EK-7 Dual Digitizer

Assembly & Using Instructions

There are plenty of times when a switch is a great way to control things- like when you want to turn something on and off, or select a preset. But when you're just playing around looking for the right sound, there's nothing quite like a knob. Unless it's a joystick.

Knob or joystick, either one- we need some way to digitize it's position so a computer can read, save and manipulate the data various ways. And preferably it should be a cheap and simple way.

We need something we'll call a digitizer. It's an analog-to-digital converter, really; the only reason I don't think we should call it an ADC is that we reserve that term for something more elaborate than what we are getting into. This is really simple.

In every electronic scheme that I know of to convert an analog parameter to a digital one there is a thing called a comparator. See figure 1. The thing it compares are the voltages at its "+" and "-" inputs. If the voltage at the "+" input is greater, the output is at a high voltage. If the "-" input is greater, the output is driven to a low voltage.

The elaborate ADC's use the comparator as only a small part of a larger circuit that will probably look something like figure 2. When it's time to quantize the voltage to be measured, the counter is reset and its digital output goes to zero. Because of this, the D/A puts out a low voltage (in this scheme you must first have a digital to analog conversion before you can have the reverse). The output of the D/A will probably be lower than the voltage that is being measured, so the output of the comparator is high and allows pulses to pass from the clock through the NAND gate to the counter. The counter counts up and, as it does, the output of the D/A increases. When the output of the D/A exceeds the voltage to be measured, the comparators output goes low and clock pulses can no longer pass through the gate to the counter. At that point, the counter's output is a digital representation of the analog voltage being measured.

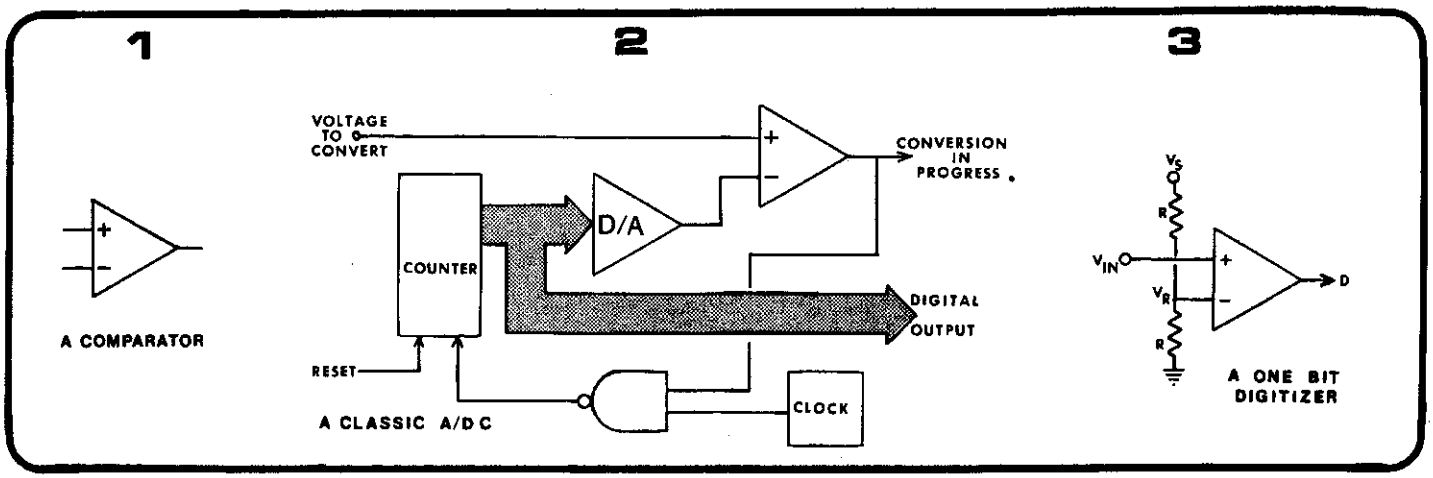
There are a number of variations on this design that have to do with the way the counter works, and in a computer based system it is common to replace both the clock and counter with software. Unfortunately, the common features of all these variations are modest complexity and/or relatively slow conversion rate.

hardware

Now, for a <u>really simple</u> digitizer, take a look at figure 3. Since the resistors in the divider that detirmines the reference voltage (Vr) are equal, the digital output is a 1 (high) if the voltage is greater than 1/2 the supply voltage and 0 if the input is less then Vs/2. I know what you're thinking, and you're right. A one bit digitizer isn't exactly an improvement over a switch in most cases.

OK, let's add another stage. Only on this one, let's make the reference voltage a function of the output state of the first stage. Schematically, this is represented in figure 4.

In order to easily see how this circuit works, you have to assume that Vr1 (the voltage at the junction of the two R1's) is constant at Vs/2. In fact, this voltage will change as the comparator output D1 changes



and alternately sinks or sources current through the two resistors, R2. But as long as the value R1 is kept much lower than the value R2 (the lower the better, at least 1/10), the change in Vr1 will not be too significant.

Imagine that a voltage which is increasing from ground to supply is applied to the input of the digitizer. When at ground, the voltage is less than Vs/2, so D1 is low (ground). The two R2's now form a voltage divider at the junction of which is a voltage equal to 1/2 of Vs/2, or 1/4 of the supply voltage (Vs/4). This voltage (Vs/4) is the reference voltage for the new stage. Since we said that our input voltage was initially at ground (which is less than Vs/4), the output of the new stage is also low. In binary, the output of the two stages is 00. An equivalent circuit would look like figure 5.

Now we increase the input voltage and, as it exceeds Vs/4, the output of the new stage changes from low to high. That's all that happens; the binary output of the two stages is now 01.

We continue to increase the input voltage and, as it exceeds Vs/2, the output of the first stage goes high. But, that's not all, because with the output high (at Vs), an equivalent circuit of the voltage divider that forms the reference for the new stage looks like figure 6. Since the input voltage is less than 3/4 of the supply voltage, the new stage changes state back to low and all is once again stable with a binary output of 10.

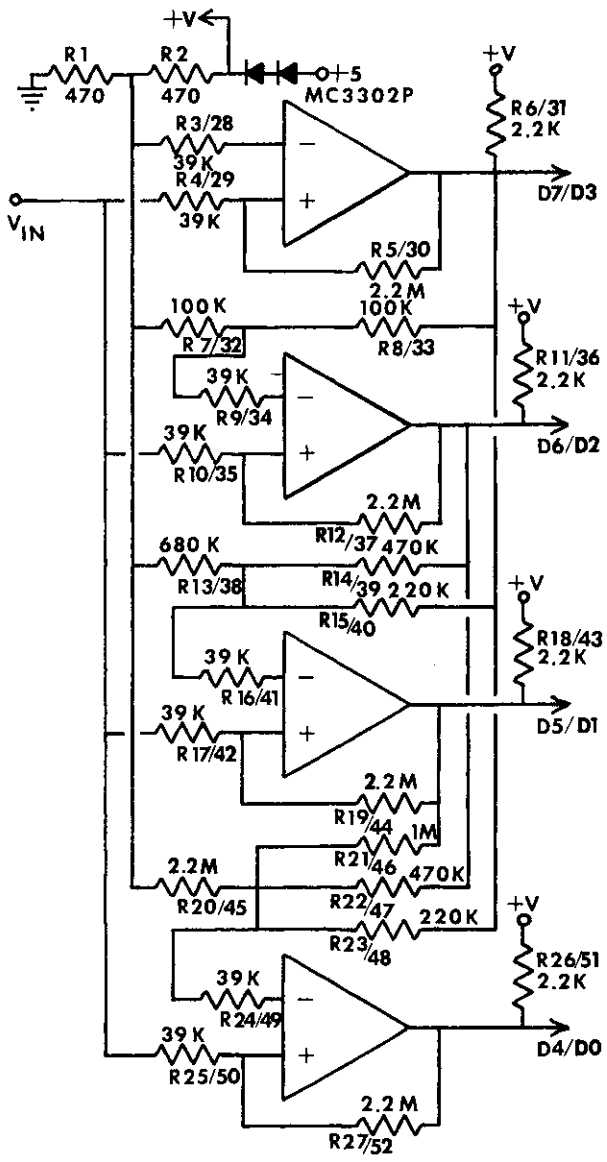
Increasing the input voltage further will exceed 3Vs/4. The new stage again changes to a high state and the binary output of the two stages reads 11.

Additional stages can be added in much the same way we just added the second stage. Each new stage becomes the least significant bit of the digitizer and its reference voltage is a weighted sum of the outputs of the more significant stages. Using 5% resistors, the scheme can be carried to 5 bits. 1%'ers would probably take us up to 6 bit resolution; 7 or 8 bit resolution should be realizable by going to active summing amps instead of the passive summing we've used. But, then you're back to complicated again.

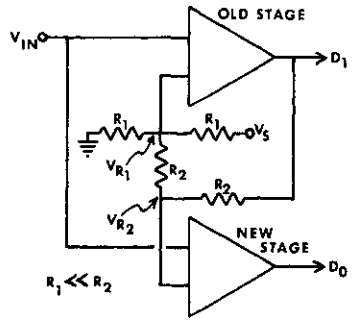
Instead, we'll stop at an easily obtainable 4 bits with the design shown in figure 7. Since the MC3302P is a quad comparator, only one IC is used in this circuit. Like I said, it's simple. Resistors R5, R12, R19, and R27 have been added to give just the slightest hysteresis (positive feedback) to each stage to help overcome any uncertainty at input voltages that correspond exactly to 'change of state' points. When powered from a computer's 5 volt supply, the range of input voltages is also 0 to 5 volts and the pot to be digitized is hung across the supply as laboriously

7

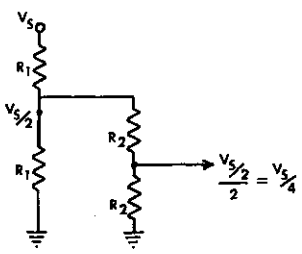
A FOUR BIT DIGITIZER



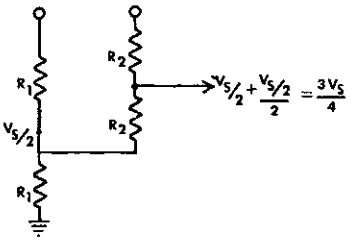
4



5

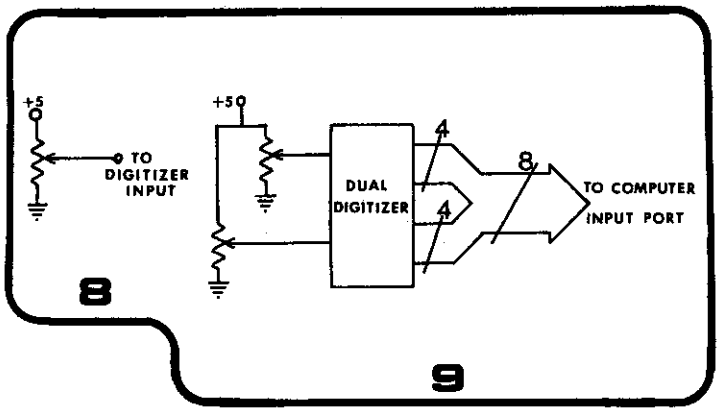


6



depicted in the formidable technical drawing of figure 8. At this point, we may as well establish the standard that the pot should be wired so clockwise rotation of the control causes the output of the digitizer to go from $0 to $F (see test program).

I believe that the most useful configuration for this circuitry is actually two digitizers on a single board, with each half providing half of an 8 bit word. The configuration shown in figure 9 is Paia's EK-7 and is made to plug directly into input port #2 of a Paia 8700 computer. It can also be connected to any 8 bit input port of any computer.



assembly

INSTALL 13 JUMPERS

INSTALL CAPACITORS

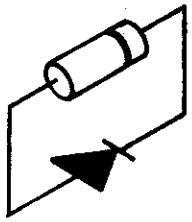
C1	.05	CERAMIC DISK
C2	33MFD/6V.	ELECTROLYTIC
C3	33MFD/6V.	ELECTROLYTIC

OBSERVE POLARITY OF C2 AND C3

INSTALL RESISTORS (ALL 1/8 W. 5%)

R1	470 OHM	YEL-VIO-BRN
R2	470 OHM	"
R3	39K	ORA-WHT-ORA
R4	39K	"
R5	2.2 MEG	RED-RED-GRN
R6	2200 OHM	RED-RED-RED
R7	100K	BRN-BLK-YEL
R8	100K	"
R9	39K	ORA-WHT-ORA
R10	39K	"
R11	2200 OHM	RED-RED-RED
R12	2.2 MEG	RED-RED-GRN
R13	680K	BLU-GREY-YEL
R14	470K	YEL-VIO-YEL
R15	220K	RED-RED-YEL
R16	39K	ORA-WHT-ORA
R17	39K	"
R18	2200	RED-RED-RED
R19	2.2 MEG	RED-RED-GRN
R20	2.2 MEG	"
R21	1 MEG	BRN-BLK-GRN
R22	470K	YEL-VIO-YEL
R23	220K	RED-RED-YEL
R24	39K	ORA-WHT-ORA
R25	39K	"
R26	2200	RED-RED-RED
R27	2.2 MEG	RED-RED-GRN
R28	39K	ORA-WHT-ORA
R29	39K	"
R30	2.2 MEG	RED-RED-GRN
R31	2200 OHM	RED-RED-RED
R32	100K	BRN-BLK-YEL
R33	100K	"
R34	39K	ORA-WHT-ORA
R35	39K	"
R36	2200 OHM	RED-RED-RED
R37	2.2 MEG	RED-RED-GRN
R38	680K	BLU-GREY-YEL
R39	470K	YEL-VIO-YEL
R40	220K	RED-RED-YEL
R41	39K	ORA-WHT-ORA
R42	39K	"
R43	2200	RED-RED-RED
R44	2.2 MEG	RED-RED-GRN
R45	2.2 MEG	"
R46	1 MEG	BRN-BLK-GRN
R47	470K	YEL-VIO-YEL
R48	220K	RED-RED-YEL
R49	39K	ORA-WHT-ORA
R50	39K	"
R51	2200	RED-RED-RED
R52	2.2 MEG	RED-RED-GRN

INSTALL THE TWO 1N914 DIODES
OBSERVE POLARITY AS ILLUSTRATED

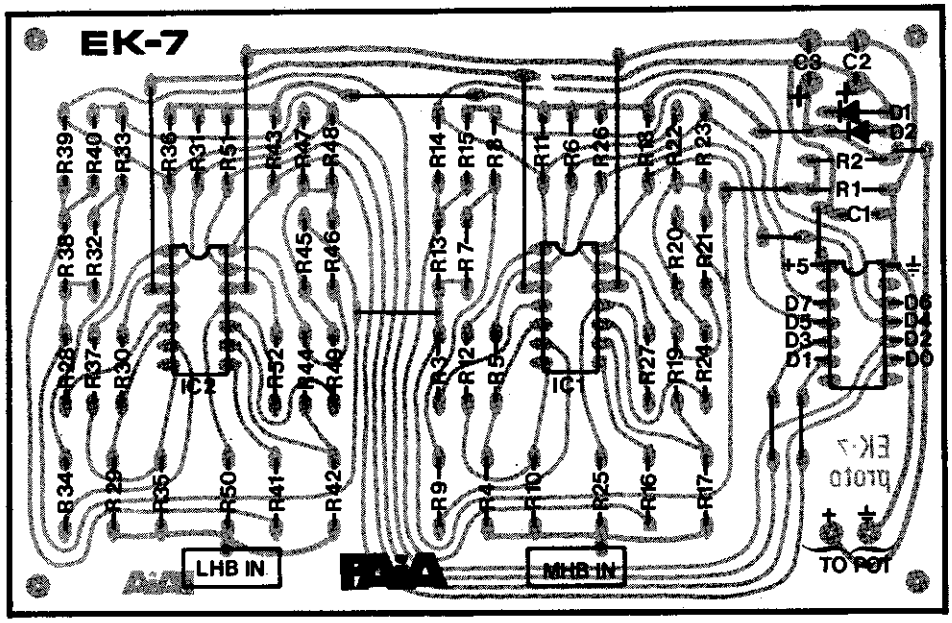


D1	1N914	SILICON DIODE
D2	1N914	

INSTALL 3 - 14 PIN DIP SOCKETS ONE EACH AT IC1 AND IC2 AND THE LAST AT THE DATA CONNECTOR

INSTALL IC1 AND IC2

IC1 MC3302 QUAD COMP (OR EQUIV)
IC2 "



software consideration

The nicest thing about the digitizer is that it is easy to program for. There are no clocks to worry about and no elaborate software overhead (in fact, none at all). You just read the port to which the digitizer is connected to find the state of the knobs.

A good first example is the short program written for an 8700 to test the unit's operation shown in Listing 1. This program reads the output of the digitizer and shows the result in the 8700's displays. When the value of either of the digitizer outputs changes, the beeper sounds. As the knobs are rotated, the displays should show that the output increases or decreases sequentially without skipping any of the hexadecimal numerals $0 - $F and that there is no interaction between the two digitizer sections.

```
                0010  :TEST FOR 4 BIT DIGITIZER
                0020  :DIGITIZER INPUTS TO PORT #2
                0030  :
                0040  : A) SHOW OUTPUT OF DIGITIZER
                0050  : B) BEEP WHEN VALUE CHANGES
                0130  :
1000- AD 08 08  0140  STAR LDA INPT     :GET DIGIT
1003- C5 20     0160       CMP *TEMP    :SAME AS LAST?
1005- F0 04     0170       BEQ LP1      :YES-BRANCH
1007- 18        0180       CLC          :PREPARE
1008- 20 22 0F  0190       JSR BEEP     :AND BEEP
100B- 85 20     0200  LP1  STA *TEMP    :SAVE VALUE
100D- 8D 20 08  0210       STA DISP     :SHOW VALUE
1010- 4C 00 10  0220       JMP STAR     :AND SO ON...
                0230  :
                0240  :
                0250  END  .EN
```

list1

The fact that there are two digitizer sections on the EK-7, one contributing the upper half-byte and the other the lower half-byte is going to be of great significance in some future software and hardware that we'll be doing.

For now, we'll use the PINK TUNES software (Polyphony July/August 78, pp. 22-26) as an example. When you review PINK TUNES, you'll notice that the statistical properties of the note durations (half notes, quarter notes, dotted notes, etc.) are controlled by the upper half-byte (UHB) and lower half byte (LHB) of memory locations we call MASK and TIME. We don't have the space here to duplicate the detailed explanation of how these variables interact which appeared in Polyphony, and is reprinted in "Friendly Stories About Computers/Synthesizers"; but briefly, both UHBs interact to determine the probability of a dotted note. The LHB of TIME sets the minimum note duration that will occur, while the MASK's LHB controls the range of possible note durations.

These dual half-byte control words are just right for use with a dual half-byte digitizer. From a programming standpoint, all we have to do is read the memory location where the knobs are ($808 on an 8700) and put the result in the memory location where PINK TUNES is going to look for the variable that we're changing.

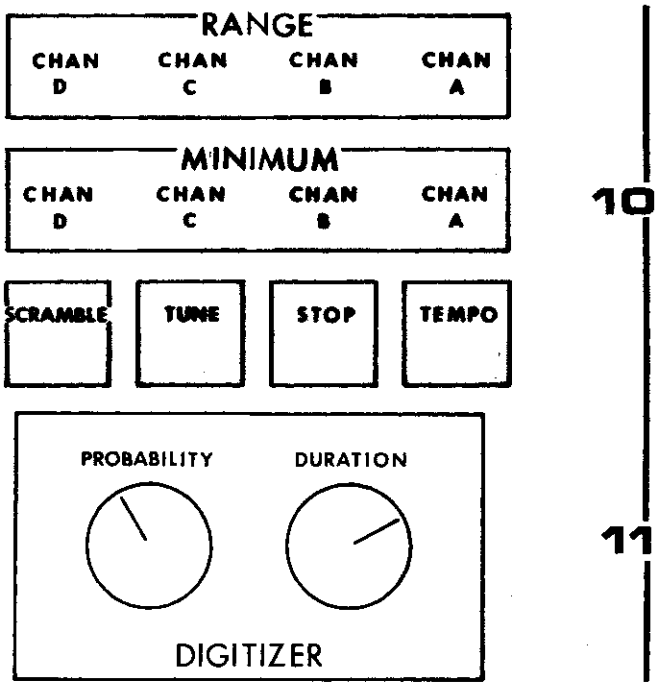
Now, there's a minor difficulty as we have 8 bytes worth of variables (MASK and TIME for each of 4 channels) to set with only two knobs. At some point in the future we'll look at hardware ways to multiplex our digitizer so it can be fed by multiple addressable pots (something like QuASHes in reverse), but for now we're going to actually multiplex the knobs - with software.

Depending on which command pad is being touched, a knob may be controlling minimum note duration on channel A, or the range of durations on channel C, or any of the other possibilities. The program shown in Listing 2 can be added to PINK TUNES to make all this happen. With the software running, the first 12 pads of

```
                0010  :                KNOBS FOR PINK TUNES
                0080  :
                0090      .OR 1066
                0100  :
                0110  :BEFORE WE BEGIN, NOTE THAT THE FOLLOWING SECTION REPLACES PART OF THE
                0120  :EXISTING PINK TUNES PROGRAM.  PRIMARILY, WE CHANGE THE BRANCH DESTINATION
                0130  :FOR THE BNE AT LOCATION $0866 SO THAT THE BRANCH IS TO THE TESTS WHICH
                0140  :FOLLOW RATHER THAN BACK TO THE START OF THE PROGRAM AS IT WAS ORIGINALLY.
                0150  :
1066-  D0 07    0160      BNE TST5     :RATHER THAN TO LP0 AS ORIGINALLY WRITTEN
1068-  20 71 11 0170      JSR SET
106B-  20 2B 1D 0180      JSR NOTE
106E-  00       0190      BRK
                0200  :
                0210  :AS WE JOIN OUR PROGRAM, TEST HAVE ALREADY BEEN MADE TO SEE IF COMMAND
                0220  :FROM KEYBOARD WAS FOR SCRAMBLE, TUNE, OR STOP.  NOW WE ADD TESTS FOR
                0230  :CHANGE TEMPO OR TIME AND MASK PARAMETERS
                0240  :
106F-  C9 0C    0250  TST5 CMP 0C       :IS THERE A COMMAND AT ALL?
1071-  B0 9B    0260      BCS LP0      :NO, JUST GO AHEAD AND BRANCH TO KEEP ON TRUCKIN'
1073-  E9 03    0270      SBC 03       :NORMALIZE COMMAND FOR POINTER USE (CARRY WAS CLEAR)
1075-  08       0280      PHP          :SAVE THE + OR - STATUS OF THE SUBTRACTION FOR LATER
1076-  AA       0290      TAX          :AND TRANSFER THE RESULT TO POINTER, MAY USE
1077-  AD 08 18 0300      LDA DGIT     :GET THE DIGITIZER OUTPUT
107A-  28       0310      PLP          :NOW RECOVER THE + OR - STATUS OF THAT SUBRACTION
107B-  10 06    0320      BPL TST6     :IF THE POINTER IS =>0 BRANCH TO CHANGE MASK OR TIME
107D-  09 F0    0330      ORA 0F0      :TEMPO CHANGE, SET ALL UHB BITS TO 1'S WITH THIS MASK
107F-  85 A9    0340      STA *TMPO    :THEN SAVE RESULT AS TEMPO CHANGE
1081-  D0 0B    0350      BNE LP0      :AND BRANCH ALWAYS TO CONTINUE
1083-  95 0B    0360  TST6 STA *TIME,X :CHANGE TIME OR MASK PARAMETERS
1085-  10 07    0370      BPL LP0      :BRANCH ALWAYS TO KEEP ON
                0380  :
                0390      .EN
```

list2

the 8700's command keyboard take on the responsibilities depicted in figure 10.



10

11

The first three keys on the computer serve the same function they did in the un-altered PINK TUNES, but from there on it's all new. When TEMPO is touched, the knob corresponding to the LHB of the digitizer provides a coarse control of tempo; the other control has no effect. Touching one of the pads $4 - $B causes the selected parameter for the selected channel to be read from the pots. By the way, thinking of the pots as being labeled as shown in figure 11 will help you keep their functions straight in your mind (particularly if you remember that TEMPO is a duration function).

Yep, the knobs are definitely a plus for PINK TUNES. You can really try things out fast without having to shut everything down and scratch your head each time you want to change a channel from quarter to half notes, and so on. Also, the first program is a good example of how to program the knobs when you're setting variables that are organized as 4 bits each, two to the byte.

But there are other ways that the knobs can be programmed. For example, some parameters simply require more resolution than the 16 quantizing levels that 4 bits provide. An obvious answer is to think of the two knobs as both controlling one value, in which case the UHB knob can be thought of as a coarse range control while the LHB knob is fine tuning (our first test program can be thought of as acting this way). We'll look at another way that resolution can be extended in a moment.

In some cases the 16 quantizing levels provided by a single digitizer "channel" is sufficient resolution, but the resulting parameter must have a greater range than 4 bits allow. A brute force method of dealing with this is to use the output of the digitizer as a pointer to a table of parameter values like the code in Listing 3. This program reads a value from the table based on

```
                        !L
                        0010 :                  TABLE LOOK-UP DEMO
                        0020 :
                        0030 :THE DIGITIZER IS READ AND THE LHB IS MASKED OFF WITH AN 'AND'.  THE
                        0040 :RESULT IS PLACED IN THE X REGISTER FOR USE AS A POINTER TO THE TABLE OF
                        0045 :VALUES WHICH OCCUPIES THE MEMORY IMMEDIATELY FOLLOWING THE PROGRAM.
                        0050 :THE VALUE CORRESPONDING TO THE KNOB POSITION IS FETCHED AND SHOWN IN
                        0060 :THE DISPLAYS.
                        0070 :
1000-  AD 00 10         0080 STAR LDA DGIT    :READ THE DIGITIZER
1003-  29 0F            0090      AND 0F      :'AND' WITH MASK (00001111) TO MAKE UHB ZERO
1005-  AA               0100      TAX         :THEN PUT TO X REGISTER TO USE AS POINTER
1006-  B5 0E            0110      LDA *TABL,X :GET THE PARAMETER VALUE FOR THIS KNOB SETTING
1008-  8D 20 18         0120      STA DISP    :SHOW THE PARAMETER VALUE
100B-  4C 00 10         0130      JMP STAR    :THEN LOOP FOR MORE
                        0140 :
                        0150 TABL .HS 0019324A62788E284C5D4E1ECF4FAFE
                        0160 :
                        0170      .EN
```

list 3

the setting of the LHB knob and shows it in the displays. In this case the table is an approximation of 1/4 cycle of a sine wave, but it could be anything.

In some cases the digitizer's output can be used in some way to calculate the parameter value.

One of the difficulties with software multiplexing of the knobs is that unless you're one of those people blessed with eidetic memory you have little way to know what the position of the knob was the last time you set it. In some cases this isn't important, but in others (when you want to smoothly change a parameter from what it is to what you want it to be) it can cause problems. You punch in to change a value and the value immediately jumps to correspond to the current setting of the control. Glitch-ville.

A solution is to use the knob not to set the parameter, but to change it. That may not sound like a big difference, but it is. Using the knob to change the parameter means that when a function is punched in, the current setting of the knob is not important. As the knob is turned, though, the change in its position produces a corresponding change in the parameter. Try running the software in Listing 4.

list 4

```
                        0010 :                  DELTA TUNE DEMO
                        0020 :
                        0030 :AFTER A SHORT DELAY GENERATED BY CALLING THE MUS 1.0 SUBROUTINE LOOK,
                        0040 :WE READ THE COMMAND KEYBOARD BY CALLING THE MONITOR SUBROUTINE DECODE.
                        0050 :ON RETURNING FROM THIS SUBROUTINE THE ACCUMULATOR AND Y REGISTER CONTAIN
                        0060 :THE NUMBER OF THE LOWEST KEY THAT WAS PRESSED ($10 FOR NO KEY).  THE
                        0070 :CARRY FLAG IS CLEARED BY DECODE IF THE KEY WAS TOUCHED THIS SCAN BUT NOT
                        0080 :THE LAST, I.E. IF THE KEY WAS JUST TOUCHED
                        0090 :
1000-  20 4E 1D         0100 DLTA JSR LOOK    :THE CAPACITIVE KEYBOARD REQUIRES A DELAY BETWEEN SCANS
1003-  20 00 1F         0110      JSR DECD    :READ THE COMMAND KEYBOARD
1006-  D0 F8            0120      BNE DLTA    :IF ZERO KEY NOT TOUCHED, LOOP
1008-  AD 00 10         0130      LDA DGIT    :CHANGE COMMAND ASSERTED, SO GET THE DIGITIZER OUTPUT
                        0140 :
                        0150 :NOW THE ACCUMULATOR HAS THE DIGITIZED KNOB POSITION.  WE'RE REALLY ONLY
                        0160 :INTERESTED IN THE LHB, SO WE MAKE THE UHB ZERO WITH AN 'AND'.  IF THE
                        0170 :COMMAND WAS JUST ASSERTED, WE SKIP THE CALCULATION OF CHANGE IN SETTING
                        0180 :AND SIMPLY SAVE THE CURRENT SETTING AS THE STARTING VALUE
                        0190 :
100B-  29 0F            0200      AND 0F      :'AND' WITH MASK TO MAKE UHB ZERO
100D-  90 17            0210      BCC DNX0    :IF COMMAND JUST ASSERTED, SKIP CALCULATING CHANGE
100F-  48               0220      PHA         :SAVE KNOB POSITION ON THE STACK FOR USE LATER
1010-  38               0230      SEC         :PREPARE FOR SUBTRACTION TO FOLLOW
                        0240 :
                        0250 :IT'S TIME TO SEE HOW THE KNOB HAS CHANGED.  CURRENT SETTING IS SUBTRACTED
                        0260 :FROM PREVIOUS SETTING AND THE DIFFERENCE (MAY BE + OR -) IS ADDED TO THE
                        0270 :CURRENT VALUE OF THE PARAMETER.  TESTS ARE MADE TO SEE THAT WE'RE WITHIN
                        0280 :THE ARBITRARY RANGE $00-$3F AND IF OUT OF RANGE THE LIMIT IS SUBSTITUTED
                        0290 :FOR THE CURRENT PARAMETER
                        0300 :
1011-  E5 00            0310      SBC *TEMP   :TEMP IS THE POSITION OF THE KNOB THE LAST TIME THROUGH
1013-  18               0320      CLC         :NOW PREPARE FOR ADDITION
1014-  65 01            0330      ADC *PARM   :ADD THE DIFFERENCE BETWEEN NOW AND LAST TIME TO VALUE
1016-  10 02            0340      BPL DNX1    :IF GREATER THAN ZERO, SKIP THE NEXT INSTRUCTION
1018-  A9 00            0350      LDA 00      :IF WE'RE HERE, WE'RE UNDER-RANGE.  MAKE PARAMETER ZERO
101A-  C9 3F            0360 DNX1 CMP 3F      :ARE WE GREATER THAN THE MAX ALLOWABLE FOR PARAMETER?
101C-  90 02            0370      BCC DNX2    :NO, SO BRANCH TO SKIP NEXT INSTRUCTION
101E-  A9 3F            0380      LDA 3F      :OVER-RANGE, MAKE PARAMETER EQUAL TO MAX LIMIT
1020-  85 01            0390 DNX2 STA *PARM   :SAVE THE NEW VALUE OF THE PARAMETER
1022-  8D 20 18         0400      STA DISP    :AND SHOW IT IN THE DISPLAYS
                        0410 :
                        0420 :NOW WE GET READY FOR THE NEXT PASS BY SAVING THE CURRENT KNOB POSITION
                        0430 :
1025-  68               0440      PLA         :PULL THE DIGITIZER OUTPUT FROM THE STACK
1026-  85 00            0450 DNX0 STA *TEMP   :AND SAVE IT TO DETERMINE CHANGE IN SETTING NEXT TIME
1028-  4C 00 10         0460      JMP DLTA    :THEN JUMP TO START TO CONTINUE
                        0470 :
                        0480      .EN
```

With the code operating, any changes in the setting of the LHB knob are ignored completely until the parameter change is called for by touching the "0" command pad. Then, as the knob is rotated clockwise, the parameter (as shown in the displays) increases. Unlike the other code that we've examined, when the end of control rotation is reached, you can release the command pad, turn the knob fully counterclockwise, touch the pad again and continue increasing the parameter. This technique not only provides smooth control over a value without having to know its current state, it also extends the range of values that can be set with the knob.

The things that we've covered here are not all the possibilities, but hopefully they will get you started in adding variables to your software. It's really hard to beat a knob.

The following are available from Paia Electronics, Inc., 1020 W. Wilshire, Oklahoma City, OK 73116:
** EK-7 Dual Digitizer kit, with PC board and all parts (including pots, knobs, and sockets); $14.95 + $1 postage and handling.
** "Friendly Stories About Computers/Synthesizers", a compendium of past Lab Notes from Polyphony; $5.00 ppd.

NOTES

WWW.PAIA.COM