DIGITALKER 128

Free Spirit Software

Welcome To DigiTalker 128!

DigiTalker 128 is a new programming utility for BASIC 8 (c) and Basic 7 programs. With it and one of the DigiTalker ClipSound Disks you can easily add digitized speech to your C128 80 column programs.

One of the first things to understand is that DigiTalker 128 is not a speech synthesizer. Instead it is a set of machine language routines that can play digitized sound samples (electronically recorded) from with a program. These sounds can be found on a ClipSound disk, which in most cases have hundreds of thousands of bytes of sound on them. (For example, ClipSound Volume 1 has over 500,000 bytes of sound in it.)

The machine language comes in two files, one for the C128 Basic 7 80 column mode (TALK.BASIC7) and one for use in BASIC 8 programs (TALK.BASIC8). Once these have been installed in memory using a simple BLOAD command, you can then begin to load and play your sounds.

DigiTalker supports, but does not require a Ram Expansion Unit, working with both the 128K 1700 and the 512K 1750 cartridges. While these REU's may not be *required*, they are certainly reccommended! Because of the huge amount of RAM required to record sounds (I recorded them at approx. 10K per second of sound), the SOUN. files on your disk are fairly large. I used a special file compression technique to keep the files

All in all there are five demonstration programs on the disks. Three are Basic 8 programs, and two are Basic 7. Some require a 17xx REU, while others will run on a plain C128.

To start them, turn off your computer and insert the DigiTalker 128 disk in drive 8, then turn the computer back on (make sure you are in 80 column mode, as that is the only mode supported by DigiTalker 128). A title screen will load, and you will be prompted to press a key. When you are presented with a menu asking you to choose between Basic 7 or Basic 8, select the one you want. Even if you don't own Basic 8 you can still run the programs, as a complete Basic 8 RunTime System is on the disk and will be loaded automatically if you make that selection.

Basic 7

There are two Basic 7 programs supplied. Here is a brief description of each.

B7.NUMBERS - This Basic 7 program will run without an REU. To use it, just type

RUN "B7.NUMBERS"

It will automatically load the TALK.BASIC7 ml and all sound files.

B7.USE REU DEMO - As the name implies, this is a Basic 7

demo that uses (and requires) a REU. Don't bother trying it if you don't have one installed. The demo is much like the number demo, except it uses all 26 letters, which explains why a REU is required. To run, type RUN"B7.USE REU DEMO"

BASIC 8

There are three BASIC 8 programs on the disk. Two are games, while the other is a useful utility. To use them either select the Basic 8 option when you boot up (which will take you to Workbench), or if you already own Basic 8 boot the editor disk and then insert the DigiTalker 128 disk and run the program you want.

B8.15PUZZLE - This is a Talking version of the classic 15 puzzle game, using Basic 8's hires color graphics and DigiTalker 128's speech routines. The result is a unique, entertaining and instructive demo. This program will run on any C128 and does not require a REU. To play you can click on its icon from the Workbench, or type RUN "B8.15PUZZLE"

B8.TALKMATCHGAME - This game requires a REU to be in place before you can begin. It is a graphic oriented version of concentration that uses a mouse (port 1) or joystick (port 2) as well as speech. From one to four players. To play click on its icon from the Workbench, or type

RUN "B8.TALKMATCHGAME"

B8.ALTERSPD - This is a utility that can be used to listen to any sound on drive 8 or 9, and can be used to make the sound play faster or slower, then this altered sound is save back to disk, replacing the old sound. For That reason you should only alter sounds from a backup copy of your ClipSound disk. ALTERSPD is useful to both Basic 8 and Basic 7 programmers, and it lets you change any sound. To run it, click on its icon from Workbench, or type

RUN "B8.ALTERSPD"

In the next two sections you will find information on programming the new Basic 7 or BASIC 8 commands. But before we start the technical details on programming, I wanted to describe the types of sounds you will find on the disk(s). This first volume of ClipSounds concentrates on some basic words and phrases, especially those that are most useful in a computer environment.

First, the complete alphabet (A-Z) is recorded. This is great for 'talking keyboards', and you will find two such demonstration programs for Basic 7, one for use with an REU, and another for the standard 128K machine.

There is also a complete set of numbers, and by combining them you can create any number upto 1 million.

Next, you will find all 16 of the C128's colors recorded. Since DigiTalker only works in the 80 column FAST mode, these color names are a little different from the 40 column versions.

There are words like up, down, left and right. Top and bottom is also available. Then you will find a wide array of computer terms, like mouse, joystick, printer, paper, disk, drive and many others. There are also complete phrases such as "Press Any Key When Ready", "Press The Left Mouse Button" or "I Could Not Find That File".

So there you have it. Now we will get into the nuts and

bolts of adding speech to your Basic 7 or Basic 8 files. I hope you like the C128's new 'talking abilities'!

BASIC 8

Programming DigiTalker 128 in Basic 8 programs is extremely simple. This is because the SOUN. files used by the TALK routines are a special Basic 8 data structure file. Because of this, the Basic 8 structure commands, @LSTRUCT (load a structure) and @SSTRUCT (save a structure) allow one easy access to the sounds, and a convenient method of managing memory, since these sound data structures can be mixed in with picture, brush, pattern and logo structures, using standard BASIC 8 structure programming techniques. (For more info on structures, see your BASIC 8 manual.)

Once a sound is in memory as a structure, all you have to do to play the sound from your program is issue a simple SYS command.

```
SYS DEC("0B80"),Structure Number.
```

Here is an example of loading some sounds from drive 8 int a Buffer in a REU, then playing them back.

```
10 @BUFFER,2,0,65535 :REM DEFINE A BUFFER IN THE 1ST REU BANK
20 AD=0
30 @LSTRUCT,0,8,2,AD,"SOUN.A":AD=@SEND
40 @LSTRUCT,1,8,2,AD,"SOUN.B":AD=@SEND
50 @LSTRUCT,2,8,2,AD,"SOUN.C":AD=@SEND
60 BLOAD "TALK.BASIC8",B0 :REM MAKE SURE TALK IS PRESENT
70 FOR I=0 TO 2
80 SYS DEC("0B80"),I :REM PASS THE STRUCTURE NUMBER
90 NEXT I
100 END
```

If you boot with your main DigiTalker disk, one of the options available is to install a Basic 8 RunTime system. With this installed, you can run the various demos on the disk, including a Talking Match Game and a very nifty talking 15 Puzzle! And, there is a program called B8.ALTERSND, which allows you to modify the playing speed of a sound, hear it and then save it back. REMEMBER, use a backup copy of your DigiTalker disk(s) before altering the sounds! And, if you are already a Basic 8 owner, you can just boot with your Basic 8 Editor disk, then load and run the Basic 8 demos on the DigiTalker disk(s).

BASIC 7

For those who want to use DigiTalker 128 in their 80 column Basic 7 programs, there is a second machine language file on the disk called TALK.BASIC7. This must be BLOADed into memory at DEC("0B00"), which can be done by entering

```
BLOAD "TALK.BASIC7",B0
```

from either direct mode or from within your programs. Once it is in memory you can use it to play digitized SOUN. files from any bank of ram you want, including both internal banks (0-1) and upto 8 external 64K ram banks if you have a REU (banks 2-9).

Programming the TALK command in Basic 7 is fairly straighforward, but it does require more work on the part of the programmer than BASIC 8 does. Mostly this is due to the fact that Basic 7 has very little control over the REU, being limited to SWAP, STASH and FETCH. These commands can be used to manipulate data in the normal 128K RAM space into the REU. Once there the TALK command can be used to play the sounds directly out of the REU.

Of course, even if you don't have an REU, you can still use DigiTalker 128, but you are limited to the number of sound files you can have resident at once. And, no matter where you put the

sounds, system ram banks 0 or 1, or the REU, you must know the address of the sounds and the bank they reside in.

Here is the syntax of the Basic 7 TALK command.

SYS DEC("0B00"),Ram Bank#,AdrLo,AdrHi

Ram Bank# - The ram bank containing the sound 0-9, with 2-9 being the REU banks 0-7.

AdrLo - The low byte of the address where the sound is stored. For example, if the sound was stashed away at $C080 (49280 decimal), the low byte is DEC("80"), or 128.

AdrHi - This is the high byte of the address where the sound is found. Using the example above, the high byte would be DEC("C0") or 192.

One thing you must do is set aside a section of ram for your sound buffer. Obviously, you don't want the sounds to overwrite your basic text or variables. One possible buffer is

the 40 column graphic screen area. If you enter a GRAPHIC 1,1:GRAPHIC 5 at the beginning of your program, this will reserve memory addresses 7168-16191 in bank 0 for graphics. Since 80 column mode Basic 7 doesn't use or support graphics (for that you need BASIC 8), you can use this area for your sounds. Just keep in mind that the sounds cannot be longer than 9K (36 disk blocks).

Other techniques that can be used to protect memory are to move the start of basic variables in bank 1 up by poking to memory addresses that control the start of variable storage. Address 47-48 contain the normal start of variables ($0400), but you can change that by having as the first line of your program a POKE to these locations, followed by a CLR. Here is an example, which moves variable start from $0400 to $8000, giving you about 30K of buffer for sound, then loads a sound into memory at $0400 and plays it.

```
10 POKE 47,0:POKE 48,128:CLR
20 BLOAD "SOUN.BASIC8",B1,P(DEC("0400"))
30 SYS DEC("0B00"),1,0,4 : REM Bank#,AdrLo,AdrHi
```

Other memory locations can be very useful in helping you manage memory. For example, if you BLOAD a file into memory, it would be nice to know what the next byte of memory available is,

and how many bytes long the loaded file was. The C128 has some easy ways to determine this. For example, to see what the next available address is after a BLOAD, just PEEK location 174 and 175, like this.

```
BLOAD "some file",b0,P(Address)
NextByte=PEEK(174)+PEEK(175)*256
```

The length of the file is gotten by subtracting the load address from the next byte.

```
Length in Bytes=NextByte-Address
```

Since we must pass the address of the sound in separate bytes, you might ask how can you find the lo and hi bytes easily? My solution was to create function definitions that would return the hi or lo bytes of a 16 bit number.
Here they are.

```
DEF FNHB(n)=INT(n/256)
DEF FNLB(n)=n-(FNHB(n)*256)
```

So, to get the high or low byte of a number, for example, 32768, I would have a line in my program that called the

LIMITED WARRANTY

Free Spirit Software, Inc. warrants that the diskette on which the enclosed program is recorded will be free from defects in materials and workmanship for a period of 90 days from the date of purchase. If within 90 days from the date of purchase, the diskette proves defective in any way, you may return it to Free Spirit Software, Inc., 58 Noble Street, Kutztown, Pa. 19530, and Free Spirit will replace it free of charge.

Free Spirit Software, Inc. makes no warranties, either expressed or implied, with respect to the software program recorded on the diskette or the instructions, their quality, performance, merchantability or fitness for any particular purpose. The program and instructions are sold "as is". The entire risk as to their quality and performance is with the buyer. In no event will Free Spirit Software, Inc. be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the programs or instructions even if Free Spirit Software, Inc. has been advised of the possibility of such damages.

The enclosed software programs and instructions are copyrighted. All rights reserved.

DIGITALKER 128