# COMPUTE

the Club Of Microprocessor Programmers, Users, and Technical Experts

Georgia Marszalek, Editor • David Graves, Editor • Doug Hall, Hardware Consultant

Sponsored by National Semiconductor Corp., Santa Clara, CA. 95051 Vol. 3, No. 6, June 1977


## SUPPORT CIRCUITS, FASTER 8080A'S ADDED

Less than one year ago we entered the 8080A marketplace with our INS8080A – a pin-for-pin, function-for-function replacement for you-know-who's MPU. But that was only the start. Since then we've added two more versions of that microprocessor, as well as a complete family of support circuits.

The new versions of our original 2-$\mu$s cycle time INS8080A are the INS8080A-1, which has a 1.3-$\mu$s cycle time, and the INS8080A-2, with a 1.5-$\mu$s cycle time.

In addition to the faster 8080A's, we now offer ten types of interface circuits to support 8080A system design.

- DP 8212 is a 8-bit I/O port that you can use to implement all major peripheral and MPU I/O system functions.
- INS8255 is a programmable peripheral I/O interface that features direct bit set/reset capability.
- DP8301 is a microprocessor interface latch element (MILE) with on-chip status flags for 'handshake' control and interrupt generation. It drives TTL, NMOS, PMOS, and CMOS circuitry.
- DP8224 is a crystal-controlled clock generator and driver, which also provides a status strobe and oscillator outputs for external circuits.
- DP8228/8238 are system controller and bus driver circuits that generate all needed read/write control signals, provide drive and isolation for the 8080A's bidirectional data bus, and a user-selected single-level interrupt vector.
- DP8304 is an 8-bit bidirectional bus transceiver with high active outputs to both ports, a Tri-State® chip enable control, and transmit/receive control.
- INS8251 is a universal communications interface (USART) for data communication in 8080A and other bus-structured systems.
- DP8216/8226 are I/O buffer drivers (4-bit parallel transceivers) suited to both 8080A and general MPU applications.

# Line-By-Line Resident Assembler For "SC/MP" MPU Enhances Development System

National is now offering a line-by-line resident assembler firmware kit, designed for use with its SC/MP LCDS (Low Cost Development System). Known as SUPAK, the $300 assembler is contained in eight PROM/ROM devices that can be plugged into a blank ROM/PROM card (ISP-8B/004B) which is available from National. The entire assembled card is then inserted into the LCDS/teletype system.

SUPAK is the only firmware kit of its kind. It greatly enhances the flexibility and capability of the SC/MP LCDS, making it both the most inexpensive and effective SC/MP development tool. SUPAK is a 4K byte package that consists of three programs: a line-by-line assembler, a paper tape line editor and a PROM tape punch program.

The line-by-line assembler accepts a program in limited assembly language from a keyboard or paper reader, and then assembles it directly into RAM.

The paper tape line editor, which allows insertion, deletion or replacement of lines of program source code, punches either leader or trailer.

The PROM tape punch program punches the contents of a specified memory range, in BPNF or complemented binary format, onto paper tape. This tape could be used to program memories using a standard, commercially available PROM programmer such as the DATA I/O.

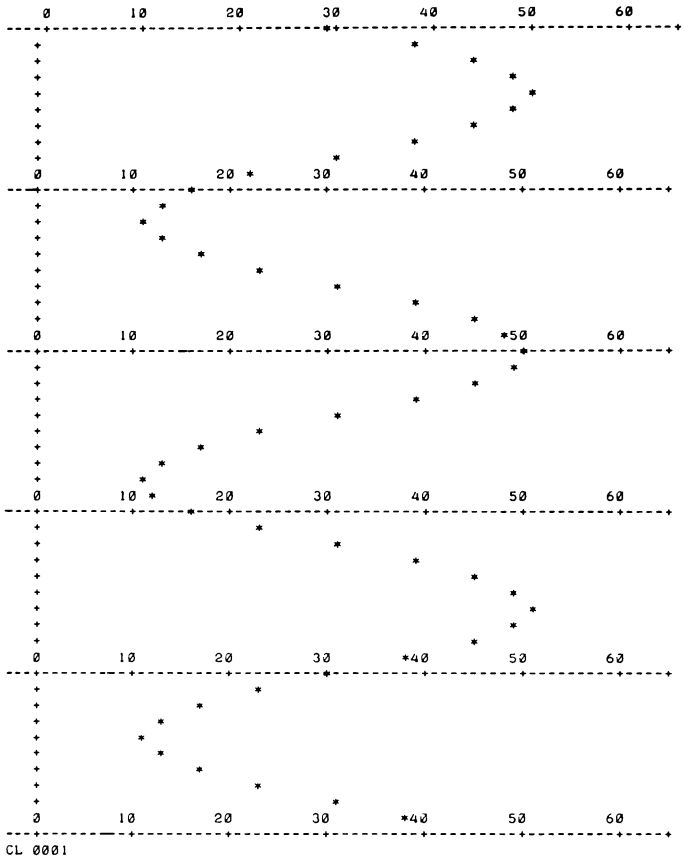
SUPAK requires the LCDS firmware, but will run on either a SC/MP or a SC/MP II (n-channel) LCDS. It comes as a set of eight MM5204/MM5214 ROM IC's, designated ISP-8F/111.

# NEW LIBRARY PROGRAM

SC/MP Program SL0047A-PLOT
(from Jermyn MicroComputer Center, Kent, England)
Program Listing — see page 3
Source paper tape — $5.00 each

A sample plot of a SINE wave drawn on a Teletype by the PLOT routine is shown below.

**GRAPH PLOTTER FROM RAM A JERMYN UTILITY PROGRAM**




## Programming Tidbit

When using the PACE or IMP-16 you may use a version of the following assembly instruction to load an ASCII character into one of the Accumulators

LI Ø, '9'/256

an ASCII instruction will result in object code that will result in an ASCII 9 or X'39 to be loaded left justified in ACØ when executed.

## SM/PL TIMER PROGRAM

(by Bob Edwards
LECO Corporation
(616) 983-5531)

This is a SM/PL program that works with a 1200 baud CRT. This program prints the time from 0 to 99 hrs 59 mins and 59 secs. It was developed using an IMP-16P with 16K and the IMP-16 Disc system.

```
 1   /* TIMEP PROGRAM FOR UP TO 99HRS 59MIN 59SEC */;
 2   ;
 3   DECLARE (A, B, C, D, E, F, I) WORD;
 4   DECLARE (AA, AB, AC, AD, AE, AF) WORD;
 5   DECLARE SP LITERALLY '20H';
 6   DECLARE CR LITERALLY '0DH';
 7   DECLARE LF LITERALLY '0AH';
 8   PUTC:ASMPROC(1, 7E59H);
 9   ST: DO I=0 TO 25;
10       CALL PUTC(LF);
11       END ST;
12   FL: DO F=0 TO 9;
13   AF=F+30H;
14     EL:DO E=0 TO 9;
15     AE=E+30H;
16       DL:DO D=0 TO 5;
17       AD=D+30H;
18         CL:DO C=0 TO 9;
19         AC=C+30H;
20           BL:DO B=0 TO 5;
21           AB=B+30H;
22             AL:DO A=0 TO 9;
23             AA=A+30H;
24             CALL TIME(6500);
25               SL:DO I=0 TO 30;
26               CALL PUTC(SP);
27               END SL;
28             CALL PUTC(AF);
29             CALL PUTC(AE);
30             CALL PUTC(SP);
31             CALL PUTC(SP);
32             CALL PUTC(AD);
33             CALL PUTC(AC);
34             CALL PUTC(SP);
35             CALL PUTC(SP);
36             CALL PUTC(AB);
37             CALL PUTC(AA);
38             CALL PUTC(CR);
39             CALL PUTC(LF);
40             END AL;
41           END BL;
42         END CL;
43       END DL;
44     END EL;
45   END FL;
46   EOF ST;

 ?
```

# *SM/PL-A High Level Language For IMP-16*

A few words on the SM/PL Compiler for IMP-16 . . .

- SM/PL (Smart or Simple Programming Language) is a high level language compiler for the IMP-16 only.
- To compile a SM/PL program and accommodate the compiler 16K of R/W memory is necessary.
- It is not a supported product and is only available through the Microprocessor Users Group at a cost of $100.00.
- The $100.00 price tag includes the source listing, SM/PL Programming Manual and the object (machine language) tape.
- Ordering SM/PL is done by sending a check or Purchase Order to Compute/208, National Semiconductor, 2900 Semiconductor Dr, Santa Clara, Ca. 95051 (see the library order form on page 15).
- The August 1976 issue of the COMPUTE (Vol. 2 #8) newsletter contains a description of the language features.

```
                .TITLE PLOT,'SC/MP PLOTTER FROM RAM'
                ;LIBRARY PROGRAM SL0047A
                ;
                ; THIS PROGRAM FETCHES DATA STORED IN RAM
                ; AND DISPLAYS THE CONTENT IN GRAPHICAL
                ; FORM ON THE TELETYPE CONSOLE.
                ; AFTER LOADING SET THE TARGET PROGRAM
                ; COUNTER TO HEX 0001 (INITIALISE) AND
                ; THEN RUN.
                ;
                ; THIS PROGRAM WILL RUN WITTHIN AN L.C.D.S
                ; CONTAINING 2K BYTES OF RAM (ISP-8C/002).
                ; DATA FOR PLOTTING IS FETCHED FROM LOCATION
                ; MDATA UPWARDS AND LOOPS AROUND AT LOCATION
                ; DATEND.
                ;
        0001    P1      =       1
        0002    P2      =       2
        0003    P3      =       3
        07FD    VECTOR  =       X'07FD
        07FF    COUNT   =       X'07FF
        0000    ZERO    =       X'00
        0004    EOT     =       X'04
        07FC    STACK   =       X'07FC
        7AE2    PUTC    =       X'7AE2
        000D    CR      =       X'0D
        000A    LF      =       X'0A
        0008    BS      =       X'08
        FFFF    MINONE  =       -1
        FFF6    MINTEN  =       -10
        FFBF    MIN65   =       -65
        0020    SP      =       X'20
        0007    BEL     =       X'07
        003F    RSTCYL  =       X'3F

0000                    .=X'0001
0001 C4FC   BEGIN:  LDI     L(STACK)
0003 32             XPAL    P2
0004 C407           LDI     H(STACK)
0006 36             XPAH    P2
0007 C4FD           LDI     L(VECTOR)
0009 31             XPAL    P1
000A C407           LDI     H(VECTOR)
000C 35             XPAH    P1
000D C402           LDI     H(MDATA)
000F CD01           ST      @1(P1)
0011 C42F           LDI     L(MDATA)
0013 C900           ST      (P1)
0015 C454           LDI     L(MIAXIS)
0017 31             XPAL    P1
0018 C401           LDI     H(MIAXIS)
001A 35             XPAH    P1
001B C400           JS      P3,PRINT
001D 37C4
001F D333
0021 3F
0022 C400           LDI     L(MCRLF)
0024 31             XPAL    P1
0025 C401           LDI     H(MCRLF)
0027 35             XPAH    P1
0028 C400           JS      P3,PRINT
002A 37C4
002C D333
002E 3F
002F C4FF   CLEAR:  LDI     L(COUNT)
0031 31             XPAL    P1
0032 C407           LDI     H(COUNT)
0034 35             XPAH    P1
0035 C400           LDI     ZERO
0037 C900           ST      (P1)
0039 C4D8           LDI     L(MCURSR)
003B 31             XPAL    P1
003C C401           LDI     H(MCURSR)
003E 35             XPAH    P1
003F 9033           JMP     PRICYC
0041 9CEC   NOSAX:  JNZ     CLEAR
0043 C401           LDI     H(MAXIS)
0045 35             XPAH    P1
0046 C491           LDI     L(MAXIS)
0048 31             XPAL    P1
0049 9029           JMP     PRICYC
004B C407   OVER:   LDI     L(MOVER)
004D 31             XPAL    P1
004E C401           LDI     H(MOVER)
0050 35             XPAH    P1
0051 9006           JMP     PRIRET
0053 C403   POINT:  LDI     L(MPOINT)
0055 31             XPAL    P1
0056 C401           LDI     H(MPOINT)
0058 35             XPAH    P1
0059 C400   PRIRET: JS      P3,PRINT
005B 37C4
005D D333
005F 3F
0060 C407   RETURN: LDI     H(COUNT)
0062 35             XPAH    P1
0063 C4FF           LDI     L(COUNT)
0065 31             XPAL    P1
0066 A900           ILD     (P1)
0068 C900           ST      (P1)
006A F4F6           ADI     MINTEN
006C 94D3           JP      NOSAX
006E C402           LDI     H(MSAXIS)
0070 35             XPAH    P1
0071 C424           LDI     L(MSAXIS)
0073 31             XPAL    P1
0074 C400   PRICYC: JS      P3,PRINT
0076 37C4
0078 D333
007A 3F
007B C4FD   CYCLE:  LDI     L(VECTOR)
007D 31             XPAL    P1
007E C407           LDI     H(VECTOR)
0080 35             XPAH    P1
0081 C501           LD      @1(P1)
0083 01             XAE
0084 C100           LD      (P1)
0086 31             XPAL    P1
0087 40             LDE
0088 35             XPAH    P1
0089 40             LDE
008A E402           XRI     H(DATEND)
008C 9C0F           JNZ     FETCH
008E 31             XPAL    P1
008F 01             XAE
0090 40             LDE
0091 31             XPAL    P1
0092 40             LDE
0093 E43F           XRI     L(DATEND)
0095 9C06           JNZ     FETCH
0097 C402           LDI     H(MDATA)
0099 35             XPAH    P1
009A C42F           LDI     L(MDATA)
009C 31             XPAL    P1
009D C501   FETCH:  LD      @1(P1)      ;DATA FETCH
009F CEFF           ST      @-1(P2)     ;AND TEST
00A1 C4FD           LDI     L(VECTOR)   ;ROUTINE
00A3 31             XPAL    P1
00A4 01             XAE
00A5 C407           LDI     H(VECTOR)
00A7 35             XPAH    P1
00A8 CD01           ST      @1(P1)
00AA 01             XAE
00AB C900           ST      (P1)
00AD C601           LD      @1(P2)
00AF 98A2           JZ      POINT
00B1 01             XAE
00B2 06             CSA
00B3 D43F           ANI     RSTCYL
00B5 07             CAS
00B6 40             LDE
00B7 F4BF           ADI     MIN65
00B9 9490           JP      OVER
00BB C4FE   SPACE:  LDI     L(MSPACE)
00BD 31             XPAL    P1
00BE C400           LDI     H(MSPACE)
00C0 35             XPAH    P1
00C1 C400           JS      P3,PRINT
00C3 37C4
00C5 D333
00C7 3F
00C8 06             CSA
00C9 D43F           ANI     RSTCYL
00CB 07             CAS
00CC 40             LDE
00CD F4FF           ADI     MINONE
00CF 9882           JZ      POINT
00D1 01             XAE                 ;DATA TO EXT
00D2 90E7           JMP     SPACE
                ;
00D4 01     PRINT:  XAE                 ;MESSAGE PRINT
00D5 CEFF           ST      @-1(P2)     ;ROUTINE INDEXED
00D7 06             CSA                 ;BY POINTER 1
00D8 CEFF           ST      @-1(P2)
00DA 37             XPAH    P3
00DB CEFF           ST      @-1(P2)
00DD 33             XPAL    P3
00DE CEFF           ST      @-1(P2)
00E0 C501   OBTAIN: LD      @1(P1)
00E2 01             XAE
00E3 C404           LDI     EOT
00E5 60             XRE
00E6 9809           JZ      EXIT
00E8 C4E1           LDI     L(PUTC)-1   ;CALL PUTC IN
                                        ;L.C.D.S. MONITOR
00EA C47A           LDI     H(PUTC)     ;TO OUTPUT
00EC 37             XPAH    P3          ;CHARACTER
00ED 40             LDE
00EE 3F             XPPC    P3
00EF 90EF           JMP     OBTAIN
00F1 C601   EXIT:   LD      @1(P2)
00F3 33             XPAL    P3
00F4 C601           LD      @1(P2)
00F6 37             XPAH    P3
00F7 C601           LD      @1(P2)
00F9 07             CAS
00FA C601           LD      @1(P2)
00FC 01             XAE
00FD 3F             XPPC    P3
                ;
00FE 20     MSPACE: .BYTE   SP,EOT
00FF 04
0100 0A     MCRLF:  .BYTE   LF,CR,EOT
0101 0D
0102 04
0103 2A     MPOINT: .ASCII  '*'         ;OR OPTIONS
0104 0A             .BYTE   LF,CR,EOT
0105 0D
0106 04
0107 2D2D   MOVER:  .ASCII  '----------------'
```

```
                0109 2D2D
                010B 2D2D
                010D 2D2D
                010F 2D2D
                0111 2D2D
                0113 2D2D
                0115 2D2D
191 0117 2D2D           ASCII '-----------------'
                0119 2D2D
                011B 2D2D
                011D 2D2D
                011F 2D2D
                0121 2D2D
                0123 2D2D
                0125 2D2D
192 0127 2D2D           .ASCII '-----------------'
                0129 2D2D
                012B 2D2D
                012D 2D2D
                012F 2D2D
                0131 2D2D
                0133 2D2D
                0135 2D2D
193 0137 2D56           ASCII '-VALUE OVER RANGE---->'
                0139 414C
                013B 5545
                013D 204F
                013F 5645
                0141 5220
                0143 5241
                0145 4E47
                0147 452D
                0149 2D2D
                014B 2D3E
194 014D 07             BYTE  BEL,BEL,CR,LF,BEL,BEL,EOT
                014E 07
                014F 0D
                0150 0A
                0151 07
                0152 07
                0153 04
195 0154 0D     MIAXIS: BYTE  CR,LF,LF,BEL,BEL,LF,LF
                0155 0A
                0156 0A
                0157 07
                0158 07
                0159 0A
                015A 0A
196 015B 2020           .ASCII '    GRAPH PLOTTER FROM RAM
                015D 2020
                015F 4752
                0161 4150
                0163 4820
                0165 504C
                0167 4F54
                0169 5445
                016B 5220
                016D 4652
                016F 4F4D
                0171 2052
                0173 414D
                0175 20
197 0176 4120           .ASCII 'A JERMYN UTILITY PROGRAM'
                0178 4A45
                017A 524D
                017C 594E
                017E 2055
                0180 5449
                0182 4C49
                0184 5459
                0186 2050
                0188 524F
                018A 4752
                018C 414D
198 018E 0A             .BYTE  LF,CR,LF
                018F 0D
                0190 0A
199 0191 2020   MAXIS:  .ASCII '    0         10'
                0193 2020
                0195 3020
                0197 2020
                0199 2020
                019B 2020
                019D 2031
                019F 30
200 01A0 2020           .ASCII '         20'
                01A2 2020
                01A4 2020
                01A6 2020
                01A8 3230
201 01AA 2020           .ASCII '         30'
                01AC 2020
                01AE 2020
                01B0 2020
                01B2 3330
202 01B4 2020           .ASCII '         40'
                01B6 2020
                01B8 2020
                01BA 2020
                01BC 3430
203 01BE 2020           .ASCII '         50'
                01C0 2020
                01C2 2020
                01C4 2020
                01C6 3530
204 01C8 2020           .ASCII '         60'
                01CA 2020
                01CC 2020
                01CE 2020
                01D0 3630
205 01D2 0D             BYTE  CR,SP,SP,SP,SP,EOT
                01D3 20
                01D4 20
                01D5 20
                01D6 20
                01D7 04
206 01D8 2D2D   MCURSR: .ASCII '----+---------'
                01DA 2D2D
                01DC 2B2D
                01DE 2D2D
                01E0 2D2D
                01E2 2D2D
                01E4 2D2D
207 01E6 2B2D           .ASCII '+---------'
                01E8 2D2D
                01EA 2D2D
                01EC 2D2D
                01EE 2D2D
208 01F0 2B2D           .ASCII '+---------'
                01F2 2D2D
                01F4 2D2D
                01F6 2D2D
                01F8 2D2D
209 01FA 2B2D           .ASCII '+---------'
                01FC 2D2D
                01FE 2D2D
                0200 2D2D
                0202 2D2D
210 0204 2B2D           .ASCII '+---------'
                0206 2D2D
                0208 2D2D
                020A 2D2D
                020C 2D2D
211 020E 2B2D           .ASCII '+---------'
                0210 2D2D
                0212 2D2D
                0214 2D2D
                0216 2D2D
212 0218 2B2D           ASCII '+----+'
                021A 2D2D
                021C 2D2B
213 021E 0D             .BYTE  CR,SP,SP,SP,SP,EOT
                021F 20
                0220 20
                0221 20
                0222 20
                0223 04
214 0224 2020   MSAXIS: .ASCII '    +'
                0226 2020
                0228 2B
215 0229 0D             .BYTE  CR,SP,SP,SP,SP,EOT
                022A 20
                022B 20
                022C 20
                022D 20
                022E 04
216                     ;
217                     ;SINE WAVE /\/\/\/
218                     ;TEST PATTERN FOR PLOTTER PROGRAM
219 022F 1E     MDATA:  .BYTE  30,38,44,48,50,48,44,38
                0230 26
                0231 2C
                0232 30
                0233 32
                0234 30
                0235 2C
                0236 26
220 0237 1E             BYTE  30,22,16,12,10,12,16,22
                0238 16
                0239 10
                023A 0C
                023B 0A
                023C 0C
                023D 10
                023E 16
221       023F  DATEND  =                 ;EQU TO PGM CNTR
222             ;
223       0000          .END


BEGIN  0001 *   BEL    0007     BS     0008 *
CLEAR  002F     COUNT  07FF     CR     000D
CYCLE  007B *   DATEND 023F     EOT    0004
EXIT   00F1     FETCH  009D     LF     000A
MAXIS  0191     MCRLF  0100     MCURSR 01D8
MDATA  022F     MIAXIS 0154     MIN65  FFBF
MINONE FFFF     MINTEN FFF6     MOVER  0107
MPOINT 0103     MSAXIS 0224     MSPACE 00FE
NOSAX  0041     OBTAIN 00E0     OVER   004B
P1     0001     P2     0002     P3     0003
POINT  0053     PRICYC 0074     PRINT  00D4
PRIRET 0059     PUTC   7AE2     RETURN 0060 *
RSTCYL 003F     SP     0020     SPACE  00BB
STACK  07FC     VECTOR 07FD     ZERO   0000


NO ERROR LINES
SOURCE CHECKSUM = CA1E
```

# NEW CB BOOK

A new booklet describing electronic components designed for use in citizen's band radio manufacture is now available from National.

The products described in the booklet include synthesizer systems, 5-pin audio amplifiers, microprocessor controlled tuning systems, linear IC's, light emitting diodes (LED's), clock modules, RF output discretes, and regulators.

Titled *National Semiconductor Personal Communications: CB Radio*, the booklet is available without charge from National Semiconductor Corp., 2900 Semiconductor Drive, Santa Clara, Calif. 95051.

## How To Build A Digital Thermometer

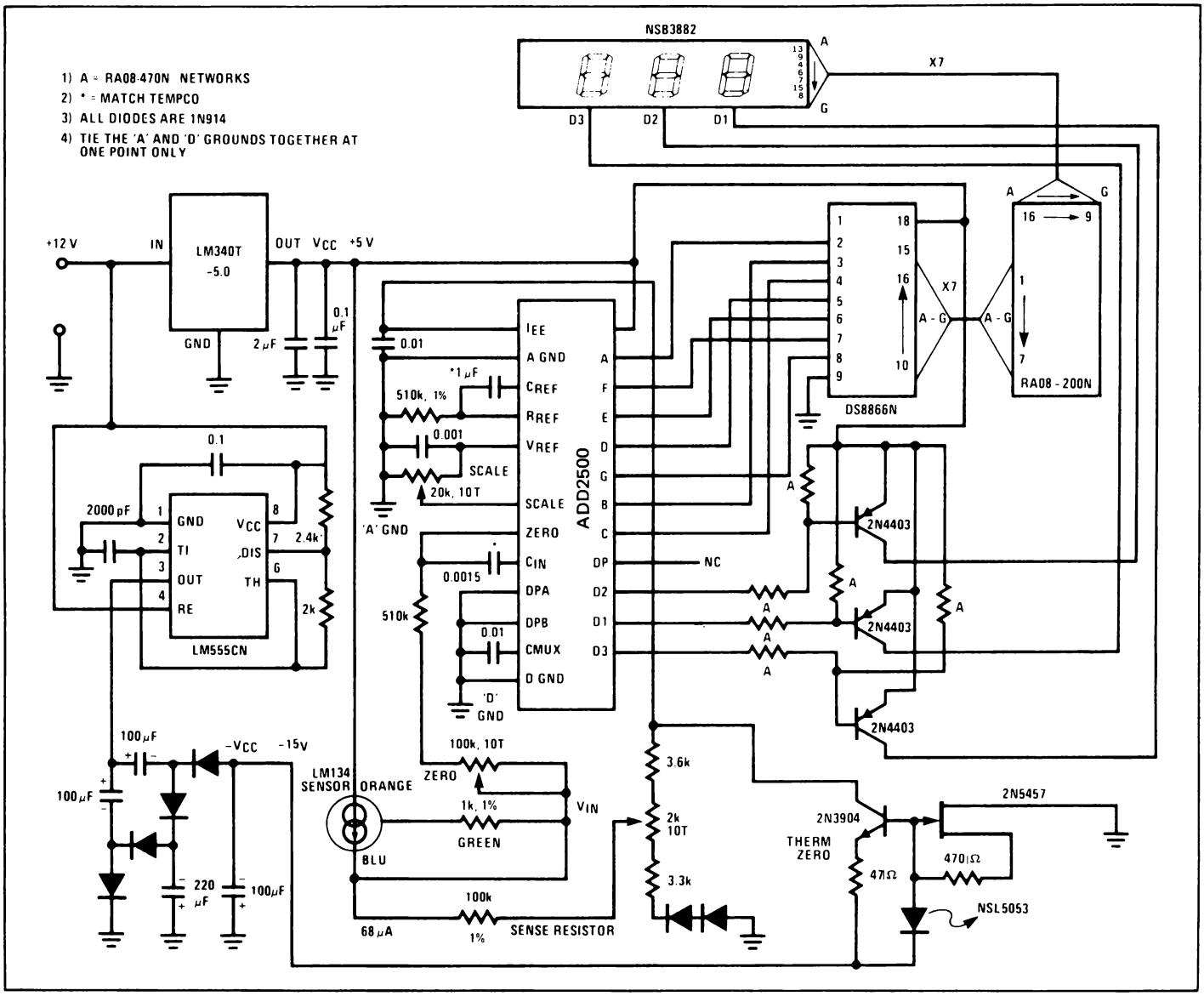
Analog electronic thermometers have been available for some time, but they are generally difficult to read and, besides, are relatively fragile. Digital thermometers, on the other hand, are both easy to read and rugged.

Besides a +5 V input, the ADD2500 draws 18 mA from a negative supply. This comes from the dc/dc converter (at −15 V) as a regulated current via the 2N5457 FET, the LED, and the 2N3904. The negative supply of the ADD2500 is internally Zener regulated; it, together with the two diodes and the resistor string between ground and $I_{EE}$, establish a low-drift offset voltage for the LM134's sense resistor.

The finished thermometer requires only a single, unregulated +12 V supply, and operates from −29°C to +60°C (−20°F to +140°F).

The digital thermometer described here uses a ADD2500 2½-digit DPM chip for A/D conversion and display decoding. The LM134 programmable current source operates here as the temperature sensor, and the LM555 timer as a dc/dc converter. The DS8866 and the pnp transistors drive the NSB3882 display.

The LM134 makes an excellent temperature sensor; it has a constant temperature coefficient of +0.30%/°C (0.167%/°F); and its noise immunity and current programmability make it ideal for remote sensing use. Output-current flow through a sense resistor scales the LM134's output voltage—in this case, to 10 mV/°F, which is one count of the DPM or 1°F displayed.



# Tough mathematical tasks are child's play for Number Cruncher

## New special-purpose microprocessor combines best features of general-purpose and calculator chips

by Alan J. Weissberger and Ted Toal, *National Semiconductor Corp., Santa Clara, Calif.**

Reprinted from *Electronics*, February 17, 1977;


☐ There is one hurdle that the general-purpose microprocessor clears awkwardly: complex mathematical computations. For such applications, designers have had to spend considerable time learning to use efficiently the chosen device's instruction set and unique input/output transfer characteristics. Then they have had to sweat out the development of complex software to perform the desired mathematical operations or algorithms.

A few, hardy designers have put up with these chores in order to gain the benefits of large-scale-integrated technology, but even they would prefer a special-purpose microprocessor designed specifically for complex calculations. A new microprocessor, the 57109 or Number Crunching Unit, does this.

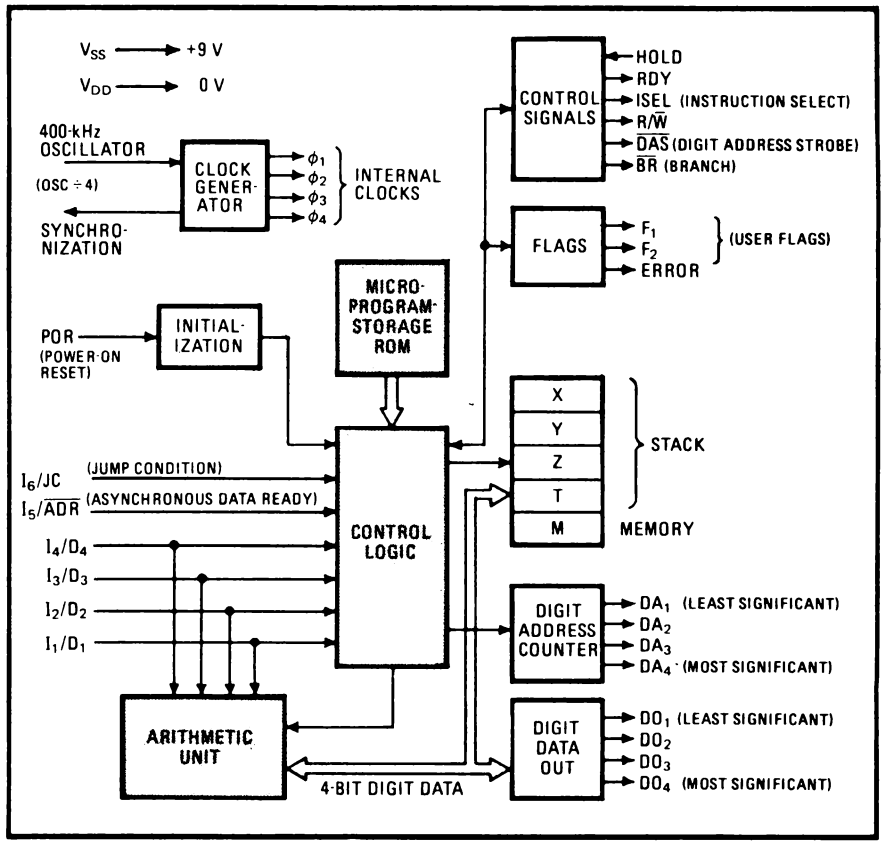
The NCU, presently being built with p-channel metal-oxide-semiconductor technology, can serve in machine process controllers, navigation systems, and measurement and test equipment. It can also extend a mini- or microcomputer's processing power when connected as a peripheral device on the host processor's bus.

In such processing applications, software development time can drop significantly with the NCU. Its instruction set is like those of scientific calculators, which means that the Number Cruncher already has most of the required calculation software. Trigonometric, logarithmic, and exponential functions, for example, are performed directly.

Data formats at the input or the output may be in floating-point or scientific notation. Digit lengths may range up to an eight-digit mantissa, with one or two digits for the exponent.

The 57109 combines the best features of calculator

*Al Weissberger is now with Signetics Corp., Sunnyvale, Calif.




**1. Digit handler.** Major functional blocks of the Number Cruncher are the control-logic and arithmetic units and the program-storage ROM, which holds about 1,500 8-bit microinstruction words. The device handles 4-bit binary-coded-decimal digits directly. They enter the control-logic block through the $I_{1,4}$ lines, and the results go out through the digit-data-out block. The digit-address-counter block sequences each digit during input/output operations. Programmed instructions, 6 bits long, enter through the $I_{1,6}$ lines and are converted to sequences of microinstructions.

chips and general-purpose microprocessors (Table 1). For example, its I/O functions are more flexible than those of the calculator, which is limited to inputs from a keyboard and outputs to a display. But it is more directly useful for calculations than microprocessors. The NCU can accept a sequence of binary-coded-decimal digits with a single input instruction, an asynchronous digit input, or single-bit inputs for control purposes. In contrast, microprocessors work only on data bytes.

Unlike calculators, the Number Cruncher is controlled by a program stored in an external read-only memory and can perform conditional and unconditional program branches. As in processors, a HOLD input allows handling asynchronous instructions and single stepping, while test and branch instructions facilitate decision-making within programs.

The NCU's major functional blocks (Fig. 1) are the control-logic and arithmetic units and the program-storage ROM, which holds about 1,500 8-bit microinstruction words. Programmed instructions, 6 bits long, enter through the $I_{1\text{-}6}$ lines and are converted to sequences of microinstructions. Binary-coded-decimal 4-bit data words enter the control-logic block through the $I_{1\text{-}4}$ lines.

Output data passes through the digit-data-out block, while the digit-address-counter block sequences each digit during I/O operations. Logic levels are compatible with low-power logic families, and the device has on-chip generation of input/output strobes and timing signals.

Examples of the 6-bit operation codes are given in Table 2. (If 8-bit instruction memories are used, external hardware can use the additional 2 bits for device addresses.) Instruction executions vary in time from 1 to 500 milliseconds, although most require 5 to 10 ms. Although these speeds may seem rather slow, they compare favorably with similar functions implemented as subroutines in low-cost microprocessors.

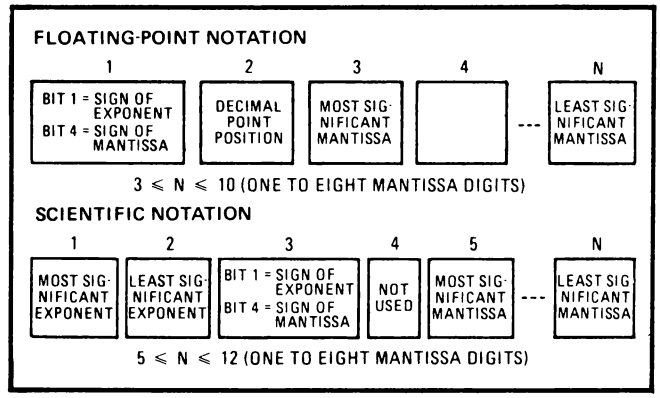
Conditional-test-and-skip/branch instructions permit decision making within the user's program. The conditional-test instructions operate on the results of computations or from an external jump-condition sense input on line $I_6$. The two flag outputs ($F_1$ and $F_2$) may be used to activate external devices. A four-register stack (X, Y, Z, T) holds operands and temporary results, and a

TABLE 1. COMPARISON OF LARGE SCALE INTEGRATED PROCESSING CHIPS

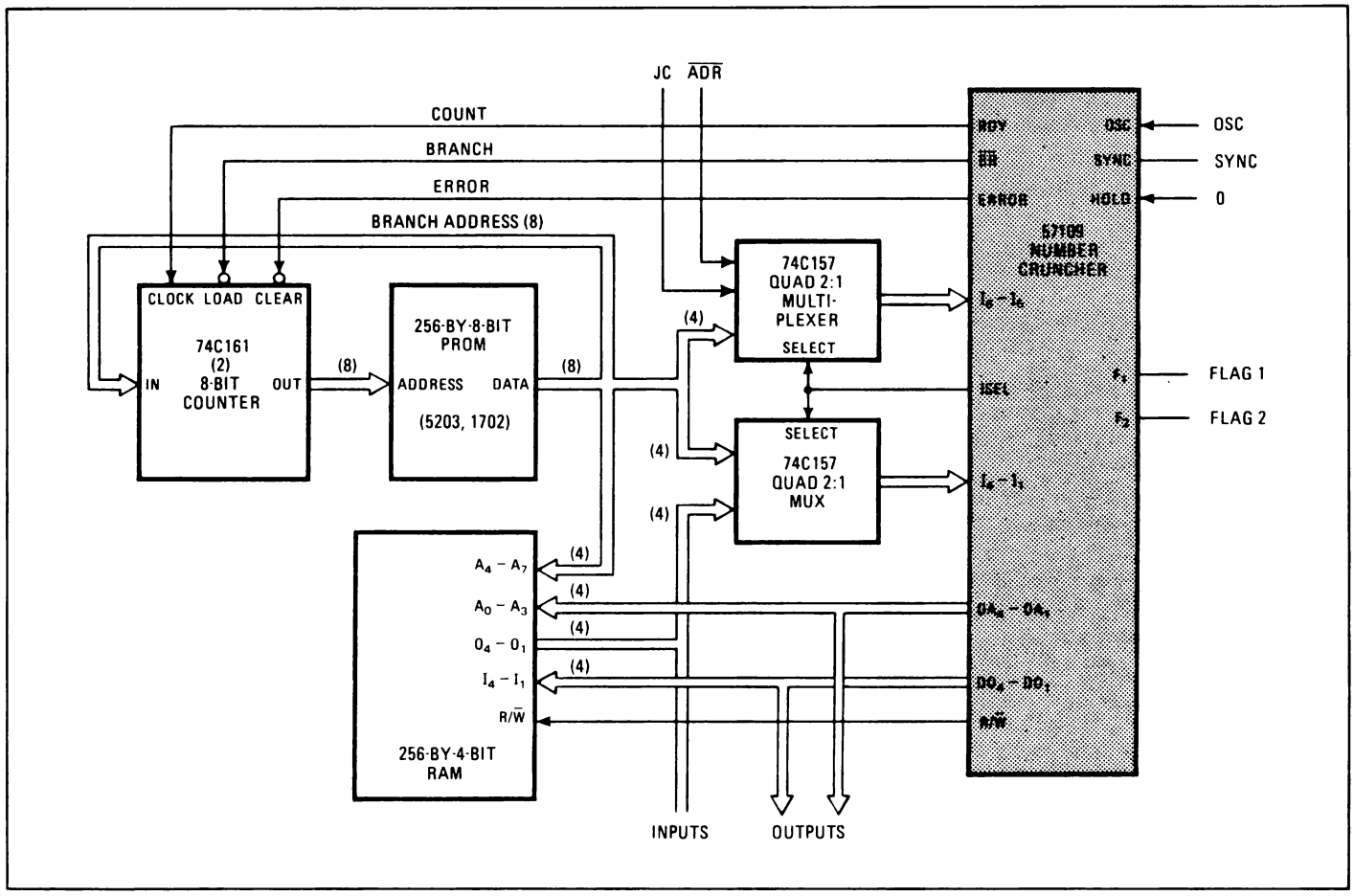
| Function | Calculator | Number Cruncher | Microprocessor |
|---|---|---|---|
| Input/output | keyboard and display | multidigit, asynchronous digit, single bit | data bytes, single bit |
| Data format | floating-point or scientific notation | floating-point or scientific notation | binary |
| Data length | fixed | variable (1 to 8 digits for mantissa) | fixed |
| Program | key sequence | external ROM/program counter, microprocessor, or first-in, first-out buffer memory | external ROM, internal PC |
| Speed (math or I/O operations) | 14 – 400 ms | 0.5 – 400 ms | 0.5 – 500 ms |
| Minimum number of chips for CPU and RAM | 1 – 3 | 1 (external PC) | 2 – 6 |

TABLE 2. NCU INSTRUCTION CLASSES

| | |
|---|---|
| **Digit entry:** | |
| 0 – 9 | Each digit is entered into the X register mantissa or exponent if in enter-exponent mode. |
| . | Fixes decimal point of mantissa of number being entered. |
| EE | Set enter-exponent mode. |
| CS | Change sign of mantissa or exponent. |
| PI | $\pi \rightarrow$ X register. |
| EN | Number entry terminated and stack is pushed. $X \rightarrow Y \rightarrow Z \rightarrow T$ |
| **Move:** | |
| Roll | Roll stack. $X \rightarrow T \rightarrow Z \rightarrow Y \rightarrow X$ |
| POP | Pop stack. $X \leftarrow Y \leftarrow Z \leftarrow T \leftarrow 0$ |
| XEY | Exchange X and Y. $X \leftrightarrow Y$ |
| XEM | Exchange X with memory. $X \leftrightarrow M$ |
| MS | Memory store. $X \rightarrow M$ |
| MR | Memory recall. $X \leftarrow M$ |
| **Math:** | |
| $X \leftarrow Y + X, X \leftarrow Y - X$<br>$X \leftarrow Y * X, X \leftarrow Y \div X$<br>$X \leftarrow Y^x$ | Result in X, stack popped. $Y \leftarrow Z \leftarrow T \leftarrow 0$ |
| $M \leftarrow M + X, M \leftarrow M - X$<br>$M \leftarrow M * X, M \leftarrow M \div X$ | Result in memory. |
| $1/X, \sqrt{X}, X^2 10^x, e^x, \ln x, \log X$ | Result in X, previous X lost, stack unchanged. |
| $SIN(X), COS(X), TAN(X)$<br>$SIN^{-1}(X), COS^{-1}(X), TAN^{-1}(X)$ | Result in X, previous X lost, stack unchanged. |
| RTD, DTR | Convert X from radians to degrees or vise versa. Previous X lost, stack unchanged. |
| **Branch:** | |
| JMP | Unconditional jump. On call branch instructions, second word of instruction is the branch address, which is loaded into an external program counter by a load pulse from the NCU. |
| TJC | Test external jump condition, branch if true. |
| **Input/output:** | |
| IN | Multidigit synchronous input from RAM or peripheral into X register. |
| OUT | Multidigit synchronous output to RAM or peripheral from X register. |
| AIN | Single digit asynchronous input. Wait for asynchronous data ready ($\overline{ADR}$) to go low, then read data and pulse acknowledge flag $F_2$ |
| **Mode control:** | |
| SMDC | Set mantissa digit count from one to eight digits. |




**2. Two formats.** The NCU can operate on data in floating-point or scientific notation formats with one to eight mantissa digits, depending on the setting of the digit count. It takes only one instruction to input or output a string of digits.




**3. Stand-alone.** The Number Cruncher can be used by itself in many control applications. Here a programmable ROM stores instructions, controlled by an external program counter, and a 256-by-4-bit RAM extends the internal memory. Multiplexers enter data or instructions.

memory register can store constants or temporary results or can serve as a loop counter for data transfer or program control. Additional data storage may be provided by external 256-word-by-4-bit random-access memories.

The two data formats are shown in Fig. 2. No reformatting is necessary when data is extracted from the 57109 or reentered from an external RAM. An asynchronous digit-input (AIN) instruction will accept a single digit when a data-ready signal indicates valid data.

Error detection is facilitated by an error flag, set by an arithmetic or output error. The TERR instruction tests the flag or can clear the external program counter, resulting in a hardware jump to memory location 0, the error recovery location. In either case, an ECLR instruction must be executed to clear the error flag.

## The basic setup

The basic Number Cruncher system in Fig. 3 includes a programmable ROM for instructions, an external program counter, and a RAM for memory expansion. To fetch an instruction from the PROM, the NCU raises its ready line after it has executed the previous instruction. This signal is used as a clock to advance the program counter.

The PROM then accepts the new 8-bit address supplied by the PC, executes a read cycle, and supplies the instruction to the NCU. To facilitate entry of asynchronous instructions, the 57109 does not lower RDY and begin execution until its HOLD input is low.

When the incoming instruction is a test and skip, the chip activates RDY to advance the PC and obtain the next word on the PROM output lines. This word is actually a branch address.

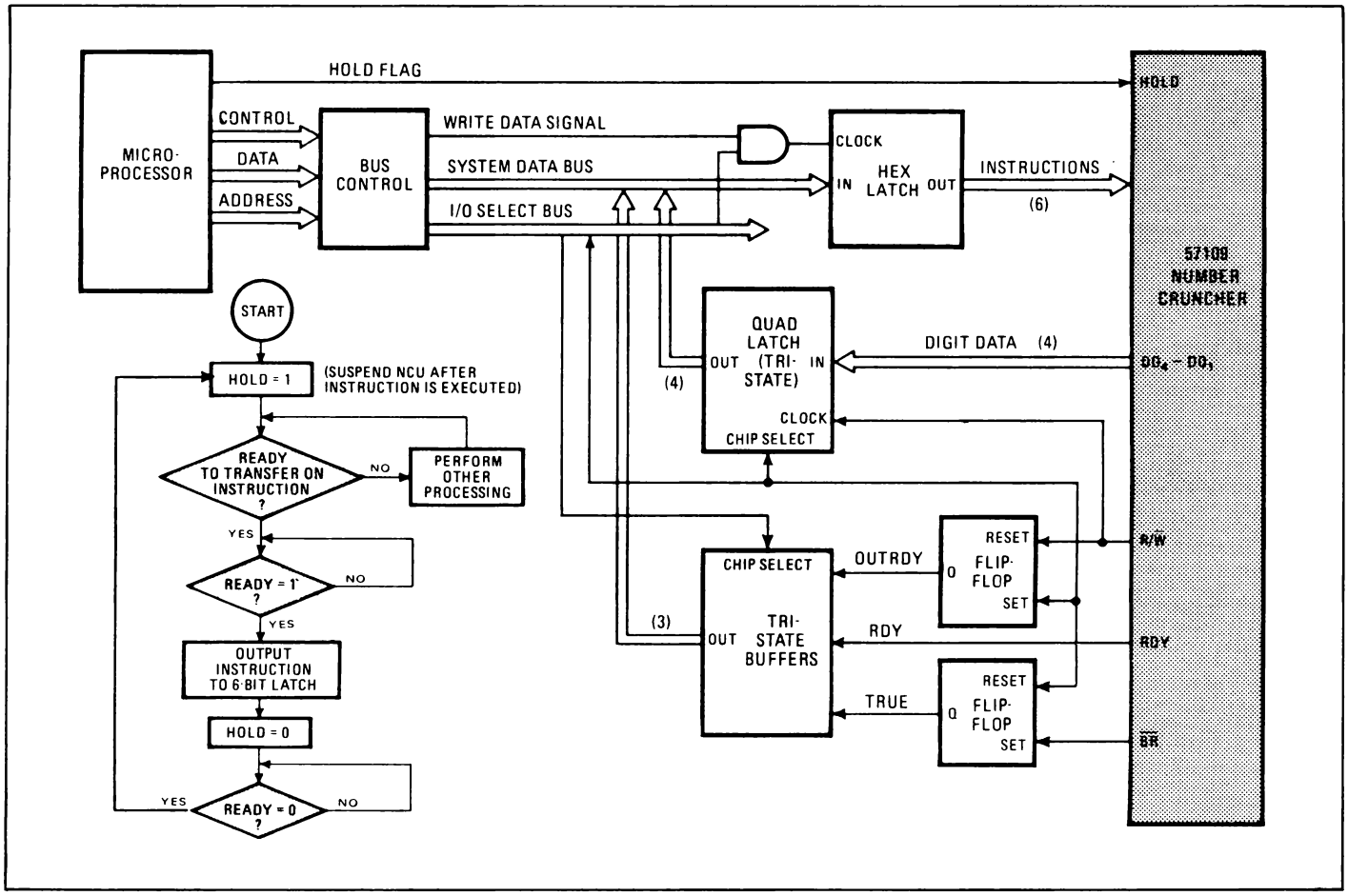
If the branch condition is true, the NCU's branch signal gates this address back to the program counter by parallel-loading it on the leading edge of the next RDY signal. When the PC is loaded, the PROM outputs will be the contents of the instruction at the branch address.

If the branch condition is not true, RDY is raised to step the PC so that it will point to the next sequential instruction at the time of the next instruction fetch.

The instruction-select signal (ISEL) selects which type of input will be used: instructions or data. The 2:1 multiplexers supply the Number Cruncher with data signals or instructions on the six input lines. This multiplexing saves pins so that the NCU can fit into a 28-pin package.

For a data-input instruction (IN), the Number Cruncher again raises RDY to advance the PC to the next instruction word, which contains a 4-bit high-order RAM address. The NCU supplies a 4-bit low-order digit address to the RAM from the digit-address DA lines and reads the RAM digit data on its input lines, having set ISEL low to select data instead of instructions. On a single IN instruction, 3 to 12 digits are input.

The OUT instruction procedure is similar to that for IN,




**4. Partners.** The NCU can extend processing power of a general-purpose microprocessor by taking instructions and data from the processor's bus and executing the instructions at its own pace. Flow chart shows software for microprocessor control of interface.

except that digit-output data is supplied on the data-output DO lines and the read/write line is pulsed to write each digit into the RAM. After putting out 3 to 12 digits, the 57109 enters a fetch cycle to obtain the next instruction.

## Extending a processor

Software overhead can be staggering for microprocessor applications requiring mathematical functions or BCD operations. Sophisticated subroutines must be written for multiply, divide, square root, log, exponential, and trigonometric functions. The data must be scaled to fix the decimal-point position and to assure there will be no register overflow as a result of an operation. Further conversion is necessary if the result is to be given in floating-point or scientific notation.

However, the Number Cruncher provides a microprocessor with a convenient peripheral unit for performing these specialized calculations. The microprocessor controls the NCU simply by supplying it with valid instructions, directly or through a buffer memory. Overlapping execution in the two devices gives much greater throughput than when the microprocessor performs the calculations itself.

A straightforward processor-NCU interface can be built with a pair of latches (Fig. 4); one for instructions and input data, the other for output data. The processor suspends the NCU's operation through the latter's HOLD signal. When the microprocessor is ready, it loads the instruction latch with a 6-bit instruction code and sets HOLD low.

The Number Cruncher executes the 6-bit code. The microprocessor senses succeeding RDY signals from the 57109 (as an interrupt or jump-condition input) and then loads the latch with the next instruction. It supplies input data to the Number Cruncher on a digit-by-digit basis in the same manner as it does 6-bit instructions.

When the NCU has data to send back, it uses a 4-bit latch. The microprocessor reads and stores this data as it is loaded into the latch.

## Using a FIFO

In another method for extending a microprocessor system with the Number Cruncher, a first-in, first-out buffer memory is a dynamic instruction store (Fig. 5). The microprocessor loads the FIFO, and the NCU draws instructions from it. Another FIFO is used for output data from the 57109. Since these memories are totally asynchronous, with separate input and output controls, the processor and the Number Cruncher can run at full speed in parallel for maximum system throughput.

This setup is useful in applications where the sequence of instructions executed by the NCU may change. Since the FIFO is a dynamic memory, it permits easy alteration of the sequence. Because instructions are stored only until the 57109 removes them, it is possible to load a very large sequence in a very small space.

When the microprocessor has a job for the NCU, it

loads a linear sequence of instructions (no branches) into the instruction FIFO, which was initially cleared. Once loading has been completed, the processor is free to process data or control devices. The FIFO can be used as the storage medium for many different instruction sequences with only minimum microprocessor software required for loading.

The FIFO stacks the instruction words in the same order as they are entered and makes them available at the output in the same sequence. The processor treats instructions to the 57109 as output data, as if they were to be written into a RAM or loaded into a register. But it selects the FIFO as an I/O device by putting that memory's address on its address bus. Next it puts the NCU instruction data on the data bus followed by a write strobe, which the FIFO uses as an input data clock.

This sequence is repeated each time a word is loaded into the FIFO. Before transmitting each instruction word, the processor checks the FIFO's status-indicator flag. If the FIFO is full, the microprocessor waits until it is ready before resuming data transfer.

While the processor is loading data into the FIFO, the NCU is fetching instructions from the FIFO output ports at its own speed, executing them one by one. An output-indicator flag signals the 57109 when the ports are ready (FIFO not empty) or not ready (FIFO empty). When the processor has loaded the first instruction word, the FIFO is ready and may be interrogated. The Number Cruncher's ready line is used as a FIFO output clock to extract data and gate it onto its instruction input lines.

The last instruction executed empties the FIFO, which forces its output indicator to not ready. This flag is the hold input for the NCU and an interrupt input for the microprocessor, which senses that the Number Cruncher has completed its instructions.

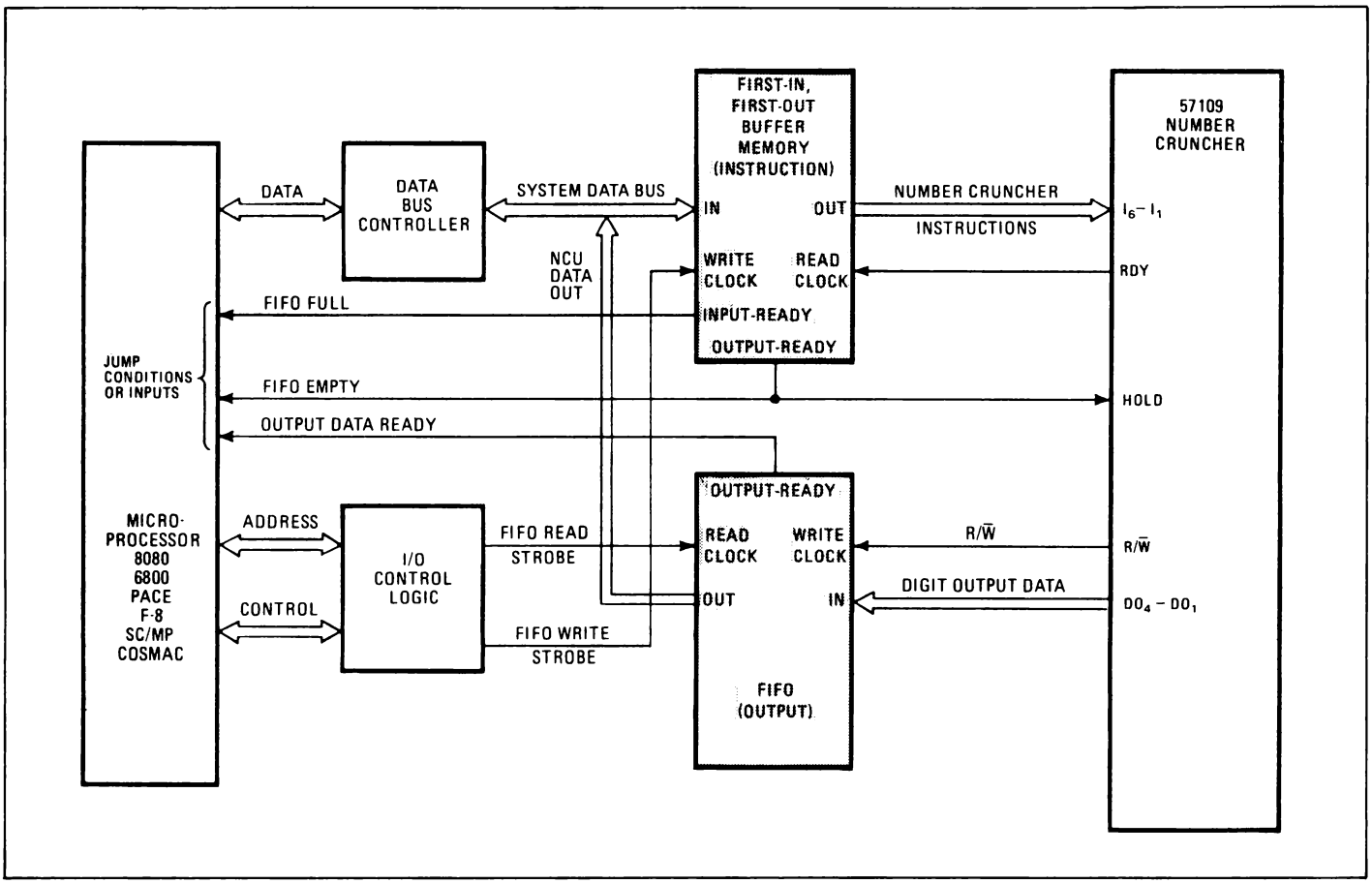
The 57109 continues to execute instructions until it has completed its specialized calculations. It sends its results to the output FIFO using an OUT instruction. If output data is present in the FIFO, the processor senses this via an interrupt or jump condition. It obtains the results if needed or sends them on to an output device.

### Control by a ROM

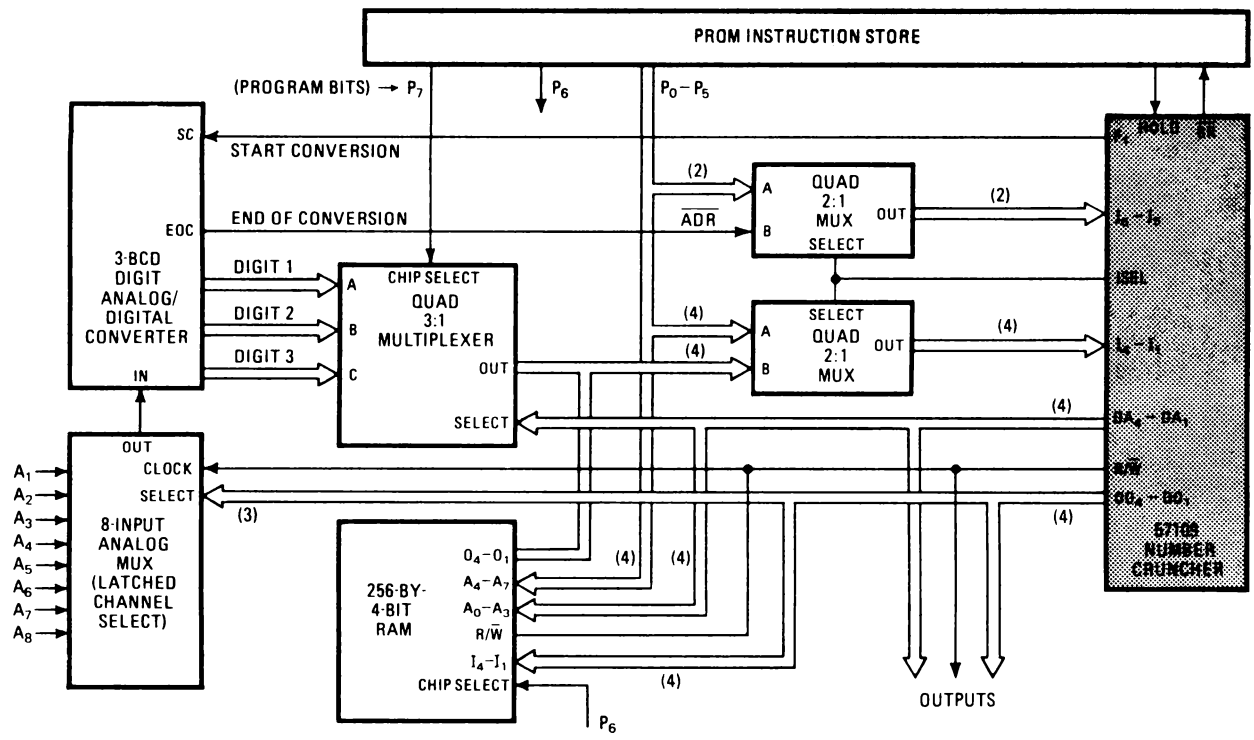
To transfer instructions where only a few sequences are necessary, a ROM can be programmed to contain the sequence of instructions for the NCU. This setup is similar to the stand-alone system in Fig. 3. It permits conditional test instructions not possible with the FIFO interface.

An AIN instruction suspends the Number Cruncher until the microprocessor requests a calculation. At that time the processor sets the asynchronous data ready ($\overline{ADR}$) to 0 and supplies a 4-bit starting address code. The NCU decodes the starting address, branches to that address in the ROM, and executes the calculation routine there. As in the FIFO setup, an interrupt notifies the microprocessor when the 57109 has completed its task.

In this setup, the microprocessor does not have to load




**5. FIFO interface.** A microprocessor can control the Number Cruncher through first-in first-out memories. The microprocessor enters data and instructions into the instruction FIFO, and the Number Cruncher extracts them asynchronously.




| INSTRUCTION FORMAT | | | |
|---|---|---|---|
| Type of instruction | $P_7$ | $P_6$ | $P_5$ $P_4$ $P_3$ $P_2$ $P_1$ $P_0$ |
| Select analog channel | 1 | 1 | OUT instruction |
| Analog-to-digital input | 0 | 1 | AIN instruction |
| RAM I/O | 1 | 0 | IN/OUT instructions for first word. Second word $P_{0-3}$ are high-order RAM address. |
| Others | 1 | 1 | other instructions |

| CODING | | | | |
|---|---|---|---|---|
| | $P_7$ | $P_6$ | $P_{5-0}$ | Comment |
| | 1 | 1 | 4 | number of channels to be input |
| | 1 | 1 | MS | store in M |
| | 1 | 1 | SMDC1 | mantissa digit count = 1 |
| LOOP | 1 | 1 | MR | retrieve channel number |
| | 1 | 1 | OUT0 | select analog channel |
| | 1 | 1 | PF1 | start A/D converter |
| | 1 | 1 | EN | push stack |
| | 0 | 1 | AIN (3) | read A/D converter digits 1 to 3 when EOC = 0 |
| | 1 | 1 | DBZ DONE | update channel number and check |
| | 1 | 1 | JMP LOOP | if 0 |

**6. Data acquisition.** The Number Cruncher can handle complex data in an analog system by controlling the analog multiplexer, which sends analog data to an analog-to-digital converter. The program listing shows how single instructions from the NCU handle complex operations.

a buffer memory to provide the NCU with instructions, but merely switches it on by supplying a single input (ADR). The result is very high system throughput and parallel processing.

In the analog data-acquisition system in Fig. 6, the PROM program controls the Number Cruncher. It makes the NCU measure analog variables, perform some digital transformation on them, compare the results to certain specified limits, and send out control information on the DO lines.

## Acquiring analog data

An eight-input analog multiplexer selects the desired analog input channel based on a 3-bit address supplied by the NCU on $DO_{3-1}$. This address is latched by the multiplexer and will not change during the conversion time of the three-BCD digit analog-to-digital converter via Flag 1. After starting the conversion via Flag 1, the NCU's AIN instruction waits for the end-of-convert signal before reading one of the three digits through the 3:1 digital multiplexer. The second and third AIN instructions read the second and third BCD digits with the results stored internally.

The PROM program updates the analog address and tests to see if all analog channels have been interrogated. If so, the program will output the digitized data to the RAM or will process the data as required. If not all analog channels have been interrogated, the PROM program scans the next one.

This system uses internal NCU storage for simultaneous calculations on four three-digit numbers. Additional storage is provided by the 256-by-4 RAM so that the 57109 can operate on an array of data. Addressing the data in the RAM is facilitated by the IN and OUT instructions. The first instruction word is either IN or OUT, and the second supplies a 5-bit address to select one of 32 numbers in the RAM. This address is stable on the instruction input lines ($I_{6-1}$) and is valid throughout the data transfer cycle.

The Number Cruncher generates a 4-bit address ($DA_{4-1}$) to select a digit each time it is ready to input or output 4-bit data. For an OUT instruction, digit data is output on $DO_{4-1}$ and clocked into the RAM by the R/$\overline{W}$ strobe. For an IN instruction, the high level on R/$\overline{W}$ causes the RAM to go into a read cycle and supply digit data to the NCU through the quad 2:1 multiplexers. □

# the Bit • Bucket

Dear Georgia:

I recently noticed that you are providing a list of microprocessor consultants for your readers in each issue of COMPUTE Newsletter. Please consider our corporation as a possible addition to that list.

Texas Microsystems, Inc. is a Houston, Texas based consulting firm started several years ago. In November of 1976, I left National to open the West Coast Office of TMI. TMI has the capability to provide complete microcomputer system design and development, including hardware and software design. We can handle the client's entire needs from evaluation and specification to system prototype and debug. Our experience ranges from IMP-16, PACE and SC/MP systems, to systems involving the 8080A, 8085 and 8048, as well as other microprocessors.

Before November 1976, I spent three years as a microprocessor systems design engineer with National, working on such projects, as the PACE/16-P interface card, PACE application cards, PACE LCDS and custom SCMP microcomputer systems. I am currently a member of COMPUTE.

Thanks for considering Texas Microsystems, Inc.

Sincerely,

TEXAS MICROSYSTEMS, INC.

GARY A. MILLER
Regional Director
West Coast Office
1530 The Alameda
Suite 200
San Jose, CA 95126
(408) 292-4004

Sir,

My SC/MP based microcomputer is up and running. I built it for under $100 with 1K of RAM (2102's) yet!

On page 5-3 of the technical description I read of a user group and imagine you have information to share about using this CPU chip.

Please add me to the club. My particular interest is circuits showing how to expand my unit to TV, keyboard and cassette.

Any help at all is OK and let me know how I can aid you too.

Sincerely,

Bob Weir
318 N. 7th
Canon City, CO 81212

Dear COMPUTE:

The next time you list microprocessor consultants please add us to your list. We are presently working with several of the popular 8-bit processors as well as with IMP.

Thank you,

Ron Tipton, President
TDL Electronics
Route 7
Fayetteville, Arkansas 72701
(501) 643-2191

*Done.*

Dear Ms. Marszalek:

Since I wrote you last week about the problems with the SC/MP Cassette system, I discovered a potential problem with the software in the SC/MP keyboard kit SKMPKB. It contains a re-executable subroutine KYBD at location X'0185 which can be used by other software for display and keyboard input. It should be pointed out to users of this subroutine that the Carry/Link (CY/L) flag must be reset (cleared, CCL instruction) prior to calling KYBD. If CY/L is set, the number returned in the E register will be incorrect for keyboard keys 8 thru F.

Is it possible to obtain a corrected copy of the SC/MP Keyboard Kit Schematic Diagram (Drawing NS10634) which is in the Keyboard Kit User's Manual? The one in my manual contains a large number of errors.

The SC/MP Cassette continues to work well — zero errors after many 1k byte reads and writes.

Sincerely,

Ronald G. Parsons
9001 Laurel Grove Drive
Austin, Texas 78758

*Drawing NS10634 is replaced by NS10586 (shown on page 13 this issue) in revision B of SC/MP Keyboard Kit User's Manual.*

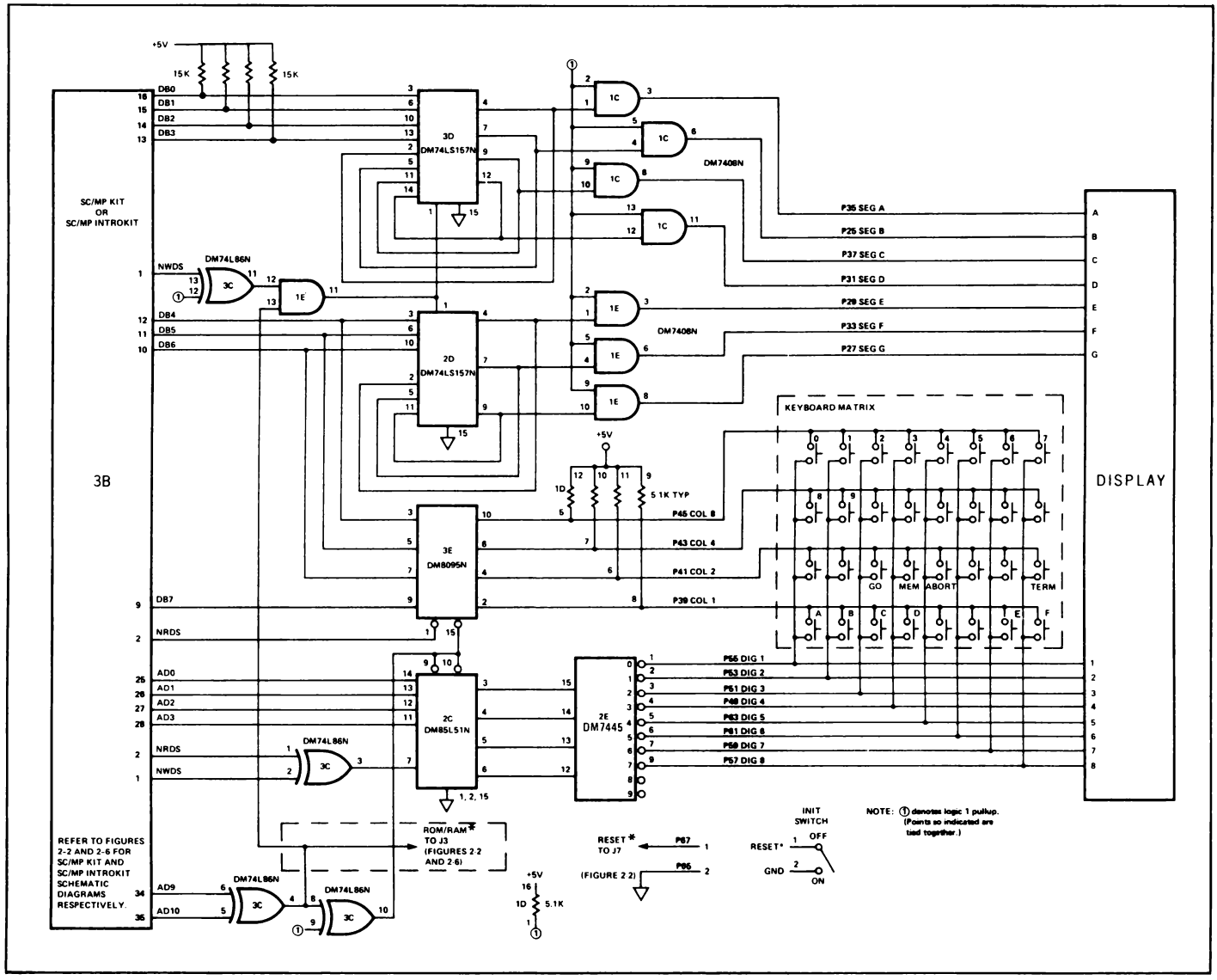
Dear Georgia:

I have built a Homebrew SC/MP, with 1K of 1702A, 1K of 2102LI, ASCII Keyboard, and selectric printing unit. I've implemented the hold and continue control lines and have a full front-panel LED display of addresses, data (read or write), and all flags and status indicators. I need some programming hints on subroutine linkage, etc.

Thanks,

Olin R. Boyer
P.O. Box 3000
Tulsa, OK 74102

*See the SC/MP Programming and Assembler Manual Chap. 6 for some programming details. Also the SC/MP Applications Handbook has many programming examples.*




# SC/MP KEYBOARD KIT REPAIR POLICY

Kits may be returned to the Microcomputer Service Center for repair on a consignment basis only! *NO DEBIT MEMO'S* . The customer should return the kit, not the distributor. The following information *MUST* be supplied or kit will be returned.

1) Name of customer contact
2) Telephone number of contact
3) Data purchased
4) Purchased from
5) Symptom of problem

Upon receipt of the unit in the service center the technician will determine if the kit will be repaired under warranty or if the customer is responsible for its repair.

If the customer is found to be responsible, a charge of $35.00 per hour plus parts will be charged. A purchase order or check for the amount of the repairs must be sent to the service center prior to the return of the repaired kit.

Kits must be returned to:

NATIONAL SEMICONDUCTOR CORP.
2921 COPPER RD.
SANTA CLARA, CA. 95051
ATTN: MICROCOMPUTER SERVICE CENTER

The following spare parts are also available from the service center. A check payable to National Semiconductor must accompany all orders.

P/N482305235 – 001 KEYBOARD KIT ROM @ $25.00
P/N980305232 – 001 KEYBOARD & CABLE @ $35.00

# NCSS Expands Its Service

National CSS has extended its telephone access and service through the world-wide TELEX network. This enables customers to access any of our host machines by dialing the NCSS TELEX rotor, 965806. International users may also access TWX by dialing 710-474-3540.

Line charges to the customer will be from the originating country to Stamford, Conn. All TELEX users should type (LTRS) NCSS after the connect light illuminates. Any problem during login should be reported to NCSS at (203) 327-9100 extension 381.

NATIONAL CSS, INC.
542 Westport Avenue
Norwalk, Connecticut 06851
(203) 853-7200

## UNDERGROUND BUYING GUIDE TELLS ELECTRONIC HOBBYISTS WHERE TO GET IT

A new directory has just been published that helps amateurs, CBers, experimenters and computer hobbyists locate equipment, parts, supplies and services.

Over 600 sources of standard and hard-to-find gear are listed in the handy guide. Many of the 600 sources are mail order firms and discounters. All are firms that do business with electronic hobbyists.

The Underground Buying Guide is available direct mail from PMS Publishing, 12625 Lido Way, Saratoga, CA. 95070. The price is $5.95 plus 55¢ postage and handling. Californians add 39¢ sales tax.

For further information contact:

Dennis A. King
PMS Publishing
12625 Lido Way
Saratoga, CA. 95070
(408) 996-0471


# MICROPROCESSOR USER'S GROUP

The following programs are available from the COMPUTE User Group Library. Copies can be ordered from COMPUTE/208, National Semiconductor, 2900 Semiconductor Drive, Santa Clara, Ca. 95051.

These programs are versions of assemblers, interpreters or compilers available for IMP, PACE, and SC/MP. Included in this list is also a listing of modifications for the NOVA assembler that will allow it to produce PACE object code and a PACE assembler written in FORTRAN with modified assembly mnemonics.

Note: As part of the User Library, these programs are not supported as National products.

| Program Number | Program Name | Description |
|---|---|---|
| SL0034A NOVA | | Instructions and listing for conversion of DATA GENERAL's Nova Assembler to a PACE cross assembler. Limited copies available. |
| SL0040A IMP-16 | 8080X | 8080 Cross Assembler for IMP-16. Object Module and source listing only available *$15.00.* Uses assembly directives similar to other National assemblers. Requires the following definitions: B=0, C=1, D=2, E=3, H=4, L=5, A=7, memory=6, PSW=6, Stack Pointer=7. |
| SL0042A IMP-16 | SM/PL | SM/PL is a high level programming language compiler for the IMP-16. See Compute Vol. 2, #8 for language features. It requires 16K of R/W memory and can be used with the IMP-16 disc system. Cost is $100.00 for the manual, object module paper tape and source listing. |
| SL0043A SC/MP | NIBL | NIBL is a version of Tiny Basic for SC/MP. It requires 4K of memory for the interpreter and an additional 2-4K for the NIBL source program. Cost is $15.00 for the paper tape load module and source listing. Both p- and n-channel versions are available. |
| SL0045A PDP-15 | PACE X-Assembler | FORTRAN cross-assembler with modified mnemonics. Reference: BYTE May 1976, "Simplifying Your Home-made Assembler" by Greg Jewell. |
| SL046A PDP-8 | SC/MP-Basic X-Assembler | Cross-assembler written in BASIC for PDP-8E with disc operating system. Listing only available. Contributed by R. Gitzel, University of Alberta, Edmonton, Canada. |

# USERS LIBRARY ORDER FORM

| PROGRAM NUMBER | NAME | NUMBER OF PROGRAM LISTINGS | SOURCE PAPER TAPES QUANTITY | SOURCE PAPER TAPES UNIT COST | TOTAL COST |
|---|---|---|---|---|---|
| **IMP PROGRAMS** | | | | | |
| SL001A | BINBCD | _____ | NA | | |
| SL002A | BCD | _____ | NA | | |
| SL003A | MD | _____ | NA | | |
| SL004A | PTBIN | _____ | NA | | |
| SL005A | BINASC | _____ | NA | | |
| SL006A | BINGRAY | _____ | NA | | |
| SL007B | BCDBIN | _____ | NA | | |
| SL008A | PNMULT | _____ | NA | | |
| SL010A | MEMORY DUMP | _____ | _____ | $5.00 | $ _____ |
| SL011A | GALPAT | _____ | _____ | $5.00 | $ _____ |
| SL012B | RAMDUMP | _____ | _____ | $5.00 | $ _____ |
| SL013A | TAPE TITLER | _____ | _____ | $5.00 | $ _____ |
| SL014A | GRAY CODE | _____ | _____ | $5.00 | $ _____ |
| SL016A | PRTPLT | _____ | _____ | $5.00 | $ _____ |
| SL017A | TSTPLT | _____ | _____ | $5.00 | $ _____ |
| SL019A | MESGH | _____ | _____ | $5.00 | $ _____ |
| SL020A | CHARST | _____ | _____ | $5.00 | $ _____ |
| SL021A | CONTAP | _____ | _____ | $5.00 | $ _____ |
| SL023A | DISC RLM-PROMSFT-B | _____ | NA | | |
| SL024A | DISC RLM-PROMSFT-C | _____ | NA | | |
| SL026A | TABTAP | _____ | _____ | $5.00 | $ _____ |
| SL028A | SORT | _____ | NA | | |
| SL030A | TITLER | _____ | NA | | |
| SL031A | DORG | _____ | _____ | $5.00 | $ _____ |
| SL038A | TAPE | _____ | _____ | $5.00 | $ _____ |
| SL040A | 8080-X | _____ | * | $15.00 | $ _____ |
| SL042A | SM/PL | _____ | *† | $100.00 | $ _____ |
| SL044A | DECIM8 | _____ | _____ | $5.00 | $ _____ |
| **PACE PROGRAMS** | | | | | |
| SL015A | PACRAM | _____ | _____ | $5.00 | $ _____ |
| SL018A | CALCULATOR | _____ | _____ | $5.00 | $ _____ |
| SL022A | NUMPRG | _____ | _____ | $5.00 | $ _____ |
| SL025A | PALM | _____ | _____ | $5.00 | $ _____ |
| SL026A | TABTAP | _____ | _____ | $5.00 | $ _____ |
| SL029A | BINBCD | _____ | NA | | |
| SL032A | DIVIDE | _____ | NA | | |
| SL033A | DELSEM | _____ | _____ | $5.00 | $ _____ |
| SL035A | PRNTLM | _____ | _____ | $5.00 | $ _____ |
| SL036A | BASCII | _____ | _____ | $5.00 | $ _____ |
| SL037A | JITTER | _____ | _____ | $5.00 | $ _____ |
| **SC/MP PROGRAMS** | | | | | |
| SL027B | SC/MP MATH | _____ | _____ | $5.00 | $ _____ |
| SL039A | TAPEI/O | _____ | _____ | $5.00 | $ _____ |
| SL041A | SCSORT | _____ | _____ | $5.00 | $ _____ |
| SL043A | NIBL | _____ | * | $15.00 | $ _____ |
| SL047A | PLOT | _____ | _____ | $5.00 | $ _____ |
| **NOVA PROGRAMS** | | | | | |
| SL034A | PACE-X | _____ | NA | | |
| **PDP-15 PROGRAMS** | | | | | |
| SL045A | PACE-X | _____ | _____ | $5.00 | $ _____ |
| **PDP-8 PROGRAMS** | | | | | |
| SL046A | SC/MP-X | _____ | NA | | |
| | | | | TOTAL | $ _____ |

*Price includes the manual, program listing, and paper tape load module.
†Available from the Melbourne Training Centre in Australia for DLR 100.00 for SM/PL and DLR 15.00 for NIBL.

Please make sure the programs you select are for the microprocessor you have.
Notes: 1. There is no charge for program listings, but the number of listings per order is limited to three (3). 2. NA indicates not available.

NAME _____
TITLE _____
COMPANY _____
ADDRESS _____
CITY _____

Fill out the form completely, make your check payable to COMPUTE, and mail to:

UNITED STATES
COMPUTE/208
National Semiconductor
2900 Semiconductor Drive
Santa Clara, CA 95051
(408) 247-7924

GERMANY
National Semiconductor GmBH
808 Fuerstenfeldbruck
Industriestrasse 10
Tel:08141/1371
Telex: 05-27649

AUSTRALIA
NS Electronics Pty Ltd.
Cnr. Stud Road & Mtn. Highway
Bayswater, Victoria 3153
Tel: 03-729-6333
Telex: 32096

## CALL FOR PAPERS

# IECI 78 CONFERENCE

## "INDUSTRIAL APPLICATIONS OF MICROPROCESSORS"

**SHERATON HOTEL • Philadelphia, Pennsylvania • MARCH 20-22, 1978**

Papers on the Following Subjects are Invited:

- Industrial Uses of Microprocessors
- Microprocessor System Hardware Architecture
- Microprocessor Software and Standardization
- Microprocessor in Thyristor Controls
- Computerized Data Acquisition Systems
- Programmable Controllers
- MSI and LSI in Process Control
- Automotive Diagnosis and Operation
- Vehicle Control
- Automatic Inspection
- "Intelligent" Test Instrumentation
- Transducers
- Textile Manufacturing
- Food Processing
- Petroleum Refining
- Geophysics
- Metal Fabrication
- Power Generation
- Education
- The State-of-the-Art in Microprocessor Standards.

PAPER REQUIREMENTS

Ten copies of the paper in summary form no longer than 600 words and an abstract of no more than 60 words, describing work not generally published or previously presented. The copies should be mailed by August 25, 1977 to:

H. W. MERGLER
Leonard Case Professor of Electrical Engineering
CASE WESTERN RESERVE UNIVERSITY
CLEVELAND, OHIO 44106
216/368-4574

The paper summary will be used for paper selection and session assignment and thus should clearly define the salient concepts and NOVEL features of the work described.

Notification of acceptance and format required for publication in the IECI '77 Proceedings will be sent to you by September 25, 1977. Final manuscripts of papers accepted for publication in the IECI proceedings must be received by November 25, 1977.

---

**UNITED STATES**

COMPUTE/208
National Semiconductor
2900 Semiconductor Dr.
Santa Clara, CA 95051
Tel: (408) 247-7924
TWX: 910-338-0537

**GERMANY**

National Semiconductor Corp. Gmbh
808 Fuerstenfeldbruck
Industriestrasse 10
Tel: 08141/1371
Telex: 05-27649

**AUSTRALIA**

NS Electronics Pty Ltd.
Cnr. Stud Road & Mtn. Highway
Bayswater, Victoria 3153
Tel: 03-729-6333
Telex: 32096