OAK
RIDGE
NATIONAL
LABORATORY

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

NUREG/CR-1548
ORNL/NUREG/TM-390

MASTER

The NDT-COMP9 Microcomputer

C. V. Dodd
R. F. Cowan

DO NOT MICROFILM
COVER

Prepared for the U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Under Interagency Agreements DOE 40-551-75

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161

Available from
GPO Sales Program
Division of Technical Information and Document Control
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DO NOT REPRODUCE
COVER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

NUREG/CR-1548
ORNL/NUREG/TM-390
Distribution
Category R5

Contract No. W-7405-eng-26

METALS AND CERAMICS DIVISION

NUREG/CR--1548

TI86 002701

THE NDT-COMP9 MICROCOMPUTER

C. V. Dodd and R. F. Cowan

Manuscript Completed - August 13, 1980

Date Published - September 1980

NOTICE: This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

Prepared for the
U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Washington, D.C. 20555
Under Interagency Agreement DOE 40-551-75,
NRC FIN No. B0417

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
DEPARTMENT OF ENERGY

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

jsu

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

CONTENTS

ABSTRACT	1
1. INTRODUCTION	1
2. MODULE HARDWARE DESCRIPTION	2
2.1 GENERAL	2
2.2 ARITHMETIC PROCESSING UNIT	4
2.3 MEMORY	5
2.4 OFF-BOARD HARDWARE	6
3. OPERATIONS	7
3.1 INPUT-OUTPUT	7
3.2 HARDWARE PROCESSORS	9
3.2.1 Central Processing Unit	9
3.2.2 Arithmetic Processing Unit	9
4. COMP9 SYSTEM SOFTWARE PROCESSORS	11
4.1 COMP9 MONITOR	11
4.2 COMP9 MATH COMMANDS	16
5. CONSTRUCTION DETAILS	18
6. PROM PROGRAMMING	18
7. POWER REQUIREMENTS	24
ACKNOWLEDGMENTS	24
REFERENCES	24
APPENDIX A — 8080/9080 INSTRUCTION SUMMARY	29
APPENDIX B — APU INSTRUCTION SUMMARY	33
APPENDIX C — MATH9 SUBROUTINE CALLS	37
APPENDIX D — COMP9 SYSTEM MONITOR ASSEMBLY LISTING	41
APPENDIX E — MATH9 MATHEMATICS PACKAGE ASSEMBLY LISTING	73
APPENDIX F — MATH CONVERSION PROGRAM	101
APPENDIX G — NDT-COMP9 PARTS LIST	105

THE NDT-COMP9 MICROCOMPUTER

C. V. Dodd and R. F. Cowan*

ABSTRACT

An 8080-based microcomputer system, the NDT-COMP9, has been designed for instrumentation control and data analysis in eddy-current tests. The NDT-COMP9 represents a significantly more powerful computer system than the NDT-COMP8 microcomputer from which it was developed. The NDT-COMP9 system is contained on a 240- by 120-mm (9.5- by 4.8-in.) circuit board and will fit in a four-wide Nuclear Instrumentation Module (NIM) BIN with 26-pin edge connectors. In addition to the 8080-compatible central processing unit (CPU), an arithmetic processing unit (APU) is available to provide up to 32-bit fixed- or floating-point, basic or transcendental math functions. The 16K of read only memory (ROM) and random access memory (RAM), one serial input-output (I/O) port (RS-232-C at a maximum speed of 9600 baud), and 72 parallel I/O ports are available. The baud rate is under software control. A system monitor and math package are available for use with the microcomputer.

1. INTRODUCTION

The NDT-COMP9 microcomputer is an extensively modified, updated version of the earlier NDT-COMP8¹ microcomputer. The NDT-COMP9 is a self-contained, 8-bit microcomputer system designed specifically to replace the NDT-COMP8 for experiment control and data analysis with modular eddy-current instrumentation. The unit is a high-performance, small-package design with 32-bit, floating-point hardware math operations and a sophisticated I/O structure for instrumentation control. Various compatible modules are in use or in the design stages, including tape drive interfaces, electrostatic plotter interfaces, and analog-digital (A/D) and digital-analog (D/A) modules. Specific applications for these modules are discussed in other reports.² The advanced processing capability, large memory, I/O structure, and small physical size make the unit ideal for a number of other diverse applications.

*Summer student, 1978, and summer research employee, 1979.

At the time of this report, no high-level language support* is available for the NDT-COMP9, so the programmer must have some familiarity with the Intel 8080 machine code.³ However, considerable software already exists for certain applications. A math package alleviates the need for the user to have a detailed knowledge of the arithmetic processor: simple subroutine calls can be made for all the math functions.

This report is intended for use as a reference manual for both NDT-COMP9 hardware and software and as a "user's guide." Sufficient detail is included to allow relatively easy and efficient use of the NDT-COMP9 by persons having limited backgrounds in microcomputer systems.

In addition to the computer board, only an RS-232-C terminal, which runs between 300 and 9600 baud, and a power supply are required.

2. MODULE HARDWARE DESCRIPTION

2.1 GENERAL

A block diagram of the NDT-COMP9 system is presented in Fig. 1. The system includes the CPU (8080 or 9080), the APU (9511), the system controller (8238), the crystal-controlled clock generator (8224), a real-time clock (8253), a serial I/O port (8251), three parallel I/O ports (8255), 16K of ROM and RAM memory, and three busses for addressing, data transfer, and control.

The 8224 clock uses the 18.432-MHz crystal signal as a time base. After an internal divide-by-9, the outputs at 2.0480 MHz provide the time base for the rest of the system. The 8253 real-time clock (RTC) contains three independent timer-counters. Counter 2 is used as a divide-by-N to derive the various baud rates for the 8251 serial interface.

*Some high-level languages are commercially available for 8080 systems, including FORTRAN and PASCAL, but none seems to be especially suitable for use with the NDT-COMP9 to date (August 1979). If a potential user is interested, one possible supplier of high-level microcomputer languages is the Boston Systems Office, Inc., 469 Moody Street, Waltham, MA 02154.

Fig. 1. Block Diagram of the NDT-COMP9.

The three busses (16-bit address bus, 8-bit data bus, and 6-bit control bus) provide communications between the various parts of the system. The CPU drives the address bus directly and selects memory locations or functions for the APU, RTC, and I/O ports. The control bus is driven by the system controller and determines read-write and memory-I/O functions. Data are transferred via the data bus, which can be written into and/or read by the system controller (a transparent buffer for the CPU), the memory, the APU, the RTC, and the I/O ports.

The memory consists of a maximum of eight 2K-by 8-bit ROM, programmable read only memory (PROM), ultraviolet erasable programmable read only memory (EPROM), and RAM chips, which are all interchangeable. Typically, Intel 2716 EPROMs (or the equivalent Texas Instrument TMS 2516) were used to provide the nonvolatile memory.

All I/O functions are performed by the three 8255 parallel ports or the 8251 serial port. Each parallel port consists of 24 bidirectional ports arranged in 3 groups of 8 bits each. Each group of eight bits can be programmed for input or output under software control. The 8251 serial interface and external line driver-receiver buffers provide an RS-232-C compatible interface with data transfer rates of 300 to 9600 baud under software control.

2.2 ARITHMETIC PROCESSING UNIT

The AM9511 is a high-performance arithmetic processor that provides the NDT-COMP9 with mathematical operations of up to 32-bit floating-point complexity in hardware. This alleviates the need for a software math package that normally would execute commands on an 8080 CPU. This APU yields much faster execution times for all operations, with increased accuracy and with more operations available, than the earlier NDT-COMP8 software package.

A software controlling package, MATH9, is included in the monitor chip. This routine allows relatively easy access to the APU math routines without requiring a detailed understanding of its instructions and operation. If the user desires, however, he may access the APU directly via his own software. A summary of APU instructions and operation codes is given in Appendix B. A summary of the MATH9 subroutine calls is in Appendix C. The APU is addressed as a conventional I/O port.

2.3 MEMORY

The memory map is presented in Fig. 2. The memory consists of eight plug-in sockets that contain 16K bytes of memory, 2K bytes in each socket. The memory is arranged so that the monitor and math library are in a PROM in the first 2K (7FF hex) bytes (on the right-hand side of the board). The last socket (on the left-hand side of the board) is reserved for RAM. The microcomputer stack starts at 3FFF hex and counts down into this section of RAM. The amount of RAM used depends on the amount of data stored on the stack. The memory between 800 hex and 37FF hex (A20 through A15) can contain any combination of RAM and PROM. The PROM used can be either Intel 2716 or Texas Instruments 2516, and the RAM is Mostek 4118. This is a 1K- by 8-bit RAM, and even though both Mostek and Texas Instruments have announced a 2K- by 8-bit static RAM, none is yet on the market. Pin 19 should be removed from the socket and tied to pin 24 with a 1-k Ω resistor. This causes the RAM to appear at two locations, exactly 1K apart. Thus, if the stack writes from 3FFF down to 3FD0, the same data will appear from 3BFF to 3BD0. Attempts by the user to overwrite this region will destroy the stack.

Y-172049

Fig. 2. Memory Map of the NDT-COMP9.

2.4 OFF-BOARD HARDWARE

An Electronic Industries Association (EIA) standard RS-232-C compatible terminal should be used as the serial I/O device connected via J4. This port is designed for 300- to 9600-baud devices. The following terminals and baud rates have been successfully used with the NDT-COMP9: LSI ADM-3A (300 to 9600), DEC DECWRITER-II (300), and Tekronix 4014 (9600). The serial interface is also designed to be compatible with the TI SILENT 700 at 300 baud.

A number of system timing signals are available at J4, including the $\phi 2$ transistor-transistor logic (TTL) 2.0480-MHz clock and the software-controlled baud rate clock. Other RTC signals are present at J4 also. Table 1 lists J4 pin connections.

Peripheral equipment can be driven via J1 through J3, which communicate with the three parallel I/O ports. Table 2 shows J1, J2, and J3 pin connections.

Table 1. J4 Pin Assignments

J4 Pin	Devices	Function
1, 3	Power supply	+12.0 V dc
2, 6, 8, 10	Power supply	GND
4	TERMINAL (RS-232-C)	SERIAL DATA GND
5	Power supply	-12.0 V dc
7	Power supply	-5.0 V dc
9	TERMINAL (RS-232-C)	SERIAL DATA OUT
11	TERMINAL (RS-232-C)	SERIAL DATA IN
12, 14, 16, 18	Power supply	+5.0 V
13	TI SILENT 700 interface	+12.0 V dc (limited)
15	8224	RESET IN
17	8253 (RTC)	CLK 2
19	8253 (RTC)	OUT 2
20	8253 (RTC)	GATE $\emptyset$
21	74LS00	INT REQ
22	8253 (RTC)	OUT $\emptyset$
23	8253 (RTC)	CLK 1
24	8253 (RTC)	CLK $\emptyset$
25	8253 (RTC)	GATE 1
26	8253 (RTC)	OUT 1

Table 2. J1, J2, and J3 Pin Assignments

Jx Pin	Function	Function	Jx Pin
1	PA3	PA0	7
2	PA4	PA1	5
3	PA2	PA2	3
4	PA5	PA3	1
5	PA1	PA4	2
6	PA6	PA5	4
7	PA0	PA6	6
8	PA7	PA7	8
9	PC7	PB0	23
10	PC0	PB1	21
11	PC6	PB2	19
12	PC1	PB3	17
13	PC5	PB4	18
14	PC2	PB5	20
15	PC4	PB6	22
16	PC3	PB7	24
17	PB3	PC0	10
18	PB4	PC1	12
19	PB2	PC2	14
20	PB5	PC3	16
21	PB1	PC4	15
22	PB6	PC5	13
23	PB0	PC6	11
24	PB7	PC7	9
25	NOT CONNECTED	GND	26
26	GND		

The system reset line is available on J4. A system reset is initiated by a short to ground, usually through a mechanical switch. A TTL interrupt signal may be input at J4. A logic low state will generate an interrupt.

3. OPERATIONS

3.1 INPUT-OUTPUT

Six chips on the COMP9 are accessed via the 8080 I/O instructions, IN and OUT (DBH and D3H, respectively). Figure 3 shows the address format. Data and control words are transferred from the CPU to the

Fig. 3. Input-Output Address Decoding.

desired chip by loading the value to be output into register A of the CPU and then issuing an instruction of the form "OUT addr," where addr is the two-digit hexadecimal address of the desired chip and function. Similarly, data can be input to register A simply by issuing an "IN addr" instruction. Hexadecimal addresses used in I/O instructions are listed in Table 3.

The I/O chips are selected by the presence of an I/OR or I/OW signal, as appropriate, and the presence of the proper address on the address bus lines AB0 through AB7. Note that the address is repeated on lines AB8 through AB15.

The serial interface is programmed by the monitor as the first task whenever a reset is issued. The RTC is programmed to generate the desired baud rate (nominally 9600 Hz) by using counter 2, which prevents the use of this counter for any other purpose. However, counters 0 and 1 are still available.

Detailed information on programming the 8255s is available in Intel Corporation publications.⁴

Table 3. Input-Output Addresses

Hex Address	Device and Function
7E	9511 (APU) data word
7F	9511 (APU) control word
BC	8253 (RTC) select counter 0
BD	8253 (RTC) select counter 1
BE	8253 (RTC) select counter 2
BF	8253 (RTC) mode word
DC	8255 (port 3) port 3A select
DD	8255 (port 3) port 3B select
DE	8255 (port 3) port 3C select
DF	8255 (port 3) control word
EC	8255 (port 2) port 2A select
ED	8255 (port 2) port 2B select
EE	8255 (port 2) port 2C select
EF	8255 (port 2) control word
F4	8255 (port 1) port 1A select
F5	8255 (port 1) port 1B select
F6	8255 (port 1) port 1C select
F7	8255 (port 1) control word
FA	8251 (USART) data word
FB	8251 (USART) control word

3.2 HARDWARE PROCESSORS

3.2.1 Central Processing Unit

The actual chip used on the COMP9 board is the Advanced Micro Devices AM9080ADC CPU. This chip executes the standard 8080 instruction set. Specific programming information is provided in publications by the Intel Corporation⁵ and Advanced Micro Devices, Inc.⁶ A summary of the 8080A instruction set is provided in Appendix A.

3.2.2 Arithmetic Processing Unit

The Advanced Micro Devices AM9511DC APU allows fixed- and floating-point operations. Figure 4 shows the various data formats. Fixed-point formats are in two's complement representation, with a range of -32,768 to

ORNL-DWG 80-13321

Fig. 4. Data Format for the 9511 APU.

+32,767 for 16-bit operands and a range of -2,147,483,648 to +2,147,483,647 for 32-bit operands. The leftmost bit is the sign bit. For floating operands, only a 32-bit format is allowed. The leftmost, most significant bit of the expression is the sign of the mantissa (0 = positive, 1 = negative). The next seven bits are the exponent in two's complement notation, with a range of -64 to +63. The remaining 24 bits represent the mantissa, with the binary point to the left of the most significant mantissa bit (bit 23). The mantissa is normalized so that bit 23 is always a one unless the value being represented is zero. This format can represent values in the range $\pm 2.7 \times 10^{-20}$ to $\pm 9.2 \times 10^{18}$ and zero.

The APU operates by using an internal stack that can hold up to eight 16-bit values or four 32-bit values. Whenever a data value is output to the APU, it is pushed onto this internal stack. The last byte output is the most significant byte (MSB) of the top of stack (TOS). Similarly, whenever data are requested from the APU, the operand at the TOS position

is input (one byte per instruction) from the APU. The last byte input will be the least significant byte (LSB) of the operand, and the next value on the stack (NOS) becomes the new TOS.

A summary of APU instructions is presented in Appendix B. Users may avoid direct programming of the APU via the software math package provided, MATH9, which programs the APU itself (see Sect. 4.2). For more detailed information on APU programming, consult the appropriate publication of Advanced Micro Devices.⁷

4. COMP9 SYSTEM SOFTWARE PROCESSORS

The COMP9 software processors consist of the monitor and the math software, which are stored in the first PROM. The monitor allows the user to perform a number of editing and test functions, and the math package allows the user easy access to the hardware APU.

4.1 COMP9 MONITOR

When the COMP9 is first turned on (or the reset button is pushed) the computer begins execution at 0000 memory, which is the first PROM address in the monitor. The monitor first initializes the RTC for the proper baud rate, then sets up the universal synchronous-asynchronous receiver-transmitter (USART), prints "NDT-COMP9" and then a period, ".", as a prompt for the next command. When it completes a command, another period is printed as a prompt. A summary of the commands is listed in Table 4. The brackets contain hex representations of the memory addresses, and leading zeros may be omitted. Spaces or commas may be used as delimiters between the addresses.

A summary of the COMP9 monitor routines is presented in Table 5. These routines can be called from any user program. A complete listing of the monitor written in Intel 8080 assembly language is given in Appendix D. Many of the routines are listed as public in this program and may be called by using an external statement in the user program. The link loader will link these programs together when the monitor and the user program are link edited and will determine the addresses for any subroutine calls.

Table 4. Summary of COMP9 Monitor Commands

Command ^a	Function
C <LO> <HI> <DEST> CR	Compares a block of data from LO through HI to the block starting at DEST
D <LO> <HI> CR	Displays the data from LO through HI on the terminal
G <DEST> CR	Begins execution at memory address DEST
I<DEST> CR Data	Allows hex data to be inserted into the memory starting at DEST. Carriage returns (after the first one) and spaces are ignored. An escape or illegal character terminates the command
M <LO> <HI> <DEST> CR	Move (copy) data block from LO through HI to RAM starting at DEST
S <DEST>	The byte of hex data at memory location DEST is displayed on the terminal. To change the value, type in a new byte; to keep the same value, type in a space. The next memory location will then be displayed, and the process repeated. Type in a carriage return to terminate the command
X	This command with a carriage return will display all the 8080 registers. The command with a space will display each register, and a new data value can be typed in
Z <LO> <HI> <DATA> CR	The value of hex data will be copied into RAM from LO through HI. This command is used to set a whole block of RAM to a given value

^aCR = Carriage return.

Table 5. COMP9 Monitor Routines

Call (Inverted Address)	Name	Function	Destroys Regs. ^a (F = Flags)
3E00	GETCM	Recognizes monitor commands C, D, G, I, M, S, X, and Z from the terminal. Transfers control to appropriate location for command execution. Reentrant point upon termination of user program	ABCHLF
6B00	DCMD	Executes "display memory" (D) command	ALL
A200	GCMD	Executes "begin execution" (G) command	ALL
C000	ICMD	Executes "insert into memory" (I) command	ALL
0A01	MCMD	Executes "more data within memory" (M) command	ALL
2401	SCMD	Executes "substitute in memory" (S) command	ALL
4801	XCMD	Executes "register examine and change" (X) command	ALL
C401	ZCMD	Executes "fill memory block" (Z) command	ALL
DD01	CCMD	Executes "compare data" (C) command	ALL
3502	BREAK	Checks for a break character from the terminal (usually an ESC). If a break character is present, a success return is taken, otherwise a failure return is taken	AF
4802	CI	Reads in one character from the terminal and places it in the A reg.	AF
5202	CNVBN	Converts the ASCII character in reg. C to its binary value. Stores result in A. No validity check on the original character is performed	AF
5B02	CO	Outputs to the terminal one ASCII character from reg. C	AF
6602	CROUT	Prints a <CR>, <LF> sequence on the terminal	ABCF

Table 5. (Continued)

Call (Inverted Address)	Name	Function	Destroys Regs. ^a (F = Flags)
6C02	ECHO	Prints the ASCII character in reg. C on the terminal. An ESC is echoed as a \$, and a <CR> as a <CR>, <LF> combination	ABF
8502	ERROR	Prints the error character on the terminal (usually a *), then a <CR>, <LF>, and then returns to the command recognizer	ABCF
9002	FRET	Indicates a failure return by setting carry false (equal to 0) and returns	F (carry only)
9302	GETCH	Places the next character from the terminal into reg. C and clears any parity bit to 0	ACF
9A02	GETHX	Converts the last four hex digits to a 16-bit binary value in regs. B, C. Illegal characters create an error condition. Returns when a delimiter is encountered	ABCDEF
CF02	CETNM	Finds the number of numbers in reg. C in the input stream and returns their binary values on the stack (last on top). 1, 2, or 3 values may be requested. If the first is greater than the second, it is set equal to the second. The last number must be followed by a <CR>	ALL
1403	HILO	Compares two unsigned 16-bit integers in HL and DE. If HL < DE, carry is set to 0; otherwise carry is set to 1	F
1F03	NMOUT	Prints the binary value in reg. A in hex on the terminal	ABCF
3A03	PRVAL	Converts a binary number in reg. C between 0H and 0FH into the corresponding ASCII character '0'-'9' or 'A'-'F' and puts the result in reg. C. No validity check is made	BCHLF

Table 5. (Continued)

Call (Inverted Address)	Name	Function	Destroys Regs. ^a (F = Flags)
4203	REGDS	Displays the previous register contents from memory in formatted form	ALL
7303	RGADR	Takes a single character in reg. C denoting a register and returns the address of the register contents in memory in register pair BC. If the register identifier is invalid, an error condition occurs	ALL
8A03	RSTTF	Restores all registers and flags from the memory save locations and then executes the G command	ALL
9F03	SRET	Indicates a success return by setting carry true (equal to 1) and returns	F (carry only)
A103	STHFØ	Checks the half-byte flag stored at location TEMP (3FF9H) in RAM. If it is set lower, it stores a 0 in the four least significant bits of the byte addressed by DE. Otherwise, no action is taken	ABCHLF
AC03	STHLF	Puts 4-bit value in C into byte addressed by DE. The lower or upper position is specified by TEMP (3FF9) in RAM	ABCHLF
CB03	VALDG	Checks to see if ASCII character in reg. C is a valid hex digit. Returns success if it is, and failure otherwise	AF
E603	VALDL	Checks to see if ASCII character in reg. C is a valid delimiter (space, comma, or <CR>). Returns success if it is, and failure otherwise	AF

^aReg. = register.

The user programs are usually started at hex location 800. When the interrupt is taken low, program control is transferred to location 800. By so doing, user programs can be started by pushing a button rather than by using a terminal.

4.2 COMP9 MATH COMMANDS

The COMP9 math commands contained in the program MATH9 permit the direct use of the 9511 APU by using only subroutine calls rather than the user having to move data in and out of the APU. These commands are summarized in Appendix E. In addition to the regular math functions, MATH9 contains subroutines for conversion from both fixed- and floating-point binary to decimal, subroutines to print the top of the APU stack in E (floating) or F (fixed) format on the terminal, and subroutines to print blocks of numbers, addressed by the HL registers on the terminal.

The data are stored in RAM in the order: HI XX XX LO. For commands that write data from memory to the APU stack, such as WRT4 or any of the operations with an A or B suffix, the HL register should contain the address of the lowest order byte (or DE registers for the B data). For the commands that move the top of the APU stack into RAM, such as TOS4, TOS2, and TOS1, the HL register should address the highest order byte of the data.

Some examples of the representation of the various types of numbers used in the APU are listed in Table 6.

The integers are stored as two's complement numbers with the 16-bit numbers going from -32,768 to +32,767 and the 32-bit numbers going from -2,147,483,648 to +2,147,483,647. The floating-point numbers go from $\pm 2.7 \times 10^{-20}$ to $\pm 9.2 \times 10^{18}$, with a 24-bit mantissa. The accuracy is about $\pm 1 \times 10^{-7}$ for most of the calculations, decreasing to about $\pm 5 \times 10^{-7}$ for some of the transcendental functions.

A Fortran program, MATHCON, to convert floating numbers on the ModComp IV to floating numbers for the COMP9 is presented in Appendix F.

The commands in MATH9 are listed as public and can be called from any user program by using the external statement. The addresses will be automatically determined when the programs are link edited.

Table 6. APU Representation of Some Numbers

Value	16-Bit Fixed (Hex Rep)	32-Bit Fixed (Hex)		32-Bit Floating (Hex)	
0	0000	0000	0000	0000	0000
1	0001	0000	0001	0180	0000
2	0002	0000	0002	0280	0000
4	0004	0000	0004	0380	0000
8	0008	0000	0008	0480	0000
16	0010	0000	0010	0580	0000
32	0020	0000	0020	0680	0000
64	0040	0000	0040	0780	0000
100	0064	0000	0064	07C8	0000
1000	03E8	0000	03E8	0AFA	0000
-1	FFFF	FFFF	FFFF	8180	0000
-1200	FB50	FFFF	FB50	8B96	0000
π				02C9	0FDA
π^2				049D	E9E5
0.1				7DCC	CCCC
0.01				7AA3	D708
-4.8828125 E-4				F680	0000
$e^{-0.2}$				00D1	9856
$\cos(-0.2)$				00FA	E5A6
$\sin(-0.2)$				FECB	6FF4
$\ln(0.2)$				81CE	0210
$1000^{0.5}$				05FC	FB73
$0.2^{0.1}$				00D9	F169

The math commands use RAM locations from 3C00 through 3C08 for scratch computations; the user should avoid these addresses. Because of the dual writing nature of the 1K- by 8-bit memory, locations 3800 through 380E should also be avoided when this memory type is used.

5. CONSTRUCTION DETAILS

The NDT-COMP9 microcomputer is assembled on a double-sided printed circuit board with plated-through holes. The board is 242 mm (9.5 in.) by 122 mm (4.8 in.), has 26-pin card edge connectors, and will fit sideways in a 4-width NIM BIN. Figure 5 shows the component layouts for the board. Foil layouts for both sides are shown in Figs. 6 and 7. Figures 8 and 9 are circuit diagrams for the processor and memory sections of the NDT-COMP9, respectively.

It is recommended that integrated circuit sockets should be used for mounting all integrated circuit chips to ensure ease of debugging and replacement of defective components. Preferably all conductors on the board would be gold plated.

Except for the initial artwork design, expenses are approximately \$30 for the printed circuit board and \$110 for all chips except the APU and memory. The APU is approximately \$200; 2K- by 8-bit EPROMs, \$35 each; and 1K- by 8-bit RAMs, \$30 each. A minimal memory version with the APU (1 EPROM and 1 RAM) would be approximately \$410.

See Appendix F for a detailed list of components.

6. PROM PROGRAMMING

The NDT-COMP9 uses Intel 2716 2K- by 8-bit EPROMs or equivalent (e.g., the Texas Instrument TMS 2516) for the nonvolatile portion of the memory. A PROM programmer has not been designed for use directly with the NDT-COMP9 board. In its place, a more versatile NIM BIN compatible programmer has been built that interfaces directly to a MODCOMP IV mini-computer. Commercial programmers for these EPROMs are readily available from several sources, including Intel* and Pro-Log Corporation.†

*Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

†Pro-Log Corporation, 2411 Garden Road, Monterey, CA 93940.

Notes: 1. CR1 to stand upright.

2. Capacitors are in μ F unless noted.

3. Resistors are in ohms and are $\frac{1}{4}$ watt unless noted.

4. Card edge connector terminal spacing is 0.100 inch.

5. Last C # used was 32.

Fig. 5. Component Layout for the NDT-COMP9A.

- NOTES:**
1. Material to be 1.58 mm [.062 in.] thick epoxy fiberglass type GNP, per MIL-P-13949/5. Specific designation shall be FL-GNP-062-C-1/B-2-B. Off-white color preferred.
 2. All holes shall be 100% plated thru with .038 mm [.015 in.] copper and .013 mm [.0005 in.] tin-lead inside holes. Front to rear registration to be within .127 mm [.005 in.]
 3. Clean and plate per ORNL Spec. Q-800-13.12. All copper, except connector contacts, to be plated with electro-deposited tin-lead. Plating shall contain 55% to 65% tin. Connector contacts shall be gold plated per ORNL Spec. Q-800-13.13.
 4. After processing, etched boards to be free of voids, scratches and warpage; and to be flat within .38 mm [.015 in.] overall.
 5. Do not use this drawing as a work sheet. Master negative(s) of wiring will be supplied by ORNL.
 6. Trim lines are denoted by broken lines [-----]. Flat cut required. Limit on all dimensions .38 mm [.015 in.] unless otherwise noted.
 7. None all edges smooth.
 8. Material required: 365 sq. cm. [57 sq. in.].
 9. Hole location tolerance 13 mm [.005 in.]
 10. Notches: 5 Key slots: 4 $\left[\begin{smallmatrix} .117mm \\ .046in. \end{smallmatrix} \times \begin{smallmatrix} .833mm \\ .325in. \end{smallmatrix} \right]$
 11. All holes to be ≈ 60 1.01 mm [.040 in.] drill 92 holes, except as noted below.

Symbol
Hole Dia.	mm	3.28	0.81	1.32			
	in.	0.129	0.032	0.052			
Drill size		30	67	55			
Quantity		4	659	2

Fig. 7. Component Side for the Printed Circuit Board.

NOTES:

1. EPROM IS INTEL 2716 OR TI TMS 2516
2. RAM IS TI TMS 4016 OR MODIFIED MOSTEK 4118

Fig. 9. Memory Schematic for the NDT-COMP9.

7. POWER REQUIREMENTS

The typical power requirements for a fully loaded NDT-COMP9 board are:

- +12 V at 105 mA
- +5 V at 690 mA
- 5 V at 2 μ A
- 12 V at 7 mA

ACKNOWLEDGMENTS

The authors would like to thank W. W. King for laying out the printed circuit boards, N. W. McCoy and C. F. Ruffner for constructing the computers, L. D. Chitwood for checking out the computers. The manuscript was reviewed by W. A. Simpson and W. A. Deeds, edited by Lydia Corrill, and prepared for reproduction by Denise Campbell.

REFERENCES

1. C. V. Dodd and G. D. Connell, *The NDT-COMP8 Microcomputer*, ORNL/TM-5113 (March 1977).
2. C. V. Dodd and L. D. Chitwood, *Three-Frequency Eddy-Current Instrument for Multiple Property Problems*, ORNL-5495 (March 1979).
3. Intel Corporation, *8080/8085 Assembly Language Programming Manual*, 9800301B, 1977.
4. Intel Corporation, *MCS-85TM User's Manual*, Santa Clara, Calif., January 1978, and *Component Data Catalog*, 1978.
5. Intel Corporation, *8080/8085 Assembly Language Programming Manual*, Santa Clara, Calif., 1977.
6. Advanced Micro Devices, Inc., *The 8080A/9080A MOS Microprocessor Handbook*, Sunnyvale, Calif., 1977.
7. Advanced Micro Devices, Inc., *Algorithm Details for the AM9511 Arithmetic Processing Unit*, Sunnyvale, Calif., 1978, and AMD data sheets.

APPENDIX A

8080/9080 INSTRUCTION SUMMARY

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table A1. 8080/9080 Instruction Summary

JUMP		CALL		RETURN		RESTART		ROTATE ^a		MOVE (cont.)		ACCUMULATOR ^b		ACCUMULATOR ^b (cont.)		CONSTANT DEFINITION		
C3 JMP	} Adr ^c	CD CALL	} Adr ^c	C9 RET	C7 RST 0	07 RLC	58 MOV E,B	80 ADD B	A8 XRA B	B								
C2 JNZ		CC CNZ		C0 RNZ	CF RST 1	0F RRC	59 MOV E,C	81 ADD C	AA XRA C	C								
CA JZ		CC CZ		C8 RZ	D7 RST 2	17 RAL	5A MOV E,D	82 ADD D	AB XRA D	E								
D2 JNC		D4 CNC		D0 RNC	DF RST 3	1F RAR	5B MOV E,E	83 ADD E	AC XRA H	H								
DA JC		DC CC		D8 RC	E7 RST 4		5C MOV E,H	84 ADD H	AD XRA L	L								
E2 JPO		E4 CPO		E0 RPO	EF RST 5		5D MOV E,L	85 ADD L	AE XRA M	M								
EA JPE		EC CPE		E8 RPE	F7 RST 6		5E MOV E,M	86 ADD M	AF XRA A	A								
F2 JP		F4 CP		F0 RP	FF RST 7		5F MOV E,A	87 ADD A										
FA JM	FC CM	F8 RM																
E9 PCHL																		

^aOnly CARRY affected.^bAll Flags (C,Z,S,P,AC) affected.^cAdr = 16-bit address. D8 = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity. D16 = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity.^dAll Flags except CARRY affected; (exception INX & DCX affect no Flags).

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

APPENDIX B

APU INSTRUCTION SUMMARY

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table B1. APU Instruction Summary

Command Mnemonic	Hex Code (sr = 0)	Execution Cycles	Summary Description
<u>16-Bit Fixed-Point Operations</u>			
SADD	6C	16-18	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	6D	30-32	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	6E	84-94	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
SMUU	76	80-98	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
SDIV	6F	84-94	Divide NOS by TOS. Result to NOS. Pop Stack.
<u>32-Bit Fixed-Point Operations</u>			
DADD	2C	20-22	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	2D	38-40	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	2E	194-210	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
DMUU	36	182-218	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
DDIV	2F	196-210	Divide NOS by TOS. Result to NOS. Pop Stack.
<u>32-Bit Floating-Point Primary Operations</u>			
FADD	10	54-368	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	11	70-370	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	12	146-168	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	13	154-184	Divide NOS by TOS. Result to NOS. Pop Stack.
<u>32-Bit Floating-Point Derived Operations</u>			
SQRT	01	782-870	Square root of TOS. Result to TOS.
SIN	02	3796-4808	Sine of TOS. Result to TOS.
COS	03	3840-4878	Cosine of TOS. Result to TOS.
TAN	04	4894-5886	Tangent of TOS. Result to TOS.
ASIN	05	6230-7938	Inverse Sine of TOS. Result to TOS.
ACOS	06	6304-8284	Inverse Cosine of TOS. Result to TOS.
ATAN	07	4992-6536	Inverse Tangent of TOS. Result to TOS.
LOG	08	4474-7132	Common Logarithm of TOS. Result to TOS.
LN	09	4298-6956	Natural Logarithm of TOS. Result to TOS.
EXP	0A	3794-4878	e raised to power in TOS. Result to TOS.
PWR	0B	8290-12032	NOS raised to power in TOS. Result to NOS. Pop Stack.

Table B1. (Continued)

Command Mnemonic ^a	Hex Code (sr = 0)	Execution Cycles	Summary Description
<u>Data and Stack Manipulation Operations</u>			
NOP	00	4	No Operation. Clear or set SVREQ.
FIXS	1F	90-214	Convert TOS from floating-point format to fixed-point format.
FIXD	1E	90-336	
FLTS	1D	62-156	Convert TOS from fixed-point format to floating-point format.
FLTD	1C	56-342	
CHSS	74	22-24	Change sign of fixed-point operand on TOS.
CHSD	34	26-28	
CHSF	15	16-20	Change sign of floating-point operand on TOS.
PTOS	77	16	Push stack. Duplicate NOS in TOS.
PTOD	37	20	
PTOF	17	20	
POPS	78	10	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom.
POPD	38	12	
POPF	18	12	
XCHS	79	18	Exchange TOS and NOS.
XCHD	39	26	
XCHF	19	26	
PUPI	1A	16	Push floating-point constant π onto TOS. Previous TOS becomes NOS.

^aCommand mnemonics:

ACOS = ARCCOSINE
 ASIN = ARCSINE
 ATAN = ARCTANGENT
 CHSD = CHANGE SIGN DOUBLE
 CHSF = CHANGE SIGN FLOATING
 CHSS = CHANGE SIGN SINGLE
 COS = COSINE
 DADD = DOUBLE ADD
 DDIV = DOUBLE DIVIDE
 DMUL = DOUBLE MULTIPLY LOWER
 DMUU = DOUBLE MULTIPLY UPPER
 DSUB = DOUBLE SUBTRACT
 EXP = EXPONENTIATION (e^x)
 FADD = FLOATING ADD
 FDIV = FLOATING DIVIDE
 FIXD = FIX DOUBLE
 FIXS = FIX SINGLE
 FLTD = FLOAT DOUBLE
 FLTS = FLOAT SINGLE
 FMUL = FLOATING MULTIPLY
 FSUB = FLOATING SUBTRACT

LOG = COMMON LOGARITHM
 LN = NATURAL LOGARITHM
 NOP = NO OPERATION
 POPD = POP STACK DOUBLE
 POPF = POP STACK FLOATING
 POPS = POP STACK SINGLE
 PTOD = PUSH STACK DOUBLE
 PTOF = PUSH STACK FLOATING
 PTOS = PUSH STACK SINGLE
 PUPI = PUSH π
 PWR = POWER (x^y)
 SADD = SINGLE ADD
 SDIV = SINGLE DIVIDE
 SIN = SINE
 SMUL = SINGLE MULTIPLY LOWER
 SMUU = SINGLE MULTIPLY UPPER
 SQRT = SQUARE ROOT
 SSUB = SINGLE SUBTRACT
 TAN = TANGENT
 XCHD = EXCHANGE OPERANDS DOUBLE
 XCHF = EXCHANGE OPERANDS FLOATING
 XCHS = EXCHANGE OPERANDS SINGLE

APPENDIX C

MATH9 SUBROUTINE CALLS

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table C1. MATH9 Subroutine Calls

Sub-routine	Call	Function	With 1 Argument Addressed by H, L in RAM	Call	With A Argument Addressed by H, L; B ARG by D, E	Call
ACOS	6304	Arccosine	ACOSA	6004		
ASIN	6804	Arcsine	ASINA	6804		
ATAN	7304	Arctangent	ATANA	7004		
BIDEC	B904	Fixed binary to decimal				
BIDECF	7804	Floating binary to decimal				
CHSD	2105	Change sign double	CHSDA	1E05		
CHSF	2905	Change sign floating	CHSFA	2605		
CHSS	3105	Change sign single	CHSSA	2E05		
COS	3905	Cosine	COSA	3605		
DADD	4505	Double add	DADDA	4205	DADDB	3E05
DDIV	5105	Double divide (B/A)	DDIVA	4E05	DDIVB	4A05
DMUL	5D05	Double multiply lower	DMULA	5A05	DMULB	5605
DMUJ	6905	Double multiply upper	DMUJA	6605	DMUJB	6205
DSUB	7505	Double subtract (B-A)	DSUBA	7205	DSUBB	6E05
EPRNT	0007	Prints in "E" format, D/LINE for E lines starting at LOC HL				
EXP	7D05	Exponentiation (e^x)	EXPA	7A05		
FADD	8905	Floating add	FADDA	8605	FADDB	8205
FDIV	9505	Floating divide (B/A)	FDIVA	9205	FDIVB	8E05
FIXD	A505	Fix double	FIXDA	A205		
FIXS	9D05	Fix single	FIXSA	9A05		
FLTD	AD05	Float double	FLTDA	AA05		
FLTS	B505	Float single	FLTSA	B205		
FMUL	C105	Floating multiply	FMULA	BE05	FMULB	BA05
FPRNT	2907	Prints in fixed format B.C, D/LINE for E lines starting at LOC HL				
FSUB	CD05	Floating subtract (B-A)	FSUBA	CA05	FSUBB	C605
LOG	D505	Common logarithm	LOGA	D205		
LN	DD05	Natural logarithm	LNA	DA05		
NOP	E205	No operation				
POPD	E705	POP stack double				
POPS	EC05	POP stack single				
PRINT	9206	Prints TOS in floating decimal				
PRINTF	4C07	Prints TOS in fixed format B.C				
PTOD	F105	Push stack double				
PTOS	F605	Push stack single				
PUPI	FB05	Push π				
PWR	0706	Power ($B^{**}A$)	PWRA	0406	PWRB	0006
SADD	1306	Single add	SADDA	1006	SADDB	0C06
SDIV	1F06	Single divide B/A	SDIVA	1C06	SDIVB	1806
SIN	2706	Sine	SINA	2406		
SMUL	3306	Single multiply lower	SMULA	3006	SMULB	2C06
SMUJ	3F06	Single multiply upper	SMUJA	3C06	SMUJB	3806
SQRT	4706	Square root	SQRTA	4406		
SSUB	5306	Single subtract (B-A)	SSUBA	5006	SSUBB	4C06
TAN	5B06	Tangent	TANA	5806		
TOS4	6006	Move top of stack to H.L in RAM	TOS2	6806	TOS1	6C06
WRT4	7006	Copy H, L in RAM to TOS	WRT2	7806	WRT1	7C06
XCHD	8006	Exchange operands double				
XCHS	8506	Exchange operands single				

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

APPENDIX D

COMP9 SYSTEM MONITOR ASSEMBLY LISTING

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK.

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 1

```

1      TITLE      'NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979'
2      NAME      COMP9
3      LIST      B,G,O,T ; LIST SYMBOL TABLE ONLY - PUT SYMBOLS
4                  ; INTO OBJECT MODULE TO ALLOW CREATION
5                  ; OF A LOAD MAP
6      ;          NLIST      I,M,S,X ; DO NOT LIST SOURCE TEXT
7      PUBLIC    CI,CNVBN,CO,COPDT,CROUT,DCMD,ECHO,ERROR,FRET
8      PUBLIC    GCMD,GETCH,GETCM,GETHX,GETNM,HILO,ICMD,NMOUT
9      PUBLIC    PRVAL,SCMD,VALDG,VALDL,XCMD
10     CSEG      ; USE RELOCATABLE CODE COUNTER
11
12
13     ;*****
14     ;*****
15     ;
16     ;
17     ;          NDT-COMP9 MONITOR
18     ;
19     ;          AUTHORS C. V. DODD AND R. F. COWAN
20     ;
21     ;          NONDESTRUCTIVE TESTING GROUP
22     ;          METALS AND CERAMICS DIVISION
23     ;          OAK RIDGE NATIONAL LABORATORIES
24     ;          OAK RIDGE, TENNESSEE 37830
25     ;
26     ;          NOTE: THE COMP9 MONITOR WAS ADAPTED FROM THE COMP8,
27     ;          WHICH IS A MODIFIED VERSION OF THE INTEL 8080A
28     ;          BOARD MONITOR.
29     ;
30     ;
31     ;
32     ;*****
33     ;*****
34     ;
35     ;
36     ;
37     ;
38     ;*****
39     ;
40     ;
41     ;          SYMBOL DEFINITIONS
42     ;
43     ;
44     ;
45     ;*****
46     ;
47
48 001B BRCHR EQU 1BH ; ASCII CODE FOR BREAK CHAR (ESCAPE)
49 03FA BRTAB EQU 3FAH ; LOCATION OF START OF BRANCH TABLE IN PROM
50 0027 CMD EQU 27H ; COMMAND INSTRUCTION FOR USART INITIALIZATION
51 00FB CNCTL EQU 0FBH ; CONSOLE (USART) CONTROL PORT
52 00FA CNIN EQU 0FAH ; CONSOLE INPUT PORT
53 00FA CNOUT EQU 0FAH ; CONSOLE OUTPUT PORT
54 00FB CONST EQU 0FBH ; CONSOLE STATUS INPUT PORT

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 2

```

55 000D CR EQU 0DH ; ASCII CODE FOR CARRIAGE RETURN
56 3F00 DATA EQU 16*1024-256 ; START OF MONITOR RAM USAGE AT
57 ; (MEM SIZE IN K)*1K-256 BYTES
58 0010 ESC EQU 10H ; ASCII CODE FOR BREAK CHAR
59 000F HCHAR EQU 0FH ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
60 00FF INVRT EQU 0FFH ; MASK TO INVERT HALF BYTE FLAG
61 000A LF EQU 0AH ; ASCII CODE FOR LINE FEED
62 0000 LOWER EQU 0 ; DENOTES LOWER HALF OF BYTE IN ICMD
63 ;LSGNON -- LENGTH OF SIGNON MESSAGE - DEFINED LATER
64 00CE MODE EQU 0CEH ; MODE SET FOR USART INITIALIZATION
65 ;NCMDS -- NUMBER OF VALID COMMANDS - DEFINED LATER
66 000D NLSB EQU 00DH ; LSB OF RTC DIVIDE COUNT
67 0000 NMSB EQU 00H ; MSB OF RTC DIVIDE COUNT
68 000F NEWLN EQU 0FH ; MASK FOR CHECKING MEMORY ADDR DISPLAY
69 007F PRTY0 EQU 07FH ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
70 0002 RBR EQU 2 ; MASK TO TEST RECEIVER STATUS FOR A 1
71 3FED REGS EQU DATA+255-10 ; START OF REGISTER SAVE AREA
72 0030 RSTU EQU 30H ; TRANSFER LOC FOR RST7 INSTRUCTION
73 000F RTCCTL EQU 00FH ; REAL-TIME CLOCK CONTROL PORT
74 000E RTCDAT EQU 00EH ; REAL-TIME CLOCK DATA (PORT*2)
75 000E RTCMD EQU 00EH ; MODE WORD FOR REAL-TIME CLOCK
76 ;RTABS -- SIZE OF ENTRY IN RTAB TABLE - DEFINED LATER
77 0010 TERM EQU 10H ; ASCII CODE FOR ICMD TERMINATING CHAR (ESCAPE)
78 0001 TRDY EQU 1 ; MASK TO TEST TRANSMITTER STATUS
79 00FF UPPER EQU 0FFH ; DENOTES UPPER HALF OF BYTE IN ICMD
80 0000 USINT EQU 0000H ; INTERRUPT BRANCH LOCATION
81 ;
82 ;
83 ;
84 ;
85 ;*****
86 ;
87 ;
88 ;
89 ;
90 ;
91 ;*****
92 ;
93 ;
94 3FED MSTAK EQU 3FEDH ; START OF MONITOR STACK
95 3FF2 ASAVE EQU 3FF2H ; REGISTER SAVE LOCS
96 3FF0 BCNVE EQU 3FF0H
97 3FEF CSAVE EQU 3FEFH
98 3FEE DSAVE EQU 3FEEH
99 3FED ESAVE EQU 3FEDH
100 3FF1 FSAVE EQU 3FF1H
101 3FF4 HSAVE EQU 3FF4H
102 3FF3 LSAVE EQU 3FF3H
103 3FF5 PSAVE EQU 3FF5H ; PC SAVE LOC
104 3FF7 SSAVE EQU 3FF7H ; USER SP SAVE LOC
105 3FF9 TEMP EQU 3FF9H ; TEMPORARY MONITOR CELL
106 ;
107 ;
108 ;*****

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 3

```

109 ;
110 ;
111 ; MONITOR MACROS
112 ;
113 ;
114 ;*****
115 ;
116 ;
117 TRUE MACRO WHERE ; BRANCH IF FUNCTION RETURNS TRUE (SUCCESS)
118 JC WHERE
119 ENDM
120 ;
121 ;
122 FALSE MACRO WHERE ; BRANCH IF FUNCTION RETURNS FALSE (FAILURE)
123 JNC WHERE
124 ENDM
125 ;
126 ;
127 ;
128 ;
129 ;*****
130 ;
131 ;
132 ; 8253 REAL-TIME CLOCK INITIALIZATION
133 ;
134 ;
135 ; THE REAL-TIME CLOCK MUST BE INITIALIZED FIRST, SINCE IT
136 ; DETERMINES THE CLOCK FREQUENCY AND HENCE THE BAUD RATE
137 ; FOR THE USART. A MODE WORD IS SENT FIRST SPECIFYING WHICH
138 ; COUNTER AND MODE, AND THEN A 16-BIT INTEGER N TO PROVIDE A
139 ; DIVIDE-BY-N FUNCTION. THE TABLE BELOW SPECIFIES WHICH INTEGE
140 ; TO USE FOR A GIVEN TIME-BASE:
141 ;
142 ;
143 ; BAUD RATE N WITH 18.432 XTAL N WITH 18.000 XTAL
144 ; 9600 000D 000D
145 ; 4800 001B 001A
146 ; 2400 0035 0034
147 ; 1200 006B 006B
148 ; 300 01AA 01A1
149 ;
150 ; THIS ASSUMES A DIVIDE-BY-16 FACTOR FOR THE USART.
151 ;
152 ; THE MODE WORD SPECIFIES:
153 ;
154 ; SELECT COUNTER #2
155 ; LOAD LSB FIRST, THEN MSB OF N
156 ; RUN IN MODE 3 (SQUARE WAVE OUTPUT)
157 ; LOAD IN BINARY
158 ;
159 ;*****
160 ;
161 ;
162 0000 3E BE START: MVI A,RTCMD

```


ERR LINE	ADDR	B1 B2 B3 B4	NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979	PAGE 4
163	0002	D3 BF	OUT RTCTL ; SEND MODE WORD TO RTC	
164	0004	3E 0D	MVI A,NLSB	
165	0006	D3 BE	OUT RTCDAT ; SEND LSB OF DIVIDE COUNT	
166	0008	3E 00	MVI A,NMSB	
167	000A	D3 BE	OUT RTCDAT ; SEND MSB OF DIVIDE COUNT	
168			;	
169			;	
170			;	
171			;	
172			*****	
173			;	
174			;	
175			USART INITIALIZATION CODE	
176			;	
177			;	
178			THE USART IS ASSUMED TO COME UP IN THE RESET POSITION (THIS	
179			FUNCTION IS TAKEN CARE OF BY THE HARDWARE). THE USART WILL	
180			BE INITIALIZED FOR A CRT INTERFACE. THE PARAMETERS BELOW	
181			APPLY:	
182			;	
183			MODE INSTRUCTION	
184			-----	
185			;	
186			2 STOP BITS	
187			PARITY DISABLED	
188			8 BIT CHARS	
189			INTERNAL FREQUENCY DIVIDE-BY-16	
190			;	
191			COMMAND INSTRUCTION	
192			-----	
193			;	
194			NO HUNT MODE	
195			NOT(RTS) FORCED TO ZERO	
196			RECEIVE ENABLED	
197			DATA TERMINAL READY	
198			TRANSMIT ENABLED	
199			;	
200			;	
201			*****	
202			;	
203			;	
204	000C	3E CE	MVI A,MODE ;.16X,2 STOP BITS,NO PARITY,8 BIT CHARACTER.	
205	000E	D3 FB	OUT CNCTL ; OUTPUT MODE SET TO USART	
206	0010	3E 27	MVI A,CMD	
207	0012	D3 FB	OUT CNCTL ; OUTPUT COMMAND WORD TO USART	
208			;	
209			;	
210			*****	
211			;	
212			;	
213			RESTART ENTRY POINT	
214			;	
215			;	
216			*****	

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 5

```

217 ;
218 ;
219 0014 22 F3 3F GO: SHLD LSAVE ; SAVE HL REGISTERS
220 0017 E1 POP H ; GET TOP OF STACK ENTRY
221 0018 22 F5 3F SHLD PSAVE ; ASSUME THIS IS LAST PC
222 0018 21 00 00 LXI H,0 ; CLEAR HL
223 001E 39 DAD SP ; GET STACK POINTER VALUE
224 001F 22 F7 3F SHLD SSAVE ; SAVE USER'S STACK POINTER
225 0022 21 F3 3F LXI H,ASAVE+1 ; NEW VALUE FOR STACK POINTER
226 0025 F9 SPHL ; SET MONITOR STACK POINTER FOR REG SAVE
227 0026 F5 PUSH PSW ; SAVE A AND FLAGS
228 0027 C5 PUSH B ; SAVE B AND C
229 0028 D5 PUSH D ; SAVE D AND E
230 ;
231 ;
232 ; *****
233 ;
234 ;
235 ; PRINT LOGON MESSAGE
236 ;
237 ;
238 ; *****
239 ;
240 ;
241 0029 21 F9 03 C LXI H,SGNON ; GET ADDR OF SIGNON MESSAGE
242 002C 06 0D MVI B,LSGNON ; COUNTER FOR CHARACTERS IN MESSAGE
243 002E 4E MSGL: MOV C,M ; FETCH NEXT CHAR TO C REG
244 002F CD 5B 02 C CALL CO ; SEND IT TO THE CONSOLE
245 0032 23 INX H ; POINT TO NEXT CHARACTER
246 0033 05 DCR B ; DECREMENT BYTE COUNTER
247 0034 C3 3B 00 C JMP PRSGN ; WANT TO LEAVE ROOM FOR RST BRANCH
248 ;
249 ORG RSTU ; ORG TO RST TRANSFER LOC
250 003B C3 00 00 JMP USINT ; JUMP TO USER INTERRUPT LOC
251 ;
252 003B C2 2E 00 C PRSGN: JNZ MSGL ; RETURN FOR NEXT CHARACTER
253 ;
254 ;
255 ; *****
256 ;
257 ;
258 ;
259 ; ROUTINE GETCM
260 ;
261 ; THIS ROUTINE RECOGNIZES COMMAND CHAR'S INPUT BY THE USER.
262 ; UPON RECEIPT OF A CHAR FROM THE TERMINAL, IT CHECKS IF
263 ; IT IS IN THE COMMAND CHARACTER TABLE. IF THE CHAR IS FOUND,
264 ; THE CORRECT ROUTINE IS SELECTED FROM A TABLE OF COMMAND ROUTI
265 ; ADDRESSES AND CONTROL IS TRANSFERRED TO THAT ROUTINE. IF THE
266 ; ORIGINAL CHAR IS NOT A VALID COMMAND, AN ERROR CONDITION IS
267 ; CREATED. THIS IS THE NORMAL RETURN POINT TO MONITOR FOR TEST
268 ; ROUTINES. A PROMPT "." IS PRINTED WHEN MONITOR IS READY.
269 ;
270 ;
271 ; EXTERNAL REFERENCES:
272 ; -----

```

```

ERR LINE ADDR B1 B2 B3 B4 NDT-COMP# MONITOR ASSEMBLY VER. 9 APRIL 1979 PAGE 6

271 ;
272 ; ECHO ERROR GETCH
273 ;
274 ; REGISTERS AFFECTED: A,B,C,H,L,FLAGS
275 ;
276 ;
277 ;
278 ;
279 ;
280 003E 21 ED 3F GETCM: LXI H,MSTAK ; ALLWAS WANT TO RESET STACK PTR TO MONITOR
281 0041 F9 SPHL ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
282 0042 0E 2E MVI C,'.' ; PROMPT CHARACTER TO C
283 0044 CD 6C 02 C CALL ECHO ; SEND PROMPT CHAR TO USER TERMINAL
284 0047 F8 EI ; ENABLE INTERRUPTS
285 0048 CD 93 02 C CALL GETCH ; GET COMMAND CHARACTER TO A
286 0048 CD 6C 02 C CALL ECHO ; ECHO CHARACTER TO USER
287 004E 79 MOV A,C ; PUT COMMAND CHAR INTO ACCUMULATOR
288 004F 01 00 00 LXI B,NCMD5 ; C CONTAINS LOOP AND INDEX COUNT
289 0052 21 10 04 C LXI H,CTAB ; HL POINTS INTO COMMAND TABLE
290 0055 0E GTC05: CMP H ; COMPARE TABLE ENTRY AND CHAR
291 0056 CA 61 00 C JZ GTC10 ; BRANCH IF EQUAL - COMMAND RECOGNIZED
292 0059 23 INX H ; ELSE, INCREMENT TABLE POINTER
293 005A 0D DCR C ; DECREMENT LOOP COUNT
294 0058 C2 55 00 C JNZ GTC05 ; BRANCH IF NOT AT TABLE END
295 005E C3 85 02 C JMP ERROR ; ELSE, COMMAND CHAR IS ILLEGAL
296 0061 21 06 04 C GTC10: LXI H,CADR ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
297 0064 09 DAD B ; ADD WHAR IS LEFT OF LOOP COUNT
298 0065 09 DAD B ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
299 0066 7E MOV A,M ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
300 0067 23 INX H ; POINT TO NEXT BYTE IN TABLE
301 0068 66 MOV H,M ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
302 0069 6F MOV L,A ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
303 006A E9 PCHL ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
304 ;
305 ;
306 ;
307 ;
308 ;
309 ;
310 ; COMMAND IMPLEMENTING ROUTINES
311 ;
312 ;
313 ;
314 ;
315 ;
316 ;
317 ; ROUTINE DCMD
318 ;
319 ; THIS ROUTINE IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND.
320 ;
321 ;
322 ; EXTERNAL REFERENCES:
323 ; -----
324 ;
325 ; CROUT ECHO GETCM GETNM HILO

```

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 7

325      ;      NMOUT
326      ;
327      ;      REGISTERS AFFECTED:  ALL
328      ;
329      ;
330      ;
331      ;
332      ;
333      006B  0E 02      DCM0:  MVI    C,2      ; GET 2 NUMBERS FROM INPUT STREAM
334      006D  CD CF 02      C      CALL    GETNM
335      0070  D1          POP    D      ; ENDING ADDRESS TO DE
336      0071  E1          POP    H      ; STARTING ADDRESS TO HL
337      0072  CD 66 02      C      DCM05: CALL    CROUT ; ECHO CARRIAGE RETURN/LINE FEED
338      0075  7C          MOV    A,H      ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
339      0076  CD 1F 03      C      CALL    NMOUT
340      0079  7D          MOV    A,L      ; ADDRESS IS 2 BYTES LONG
341      007A  CD 1F 03      C      CALL    NMOUT
342      007D  0E 20      DCM10: MVI    C, " "
343      007F  CD 6C 02      C      CALL    ECHO      ; USE BLANK AS SEPARATOR
344      0082  7E          MOV    A,M      ; GET CONTENTS OF NEXT MEMORY LOC
345      0083  CD 1F 03      C      CALL    NMOUT      ; DISPLAY CONTENTS
346      0086  CD 35 02      C      CALL    BREAK      ; SEE IF USER WANTS OUT
347      0089          TRUE    DCM12      ; IF SO, BRANCH
348      0089  DA 92 00      C +      JC      DCM12
349      008C  CD 14 03      C      CALL    HILO      ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
350      ;                  ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
351      008F          FALSE    DCM15      ; IF NOT, MORE TO DISPLAY
352      008F  D2 98 00      C +      JNC     DCM15
353      0092  CD 66 02      C      DCM12: CALL    CROUT ; CARRIAGE RETURN/LINE FEED TO END LINE
354      0095  C3 3E 00      C      JMP      GETCM      ; ALL DONE
355      0098  23          DCM15: INX    H      ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
356      0099  7D          MOV    A,L      ; GET LOW ORDER BITS OF NEW ADDRESS
357      009A  E6 0F          ANI    NEWLN      ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
358      ;                  ; /START OF NEW LINE
359      009C  C2 7D 00      C      JNZ     DCM10      ; NO - NOT AT END OF LINE
360      009F  C3 72 00      C      JMP      DCM05      ; YES - START NEW LINE WITH ADDRESS
361      ;
362      ;
363      ;
364      ;
365      ;
366      ;      ROUTINE GCM0
367      ;
368      ;      THIS ROUTINE IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
369      ;
370      ;
371      ;      EXTERNAL REFERENCES:
372      ;      -----
373      ;      ERROR      GETHX      RSTYP
374      ;
375      ;      REGISTERS AFFECTED:  ALL
376      ;
377      ;
378      ;

```

ERR LINE	ADDR	B1 B2 B3 B4	NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979	PAGE 8
379			;	
380			;	
381	00A2	CD 9A 02	C GCM0: CALL GETHX ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM	
382	00A5		FALSE GCM05 ; BRANCH IF NO NUMBER PRESENT	
383	00A5	D2 07 00	C + JNC GCM05	
384	00A8	7A	MOV A,D ; ELSE, GET TERMINATOR	
385	00A9	FE 0D	CPI CR ; SEE IF CARRIAGE RETURN	
386	00AB	C2 05 02	C JNZ ERROR ; ERROR IF NOT PROPERLY TERMINATED	
387	00AE	21 F5 3F	LXI H,PSAVE ; WANT NUMBER TO REPLACE SAVE PGM COUNTER	
388	00B1	71	MOV M,C	
389	00B2	23	INX H	
390	00B3	70	MOV M,B	
391	00B4	C3 0D 00	C JMP GCM10	
392	00B7	7A	GCM05: MOV A,D ; IF NO STARTING ADDRESS, MAKE SURE THAT	
393	00B8	FE 0D	CPI CR ; /CARRIAGE RETURN TERMINATED COMMAND	
394	00BA	C2 05 02	C JNZ ERROR ; ERROR IF NOT	
395	00BD	C3 0A 03	C GCM10: JMP RSTTF ; RESTORE REGISTERS AND BEGIN EXECUTION	
396			;	
397			;	
398			*****	
399			;	
400			;	
401			ROUTINE ICMD	
402			;	
403			THIS ROUTINE IMPLEMENTS THE INSERT INTO MEMORY (I) COMMAND.	
404			;	
405			EXTERNAL REFERENCES:	
406			-----	
407			;	
408			CNVBN CROUT ECHO ERROR GETCH	
409			GETNM STHLF VALDG VALDL	
410			;	
411			REGISTERS AFFECTED: ALL	
412			;	
413			;	
414			*****	
415			;	
416			;	
417	00C0	0E 01	ICMD: MVI C,1	
418	00C2	CD CF 02	C CALL GETNM ; GET SINGLE NUMBER FROM INPUT STREAM	
419	00C5	7F FF	MVI A,UPPER	
420	00C7	32 F9 3F	STA TEMP ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG	
421	00CA	D1	POP D ; DE GETS ADDR OF START	
422	00CB	CD 93 02	C ICM05: CALL GETCH ; GET ONE CHAR FROM INPUT STREAM	
423	00CE	4F	MOV C,A	
424	00CF	CD 6C 02	C CALL ECHO ; ECHO THE CHAR	
425	00D2	79	MOV A,C ; RESTORE IT INTO A	
426	00D3	FE 1B	CPI TERM ; IS IT A TERMINATING CHAR?	
427	00D5	CA 01 01	C JZ ICM25 ; YES - NO MORE CHARS COMING IN	
428	00D8	CD E6 03	C CALL VALDL ; NO - IS IT A VALID DELIMITER?	
429	00D8		TRUE ICM05 ; YES - IGNORE IT	
430	00DB	DA CB 00	C + JC ICM05	
431	00DE	CD CB 03	C CALL VALDG ; NO - WELL, IS IT A VALID HEX DIGIT?	
432	00E1		FALSE ICM20 ; NO - MUST BE AN ERROR	

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 9

433 00E1 D2 FB 00      C +      JNC      ICM20
434 00E4 CD 52 02      C          CALL    CNVBN      ; YES - AT LAST A GOOD DIGIT.  CONVERT IT TO BINA
435 00E7 4F          C          MOV      C,A      ; MOVE RESULT TO C
436 00E8 CD AC 03      C          CALL    STHLF      ; STORE IN APPROPRIATE HALF WORD
437 00E8 3A F9 3F      C          LDA      TEMP      ; GET HALF BYTE FLAG
438 00EE 87          C          ORA      A      ; SET FLAGS
439 00EF C2 F3 00      C          JNZ      ICM10      ; BRANCH IF FLAG SET FOR UPPER
440 00F2 13          C          INX      D      ; IF LOWER, INC ADDR OF DESTINATION BYTE
441 00F3 EE FF          C          ICM10: XRI    INVRT      ; TOGGLE FLAG
442 00F5 32 F9 3F      C          STA      TEMP      ; PUT NEW VALUE OF FLAG BACK
443 00F8 C3 CB 00      C          JMP      ICM05      ; LOOP AND DO ANOTHER CHAR
444 00FB CD A1 03      C          ICM20: CALL  STHF0      ; ILLEGAL CHAR
445 00FE C3 85 02      C          JMP      ERROR      ; FILL BYTE AND ERROR
446 0101 CD A1 03      C          ICM25: CALL  STHF0      ; BRANCH TO HERE WHEN INPUT FINISHED
447 0104 CD 66 02      C          CALL    CROUT      ; PRINT A CR,LF
448 0107 C3 3E 00      C          JMP      GETCM      ; RETURN TO COMMAND INTERPRETER
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470 010A 0E 03      C          MCM0: MVI      C,3
471 010C CD CF 02      C          CALL    GETNM      ; GET 3 INPUT VALUES
472 010F D1          C          POP      D      ; DE=DESTINATION ADDRESS OF FIRST BYTE
473 0110 C1          C          POP      B      ; BC-END ADDRESS OF BLOCK TO BE TRANSFERED
474 0111 E1          C          POP      H      ; HL=STARTING ADDRESS OF BLOCK TO BE XFERED
475 0112 CD 18 01      C          CALL    COPLP      ; CALL COPY DATA LOOP
476 0115 C3 3E 00      C          JMP      GETCM      ; JUMP BACK TO GET COMMAND PROGRAM
477 0118 7E          C          COPLP: MOV     A,M      ; MOVE BYTE IN MEM TO A
478 0119 12          C          STAX     D      ; STORE SAME BYTE IN LOC ADDRESSED BY DE
479 011A 23          C          INX      H
480 011B 13          C          INX      D
481 011C 79          C          COPDT: MOV    A,C      ; SET UP TEST TO DETERMINE IF HL>BC
482 011D 95          C          SRA      A      ; SUBTRACT HI FROM AC - GET A "BORROW"
483 011E 78          C          MOV     A,B      ; WHEN HL>BC (CARRY IS SET)
484 011F 9C          C          SBB      H      ; THEN RETURN TO CALL SUB
485 0120 D0          C          RC          ; RETURN IF DONE (HL>BC)
486 0121 C3 18 01      C          JMP      COPLP      ; OTHERWISE DO ANOTHER BYTE

ROUTINE MCM0
THIS ROUTINE IMPLEMENTS THE MOVE DATA WITHIN MEMORY (M)
COMMAND.

EXTERNAL REFERENCES:
-----

GETCM      GETNM

REGISTERS AFFECTED:  ALL

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 18

```

487 ;
488 ;
489 ;*****
490 ;
491 ;
492 ; ROUTINE SCMD
493 ;
494 ; THIS ROUTINE IMPLEMENTS THE SUBSTITUTE IN MEMORY (S) COMMAND.
495 ;
496 ; EXTERNAL REFERENCES:
497 ; -----
498 ;
499 ; ECHO GETCM GETHX NMOUT
500 ;
501 ; REGISTERS AFFECTED: ALL
502 ;
503 ;
504 ;*****
505 ;
506 ;
507 0124 CD 9A 02 C SCMD: CALL GETHX ; GET A NUMBER, IF PRESENT, FROM INPUT
508 0127 C5 PUSH B ;
509 0128 E1 POP H ; PUT IT INTO HL
510 0129 7A SCMD5: MOV A,D ; GET TERMINATOR
511 012A FE 20 CPI ' ' ; SEE IF SPACE
512 012C CA 34 01 C JZ SCMD10 ; YES - CONTINUE
513 012F FE 2C CPI ',' ; NO - THEN CHECK IF COMMA
514 0131 C2 3E 00 C JNZ GETCM ; NO - TERMINATE COMMAND
515 0134 7E SCMD10: MOV A,M ; GET CONTENTS OF SPECIFIED LOC
516 0135 CD 1F 03 C CALL NMOUT ; DISPLAY THEM
517 0138 0E 2D MVI C,'-'
518 013A CD 6C 02 C CALL ECHO ; USE DASH FOR SEPARATOR
519 013D CD 9A 02 C CALL GETHX ; GET NEW VALUE FOR MEM LOC, IF ANY
520 0140 FALSE SCMD15 ; IF NO VALUE THERE, THEN ADVANCE
521 0140 D2 44 01 C + JNC SCMD15
522 0143 71 MOV M,C ; OTHERWISE STORE LSB OF VALUE ENTERED
523 0144 23 SCMD15: INX H ; INCREMENT ADDR OF MEM LOC TO EXAMINE
524 0145 C3 29 01 C JMP SCMD5
525 ;
526 ;
527 ;*****
528 ;
529 ;
530 ; ROUTINE XCMD
531 ;
532 ; THIS ROUTINE IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
533 ; COMMAND.
534 ;
535 ; EXTERNAL REFERENCES:
536 ; -----
537 ;
538 ; CROUT ECHO ERROR GETCH GETCM
539 GETHX NMOUT RGADR REGDS
540 ;

```

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  11

541                      ;      REGISTERS AFFECTED:  ALL
542                      ;
543                      ;
544                      ;*****
545                      ;
546                      ;
547 0148 CD 93 02      C  XCM0: CALL  GETCH  ; GET REGISTER IDENTIFIER
548 0148 4F              MOV  C,A
549 014C CD 6C 02      C      CALL  ECHO   ; ECHO IT
550 014F 79              MOV  A,C
551 0150 FE 0D          CPI   CR
552 0152 C2 5B 01      C      JNZ   XCM05  ; BRANCH IF NOT A CR
553 0155 CD 42 03      C      CALL  REGDS  ; OTHERWISE, SHOW REG CONTENTS
554 0158 C3 3E 00      C      JMP   GETCM  ; AND RETURN TO COMMAND INTERPRETER
555 015B 4F              XCM05: MOV  C,A    ; GET REGISTER IDENTIFIER TO C
556 015C CD 73 03      C      CALL  RGADR  ; CONVERT INTO RTAB ADDR
557 015F C5              PUSH B
558 0160 E1              POP  H          ; PUT POINTER TO REGISTER ENTRY INTO HL
559 0161 0E 20          MVI  C, " "
560 0163 CD 6C 02      C      CALL  ECHO   ; ECHO SPACE TO USER
561 0166 79              MOV  A,C
562 0167 32 F9 3F      STA   TEMP      ; PUT SPACE INTO TEMP AS DELIMITER
563 016A 3A F9 3F      XCM10: LDA   TEMP  ; GET TERMINATOR
564 016D FE 20          CPI   " "      ; IS IT A SPACE?
565 016F CA 77 01      C      JZ     XCM15  ; YES - GO CHECK POINTER INTO TABLE
566 0172 FE 2C          CPI   ","      ; NO - WELL, IS IT A COMMA?
567 0174 C2 3E 00      C      JNZ   GETCM  ; NO - THEN IT MUST BE A CR
568 0177 7E              XCM15: MOV  A,M
569 0178 B7              ORA   A          ; SET FLAGS
570 0179 C2 02 01      C      JNZ   XCM18  ; BRANCH IF NOT AT END OF TABLE
571 017C CD 66 02      C      CALL  CROUT  ; NOT AT END - PRINT CR,LF
572 017F C3 3E 00      C      JMP   GETCM  ; AND EXIT
573 0182 E5              XCM18: PUSH H          ; PLACE POINTER ON STACK
574 0183 5E              MOV  E,M
575 0184 16 3F          MVI  D, HIGH DATA ; FETCH ADDR OF SAVE LOC FROM TABLE
576 0186 23              INX   H
577 0187 46              MOV  B,M          ; FETCH LENGTH FLAG FROM TABLE
578 0188 D5              PUSH D          ; SAVE ADDR OF SAVE LOC
579 0189 D5              PUSH D
580 018A E1              POP  H          ; MOVE ADDR TO HL
581 018B C5              PUSH B          ; KEEP LENGTH FLAG
582 018C 7E              MOV  A,M          ; GET 8 BITS OF REG FROM SAVE LOC
583 018D CD 1F 03      C      CALL  NMOUT  ; SHOW IT TO THE USER
584 0190 F1              POP  PSW        ; GET BACK THE LENGTH FLAG
585 0191 F5              PUSH PSW        ; SAVE IT AGAIN
586 0192 B7              ORA   A          ; SET FLAGS
587 0193 CA 9B 01      C      JZ     XCM20  ; IF AN 8 BIT REG, THEN WE ARE DONE
588 0196 2B              DCX   H          ; OTHERWISE GET 8 MORE BITS
589 0197 7E              MOV  A,M
590 0198 CD 1F 03      C      CALL  NMOUT  ; NOW SHOW THEM
591 019B 0E 2D          XCM20: MVI  C, "-"
592 019D CD 6C 02      C      CALL  ECHO   ; USE DASH AS SEPARATOR
593 01A0 CD 9A 02      C      CALL  GETHX  ; IS THERE A VALUE THERE?
594 01A3                FALSE XCM30 ; NO - GO CHECK NEXT REGISTER

```



```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 12

595 01A3 D2 8B 01      C +      JNC      XCM30
596 01A6 7A              MOV      A,D
597 01A7 32 F9 3F      STA      TEMP      ; YES - SAVE THE TERMINATOR FOR A WHILE
598 01AA F1              POP      PSW      ; GET BACK LENGTH FLAG
599 01AB E1              POP      H      ; STORE SAVE LOC ADDR IN HL
600 01AC B7              ORA      A      ; SET FLAGS
601 01AD CA B2 01      C      JZ      XCM25      ; IF AN 8 BIT REG, THEN BRANCH
602 01B0 70              MOV      M,B      ; SAVE UPPER 8 BITS
603 01B1 2B              DCX      H      ; POINT TO SAVE LOC FOR LOWER 8 BITS
604 01B2 71              XCM25: MOV      M,C      ; STORE LSB OF REG
605 01B3 11 03 00      XCM27: LXI      D,RTABS      ; SIZE OF ENTRY IN RTAB TABLE
606 01B6 E1              POP      H      ; GET POINTER INTO RTAB
607 01B7 19              DAD      D      ; ADD ENTRY SIZE TO POINTER
608 01B8 C3 6A 01      C      JMP      XCM10      ; DO NEXT REG
609 01BB 7A              XCM30: MOV      A,D      ; GET TERMINATOR
610 01BC 32 F9 3F      STA      TEMP      ; SAVE IN MEM
611 01BF D1              POP      D      ; CLEAR STACK OF LENGTH FLAG AND ADDR
612 01C0 D1              POP      D      ; /OF SAVE LOC
613 01C1 C3 B3 01      C      JMP      XCM27      ; INC RTAB POINTER
614
615
616
617
618
619
620
621
622
623
624
625
626 01C4 0E 03          ZCMD: MVI      C,3      ;LOAD C FOR 3 CHARACTERS
627 01C6 CD CF 02      C      CALL     GETNM      ;GET 3 INPUT VALUES
628 01C9 D1              POP      D      ;DE=CHARACTER TO BE COPIED IN E,D FILLER
629 01CA C1              POP      B      ;END ADDRESS TO BE COPIED THROUGH
630 01CB E1              POP      H      ;HL=STARTING ADDRESS OF RAM WHERE E IS WRITTEN
631 01CC CD D2 01      C      CALL     ZERO      ;CALL ZERO ROUTINE
632 01CF C3 3E 00      C      JMP      GETCM      ;JUMP BACK TO GET NEXT COMMAND WHEN DONE
633 01D2 70              ZERO: MOV      A,E      ;DATA TO BE WRITTEN MOVED INTO RAM
634 01D3 77              MOV      M,A      ;WRITE DATA IN ACC INTO RAM ADDR BY HL
635 01D4 23              INX      H      ;INCREMENT HL FOR NEXT ADDRESS
636 01D5 79              MOV      A,C      ;SET UP TEST TO DETERMINE IF HL.GT.BC
637 01D6 95              SUB      L      ;SUBTRACT HL FROM BC IF NEG.GET A BORROW
638 01D7 70              MOV      A,B      ;WHEN HL .GT. BC ,CARRY IS SET
639 01D8 9C              SBB      H      ;THEN RETURN TO CALLING PROGRAM
640 01D9 D8              RC      ;RETURN WHEN DONE
641 01DA C3 D2 01      C      JMP      ZERO      ;JUMP BACK,WE WERE NOT DONE
642
643
644
645
646
647
648
ROUTINE CCMO
THIS ROUTINE COMPARES DATA FROM ADDR1 THROUGH ADDR2 TO

```

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 13

649          ;          DATA STARTING AT ADDR3.ANY DIFFERENCES ARE PRINTED ON
650          ;          THE TERMINAL
651          ;
652          ;*****
653          ;
654 01DD 0E 03      CCMD: MVI    C,3      ;LOAD C REGISTER TO GET 3 NUMBERS FROM INPUT
655 01DF CD CF 02    C      CALL    GETNM   ;3 NUMBERS WILL BE ON STACK
656 01E2 D1          POP     D          ;DE CONTAINS THE VALUES TO BE COMPARED TO
657 01E3 C1          POP     B          ;VALUES FROM ADDR HL TO BC WILL BE COMPARED
658 01E4 E1          POP     H          ;TO VALUES STARTING AT ADDR DE.
659 01E5 CD EB 01    C      CALL    COMPAR  ;COMPARE ROUTINE IS CALLED
660 01E8 C3 3E 00    C      JMP     GETCM   ;JUMP BACK TO GET NEXT COMMAND WHEN DONE
661 01EB 1A          COMPAR:LDAX  D      ;LOAD ACCUMULATOR WITH DATA ADDR BY DE
662 01EC 0E          CMP     M          ;COMPARE TO DATA ADDR BY HL
663 01ED CA 2B 02    C      JZ      DATCK   ;JUMP TO DATA CHECKS IF DE DATA=HL DATA
664 01F0 C5          PUSH   B          ;DATA DID NOT AGREE,SAVE BC,PRINT DATA
665 01F1 7C          MOV     A,H        ;LOAD H IN ACC
666 01F2 CD 1F 03    C      CALL    NMOUT   ;PRINT H REG VAL
667 01F5 7D          MOV     A,L        ;LOAD L IN ACC
668 01F6 CD 1F 03    C      CALL    NMOUT   ;PRINT L VALUE
669 01F9 0E 3D          MVI    C,03DH   ;LOAD ASCII "*" INTO C
670 01FB CD 5B 02    C      CALL    CO      ;PRINT IT
671 01FE 0E 20          MVI    C,020H   ;LOAD BLANK INTO C
672 0200 CD 5B 02    C      CALL    CO      ;PRINT A SPACE
673 0203 7E          MOV     A,M        ;LOAD DATA AT HL ADDR INTO A
674 0204 CD 1F 03    C      CALL    NMOUT   ;PRINT DATA AT HL ADDR
675 0207 0E 20          MVI    C,020H   ;PRINT 2 SPACES
676 0209 CD 5B 02    C      CALL    CO      ;
677 020C 0E 20          MVI    C,020H   ;
678 020E CD 5B 02    C      CALL    CO      ;
679 0211 7A          MOV     A,D        ;LOAD D ADDR INTO A
680 0212 CD 1F 03    C      CALL    NMOUT   ;
681 0215 7B          MOV     A,E        ;PRINT DE ADDRESS
682 0216 CD 1F 03    C      CALL    NMOUT   ;
683 0219 0E 3D          MVI    C,03DH   ;LOAD ASCII REP OF "*" INTO C
684 021B CD 5B 02    C      CALL    CO      ;PRINT IT
685 021E 0E 20          MVI    C,020H   ;LOAD IN SPACE
686 0220 CD 5B 02    C      CALL    CO      ;PRINT IT
687 0223 1A          LDAX  D          ;LOAD DATA ADDRESSED BY DE INTO ACC
688 0224 CD 1F 03    C      CALL    NMOUT   ;PRINT DATA ADDRESSED BY DE
689 0227 CD 66 02    C      CALL    CROUT   ;CARRIAGE RETURN,LINEFEED
690 022A C1          POP     B          ;BC RESTORED
691 022B 23          DATCK: INX  H      ;INCREMENT HL TO NEXT DATA BYTE
692 022C 13          INX  D          ;INCREMENT DE TO NEXT DATA BYTE
693 022D 79          MOV     A,C        ;SET UP TEST TO DETERMINE IF HL.GT.BC
694 022E 95          SUB     L          ;SUBTRACT HL FROM BC, WE WILL GET A
695 022F 7B          MOV     A,B        ;A "BORROW" WHEN HL.GT.BC(CARRY IS SET)
696 0230 9C          SBB  H          ;
697 0231 D8          RC      ;RETURN TO CALLING SUB WHEN DONE
698 0232 C3 EB 01    C      JMP     COMPAR ;JUMP BACK IF NOT DONE
699          ;
700          ;*****
701          ;*****
702          ;

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 14

```

703      ;
704      ;           UTILITY ROUTINES
705      ;
706      ;
707      ;*****
708      ;*****
709      ;
710      ;
711      ;           ROUTINE BREAK
712      ;
713      ;           THIS ROUTINE CHECKS FOR A BREAK CHAR FROM THE CONSOLE.
714      ;           IF THERE IS NO PENDING CHAR OR IF IT IS NOT THE BREAK
715      ;           CHAR (USUALLY AN ESC), A FAILURE RETURN IS TAKEN (CARRY=0).
716      ;           THE CHAR IS LOST. IF IT IS THE BREAK CHAR, A SUCCESS RETURN
717      ;           IS TAKEN (CARRY=1).
718      ;
719      ;           OUTPUT:                CARRY
720      ;                               BIT
721      ;                               -----
722      ;                               SUCCESS      1
723      ;
724      ;                               FAILURE      0
725      ;
726      ;           DESTROYS:  A,FLAGS
727      ;
728      ;
729      ;*****
730      ;
731      ;
732      0235 DB FB      BREAK: IN      CONST ; GET TERMINAL STATUS
733      0237 E6 02      ANI      RBR      ; IS A CHAR THERE?
734      0239 CA 90 02    C      JZ      FRET ; NO - FAILURE RETURN
735      023C DB FA      IN      CNIN     ; YES - GET THE CHAR
736      023E E6 7F      ANI      PRTY0    ; GET RID OF THE PARITY BIT
737      0240 FE 1B      CPI      BRCHR    ; IS IT A BREAK CHAR?
738      0242 CA 9F 03    C      JZ      SRET ; YES - TAKE A SUCCESS RET
739      0245 C3 90 02    C      JMP      FRET ; NO - TAKE A FAILURE RET
740      ;
741      ;
742      ;*****
743      ;
744      ;
745      ;           ROUTINE CI
746      ;
747      ;           THIS ROUTINE WAITS FOR A CHAR TO BE ENTERED AT THE
748      ;           TERMINAL AND THEN PLACES THE CHAR INTO THE A REGISTER,
749      ;           ALLOWING THE CALLING ROUTINE ACCESS TO IT.
750      ;
751      ;           OUTPUT:      THE RECEIVED CHAR IN REG A
752      ;
753      ;           REGISTERS AFFECTED:  A,FLAGS
754      ;
755      ;
756      ;*****

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 15

```

757 ;
758 ;
759 0248 DB FB      CI:  IN      CONST ; GET STATUS OF CONSOLE
760 024A E6 02      ANI      RBR      ; CHECK FOR RECEIVER READY
761 024C CA 48 02    JZ       CI       ; NOT YET, SO WAIT
762 024F DB FA      IN       CNIN     ; OK - NOW GET CHAR
763 0251 C9         RET       ; RETURN TO CALLER
764 ;
765 ;
766 ;
767 ;
768 ;
769 ;
770 ;
771 ;
772 ;
773 ;
774 ;
775 ;
776 ;
777 ;
778 ;
779 ;
780 ;
781 ;
782 ;
783 ;
784 ;
785 0252 79         CNVBN: MOV      A,C
786 0253 D6 30      SUI      '0'      ; SUBTRACT A HEX 30H FROM ASCII INPUT CODE
787 0255 FE 0A      CPI      10       ; BETWEEN 0 AND 9?
788 0257 F8         RM        ; YES - ALL DONE
789 0258 D6 07      SUI      7        ; NO - BETWEEN 17 AND 23 DECIMAL
790 025A C9         RET       ; SUBTRACT OFFSET OF 7 (DECIMAL) AND RETURN
791 ;
792 ;
793 ;
794 ;
795 ;
796 ;
797 ;
798 ;
799 ;
800 ;
801 ;
802 ;
803 ;
804 ;
805 ;
806 ;
807 ;
808 ;
809 ;
810 ;

```

ROUTINE CNVBN

THIS ROUTINE CONVERTS THE ASCII CODE OF A HEX DIGIT TO THE CORRESPONDING BINARY VALUE. NO CHECK FOR VALID INPUT IS PERFORMED.

INPUT: REGISTER C - ASCII CODE FOR ONE HEX DIGIT

OUTPUT: REGISTER A - BINARY VALUE OF THE INPUT DIGIT

REGISTERS AFFECTED: A, FLAGS

ROUTINE CO

THIS ROUTINE WAITS UNTIL THE TERMINAL IS READY AND THEN SENDS THE INPUT ASCII CODE TO IT.

INPUT: REGISTER C - ASCII CODE OF DESIRED CHAR TO BE PRINTED

OUTPUT: REGISTER C - CODE PRINTED AT TERMINAL

REGISTERS AFFECTED: A, FLAGS

```

ERR_LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 16

011 0258 D8 FB      CO:  IN      CONST ; GET TERMINAL'S STATUS
012 025D E6 01      ANI      TRDY ; SEE IF TRANSMITTER IS READY
013 025F CA 58 02    C      JZ      CO ; NO - WAIT
014 0262 79          MOV      A,C ; YES - MOVE CODE TO THE ACCUMULATOR FOR OUTPUT
015 0263 D3 FA      OUT      CNOUT ; SEND IT TO THE CONSOLE
016 0265 C9          RET
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034 0266 0E 0D      CROUT: MVI      C,CR
035 0268 CD 6C 02    C      CALL     ECHO
036 026B C9          RET
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061 026C 41          ECHO:  MOV      B,C ; SAVE ARG
062 026D 3E 18      MVI      A,ESC ; LOAD ASCII CODE OF <ESC>
063 026F 88          CMP      B ; SEE IF ARG IS AN <ESC>
064 0270 C2 75 02    C      JNZ      ECH05 ; NO - THEN BRANCH

```

ROUTINE CROUT

THIS ROUTINE PRINTS A <CR>,<LF> ON THE TERMINAL

EXTERNAL REFERENCES: ECHO

REGISTERS AFFECTED: A,B,C,FLAGS

ROUTINE ECHO

THIS ROUTINE TAKES A SINGLE CHAR AS INPUT AND PRINTS IT ON TH
USER TERMINAL. A <CR> IS ECHOED AS A <CR>,<LF> AND AN <ESC>
IS ECHOED AS \$.

INPUT: REGISTER C - ASCII CODE OF CHAR TO ECHO

OUTPUT: REGISTER C - CODE ECHOED TO TERMINAL

EXTERNAL REFERENCES: CO

REGISTERS AFFECTED: A,B,FLAGS

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE 17

      865 0273 0E 24      MVI    C,'$'    ; YES - THEN ECHO A $
      866 0275 CD 5B 02    C ECH05: CALL  CO      ; DO OUTPUT THROUGH MONITOR
      867 0278 3E 0D      MVI    A,CR
      868 027A B8          CMP    B          ; WAS CHAR A <CR>?
      869 027B C2 83 02    C JNZ    ECH10    ; NO - NO SPECIAL ACTION NEEDED
      870 027E 0E 0A      MVI    C,LF      ; YES - ADD A <LF>
      871 0280 CD 5B 02    C CALL    CO
      872 0283 48          ECH10: MOV   C,B      ; RESTORE ARG
      873 0284 C9          RET          ; RETURN TO CALLER
      874
      875
      876 ;*****
      877 ;
      878 ;
      879 ;
      880 ;
      881 ;
      882 ;
      883 ;
      884 ;
      885 ;
      886 ;
      887 ;
      888 ;
      889 ;
      890 ;
      891 0285 0E 2A      ERROR: MVI    C,'*'
      892 0287 CD 6C 02    C CALL    ECHO    ; PRINT A '*' ON THE TERMINAL
      893 028A CD 66 02    C CALL    CROUT   ; GO TO THE NEXT LINE
      894 028D C3 3E 00    C JMP     GETCM   ; RETURN TO COMMAND INTERPRETER
      895
      896 ;
      897 ;
      898 ;
      899 ;
      900 ;
      901 ;
      902 ;
      903 ;
      904 ;
      905 ;
      906 ;
      907 ;
      908 ;
      909 ;
      910 ;
      911 ;
      912 ;
      913 ;
      914 0290 37      FRET: STC          ; FIRST SET CARRY TRUE
      915 0291 3F      CMC          ; COMPLEMENT AND MAKE IT FALSE
      916 0292 C9      RET          ; RETURN TO CALLER
      917
      918

```

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  18

919      ;*****
920      ;
921      ;
922      ;      ROUTINE GETCH
923      ;
924      ;      THIS ROUTINE RETURNS THE NEXT CHAR IN THE INPUT STREAM TO
925      ;      THE CALLING ROUTINE.
926      ;
927      ;      OUTPUT:  REGISTER C - THE NEXT CHAR IN INPUT STREAM
928      ;
929      ;      EXTERNAL REFERENCES:  CI
930      ;
931      ;      REGISTERS AFFECTED:  A,C,FLAGS
932      ;
933      ;*****
934      ;
935      ;
936      ;
937      0293  CD 48 02      C  GETCH: CALL    CI      ; GET CHAR FORM TERMINAL
938      0296  E6 7F          ANI    PRTY0    ; TURN OFF PARITY BIT IF SET
939      0298  4F             MOV    C,A      ; PUT VALUE IN C REG FOR RETURN
940      0299  C9             RET          ; RETURN TO CALLER
941      ;
942      ;
943      ;*****
944      ;
945      ;
946      ;      ROUTINE GETHX
947      ;
948      ;      THIS ROUTINE TAKES THE LAST FOUR HEX DIGITS FROM THE INPUT
949      ;      STREAM AND CONVERTS THEIR VALUE TO A 16-BIT BINARY VALUE.
950      ;      A VALID DELIMITER TERMINATES THE INPUT NUMBER AND IS RETURNED
951      ;      WITH THE BINARY VALUE AS AN OUTPUT.  ANY ILLEGAL CHARS
952      ;      CREATE AN ERROR CONDITION.  IF THE FIRST (VALID) CHAR IS NOT
953      ;      A DELIMITER, THE ROUTINE SETS THE CARRY TO 1 AND RETURNS;
954      ;      OTHERWISE, THE CARRY IS SET TO 0 AND THE CONTENTS OF BC ARE
955      ;      UNDEFINED.
956      ;
957      ;      OUTPUT:
958      ;      -----
959      ;
960      ;      REGISTER B,C - 16 BIT BINARY INTEGER
961      ;
962      ;      REGISTER D - CHAR THAT TERMINATED THE INTEGER INPUT
963      ;
964      ;      FLAGS (CARRY) - 1 IF FIRST CHAR NOT A DELIMITER
965      ;                        0 IF IT IS (THEN B,C ARE UNDEFINED)
966      ;
967      ;      EXTERNAL REFERENCES:
968      ;      -----
969      ;
970      ;      CNVBN    ECHO    ERROR    GETCH    VALDG
971      ;      VALDL
972      ;

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 19

```

973          ;          REGISTERS AFFECTED: A,B,C,D,E,FLAGS
974          ;
975          ;
976          ;*****
977          ;
978          ;
979 029A E5      GETHX: PUSH    H          ; SAVE HL
980 029B 21 00 00 LXI      H,0          ; INIT RETURN VALUE
981 029E 1E 00   MVI      E,0          ; INIT DIGIT FLAG TO FALSE
982 02A0 CD 93 02 C      GHX05: CALL   GETCH    ; GET ONE CHAR
983 02A3 4F      MOV      C,A          ;
984 02A4 CD 6C 02 C      CALL   ELCHU    ; ECHO IT
985 02A7 CD E6 03 C      CALL   VALDL    ; SEE IF IT IS A DELIMITER
986 02AA         FALSE    GHX10        ; NO - BRANCH
987 02AA D2 B9 02 C +    JNC      GHX10
988 02AD 51      MOV      D,C          ; YES - FINISHED, MUST RETURN DELIMITER
989 02AE E5      PUSH    H
990 02AF C1      POP     B          ; STORE VALUE IN BC
991 02B0 E1      POP     H          ; RESTORE HL
992 02B1 7B      MOV     A,E          ; GET FLAG
993 02B2 87      ORA     A          ; SET FLAGS
994 02B3 C2 9F 03 C      JNZ      SRET    ; NONZERO FLAG - NUMBER FOUND
995 02B6 CA 90 02 C      JZ       FRET    ; ZERO FLAG - DELIMITER WAS 1ST CHAR
996 02B9 CD CB 03 C      GHX10: CALL   VALDG    ; IF NOT A DELIMITER, SEE IF A HEX DIGIT
997 02BC         FALSE    ERROR        ; NO - ERROR
998 02BC D2 85 02 C +    JNC      ERROR
999 02BF CD 52 02 C      CALL   CNVBN    ; CONVERT TO ITS BINARY VALUE
1000 02C2 1E FF   MVI      E,0FFH      ; SET DIGIT FLAG TO TRUE
1001 02C4 29      DAD     H          ; *2
1002 02C5 29      DAD     H          ; *4
1003 02C6 29      DAD     H          ; *8
1004 02C7 29      DAD     H          ; *16
1005 02C8 06 00   MVI      B,0          ; CLEAR THE HIGH 8 BITS OF BC
1006 02CA 4F      MOV     C,A          ; GET BINARY VALUE OF CHAR INTO C
1007 02CB 09      DAD     B          ; ADD THIS VALUE TO PARTIAL RESULT
1008 02CC C3 A0 02 C      JMP      GHX05    ; NOW GET THE NEXT CHAR
1009          ;
1010          ;
1011          ;*****
1012          ;
1013          ;
1014          ;          ROUTINE GETNM
1015          ;
1016          ;          THIS ROUTINE FINDS A SPECIFIED NUMBER OF INPUT NUMBERS
1017          ;          (BETWEEN 1 AND 3) AND RETURNS THEIR VALUES ON THE STACK.
1018          ;
1019          ;          IF TWO OR MORE ARE REQUESTED, THEN THE FIRST MUST BE LESS
1020          ;          THAN OR EQUAL TO THE SECOND. OTHERWISE THE FIRST AND SECOND
1021          ;          NUMBERS WILL BE SET EQUAL. THE LAST NUMBER MUST BE FOLLOWED
1022          ;          BY A <CR> OR AN ERROR CONDITION WILL OCCUR.
1023          ;
1024          ;          INPUT:  REG C - NUMBER OF INPUT VALUES TO READ
1025          ;
1026          ;          OUTPUT: TOP OF STACK - VALUES OF INPUTS (LAST ON TOP)

```


ERR LINE	ADDR	B1	B2	B3	B4	NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979		PAGE 28
1027								
1028							EXTERNAL REFERENCES:	
1029							-----	
1030								
1031							ERROR HILO GETHX	
1032								
1033							REGISTERS AFFECTED: ALL	
1034								
1035								
1036								
1037								
1038	02CF	2E	03			GETNM:	MVI L,3	; PUT MAX ARGUMENT COUNT INTO L
1039	02D1	79					MOV A,C	; GET THE ACTUAL ARG COUNT
1040	02D2	E6	03				ANI 3	; FORCE TO MAX OF 3
1041	02D4	C8					RZ	; IF 0, DON'T DO ANYTHING
1042	02D5	67					MOV H,A	; ELSE, PUT THE ACTUAL COUNT IN H
1043	02D6	CD	9A	02	C	GNM05:	CALL GETHX	; GET A NUMBER FROM THE INPUT STREAM
1044	02D9						FALSE	ERROR ; ERROR IF NOT THERE (TOO FEW INPUTS)
1045	02D9	D2	85	02	C +		JNC ERROR	
1046	02DC	C5					PUSH B	; ELSE, SAVE NUMBER ON THE STACK
1047	02DD	2D					DCR L	; DECREMENT ACTUAL ARG COUNT
1048	02DE	25					DCR H	; DECREMENT ACTUAL ARG COUNT
1049	02DF	CA	EB	02	C		JZ GNM10	; BRANCH IF NO MORE INPUTS ARE NEEDED
1050	02E2	7A					MOV A,D	; ELSE, GET NUMBER TERMINATOR TO A
1051	02E3	FE	0D				CPI CR	; IS IT A <CR>?
1052	02E5	CA	85	02	C		JZ ERROR	; YES - ERROR (TOO FEW NUMBERS)
1053	02E8	C3	D6	02	C		JMP GNM05	; NO - DO ANOTHER NUMBER
1054	02E8	7A				GNM10:	MOV A,D	; WHEN DOWN TO ZERO, CHECK LAST CHAR
1055	02EC	FE	0D				CPI CR	
1056	02EE	C2	85	02	C		JNZ ERROR	; ERROR IF IT ISN'T A <CR>
1057	02F1	01	FF	FF			LXI B,0FFFFH	; HL GETS THE LARGEST VALUE
1058	02F4	7D					MOV A,L	; GET WHAT'S LEFT OF THE MAX ARG COUNT
1059	02F5	B7					ORA A	; CHECK IF 0
1060	02F6	CA	FE	02	C		JZ GNM20	; YES - 3 NUMBERS WERE INPUT
1061	02F9	C5				GNM15:	PUSH B	; NO - FILL UNUSED ARGS WITH 0FFFFH
1062	02FA	2D					DCR L	
1063	02FB	C2	F9	02	C		JNZ GNM15	
1064	02FE	C1				GNM20:	POP B	; NOW GET THE THREE ARGS OUT
1065	02FF	D1					POP D	
1066	0300	E1					POP H	
1067	0301	CD	14	03	C		CALL HILO	; SEE IF FIRST VAL >= THE SECOND
1068	0304						FALSE	GNM25 ; NO - TAKE A BRANCH
1069	0304	D2	09	03	C +		JNC GNM25	
1070	0307	54					MOV D,H	
1071	0308	5D					MOV E,L	; YES - SET SECOND EQUAL TO FIRST
1072	0309	E3				GNM25:	XTHL	; PUT THE FIRST ON THE STACK AND GET RETURN ADDR
1073	030A	D5					PUSH D	; PUT THE SECOND ON STACK
1074	030B	C5					PUSH B	; PUT THIRD ON THE STACK
1075	030C	E5					PUSH H	; PUT RET ADDR ONTO STACK
1076	030D	3D				GNM30:	DCR A	; DECREMENT RESIDUAL COUNT
1077	030E	F8					RM	; IF IT IS NEGATIVE, THEN CORRECT
1078								; NUMBERS ARE ON THE STACK
1079	030F	E1					POP H	; OTHERWISE, GET RETURN ADDR
1080	0310	E3					XTHL	; REPLACE TOP RESULT WITH RET ADDR

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  21

1081 0311 C3 0D 03      C      JMP      GNM30      ; TRY AGAIN
1082
1083
1084 ;*****
1085
1086
1087 ;
1088 ;
1089 ;
1090 ;
1091 ;
1092 ;
1093 ;
1094 ;
1095 ;
1096 ;
1097 ;
1098 ;
1099 ;
1100 ;
1101 ;
1102 ;
1103 ;
1104 ;
1105 ;*****
1106 ;
1107 ;
1108 0314 C5      HILO: PUSH      B      ; SAVE BC
1109 0315 47      MOV      B,A      ; SAVE A IN B REG
1110 0316 18      DCX      D      ; DE DECREMENTED SO THAT WE WILL GET CARRY
1111 0317 78      MOV      A,E      ; WHEN HL.GE.DE
1112 0318 95      SUB      L      ; SUBTRACT HL FROM DE-GET A BORROW WHEN
1113 0319 7A      MOV      A,D      ; WHEN HL.GT.DE-1(CARRY IS SET)
1114 031A 9C      SBB      H
1115 031B 13      INX      D      ; DOESN'T WORK IF DE=0,RESTORE DE
1116 031C 78      MOV      A,B      ; RESTORE ACCUMULATOR
1117 031D C1      POP      B      ; RESTORE BC
1118 031E C9      RET      ; RETURN TO CALLING ROUTINE
1119
1120 ;
1121 ;*****
1122 ;
1123 ;
1124 ;
1125 ;
1126 ;
1127 ;
1128 ;
1129 ;
1130 ;
1131 ;
1132 ;
1133 ;
1134 ;

```

ROUTINE HILO

THIS ROUTINE COMPARES TWO UNSIGNED 16-BIT INTEGERS IN HL AND DE. THE CARRY FLAG IS SET ACCORDING TO THE COMPARISON.

INPUT: DE - 16-BIT INTEGER
HL - 16-BIT INTEGER

OUTPUT: CASE CARRY

HL<DE 0
HL>=DE 1

REGISTERS AFFECTED: FLAGS

ROUTINE NMOUT

THIS ROUTINE PRINTS THE CONTENTS OF REGISTER A IN HEX ON THE TERMINAL.

INPUT: REGISTER A - 8-BIT INTEGER

EXTERNAL REFERENCES:

ECHO PRVAL

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  22

1135      ;
1136      ;   REGISTERS AFFECTED:  A,B,C,FLAGS
1137      ;
1138      ;
1139      ; ~~~~~
1140      ;
1141      ;
1142 031F  E5      NMOUT: PUSH    H      ; SAVE HL - DESTROYED BY PRVAL
1143 0320  F5      PUSH    PSW     ; SAVE ARG
1144 0321  0F      RRC
1145 0322  0F      RRC
1146 0323  0F      RRC
1147 0324  0F      RRC      ; MOVE HI 4 BITS TO LO 4 BITS
1148 0325  E6 0F  ANI      HCHAR   ; MASK OUT HI 4 - ONLY WANT ONE HEX CHAR
1149 0327  4F      MOV      C,A
1150 0328  CD 3A 03  CALL    PRVAL   ; CONVERT LOWER 4 BITS TO ASCII
1151 0328  CD 6C 02  CALL    ECHO   ; SEND TO TERMINAL
1152 032E  F1      POP      PSW     ; GET BACK ARG
1153 032F  E6 0F  ANI      HCHAR   ; MASK OUT UPPER 4 BITS - JUST WANT 1 HEX CHAR
1154 0331  4F      MOV      C,A
1155 0332  CD 3A 03  CALL    PRVAL
1156 0335  CD 6C 02  CALL    ECHO
1157 0338  E1      POP      H      ; RESTORE ORIGINAL HL
1158 0339  C9      RET          ; RETURN TO CALLER
1159      ;
1160      ; ~~~~~
1161      ;
1162      ;
1163      ;
1164      ;   ROUTINE PRVAL
1165      ;
1166      ;   THIS ROUTINE CONVERTS A BINARY NUMBER BETWEEN 0H AND 0FH,
1167      ;   INCLUSIVE, INTO THE CORRESPONDING ASCII CHAR '0' - '9' OR
1168      ;   'A' - 'F'. NO CHECK IS MADE OF THE VALIDITY OF THE INPUT.
1169      ;
1170      ;   INPUT:  REGISTER C - INTEGER N, 0<=N<=0FH
1171      ;
1172      ;   OUTPUT: REGISTER C - CORRESPONDING ASCII CODE
1173      ;
1174      ;   REGISTERS AFFECTED:  B,C,H,L,FLAGS
1175      ;
1176      ; ~~~~~
1177      ;
1178      ;
1179      ;
1180 033A  21 20 04  C  PRVAL: LXI    H,DIGTB ; ADDR OF TABLE
1181 033D  06 00      MVI      B,0      ; CLEAR HI 8 BITS OF BC
1182 033F  09      DAD      B      ; ADD DIGIT VAL TO HL ADDR
1183 0340  4E      MOV      C,M      ; FETCH ASCII CODE FROM MEM
1184 0341  C9      RET          ; RETURN TO CALLER
1185      ;
1186      ; ~~~~~
1187      ;
1188      ;

```

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  23

1189      ;
1190      ;
1191      ;
1192      ;
1193      ;
1194      ;
1195      ;
1196      ;
1197      ;
1198      ;
1199      ;
1200      ;
1201      ;
1202      ;
1203      ;
1204      ;
1205      ;
1206      ;
1207      ;
1208  0342  21 30 04      C  REGDS: LXI      H,RTAB ; LOAD HL WITH ADDR OF START OF TABLE
1209  0345  4E              REG05: MOV      C,M   ; GET PRINT SYMBOL OF REG
1210  0346  79              MOV      A,C
1211  0347  B7              ORA      A       ; TEST FOR 0 - END OF TABLE
1212  0348  C2 4F 03      C      JNZ      REG10 ; NO - THEN BRANCH
1213  0348  CD 66 02      C      CALL     CROUT ; YES - SEND <CR>,<LF>
1214  034E  C9              RET          ; RETURN TO CALLER
1215  034F  CD 6C 02      C  REG10: CALL    ECHO ; ECHO THE CHAR
1216  0352  0E 3D          MVI      C,"-"
1217  0354  CD 6C 02      C      CALL     ECHO ; OUTPUT EQUALS SIGN
1218  0357  23              INX      H       ; POINT TO START OF SAVE LOC ADDR
1219  0358  5E              MOV      E,M   ; GET LSB OF SAVE LOC ADDR TO E
1220  0359  16 3F          MVI      D, HIGH DATA ; PUT MSB OF SAVE LOC ADDR INTO D
1221  0358  23              INX      H       ; POINT TO LENGTH FLAG
1222  035C  1A              LDAX     D       ; GET CONTENTS OF SAVE ADDR
1223  035D  CD 1F 03      C      CALL     NMOUT ; PRINT THEM ON THE TERMINAL
1224  0360  7E              MOV      A,M   ; GET LENGTH FLAG
1225  0361  B7              ORA      A       ; SET SIGN FLAG
1226  0362  CA 6A 03      C      JZ       REG15 ; IF 0, REGISTER IS 8 BITS
1227  0365  1B              DCX      D       ; OTHERWISE, 16-BIT REG SO MORE TO DISPLAY
1228  0366  1A              LDAX     D       ; GET LO 8 BITS
1229  0367  CD 1F 03      C      CALL     NMOUT ; PRINT THEM
1230  036A  0E 20          REG15: MVI      C," "
1231  036C  CD 6C 02      C      CALL     ECHO
1232  036F  23              INX      H       ; POINT TO START OF NEXT TABLE ENTRY
1233  0370  C3 45 03      C      JMP      REG05 ; DO NEXT REG
1234      ;
1235      ;
1236      ;
1237      ;
1238      ;
1239      ;
1240      ;
1241      ;
1242      ;

```

ROUTINE REGDS

THIS ROUTINE SHOWS THE REGISTER SAVE LOC'S ON THE TERMINAL IN FORMATTED FORM. THIS ROUTINE IS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS THE REG'S PRINT SYMBOL, SAVE LOC ADDR, AND LENGTH (8 OR 16 BITS).

EXTERNAL REFERENCES:

CROUT ECHO ERROR NMOUT

REGISTERS AFFECTED: ALL

ROUTINE RGADR

THIS ROUTINE TAKES ONE CHAR, DENOTING A REGISTER, AND SEARCHES THE RTAB TABLE FOR A MATCH. IF FOUND, THE ROUTINE

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  24

1243      ;      RETURNS THE ADDR OF THE ADDR OF THE SAVE LOC OF THE REG.
1244      ;      THIS ADDR POINTS INTO RTAB.  IF THERE IS NO MATCH, THEN THE
1245      ;      REGISTER IDENTIFIER IS ILLEGAL AND AN ERROR CONDITION OCCURS.
1246      ;
1247      ;      INPUT:      REGISTER C - ASCII CODE OF REG SYMBOL
1248      ;
1249      ;      OUTPUT:     REGISTERS B,C - ADDR OF ENTRY IN RTAB FOR REG DESI
1250      ;
1251      ;      EXTERNAL REFERENCES:
1252      ;      -----
1253      ;
1254      ;      ERROR
1255      ;
1256      ;      REGISTERS AFFECTED:  ALL
1257      ;
1258      ;
1259      ; *****
1260      ;
1261      ;
1262 0373 21 30 04      C  RGADR: LXI      H,RTAB ; HL GETS ADDR OF START OF TABLE
1263 0376 11 03 00      LXI      D,RTABS ; DE GETS SIZE OF AN ENTRY
1264 0379 7E           RGA05: MOV      A,M ; GET REGISTER IDENTIFIER
1265 037A 87           ORA      A ; CHECK FOR TABLE END (IDENTIFIER IS 0)
1266 037B CA 05 02      C      JZ      ERROR ; IF AT END OF TABLE, ARG IS ILLEGAL
1267 037E 09           CMP      C ; ELSE, COMPARE TABLE ENTRY AND ARG
1268 037F CA 06 03      C      JZ      RGA10 ; EQUAL - REG IS FOUND
1269 0382 19           DAD      D ; NOT EQUAL - INC POINTER AND TRY AGAIN
1270 0383 C3 79 03      C      JMP      RGA05
1271 0386 23           RGA10: INX      H ; IF A MATCH, INC TABLE OPOINTER TO SAVE LOC ADDR
1272 0387 44           MOV      B,H
1273 0388 4D           MOV      C,L ; RETURN THIS VALUE
1274 0389 C9           RET      ; RETURN TO CALLER
1275      ;
1276      ;
1277      ; *****
1278      ;
1279      ;
1280      ;      ROUTINE RSTTF
1281      ;
1282      ;      THIS ROUTINE RESTORES ALL REGS, ALL FLIP/FLOPS, THE SP AND
1283      ;      THE PC FROM THE SAVE LOCS IN MEMORY.  THE ROUTINE RETURNS
1284      ;      CONTROL TO THE NFM PC LOC WITH ALL INTERRUPTS ENABLED.
1285      ;
1286      ;      REGISTERS AFFECTED:  ALL
1287      ;
1288      ;
1289      ; *****
1290      ;
1291      ;
1292      ;
1293 038A F3           RSTTF: DI      ; DISABLE INTERRUPTS WHILE RESTORING THINGS
1294 038B 21 ED 3F      LXI      H,MSTAK ; SET MONITOR SP TO START OF STACK
1295 038E F9           SPHL
1296 038F D1           POP      D ; START ALSO END OF REG SAVE AREA

```

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 25

1297 0390 C1
 1298 0391 F1
 1299 0392 2A F7 3F
 1300 0395 F9
 1301 0396 2A F5 3F
 1302 0399 E5
 1303 039A 2A F3 3F
 1304 039D FB
 1305 039E C9

POP B
 POP PSW
 LHL D SSAVE ; RESTORE USER STACK POINTER.
 SPHL
 LHL D PSAVE
 PUSH H ; PUT USER RET ADDR ON USER STACK
 LHL D LSAVE ; RESTORE HL REGS
 EI ; ENABLE INTERRUPTS NOW
 RET ; JUMP TO RESTORED PC LOC

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324 039F 37

1325 03A0 C9

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

ROUTINE SRET

THIS ROUTINE IS CALLED BY OTHER ROUTINES WHICH WANT TO
 RETURN SUCCESS. IT SETS THE CARRY TRUE (1).

OUTPUT: CARRY = 1

REGISTERS AFFECTED: FLAGS (CARRY ONLY)

SRET: STC ; SET CARRY TRUE
 RET ; RETURN TO CALLER

ROUTINE STHF0

THIS ROUTINE CHECKS THE HALF-BYTE FLAG IN TEMP. TO SEE IF
 IT IS SET TO LOWER. IF SO, STHF0 STORES A 0 TO PAD OUT
 THE LOWER HALF OF THE ADDRESSED BYTE; OTHERWISE, IT DOES
 NOTHING.

INPUT: REGISTERS DE - 16-BIT ADDR OF BYTE TO BE STORED IN

EXTERNAL REFERENCES:

STHLF

REGISTERS AFFECTED: A,B,C,H,L,FLAGS

ERR LINE	ADDR	B1 B2 B3 B4	NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979	PAGE 26
1351	03A1	3A F9 3F	STHF0: LDA TEMP ; GET HALF BYTE FLAG	
1352	03A4	B7	ORA A ; SET FLAGS	
1353	03A5	C0	RNZ ; IF SET TO UPPER, DON'T DO ANYTHING	
1354	03A6	0E 00	MVI C,0 ; ELSE, WANT TO STORE THE VALUE 0	
1355	03A8	CD AC 03	CALL STHLF ; DO IT	
1356	03AB	C9	RET ; RETURN TO CALLER	
1357				
1358				
1359				
1360				
1361				
1362				
1363			ROUTINE STHLF	
1364				
1365			THIS ROUTINE TAKES THE 4-BIT VALUE IN C AND STORES IT IN	
1366			HALF OF THE BYTE ADDRESSED BY REGS DE. THE HALF-BYTE USED	
1367			(UPPER OR LOWER) IS DENOTED BY THE FLAG IN TEMP. STHLF	
1368			ASSUMES THAT THIS FLAG HAS ALREADY BEEN SET, USUALLY BY	
1369			ICMD.	
1370			INPUT: REGISTER C - 4-BIT VALUE TO STORE IN HALF-BYTE	
1371			REGISTERS DE - 16-BIT ADDR OF BYTE TO BE STORED IN	
1372				
1373			REGISTERS AFFECTED: A,B,C,H,L,FLAGS	
1374				
1375				
1376				
1377				
1378				
1379	03AC	D5	STHLF: PUSH D	
1380	03AD	E1	POP H ; MOVE ADDR OF BYTE INTO HL	
1381	03AE	79	MOV A,C ; GET VAL	
1382	03AF	E6 0F	ANI 0FH ; FORCE TO 4-BIT LENGTH	
1383	03B1	4F	MOV C,A ; PUT VAL BACK	
1384	03B2	3A F9 3F	LDA TEMP ; GET HALF BYTE FLAG	
1385	03B5	B7	ORA A ; CHECK FOR LOWER HALF	
1386	03B6	C2 0F 03	JNZ STH05 ; BRANCH IF NOT	
1387	03B9	7E	MOV A,M ; ELSE, GET BYTE	
1388	03BA	E6 F0	ANI 0F0H ; CLEAR LOWER 4 BITS	
1389	03BC	B1	ORA C ; OR IN VAL	
1390	03BD	77	MOV M,A ; PUT BYTE BACK	
1391	03BE	C9	RET ; RETURN TO CALLER	
1392	03BF	7E	STH05: MOV A,M ; IF UPPER HALF, GET BYTE	
1393	03C0	E6 0F	ANI 0FH ; CLEAR UPPER 4 BITS	
1394	03C2	47	MOV B,A ; SAVE BYTE IN B	
1395	03C3	79	MOV A,C ; GET VALUE	
1396	03C4	0F	RRC	
1397	03C5	0F	RRC	
1398	03C6	0F	RRC	
1399	03C7	0F	RRC ; ALIGN TO UPPER 4 BITS	
1400	03C8	B0	ORA B ; OR IN ORIGINAL LOWER 4 BITS	
1401	03C9	77	MOV M,A ; PUT NEW CONFIGURATION BACK	
1402	03CA	C9	RET ; RETURN TO CALLER	
1403				
1404				

```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  27

1405      ;
1406      ;
1407      ;
1408      ;
1409      ;
1410      ;
1411      ;
1412      ;
1413      ;
1414      ;
1415      ;
1416      ;
1417      ;
1418      ;
1419      ;
1420      ;
1421      03C8  79      VALDG: MOV      A,C
1422      03CC  FE 30      CPI      '0'      ; TEST CHAR AGAINST '0'
1423      03CE  FA 90 02      C      JM      FRET      ; CANNOT BE A HEX DIGIT IF LESS THAN ASCII FOR ZE
1424      03D1  FE 39      CPI      '9'      ; ELSE, SEE IF IN RANGE '0' - '9'
1425      03D3  FA 9F 03      C      JM      SRET      ; LESS THAN '9'??
1426      03D6  CA 9F 03      C      JZ      SRET      ; EQUAL TO '9'?
1427      03D9  FE 41      CPI      'A'      ; NOT A NUMERAL, SO TRY A DIGIT
1428      03DB  FA 90 02      C      JM      FRET      ; NO - RETURN
1429      03DE  FE 47      CPI      'G'
1430      03E0  F2 90 02      C      JP      FRET      ; NO - CODE GREATER THAN 'F'
1431      03E3  C3 9F 03      C      JMP     SRET      ; YES - CODE BETWEEN 'A' - 'F', INCLUSIVE
1432      ;
1433      ;
1434      ;
1435      ;
1436      ;
1437      ;
1438      ;
1439      ;
1440      ;
1441      ;
1442      ;
1443      ;
1444      ;
1445      ;
1446      ;
1447      ;
1448      ;
1449      ;
1450      ;
1451      ;
1452      ;
1453      ;
1454      03E6  79      VALDL: MOV      A,C
1455      03E7  FE 2C      CPI      ','      ; SEE IF COMMA
1456      03E9  CA 9F 03      C      JZ      SRET
1457      03EC  FE 0D      CPI      CR      ; SEE IF <CR>
1458      03EE  CA 9F 03      C      JZ      SRET

```



```

ERR LINE  ADDR  B1 B2 B3 B4      NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979      PAGE  28

1459 03F1 FE 20      CPI      ' '      ; SEE IF SPACE
1460 03F3 CA 9F 03    C      JZ      SRET
1461 03F6 C3 90 02    C      JMP     FRET      ; ERROR IF NONE OF THE ABOVE
1462      ;
1463      ;
1464      ;*****
1465      ;*****
1466      ;
1467      ;
1468      ;      MONITOR TABLES
1469      ;
1470      ;
1471      ;*****
1472      ;*****
1473      ;
1474      ;
1475 03F9 0D 0A 4E 44  SGNON: DB      CR,LF,'NDT-COMP9',CR,LF ; LOGON MESSAGE
1476 03FD 54 2D 43 4F
1477 0401 4D 50 39 0D
1478 0405 0A
1479 000D      LSGNON EQU      $-SGNON ; LENGTH OF SIGN-ON MESSAGE
1480      ;
1481      ;
1482 0406 00 00      CADR: DW      0      ; TABLE OF COMMAND ROUTINE ADDR'S - 1ST VAL IS DU
1483 0408 C4 01      DW      ZCMD
1484 040A 40 01      DW      XCMD
1485 040C 24 01      DW      SCMD
1486 040E 0A 01      DW      MCMD
1487 0410 C0 00      DW      ICMD
1488 0412 A2 00      DW      GCMD
1489 0414 60 00      DW      DCMD
1490 0416 DD 01      DW      CCMO
1491      ;
1492      ;
1493 0418 43      CTAB: DB      'C'      ; TABLE OF VALID COMMAND CHARS
1494 0419 44      DB      'D'
1495 041A 47      DB      'G'
1496 041B 49      DB      'I'
1497 041C 4D      DB      'M'
1498 041D 53      DB      'S'
1499 041E 50      DB      'X'
1500 041F 5A      DB      'Z'
1501 0000      NCMD5 EQU      $-CTAB ; NUMBER OF VALID COMMANDS
1502      ;
1503      ;
1504 0420 30      DIGTB: DB      '0'      ; TABLE OF ASCII CODE OF HEX DIGITS
1505 0421 31      DB      '1'
1506 0422 32      DB      '2'
1507 0423 33      DB      '3'
1508 0424 34      DB      '4'
1509 0425 35      DB      '5'
1510 0426 36      DB      '6'
1511 0427 37      DB      '7'
1512 0428 38      DB      '8'

```

ERR LINE	ADDR	B1	B2	B3	B4	NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979	PAGE	29
1513	0429	39				DB '9'		
1514	042A	41				DB 'A'		
1515	042B	42				DB 'B'		
1516	042C	43				DB 'C'		
1517	042D	44				DB 'D'		
1518	042E	45				DB 'E'		
1519	042F	46				DB 'F'		
1520						;		
1521						;		
1522	0430	41				RTAB: DB 'A' ; REGISTER IDENTIFIER		
1523	0431	F2				DB .LOW.ASAVE; ADDR OF REG SAVE LOC		
1524	0432	00				DB 0 ; LENGTH FLAG - 0=8 BITS, 1=16 BITS		
1525	0003					RTABS EQU \$-RTAB ; SIZE OF AN ENTRY IN THIS TABLE		
1526	0433	42				DB 'B'		
1527	0434	F0				DB .LOW.BSAVE		
1528	0435	00				DB 0		
1529	0436	43				DB 'C'		
1530	0437	EF				DB .LOW.CSAVE		
1531	0438	00				DB 0		
1532	0439	44				DB 'D'		
1533	043A	EE				DB .LOW.DSAVE		
1534	043B	00				DB 0		
1535	043C	45				DB 'E'		
1536	043D	ED				DB .LOW.ESAVE		
1537	043E	00				DB 0		
1538	043F	46				DB 'F'		
1539	0440	F1				DB .LOW.FSAVE		
1540	0441	00				DB 0		
1541	0442	48				DB 'H'		
1542	0443	F4				DB .LOW.HSAVE		
1543	0444	00				DB 0		
1544	0445	4C				DB 'L'		
1545	0446	F3				DB .LOW.LSAVE		
1546	0447	00				DB 0		
1547	0448	4D				DB 'M'		
1548	0449	F4				DB .LOW.HSAVE		
1549	044A	01				DB 1		
1550	044B	50				DB 'P'		
1551	044C	F6				DB .LOW.(PSAVE+1)		
1552	044D	01				DB 1		
1553	044E	53				DB 'S'		
1554	044F	F8				DB .LOW.(SSAVE+1)		
1555	0450	01				DB 1		
1556	0451	00				DB 0 ; END OF TABLE MARKERS		
1557	0452	00				DB 0		
1558						;		
1559						;		
1560	0453	06 04				MVI B,04H ; PRINT FOUR <LF> CHARS		
1561	0455	0E 0A				MVI C,0AH ; LOAD REG C		
1562	0457	CD 5B 02	C	RPT:	CALL CO ; PRINT A <LF>			
1563	045A	05				DCR B ; DECREMENT COUNTER		
1564	045B	C8				RZ ; RETURN TO CALLER WHEN DONE		
1565	045C	C3 57 04	C		JMP RPT ; LOOP ONCE MORE			
1566						;		

ERR LINE ADDR B1 B2 B3 B4

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 30

```

1567 ;
1568 ;*****
1569 ;*****
1570 ;
1571 ;
1572 ;           END OF MONITOR
1573 ;
1574 ;
1575 ;*****
1576 ;*****
1577 ;
1578 ;
1579 045F          END      START

```

ASSEMBLER ERRORS - 0

NDT-COMP9 MONITOR ASSEMBLY VER. 9 APRIL 1979

PAGE 31

SYMBOL TABLE

A	0007	ASAVE	3FF2	B	0000	BRCHR	001B
BREAK	C 0235	BRTAB	03FA	BSAVE	3FF0	C	0001
CADR	C 0406	CCMD	C 01DD	CI	C 0240	CMD	0027
CNCTL	00FB	CNIN	00FA	CNOUT	00FA	CNVBN	C 0252
CO	C 0250	COMPAR	C 01EB	CONST	00FB	COPDT	C 011C
COPLP	C 0110	CR	000D	CROUT	C 0266	CSAVE	3FEF
CTAB	C 0410	D	0002	DATA	3F00	DATCK	C 0220
DCM05	C 0072	DCM10	C 007D	DCM12	C 0092	DCM15	C 0090
DCMD	C 0060	DIGTB	C 0420	DSAVE	3FEE	E	0003
ECH05	C 0275	ECH10	C 0203	ECHO	C 026C	ERROR	C 0205
ESAVE	3FED	ESC	0010	FRET	C 0290	FSAVE	3FF1
GCM05	C 0007	GCM10	C 000D	GCMD	C 00A2	GETCH	C 0293
GETCM	C 003E	GETHX	C 029A	GETNM	C 02CF	GHX05	C 02A0
GHX10	C 0209	GNM05	C 02D6	GNM10	C 02EB	GNM15	C 02F9
GNM20	C 02FE	GNM25	C 0309	GNM30	C 030D	GO	C 0014
GT0A5	C 0A55	GT01A	C 0A51	H	000A	HCHAR	000F
HILO	C 0314	HSAVE	3FF4	ICM05	C 00CB	ICM10	C 00F3
ICM20	C 00FB	ICM25	C 0101	ICMD	C 00C0	INVRT	00FF
L	0005	LF	000A	LOWER	0000	LSAVE	3FF3
LSGNON	000D	M	0006	MCMD	C 010A	MEMORY	M 0000
MODE	00CE	MSGL	C 002E	MSTAK	3FED	NCMDS	0000
NEWLN	000F	NLSB	000D	NMOUT	C 031F	NMSB	0000
PRSGN	C 0030	PRTY0	007F	PRVAL	C 033A	PSAVE	3FF5
PSW	0006	RBR	0002	REG05	C 0345	REG10	C 034F
REG15	C 036A	REGDS	C 0342	REGS	3FED	RGA05	C 0379
RGA10	C 0306	RGADR	C 0373	RPT	C 0457	RSTTF	C 030A
RSTU	0030	RTAB	C 0430	RTABS	0003	RTCCTL	000F
RTCDAT	000E	RTCMD	000E	SCM05	C 0129	SCM10	C 0134
SCM15	C 0144	SCMD	C 0124	SGNON	C 03F9	SP	0006
SRET	C 039F	SSAVE	3FF7	STACK	S 0000	START	C 0000
STH05	C 030F	STHF0	C 03A1	STHLF	C 03AC	TEMP	3FF9
TERM	0010	TRDY	0001	UPPER	00FF	USINT	0000
VALDG	C 03CB	VALDL	C 03E6	XCM05	C 015B	XCM10	C 016A
XCM15	C 0177	XCM10	C 0102	XCM20	C 019B	XCM25	C 0102
XCM27	C 0103	XCM30	C 010B	XCMD	C 0140	ZCMD	C 01C4
ZERO	C 01D2						

APPENDIX E

MATH9 MATHEMATICS PACKAGE ASSEMBLY LISTING

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 1

```

1      TITLE 'MATH9 ASSEMBLY VER 9 APRIL 1979'
2      NAME MATH9
3      LIST      B,G,O,T ; LIST CROSS REFERENCE TABLE ONLY - PUT SYMBOLS
4                  ; INTO OBJECT MODULE TO ALLOW CREATION
5                  ; OF A LOAD MAP
6      ;      NLIST      I,M,S,X ; DO NOT LIST SOURCE TEXT OR CROSS-
7                  ; REFERENCE
8      PUBLIC      ACOS,ACOSA,ASIN,ASINA,ATAN,ATANA,BIDEC,BIDECF
9      PUBLIC      CHSD,CHSDA,CHSF,CHSFA,CHSS,CHSSA,COS,COSA
10     PUBLIC      DADD,DADDA,DADDB,DDIV,DDIVA,DDIVB,DECNO
11     PUBLIC      DMUL,DMULA,DMULB,DMUU,DMUUA,DMUUB,DSUB,DSUBA,DSUBB
12     PUBLIC      EXP,EXPA,EXPB,FADD,FADDA,FADDB,FDIV,FDIVA,FDIVB
13     PUBLIC      FIXD,FIXDA,FXS,FXSA,FLTD,FLTDA,FLTS,FLTSA
14     PUBLIC      FMUL,FMULA,FMULB,FSUB,FSUBA,FSUBB,LN,LNA,LOG,LOGA
15     PUBLIC      MDAD,NOP,POPD,POPS,PTOD,PTOS,PUP1,PWR,PWRA,PWRB
16     PUBLIC      SADD,SADDA,SADDB,SDIV,SDIVA,SDIVB,SIN,SINA
17     PUBLIC      SMUL,SMULA,SMULB,SMUU,SMUUA,SMUUB,SORT,SORTA
18     PUBLIC      SSUB,SSUBA,SSUBB,TAN,TANA,TOS1,TOS2,TOS4
19     PUBLIC      WRT1,WRT2,WRT4,XCHD,XCHS
20     PUBLIC      PRINT,PRINTF,EPRNT,FPRNT,COMPAR,ZERO
21     EXTRN      CO,CROUT,ECHO,NMOUT,PRVAL
22     ORG      0460H      ; START AT AFTER COMP9 MONITOR
23
24
25     ;*****
26     ;*****
27
28     ;
29     ;      FLOATING-POINT MATH PACKAGE FOR THE NDT-COMP9
30     ;
31     ;      THESE ROUTINES ALLOW THE USER TO UTILIZE THE AMD9511 ARITH-
32     ;      METIC PROCESSING UNIT ON THE NDT-COMP9.  THIS PACKAGE HAS
33     ;      32-BIT FLOATING-POINT AND 32- OR 16-BIT FIXED-POINT OPERA-
34     ;      TIONS.  IN ADDITION TO THE FOUR BASIC FUNCTIONS, TRANSCEN-
35     ;      DENTAL FUNCTIONS AND CONVERSION AND CONTROL OPERATIONS ARE
36     ;      AVAILABLE.
37     ;
38     ;
39     ;      AUTHORS C. V. DODD AND R. F. COWAN
40     ;
41     ;
42     ;      NONDESTRUCTIVE TESTING GROUP
43     ;      METALS AND CERAMICS DIVISION
44     ;      OAK RIDGE NATIONAL LABORATORIES
45     ;      OAK RIDGE, TENNESSEE 37830
46     ;
47     ;
48     ;*****
49     ;*****
50     ;
51     ;
52     ;      DATA TRANSFER TO THE APU:
53     ;      -----
54     ;

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 2

```

55      ;      ALL ARITHMETIC OPERATIONS ARE PERFORMED ON DATA STORED
56      ;      ON A LAST IN FIRST OUT STACK, WHICH MUST BE LOADED IF THE
57      ;      VALUES ARE NOT ALREADY PRESENT FROM A PREVIOUS OPERATION.
58      ;      THE REGISTERS ARE DESIGNATED A,B,C,D...AND ARE 4 32 BIT
59      ;      OR 8 16 BIT REGISTERS. A CALL WITH AN A SUFFIX WILL LOAD
60      ;      REGISTER A, AND A CALL WITH A B SUFFIX WILL LOAD B, THEN A,
61      ;      AND A CALL WITH NO SUFFIX ASSUMES A & B (IF NEEDED) ARE
62      ;      ALREADY ON THE STACK. WHEN A SINGLE OR DOUBLE-OPERAND IS
63      ;      LOADED, THE FIRST OPERAND (B IF TWO, A IF ONE) IS ASSUMED
64      ;      TO BE STORED IN RAM WITH THE H,L REGISTER PAIR
65      ;      POINTING TO THE LSB. THE H,L ADDRESS IS DECREMENTED TO
66      ;      ACCESS THE HIGHER-ORDER BYTES. SIMILARLY, FOR DOUBLE-OPERAND
67      ;      INSTRUCTIONS, THE D,E REGISTER PAIR IS ASSUMED TO POINT TO
68      ;      THE LSB OF THE SECOND OPERAND AND IS DECREMENTED TO ACCESS
69      ;      THE HIGHER-ORDER BYTES.
70      ;
71      ;
72      ;      AFTER AN OPERATION IS COMPLETED, THE RESULT IS STORED ON THE
73      ;      TOP OF THE INTERNAL STACK IN THE APU. THIS RESULT CAN BE
74      ;      ACCESSED VIA THE TOS2 OR THE TOS4 ROUTINES AS APPROPRIATE.
75      ;
76      ;
77      ;*****
78      ;
79      ;
80      ;
81      ;
82      ;*****
83      ;
84      ;
85      ;      SYMBOL DEFINITIONS
86      ;
87      ;
88      ;*****
89      ;
90      ;
91      007F      APCMD EQU      7FH      ; ARITHMETIC PROCESSOR CONTROL PORT
92      007E      APDATA EQU     7EH      ; ARITHMETIC PROCESSOR DATA PORT
93      007F      APSTS EQU      7FH      ; ARITHMETIC PROCESSOR STATUS PORT
94      0000      MASK F0H      RAM      ; MASK FOR BUSY BIT
95      ;
96      3C00      EXP10 EQU      3C00H     ; START OF RAM STORAGE, SIGN/EXPONENT LOCATION
97      3C01      BINNO EQU      EXP10+1   ; 4 STORAGE LOCATIONS FOR BINARY NUMBER
98      3C05      DUBLE EQU      BINNO+4   ; 5 BYTES, SCRATCH STORAGE FOR BI TO DEC CONVERSION
99      3C0A      DECNO EQU      DUBLE+5   ; MUST FOLLOW DUBLE, FIXED DECIMAL NO STORED HERE
100     ;
101     ;      NEXT BLANK RAM=DECNO+5
102     ;*****
103     ;
104     ;
105     ;      ROUTINE ACOS - 32-BIT, FLOATING-POINT INVERSE COSINE
106     ;
107     ;      R <-- ARCCOS(A)
108     ;

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 3

```

109      ;          DESTROYS:  A,H,L
110      ;
111      ;
112      ;*****
113      ;
114      ;
115 0460 CD 70 06 ACOSA: CALL  WRT4      ; GET 32-BIT ARG
116 0463 3E 06 ACOS:  MVI   A,06H    ; LOAD ACOS OP CODE
117 0465 D3 7F      OUT   APCMD     ; ISSUE COMMAND
118 0467 C9      RET               ; RETURN TO CALLER
119      ;
120      ;
121      ;*****
122      ;
123      ;
124      ;          ROUTINE ASIN - 32-BIT, FLOATING-POINT INVERSE SINE
125      ;
126      ;          R <-- ARCSIN(A)
127      ;
128      ;          DESTROYS:  A,H,L
129      ;
130      ;
131      ;*****
132      ;
133      ;
134 0468 CD 70 06 ASINA: CALL  WRT4      ; GET 32-BIT ARG
135 046B 3E 05 ASIN:  MVI   A,05H    ; LOAD OP CODE
136 046D D3 7F      OUT   APCMD     ; ISSUE COMMAND
137 046F C9      RET               ; RETURN TO CALLER
138      ;
139      ;
140      ;*****
141      ;
142      ;
143      ;          ROUTINE ATAN - 32-BIT, FLOATING-POINT INVERSE TANGENT
144      ;
145      ;          R <-- ARCTAN(A)
146      ;
147      ;          DESTROYS:  A,H,L
148      ;
149      ;
150      ;*****
151      ;
152      ;
153 0470 CD 70 06 ATANA: CALL  WRT4      ; GET 32-BIT ARG
154 0473 3E 07 ATAN:  MVI   A,07H    ; LOAD OP CODE
155 0475 D3 7F      OUT   APCMD     ; ISSUE COMMAND
156 0477 C9      RET               ; RETURN TO CALLER
157      ;
158      ;
159      ;*****
160      ;
161      ;          ROUTINE BIDECE - CONVERTS A 32 BIT FLOATING POINT NUMBER
162      ;          NUMBER TO A FLOATING POINT DECIMAL NUMBER

```


ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 4

```

163 ; ROUTINE BIDECE - ENTRY POINT FOR A FIXED POINT NUMBER
164 ;
165 ;*****
166 ;
167 ;
168 0478 DB 7F BIDECE: IN APSTS ;LOOK AT THE STATUS OF NO ON TOS
169 047A E6 36 ANI 036H ;LOOK AT ERROR & ZERO BITS
170 047C C2 02 05 JNZ ERROR ;SOMETHING'S KINKY,CHECK IT OUT
171 047F DB 7E IN APDATA ;GET EXPONENT AND SIGN FROM TOS
172 0481 D3 7E OUT APDATA ;RESTORE STACK
173 0483 4F MOV C,A ;MAKE A COPY IN C REGISTER
174 0484 E6 00 ANI MASK ;DUMP ALL BUT THE SIGN BIT (BIT 0)
175 0486 C6 20 ADI 020H ;BASE 10 EXPONENT WILL BE OFFSET BY 20
176 0488 47 MOV B,A ;SAVE SIGN IN THE B REGISTER
177 0489 79 MOV A,C ;RESTORE EXPONENT
178 048A E6 7F CHECK: ANI 07FH ;DUMP SIGN BIT
179 048C C6 C0 ADI 0C0H ;ADD OFF-SET (CONVERTS FROM 2'S COMPLEMENT TO
180 048E FE D6 CPI 0D6H ;OFFSET BINARY) COMPARE EXPONENT TO 22,IF .LT.
181 0490 DA 9B 04 JC MLTPLY ;CARRY IS GENERATED AND WE JUMP TO THE MULTIPLY
182 0493 FE DA CPI 0DAH ;LOOP,COMPARE EXP TO 26,IF.GT.NO CARRY GENERATED
183 0495 D2 A2 04 JNC DIVIDE ;AND WE JUMP TO DIVIDE
184 0498 C3 00 04 JMP CONT ;EXPONENT BETWEEN 22 AND 26,CONTINUE CALCULATIONS
185 049B 21 91 06 MLTPLY:LXI H,FACT10+3;LOAD ADDRESS OF FLOATING POINT 10
186 049E 05 DCR B ;EXPONENT IS DECREMENTED
187 049F C3 A6 04 JMP CONT1 ;JUMP AROUND
188 04A2 21 0D 06 DIVIDE:LXI H,FACT01+3;LOAD ADDRESS OF 0.1 INTO H,L
189 04A5 04 INR B ;EXPONENT IS INCREMENTED
190 04A6 CD BE 05 CONT1: CALL FMULA ;CALL FLOATING POINT MULTIPLY ROUTINE,LOAD A
191 04A9 DB 7E IN APDATA ;GET EXPONENT AND SIGN FROM MULTIPLICATION
192 04AB D3 7E OUT APDATA ;RESTORE STACK ON 9511.
193 04AD C3 0A 04 JMP CHECK ;LOOP BACK AROUND UNTIL NUMBER IS IN PROPER RANGE
194 04B0 CD A5 05 CONT: CALL FIXD ;NUMBER IS FIXED
195 04B3 78 MOV A,B ;THE EXPONENT AND SIGN ARE STORED
196 04B4 32 00 3C STA EXP10 ;IN MEMORY EXP10. LOCATION.
197 04B7 16 04 MVI D,04H ;SET D FOR A 4 BYTE (8 DIGIT) DECIMAL NUMBER
198 ;
199 ;
200 ; 32 BIT FIXED BINARY TO DECIMAL ROUTINE
201 ;
202 04B9 DB 7E BIDECE: IN APDATA ;GET SIGN OF NUMBER FROM TOP OF STACK
203 04BB D3 7E OUT APDATA ;RESTORE STACK
204 04BD E6 00 ANI MASK ;TEST SIGN BIT
205 04BF CA C5 04 JZ CONT2 ;JUMP AROUND IF NUMBER WAS POSITIVE
206 04C2 CD 21 05 CALL CHSD ;CHANGE SIGNS
207 04C5 21 01 3C CONT2: LXI H,BINNO ;LOAD ADDRESS WHERE BINARY NO IS TO BE STORED
208 04C8 CD 60 06 CALL TOS4 ;DATA FROM STACK OF 9511 STORED IN RAM,LSB,2SB
209 04CB 21 05 3C LXI H,DUBLE ;ADDRESS OF LOW ORDER BYTE OF NO TO BE DOUBLED
210 04CE 0E 0A MVI C,0AH ;SET COUNTER TO 10
211 04D0 AF XRA A ;ZERO ACCUMULATOR
212 04D1 CD 0B 05 CALL ZERO ;ZERO MEMORY FROM HL TO HL +C
213 04D4 3C INR A ;A REG HAS 01
214 04D5 32 05 3C STA DUBLE ;DUBLE HAS A VALUE OF 1
215 04D8 1E 1F MVI E,01FH ;SET FOR 31 BITS
216 04DB 4A LOOP4: PUSH D ;SAVE A COPY OF D,E ON THE STACK
MOV C,D ;THE NUMBER OF BYTES TO BE TESTED MOVED TO C

```

ERR LINE	ADDR	B1	B2	B3	B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 5
217	04DC	21	01	3C		LXI H,BINNO ;THE ADDRESS OF THE BINARY NUMBER LOADED IN H,L	
218	04DF	AF				XRA A ;CARRY BIT CLEARED	
219	04E0	7E				LOOPS: MOV A,M ;MULTI BYTE RIGHT ROTATION,MOVE BYTE TO A	
220	04E1	1F				RAR ;ROTATE ACCUMULATOR RIGHT,LSB>CARRY,CARRY>MSB	
221	04E2	77				MOV M,A ;ROTATED BYTE SAVED	
222	04E3	23				INX H ;ADDRESS INCREMENTED FOR NEXT BYTE	
223	04E4	0D				DCR C ;BYTE COUNTER DECREMENTED	
224	04E5	C2	E0	04		JNZ LOOPS ;JUMP BACK IF NOT FINISHED	
225	04E8	01	0A	3C		LXI B,DECNO ;LOAD ADDRESS OF DECIMAL NUMBER INTO B,C	
226	04E8	21	05	3C		LXI H,DUBLE ;LOAD ADDRESS OF DUBLED NUMBER INTO H,L	
227	04EE	DC	12	05		CC MDAD ;CALL MULTI BYTE ADD IF LSB OF LAST BYTE WAS 1	
228	04F1	D1				POP D ;GET A NEW COPY OF D,E	
229	04F2	D5				PUSH D ;SAVE ANOTHER	
230	04F3	21	05	3C		LXI H,DUBLE ;LOAD ADDRESS OF DUBLED NUMBER INTO H,L	
231	04F6	01	05	3C		LXI B,DUBLE ;LOAD ADDRESS OF DUBLED NUMBER INTO B,C	
232	04F9	CD	12	05		CALL MDAD ;DOUBLE THE NUMBER AND STORE IN DUBLE	
233	04FC	D1				POP D ;RESTORE D,C	
234	04FD	1D				DCR E ;DECREMENT THE BIT COUNTER	
235	04FE	C2	DA	04		JNZ LOOP4 ;GO BACK AND REPEAT FOR NEXT BIT IF NOT FINISHED	
236	0501	C9				RET ;GO BACK TO THE CALLING ROUTINE;DECIMAL NO IN RAM	
237	0502	32	00	3C		ERROR: STA EXP10 ;STORE ERROR OR ZERO AS EXPONENT	
238	0505	AF				XRA A ;ZERO ACCUMULATOR	
239	0506	21	0A	3C		LXI H,DECNO ;LOAD ADDR OF DECIMAL NO IN H	
240	0509	0E	05			MVI C,05H ;NUMBER OF BYTES TO BE WRITTEN IN C	
241							
242							
243						ROUTINE ZERO	
244						ROUTINE MOVES VALUE IN A INTO RAM FROM HL TO	
245	0508	77				HL+C,REGS H,L&C ARE DESTROYED	
246	050C	23				ZERO: MOV M,A ;MOVE ACCUMULATOR TO MEMORY	
247	050D	0D				INX H ;INCREMENT HL COUNTER	
248	050E	C2	0B	05		DCR C ;DECREMENT C COUNTER	
249	0511	C9				JNZ ZERO ;JUMP BACK IF NOT ZERO	
250						RET ;RETURN TO CALLING ROUTINE	
251							
252							
253							
254						ROUTINE MDAD	
255						MULTI BYTE DECIMAL ADD - ADD STRING ADDRESSED	
256						BY H,L TO STRING ADDRESSED BY B,C.THE RESULT REPLACES B,C.	
257						D CONTAINS THE NUMBER OF BYTES TO BE ADDED.A,B,C,D,H,L USED.	
258						THE BYTES ARE STORED:LSB,2SB,3SB,4SB,MSB.	
259	0512	AF				MDAD: XRA A ;CLEAR CARRY BIT	
260	0513	0A				LDAX B ;LOAD BYTE ADDRESSED BY B,C INTO A	
261	0514	0E				ADC M ;ADD WITH CARRY TO BYTE ADDRESSED BY H,L	
262	0515	27				DAA ;DECIMAL ADJUST ACCUMULATOR	
263	0516	02				STAX B ;REPLACE B,C ADDRESSED BYTE	
264	0517	15				DCR D ;DECREMENT COUNTER IN D	
265	0518	C0				RZ ;RETURN IF FINISHED	
266	0519	03				INX B ;POINT B TO NEXT HIGHER BYTE	
267	051A	23				INX H ;POINT H TO NEXT HIGHER BYTE	
268	051B	C3	13	05		JMP LOOP ;RETURN TO DO NEXT BYTE	
269							
270							

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 6

```

271 ;*****
272 ;
273 ;
274 ; ROUTINE CHSD - 32-BIT, FIXED-POINT SIGN CHANGE
275 ;
276 ; R <-- -(A)
277 ;
278 ; DESTROYS: A,H,L
279 ;
280 ;
281 ;*****
282 ;
283 ;
284 051E CD 70 06 CHSDA: CALL WRT4 ; GET 32-BIT ARG
285 0521 3E 34 CHSD: MVI A,34H ; LOAD OP CODE
286 0523 D3 7F OUT APCMD ; ISSUE COMMAND
287 0525 C9 RET ; RETURN TO CALLER
288 ;
289 ;
290 ;*****
291 ;
292 ;
293 ; ROUTINE CHSF - 32-BIT, FLOATING-POINT SIGN CHANGE
294 ;
295 ; R <-- -(A)
296 ;
297 ; DESTROYS: A,H,L
298 ;
299 ;
300 ;*****
301 ;
302 ;
303 0526 CD 70 06 CHSFA: CALL WRT4 ; GET 32-BIT ARG
304 0529 3E 15 CHSF: MVI A,15H ; LOAD OP CODE
305 052B D3 7F OUT APCMD ; ISSUE COMMAND
306 052D C9 RET ; RETURN TO CALLER
307 ;
308 ;
309 ;*****
310 ;
311 ;
312 ; ROUTINE CHSS - 16-BIT, FIXED POINT SIGN CHANGE
313 ;
314 ; R <-- -(A)
315 ;
316 ; DESTROYS: A,H,L
317 ;
318 ;
319 ;*****
320 ;
321 ;
322 052E CD 70 06 CHSSA: CALL WRT2 ; GET 16-BIT ARG
323 0531 3E 74 CHSS: MVI A,74H ; LOAD OP CODE
324 0533 D3 7F OUT APCMD ; ISSUE INSTRUCTION

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 7

325 0535 C9

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341 0536 CD 70 06

342 0539 3E 03

343 053B D3 7F

344 053D C9

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361 053E CD 70 06

362 0541 EB

363 0542 CD 70 06

364 0545 3E 2C

365 0547 D3 7F

366 0549 C9

367

368

369

370

371

372

373

374

375

376

377

378

RET ; RETURN TO CALLER

ROUTINE COS - 32-BIT, FLOATING-POINT COSINE

R <-- COS(A)

DESTROYS: H,H,L

```

COSA: CALL WRT4 ; GET 32-BIT ARG
COS: MVI A,03H ; LOAD OP CODE
OUT APCMD ; ISSUE COMMAND
RET ; RETURN TO CALLER

```

ROUTINE DADD - 32-BIT, FIXED-POINT ADDITION

R <-- A+B

DESTROYS: A,D,E,H,L

```

DADDB: CALL WRT4 ; GET ARG B
XCHG ; SWAP MEM POINTERS
DADDA: CALL WRT4 ; GET ARG A
DADD: MVI A,2CH ; LOAD OP CODE
OUT APCMD ; ISSUE COMMAND
RET ; RETURN TO CALLER

```

ROUTINE DDIV - 32-BIT, FIXED-POINT DIVIDE

R <-- B/A (INTEGER ONLY - NO REMAINDER GENERATED)

DESTROYS: A,D,E,H,L

ERR LINE	ADDR	B1	B2	B3	B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 8
379						;	
380						;	
381						;	
382	054A	CD	70	06		DDIVB: CALL WRT4 ; GET ARG B	
383	054D	EB				XCHG ; SWAP MEM POINTERS	
384	054E	CD	70	06		DDIVA: CALL WRT4 ; GET ARG A	
385	0551	3E	2F			DDIV: MVI A,2FH ; LOAD OP CODE	
386	0553	D3	7F			OUT APCMD ; ISSUE COMMAND	
387	0555	C9				RET ; RETURN TO CALLER	
388						;	
389						;	
390						;	
391						;	
392						;	
393						ROUTINE DMUL - 32-BIT, FIXED-POINT MULTIPLY, LOWER	
394						;	
395						R <-- A*B (LOWER 32 BITS ONLY)	
396						;	
397						DESTROYS: A,D,E,H,L	
398						;	
399						;	
400						;	
401						;	
402						;	
403	0556	CD	70	06		DMULB: CALL WRT4 ; GET ARG B	
404	0559	EB				XCHG ; SWAP MEM POINTERS	
405	055A	CD	70	06		DMULA: CALL WRT4 ; GET ARG A	
406	055D	3E	2E			DMUL: MVI A,2EH ; LOAD OP CODE	
407	055F	D3	7F			OUT APCMD ; ISSUE COMMAND	
408	0561	C9				RET ; RETURN TO CALLER	
409						;	
410						;	
411						;	
412						;	
413						;	
414						ROUTINE DMUJ - 32-BIT, FIXED-POINT MULTIPLY, UPPER	
415						;	
416						R <-- A*B (UPPER 32 BITS ONLY)	
417						;	
418						DESTROYS: A,D,E,H,L	
419						;	
420						;	
421						;	
422						;	
423						;	
424	0562	CD	70	06		DMUJB: CALL WRT4 ; GET ARG B	
425	0565	EB				XCHG ; SWAP MEM POINTERS	
426	0566	CD	70	06		DMUJA: CALL WRT4 ; GET ARG A	
427	0569	3E	36			DMUJ: MVI A,36H ; LOAD OP CODE	
428	056B	D3	7F			OUT APCMD ; ISSUE COMMAND	
429	056D	C9				RET ; RETURN TO CALLER	
430						;	
431						;	
432						;	

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 9

```

433 ;
434 ;
435 ; ROUTINE DSUB - 32-BIT, FIXED-POINT SUBTRACT
436 ;
437 ; R <-- B-A
438 ;
439 ; DESTROYS: A,D,E,H,L
440 ;
441 ;
442 ;
443 ;
444 ;
445 056E CD 70 06 DSUB: CALL WRT4 ; GET ARG B
446 0571 EB XCHG ; SWAP MEM POINTERS
447 0572 CD 70 06 DSUB: CALL WRT4 ; GET ARG A
448 0575 3E 2D DSUB: MVI A,2DH ; LOAD OP CODE
449 0577 D3 7F OUT APCMD ; ISSUE COMMAND
450 0579 C9 RET ; RETURN TO CALLER
451 ;
452 ;
453 ;
454 ;
455 ;
456 ; ROUTINE EXP - 32-BIT, FLOATING-POINT EXPONENTIATION
457 ;
458 ; R <-- EXP(A)
459 ;
460 ; DESTROYS: A,D,E,H,L
461 ;
462 ;
463 ;
464 ;
465 ;
466 057A CD 70 06 EXP: CALL WRT4 ; GET ARG A
467 057D 3E 0A EXP: MVI A,0AH ; LOAD OP CODE
468 057F D3 7F OUT APCMD ; ISSUE COMMAND
469 0581 C9 RET ; RETURN TO CALLER
470 ;
471 ;
472 ;
473 ;
474 ;
475 ; ROUTINE FADD - 32-BIT FLOATING POINT ADD
476 ;
477 ; R <-- A+B
478 ;
479 ; DESTROYS: A,D,E,H,L
480 ;
481 ;
482 ;
483 ;
484 ;
485 0582 CD 70 06 FADD: CALL WRT4 ; GET ARG B
486 0585 EB XCHG ; SWAP MEM POINTERS

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 10

487 0586 CD 70 06
 488 0589 3E 10
 489 058B D3 7F
 490 058D C9

FADDA: CALL WRT4 ; GET ARG A
 FADD: MVI A,10H ; LOAD OP CODE
 OUT APCMD ; ISSUE COMMAND
 RET ; RETURN TO CALLER

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508 058E CD 70 06

509 0591 EB

510 0592 CD 70 06

511 0595 3E 13

512 0597 D3 7F

513 0599 C9

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531 059A CD 70 06

532 059D 3E 1F

533 059F D3 7F

534 05A1 C9

535

536

537

538

539

540

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

ROUTINE FDIV - 32-BIT FLOATING POINTDIVIDE

R <-- B/A

DESTROYS: A,D,E,H,L

FDIVB: CALL WRT4 ; GET ARG B
 XCHG ; SWAP MEM POINTERS
 FDIVA: CALL WRT4 ; GET ARG A
 FDIV: MVI A,13H ; LOAD OP CODE
 OUT APCMD ; ISSUE COMMAND
 RET ; RETURN TO CALLER

ROUTINE FIXS - 32-BIT, FLOATING-POINT TO 16-BIT FIXED-POINT

R <-- FIX(A)

DESTROYS: A,H,L

FIXSA: CALL WRT4 ; GET 32-BIT ARG
 FIXS: MVI A,1FH ; LOAD OP CODE
 OUT APCMD ; ISSUE COMMAND
 RET ; RETURN TO CALLER

ROUTINE FIXD - 32-BIT, FLOATING-POINT TO 32-BIT FIXED-POINT

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 11

```

541 ;
542 ; R <-- FIX(A)
543 ;
544 ; DESTROYS: A,H,L
545 ;
546 ;
547 ;*****
548 ;
549 ;
550 05A2 CD 70 06 FIXDA: CALL WRT4 ; GET 32-BIT ARG
551 05A5 3E 1E FIXD: MVI A,1EH ; LOAD OP CODE
552 05A7 D3 7F OUT APCMD ; ISSUE COMMAND
553 05A9 C9 RET ; RETURN TO CALLER
554 ;
555 ;
556 ;*****
557 ;
558 ;
559 ; ROUTINE FLTD - 32-BIT FIXED-POINT TO 32-BIT FLOATING-POINT
560 ;
561 ; R <-- FLOAT(A)
562 ;
563 ; DESTROYS: A,H,L
564 ;
565 ;
566 ;*****
567 ;
568 ;
569 05AA CD 70 06 FLTDA: CALL WRT4 ; GET 32-BIT ARG
570 05AD 3E 1C FLTD: MVI A,1CH ; LOAD OP CODE
571 05AF D3 7F OUT APCMD ; ISSUE COMMAND
572 05B1 C9 RET ; RETURN TO CALLER
573 ;
574 ;
575 ;*****
576 ;
577 ;
578 ; ROUTINE FLTS - 16-BIT FIXED-POINT TO 32-BIT FLOATING-POINT
579 ;
580 ; R <-- FLOAT(A)
581 ;
582 ; DESTROYS: A,H,L
583 ;
584 ;
585 ;*****
586 ;
587 ;
588 05B2 CD 7A 06 FLTSA: CALL WRT2 ; GET 16-BIT ARG
589 05B5 3E 1D FLTS: MVI A,1DH ; LOAD OP CODE
590 05B7 D3 7F OUT APCMD ; ISSUE COMMAND
591 05B9 C9 RET ; RETURN TO CALLER
592 ;
593 ;
594 ;

```


ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 12

```

595 ;*****
596 ;
597 ;
598 ;      ROUTINE FMUL - 32-BIT FLOATING POINTMULTIPLY
599 ;
600 ;      R <-- A*B
601 ;
602 ;      DESTROYS:  A,D,E,H,L
603 ;
604 ;
605 ;*****
606 ;
607 ;
608 058A CD 70 06 FMULB: CALL  WRT4      ; GET ARG B
609 058D EB                XCHG          ; SWAP MEM POINTERS
610 058E CD 70 06 FMULA: CALL  WRT4      ; GET ARG A
611 05C1 3E 12 FMUL:  MVI    A,12H      ; LOAD OP CODE
612 05C3 D3 7F      OUT    APCMD      ; ISSUE COMMAND
613 05C5 C9                RET          ; RETURN TO CALLER
614 ;
615 ;
616 ;
617 ;
618 ;*****
619 ;
620 ;
621 ;      ROUTINE FSUB - 32-BIT FLOATING POINT SUBTRACT
622 ;
623 ;      R <-- B-A
624 ;
625 ;      DESTROYS:  A,D,E,H,L
626 ;
627 ;
628 ;*****
629 ;
630 ;
631 05C6 CD 70 06 FSUBB: CALL  WRT4      ; GET ARG B
632 05C9 EB                XCHG          ; SWAP MEM POINTERS
633 05CA CD 70 06 FSUBA: CALL  WRT4      ; GET ARG A
634 05CD 3E 11 FSUB:  MVI    A,11H      ; LOAD OP CODE
635 05CF D3 7F      OUT    APCMD      ; ISSUE COMMAND
636 05D1 C9                RET          ; RETURN TO CALLER
637 ;
638 ;
639 ;*****
640 ;
641 ;
642 ;      ROUTINE LOG - 32-BIT, FLOATING-POINT COMMON LOGARITHM
643 ;
644 ;      R <-- LOG(A)
645 ;
646 ;      DESTROYS:  A,D,E,H,L
647 ;
648 ;

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 13

```

649 ;*****
650 ;
651 ;
652 05D2 CD 70 06 LOGA: CALL WRT4 ; GET ARG A
653 05D5 3E 08 LOG: MVI A,08H ; LOAD OP CODE
654 05D7 D3 7F OUT APCMD ; ISSUE COMMAND
655 05D9 C9 RET ; RETURN TO CALLER
656 ;
657 ;
658 ;*****
659 ;
660 ;
661 ; ROUTINE LN - 32-BIT, FLOATING-POINT NATURAL LOGARITHM
662 ;
663 ; R <-- LN(A)
664 ;
665 ; DESTROYS: A.D.E.H.L
666 ;
667 ;
668 ;*****
669 ;
670 ;
671 05DA CD 70 06 LNA: CALL WRT4 ; GET ARG A
672 05DD 3E 09 LN: MVI A,09H ; LOAD OP CODE
673 05DF D3 7F OUT APCMD ; ISSUE COMMAND
674 05E1 C9 RET ; RETURN TO CALLER
675 ;
676 ;*****
677 ;
678 ;
679 ; ROUTINE NOP - NO OPERATION, CLEARS STATUS
680 ;
681 ; DESTROYS: A
682 ;
683 ;
684 ;*****
685 ;
686 ;
687 05E2 3E 00 NOP: MVI A,00H ; LOAD OP CODE
688 05E4 D3 7F OUT APCMD ; ISSUE COMMAND
689 05E6 C9 RET ; RETURN TO CALLER
690 ;
691 ;
692 ;*****
693 ;
694 ;
695 ; ROUTINE POPD - 32-BIT APU STACK POP. TOS ROTATES TO BOTTOM,
696 ; OLD NOS BECOMES NEW TOS.
697 ;
698 ; DESTROYS: A
699 ;
700 ;
701 ;*****
702 ;

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 14

```

703
704 05E7 3E 30      ; POPD: MVI    A,30H ; LOAD OP CODE
705 05E9 D3 7F      ;      OUT    APCMD ; ISSUE COMMAND
706 05EB C9         ;      RET    ; RETURN TO CALLER
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721 05EC 3E 70      ; POPS: MVI    A,70H ; LOAD OP CODE
722 05EE D3 7F      ;      OUT    APCMD ; ISSUE COMMAND
723 05F0 C9         ;      RET    ; RETURN TO CALLER
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738 05F1 3E 37      ; PTOD: MVI    A,37H ; LOAD OP CODE
739 05F3 D3 7F      ;      OUT    APCMD ; ISSUE COMMAND
740 05F5 C9         ;      RET    ; RETURN TO CALLER
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755 05F6 3E 77      ; PTOS: MVI    A,77H ; LOAD OP CODE
756 05F8 D3 7F      ;      OUT    APCMD ; ISSUE COMMAND

```

ROUTINE POPS - 16-BIT APU STACK POP. TOS ROTATES TO BOTTOM,
OLD NOS BECOMES NEW TOS.

DESTROYS: A

ROUTINE PTOD - PUSH 32-BIT STACK DOWN AND COPY OLD TOS
INTO NEW TOS.

DESTROYS: A

ROUTINE PTOS - PUSH 16-BIT STACK DOWN AND COPY THE PREVIOUS
TOS INTO THE NEW TOS.

DESTROYS: A

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 15

```

757 05FA C9          RET          ; RETURN TO CALLER
758                  ;
759                  ;
760                  ;
761                  ;
762                  ;
763                  ;
764                  ;
765                  ;
766                  ;
767                  ;
768                  ;
769                  ;
770                  ;
771 05FB 3E 1A      PUPI: MVI      A,1AH  ; LOAD OP CODE
772 05FD D3 7F      OUT        APCMD  ; ISSUE COMMAND
773 05FF C9        RET          ; RETURN TO CALLER
774                  ;
775                  ;
776                  ;
777                  ;
778                  ;
779                  ;
780                  ;
781                  ;
782                  ;
783                  ;
784                  ;
785                  ;
786                  ;
787                  ;
788                  ;
789 0600 CD 70 06    PLRB: CALL     WRT4   ; GET ARG B
790 0603 EB          XCHG          ; SWAP MEM POINTERS
791 0604 CD 70 06    PLRA: CALL     WRT4   ; GET ARG A
792 0607 3E 0B      PLR:  MVI      A,0BH  ; LOAD OP CODE
793 0609 D3 7F      OUT        APCMD  ; ISSUE COMMAND
794 060B C9        RET          ; RETURN TO CALLER
795                  ;
796                  ;
797                  ;
798                  ;
799                  ;
800                  ;
801                  ;
802                  ;
803                  ;
804                  ;
805                  ;
806                  ;
807                  ;
808                  ;
809                  ;
810 060C CD 70 06    SADD: CALL     WRT2   ; GET ARG B

```

ROUTINE PUPI - PUSH 32-BIT FLOATING-POINT PI ONTO STACK

DESTROYS: A

ROUTINE PLR - RAISE B TO THE A POWER

R <-- B**A

DESTROYS: A,D,E,H,L

ROUTINE SADD - 16-BIT, FIXED-POINT ADDITION

R <-- A+B

DESTROYS: A,D,E,H,L

ERR LINE	ADDR	B1 B2 B3 B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 16
811	060F	EB	XCHG ; SWAP MEM POINTERS	
812	0610	CD 78 06	SADDA: CALL WRT2 ; GET ARG A	
813	0613	3E 08	SADD: MVI A,08H ; LOAD OP CODE	
814	0615	D3 7F	OUT APCMD ; ISSUE COMMAND	
815	0617	C9	RET ; RETURN TO CALLER	
816			;	
817			;	
818			;	
819			;	
820			;	
821			ROUTINE SDIV - 16-BIT, FIXED-POINT DIVIDE, NO REMAINDER	
822			;	
823			R <-- B/A (NO REMAINDER IS GENERATED)	
824			;	
825			DESTROYS: A,D,E,H,L	
826			;	
827			;	
828			;	
829			;	
830			;	
831	0618	CD 78 06	SDIVB: CALL WRT2 ; GET ARG B	
832	0618	EB	XCHG ; SWAP MEM POINTERS	
833	061C	CD 78 06	SDIVA: CALL WRT2 ; GET ARG A	
834	061F	3E 6F	SDIV: MVI A,6FH ; LOAD OP CODE	
835	0621	D3 7F	OUT APCMD ; ISSUE COMMAND	
836	0623	C9	RET ; RETURN TO CALLER	
837			;	
838			;	
839			;	
840			;	
841			;	
842			ROUTINE SIN - 32-BIT, FLOATING-POINT SINE	
843			;	
844			R <-- SIN(A)	
845			;	
846			DESTROYS: A,H,L	
847			;	
848			;	
849			;	
850			;	
851			;	
852	0624	CD 78 06	SINA: CALL WRT4 ; GET ARG A	
853	0627	3E 02	SIN: MVI A,02H ; LOAD OP CODE	
854	0629	D3 7F	OUT APCMD ; ISSUE COMMAND	
855	062B	C9	RET ; RETURN TO CALLER	
856			;	
857			;	
858			;	
859			;	
860			;	
861			ROUTINE SMUL - 16-BIT, FIXED-POINT MULTIPLY, LOWER	
862			;	
863			R <-- A*B (LOWER 16 BITS ONLY)	
864			;	

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 17

```

865                                ; DESTROYS: A,D,E,H,L
866                                ;
867                                ;
868                                ; *****
869                                ;
870                                ;
871 062C CD 78 06 SMULB: CALL WRT2 ; GET ARG B
872 062F EB XCHG ; SWAP MEM POINTERS
873 0638 CD 78 06 SMULA: CALL WRT2 ; GET ARG A
874 0633 3E 6E SMUL: MVI A,06EH ; LOAD OP CODE
875 0635 D3 7F OUT APCMD ; ISSUE COMMAND
876 0637 C9 RET ; RETURN TO CALLER
877                                ;
878                                ;
879                                ; *****
880                                ;
881                                ;
882                                ; ROUTINE SMU - 16-BIT, FIXED-POINT MULTIPLY, UPPER
883                                ;
884                                ; R <-- A*B (UPPER 16 BITS ONLY)
885                                ;
886                                ; DESTROYS: A,D,E,H,L
887                                ;
888                                ;
889                                ; *****
890                                ;
891                                ;
892 0638 CD 78 06 SMUUB: CALL WRT2 ; GET ARG B
893 063B EB XCHG ; SWAP MEM POINTERS
894 063C CD 78 06 SMUUA: CALL WRT2 ; GET ARG A
895 063F 3E 76 SMUJ: MVI A,76H ; LOAD OP CODE
896 0641 D3 7F OUT APCMD ; ISSUE COMMAND
897 0643 C9 RET ; RETURN TO CALLER
898                                ;
899                                ;
900                                ; *****
901                                ;
902                                ;
903                                ; ROUTINE SQRT - 32-BIT, FLOATING-POINT SQUARE ROOT
904                                ;
905                                ; R <-- SQRT(A)
906                                ;
907                                ; DESTROYS: A,H,L
908                                ;
909                                ;
910                                ; *****
911                                ;
912                                ;
913 0644 CD 78 06 SQRTA: CALL WRT4 ; GET ARG A
914 0647 3E 01 SQRT: MVI A,01H ; LOAD OP CODE
915 0649 D3 7F OUT APCMD ; ISSUE COMMAND
916 064B C9 RET ; RETURN TO CALLER
917                                ;
918                                ;

```

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 18

```

919 ;
920 ;
921 ;
922 ;
923 ;
924 ;
925 ;
926 ;
927 ;
928 ;
929 ;
930 ;
931 ;
932 064C CD 78 06 SSUBB: CALL WRT2 ; GET ARG B
933 064F EB XCHG ; SWAP MEM POINTERS
934 0650 CD 78 06 SSUBA: CALL WRT2 ; GET ARG A
935 0653 3E 6D SSUB: MVI A,6DH ; LOAD OP CODE
936 0655 D3 7F OUT APCMD ; ISSUE COMMAND
937 0657 C9 RET ; RETURN TO CALLER
938 ;
939 ;
940 ;
941 ;
942 ;
943 ;
944 ;
945 ;
946 ;
947 ;
948 ;
949 ;
950 ;
951 ;
952 ;
953 0658 CD 78 06 TANA: CALL WRT4 ; GET ARG A
954 065B 3E 04 TAN: MVI A,04H ; LOAD OP CODE
955 065D D3 7F OUT APCMD ; ISSUE COMMAND
956 065F C9 RET ; RETURN TO CALLER
957 ;
958 ;
959 ;
960 ;
961 ;
962 ;
963 ;
964 ;
965 ;
966 ;
967 ;
968 ;
969 ;
970 ;
971 ;
972 ;

```

ROUTINE SSUB - 16-BIT, FIXED-POINT SUBTRACT

R <-- B-A

DESTROYS: A,D,E,H,L

ROUTINE TAN - 32-BIT, FLOATING-POINT TANGENT

R <-- TAN(A)

DESTROYS: A,H,L

ROUTINE TOS4 - STORE THE APU TOS INTO MEMORY LOC ADDRESSED BY H,L. THE MSB WILL GO INTO THE INITIAL LOC AND H,L WILL BE INCREMENTED AS SUCCEEDING BYTES ARE STORED, RESULTING IN H,L POINTING TO THE LSB OF THE VALUE ON RETURN.

DESTROYS: A,H,L

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 19

973 0660 DB 7E
 974 0662 77
 975 0663 23
 976 0664 DB 7E
 977 0666 77
 978 0667 23
 979 0668 DB 7E
 980 066A 77
 981 066B 23
 982 066C DB 7E
 983 066E 77
 984 066F C9

TOS4: IN APDATA ; GET MSB OF TOS
 MOV M,A ; STORE IN MEMORY
 INX H ; GET POINTER TO NEXT BYTE
 IN APDATA ; REPEAT FOR ALL 4 BYTES
 MOV M,A
 INX H
 TOS2: IN APDATA ; ENTRY POINT FOR TWO BYTES.
 MOV M,A
 INX H
 TOS1: IN APDATA ; ENTRY POINT FOR A SINGLE BYTE.
 MOV M,A
 RET ; RETURN TO CALLER

;
 ;*****

ROUTINE WRT4 - TRANSFER 32-BIT QUANTITY FROM LOCATION H,L TO
 THE APU TOS. HL IS DECREMENTED TO OBTAIN THE MORE SIGNIFI-
 CANT BYTES. FOUR,TWO,OR ONE BYTES ARE LOADED.

DESTROYS: A,H,L

;
 ;*****

999 0670 7E
 1000 0671 D3 7E
 1001 0673 2B
 1002 0674 7E
 1003 0675 D3 7E
 1004 0677 2B
 1005 0678 7E
 1006 0679 D3 7E
 1007 067B 2B
 1008 067C 7E
 1009 067D D3 7E
 1010 067F C9

WRT4: MOV A,M ; GET LSB FROM RAM
 OUT APDATA ; WRITE ONTO STACK
 DCX H ; NEW POINTER
 MOV A,M ; REPEAT FOR ALL 4 BYTES
 OUT APDATA
 DCX H
 WRT2: MOV A,M ;ENTRY POINT FOR 2 BYTES
 OUT APDATA
 DCX H
 WRT1: MOV A,M ;ENTRY POINT FOR SINGLE BYTE
 OUT APDATA
 RET ; RETURN TO CALLER

;
 ;
 ;*****

ROUTINE XCHD - EXCHANGE THE 32-BIT APU TOS AND NOS

DESTROYS: A

;
 ;*****

1024 0680 3E 39
 1025 0682 D3 7F
 1026 0684 C9

XCHD: MVI A,39H ; LOAD OP CODE
 OUT APCMD ; ISSUE COMMAND
 RET ; RETURN TO CALLER


```

ERR LINE  ADDR  B1 B2 B3 B4      MATH9 ASSEMBLY VER 9 APRIL 1979      PAGE  20

1027      ;
1028      ;
1029      ;*****
1030      ;
1031      ;
1032      ;      ROUTINE XCHS - EXCHANGE 16-BIT TOS AND NOS
1033      ;
1034      ;      DESTROYS:  A
1035      ;
1036      ;
1037      ;*****
1038      ;
1039      ;
1040      0685  3E 79      XCHS:  MVI    A,79H    ; LOAD OP CODE
1041      0687  D3 7F      OUT     APCMD  ; ISSUE COMMAND
1042      0689  C9          RET     ; RETURN TO CALLER
1043      ;
1044      ;
1045      ;*****
1046      ;
1047      ;      CONSTANTS
1048      ;
1049      ;*****
1050      ;
1051      068A  7D CC CC CC  FACT01:DB    07DH,0CCH,0CCH,0CCH  ;0.1 IN FLOATING POINT BINARY
1052      068E  04 A0 00 00  FACT10:DB    04H,0A0H,00H,00H    ;10 IN FLOATING POINT BINARY
1053      ;
1054      ;*****
1055      ;
1056      ;*****
1057      ;*****
1058      ;
1059      ;
1060      ;      END OF MATH9
1061      ;
1062      ;
1063      ;*****
1064      ;*****
1065      ;
1066      ;
1067      ;
1068      ;      UTILITY OUTPUT ROUTINE
1069      ;      TAKE A FLOATING POINT NUMBER FROM THE TOP OF THE
1070      ;      9511 STACK, CONVERT IT TO DECIMAL AND PRINT IT ON
1071      ;      THE SERIAL OUTPUT DEVICE.
1072      ;
1073      0692  CD 78 04      PRINT: CALL  BIDECE  ;CONVERT FLOATING POINT NO TO DECIMAL
1074      0695  0E 20      MVI     C,020H  ;LOAD A SPACE INTO C REG.
1075      0697  3A 00 3C      LDA     EXP10  ;LOAD EXPONENT INTO ACCUMULATOR
1076      069A  E6 00      ANI     MASK   ;LOOK AT SIGN OF NUMBER
1077      069C  C4 FA 06      CNZ     MINUS  ;MINUS LOADS A MINUS SIGN INTO C REG
1078      069F  CD 00 00      CALL    CO     ;OUTPUT A SPACE IF +, A - IF NEG
1079      06A2  21 0D 3C      LXI     H,DECNO+3;ADDRESS OF HIGHEST ORDER DECIMAL NO LOADED
1080      06A5  7E          MOV     A,M     ;LOADED INTO A

```

ERR LINE	ADDR	B1	B2	B3	B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 21
1081	06A6	E5				PUSH H ;H,L SAVED	
1082	06A7	F5				PUSH PSW ;NUMBER IN A SAVED	
1083	06A8	0F				RRC	
1084	06A9	0F				RRC	
1085	06AA	0F				RRC	
1086	06AB	0F				RRC	
1087	06AC	E6 0F				ANI 0FH ;SELECT FIRST DIGIT	
1088	06AE	4F				MOV C,A ;LOAD INTO C	
1089	06AF	CD 00 00	E			CALL PRVAL ;CONVERT TO ASCII	
1090	06B2	CD 00 00	E			CALL ECHO ;SEND TO TERMINAL	
1091	06B5	0E 2E				MVI C,02EH ;LOAD DECIMAL POINT	
1092	06B7	CD 00 00	E			CALL CO ;OUTPUT CHARACTER IN C	
1093	06BA	F1				POP PSW ;RESTORE A	
1094	06BB	E6 0F				ANI 0FH ;GET SECOND DIGIT	
1095	06BD	4F				MOV C,A ;CONVERT TO ASCII	
1096	06BE	CD 00 00	E			CALL PRVAL	
1097	06C1	CD 00 00	E			CALL ECHO ;PRINT OUT	
1098	06C4	E1				POP H	
1099	06C5	2B				DCX H	
1100	06C6	7E				MOV A,M ;LOAD NEXT SET	
1101	06C7	CD 00 00	E			CALL NMOUT ;PRINT NEXT SET	
1102	06CA	2B				DCX H	
1103	06CB	7E				MOV A,M	
1104	06CC	CD 00 00	E			CALL NMOUT	
1105	06CF	2B				DCX H	
1106	06D0	7E				MOV A,M	
1107	06D1	CD 00 00	E			CALL NMOUT	
1108	06D4	0E 20				MVI C,020H ;LOAD ANOTHER BLANK	
1109	06D6	C5				PUSH B	
1110	06D7	CD 00 00	E			CALL CO ;PRINT IT	
1111	06DA	0E 45				MVI C,045H ;LOAD AN E	
1112	06DC	CD 00 00	E			CALL CO	
1113	06DF	3A 00 3C				LDA EXP10 ;LOAD EXPONENT BACK IN	
1114	06E2	E6 7F				ANI 07FH ;DUMP SIGN BIT	
1115	06E4	D6 19				SUI 019H ;SUBTRACT OFFSET OF 20 +7 THAT WAS SHIFTED	
1116	06E6	C1				POP B ;RESTORE C	
1117	06E7	DC FA 06				CC MINUS ;LOAD A MINUS SIGN IF EXPONENT WAS NEGATIVE	
1118	06EA	DC FD 06				CC COMPAC ;COMPLIMENT ACCUMULATOR IF IT WAS NEGATIVE	
1119	06ED	F5				PUSH PSW ;SAVE EXPONENT	
1120	06EE	CD 00 00	E			CALL CO ;PRINT BLANK OR SIGN	
1121	06F1	F1				POP PSW ;RESTORE EXPONENT	
1122	06F2	4F				MOV C,A ;SAVE EXPONENT IN C REGISTER	
1123	06F3	AF				XRA A ;ZERO A,CLEAR CARRY	
1124	06F4	09				ADC C ;BRING EXPONENT BACK INTO A	
1125	06F5	27				DAA ;DECIMAL ADJUST ACCUMULATOR-CONVERT EXP TO DEC	
1126	06F6	CD 00 00	E			CALL NMOUT ;PRINT EXPONENT	
1127	06F9	C9				RET	
1128	06FA	0E 2D				MINUS: MVI C,'-' ;LOAD A MINUS SIGN INTO C REG	
1129	06FC	C9				RET	
1130	06FD	2F				COMPAC: CMA ;COMPLIMENT ACCUMULATOR	
1131	06FE	3C				INR A ;INCREMENT THE ACCUMULATOR (TAKE NEG OF NEG NO)	
1132	06FF	C9				RET	
1133							
1134						ROUTINE EPRNT	

ERR LINE	ADDR	B1	B2	B3	B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 22
1135						; PRINTS NUMBERS IN E FORMAT - STARTING AT RAM ADDRESSED BY HL	
1136						; D REG CONTAINS THE NUMBERS PER LINE,E REG NO. OF LINES	
1137						;	
1138	0700	D5				EPRNT PUSH D COPY OF NUM/LINE,NUM LINES SAVED	
1139	0701	23				EPRNL INX H HL INCREMENTED 3 TIMES BEFORE WRT4 CALL	
1140	0702	23				INX H	
1141	0703	23				INX H	
1142	0704	E5				PUSH H ON STACK SINCE WRT4,EPRIN,OVERWRT HL	
1143	0705	D5				PUSH D D ALSO COPIED AGAIN	
1144	0706	CD 70 06				CALL WRT4 DATA AT HL WRITTEN ON STACK	
1145	0709	CD F1 05				CALL PTOD PUSH STACK,SETS STATUS REG IN AP	
1146	070C	CD 92 06				CALL PRINT NUMBER PRINTED IN E FORMAT	
1147	070F	0E 20				MVI C,020H SPACE LOADED IN C	
1148	0711	CD 00 00	E			CALL CO SPACE IN C PRINTED	
1149	0714	0E 20				MVI C,020H 2ND SPACE LOADED	
1150	0716	CD 00 00	E			CALL CO	
1151	0719	D1				POP D DE RESTORED	
1152	071A	E1				POP H HL ALSO RESTORED FROM STACK	
1153	071B	23				INX H INCREMENT HL TO POINT TO NEXT RAM NUMBER	
1154	071C	15				DCR D TEST D TO SEE IF WE ARE DONE WITH LINE	
1155	071D	C2 01 07				JNZ EPRNL STAY IN LINE LOOP UNTIL LINE IS DONE	
1156	0720	CD 00 00	E			CALL CROUT LINEFEED,CARRIAGE RETURN	
1157	0723	D1				POP D LOOK AT NEXT LINE	
1158	0724	1D				DCR E DECREMENT NO OF LINES COUNTER	
1159	0725	C2 00 07				JNZ EPRNT DO NEXT LINE IF NOT DONE	
1160	0728	C9				RET RETURN TO CALLING PROGRAM	
1161						;	
1162						; ROUTINE FPRNT	
1163						; PRINTS NUMBERS IN F FORMAT - STARTING AT RAM ADDRESSED BY HL	
1164						; D REGISTER CONTAINS THE NUMBER PER LINE,E REG NO. PER LINE	
1165						; THE NO OF DIGITS IS IN B,THE DIGITS AFTER THE DEC IN C	
1166						;	
1167	0729	D5				FPRNT PUSH D COPY OF NUM/LINE,NUM LINES SAVED	
1168	072A	23				FPRNL INX H HL INCREMENTED 3 TIMES BEFORE WRT4 CALL	
1169	072B	23				INX H	
1170	072C	23				INX H	
1171	072D	E5				PUSH H ON STACK SINCE WRT4,EPRIN,OVERWRT HL	
1172	072E	D5				PUSH D D ALSO COPIED AGAIN	
1173	072F	C5				PUSH B COPY OF BC STORED ON STACK	
1174	0730	CD 70 06				CALL WRT4 DATA AT HL WRITTEN ON STACK	
1175	0733	CD F1 05				CALL PTOD PUSH STACK,SETS STATUS REG IN AP	
1176	0736	CD 4C 07				CALL PRINTF NUMBER PRINTED IN F FORMAT	
1177	0739	C1				POP B BC REGISTERS RESTORED	
1178	073A	D1				POP D DE RESTORED	
1179	073B	E1				POP H HL ALSO RESTORED FROM STACK	
1180	073C	23				INX H INCREMENT HL TO POINT TO NEXT RAM NUMBER	
1181	073D	15				DCR D TEST D TO SEE IF WE ARE DONE WITH LINE	
1182	073E	C2 2A 07				JNZ FPRNL STAY IN LINE LOOP UNTIL LINE IS DONE	
1183	0741	C5				PUSH B SAVE BC SINCE CROUT WILL DESTROY THEM	
1184	0742	CD 00 00	E			CALL CROUT LINEFEED,CARRIAGE RETURN	
1185	0745	C1				POP B RESTORE BC REGISTERS	
1186	0746	D1				POP D LOOK AT NEXT LINE	
1187	0747	1D				DCR E DECREMENT NO OF LINES COUNTER	
1188	0748	C2 29 07				JNZ FPRNT DO NEXT LINE IF NOT DONE	

ERR LINE	ADDR	B1	B2	B3	B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 23
1189	0740	C9				RET RETURN TO CALLING PROGRAM	
1190						:	
1191						:	
1192						PRINTF PROGRAM TO PRINT A FLOATING POINT NUMBER ON THE TOP	
1193						OF THE APSTACK AND PRINT IT IN FIXED FORMAT	
1194						THE NUMBER OF DIGITS IS CONTAINED IN B, THE NUMBER OF DIGITS	
1195						AFTER THE DECIMAL POINT IS IN C.	
1196	074C	C5				PRINTF PUSH B STORE NO DIGITS, NO DIGITS AFTER DEC PT ON	
1197	074D	CD	78	04		CALL BIDECE CONVERT B1 FLT PT TO DEC FLT PT	
1198	0750	21	0D	3C		LXI H, DECNO+3 ADDRESS OF HIGHEST ORDER DEC NO IN HL	
1199	0753	11	01	3C		LXI D, BINNO NUMBER WILL BE COPIED TO HERE	
1200	0756	0E	04			MVI C, 04H SET C COUNTER TO DO 4 BYTES	
1201	0758	3A	00	3C		LDA EXP10 LOAD EXPONENT INTO A	
1202	075B	E6	7F			ANI 07FH DUMP SIGN	
1203	075D	47				MOV B, A AND STORE IN REG B	
1204	075E	7E				MOV A, M MOVE DEC NO INTO ACCUMULATOR	
1205	075F	0F				RRC ROTATE RIGHT 4 TIMES TO LINE UP NUMBER	
1206	0760	0F				RRC	
1207	0761	0F				RRC	
1208	0762	0F				RRC	
1209	0763	E6	0F			ANI 0FH MASK OFF HI BYTE	
1210	0765	CA	6B	07		JZ CONT3 JUMP AROUND IF 1ST HI BYTE WAS ZERO	
1211	0768	12				STAX D NUMBER WAS NOT ZERO, STORE IT	
1212	0769	13				INX D AND INCREMENT D	
1213	076A	04				INR B INR EXPT, DEC PT WILL BE TO LFT OF 1ST NO	
1214	076B	7E				MOV A, M LOAD ACC FROM MEM ADDR BY HL	
1215	076C	E6	0F			ANI 0FH DUMP 4 HI BITS	
1216	076E	12				STAX D STORE IN RAM	
1217	076F	04				INR B INR EXPT, DEC PT WILL BE TO LFT OF 1ST NO	
1218	0770	0D				DCR C CHECK TO SEE IF WE ARE DONE	
1219	0771	CA	83	07		JZ OUT1 JUMP OUT OF LOOP WHEN DONE	
1220	0774	2B				DCX H DECREMENT HL COUNTER	
1221	0775	13				INX D POINT DE TO NEXT LOCATION	
1222	0776	7E				MOV A, M LOAD NEXT BYTE OF DECIMAL NO	
1223	0777	0F				RRC ROTATE RIGHT 4 TIMES TO LINE UP BITS	
1224	0778	0F				RRC WHEN DONE, DECIMAL NUMBER WILL BE STORED	
1225	0779	0F				RRC STARTING WITH THE MOST SIGNIFICANT DIGIT	
1226	077A	0F				RRC AT BINNO, TO L.S. DIGIT AT BINNO+8 OR 9	
1227	077B	E6	0F			ANI 0FH DUMP HI BITS	
1228	077D	12				STAX D STORE AT RAM ADDRESSED BY DE	
1229	077E	04				INR B INR EXPT, DEC PT WILL BE TO LFT OF 1ST NO	
1230	077F	13				INX D INCREMENT DE COUNTER	
1231	0780	C3	6B	07		JMP CONT3 JUMP BACK	
1232	0783	AF				XRA A ZERO ACCUMULATOR	
1233	0784	13				INX D A COUPLE OF EXTRA ZEROS WILL BE ADDED	
1234	0785	12				STAX D TO THE RAM LOCATIONS IN CASE THE	
1235	0786	13				INX D NUMBER NEEDS PADDING OR THE HL COUNTER	
1236	0787	12				STAX D OVER RUNS THE 7 OR 8 DIGITS SET	
1237	0788	50				MOV E, B EXPONENT 1112011 IN E REGISTER	
1238	0789	C1				POP B BC RESTORED	
1239	078A	21	01	3C		LXI H, BINNO HL ARE ADDRESSING 1ST NUMBER	
1240	078D	70				MOV A, B MOVE NO DIGITS INTO ACC	
1241	078E	91				SUB C * DIG-(#DIG AFT DEC + 1) IN ACC	
1242	078F	C6	1F			ADI 01FH B-C+01FH IN ACCUMULATOR	

ERR LINE	ADDR	B1 B2 B3 B4	MATH9 ASSEMBLY VER 9 APRIL 1979	PAGE 24
1243	0791	57	MOV D,A	AND STORED IN D
1244	0792	BB	CMP E	IF EXP+1+20.GT.B-C+1F,CARRY SET;OVERFLO
1245	0793	DC 07 07	CC PRST	PRINT OVERFLOW (*) AND DUMP HI DIGITS
1246			MAIN LOOP,TEST EXPONENT AGAINST DIGIT COUNT(E VS D)	
1247	0796	7A	XPTST	B-C+1F MOVED TO ACC
1248	0797	BB	CMP E	COMPARE B-C+1F TO EXP+1+20
1249	0798	DA E8 07	JC PRTHO	IF EXP+1+20.GT.B-C+1F,PRINT NO
1250	079B	CA D3 07	JZ PRTHO	IF EXP+1+20.EQ.B-C+1F PRINT SIGN
1251			EXP+1+20 WAS LT B-C-1F;TEST B-C-1F IN ACC AGAINST 021H	
1252	079E	FE 21	CPI 021H	CARRY SET IF D.LT.21,0 SET IF D.EQ.21
1253	07A0	CA D3 07	JZ PRTHO	PRINT THE SIGN IF D.EQ.21
1254	07A3	D2 C3 07	JNC PRTHO	PRINT A BLANK IF D.GT.21
1255	07A6	0E 30	MVI C,030H	ASCII REP OF 0 IN C
1256	07A8	CD 00 00	CALL CO	OUTPUT ASCII CHARACTER IN REG C
1257	07AB	7A	DECTS	TEST FOR DEC PT,LOAD B-C+1F INTO ACC
1258	07AC	FE 20	CPI 020H	WILL PRINT DEC PT WHEN B-C+1F=20
1259	07AE	CA CB 07	JZ PRTHO	PRINT DECIMAL POINT IF B.EQ C+1
1260	07B1	15	DCRMB	DECREMENT B-C+1F,DIGIT COUNT IN D
1261	07B2	05	DCR B	DECREMENT DIGIT COUNT IN B
1262	07B3	C2 96 07	JNZ XPTST	GO BACK TO MAIN LOOP IF NOT DONE
1263	07B6	C9	RET	DONE,RETURN TO CALLING PROG
1264				
1265	07B7	23	PRTHO	H,L REG INCREMENTED,DUMPING HI DIGIT
1266	07B8	1D	DCR E	EXPONENT DECREMENTED FOR NEXT DIGIT
1267	07B9	BB	CMP E	NEW EXP+1+20 COMP TO B-C+20,CAR IF .GT.
1268	07BA	DA B7 07	JC PRTHO	STAY IN LOOP,DUMPING HI DIGITS UNTIL D=E
1269	07BD	0E 2A	MVI C,02AH	ASCII REP OF * IN C
1270	07BF	CD 00 00	CALL CO	OUTPUT ASCII CHAR IN REG C
1271	07C2	C9	RET	RETURN TO CALLING ROUTINE
1272				
1273	07C3	0E 20	PRTHO	ASCII REP OF BLANK IN C
1274	07C5	CD 00 00	CALL CO	OUTPUT ASCII CHAR IN REG C
1275	07C8	C3 B1 07	JMP DCRMB	JUMP TO DECREMENT B
1276				
1277	07CB	0E 2E	PRTHO	ASCII REP OF . IN C
1278	07CD	CD 00 00	CALL CO	OUTPUT ASCII CHAR IN REG C
1279	07D0	C3 B1 07	JMP DCRMB	JUMP TO DECREMENT B
1280				
1281	07D3	FE 21	PRTHO	COMPARE B-C+1F TO 21,IF D.LT.21 GO TO PRTHO
1282	07D5	DA A6 07	JC PRTHO	PRINT ZERO;DIGIT B-C.LT.1
1283	07D8	0E 20	MVI C,020H	LOAD ASCII REP OF BLANK IN C
1284	07DA	3A 00 3C	LDA EXP10	LOAD EXPONENT INTO ACCUMULATOR
1285	07DD	E6 00	ANI 000H	LOOK AT SIGN OF NUMBER
1286	07DF	C4 FA 06	CMZ MINUS	MINUS LOADS A - IN C REG ,OVERWRITING BLA
1287	07E2	CD 00 00	CALL CO	OUTPUT ASCII CHAR IN REG C
1288	07E5	C3 B1 07	JMP DCRMB	JUMP TO DECREMENT B
1289				
1290	07E8	7E	PRTHO	LOAD ACC FROM HL ADDR IN RAM
1291	07E9	C6 30	ADI 030H	CONVERT TO ASCII REP OF DEC NO
1292	07EB	4F	MOV C,A	LOAD INTO REG C
1293	07EC	CD 00 00	CALL CO	PRINT ASCII CHARACTER IN REG C
1294	07EF	23	INX H	MOVE HL POINTER TO NEXT LOCATION
1295	07F0	C3 AB 07	JMP DECTS	JUMP BACK TO TEST FOR DECIMAL POINT
1296				

ERR LINE ADDR B1 B2 B3 B4

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 25

1297 07F3

END

ASSEMBLER ERRORS = 0

MATH9 ASSEMBLY VER 9 APRIL 1979

PAGE 26

SYMBOL TABLE

A	0007	ACOS	0463	ACOSA	0460	APCMD	007F
APDATA	007E	APSTS	007F	ASIN	046B	ASINA	046B
ATAN	0473	ATANA	0470	B	0000	BIDEC	04B9
BIDECF	0470	BINNO	3C01	C	0001	CHECK	048A
CHSD	0521	CHSDA	051E	CHSF	0529	CHSFA	0526
CHSS	0531	CHSSA	052E	CO	E 0000	COMPAC	06FD
CONT	0480	CONT1	04A6	CONT2	04C5	CONT3	076B
COS	0539	COSA	0536	CROUT	E 0001	D	0002
DADD	0545	DADDA	0542	DADDB	053E	DCRMB	07B1
DDIV	0551	DDIVA	054E	DDIVB	054A	DECNO	3C0A
DECTS	07A8	DIVIDE	04A2	DMUL	055D	DMULA	055A
DMULB	0556	DMUJ	0569	DMUJA	0566	DMUJB	0562
DSUB	0575	DSUBA	0572	DSUBB	056E	DUBLE	3C05
E	0003	ECHO	E 0002	EPRHL	0701	EPRNT	0700
ERROR	0502	EXP	057D	EXP10	3C00	EXPA	057A
FACT01	060A	FACT10	060E	FADD	0509	FADDA	0506
FADDB	0502	FDIV	0595	FDIVA	0592	FDIVB	050E
FIXD	05A5	FIXDA	05A2	FIXS	059D	FIXSA	059A
FLTD	05AD	FLTDA	05AA	FLTS	05B5	FLTSA	05B2
FMUL	05C1	FMULA	05BE	FMULB	05BA	FPRNL	072A
FPRNT	0729	FSUB	05CD	FSUBA	05CA	FSUBB	05C6
H	0004	L	0005	LN	05DD	LNA	05DA
LOG	05D5	LOGA	05D2	LOOP	0513	LOOP4	04DA
LOOP5	04E0	M	0006	MASK	0000	MDAD	0512
MEMORY	M 0000	MINUS	06FA	MLTPLY	049B	NMOUT	E 0003
NOP	05E2	OUT1	0703	POPD	05E7	POPS	05EC
PRINT	0692	PRINTF	074C	PRTBL	07C3	PRTDP	07CB
PRTNO	07E0	PRTSN	07D3	PRTST	07B7	PRTZR	07A6
PRVAL	E 0004	PSW	0006	PTOD	05F1	PTOS	05F6
PUP1	05FB	PWR	0607	PWRA	0604	PWRB	0600
SADD	0613	SADDA	0610	SADDB	060C	SDIV	061F
SDIVA	061C	SDIVB	0618	SIN	0627	SINA	0624
SMUL	0633	SMULA	0630	SMULB	062C	SMUJ	063F
SMUJA	063C	SMUJB	0638	SP	0006	SQRT	0647
SQRTA	0644	SSUB	0653	SSUBA	0650	SSUBB	064C
STACK	S 0000	TAN	065B	TANA	0658	TOS1	066C
TOS2	0660	TOS4	0660	WRT1	067C	WRT2	0678
WRT4	0670	XCHD	0600	XCHS	0605	XPTST	0796
ZERO	050B						

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

APPENDIX F

MATH CONVERSION PROGRAM

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table Fl. Math Conversion Program

```

1      PROGRAM MATHCON
2      DIMENSION NCONV(4)
3      20 WRITE(1,30)
4      30 FORMAT(' ? ')
5      READ(1,*)XNUM
6      CALL CONVR9(XNUM,1,NCONV)
7      WRITE(5,50)XNUM,(NCONV(II),II=1,4)
8      50 FORMAT(1X,F15.5,2X,Z2,1X,Z2,1X,Z2,1X,Z2)
9      IF (XNUM.LT.1.E40)GO TO 20
10     STOP JOB
11     END
12     SUBROUTINE CONVR9(XNUM,NTYPE,NCONV)
13     INTEGER*2 NEXP1,NHI1,NMD1,NLO1
14     DIMENSION NCONV(4)
15 C
16 C     SUBROUTINE TO CONVERT MODCOMP FLOATING POINT NUMBERS TO THE
17 C     HEX REPRESENTATION OF THE FLOATING POINT BINARY THAT THE NDT-COMP9
18 C     USES.  THE FORM IS EXPONENT, FRACTION WITH THE FIRST TWO NUMBERS
19 C     THE EXPONENT AND THE LAST SIX THE FRACTION.  VERSION 16 11 77
20 C
21     ZNUM=XNUM
22     INLINE
23     LDMD,6          ZNUM      LOAD ZNUM INTO REG 6&7
24     LLQ,4,1          SHIFT QUAD REG 4-7 TO LEFT
25     LAD,6,2          ARTH SHIFT DOUB REG TO LEFT
26     ADI,6            #8000    ADD EXPONENT OFFSET TO HI BYTE
27     RLQ,4,1          SHIFT QUAD REG TO RT,DUMP 2 EXPT BIT
28     TBRB,6,0        SKIP     SKIP AROUND IF ZNUM WAS NEG
29     BRU              RETN     GO TO RETURN IF NO WAS POS
30 SKIP ABR,6,0        INCREMENT BIT 0,SIGN BIT
31     TTRD,6,6        TWOS COMPLEMENT OF 6,7-CHANGS TO POS
32 RETN TBRB,6,8        NZRO     TESTS BIT 8,XFERS IF NONZERO
33     LDI,6            #0000    LOADS B000 IN REG 6 IF IT WAS 0
34 NZRO MBR,2,6        THE EXPONENT IS TRANSFERRED TO 2
35     STM,2            NEXP1    AND STORED IN MEMORY.
36     MLR,2,6        THE HI ORDER MANTISSA IS XFERED TO I
37     STM,2            AND STORED.
38     MBR,2,7        NHI1      THE MID ORDER MANTISSA IS MOVED TO I
39     STM,2            AND STORED.
40     MLR,2,7        NMD1      THE LOW ORDER BYTE IS MOVED TO 2
41     STM,2            NLO1    AND STORED.
42     FINI
43     NCONV(1)=NEXP1
44     NCONV(2)=NHI1
45     NCONV(3)=NMD1
46     NCONV(4)=NLO1
47     RETURN
48     END
TOTAL RECORDS WRITTEN = 49

```

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

APPENDIX G

NDT-COMP9 PARTS LIST

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table G1. NDT-COMP9 Parts List

Component Designation	Device ^a	Description
<u>I. Integrated Circuits</u>		
A1	8253	Programmable interval timer-real-time clock
A2, A3, A4	8255A/AM9555	Programmable peripheral interface
A12	8251A	Programmable communication interface
A11	SN74LS00	Quad positive-logic nand gate
A10	8205/AM25LS138	3-line to 8-line decoder-demultiplexer
A21	Intel 2716/TI TMS 2516	2K × 8-bit UV erasable PROM (preprogrammed with monitor and MATH9)
A17, A15, A16, A18, A19, A20		2K × 8-bit EPROM or RAM memory (user defined)
A9	8238	System controller and bus driver with advanced IOW/MEMW
A8	8080A/AM9080A	Central processing unit (CPU) MOS microprocessor
A13	AM9511	Arithmetic processing unit (APU)
A14	TMS 4016/MOSTEK 4118	2K × 8-bit RAM or modified 1K × 8-bit RAM
A7	8224	Clock generator and driver
A6	SN75150	Dual line driver, TTL to RS-232-C
A5	μA9637A	Dual differential line receiver, RS-422 and RS-423 to TTL

Table G1. (Continued)

Component Designation	Device ^a	Description
<u>II. Diodes</u>		
D1	1N4446	General-purpose silicon
<u>III. Resistors</u>		
R1	47 Ω	1/4 W, 1% metal film
R2, R3	1K Ω	1/4 W, 1% metal film
R4	560 K Ω	1/4 W, 1% metal film
<u>IV. Capacitors</u>		
C2, C4, C6, C9, C13-C17, C22-C31,	0.01 μ f	15 V, ceramic bypass
C10, C12, C19, C21, C32	0.01 μ f	25 V, ceramic bypass
C3, C5	1 μ f	15 V, tantalum
C1, C11, C18	22 μ f	25 V, tantalum
C20	10 pf	500 V, silver mica
<u>V. Crystal</u>		
CR1	18.432 MHz	Intel
<u>VI. Connectors</u>		
J1, J2, J3, J4	Card-edge connectors formed on printed circuit board	
4 each	40-pin I. C. socket	
2 each	28-pin I. C. socket	
10 each	24-pin I. C. socket	
3 each	16-pin I. C. socket	
1 each	14-pin I. C. socket	
<u>VII. Printed Circuit Board</u>		
1 each	NDT-COMP9A ECI-1592/3	Double sided with plated-through holes, gold plated

^aI.C. = integrated circuit.

APPENDIX H

HEXADECIMAL ASCII CODE TABLE

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

Table H1. ASCII Code Table

Hex Code	ASCII Character	Hex Code	ASCII Character	Hex Code	ASCII Character
00	NUL	2B	+	56	V
01	SOH	2C	,	57	W
02	STX	2D	-	58	X
03	ETX	2E	.	59	Y
04	EOT	2F	/	5A	Z
05	ENQ	30	0	5B	[
06	ACK	31	1	5C	\
07	BEL	32	2	5D	]
08	BS	33	3	5E	^
09	HT	34	4	5F	_
0A	LF	35	5	60	`
0B	VT	36	6	61	a
0C	FF	37	7	62	b
0D	CR	38	8	63	c
0E	SO	39	9	64	d
0F	SI	3A	:	65	e
10	DLE	3B	;	66	f
11	DC1	3C	<	67	g
12	DC2	3D	=	68	h
13	DC3	3E	>	69	i
14	DC4	3F	?	6A	j
15	NAK	40	@	6B	k
16	SYN	41	A	6C	l
17	ETB	42	B	6D	m
18	CAN	43	C	6E	n
19	EM	44	D	6F	o
1A	SUB	45	E	70	p
1B	ESC	46	F	71	q
1C	FS	47	G	72	r
1D	GS	48	H	73	s
1E	RS	49	I	74	t
1F	US	4A	J	75	u
20	SP	4B	K	76	v
21	!	4C	L	77	w
22	"	4D	M	78	x
23	#	4E	N	79	y
24	\$	4F	O	7A	z
25	%	50	P	7B	{
26	&	51	Q	7C	
27	~	52	R	7D	}
28	(	53	S	7E	~
29	)	54	T	7F	DEL
2A	*	55	U		

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

NUREG/CR-1548
 ORNL/NUREG/TM-390
 Distribution
 Category R5

INTERNAL DISTRIBUTION

1-2. Central Research Library	33. D. O. Hobson
3. Document Reference Section	34. J. M. Jansen, Jr.
4-5. Laboratory Records Department	35. E. H. Lee
6. Laboratory Records, ORNL RC	36. Jar-Shyong Lin
7. ORNL Patent Section	37-38. R. W. McClung
8. Nuclear Safety Information Center	39. A. J. Moorhead
9. O. B. Cavin	40. F. R. Mynatt
10. W. A. Coghlan	41. G. W. Scott
11-15. R. F. Cowan	42. W. A. Simpson
16-25. C. V. Dodd	43. G. M. Slaughter
26. B. G. Eads	44. J. H. Smith
27. L. C. Emerson	45. R. O. Williams
28-32. M. R. Hill	46. H. L. Yakel

EXTERNAL DISTRIBUTION

47. NRC, Office of Nuclear Regulatory Research, Washington, DC 20555
 Program Sponsor
48. DOE, Oak Ridge Operations Office, P.O. Box E, Oak Ridge, TN 37830
 Assistant Manager for Energy Research and Development
- 49-50. DOE, Technical Information Center, P.O. Box 62, Oak Ridge, TN
 37830
- 51-440. For Distribution Category R5 (10 - NTIS)