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 Adapter (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 ½" X 5 ¼ double sided, plated thru hole circuit board plugged onto one of tie 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 ail 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 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 feedthru'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 gives the "lowest" address of each interface card position.

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

```
86 7F     INITAL  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
39                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    LOOP 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
39                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 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 nor destroyed.

```
7F 00 20  SETMEM  CLR    $20       CLEAR $0020
CE 00 20          LDX    #$20      BOTTOM OF BUFFER
C6 20             LDA B  #$20
08        LOOP1   INX
E7 00             STA B  0,X       STORE A SPACE
8C 00 2B          CPX    #$2B      CHEXC 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 (22_{16}) 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 ($2F_{16}$) 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 ($2B_{16}$) 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-POS 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 INTO 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 01 C6          STA B  POINT2+1  STORE OUT DATA SEQUENTIALLY
97 00     POINT2  STA A  $0        SELF MODIFING CODE
20 E2             BRA    WAIT3
86 36     CONFLG  LDA A  #$36      HIGH HOLD-POS 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 ∏ is entered with an 0-9, DP, or PI instruction, the stack is first pushed and the X register cleared: Z -> T, Y -> Z, X -> Y, 0 -> 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 ∏ 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 Adapter (PIA) ICI 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, IN, 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

____	R1	47K ohm ¼ watt resistor
____	R2	1K ohm ¼ watt resistor
____	R3	10K ohm ¼ watt resistor
____	R4	10K ohm ¼ watt resistor
____	R5	10K ohm ¼ watt resistor
____	R6	10K ohm ¼ watt resistor
____	R7	10K ohm ¼ watt resistor
____	R8	22K ohm ¼ watt resistor
____	R9	22K ohm ¼ watt resistor
____	R10	22K ohm ¼ watt resistor
____	R11	22K ohm ¼ watt resistor
____	R12	12K ohm ¼ watt resistor
____	R13	27 ohm ¼ watt resistor
____	R14	3.3K ohm ¼ watt resistor
____	R15	10K ohm ¼ watt resistor
____	R16	47K ohm ¼ watt resistor
____	R17	10K ohm ¼ watt resistor

Capacitors

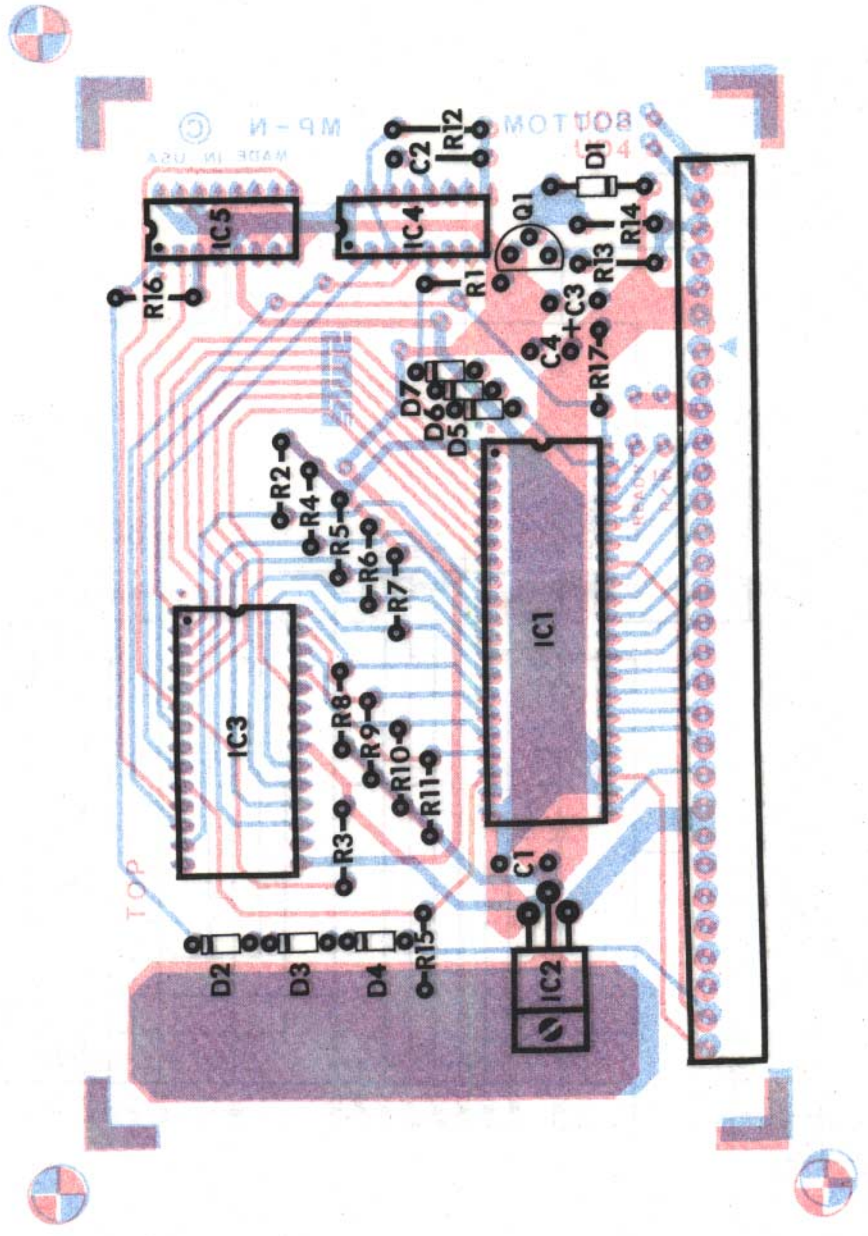
____	C1	0.1 mfd capacitor
____	C2	100 pfd capacitor
____	C3	0.1 mfd capacitor
____	C4*	10 mfd@ 15 VDC electrolytic

Diodes and Transistors

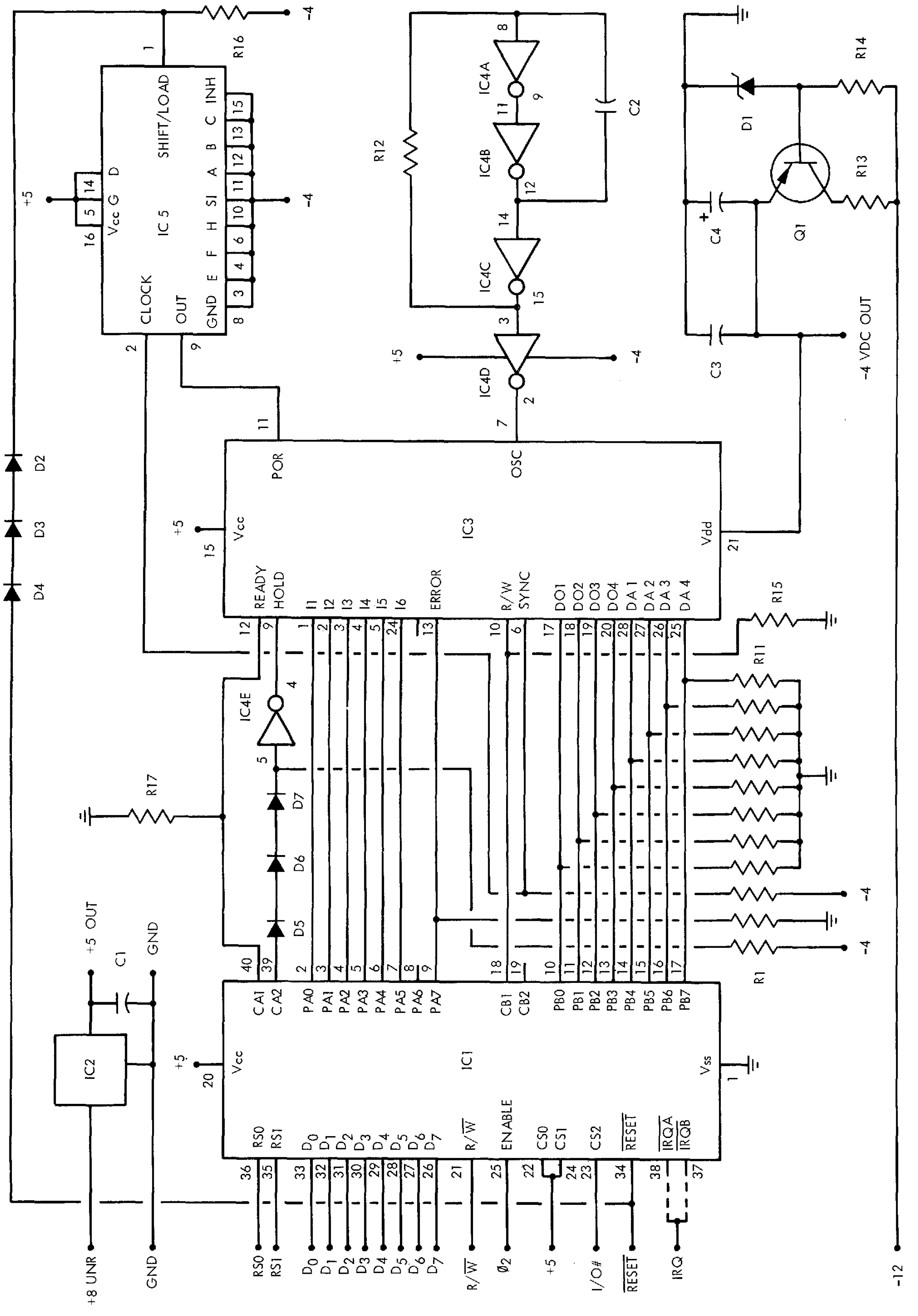
____	D1*	4.7 volt 400 mw zener diode 1N5230 or 1N4732
____	D2*	1N4148 silicon diode
____	D3*	1N4148 silicon diode
____	D4*	1N4148 silicon diode
____	D5*	1N4148 silicon diode
____	D6*	1N4148 silicon diode
____	D7*	1N4148 silicon diode
____	Ql*	2N5087 transistor

Integrated Circuits

____	IC1*	6820 MOS peripheral interface adapter
____	IC2*	7805 voltage regulator
____	IC3	MM57109 FAN MOS calculator chip
____	IC4*	4009 or 14009 MOS hex inverter
____	IC5*	74C165 MOS shift register


IC1
IC2
IC3
IC4
IC5
Q1
C1
C2
+C3
C4
D1
D2
D3
D4
D5
D6
D7
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
TOP

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 5 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 21 ORG's the program at memory location 0100 where the terminal's screen is cleared and titled. Line 25 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/0 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	8000
PORT4	8010
PORT5	8014
PORT6	8018
PORT7	801C

Lines 28 thru 37 contain the INITAL subroutine described in detail earlier. lines 38 thru 41 accept entered keyboard commands, lookup the selected calculator instructions and deposit the data or instruction in the A accumulator. Lines 46 thru 57 contain the OUTINS subroutine described in detail earlier. Lines 57 thru 76 check to see what instruction or data has been entered so the result may be output if appropriate. Line 73 looks for the TOGM instruction so the program knows which display mode to use when outputting data. Lines 79 thru 86 contain the SETMEM subroutine described in detail earlier. Since the SETMEM subroutine destroys the contents of the index register, line 87 reloads it before proceeding to the OUTANS subroutine contained in lines 90 thru 122. Line 123 checks to see of the ERROR flag was set during the last output sequence. If so, program control is transferred to lines 124 thru 137 where an error message is output and the error flag cleared by sending an ECLR instruction to the calculator chip. Line 140 tests to see if the calculator is in the floating point or scientific mode. If floating point, control is transferred to lines 142 thru 169. If scientific, control is transferred to lines 170 thru 201. In both modes the data is output to the display in the selected mode and program control is transferred back to line 38 where new commands or data may be entered

The original listing had line numbers that incremented by 10. Line 28 in this listing was line 280 in the original. The were two lines that did not increment by 10 (412 and 414) in the original. In the listing the lines above 41 are off by 2, line 53 was line 510.

```
CALC-1                                           TSC ASSEMBLER  PAGE    1

    1                            NAM    CALC-1
    2                    *A DRIVER ROUTINE FOR THE MP-N BOARD
    3                            OPT    PAG
    4  0080                      ORG    $0080
    5  0080 0F                   FCB    $0F,$0F,$0F,$0F,$0F,$0F,$0F,$0F
       0081 0F 0F
       0083 0F 0F
       0085 0F 0F
       0087 0F
    6  0088 0F                   FCB    $0F,$0F,$0F,$0F,$0F,$21,$0F,$0F
       0089 0F 0F
       008B 0F 0F
       008D 21 0F
       008F 0F
    7  0090 0F                   FCB    $0F,$0F,$0F,$0F,$0F,$0F,$0F,$0F
       0091 0F 0F
       0093 0F 0F
       0095 0F 0F
       0097 0F
    8  0098 2F                   FCB    $2F,$0F,$0F,$0F,$0F,$0F,$0F,$0F
       0099 0F 0F
       009B 0F 0F
       009D 0F 0F
       009F 0F
    9  00A0 21                   FCB    $21,$0F,$0F,$0F,$0F,$0F,$0F,$0F
       00A1 0F 0F
       00A3 0F 0F
       00A5 0F 0F
       00A7 0F
   10  00A8 0F                   FCB    $0F,$0F,$3B,$39,$0F,$3A,$0A,$3C
       00A9 0F 3B
       00AB 39 0F
       00AD 3A 0A
       00AF 3C
   11  00B0 00                   FCB    $00,$01,$02,$03,$04,$05,$06,$07
       00B1 01 02
       00B3 03 04
       00B5 05 06
       00B7 07
   12  00B8 08                   FCB    $08,$09,$0F,$0F,$0F,$0F,$22,$0F
       00B9 09 0F
       00BB 0F 0F
       00BD 0F 22
       00BF 0F
   13  00C0 0F                   FCB    $0F,$1B,$36,$25,$2D,$0B,$2C,$1C
       00C1 1B 36
       00C3 25 2D
       00C5 0B 2C
       00C7 1C
   14  00C8 1D                   FCB    $1D,$20,$0F,$0F,$0F,$18,$35,$23
       00C9 20 0F
       00CB 0F 0F
       00CD 18 35
       00CF 23
   15  00D0 0D                   FCB    $0D,$33,$37,$24,$26,$32,$34,$31
       00D1 33 37
       00D3 24 26
```

```
00D5 32 34
00D7 31
00D8 30                FCB    $30,$2B,$0C,$0F,$0F,$0F,$38,$0F
00D9 2B 0C
00DB 0F 0F
00DD 0F 38
00DF 0F
00E0 0F                FCB    $0F,$0F,$36,$25,$2D,$0B,$2C,$1C
00E1 0F 36
00E3 25 2D
00E5 0B 2C
00E7 1C
00E8 1D                FCB    $1D,$20,$0F,$0F,$0F,$18,$35,$23
00E9 20 0F
00EB 0F 0F
00ED 18 35
00EF 23
00F0 0D                FCB    $0D,$33,$37,$24,$26,$32,$34,$31
00F1 33 37
00F3 24 26
00F5 32 34
00F7 31
00F8 30                FCB    $30,$2B,$0C,$0F,$0F,$0F,$0F,$0F
00F9 2B 0C
00FB 0F 0F
00FD 0F 0F
00FF 0F
0100                   ORG    $0100
0100 8E A0 47  START   LDS    #$A047    DECREMENT STACK
0103 CE 02 87          LDX    #CLRSCN
0106 BD E0 7E          JSR    PDATA1    CLEAR AND TITLE TERM
0109 FE A0 02          LDX    PARADR
010C 8D 02             BSR    INITAL
010E 20 13             BRA    COMAND
0110 86 7F     INITAL  LDA A  #$7F      INIT A SIDE OF PIA
0112 A7 00             STA A  0,X
0114 86 36             LDA A  #$36      HIGH HOLD-POS READY
0116 A7 01             STA A  1,X
0118 86 00             LDA A  #$00      INIT B SIDE OF PIA
011A A7 02             STA A  2,X
011C 86 34             LDA A  #$34      NEG R/W
011E A7 03             STA A  3,X
0120 A6 02             LDA A  2,X       CLEAR R/W FLAG
0122 39                RTS
0123 BD E1 AC  COMAND  JSR    INEEE     GET OPERATOR DATA
0126 8A 80             ORA A  #$80      POSITION TO TOP OF TABLE
0128 B7 01 2C          STA A  POINT+1
012B 96 00     POINT   LDA A  $00       SELF MODIFING CODE
012D 81 21             CMP A  #$21
012F 27 43             BEQ    ZERMEM
0131 8D 02             BSR    OUTINS
0133 20 17             BRA    CHRCHK
0135 E6 01     OUTINS  LDA B  1,X       WAIT FOR READY
0137 2A FC             BPL    OUTINS
```

```
0139 A7 00             STA A  0,X       FORWARD INSTRUCTION TO CALC
013B E6 00             LDA B  0,X       CLEAR FLAG BIT
013D C6 3C             LDA B  #$3C      LOW HOLD-NEG READY
013F E7 01             STA B  1,X       BRING HOLD LINE LOW
0141 E6 01     WAIT10  LDA B  1,X
0143 2A FC             BPL    WAIT10    LOOP FOR READY LOW
0145 E6 00             LDA B  0,X       CLEAR FLAG BIT
0147 C6 36             LDA B  #$36      HIGH HOLD-POS READY
0149 E7 01             STA B  1,X       RETURN HOLD LINE HIGH
014B 39                RTS
014C 81 2F     CHRCHK  CMP A  #$2F
014E 26 03             BNE    SKIP75
0150 7F 02 AE          CLR    FORMAT
0153 7D 02 AF  SKIP75  TST    SMDC      CHECK FOR PREVIOUS SMDC INSTR
0156 26 1C             BNE    ZERMEM
0158 81 0F     CONT50  CMP A  #$0F
015A 27 C7             BEQ    COMAND    GET MOR DATA IF NOP
015C 81 18             CMP A  #$18
015E 26 05             BNE    SKIP25
0160 73 02 AF          COM    SMDC
0163 20 BE             BRA    COMAND    GET MORE DATA IF SMDC
0165 81 20     SKIP25  CMP A  #$20
0167 27 BA             BEQ    COMAND    GET MORE DATA IF INV
0169 81 0B             CMP A  #$0B
016B 23 B6             BLS    COMAND    GET MORE DATA IF NUMBERS
016D 81 22             CMP A  #$22      LOOK FOR TOGM
016F 26 03             BNE    ZERMEM
0171 73 02 AE          COM    FORMAT
0174 7F 02 AF  ZERMEM  CLR    SMDC      ZERO SMDC
0177 8D 02             BSR    SETMEM
0179 20 11             BRA    LODADR
017B 7F 00 20  SETMEM  CLR    $20       CLEAR $0020
017E CE 00 20          LDX    #$20      BOTTOM OF BUFFER
0181 C6 20             LDA B  #$20
0183 08        LOOP1   INX
0184 E7 00             STA B  0,X       STORE A SPACE
0186 8C 00 2B          CPX    #$2B      CHEXC FOR TOP OF BUFFER
0189 26 F8             BNE    LOOP1
018B 39                RTS
018C FE A0 02  LODADR  LDX    PARADR
018F 8D 02             BSR    OUTANS
0191 20 3D             BRA    OUTCHR
0193 E6 01     OUTANS  LDA B  1,X
0195 2A FC             BPL    OUTANS
0197 A6 00             LDA A  0,X       CLEAR FLAG BIT
0199 86 16             LDA A  #$16      SEND AN OUT
019B A7 00             STA A  0,X
019D C6 3E             LDA B  #$3E      LOW HOLD-POS READY
019F E7 01             STA B  1,X       BRING HOLD LINE LOW
01A1 E6 01     WAIT30  LDA B  1,X       WAIT FOR SECOND READY
01A3 2A FC             BPL    WAIT30
01A5 E6 00             LDA B  0,X       CLEAR FLAG BIT
01A7 86 0F             LDA A  #$0F
01A9 A7 00             STA A  0,X       SEND A NOP
```

```
01AB E6 03     WAIT3   LDA B  3,X       LOOK FOR R/W STROBE
01AD 2B 06             BMI    OUTDIG    TRANSFER CALC DATA INTO MEMORY
01AF E6 01             LDA B  1,X       LOOK FOR READY STROBE
01B1 2B 16             BMI    CONFLG    PRINT MEMORY CONTENTS
01B3 20 F6             BRA    WAIT3
01B5 A6 02     OUTDIG  LDA A  2,X       LOAD OUT DATA INTO A
01B7 16                TAB
01B8 84 0F             AND A  #$0F      ELIMINATE UPPER 4 BITS
01BA 8A 30             ORA A  #$30      CONVERT TO ASCII DATA
01BC 54                LSR B
01BD 54                LSR B
01BE 54                LSR B
01BF 54                LSR B
01C0 CA 20             ORA B  #$20      INCREMENT ADDRESSES BY $20
01C2 F7 01 C6          STA B  POINT2+1  STORE OUT DATA SEQUENTIALLY
01C5 97 00     POINT2  STA A  $0        SELF MODIFING CODE
01C7 20 E2             BRA    WAIT3
01C9 86 36     CONFLG  LDA A  #$36      HIGH HOLD-POS READY
01CB A7 01             STA A  1,X       BRING HOLD LINE HIGH
01CD A6 00             LDA A  0,X       CLEAR FLAG BIT
01CF 39                RTS
01D0 2A 1E     OUTCHR  BPL    CONT1     SKIP IF NO ERROR
01D2 E6 01     WAIT70  LDA B  1,X       WAIT FOR READY
01D4 2A FC             BPL    WAIT70
01D6 86 2B             LDA A  #$2B      ERROR CLEAR INSTRUCTION
01D8 A7 00             STA A  0,X
01DA E6 00             LDA B  0,X       CLEAR FLAG BIT
01DC C6 3C             LDA B  #$3C      LOW HOLD-NEG READY
01DE E7 01             STA B  1,X       BRING HOLD LOW
01E0 E6 01     WAIT71  LDA B  1,X
01E2 2A FC             BPL    WAIT71
01E4 E6 00             LDA B  0,X       CLEAR FLAG BIT
01E6 C6 36             LDA B  #$36      HIGH HOLD-POS READY
01E8 E7 01             STA B  1,X       RETURN HOLD HIGH
01EA CE 02 B0          LDX    #ERRMSG
01ED BD E0 7E          JSR    PDATA1
01F0 CE 02 A8  CONT1   LDX    #CRLF
01F3 BD E0 7E          JSR    PDATA1
01F6 7D 02 AE          TST    FORMAT
01F9 2B 3F             BMI    SCINOT
01FB CE 00 22  FLOPNT  LDX    #$22      FLOTING POINT NOTATION
01FE A6 00             LDA A  0,X       INPUT MANTISSA SIGN DATA
0200 84 08             AND A  #$08      MASK BIT 4
0202 26 04             BNE    MINPNT
0204 86 20             LDA A  #$20      LOAD A SPACE
0206 20 02             BRA    PRINT1
0208 86 2D     MINPNT  LDA A  #$2D      LOAD MINUS
020A BD E1 D1  PRINT1  JSR    OUTEEE    PRINT CHARACTER
020D 08        DPIND   INX
020E E6 00             LDA B  0,X
0210 C4 0F             AND B  #$0F
0212 E7 00             STA B  0,X
0214 C6 2F             LDA B  #$2F
0216 E0 00             SUB B  0,X
```

```
0218 D7 21              STA B  $21       STORE DEC PT POSITION IND
021A 08         DIGLOP  INX
021B A6 00              LDA A  0,X
021D BD E1 D1           JSR    OUTEEE    OUTPUT ASCII NUMBER
0220 9C 20              CPX    $20       TIME FOR DEC PT
0222 26 05              BNE    ENDCH1
0224 86 2E              LDA A  #$2E
0226 BD E1 D1           JSR    OUTEEE
0229 8C 00 2B  ENDCH1   CPX    #$2B      CHECK FOR LAST DIGIT
022C 26 EC              BNE    DIGLOP    GET NEXT DIGIT
022E CE 02 A8           LDX    #CRLF
0231 BD E0 7E           JSR    PDATA1    PRINT CR/LF
0234 FE A0 02           LDX    PARADR
0237 7E 01 23           JMP    COMAND
023A 96 22      SCINOT  LDA A  $22       SCIENTIFIC NOTATION
023C 84 08              AND A  #$08      LOOK FOR NEGATIVE MANTISSA
023E 26 04              BNE    NEGPNT
0240 86 20              LDA A  #$20      SPACE IF NOT
0242 20 02              BRA    PRINT2
0244 86 2D      NEGPNT  LDA A  #$2D
0246 BD E1 D1   PRINT2  JSR    OUTEEE    PRINT SIGN
0249 CE 00 23           LDX    #$23
024C 08         NUMLOP  INX
024D A6 00              LDA A  0,X
024F BD E1 D1           JSR    OUTEEE
0252 8C 00 24           CPX    #$24      LOOK FOR DEC PT DIGIT
0255 26 05              BNE    SKIPDP
0257 86 2E              LDA A  #$2E
0259 BD E1 D1           JSR    OUTEEE    PRINT DEC PT
025C 8C 00 2B   SKIPDP  CPX    #$2B      CHECK FOR LAST DIGIT
025F 26 EB              BNE    NUMLOP
0261 86 45              LDA A  #$45
0263 BD E1 D1           JSR    OUTEEE    PRINT AN E
0266 96 22              LDA A  $22       LOAD SIGN BYTE
0268 84 01              AND A  #$01
026A 27 05              BEQ    SKPSGN
026C 86 2D              LDA A  #$2D
026E BD E1 D1           JSR    OUTEEE    PRINT A -
0271 96 20      SKPSGN  LDA A  $20
0273 BD E1 D1           JSR    OUTEEE    PRINT EXPONENT MSD
0276 96 21              LDA A  $21
0278 BD E1 D1           JSR    OUTEEE    PRINT EXPONENT LSD
027B CE 02 A8           LDX    #CRLF
027E BD E0 7E           JSR    PDATA1    PRINT CR/LF
0281 FE A0 02           LDX    PARADR
0284 7E 01 23           JMP    COMAND
0287 0D         CLRSCN  FCB    $0D,$0A,$10,$16,$00
0288 0A 10
028A 16 00
028C 53                 FCC    /SWTPC 6800 CALC-1 CALCULATOR/
028D 57 54
028F 50 43
0291 20 36
0293 38 30
```

```
CALC-1                                          TSC ASSEMBLER  PAGE    6

       0295 30 20
       0297 43 41
       0299 4C 43
       029B 2D 31
       029D 20 43
       029F 41 4C
       02A1 43 55
       02A3 4C 41
       02A5 54 4F
       02A7 52
       02A8 0D         CRLF    FCB    $0D,$0A,$00,$00,$00,$04
       02A9 0A 00
       02AB 00 00
       02AD 04
       02AE 00         FORMAT  FCB    $00
       02AF 00         SMDC    FCB    $00
       02B0 0D         ERRMSG  FCB    $0D,$0A,$00,$00
       02B1 0A 00
       02B3 00
       02B4 45                 FCC    /ERROR/
       02B5 52 52
       02B7 4F 52
       02B9 04                 FCB    $04
       E07E            PDATA1  EQU    $E07E
       A002            PARADR  EQU    $A002
       E1AC            INEEE   EQU    $E1AC
       E1D1            OUTEEE  EQU    $E1D1
       A002                    ORG    $A002
       A002 80 0C              FDB    $800C
                               END    START

NO ERROR(S) DETECTED


CALC-1                                          TSC ASSEMBLER  PAGE    7


   SYMBOL TABLE:

CHRCHK 014C   CLRSCN 0287   COMAND 0123   CONFLG 01C9   CONT1  01F0
CONT50 0158   CRLF   02A8   DIGLOP 021A   DPIND  020D   ENDCH1 0229
ERRMSG 02B0   FLOPNT 01FB   FORMAT 02AE   INEEE  E1AC   INITAL 0110
LODADR 018C   LOOP1  0183   MINPNT 0208   NEGPNT 0244   NUMLOP 024C
OUTANS 0193   OUTCHR 01D0   OUTDIG 01B5   OUTEEE E1D1   OUTINS 0135
PARADR A002   PDATA1 E07E   POINT  012B   POINT2 01C5   PRINT1 020A
PRINT2 0246   SCINOT 023A   SETMEM 017B   SKIP25 0165   SKIP75 0153
SKIPDP 025C   SKPSGN 0271   SMDC   02AF   START  0100   WAIT10 0141
WAIT3  01AB   WAIT30 01A1   WAIT70 01D2   WAIT71 01E0   ZERMEM 0174
```

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

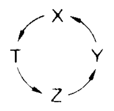
CLASS	SUBCLASS	MNEMONIC	OCTAL OP CODE	FULL NAME	DESCRIPTION
Digit Entry		0	00	0	Mantissa or exponent digits. On first digit (d) the following occurs:
		1	01	1	Z → T
		2	02	2	Y → Z
		3	03	3	X → Y
		4	04	4	d → X
		5	05	5	See description of number entry on page 11
		6	06	6	
		7	07	7	
		8	10	8	
		9	11	9	
		DP	12	Decimal Point	Digits that follow will be mantissa fraction.
		EE	13	Enter Exponent	Digits that follow will be exponent.
		CS	14	Change Sign	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.
		PI	15	Constant ∏	3.1415927 → X, stack not pushed.
		EN	41	Enter	Terminates digit entry and pushes the stack. The argument entered will be in X and Y. Z → T Y → Z X → Y
		NOP	77	No 0peration	Do nothing instruction that will terminate digit entry.
		HALT	17	Halt	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.
Move		ROLL	43	Roll	Roll Stack, 
		POP	56	Pop	Pop Stack. Y → X Z → Y T → Z O → T
		XEY	60	X exchange Y	Exchange X and Y. X ↔ Y
		XEM	33	X exchange M	Exchange X with memory. X ↔ M
		MS	34	Memory Store	Store X in Memory. X → M
		MR	35	Memory Recall	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
Branch	Count	IBNZ	31	Increment memory and branch if M ≠ 0	M + 1 → 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 ≠ 0	M - 1 → M. If M = 0, skip second instruction word. Otherwise, branch to address specified by second instruction word.
I/0	Multi-digit	IN*	27	Multidigit input to X	The processor supplies a 4-bit digit address (DA4-DA1) accompanied by a digit address strobe (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 notation or floating point) and the mantissa digit count (See Data Formats and Instruction Timing). Data to be input s stored in X and the stack is pushed (X → Y → Z → 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 the output on D04-D01 and pulses the R/W line active low. At the conclusion of output, D04-D01=0 and DA4-DA1=0.
I/0	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 ADR = 0 (ISEL = 0 selects ADR instead of I_5), indicating data valid F2 is pulsed active low to acknowledge data just read.
I/0	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 ahead y high, this results in it being set low.
		PRW1	75	Pulse R/W 1	Generates R/W active low pulse which may be used as a strobe or to clock extra instruction bits into a flip-flop or register.
		PRW2	76	Pulse R/W 2	Identical to PRW1 instruction. Advantage may be taken of the fact that the last 2 hits 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 R/W pulse.
Mode Control		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 (Continued) (* Indicates 2-word instruction)

CLASS	SUBCLASS	MNEMONIC	OCTAL OP CODE	FULL NAME	DESCRIPTION
Math	F(X,Y)	+	71	Plus	Add X to Y. X + Y → X On +, -, x, /, and YX instructions stack is popped as follows Z → Y T → Z O → T Former X, Y are lost.
		-	72	Minus	Subtract X from Y. Y - X → X I
		x	73	Times	Multiply X times Y. Y x X → X
		/	74	Divide	Divide X into Y. Y / X → X
		YX	70	Y to X	Raise Y to X power $Y^X \rightarrow X$
	F(X,M)	INV+*	40, 71	Memory Plus	Add X to memory M + X → M On INV +, -, x, and / instructions, X, Y, Z and T are unchanged.
		INV-*	40, 72	Memory Minus	Subtract X from memory. M - X → M
		INVx*	40, 73	Memory Times	Multiply X times memory. M x X → M
		INV/*	40, 74	Memory Divide	Divide X into memory. M / X → M
	F(X) Math	1/X	67	One Divided by X	1 / X → X. On all F(X) math Instructions Y, Z, T and M are unchanged and previous X is lost.
		SORT	64	Square Root	√X → X
		SQ	63	Square	$X^2 \rightarrow X$
		10X	62	Ten to X	$10^X \rightarrow X$
		EX	61	E to X	$e^X \rightarrow X$
		LN	65	Natural log of X	ln X → X
		LOG	66	Base 10 log of X	log X → X
	F(X) Trig	SIN	44	Sine X	SIN(X) → X On all F(X) trig functions Y, Z, T and M are unchanged and the previous X is lost
		COS	45	Cosine X	COS(X) → X
		TAN	46	Tangent X	TAN(X) → X
		INV SIN *	40, 44	Inverse sine X	$SIN^{-1}(X) \rightarrow X$
		INV COS*	40,4 5	Inverse cosine X	$COS^{-1}(X) \rightarrow X$
		INV TAN*	40,46	Inverse tan X	$TAN^{-1}(X) \rightarrow X$
		DTR	55	Degrees to radians	Convert X from degrees to radians.
		RTD	54	Radians to degrees	Convert X from radians to degrees.
Clear		MCLR	5 7	Master Clear	Clear all internal registers and memory, initialize I/O control signals, MDC = 8, MODE floating point. (See initialization.)
		ECLR	53	Error flag clear	0 → Error flag
Branch	Test	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 (16) 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 traction.)
		TXLTO*	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)

I_4 -I_1	I_6I_5			
	0	1	2	3
0	0	TJC*	IN V	XEY
1	1	TX=O*	EN	EX
2	2	TXLTO*	TOG M	10X
3	3	TXF*	ROLL	SQ
4	4	TERR*	SIN(SIN-1)	SORT
S	5	JMP	COS(COS-1)	LN
6	6	OUT*	TAN(TAN-1)	LOG
7	7	IN*	SF1	1,X
8	8	SMDC*	PF1	YX
9	9	IBNZ*	SF2	+(M+)
A	DP	DBNZ*	PF2	-(M-)
B	EE	XEM	ECLR	x(Mx)
C	CS	MS	RTD	/(M/)
D	PI	MR	DTR	PRW1
E	AIN	LSH	POP	PRW2
F	HALT	RSH	MCLR	NOP

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

FULL NAME	HEX OP CODE	MNEMONIC	ASCII CHARACTER
0	00	00	0
1	01	01	1
2	02	02	2
3	03	03	3
4	04	04	4
5	05	05	5
6	06	06	6
7	07	07	7
8	08	08	8
9	09	09	9
Decimal Point	OA	DP	
Enter Exponent	OB	EE	E
Change Sign	OC	CS	Z
Constant PI	OD	PI	P
Asynchronous Input	OE	AIN	
Halt	OF	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	
Set Mantissa Digit Count	18	SMDC	M
Inc & Branch if M≠0	19	IBNZ	
Dec & Branch if M=0	lA	DBNZ	
X Exchange M	1B	XEM	A
Memory Store	1C	MS	G
Memory Recall	1D	MR	H
Left shift Xm	lE	LSH	
Right shift Xm	1F	RSH	
Inverse Mode	20	INV	I
Enter	21	EN	
Toggle Mode	22	TOGM	
Roll Stack	23	ROLL	0
Sine X	24	SIN	S
Cosine X	25	COS	C
Tangent X	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	Cntrl X

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

NAME	HEX OP CODE	MNEMONIC	ASCII CHARACTER
X exchange Y	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	IN	N
Base 10 Log of X	36	LOG	B
One divided by X	37	1/X	R
Y to X	38	YX	n
Plus	39	+	+
Minus	3A	-	-
Times	3B	k	
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

Memory Location	DP POS	D7	D6	D5	D4	D3	D2	D1	D0
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	OB	0	0	1	1	BCD digit(left most)			
25	OA	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 IQ - Scientific Mode OUT data storage

Memory Location	D7	D6	D5	D4	D3	D2	D1	D0
20	0	0	1	1	Most significant exp. digit			
21	0	0	1	1	Least significant 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	00	1	1	BCD digit (left most)			

Notes:

1) If the Mantissa significant Digit Count (set by SMDC instruction, initially 8) is less than 8, the unused digit memory locations will be filled with ASCII spaces (20_{16})

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

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

Example: An ASCII P is a hex 50 which points in the table to a OD 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 90^0 , 270^0, 450^0 , etc.

4) SIN X, COS X, TAN X when $|X| \geq 9000^0$

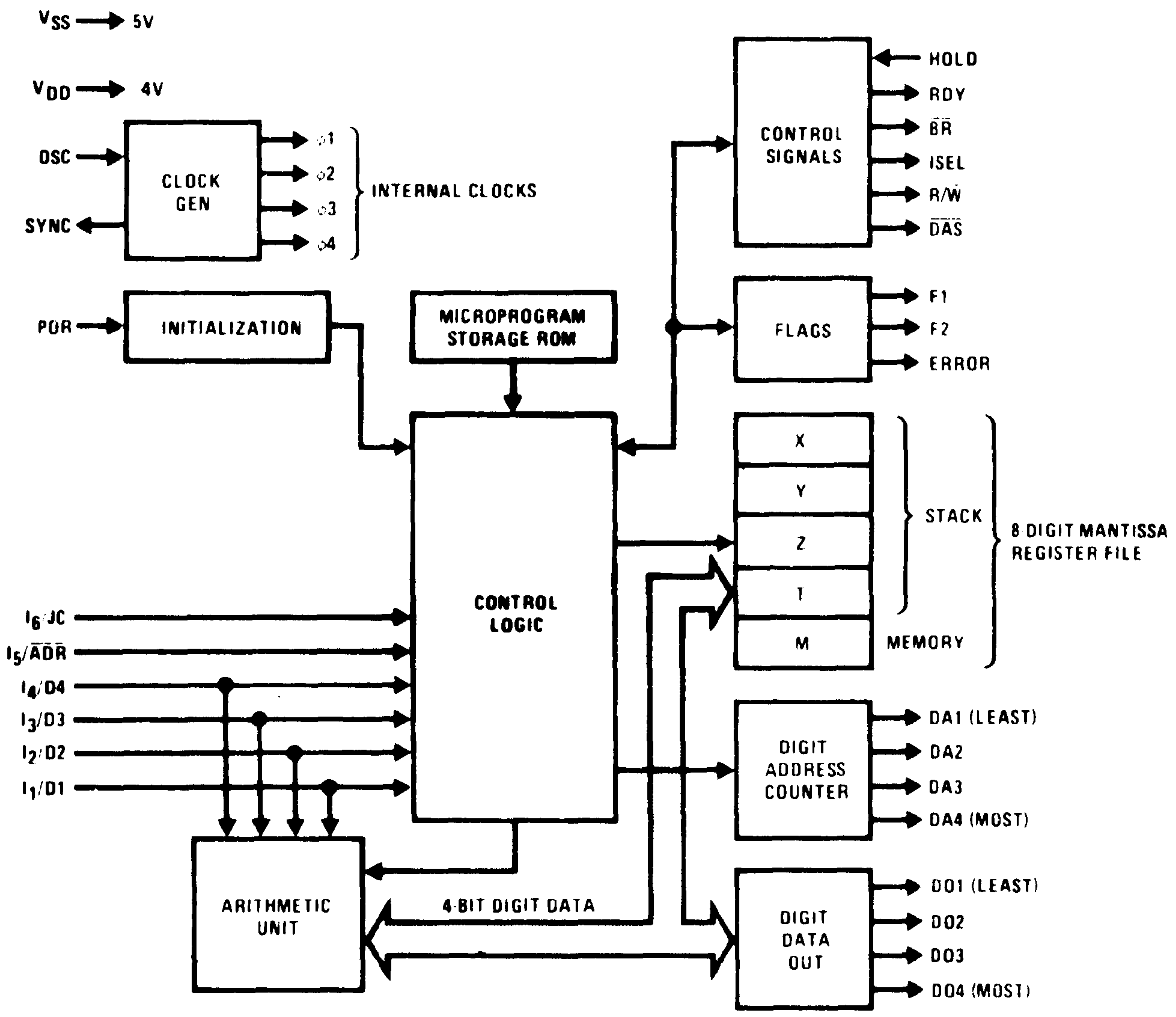
5) SIN^{-1} X, COS^{-1} X when $|X| > 1$ $|X| < 10^{-50}$

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 I


VSS 5V
VDD 4V
OSC
SYNC
CLOCK GEN
φ1
φ2
φ3
φ4
INTERNAL CLOCKS
POR
INITIALIZATION
MICROPROGRAM STORAGE ROM
CONTROL LOGIC
CONTROL SIGNALS
HOLD
RDY
BR
ISEL
R/W
DAS
FLAGS
F1
F2
ERROR
X
Y
Z
T
M
STACK
MEMORY
8 DIGIT MANTISSA REGISTER FILE
I6/JC
I5/ADR
I4/D4
I3/D3
I2/D2
I1/D1
ARITHMETIC UNIT
4-BIT DIGIT DATA
DIGIT ADDRESS COUNTER
DA1 (LEAST)
DA2
DA3
DA4 (MOST)
DIGIT DATA OUT
DO1 (LEAST)
DO2
DO3
DO4 (MOST)