SUPER 99 MONTHLY

Music is in the air! And on our pages this month. We felt that with so many family gatherings and Holiday celebrations in December, it would be a great time to cover music and sound effects.

This month, we've included LOGO and Assembly Language! We promised a variety of languages and here you have it! Thanks for all the requests for these two important languages. We'll try to fit them in on a more regular basis.

We have reformatted this issue so that certain areas of interest will be grouped and therefore be easier to locate. We will try to maintain the same sequence of categories from month to month. As not all articles fall distinctly into a category, some articles will remain uncategorized. Within the next few issues, we expect to have most articles categorized and a complete format intact. We've had several requests for this and our objective is certainly to offer a concise and practical format.

Transposing Music to "Computereze"

STANDARD: 1A 9A

The TI-99/4A is very versatile in creating a variety of musical effects. You will need to have at least an elementary knowledge of reading music to best follow our examples. Just as other musical forms require structure, computer music does, too.

Our TI uses the CALL SOUND statement, which has three different parameters - duration, frequency, and volume. Let's begin with the duration. By using a variable for the duration throughout a song or portion of a song, we can easily adjust the tempo. Otherwise, we would have to change every CALL SOUND in the program to make one tempo change! Some of you have probably already discovered what a big mistake that can be! Here is a short example of the use of a variable for duration:

```
>100 D=150
>110 FOR I=1 TO 10
>120 CALL SOUND(D,262,0)
>130 NEXT I
```

Now try changing the value of D. Oh, the beat is dull. You're right. We need some eights and sixteenths, etc. Try this:

```
>100 D=1000
>110 FOR I=1 TO 10
>120 READ TYPE
>130 CALL SOUND(D/TYPE,262,0)
>140 NEXT I
>150 DATA 8,16,16,4,2
>160 DATA 2,4,8,16,16
```

The DATA statements now tell us the type of note being used - quarter, half, etc. Note that our DATA statements add up for one measure in 4/4 time. It is a very good idea to keep your DATA statements separated by measures so that you can total the counts for the measure and be able to

-->

-1-

easily locate the measure that you are working with (REM's may be needed).

The duration is in milliseconds (1 second = 1000 milliseconds) and can vary as much as 1/60 seconds due to interrupt timings. The duration may be either positive or negative. With positive durations, the sound will continue for the specified duration. With a negative duration, the sound continues either through the specified duration or until a new CALL SOUND is specified.

The next parameter in the CALL SOUND statement is the frequency. The frequency is a measurement of Hz. Specifying 110 equates to producing a tone of 110 Hz.

The frequency produced is not an extremely exact one. In fact, the TI BASIC Reference Manual states that a frequency may vary from 0 to 10 percent depending on the frequency. However, the sound produced is, for instance, accurate enough to dial a telephone number on a push-button phone, so we will assume that it is adequate for most of our uses. For some applications, such as random patterns, it is important to note that musical half-steps increase by 2^(1/12) Hz. However, BASIC rounds all SOUND parameters to an integer, so most manipulations to arrive at a precise tone are a waste of time.

The half-step formula does lead to a very useful bit of knowledge. An octave interval is the Hz value doubled. This sometimes results in a value that is one Hz off of the value listed in the BASIC Reference Manual, but most of us cannot tell the difference anyway. Let's look at an example of the interval concept:

```
>100 A=220
>110 CALL SOUND(1000,A,0,A*2,
 0)
```

The frequency can be specified up to 44733 Hz, which is above human hearing limits. This allows a very good method of including musical rests, especially when combined with a low volume (see below).

The frequency can also be positive or negative. The negative frequencies are what is known in electronics as noises and that is what they sound like -- what most of us refer to as noise. The noises may range from -1 to -8. A maximum of three tones and one noise may be activated at one time. When accessed in certain patterns, the noises can produce otherwise unobtainable frequencies (see Tigercub Software "Tips from the Tigercub No. 2"). So, playing around with various sound combinations can lead to some interesting results!

The volume can be specified as from 0 to 30, with 0 being loudest. Use the 30 value for musical rests.

It is very important to note that the manner in which a program is constructed can greatly affect the resulting music or "music"! You must be creative at times. The faster the tempo or "run" you seek, the more critical program timing becomes. Complicated formulas and/or DEF statements can slow your program to a crawl. If you want to do a professional job, you may want to do timing experiments on various BASIC statements. This differs from most of the applications we have previously covered and makes programming music a bigger challenge than we sometimes first imagine for some songs.

Returning to the half-step interval formula, you can use it to establish a relative distance of tones from one another. Number the tones in the BASIC Manual from 0 to 48 (some are repeated in the manual), with 110 being 0. We will refer to our new numbering system with the variable NN. Here is the formula for the frequency:

```
FREQUENCY=55*2^((NN+12)/12)
```

To improve program speed, try this:

```
>100 NN(9)=3.363585661
>110 CALL SOUND(1000,55*NN(9)
 ,0)
```

Due to the interrupt system on the TI-99/4A, combining music and graphics is cumbersome at best and a true mix is impossible. Combining music and graphics represents one of the most challenging forms of programming the TI, but we've seen some impressive efforts, so it is very possible to obtain a good program. A hint is that you may need to work on the timings of BASIC statements!

Our BASIC musical example is a traditional Christmas carol -- "Silent Night":

```
>100 FOR I=1 TO 46
>110 READ D,DOT,F1,V1,F2,V2
>120 CALL SOUND(5000/(D/DOT),
 F1,V1,F2,V2+5,F1*2,V1+3)
>130 NEXT I
>999 END
>1000 DATA 8,1.5,330,0,392,0
>1010 DATA 16,1,349,0,440,0
>1020 DATA 8,1,330,0,392,0
>1030 DATA 4,1.5,262,0,330,0
>1035 REM
>1040 DATA 8,1.5,330,0,392,0
>1050 DATA 16,1,349,0,440,0
>1060 DATA 8,1,330,0,392,0
>1070 DATA 4,1.5,262,0,330,0
>1075 REM
>1080 DATA 4,1,349,0,587,0
>1090 DATA 8,1,349,0,587,0
>1100 DATA 4,1.5,349,0,494,0
>1105 REM
>1110 DATA 4,1,330,0,523,0
>1120 DATA 8,1,330,0,523,0
>1130 DATA 4,1.5,330,0,392,0
>1135 REM
>1140 DATA 4,1,349,0,440,0
>1150 DATA 8,1,349,0,440,0,0
>1160 DATA 8,1.5,440,0,523,0
>1170 DATA 16,1,392,0,494,0
>1180 DATA 8,1,349,0,440,0
>1185 REM
>1190 DATA 8,1.5,330,0,392,0
>1200 DATA 16,1,349,0,440,0
>1210 DATA 8,1,330,0,392,0
>1220 DATA 4,1.5,262,0,330,0
>1225 REM
>1230 DATA 4,1,262,0,440,0
>1240 DATA 8,1,349,0,440,0
>1250 DATA 8,1.5,440,0,523,0
>1270 DATA 8,1,349,0,440,0
>1275 REM
>1280 DATA 8,1.5,330,0,392,0
>1290 DATA 16,1,349,0,440,0
>1300 DATA 8,1,330,0,392,0
>1310 DATA 4,1.5,262,0,330,0
>1315 REM
>1320 DATA 4,1,349,0,587,0
>1330 DATA 8,1,349,0,587,0
>1340 DATA 8,1.5,294,0,698,0
>1350 DATA 16,1,349,0,587,0
>1360 DATA 8,1,349,0,494,0
>1365 REM
>1370 DATA 4,1.5,330,0,523,0
>1380 DATA 4,1.5,392,0,659,0
>1385 REM
>1390 DATA 8,1,330,0,523,0
>1400 DATA 8,1,392,0,44733,25
>1410 DATA 8,1,262,0,330,0
>1420 DATA 8,1.5,247,0,392,0
>1430 DATA 16,1,247,0,349,0
>1440 DATA 8,1,247,0,294,0
>1445 REM
>1450 DATA 2,1.5,262,0,262,0
>1460 REM
```

Note that "Silent Night" is in 6/8 time. Therefore, the measures add up to 3/4 instead of 1. Also, the song uses dotted notes, so we have used the variable DOT to indicate the notes that need a 1.5 times count to make it easy to compare to the sheet music. We have also taken the first frequency and raised it an octave for the third frequency. You can vary the tempo (for listening or singing, for example) by changing 5000 to another value.

We hope this gives you a good start on setting your music into "computereze".

Program Development: Structuring in Extended BASIC

STANDARD: 1A 2A 9A

BASIC is now nearly 20 years old. The general usage of the language has deteriorated during that period to such a degree that it is now often regarded as one of the least desirable languages available -- sort of the street slang of the computer world. The decline of BASIC can be directly linked to it being one of the few languages without a standard. A Minimal Standard was adopted several

-->

years ago (though TI BASIC is still one of the few BASIC's that properly follows the guidelines). Expectations are that in 1985, a full standard for BASIC will finally be adopted. Ironically, the "new" BASIC is actually the original Dartmouth BASIC with its Darmouth evolutions!

The primary emphasis of the proposed standard (hereafter referred to as ANSI BASIC) is on structuring. Using ANSI BASIC, an entire program can be written without line numbers! This results in a well-structured program that can be easily read by any knowledgeable programmer ("idiot jump" GOTO's, etc. so common in today's "spaghetti code" are discouraged).

In TI Extended BASIC, we do not have a few of the tools of ANSI BASIC (if you're familiar with languages, ANSI BASIC includes DO loops, CASE structures, etc.). Plus, TI Extended BASIC requires line numbers for every five or so lines of code.

However, we can benefit greatly by using structuring techniques as was intended for Extended and ANSI BASIC.

One fundamental aspect of program structuring is known as modularity (we have postponed this discussion until now to be able to offer a more thorough examination). The modularity concept is based on being able to change parts of a program (modules) without affecting the unchanged portions. The subprogram, with its passed parameters, is the ideal method of structuring in a modular format. Only the passed parameters affect the rest of the program. One glance at the parameter line tells us and the computer where to trace the variables in the program.

So, what is really so great about all of this structuring? After all, it requires an entirely different approach from what most micro owners have become accustomed. Let's take a a closer look.

We have already covered that a structured program is more easily read and it is easier to trace the variables. Subprograms offer the advantage that you can usually test a subprogram as soon as it is completed, much like a FORTH word can be tested in immediate mode. Subprograms can be stored to disk in MERGE format under the name of the subprogram (or a similar name). Complicated formulas and algorithms can be reduced to one normally difficult programming task, followed by plugging in a few parameters. Let's look at a very simple example first.

```
>1 REM SIZING AN AQUARIUM
>100 CALL GET_DIMENSION(HEIGH
 T,ALENGTH,WIDTH)
>110 CALL SHOW_GALLONS(HEIGHT
 ,ALENGTH,WIDTH)
>999 END
>1000 SUB GET_DIMENSION(HEIGH
 T,ALENGTH,WIDTH)
>1010 CALL CLEAR
>1020 DISPLAY AT(4,5):"SIZING
  AN AQUARIUM"
>1030 DISPLAY AT(7,1):"ENTER
 ALL DIMENSIONS IN INCHES"
>1040 DISPLAY AT(10,1):"HEIGH
 T": :"LENGTH": :"WIDTH"
>1050 ACCEPT AT(10,8)BEEP VAL
 IDATE(DIGIT)SIZE(2):HEIGHT
>1060 ACCEPT AT(12,8)BEEP VAL
 IDATE(DIGIT)SIZE(2):ALENGTH
>1070 ACCEPT AT(14,8)BEEP VAL
 IDATE(DIGIT)SIZE(2):WIDTH
>1080 SUBEND
>2000 SUB SHOW_GALLONS(HEIGHT
 ,ALENGTH,WIDTH)
>2010 GALLONS=(HEIGHT*ALENGTH
 *WIDTH)/231
>2020 DISPLAY AT(24,1):GALLON
 S;" GALLONS"
>2030 SUBEND
```

Possibly the most striking aspect of this program is that it was written from the TOP DOWN! This is known as, of course, Top Down Programming. The fundamental objectives of the program were spelled out in the body -- lines 100 and 110. When it came time to write the subprograms, the objectives of the subprogram were very clear. And, anyone who has read a few Extended BASIC programs should be able to clearly follow every line of code. Much more complex tasks can be

-->

approached in exactly the same manner. "Outline" at the top, detail at the bottom, keep it simple.

Well, hopefully this article has brought together the loose ends we've left in previous issues. Practice writing structured programs and chances are that in a few weeks or months you'll be operating while your friends are still struggling with the program!

Product Review: "Nuts & Bolts"

STANDARD: 1A 2A 4B 5A 6B 7A 9A

"Nuts & Bolts" is a new software package from Tigercub Software, 156 Collinwood Ave., Columbus, OH 43213.

"Nuts & Bolts" is a package of 100 Extended BASIC subprograms on one disk. Our original intention was to test the entire package. To be quite honest, there are so many uses for the subprograms that after 6 hours of testing and getting side-tracked on the utility and/or enjoyment of the routines, we only tested about 70 of the routines and doubt that anyone will be able to use all of this wealth of software within several months!

The subprograms are divided into 15 categories and each subprogram is on disk in MERGE format. All of the line numbers are at 20000 and above to avoid conflict with the line numbers to be used in the main program. The categories are Type Fonts; Text Displays; Screen Wipes; Pauses; Programming Aids; Data Saving, Reading; Displays; Time and Date; Music; Sorts and Scrambles; Printer Aids; Keyboard and Joystick Control; Math; Protection; Miscellaneous.

Some of the more impressive routines are listed here:

CALL SLANT -- converts all uppercase letters, numerals and punctuation to a slanted form, similar to italics.
CALL CHARSET2 -- will restore the characters not restored by CALL CHARSET
CALL FORMATTER -- will reformat text from DATA statements to be displayed on screen without words wrapping around.
CALL OUTSIDE_IN -- will erase the screen from the edges inward.
CALL CURSOR(A) -- will redefine the cursor to the shape, normal or redefined, of the ASCII of variable A.
CALL CLOCK(R,CC) -- displays seconds at row R, column CC until key is pressed.
CALL NO_REPEAT(A,B,X) -- randomly selects number X (up to 255) from a sequence A to B.
CALL HEX_DEC(H$,D) -- converts hex numbers to decimal equivalent.
CALL CARDS -- shuffles a deck of cards and selects deals (no duplication).

These are only a few of the many useful routines on this disk!

So that those of you who are not familiar with the users group articles called "Tips from the Tigercub" will not be surprised, there are a few routines on the disk that are better categorized as interesting examples than practical routines (some of the tips offered are the same way). On the disk, you'll find CALL FLAG, which unfurls an American flag in a minimum amount of program lines -- we haven't fit that one into a program yet, but it is a good programming example!

Accompanying the disk is 5 pages of documentation. There are examples of the use of most of the subprograms. For the most part the documentation is adequate, but if you have problems, write to Tigercub (test it all first to save you and them time) -- they're usually glad to offer support.

Priced at $19.95, "Nuts & Bolts" is a real winner. It represents 80 hours of work after many of the routines had already been developed, so if your time is worth only a quarter an hour, you may want to try programming all of this yourself. We recommend the easy way -- buy it.

-->

ECHOE....echoe

STANDARD: 1A 9A

Echoes and reverberations are interesting and often useful sound effects. Technically, an echo takes 1/10 second or longer. Here is a program to simulate an echo:

```
>100 FREQ=440
>110 TIME=34
>120 REPITITIONS=24
>130 INTERVAL=3
>140 TONE_ADJ=1
>150 FOR VOLUME=0 TO REPITITI
 ONS STEP INTERVAL
>160 CALL SOUND(-TIME,FREQ*1,
 VOLUME)
>170 FOR DELAY=1 TO TIME-3
>180 NEXT DELAY
>190 CALL SOUND(-TIME,FREQ*TO
 NE_ADJ,VOLUME+5)
>200 FOR DELAY=1 TO TIME-3
>210 NEXT DELAY
>220 NEXT V
```

We can convert this to a synthetic reverb effect as is used in many songs and gadgets by changing lines 100 to 140. By changing the lines to INPUT instead of the implied LET, the program becomes a test for sounds we are creating. Try entering a time of 4, an interval of 1, and a a tone adjustment of 2, leaving the other values as they were. Adding the following lines will allow you to hear the sound several times consecutively and then return to the INPUT section:

```
>145 FOR I=1 TO 5
>230 NEXT I
>240 GOTO 100
```

ASSEMBLY

Quick Disk Load of Low Memory

from Extended BASIC

STANDARD: 1A 2AC 4B 5A 6B 7A 9A

Editor's Note: A million thanks go to Ray Robinson of Lake Charles, Louisiana for lending his expertise in helping to develop this article. Ray has more ideas that we'll try to share with you in future issues. Ray's philosophy has been that if TI did not provide a way, it can probably be done anyway! You'll soon see what we mean!

****

Before we get into the specifics of the program listed on the following pages, let's look at the utility of the overall concepts of the program. First, it includes an Extended BASIC DSRLNK, which TI did not provide. The DSRLNK is needed to access peripherals from Assembly Language. Both Mini Memory and the Editor/Assembler have a built-in DSRLNK. Next, this program was developed by combining several of Ray Robinson's ideas into one program. So, if you do not have the equipment required to accomplish our recommended goal, maybe you can come up with a use for your own configuration. Let us know what you come up with!

Our program will load 7 screens from disk and display them in 12 seconds using one Extended BASIC statement to load the program, one Extended BASIC statement to execute the program, and one Extended BASIC CorComp Toolshed statement each to display the 7 screens (the 7 displays could be in a loop)!!! An additional statement could be used to change to the names other screens are stored under to load more sets of 7 screens! Imagine, an Extended BASIC program with no screen set-ups! Although we have not tried it, theoretically at least 448 different screens could be loaded from one disk in less than 13 minutes from one Extended BASIC program! Note that our figures are based on using a DS/DD TEAC disk drive with the CorComp Controller head step setting adjusted (there are dip switches inside the card) to 3 ms.

Before you conclude that we are referring to limiting the usage of this program to fancy games, think about the possibilities of a really fast data base file. That's right, 7 screens of text on a selected topic in a flash with up to 64 topics on one disk (22 sectors per 7 screens)! Many

-->

programs use as much as half or more of the program's space to build screens. Our Assembly Language program could therefore sometimes expand a program to the equivalent of twice its length!

Let's look at the specifics of how all of the above can be done. To begin with, Assemble the program listing with the object code being named "CPUDSK". Then, change the labels called "CPUDSK" in the source file to "DSKCPU". Add VMBR EQU >202C. Change the first data labeled "PDATA" from >0600 to >0500. Move the 4 lines preceding BLWP @DSRLNK to after DATA 8. On the 4 lines moved, change the BLWP @VMBW to BLWP @VMBR. Assemble this new source into an object file called "DSKCPU". Refer to the File Management section of the Editor/Assembler Manual if you have difficulties in understanding the logic of the program.

"CPUDSK" saves >1500 (decimal 5376, which is equivalent to 768 screen positions times 7) bytes of Low Memory in a memory image format on disk. This memory image file is currently set up to be called "CPULO" and will show up on a catalog with the file type being "Program" (cannot be loaded via BASIC OLD command).

"DSKCPU" simply reverses the process, loading the same >1500 bytes almost immediately after the program, beginning at >2710 (decimal 10000). So, you should bank your screens in and out of Low Memory in the same way as we banked the Pattern Descriptor Table in last month's issue, beginning the storing at decimal 10000.

Although we keep referring to screens, the program is actually storing whatever is in Low Memory. It could be any of VDP copied to Low Memory or whatever. It could also be used by someone who does not have a CorComp card if some short Assembly programs were written to copy bytes around, say from VDP.

Build your screens in Extended BASIC. Bank the screens to Low Memory using the Toolshed statements. Save Low Memory using "CPUDSK". Key NEW. Write your program and use "DSKCPU" to load your previous section of Low Memory. Bank in screens as needed.

The file name is generated by the TEXT statement that begins at >258A in "DSKCPU" and >258C in "CPUDSK". You can change 'DSK1.CPULO' to any valid file name of the same length by using LOAD to poke the ASCII values of the characters in the new file name. The last DATA in PDATA is the length of the filename if you want to change the length.

A short test (example) program in Extended BASIC follows the listing.

```
        DEF   CPUDSK
VMBW    EQU   >2024
PABBUF  EQU   >1000
PAB     EQU   >F80
BUFFER  EQU   >2710
STATUS  EQU   >837C
PNTR    EQU   >8356
        AORG  >2560
MYREG   BSS   >20
PDATA   DATA  >0600,>1000,>0000,>1500
* note:  next data can be placed on
* same line as PDATA above.
        DATA  >000A
        TEXT  'DSK1.CPULO'
* This EVEN is a reminder in case you
* change the length to odd.
        EVEN
CPUDSK  LWPI  MYREG
        LI    R0,PAB
        LI    R1,PDATA
        LI    R2,>26
        BLWP  @VMBW
        LI    R6,PAB+9
        MOV   R6,@PNTR
        LI    R0,PABBUF
        LI    R1,BUFFER
        LI    R2,>1500
        BLWP  @VMBW
        BLWP  @DSRLNK
        DATA  8
        CLR   R0
        MOVB  R0,@STATUS
        LWPI  >83E0
        B     @>0070
* Extended BASIC DSRLNK Routine
NAMBUF  BSS 8
D1      DATA >2000
D2      DATA >2EAA
```

-->

```
WP      BSS 10
D3      BSS 22
DSRLNK  DATA WP,$+2
        MOV  *R14+,R5
        SZCB @D1,R15
        MOV  @>8356,R0
        AI   R9,>FFF8
        BLWP @>2028
        MOVB R1,R3
        SRL  R3,8
        SETO R4
        LI   R2,NAMBUF
LL2     INC  R0
        INC  R4
        C    R4,R3
        JEQ  LL1
        BLWP @>2028
        MOVB R1,*R2+
        CB   R1,@D2
        JNE  LL2
LL1     MOV  R4,R4
        JEQ  LL3
        CI   R4,>0007
        JGT  LL3
        CLR  @>83D0
        MOV  R4,@>8354
        INC  R4
        A    R4,@>8356
        LWPI >83E0
        CLR  R1
        LI   R12,>0F00
LL6     MOV  R12,R12
        JEQ  LL4
        SBZ  0
LL4     AI   R12,>0100
        CLR  @>83D0
        CI   R12,>2000
        JEQ  LL5
        MOV  R12,@>83D0
        SBO  0
        LI   R2,>4000
        CB   *R2,@D2+1
        JNE  LL6
        A    @D3,R2
        JMP  LL7
LL9     MOV  @>83D2,R2
        SBO  0
LL7     MOV  *R2,R2
        JEQ  LL6
        MOV  R2,@>83D2
        INCT R2
        MOV  *R2+,R9
        MOVB @>8355,R5
        JEQ  LL8
        CB   R5,*R2+
        JNE  LL9
        SRL  R5,8
        LI   R6,NAMBUF
```

add MOV R0,R9

```
LL10    CB   *R6+,*R2+
        JNE  LL9
        DEC  R5
        JNE  LL10
LL8     INC  R1
        BL   *R9
        JMP  LL9
        SBZ  0
        LWPI WP
        MOV  R9,R0
        BLWP @>2028
        SRL  R1,13
        JNE  LL11
        RTWP
LL5     LWPI WP
LL3     CLR  R1
LL11    SWPB R1
        MOVB R1,*R13
        SOCB @D1,R15
        RTWP
        END
```

Here is a test program to be sure the Assembly program is working:

```
>100 CALL INIT :: DELETE "LD-
 CMDS" :: CALL LOAD("DSK1.CPU
 DSK")
>110 FOR I=1 TO 7
>120 CALL CLEAR
>130 CALL HCHAR(1,1,48+I,768)
>140 ADDRESS=10000+((I-1)*8
 )
>150 CALL LINK("MOVEM")(2,0,A
 DDRESS,768)
>160 NEXT I
>170 CALL LINK("CPUDSK")
>180 CALL CLEAR
>190 FOR I=1 TO 7
>200 ADDRESS=10000+((I-1)*8
 )
>210 CALL LINK("MOVEM")(2,0,A
 DRESS,768)
>220 NEXT I
>230 CALL LOAD("DSK1.DSKCPU")
>240 CALL LINK("DSKCPU")
>250 FOR I=1 TO 7
>260 ADDRESS=10000+((I-1)*8
 )
>270 CALL LINK("MOVEM")(3,ADD
 RESS,0,768)
>280 NEXT I
```

If you have a lot of difficulty with this, send us an initialized disk and $2.00 for mailing and handling.

-->

TI-Writer

Masters and Templates

STANDARD: 1A 2E 3B 4B 5A 6B 7A 9A 10A

TI-Writer allows the capability of loading a file, making modifications, and then saving the file under whatever name you wish with the press of only a few keys. This, in conjunction with other features of TI-Writer makes it easy to create master documents, sometimes known as templates.

A master does not have to consist of an entire document. Charles Foster, of New Orleans, writes that he has a file of transliterated printer commands that he uses to begin new letters, etc. He uses the characters that he normally uses the least as the transliterate characters, such as the braces, brackets, reverse slant, and tilde. Here is an example:

.TL 123:27,14

This transliterate sets the left brace as double width (expanded print) on for his Epson FX-80.

Mike Kelley, of Irish Input in San Diego (TIBBS (619) 276-3173), says he sets printer commands in a file by direct commands. For instance, on his Gemini 10X, <CTRL><u><O><CTRL><u> sets condensed print. After we get a chance to review Mike's entire list, we'll pass it along.

As we discussed last month, pages can be written over and saved under a different name to save formatting time. However, we recommend that you have a backup of your master before doing this as it is easy to overwrite the master with the new file. An alternative to a backup is to have the master on a separate disk and remove that disk immediately after loading in the master.

Another handy trick is to have a master for your letters. The one item that you are most likely to forget to update is the date. Use the Define Prompt Command to insert the date from TEXT FORMATTER, as follows:

.DP 1 Date

1

Up to 28 characters can be input using the DEFINE PROMPT. The example above works out very well for letters that may not be completed on the same day started, too! DEFINE PROMPT and other features of TI-Writer can also be used for form letters, but form letters are a little different from permanent masters, so we'll reserve that topic for some other time.

Masters can be quite complex to set up, but the time spent may be more than offset by the time saved in producing documents. In future issues, we'll tie the master to a form document and mail merge to show the total picture of just how powerful TI-Writer is!

We were surprised that nobody wrote in about last month's TI-Writer column. Next month we'll show how to produce the same results through a similar, but more flexible technique!

LOGO

LOGO = Easy Music

STANDARD: 1A 2F 4B(o) 5A 6B(o) 7A 9A

LOGO is really well designed for working with music because it uses a number of actual musical terms. After all, how many composers know what TI BASIC's CALL SOUND means? In LOGO, we can use PLAYMUSIC, REST, STACCATO, SETTEMPO, etc.

The LOGO Manual's music section is one of the best documented portions of the book. Setting up the round "Frere Jacques" as in the Manual is easy and a good introduction to LOGO music. LOGO is great for children, music students, and adults alike! As LOGO is so simple for anyone with a basic knowledge of music, we'll go directly to a song on the next page.

-->

"My Old Kentucky Home"

```
TO K1
MUSIC [11 ] [4 ]
END
TO K2
MUSIC [11 11 7 9 11 ] [4 4 4 2 2 ]
END
TO K3
MUSIC [12 11 12 16 14 12 ] [3 1 2 2 6
2 ]
END
TO K4
MUSIC [11 9 7 7 6 7 ] [2 4 2 4 2 2 ]
MUSIC [9 9 ] [12 4 ]
END
TO K5
MUSIC [12 11 12 16 14 7 9 ] [3 1 2 2 4
2 2 ]
END
TO K6
MUSIC [11 11 9 7 11 9 ] [4 4 2 2 3 1 ]
MUSIC [7 11 ] [12 4 ]
END
TO K7
MUSIC [11 9 7 7 6 7 ] [2 4 2 2 4 2 ]
MUSIC [9 2 ] [12 4 ]
END
TO K8
MUSIC [11 7 12 11 9 7 ] [2 2 2 2 6 2 ]
MUSIC [7 ] [12 ]
REST 4
MUSIC [14 11 12 16 ] [6 2 6 2 ]
MUSIC [14 11 ] [2 6 ]
REST 4
MUSIC [9 ] [4 ]
MUSIC [7 9 7 4 ] [6 2 4 4 ]
MUSIC [7 7 9 ] [12 2 2]
END
TO K9
MUSIC [11 7 12 11 9 9 6 ] [3 1 2 2 4 3
1 ]
MUSIC [7 ] [12 ]
END
TO KPATTERN
K1 K2 K3 K4 K2 K5 K6 K2 K3 K7 K2 K5
K8 K2 K5 K9
END
TO KENTUCKY
SETTEMPO 325
SETVOICE 0
SETVOICE 1
SETVOLUME 15
KPATTERN
PLAYMUSIC
END
```

FORTH

Programmer's Screens

STANDARD: 1A 2C 4B 5A 6B 7A 9A

When using FORTH, it is very easy to lose track of what you have placed on the various screens. Therefore, it is a good idea to reserve several consecutive screens for "housekeeping".

One idea for these reserved screens is a catalog of what you have on the disk and on which screen. Remember to update this screen often. It is also a good idea to give it a name on screen 3 like this:

```
: DIR  120 LOAD ;
```

Your next housekeeping chore is to store your favorite words. You may want to set this screen up on the boot screen (3) also. Your words should include as many stack manipulation words as possible. There are several converted from Brodie's book in the cross-reference section of the TI FORTH Manual. Miller's Graphics also printed several of these a few issues back. Hopefully, you've developed a few on your own. The stack is very important in FORTH and having a good stack dictionary is very helpful!

Next, you might have other special words for special purposes, such as screen graphics, mode changes, etc. By the way, if you use -64SUPPORT, here is a word to keep from getting to one of those "never-never" displays:

```
: NEW  TEXT COLD ;
```

That will set your screen off of bit map and onto text before you re-boot.

As you now know, there are many reasons for reserving screens for housekeeping, but if you don't go in occassionally to update, many of the screens will become useless. Check your directory at least once a month!

-->

99 POTPOURRI

News, Corrections, Updates, Editorials, Kudos, and Come-what-may

Well, word is in that the rights to Extended BASIC are now held by Exceltec, Inc., formerly known as Sunware Ltd. The firm indicates that the "new" Extended BASIC will be essentially the same product as was offered by TI, including materials and documentation. This is certainly good news as Extended BASIC is extremely useful and the supply had been dwindling to almost zero. The new prices are $99.95 each, $89.95/3, and $79.95/5 and up.

We were unable to ascertain from Exceltec whether the firm remains in the business of making ROM cartridges for third party software producers.

CorComp reports that their 9900 Disk Controller is compatible with the Foundation 128K card and virtually all TI-99/4A products.

Last month we reported that the 99er Users Group Association (3535 So. H St., #93, Bakersfield, CA 93304), whose publication is now known as The National Ninety-Niner, had DS FORTH available in a back issue. Their writer, Jim Vincent, has now developed DS/DD FORTH, so look for it in their September issue.

Somehow we've overlooked mentioning a couple of items. One is Dragonslayer's 99/4A Auto Spell-Check program to check your TI-Writer spelling. The program lists for $49.95 and is available from Triton.

If you're into D and D type games you should try "Doom of Mondular", available from Symbiotech, Inc., P.O. Box 320, Roscoe, IL 61073 for $24.95, plus $2.50 handling (IL residents add sales tax).

Quiz: What company invented the integrated circuit, the microprocessor and the microcomputer? If you need a hint (?), look on the back of the TI manuals, such as TI-Writer.

COMING IN JANUARY - Sprites and Graphics issue.

CORRECTIONS TO OCTOBER:

PAGE 4: We have recieved reports that our parallel default may not work with the TI Impact Printer, causing several pages to be ejected when printing from TEXT FORMATTER. If you own an Epson printer, the same might be true as the TI Printer was made by Epson.

SEASON'S GREETINGS

We hereby acknowledge any registrations, copyrights, or other legal rights held in association with the company names and/or products listed below:

Texas Instruments, Inc.: TI, TI-99/4A, TI-Writer, TI Extended BASIC, TI BASIC, TI LOGO II, TI Impact Printer
Epson: FX-80
Star Micronics, Inc.: Gemini 15-X PC, Gemini 10-X
CorComp, Inc.: 9900 Disk Controller Card
TEAC: TEAC 55B

-->

Super 99 Monthly is published monthly by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663. Subscription rate in U.S. and possessions is $12.00 per year; all other countries $16.00 U.S. funds for surface mail. All correspondence received will be considered unconditionally assigned for publication and copyright and subject to editing and comments by the editors of Super 99 Monthly. Each contribution to this issue and the issue as a whole Copyright 1984 by Bytemaster Computer Services. All rights reserved. Copying done for other than personal archival or internal reference use without the permission of Bytemaster Computer Services is prohibited. Bytemaster Computer Servies assumes no liability for errors in articles.

STANDARD KEY

1	Computer	A	TI-99/4A (LTA4583)
2	Cartridge	A	Extended BASIC
		C	Editor/Assembler
		E	TI-Writer
		F	LOGO II
3	RS-232	B	TI
4	Disk Drive	A	TI
		B	TEAC 55B
5	Expansion Box	A	TI
6	Disk Controller	A	TI
		B	CorComp
7	32K Card	A	TI
9	Monitor or TV	A	TV & RF Modulator
10	Printer	A	Gemini 15-X PC

Note: List adjusted monthly.

Addendum - December Super 99 Monthly

Error November: Our apologies to the 99er User Group Association. The subscription rate for their publication is $12.00/yr. As this is a non-profit organization, if you sent in the amount we quoted in Nov., please send them the difference. Otherwise, the Association could deem it necessary to pro-rate your subscription.

December Notes: "Doom of Mondular" is $5.00 off through December 31, 1984.

The FORTi Music System is available in quantities of 4 or more at a discounted price of $199.95 ($100 off) through Dec. 21, 1984, from Texas Peripherals, Inc., 624 Sorita, Heath, TX 75087 (SOURCE I.D. TI0004). The firm is headed by Don Bynum, former head of Home Computer engineering at TI.

Error December: In the Assembly program on p. 8, in the section with label DSRLNK, after MOV *R14+,R5 add a line for MOV R0,R9
In the Extended BASIC test for the Assembly program, insert an asterisk before the 8 in lines 140, 200 and 260.