Practical Circuitry

MICRO DRUMS

PART I

By: Tom Henry

figure 2

Analog design has always fascinated me, and as a consequence I've never really gotten into digital or computer type circuits. However, just recently I dipped into this field and was amazed at the things that even a novice (like myself) can get a computer to do. As my first venture into this area, I came up with a computer controlled drum unit, called "Micro-Drums", which has absolutely revolutionized the manner in which I approach composition. Not being a drummer, my music has always been hampered by a lack of rhythmic expression, but this new circuit has changed all that.

What is Micro-Drums? Quite simply, it is a hardware/software combination which causes the PAIA 8700 microcomputer to think that it's a drummer. Up to eight drums can be polytonically controlled, over the range of an entire song. Nuances, bridges, breaks, lead-ins, even mistakes can be programmed into it so that the unit really drums as if it were a person. Depending on various factors, three to ten minute songs can be programmed with a great amount of depth and variation.

Editing a drum score is quite easy since an hierarchical approach is taken which closely approximates the manner in which you would write a song anyway. Once a score has been worked out, it can be saved to cassette and re-loaded at a later date. Thus, it is possible to create a library of drum scores and this eases the task of creating new songs later on.

But here's the real kicker. By using a special technique with the computer (the $\overline{\text{IRQ}}$, to be discussed later), we can sync the drum unit to click tracks on tape, other rhythm generators, sequencers -- or we can go the other way around! That is, Micro-Drums can be either a master or slave with equal ease. And best of all, this special technique drastically reduces both the hardware and software needed.

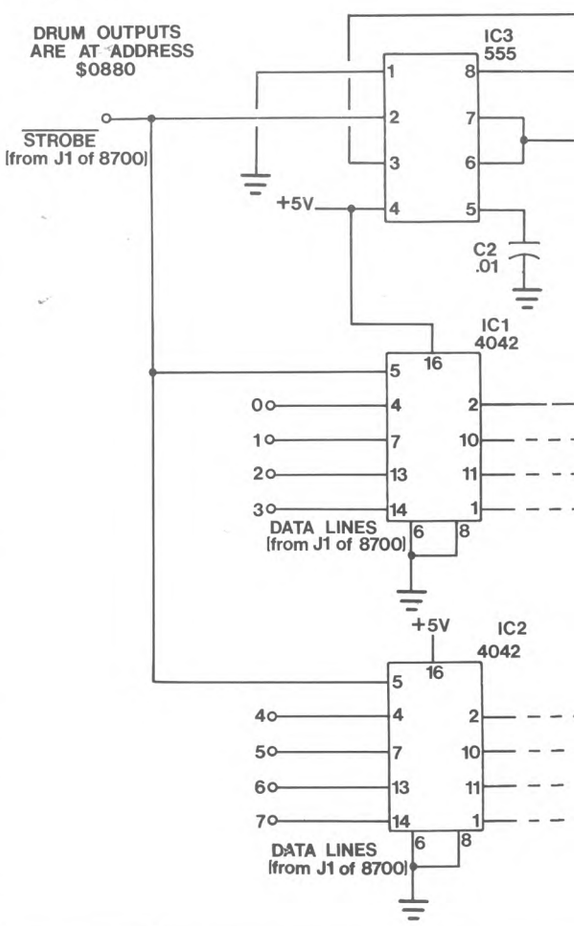
As mentioned, this project is based on the PAIA 8700 microcomputer. However, the same methods will work with just about any other computer using a member of the 6500 family, including the VIC-20, Commodore 64, and PET. As long as you can find one uncommitted address and can get access to the $\overline{\text{IRQ}}$ pin of the CPU chip, you can make it work. By the way, if you do configure this around the 8700 you'll be glad to know that the new hardware in no way interferes with the rest of the computer. You can still use it for your other applications.

Like most computer projects, the basic principle is simple although the explanations get quite long-winded. To keep things orderly, this installment of "Practical Circuitry" details the hardware needed for Micro-Drums, while next time the necessary software and programming instructions are described.

Refer to the schematic in figure 1. Essentially, we set up one address in the 8700's operating system to act as a drum output port. This port is memory mapped, and each bit in it controls a separate drum. Writing a byte into this port triggers the drums corresponding to the various bits. Address $0880 ("$" means that the number is in hexadecimal notation) is chosen for the drum output port, since this leaves the one at $0840 free for other synthesizer applications.

IC1 and IC2 are quad latches; their duty is to store the byte which is written to the drum port. Since a typical "write" operation on the data bus is only several microseconds long, we need these two chips to grab the desired byte and hold on to it.

When a "write" occurs, the $\overline{\text{STROBE}}$ line of the 8700 goes low for a microsecond or two. This line goes to pin 5 of each of the latches, hence causing them to latch the bytes currently on the data bus.



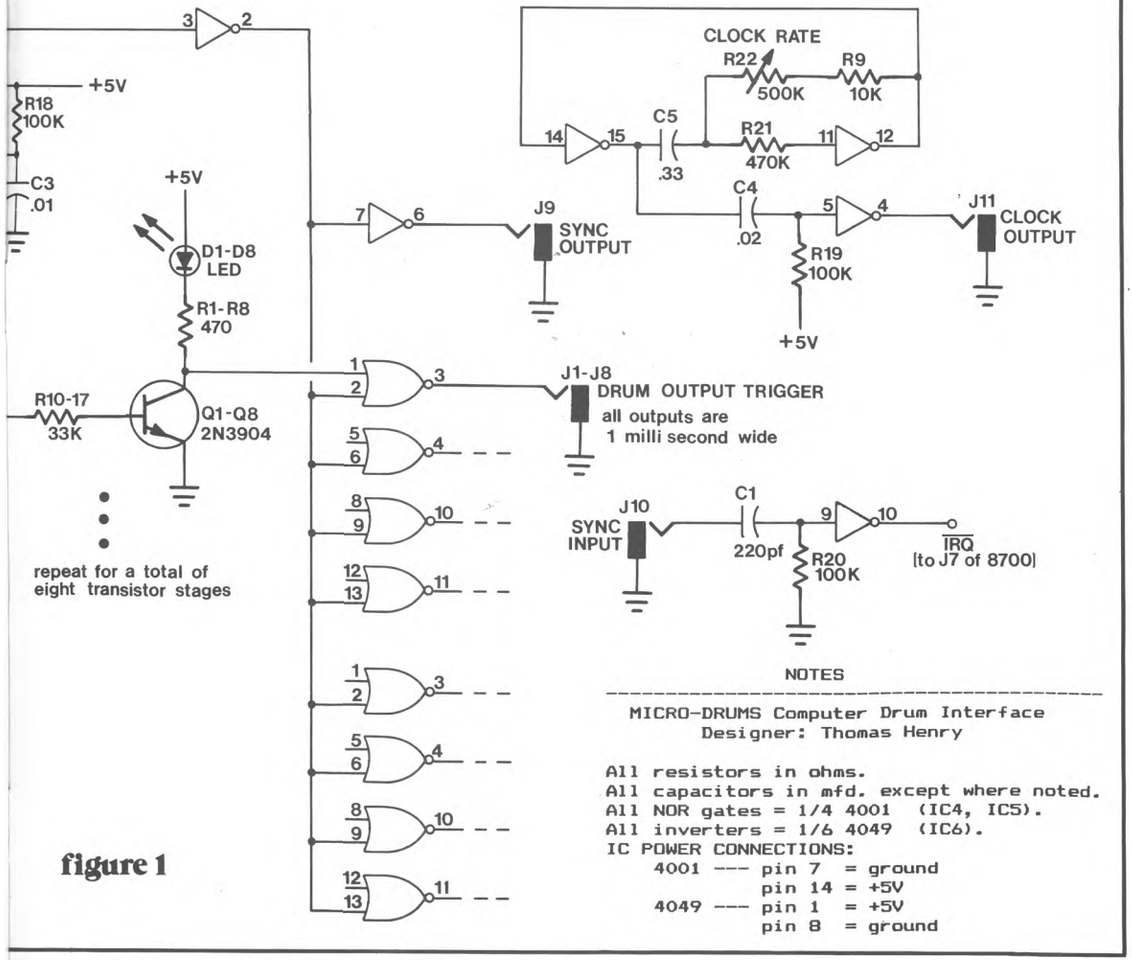


figure 1

NOTES

MICRO-DRUMS Computer Drum Interface
Designer: Thomas Henry

All resistors in ohms.
All capacitors in mfd. except where noted.
All NOR gates = 1/4 4001 (IC4, IC5).
All inverters = 1/6 4049 (IC6).
IC POWER CONNECTIONS:
4001 --- pin 7 = ground
pin 14 = +5V
4049 --- pin 1 = +5V
pin 8 = ground

But the STROBE signal also goes to IC3, which is a 555 timer set up as a one-shot. The output of this one-shot, upon being fired, goes high for a period of 1 millisecond. In effect, we have stretched out the STROBE signal (which is several microseconds long) to a 1 millisecond pulse. You will recognize this figure as being one of the standards we've talked about previously -- it's just the right length of time to fire envelope generators and drum circuits. In point of fact, every output in Micro-Drums generates a +5V, 1 millisecond wide pulse, and thus the circuit is compatible with any of the projects described in "Practical Circuitry".

By the way, notice how everything works out conveniently for the 555 input at pin 2. Normally, you have to provide some input conditioning to this pin, but as it turns out the STROBE signal from the 8700 has the correct polarity and pulse width so that all you have to do is just hook it up directly to pin 2. That ought to refute Murphy's Law!

Now backtrack just a bit to the latch outputs. Each output is buffered by a transistor (Q1 through Q8, one for each bit). Note that the transistor inverts the bit; however, the NOR gate following the transistor inverts the bit again. The net effect is that the bit passes to jacks J1 through J8 exactly as it was written to the drum port. To put it another way, write the number $FF to the drum port and all of the drums will be fired; write a number $00 to the port, and none of them will be fired.

Note that LEDs D1 through D8 turn on according to the number written to the port. These provide an excellent indication of what's going on while you're composing a song.

So, the latched data is sent to the transistors and then to the NOR gates. Now one input of each NOR gate is tied to the stretched out STROBE signal (from IC3 and the inverter). This causes the selected NOR gate to go high for a period of 1 millisecond, and as mentioned above, this is just what envelope generators and drum circuits like to see.

Notice that the stretched out STROBE signal is also available at J9. This signal may be used as a SYNC output and can lock sequencers, rhythm generators, and other computers to the main timing logic. Its pulse width is also 1 millisecond.

This takes care of the drum output support hardware. As you can tell, there really isn't very much to it, and in fact it is quite similar to the other output port at $0840 on the 8700. The rest of the circuitry in figure 1 has nothing to do with the drum output, but instead provides the necessary housekeeping circuitry to round out the complete system.

Let's look at the peripheral circuitry. Since we will be syncing the computer through its IRQ line (interrupt request), we must condition this input somewhat. J10 gives us access to this line. C1, R20, and the inverter form a half-monostable and as such take an input pulse of variable length and transform it into a precise 10 microsecond trigger. This trigger couples to the computer via IRQ, found at J7 on the 8700.

Why should this be a 10 microsecond pulse? The answer to this lies in the nature of the IRQ line in general. When the IRQ line of the 6503 CPU is brought low, the computer will cease whatever it is doing and jump to the service routine pointed at by the vector in location $0FFE and $0FFF. Control is then sent to this service routine, and the instructions found there will be executed until an RTI (return) command is encountered. Control is then sent back to the main program.

Now suppose that the IRQ signal which caused all of this to happen is still low. (In other words, the execution time of the service routine was shorter than the pulse width of the IRQ signal.) What will happen? Just what you would expect; the routine is called again! The upshot is that one IRQ signal caused the service routine to be called twice. We clearly don't need that, so the IRQ pulse is shaved down to 10 microseconds. With the 8700, 10 microseconds corresponds to about 5 program instructions, so as long as the service routine is more than 5 instructions long, all will work well. Incidentally, it should be clear that the IRQ pin responds to "levels", not "edges".

As you will see next time, when we discuss the software aspects of Micro-Drums, this IRQ business is the key to the entire system. Not only does it make master/slave relationships possible, but it also allows use to achieve a remarkable analog to digital conversion for the price of a patch cord! And as mentioned before, both the hardware and software can be drastically simplified.

The remaining three inverters of the 4049 package are put to use as a variable clock. There's nothing clever here since this circuit has been around for years. But one interesting aspect is that C4, R19 and an inverter are set up as another half-monostable. This time the pulse width is made to be millisecond wide (our old standard). R23 sets the clock rate, and with this control the output frequency can be continuously adjusted. Even though the frequency can be changed the pulse width will remain fixed at 1 millisecond.

Just to give you a sneak preview, a patch cord will be used to connect the CLOCK OUTPUT (J11) to the SYNC INPUT (J10), and thus interrupts can be made to occur at an adjustable frequency. This method will be employed to set the tempo of the song.

This just about covers the hardware aspects of Micro-Drums, but perhaps a few words about construction methods should be said. I built this circuit on a prototype circuit board (from Radio Shack), using ordinary hookup wire. If you employ this method, be sure to ponder the layout so that you won't run out of room at a crucial moment! Watch your power supply connections, but since the circuit only uses a +5V supply, this shouldn't provide any great problem. Also (need I say it?), use sockets, since this project employs CMOS integrated circuits which can be damaged by static electricity. Figure two shows the complete parts list for Micro-Drums.

After constructing the circuit, give a great deal of thought on how you will interface it with the computer. I used ribbon cable and headers to complete the connections to J1 and J7 on the 8700. After making the connections I mounted the board to the back of the 8700 computer itself. If you already have this computer, then you will know that it is a double-decked circuit board arrangement. By adding the Micro-Drums card, you will be left with a triple-decked affair.

And now is as good a time as any to mention a modification to the 8700 that you really ought to

think about. I found that with just the bare-bones computer (no Micro-Drums added on), the power supply ran extremely hot. I took some measurements and discovered that the unit was drawing almost 900 mA! This is clearly way too much for the 7805 regulator to handle with such a small heatsink. The culprit, of course, is the RAM -- each chip consumes almost 70 mA. Multiply that by 8 (the number of 2112s in the 8700), and you've got quite a load for the regulator to handle.

In general I like to have at least a 2:1 margin of safety, so I decided to modify the computer accordingly. I simply built another +5V power supply and put the RAM on their own circuit. It's a crazy scheme, I know, but it does work and both power supplies now run considerably cooler. What's more, in the future I will be able to add on extra circuitry since I have a little more juice to play with now.

If what I've said doesn't make any sense to you, then don't perform the modification!!! Your 8700 is a valuable instrument and you won't want to wreck it. On the other hand, if you understand about power supplies, decoupling, and computers -- and if you have a steady hand for cutting PC board traces -- you might want to give this a try. Remember, the RAMs must be completely on their own circuit; it's no good just wiring two power supplies in parallel (unless you get off on rampant destruction of valuable equipment and enjoy fireworks).

After building the Micro-Drums card and performing the modification (if this applies to you), you can then put the thing in a suitable house. figure 2 shows how I did it. This is a home-made wooden box with two sheets of steel for a top and bottom. The 8700 is bolted to the top panel, and the keyboard shines through a square hole cut in the metal. I put some foam rubber around the hole and this keeps dust and moisture out. By the way, the fuse, switch, and line cord are on their own small panel mounted on the back surface of the box.

If you are using the PAIA 8700 power supply, then bring out the 60 Hz signal output to a front panel jack as well. (This is a logic level signal, NOT the line voltage!!!) Since this is a reliable frequency source, it might come in handy for future use.

As you examine the photo fig. 2 you will probably notice some features not described in the article (knobs, connectors, etc.). These have nothing to do with the Micro-Drums interface. For example, there are two D-25 jacks on the right side; one of these is a dummy (for future expansion) and the other is an interface to my keyboard synthesizer. When you build your unit, you might like to plan for the future, too, and leave some extra room for more connectors and whatnot.

Well, that wraps up the Micro-Drums hardware and it's a good thing too, since we're out of space. But come back next time for the concluding episode and see how to implement the software for a complete drum computer. Until then, here's a challenge for you to ponder: how would you create a real time clock using the SYNC input, the 60 Hz output, a patch cord, and a bit of software?

re-view

continued from page 9

front-to-rear as well as left-to-right. This 3-dimensionality makes his heavily-produced electronic pop tunes a listening experience which goes beyond their significance as pop tunes. Other influences might be Zappa and Godley/Creme.

Berlin **Pleasure Victim** (Geffen 2036). I didn't want to like this group -- their music is too trendy and their videos have been a little pretentious. But after having played the record numerous times looking for a weakness to attack, I have to admit it grew on me; Terri Nunn's voice has a cloying innocence and John Crawford's and David Diamond's synthi backing is very professional. It's well defended from sharks like me.

Men at Work **Business as Usual** (Columbia FC37978); **Cargo** (Columbia QC38660). MAW has been called "the Velveeta of pop". However, it's precisely because their strong, well-crafted tunes appeal to even a jaded old reviewer that they're selling so many records. Despite the cheese.

Deckard/Cardwell/Vosh **Sound** (cassette). "Sound" is what it's all about, as this synthesizer trio spins long, introspective pieces full of original synthesizer sounds. Sounds that soothe, sounds that startle, sounds that bounce off the wall and refuse to leave -- always the devotion for The Sound. $4 postpaid from David Vosh, 6300 Goldenrod Court, Upper Marlboro, MD 20772.

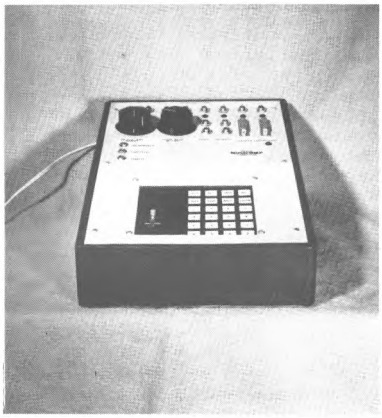
Everfriend **Key Essentials** (cassette). Keyboard artist Bill Rhodes displays his skills, from Keith Emerson-like classical rock to piano fusion jazz to the dramatic "Life and Death of a Star" (reviewed May/June '81 as an EP). There's a couple vocal numbers too -- too bad the vocalist he chose sounds a little tentative. Never mind; the rest is top drawer. Contact Bill at 1 Windemere Rd., Piscataway, NJ 08854.

continued on page 44

Practical Circuitry

MICRO DRUMS

PART II



By: Tom Henry

Last time in "Practical Circuitry" we discussed the hardware needed to implement Micro-Drums, a computer controlled drum unit. In this installment we will consider the software side of things. Since a lot needs to be said, let's jump right in and see how to punch up the program required to get Micro-Drums off and drumming!

Entering the program. Figure 1 shows the complete listing for Micro-Drums software. Since I used the PAIA 8700 computer, the code is written in 6502 assembly language. Those of you who plan to use Micro-Drums with some computer other than the PAIA unit will need to change the appropriate equates (at the start of the program) and may also need to alter the starting address. The source code in Figure 1 is heavily annotated, so you should be able to figure out how it works quite easily.

8700 users can ignore the line numbers, labels, mnemonics and comments if desired, since all that is needed to enter the program is the start address and the required code (under the heading "CODE", in the listing). Refer to your 8700 Computer/Controller manual for help in deciphering an assembler listing if you experience any difficulty.

Follow these steps to enter the program:

(1) Turn on the computer and hit the reset button.

(2) Load location $ED with the byte $1F. This sets the stack to a known condition needed by Micro-Drums.

(3) Get ready to start loading data at $0120, by punching in this address and hitting the DISP key.

(4) Using Figure 1 as a guide, start entering the data. The first few bytes are $20, $34, $0F, A5, etc.

(5) Keep entering data until you hit the last byte in the listing. This is $02 (at location $028B). Use the PCH and PCL keys to confirm that you're at the right place.

(6) You're now ready to save the program to tape. Follow the normal 8700 Cassette Interface protocol for saving a program. The start address in $1020, the end address is $028B, and you can use $01 for an identifier.

If all has gone well, you now have a working copy of the Micro-Drums program. After debugging your work (if needed), make a few backup copies as well. Now, let's see how to use the complete Micro-Drums system.

Using Micro-Drums. To use Micro-Drums, follow these instructions:

(1) Reset the 8700 computer.

(2) Load location $ED with the byte $1F. This sets the stack to a known condition needed by Micro-Drums. Do not forget this step; the program will not load or run correctly if it is left out!

(3) Using the standard loading procedure, load the Micro-Drums software. The start address is $0120, the end address is $028B and the identifier is $01.

(4) Run the program by typing $025F and hitting the RUN key. If everything has gone well, you will hear a long beep. This long beep indicates that you are in the main loop, and the computer is awaiting your instructions.

When in the main loop (as you now are), you have a choice of four commands. They are COARSE EDIT, FINE EDIT, TAPE and PLAY. After any of these commands have been executed, you will always be ushered back to the main loop. Even though the 8700 has limited display capabilities, you can always tell when you are back in the main loop by the long beep. Also, if you hit an invalid key when in the main loop, a long beep will occur. By listening for this beep, you can always tell what's happening at any moment.

Here follows a description of the four main commands. Note that when within the four main commands, there are other minor subcommands possible.

FINE EDIT. The fine edit command defines measures of patterns which will be used as the basis for the entire song. You may define up to eight different patterns, each one up to 32 beats long. Or you may partition this in other ways; for example, four patterns of 64 beats each.

Each pattern is given a number-name. The numbers 0 through 7

are used for this purpose. To start editing a given pattern, when in the main loop type the number-name and then hit the PCL key. (Mnemonic: think of PCL as "low", for lowest level of editing, the pattern.) So to start editing pattern zero, type $00 and then hit the PCL key.

You will now be sent from the main loop to the FINE EDIT routine. The display will show the current beat number, and the drum LEDs will show the drums to be played during that beat. To turn a drum on for the beat, touch one of the keys from 0 to 7. The corresponding LED will light up, and all of the currently selected drums will be played as a test. You can turn a drum off by touching the corresponding drum key again. Thus, the drum keys are like toggle switches and may be used to turn drums either on or off. There is one limitation, though: you may not have a particular beat play all eight drums at once. You can have up to seven drums playing at once, but not all of them. (The software uses the all-eight condition as an end-of-the-pattern marker.)

When you have achieved the selection of drums desired, touch the ENTER key. This will store the selection, then increment the beat pointer. The display will show the next beat number, and you are all set to enter the next selection. Also, when the display increments, the drums currently selected for the new beat will be triggered once. Thus you can single step through a pattern, for trial purposes, just by touching the ENTER key repeatedly. You can back step with the BACK key just as easily. And if you want to hear the current beat several times, touch the DISP key. This will sound the current drum selection.

As mentioned, keys 0 to 7 stand for drums 0 to 7. Key 8 stands for a rest. This clears out the drums selected for that beat. When you are done editing a pattern, touch the 9 key and you will be ushered back to the main loop. Note that even though the pattern may be up to 32 beats long, you are not obligated to use all of the beats. For example, with 5/4 time you might want to only use 10 beats.

COARSE EDIT. After creating some patterns (see above), you

```
00001  0000                ;*************************************
00002  0000                ;*                                   *
00003  0000                ;*            MICRO-DRUMS            *
00004  0000                ;*  MICROCOMPUTER CONTROLLED DRUM UNIT *
00005  0000                ;*                                   *
00006  0000                ;*       (C) 1983  THOMAS HENRY      *
00007  0000                ;*    VERSION 2.1  FEBRUARY 5, 1983  *
00008  0000                ;*                                   *
00009  0000                ;*************************************
00010  0000                ;
00011  0000                ;
00012  0000                IRQVEC = $00          ;IRQ VECTOR.
00013  0000                PATTER = $03          ;PATTERN POINTER BASE.
00014  0000                SELECT = $05          ;DRUM SELECT BIT PATTERNS.
00015  0000                PARAMS = $0D          ;TAPE PARAMETERS.
00016  0000                BEAT   = $14          ;CURRENT BEAT POINTER.
00017  0000                SPOINT = $15          ;CURRENT EVENT SELECTED.
00018  0000                REPEAT = $16          ;CURRENT REPEAT COUNTER.
00019  0000                BUFFER = $F0          ;KEYBOARD BUFFER.
00020  0000                SCORE  = $0280        ;DRUM SCORE AREA.
00021  0000                DISPLA = $0820        ;DISPLAY ADDRESS.
00022  0000                DRUMS  = $0880        ;DRUM OUTPUT ADDRESS.
00023  0000                RELAYS = $0E25        ;TURN ON TAPE RELAYS.
00024  0000                CASS   = $0EAA        ;PERFORM CASSETTE OPERATION.
00025  0000                DECODE = $0F00        ;INPUT A BYTE.
00026  0000                GETKEY = $0F1F        ;GET A BYTE.
00027  0000                BEEP   = $0F22        ;BEEP THE BEEPER.
0002B  0000                LBEEP  = $0F24        ;WITH SETUP, GIVES LONG BEEP.
00029  0000                SHIFT  = $0F34        ;SHIFT BUFFER BY ONE DIGIT.
00030  0000                ;
00031  0000                ;
00032  0000                ;*** MAIN LOOP ***
00033  0000                ;
00034  0000                ;
00036  0000                      * = $0120
00037  0120                ;
0003B  0120  20 34 0F     NUMBER JSR SHIFT      ;SHIFT IN NEW DIGIT.
00039  0123  A5 F0              LDA BUFFER      ;FETCH PACKED ENTRY.
00040  0125  8D 20 08           STA DISPLA      ;THEN UPDATE THE DISPLAY.
00041  0128  4C 31 01           JMP INPUT       ;GO GET NEXT INPUT.
00042  012B  A2 FF       MAIN   LDX #$FF        ;GET READY FOR LONG BEEP.
00043  012D  18                 CLC
00044  012E  20 24 0F           JSR LBEEP       ;DO LONG BEEP.
00045  0131  20 1F 0F     INPUT  JSR GETKEY      ;WAIT FOR KEYSTROKE.
00046  0134  C9 10               CMP #$10        ;IS IT A NUMBER?
00047  0136  90 E8               BCC NUMBER      ;YES, BRANCH BACK AND GET
00048  013B  C9 10       FIND   CMP #$10        ;IS IT 'PLAY'?
00049  013A  F0 0F               BEQ PCMD
00050  013C  C9 14               CMP #$14        ;IS IT 'COARSE'?
00051  013E  F0 52               BEQ COARSE
00052  0140  C9 15               CMP #$15        ;IS IT 'FINE'?
00053  0142  F0 0A               BEQ FINE
00054  0144  C9 16               CMP #$16        ;IS IT 'TAPE'?
00055  0146  D0 E3               BNE MAIN        ;RAN OUT OF COMMANDS.
00056  0148  4C 37 02           JMP TAPE
00057  014B  4C E3 01     PCMD   JMP PLAY
0005B  014E                ;
00059  014E                ;
00060  014E                ;*** FINE EDIT COMMAND ***
00061  014E                ;
00062  014E                ;
00063  014E  A5 F0        FINE   LDA BUFFER      ;GET PATTERN NUMBER.
00064  0150  20 57 02           JSR OFFSET      ;GET PATTERN OFFSET.
00065  0153  A0 00               LDY #$00        ;ZERO OUT THE BEAT POINTER.
00066  0155  C8           SHOWIT INY
00067  0156  88           BACKUP DEY
00068  0157  8C 20 08           STY DISPLA      ;DISPLAY IT.
00069  015A  B1 03               LDA (PATTER),Y  ;GET SELECTED BEAT,
00070  015C  8D B0 08           STA DRUMS       ;AND PLAY IT.
00071  015F  20 4F 02     FEDIT  JSR FETCH       ;GET EDIT KEYSTROKE.
00072  0162  C9 12               CMP #$12        ;IS IT A 'BACK'?
00073  0164  F0 F0               BEQ BACKUP      ;YES, BACKSPACE ONCE.
00074  0166  C9 11               CMP #$11        ;IS IT A 'DISP'?
00075  0168  F0 EB               BEQ SHOWIT      ;YES, PLAY CURRENT BEAT.
00076  016A  C9 0A               CMP #$0A        ;IS IT A DRUM NUMBER (0-9)?
00077  016C  90 08               BCC DENTER      ;YES, GO ENTER DRUM BEAT.
00078  016E  C9 13               CMP #$13        ;IS IT AN 'ENTER'?
00079  0170  D0 ED               BNE FEDIT       ;NO, RAN OUT OF COMMANDS.
00080  0172  C8                 INY             ;YES, ADVANCE TO NEXT BEAT.
00081  0173  4C 55 01           JMP SHOWIT
00082  0176  C9 09        DENTER CMP #$09        ;#$09 MEANS END OF PATTERN.
00083  0178  D0 06               BNE NEXT1
00084  017A  A9 FF               LDA #$FF        ;END OF PATTERN MARKER.
00085  017C  91 03               STA (PATTER),Y
00086  017E  D0 AB               BNE MAIN        ;BRANCH ALWAYS.
00087  0180  C9 08        NEXT1  CMP #$08        ;#$08 MEANS 'REST'
00088  0182  D0 04               BNE NEXT2
00089  0184  A9 00               LDA #$00
00090  0186  F0 05               BEQ STORE       ;BRANCH ALWAYS.
00091  0188  AA           NEXT2  TAX             ;INDEX INTO BIT PATTERN.
00092  0189  B5 05               LDA SELECT,X    ;GET PROPER BIT PATTERN.
00093  018B  51 03               EOR (PATTER),Y  ;ADD IN NEW BEAT.
00094  018D  91 03        STORE  STA (PATTER),Y  ;AND SAVE IT.
00095  018F  4C 55 01           JMP SHOWIT      ;SOUND THE DRUM BEAT.
```

```
00096  0192                    ;
00097  0192                    ;
00098  0192                    ;*** COARSE EDIT COMMAND ***
00099  0192                    ;
00100  0192                    ;
00101  0192  A6 F0             COARSE LDX BUFFER       ;GET DESIRED EVENT NUMBER.
00102  0194  86 15                    STX SPOINT       ;STORE AT CURRENT EVENT.
00103  0196  A6 15             REVEAL LDX SPOINT
00104  0198  BD B0 02                 LDA SCORE,X      ;GET CONTENTS OF EVENT.
00105  019B  85 F0                    STA BUFFER       ;PUT IN BUFFER AND
00106  019D  BD 20 08          VIEW   STA DISPLA       ;SHOW IT TOO.
00107  01A0  20 1F 0F          LOOP   JSR GETKEY       ;GET KEYSTROKE.
00108  01A3  C9 10                    CMP #$10         ;CHECK FOR NUMBER.
00109  01A5  B0 08                    BCS NONUM        ;NOT A NUMBER, BRANCH.
00110  01A7  20 34 0F                 JSR SHIFT        ;SHIFT IN NEW DIGIT.
00111  01AA  A5 F0                    LDA BUFFER       ;FETCH PACKED ENTRY.
00112  01AC  4C 9D 01                 JMP VIEW         ;AND UPDATE DISPLAY.
00113  01AF  C9 13             NONUM  CMP #$13         ;IS IT AN 'ENTER'?
00114  01B1  D0 0C                    BNE NEXT3        ;NO, GO ON.
00115  01B3  A6 15                    LDX SPOINT       ;RE-GET EVENT NUMBER.
00116  01B5  A5 F0                    LDA BUFFER       ;FETCH INPUT NUMBER.
00117  01B7  9D B0 02                 STA SCORE,X      ;STORE IN SCORE.
00118  01BA  E6 15                    INC SPOINT       ;UPDATE EVENT NUMBER.
00119  01BC  4C 96 01                 JMP REVEAL       ;UPDATE DISPLAY.
00120  01BF  C9 12             NEXT3  CMP #$12         ;IS IT A BACKSPACE?
00121  01C1  D0 05                    BNE NEXT4        ;NO, BRANCH ON.
00122  01C3  C6 15                    DEC SPOINT       ;DECREMENT EVENT COUNTER.
00123  01C5  4C 96 01                 JMP REVEAL       ;SHOW CONTENTS OF EVENT.
00124  01C8  C9 14             NEXT4  CMP #$14         ;IS IT A 'PCH'?
00125  01CA  D0 05                    BNE NEXT5        ;NO, BRANCH ON.
00126  01CC  A5 15                    LDA SPOINT       ;GET CURRENT EVENT NUMBER.
00127  01CE  4C 9D 01                 JMP VIEW         ;AND SHOW IT.
00128  01D1  C9 11             NEXT5  CMP #$11         ;IS IT A 'DISP'?
00129  01D3  F0 C1                    BEQ REVEAL       ;IF SO, SHOW CONTENTS.
00130  01D5  C9 17                    CMP #$17         ;'REL' STANDS FOR ALL DONE.
00131  01D7  D0 C7                    BNE LOOP         ;RAN OUT OF COMMANDS.
00132  01D9  A6 15                    LDX SPOINT       ;RE-GET EVENT NUMBER.
00133  01DB  A9 00                    LDA #$00         ;END OF SCORE MARKER.
00134  01DD  9D B0 02                 STA SCORE,X
00135  01E0  4C 2B 01                 JMP MAIN         ;RETURN TO MAIN LOOP.
00136  01E3                    ;
00137  01E3                    ;
00138  01E3                    ;*** 'PLAY' COMMAND ENTRY ***
00139  01E3                    ;
00140  01E3                    ;
00141  01E3  A9 00             PLAY   LDA #$00         ;ZERO OUT REPEAT AND
00142  01E5  85 16                    STA REPEAT       ;SCORE POINTER.
00143  01E7  A9 FF                    LDA #$FF
00144  01E9  85 15                    STA SPOINT
00145  01EB  58                       CLI              ;PREPARE FOR IRQ.
00146  01EC  C9 FF             TIGHT  CMP #$FF         ;#$FF MEANS KEEP PLAYING.
00147  01EE  F0 FC                    BEQ TIGHT        ;STAY IN TIGHT LOOP.
00148  01F0  4C 2B 01                 JMP MAIN         ;ABORT 'PLAY' NOW.
00149  01F3                    ;
00150  01F3                    ;
00151  01F3  20 00 0F          IRQRTN JSR DECODE       ;SEE IF ZERO KEY IS PUSHED.
00152  01F6  C9 00                    CMP #$00
00153  01F8  D0 04                    BNE PLAMOR       ;IT ISN'T, SO PLAY MORE.
00154  01FA  28                FINISH PLP              ;SET INTERRUPT FLAG
00155  01FB  78                       SEI              ;SO NO MORE OCCUR.
00156  01FC  08                       PHP
00157  01FD  40                RETURN RTI
00158  01FE                    ;
00159  01FE                    ;
00160  01FE  A5 16             PLAMOR LDA REPEAT       ;REPEATED OLD PATTERN ENOUGH?
00161  0200  D0 19                    BNE MORE         ;NO, KEEP GOING WITH OLD ONE.
00162  0202  E6 15                    INC SPOINT       ;YES, UPDATE SCORE POINTER.
00163  0204  A6 15                    LDX SPOINT
00164  0206  BD B0 02                 LDA SCORE,X      ;GET REPEAT TIME DATA.
00165  0209  F0 EF                    BEQ FINISH       ;DONE PLAYING WHOLE SCORE.
00166  020B  85 16                    STA REPEAT       ;CONTAINS NUMBER OF REPEATS.
00167  020D  E6 15                    INC SPOINT       ;UPDATE SCORE POINTER.
00168  020F  A6 15                    LDX SPOINT
00169  0211  BD B0 02                 LDA SCORE,X      ;GET PATTERN NAME DATA.
00170  0214  20 57 02                 JSR OFFSET       ;GET PATTERN ADDRESS OFFSET.
00171  0217  A9 00                    LDA #$00
00172  0219  85 14                    STA BEAT         ;ZERO OUT BEAT POINTER.
00173  021B  A4 14             MORE   LDY BEAT         ;Y INDEXES TO PROPER BEAT.
00174  021D  B1 03                    LDA (PATTER),Y   ;GET OUTPUT DATA.
00175  021F  C9 FF                    CMP #$FF         ;END OF PATTERN?
00176  0221  D0 08                    BNE OKAY         ;NO, GO PLAY THE BEAT.
00177  0223  C6 16                    DEC REPEAT       ;DECREMENT REPEAT TIME.
00178  0225  A9 00                    LDA #$00         ;YES, RESET BEAT COUNTER.
00179  0227  85 14                    STA BEAT         ;THEN TRY AGAIN.
00180  0229  F0 C8                    BEQ IRQRTN       ;BRANCH ALWAYS.
00181  022B  8D 80 08          OKAY   STA DRUMS
00182  022E  8C 20 08                 STY DISPLA
00183  0231  E6 14                    INC BEAT         ;UPDATE BEAT POINTER.
00184  0233  A9 FF                    LDA #$FF
00185  0235  D0 C6                    BNE RETURN       ;BRANCH ALWAYS.
00186  0237                    ;
00187  0237                    ;
00188  0237                    ;*** 'LOAD' AND 'SAVE' COMMAND ***
00189  0237                    ;
```

will then string them together in various arrangements to form the complete song. This is COARSE editing. You will create a score by entering some events; each event consists of two entries. The first entry is the number of times you wish a pattern to repeat, and the second entry is the number-name of the pattern which is to be repeated. There is room for 64 events total. This will allow songs up to fifteen minutes long to be programmed! To get into the COARSE EDIT mode from the main loop, type the number of the event you wish to start at (usually a $00) and then the PCH key. (Mnemonic: think of PCH as "high", the highest level of editing.)

The display will now show the contents of the current event. To enter a new event, type the desired number and hit the ENTER key. The event will be recorded, and the score pointer is incremented once. For example, starting at event zero, to get sixteen repeats of one, type $10, ENTER, $01, ENTER. Note that all numbers are in hexadecimal and that each entry must be followed by an ENTER.

You can backspace through a score with the BACK key. Also, to see the current event number, touch the PCH key at any time. To see the contents of the event, type DISP. Using the keys just mentioned, you can step through an entire score in a matter of minutes and change or update it as needed.

To finish off a score, touch the REL key. This puts in an end of score marker and returns you to the main loop. A long beep will occur.

PLAY. Playing a score is easy. First make sure that the SYNC INPUT jack has some source of triggers. You may sync the drum score off of the internal variable clock, an external clock, keyboard triggers, sequencer triggers, or click tracks from a tape deck. The input pulses should be +5V in magnitude. Note that Micro-Drums' internal variable clock meets this need and is perhaps the easiest to use. In addition it allows easy adjustment of the tempo: just dial in the desired speed. This may not seem like much, but consider what we've just done: a potentiometer controls the tempo, continuously, without the intervention of an analog to digital con-

verter. How's that for saving money and keeping things simple!

After providing some source of sync pulses, you may start playing the score simply by touching the RUN key. When the song is finished, you will be sent back to the main loop and a long beep will occur. You can also abort a song while it is playing by touching the 0 key. Once again, you will return to the main loop.

As you can see, the SYNC INPUT (alias the IRQ) is the key to the power of Micro-Drums. Any circuit which can put out a series of pulses can cause Micro-Drums to step through the song, beat after beat. You are not constrained to meet this or that condition, and the circuitry is perfectly general. Simply send the computer some pulses and the song commences! And don't forget the SYNC OUTPUT jack either. You can cause some other circuit (like a sequencer) to follow Micro-Drums just as easily, so Micro-Drums can thus play the role of master or slave with equal ease.

TAPE. You can save or load scores using this command. To save a score, start the recorder going in the record mode, type $DD and touch the TAPE key. The computer will do the rest; there is no need to enter any addresses, etc.

Loading a score is just as easy. Start the recorder going in the play mode, type $11 and hit the TAPE key. The score will be loaded.

At the end of any tape operation you will be sent back to the main loop, and a long beep will indicate this fact. Note that the load and save options affect the entire score and pattern memory, so don't be alarmed if the operation takes up to a minute or so. If you experience any trouble, refer to the 8700 Cassette Interface manual and review how to set volume levels and so on.

```
00190  0237                    ;
00191  0237  A2 07      TAPE    LDX #$07        ;PREPARE TAPE PARAMETERS.
00192  0239  B5 0C      SETFIL  LDA PARAMS-1,X  ;GET PARAMETERS.
00193  023B  95 F0              STA BUFFER,X    ;AND STUFF IN PLACE.
00194  023D  CA                 DEX
00195  023E  D0 F9              BNE SETFIL      ;KEEP STUFFING IF NEEDED.
00196  0240  A5 F0              LDA BUFFER      ;GET LOAD/SAVE TOKEN.
00197  0242  20 25 0E           JSR RELAYS      ;TURN ON RELAYS.
00198  0245  20 AA 0E           JSR CASS        ;PERFORM LOAD OR SAVE.
00199  0248  18                 CLC
00200  0249  20 22 0F           JSR BEEP        ;TURN OFF RELAYS AND BEEP.
00201  024C  4C 2B 01           JMP MAIN        ;ALL DONE!
00202  024F                    ;
00203  024F                    ;
00204  024F  84 14      FETCH   STY BEAT        ;GET A KEY, BUT SAVE
00205  0251  20 1F 0F           JSR GETKEY      ;CURRENT Y-REGISTER.
00206  0254  A4 14              LDY BEAT
00207  0256  60                 RTS
00208  0257                    ;
00209  0257                    ;
00210  0257  0A        OFFSET  ASL A           ;FIND OFFSET BY
00211  0258  0A                 ASL A          ;MULTIPLYING ACCUMULATOR
00212  0259  0A                 ASL A          ;BY SIXTEEN.
00213  025A  0A                 ASL A
00214  025B  0A                 ASL A
00215  025C  85 03              STA PATTER     ;OFFSET ADDRESS IS HERE.
00216  025E  60                 RTS
00217  025F                    ;
00218  025F                    ;
00219  025F                    ;*** INITIALIZATION ROUTINE ***
00220  025F                    ;
00221  025F                    ;
00222  025F  78                 SEI
00223  0260  A2 00              LDX #$00
00224  0262  BD 78 02   MOVE    LDA DATA,X      ;GET DATA BYTE.
00225  0265  95 00              STA IRQVEC,X    ;STUFF IT INTO 0-PAGE.
00226  0267  E8                 INX
00227  0268  E0 14              CPX #$14        ;NUMBER OF BYTES+1.
00228  026A  D0 F6              BNE MOVE
00229  026C  A0 00              LDY #$00        ;CLEAR PATTERN AREA.
00230  026E  A9 00              LDA #$00
00231  0270  91 03      CLEAR   STA (PATTER),Y
00232  0272  88                 DEY
00233  0273  D0 FB              BNE CLEAR
00234  0275  4C 2B 01           JMP MAIN        ;GO START UP MICRO-DRUMS.
00235  0278                    ;
00236  0278                    ;
00237  0278                    ;*** DATA AND ADDRESS TABLES ***
00238  0278                    ;
00239  0278                    ;
00240  0278  4C          DATA    .BYTE $4C       ;OPCODE FOR 'JMP'.
00241  0279  F3 01              .WORD IRQRTN    ;START OF IRQ ROUTINE.
00242  027B  00 03              .WORD $0300     ;PATTERN BASE ADDRESS.
00243  027D  01                 .BYTE $01, $02  ;DRUM SELECT BIT PATTERNS.
00243  027E  02
00244  027F  04                 .BYTE $04, $08
00244  0280  08
00245  0281  10                 .BYTE $10, $20
00245  0282  20
00246  0283  40                 .BYTE $40, $80
00246  0284  80
00247  0285  00                 .BYTE $00       ;FILE PARAMETER.
00248  0286  FF 03              .WORD $03FF     ;TAPE END ADDRESS.
00249  0288  80 02              .WORD $0280     ;TAPE START ADDRESS.
00250  028A  80 02              .WORD $0280     ;TAPE POINTER.
00251  028C                     .END

ERRORS = 00000

END OF ASSEMBLY
```

The future. Well, that just about wraps up how to use Micro-Drums. Of course, all we have done here is talk about the mechanics of using the unit; it's up to you to think about the musical side of things. For example, if you know that a particular pattern is to contain both eighth notes and triplets, then you will need to divide the pattern into groups of twenty-four (three times eight). This is just one example of one of the musical considerations that must be taken into account with Micro-Drums. However, you will find that the more you play with Micro-Drums, the better you will become at visualizing what needs to be done.

Well, we've run out of room and need to start planning other projects. But in the meanwhile, as you play with the unit, think about how you would implement sequencer interfaces with Micro-Drums. The procedure is actually quite simple due to the "magical" way in which the SYNC INPUT works. Then consider synchro-sonic recording; how would you do this with Micro-Drums? Once again, the basic principle is quite simple. Think about these things and perhaps later on in the pages of "Practical Circuitry" we can compare notes.