

Notes on the
SWTPC MP-N Calculator Interface
and the
Calc-1 Program

prepared for.
Dr. Daniel W. Scott
CSCI 490.001

by
Daniel Paul Long
May 8, 1979

Notes on the
SWTPC MP-N Calculator Interface
and the
Calc-1 Program

I assembled an SWTPC MP-N Calculator Interface and implemented it using the Calc-1 Program supplied with the kit. The following are my corrections to the documentation and my observations about the interface board and software.

This interface was bought to perform floating-point arithmetic and for its function capabilities such as SIN, COS, and e^x . My application required an integer truncation function that is not performed by this calculator, so I wrote a small assembly language subroutine to do it. A potentially irritating problem is that the calculator chip does not automatically convert to scientific notation if the numbers become too big to display in floating point. The control program must keep track of the display mode.

My application requires fast numerical processing, so, after observing the calculator's computational speed using the Calc-1 program, I decided that its trigonometric functions were too slow. I developed another approach to the problem using pure assembly language rather than partially relying on peripheral hardware processing.

Though I found no major flaws with the Calc-1 software, I did find some mistakes in its documentation. In Table V, the ASCII to Calculator Instruction Lookup Table, there are three errors: the hex value in the table where MSB is 0 and LSB is D should be 21, where MSB is 1 and LSB is 8 should be 2F, and where MSB is 2 and LSB is E, the hex value should be 0A. I added a one page summary of the Calc-1 instruction set to the documentation because the supplied documentation is somewhat confusing.

The Calc-1 program uses a part of RAM also used by the monitor. This provides problems when Calc-1 is halted to record it on cassette tape and resume after it has been recorded. ^{ed}This can be alleviated by changing the "LDX PARADR" instructions in lines 250, 850, 1660, and 1980 to "LDX ~~##~~800C," or, in hex, "FE A002" to "CE 800C." However, the interface must now always be plugged into I/O port 3. The data at hex address 028A is output to the terminal to clear the screen. The ADM terminals we ^{have} ~~use~~ use a different character to clear the screen, so I changed the data at that address to hex 1A. Calc-1 also uses address modification within itself so that, as a result, it cannot be implemented in ROM.

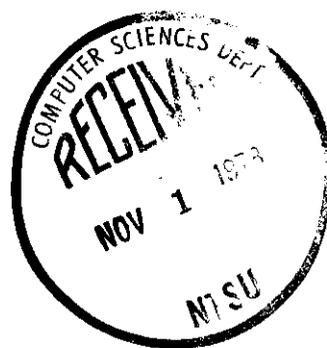
I tested the MP-N interface with the Calc-1 program by making various operand and operator entries and checking the results against the results I obtained

with my own pocket scientific calculator. I noticed
no major discrepancies between the two sets of results.

S W T P C

MP-N Calculator Interface

Documentation



Assembly Instructions MP-N Calculator Interface Kit

The Southwest Technical Products MP-N Calculator Interface interfaces the SWTPC 6800 Computer System thru a peripheral interface Adaptor (PIA) to the National Semiconductor MM57109 Number Oriented Processor. This "processor" is a Reverse Polish Notation (RPN) calculator chip without the internal keypad interfacing circuitry which has made interfacing to calculator chips so difficult in the past. This chip allows data and instruction entry in conventional binary form and speeds entry with the elimination of the debounce circuitry built into conventional calculator chips. It is called a processor because it has instructions and control lines which allow it to operate in conjunction with ROM and RAM as a stand alone numerical processor. It may however be operated as a computer peripheral for numerical calculation and this is the configuration in which the chip has been implemented.

All interfacing from the 6800 Computer System to the calculator chip has been done thru a 6820 PIA. Both the PIA and calculator chip reside on a 3 1/2" X 5 1/4" double sided, plated thru hole circuit board plugged onto one of the seven available interface card positions on the mother board of the 6800 Computer. All data and instructions fed to and all results received from the calculator chip are handled by your own assembler or machine language program. The calculator features reverse Polish notation, floating point or scientific notation, up to an eight digit mantissa and two digit exponent, trig functions, base 10 and natural logarithms, and overflow indicator.

PC Board Assembly

NOTE: Since all of the holes on the PC board have been plated thru, it is only necessary to solder the components from the bottom side of the board. The plating provides the electrical connection from the "BOTTOM" to the "TOP" foil of each hole. Unless otherwise noted it is important that none of the connections be soldered until all of the components of each group have been installed on the board. This makes it much easier to interchange components if a mistake is made during assembly. Be sure to use a low wattage iron (not a gun) with a small tip. Do not use acid core solder or any type of paste flux. We will not guarantee or repair any kit on which either product has been used. Use only the solder supplied with the kit or a 60/40 alloy resin core equivalent. Remember all of the connections are soldered on the bottom side of the board only. The plated-thru holes provide the electrical connection to the top foil.

- () Before installing any parts on the circuit board, check both sides of the board over carefully for incomplete etching and foil "bridges" or "breaks". It is unlikely that you will find any, but should there be one, especially on the "TOP" side of the board, it will be very hard to locate and correct after all of the components have been installed on the board.
- () Starting from one end of the circuit board install each of the three, 10 pin Molex female edge connectors along the lower edge of board. These connectors must be inserted from the "TOP" side of the board and must be pressed down firmly against the circuit board, so that each pin extends completely into the holes on the circuit board. Not being careful here will cause the board to either wobble and/or be crooked when plugging it onto the mother board. It is

either wobble and/or be crooked when plugging it onto the mother board. It is suggested that you solder only the two end pins of each of the three connectors until all have been installed at which time if everything looks straight and rigid you should solder the as yet unsoldered pins.

- () Insert the small nylon indexing plug into the lower edge connector pin indicated by the small triangular arrow on the "BOTTOM" side of the circuit board. This prevents the board from being accidentally plugged on incorrectly.
- () Attach all of the resistors to the board. As with all other components unless noted, use the parts list and component layout drawing to locate each part and install from the "TOP" side of the board bending the leads along the "BOTTOM" side of the board and trimming so that 1/16" to 1/8" of wire remains. Solder.
- () Install the capacitors on the circuit board. Be sure to orient electrolytic capacitor C4 so its polarity matches with that shown on the component layout drawing. Solder.
- () Install the transistor and diode. These components must be oriented to match the component layout drawing. Solder.
- () Install integrated circuit IC2 on the circuit board. This component must be oriented so its metal face is facing the circuit board and is secured to the circuit board with a #4 - 40 X 1/4" screw, lockwasher and nut. A heatsink is not used. The three leads of the integrated circuit must be bent down into each of their respective holds. Solder.

NOTE: MOS integrated circuits are susceptible to damage by static electricity. Although some degree of protection is provided internally within the integrated circuits, their cost demands the utmost in care. Before opening and/or installing any MOS integrated circuits you should ground your body and all metallic tools coming into contact with the leads, thru a 1 M ohm 1/4 watt resistor (supplied with the kit). The ground must be an "earth" ground such as a water pipe, and not the circuit board ground. As for the connection to your body, attach a clip lead to your watch or metal ID bracelet. Make absolutely sure you have the 1 Meg ohm resistor connected between you and the "earth" ground, otherwise you will be creating a dangerous shock hazard. Avoid touching the leads of the integrated circuits any more than necessary when installing them, even if you are grounded. On those MOS IC's being soldered in place, the tip of the soldering iron should be grounded as well (separately from your body ground), either with or without a 1 Meg ohm resistor. Most soldering irons having a three prong line cord plug already have a grounded tip. Static electricity should be an important consideration in cold, dry environments. It is less of a problem when it is warm and humid.

- () Install MOS integrated circuits IC1, IC3, IC4 and IC5 following the precautions given in the preceding section. As they are installed, make sure they are down firmly against the board before soldering all of their leads. Do not bend the leads on the back side of the board. Doing so makes it very difficult to remove the integrated circuit should replacement ever be necessary. The "dot" or "notch" on the end of the package is used for orientation purposes and must match with that shown on the component layout drawing for the IC. Solder.
- () Working from the "TOP" side of the circuit board, fill in all of the feed-thru's with molten solder. The feed-thru's are those unused holes on the

board whose internal plating connects the "TOP" and "BOTTOM" circuit connections. Filling these feed-thru's with molten solder guarantees the integrity of the connections and increases the current handling capability.

- () Now that all of the components have been installed on the board, double check to make sure all have been installed correctly in their proper location.
- () Check very carefully to make sure that all connections have been soldered. It is very easy to miss some connections when soldering which can really cause some hard to find problems later during checkout. Also look for solder "bridges" and "cold" solder joints which are another common problem.

Since the MP-N circuit board now contains MOS devices, it is susceptible to damage from severe static electrical sources. One should avoid handling the board any more than necessary and when you must, avoid touching or allowing anything to come into contact with any of the conductors on the board.

Using the Calculator Interface

Table I gives a complete list and description of the calculator chip's instruction set. Remember that some of the instructions are for stand alone processing systems and are not used on this interface. All numerical entry is in Reverse Polish Notation (RPN) and anyone familiar with Hewlett Packard calculators should have no problem with the data entry sequence. For those not familiar with RPN, the following should be helpful:

To add $7 + 8$, enter the following
7 enter 8 + (4 entries)

The answer is now stored in the X accumulator within the calculator chip
The OUT instruction may be used to output the answer

To find the inverse sine of 0.5, enter the following:
0.5 INV SIN (5 entries)

The answer is now stored in the X accumulator within the calculator chip.
The OUT instruction may be used to output the answer.

In order to simplify the interfacing between your program and the calculator interface, you will probably want to incorporate the following subroutines into your program.

INITAL SUBROUTINE

The INITAL or initialize subroutine configures the PIA interfacing to the calculator chip. This subroutine need only be used once; and is best placed somewhere at the beginning of your program. It is responsible for initializing the data direction registers and control registers of the PIA. The subroutine requires that the index register be loaded with the "lowest" address of the PIA interfacing to the calculator chip prior to execution.

This "lowest" address depends upon which interface port position the MP-N calculator card is plugged. The table below give the "lowest" address of each interface card position.

Address Assignments

PORT0	8000
PORT1	8004
PORT2	8008
PORT3	800C
PORT4	8010
PORT5	8014
PORT6	8018
PORT7	801C

```
86 7F  INITIAL LDA A  #7F  INIT  A SIDE OF PIA
A7 00          STA A  0,X
86 36          LDA A  #36  HIGH HOLD-POS READY
A7 01          STA A  1,X
86 00          LDA A  #00  INIT  B SIDE OF PIA
A7 02          STA A  2,X
86 34          LDA A  #34  NEG  R/W
A7 03          STA A  3,X
A6 02          LDA A  2,X  CLEAR R/W FLAG
89            RTS
```

OUTINS SUBROUTINE

The OUTINS or out instruction subroutine is used to get program data and instructions into the calculator. To send a digit or instruction to the calculator chip, use Table II to find the OP code of the instruction you wish to send. Load this OP code into the A accumulator and jump or branch to the OUTINS subroutine. If you have a string of data you wish to send, just recycle thru this subroutine as many times as necessary. The subroutine takes care of all of the READY and HOLD signals to the calculator chip so there is no worry of sending data faster than the calculator chip can accept it. The subroutine destroys the contents of the B accumulator during execution while the contents of the A accumulator and index register are not destroyed.

```
E6 01  OUTINS LDA B  1,X  WAIT FOR READY
2A FC          BPL   OUTINS
A7 00          STA A  0,X  FORWARD INSTRUCTION TO CALC
E6 00          LDA B  0,X  CLEAR FLAG BIT
C6 3C          LDA B  #3C  LOW HOLD-NEG READY
E7 01          STA B  1,X  BRING HOLD LINE LOW
E6 01  WAIT10 LDA B  1,X
2A FC          BPL   WAIT10  LOOK FOR READY LOW
E6 00          LDA B  0,X  CLEAR FLAG BIT
C6 36          LDA B  #36  HIGH HOLD-POS READY
E7 01          STA B  1,X  RETURN HOLD LINE HIGH
89            RTS
```

SETMEM SUBROUTINE

The SETMEM or set memory subroutine initializes the memory locations to which the calculator's output data will be stored. This subroutine must be executed immediately before the OUTANS subroutine is used. Although it can be changed, memory locations 0020 thru 002B have been designated the temporary storage locations for the calculator's output data. The subroutine sets memory location 0020 to a 00 while locations 21 thru 2B are set to 20 (ASCII spaces). This subroutine destroys the contents of the index register and B accumulator. The contents of the A accumulator are not destroyed.

```

7F 0020 SETMEM CLR    $20      CLEAR $0020
CE 0020          LDX   $$20    BOTTOM OF BUFFER
C6 20          LDA  B  $$20
08          LOOP1  INX
E7 00          STA  B  0,X     STORE A SPACE
8C 002B       CPX   $$2B     CHECK FOR TOP OF BUFFER
26 F8          BNE   LOOP1
39          RTS

```

OUTANS SUBROUTINE

The OUTANS or output answer subroutine outputs the contents of the X register within the calculator chip in BCD to memory locations 0020 thru 002B. Since the mantissa digit count of the calculator is variable, the previous SETMEM subroutine blanks out any digit location not filled by the OUTANS subroutine. It is very important that the SETMEM subroutine be used each time before executing the OUTANS subroutine. The OUTANS subroutine outputs data in two different formats depending upon whether the calculator chip is in the floating point or scientific mode. The calculator initially starts out in the floating point mode where it will remain until changed by the TOGM (22₁₆) instruction. This calculator does not automatically convert to scientific notation if the numbers become too big to handle in floating point as many do. An MCLR (2F₁₆) instruction will always reset the calculator chip to the floating point mode regardless of what mode it was in originally. Since the calculator chip does not tell you what mode it is in when it is outputting data, your program must know so you can process the data accordingly. Table IV shows the format in which the data is stored. At the end of the OUTANS subroutine, the N bit of the condition code register is set if an error has transpired since the last execution of the OUTANS subroutine. You may use a BMI instruction to catch and branch to an error routine to note the error. You should then send an ECLR (2B₁₆) instruction to the calculator chip to reset the calculator chip's error flag. Disregarding the error flag on the calculator chip will cause no problems. The chip will continue to function regardless of the state of the flag. The subroutine requires that the index register be loaded with the "lowest" address of the PIA interfacing to the calculator chip prior to execution. Since the SETMEM subroutine usually run prior to this destroys the contents of the index register, don't forget to reload the index register before branching to the OUTANS subroutine. The OUTANS subroutine destroys the contents of both the A and B accumulators during execution while the contents of the index register is not changed.

E6 01	OUTANS	LDA B	1, X	
2A FC		BPL	OUTANS	
A6 00		LDA A	0, X	CLEAR FLAG BIT
86 16		LDA A	##16	SEND AN OUT
A7 00		STA A	0, X	
C6 3E		LDA B	##3E	LOW HOLD-PDS READY
E7 01		STA B	1, X	BRING HOLD LINE LOW
E6 01	WAIT30	LDA B	1, X	WAIT FOR SECOND READY
2A FC		BPL	WAIT30	
E6 00		LDA B	0, X	CLEAR FLAG BIT
86 0F		LDA A	##0F	
A7 00		STA A	0, X	SEND A NOP
E6 03	WAIT3	LDA B	3, X	LOOK FOR R/W STROBE
2B 06		BMI	OUTDIG	TRANSFER CALC DATA TO MEMORY
E6 01		LDA B	1, X	LOOK FOR READY STROBE
2B 16		BMI	CONFLG	PRINT MEMORY CONTENTS
20 F6		BRA	WAIT3	
A6 02	OUTDIG	LDA A	2, X	LOAD OUT DATA INTO A
16		TAB		
84 0F		AND A	##0F	ELIMINATE UPPER 4 BITS
8A 30		ORA A	##30	CONVERT TO ASCII DATA
54		LSR B		
54		LSR B		
54		LSR B		
54		LSR B		
CA 20		ORA B	##20	INCREMENT ADDRESSES BY #20
F7 01C6		STA B	POINT2+1	STORE OUT DATA SEQUENTIALLY
97 00	POINT2	STA A	\$0	
20 E2		BRA	WAIT3	
86 36	CONFLG	LDA A	##36	HIGH HOLD-PDS READY
A7 01		STA A	1, X	BRING HOLD LINE HIGH
A6 00		LDA A	0, X	CLEAR FLAG BIT
39		RTS		

Number Entry Rules

When a digit, decimal point, or π is entered with an 0-9, DP, or PI instruction, the stack is first pushed and the X register cleared: Z \rightarrow T, Y \rightarrow Z, X \rightarrow Y, 0 \rightarrow X. This process is referred to as "initiation of number entry." Following this, the digit and future digits are entered into the X mantissa. Subsequent entry of digits or DP, EE, or CS instructions do not cause initiation of number entry. Digits following the eighth mantissa digit are ignored. This number entry mode is terminated by any instruction except 0-9, DP, EE, CS, PI, or HALT. Termination of number entry means two things. First, the number is normalized by adjusting the exponent and decimal point position so that the decimal point is to the right of the first mantissa digit. Second, the next digit, decimal point, or π entered will cause initiation of number entry, as already described. There is one exception to the number entry initiation rule. The stack is not pushed if the instruction prior to the entered digit was an ENTER. However, the X register is still cleared and the entered digit put in X.

The ENTER key itself terminates number entry and pushes the stack. The OUT instruction terminates number entry and prepares the stack for pushing upon the next entry of data. This means that if you use the ENTER and OUT instructions consecutively, the stack gets pushed twice which is not what you want. If you wish to ENTER data and immediately OUT the result, use only the OUT instruction. The OUT performs the entry. If you do not wish to OUT the ENTER'ed data, just use the ENTER instruction by itself.

The AIN and IN instructions should not be used for number entry. Provisions have not been made for their use on this interface.

How It Works

Peripheral Interface Adaptor (PIA) IC1 interfaces the MM57109 calculator chip, IC3, to the SWTPC 6800 buss. The first six bits of the A side of the PIA are used to feed instructions to the calculator chip while the eighth is used as an input to monitor the ERROR output of the calculator. Control line CA1 outputs HOLD signals to, while control line CA2 inputs READY signals from the calculator chip. The first four bits of the B side of the PIA are used to input BCD digit data while the last four bits input digit addresses. The CB1 line inputs READ/WRITE signals while the CB2 control line is not used. Hex inverter/buffer, IC4, is used primarily as the 320 to 400 Khz single phase oscillator required by the calculator chip. One section is used to invert the HOLD signal going to the calculator. Shift register IC5 generates the POR signal required for proper startup and initialization. +5 VDC power required by the board is supplied by voltage regulator IC2 while -4 VDC voltage is supplied by transistor Q1 and its associated components. Figure I shows a block diagram for the internal construction of the calculator chip.

Parts List MP-N Calculator Interface

Resistors

<u>✓</u>	R1	47K ohm 1/4 watt resistor
<u>✓</u>	R2	1K " " "
<u>✓</u>	R3	10K " " "
<u>✓</u>	R4	10K " " "
<u>✓</u>	R5	10K " " "
<u>✓</u>	R6	10K " " "
<u>✓</u>	R7	10K " " "
<u>✓</u>	R8	22K " " "
<u>✓</u>	R9	22K " " "
<u>✓</u>	R10	22K " " "
<u>✓</u>	R11	22K " " "
<u>✓</u>	R12	12K " " "
<u>✓</u>	R13	27 " " "
<u>✓</u>	R14	3.3K " " "
<u>✓</u>	R15	10K " " "
<u>✓</u>	R16	47K " " "
<u>✓</u>	R17	10K " " "

Capacitors

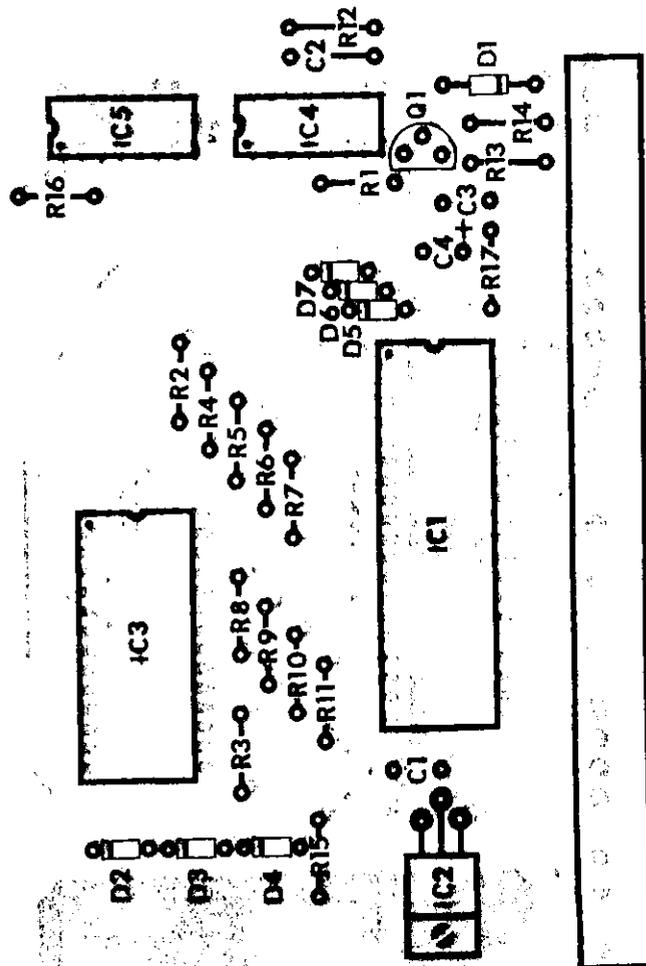
<u>✓</u>	C1	0.1 mfd capacitor
<u>✓</u>	C2	100 pfd capacitor
<u>✓</u>	C3	0.1 mfd capacitor
<u>✓</u>	C4*	10 mfd@ 15 VDC electrolytic

Diodes and Transistors

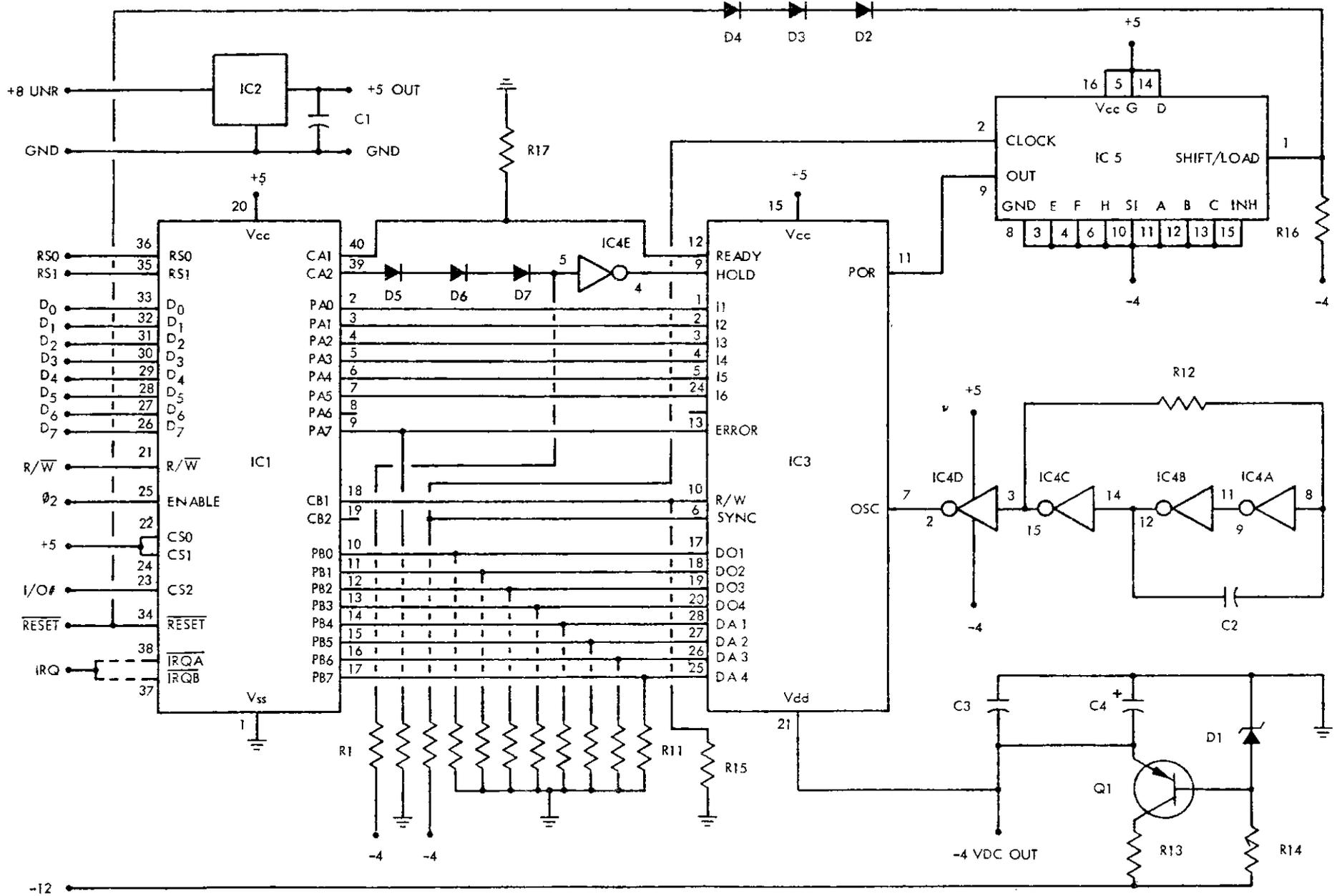
<u>✓</u>	D1*	4.7 volt 400 mw zener diode 1N5230 or 1N4732
<u>✓</u>	D2*	1N4148 silicon diode
<u>✓</u>	D3*	" " "
<u>✓</u>	D4*	" " "
<u>✓</u>	D5*	" " "
<u>✓</u>	D6*	" " "
<u>✓</u>	D7*	" " "
<u>✓</u>	Q1*	2N5087 transistor

Integrated Circuits

<u>✓</u>	IC1*	6820 MOS peripheral interface adaptor 6821
<u>✓</u>	IC2*	7805 voltage regulator
<u>✓</u>	IC3*	MM57109 FAN MOS calculator chip
<u>✓</u>	IC4*	4009 or 14009 MOS hex inverter
<u>✓</u>	IC5*	74C165 MOS shift register



MP-N CALCULATOR INTERFACE



CALC-1 Program

In order to see how the calculator chip is used and how to incorporate these subroutines into a program, the CALC-1 program listing is given. CALC-1 allows the operator to use the calculator chip just as you would a standard RPN desk calculator with the same features. All communication to the chip is done thru the terminal's keyboard with all results displayed on the terminal's display. Since the terminal's keyboard just has standard ASCII characters rather than the labeling found on calculator keys; selected ASCII characters have been substituted for normal calculator function keys. It is the job of the CALC-1 program to accept all data and instruction commands from the terminal's keyboard, send them to the calculator chip and display all results on the terminal's display. The program resides from memory locations 0020 thru 02C0 which is approximately 700 bytes of code. Since most of the lower 256 bytes are used for the ASCII character lookup table and some of the upper is used for terminal interfacing, you should be able to incorporate the package into your program using somewhat less memory than was used here.

The program starts at line 50 by storing the ASCII lookup table from memory locations 0080 thru 00FF. This table covers the entire 128 character ASCII set. Whenever an ASCII character is received from the keyboard it is OR'ed with 80, and the resulting address contains the selected command or instruction for the calculator chip. Line 210 ORG's the program at memory location 0100 where the terminal's screen is cleared and titled. Line 250 loads the index register extended with the contents of memory locations A002 and A003 with 800C, the starting address of Port 3. If you wish to plug the calculator board onto an I/O port other than PORT 3. Use the table below to find the address to be loaded into memory locations A002 and A003 prior to executing the program.

PORT0	8000
PORT1	8004 (Serial control interface only)
PORT2	8008
PORT3	800C
PORT4	8010
PORT5	8014
PORT6	8018
PORT7	801C

Lines 280 thru 370 contain the INITIAL subroutine described in detail earlier. Lines 380 thru 410 accept entered keyboard commands, lookup the selected calculator instructions and deposit the data or instruction in the A accumulator. Lines 440 thru 550 contain the OUTINS subroutine described in detail earlier. Lines 550 thru 740 check to see what instruction or data has been entered so the result may be output if appropriate. Line 710 looks for the TOGM instruction so the program knows which display mode to use when outputting data. Lines 770 thru 840 contain the SETMEM subroutine described in detail earlier. Since the SETMEM subroutine destroys the contents of the index register, line 850 reloads it before proceeding to the OUTANS subroutine contained in lines 880 thru 1200. Line 1210 checks to see if the ERROR flag was set during the last output sequence. If so, program control is transferred to lines 1220 thru 1350 where an error message is output and the error flag cleared by sending an ECLR instruction to the calculator chip. Line 1380 tests to see if the calculator is in the floating point or scientific mode. If floating point, control is transferred to lines 1400 thru 1670. If scientific, control is transferred to lines 1680 thru 1990. In both modes the data is output to the display in the selected mode and program control is transferred back to line 380 where new commands or data may be entered.

00010		NAM	CALC-1
00020		OPT	0
00030		OPT	5
00040	0080	ORG	\$0080
00050	0080 OF	FCB	\$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F
	0081 OF		
	0082 OF		
	0083 OF		
	0084 OF		
	0085 OF		
	0086 OF		
	0087 OF		
00060	0088 OF	FCB	\$0F, \$0F, \$0F, \$0F, \$0F, \$21, \$0F, \$0F
	0089 OF		
	008A OF		
	008B OF		
	008C OF		
	008D 21		
	008E OF		
	008F OF		
00070	0090 OF	FCB	\$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F
	0091 OF		
	0092 OF		
	0093 OF		
	0094 OF		
	0095 OF		
	0096 OF		
	0097 OF		
00080	0098 2F	FCB	\$2F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F
	0099 OF		
	009A OF		
	009B OF		
	009C OF		
	009D OF		
	009E OF		
	009F OF		
00090	00A0 21	FCB	\$21, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F, \$0F
	00A1 OF		
	00A2 OF		
	00A3 OF		
	00A4 OF		
	00A5 OF		
	00A6 OF		
	00A7 OF		
00100	00A8 OF	FCB	\$0F, \$0F, \$3B, \$39, \$0F, \$3A, \$0A, \$3C
	00A9 OF		
	00AA 3B		
	00AB 39		
	00AC OF		
	00AD 3A		
	00AE 0A		
	00AF 3C		
00110	00B0 00	FCB	\$00, \$01, \$02, \$03, \$04, \$05, \$06, \$07
	00B1 01		

	00B2	02		
	00B3	03		
	00B4	04		
	00B5	05		
	00B6	06		
	00B7	07		
00120	00B8	08	FCB	\$08, \$09, \$0F, \$0F, \$0F, \$0F, \$22, \$0F
	00B9	09		
	00BA	0F		
	00BB	0F		
	00BC	0F		
	00BD	0F		
	00BE	22		
	00BF	0F		
00130	00C0	0F	FCB	\$0F, \$1B, \$36, \$25, \$2D, \$0B, \$2C, \$1C
	00C1	1B		
	00C2	36		
	00C3	25		
	00C4	2D		
	00C5	0B		
	00C6	2C		
	00C7	1C		
00140	00C8	1D	FCB	\$1D, \$20, \$0F, \$0F, \$0F, \$18, \$35, \$23
	00C9	20		
	00CA	0F		
	00CB	0F		
	00CC	0F		
	00CD	18		
	00CE	35		
	00CF	23		
00150	00D0	0D	FCB	\$0D, \$33, \$37, \$24, \$26, \$32, \$34, \$31
	00D1	33		
	00D2	37		
	00D3	24		
	00D4	26		
	00D5	32		
	00D6	34		
	00D7	31		
00160	00D8	30	FCB	\$30, \$2E, \$0C, \$0F, \$0F, \$0F, \$38, \$0F
	00D9	2E		
	00DA	0C		
	00DB	0F		
	00DC	0F		
	00DD	0F		
	00DE	38		
	00DF	0F		
00170	00E0	0F	FCB	\$0F, \$0F, \$36, \$25, \$2D, \$0B, \$2C, \$1C
	00E1	0F		
	00E2	36		
	00E3	25		
	00E4	2D		
	00E5	0B		
	00E6	2C		
	00E7	1C		

```

00180 00E8 1D          FCB    $1D, $20, $0F, $0F, $0F, $18, $35, $23
      00E9 20
      00EA 0F
      00EB 0F
      00EC 0F
      00ED 18
      00EE 35
      00EF 23

```

```

00190 00F0 0D          FCB    $0D, $33, $37, $24, $26, $32, $34, $31
      00F1 33
      00F2 37
      00F3 24
      00F4 26
      00F5 32
      00F6 34
      00F7 31

```

```

00200 00F8 30          FCB    $30, $2B, $0C, $0F, $0F, $0F, $0F, $0F
      00F9 2B
      00FA 0C
      00FB 0F
      00FC 0F
      00FD 0F
      00FE 0F
      00FF 0F

```

```

00210 0100          ORG    $0100
00220 0100 8E A047 START LDS    #$A047    DECREMENT STACK
00230 0103 CE 0287     LDX    #CLRSCN
00240 0106 BD E07E     JSR    PDATA1    CLEAR AND TITLE TERM.
00250 0109 FE A002     LDX    PARADR
00260 010C 8D 02      BSR    INITAL
00270 010E 20 13      BRA    COMAND
00280 0110 86 7F      INITAL LDA A    #$7F    INIT. A SIDE OF PIA
00290 0112 A7 00      STA A    0, X
00300 0114 86 36      LDA A    #$36    HIGH HOLD-POS READY
00310 0116 A7 01      STA A    1, X
00320 0118 86 00      LDA A    #$00    INIT. B SIDE OF PIA
00330 011A A7 02      STA A    2, X
00340 011C 86 34      LDA A    #$34    NEG R/W
00350 011E A7 03      STA A    3, X
00360 0120 A6 02      LDA A    2, X    CLEAR R/W FLAG
00370 0122 39          RTS
00380 0123 BD E1AC COMAND JSR    INEE     GET OPERATOR DATA
00390 0126 8A 80          ORA A    #$80    POSITION TO THE TOP OF TABLE
00400 0128 B7 012C      STA A    POINT+1
00410 012B 96 00      POINT LDA A    $00
00412 012D 81 21          CMP A    #$21
00414 012F 27 43          BEQ    ZERMEM
00420 0131 8D 02          BSR    OUTINS
00430 0133 20 17          BRA    CHRCHK
00440 0135 E6 01      OUTINS LDA B    1, X    WAIT FOR READY
00450 0137 2A FC          BPL    OUTINS
00460 0139 A7 00      STA A    0, X    FORWARD INSTRUCTION TO CALC.
00470 013B E6 00      LDA B    0, X    CLEAR FLAG BIT
00480 013D C6 3C      LDA B    #$3C    LOW HOLD-NEG READY

```

00490	013F	E7	01		STA B	1, X	BRING HOLD LINE LOW
00500	0141	E6	01	WAIT10	LDA B	1, X	
00510	0143	2A	FC		BFL	WAIT10	LOOK FOR READY LOW
00520	0145	E6	00		LDA B	0, X	CLEAR FLAG BIT
00530	0147	C6	36		LDA B	##36	HIGH HOLD-POS READY
00540	0149	E7	01		STA B	1, X	RETURN HOLD LINE HIGH
00550	014B	39			RTS		
00560	014C	81	2F	CHRCHK	CMP A	##2F	
00570	014E	26	03		BNE	SKIP75	
00575	0150	7F	02AE		CLR	FORMAT	
00580	0153	7D	02AF	SKIP75	TST	SMDC	CHECK FOR PREVIOUS SMDC INSTR
00590	0156	26	1C		BNE	ZERMEM	
00600	0158	81	0F	CONT50	CMP A	##0F	
00620	015A	27	C7		BEQ	COMMAND	GET MORE DATA IF NOP
00630	015C	81	18		CMP A	##18	
00640	015E	26	05		BNE	SKIP25	
00650	0160	73	02AF		COM	SMDC	
00660	0163	20	BE		BRA	COMMAND	GET MORE DATA IF SMDC
00670	0165	81	20	SKIP25	CMP A	##20	
00680	0167	27	BA		BEQ	COMMAND	GET MORE DATA IF INV
00690	0169	81	0B		CMP A	##0B	
00700	016B	23	B6		BLS	COMMAND	GET MORE DATA IF NUMBERS
00710	016D	81	22		CMP A	##22	LOOK FOR TOGM
00720	016F	26	03		BNE	ZERMEM	
00730	0171	73	02AE		COM	FORMAT	
00740	0174	7F	02AF	ZERMEM	CLR	SMDC	ZERO SMDC
00750	0177	8D	02		BSR	SETMEM	
00760	0179	20	11		BRA	LOADADR	
00770	017B	7F	0020	SETMEM	CLR	##20	CLEAR ##0020
00780	017E	CE	0020		LDX	##20	BOTTOM OF BUFFER
00790	0181	C6	20		LDA B	##20	
00800	0183	08		LOOP1	INX		
00810	0184	E7	00		STA B	0, X	STORE A SPACE
00820	0186	8C	002B		CPX	##2B	CHECK FOR TOP OF BUFFER
00830	0189	26	F8		BNE	LOOP1	
00840	018B	39			RTS		
00850	018C	FE	A002	LOADADR	LDX	PARADR	
00860	018F	8D	02		BSR	OUTANS	
00870	0191	20	3D		BRA	OUTCHR	
00880	0193	E6	01	OUTANS	LDA B	1, X	
00890	0195	2A	FC		BFL	OUTANS	
00900	0197	A6	00		LDA A	0, X	CLEAR FLAG BIT
00910	0199	86	16		LDA A	##16	SEND AN OUT
00920	019B	A7	00		STA A	0, X	
00930	019D	C6	3E		LDA B	##3E	LOW HOLD-POS READY
00940	019F	E7	01		STA B	1, X	BRING HOLD LINE LOW
00950	01A1	E6	01	WAIT30	LDA B	1, X	WAIT FOR SECOND READY
00960	01A3	2A	FC		BFL	WAIT30	
00970	01A5	E6	00		LDA B	0, X	CLEAR FLAG BIT
00980	01A7	86	0F		LDA A	##0F	
00990	01A9	A7	00		STA A	0, X	SEND A NOP
01000	01AB	E6	03	WAIT3	LDA B	0, X	LOOK FOR R/W STROBE
01010	01AD	2B	06		BMI	OUTDIS	TRANSFER CALC DATA TO MEMORY
01020	01AF	E6	01		LDA B	1, X	LOOK FOR READY STROBE

```

01030 01B1 2B 16          BMI    CONFLG  PRINT MEMORY CONTENTS
01040 01B3 20 F6          BRA    WAIT3
01050 01B5 A6 02    OUTDIG LDA A    2,X    LOAD OUT DATA INTO A
01060 01B7 16          TAB
01070 01B8 84 0F          AND A    #0F    ELIMINATE UPPER 4 BITS
01080 01BA 8A 30          ORA A    #30    CONVERT TO ASCII DATA
01090 01BC 54          LSR B
01100 01BD 54          LSR B
01110 01BE 54          LSR B
01120 01BF 54          LSR B
01130 01C0 CA 20          ORA B    #20    INCREMENT ADDRESSES BY #20
01140 01C2 F7 01C6       STA B    POINT2+1 STORE OUT DATA SEQUENTIALLY
01150 01C5 97 00C1 POINT2 STA A    $0
01160 01C7 20 E2          BRA    WAIT3
01170 01C9 86 36    CONFLG LDA A    #36    HIGH HOLD-POS READY
01180 01CB A7 01          STA A    1,X    BRING HOLD LINE HIGH
01190 01CD A6 00          LDA A    0,X    CLEAR FLAG BIT
01200 01CF 39          RTS
01210 01D0 2A 1E    OUTCHR BPL    CONT1    SKIP IF NO ERROR
01220 01D2 E6 01    WAIT70 LDA B    1,X    WAIT FOR READY
01230 01D4 2A FC          BPL    WAIT70
01240 01D6 86 2B          LDA A    #2B    ERROR CLEAR INSTRUCTION
01250 01D8 A7 00          STA A    0,X
01260 01DA E6 00          LDA B    0,X    CLEAR FLAG BIT
01270 01DC C6 3C          LDA B    #3C    LOW HOLD-NEG READY
01280 01DE E7 01          STA B    1,X    BRING HOLD LOW
01290 01E0 E6 01    WAIT71 LDA B    1,X
01300 01E2 2A FC          BPL    WAIT71
01310 01E4 E6 00          LDA B    0,X    CLEAR FLAG BIT
01320 01E6 C6 36          LDA B    #36    HIGH HOLD-POS READY
01330 01E8 E7 01          STA B    1,X    RETURN HOLD HIGH
01340 01EA CE 02B0       LDX    #ERRMSG
01350 01ED BD E07E       JSR    FDATA1
01360 01F0 CE 02A8 CONT1 LDX    #CRLF
01370 01F3 BD E07E       JSR    FDATA1
01380 01F6 7D 02AE       TST    FORMAT
01390 01F9 2B 3F          BMI    SCINDT
01400 01FB CE 0022 FLOPNT LDX    #22    FLOATING POINT NOTATION
01410 01FE A6 00          LDA A    0,X    INPUT MANTISSA SIGN DATA
01420 0200 84 08          AND A    #08    MASK BIT 4
01430 0202 26 04          BNE    MINFNT
01440 0204 86 20          LDA A    #20    LOAD A SPACE
01450 0206 20 02          BRA    PRINT1
01460 0208 86 2D    MINFNT LDA A    #2D    LOAD A MINUS
01470 020A BD E1D1 PRINT1 JSR    OUTEEE    PRINT CHARACTER
01480 020D 08          DPIND  INX
01490 020E E6 00          LDA B    0,X
01500 0210 C4 0F          AND B    #0F
01510 0212 E7 00          STA B    0,X
01520 0214 C6 2F          LDA B    #2F
01530 0216 E0 00          SUB B    0,X
01540 0218 D7 21          STA B    $21    STORE DEC. PT POSITION IND.
01550 021A 08          DIGLOP INX
01560 021B A6 00          LDA A    0,X

```

```

01570 021D BD E1D1      JSR      OUTEEE      OUTPUT ASCII NUMBER
01580 0220 9C 20      CPX      $20      TIME FOR DEC. PT. ?
01590 0222 26 05      BNE      ENDCH1
01600 0224 86 2E      LDA      A      #$2E
01610 0226 BD E1D1      JSR      OUTEEE
01620 0229 8C 002B ENDCH1 CPX      $$2B      CHECK FOR LAST DIGIT
01630 022C 26 EC      BNE      DIGLOP      GET NEXT DIGIT
01640 022E CE 02A8      LDX      #CRLF
01650 0231 BD E07E      JSR      PDATA1      PRINT CR/LF
01660 0234 FE A002      LDX      PARADR
01670 0237 7E 0123      JMP      COMMAND
01680 023A 96 22      SCINOT LDA      A      $22      SCIENTIFIC NOTATION
01690 023C 84 08      AND      A      $$08      LOOK FOR NEGATIVE MANTISSA
01700 023E 26 04      BNE      NEGPNT
01710 0240 86 20      LDA      A      $$20      SPACE IF NOT
01720 0242 20 02      BRA      PRINT2
01730 0244 86 2D      NEGPNT LDA      A      $$2D
01740 0246 BD E1D1 PRINT2 JSR      OUTEEE      PRINT SIGN
01750 0249 CE 0023      LDX      $$23
01760 024C 08      NUMLOP INX
01770 024D A6 00      LDA      A      0,X
01780 024F BD E1D1      JSR      OUTEEE
01790 0252 8C 0024      CPX      $$24      LOOK FOR DEC. PT. DIGIT
01800 0255 26 05      BNE      SKIPDF
01810 0257 86 2E      LDA      A      $$2E
01820 0259 BD E1D1      JSR      OUTEEE      PRINT DEC. PT.
01830 025C 8C 002B SKIPDF CPX      $$2B      CHECK FOR LAST DIGIT
01840 025F 26 EB      BNE      NUMLOP
01850 0261 86 45      LDA      A      $45
01860 0263 BD E1D1      JSR      OUTEEE      PRINT AN E
01870 0266 96 22      LDA      A      $22      LOAD SIGN BYTE
01880 0268 84 01      AND      A      $01
01890 026A 27 05      BEQ      SKPSGN
01900 026C 86 2D      LDA      A      $$2D
01910 026E BD E1D1      JSR      OUTEEE      PRINT A -
01920 0271 96 20      SKPSGN LDA      A      $20
01930 0273 BD E1D1      JSR      OUTEEE      PRINT EXPONENT MSD
01940 0276 96 21      LDA      A      $21
01950 0278 BD E1D1      JSR      OUTEEE      PRINT EXPONENT LSD
01960 027B CE 02A8      LDX      #CRLF
01970 027E BD E07E      JSR      PDATA1      PRINT CR/LF
01980 0281 FE A002      LDX      PARADR
01990 0284 7E 0123      JMP      COMMAND
02000 0287 0D      CLRSCN FCB      $0D, $0A, $10, $1A, $00
      0288 0A
      0289 10
      028A 1A
      028B 00
02010 028C 53      FCC      "SWTPC 6800 CALC-1 CALCULATOR"
      028D 57
      028E 54
      028F 50
      0290 43
      0291 20

```

	0292	36			
	0293	38			
	0294	30			
	0295	30			
	0296	20			
	0297	43			
	0298	41			
	0299	4C			
	029A	43			
	029B	2D			
	029C	31			
	029D	20			
	029E	43			
	029F	41			
	02A0	4C			
	02A1	43			
	02A2	55			
	02A3	4C			
	02A4	41			
	02A5	54			
	02A6	4F			
	02A7	52			
02020	02A8	0D	CRLF	FCB	\$0D, \$0A, \$00, \$00, \$00, \$04
	02A9	0A			
	02AA	00			
	02AB	00			
	02AC	00			
	02AD	04			
02030	02AE	00	FORMAT	FCB	\$00
02040	02AF	00	SMDC	FCB	\$00
02050	02B0	0D	ERRMSG	FCB	\$0D, \$0A, \$00, \$00
	02B1	0A			
	02B2	00			
	02B3	00			
02060	02B4	45		FCC	ERROR
	02B5	52			
	02B6	52			
	02B7	4F			
	02B8	52			
02070	02B9	04		FCB	\$04
02080		E07E	FDATA1	EQU	\$E07E
02090		A002	PARADR	EQU	\$A002
02100		E1AC	INEEE	EQU	\$E1AC
02110		E1D1	OUTEEE	EQU	\$E1D1
02120	A048			ORG	\$A048
02130	A048	0100		FDB	\$0100
02140	A002			ORG	\$A002
02150	A002	800C		FDB	\$800C
02160				END	
START	0100				
INITAL	0110				
COMAND	0123				
POINT	012B				
OUTINS	0135				

PAGE 008 CALC-1

WAIT10 0141
CHRCHK 014C
SKIP75 0153
CONT50 0158
SKIP25 0165
ZERMEM 0174
SETMEM 017B
LOOP1 0183
LODADR 018C
OUTANS 0193
WAIT30 01A1
WAIT3 01AB
OUTDIG 01B5
FOINT2 01C5
CONFLG 01C9
OUTCHR 01D0
WAIT70 01D2
WAIT71 01E0
CONT1 01F0
FLOPNT 01FB
MINPNT 0208
PRINT1 020A
DPIND 020D
DIGLOP 021A
ENDCH1 0229
SCINOT 023A
NEGPNT 0244
PRINT2 0246
NUMLOP 024C
SKIPDP 025C
SKPSGN 0271
CLRSCN 0287
CRLF 02AB
FORMAT 02AE
SMDC 02AF
ERRMSG 02B0
PDATA1 E07E
PARADR A002
INEEE E1AC
OUTEEE E1D1

TOTAL ERRORS 00000

RPN—the only language that lets you “speak” with confidence and consistency to a pocket-sized computer calculator.

In 1967, Hewlett-Packard embarked on a major new development effort: to design a family of advanced computer calculators powerful enough to solve complex engineering/scientific problems yet simple enough to be used by anyone who works with numbers.

As part of this effort, HP carefully evaluated the strengths and weaknesses of the various languages which an operator might use to communicate with an electronic calculating device. Among those studied were:

- computer languages such as BASIC and FORTRAN,
- various forms of algebraic notation, and
- RPN (*Reverse Polish Notation*), a parenthesis-free but unambiguous language derived from that developed by the Polish mathematician, Jan Lukasiewicz.

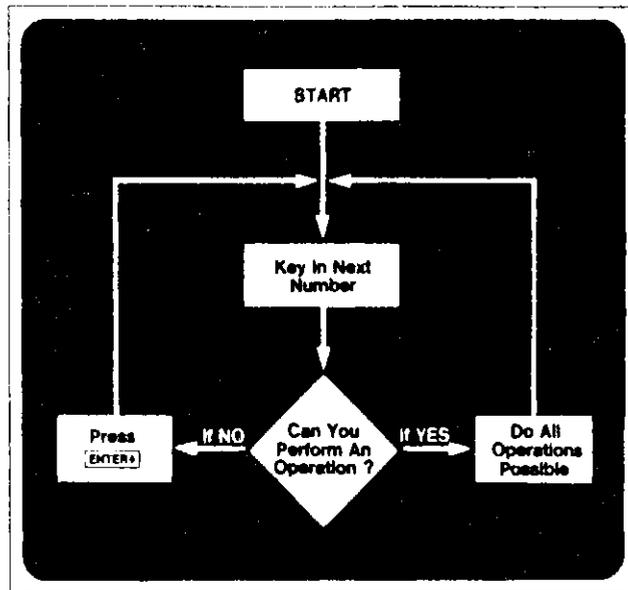
As might be expected, each of these languages was found to excel in a particular application. For its biggest programmable desktop calculators, HP selected BASIC. For its other powerful desktop calculators, with less extensive storage capacity, HP chose algebraic notation.

But, given the design constraints of a pocket-sized scientific computer calculator, RPN proved the simplest, most efficient, most consistent way to solve complex mathematical problems.

Only RPN offers these powerful advantages

Compared to alternative logic systems, Hewlett-Packard believes that only RPN—in combination with a 4-register operational memory stack—gives you these powerful advantages.

1. You can always enter your data the same way, i.e., from left to right—the same way you read an equation. Yet, there is no need for a parenthesis key; nor for a complicated “operational hierarchy.”
2. You can always proceed through your problem the same way. Once you've entered a number, you ask: “Can I perform an operation?” If yes, you do it. If no, you press **ENTER** and key in the next number.
3. You always see all intermediate answers—as they are calculated—so that you can check the progress of your calculation as you go. As important, you can review all numbers stored in the calculator at any time by pressing a few keys. There is no “hidden” data.
4. You don't have to think your problem all the way through beforehand unless the problem is so complex that it may require simultaneous storage of three or more intermediate answers.
5. You can easily recover from errors since all operations are performed sequentially, immediately after pressing the appropriate key.



The RPN method consists of four, easy-to-remember steps. Once learned, it can be applied to almost any mathematical expression.

6. You don't have to write down and re-enter intermediate answers, a real time-saver when working with numbers of eight or nine digits each.
7. You can communicate with your calculator confidently, consistently because you can always proceed the same way.

If all this sounds too good to be true, bear with us—you'll soon get the chance to see for yourself. But first, we need to describe how RPN and the 4-register operational stack operate.

The RPN method—it takes a few minutes to learn but can save years of frustration.

Yes, the RPN method does take some getting used to. But, once you've learned it, you can use the RPN method to solve almost any mathematical expression—confidently, consistently.

There are only four easy-to-follow steps:

1. Starting at the left side of the problem, key in the first or next number.
2. Determine if any operations can be performed. If so, do all operations possible.
3. If not, press **ENTER** to save the number for future use.
4. Repeat steps 1 through 3 until your calculation is completed.

A diagram of the RPN method is shown above.

Simple arithmetic, the RPN way.

Just to show how it works, let's try the RPN method on two simple problems (we'll use them again in the comparisons that begin on the next page).

Problem: $3 \times 4 = 12$

RPN solution:

Step	Press	See Displayed
1. Key in first number.	3	3
2. Since only one number has been keyed in, no operations are possible. Press ENTER .	ENTER	3
3. Key in next number.	4	4
4. Since both numbers are now in calculator, multiplication can be performed.	X	12

Problem: $(3 \times 4) + (5 \times 6) = 42$

RPN solution:

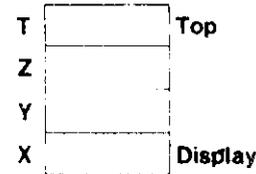
Step	Press	See Displayed
1. Key in first number.	3	3
2. No operations possible. Press ENTER .	ENTER	3
3. Key in second number.	4	4
4. Since both numbers are in calculator, first multiplication is possible.	X	12
5. Key in next number. (First intermediate answer will be automatically stored for future use.)	5	5
6. No operations possible. Press ENTER .	ENTER	5
7. Key in next number.	6	6
8. Second multiplication is possible since both numbers are in calculator.	X	30
9. Addition is possible since both intermediate answers have been calculated and are stored in 4-register operational stack.	+	42

If you've followed us this far, you've noticed two important facts:

1. Both of these problems were solved in the same, consistent manner, using the same simple set of rules.
2. All intermediate answers were displayed as they were calculated, and stored and retrieved as needed to complete the calculation. With RPN and a 4-register operational memory stack, there is almost never a need to write down intermediate answers.

How the operational stack works.

The four registers of HP's exclusive operational stack can be represented by the following diagram.



When a number is keyed in, it goes into the X register for display. Pressing the **ENTER** key duplicates the contents of the X register into the Y register and moves all other numbers in the stack up one position.

When an operation key (**+**, **-**, **X**, **÷**, **x^y**) is pressed the operation is performed on the numbers in the X and Y registers, and the answer appears in the X register for display. Numbers in the other registers automatically drop one position.

To demonstrate these points, we'll show what happens to the stack as we solve the problem: $(3 \times 4) + (5 \times 6) = 42$.

T									
Z					12	12			
Y		3	3		12	5	5	12	
X	3	3	4	12	5	5	6	30	42
	3	ENTER	4	X	5	ENTER	6	X	+

As you can see, all numbers are automatically positioned in the stack on a last-in-first-out basis, in the proper order for subsequent use.

Now that we've described how RPN logic operates, we can proceed with our problem-by-problem comparison of this system versus two others used in today's scientific pocket calculators.

We think you will find it interesting.

Calc-1 Instruction Set

<u>Full Name</u>	<u>ASCII Character</u>
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
Decimal point	.
Enter exponent	E
Change sign	Z
Constant PI	P
Set mantissa digit count	M
X exchange M	A
Memory store	G
Memory recall	H
Inverse mode	I
Enter	sp or cr
Toggle mode	>
Roll stack	O
Sine X	S
Cosine X	C
Tangent X	T
Error clear	Y
Radians to degrees	F
Degrees to radians	D
Master clear	cntrl X
X exchange Y	X
E to X	W
Ten to X	U
Square	Q
Square root	V
Natural log of X	N
Base 10 log of X	B
One divided by X	R
Y to X	^
Plus	+
Minus	-
Times	*
Divide	/

Table I

MM57109 Instruction Description Table (Continued) (* Indicates 2-word instruction)

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION
Branch	Count	IBNZ	31	Increment memory and branch if $M \neq 0$	$M + 1 \rightarrow M$. If $M = 0$, skip second instruction word. Otherwise, branch to address specified by second instruction word.
		DBNZ	32	Decrement memory and branch if $M \neq 0$	$M - 1 \rightarrow M$. If $M = 0$, skip second instruction word. Otherwise, branch to address specified by second instruction word.
I/O	Multi-digit	IN*	27	Multidigit input to X	The processor supplies a 4-bit digit address (DA4-DA1) accompanied by a digit address strobe (\overline{DAS}) for each digit to be input. The high order address for the number to be input would typically come from the second instruction word. The digit is input on D4-D1, using $ISEL = 0$ to select digit data instead of instructions. The number of digits to be input depends on the calculation mode (scientific notation or floating point) and the mantissa digit count (See Data Formats and Instruction Timing). Data to be input is stored in X and the stack is pushed ($X \rightarrow Y \rightarrow Z \rightarrow T$). At the conclusion of the input, DA4-DA1 = 0.
		OUT*	26	Multidigit output from X	Addressing and number of digits is identical to IN instruction. Each time a new digit address is supplied, the processor places the digit to be output on DO4-DO1 and pulses the R/W line active low. At the conclusion of output, DO4-DO1 = 0 and DA4-DA1 = 0.
I/O	Single-digit	AIN	16	Asynchronous Input	A single digit is read into the processor on D4-D1. $ISEL = 0$ is used by external hardware to select the digit instead of instruction. It will not read the digit until $\overline{ADR} = 0$ ($ISEL = 0$ selects \overline{ADR} instead of I_5), indicating data valid. F2 is pulsed active low to acknowledge data just read.
I/O	Flags	SF1	47	Set Flag 1	Set F1 high, i.e. $F1 = 1$.
		PF1	50	Pulse Flag 1	F1 is pulsed active high. If F1 is already high, this results in it being set low.
		SF2	51	Set Flag 2	Set F2 high, i.e. $F2 = 1$.
		PF2	52	Pulse Flag 2	F2 is pulsed active high. If F2 is already high, this results in it being set low.
		PRW1	75	Pulse $\overline{R/W} 1$	Generates $\overline{R/W}$ active low pulse which may be used as a strobe or to clock extra instruction bits into a flip-flop or register.
Mode Control		PRW2	76	Pulse $\overline{R/W} 2$	Identical to PRW1 instruction. Advantage may be taken of the fact that the last 2 bits of the PRW1 op code are 10 and the last 2 bits of the PRW2 op code are 01. Either of these bits can be clocked into a flip-flop using the $\overline{R/W}$ pulse.
		TOGM	42	Toggle Mode	Change mode from floating point to scientific notation or vice-versa, depending on present mode. The mode affects only the IN and OUT instructions. Internal calculations are always in 8-digit scientific notation.
		SMDC*	30	Set Mantissa Digit Count	Mantissa digit count is set to the contents of the second instruction word (=1 to 8).
		INV	40	Inverse Mode	Set inverse mode for trig or memory function instruction that will immediately follow. Inverse mode is for next instruction only.

Table I

MM57109 Instruction Description Table (* Indicates 2-word instruction)

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION
Digit Entry			00		Mantissa or exponent digits. On first digit (d) the following occurs: Z → T Y → Z X → Y d → X See description of number entry on page 11. Digits that follow will be mantissa fraction. Digits that follow will be exponent. Change sign of exponent or mantissa. Xm = X mantissa Xe = X exponent CS causes -Xm → Xm or -Xe → Xe depending on whether or not an EE instruction was executed after last number entry initiation. 3.1415927 → X, stack not pushed. Terminates digit entry and pushes the stack. The argument entered will be in X and Y. Z → T Y → Z X → Y Do nothing instruction that will terminate digit entry. External hardware detects HALT op code and generates HOLD = 1. Processor waits for HOLD = 0 before continuing. HALT acts as a NOP and may be inserted between digit entry instructions since it does not terminate digit entry.
			01		
			02		
			03		
			04		
			05		
			06		
			07		
			10		
			11		
			12		
			13		
			14		
				15	
			41		
		NOP	77	No Operation	
		HALT	17	Halt	
Move		ROLL	43		Roll Stack.
		POP	56	Pop	Pop Stack. Y → X Z → Y T → Z O → T
		KEY	60		Exchange X and Y. X ↔ Y
		XEM	33		Exchange X with memory. X ↔ M
		MS	34		Store X in Memory. X → M
		MR	35		Recall Memory into X. M → X
		LSH	36	Left Shift Xm	X mantissa is left shifted while leaving decimal point in same position. Former most significant digit is saved in link digit. Least significant digit is zero.
		RSH	37	Right Shift Xm	X mantissa is right shifted while leaving decimal point in same position. Link digit, which is normally zero except after a left shift, is shifted into the most significant digit. Least significant digit is lost.

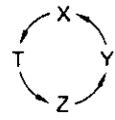


Table I

MM57109 Instruction Description Table (Continued) (* Indicates 2-word instruction)

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION	
Math	F (X,Y)	+	71		Add X to Y $X + Y \rightarrow X$. On +, -, x, / and YX instructions, stack is popped as follows: $Z \rightarrow Y$ $T \rightarrow Z$ $O \rightarrow T$ Former X, Y are lost.	
		-	72		Subtract X from Y. $Y - X \rightarrow X$	
		x	73		Multiply X times Y. $Y \times X \rightarrow X$	
		/	74		Divide X into Y. $Y \div X \rightarrow X$	
		YX	70		Raise Y to X power. $Y^X \rightarrow X$	
		F (X,M)	INV +*	40, 71	Memory Plus	Add X to memory. $M + X \rightarrow M$ On INV +, -, x and / instructions, X, Y, Z, and T are unchanged.
			INV -*	40, 72	Memory Minus	Subtract X from memory. $M - X \rightarrow M$
	INV x*		40, 73	Memory Times	Multiply X times memory. $M \times X \rightarrow M$	
	F (X) Math	INV /*	40, 74	Memory Divide	Divide X into memory. $M \div X \rightarrow M$	
		1/X	67		$1 \div X \rightarrow X$. On all F (X) math instructions Y, Z, T and M are unchanged and previous X is lost.	
		SQRT	64		$\sqrt{X} \rightarrow X$	
		SQ	63		$X^2 \rightarrow X$	
		10X	62		$10^X \rightarrow X$	
		EX	61		$e^X \rightarrow X$	
		LN	65		$\ln X \rightarrow X$	
		LOG	66		$\log X \rightarrow X$	
		F (X) Trig	SIN	44		$\text{SIN}(X) \rightarrow X$. On all F(X) trig functions, Y, Z, T, and M are unchanged and the previous X is lost.
			COS	45		$\text{COS}(X) \rightarrow X$
	TAN		46		$\text{TAN}(X) \rightarrow X$	
	INV SIN*		40, 44	Inverse sine X	$\text{SIN}^{-1}(X) \rightarrow X$	
INV COS*	40, 45		Inverse cosine X	$\text{COS}^{-1}(X) \rightarrow X$		
INV TAN*	40, 46		Inverse tan X	$\text{TAN}^{-1}(X) \rightarrow X$		
DTR	55			Convert X from degrees to radians.		
RTD	54		Convert X from radians to degrees.			
Clear		MCLR	57		Clear all internal registers and memory; initialize I/O control signals, MDC = 8, MODE = floating point. (See initialization.)	
Branch	Test	ECLR	53		$O \rightarrow \text{Error flag}$	
		JMP*	25	Jump	Unconditional branch to address specified by second instruction word. On all branch instructions, second word contains branch address to be loaded into external PC.	
		TJC*	20	Test jump condition	Branch to address specified by second instruction word if JC (lg) is true (=1). Otherwise, skip over second word.	
		TERR*	24	Test error	Branch to address specified by second instruction word if error flag is true (=1). Otherwise, skip over second word. May be used for detecting specific errors as opposed to using the automatic error recovery scheme dealt with in the section on Error Control.	
		TX = 0*	21	Test X = 0	Branch to address specified by second instruction word if X = 0. Otherwise, skip over second word.	
		TXF*	23	Test X < 1	Branch to address specified by second instruction word if $ X < 1$. Otherwise, skip over second word. (i.e. branch if X is a fraction.)	
		TXLT0*	22	Test X < 0	Branch to address specified by second instruction word if $X < 0$. Otherwise, skip over second word.	

Table II

MM57109 Instruction Summary Table (* = 2-word instruction)

14-11	165			
	0	1	2	3
0	[REDACTED]	TJC*	[REDACTED]	[REDACTED]
1	[REDACTED]	TX-0*	[REDACTED]	[REDACTED]
2	[REDACTED]	TXLTO*	[REDACTED]	[REDACTED]
3	[REDACTED]	TXF*	[REDACTED]	[REDACTED]
4	[REDACTED]	TERR*	[REDACTED]	[REDACTED]
5	[REDACTED]	JMP*	[REDACTED]	[REDACTED]
6	[REDACTED]	OUT*	[REDACTED]	[REDACTED]
7	[REDACTED]	IN*	SF1	[REDACTED]
8	[REDACTED]	[REDACTED]	PF1	[REDACTED]
9	[REDACTED]	IBNZ*	SF2	M+1
A	[REDACTED]	DBNZ*	PF2	(M-)
B	[REDACTED]	[REDACTED]	[REDACTED]	(Mx1)
C	[REDACTED]	[REDACTED]	[REDACTED]	(M÷)
D	[REDACTED]	[REDACTED]	[REDACTED]	PRW1
E	AIN	LSH	POP	PRW2
F	HALT	RSH	[REDACTED]	NOP

SET memory locations A002-A003 to 800C!
prior to running program.

Table III - CALC-1 Instruction to ASCII Character Lookup Table

<u>FULL NAME</u>	<u>HEX OP CODE</u>	<u>MNEMONIC</u>	<u>ASCII CHARACTER</u>
	00	00	0
	01	01	1
	02	02	2
	03	03	3
	04	04	4
	05	05	5
	06	06	6
	07	07	7
	08	08	8
	09	09	9
	0A	DP	.
	0B	EE	E
	0C	CS	Z
	0D	PI	P
Asynchronous Input	0E	AIN	
Halt	0F	HALT	
Test Jump	10	TJC	
Test X=0	11	TX=0	
Test X<0	12	TXLTO	
Test 1 X 1<1	13	TXF	
Test Error	14	TERR	
Jump	15	JMP	
Multidigit Out	16	OUT	
Multidigit In	17	IN	
	18	SMDC	M
Inc & Branch if M=0	19	IBNZ	
Dec & Branch if M=0	1A	DBNZ	
	1B	XEM	A
	1C	MS	G
	1D	MR	H
Left shift Xm	1E	LSH	
Right shift Xm	1F	RSH	
	20	INV	I
	21	EN	space or carriage return
	22	TOGM	>
	23	ROLL	O
	24	SIN	S
	25	COS	C
	26	TAN	T
Set Flag 1	27	SF1	
Pulse Flag 1	28	PF1	
Set Flag 2	29	SF2	
Pulse Flag 2	2A	PF2	
Error Clear	2B	ECLR	Y
Radians to Degrees	2C	RTD	F
Degrees to Radians	2D	DTR	D
Pop	2E	POP	
Master Clear	2F	MCLR	Ctrl X

Table III - CALC-1 Instruction to ASCII Character Lookup Table

<u>NAME</u>	<u>HEX OP CODE</u>	<u>MNEMONIC</u>	<u>ASCII CHARACTER</u>
X exchange	30	XEY	X
E to X	31	EX	W
Ten to X	32	1QX	U
Square	33	SQ	Q
Square Root	34	SQRT	V
Natural Log of X	35	LN	N
Base 10 Log of X	36	LOG	B
One divided by X	37	1/X	R
Y to X	38	YX	^
Plus	39	+	+
Minus	3A	-	-
Times	3B	X	*
Divide	3C	/	/
Pulse R/W 1	3D	PRW1	
Pulse R/W 2	3E	PRW2	
No Operation	3F	NOP	

Table IV - Floating Point Mode OUT data storage

<u>Memory Location</u>	<u>DP POS</u>	<u>D7</u>	<u>D6</u>	<u>D5</u>	<u>D4</u>	<u>D3</u>	<u>D2</u>	<u>D1</u>	<u>D0</u>
20		0	0	0	0	0	0	0	0
21		0	0	1	0	0	0	0	0
22		0	0	1	1	Sm	0	0	0
23		0	0	1	1	Dp	POS		
24	0B	0	0	1	1	BCD	digit(left most)		
25	0A	0	0	1	1	BCD	digit		
26	09	0	0	1	1	BCD	digit		
27	08	0	0	1	1	BCD	digit		
28	07	0	0	1	1	BCD	digit		
29	06	0	0	1	1	BCD	digit		
2A	05	0	0	1	1	BCD	digit		
2B	04	0	0	1	1	BCD	digit(right most)		

Table IV - Scientific Mode OUT data storage

<u>Memory Location</u>	<u>D7</u>	<u>D6</u>	<u>D5</u>	<u>D4</u>	<u>D3</u>	<u>D2</u>	<u>D1</u>	<u>D0</u>
20	0	0	1	1	Most	signif.	exp.	digit
21	0	0	1	1	Least	signif.	exp.	digit
22	0	0	1	1	Sm	0	0	Se
23	NOT USED							
24	0	0	1	1	BCD	digit	(left	most)
25	0	0	1	1	BCD	digit		
26	0	0	1	1	BCD	digit		
27	0	0	1	1	BCD	digit		
28	0	0	1	1	BCD	digit		
29	0	0	1	1	BCD	digit		
2A	0	0	1	1	BCD	digit		
2B	0	0	1	1	BCD	digit	(left	most)

Notes:

- 1) If the Mantissa Digit Count (set by SMDC instruction, initially 8) is less than 8, the unused digit memory locations will be filled with ASCII spaces (20₁₆)
- 2) Sm is the sign of the mantissa. 0 = positive 1 = negative
- 3) Se is the sign of the exponent 0 = positive 1 = negative
- 4) DP POS is the decimal point position. The decimal point should follow the digit whose address is stored in memory location 24 when in the Scientific mode. In the Floating Point mode AND the data in memory location 23 with 0F and subtract the result from 2F and OR this with 20. The decimal point should follow the digit whose address is given by the result.

Table V - ASCII to CALCULATOR INSTRUCTION LOOKUP TABLE

<u>LSB</u>	<u>MSB</u>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>
0		OF	OF	21	00	OF	0D	OF	0D
1		OF	OF	OF	01	1B	33	OF	33
2		OF	OF	OF	02	36	37	36	37
3		OF	OF	OF	03	25	24	25	24
4		OF	OF	OF	04	2D	26	2D	26
5		OF	OF	OF	05	0B	32	0B	32
6		OF	OF	OF	06	2C	34	2C	34
7		OF	OF	OF	07	1C	31	1C	31
8		OF	2F	OF	08	1D	30	1D	30
9		OF	OF	OF	09	20	2B	20	2B
A		OF	OF	3B	OF	OF	0C	OF	0C
B		OF	OF	39	OF	OF	OF	OF	OF
C		OF							
D		21	OF	3A	OF	18	OF	18	OF
E		OF	OF	0A	22	35	38	35	OF
F		OF	OF	3C	OF	23	OF	23	OF

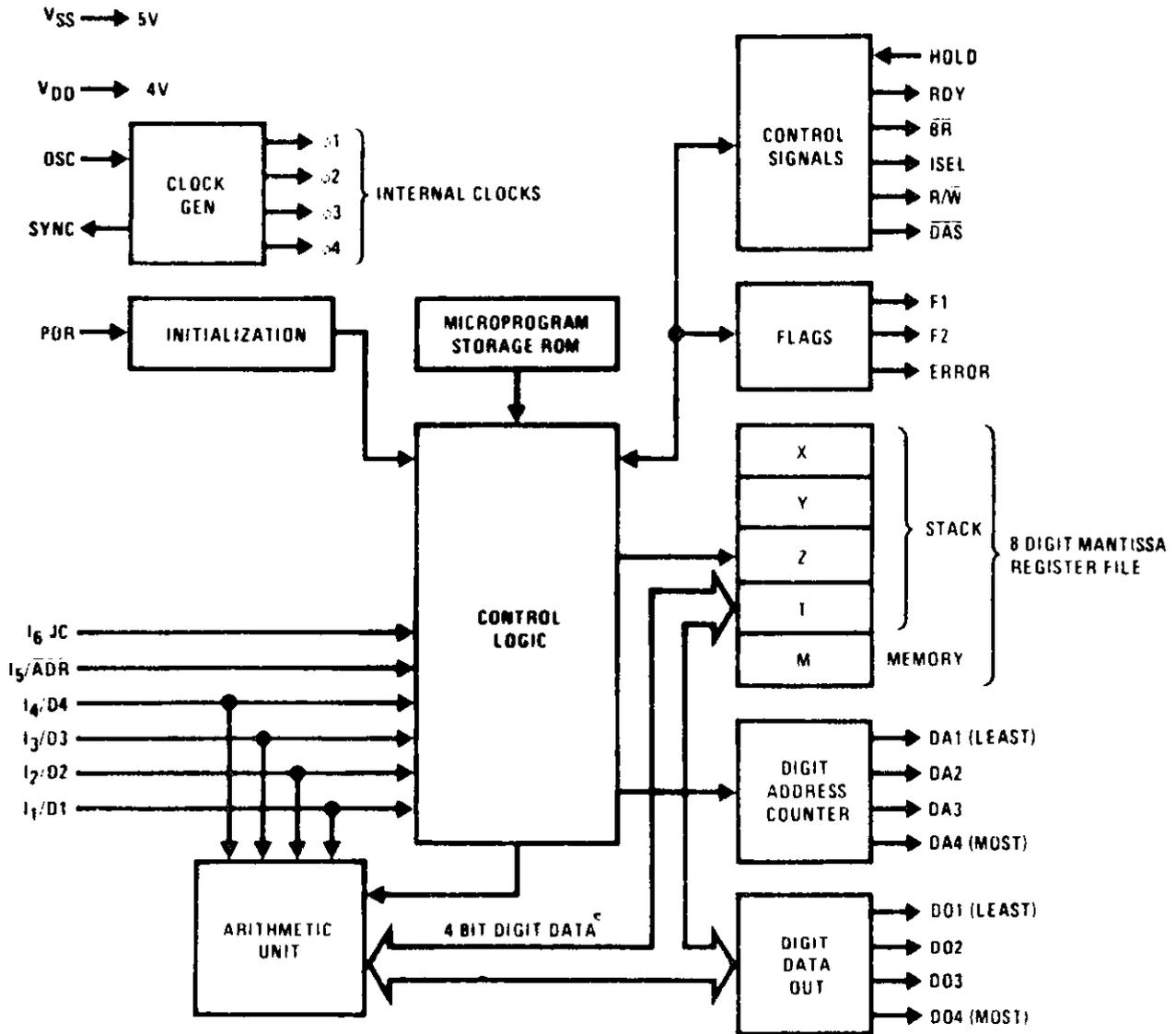
Example: An ASCII P is a hex 50 which points in the table to a 0D which is the constant PI instruction for the calculator chip

TABLE VI- ERROR CONDITIONS

The ERROR flag on the calculator chip is set when:

- 1) LN X when $X \leq 0$ LOG X when $X \leq 0$
- 2) Any result $< 10^{-99}$ Any result $\geq 10^{99}$
- 3) TAN 90° , 270° , 450° , etc.
- 4) SIN X, Cos X, TAN X when $|X| \geq 9000^\circ$
- 5) $\text{SIN}^{-1} X$, $\text{COS}^{-1} X$ when $|X| > 1$ or $|X| \leq 10^{-50}$
- 6) SQRT X when $X < 0$
- 7) dividing by 0
- 8) Outputting a number in floating point mode if the number of mantissa digits to the left of the decimal point is greater than the mantissa digit count.

Figure 1



ASCII to Hexadecimal Conversion Table

MSB LSB	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	END	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL