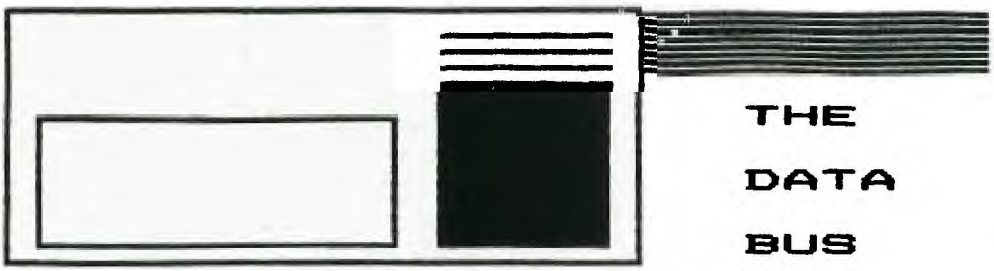
NEXT CHRISTIANA MEETING - DEC. 3, 1987



THE DATA BUS

ISSUE :
VOL. 5
NO. 9
- - - -
OCT/NOV
1987

THE DELAWARE VALLEY USERS GROUP

DEDICATED TO THE TI AND COMPATIBLE HOME COMPUTER FAMILY

P.O. BOX 6240 STANTON BRANCH, WILMINGTON DE 19804 - 0240

USING SPRITES - PART III SPEECH

by Jim Davis

This part of the series deals with the SPEECH SYNTHESIZER accessory. Although we're trying to make the DOG bark, the speech synthesizer can bark too, since humans can imitate dogs barking. For the TI99/4 computer, there are 3 levels on which the speech may be handled:

* RESIDENT VOCABULARY
* ALLOPHONIC SPEECH
* LPC10 (LINEAR PREDICTIVE CODE)

The simplest, using the RESIDENT VOCABULARY, is not a very realistic dog bark, since it depends on the vocabulary in the speech synthesizer. The program below illustrates its use as well as a means to detect the FIRE button on the joystick.

```
100 CALL CLEAR
200 CALL CHAR(96,"050787FAFC
FECA00")
300 CALL SPRITE(#1,96,2,100,
100)
400 CALL MAGNIFY(2)
500 CALL JOYST(1,X,Y)
610 VX=VX * 0.9 + X * 0.5
620 VY=VY * 0.9 - Y * 0.5
630 CALL MOTION(#1,VY,VX)
700 CALL KEY(1,K,S)
710 IF S<>0 THEN 500
720 IF K<>18 THEN 500
730 CALL SAY("ARE ARE")
800 GOTO 500
```

Notice the KEY subroutine is setup to use the option for the lefthand side of the keyboard. The #1 joystick fire button parallels the keyboard fire button, ie. the "Q", which is coded as # 18. The value of S is 0 if the button has not been pressed, +1 if the button was just pressed, and -1 if the button was held down (so called automatic rollover). We want only the fire button to activate the speech, hence all other inputs are ignored. Of course, all of the above could be tailored for joystick #2.

ALLOPHONIC speech is more flexible, since it is created from the basic sounds of a

Continued On Page 7

WANT TO PROGRAM YOUR TI-99/4A IN SUPER-FAST ASSEMBLY LANGUAGE?

We are forming an Assembly Language course of 6 sessions to cover assembly language. The level of the course will depend on the people present.

I would like to go from beginning assembly through advanced programming techniques. (This may take more than one course).

I would hope to have everybody in the class write a small routine or program as a project. The registration fee is $5.00 per person.

If you are interested, please contact Norm Sellers after 7:00 PM at (215)353-0475 or fill in the questions on the sign-up sheet at the DVUG meeting.

The exact time, location and text book are yet to be determined.

You should be able to write a program in BASIC before taking this course.

Super Space II Report

by Norm Sellers

A while back, I bought a Super Space II(tm) Cartridge from DataBioTics, Inc. It is really a nice and unique piece of hardware. It allows you to save your assembly programs in a cartridge complete with an automatic cold start menu facility that allows you to quickly and easily choose which program you wish to run with a single key stroke. It comes with a beautiful set of programs on what it calls the SSLDR (Software Support Loader) Menu. My favorite program on this menu is the TI-Writer with the option for editing programs. It is patched under this option so there is no tab record written when a program is Saved with SF, but the tabs are fixed while editing the program the same as in the Editor/Assembler editor.

I did not plan to write a review at this time. I felt obligated, however, to write of a problem I had in case anybody else should have this problem. The first time I put the Super Space II on the computer, I also loaded my MG

Continued On Next Page

DVUG EXECUTIVE COMMITTEE MEMBERS IN 1987

PRESIDENT	TOM AUGUST
VICE PRESIDENT	JIM DAVIS
SECRETARY	TIM EVERS
TREASURER	TOM KLEIN
SGT. AT ARMS	JIM FOLZ
DELMARVA CHAPTER CHR	CHARLES BOWER
SO.JERSEY CHAPTER CHR	TONY DIFEBBO
SHORE CHAPTER CHR	HARVEY ADAMS

NORMAL MEETING SCHEDULE

CHRISTIANA GROUP	4th Thursday	6:30-9:30
DELMARVA CHAPTER	2nd Monday	7:00-9:00
SOUTH JERSEY CHAPTER	3rd Monday	6:45-9:00
SHORE CHAPTER	1st Thursday	7:30-9:00

MEETING PLACES

CHRISTIANA GROUP: Delaware's Christiana Mall on Rte. 7, at I-95 Exit 4-S. We meet in the Community Room. Enter between J. C. Penney and Liberty Travel inside the Mall.

DELMARVA CHAPTER: Kent County Courthouse, Basement Conference Rm #25, Green & State Sts, Dover, De. Use the Green St. side entrance.

SOUTH JERSEY CHAPTER: Deptford Municipal Bldg, Cooper Ave. and Delsea Drive, (Rtes. 534 & 47), in Gloucester County. Enter and park in rear of the building.

SHORE CHAPTER: Scullville Firehouse #1, County Rte. 559 (on left, between mile markers 4 and 3), in Atlantic County. Ignore Station #2 on right enroute.

DVUG BULLETIN BOARDS

(302)322-3999	Anytime
(302)674-1449	Anytime
(609)429-7792	Monday-Thursday 3:00 PM-7:00 AM
	Friday 3:00 PM-Monday 7:00 AM

For general information, you may contact

TOM KLEIN	Pa.	(215)494-1372
JIM FOLZ	Del.	(302)995-6848
BUTCH FISHER	N.J.	(609)783-8276
GUS LEWIS	N.J.	(609)927-5601

Delaware Valley Users Group membership includes: library and software privileges, monthly DATABUS newsletter, plus other special benefits. Annual membership rates are: Family or Individual $15; Student $10; Newsletter only (beyond 75 mi) $10.

TRANSMIT YOUR NEWSLETTER COPY TO: The Data Bus Editor --- Jim Folz, Telephone (302)995-6848, or use the DVUG mailing address shown on Page One. PLEASE SUBMIT NEWSLETTER ARTICLES FOR AN ISSUE BEFORE THE 2ND THURSDAY OF EACH MONTH.

DVUG ADVERTISING RATES FOR THE DATA BUS:

1/4 page = $ 5/issue, or $ 45/12 issues
1/2 page = $ 8/issue, or $ 75/12 issues
Full page = $15/issue, or $125/12 issues

Continued From Page 1

Explorer intending to see what was in the cartridge when I bought it. I quickly noticed something strange. When I switch the memory window of the Explorer from Hex to Ascii Character Conversion, bytes here and there would change in value in the cartridge. I immediately wrote DataBiotics about this. I waited for a reply from them which never came in 3 months.

Incidentally, the SSLDR and other programs worked beautifully in the default memory bank (the bank you get upon cold start). This made me wonder if something was really wrong or not.

I finally decided to put the cartridge to a test. I loaded all 4 memory banks with high values only to find that I could not even load the SSLDR menu program for about 6 hours after the test. I repeated the test loading all 4 memory banks with low values with the same result. The program I used to do this follows:

```
* SOURCE FILLSSPCE      FILL ALL 4 SUPER SPACE II
* OBJECT FILLSP         **********************
*
        DEF  FILL       MEMORY BANKS WITH >FFFF.
MYREG   BSS  32         **********************
FILL    LWPI MYREG
        LI   R0,1
        BL   @BNKSW
        LI   R0,2
        BL   @BNKSW
        LI   R0,3
        BL   @BNKSW
        LI   R0,4
        BL   @BNKSW
        CLR  @>837C         SET STATUS=0
        BLWP @0
*
BNKSW   LI   R12,>0800      SET CRU ADDR
        LI   R1,2           LOAD SHIFT BIT
        SLA  R0,1           ALIGN BANK NUMBER
        SLA  R1,0           ALIGN SHIFT BIT
        LDCR R1,0           SWITCH BANKS
        SRL  R0,1           RESTORE BANK NUMBER
        LI   R4,>6000
        LI   R5,>FFFF       OR >0000
AGN     MOV  R5,*R4+
        CI   R4,>8000
        JL   AGN
        RT
        END
```

To put low values in the cartridge, I changed the statement:

```
LI R5,>FFFF
```

to:

```
LI R5,>0000
```

The cartridge works beautifully if only one bank is loaded, but when other banks are loaded, the memory changed as a function of time.

CONTENTS OF THE OCTOBER ISSUE OF THE DATA BUS:

BASIC/XBASIC Programming Techniques
by Jack Shattuck Phone: (302) 764-8619

STRING$ AND THINGS ...

Computers long have been seen as used for number crunching; a census-taker is credited with development of the first computer a century ago. But surveys time and again show the biggest use of home computers is for word processing, by more than 5 of 6 computer owners. This month we'll look at some of the functions dealing with variable character strings on the TI-99/4A.

In analyzing these functions, let's use the following: The larger string, A$="COMPUTER". The smaller (sub)string, B$="PUT". N=4, L=3 and X=1. The reason for these variables will be explained below. Here are the major functions with which you should be familiar:

ASC(A$) = the ASCII code of the first character in the string. Using the above variables, here ASC(A$)=67.

CHR$(67) = the reverse process, telling you what character letter equates to ASCII 67. Here, CHR$(67)="C".

DATA marks the program line containing string values, such as 100 DATA COMPUTER, PUT. These values are found by the command to READ A$,B$. Data will be read from the first or next available line that has DATA within it, unless RESTORE [line number] is used to indicate a different line to use for the DATA source. (On other occasions, you also could read numeric data, such as A,B instead of A$,B$.)

LEN(A$) = number of characters in a string. Here, LEN(A$)=8.

POS(A$,B$,X) = what number character within A$ is the position where substring B$ begins. X indicates at which character you start searching. Starting with the first character as our above example suggests, if N=POS(A$,B$,X), then N=4.

SEG$(A$,N,L) = a segment of string A$ beginning at position N, continuing for a length of L characters. Here, SEG$(A$,N,L)="PUT".

STR$(X) = converts a numerical value X into a string which looks like a number, but which can't have numerical functions (such as SQR) applied to it. Here, STR$(X) converts 1 into "1" (i.e., the number one becomes string character "1").

VAL(X$) = opposite function of STR$(X). Here, VAL(X$) would convert the string "1" into the numeric value 1.

All functions discussed above can be used in TI Basic. One other function applicable to strings works only with XBasic. That's RPT$(A$,n), used to extend a string by repeating it n times without a break. That can be accomplished in a somewhat similar fashion in BASIC through a concatenation of A$&A$&A$ etc. n times.

We're not quite finished. Other computers use LEFT$,MID$ and RIGHT$ to obtain a segment (that is, a substring) from the left, middle or right hand part of the main string.

For instance, B$=LEFT$(A$,L) sets B$ equal to the first L characters in the string A$ on another computer. B$=SEG$(A$,1,L) is TI's equivalent.

Other computers' B$=MID$(A$,N,L) is the exact same function as TI's B$=SEG$(A$,N,L), wherein substring B$ starts at position N for a length of L characters.

Other computers' B$=RIGHT$(A$,L) derive substring B$ from the last L characters of A$. TI's equivalent is B$=SEG$(A$,(LEN(A$)-L),L). Here, if L=3, then B$="TER" (the last three letters of COMPUTER).

One other function I'd like to see is the Reverse string display, to print A$ backwards. I'm not aware of a single function to achieve that, but it can be printed or displayed with a For-Next loop. First, in XBasic:

```
1 A$="COMPUTER"
2 X=LEN(A$):: FOR C=1 TO LEN
(A$):: DISPLAY AT(4,C):SEG$(
A$,X,1):: X=X-1 :: NEXT C ::
 END
```

The difference in BASIC (aside from using individual instead of multiple lines) is to substitute CALL HCHAR(4,C+2,ASC(SEG$(A$,X,1))) instead of the "Display At" portion.

So much for what the functions can do. When would one use them? Well, suppose you want to display the letters of the alphabet in groups of seven letters indented on successive lines, for example:

```
 ABCDEFG
  BCDEFGH
   CDEFGHI
```

etc. until you came to TUVWXYZ. (Seven letters were chosen to keep the display within the screen capability of 24 lines, as you'll soon see.) Successive PRINT or DISPLAY AT lines will absorb mucho memory. Instead try:

```
100 A$="ABCDEFGHIJKLMNOPQRST
UVWXYZ"
110 FOR LOCATION=1 TO 20
120 PRINT TAB(LOCATION);
130 PRINT SEG$(A$,LOCATION,7
)
140 NEXT LOCATION
```

(See Herbert D. Peckham, "Programming BASIC with the TI Home Computer", McGraw-Hill, 1979, p.159-160.)

Run it, then change line 110 to read FOR LOCATION=20 TO 1 STEP -1. Then RUN. Change line 110 back to its original version, then add the revised line 110 as line 150. Retype lines 120, 130 and 140 as lines 160, 170 and 180 respectively. Now run it. Does the pattern look familiar? Like a skier slalom? Imagine if you used CALL CHAR to reshape those characters into a landscape, then ran this program ...

There's always someone who insists on practical programming. One common example is in looking for an expected answer to the programmer's question, and comparing the response to the desired answer. This is in the form, IF SEG$(AN$,1,1)<>"Y" THEN ...

Printing text without scrolling is solved in BASIC (which doesn't have DISPLAY AT a particular point) by the CALL HCHAR command with a For-Next loop, such as was shown in the Reverse text example, bringing up part of a string segment at a time.

(Continued on next page)

STRING$ AND THINGS ... (Continued)

How about the continual problem of label-making from a data collection of names and addresses and assorted information whose length varies from case to case, invariably including material too long for a single line? Why should you worry about it when the computer can fix it for you?

In the original TI Mail List, the first disk-based program written in 1980 (in BASIC), the first label line includes a title and an individual's name, but has to fit in all within 32 spaces. Regardless of what length first and last name you may have entered, Mail List chops it into a Title (a maximum of 4 spaces), space, then 27 letters for the name - 12 characters for the first name and 15 for the last name. (See p. 32 of the TI Mail List instruction booklet.) In the program, when Field 13 [F$(13)] is the title, and Field 3 [F$(3)] the name -- last name then first name -- this translates into F$(13)& " "&SEG$(F$(3),16,12)&SEG$(F$(3),1,15) as the line to be printed. (See program lines 3730 and 3750.)

Manipulation of data throughout the Mail List program relies on data segments. What about sorting and lookup routines? Aren't they comparisons of string segment locations?

Replace String functions in TI-Writer's Editor are certainly an important activity, as another example. Don't forget spacing variations for a different length replacement text. Replacement by a text with uniformly altered ASCII code is the basis for simple Cryptograms.

A program in Manning and Inglsbe's "Get Personal with Your TI-99", Oilithium Press, 1984, p.188ff., gives some good examples of string functions, including a conversion of dates from the MM/DD/YY format into YYMMDD (10/01/87 to 871001). Simply multiply the VAL(SEG$(A$,7,2)) [year] (did you remember the "slashes"?) by 10000, VAL(SEG$(A$,1,2)) [month] by 100, and add the remaining VAL(SEG$(A$,4,2)) [day] to get the "digital" date. Don't forget STR$ to convert the number back into a string. Sure is simple when you know how the computer does it; now you know that language and you can program it yourself.

Another use of string functions is to match the segment of a word in a Hangman game. You can figure out the logic for word games if you know the computer's capabilities. Computerized Scrabble scoring also derives from string placement. But word games aren't the only use for string segmentation. If you consider yourself a math whiz, and have been around for awhile, try interpreting Barry Traver's multiple number base conversion program (THE DATA BUS, Vol. 3:6, July, 1985, p.8), which also works on segment manipulation. Then again, graphic programs also use string segmentation to save memory in character definitions. Would you call that use of "word" strings? That's simple computer handling of parts -- bits and bytes -- of building blocks of computer data.

Here's leaving you with a little broader perspective at your computer's role in some fairly frequent string variable handling routines you may have been overlooking.

PRINTER PRATTLE FOLLOWUP FOR NEC/PROWRITER

Last month's column discussed the new Correspondence Quality (CQ) font for the NEC 8023A-C and C.Itoh Prowriter printers. I bought one for my NEC, for $30 plus $5 shipping, and I think its font is a very nice quality, which can fairly be called Near Letter Quality on my 7x9 (alphanumeric) or 8x8 (graphic mode) dot matrix.

The vendor describes it slightly more cautiously as Correspondence Quality, but I think it comes as close as possible under the circumstances.

The CQ, or NLQ, chip replaces the proportional chip, and requires removal of the circuit board and resoldering. Send a SASE beforehand, if you're interested in details, and also to see a print sample. Anyway, it took about one hour for one of my gracious fellow DVUG members, a Du Pont engineer, to accomplish the task for me.

He thoughtfully put the CQ chip onto a socket board, which would allow me to conveniently "unplug" it and replace the old proportional font without resoldering, should I ever need to do so. (It would still take some unplugging and unscrewing of parts, but at least I wouldn't need any special equipment or skill.)

Curtis Josey, Sr., owner of House of Hardware, R.D. #1, Box 227, Burdett, NY 14818, also sends the following information:

"I offer a discount to groups. With an initial order of 3 or more ROMs [chips], I will reduce the price drastically. The $30 ROM [described above] costs $18 each and the $45 ROM [for another version of the printer] is reduced to $24 each. Plus a $5 shipping/handling charge for the whole shipment. The only requirement for the deal is that I ship to 1 address. You can mix various ROMs to get the first order up to 3. Subsequent orders can be as low of a quantity as 1."

(Interested DVUG members should contact Jack Shattuck for more details. - Ed.)

The CQ chip does not change the Pica (80 characters per line) font, nor the Elite (96 characters) font, but rather is a separate font itself, uniform in width for a length of 91 characters per line. Josey writes that rather than stick to a preconceived "conventional" length, the more significant consideration was that it look good when developed. It does.

Readers can also reach JOSEY under that Username on the DELPHI BBS network.

TI Console and Cassette Recorder - $50
David Porter
3111 Winterhaven Dr.
Newark, De. 19702
(302) 737-6852

FOLLOW-UP REMARKS FROM PREVIOUS COLUMNS
by Jack Shattuck

CATCAT

The CATCAT program (Categorical Cataloguer) which I published in the August DATA BUS (Vol. 5:7, p. 10), left it to the reader to insert printer command lines. The version as printed went to the screen alone.

I left it that way due to publishing deadlines. I dislike smug programmers who note casually that readers can easily adapt listings to their own conveniences without explaining how. So here's an addendum to produce a printed CATCAT.

Add these lines:

```
185 DISPLAY AT(15,1):"Use Pr
inter (Y/N)? N"::ACCEPT AT(1
5,20)SIZE(-1)VALIDATE("NYny"
):PR$::IF PR$="Y" OR PR$="y"
 THEN YES=1

195 IF YES<>1 THEN 200::OPEN
 #2:"PIO"::PRINT #2:"Single
Category Catalog of":SEG$(D$
,1,4)&" - Diskname= "&N$

205 IF YES<>1 THEN 210::PRIN
T #2:"Available=";K;"Used=";J
-K:"Filename Size Type P
";"-------- ---- ---- -"

285 IF YES<>1 THEN 290::PRIN
T #2:P$;TAB(12);J;TAB(17);T$
(ABS(A));::IF V(NN,1)<1 THEN
 PRINT #2:TAB(24);"Y" ELSE P
RINT #2:TAB(24);""
```

Change line 340 to read:

```
340 CALL HCHAR(23,1,32,32)::
CLOSE #1::IF YES=1 THEN PRIN
T #2:"Sectors =";TAB(12);SX:
:CLOSE #2::END
```

and move V(NN,1)=A from the start of line 290 to the end of line 280 instead.

Barry Traver has added a note to one of his mailings to caution about Accept At statements. While you may use the SIZE(-1) statement to pick up your default value, it's possible to leave blank the location of the response, and simply use a VALIDATE command to limit responses to those desired. However, notes Barry, the input will still accept a null string (blank space) IN ADDITION TO the validated responses. To avoid unforeseen happenings, he suggests you use this formulation:

```
100 DISPLAY AT(3,1):"REPEAT?
(Y/N)"::ACCEPT AT(3,15)VALID
ATE("NY"):R$ :: IF R$="" THE
N 100
```

Thus a blank input returns you to the question.

STRING$ AND THINGS:

Another use of the SEG$ command allows you to scroll a message across the screen like a ticker tape. Here are two versions of the CRAWL routine, in which T$=text to be used, and D=delay time.

```
100 T$=RPT$(" ",28)&T$::FOR
X=1 TO LEN(T$)::CALL HCHAR(2
,X+1,ASC(SEG$(T$,X,28)))::FO
R J=0 TO D::NEXT J::CALL KEY
(0,K,S)::IF S<>0 THEN 120

110 NEXT X::GOTO 100
```

Also:

```
100 DISPLAY AT(2,1):SEG$(T$,
1,28)

110 FOR D=1 TO 10::NEXT D

120 T$=SEG$(T$,2,LEN(T$)-1)&
SEG$(T$,1,1)::GOTO 100
```

ASGARD SOFTWARE OFFERINGS:

Font Writer II, previously discussed in these pages, was issued as TI/Epson printer compatible only, not designed for the NEC/Prowriter at the present time. The latter printer reverses the location of the Least Significant Bit, and thus has the effect of providing an inverted code feed for graphics. Maybe at a future date ...

Meanwhile, Asgard has issued a new catalog with three new games, three new graphic items, four cookbook offerings, and a powerful macro editor, EZ-Keys, which can be customized for innumerable uses in programming convenience. They now have 8 volumes of TI-Artist Instances, and 4 GRAPHX Companions. All eight Instance volumes can be purchased at a group rate of $49.95, or $26.95 for a set of four ($8.95 each, a new rate).

Although most offerings need 32K, XB and disk drive, the new Missile Wars game (Catlog #FE-04, $5.95) uses cassette and console 16K in XBasic. For further info, contact ASGARD SOFTWARE, P.O. Box 10306, Rockville, MD 20850. Tell 'em DVUG sent you, please.

TI Console, P. E. Box, 32K Memory Expansion, Disk Drive Controller, RS 232/Printer Interface, Internal Drive, External Drive w/Power Supply, Speech Synthesizer, Axiom Printer, Hayes Smartmodem (300 baud only), Joysticks, Cables, Cover, Books, Programs - $600 or will sell by the piece
Bill Schwoer
(609) 652-6557

Call Load To Assembly And Back
by Tom Freeman

This article and the programs that accompany it are another in my intermittant series to help those interested in understanding assembly programs better. You will find XBasic programs that will convert assembly language programs in various formats to other formats, which might have one of two purposes. Either you wish to increase the portability or printability of a program, or you wish to disassemble it to understand what the programmer was doing.

Many programs that use assembly subprograms are published in a "CALL LOAD" format. In other words, the XBasic program directly "pokes" the assembly program into memory, byte by byte. This is done because it might be cumbersome to type in the source code for the assembly program and then assemble it, or you might not have the Editor/Assembler (everyone should, however!). Nevertheless, I have published most of my programs this way. The author might also publish the (uncompressed) DIS/FIX 80 object file, but if you have ever looked at one of these, each line is just a long string of numbers and letters that make no sense, and it would be almost impossible to avoid a typing mistake! The CALL LOAD's, on the other hand, are full of commas and easily read numbers, so typing them in is easier. However, that portion of the program must be run every time the program is run, which takes extra time, so it would be nice to be able to convert them to a real assembly file. Two recent examples of programs that use this method are: "Artist to XB" in Smart Programmer, September 1985 - contains two columns packed with CALL LOAD's, and Improved Unrunnable Basic in Topics, September 1986.

The first program, entitled CL/ASL, that follows this article (I have placed all the programs together, for neatness' sake, so that they could be in 28 column format which looks EXACTLY the way you type it in) provides a method of turning a CALL LOAD XBasic program into either a source code file, which can be run through the Assembler to produce an object file, or an object file directly. Thus there are really two programs here - lines 190-280 could be deleted if you only want to make source code, or 290-350 for only object code. I haven't been able to test this program on LOTS of files, so I suggest you use them both, in case one produces errors. Naturally, I have tried to account for all the errors I could think of! One that cropped up was when the CALL LOAD began with an odd address. Assembly files normally insist on even addresses. I compensated for this by backing up to the even address one lower and beginning with the last byte from the previous line. Try this out with a sample two or three line file to see what I mean. The assembler automatically backs up one byte if the AORG or RORG address is odd, and inserts a zero byte first. This would mess up the code which is why I retained the previous byte.

The only constraints on the input file are that it must 1) be saved in merge format (DIS/VAR 163) not as a program file, 2) contain only CALL LOAD's (delete all other lines and any other statements on the CALL LOAD lines before saving) and 3) only one CALL LOAD(address,byte, byte,...) per line. The program makes heavy use of a knowledge of how the program lines are tokenized. You can see this for yourself by running the last program in this article on a sample small file and comparing the bytes generated with the list of tokens also provided.

I found one interesting quirk in the way TI handles these assembly DIS/FIX 80 files. Normally, the author of a CALL LOAD type program needs to set the REF/DEF table just below 16384 (hex >4000) byte by byte, and then insert the address of the beginning of the table into 8196 (>2004). I originally tried to do this just with ADRGs, but the XB loader just won't insert the bytes there even if the assembly file tells it to! CALL LOAD works fine however. I fixed this up by assuming that all code above 16225 is for the REF/DEF table (this leaves room for 20 DEF's, and it appears that no one ever has actual assembly code at this location) and then actually construct a real DEF table. Then the loader sets the proper address into >2004 by itself.

Now when the file is ready you can replace ALL the CALL LOADs by CALL LOAD("DSK1.YOURFILE") where YOURFILE is whatever you named your DIS/FIX 80 file (produced directly by my program, or assembled from the source code it produced). By the way, I lied a little when I wrote above that the assembly program needs to be reloaded every time you RUN the XB program. When a program is finished, the assembly code remains in memory unless you quit or CALL INIT again. So you can add a line to any such program that "PEEK"s at a couple of bytes that you know the value of (do the peeking after the program is run the first time) and then bypass the CALL INIT and the CALL LOAD if the bytes are what they should be. This works with either method of loading the assembly file (CALL LOAD (dis/fix 80 file) or CALL LOAD(address,bytes)).

By the way, the program takes quite a bit of time to run, especially if the CALL LOAD's are numerous, but at least it only has to be done once!

The second program, entitled ASL/CL, reverses the process. Why would you want to do this? There are two possible reasons: one might be that you have an XB program and wish to publish it, or list it for a friend. Putting the assembly code into CALL LOAD format makes it all readable in one program. Another reason could be that you wish to have the program on tape for someone who has memory expansion but not a disk drive (my son was originally in this position). The program as listed also is a "double" program, as it allows you to construct the CALL LOAD file from a memory range, or directly from a DIS/FIX 80 file. Most object files can be simply loaded from command mode by CALL INIT :: CALL LOAD("DSK1.XXX") and my program then run with the memory range option. (This part of the program runs considerably faster.) WARNING - a few files insert the start address into the ISR hook at >83C4, and will thus auto start. You will need to run the program on the DIS/FIX file directly or use a sector editor to change that value (you would find at the end of the file something like 983C4BXXXX where XXXX is the Start Address - it should be changed to 0000). Please note that the program ends with a statement on the SCREEN that you should type in one or two extra CALL LOAD's. I could have had the program do this, but I didn't get around to it and time is short! (Please note that if the program does use the above auto-start method, then you will need to add one additional CALL LOAD(-31184,x,y) where x and y are the decimal representations of the two bytes following 983C4B above, e.g. if you saw 24F4 then x and y would be 36 and 244.)

Continued On Next Page

If you are going to use the memory range option after loading the DIS/FIX file, there is an additional program that will help you, called ORIGINS. Many object files do not load all the bytes in the whole range of address used, but instead leave some blank, to be used later by the program (this is signalled by the BSS directive in the source code). ORIGINS will search for these breaks - actually it just lists all the origins, and you can see if there are large gaps as normally a single DIS/FIX record can only contain about 22 bytes. You can then specify each memory range separately in the program and not waste a lot of CALL LOAD's unnecessarily.

There is one additional type of assembly file that I haven't mentioned. Some authors have written assembly code, and then "hidden" it in the X8 file, using various methods such as Barry Traver's ALSAVE program. You should suspect this when the X8 program as listed has more sectors than could be accounted for by the number of lines you see, or if you see a CALL LINK or a CALL LOAD(-31804,x,y) when no assembly file was loaded. The program called HIDDEN will search for the area containing the assembly file and inform you of the range. If you save the ASL/CL program in merge format, and then merge it into the hidden program, you can specify the memory range and produce a CALL LOAD file. I must warn you, however, many of these are quite long and would produce a gigantic CALL LOAD file! You would probably be better off in that case to use SAVE to produce a separate program image file and then DISkASSEMBLE it! (See my article in November 1986 Topics to see how to use SAVE).

Finally, the last program is called PRINTMERGE. This will take a MERGE type file and produce a neat listing in compressed format on a printer of each byte of each line and the ASCII representation, if possible, underneath it. You can do this on a few lines of code to see how program lines are tokenized. If you run it on a single CALL LOAD line, for instance, you would find the following: the first two bytes represent the line number (multiply the first by 256 and add the 2nd). The rest of the bytes are the tokens, or strings, and the last is always a 0. After the line number bytes you will find: 157 (CALL), 200 (unquoted string), 4 (length of next string), 76, 79, 65, 68 (L, O, A, D), 183 (left parenthesis), 200 (unquoted string), x (the length of the string), x x x (the actual string, in this case the address to be loaded), etc. Have fun with this one, but DON'T use it on large files, unless you have lots of paper!

Ultimately, my purpose in writing these programs was to be able to disassemble the CALL LOAD's to understand them. What I did was to produce files that DISkASSEMBLER could read. Reversing the process merely became a challenge! Here's hoping you find these programs useful. Enjoy.

SEE PAGES 8,9 FOR PROGRAM LISTINGS. SEE PAGE 10 FOR TOKEN LIST.

FROM PG 1

language (English in this case). However, allophonic speech does not sound as realistic as the resident vocabulary or LPC10 speech. The necessary support programs are available in the TE-II cartridge, the SPEECH EDITOR cartridge, or the TEXT-TO-SPEECH (English) (Model PHD 5076). The latter is necessary for use with EXTENDED BASIC since the cartridge port is already in use. However, the cartridges can be used with TI BASIC for other programs where SPRITES are not used. Using the following program we can try different TEXT until something reasonable comes out. We could also go directly to the table of allophones in the TI documentation with the cartridges/disks:

```
100 REM AFTER THE SUBPROGRAM
S ARE LOADED INTO MEMORY, YO
U CAN DELETE THE REM IN THE
NEXT STATEMENT.
110 REM GOTO 200
120 CALL INIT("DSK1.SETUP","
DSK1.XLAT","DSK1.SPEAK")
130 CALL LINK("SETUP","DSK1.
DATABASE")
200 INPUT "PHRASE= ";X$
210 CALL LINK("XLAT",X$,Y$)
220 CALL LINK("SPEAK",Y$,43,
128")
300 Z$=""
310 FOR I=1 TO LEN(Y$)
320 Z$=Z$&STR$(ASC(SEG$(Y$,I
,1)))&" "
330 NEXT I
340 PRINT Z$
350 GOTO 200
```

You can vary the pitch by changing the parameters in line 220. Changing 43, the pitch period, to a smaller number gives a higher pitch. The second number, the pitch slope, is usually set to 32*INT(pitch period/10). Thus a little dog is 16,32. It is more convenient to change the sound by purposely misspelling the phrases, to force the translator to select different allophones.

The program using allophonic speech is:

```
10 REM GOTO 50
20 CALL INIT("DSK1.SETUP","D
SK1.XLAT","DSK1.SPEAK")
30 CALL LINK("SETUP","DSK1.D
ATABASE")
50 B$=CHR$(250)&CHR$(255)&CH
R$(1)&CHR$(29)&CHR$(105)
100 CALL CLEAR
200 CALL CHAR(96,"050787FAFC
FECA00")
300 CALL SPRITE(#1,96,2,100,
100)
400 CALL MAGNIFY(2)
500 CALL JOYST(1,X,Y)
610 UX=UX * 0.9 + X * 0.5
620 UY=UY * 0.9 - Y * 0.5
630 CALL MOTION(#1,UY,UX)
700 CALL KEY(1,K,S)
710 IF S<>0 THEN 500
720 IF K<>18 THEN 500
730 CALL LINK("SPEAK",B$,16,
32)
800 GOTO 500
```

Note the computer is so busy with the speech systhesizer that the dog stops moving. The subroutine "SPEAK" converts the allophones as noted by there corresponding number into LPC10 code that the synthesizer uses. We'll deal with LPC10 next time.

CL/ASL

```
100 !CONVERT CALL LOAOS TO A
SSEMBLY SOURCE OR OBJECT FIL
E, BY TOM FREEMAN, LA 99ERs
110 BYTE$=" BYTE " :: H$="01
23456789ABCOEF" :: ENO$="8FF
FFF" :: ST$="00" :: VALUE$="
0"
120 OISPLAY AT(2,1)ERASE ALL
:"NAME OF INPUT MERGE FILE?
 OSK1." :: ACCEPT AT(3, 1)SI
ZE(-15)BEEP:I$ :: OPEN #1:I$
,VARIABLE 163,INPUT
130 OISPLAY AT(4,1):"PROOUCE
 OBJECT OR SOURCE     COOE?
(O/S) S" :: ACCEPT AT(5,14)S
IZE(-1)VALIOATE("OS")BEEP:CO
OE$ :: IF CODE$="O" THEN 150
140 OISPLAY AT(7,1):"NAME FO
R OUTPUT SOURCE FILE?DSK2."
:: ACCEPT AT(8,1)SIZE(-15)BE
EP:O$ :: OPEN #2:O$ :: GOTO
160
150 OISPLAY AT(7,1):"NAME FO
R OUTPUT OBJECT FILE?OSK2."
G: ACCEPT AT(8,1)SIZE(-15)BE
EP:O$ :: OPEN #2:O$,FIXEO
160 OISPLAY AT(10,1)BEEP:"IN
PUT FILE MUST CONTAIN ONLY'C
ALL LOAO'S, AND ONLY ONE  PE
R PROGRAM LINE"
170 OISPLAY AT(14,1):"RELOCA
TABLE/ABSOLUTE?(R/A) R" :: A
CCEPT AT(14,28)SIZE(-1)VALIO
ATE("RA")BEEP:R$ :: IF R$="R
" THEN RFLAG=1
180 IF CODE$="S" THEN 250
190 PRINT #2
200 LINPUT #1:C$ :: IF LEN(C
$)=2 THEN PRINT #2:":" :: GO
TO 280
210 GOSUB 360 :: GOSUB 380 :
: RCOOE=MAX(RCOOE,Q*N)
220 IF N>16225 THEN GOSUB 42
0 :: GOTO 200
230 GOSUB 480 :: CALL HEX(AO
OR$,HEX$,H$,OFLAG):: PRINT #
2:CHR$(57+Q*8)&HEX$;
240 IF OFLAG THEN PRINT #2:"
B";ST$;:: BFLAG=1 ELSE BFLAG
=0
250 C$=SEG$(C$,L+4,155):: IF
 ASC(C$)=182 THEN IF BFLAG=1
 THEN PRINT #2:"00";ENO$ ::
ST$=HEX$ :: GOTO 200 ELSE PR
INT #2:ENO$ :: GOTO 200
260 L=ASC(SEG$(C$,3,1)):: VA
LUE$=SEG$(C$,4,L):: CALL HEX
1(VALUE$,HEX$,H$):: IF BFLAG
=0 THEN PRINT #2:"B";:: BFLA
G=1 ELSE BFLAG=0
270 RCOOE=RCOOE+Q :: PRINT #
2:HEX$;:: GOTO 250
280 RCOOE$=STR$(RCOOE):: CAL
L HEX(RCOOE$,HEX$,H$,O):: RE
STORE #2 :: PRINT #2:"O";HEX
$;RPT$(" ",8);END$ :: CLOSE
#1 :: CLOSE #2 :: STOP
290 LINPUT #1:C$ :: IF LEN(C
$)=2 THEN PRINT #2:" ENO" ::
 CLOSE #1 :: CLOSE #2 :: STO
P
300 GOSUB 360 :: GOSUB 380
310 IF N>16225 THEN GOSUB 42
0 :: GOTO 290
320 GOSUB 480 :: PRINT #2:"
  ";CHR$(65+Q*17);"ORG ";N-O
FLAG
330 IF OFLAG THEN PRINT #2:B
YTE$;VALUE$
340 C$=SEG$(C$,L+4,155):: IF
 ASC(C$)=182 THEN 290
350 L=ASC(SEG$(C$,3,1)):: VA
LUE$=SEG$(C$,4,L):: PRINT #2
:BYTE$;VALUE$ :: GOTO 340
360 C$=SEG$(C$,10,155):: IF
SEG$(C$,2,1)=CHR$(194)THEN M
FLAG=1 :: C$=SEG$(C$,2,155)E
LSE MFLAG=0
370 RETURN
380 L=ASC(SEG$(C$,3,1)):: N=
VAL(SEG$(C$,4,L))
390 Q=ABS(RFLAG*(MFLAG=0)*(N
>9459 AND N<16226)):: IF MFL
AG THEN N=-N
400 N=N-Q*9460 :: ADDR$=STR$
(N)
410 RETURN
420 NAME$=""
430 FOR X=1 TO 8 :: C$=SEG$(
C$,L+4,155):: L=ASC(SEG$(C$,
3,1)):: VALUE$=SEG$(C$,4,L):
: NAME$=NAME$&CHR$(VAL(VALUE
$)):: NEXT X
440 V1=ASC(SEG$(NAME$,7,1)):
: V2=ASC(SEG$(NAME$,8,1)):: 
NAME$=SEG$(NAME$,1,6):: N=V1
*256+V2 :: IF N>32767 THEN M
FLAG=1 ELSE MFLAG=0
450 GOSUB 390 :: CALL HEX(AD
OR$,HEX$,H$,O)
460 IF CODE$="O" THEN PRINT
#2:CHR$(54-Q);HEX$;NAME$;ENO
$ ELSE PRINT #2:" OEF ";NAME
$;"  EQU ";N+Q*9460
470 IF ASC(SEG$(C$,L+4,155))
-182 THEN RETURN ELSE 420
480 IF INT(N/2)=N/2 THEN OFL
AG=0 ELSE OFLAG=1
490 RETURN
500 SUB HEX1(VALUE$,HEX$,H$)
510 N=VAL(VALUE$)
520 U=INT(N/16):: L=N-16*U
530 HEX$=SEG$(H$,U+1,1)&SEG$
(H$,L+1,1)
540 SUBEND
550 SUB HEX(AODR$,HEX$,H$,OF
LAG)
560 HEX$=""
570 ADOR=VAL(ADOR$)-OFLAG ::
 AODR=ADDR-65536*(AODR<0)
580 FOR X=1 TO 4 :: P=16^(4-
X):: U=INT(ADOR/P):: ADOR=AD
DR-P*U :: HEX$=HEX$&SEG$(H$,
U+1,1):: NEXT X
590 SUBENO
```

ASL/CL

```
100 !CONVERT MEMORY OR OIS/F
IX80 FILE TO MERGEABLE CALL
LOAO FILE, BY TOM FREEMAN, L
A 99ER5
110 OC$=CHR$(179)&CHR$(200):
: CC$=CHR$(182)&CHR$(0):: H$
="0123456789ABCOEF" :: A1,A2
-9460 :: OEFA00-16384
120 OISPLAY AT(3,1)ERASE ALL
:"CONVERT TO ""CALL LOAOS"""
:"        by Tom Freeman":"CH
OOSE":"  1. MEMORY RANGE":"
 2. OIS/FIX 80 FILE       1"
130 ACCEPT AT(7,28)SIZE(-1)V
ALIOATE("12")BEEP:CH$ :: IF
CH$="2" THEN 260
140 GOSUB 560
150 OISPLAY AT(11,1):"NEXT T
WO #'S MUST BE OECIMALLAST W
ILL BE included": :"FIRST AO
ORESS ";A1:"LAST  AOORESS ";
A2
160 ACCEPT AT(14,15)SIZE(-6)
VALIOATE(OIGIT," -")BEEP:A1
:: A1=2*INT(A1/2)
170 ACCEPT AT(15,15)SIZE(-6)
VALIOATE(OIGIT," -")BEEP:A2
180 OISPLAY AT(16,1):"CORREC
T? (Y/N) Y" :: ACCEPT AT(16,
16)SIZE(-1)VALIOATE("YN")BEE
P:Y$ :: IF Y$="N" THEN 150
190 GOSUB 490
200 OISPLAY AT(24,1):"DO MOR
E IN THIS FILE?(Y/N) N"
210 ACCEPT AT(24,28)SIZE(-1)
VALIOATE("YN")BEEP:M$ :: IF
M$="Y" THEN 150
220 OISPLAY AT(24,1):"# BYTE
S IN OEF TABLE?    O"
230 ACCEPT AT(24,26)SIZE(-3)
VALIOATE(OIGIT)BEEP:BY
240 IF BY THEN A2=16384 :: A
1=A2-BY :: GOSUB 490 :: OEFA
00=16384-BY :: GOTO 540
250 PRINT #1:CHR$(255);CHR$(
255):: CLOSE #1 :: STOP
260 GOSUB 570 :: GOSUB 560
270 LINPUT #2:A$ :: RELOC$=S
EG$(A$,2,4):: CALL OEC(RELOC
$,RELOC):: A$=SEG$(A$,14,80)
280 T$=SEG$(A$,1,1):: IF T$=
":" THEN 530 ELSE P=POS(H$,T
$,1):: ON P-1 GOTO 290,290.3
00,300,310,320,330,330,340,3
50,360,370,5
290 CALL WARN1 :: A$=SEG$(A$
,6,80):: GOTO 280
300 CALL WARN2 :: A$=SEG$(A$
,12,80):: GOTO 280
310 RLFLAG=1 :: GOTO 390
320 RLFLAG=0 :: GOTO 390
330 GOSUB 480 :: LINPUT #2:A
$ :: GOTO 280
340 RLFLAG=0 :: GOTO 430
350 RLFLAG=1 :: GOTO 430
360 RLFLAG=0 :: GOTO 450
370 RLFLAG=1 :: GOTO 450
380 CALL WARN3
390 OEFADD=DEFA00-8 :: GOSUB
 480
400 LN=LN+10 :: CALL START(L
N,OEFAOD):: GOSUB 460 :: GOS
UB 470
410 NAME$=SEG$(A$,6,6):: FOR
 X=1 TO 6 :: N=ASC(SEG$(NAME
$,X,1)):: N$=STR$(N):: PRINT
 #1:OC$;CHR$(LEN(N$));N$;::
NEXT X :: A$=SEG$(A$,12,80)
420 PRINT #1:OC$;CHR$(U);U$;
OC$;CHR$(L);L$;CC$ :: GOTO 2
80
430 GOSUB 460 :: LN=LN+10 ::
 GOSUB 480 :: CALL START(LN,
AOOR):: PFLAG=1
440 A$=SEG$(A$,6,80):: GOTO
280
450 GOSUB 460 :: GOSUB 470 :
: PRINT #1:OC$;CHR$(U);U$;OC
$;CHR$(L);L$;:: GOTO 440
460 AOOR$=SEG$(A$,2,4):: CAL
L OEC(AOOR$,AOOR):: AOOR=AOO
R+RLFLAG*9460 :: RETURN
470 ADOR=AOOR-65536*(AOOR<0)
:: U=INT(AOOR/256):: L=AOOR-
256*U :: U$=STR$(U):: L$=STR
```

```
$(L):: U=LEN(U$):: L=LEN(L$)
:: RETURN
480 IF PFLAG THEN PRINT #1:C
C$ :: PFLAG=0 :: RETURN ELSE
 RETURN
490 FOR X=A1 TO A2 STEP 22 :
: LN=LN+10 :: CALL START(LN,
X)
500 FOR Y=0 TO 21 :: IF X+Y>
A2 THEN 520
510 CALL PEEK(X+Y,A):: A$=ST
R$(A):: L=LEN(A$):: PRINT #1
:OC$;CHR$(L);A$;:: NEXT Y
520 PRINT #1:CC$ :: NEXT X :
: RETURN
530 CLOSE #2 :: IF RELOC THE
N ADDR=RELOC+9460 :: GOSUB 4
70 :: DISPLAY AT(16,1)BEEP:"
REMEMBER TO ADD":"  CALL LOA
D(8194,";U$;",";L$;")"
540 IF DEFADD<16384 THEN ADD
R=DEFADD :: GOSUB 470 :: DIS
PLAY AT(19,1)BEEP:"REMEMBER
TO ADD":"  CALL LOAD(8196,";
U$;",";L$;")"
550 GOTO 250
560 DISPLAY AT(9,1):"OUTPUT
FILE? DSK1." :: ACCEPT AT(9,
14)SIZE(-15)BEEP:O$ :: OPEN
#1:O$,VARIABLE 163,OUTPUT ::
 RETURN
570 DISPLAY AT(8,1):"INPUT
FILE? DSK1." :: ACCEPT AT(8,
14)SIZE(-15)BEEP:I$ :: OPEN
#2:I$,INPUT ,FIXED :: RETURN
580 SUB START(LN,X)
590 A=INT(LN/256):: B=LN-256
*A :: Y=ABS(X):: Y$=STR$(Y):
: L=LEN(Y$)
600 PRINT #1:CHR$(A);CHR$(B)
;CHR$(157);CHR$(200);CHR$(4)
;"LOAD";CHR$(183);
610 IF X<0 THEN PRINT #1:CHR
$(194);
620 PRINT #1:CHR$(200);CHR$(
L);Y$;
630 SUBEND
640 SUB DEC(A$,A):: A=0
650 L=LEN(A$):: FOR X=1 TO L
 :: A1=ASC(SEG$(A$,X,1)):: A
2=A1-48+7*(A1>57):: A=A+A2*1
6^(L-X):: NEXT X
660 A=A+65536*(A>32767):: SU
BEND
670 SUB WARN1 :: DISPLAY AT(
22,1)BEEP:"WARNING! FILE CON
TAINS AUTO START AND MAY NOT
 BE XBASIC   PRESS ANY KEY T
O CONTINUE"
680 CALL PRESS :: SUBEND
690 SUB WARN2 :: DISPLAY AT(
22,1)BEEP:"WARNING! FILE CON
TAINS EXT  REF'S AND MAY NOT
 BE XBASIC   PRESS ANY KEY T
O CONTINUE"
700 CALL PRESS :: SUBEND
710 SUB WARN3 :: DISPLAY AT(
22,1)BEEP:"WARNING! FILE CON
TAINS A BADTAG! PROGRAM ABOR
TED"
720 PRINT #1:CHR$(255);CHR$(
255):: CLOSE #1 :: CLOSE #2
:: STOP :: SUBEND
730 SUB PRESS
740 CALL KEY(0,K,S):: IF S=0
 THEN 740
750 SUBEND
```

(Note that line 280 can send you to non-existent line 5. I called Tom Freeman and he reported no problems with the program, as written. I suggest you add a line 5 to stop execution with a note to remind you. You will also need a line 1 GOTO 100. Ed.)

ORIGINS

```
100 !DETERMINE ORIGINS OF A
D/F 80 FILE, I.E.ADDRESS RAN
GE LOADED, BY TOM FREEMAN, L
A 99'ERS
110 DEF$="56" :: ER$="78" ::
 ORG$="9A" :: DAT$="BC"
120 INPUT "NAME OF DIS/FIX 8
0 FILE TO  BE ANALYZED ":F$
:: OPEN #1:F$,FIXED,INPUT
130 INPUT "PRINTER? (PRESS E
NTER FOR   SCREEN DISPLAY) "
:P$ :: IF P$<>"" THEN P=2 ::
 OPEN #P:P$,VARIABLE 136 ::
PRINT #P:CHR$(15);
140 LINPUT #1:A$ :: RL$=SEG$
(A$,2,4):: CALL DEC(RL$,RL):
: PRINT #P:RL;"BYTES OF RELO
CATABLE CODE"
150 A$=SEG$(A$,14,80):: GOTO
 170
160 LINPUT #1:A$ :: L=L+1 ::
 PRINT :"RECORD";L;
170 I$=SEG$(A$,1,1):: IF I$=
":" THEN 250 ELSE IF POS(ER$
,I$,1)THEN 160
180 IF POS(DAT$,I$,1)THEN A$
=SEG$(A$,6,80):: BSFLAG=0 ::
 GOTO 170
190 ORG=POS(ORG$,I$,1):: IF
ORG=0 THEN 240
200 R=(ORG=2):: GOSUB 290
210 IF BSFLAG THEN PRINT #P:
"BSS";
220 IF ORG=2 THEN PRINT #P:"
R";
230 PRINT #P:AD;:: BSFLAG=1
:: A$=SEG$(A$,6,80):: GOTO 1
70
240 RDEF=POS(DEF$,I$,1):: IF
 RDEF=0 THEN PRINT #P:"ERROR
-NOT AN XB DF80 FILE" :: GOT
O 270
250 DT=DT+8 :: R=(RDEF=1)::
GOSUB 290 :: PRINT #P:"DEF "
;SEG$(A$,6,6);AD :: A$=SEG$(
A$,12,80):: GOTO 170
260 PRINT #P:" ";DT;"BYTES I
N DEF TABLE"
270 CLOSE #1 :: IF P THEN CL
OSE #P
280 STOP
290 AD$=SEG$(A$,2,4):: CALL
DEC(AD$,AD):: AD=AD-9460*R :
: RETURN
300 SUB DEC(A$,A):: A=0
310 FOR X=1 TO 4 :: A1=ASC(S
EG$(A$,X,1)):: A2=A1-48+7*(A
1>57):: A=A+A2*16^(4-X):: NE
XT X
320 A=A+65536*(A>32767):: SU
BEND
```

HIDDEN

```
32760 !QUICKLY DETERMINE THE
 ADDRESS RANGE OF A HIDDEN A
SSEMBLY PROGRAM IN AN XB PRO
GRAM, BY TOM FREEMAN, LA 99E
RS
32761 !MERGE THIS FILE INTO
THE XB FILE, THEN RUN 32762
32762 CALL PEEK(-31952,21,22
,23,24):: Z5=21*256+22-65536
 :: Z6=23*256+24-65536
32763 Z4=-32700 :: FOR Z7=Z5
+2 TO Z6-1 STEP 4 :: CALL PE
EK(Z7,21,22):: Z3=21*256+22-
65536 :: Z4=MAX(Z4,Z3):: NEX
T Z7
32764 CALL PEEK(Z4-1,Z1):: Z
4=Z4+Z1-1
32765 PRINT "PROGRAM RUNS FR
OM ";Z4;"TO -25"
```

PRINTMERGE

```
100 CALL CLEAR :: INPUT "NAM
E OF INPUT MERGE FILE?   ":F
$
110 OPEN #1:F$,VARIABLE 163
120 OPEN #2:"PIO",VARIABLE 1
32 :: PRINT #2:CHR$(15)
130 DIM A$(163)
140 LINPUT #1:B$ :: L=LEN(B$
)
150 FOR X=1 TO L :: A$(X)=SE
G$(B$,X,1):: NEXT X
160 FOR X=1 TO 33 :: IF X+Y>
L THEN 180 ELSE B=ASC(A$(X+Y
)):: PRINT #2:STR$(B);TAB(X*
4+1);
170 NEXT X
180 PRINT #2
190 FOR X=1 TO 33 :: IF X+Y>
L THEN 230
200 B=ASC(A$(X+Y)):: IF B<32
 OR B>126 THEN 210 ELSE PRIN
T #2:CHR$(B);
210 PRINT #2:TAB(X*4+1);
220 NEXT X
230 PRINT #2 :: IF X+Y<=L TH
EN Y=Y+33 :: GOTO 160
240 LINPUT #1:B$ :: IF EOF(1
)THEN 250 ELSE Y=0 :: L=LEN(
B$):: GOTO 150
250 CLOSE #1 :: CLOSE #2
```

SPECIAL NOTICE

NO NOVEMBER CHRISTIANA MEETING

DUE TO HOLIDAYS

NEXT MEETING

DECEMBER 3, 1987

NEW	0	00	)	182	B6	IMAGE	163	A3	ERASE	239	EF
CONTINUE/CON	1	01	(	183	B7	ACCEPT	164	A4	AT	240	F0
LIST	2	02	&	184	B8	ERROR	165	A5	BASE	241	F1
BYE	3	03	OR	186	BA	WARNING	166	A6	VARIABLE	243	F3
NUMBER/NUM	4	04	AND	187	BB	SUBEXIT	167	A7	RELATIVE	244	F4
OLD	5	05	XOR	188	BC	SUBEND	168	A8	INTERNAL	245	F5
RESEQUENCE/RES	6	06	NOT	189	BD	RUN	169	A9	SEQUENTIAL	246	F6
SAVE	7	07	=	190	BE	LINPUT	170	AA	OUTPUT	247	F7
MERGE	8	08	<	191	BF	THEN	176	B0	UPDATE	248	F8
DEL	9	09	>	192	C0	TO	177	B1	APPEND	249	F9
COPY	10	0A	+	193	C1	STEP	178	B2	FIXED	250	FA
MOVE	11	0B	-	194	C2	,	179	B3	TAB	252	FC
ELSE	129	81	*	195	C3	;	180	B4	#	253	FD
::	130	82	/	196	C4	:	181	B5	VALIDATE	254	FE
!	131	83	^	197	C5						
IF	132	84	EOF	202	CA						
GO	133	85	ABS	203	CB						
GOTO	134	86	ATN	204	CC						
GOSUB	135	87	COS	205	CD						
RETURN	136	88	EXP	206	CE						
DEF	137	89	INT	207	CF						
DIMENSION/DIM	138	8A	LOG	208	D0						
END	139	8B	SGN	209	D1						
FOR	140	8C	SIN	210	D2						
LET	141	8D	SQR	211	D3						
BREAK	142	8E	TAN	212	D4						
UNBREAK	143	8F	LEN	213	D5						
TRACE	144	90	CHR$	214	D6						
UNTRACE	145	91	RND	215	D7						
INPUT	146	92	SEG$	216	D8						
DATA	147	93	POS	217	D9						
RESTORE	148	94	VAL	218	DA						
RANDOMIZE	149	95	STR$	219	DB						
NEXT	150	96	ASC	220	DC						
READ	151	97	PI	221	DD						
STOP	152	98	REC	222	DE						
DELETE	153	99	MAX	223	DF						
REM	154	9A	MIN	224	E0						
ON	155	9B	RPT$	225	E1						
PRINT	156	9C	NUMERIC	232	E8						
CALL	157	9D	DIGIT	233	E9						
OPTION	158	9E	UALPHA	234	EA						
OPEN	159	9F	SIZE	235	EB						
CLOSE	160	A0	ALL	236	EC						
SUB	161	A1	USING	237	ED						
DISPLAY	162	A2	BEEP	238	EE						

NOTES:

1 In addition to the list of tokens above-

Quoted String (follows)	199	C7
Unquoted String (follows)	200	C8
Line Number (2 bytes follow)	201	C9

2 This is Danny Michaels XB for GK. There are a few tokens not in TI XB.

SPECIAL NOTICE

NO NOVEMBER CHRISTIANA MEETING!

DUE TO HOLIDAYS

NEXT MEETING

DECEMBER 3, 1987

DELAWARE VALLEY USERS' GROUP
P. O. BOX 6240
STANTON BRANCH
WILMINGTON, DE 19804-0840

Subscription expires with date on label

FIRST CLASS MAIL

TO:

0000 EDMONTON 99'ER ONLINE
P.O. BOX 11983
EDMONTON ALB. CANADA T5J 3L1