        RRA
0370  38DD      0786        JR    C,TRYREC    ;FALSE START => TRYREC
0372  23        0787        INC   HL          ;TIMING DELAY
0373  2B        0788        DEC   HL          ;TIMING DELAY
0374  2AFD07    0789 DATAIN LD    HL,(SCRTCH) ;GET TIMER CONSTANT
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0377  19        0790 LOOP4  ADD   HL,DE       ;DELAY ONE BIT
0378  D27703    0791        JP    NC,LOOP4
037B  23        0792        INC   HL           ;DUMMY STATEMENT => TIME DELAY
037C  2B        0793        DEC   HL           ;MORE OF SAME
037D  23        0794        INC   HL           ;MORE OF SAME
037E  2B        0795        DEC   HL           ;MORE OF SAME
037F  3A001C    0796        LD    A,(PORT2)    ;SAMPLE DATA
0382  CB3F      0797        SRL   A            ;DATA TO CARRY
0384  CB19      0798        RR    C            ;DATA TO MSB OF C
0386  10EC      0799        DJNZ  DATAIN       ;DONE? - => DATAIN
0388  00        0800        NOP                ;TIMING DELAY
0389  00        0801        NOP                ;TIME DELAY
038A  79        0802        LD    A,C          ;DATA TO A
038B  08        0803        EX    AF,AF'       ;DATA TO A'
038C  0602      0804        LD    B,02H        ;NUMBER OF EXPECTED STOP BITS
038E  2AFD07    0805 STOPIN LD    HL,(SCRTCH)   ;GET DELAY CONSTANT
0391  19        0806 LOOP5  ADD   HL,DE       ;DELAY ONE BIT LENGTH
0392  D29103    0807        JP    NC,LOOP5
0395  3A001C    0808        LD    A,(PORT2)    ;GET STOP BIT
0398  23        0809        INC   HL           ;TIMING DELAY
0399  2B        0810        DEC   HL           ;TIMING DELAY
039A  23        0811        INC   HL           ;TIMING DELAY
039B  CB3F      0812        SRL   A            ;DATA TO CARRY
039D  CB19      0813        RR    C            ;DATA TO MSB OF C
039F  10ED      0814        DJNZ  STOPIN       ;MORE DATA? +=> STOPIN
03A1  3EFF      0815        LD    A,0FFH       ;RESET ALL I/O FUNCTIONS
03A3  32001C    0816        LD    (PORT2),A
03A6  79        0817        LD    A,C          ;STOP BITS TO ACC-A
03A7  0EC0      0818        LD    C,0C0H       ;MASK CONSTANT
03A9  A1        0819        AND   C            ;MASK
03AA  A9        0820        XOR   C
03AB  2813      0821        JR    Z,OVERRR     ;ON NO STOP BIT ERROR JUMP OVER
                0822 ;                           THE FOLLOWING ERROR ROUTINE
                0823 ;*****************************************************
                0824 ;              RECEIVE ERROR HANDLING ROUTINE
                0825 ;*****************************************************
03AD  CB79      0826        BIT   7,C          ;BIT 10 IN ERROR?
03AF  2804      0827        JR    Z,OVERI1     ;  -=> OVERI1
03B1  215000    0828        LD    HL,ERRTEN    ;POINTER TO ERROR CODE
03B4  F7        0829        RST   30H          ;OUTPUT SAME
03B5  CB71      0830 OVERI1 BIT   6,C          ;BIT 9 IN ERROR?
03B7  2807      0831        JR    Z,OVERRR     ;  -=> EXIT
03B9  3E39      0832        LD    A,39H        ;"9"
03BB  EF        0833        RST   28H
03BC  215200    0834        LD    HL,ERRTIN    ;POINTER TO ERROR CODE
03BF  F7        0835        RST   30H          ;OUTPUT SAME
                0836 ;
                0837 ;*****END OF ERROR HANDLING ROUTINE*****
                0838 ;
03C0  08        0839 OVERRR EX    AF,AF'       ;DATA TO A
03C1  CB3F      0840        SRL   A            ;SHIFT A 0 AS MSB OF DATA
03C3  F5        0841        PUSH  AF           ;SAVE DATA
```

```
ADDR  CODE      STMT SOURCE STATEMENT

03C4  D9        0842        EXX             ;RESTORE ENVIRONMENT
03C5  EF        0843        RST   28H       ;ECHO DATA
03C6  F1        0844        POP   AF        ;RECOVER DATA
03C7  C9        0845        RET
ERRORS=0000
```

MONIT1.91 Object listing

```
$MONIT9050025
:0700000031F707FBC3E50027
:03000800C3080426
:03001000C3100416
:03001800C3180406
:03002000C32004F6
:03002C00C32C04DE
:03003400C33404CE
:03003C00C33C04BE
:03002800C3F80218
:03003000C37E028A
:03003800C3F10011
:04000B003F3F3F87AD
:04001300546F3FA047
:03001B000A0DBE0D
:2000400030313233343536373839414243444546313O2D53544F5020424954204558504 5D9
:05006000435445448 7F4
:20006600C3F1000A0D46726F6D3FA04C656E6774683FA065FFC6FE77FE89FDB4FF65FF3D84
:20008600FFC6FEDCFFB4FFA0FF65FFF0FFDCFFD2FFB4FF4D280147640156230252BD0143C8
:2000A600E1014E14024265020D4E5343383030204D6F6E69746F722056657273696F6E2094
:2000C600312E39310A0D4B616E7361732053746174652056E697665727369747 90A8D3E81
:2000E600FF4710FECD870221AF00F7E32BE3FDE5DDE5E5D5C5F508D9E5D5C5F5CD870221B4
:200106001B00F7CD4803219700010800232 3EDA1E2200120F75E2356EBE928E3210B00F727
:2001260018DDCDAE02DA22012AFB07CD93023E3EEF7E32FC070602CD98023E20EF1E02CDF5
:20014600B002300CFE53CA0501FE0A280BC322013AFB0777BEC22201233E0DEF18CDCDAE56
:2001660002DA22013EC332FA07F1C1D1E108D9F1C1D1E1DDE1FDE1C3FA070A0D5350202043
:2001860020412746272042274327204427452720482 74C27204120462020422043 20204448
:2001A6002045202048204C2020495820202049592020205043 0A8DED73FB07218001F72A4E
:2001C600FB07060CC5CD96023E20EF7E32FB07237E32FC0723C110ECC30501216900F7CD0F
:2001E600AE0238272AFB07E5211300F7CDAE0238192AFB07E5217100F7CDAE02380BED4B4E
:20020600FB07D1E1EDB0C30501E1E1C32201F1C1D1E108D9F1C1D1E1DDE1FDE1C9210000E7
:2002260022FB07CDAE02FE2CC222012AFB07CDAE02DA2201ED5BFB0713E521E300F7E1CD77
:2002460093023E20EF7E32FC070602CD9802237DBB20057CBACA05017DE60F28DC18E3CDD0
:20026600 8702CDAE02DA22012AFB077E22F80732F7073EFF77C305017EF5EFF117D8231880
:20028600F72AF8073AF70777AF32F907C922FB070604E521FB073E30ED6F23ED6FFE3A38F0
:2002A60002C607EF10EDE1C91E00E5CD4803FE0D281C214000011000EDA1E2C50220F92087
:2002C600172D7D21FB07ED6F23ED6F1C18DD21E300F73EFC833FE1C937E1C93A001CE61E6C
:2002E60026006F017900095E2356EB22FD07110100C908D93EDF32001C3A001C171738F91C
:20030600CDE10208CB274F06082AFD07000079E6DFF6DE32001C19D21C03CB3910EB060330
:200326000EFF2AFD0779E6DFF6DE32001C000019D23503CB2910EB060210FE007932001C2D
:20034600D9C93A001C0738FAD9CDE10237CB1CCB1D233EEF060732001C3A001C1F38FA2362
:200366002B0019D268033A001C1F38DD232B2AFD0719D27703232B232B3A001CCB3FCB19AB
:2003860010EC000079080602 2AFD0719D291033A001C232B23CB3FCB1910ED3EFF32001CE8
:2003A600790EC0A1A92813CB792804215000F7CB7128073E39EF215200F708CB3FF5D9EF89
:0203C600F1C97B
:00000001FF
```

NorthStar Second Serial Interface Header Configuration

1-11	2-16
3-15	4-13
5-12	6-NC
7-8	9-10
12-NC	

Table C-1.

Connector Configuration

NSC800 Pin (J28)	NorthStar Pin (second serial)
1	1
2	3
3	2
4	5
11	6
12	20

Table C-2.

Hardware Block Diagram

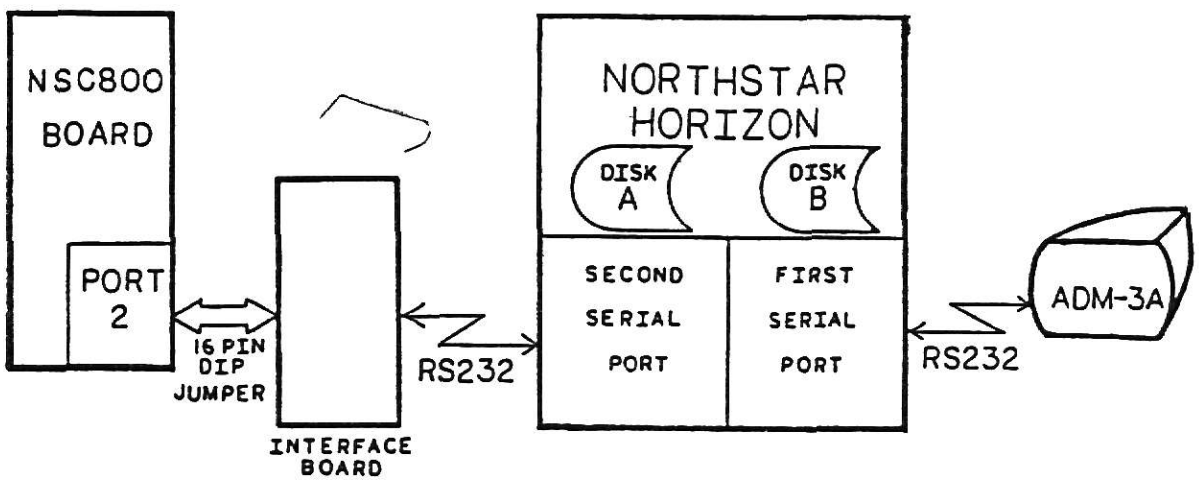


Figure C-1.

1. Assemble the connector and header specified in Tables C-1 and C-2. Install the second serial header in the appropriate dip inside the NorthStar Horizon. Install the specified DB25 male to male connector (see Table C-2) between the second serial port of the NorthStar Horizon and the NSC800 board. Please take note that this connector will work in one direction only, ie. it is not symmetric with respect to the interface board and the NorthStar. WARNING: Make sure that the baud rate selected in hardware for the second serial port is the same as that expected by the NSC800 board. {The second serial port on the NorthStar is the one on the left as you look at the back.} For assembly instructions for the NSC800 interface board see Appendix B.

2. Power up all equipment with the appropriate system disk in drive A (left). {A working knowledge of the CP/M operating system and its associated assemblers is assumed by the author.} The CP/M operating system should be in effect at this time.

3. This step is optional. Write, edit, and assemble a source program under CP/M. An object file of type xxxxxxxx.OBJ* or xxxxxxxx.HEX* must be created for downloading purposes.

4. Make sure that NORTH.COM is on the specified disk by typing "DIR NORTH.*<cr>". CP/M will respond with all of the file names NORTH with different extensions. The NorthStar cannot execute a program that is not on the specified disks.

Perform either 5a or 5b.

5a. To start the program type "NORTH <object program name>.<extension><cr>" where <object Program name> is the optional

* "xxxxxxxx" may be any user-selected CP/M file name.

object program name. NORTH.COM will respond with the appropriate banner wakeup message and a command summary. If an error was detected when trying to open the specified input file, an appropriate error message will be displayed. The user now has the option of discontinuing program execution (by typing 'Y' as the answer to the program prompt) or continuing in the immediate mode (no downloading) of operation only.

5b. If no object program has been created then type "NORTH<cr>". The program assumes that an input file is suggested and warns the user that an input file is not available. Answer the program prompt with any key except 'Y.' This will put the user in the immediate mode (no downloading) of operation.

6. The user is now in communication with the NSC800 the same as if he were directly connected to the ADM-3A. Any of the monitor functions may now be executed from the keyboard of the NorthStar. For more information on the NSC800 monitor functions see Appendix B of this report. Two commands are blocked and interpreted by the interface program within the NorthStar. These commands allow the user to download an object program (^[, ESCape) and to exit the program and return to CP/M (^C). . ESCape will only perform properly when an acceptable object program has been previously specified (see 5a).

WARNING: If the NSC800 monitor command prompt is not displayed at this time try pressing the carriage return. If garbage appears on the screen the probable cause is an incorrect matching of baud rates between the NSC800 and the NorthStar. If nothing appears on

the screen try pressing the reset button on the NSC800. If the NSC800 halts (red status LED on the cpu board lighted) then reseat the EPROMs on the cpu board and press the reset button again. If the monitor still does not respond then get out a logic probe and start trouble shooting the communication interface.

{Author's note: On NorthStar Horizon 2 a special duplicate DB25 connector is installed for each of the first and second serial ports. These should NOT be used for the above application. It has been found that the routines require the handshaking lines not provided on these connectors. Data will be lost if the three wire connectors are used.}

Notes on Downloading.

1. Addressing is absolute with respect to the NSC800 memory map. On the existing system this means that programs are limited to about 950D bytes of code (the monitor requires some memory for operation). Source assembly programs using the existing NSC800 system must map the object code to absolute addresses on available system RAM (0400H-07C0). Mapping to other areas will cause a monitor error. The interface program has no method of error recovery except a manual abort of the download (press any key during data transmisssion and the routine will eventually stop and return to the monitor).

2. Downloading may be terminated at the end of any record of object file by pressing any key on the keyboard. Command will return to the NSC800 monitor.

Sample Run*

```
{Author's note:  All user inputs appear underlined.}
[reset]
CP/M2 on North Star disk
Double density - Quad capacity
32K vers 2.20
Copyright (C) 1980 Lifeboat Associates

A>dir a:north.*[cr]
A: NORTH    ASM : NORTH    COM
A>dir b:test.*[cr]
B: TEST         : TEST     OBJ
A>a:north[cr]
NorthStar <=> NSC800 interface program.

Command Summary:
     All NSC800 Monitor commands may be issued with the following
  additions.  1)  ^[ (ESCape) causes the designated source
  file to be placed in the NSC800 RAM (see this program's
  listing for further information).  2)  ^C causes this
  program to terminate and control to be transferred to CP/M.



WARNING:  NO INPUT FILE PRESENT ON DISK
ABORT? ('Y' OR RETURN) Y

A>north b:test.obj[cr]
NorthStar <=> NSC800 interface program.

Command Summary:
     All NSC800 Monitor commands may be issued with the following
  additions.  1)  ^[ (ESCape) causes the designated source
  file to be placed in the NSC800 RAM (see this program's
  listing for further information).  2)  ^C causes this
  program to terminate and control to be transferred to CP/M.

[NSC800:  power ON]                              {Press NSC800 reset button.}
NSC800 Monitor Version 1.91
Kansas State University

>R                                               {View existing registers.}
SP   A'F' B'C' D'E' H'L' A F  B C  D E  H L  IX   IY   PC
07E3 0200 00FF 0001 0000 1A4D 0004 55ED 00E4 5555 5555 FFFE
>V,0017[cr]                                      {View a memory block.}
0000 31 F7 07 FB C3 E5 00 FF C3 08 04 3F 3F 3F 87 FF
0010 C3 10 04 54 6F 3F A0 FF
>[ESCape]M0400                                   {Start automatic transfer.}
0400>FF DD
0401>FF 21
0402>FF 00
0403>FF 00
0404>FF FD
0405>FF 21
```

* Nonprintable characters appear in brackets, eg. [cr].

```
0406>FF 01
0407>FF 00
0408>FF DD
0409>FF 7E
040A>FF 00
040B>FF FD
040C>FF 46
040D>FF 00
040E>FF DD
040F>FF 4E
0410>FF 01
0411>FF FD
0412>FF 56
0413>FF 01
0414>FF DD
0415>FF 5E
0416>FF 02
0417>FF FD
0418>FF 66
0419>FF 02
041A>FF DD
041B>FF 6E
041C>FF 03
041D>FF DD
041E>FF 23
041F>FF FD
0420>FD S
>M0420
0410>FF 23
0411>FF D9
0412>FF 21
0413>FF 34
0414>FF 12
0415>FF 11
0416>FF 78
0417>FF 56
0418>FF 01
0419>FF BC
041A>FF 9A
041B>FF S
>[beep]Done filling NSC800 buffer with data file.[beep]
>G0400[cr]                                          {Run the inputted program.}

>R                                                  {Look at program results.}
SP   A'F' B'C' D'E' H'L' A F  B C  D E  H L  IX   IY   PC
07E1 0200 F7F7 0707 FBFB 314D 9ABC 5678 1234 0001 0002 041B
>[cr]                                               {NOP.}
>[^C]                                              {Terminate secession.}

A>

{END OF SAMPLE RUN.}
```

Test Program Print Listing

```
                                        SD SYSTEMS Z80 ASSEMBLER PAGE 0001
 ADDR  CODE      STMT SOURCE STATEMENT

                 0001 ;       Dwight W. Gordon                6-21-82
                 0002 ;       Kansas State University
                 0003 ;       Department of Electrical Engineering
                 0004 ;       Seaton Hall
                 0005 ;       Manhattan, Kansas 66506
                 0006 ;
                 0007 ;       This program is a simple test of the programming
                 0008 ; mode of the NORTH.ASM program.
                 0009 ;       Source code is Z80.
                 0010 ;
                 0011         PSECT   ABS
                 0012         NAME    TEST1
 >0400           0013         ORG     0400H
 0400  DD210000  0014         LD      IX,0000H
 0404  FD210100  0015         LD      IY,0001H
 0408  DD7E00    0016         LD      A,(IX)
 040B  FD4600    0017         LD      B,(IY)
 040E  DD4E01    0018         LD      C,(IX+1)
 0411  FD5601    0019         LD      D,(IY+1)
 0414  DD5E02    0020         LD      E,(IX+2)
 0417  FD6602    0021         LD      H,(IY+2)
 041A  DD6E03    0022         LD      L,(IX+3)
 041D  DD23      0023         INC     IX
 041F  FD23      0024         INC     IY
 0421  D9        0025         EXX
 0422  213412    0026         LD      HL,1234H
 0425  117856    0027         LD      DE,5678H
 0428  01BC9A    0028         LD      BC,9ABCH
                 0029         END

ERRORS=0000
```

Test Program Object Listing

```
$TEST1 05003E
:20040000DD210000FD210100DD7E00FD4600DD4E01FD5601DD5E02FD6602DD6E03DD23FDB4
:0B04200023D921341211785601BC9A38
:00040001FB
```

NORTH.ASM Print Listing*

```
                ;*****************************************************************
                ;       DWIGHT W. GORDON                    10-27-81          *
                ;       KANSAS STATE UNIVERSITY       REV. 5-19-82          *
                ;      DEPARTMENT OF ELECTRICAL ENGINEERING                   *
                ;      SEATON HALL                                            *
                ;      MANHATTAN, KS 66506                                    *
                ;*****************************************************************
                ;      NORTH.ASM IS AN 8080A BASED PROGRAM TO INTERFACE THE
                ; NSC800 BOARD DEVELOPED BY MAC CODY (1980) WITH THE INTERFACE
                ; EXTENSION AND MONITOR (VERSION 1.8+) WRITTEN BY THE AUTHOR
                ; TO THE SECOND SERIAL PORT OF THE NORTHSTAR uP.  THE PROGRAM
                ; ALLOWS THE USER TO SPECIFY AN INPUT OBJECT FILE TO BE USED AS
                ; A SOURCE FILE FOR DATA TRANSFER TO THE NSC800.  ALSO, THE
                ; PROGRAM PROVIDES BIDIRECTIONAL COMMUNICATIONS BETWEEN THE
                ;  NORTHSTAR AND THE NSC800 MONITOR (VERSION 1.8+).
                ;      ALL CHARACTERS INPUTTED BY THE TERMINAL WILL BE ECHOED
                ; TO THE NSC800 AND VERIFIED EXCEPT FOR THE FOLLOWING SPECIAL
                ; CHARACTERS:
                ;
                ;       ^[  (ESCape)    RESET THE BEGINNING OF FILE POINTER AND
                ;                       INPUT THE SPECIFIED INPUT FILE TO THE
                ;                       NSC800 VIA THE Memory change COMMAND OF
                ;                       THE NSC800 MONITOR.
                ;
                ;       Y               EXIT THE PROGRAM.  (THIS COMMAND IS USED
                ;                       IN RESPONSE TO A PROGRAM PROMPT
                ;                       CONCERNING AN INVALID INPUT FILE.)
                ;
                ;       ^C  (CNTRL-C)   EXIT THE PROGRAM.  THIS STANDARD CP/M
                ;                       REBOOT COMMAND IS KEPT.  THE FUNCTION IS
                ;                       THE SAME AS THAT OF THE 'Y' ABOVE.
                ;
                ;WARNING:
                ;      INPUT FILES MUST BE OF TYPE xxxxxxxx.OBJ or xxxxxxxx.HEX
                ;
 0100                   ORG     100H
 0005 =         BDOS    EQU     0005H   ;DOS ENTRY POINT
 0001 =         CONS    EQU     1       ;READ CONSOLE
 0002 =         TYPEF   EQU     2       ;TYPE FUNCTION
 0009 =         PRINTF  EQU     9       ;BUFFER PRINT ENTRY
 000B =         BRKF    EQU     11      ;BREAK KEY FUNCTION (TRUE IF CHAR READY
>000F =         OPENF   EQU     15      ;FILE OPEN
 0014 =         READF   EQU     20      ;READ FUNCTION
                ;
 005C =         FCB     EQU     5CH     ;FILE CONTROL BLOCK ADDRESS
 0080 =         BUFF    EQU     80H     ;INPUT DISK BUFFER ADDRESS
                ;
                ;       NON GRAPHIC CHARACTERS
 000D =         CR      EQU     0DH     ;CARRIAGE RETURN
 000A =         LF      EQU     0AH     ;LINE FEED
```

* Minor modifications allow this listing to fit within the margins.

```
                ;
                ;       FILE CONTROL BLOCK DEFINITIONS
005C =          FCBDN   EQU     FCB+0   ;DISK NAME
005D =          FCBFN   EQU     FCB+1   ;FILE NAME
0065 =          FCBFT   EQU     FCB+9   ;DISK FILE TYPE (3 CHARACTERS)
0068 =          FCBRL   EQU     FCB+12  ;FILE'S CURRENT REEL NUMBER
006B =          FCBRC   EQU     FCB+15  ;FILE'S RECORD COUNT (0 TO 128)
007C =          FCBCR   EQU     FCB+32  ;CURRENT (NEXT) RECORD NUMBER (0 TO 127)
007D =          FCBLN   EQU     FCB+33  ;FCB LENGTH
                ;
                ;       SET UP STACK
0100 210000             LXI     H,0     ;
0103 39                 DAD     SP      ;
                ;       ENTRY STACK POINTER IN HL FROM THE CCP
0104 22AF05             SHLD    OLDSP   ;
                ;       SET SP TO LOCAL STACK AREA (RESTORED AT FINIS)
0107 31D105             LXI     SP,STKTOP
                ;       RECONFIGURE BOTH SERIAL INPUT PORTS
010A AF                 XRA     A       ;I/O MASTER RESET
010B D306               OUT     6       ;
010D D306               OUT     6       ;SPARES
010F D306               OUT     6       ;
0111 D306               OUT     6       ;
0113 3ECE               MVI     A,0CEH  ;16x,2stop,no parity,8bit data
0115 D303               OUT     3       ;TERMINAL
0117 3ECA               MVI     A,0CAH  ;16x,2stop,no parity,7bit data
0119 D305               OUT     5       ;NSC800
011B 3E37               MVI     A,37H   ;SET UP TERMINAL HANDSHAKE LINES
011D D303               OUT     3       ;
011F 3E17               MVI     A,17H   ;SET UP NSC800 HANDSHAKE LINES
0121 D305               OUT     5       ;
0123 CDE801             CALL    DELAYS  ;LET LINES SETTLE
0126 115503             LXI     D,SIGNON ;TYPE SIGN ON MESSAGE
0129 CD0203             CALL    ERR     ;
012C CDB602             CALL    CRLF0   ;RESET TERMINAL CURSOR POSITION
                ;
012F CD3903     RESETR  CALL    SETUP   ;SET UP INPUT FILE
0132 FEFF               CPI     255     ;255 IF FILE NOT PRESENT
0134 C25A01             JNZ     OPENOK  ;SKIP IF OPEN IS OK
                ;
                ;       FILE NOT THERE, GIVE ERROR MESSAGE AND EXIT IF DESIRED
                ;
0137 11C904     OPENERR LXI     D,OPNMSG
013A CD0203             CALL    ERR     ;PROGRAM PRINT FUNCTION
013D DB03       ERRINX  IN      3       ;GET NSC800 BUFFER INPUT STATUS
013F 2F                 CMA             ;
0140 E602               ANI     2       ;
0142 C23D01             JNZ     ERRINX  ;NO DATA => ERRINX
0145 DB02               IN      2       ;
0147 E67F               ANI     7FH     ;MASK DATA
0149 47                 MOV     B,A     ;ECHO DATA TO TERMINAL
014A 3E00               MVI     A,0     ;
014C CDBB01             CALL    XMIT    ;
014F FE59               CPI     'Y'     ;CHECK FOR EXIT
```

```
0151 CAFA02             JZ      FINIS   ;
0154 CDB602             CALL    CRLF0   ;OUTPUT A NEW LINE TO TERMINAL
0157 C36601             JMP     CIN     ;
                ;
                OPENOK: ;OPEN OPERATION OK, SET BUFFER INDEX TO END
015A 3E80               MVI     A,80H   ;
015C 32AD05             STA     IBP     ;SET BUFFER POINTER TO 80H
015F 060D               MVI     B,CR    ;SEND A <cr> TO NSC800
0161 3E01               MVI     A,1     ;
0163 CDBB01             CALL    XMIT    ;
                ;
                ;       INPUT POLLING GIVES EQUAL PRIORITY TO BOTH DEVICES
                ;
0166 DB03       CIN     IN      3       ;CHECK IF TERMINAL BUFFER FULL
0168 2F                 CMA             ;
0169 E602               ANI     2       ;
016B CA7901             JZ      IN0     ;  + => IN0
016E DB05               IN      5       ;CHECK IF NSC800 BUFFER FULL
0170 2F                 CMA             ;
0171 E602               ANI     2       ;
0173 CA9001             JZ      IN1     ;  + => IN1
0176 C36601             JMP     CIN     ;POLL INPUTS ONCE AGAIN
              ;
              ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
              ;       TERMINAL INPUT HANDLING ROUTINE                       +
              ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
              ;
0179 DB02      IN0      IN      2       ;GET BUFFER DATA
017B E67F               ANI     7FH     ;MASK OFF PARITY (?)
017D FE03               CPI     3       ;^C => EXIT
017F CAFA02             JZ      FINIS   ;
0182 FE1B               CPI     1BH     ;^[ (ESC) => SEND INPUT FILE
0184 CACE02             JZ      XMFILE  ;
0187 47                 MOV     B,A     ;DEFAULT IS SEND DATA TO NSC800
0188 3E01               MVI     A,01    ;
018A CDBB01             CALL    XMIT    ;
018D C36601             JMP     CIN     ;RETURN TO INPUT POLLING ROUTINE
              ;
              ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              ;       NSC800 INPUT HANDLING ROUTINE                         %
              ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              ;
0190 DB04      IN1      IN      4       ;GET DATA
0192 E67F               ANI     7FH     ;MASK AS BEFORE
0194 47                 MOV     B,A     ;SEND TO TERMINAL
0195 3E00               MVI     A,0     ;
0197 CDBB01             CALL    XMIT    ;
019A C36601             JMP     CIN     ;RETURN TO INPUT POLLING ROUTINE
```

```
                    ;
                    ;###################################################################
                    ;       THE FOLLOWING ROUTINE TRIES 255 TIMES (WITH A DELAY       #
                    ; BETWEEN EACH ATTEMPT) TO READ A CHARACTER FROM THE NSC800       #
                    ; BUFFER.  IT IS USED TO OBTAIN RESPONSE DATA PROVIDED BY THE     #
                    ; NSC800 MONITOR ROUTINES.                                       #
                    ;###################################################################
                    ;
019D C5             CINE    PUSH    B       ;SAVE USER STATUS
019E 0600                   MVI     B,00    ;CONSTANT = NUMBER OF TRIES + 1
01A0 05             CINER   DCR     B       ;DONE?
01A1 CAB901                 JZ      OUTCIN  ;  + => OUTCIN
01A4 CDE801                 CALL    DELAYS  ;DELAY
01A7 DB05                   IN      5       ;CHECK INPUT BUFFER STATUS
01A9 2F                     CMA             ;
01AA E602                   ANI     2       ;
01AC C2A001                 JNZ     CINER   ;EMPTY => TRY AGAIN
01AF DB04           INLE    IN      4       ; ELSE GET DATA
01B1 E67F                   ANI     7FH     ;MASK
01B3 47                     MOV     B,A     ;SEND TO TERMINAL
01B4 3E00                   MVI     A,0     ;
01B6 CDBB01                 CALL    XMIT    ;
01B9 C1             OUTCIN  POP     B       ;RECOVER USER STATUS
01BA C9                     RET
                    ;
                    ;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                    ;       XMIT SENDS THE DATA IN ACC-B TO THE DEVICE SPECIFIED      &
                    ; BY ACC-A (1 => NSC800, ANYTHING ELSE => TERMINAL).             &
                    ;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                    ;
01BB CDF901         XMIT    CALL    OST     ;BUFFER EMPTY (+ => Z BIT SET)
01BE C2BB01                 JNZ     XMIT    ; - => TRY AGAIN
01C1 FE01                   CPI     1       ;IS DEVICE THE NSC800?
01C3 CACA01                 JZ      COUTT1  ; + => COUTT1
01C6 78             COUTT0  MOV     A,B     ;DEFAULT => TERMINAL
01C7 D302                   OUT     2       ;SEND DATA
01C9 C9                     RET
                    ;
                    ;       NSC800 OUTPUT HANDLING ROUTINE.
                    ;               NOTE:  THIS ROUTINE IS BASED ON DELAYING THE
                    ;               NORTHSTAR UNTIL THE NSC800 IS READY.  FULL
                    ;               HANDSHAKING IS NOT USED.  TEST CONDITIONS ARE
                    ;               TERMINAL @ 9600 BAUD, NSC800 @ 2400 BAUD & 2.5
                    ;               MHz.
                    ;
01CA 3E37           COUTT1  MVI     A,37H   ;SEND REQUEST TO SEND
01CC D305                   OUT     5       ;
01CE CDE801                 CALL    DELAYS  ;WAIT FOR A WHILE AS NSC800 GETS READY
01D1 DB05           NTXRDY  IN      5       ;BUFFER EMPTY AND READY?
01D3 E601                   ANI     01H     ;
01D5 CAD101                 JZ      NTXRDY  ; - => TRY AGAIN
01D8 78                     MOV     A,B     ;SEND DATA
01D9 D304                   OUT     4       ;
01DB CDE801                 CALL    DELAYS  ;LEAVE REQUEST TO SEND ACTIVE FOR WHILE
```

```
01DE CDE801           CALL    DELAYS  ;
01E1 F5               PUSH    A       ;SAVE DATA
01E2 3E17             MVI     A,17H   ;REMOVE REQUEST TO SEND
01E4 D305             OUT     5       ;
01E6 F1               POP     A       ;RECOVER DATA TO ACC-A
01E7 C9               RET
                ;
                ;       DELAY ROUTINE USES NESTED DELAYS.  UPON EXIT, STATUS
                ; OF USED REGISTER IS RESTORED.
                ;
01E8 C5         DELAYS  PUSH    B       ;SAVE REGISTER STATUS
01E9 0602             MVI     B,2H    ;FIRST DELAY CONSTANT
01EB C5         AROUND  PUSH    B       ;SAVE
01EC 0680             MVI     B,80H   ;SECOND DELAY CONSTANT
01EE 05         ARNDI   DCR     B       ;
01EF C2EE01           JNZ     ARNDI   ;END OF FIRST LOOP
01F2 C1               POP     B       ;
01F3 05               DCR     B       ;
01F4 C2EB01           JNZ     AROUND  ;END OF SECOND LOOP
01F7 C1               POP     B       ;RECOVER STATUS
01F8 C9               RET
                ;
                ;       CHECK BUFFER STATUS OF DEVICE SPECIFIED BY ACC-A
                ;       BUFFER EMPTY => Z BIT SET
                ;
01F9 FE01       OST     CPI     1       ;NSC800?
01FB CA0602           JZ      OST1    ; + => OST1
01FE DB03       OST0    IN      3       ;DEFAULT => TERMINAL
0200 2F               CMA             ;
0201 E601             ANI     1       ;
0203 3E00             MVI     A,0     ;RESTORE DEVICE NUMBER
0205 C9               RET
0206 DB05       OST1    IN      5       ;
0208 2F               CMA             ;
0209 E601             ANI     1       ;
020B 3E01             MVI     A,1     ;RESTORE DEVICE NUMBER
020D C9               RET
                ;
                ;*****************************************************************
                ;       ROUTINES FOR FILE TRANSFER                              *
                ;*****************************************************************
                ;       NUM SETS THE FILE BUFFER POINTER TO THE FIRST PAIR OF
                ; NIBBLES FROM THE INPUT FILE.  IT ALSO PLACES THE LENGTH OF
                ; THE CURRENT LINE OF INPUT (LENGTH=HEX BYTES) IN ACC-B.
                ;
020E CD0803     NUM     CALL    GNB     ;Get Next file data Byte
0211 FE3A             CPI     ':'     ;
0213 C20E02           JNZ     NUM     ;WAIT UNTIL A ':' IS FOUND (IE. START
                ; OF A RECORD)  FOR FURTHER INFORMATION ON THE FORMAT OF A
                ; TYPICAL xxxxxxxx.HEX OR xxxxxxxx.OBJ FILE, IT IS SUGGESTED
                ; THAT THE PROGRAM MODIFIER LOOK AT AN EXAMPLE OF EACH.
0216 CD0803           CALL    GNB     ;GET MOST SIG NIBBLE OF LENGTH
0219 07               RLC             ;   NOTE: MS NIBBLE := [0..9]
021A 07               RLC             ;
```

```
021B 07              RLC             ;
021C 07              RLC             ;
021D E6F0            ANI     0F0H    ;MASK OFF GARBAGE
021F F5              PUSH    A       ;SAVE ON STACK
0220 CD0803          CALL    GNB     ;GET LOWER NIBBLE
0223 C1              POP     B       ;UPPER NIBBLE TO B
0224 D630            SUI     '0'     ;DO FULL ASCII->HEX CONVERSION
0226 FE0A            CPI     10      ;
0228 DA2D02          JC      P101    ;
022B D607            SUI     7       ;
022D 80      P101    ADD     B       ;ADD UPPER AND LOWER NIBBLES = LENGTH
                                     ; OF RECORD.
022E 47              MOV     B,A     ;SAVE IN ACC-B
022F C9              RET
                ;
                ;       OUTPUT THE COMMAND BLOCK TO THE NSC800:
                ;               'M<starting address><cr>'
                ;
0230 C5      CMD     PUSH    B       ;SAVE RECORD LENGTH
0231 CD9D01          CALL    CINE    ;GET ANY CHARACTERS FROM NSC800
0234 3E01            MVI     A,1     ;OUTPUT THE COMMAND CHARACTER
0236 064D            MVI     B,'M'   ;
0238 CDBB01          CALL    XMIT    ;
023B CD9D01          CALL    CINE    ;GET ECHO COMMAND CHARACTER
023E 0604            MVI     B,4     ;OUTPUT FOUR DIGIT STARTING ADDRESS
0240 C5      CMD1    PUSH    B       ;  FROM INPUT FILE
0241 CD0803          CALL    GNB     ;
0244 47              MOV     B,A     ;
0245 3E01            MVI     A,1     ;SEND EACH CHARACTER TO NSC800
0247 CDBB01          CALL    XMIT    ;
024A CD9D01          CALL    CINE    ;ECHO BACK EACH CHARACTER
024D C1              POP     B       ;
024E 05              DCR     B       ;
024F C24002          JNZ     CMD1    ;
0252 060D            MVI     B,CR    ;SIGNAL END OF COMMAND WORD
0254 3E01            MVI     A,1     ;
0256 CDBB01          CALL    XMIT    ;
0259 CD0803          CALL    GNB     ;SKIP THE NEXT TWO BYTES OF THE INPUT
025C CD0803          CALL    GNB     ; FILE.  THEY ARE NOT PERTINANT DATA
025F 060A            MVI     B,10    ;GET THE LENGTHY NSC800 RESPONSE
0261 C5      ECHOI   PUSH    B       ; AND SEND TO THE TERMINAL
0262 CD9D01          CALL    CINE    ;
0265 C1              POP     B       ;
0266 05              DCR     B       ;
0267 C26102          JNZ     ECHOI   ;
026A C1              POP     B       ;RECOVER THE RECORD LENGTH
026B C9              RET             ;
                ;
                ;###############################################################
                ;       THE FOLLOWING ROUTINE WILL OUTPUT THE NUMBER OF BYTES #
                ; OF DATA (WHICH IS SPECIFIED BY THE CONTENTS OF REGISTER B)  #
                ; FROM THE INPUT FILE TO THE NSC800 MONITOR.  THIS ROUTINE IS #
                ; USED TO OBTAIN THE TRANSFER OF A RECORD OF DATA.  NO ERROR  #
                ; CHECKING IS DONE BY THE ROUTINE.  IF AN ERROR SHOULD OCCUR  #
```

```
                ; IN THE TRANSFER AND THE NSC800 MONITOR DETECTS SAME BY      #
                ; OUTPUTTING AN ERROR CODE, THEN THIS ROUTINE HAS NO METHOD   #
                ; OF RECOVERY.  IT IS SUGGESTED THAT THE USER KEEP AN EAR TO  #
                ; THE TERMINAL DURING A DATA TRANSFER.  IF THE TERMINAL BELL  #
                ; RINGS FOR ANYTHING OTHER THAN THE END OF TRANSFER "DONE,"   #
                ; THE USER SHOULD SUSPECT INVALID DATA TRANSFER AND TAKE THE  #
                ; APPROPRIATE ACTION.                                         #
                ;##############################################################
                ;
026C C5         OUTT    PUSH    B       ;SAVE RECORD LENGTH
026D CD9D01             CALL    CINE    ;CHECK FOR ANY NSC800 TRANSMISSION
0270 CD0803             CALL    GNB     ;GET FIRST ASCII NIBBLE
0273 47                 MOV     B,A     ;SEND TO NSC800
0274 3E01               MVI     A,1     ;
0276 CDBB01             CALL    XMIT    ;
0279 CD9D01             CALL    CINE    ;WAIT FOR ECHO TO TERMINAL
027C CD0803             CALL    GNB     ;GET SECOND ASCII NIBBLE (LSN)
027F 47                 MOV     B,A     ;SEND TO NSC800
0280 3E01               MVI     A,1     ;
0282 CDBB01             CALL    XMIT    ;
0285 CD9D01             CALL    CINE    ;WAIT FOR ECHO TO TERMINAL
0288 060D               MVI     B,CR    ;OUTPUT END OF BYTE MARKER <cr>
028A 3E01               MVI     A,1     ;
028C CDBB01             CALL    XMIT    ;
028F 060B               MVI     B,11    ;GET NSC800 RESPONSE TO TERMINAL
0291 C5         MX1     PUSH    B       ;
0292 CD9D01             CALL    CINE    ;
0295 C1                 POP     B       ;
0296 05                 DCR     B       ;
0297 C29102             JNZ     MX1     ;
029A C1                 POP     B       ;RECOVER BYTE COUNTER
029B 05                 DCR     B       ;LAST BYTE ON RECORD
029C C26C02             JNZ     OUTT    ; - => OUTT
029F CD9D01             CALL    CINE    ;GET ANY HANGING NSC800 TRANSMISSION
02A2 0653               MVI     B,'S'   ;OUTPUT END OF COMMAND FLAG TO NSC800
02A4 3E01               MVI     A,1     ;
02A6 CDBB01             CALL    XMIT    ;
02A9 CD9D01             CALL    CINE    ;GET NSC800 RESPONSE TO END OF COMMAND
02AC CD9D01             CALL    CINE    ;
02AF CD9D01             CALL    CINE    ;
02B2 CD9D01             CALL    CINE    ;
02B5 C9                 RET
                ;
                ;       OUTPUT A <lf>, <cr> TO THE TERMINAL
                ;
02B6 060A       CRLF0   MVI     B,LF    ;
02B8 3E00               MVI     A,0     ;
02BA CDBB01             CALL    XMIT    ;
02BD 060D               MVI     B,CR    ;
02BF 3E00               MVI     A,0     ;
02C1 CDBB01             CALL    XMIT    ;
02C4 C9                 RET             ;
```

```
                ;
                ;       OUTPUT A MESSAGE TO THE TERMINAL UPON COMPLETION OF
                ; A TRANSFER OF A DATA FILE.
                ;
02C5 110C05      DONEMG: LXI    D,DONMSG ;START OF MESSAGE ENDING IN '$'
02C8 CD0203             CALL   ERR      ;OUTPUT ROUTINE
02CB C36601             JMP    CIN      ;RETURN TO INPUT POLLING
                ;
                ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                ;       THE FOLLOWING IS THE MAIN ROUTINE FOR A TRANSFER OF   %
                ; A FILE OF DATA.                                             %
                ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                ;
02CE CD3903      XMFILE CALL   SETUP    ;SET UP INPUT FILE
02D1 FEFF               CPI    255      ; IF AN ERROR ON FILE OPEN THEN OPENERR
02D3 CA3701             JZ     OPENERR ;
02D6 DB03        FILEH  IN     3        ;GET TERMINAL BUFFER STATUS
02D8 E602               ANI    02H      ;
02DA CAE802             JZ     OVERRT   ;BUFFER EMPTY => CONTINUE
02DD DB02               IN     2        ;DUMMY INPUT TO CLEAR BUFFER
02DF 114005             LXI    D,INTERP ;OUTPUT APPROPRIATE MESSAGE
02E2 CD0203             CALL   ERR      ;
02E5 C36601             JMP    CIN      ;RETURN TO INPUT POLLING
02E8 CD0E02      OVERRT CALL   NUM      ;GET CURRENT RECORD LENGTH & ADJUST
                                        ;  FILE POINTER
02EB 3EFF               MVI    A,0FFH   ;CHECK FOR ZERO LENGTH
02ED A0                 ANA    B        ;
02EE CAC502             JZ     DONEMG   ;LENGTH=0 => SEND DONE MESSAGE
02F1 CD3002             CALL   CMD      ; ELSE OUTPUT COMMAND WORD
02F4 CD6C02             CALL   OUTT     ; TRANSFER A RECORD
02F7 C3D602             JMP    FILEH    ; REPEAT
                ;
                FINIS:
                ;       END OF PROGRAM, RETURN TO CCP
                ;       (NOTE THAT A JMP TO 0000H REBOOTS)
02FA CDB602             CALL   CRLF0    ;
02FD 2AAF05             LHLD   OLDSP    ;
0300 F9.                SPHL            ;
                 ;      STACK POINTER CONTAINS CCP'S STACK LOCATION
0301 C9                 RET             ;TO THE CCP
                ;
                ;****************************************************************
                ;    PRINT A STRING OF ASCII CHARACTERS WHICH ENDS WITH A '$.'
                ;    START OF MESSAGE IS AT THE LOCATION SPECIFIED BY DE PAIR.
                ;
0302 0E09        ERR:   MVI    C,PRINTF ;PRINT BUFFER FUNCTION
0304 CD0500             CALL   BDOS     ;CP/M PRINT MACRO
0307 C9                 RET
```

```
                        ;
                        ;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                        ;       ROUTINE TO READ A BYTE OF AN INPUT FILE                 &
                        ;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                        ;
0308 3AAD05     GNB:    LDA     IBP     ;FILE BYTE POINTER
030B FE80               CPI     80H     ;CHECK FOR END OF CURRENT 'WINDOW'
030D C22903             JNZ     G0      ;NOT END => G0
0310 CD4603             CALL    DISKR   ;CALL DISK READ MACRO
0313 B7                 ORA     A       ;ZERO VALUE IF READ OK
0314 CA2903             JZ      G0      ;FOR ANOTHER BYTE
0317 F5                 PUSH    PSW     ;****DISK READ ERROR****SAVE ACC STATUS
0318 E67F               ANI     07FH    ;MASK TO SEVEN BITS
031A 47                 MOV     B,A     ;SEND CHARACTER TO TERMINAL
031B 3E00               MVI     A,0     ;
031D CDBB01             CALL    XMIT    ;
0320 118C05             LXI     D,FREADR ;OUTPUT ERROR MESSAGE
0323 CD0203             CALL    ERR     ;
0326 F1                 POP     PSW     ;RECOVER DATA
0327 37                 STC             ;FLAG ERROR
0328 C9                 RET             ;
0329 5F         G0:     MOV     E,A     ;LS BYTE OF BUFFER INDEX
032A 1600               MVI     D,0     ;DOUBLE PRECISION INDEX TO DE
032C 3C                 INR     A       ;INDEX=INDEX+1
032D 32AD05             STA     IBP     ;BACK TO MEMORY
0330 218000             LXI     H,BUFF  ;INDEX TO MEMORY
0333 19                 DAD     D       ;CALCULATE REAL ADDRESS FROM OFFSETS
0334 7E                 MOV     A,M     ;DATA TO ACC-A
0335 B7                 ORA     A       ;RESET CARRY BIT
0336 E67F               ANI     7FH     ;MASK PARITY
0338 C9                 RET
                        ;
                        ;       FILE OPEN FUNCTION THROUGH CP/M MACROS
                        ;
0339 AF         SETUP:  XRA     A       ;ZERO TO ACCUM
033A 327C00             STA     FCBCR   ;CLEAR CURRENT RECORD
033D 115C00             LXI     D,FCB   ;BEGINNING OF FILE INFORMATION
0340 0E0F               MVI     C,OPENF ;OPEN FUNCTION
0342 CD0500             CALL    BDOS    ;JUMP TO CP/M
0345 C9                 RET
                        ;
                        ;       DISK READ FUNCTION THROUGH CP/M MACROS
                        ;
0346 E5D5C5     DISKR:  PUSH H! PUSH D! PUSH B
0349 115C00             LXI     D,FCB   ;FILE ADDRESS BLOCK
034C 0E14               MVI     C,READF ;READ FUNCTION
034E CD0500             CALL    BDOS    ;CP/M MACRO
0351 C1D1E1             POP B! POP D! POP H
0354 C9                 RET
                        ;
                        ;       MESSAGES USED BY THE PROGRAM (NOTE ALL END WITH '$').
                        ;
0355 0D0A4E6F72SIGNON:  DB      CR,LF,'NorthStar <=> NSC800 interface program.'
                        DB      CR,LF,LF
```

```
0381 436F6D6D61          DB      'Command Summary:',CR,LF,'     '
0398 416C6C204E          DB      'All NSC800 Monitor commandsmaybeissued with'
                         DB      ' the following '
03D5 0D0A202061          DB      CR,LF,'  additions.  1)  ^[ (ESCape) causes the'
                         DB      ' designated '
040B 736F757263          DB      'source',CR,LF,'  file to be placedinthe NSC8'
                         DB      '00 RAM (see '
043E 7468697320          DB      'this program''s',CR,LF,'  listing for further'
                         DB      ' information).'
0471 2020322920          DB      '  2)  ^C causes this ',CR,LF,'  program to '
                         DB      'terminate and '
04A3 636F6E7472          DB      'control to be transferred to CP/M.',CR,LF,LF,'$'
04C9 0D0A574152OPNMSG:   DB      CR,LF,'WARNING:  NO INPUT FILE PRESENT ON DISK'
04F2 0D0A41424F          DB      CR,LF,'ABORT? (''Y'' OR RETURN) $'
050C 0707070745DONMSG:   DB      07H,07H,07H,07H,'Done filling NSC800 '
0524 6275666665          DB      'buffer with data file.',CR,LF,07H,07H,'>$'
0540 070D0A4669INTERP:   DB      07H,CR,LF,'File transfer INTERRUPTED.',07H,CR,LF
0560 436F6E7472          DB      'Control is transferred to NSC800 Monitor.',CR
                         DB      LF,'$'
058C 0D0A072A2AFREADR:   DB      CR,LF,07H,'*****FILE READ ERROR.*****',CR,LF
                         DB      07H,'$'
05AD             IBP:    DS      2
05AF             OLDSP:  DS      2
05B1             DS      32
                 STKTOP:
                  ;
05D1                     END
```

NORTH.ASM Object Listing

```
:10010000210003922AF0531D105AFD306D306D384
:1001100006D3063ECED3033ECAD3053E37D3033EB5
:1001200017D305CDE801115503CD0203CDB602CD9D
:100130003903FEFFC25A0111C904CD0203DB032FAC
:10014000E602C23D01DB02E67F473E00CDBB01FE79
:1001500059CAFA02CDB602C366013E8032AD050629
:100160000D3E01CDBB01DB032FE602CA7901DB05A1
:100170002FE602CA9001C36601DB02E67FFE03CAD6
:10018000FA02FE1BCACE02473E01CDBB01C3660187
:10019000DB04E67F473E00CDBB01C36601C5060018
:1001A00005CAB901CDE801DB052FE602C2A001DBDB
:1001B00004E67F473E00CDBB01C1C9CDF901C2BBFA
:1001C00001FE01CACA0178D302C93E37D305CDE882
:1001D00001DB05E601CAD10178D304CDE801CDE801
:1001E00001F53E17D305F1C9C50602C5068005C253
:1001F000EE01C105C2EB01C1C9FE01CA0602DB0363
:100200002FE6013E00C9DB052FE6013E01C9CD08FE
:1002100003FE3AC20E02CD080307070707E6F0F512
:10022000CD0803C1D630FE0ADA2D02D6078047C9B1
:10023000C5CD9D013E01064DCDBB01CD9D010604FE
:10024000C5CD0803473E01CDBB01CD9D01C105C20F
:1002500040020 60D3E01CDBB01CD0803CD080306CB
:100260000AC5CD9D01C105C26102C1C9C5CD9D01AF
```

NORTH.ASM Object Listing (cont.)

```
:10027000CD0803473E01CDBB01CD9D01CD0803470D
:100280003E01CDBB01CD9D01060D3E01CDBB01065A
:100290000BC5CD9D01C105C29102C105C26C02CD45
:1002A0009D0106533E01CDBB01CD9D01CD9D01CDEC
:1002B0009D01CD9D01C9060A3E00CDBB01060D3E44
:1002C00000CDBB01C9110C05CD0203C36601CD39B8
:1002D00003FEFFCA3701DB03E602CAE802DB0211B4
:1002E0004005CD0203C36601CD0E023EFFA0CAC584
:1002F00002CD3002CD6C02C3D602CDB6022AAF05C4
:10030000F9C90E09CD0500C93AAD05FE80C2290321
:10031000CD4603B7CA2903F5E67F473E00CDBB01B2
:10032000118C05CD0203F137C95F16003C32AD05D3
:1003300021800019EB7E67FC9AF327C00115C00D6
:100340000E0FCD0500C9E5D5C5115C000E14CD0515
:1003500000C1D1E1C90D0A4E6F72746853746172A5
:10036000203C3D3E204E534338303020696E74654A
:1003700072666163652070726F6772616D2E0D0A1F
:100380000A436F6D6D616E642053756D6D61727996
:100390003A0D0A2020202020416C6C204E53433817
:1003A0003030204D6F6E69746F7220636F6D6D61B8
:1003B0006E6473206D617920626520697373756561
:1003C000642077697468207468652066F6C6C6F50
:1003D00077696E67200D0A2020616464697469F13
:1003E0006E732E2020312920205E5B202845534348
:1003F000617065292063617573657320746865207 9
:10040000646573696765617465642073675726388
:10041000650D0A202066696C6520746F2062652076
:10042000706C6163656420696E20746865204E534A
:1004300043383030205241 4D202873656520746860
:1004400069732070726F6772616D27730D0A2020C7
:100450006C697374696E6720666F7220667572745A
:1004600068657220696E666F726D6174696F6E295E
:100470002E2020322920205E43206361757365732E
:1004800020746869732 00D0A202070726F67726192
:100490006D20746F207465726D696E617465206182
:1004A0006E6420636F6E74726F6C20746F2062656F
:1004B000207472616E736665727265642074 6F2059
:1004C00043502F4D2E0D0A0A240D0A5741524E4912
:1004D0004E473A20204E4F20494E50555420464911
:1004E0004C452050524553454E54204F4E204449D0
:1004F000534B0D0A41424F52543F20282759272081
:100500004F522052455455524E29202407070707C1
:10051000446F6E652066696C6C696E67204E53434C
:1005200038303020627566666572207697468209D
:1005300064617461206669 6C652E0D0A07073E24AC
:1005400070D0A46696C65207472616E7366657288
:1005500020494E544552525555054454 42E070D0AD9
:10056000436F6E74726F6C20697320747261 6E7366
:1005700066657272656420746F204E534338303064
:10058000204D6F6E69746F722E0D0A240D0A072AB2
:100590002A2A2A2A46494C4520524541442045 52A0
:0D05A000524F522E2A2A2A2A2A0D0A072419
:0000000000
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0001          PSECT   ABS
                0002          NAME    WIDROW
                0004 ;
                0005 ;        THIS PROGRAM IS A VERSION OF THE WIDROW ADAPTIVE PREDICTOR
                0006 ; ALGORITHM FOR THE NSC800 SYSTEM DEVELOPED BY MAC CODY AT KANSAS
                0007 ; STATE UNIVERSITY.  THIS IS BASED UPON THE PROGRAM PRESENTED BY
                0008 ; D. NICKEL AS PART OF HIS MASTER'S THESIS IN 1979.
                0009 ;
                0010 ;        PROGRAMMER:     Dwight W. Gordon
                0011 ;                        Kansas State University
                0012 ;                        Department of Electrical Engineering
                0013 ;                        Seaton Hall
                0014 ;                        Manhattan, Kansas 66506
                0015 ;        May 1982  Rev. June 9, 1982
                0016 ;
                0017 ;        MEMORY DECLARATIONS AND DATA TABLE:
                0018 ;
>0700           0019 GLOW     EQU     0700H
>0702           0020 GHIGH    EQU     GLOW+2
>0704           0021 QM       EQU     GHIGH+2
>0706           0022 VSTARE   EQU     QM+2
>0708           0023 EM16     EQU     VSTARE+2
>0728           0024 EM       EQU     EM16+32
>072A           0025 FM16     EQU     EM+2
>074A           0026 FM       EQU     FM16+32
>074C           0027 BM16     EQU     FM+2
>076A           0028 BM1      EQU     BM16+30
>076C           0029 TBLTOP   EQU     BM1+2
                0030 ;
                0031 ;        SYSTEM CONSTANTS AND LOCATIONS:
                0032 ;
>0040           0033 SOFSET   EQU     0040H                   ;Second OFfSET
>0062           0034 IOFSET   EQU     BM16-FM16+SOFSET        ;Initial OFfSET
>0020           0035 HALFS    EQU     SOFSET/2                ;MORE OFFSETS
>0042           0036 HALFS2   EQU     HALFS-FM16+BM16         ;MORE OFFSETS
>1402           0037 ADC      EQU     1402H
>1400           0038 DAC      EQU     1400H
>1000           0039 MDU      EQU     1000H
                0040 ;
                0041 ;        PROGRAM INITIALIZATION
                0042 ;
>0400           0043          ORG     0400H
0400  FD210010  0044          LD      IY,MDU                  ;IY AS POINTER TO MDU
0404  FD360320  0045          LD      (IY+3),00100000B        ;INITIALIZE MDU
0408  066C      0046          LD      B,TBLTOP-GLOW   ;ITERATION CONSTANT
040A  AF        0047          XOR     A               ;CLEAR ACC-A
040B  216B07    0048          LD      HL,TBLTOP-1     ;TOP OF DATA TABLE TO POINTER
040E  77        0049 CLEAR    LD      (HL),A          ;CLEAR MEMORY LOCATION
040F  2B        0050          DEC     HL              ;POINT TO NEXT
0410  10FC      0051          DJNZ    CLEAR           ;REPEAT UNTIL FINISHED
```

```
                    0052 ;
                    0053 ;       BEGINNING OF PROGRAM LOOP:    DATA ACQUISITION AND SCALING.
                    0054 ;                                                          T-cycles
0412  3A0214        0055 START   LD      A,(ADC)          ;GET DATA                    13
0415  EE7F          0056         XOR     7FH              ;DATA SCALING                 7
0417  67            0057         LD      H,A              ;                             4
0418  AF            0058         XOR     A                ;FORCE LS NIBBLE TO ZERO      4
0419  CB2C          0059         SRA     H                ;SCALE SAMPLE: DIVIDE BY 2    8
041B  1F            0060         RRA                      ;                             4
041C  6F            0061         LD      L,A              ;DATA TO ACC-L                4
041D  224A07        0062         LD      (FM),HL          ;STORE SCALED F(M)           16
0420  210000        0063         LD      HL,0000H         ;CLEAR G BUFFER              10
0423  220207        0064         LD      (GHIGH),HL       ;                            16
0426  DD21EA06      0065         LD      IX,FM16-SOFSET   ;MEMORY/DATA TABLE POINTER   20
042A  0610          0066         LD      B,10H            ;LOOP CONSTANT                7
                    0067 ;                                                             ---
                    0068 ;                                                             113
                    0069 ;       FORM G(M) = SIGMA [ B(M,K) * F(M-K) ]
                    0070 ;
042C  DD6663        0071 SUM1    LD      H,(IX+IOFSET+1)  ;B(M,K) HIGH BYTE            19
042F  DD5641        0072         LD      D,(IX+SOFSET+1)  ;F(M-K) HIGH BYTE            19
0432  AF            0073 MULT1   XOR     A                ;CLEAR LS BYTE OF ONE OPERAND 4
0433  4F            0074         LD      C,A              ;                             4
0434  CB7A          0075         BIT     7,D              ;TEST FOR NEGATIVE            8
0436  2802          0076         JR      Z,NADJ11         ;                           12/7
0438  7C            0077         LD      A,H              ;                             4
0439  4C            0078         LD      C,H              ;                             4
043A  CB7C          0079 NADJ11  BIT     7,H              ;TEST FOR NEGATIVE            8
043C  2802          0080         JR      Z,NADJ21         ;                           12/7
043E  82            0081         ADD     A,D              ;                             4
043F  4F            0082         LD      C,A              ;                             4
0440  FD36036C      0083 NADJ21  LD      (IY+3),01101100B ;SETUP MDU'S                 19
0444  7A            0084         LD      A,D              ;LOAD OPERANDS TO MDU         4
0445  320010        0085         LD      (MDU),A          ;                            13
0448  AF            0086         XOR     A                ;                             4
0449  320010        0087         LD      (MDU),A          ;                            13
044C  7C            0088         LD      A,H              ;SECOND OPERAND               4
044D  320110        0089         LD      (MDU+1),A        ;                            13
0450  FD360369      0090         LD      (IY+3),69H       ;START MULTIPLY              19
0454  DD23          0091         INC     IX               ;POINT TO NEXT SET OF VALUES 10
0456  DD23          0092         INC     IX               ;                            10
0458  ED5B0207      0093         LD      DE,(GHIGH)       ;GET G HIGH BYTE FOR LATER   20
045C  3A0210        0094         LD      A,(MDU+2)        ;GET RESULT                  13
045F  91            0095         SUB     C                ;CORRECT FOR SIGNED MULTIPLY  4
0460  67            0096         LD      H,A              ;                             4
0461  3A0210        0097         LD      A,(MDU+2)        ;                            13
0464  6F            0098         LD      L,A              ;                             4
0465  19            0099         ADD     HL,DE            ;                            11
0466  220207        0100         LD      (GHIGH),HL       ;STORE HIGH WORD             16
0469  10C1          0101         DJNZ    SUM1             ;REPEAT UNTIL LAST         8/13
                    0102 ;                                                             ---
                    0103 ;                                                            4779
```

```
                0104 ;       COMPUTE E(M) = [F(M) - G(M)] / 16
                0105 ;
046B  EB        0106         EX      DE,HL           ;GHIGH TO ACC-DE                4
046C  2A4A07    0107         LD      HL,(FM)         ;F(M) TO ACC-HL                16
046F  AF        0108         XOR     A               ;CLEAR CARRY                    4
0470  ED52      0109         SBC     HL,DE           ;FORM E(M) UNSCALED            15
0472  7D        0110         LD      A,L             ;LOW BYTE OF E(M) UNSCALED      4
0473  CB2C      0111         SRA     H               ;DIVIDE BY 16 BY SHIFTING RIGHT 8
0475  1F        0112         RRA                     ;  FOUR TIMES                   4
0476  CB2C      0113         SRA     H               ;                               8
0478  1F        0114         RRA                     ;                               4
0479  CB2C      0115         SRA     H               ;                               8
047B  1F        0116         RRA                     ;                               4
047C  CB2C      0117         SRA     H               ;                               8
047E  1F        0118         RRA                     ;                               4
047F  6F        0119         LD      L,A             ;RETURN LOW BYTE OF E(M)        4
0480  222807    0120         LD      (EM),HL         ;STORE VALUE IN MEMORY         16
0483  220607    0121         LD      (VSTARE),HL     ;V-STAR-E <= E(M)              16
0486  0610      0122         LD      B,10H           ;SET UP LOOP CONSTANT           7
                0123 ;                                                             ----
                0124 ;                                                              134
                0125 ;       UPDATE THE WEIGHTS:
                0126 ;              B(M+1,K) = v * B(M,K) + u * E(M) * F(M-K)
                0127 ;       where v = 1-2**-10 and u = 2**-4
                0128 ;
0488  AF        0129         XOR     A               ;INITIALIZE TO ZERO             4
0489  C5        0130 WTCHG   PUSH    BC              ;SAVE COUNTER                  11
048A  2A0607    0131         LD      HL,(VSTARE)     ;                              16
048D  DD5621    0132         LD      D,(IX+HALFS+1)  ;LOAD HIGH BYTE OF F(M)        19
0490  4F        0133         LD      C,A             ;CLEAR LS BYTE OF ONE OPERAND   4
0491  CB7A      0134         BIT     7,D             ;TEST FOR NEGATIVE              8
0493  2802      0135         JR      Z,NADJ12        ;                            12/7
0495  7C        0136         LD      A,H             ;                               4
0496  4C        0137         LD      C,H             ;                               4
0497  CB7C      0138 NADJ12  BIT     7,H             ;TEST FOR NEGATIVE              8
0499  2802      0139         JR      Z,NADJ22        ;                            12/7
049B  82        0140         ADD     A,D             ;                               4
049C  4F        0141         LD      C,A             ;                               4
049D  FD36036C  0142 NADJ22  LD      (IY+3),01101100B ;SETUP MDU'S                  19
04A1  7A        0143         LD      A,D             ;LOAD OPERANDS TO MDU           4
04A2  320010    0144         LD      (MDU),A         ;                              13
04A5  AF        0145         XOR     A               ;                               4
04A6  320010    0146         LD      (MDU),A         ;                              13
04A9  47        0147         LD      B,A             ;INITIALIZE ACC-B FOR LATER     4
04AA  7C        0148         LD      A,H             ;SECOND OPERAND                 4
04AB  320110    0149         LD      (MDU+1),A       ;                              13
04AE  FD360369  0150         LD      (IY+3),69H      ;START MULTIPLY                19
04B2  DD6E42    0151         LD      L,(IX+HALFS2)   ;GET B(M,K) AS WE WAIT FOR THE 19
04B5  DD6643    0152         LD      H,(IX+HALFS2+1) ;  MULTIPLY TO FINISH          19
04B8  3A0210    0153         LD      A,(MDU+2)       ;GET RESULT                    13
04BB  91        0154         SUB     C               ;CORRECT FOR SIGNED MULTIPLY    4
04BC  57        0155         LD      D,A             ;                               4
```

```
04BD 3A0210   0156          LD    A,(MDU+2)          ;                          13
04C0 5F       0157          LD    E,A                ;                           4
04C1 4C       0158          LD    C,H                ;BC = V*E(M)*F(M-K)         4
04C2 CB29     0159          SRA   C                  ;SCALE ACC-C                8
04C4 CB29     0160          SRA   C                  ;                           8
04C6 F2CB04   0161          JP    P,OVERB            ;FILL ACC-B WITH SIGN BIT  10
04C9 06FF     0162          LD    B,0FFH             ;                           7
04CB AF       0163 OVERB    XOR   A                  ;CLEAR CARRY                4
04CC ED42     0164          SBC   HL,BC              ;PERFORM ADDS TO UPDATE W(M,K) 15
04CE ED5A     0165          ADC   HL,DE              ;                          15
04D0 DD7542   0166          LD    (IX+HALFS2),L      ;UPDATE MEMORY VALUE       19
04D3 DD23     0167          INC   IX                 ;POINT TO HIGH BYTE        10
04D5 DD7442   0168          LD    (IX+HALFS2),H      ;                          19
04D8 DD23     0169          INC   IX                 ;POINT TO NEXT WEIGHT      10
04DA C1       0170          POP   BC                 ;RECOVER COUNTER           10
04DB 10AC     0171          DJNZ  WTCHG              ;REPEAT UNTIL DONE        8/13
              0172 ;                                                            ----
              0173 ;                                                            6671
              0174 ;        FORM ERROR
              0175 ;
04DD 2A0807   0176          LD    HL,(EM16)          ;                          16
04E0 EB       0177          EX    DE,HL              ;                           4
04E1 2A2807   0178          LD    HL,(EM)            ;                          16
04E4 AF       0179          XOR   A                  ;CLEAR CARRY                4
04E5 ED52     0180          SBC   HL,DE              ;                          15
04E7 EB       0181          EX    DE,HL              ;                          15
04E8 2A0407   0182          LD    HL,(QM)            ;                          16
04EB 19       0183          ADD   HL,DE              ;                          15
04EC 220407   0184          LD    (QM),HL            ;                          16
04EF 54       0185          LD    D,H                ;COPY Q(M) TO ACC-DE        4
04F0 5D       0186          LD    E,L                ;                           4
04F1 4F       0187          LD    C,A                ;CLEAR LS BYTE OF OPERAND   4
04F2 CB7A     0188          BIT   7,D                ;TEST FOR NEGATIVE          8
04F4 2803     0189          JR    Z,NADJ13           ;                         12/7
04F6 7C       0190          LD    A,H                ;MODIFY FUDGE REGISTER  IF  4
04F7 82       0191          ADD   A,D                ;  NEGATIVE                 4
04F8 4F       0192          LD    C,A                ;                           4
04F9 FD36036C 0193 NADJ13   LD    (IY+3),01101100B   ;SETUP MDU'S               19
04FD 7A       0194          LD    A,D                ;LOAD OPERANDS TO MDU       4
04FE 320010   0195          LD    (MDU),A            ;                          13
0501 AF       0196          XOR   A                  ;                           4
0502 320010   0197          LD    (MDU),A            ;                          13
0505 7C       0198          LD    A,H                ;SECOND OPERAND             4
0506 320110   0199          LD    (MDU+1),A          ;                          13
0509 FD360369 0200          LD    (IY+3),69H         ;START MULTIPLY            19
050D D9       0201          EXX                      ;SAVE ENVIRONMENT           4
```

```
                    0202 ;
                    0203 ;
                    0204 ;          ROTATE THE CIRCULAR BUFFERS (WAIT ON MULTIPLY)
                    0205 ;
050E  110807        0206            LD      DE,EM16         ;                       10
0511  210A07        0207            LD      HL,EM16+2       ;                       10
0514  014400        0208            LD      BC,BM16-EM16    ;                       10
0517  EDB0          0209            LDIR                    ;                    16/21
                    0210 ;                                                      ------
                    0211 ;                                                        1453
                    0212 ;
                    0213 ;          CONTINUE FORMING THE ERROR
                    0214 ;
0519  D9            0215            EXX                     ;RESTORE THE ENVIRONMENT    4
051A  3A0210        0216            LD      A,(MDU+2)       ;GET RESULT                13
051D  91            0217            SUB     C               ;CORRECT FOR SIGNED MULTIPLY    4
051E  67            0218            LD      H,A             ;                           4
051F  3A0210        0219            LD      A,(MDU+2)       ;                          13
0522  6F            0220            LD      L,A             ;                           4
0523  7C            0221            LD      A,H             ;SCALE Q(M)**2              4
0524  CB25          0222            SLA     L               ;                           8
0526  17            0223            RLA                     ;                           4
0527  CB25          0224            SLA     L               ;                           8
0529  17            0225            RLA                     ;                           4
052A  CB25          0226            SLA     L               ;                           8
052C  17            0227            RLA                     ;                           4
052D  CB25          0228            SLA     L               ;                           8
052F  17            0229            RLA                     ;                           4
0530  320014        0230            LD      (DAC),A         ;OUTPUT RESULT TO DAC      13
0533  C31204        0231            JP      START           ;                          10
                    0232 ;                                                      ------
                    0233 ;                                                         366
                    0234 ;
                    0235 ;
                    0236            END
```

Symbol Table

```
ADC             1402 BM1        076A BM16       074C CLEAR      040E
DAC             1400 EM         0728 EM16       0708 FM         074A
FM16            072A GHIGH      0702 GLOW       0700 HALFS      0020
HALFS2          0042 IOFSET     0062 MDU        1000 MULT1      0432
NADJ11          043A NADJ12     0497 NADJ13     04F9 NADJ21     0440
NADJ22          049D OVERB      04CB QM         0704 SOFSET     0040
START           0412 SUM1       042C TBLTOP     076C VSTARE     0706
WTCBG           0489
```

CROSS REFERENCE LISTING

SYMBOL	VALUE	TYPE	STMT	STATEMENT REFERENCES							
ADC	1402		0037	0055							
BM1	076A		0028	0029							
BM16	074C		0027	0208	0036	0034	0028				
CLEAR	040E		0049	0051							
DAC	1400		0038	0230							
EM	0728		0024	0178	0120	0025					
EM16	0708		0023	0208	0207	0206	0176	0024			
FM	074A		0026	0107	0062	0027					
FM16	072A		0025	0065	0036	0034	0026				
GHIGH	0702		0020	0100	0093	0064	0021				
GLOW	0700		0019	0046	0020						
HALFS	0020		0035	0132	0036						
HALFS2	0042		0036	0168	0166	0152	0151				
IOFSET	0062		0034	0071							
MDU	1000		0039	0219	0216	0199	0197	0195	0156	0153	0149
-				0146	0144	0097	0094	0089	0087	0085	0044
MULT1	0432		0073								
NADJ11	043A		0079	0076							
NADJ12	0497		0138	0135							
NADJ13	04F9		0193	0189							
NADJ21	0440		0083	0080							
NADJ22	049D		0142	0139							
OVERB	04CB		0163	0161							
QM	0704		0021	0184	0182	0022					
SOFSET	0040		0033	0072	0065	0035	0034				
START	0412		0055	0231							
SUM1	042C		0071	0101							
TBLTOP	076C		0029	0048	0046						
VSTARE	0706		0022	0131	0121	0023					
WTCHG	0489		0130	0171							

ERRORS=0000

Program Object Listing

```
∫WIDROW05005B
:20040000FD210010FD360320066CAF216B07772B10FC3A0214EE7F67AFCB2C1F6F224A072B
:2004200021000022020 7DD21EA060610DD6663DD5641AF4FCB7A28027C4CCB7C2802824FDB
:20044000FD36036C7A320010AF3200107C320110FD360369DD23DD23ED5B02073A021091C1
:20046000673A02106F1922020710C1EB2A4A07AFED527DCB2C1FCB2C1FCB2C1FCB2C1F6FAD
:2004800022280722060706 10AFC52A0607DD56214FCB7A28027C4CCB7C2802824FFD3603C9
:2004A0006C7A320010AF320010477C320110FD360369DD6E42DD66433A021091573A0210EB
:2004C0005F4CCB29CB29F2CB0406FFAFED42ED5ADD7542DD23DD7442DD23C110AC2A0807C1
:2004E000EB2A2807AFED52EB2A0407192204075 45D4FCB7A28037C824FFD36036C7A320059
:2005000010AF3200107C320110FD360369D9110807210A07014400EDB0D93A021091673A18
:1605200002106F7CCB2517CB2517CB2517CB2517320014C312048D
:00040001FB
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0001         PSECT   ABS
                0002         NAME    WIDROW
                0003 ;
                0004 ;       THIS PROGRAM IS A VERSION OF THE WIDROW ADAPTIVE PREDICTOR
                0005 ; ALGORITHM AS IT APPLIES TO THE NSC800 SYSTEM DEVELOPED BY MAC
                0006 ; CODY AT KANSAS STATE UNIVERSITY.  THIS IS BASED UPON THE PROGRAM
                0007 ; PRESENTED BY D. NICKEL AS PART OF HIS MASTER'S THESIS IN 1979.
                0008 ;
                0009 ;       PROGRAMMER:     Dwight W. Gordon
                0010 ;                       Kansas State University
                0011 ;                       Department of Electrical Engineering
                0012 ;                       Seaton Hall
                0013 ;                       Manhattan, Kansas 66506
                0014 ;       May 1982  Rev. June 28,1982
                0015 ;
                0016 ;       MEMORY DECLARATIONS AND DATA TABLE:
                0017 ;
>0700           0018 GLOW    EQU     0700H
>0702           0019 GHIGH   EQU     GLOW+2
>0704           0020 QM      EQU     GHIGH+2
>0706           0021 VSTARE  EQU     QM+2
>0708           0022 EM16    EQU     VSTARE+2
>0728           0023 EM      EQU     EM16+32
>072A           0024 FM16    EQU     EM+2
>074A           0025 FM      EQU     FM16+32
>074C           0026 BM16    EQU     FM+2
>076A           0027 BM1     EQU     BM16+30
>076C           0028 TBLTOP  EQU     BM1+2
                0029 ;
                0030 ;       SYSTEM CONSTANTS AND LOCATIONS:
                0031 ;
>0040           0032 SOFSET  EQU     0040H                   ;Second OFfSET
>0062           0033 IOFSET  EQU     BM16-FM16+SOFSET        ;Initial OFfSET
>0020           0034 HALFS   EQU     SOFSET/2                ;MORE OFFSETS
>0042           0035 HALFS2  EQU     HALFS-FM16+BM16         ;MORE OFFSETS
>1402           0036 ADC     EQU     1402H
>1400           0037 DAC     EQU     1400H
>1000           0038 MDU     EQU     1000H
                0039 ;
                0040 ;       PROGRAM INITIALIZATION
                0041 ;
>0400           0042         ORG     0400H
0400  FD210010  0043         LD      IY,MDU                  ;IY AS POINTER TO MDU
0404  FD360320  0044         LD      (IY+3),00100000B        ;INITIALIZE MDU
0408  066C      0045         LD      B,TBLTOP-GLOW   ;ITERATION CONSTANT
040A  AF        0046         XOR     A               ;CLEAR ACC-A
040B  216B07    0047         LD      HL,TBLTOP-1     ;TOP OF DATA TABLE TO POINTER
040E  77        0048 CLEAR   LD      (HL),A          ;CLEAR MEMORY LOCATION
040F  2B        0049         DEC     HL              ;POINT TO NEXT
0410  10FC      0050         DJNZ    CLEAR           ;REPEAT UNTIL FINISHED
```

```
                    0051 ;
                    0052 ;       BEGINNING OF PROGRAM LOOP:    DATA ACQUISITION AND SCALING.
                    0053 ;
0412  3A0214        0054 START   LD      A,(ADC)          ;GET DATA                      13
0415  EE7F          0055         XOR     7FH              ;DATA SCALING                   7
0417  67            0056         LD      H,A              ;DATA TO ACC-H                  4
0418  3A0114        0057         LD      A,(ADC-1)        ;GET LS NIBBLE                 13
041B  EEF0          0058         XOR     0F0H             ;SCALE                          7
041D  CB2C          0059         SRA     H                ;SCALE SAMPLE: DIVIDE BY 2      8
041F  1F            0060         RRA                      ;                               4
0420  6F            0061         LD      L,A              ;DATA TO ACC-L                  4
0421  224A07        0062         LD      (FM),HL          ;STORE SCALED F(M)             16
0424  210000        0063         LD      HL,0000H         ;CLEAR G BUFFER                10
0427  220207        0064         LD      (GHIGH),HL       ;                              16
042A  DD21EA06      0065         LD      IX,FM16-SOFSET   ;MEMORY/DATA TABLE POINTER     20
042E  312A07        0066         LD      SP,FM16          ;POINT TO TABLE                10
0431  0610          0067         LD      B,10H            ;LOOP CONSTANT                  7
                    0068 ;                                                               ---
                    0069 ;                                                               139
                    0070 ;       FORM G(M) = SIGMA [ B(M,K) * F(M-K) ]
                    0071 ;
0433  D9            0072 SUM1    EXX                      ;SAVE COUNTER                   4
0434  DD6E62        0073         LD      L,(IX+IOFSET)    ;B(M,K) LOW BYTE               19
0437  DD23          0074         INC     IX               ;POINT TO HIGH BYTES           10
0439  DD6662        0075         LD      H,(IX+IOFSET)    ;B(M,K) HIGH BYTE              19
043C  D1            0076         POP     DE               ;F(M-K)                        10
043D  DD23          0077         INC     IX               ;POINT TO NEXT SET OF VALUES   10
043F  FD36036C      0078         LD      (IY+3),01101100B ;SETUP MDU'S                   19
0443  7A            0079         LD      A,D              ;LOAD OPERANDS TO MDU           4
0444  320010        0080         LD      (MDU),A          ;                              13
0447  7B            0081         LD      A,E              ;                               4
0448  320010        0082         LD      (MDU),A          ;                              13
044B  7C            0083         LD      A,H              ;SECOND OPERAND                 4
044C  320110        0084         LD      (MDU+1),A        ;                              13
044F  7D            0085         LD      A,L              ;                               4
0450  320110        0086         LD      (MDU+1),A        ;                              13
0453  FD360369      0087         LD      (IY+3),69H       ;START MULTIPLY                19
0457  010000        0088 MULT1   LD      BC,0000H         ;INITIALIZE FUDGE REGISTER     10
045A  CB7C          0089         BIT     7,H              ;TEST FOR NEGATIVE              8
045C  2802          0090         JR      Z,NOAD11         ;                            12/7
045E  42            0091         LD      B,D              ;                               4
045F  4B            0092         LD      C,E              ;                               4
0460  CB7A          0093 NOAD11  BIT     7,D              ;TEST FOR NEGATIVE              8
0462  2803          0094         JR      Z,NOAD21         ;                            12/7
0464  09            0095         ADD     HL,BC            ;                              11
0465  44            0096         LD      B,H              ;                               4
0466  4D            0097         LD      C,L              ;                               4
0467  AF            0098 NOAD21  XOR     A                ;CLEAR Carry                    4
0468  3A0210        0099         LD      A,(MDU+2)        ;GET RESULT                    13
046B  67            0100         LD      H,A              ;PLACE IN HL PAIR               4
046C  3A0210        0101         LD      A,(MDU+2)        ;                              13
046F  6F            0102         LD      L,A              ;                               4
```

```
0470 ED42     0103          SBC     HL,BC           ;CORRECT FOR SIGNED MULTIPLY   11
0472 ED4B0207 0104          LD      BC,(GHIGH)      ;UPDATE HIGH WORD              20
0476 09       0105          ADD     HL,BC           ;                              11
0477 220207   0106          LD      (GHIGH),HL      ;STORE HIGH WORD               16
047A D9       0107          EXX                     ;RECOVER COUNTER                4
047B 10B6     0108          DJNZ    SUM1            ;REPEAT UNTIL LAST            8/13
              0109 ;                                                               -----
              0110 ;                                                               5723
              0111 ;        COMPUTE E(M) = [F(M) - G(M)] / 16
              0112 ;
047D D9       0113          EXX                     ;RECOVER GHIGH                  4
047E EB       0114          EX      DE,HL           ;GHIGH TO ACC-DE                4
047F 2A4A07   0115          LD      HL,(FM)         ;F(M) TO ACC-HL                16
0482 AF       0116          XOR     A               ;CLEAR CARRY                    4
0483 ED52     0117          SBC     HL,DE           ;FORM E(M) UNSCALED            15
0485 7D       0118          LD      A,L             ;LOW BYTE OF E(M) UNSCALED      4
0486 CB2C     0119          SRA     H               ;DIVIDE BY 16 BY SHIFTING RIGHT 8
0488 1F       0120          RRA                     ;  FOUR TIMES                   4
0489 CB2C     0121          SRA     H               ;                               8
048B 1F       0122          RRA                     ;                               4
048C CB2C     0123          SRA     H               ;                               8
048E 1F       0124          RRA                     ;                               4
048F CB2C     0125          SRA     H               ;                               8
0491 1F       0126          RRA                     ;                               4
0492 6F       0127          LD      L,A             ;RETURN LOW BYTE OF E(M)        4
0493 222807   0128          LD      (EM),HL         ;STORE VALUE IN MEMORY         16
0496 220607   0129          LD      (VSTARE),HL     ;V-STAR-E <= E(M)              16
0499 E1       0130          POP     HL              ;ADJUST POINTER                10
049A 0610     0131          LD      B,10H           ;SET UP LOOP CONSTANT           7
              0132 ;                                                               -----
              0133 ;                                                               148
              0134 ;        UPDATE THE WEIGHTS:
              0135 ;                B(M+1,K) = u * B(M,K) + v * E(M) * F(M-K)
              0136 ;                where u = 1-2**-10 and v = 2**-4
              0137 ;
049C D9       0138 WTCHG    EXX                     ;SAVE COUNTER                  11
049D 2A0607   0139          LD      HL,(VSTARE)     ;                              16
04A0 DD5E20   0140          LD      E,(IX+HALFS)    ;LOAD F(M)                     19
04A3 DD5621   0141          LD      D,(IX+HALFS+1)  ;                              19
04A6 FD36036C 0142          LD      (IY+3),01101100B ;SETUP MDU'S                  19
04AA 7A       0143          LD      A,D             ;LOAD OPERANDS TO MDU           4
04AB 320010   0144          LD      (MDU),A         ;                              13
04AE 7B       0145          LD      A,E             ;                               4
04AF 320010   0146          LD      (MDU),A         ;                              13
04B2 7C       0147          LD      A,H             ;SECOND OPERAND                 4
04B3 320110   0148          LD      (MDU+1),A       ;                              13
04B6 7D       0149          LD      A,L             ;                               4
04B7 320110   0150          LD      (MDU+1),A       ;                              13
04BA FD360369 0151          LD      (IY+3),69H      ;START MULTIPLY                19
04BE 010000   0152 MULT2    LD      BC,0000H        ;INITIALIZE FUDGE REGISTER     10
04C1 CB7C     0153          BIT     7,H             ;TEST FOR NEGATIVE              8
04C3 2802     0154          JR      Z,NOAD12        ;                            12/7
```

```
04C5  42        0155          LD      B,D             ;                          4
04C6  4B        0156          LD      C,E             ;                          4
04C7  CB7A      0157 NOAD12   BIT     7,D             ;TEST FOR NEGATIVE         8
04C9  2803      0158          JR      Z,NOAD22        ;                       12/7
04CB  09        0159          ADD     HL,BC           ;                         11
04CC  44        0160          LD      B,H             ;                          4
04CD  4D        0161          LD      C,L             ;                          4
04CE  AF        0162 NOAD22   XOR     A               ;CLEAR Carry               4
04CF  3A0210    0163          LD      A,(MDU+2)       ;GET RESULT               13
04D2  67        0164          LD      H,A             ;PLACE IN HL PAIR          4
04D3  3A0210    0165          LD      A,(MDU+2)       ;                         13
04D6  6F        0166          LD      L,A             ;                          4
04D7  ED42      0167          SBC     HL,BC           ;CORRECT FOR SIGNED MULTIPLY 11
04D9  D1        0168          POP     DE              ;LOAD B(M,K)              10
04DA  EB        0169          EX      DE,HL           ;DE=v*E(M)*F(M-K); HL=B(M,K)  4
04DB  0600      0170          LD      B,00H           ;CLEAR ACC-B               7
04DD  4C        0171          LD      C,H             ;BC = v*E(M)*F(M-K)        4
04DE  CB29      0172          SRA     C               ;SCALE ACC-C               8
04E0  CB29      0173          SRA     C               ;                          8
04E2  F2E704    0174          JP      P,OVERB         ;FILL ACC-B WITH SIGN BIT 10
04E5  06FF      0175          LD      B,0FFH          ;                          7
04E7  AF        0176 OVERB    XOR     A               ;CLEAR CARRY               4
04E8  ED42      0177          SBC     HL,BC           ;HL = B(M,K) * (1-2**-10)  15
04EA  19        0178          ADD     HL,DE           ;HL = u*B(M,K) + v*E(M)*F(M-K) 11
04EB  E5        0179          PUSH    HL              ;UPDATE MEMORY VALUE      11
04EC  E1        0180          POP     HL              ;POINT TO NEXT WEIGHT     10
04ED  DD23      0181          INC     IX              ;POINT TO NEXT WEIGHT     10
04EF  DD23      0182          INC     IX              ;POINT TO NEXT WEIGHT     10
04F1  D9        0183          EXX                     ;RECOVER COUNTER           4
04F2  10A8      0184          DJNZ    WTCHG           ;REPEAT UNTIL DONE      8/13
                0185 ;                                                          -----
                0186 ;                                                          6811
                0187 ;        FORM ERROR
                0188 ;
04F4  2A0807    0189          LD      HL,(EM16)       ;                         16
04F7  EB        0190          EX      DE,HL           ;                          4
04F8  2A2807    0191          LD      HL,(EM)         ;                         16
04FB  AF        0192          XOR     A               ;CLEAR CARRY               4
04FC  ED52      0193          SBC     HL,DE           ;                         15
04FE  EB        0194          EX      DE,HL           ;                         15
04FF  2A0407    0195          LD      HL,(QM)         ;                         16
0502  19        0196          ADD     HL,DE           ;                         15
0503  220407    0197          LD      (QM),HL         ;                         16
0506  FD36036C  0198          LD      (IY+3),01101100B ;SETUP MDU'S             19
050A  7C        0199          LD      A,H             ;LOAD OPERANDS TO MDU      4
050B  320010    0200          LD      (MDU),A         ;                         13
050E  7D        0201          LD      A,L             ;                          4
050F  320010    0202          LD      (MDU),A         ;                         13
0512  7C        0203          LD      A,H             ;SECOND OPERAND            4
0513  320110    0204          LD      (MDU+1),A       ;                         13
0516  7D        0205          LD      A,L             ;                          4
0517  320110    0206          LD      (MDU+1),A       ;                         13
```

```
051A FD360369 0207         LD    (IY+3),69H    ;START MULTIPLY              19
051E 010000   0208 MULT3   LD    BC,0000H      ;INITIALIZE FUDGE REGISTER   10
0521 CB7C     0209         BIT   7,H           ;TEST FOR NEGATIVE            8
0523 2805     0210         JR    Z,NOADJ2      ;                          12/7
0525 44       0211         LD    B,H           ;                             4
0526 4D       0212         LD    C,L           ;                             4
0527 09       0213         ADD   HL,BC         ;                            11
0528 44       0214         LD    B,H           ;                             4
0529 4D       0215         LD    C,L           ;                             4
052A AF       0216 NOADJ2  XOR   A             ;CLEAR Carry                  4
052B 3A0210   0217         LD    A,(MDU+2)     ;GET RESULT                  13
052E 67       0218         LD    H,A           ;PLACE IN HL PAIR             4
052F 3A0210   0219         LD    A,(MDU+2)     ;                            13
0532 6F       0220         LD    L,A           ;                             4
0533 ED42     0221         SBC   HL,BC         ;CORRECT FOR SIGNED MULTIPLY 11
0535 7C       0222         LD    A,H           ;SCALE Q(M)**2                4
0536 CB25     0223         SLA   L             ;                             8
0538 17       0224         RLA                 ;                             4
0539 CB25     0225         SLA   L             ;                             8
053B 17       0226         RLA                 ;                             4
053C CB25     0227         SLA   L             ;                             8
053E 17       0228         RLA                 ;                             4
053F CB25     0229         SLA   L             ;                             8
0541 17       0230         RLA                 ;                             4
0542 320014   0231         LD    (DAC),A       ;OUTPUT RESULT TO DAC        13
              0232 ;                                                        ----
              0233 ;                                                         389
              0234 ;       ROTATE THE CIRCULAR BUFFERS
              0235 ;
0545 110807   0236         LD    DE,EM16       ;                            10
0548 210A07   0237         LD    HL,EM16+2     ;                            10
054B 014400   0238         LD    BC,BM16-EM16  ;                            10
054E EDB0     0239         LDIR                ;                          16/21
0550 C31204   0240         JP    START         ;                            10
              0241 ;                                                        ----
              0242 ;                                                        1463
              0243 ;
              0244 ;
              0245         END
```

Symbol Table

```
ADC      1402 BM1    076A BM16   074C CLEAR  040E
DAC      1400 EM     0728 EM16   0708 FM     074A
FM16     072A GHIGH  0702 GLOW   0700 HALFS  0020
HALFS2   0042 IOFSET 0062 MDU    1000 MULT1  0457
MULT2    04BE MULT3  051E NOAD11 0460 NOAD12 04C7
NOAD21   0467 NOAD22 04CE NOADJ2 052A OVERB  04E7
QM       0704 SOFSET 0040 START  0412 SUM1   0433
TBLTOP   076C VSTARE 0706 WTCHG  049C
```

CROSS REFERENCE LISTING

SYMBOL	VALUE	TYPE	STMT	STATEMENT REFERENCES
ADC	1402		0036	0057 0054
BM1	076A		0027	0028
BM16	074C		0026	0238 0035 0033 0027
CLEAR	040E		0048	0050
DAC	1400		0037	0231
EM	0728		0023	0191 0128 0024
EM16	0708		0022	0238 0237 0236 0189 0023
FM	074A		0025	0115 0062 0026
FM16	072A		0024	0066 0065 0035 0033 0025
GHIGH	0702		0019	0106 0104 0064 0020
GLOW	0700		0018	0045 0019
HALFS	0020		0034	0141 0140 0035
HALFS2	0042		0035	
IOFSET	0062		0033	0075 0073
MDU	1000		0038	0219 0217 0206 0204 0202 0200 0165 0163
-				0150 0148 0146 0144 0101 0099 0086 0084
-				0082 0080 0043
MULT1	0457		0088	
MULT2	04BE		0152	
MULT3	051E		0208	
NOAD11	0460		0093	0090
NOAD12	04C7		0157	0154
NOAD21	0467		0098	0094
NOAD22	04CE		0162	0158
NOADJ2	052A		0216	0210
OVERB	04E7		0176	0174
QM	0704		0020	0197 0195 0021
SOFSET	0040		0032	0065 0034 0033
START	0412		0054	0240
SUM1	0433		0072	0108
TBLTOP	076C		0028	0047 0045
VSTARE	0706		0021	0139 0129 0022
WTCHG	049C		0138	0184

ERRORS=0000

Program Object Listing

```
∫WIDROW05005B
:20040000FD210010FD360320066CAF216B07772B10FC3A0214EE7F673A0114EEF0CB2C1F8F
:200420006F224A07210000220207DD21EA06312A07061,0D9DD6E62DD23DD6662D1DD23FD2F
:2004400036036C7A3200107B3200107C3201107D320110FD360369010000CB7C2802424B61
:20046000CB7A280309444DAF3A0210673A02106FED42ED4B020709220207D910B6D9EB2A24
:200480004A07AFED527DCB2C1FCB2C1FCB2C1FCB2C1F6F222807220607E10610D92A060752
:2004A000DD5E20DD5621FD36036C7A3200107B3200107C3201107D320110FD360369010053
:2004C00000CB7C2802424BCB7A280309444DAF3A0210673A02106FED42D1EB06004CCB29C6
:2004E000CB29F2E70406FFAFED4219E5E1DD23DD23D910A82A0807EB2A2807AFED52EB2A58
:20050000040719220407FD36036C7C3200107D3200107C3201107D320110FD36036901004C
:2005200000CB7C2805444D09444DAF3A0210673A02106FED427CCB2517CB2517CB2517CB74
:13054000251732001411080721 0A07014400EDB0C3120419
:00040001FB
```

This program is provided as a means of emulating the NSC800 monitor on the NorthStar Horizon. The user is allowed to utilize the NorthStar Horizon as a software debugging tool for programs to be implemented on the NSC800 board. A working knowledge of CP/M and the NSC800 monitor is assumed.

Most of the features of the NSC800 monitor are available on this emulation. Exceptions include the MDUs, ADC and DAC. The MDUs may be emulated by appropriate software patched onto the user program as a subroutine. The ADC may be implemented by a software routine which sequentially accesses a data file which may be LINKed* to the user program. The only substitute which the author has utilized for the DAC is a block of sequential memory to which the program may sequentially write. This, too, may be implemented in a subroutine patched onto the end of the user program.

The first step in system utilization is the powering-up of all of the equipment. CP/M will respond with the appropriate wake-up banner to indicate that the system is responsive. More information on power-up may be obtained from the appropriate manuals[7,8,10].

It is assumed that the user wishes to use this program as a software development tool. Under this assumption the appropriate file of type "OBJ" must be created. {Author's note: The emulator will execute properly without a user program patched onto the emulator program.} This file must be created with the use of the ZASM assembly of a Z80 source file.

Each time that a different source file is to be tested a new

* LINK is a CP/M utility for ZASM use.

file of type "COM" must be created. To do this it is necessary to utilize the LINK utility. For more information see the appropriate manual[9]. The user program must not be loaded over the emulator program. This means that in order to transfer the user program listing to a listing which may be run on the NSC800 board it is necessary to change the ORG statement in the user program. {Author's note: It was much easier to make the user program origin at 1400H. This means that all addresses utilized in the emulation were exactly 1000H greater than those accepted by the NSC800.} All restrictions imposed by the LINK program are in effect at this time.

Complete documentation for the NSC800 monitor may be found in Appendix B of this report. To exit the NSC800 monitor emulator it is necessary to implement a Go to location 0000H. This causes a warm start of the CP/M operating system.

Hardware Block Diagram

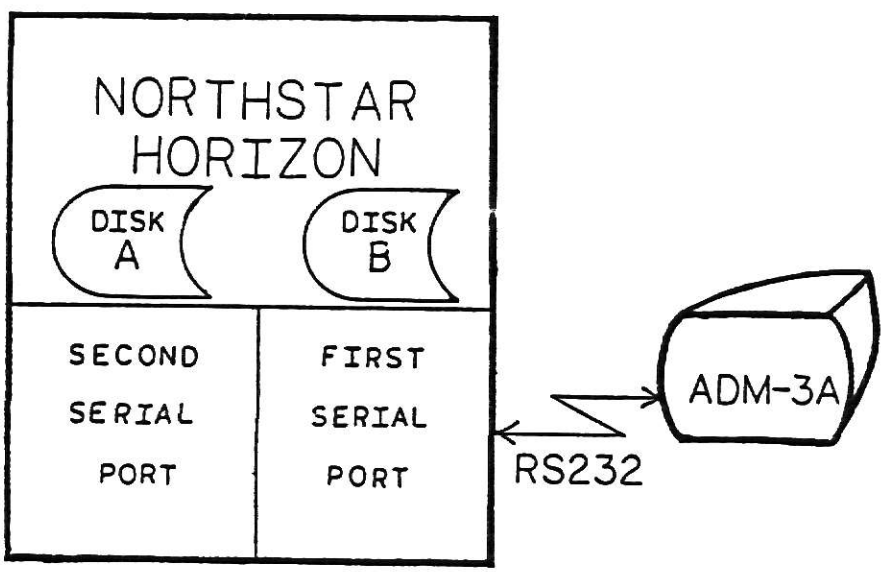


Figure E-1.

```
ADDR  CODE      STMT SOURCE STATEMENT

                0001         NAME    NSMTR
                0002         PSECT   ABS
                0004 ;#############################################################
                0005 ;#NSC800 <=> ADM-3A SYSTEM MONITOR                            #
                0006 ;#PROGRAMMER:   DWIGHT W. GORDON          3-18-82            #
                0007 ;#KANSAS STATE UNIVERSITY                 Rev. 6-21-82       #
                0008 ;#DEPARTMENT OF ELECTRICAL ENGINEERING                       #
                0009 ;#SEATON HALL                                                #
                0010 ;#MANHATTAN, KANSAS 66506                                    #
                0011 ;#############################################################
                0012 ;#    THIS PROGRAM HAS BEEN DESIGNED AS A MINIMAL MONITOR#
                0013 ;# FOR THE NEWLY DEVELOPED NSC800 uP.                       #
                0014 ;#     THIS PROGRAM HAS NOT BEEN MINIMIZED OR OPTIMIZED. #
                0015 ;# USE IT AT YOUR OWN RISK.  NORTHSTAR TEST                 #
                0016 ;#############################################################
>1C00           0017 PORT2  EQU   1C00H       ;INPUT/OUTPUT PORT
>0400           0018 RAMST  EQU   0400H       ;STARTING LOCATION OF SYSTEM RAM
>0800           0019 RAMEND EQU   0800H       ;END OF SYSTEM RAM + 1
>07FD           0020 SCRTCH EQU   RAMEND-3    ;AREA FOR ASYNC COMM'CATION CONST
>07FB           0021 BREAKA EQU   SCRTCH-2    ;BREAKPOINT ADDRESS
>07FA           0022 CODE   EQU   BREAKA-1    ;BREAKPOINT CODE ADDRESS
>07F8           0023 FIX    EQU   CODE-2      ;SCRATCHPAD MEMORY FOR MONITOR
>07F7           0024 STACK  EQU   FIX-1       ;MONITOR BOTTOM OF STACK
>003E           0025 CURSOR EQU   3EH         ;">"
>0008           0026 CMDLEN EQU   08H         ;NUMBER OF COMMANDS ACCEPTED
>000A           0027 LF     EQU   0AH         ;LINE FEED
>000D           0028 CR     EQU   0DH         ;CARRIAGE RETURN
>0007           0029 BELL   EQU   07H         ;BELL
>0020           0030 SPACE  EQU   20H         ;ASCII SPACE
>008D           0031 CREND  EQU   CR+80H      ;END OF OUTPUT CR DELIMITER
>00BE           0032 CUREND EQU   CURSOR+80H  ;END OF OUTPUT CURSOR DELIMITER
>0087           0033 BELEND EQU   BELL+80H    ;END OF OUTPUT BELL DELIMITER
>008A           0034 LFEND  EQU   LF+80H      ;END OF OUTPUT LF DELIMITER
>00A0           0035 SPCEND EQU   SPACE+80H   ;END OF OUTPUT SPACE DELIMITER
                0036 ;#############################################################
                0037 ;#############################################################
>0100           0038         ORG   0100H
0100  31F707     0039         LD    SP,STACK   ;INITIALIZE MONITOR STACK
0103  FB         0040         EI               ;ENABLE INTERRUPTS
0104  3EC3       0041         LD    A,0C3H     ;JUMP COMMAND#####################
0106  323800     0042         LD    (0038H),A  ;SWI VECTOR INIT################
0109  21CF01     0043         LD    HL,RESET   ;JUMP LOCATION#################
010C  223900     0044         LD    (0039H),HL ;SWI###########################
010F  C3C101     0045         JP    MAIN       ;BEGIN EXECUTION OF MAIN PROGRAM
                0046 ;
                0047 ;#############################################################
                0048 ;       ERROR MESSAGE
                0049 ;
0112  3F3F3F     0050 ERRMES DEFM  '???'
0115  87         0051         DEFB  BELEND
                0052 ;
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0053 ;#########################################################
                0054 ;      MESSAGE USED BY COPY ROUTINE
                0055 ;
0116  544F3F    0056 TO     DEFM  'TO?'
0119  A0        0057        DEFB  SPCEND
                0058 ;
                0059 ;#########################################################
                0060 ;      NEWLINE DATA WITH CURSOR PROMPT
                0061 ;
011A  0A0DBE    0062 LFCRCU DEFB  LF,CR,CUREND
                0063 ;
                0064 ;#########################################################
                0065 ;#           ASCII <=> BINARY LOOKUP TABLE               #
                0066 ;#########################################################
>0140           0067        ORG   0140H
0140  30313233  0068 ASCII  DEFM  '0123456789ABCDEF'
      34353637
      38394142
      43444546
                0069 ;
                0070 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                0071 ;      DATA TRANSFER ERROR MESSAGE
                0072 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
0150  3130      0073 ERRTEN DEFM  '10'
0152  2D53544F  0074 ERRTIN DEFM  '-STOP BIT EXPECTED'
      50204249
      54204558
      50454354
      4544
0164  87        0075        DEFB  BELEND
                0076 ;
                0077 ;#########################################################
                0078 ;      MORE DATA FOR COPY ROUTINE
                0079 ;
0165  0A0D      0080 FROM   DEFB  LF,CR
0167  46524F4D  0081        DEFM  'FROM?'
      3F
016C  A0        0082        DEFB  SPCEND
016D  4C454E47  0083 LENGTH DEFM  'LENGTH?'
      54483F
0174  A0        0084        DEFB  SPCEND
                0085 ;
                0086 ;#########################################################
                0087 ;#     COMMAND DECODING AND LOOKUP ADDRESS TABLE          #
                0088 ;#########################################################
0175  4D        0089 CMDTBL DEFM  'M'         ;view and alter Memory
0176  9303      0090        DEFW  MEMCHG
0178  47        0091        DEFM  'G'         ;Go a user defined program
0179  3202      0092        DEFW  GOPROG
017B  56        0093        DEFM  'V'         ;View a block of memory
017C  FB02      0094        DEFW  MEMVUE
017E  52        0095        DEFM  'R'         ;view existing Register values
```

```
ADDR  CODE      STMT SOURCE STATEMENT

017F  8B02      0096        DEFW REGLOK
0181  43        0097        DEFM  'C'        ;Copy a block of data
0182  B302      0098        DEFW  COPY
0184  4E        0099        DEFM  'N'        ;Next command from software intrpt
0185  EC02      0100        DEFW  RESTAR
0187  42        0101        DEFM  'B'        ;SET A SINGLE BREAKPOINT
0188  E503      0102        DEFW  BRKPT      ;
018A  0D        0103        DEFB  CR         ;CR AS TERMINAL
                0104 ;
                0105 ;############WAKEUP BANNER MESSAGE#########################
018B  4E534338  0106 BANNER DEFM  'NSC800 MONITOR VERSION 1.91'
      3030204D
      4F4E4954
      4F522056
      45525349
      4F4E2031
      2E3931
01A6  0A0D      0107        DEFB  LF,CR
01A8  4B414E53  0108        DEFM  'KANSAS STATE UNIVERSITY'
      41532053
      54415445
      20554E49
      56455253
      495459
01BF  0A8D      0109 NEWLIN DEFB  LF,CREND
                0110 ;
                0111 ;############################################################
                0112 ;############################################################
                0113 ;##  BEGINNING OF THE MAIN PROGRAM                         ##
                0114 ;############################################################
                0115 ;############################################################
                0116 ;#  WARNING:  Routines invoked from the main program are
                0117 ;#     NOT subroutines and may NOT be used as such by the
                0118 ;#     user.  There are, however, many subroutines which
                0119 ;#     may be called by the user.  Care should be taken
                0120 ;#     to insure that registers disturbed by a called
                0121 ;#     subroutine, which contain information prior to the
                0122 ;#     call, are pushed in the user main program prior to
                0123 ;#     calling the desired subroutine.
                0124 ;#
01C1  3EFF      0125 MAIN   LD    A,0FFH     ;INITIALIZE I/O PORT LINES
01C3  47        0126        LD    B,A        ;ALLOW TIME FOR LINES TO SETTLE
01C4  10FE      0127 WAKEUP DJNZ  WAKEUP
01C6  CDD503    0128        CALL  REMOVE     ;REMOVE PENDING BREAKPOINTS
01C9  218B01    0129        LD    HL,BANNER  ;OUTPUT BANNER MESSAGE
01CC  CD0A02    0130        CALL  OUTPUT
01CF  E3        0131 RESET  EX    (SP),HL    ;CLEAN UP BREAKPOINT POINTER
01D0  2B        0132        DEC   HL         ;
01D1  E3        0133        EX    (SP),HL    ;
01D2  FDE5      0134        PUSH  IY         ;PUSH MAIN REGISTER SET
01D4  DDE5      0135        PUSH  IX
01D6  E5        0136        PUSH  HL
```

```
ADDR  CODE      STMT SOURCE STATEMENT

01D7  D5        0137        PUSH  DE
01D8  C5        0138        PUSH  BC
01D9  F5        0139        PUSH  AF
01DA  08        0140        EX    AF,AF'      ;GET ALTERNATE REGISTER SET
01DB  D9        0141        EXX               ;  AND PUSH SAME
01DC  E5        0142        PUSH  HL
01DD  D5        0143        PUSH  DE
01DE  C5        0144        PUSH  BC
01DF  F5        0145        PUSH  AF
01E0  CDD503    0146        CALL  REMOVE      ;CLEAN UP BREAKPOINT
01E3  211A01    0147 OK     LD    HL,LFCRCU   ;OUTPUT CURSOR
01E6  CD0A02    0148        CALL  OUTPUT
01E9  CD2302    0149        CALL  RECEIV      ;GET COMMAND
01EC  217301    0150        LD    HL,CMDTBL-2   ;GET POINTER TO CMD TABLE-2
01EF  010800    0151        LD    BC,CMDLEN     ;GET LENGTH OF SAME
01F2  23        0152 NXTCMD INC   HL          ;POINT TO COMMAND NAME
01F3  23        0153        INC   HL
01F4  EDA1      0154        CPI               ;COMPARE ENTERED-TABLE COMMANDS
01F6  E20002    0155        JP    PO,LASTCD   ;LAST CMD? +=> CR & -> LASTCD
01F9  20F7      0156        JR    NZ,NXTCMD   ;MATCH? -=> NeXTCoMmanD
01FB  5E        0157        LD    E,(HL)      ;ELSE: POINT TO LOCATION OF GO
01FC  23        0158        INC   HL          ; AND START EXECUTION OF SAME
01FD  56        0159        LD    D,(HL)
01FE  EB        0160        EX    DE,HL
01FF  E9        0161        JP    (HL)
0200  28E1      0162 LASTCD JR    Z,OK        ;IF A CR => GO TO OK
                0163 ;#############################################################
                0164 ;      EXECUTION OF THE FOLLOWING MODULE WILL ONLY OCCUR
                0165 ;      UPON DETECTION OF AN INVALID COMMAND
                0166 ;#############################################################
0202  211201    0167 NOCMD  LD    HL,ERRMES   ;START OF ERROR MESSAGE
0205  CD0A02    0168        CALL  OUTPUT
0208  18D9      0169        JR    OK          ;RETURN TO INPUT MODE
                0170 ;
                0171 ;#############################################################
                0172 ;      OUTPUT WILL TAKE THE ASCII DATA STARTING AT THE
                0173 ; INPUTTED VALUE OF THE HL REGISTER AND OUTPUT THIS DATA
                0174 ; TO THE TERMINAL.  TERMINATION WILL OCCUR WHEN THE
                0175 ; INDEXED DATA CONTAINS THE MOST SIG BIT (D7) SET.
                0176 ;#############################################################
020A  7E        0177 OUTPUT LD    A,(HL)
020B  F5        0178        PUSH  AF
020C  CD1502    0179        CALL  XMITER
020F  F1        0180        POP   AF
0210  17        0181        RLA
0211  D8        0182        RET   C
0212  23        0183        INC   HL
0213  18F5      0184        JR    OUTPUT
                0185 ;
                0186 ;#############################################################
                0187 ;      XMITER WILL OUTPUT A SINGLE ASCII CHARACTER LOCATED
                0188 ; IN ACC-A UPON ENTRY.
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0189 ;#########################################################
0215  D9        0190 XMITER EXX
0216  47        0191        LD    B,A
0217  DB03      0192 HERE1  IN    A,(3)
0219  2F        0193        CPL
021A  E601      0194        AND   01
021C  20F9      0195        JR    NZ,HERE1
021E  78        0196        LD    A,B
021F  D302      0197        OUT   (2),A
0221  D9        0198        EXX
0222  C9        0199        RET
                0200 ;#########################################################
                0201 ;      RECEIV INPUTS ONE ASCII CHARACTER INTO ACC-A AND
                0202 ; ECHOS BACK THE DATA TO THE ADM-3A
                0203 ;#########################################################
0223  DB03      0204 RECEIV IN    A,(3)
0225  2F        0205        CPL
0226  E602      0206        AND   02
0228  20F9      0207        JR    NZ,RECEIV
022A  DB02      0208        IN    A,(2)
022C  E67F      0209        AND   7FH
022E  CD1502    0210        CALL  XMITER
0231  C9        0211        RET
                0212 ;#########################################################
                0213 ;      THE FOLLOWING ROUTINE WILL INPUT A FOUR NIBBLE
                0214 ; ADDRESS AND START PROGRAM EXECUTION AT SAID ADDRESS
                0215 ; UPON DETECTION OF A CR.
                0216 ;#########################################################
0232  CD5E03    0217 GOPROG CALL  HILOW4      ;GET ADDRESS
0235  DA0202    0218        JP    C,NOCMD     ;ERROR => NOCMD
0238  3EC3      0219        LD    A,0C3H      ;Z80 JUMP COMMAND
023A  32F707    0220        LD    (FIX-1),A   ;SET UP JUMP COMMAND TO SPECIFIED
                0221 ;                          ADDRESS
023D  F1        0222        POP   AF          ;POP ALT REG SET
023E  C1        0223        POP   BC
023F  D1        0224        POP   DE
0240  E1        0225        POP   HL
0241  08        0226        EX    AF,AF'      ;GET MAIN REG SET
0242  D9        0227        EXX
0243  F1        0228        POP   AF          ;POP SAME
0244  C1        0229        POP   BC
0245  D1        0230        POP   DE
0246  E1        0231        POP   HL
0247  DDE1      0232        POP   IX
0249  FDE1      0233        POP   IY
024B  C3F707    0234        JP    FIX-1       ;AFFECTS A JUMP TO THE LOCATION
                0235 ;                          SPECIFIED BY FIX AND FIX+1.
                0236 ;
                0237 ;#########################################################
                0238 ;      REGLOK WILL DISPLAY THE PUSHED REGISTER SET
                0239 ;#########################################################
                0240 ;---------------------------------------------------------
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0241 ;                          DATA FOR THE ROUTINE
                0242 ;-----------------------------------------------------------
024E  0A0D      0243 REGHED DEFB  LF,CR         ;LF,CR
0250  53502020  0244        DEFM  'SP   A''F'' B''C'' D''E'' H''L'' '
      20412746
      27204227
      43272044
      27452720
      48274C27
      20
0269  41204620  0245        DEFM  'A F  B C  D E  H L'
      20422043
      20204420
      45202048
      204C
027B  20204958  0246        DEFM  '  IX   IY   PC'
      20202049
      59202020
      5043
0289  0A8D      0247        DEFB  LF,CREND      ;LF, CR, DELIMITER
                0248 ;//////////////////////////////////////////////////////
028B  ED73F807  0249 REGLOK LD    (FIX),SP      ;SP TO TEMP STORAGE
028F  214E02    0250        LD    HL,REGHED     ;START OF REGISTER HEADER
0292  CD0A02    0251        CALL  OUTPUT        ;OUTPUT HEADER
0295  2AF807    0252        LD    HL,(FIX)      ;INITIALIZE POINTER TO PUSHED REGs
0298  060C      0253        LD    B,0CH         ;NUMBER OF REGISTERS
029A  C5        0254 REGAGN PUSH  BC            ;SAVE COUNTER
029B  CD4403    0255        CALL  OUT4A         ;OUTPUT FOUR NIBBLES
029E  3E20      0256        LD    A,SPACE       ;SPACE
02A0  CD1502    0257        CALL  XMITER
02A3  7E        0258        LD    A,(HL)        ;MOVE DOUBLE WORD TO FIX
02A4  32F807    0259        LD    (FIX),A
02A7  23        0260        INC   HL
02A8  7E        0261        LD    A,(HL)
02A9  32F907    0262        LD    (FIX+1),A
02AC  23        0263        INC   HL            ;POINT TO NEXT  REG PAIR
02AD  C1        0264        POP   BC            ;RECOVER COUNTER
02AE  10EA      0265        DJNZ  REGAGN        ;MORE => REGAGN
02B0  C3E301    0266        JP    OK            ;ELSE RETURN
                0267 ;
                0268 ;#########################################################
                0269 ;       BLOCK COPY SUBPROGRAM COPIES A BLOCK OF CODE
                0270 ; FROM THE 'FROM' ADDRESS TO THE 'TO' ADDRESS FOR A
                0271 ; TOTAL OF 'LENGTH' (HEX) BYTES OF DATA.  NO VERIFICATION
                0272 ; IS DONE.
                0273 ;#########################################################
02B3  216501    0274 COPY   LD       HL,FROM
02B6  CD0A02    0275        CALL     OUTPUT     ;OUTPUT ROUTINE 'FROM'
02B9  CD5E03    0276        CALL     HILOW4     ;GET STARTING ADDRESS
02BC  382B      0277        JR       C,NOCMD0   ;ERROR => NOCMD0
02BE  2AF807    0278        LD       HL,(FIX)   ;GET AND SAVE DATA ON STACK
02C1  E5        0279        PUSH     HL
```

```
ADDR  CODE      STMT SOURCE STATEMENT

02C2  211601    0280        LD      HL,TO
02C5  CD0A02    0281        CALL    OUTPUT    ;OUTPUT ROUTINE 'TO'
02C8  CD5E03    0282        CALL    HILOW4    ;GET DESTINATION ADDRESS .
02CB  381B      0283        JR      C,NOCMD1  ;ERROR => NOCMD1
02CD  2AF807    0284        LD      HL,(FIX)  ;GET AND SAVE DEST. ON STACK
02D0  E5        0285        PUSH    HL
02D1  216D01    0286        LD      HL,LENGTH
02D4  CD0A02    0287        CALL    OUTPUT    ;OUTPUT ROUTINE 'LENGTH'
02D7  CD5E03    0288        CALL    HILOW4    ;GET LENGTH
02DA  380B      0289        JR      C,NOCMD2  ;ERROR => NOCMD2
02DC  ED4BF807  0290        LD      BC,(FIX)  ;GET LENGTH
02E0  D1        0291        POP     DE        ;DESTINATION
02E1  E1        0292        POP     HL        ;START
02E2  EDB0      0293        LDIR              ;MOVE
02E4  C3E301    0294        JP      OK        ;RETURN TO MAIN ROUTINE
                0295 ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                0296 ;      ERROR HANDLING ROUTINE TO AVOID CREAPING STACK
                0297 ;
02E7  E1        0298 NOCMD2 POP    HL         ;POP STACK TO ADJUST STACK PTR
02E8  E1        0299 NOCMD1 POP    HL         ; AND JUMP TO NOC(O)M(MAN)D
02E9  C30202    0300 NOCMD0 JP     NOCMD
                0301 ;
                0302 ;##########################################################
                0303 ;        SOFTWARE INTERRUPT RESTART ROUTINE PULLS ALL     #
                0304 ;# REGISTERS FROM STACK (INCLUDING PROGRAM COUNTER).      #
                0305 ;##########################################################
02EC  F1        0306 RESTAR POP  AF
02ED  C1        0307        POP  BC
02EE  D1        0308        POP  DE
02EF  E1        0309        POP  HL
02F0  08        0310        EX   AF,AF'
02F1  D9        0311        EXX
02F2  F1        0312        POP  AF
02F3  C1        0313        POP  BC
02F4  D1        0314        POP  DE
02F5  E1        0315        POP  HL
02F6  DDE1      0316        POP  IX
02F8  FDE1      0317        POP  IY
02FA  C9        0318        RET
                0319 ;
                0320 ;##########################################################
                0321 ;       MEMVUE WILL CAUSE A STARTING ADDRESS TO BE INPUT,
                0322 ; FOLLOWED BY A COMMA (','), AND CONCLUDED BY AN ENDING
                0323 ; ADDRESS.  THE ROUTINE WILL OUTPUT THE CONTENTS OF MEMORY
                0324 ; IN BYTE FORM FROM THE BEGINNING OF THE BLOCK TO THE END
                0325 ; OF THE BLOCK.
                0326 ;##########################################################
02FB  210000    0327 MEMVUE LD    HL,0000H   ;INITIALIZE DEFAULT STARTING ADDRS
02FE  22F807    0328        LD    (FIX),HL   ;
0301  CD5E03    0329        CALL  HILOW4     ;GET STARTING ADDRESS
0304  FE2C      0330        CP    ','        ;HAS A COMMA BEEN SENT
0306  C20202    0331        JP    NZ,NOCMD   ; - => ERRCR
```

```
ADDR  CODE      STMT SOURCE STATEMENT

0309  2AF807    0332        LD    HL,(FIX)    ;SAVE STARTING ADDRESS
030C  CD5E03    0333        CALL  HILOW4      ;GET ENDING ADDRESS
030F  DA0202    0334        JP    C,NOCMD     ;ERROR? + => NOCMD
0312  ED5BF807  0335        LD    DE,(FIX)    ;GET ENDING ADDRESS
0316  13        0336        INC   DE          ;POINT TO END+1
0317  E5        0337 I0     PUSH  HL          ;SAVE CURRENT ADDRESS
0318  21BF01    0338        LD    HL,NEWLIN   ;OUTPUT A NEW LINE
031B  CD0A02    0339        CALL  OUTPUT
031E  E1        0340        POP   HL          ;RECOVER CURRENT LINE
031F  CD4103    0341        CALL  OUT4        ;OUTPUT ADDRESS
0322  3E20      0342 I1     LD    A,' '       ;OUTPUT A SPACE
0324  CD1502    0343        CALL  XMITER
0327  7E        0344        LD    A,(HL)      ;GET DATA
0328  32F907    0345        LD    (FIX+1),A   ;STORE IN BUFFER
032B  0602      0346        LD    B,02H       ;INITIALIZE COUNTER
032D  CD4603    0347        CALL  OUT2        ;OUTPUT BYTE
0330  23        0348        INC   HL          ;POINT TO NEXT CHAR
0331  7D        0349        LD    A,L         ;LOW BYTE TO ACC
0332  BB        0350        CP    E           ;16 BIT COMPARE
0333  2005      0351        JR    NZ,NEXTDT   ;NOT SAME => CONTINUE
0335  7C        0352        LD    A,H         ;
0336  BA        0353        CP    D           ;
0337  CAE301    0354        JP    Z,OK        ;SAME => RETURN TO MAIN PROG.
033A  7D        0355 NEXTDT LD    A,L         ;LAST INPUT FOR LINE?
033B  E60F      0356        AND   0FH         ;LAST PER LINE? + => Z SET
033D  28D8      0357        JR    Z,I0        ; Z => I0
033F  18E1      0358        JR    I1          ;ELSE NEXT
                0359 ;
                0360 ;#########################################################
                0361 ;      THIS ROUTINE, IF ENTERED FROM THE FIRST STATEMENT
                0362 ; WILL CAUSE THE FOUR NIBBLES LOCATED IN THE HL REGISTER
                0363 ; TO BE ENCODED TO ASCII AND DISPLAYED AT THE TERMINAL.
                0364 ;      IF THE ROUTINE IS ENTERED AT THE SECOND+ STATEMENT,
                0365 ; THEN THE VALUE OF ACC-B (B=<4) IS USED TO DETERMINE
                0366 ; THE NUMBER OF OUTPUTTED NIBBLES AND THE DATA IS TAKEN
                0367 ; DIRECTLY FROM LOCATION FIX+1.
                0368 ;      ENTERING AT OUT4A IMPLIES THAT DATA IS IN (FIX) AS
                0369 ; OPPOSED TO HL UPON ENTRY.
                0370 ;#########################################################
0341  22F807    0371 OUT4   LD    (FIX),HL    ;DATA TO FIX
0344  0604      0372 OUT4A  LD    B,04H       ;NUMBER OF NIBBLES
0346  E5        0373 OUT2   PUSH  HL          ;SAVE MEMORY POINTER
0347  21F807    0374 OUTCHR LD    HL,FIX      ;LOW DATA BYTE TO HL
034A  3E30      0375        LD    A,30H       ;ASCII ZERO
034C  ED6F      0376        RLD               ;NIBBLE ROTATE LOW BYTE
034E  23        0377        INC   HL          ;POINT TO HIGH BYTE
034F  ED6F      0378        RLD               ;NIBBLE ROTATE HIGH BYTE
0351  FE3A      0379        CP    3AH         ;IS NUMBER GREATER THAN "9?"
0353  3802      0380        JR    C,OUTDTA    ;NO => ADD NO CORRECTION
0355  C607      0381        ADD   A,07H       ;ELSE ADD A CORRECTION TERM
0357  CD1502    0382 OUTDTA CALL  XMITER      ;OUT DATA
035A  10EB      0383        DJNZ  OUTCHR      ;MORE? + => OUTCHR
```

```
ADDR  CODE      STMT SOURCE STATEMENT

035C  E1        0384        POP   HL          ;RESTORE MEMORY POINTER
035D  C9        0385        RET
                0386 ;
                0387 ;#########################################################
                0388 ;      THE FOLLOWING ROUTINE WILL OBTAIN A FOUR NIBBLE
                0389 ; ADDRESS FROM DATA INPUT FROM THE TERMINAL (IF ENTERED
                0390 ; FROM THE FIRST STATEMENT).  IF THE ROUTINE IS ENTERED
                0391 ; FROM THE SECOND STATEMENT WITH AN 'E' VALUE OF 02H,
                0392 ; THEN THE ROUTINE WILL COMPUTE ONLY A TWO NIBBLE HEX
                0393 ; VALUE.
                0394 ;#########################################################
035E  1E00      0395 HILOW4 LD    E,00H       ;INITIALIZE FOR 4 NIBBLES
0360  E5        0396 HILOW2 PUSH  HL          ;SAVE MEMORY POINTER
0361  CD2302    0397 HILORP CALL  RECEIV      ;GET NIBBLE
0364  FE0D      0398        CP    CR          ;END?
0366  281C      0399        JR    Z,ENDHGH    ; + => ENDHGH
0368  214001    0400        LD    HL,ASCII    ;START OF ASCII LOOKUP
036B  011000    0401        LD    BC,10H      ;LENGTH OF SAME
036E  EDA1      0402 NXTHGH CPI               ;LAST VALUE THAT OF TABLE?
0370  E27503    0403        JP    PO,OUTAHX   ;LAST ENTRY => OUTAHX
0373  20F9      0404        JR    NZ,NXTHGH   ;VALUE? - => NXTHGH
                0405 ;EXECUTION OF THE FOLLOWING STATEMENT OCCURS IF EITHER
                0406 ; A MATCH IS FOUND (IN WHICH CASE THE JUMP IS REDUNDANTLY
                0407 ; FALSE), OR IF END OF THE TABLE IS FOUND (IN WHICH CASE
                0408 ; THE JUMP TESTS TO SEE IF THE INPUTTED CHARACTER IS A "F"
                0409 ; OR IF IT IS NOT IN THE TABLE--OUTHER).
0375  2019      0410 OUTAHX JR    NZ,OUTHER   ;IS THIS THE VALUE? - => OUTHER
0377  2D        0411        DEC   L           ;ADJUST POINTER
0378  7D        0412        LD    A,L         ;COMPUTE LOOKUP ADDRESS
0379  21F807    0413        LD    HL,FIX      ;POINTER TO LOW NIBBLE
037C  ED6F      0414        RLD               ;ROTATE IN NIBBLE
037E  23        0415        INC   HL          ;POINT TO HIGH BYTE
037F  ED6F      0416        RLD               ;ROTATE IN NIBBLE
0381  1C        0417        INC   E           ;ADJUST COUNTER
0382  18DD      0418        JR    HILORP      ;CONTINUE INPUT
0384  21BF01    0419 ENDHGH LD    HL,NEWLIN   ;OUTPUT LF,CR
0387  CD0A02    0420        CALL  OUTPUT
038A  3EFC      0421        LD    A,0FCH      ;2'S COMP OF 0004H
038C  83        0422        ADD   A,E         ;SEE IF ENOUGH NIBBLES INPUT
038D  3F        0423        CCF               ;ADJUST CARRY TO ERROR STATE
038E  E1        0424        POP   HL          ;RECOVER MEMORY POINTER
038F  C9        0425        RET
0390  37        0426 OUTHER SCF               ;SET CARRY FOR ERROR
0391  E1        0427        POP   HL          ;RECOVER MEMORY POINTER
0392  C9        0428        RET
                0429 ;
                0430 ;#########################################################
                0431 ;      MEMCHG ALLOWS THE USER TO VIEW A BLOCK OF MEMORY
                0432 ; AND ALTER ITS CONTENTS.  DETECTION OF A "S" TERMINATES
                0433 ; THE ROUTINE.  DETECTION OF A LF CAUSES THE POINTER TO
                0434 ; ADVANCE TO THE NEXT MEMORY LOCATION WITHOUT DISTURBING
                0435 ; THE CONTENTS OF THE PRESENTLY INDEXED LOCATION.
```

```
ADDR  CODE      STMT SOURCE STATEMENT

                0436 ;#########################################################
0393  CD5E03    0437 MEMCHG CALL  HILOW4     ;GET STARTING ADDRESS
0396  DA0202    0438        JP    C,NOCMD    ;ERROR => NOCMD
0399  2AF807    0439        LD    HL,(FIX)   ;GET STARTING ADDRESS
039C  CD4103    0440 NXTMEM CALL  OUT4       ;DISPLAY ADDRESS
039F  3E3E      0441        LD    A,CURSOR   ;OUT CURSOR
03A1  CD1502    0442        CALL  XMITER
03A4  7E        0443        LD    A,(HL)     ;GET MEMORY CONTENTS
03A5  32F907    0444        LD    (FIX+1),A  ;PLACE IN STORAGE
03A8  0602      0445        LD    B,02H      ;TWO NIBBLES OUTPUTTED
03AA  CD4603    0446        CALL  OUT2
03AD  3E20      0447        LD    A,SPACE    ;SPACE BETWEEN OLD AND NEW VALUES
03AF  CD1502    0448        CALL  XMITER
03B2  1E02      0449        LD    E,02H      ;TWO NIBBLES TO BE INPUT
03B4  CD6003    0450        CALL  HILOW2     ;GET TWO NIBBLES
03B7  300C      0451        JR    NC,PNOCMD  ;NO ERROR => PNOCMD
03B9  FE53      0452        CP    53H        ;ON DETECTION OF AN "S" => STOP
03BB  CAE301    0453        JP    Z,OK
03BE  FE0A      0454        CP    LF         ;ON DETECTION OF A LF => CONT.
03C0  280B      0455        JR    Z,LFINPT
03C2  C30202    0456        JP    NOCMD      ;  ELSE ERROR
03C5  3AF807    0457 PNOCMD LD    A,(FIX)    ;GET DATA
03C8  77        0458        LD    (HL),A     ;STORE
03C9  BE        0459        CP    (HL)       ;CHECK
03CA  C20202    0460        JP    NZ,NOCMD   ;NOCHECK => ERROR
03CD  23        0461 LFINPT INC   HL         ;POINT TO NEXT
03CE  3E0D      0462        LD    A,CR       ;OUTPUT CR
03D0  CD1502    0463        CALL  XMITER
03D3  18C7      0464        JR    NXTMEM     ;CONTINUE
                0465 ;#########################################################
                0466 ;       REMOVE A BREAKPOINT
                0467 ;#########################################################
03D5  2AFB07    0468 REMOVE LD    HL,(BREAKA) ;CLEAR ANY PENDING BREAKPOINT
03D8  3AFA07    0469        LD    A,(CODE)    ;
03DB  77        0470        LD    (HL),A      ;
03DC  3EFF      0471        LD    A,0FFH      ;POINT BREAKPOINT TO NOTHING
03DE  32FC07    0472        LD    (BREAKA+1),A ;
03E1  32FB07    0473        LD    (BREAKA),A   ;
03E4  C9        0474        RET               ;
                0475 ;#########################################################
                0476 ;       SET A BREAKPOINT
                0477 ;#########################################################
03E5  CDD503    0478 BRKPT  CALL  REMOVE      ;REMOVE PENDING BREAKPOINT
03E8  CD5E03    0479        CALL  HILOW4      ;GET AN ADDRESS
03EB  DA0202    0480        JP    C,NOCMD     ;ERROR => NOCMD
03EE  2AF807    0481        LD    HL,(FIX)    ;GET ADDRESS
03F1  7E        0482        LD    A,(HL)      ;GET CODE
03F2  22FB07    0483        LD    (BREAKA),HL ;STORE ADDRESS
03F5  32FA07    0484        LD    (CODE),A    ; AND CODE
03F8  3EFF      0485        LD    A,0FFH      ;BREAKPOINT COMMAND
03FA  77        0486        LD    (HL),A      ;SUBSTITUTE
03FB  C3E301    0487        JP    OK          ;
```

```
$NSMTR 05004F
:1D01000031F707FB3EC332380021CF01223900C3C1013F3F3F87544F3FA00A0DBEE1
:200140003031323334353637383941424344454631302D53544F50204249542045585045D8
:20016000435445448704A0D46524F4D3FA04C454E4754483FA04D930347320256FB02528B7F
:200180000243B3024EEC0242E5030D4E5343383030204D4F4E49544F522056455253494F36
:2001A0004E20312E39310A0D4B414E534153205354415445205554E4956455253495459 0A43
:2001C0008D3EFF4710FECDD503218B01CD0A02E32BE3FDE5DDE5E5D5C5F508D9E5D5C5F577
:2001E000CDD503211A01CD0A02CD2302217301010800232 3EDA1E2000220F75E2356EBE93B
:2002000028E1211201CD0A0218D97EF5CD1502F117D82318F5D947DB032FE60120F978D3F8
:2002200002D9C9DB032FE60220F9DB02E67FCD1502C9CD5E03DA02023EC332F707F1C1D15D
:20024000E108D9F1C1D1E1DDE1FDE1C3F7070A0D535020202041274627204227432720 44D5
:2002600027452720482 74C27204120462020422043202044204520204820 4C20204958205F
:2002800020204959202020504 30A8DED73F807214E02CD0A022AF807060CC5CD44033E20D7
:2002A000CD15027E32F807237E32F90723C110EAC3E301216501CD0A02CD5E03382B2AF840
:2002C00007E5211601CD0A02CD5E03381B2AF807E5216D01CD0A02CD5E03380BED4BF80782
:2002E000D1E1EDB0C3E301E1E1C30202F1C1D1E108D9F1C1D1E1DDE1FDE1C921000022F836
:2003000007CD5E03FE2CC202022AF807CD5E03DA0202ED5BF80713E521BF01CD0A02E1CDDC
:2003200041033E20CD15027E32F9070602CD4603237DBB20057CBACAE3017DE60F28D8187B
:20034000E122F8070604E521F8073E30ED6F23ED6FFE3A3802C607CD150210EBE1C91E005D
:20036000E5CD2302FE0D281C214001011000EDA1E2750320F920192D7D21F807ED6F23ED74
:200380006F1C18DD21BF01CD0A023EFC833FE1C937E1C9CD5E03DA02022AF807CD41033E18
:2003A0003ECD15027E32F9070602CD46033E20CD15021E02CD6003300CFE53CAE301FE0A78
:2003C000280BC302023AF80777BEC20202233E0DCD150218C72AFB073AFA07773EFF32FC75
:1E03E0000732FB07C9CDD503CD5E03DA02022AF8077E22FB0732FA073EFF77C3E301F1
:00010001FE
```

NSC800 Board Fast 8x8 Multiplies

The following appendix documents eight by eight multiplies for the Cody NSC800 board with 1855 MDUs on board. The first section of the listing documents an unsigned eight by eight multiply. The second contains the signed eight by eight multiply. In the first a five to seven microsecond delay routine must be inserted at the indicated location in order to allow the multiply to complete. {This routine is a function of the system clock and is calculated according to this frequency.} The seven microsecond delay was determined and utilized by Cody[1] when he coded the original algorithm. The algorithms assume that the NSC800 board is the host system.

If a system with only one 1855 is utilized the routine can be simplified by loading the second operand into ACC-L (as opposed to the zeroed ACC-A) and deleting all code prior to this step. Making these modifications permits the multiply to be performed in 73T (18.25 uS at 4.0 MHz.) for the unsigned and 123T (30.75 uS at 4.0 MHz.) for the signed multiply. {Author's note: The signed multiply must not delete the XOR A because this is needed for the "Fudge" manipulation.}

```
ADDR  CODE      STMT SOURCE STATEMENT

                0001        NAME    QMULT
                0003        PSECT   REL
                0004        ;
                0005        ;       DWIGHT W. GORDON
                0006        ;       DEPARTMENT OF ELECTRICAL ENGINEERING
                0007        ;       KANSAS STATE UNIVERSITY
                0008        ;       SEATON HALL
                0009        ;       MANHATTAN, KANSAS 66506
                0010        ;
                0011        ;       JULY 8, 1982
                0012        ;
                0013        ;       THIS PROGRAM CONTAINS FAST MULTIPLIES
                0014        ; FOR THE CODY NSC800 BOARD.
                0015        ;
>0069           0016 SCODE  EQU     0069H   ;1855 START CODE FOR NSC800
>1000           0017 MDU    EQU     1000H   ;LOCATION 1855'S
>0020           0018 ICODE  EQU     0020H   ;INITIALIZATION CODE FOR 1855'S
                0019        ;
                0020        ;PROGRAMMING ASSUMPTIONS:
                0021        ; 1. EIGHT BIT MULTIPLIES ON EXISTING NSC800 BOARD
                0022        ; 2. ALL REGISTERS ARE AVAILABLE
                0023        ; 3. IY HAS BEEN INITIALIZED TO POINT TO THE MDU'S
                0024        ; 4. SOURCE OPERANDS IN REGISTERS "D" AND "H"
                0025        ; 5. DESTINATION ANSWER TO REGISTER PAIR "HL"
                0026        ;
                0027        ;************************************************
                0028        ;**      VERSION 1:  UNSIGNED MULTIPLY          **
                0029        ;************************************************
>0000           0030        ORG     0000H
'0000 FD360320  0031        LD  (IY+3),ICODE  ;SETUP & RESET SEQUENCE COUNTER 19
'0004 7A        0032        LD  A,D           ;LOAD FIRST OPERAND              4
'0005 320010    0033        LD  (MDU),A       ;                               13
'0008 AF        0034        XOR A             ;CLEAR ACC-A                     4
'0009 6F        0035        LD  L,A           ;LOAD A ZERO IN LOW BYTE OF HL   4
'000A 220010    0036        LD  (MDU),HL      ;LOAD SECOND OPERAND            16
'000D FD360369  0037        LD  (IY+3),SCODE  ;START MULTIPLY                 19
                0038        ;
                0039        ;       INSERT A 5 TO 7 MICROSECOND DELAY HERE
                0040        ;
'0011 3A0210    0041        LD  A,(MDU+2)     ;OBTAIN RESULT OF MULTIPLY      13
'0014 67        0042        LD  H,A           ;  AND STORE IN REGISTER PAIR    4
'0015 3A0210    0043        LD  A,(MDU+2)     ;  HL                           13
'0018 6F        0044        LD  L,A           ;                                4
                0045 ;                                                       ----
                0046 ;                                                5(7)uS + 113T
                0047 ;       2.5 MHz. => 50.2(52.2) uS per multiply.
                0048 ;       4.0 MHz. => 33.25(35.25) uS per multiply.
                0049 ;
```

```
                    0050 ;
                    0051        ;#################################################
                    0052        ;##     VERSION 2:  SIGNED MULTIPLY             ##
                    0053        ;#################################################
>0000               0054        ORG 0000H
'0000 FD360320      0055        LD  (IY+3),ICODE  ;SETUP & RESET SEQUENCE COUNTER 19
'0004 7A            0056        LD  A,D           ;LOAD FIRST OPERAND              4
'0005 320010        0057        LD  (MDU),A       ;                               13
'0008 AF            0058        XOR A             ;CLEAR ACC-A                     4
'0009 6F            0059        LD  L,A           ;LOAD A ZERO IN LOW BYTE OF HL   4
'000A 220010        0060        LD  (MDU),HL      ;LOAD SECOND OPERAND            16
'000D FD360369      0061        LD  (IY+3),SCODE  ;START MULTIPLY                 19
'0011 CB7A          0062        BIT 7,D           ;TEST FOR NEGATIVE               8
'0013 2801          0063        JR  Z,NOADJ1      ;IF + LEAVE FUDGE ALONE       12/7
'0015 7C            0064        LD  A,H           ;IF NEGATIVE ALTER ONE FUDGE     4
'0016 CB7C          0065 NOADJ1 BIT 7,H           ;TEST FOR NEGATIVE               8
'0018 2801          0066        JR  Z,NOADJ2      ;IF + LEAVE FUDGE ALONE       12/7
'001A 82            0067        ADD A,D           ;IF NEGATIVE ALTER ONE FUDGE     4
'001B 4F            0068 NOADJ2 LD  C,A           ;PLACE FUDGE IN SECOND FUDGE     4
'001C 3A0210        0069        LD  A,(MDU+2)     ;OBTAIN RESULT OF MULTIPLY,     13
'001F 91            0070        SUB C             ;  CORRECT FOR SIGNED MULTIPLY,  4
'0020 67            0071        LD  H,A           ;  AND STORE IN REGISTER PAIR    4
'0021 3A0210        0072        LD  A,(MDU+2)     ;  HL                           13
'0024 6F            0073        LD  L,A           ;                                4
                    0074 ;                                                       ----
                    0075 ;                                                       159T
                    0076 ;       2.5 MHz. => 55.6 uS per multiply.
                    0077 ;       4.0 MHz. => 39.75 uS per multiply.
                    0078 ;
                    0079        END
```

System Clock Freq. (MHz.)	Vcc Volts	I_{Vcc} mA.	I_{+15} mA.	I_{-15} -mA.	Total Power mW
0.5	5.00	14.9	5.20	3.25	201.3
	6.00	24.0	5.02	3.25	268.2
	6.25	27.5	4.74	3.22	291.3
	7.00	45.0	4.10	3.25	425.3
1.0	5.00	23.5	5.20	3.25	244.3
	6.00	34.0	5.02	3.25	328.3
	6.25	38.5	4.76	3.23	360.5
	7.00	57.0	4.12	3.25	509.6
1.5	5.00	31.0	5.20	3.25	281.8
	6.00	43.8	5.03	3.25	387.0
	6.25	49.1	4.78	3.23	427.0
	7.00	69.0	4.16	3.25	594.2
2.0	5.00	38.0	5.20	3.25	316.8
	6.00	52.8	5.04	3.25	441.2
	6.25	60.0	4.80	3.23	495.5
	7.00	81.0	4.17	3.25	678.3
2.5	5.00	45.5	5.20	3.25	354.3
	6.00	62.3	5.04	3.25	498.2
	6.25	70.0	4.74	3.23	558.3
	7.00	92.6	4.18	3.26	759.8
3.0	6.00	71.9	5.05	3.25	555.9
	6.25	82.3	4.85	3.23	635.6
	7.00	103.6	4.19	3.26	837.0
3.5	6.00	81.5	5.08	3.25	614.0
	6.25	92.2	4.87	3.23	697.8
	7.00	115.9	4.20	3.26	923.2
3.9	6.25	98.0	4.89	3.23	734.3
4.0	7.00	126.2	4.22	3.26	995.6

Test Conditions:

1. The Cody NSC800 board was fully populated with the exception that the status LEDs (labeled "H", "R", "L", and "T") were disconnected.

2. This author's communication interface was disconnected at J8.

3. The program WIDROW16.DWG was executing in RAM at the time of the test.

4. The ADC input was a sine wave modulated between approximately 0.2 Hertz and 100. Hertz by a sawtooth wave with a one second (approximate) period.

5. The DAC output was connected to an oscilloscope. Correct operation of the cpu was determined by examining this output. When the output failed to change, or changed erratically over a period of time (order of magnitude: seconds) the cpu was diagnosed as inoperative.

6. The cpu was able to operate for at lease four hours at any of the above conditions. {Most were such that the board would function for more than seven hours without failing.}

Total Power Consumption of the NSC800 Board vs. Operating Frequency (MHz.) and Sampling Frequency of WIDROW16.DWG (sps.)

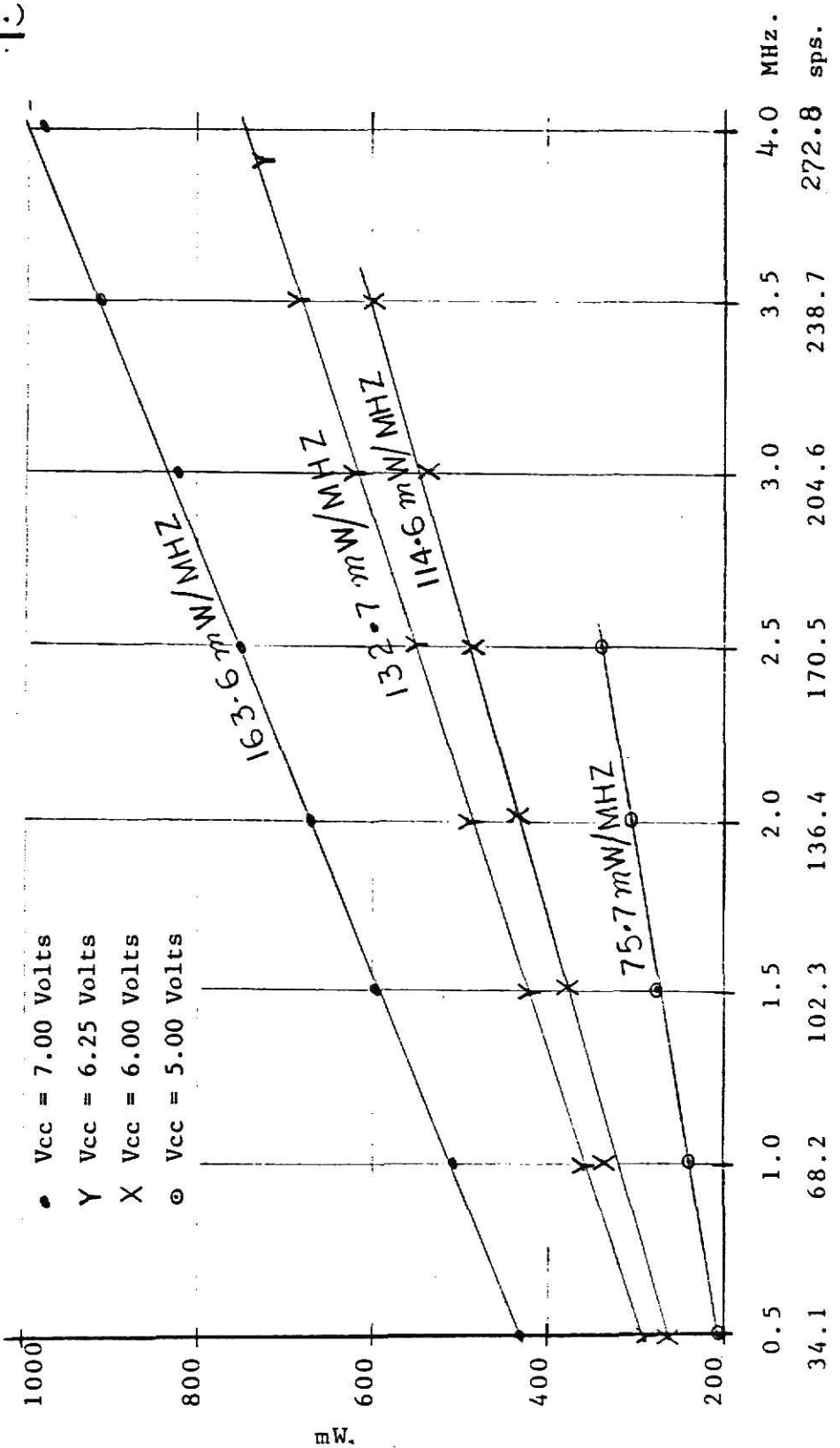


Figure G-1.

AN NSC800 DEVELOPMENT SYSTEM

by

DWIGHT WALLACE GORDON

B. A., Susquehanna University, 1981

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

ABSTRACT

This report provides complete documentation for a development system based on the NSC800 board designed by Mac Cody, Department of Electrical Engineering, Kansas State University, 1981. Included in this report is documentation for the hardware and software necessary to interface the NSC800 board to any asynchronous serial terminal with an RS232 interface and handshaking capabilities and the NorthStar Horizon. The latter interface has the capability to download object code to the NSC800. An emulation of the NSC800 monitor on the CP/M operating system of the NorthStar Horizon is provided for software development.

As an application example for the development system the author has programmed a version of the Widrow Adaptive Digital Processing algorithm to run on the NSC800 board. The original algorithm for a Z80 microprocessor was first presented by Donovan Nickel, Department of Electrical Engineering, Kansas State University, 1979, and later modified by Cody. Such an algorithm is often used for intruder detection. This use requires high speed and low power consumption. The development system was used to reduce the running time of the algorithm and determine power consumption specifications for the NSC800 board.