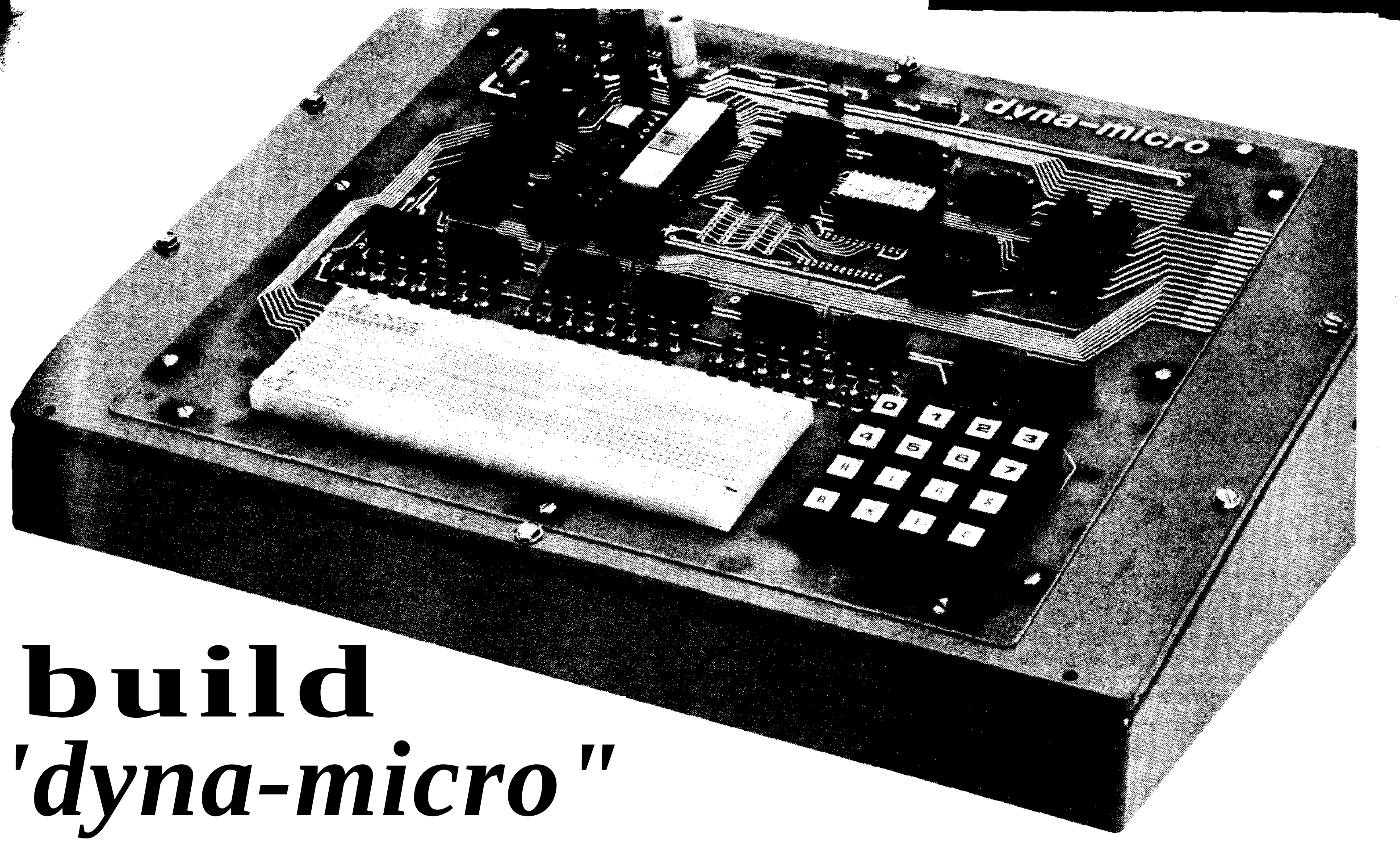


build 'dyna-micro" an BOBO ...icrocomputer

Complete with keyboard for data entry, LED readout of the address and data, breadboard socket for experimenting, SOD-bytes of PROM, SOD-bytes of RAM, expandable to 6SK and self-contained power supply

JOHN TITUS

LAST MONTH, PART-l OF THIS ARTICLE described the operation of the Dyna-MIcro and presented the schematic diagram and construction details.

This month, the foil patterns and component placement diagram are presented along with a description of how to use the Dyna-Micro.

Final check

All integrated circuits should now be in place, hopefully in sockets. Remove the 8080A IC and check for the correct Voltages at its socket. Yau should find +5 volts at pin 20, +12 volts at pin 28 and -5 volts at pin 11. You should also check the PROM for -9 volts at pins 16 and 24. A PROM must be in one of the sockets for the -9 volts to be present.

Turn on the power with all the IC's in place and check the power supply voltages. They should be at their preset levels of +5 and ± 12 volts. If these are correct, your Dyna-Micro system should be operational.

Depress the R key. The LED displays should now indicate 003 (0000011) at the HI and 000 (00000000) at the La. The OUTPUT PORT 2 LED's may have some random data present. If this doesn't happen, remove the power and carefully check your system. Things to check for are solder bridges, cold solder-joints, unsoldered IC pins and incorrect IC orientation. Plated through holes don't have to be soldered unless there is a component or other lead going through them. Also check for +5 volts and ground at all the IC's.

If the LED's display the correct pattern, depress and release the s key. Each time that this key is pressed, the LO address information should be incremented by 1. If this doesn't happen, check the keyboard encoder section and the 1/0 sections.

If the LED's are operating correctly, enter some data (0 through 7) from the keyboard. The binary codes for these keys will be entered in the Data-Register (OUTPUT PORT 2) display in the three least-significant bits. You will note that as new data is entered, the old data is shifted to the left where it finally disappears as more new data is entered. The actual operation of the KEX software to input and output data will be discussed later.

How to use the Dyna-Micro

The Keyboard Executive software is the "heart" of the Dyna-Micro system. It allows you to examine data or pro-

PARTS LIST

R1, R2, R4, R5, R6-1000 ohms, ¼ W, 10%
R3—2200 ohms, ¼ W, 10%
R27-R30—220 ohms, ¼ W, 10%
C1—33 ,uF/6.3V electrolytic
C2—5 ,uF/50V electrolytic
C3, C5-C14-0.01 μF disc ceramic
C4—3.3 μF/ 16V electrolytic
D1-D24-Small red LED (Hewlett-Packard 5082-4484, Monsanto MV5075B, or equal.)
D25-1N751A, 5.1V Zener
D26-1N746, 3.3V Zener
IC1, IC22, IC23-SN7404 Hex inverter
IC2, IC3, IC30-SN7400 Quad 2-input NAND gate
IC4-SN74174-Quad type-D flip-flop
IC5-8224 Clock generator (Intel)
IC6, IC7-8216 Bus driver (Intel)
IC8-B080A CPU (Intel) (Must be "A" version)
IC9, IC10-8111-2 RAM memory (Intel)
IC13, IC17-SN74LS05 Open-collector hex inverter
IC14-SN74LS155 Dual 2-to-41ine decoder
IC15-1702A PROM memory (Intel)
IC18-SN74L42 BCD to decimal converter
IC19-SN7402 Quad 2-input NOR gate
IC20, IC21-SN74L04 Hex inverter
IC24-IC29-SN7475 Bistable latch
IC31-DM8095 or SN74365 Buffer
IC32, IC33-SN74148 8-to-3 line priority encoder
S1-S16-Keyswitches with legends
XTAL-6.750 MHz crystal, HC-18/U holder
Solderless breadboard-SK-10 IF18
Chassis-10" x 12" x 3" (Bud type AC-413)
Misc.-Binding posts, IC sockets, hardware

Optional IC's for expanded memory
IC11, IC12-8111-2 RAM memory
IC16-1702A PROM memory

Keyswitches are available from Solid State Systems, Inc., P.O. Box 617, Columbia, MO 65201. Order type LM or LFW-LT, with legends shown in Fig. 7.

Breadboarding socket is available from Circuit Design, Inc., Box 24, Shelton, CT 06484.

The following kits are available from Circuit Design, Inc., Box 24, Shelton, CT 06484. Phone 203-735-8774. All kits, except MMD-1IC, MMD-1 PROM, and MilD. 1 RAM come complete with construction details, experiments, and tutorial material.

#MMD-1CBK- Etched, plated-through PC board, keyboard parts and breadboarding socket. $125 postpaid.

#MMD-1K-Complete kit of parts including 1702A PROM preprogrammed with KEX software and power supply. $350 postpaid.

#MMD-1A-Completely assembled and tested system. $500 postpaid

#MMD-lIC-Microprocessor IC set includes one 8080A CPU, one 8224 clock generator, two 8216 bus drivers, two 8111-2 RAM memory, one 1702A PROM preprogrammed with KEX. $100 postpaid.

#MMD-1 PROM-Additional 256-WOrd PROM (1702A). $40 postpaid.

#MMD-1 RAM-Additional 256 words of RAM (8111-2). $15 postpaid.

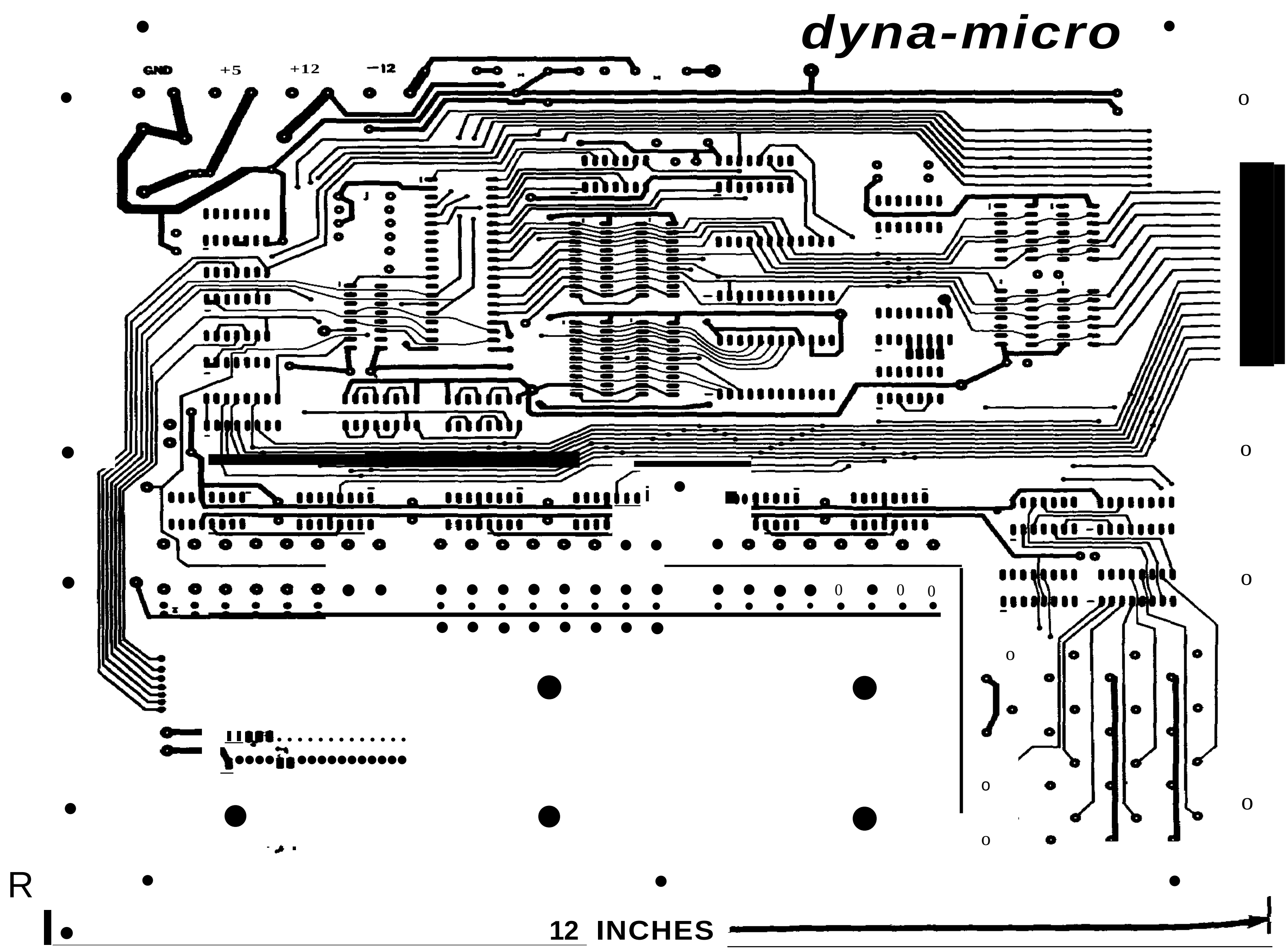


FIG. 3-FOIL PATTERN of the component-side of the double-sided board. The actual board measures 10 X 12 inches (254 X 305 mm.)

gram steps and to change data or program steps stored in the *R/W* (read/write) portion of the memory. We can also specify any address and start the program there.

The data keyswitches are labeled 0-7. When any one of these keys are depressed, data is entered in the corresponding octal format. The H key designates the **HI** address and the L key designates the **LO** address. The G represents Go and S represents See and Store. Three keys are not used by the KEX—A, B and C. The R key will always reset the computer and restart the KEX program. All manual data entry is through the keyboard in the basic Dyna-Micro system.

Whenever you want to start the system, depress keyswitch R. This will reset the KEX program and address the first location in the *R/W* section of the memory. This is HI=003 and LO=000. If you will only be using 256 words of *R/W* memory to get started, it must be in the locations allocated for IC9 and IC10. The KEX will not function without *R/W* memory.

To enter data, whether it will be used for new data or to address a memory location, simply depress the numbered keys as you would on a calculator. Data will be entered into the three least-significant (right-most) LED's and it will shift to the left as more data is entered. If a mistake is made, simply re-enter the data. Mistakes are shifted out and lost. The data-register LED's will display the data just entered from the keyboard and this may be used as the **HI** address by depressing H, or it may be used as the LO address by depressing L. These keys will transfer the data to the proper LED display register and it will be used by the 8080A to address a new memory location.

Whenever a new **HI** or LO address is specified by depressing either the H or L keys, the KEX program will always display the contents of the specified memory location on the data-register LED's. To examine the contents in the next location, depress the S key. By depressing the S key again and again, we can examine the contents in sequential memory locations. It should be noted that this S function follows increasing memory locations, *not* the sequential *flow* of a program.

To change the contents in a *R/W* memory location, simply load the address using the data input keys and the H and L keys. The old data presently in the location will immediately appear on the data-register LED's. Enter the new data into the data register using the numeric keys and then enter it into the *R/W* location by depressing the S key. After S is depressed, the new data is stored and the address is automatically incremented by one to address the next memory location. The data from the next location is now displayed on the data-register LED's.

The S key has two functions, both See and Store. How can we tell the difference? If the data has changed we will store it and see the next location. If the data hasn't changed, it will be stored in the same location that it originally came from and then the contents from the next memory location will be displayed. When we store old data back to the same location, we can't really see

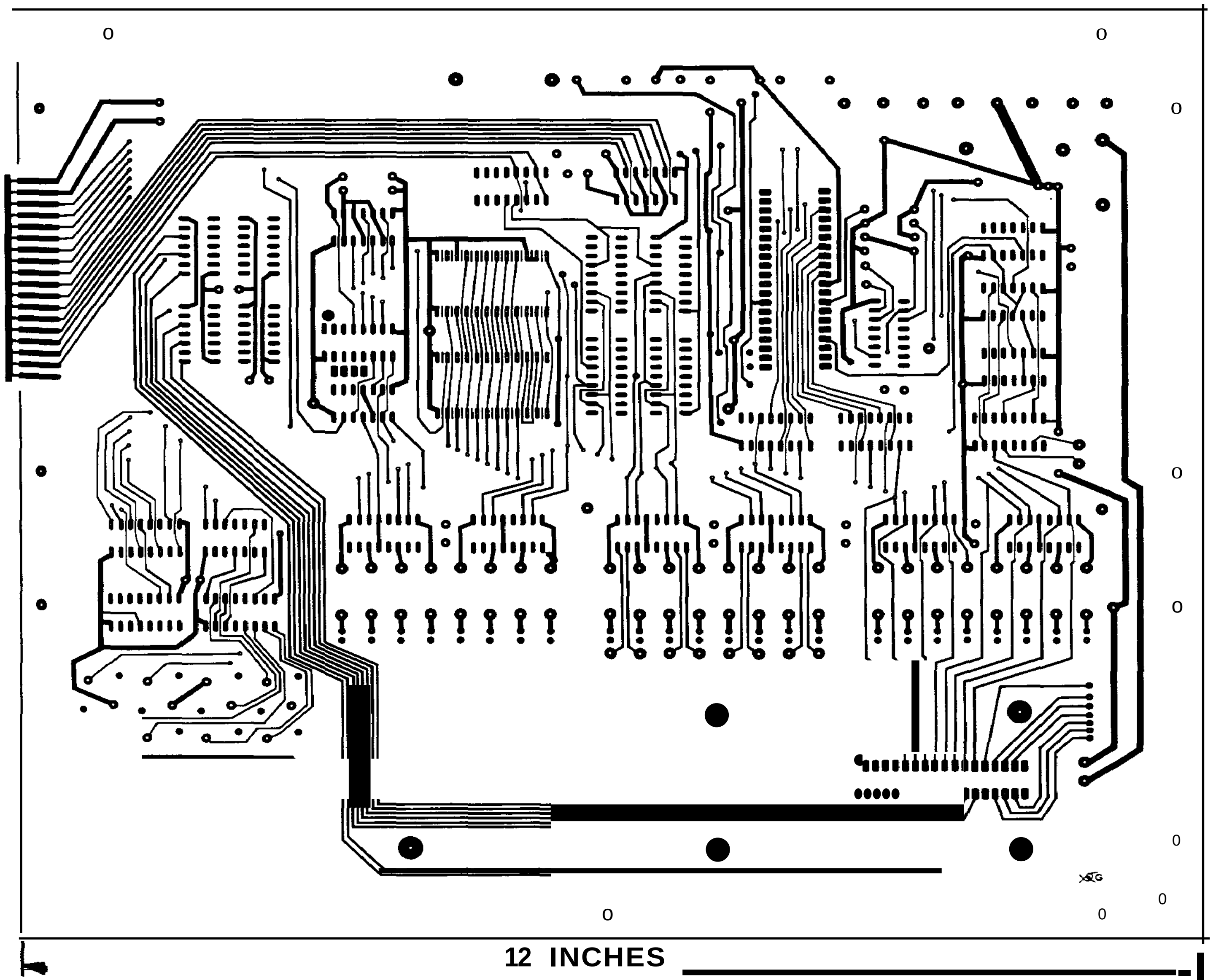


FIG. 4—FOIL PATTERN of the bottom of the double-sided board. A board with plated-through holes is recommended.

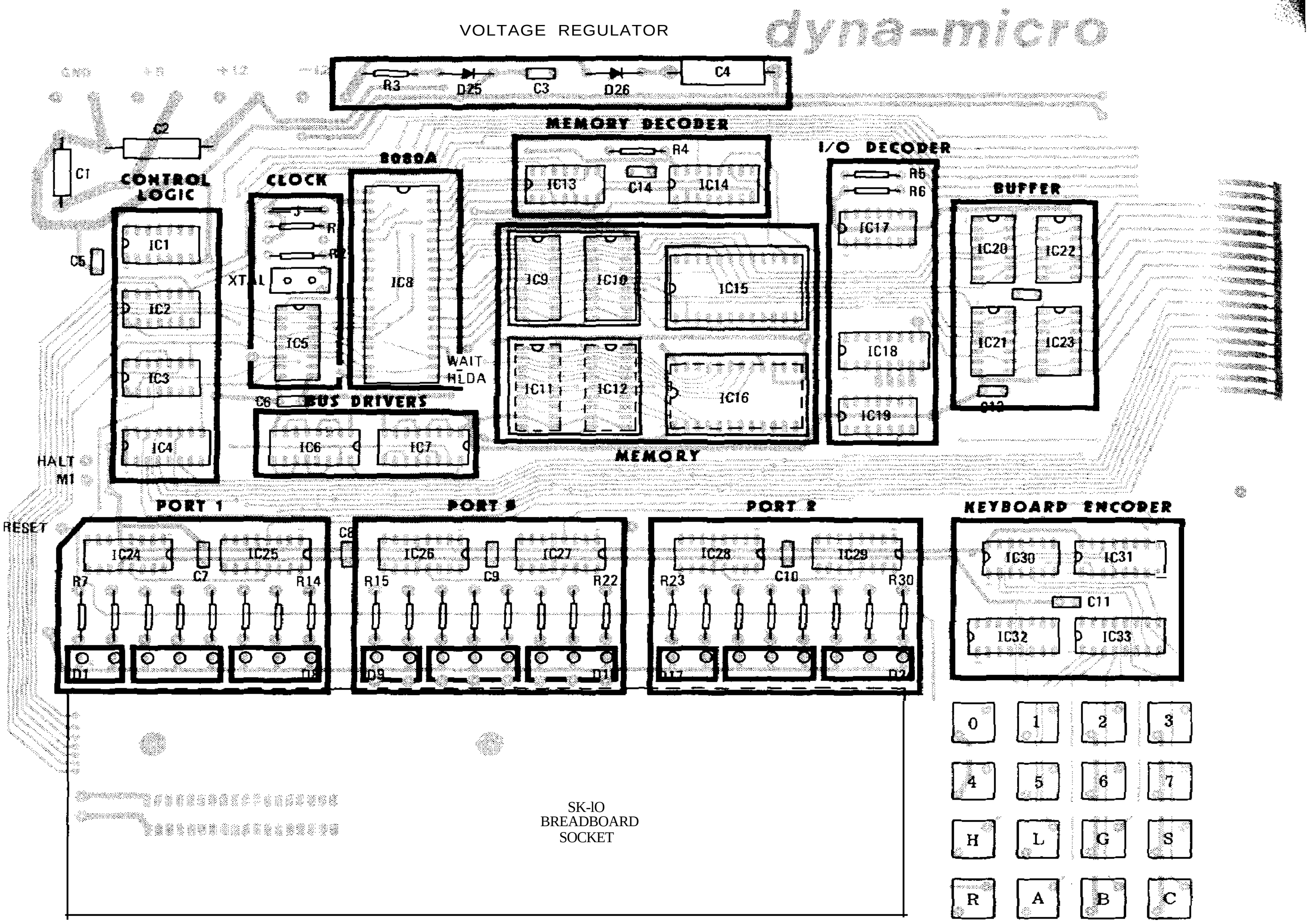


FIG. 5-COMPONENT PLACEMENT diagram. The board is shown divided into functional sections.

any change, but this is exactly what the KEX does. It displays data from a specific location, allows us to make changes and then puts it back. If no changes are made, the old data is *restored* to its *memory* location.

Once a program is entered into the computer through the keyboard, we can start it by loading *our* starting address and actuating the G key. This will transfer control from the KEX software to the *program* that we want to run. Starting addresses are loaded in the same way as previously described. Starting addresses don't .have to be in R/W memory, but can just as easily be in PROM.

If your program starts at the first location in R/W memory (003000) you can simply depressR followed by G. We can do this because KEX always resets the address back to this first R/W location.

Keys labeled A, B and C are not used by the KEX program. It should be remembered that the three LED output ports and the keyboard are not hardwired for use with the KEX program only. They are available for you to use in your programs. All fifteen keys may be used in any way you like, using software.

TABLE 1—MEMORY ALLOCATION

HI	LO	
000	000	KEY PROM
000	377	
001	000	OPTIONAL PROM
001	377	
002	000	OPTIONAL R/W MEMORY
002	377	
003	000	R/W MEMORY
003	377	
004	000	AVAILABLE FOR USER EXPANSION
377	377	

How KEX operates

The keyboard Executive software is contained in a single 1702A type PROM in the location allocated for ICIS. This contains all the necessary software to operate the keyboard and the LED displays. This is our software controlled "front panel", since the keys and LED's perform functions determined by the KEX software.

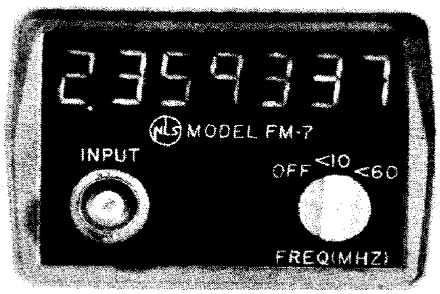
Whenever the R key is depressed, the SOSOA CPU will start to execute the program that starts at location 0. Look· ing at the software listing for the KEX program (Table II), you will see that immediately after starting at location 0, the software instructions cause the computer to jump to location HI==OOO, LO=070 (HI=OOO throughout the KEX program) where we start the program by pointing to the first R/W memory address (003 000.) The address and the data in that location are displayed on the three output ports. This is done between POINTA and POINTe in the program (see Table II). The software between POINTC and POINTD will do the necessary tasks to input new data from the keyboard and shift the data onto the LED's. The shifting is done inside the SOSOA with software instructions. Doing this by hardware would require

(contillued 011 page 84)

FREQUENCY9

MEASURE IT ANYWHERE
To 60 MHz with the FM-7,

NLS proudly announces a NEW Frequency Meter.



With Rechargeable Batteries & Charger Unit **$195**

Features Include:
• **Portable,** battery operated for measurement anywhere! _ 10 Hz to 60 MHz Input with LED 7-digit resolution. _ High sensitivity; 30 mV - 50 Hz to 30 MHz (100 mV - 10 Hz to 60 MHz). _ Input overload protection. _ Small, only 1.9"H x 2.7"W x 4"0.. • High stability internal time base.. _ For field application, servicing or **production testing.**
See your locs! distributor!
Distributor inquiries invited.

Non-Linear Systems, **Inc.**
Originator ot the digital voltmeter.
Box N. Del Mar, California 92014
Telephone (714) 755-1134 TWX 910-322-1132

Circle 72 on reader service card

BUILD A COMPUTER
(continued from page 44)

many more Ie's, but it takes relatively few software steps.

The software routines at POINTD, POINTE, POINTF and POINTG make up what is called a command decoder. The software decodes the key-switches into real actions. Depressing H or L causes the data ternporatily stored in the 8080A as numeric key inputs to be output to eIther the HI or LO set of LED's. The s key causes the current or new data to be put back into the current memory location. Depressing G causes the computer to use the HI and LO address as the startIDg POint for a new program.

The TIMOUT and KBRD software subroutines have specific tasks. The TIMOUT will count its way through various loops for about 10 milliseconds while the KBRD subroutine will input a code from the keyboard. The KBRD subroutine has some unique features that illustrate an interesting hardware. software tradeoff. The keyswitches used in the Dyna-Micro are not bounce free, so that when the switches are opened or closed, they can often re-make or re-break the contacts. This can be confusing to the computer since it can't distinguish between a real switch closure and a bounce. We don't want

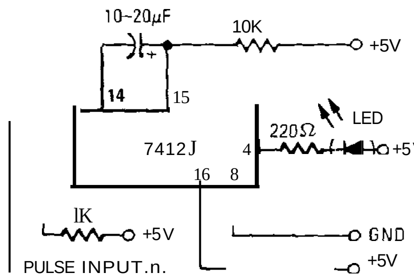


FIG. 6-MONOSTABLE·LED circuit is used for check-out of the Dyna·Micro.

0	1	2	3
4	5	6	7
H	L	G	S
R	A	B	C

FIG. 7-KEYBOARD LEGENDS are oriented as shown,

A user's group has already been formed for the Dyna-Micro. Interested people should contact:
Dr. Frank Settle, Jr.
Digital Directions
Box 1053
Lexington, VA 24450

CASTLE Timesavers
,,. f(Jf the P'(Jfessi(J1I1I1

SUBBER®
TV signal circuit analysts

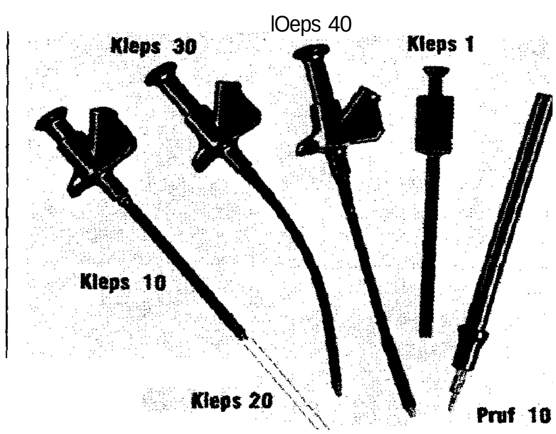
• Mode in USA.

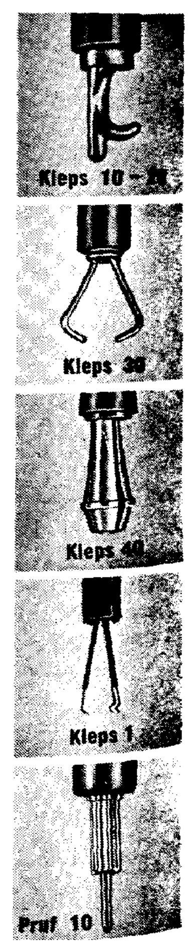
Castle products
– advanced technology
– modern styling
– and they work'

ASk Y,our electronic distributor tor them or write for more details

CASTLE ELECTRONICS, INC.
5715 N Western Ave. Chicago. Ill 60645
• Ph [312J 7281800

Circle 73 on reader service card





Clever Kleps

Test probes designed by your needs - Push to seize, push to release (all Kleps spring loaded).
Kleps 10. Boathook clamp grips wires, lugs, terminals. Accepts banana plug or bare wire lead. 4¾" long. $1.39
Kleps 20. Same, but 7" long. $1.49
Kleps 30. Completely flexible. Forked-tongue gripper. Accepts banana plug or bare lead. 6" long. $1.79
Kleps 40. Completely flexible. 3-segment automatic collet firmly grips wire ends, PC-board terminals, connector pins. Accepts banana plug or plain Wire. 61f4" long. $2.59
Kleps 1. Economy Kleps for light line work (not lab quality). Meshing claws. 41f2" long. $ .99
Pruf 10. Versatile test prod. Solder connection. Molded phenolic. Doubles as scribing tool. "Bunch" pin fits banana jack. Phone tip. 5V2" long. $ .89
All in red or black· specify. (Add 50¢ postage and handling).
Write for complete catalog of - test probes, plugs, sockets, connectors, earphones, headsets, miniature components.

A vailable through your local distributor, or write to:
RVE INDUSTRIES INC.
.128 Spencer Place, Mamaroneck, N.Y. 10543
In Canada: Rye Industries (Canada) Ltd

WE GOOFED

WE SURE DID! Last month we ran the first part of this construction article on the Dyna-Micro and we erred. It seems that one or more gremlins (little green people) found their way into our editorial offices and stole-that's right, stole-74 lines of text from the article. The text was stolen between the last paragraph on page 36 and the opening paragraph on page 74. To correct this evil crime, we are reprinting the two paragraphs in italics along with the missing text. To prevent this from happening again, we have tightened security around our offices.

The clock circuit uses an Intel 8224 integrated circuit. This is a crystal clock oscillator that provides the proper MOS clock levels for the 8080 system. It also contains circuitry for a TTL level clock (Ø2), RESET *and* READY inputs. Construction of the clock circuit begins by inserting components Rl, R2, the 6.75 MHz crystal, IC5 and the jumper. Good quality IC sockets are recommended for all the integrated circuits.

After inserting and soldering the parts, the clock section should be checked. To do this, apply power and check for voltages on the 8224 chip. You should observe +5 at pin 16, +12 at pin 9 and ground at pin 8. Clock operation can be checked at pins 10 and 11. These are the MOS level outputs that swing between +12V and ground. The signals can be observed with a good scope. The TTL output on pin 6 can also be checked with a scope or with a monostable-LED circuit. The monostable circuit can be constructed on the SK-lO socket before it is added to the system.

With the 6.75-MHz crystal, the output frequency will be 750 kHz. This is slower than the maximum 2-MHz frequency that the 8080A will operate at, but a slow frequency was chosen to allow for slow access times of the PROM's.

The output ports are constructed next by inserting and soldering all the LED's (Dl through D24). Be sure to observe the polarity as shown by the symbol near 01. Add all the 220-ohm resistors (R7 through R30) and insert the SN7475 latches. Be careful to orient the IC's in their sockets. The foil pattern has a small mark near pin 1 for all the IC's.

With the latches and parts installed and soldered, apply power to the system. All of the LED's should light. If any are not on, check the associated SN7475 latch. With power still applied, ground the input pins to the latches, one at a time. Since all the inputs are on eight common data bus-lines, only eight inputs must be grounded to check all the LED's and latches. Ground pins 2, 3, 6 and 7 on IC24 and IC25. One of the LED's in each group of eight should go out, one at a time. If this doesn't happen, again check the SN7475's.

The keyboard section consists of 16 keyswitches-15 are used to input data and one is hardwired to the 8224 chip to reset the Dyna-Micro. The keyswitch closures are encoded by two SN74148 octal encoders and the encoded binary data is gated onto the bus through a three-state DM8095 or SN74365IC.

Insert and carefully solder the keyswitches to the printed-circuit board and then insert the four integrated circuits, IC30 through IC33.

The keyboard section is tested by monitoring the data on the LED's. Carefully ground pins 1 and 15 on the three-state driver, Ie31. This will cause data from the keyswitch encoders to *constantly be fed to the bus. With these two pins grounded, apply power and depress the keys, one at a time. The binary data for each keyswitch will be indicated on the LED's at all of the output-ports simultaneously. Note that the most significant bit, D7, will be on whenever one of the keys is depressed. This is often called a 'flag' since it is used to flag down the computer and tell it that one of the switches is ready with data.*

the computer to sense each bounce as a key closure so we would like some way to filter them out. Additional circuitry including latches, clocks and monostables could do this for us, but it complicates the system. We can also do the debouncing via software.

The KBRD subroutine will recognize any key closure, but it will only input the key codes after being sure that the key is closed and not bouncing. It does this by waiting after sensing a closure and then rechecking the switch to be sure it is still closed. It also checks when we release a key to be sure that it has stopped bouncing before it tries to sense another key being depressed by the user. We have traded some additional software steps for a great deal of hardware. Since there was plenty of PROM left, it was easy to include.

The TIMOUT and KBRD software

(Table II is on page 86)
(Text continues on page 90)

Text continues on page 90

TABLE II-KEYBOARD EXECUTIVE (KEX) PROGRAM

```
                              *000 000
000 000   303                 JMP
000 001   070                 START
000 002   000                 0

                              ;JUMP UP TO R/W MEMORY TO
                               BE USED BY RESTARTS &
                               VECTORED INTERRUPTS
                              *000 010
000 010   303                 JMP
000 011   010                 010
000 012   003                 003
                              *000 020
000 020   303                 JMP
000 021   020                 020
000 022   003                 003
                              *000 030
000 030   303                 JMP
000 031   030                 030
000 032   003                 003
                              *000 040
000 040   303                 JMP
000 041   040                 040
000 042   003                 003
                              *000 J50
000 050   303                 JMP
000 051   050                 050
000 052   003                 003
                              *000 060
000 060   303                 JMP
000 061   060                 060
000 062   003                 003

                              ;BEGINNING OF MAIN PROGRAM
                              *000 070
000 070   061   START,        LXISP     ;SET STACK POINTER
                                         TO TOP OF R/W MEM.
000 071   000                 000
000 072   004                 004
000 073   041                 LXIH      /INITIAL VALUE FOR
                                         H & L
000 074   000                 000
000 075   003                 003
000 076   116   POINT A,      MOVCM     /LOAD MEM DATA INTO
                                         TEMP DATA BUFFER
000 077   174                 MOVAH     /OUTPUT HI TO LED'S
000 100   323                 OUT
000 101   001                 001
000 102   175                 MOVAL     /OUTPUT LOW TO
                                         LED'S
000 103   323                 OUT
000 104   000                 000
000 105   171   POINT B,      MOVAC     /OUTPUT TEMP. DATA
                                         BUFFER DATA TO
                                         LED'S
000 106   323                 OUT
000 107   002                 002
000 110   315   POINT C,      CALL      ;WAIT & INPUT NEXT
                                         KEY CLOSURE
000 111   315                 KBRD
000 112   000                 0
000 113   376                 CPI
000 114   010                 010
000 115   322                 JNC       / JUMP IF KEY WAS <
                                         010
000 116   134                 POINT D   /(0-7, OCTAL
                                         DIGIT)
000 117   000                 0
000 120   107                 MOVBA     /SAVE KEY CODE
000 121   171                 MOVAC     /GET OLD VALUE
000 122   027                 RAL       ;ROTATE 3 TIMES
```

```
000 123  027              RAL
000 124  027              RAL
000 125  346              ANI      jMASK OUT LEAST
                                    SIG. OCTAL DIGIT
000 126  370              370
000 127. 260              DRAB     JOR IN NEW OCTAL
                                    DIGIT
000 130  117              MOVCA    jPUT NEW DATA
                                    BACK INTO BUFFER
000 131  303              JMP
000 132  105              POINT B
000 133  000              0
000 134  376  POINT 0,    CPI
000 135  011              011      j"L" KEY
000 136  302              JNZ      /JUMP IF NOT AN "L"
000 137  145              POINT E
000 140  000              0
000 141  151              MOVLC    IPUT BUFFER DATA
                                    IN L
000 142  303              JMP
000 143  076              POINT A
000 144  000              0
000 145  376  POINT E,    CPI
000 146  010              010      I"H" KEY
000 147  302              JNZ      j JUMP IF NOT AN "H"
000 150  156              POINT F
000 151  000              0
000 152  141              MOVHC    IPUT BUFFER DATA
                                    IN H
DOD 153  303              JMP
000 154  076              POINT A
000 155  000              0
DOD 156  376  POINT F,    CPI
000 157  013              013      I"S" KEY
000 160  302              JNZ      jJUMP IF NOT "S"
000 161  170              POINT G
000 162  000              0
000 163  161              MOVMC    jPUT TEMP. DATA
                                    INTO MEMORY
DOD 164  043              INHX     jINCREMENT H & L
ODD 165  303              JMP
000 166  076              POINT A
DOD 167  000              0
000 170  376  POINT G,    CPI
000 171  012              012      j"G" KEY
000 172  302              JNZ      jJUMP IF NOT "Gil
DOD 173  110              POINT C
DOD 174  000              0
000 175  351              PCHL     IGO EXECUTE PGM
                                    POINTED TO BY
                                    H & L

                          ITHIS 10 MSEC DELAY
                           DISTURBS NO REGISTERS OR
                           FLAG

                          *000 277
DOD 277  365  TIMOUT,     PUSHPSW  jSAVE REGISTERS
000 300  325              PUSHD
                          LXID     jLOAD D & E WITH
                                    VALUE TO BE
000 301  021                        DECREMENTED
000 302  046              046
000 303  001              001
000 304  033  MORE,       DCXD     jJUMP IN THIS
                                    LOOP UNTIL
000 305  172              MOVAD    jD & E ARE BOTH
                                    ZERO
000 306  263              ORAE
000 307  302              JNZ
000 310  304              MORE
000 311  000              0
000 312  321              POPD
000 313  361              POPPSW   lRESTORE
                                    REGISTERS
000 314  311              RET

                          ITHE KBRD ROUTINE
                           DEBOUNCES KEY CLOSURES
                          lAND TRANSLATES KEY CODES
```

Table II continues on page 89 (text continues on page 90)

Table continued from page 87
Text continues on page 90

```
                          /FLAGS AND REG A ARE
                           CHANGED
                          /A0-A3 = CODE; A4-A7
                           0000

000 315  333  KBRD,   IN      /INPUT FROM
                               KEYBOARD
                               ENCODERS
000 316  000          000
000 317  267          ORAA    /SET FLAGS
000 320  372          JM      /JUMP BACK IF LAST
                               KEY NOT RELEASED
000 321  315          KBRD
000 322  000          0
000 323  315          CALL    WAIT 10 MSEC
000 324  277          TIMOUT
000 325  000          0
000 326  333  FLAGCK, IN
000 327  000          000
000 330  267          ORAA
000 331  362          JP      /JUMP BACK TO WAIT
                               FOR A NEW
000 332  326          FLAGCK  /KEY TO BE PRESSED
000 333  000          0
000 334  315          CALL    /WAIT 10 MSEC FOR
                               BOUNCING
000 335  277          TIMOUT
000 336  000          0
000 337  333          IN
000 340  000          000
000 341  267          ORAA
000 342  362          JP      /JUMP BACK IF NEW
                               KEY NOT STILL
000 343  326          FLAGCK  /PRESSED (FALSE
                               ALARM)
000 344  000          0
000 345  346          ANI     /MASK OUT ALL BUT
                               KEY CODE
000 346  017          017
000 347  345          PUSHH   /SAVE H & L
000 350  046          MVIH    /ZERO H REG
000 351  000          000
000 352  306          ADI     /ADD THE ADDRESS
                               OF THE BEGINNING
000 353  360          360     /OF THE TABLE TO
                               THE KEY CODE
000 354  157          MOVLA   /
000 355  176          MOVAM   /FETCH NEW VALUE
                               FROM TABLE
000 356  341          POPH    /RESTORE H & L
000 357  311          RET
                      /THIS TRANSLATION TABLE
                       CONVERTS THE CODE
                      /GENERATED BY KEY CLOSURES
                       TO THE CODE
                      /USED BY THE MAIN KEX
                       PROGRAM

000 360  000  TABLE,  000
000 361  001          001
000 362  002          002
000 363  003          003
000 364  004          004
000 365  005          005
000 366  006          006
000 367  007          007
000 370  013          013     /S
000 371  000          000     /THIS CODE CAN'T
                               BE GENERATED
000 372  017          017     /C
000 373  012          012     /G
000 374  010          010     /H
000 375  011          011     /L
000 376  015          015     /A
000 377  016          016     /B
```

end of Table II *(text continues on page 90)*

(continued from page 87)

segments have been set up as subroutines and can be used in your software and in the experiments. Each of these subroutines may be started with a CALL instruction, 315. The TIMOUT subroutine does not affect any of the registers or flags and it only serves to delay the software flow by 10 ms.

An important distinction between the 8008 and the 8080 processors is in the use of subroutines. In the 8008, return-pointer addresses were stored in the 8008 IC itself. In the 8080, these return-pointer addresses are stored in a portion of the R/W memory. This is called a "stack" area. Whenever a subroutine is used, we want to execute the subroutine and then return back to the normal program flow. These return addresses are very important to the computer since they provide the only link between the subroutine and the main program. If we are to store them in a portion of R/W memory, the computer must know where this storage area is if it is to be able to use the addresses properly. In the KEX software, this is preset to be the top of the R/W memory with instructions at locations 070, 071 and 072. The LXISP instruction loads an internal 8080 stack-pointer register to HI= 004, LO= 000. Since the stack-pointer register is *decremented* to point to a new location before anything is stored, the first stack location will be HI=003, LO=377. Check your 16″ bit binary numbers if this looks a little confusing.

You can use the stack as set up by the KEX (generally a good idea) or you can put your own stack anywhere you want, just by using the LXISP instruction. Remember to avoid the stack area when writing your programs. Remember, too, that you can't put the stack in an area of non-existent memory or in PROM.

You will use the stack area and you'll see how it can also be used to temporarily store data. This will be covered in the software modules. Let's see how the TIMOUT and KBRD subroutines can be used in our own software. We will use the software stack already set up in KEX.

Let's input a keyboard character, add a constant to the binary code for that character and display the result. We would first CALL the KRBD subroutine to input a binary key-code, then add the constant and display the result. The software listed in Table HI will do this:

You can enter this with the KEX program. Depress the R key and start entering data. Enter 000 at location 003 004 so we'll first add zero to the codes. This will let us check what values are assigned to each key. Write down the codes. Go back and restart

TABLE III

HI	LO	INSTR	MNEMONIC	
003	000	315	CALL	/ Input keyboard character
003	001	315	KBRD	/Subroutine's IO address
003	002	000	a	/Subroutine's HI address
003	003	306	ADI	IAdd the following DATA to
003	004	???	DATA	/the contents of register A
003	005	323	OUT	IOutput data from register A to
003	006	000	000	/device 000 (LEOs)
003	007	303	JMP	/Jump back to program at
003	010	000	000	fIO address = 000
003	011	003	003	IHI address = 003

TABLE IV

Assume HI = 003 throughout this program

LO	INSTR	MNEMONIC	
000	006	MVIB	ILoad register B with the following data
001	370	370	10ata; time constant
002	315	CALL	/ Call TIMOUT subroutine at
003	277	277	/ LO address = 277
004	000	000	IHI address = 000
005	005	OECB	/ Decrement B by 1
006	302	JNZ	/ Jump if result is not zero to
007	002	002	/ LO address = 002
010	003	003	IHI address = 003
011	076	MVIA	/ Load register A with the following data
012	377	377	10ata; all1's
013	323	OUT	IOutput to device,
014	000	000	Idevice 000 (LED's)
015	166	HLT	/ Stop once you reach here

KEX and change the value in 003 004 10, say, 005. This will add 5 to each code. Restart your software and see if this is the case. Congratulations, you have just done your first software experiment! The instructions at 003 003 and 003 004 could be changed to do other things to the data. Can you suggest one?

The 10 ms delay routine, TIMOUT, can be useful when we want a software delay that is in multiples of 10 ms. The software routine listed in Table IV will delay an output of all I's on OUTPUT PORT 0 by about 2.5 seconds after the program is started. Try it. Can you see how the time delay might be shortened? Can you see any use for programs like this?

The keyboard input subroutine, KBRD, is called at address 000 315 and the time delay subroutine, TIMOUT, is called at address 000 277.

Next month, this construction article concludes with a description of the 8080A microprocessor and how the Oyna-Micro works. This will include an explanation of how the memory is accessed and how the Oyna-Micro selects inpUt/output devices via software. R-E