an expandable control system for computer assisted synthesizer

CHAMELEON 0.25

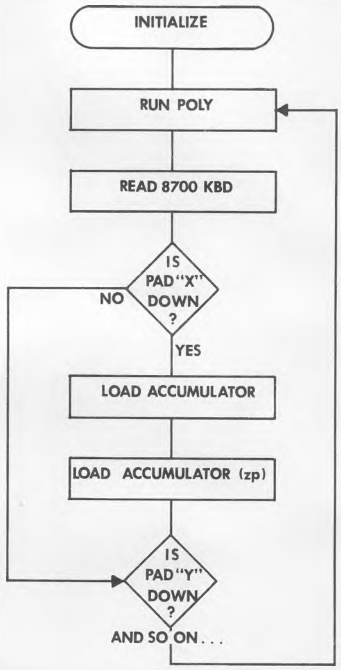
BY JON BALLERAS

There's little telling into whose hands digital music gear is falling and how it's being used. Some users running digital control type systems undoubtedly have been programming for years and are contentedly typing in their own intricate software, filled with stacks happily being pushed and pulled, indexes that know just where they're going and what to do when they get there, subroutines nesting comfortably inside subrouting, and God knows what else. Others, like me, soldered up an 8700 without knowing a bit from byte, much less being able to add $2 and $2 on the machine. Worse, users like me can easily develop a propensity toward software dependency--we become listing junkies anxiously waiting the next hit of code from Simonton or some other kindly programming wiz. Not an especially comfortable situation, especially when you realize that your high technology system doesn't do what you want it to do and, worse, Polyphony may never print a program that makes it run exactly the way you want.

All this was driven home to me a few months ago when I seriously considered playing gigs on my machine. Even though I'd done a good deal of normalizing, it became evident that I wouldn't get the same effects on a job as I did at home when leisurely laying down track by track of tape. For example, selecting glide for one channel meant shutting down the machine, remembering the transpose location for that channel, calling it up, writing in the correct code, touching ENTER, and remembering to start MUS 1 up at the correct location: eleven keystrokes in all. Switching from 1 to 2 or 4 voices meant going through an analagous process, with lots of room for error. Granted, these aren't terribly complicated procedures, but on a job there's already enough to think about: the charts, wondering if the bass player will catch the cue for the next section, what's going on in the audience. With all this happening it's nice to reserve some mental energy for just getting into the music and maybe tweaking a filter or resetting an envelope somewhere along the way.

At this point necessity plunged me into programming. My goal was to write some code that would allow on the fly control of the number of voices POLY was running and of glide and transpose values for four discrete synthesizer channels (each with hardware ADSR's). After a month of performing some simple programming exercises--reading and testing data from the command keyboard, writing a preselected value to a designated zero page location, and lots of branching -- CHAMELEON 0.25, the program at the end of this article, materialized. With one minor glitch, it does exactly what I wanted. More interestingly, it can be expanded to control other MUS 1 functions. Additionally, the body of the program looks to be a useful subrouting for programs that benefit from fast voice switching and changes in TTBL, like the "SPLITZ" portion of Bob Yannes' SHAZAM, and ECHO when run with its first preset. Experienced programmers will quickly note that CHAMELEON is far from perfect. It is extremely redundant and could easily be shortened considerably by using indexing techniques. Nevertheless, the program does run and has led me into more sophisticated coding projects. Most importantly, it has solved some vexing real time control problems that, apparently, no one else was about to take on. Programming autonomy!

The bulk of CHAMELEON consists of a long series of tests (CMP's) of the data put out by DECODE, the keyboard reading subroutine of Piebug Monitor. In the listing, this coding begins at ADDR $228. If the keypad being tested is down, the computer is instructed to write a preset value to a designated zero page location. If the keypad isn't down, the computer takes a branch (BNE) to the next CMP, and a similar routine is followed. Flowcharted, an abbreviated outline of CHAMELEON looks something like this:



some finer (yet useful) points

As the above flow chart implies, the bulk of my code is a kind of appendage to the sequence in which MUS 1 normally calls up its subroutines. As both John Simonton and Bob Yannes have pointed out in these pages, MUS 1 is not at all a monolithic chunk of programming which can only be called up at one location to do only one thing. Instead, this prom has a good number of useful entry points which you can call up when you need them with a simple JSR. If you'll look over the OPTION portion of the MUS 1 listing (beginning at ADDR $DOO) you'll find that CHAMELEON calls up almost exactly the same sequence of subroutines. But my code "opens up" MUS 1, allowing the program to continue after DECODE and perform as many KBD tests as needed. SHAZAM and ECHO are two other programs that freely enter and exit MUS 1. Studying these listings was a key to setting up CHAMELEON.

To give you a more specific idea of MUS 1's flexibility, notice that my listing contains a truncated initialization routine beginning at ADDR $205. As I experimented with each section of CHAMELEON, I found that downshifting from four voices to one, for example, didn't work. Instead of writing to QUASH channel 1 only, POLY got confused and assigned notes to all four channels. While pressing CLEAR wiped out those notes that were evidently floating around in KTBL and NTBL, TTBL was also cleared, removing any glide or transpose value the program had loaded there. Deep thought and lucky guessing led to my writing a second initilization routine, one that cleaned up those notes bouncing around in KTBL but left TTBL untouched. This rewritten version of MUS 1's analagous routine occupies ADDR's $205 to $20E. The point is that if part of MUS 1 doesn't do exactly what you need, you can always write around it. And there's nothing wrong with borrowing liberally from this prom's code, or from any other code, for that matter. If part of a program solves your problem, use it!

making it run

You'll note that CHAMELEON is written on Page 2 of memory. I started it here for several reasons: to avoid the stack on Page 1, to avoid colliding with the MUS 1 tables and other variables on zero page, and to permit the easy addition of other preset commands as they occured to me. As it happens, the basic listing can be squeezed into zero page without immediate problems, although expansion is limited. So if you're running with ½ K of RAM, you can enter CHAMELEON on zero page by changing the address of the JMP at $2A0 from OF 02 to OF OO. If for some reason you want to enter this program on Page 1, you'll have to keep the stack under control by adding this code at the beginning of the program:

```
0100    A2 FF    LDX # $FF     ;PUSH STACK
0102    9A       TXS           ;OUT OF WAY
```

The JMP at the program's end would now become 12 01. Don't forget to set both the user's stack pointer ($OFE) and the monitor pointer ($OED) to $FF before running.

Wherever you decide to locate the program, operating procedures remain the same: type in the code, saving it on tape as insurance against a bad byte blow up. Most importantly, be certain to enter the usual MUS 1 variables, starting at ADDR $OE8 ($40/$20/$01 will do the job.) When you first run the program, the displays will read $00. The 8700's KBD is now "intelligent", and the functions of all active keypads are defined as follows:

Start the program at $200. As you touch a keypad on the second rank of the keyboard, the displays will indicate the number of voices you're currently operating with. The glide/transpose channels are set up discretely, i.e., touching TRANS 2 doubles the control voltage of Channel 2 only; touching GLIDE 3 selects glide for the third QUASH channel only, and so on. Note, though, that all glide and transpose channels are cleared in one block by tcuching CLEAR. Shifting up form 1 to 2, 3, or 4 channels presents no problem, but downshifting from three or four channels can be a touch noisy if the ADSR's for these channels are still in their sustain or release cycles. As I've noted, CHAMELEON was written for a machine running with one QUASH and hardware ADSR's for each channel. If you're still running STG's, computer control of the transpose table is still possible. Just go through the listing beginning at ADDR $26E and change all STA zero page addresses designated $CE to $CD. Addresses $CD become $CB, and $CC's become $CA's. Making these changes will line up the transpose table with the CV channels of one or two STG driven QUASH. So, for absolutely quite downshifting, downshift from full polyphony only when your synthesizer is already quiet. (One deep breath after your fingers are off the AGO should do it.)

going further

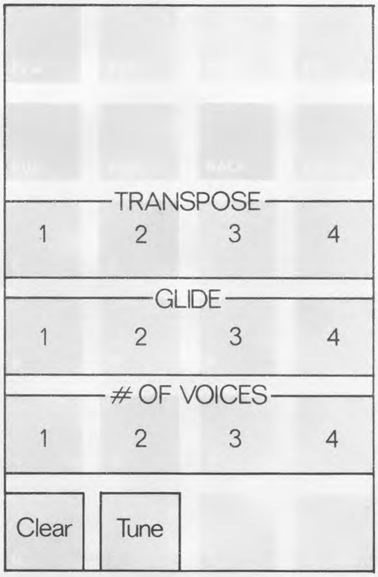
Although CHAMELEON is hardly a tight, finished program, it does have the advantage of being comfortably open ended. Once you've caught on to its basic routine of: test KBD/ branch/ write to zero page, expanding the program or rededicating the 8700's control pads to different functions isn't difficult and, it seems to me, could turn into a useful set of exercises for any beginning programmer. A control pad, for instance, could be assigned to write a 1 (or 2) octave transpose to all channels, or to write a simultaneous glide and transpose to designated channels. Even more challenging and useful would be setting up some pads as STG envelope presets. By expanding CHAMELEON's treadmill styled routine of test/branch/write, some code to select a preset STG envelope ($10/$04/$20/$10/$3F) comes out like this:

```
02XX    C9 XX    STGT 1    CMP # $XX     ;IS PAD XX DOWN?
        DO 14              BNE STGT 2    ;NO, BRANCH TO NEXT TEST
        A9 10              LDA # $10     ;YES, PREP ATCK
        85 BA              STA $BA       ;STORE ATCK
        A9 04              LDA # $04     ;PREP DCY
        85 BB              STA $BB       ;STORE DCY
        A9 20              LDA # $20     ;PREP SUST
        85 BC              STA $BC       ;STORE SUST
        A9 01              LDA # $01     ;PREP RLS
        85 BD              STA $ BD      ;STORE RLS
        A9 3F              LDA # $3F     ;PREP PEAK
        85 BE              STA $BE       ;STORE PEAK
```

Conceivably half a dozen of your most useful STG envelopes could be stored in the program and called up when you need them.

afterwords

While I certainly hope CHAMELEON will prove helpful in solving some of your computer-assisted synthesizer control problems, I'll count this article even more of a success if it encourages those of you who are "software dependent" to start fooling around with your own code. We all known there are stacks (!) of books on programming on the market (some, like



The First Book of Kim and William Barden's How to Program Microcomputers actually talk about the 6502 in enlightening ways). After reading material like this the MOS Manual begins to make some sense. However, it's been my experience that you learn programming by doing programming. Adding $2 plus $2 is a start. As you go on, more simple exercises will occur to you, and simple exercises have a way of germinating into full fledged programs, as I hope CHAMELEON shows. The sample programs in the 8700 (or whatever computer you have) manuals are ripe for study and flowcharting -- particularly helpful activities since you'll become familiar with the actual protocols of Paia equipment (or your own system). Ultimately, each programmer has to find his or her way of coming to terms with their machine. It's not that hard, and you may well be surprised at what you learn. Happy coding!

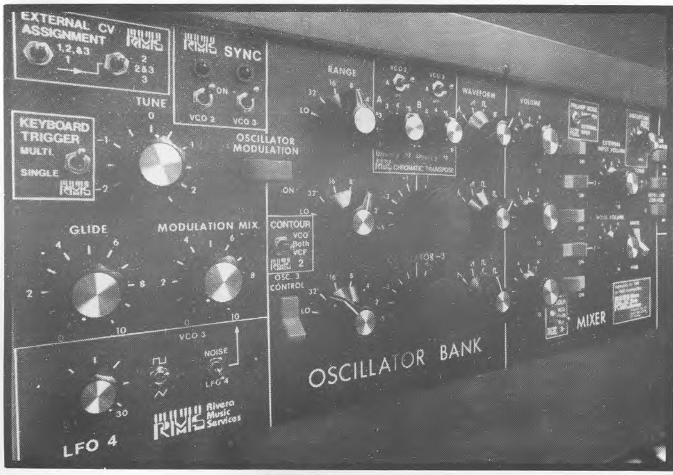
CHAMELEON 0.25

Control System For 8700 Based Synthesizers

By Jon Balleras

July 1979

```
ADDR  CODE      LABEL    INSTRUCTION     COMMENT
0200  20 21 OD  INIT 0   JSR INIT        CLEAR KTBL,NTBL,TTBL
0203  90 0A              BCC POLY        BRANCH ALWAYS POLY
0205  A9 00     INIT 1   LDA # $00       PREP TO ZERO
0207  A2 10              LDX # $10       SET POINTER
0209  95 CF     Z BUF    LDA KTBL, X     ZERO BUFFER
020B  CA                 DEX             POINT TO NEXT
020C  DO FB              BNE Z BUF       LOOP IF NOT DONE
020E  60                 RTS             BACK TO VOXTS
020F  20 71 OD  POLY     JSR POLY        ASSIGN NOTES
0212  20 C3 OD           JSR TRNGN       CALL STG'S, IF ON
0215  20 2B OD           JSR NOTE        PLAY NOTES
0218  20 00 OF           JSR DECD        READ 8700 KBD
021B  C9 01              CMP # $01       IS TUNE DOWN?
021D  90 E1              BCC INIT 0      NO,IT'S LESS,GO CLEAR
021F  D0 07              BNE VOXT 1      GO TO VOXT 1
0221  A0 5C              LDY # $5C       PREP FOR TUNE
0223  20 52 OD           JSR FILL        PUT NOTE IN ALL VOX
0226  F0 E7              BEQ POLY        PLAY TUNING NOTE
0228  C9 04     VOXT 1   CMP # $04       IS $04 DOWN?
022A  D0 0A              BNE VOXT 2      NO, GO TO VOXT 2
022C  A9 01              LDA # $01       YES, PREP 1 VOX
022E  85 EA              STA (zp) EA     PUT IN OUTS
0230  8D 20 08           STA DSPY        SHOW VOX NUMBER
0233  20 05 02           JSR INIT 1      CLEAR NTBL
0236  C9 05     VOXT 2   CMP # $05       IS $05 DOWN?
0238  D0 0A              BNE VOXT 3      NO,TO VOXT 3
023A  A9 02              LDA # $02       YES, PREP 2 VOX
023C  85 EA              STA (zp) EA     PUT IN OUTS
023E  8D 20 08           STA DSPY        SHOW VOX NUMBER
0241  20 05 02           JSR INIT 1      CLEAR NTBL
0244  C9 06     VOXT 3   CMP # $06       IS $06 DOWN?
0246  D0 0A              BNE VOXT 4      NO, TO VOXT 4
0248  A9 03              LDA # $03       YES, PREP 3 VOX
024A  85 EA              STA (zp) EA     PUT IN OUTS
024C  8D 20 08           STA DSPY        SHOW VOX NUMBER
027C  A9 80              LDA # $80       YES, PREP GLIDE
027E  85 CC              STA (zp) CC     PUT IN XPOSE CH 4
0280  C9 0C     TTST 1   CMP # $0C       IS $0C DOWN?
0282  D0 04              BNE TTST 2      NO, TO TTST 2
0284  A9 0C              LDA # $0C       YES, PREP XPOSE
0286  85 CF              STA (zp) CF     PUT IN XPOSE CH 1
0288  C9 0D     TTST 2   CMP # $0D       IS $0D DOWN?
028A  D0 04              BNE TTST 3      NO, TO TTST 3
028C  A9 0C              LDA # $0C       YES, PREP XPOSE
028E  85 CE              STA (zp) CE     PUT IN XPOSE CH 2
0290  C9 0E     TTST 3   CMP # $0E       IS $0E DOWN?
0292  D0 04              BNE TTST 4      NO, TO TTST 4
0294  A9 0C              LDA # $0C       YES, PREP XPOSE
0296  85 CD              STA (zp) CD     PUT IN XPOSE CH 3
0298  C9 0F     TTST 4   CMP # $0F       IS $0F DOWN?
029A  D0 04              BNE RETURN      NO, TO RETURN
029C  A9 0C              LDA # $0C       YES, PREP XPOSE
029E  85 CC              STA (zp) CC     PUT IN XPOSE CH 4
02A0  4C 0F 02  RETURN   JMP POLY        DO IT AGAIN
024F  20 05 02           JSR INIT 1      CLEAR NTBL
0252  C9 07     VOXT 4   CMP # $07       IS $07 DOWN?
0254  D0 0A              BNE GLDT 1      NO, TO GLDT 1
0256  A9 04              LDA # $04       YES, PREP 4 VOX
0258  85 EA              STA (zp) EA     PUT IN OUTS
025A  8D 20 08           STA DSPY        SHOW VOX NUMBER
025D  20 05 02           JSR INIT 1      CLEAR NTBL
0260  C9 08     GLDT 1   CMP # $08       IS $08 DOWN?
0262  D0 04              BNE GLDT 2      NO, TO GLDT 2
0264  A9 80              LDA # $80       YES, PREP GLIDE
0266  85 CF              STA (zp) CF     PUT IN XPOSE CH 1
0268  C9 09     GLDT 2   CMP # $09       IS $09 DOWN?
026A  D0 04              BNE GLDT 3      NO, TO GLDT 3
026C  A9 80              LDA # $80       YES, PREP GLIDE
026E  85 CE              STA (zp) CE     PUT IN XPOSE CH 2
0270  C9 0A     GLDT 3   CMP # $0A       IS $0A DOWN?
0272  D0 04              BNE GLDT 4      NO, TO GLDT 4
0274  A9 80              LDA # $80       YES, PREP GLIDE
0276  85 CD              STA (zp) CD     PUT IN XPOSE CH 3
0278  C9 0B     GLDT 4   CMP # $0B       IS $0B DOWN?
027A  D0 04              BNE TTST 1      NO, TO TTST 1
```



Do unheard of things with your Mini!

You don't have to sound like all the rest. RMS modifications like sync, distortion and chromatic transpose add versatility and unique functions to your already great standard.

For complete details, write to
Rivera Music Services,
48 Brighton Ave., #11,
Boston, Mass. 02134,
or call us at
617-782-6554.

RMS Rivera Music Services

Make your Mini an RMS modified Mini.