

4051, 4631 and Zonic DMS 5003 teamwork results in easy-to-read graphs of analog data.

Tektronix-Zonic Fast Fourier Transform (FFT), Data Acquisition and Analysis System

by L. D. Mitchell
Virginia Polytechnic Institute and State University

Virginia Polytechnic Institute and State University has assembled a very powerful data acquisition and analysis system, using the TEKTRONIX 4051 Graphic System as a key component. Working with Tektronix and Zonic Technical Laboratories, the two-function system shown above was configured. This system's first function is easy acquisition of real-world analog data and conversion of that data into digital information understood by either microprocessors or large scale computers. The system also locally processes and analyzes time domain data. The 4051 Graphic System and the Zonic Technical Laboratories High Speed Two Channel FFT share this second function.

The Zonic unit performs any one of thirty-three frequency domain type functions. Statistical data is handled

expeditiously, because the forward Fourier transforms are computed in 50 milliseconds. Once the Zonic unit completes the requested frequency domain operation, the 4051 display plots the graphic result, in the form of a log-

In This Issue

4051 Goes to Chemistry Class.....	2
Editor's Note.....	3
Membership Card.....	3
Graphics in Calculus.....	4
Programming Tips.....	4
Basic Bits.....	8
New Abstracts.....	9
Library Addresses.....	12


log, log-linear, or linear-linear graph, labeled in engineering units. Typical of such results might be a graph of the frequency response of a physical system, in both magnitude and phase. Any graphic display can be copied using the TEKTRONIX 4631 Hard Copy Unit included in the system.

Besides displaying graphic results, the 4051 is also used to handle analyses beyond the Zonic unit's capability. This can include data analysis that is outside the domain of the preprogrammed Zonic functions, and/or data analysis needs that go beyond the batch programming capability of the Zonic unit. For these cases, the system transfers either the data or a pre-analyzed frequency domain result to the 4051's cartridge tape unit. The data can then be analyzed in any way desired, using the programmable capability of the TEKTRONIX 4051 Graphic System.

As an example, consider amplitude time history data. This data cannot be analyzed for its amplitude probability density function by the Zonic High Speed FFT system.

However, the data can be transferred to the 4051 Graphic System. This allows existing Plot 50 software, such as that in Statistics Volume 1, to carry out that analysis.

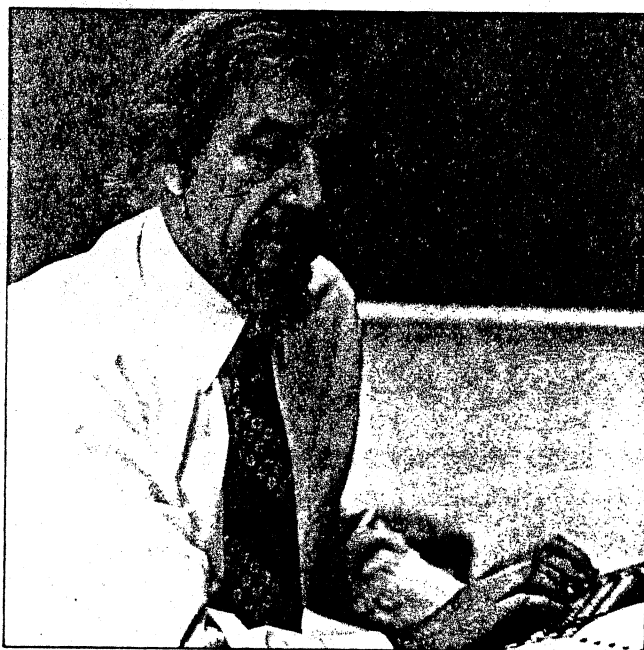
Moreover, the 4051 can be programmed to operate on the data in any way desired. If the particular analysis requires a lot of number crunching, the raw data and/or the pre-analyzed results may be transmitted out through the Option 1 RS-232 Interface to a large-scale computer. Virginia Polytechnic Institute and State University uses an IBM 370 configured with two model 158s. One machine is in batch operation with five megabytes of core; the other machine is interactive and has four megabytes of core.

This configuration of two interlinked microprocessor-based systems, which are in turn linked to a large-scale computer, yields great flexibility in data analysis. This system also capitalizes on the inherent graphics capability of the 4051 Graphic System for the presentation of analysis results. 

Binary Liquid-Vapor Boiling Point Diagrams

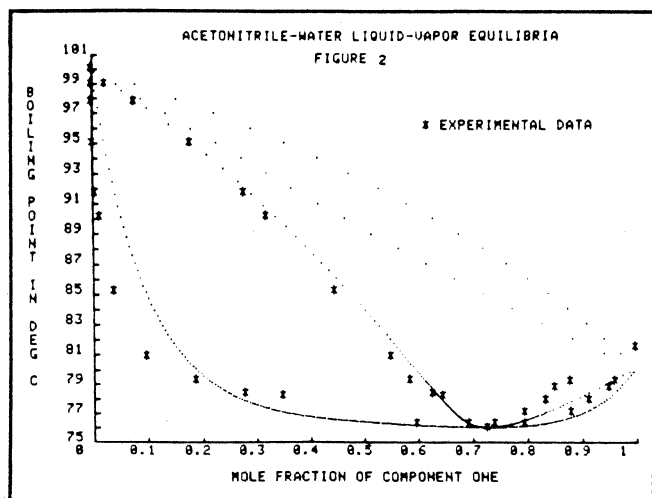
An interactive graphic program is now available, written in 4051 BASIC, that produces diagrams of liquid-vapor boiling points. The program computes and plots the liquid and vapor composition curves as a function of temperature for any volatile two-component system. Dr. Gilbert F. Pollnow of the University of Wisconsin-Oshkosh developed the program with the support of both a unique faculty development funding program and a National Science Foundation grant. The NSF funding enabled the purchase of three TEKTRONIX 4051 Graphic Systems with 32K memory, along with a Hard Copy Unit. This was all part of an interdisciplinary grant involving Chemistry, Physics, Geology and Geography.

The Binary Liquid-Vapor Boiling Point Diagrams program plots both the ideal and Van Laar representations for the liquid system. On the same plot is the experimental boiling point composition data; a sample plot is shown in Fig. 2. The points are plotted as they are generated, emphasizing the discrete and conjugate nature of the experimental and theoretical data. The program computes the Van Laar constants using the method described in F. Daniels' *Experimental Physical Chemistry*, which requires the composition and boiling point of the azeotrope, barometric pressure, and the Antoine vapor pressure constants for both pure components. Tabular output of the liquid and vapor compositions (Fig. 3) is optional. As a tutorial, the program is set up to graph literature data for the acetonitrile-water system if it is executed without modifying the parameters.



4051 graphics relieve tedium and enhance understanding of complex chemistry problems in Dr. Gilbert F. Pollnow's class at the University of Wisconsin.

At UWO physical chemistry students are required to calculate the Van Laar constants and the conjugate compositions corresponding to one of the tabular output points on their personal calculators for a known system. This reinforces their understanding of the computations prior to performing the laboratory experiment as described in the Daniels text. In addition, the Van Laar constants are also determined by the students using the least squares method, and attempts are made to rationalize the differences. This experiment also provides an excellent



opportunity to demonstrate the use of a software polynomial regression program for plotting and fitting their refractive index composition data and then using the resulting polynomial to compute the conjugate compositions from their measured refractive indices.

Dr. Pollnow has been involved with computer science and its practical applications on the university scene since 1962. Enabled by university funding and the NSF grant, his recent endeavors focus upon improving and increasing computer use to solve ordinary chemical problems which would otherwise be too tedious or complex to include in the undergraduate program.

FIGURE 3

VAN LAAR TEMPERATURE-COMPOSITION RESULTS

T	X1L	X1U	X2L	X2U
99.87	0.0005	0.0052	0.9995	0.9948
84.74	0.1005	0.4828	0.8995	0.5172
79.55	0.2005	0.6058	0.7995	0.3942
77.53	0.3005	0.6529	0.6995	0.3471
76.73	0.4005	0.6742	0.5995	0.3259
76.37	0.5005	0.6868	0.4995	0.3132
76.15	0.6005	0.6991	0.3995	0.3009
76.01	0.7005	0.7187	0.2995	0.2813
76.12	0.8005	0.7559	0.1995	0.2441
76.99	0.9005	0.8307	0.0995	0.1693

THE VAPOR PRESSURE CONSTANTS USED TO COMPUTE THE ABOVE RESULTS ARE:

A1 = 7.11989 B1 = -1314.40 C1 = 230.00
 A2 = 7.96681 B2 = -1668.21 C2 = 228.00

PRESSURE IN MM = 760.0
 MOLE FRACTION (ONE) IN AZEOTROPE = 0.7268
 BOILING POINT OF THE AZEOTROPE IN DEG C = 76.88
 VAN LAAR CONSTANTS A1 AND A2 ARE 0.7669 0.7557

From the original 1963-64 room-sized IBM machine to the new desk-top 4051 Graphic systems, a gigantic leap in the capacity of the machine, as well as accessibility for the layman, has been achieved. The graphic ability of the 4051 display screen has made possible a host of solutions hitherto unavailable to most computer users. Using this advantage, Dr. Pollnow has created a number of autotutorial program modules for his students in physical chemistry classes. The popularity of such applications is evidenced by requests from scholars from both sides of the world for copies of his programs. Copies of this program with sample output are available from Dr. Pollnow upon request.

* EDITOR'S NOTE

New Catalog

You will notice a new catalog of 4051 Applications Library programs enclosed with this issue of TEKniques. This catalog contains several new program categories and lots of new programs. Look through it. Additional copies are available for the asking. The 4051 Applications Library staff thanks all of you who contributed to the Library and made these additional programs available.

Contest Finale

By the time you read this, all of the entries in the Computer-Aided Design contest have been received and judged. Contest results will appear in the July issue of TEKniques; watch for the winning programs!

51/00-8004/0 Documentation Contains "Typo"

In the Cross Reference and List Program Variables program there are typographical errors on page 4 of the documentation. Under *Operating Hints*, all references to "FS" should be changed to "F5", and "JS" to "J5".

Hopefully this has not caused anyone anguish who purchased this program. Our thanks to John Williams of GTE Automatic Electric for pointing this out.

Dashed Lines Combined

Three subroutines for different methods of drawing dashed lines have been combined in a new Applications Library program entry. Check out Abstract No. 51/00-9508/1 in this issue.

Membership Card Enclosed

It's free. Other than sunshine and air, not too many things are. But membership in the 4051 Applications Library is free for 4051 owners and users. Pass the enclosed membership cards along to those who borrow your TEKniques. They'll get their own copy of these bits of wisdom and your newsletter remains intact. Plus, more members mean more ideas—we'll all benefit. If you need more cards, call Rory Gugliotta at (503) 682-3411, ext. 2618. Sign up those friends of the '51.

Using Graphics Technology in Calculus


by Edwin T. Hoefler
Rochester Institute of Technology

In many areas of science, a graphics demonstration in the classroom can clarify the instructor's complex point. The Rochester Institute of Technology uses the 4051 Graphic System to provide graphic classroom illustration of fundamental functions in calculus. For instance, the 4051 can draw on the screen the interesting curves that are analyzed in a traditional calculus class. A particular technique of plotting the graphic display is used to dramatically illustrate the concept of "rate of change" of a function.

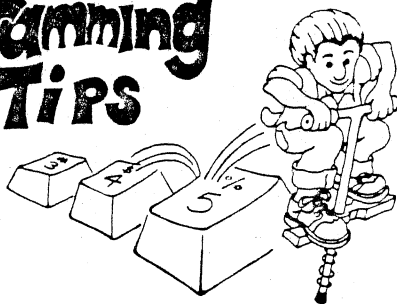
The 4051 draws curves by small line segments, usually controlled by a FOR-NEXT, so that the x-axis increments are of constant size. This causes dx/dt to remain constant. Thus, the rate of change in the y-axis increments

of the curve, $dy/dt = dy/dx$. The ratio dx/dt is directly proportional to dy/dx , so the rate at which the curve is actually drawn on the display is a visual demonstration of the rate of change in dy/dt .

Consider the familiar examples of $y = x^2$ and $y = x^3$. As these curves were drawn on the display, the class observed that it takes longer to draw the points close to (0,0) than to draw the rest of the curve. This corresponds to the fact that $dy/dt = 0$ at the point (0,0). They also clearly saw that the larger the value of dy/dt , the faster the curve is drawn.

Thus, the 4051 display graphically illustrated the calculus concept of "rate of change" for the class, through calculating and drawing the curves. 

Programming TIPS



Overlays Expand Capability

by Jay Beck

How do you run a program larger than the memory capacity of your machine? Try the overlay technique using the 4051 memory management capability combined with the 4907 FILE MANAGER.

Create overlays by developing a program as a set of tasks to be performed. Write each task as if it were to be the only one in the machine. The tasks will be loaded (and later deleted) under the control of a small program that always remains in memory. Each task is called an "overlay" because it occupies the same memory as the preceding task.

To see how the overlay technique is used, let's analyze an overlaid program comprised of three tasks. (In our example, the function of each task will be the same.)

Three things must be considered in a numbering strategy. 1) The line numbers must remain the same in key sections of the small program that loads the overlay. 2) Each overlay must begin with the same line number. 3) Each overlay's line numbers must lie within a known range.

```
10000 GO TO 30090
20000 REM-----OVL-- AREA
20010 REM
30000 REM-----
30010 REM   MAIN PROGRAM
30020 REM   MAIN
30030 REM
30040 REM
30050 REM-----Load OVL Area
30060 GOSUB 30220
30070 GO TO 30060
30080 REM-----MAIN PROGRAM
30090 INIT
30100 AS=""
30110 U=2
30120 REM
30130 REM
30140 REM
30150 REM-----Typical Overlay Request
30160 FS="PROGRAM/OVL01"
30170 GO TO 30060
30180 REM
30190 REM
30200 REM
30210 REM-----LOAD OVL AREA
30220 IF FS=AS THEN 30280
30230 DELETE 20020,29990
30240 G=MEMURY
30250 CALL 'UNIT',U
30260 APPEND FS:20010,10
30270 AS=FS
30280 RETURN
```

Fig. 1. Numbering system for the main program.

The small controlling program (Fig. 1) is written so it can be renumbered starting at line 30080 (e.g., REN 30080,-10,30080). The sections of this program that are not to be disturbed are placed at the beginning. Thus, the main program can be developed and renumbered as new common routines are added.

For our example, the overlays have line numbers from 20010 to 29990. This allows 998 lines of 4051 BASIC to be written per overlay, using a line number increment of 10.

As mentioned, the overlays are identical except for the title banner and the first line of the message displayed. Note that each overlay begins with a REMARK statement. This statement has the same line number as a REMARK statement in the small controlling main program. Thus, each time the DELETE is done, a properly numbered line remains behind to serve as a target line in the next APPEND operation.

```

:0010 REM
:0020 REM :-----:
:0030 REM : SAMPLE PROGRAM
:0040 REM : OVLO1
:0050 REM :-----:
:0060 PAGE
:0070 PRINT 'Overlay 1 Entered'
:0080 PRINT
:0090 PRINT 'CHOOSE NEXT OVERLAY'
:0100 PRINT ' OVLO1 --1--'
:0110 PRINT ' OVLO2 --2--'
:0120 PRINT ' OVLO3 --3--'
:0130 PRINT ' END --4--'
:0140 PRINT 'ENTER OVERLAY CHOICE : '
:0150 INPUT 0
:0160 IF 0=1 OR 0=4 THEN 20140
:0170 IF 0=4 THEN 20240
:0180 OS=STR(0)
:0190 OS=SEG(OS,2,10)
:0200 REM-----Request+Transfer-----
:0210 FS='@PROGRAM/OVLO'+OS
:0220 GO TO 30060
:0230 REM-----:
:0240 END

:0010 REM
:0020 REM :-----:
:0030 REM : SAMPLE PROGRAM
:0040 REM : OVLO2
:0050 REM :-----:
:0060 PAGE
:0070 PRINT 'Overlay 2 Entered'
:0080 PRINT
:0090 PRINT 'CHOOSE NEXT OVERLAY'
:0100 PRINT ' OVLO1 --1--'
:0110 PRINT ' OVLO2 --2--'
:0120 PRINT ' OVLO3 --3--'
:0130 PRINT ' END --4--'
:0140 PRINT 'ENTER OVERLAY CHOICE : '
:0150 INPUT 0
:0160 IF 0=1 OR 0=4 THEN 20140
:0170 IF 0=4 THEN 20240
:0180 OS=STR(0)
:0190 OS=SEG(OS,2,10)
:0200 REM-----Request+Transfer-----
:0210 FS='@PROGRAM/OVLO'+OS
:0220 GO TO 30060
:0230 REM-----:
:0240 END

:0010 REM
:0020 REM :-----:
:0030 REM : SAMPLE PROGRAM
:0040 REM : OVLO3
:0050 REM :-----:
:0060 PAGE
:0070 PRINT 'Overlay 3 Entered'
:0080 PRINT
:0090 PRINT 'CHOOSE NEXT OVERLAY'
:0100 PRINT ' OVLO1 --1--'
:0110 PRINT ' OVLO2 --2--'
:0120 PRINT ' OVLO3 --3--'
:0130 PRINT ' END --4--'
:0140 PRINT 'ENTER OVERLAY CHOICE : '
:0150 INPUT 0
:0160 IF 0=1 OR 0=4 THEN 20140
:0170 IF 0=4 THEN 20240
:0180 OS=STR(0)
:0190 OS=SEG(OS,2,10)
:0200 REM-----Request+Transfer-----
:0210 FS='@PROGRAM/OVLO'+OS
:0220 GO TO 30060
:0230 REM-----:
:0240 END

```

Fig. 2. Overlay numbering system.

Line numbers 30210 to 30280 actually load the overlay, but only if it is not already resident in memory. Q=MEMORY is included to compress memory before the APPEND statement is executed.

The GOTO in line 10000 transfers control to the initial statement in the overlay controller program at line 30090, bypassing the sensitively numbered section. Organizing the program in this manner also ensures that an overlaid program, terminated with an END or STOP statement, can be RUN again without reloading any part of it from the 4907.

Figure 3 portrays the file structure created for the sample program. Using the desired overlay number (lines 20140-20190), the program dynamically builds a file name (line 20210) which the system then requests for loading.

When large programs are implemented as overlays, the 4907 can be used to provide direct access and fast loading of program segments. Using such techniques allows powerful application programming systems to be built.

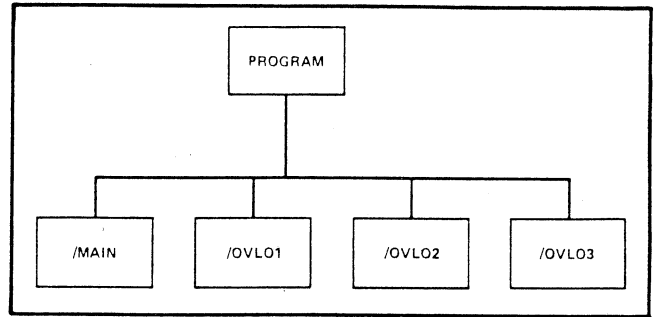


Fig. 3. 4907 file structure created for the sample program.

Save Overlay On Correct File

by Jay Beck

In developing a large program using overlays, you will be doing repeated OLDs and SAVEs. In the first few lines of each overlay include as part of the REMARK statement the name of the file in which that overlay is to be stored. Then, take time to list these lines and check the name of the file to avoid mistakenly storing an overlay on the wrong file.

4051 Subroutine Linkage

The GOSUB keyword is a very powerful and useful feature in BASIC. However, this feature can occasionally lead to undesirable results if its use is not fully understood.

Each time a GOSUB statement is executed, six more bytes of Random Access Memory is allocated to a push-down stack for the storage of return address pointers. In turn, each time a RETURN statement is executed, the stack is popped; allocated memory is then returned to the system. The stored return address is used to continue execution at the statement following the GOSUB.

This procedure works very well until a program branches out of a subroutine, using a GO TO or an IF-THEN statement. If these branches out of the subroutine do not re-enter the subroutine or execute a RETURN statement, then the stack is *not* popped. Many iterations of this process of pushing the stack (with a GOSUB) and neglecting to pop the stack (with a RETURN) results in a gradual loss of memory and, eventually, a MEMORY FULL condition.

Variable Names Take Same Bytes

In a recent issue of TEKniques, it was suggested that using single character variables instead of two-character would save memory. While this holds true for many machines, it does not for the 4051. The 4051 is designed so that no additional space is required for a two-character name.

Tracking Variables

Up to your ears with paper and code? Can't find those chunks with your variable names? Get organized. Try one of the variable tables sent in by TEKniques readers.

The first type of table is a form contributed by Bob Young of Tektronix. A handy program draws it on the 4662 Plotter. Shown below is a working example.

4051 BASIC VARIABLE TABLE									
PROGRAM: EXAMPLE #X									
	1	2	3	4	5	6	7	8	9
A									
B									
C									
D	(/a)								
E	DESCRIPTION:								
F		FACT1	FACT2	FACT3					
G		1	2	3					
H	(/a)								
I	HEADING								
J	INPUT LOOP								
K	OUTPUT LOOP								
L									
M	(/a)								
N	TABLE								
O									
P	OUTPUT DEVICE	COL 1	COL 2	COL 3	COL 4	COL 5	COL 6	COL 7	COL 8
Q									
R									
S									
T	INPUT FLAG								
U									
V									
W									
X									
Y									
Z									

REMARKS: * CRT = 32
 * 41 MATRX PRINTER = 41
 * 4662 PLOTTER = 3

```

100 INIT
110 D=1
120 PRINT @1,17:2,2
130 PRINT @D,21:10,9,98
140 PRINT @D:"4051 BASIC"
150 PRINT @D,21:13,8,93,5
160 PRINT @D:"PROGRAM:"
170 PRINT @D,21:13,67,93,3
180 PRINT @D,20:13,67,5,75,125,2,5,75,125,2,93,3,9,6,93,3,9,6,5,75
190 PRINT @D,20:13,67,5,75
200 PRINT @D,21:11,5,2,7
210 PRINT @D:"REMARK:"
220 PRINT @D,21:26,9,5,75
230 FOR I=26,9 TO 127 STEP 17,872
240 PRINT @D,21:1,5,75
250 PRINT @D,20:1,93,3,1+8.936,93,3,1+8.936,5,75
260 NEXT I
270 FOR I=8,6 TO 88 STEP 6,596
280 PRINT @D,21:125,2,1
290 PRINT @D,20:9,6,1,9,6,1+3.298,125,2,1+3.298
300 NEXT I
310 DATA "9","8","7","6","5","4","3","2","1","0","b","s"
320 FOR I=120 TO 14 STEP -8,936
330 READ A$
340 PRINT @D,21:1,91
350 PRINT @D:A$
360 NEXT I
370 DATA "A","B","C","D","E","F","G","H","I","J","K","L","M"
380 DATA "N","O","P","Q","R","S","T","U","V","W","X","Y","Z"
390 FOR I=88,4 TO 5 STEP -3,298
400 READ A$
410 PRINT @D,21:11,32,1
420 PRINT @D:A$
430 NEXT I
  
```

The second type is in the form of a BASIC program. Ed Mitchell of Tektronix wrote the program that prints variable tables on the screen or to a line printer. Ed's program provides for String Variables, Simple Variables, and Function Variables. Examples and the program listing follow.

STRINGS	
A\$	LENGTH()
B\$	LENGTH()
C\$	LENGTH()
D\$	LENGTH()
E\$	LENGTH()
F\$	LENGTH()
G\$	LENGTH()
H\$	LENGTH()
I\$	LENGTH()
J\$	LENGTH()
K\$	LENGTH()

SIMPLE			
A			
A0	A1	A2	A3
A4	A5	A6	A7
A8	A9		
B			
B0	B1	B2	B3
B4	B5	B6	B7
B8	B9		
C			
C0	C1	C2	C3
C4	C5	C6	C7
C8	C9		

FUNCTIONS		
FNA()	FNB()	FNC()
FND()	FNE()	FNF()
FNG()	FNH()	FNI()
FNJ()	FNK()	FNL()
FNR()	FNS()	FNO()
FNP()	FNQ()	FNR()
FNS()	FNT()	FNU()
FNV()	FNW()	FNX()

```

1 F=0
2 SET KEY
3 GO TO 100
4 REM ***** U/D 1 LIST STRINGS *****
5 F=1
6 PRINT @32,26:3
7 GO TO 240
8 REM ***** U/D 2 LIST SIMPLE VARIABLES *****
9 F=1
10 PRINT @32,26:3
11 GO TO 380
12 REM ***** U/D 3 LIST FUNCTIONS *****
13 F=1
14 PRINT @32,26:3
15 GO TO 760
180 PAGE
181 PRINT "ENTER DEVICE NUMBER FOR LINE PRINTER (40 OR 41), OR RETURN:"
190 PRINT "FOR SCREEN,G"
195 REMPRINT @32,26:3
198 INPUT A$
199 IF A$="" THEN 198
200 D=VAL(A$)
201 IF D=40 OR D=41 THEN 200
170 GO TO 100
180 GO TO 280
190 D=32
200 PRINT "PRESS RETURN FOR COMPLETE LIST OR PRESS USER DEFINABLE 1, 2"
210 PRINT "OR 3 FOR"
220 PRINT "PARTIAL LISTS."
230 INPUT A$
240 REM ***** START STRING LIST *****
250 PRINT @D:"L"
260 PRINT @D:""
270 FOR N=65 TO 90
280 B$=CHR(N)
290 PRINT @D:B$
300 PRINT @D:" LENGTH( )"
310 NEXT N
320 IF F<>1 THEN 300
330 REM ??? F=1 IS THE USER DEFINABLE FLAG ???
340 F=0
350 PRINT @32,26:0
  
```

```

360 END
370 REM ***** START SIMPLE VARIABLE *****
380 PRINT "L"
390 PRINT "S"
400 FOR N=65 TO 90
410 B$=CHR(N)
420 PRINT "B";B$
430 REM ***** PRINT SIMPLE VARIABLES 4 AT A TIME (SUB 670) *****
440 X=0
450 Y=1
460 GOSUB 670
470 X=X+1
480 Y=Y+1
490 GOSUB 670
500 X=X+1
510 Y=Y+1
520 GOSUB 670
530 REM ***** GIN IS TO INSURE THAT COMPLETE VARIABLES ARE PRINTED *****
540 REM ***** BY FORCING A PAGE FULL IF THERE ISNT ENOUGH ROOM. *****
550 GIN 0,M
560 IF M/20 THEN 620
570 IF D<>32 THEN 620
580 COPY
590 PAGE
600 GO TO 620
610 NEXT M
620 NEXT M
630 IF F<>1 THEN 760
640 F=0
650 PRINT "32,26:0"
660 END
670 FOR N1=X TO Y
680 C$=STR(N1)
690 C$=SEG(C$,2,1)
700 PRINT "B";C$
710 NEXT M1
720 PRINT "J";
730 RETURN
740 NEXT M
750 PRINT "J";
760 PRINT "L";
770 PRINT "S";
780 FOR N1=65 TO 90 STEP 3
790 FOR N1=0 TO 2
800 H2=N1+1
810 IF H2=90 THEN 870
820 B$=CHR(H2)
830 PRINT "F";B$;"(")
840 NEXT M1
850 PRINT "J";
860 NEXT M
870 IF D<>32 THEN 890
880 COPY
890 PRINT "32,26:0"

```

The third type of variable table is also a BASIC program contributed by Dr. P. C. Holman of the University of Wisconsin-Stevens Point. It is a quick and easy screen-only table to keep track of numeric variables or alpha strings. The table is reproduced here followed by the program listing...

PROGRAM NAME		PROGRAMMER		DATE	
PROGRAMMING AID					
NUMERIC					
					ALPHA
A)	A0)	A1)
B)	B0)	B1)
C)	C0)	C1)
D)	D0)	D1)
E)	E0)	E1)
F)	F0)	F1)
G)	G0)	G1)
H)	H0)	H1)
I)	I0)	I1)
J)	J0)	J1)
K)	K0)	K1)
L)	L0)	L1)
M)	M0)	M1)
N)	N0)	N1)
O)	O0)	O1)
P)	P0)	P1)
Q)	Q0)	Q1)
R)	R0)	R1)
S)	S0)	S1)
T)	T0)	T1)
U)	U0)	U1)
V)	V0)	V1)
W)	W0)	W1)
X)	X0)	X1)
Y)	Y0)	Y1)
Z)	Z0)	Z1)

```

LIS
100 INIT
110 REM THIS IS A PROGRAM TO HELP YOU KEEP TRACK OF THE VARIABLES IN
120 REM ANY PROGRAM AS YOU USE A NUMERIC VARIABLE OR ALPHA STRING
130 REM FILL IN THE PARENTHESIS AND CIRCLE WHAT YOU HAVE USED (E.G.
140 REM A2(25) OR A2(2,3) TO DIMENSION NUMERIC VARIABLES), OR (E.G.
150 REM A$(49) TO DIMENSION AN ALPHA STRING VARIABLE.)
160 REM ALSO THEY CAN BE USED TO DEFINE VARIABLES NOT DIMENSIONED.
170 REM (E.G. LET A$A+1), (E.G. LET A$=THE*)
180 PAGE
190 LET L$="ABCDEFGHIJKLMNORSTUVWXYZ"
200 LET N$="0123456789"
210 PRINT "PROGRAMMING AID"
220 PRINT "J";
230 PRINT "PROGRAM NAME----- PROGRAMMER-----";
240 PRINT "DATE / /";
250 PRINT "J";
260 PRINT "NUMERIC";
270 PRINT "ALPHA";
280 PRINT
290 FOR I=1 TO LEN(L$)
300 LET A$=SEG(L$,I)
310 FOR J=1 TO LEN(N$)
320 PRINT A$;
330 LET B$=SEG(N$,J)
340 PRINT B$;"(")
350 NEXT J
360 PRINT " "
370 NEXT I
380 END

```

Whatever means you use to track your variables, include it in your documentation. Your program user will be forever grateful.

Note: There are also two programs in the 4051 Applications Library to do variable cross-referencing for existing programs: 51/00-8002/0 and 51/00-8003/0.

CASE Condition and Parentheses, Brackets and Braces

by Arnie Kaber

When SET CASE (default) is in effect, the BASIC interpreter considers lower case letters equal to upper case letters in string comparisons. The following pairs of symbols are also considered equal.

< >, [], ()

However, if SET NOCASE is specified, upper and lower case letters are not equal; and neither are these symbols. The following examples illustrate results in each case:

```

LIS
100 SET NOCASE
110 IF "ABC"="(ABC)" THEN 300
120 PRINT "THEY ARE NOT EQUAL"
200 END
300 PRINT "THEY ARE EQUAL"

```

RUN
THEY ARE NOT EQUAL

```

LIS
100 SET CASE
110 IF "ABC"="(ABC)" THEN 300
120 PRINT "THEY ARE NOT EQUAL"
200 END
300 PRINT "THEY ARE EQUAL"

```

RUN
THEY ARE EQUAL

```

LIS
100 SET NOCASE
110 IF "ABC"="(ABC)" THEN 300
120 PRINT "THEY ARE NOT EQUAL"
200 END
300 PRINT "THEY ARE EQUAL"

```

RUN
THEY ARE NOT EQUAL

```

LIS
100 SET CASE
110 IF "ABC"="(ABC)" THEN 300
120 PRINT "THEY ARE NOT EQUAL"
200 END
300 PRINT "THEY ARE EQUAL"

```

RUN
THEY ARE EQUAL

Consequently, comparing strings which include parentheses, brackets and braces, may produce unanticipated results.

However, in numeric operations, the 4051 converts brackets or braces to parentheses (or drops them if unnecessary). Therefore, invalid results *due to these characters* won't occur in numeric comparisons.

For example,

```

100 A=1/2+(2+3)-(4-2)/3+(24/3)
110 PRINT A
120 END
130

```

```


LIS
100 A=1/2+(2+3)-(4-2)/3+24/3
110 PRINT A
120 END

```

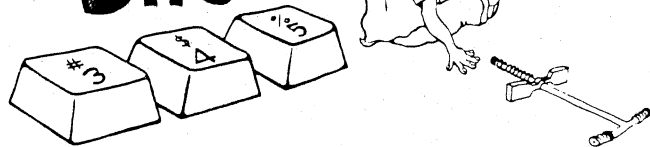
INIT Rather Than CLOSE

by Jay Beck

When your 4907 file is open for read/write operations (OPEN "TESTFILE";1,"F",AS), a pointer in the 4907 firmware indicates where the next item to be read/written is to occur. When a program fails after reading part way through a file, the user should type INIT. This resets the pointer to the beginning of the file and closes it, keeping the original file intact. If a CLOSE is typed instead, the 4907 will write an end-of-file mark at the current pointer

location and close the file. Everything between the pointer and the end of the original file is lost. This problem does not occur if the file is opened in read-only or update access modes. 

Basic Bits



Renumbering Doesn't Update Everything

When a RENUMBER statement is executed, the target lines of any CALL statement are *not* updated.

For example:

```

3500 CALL "BAPPEN",3500
2510 CALL "LINK",4500
3500 REM
4500 REM
    
```

(3500 is target line for BAPPEN)
(4500 is target line for LINK)

```

RENUMBER
LIS
    
```

```

100 CALL "BAPPEN",3500
110 CALL "LINK",4500
120 REM
130 REM
    
```

(120 is new target line for BAPPEN)
(130 is new target line for LINK)

Note that both target lines in renumbered statements 100 and 110 still retain the original line numbers.

Renumbering Might Update Wrong Line

If you have code that references non-existent lines, before renumbering either insert those lines or renumber with higher values. This program illustrates results of not paying attention to this detail.

```

147 GOTO 110
231 REM Function A
    
```

(line 110 is non-existent)

```

REN 100.10

100 GOTO 110
110 REM Function A
    
```

(still refers to original target line 110 but now it's the wrong target)

Then, after more coding, you follow with another renumber:

```
REN 200.10
```

```

200 GOTO 210
210 REM Function A
    
```

(original target line number 110 vanishes to be replaced with wrong line number leading to wrong target)

If the initial renumber had begun with a line number higher than the non-existent target line, the original target line number would have remained:

```

147 GOTO 110
231 REM Function A
    
```

(line 110 is non-existent)

```
REN 200.10
```

```

200 GOTO 110
210 REM Function A
    
```

(still non-existent but number intact)

Or, retain correct target by inserting line 110 as a REMARK statement to hold it prior to renumbering:

```

147 GOTO 110
231 REM Function A
110 REM Hold Target Line
    
```

(line 110 is non-existent)

```

REN 100.10

100 REM Hold Target Line
110 GOTO 100
120 REM Function A
    
```

(different number but correct target)

Note: BASIC EDITOR and Formatted Listings #1 of the PRINTER PROGRAMS, both contained in PLOT 50 General Utility Programs, will tell you if you have any non-existent line references.

Variable Symbols Are Not Deleted

Once a simple variable symbol has been entered into memory, it always takes up at least 13 bytes of space (5 in the table listing and 8 for the number). A DELETE of a variable will eliminate the *value* assigned to that symbol, but the symbol will remain in the table and the space will be reserved for the number. To save memory, re-use these symbols for new values when the old are no longer needed, rather than assigning new symbols.

For example:

```


300 FOR I=1 TO 8
310 GOSUB 700
320 NEXT I
330 FOR J=1 TO 5
340 GOSUB 900
350 NEXT J
    
```

Rather than using the new variable J, the variable I could have been reassigned. As it stands in the example, I and J

are both in the symbol table, with only one being used. Refer to page 4-8 in the 4051 Graphic System Reference Manual for more detail.

Two-Character Variables Desirable

Using two-character variable names might save time if you plan to edit your program using the Editor ROM or an editor program.* Whereas, searching for a single character variable will result in unwanted text being picked out, searching for an alpha character coupled with a numeric will usually locate the desired variable.

*51/00-8007/0, Text Editor, in the 4051 Applications Library, or 4050A08 BASIC EDITOR contained in PLOT 50 General Utility Programs, are two such programs available. 

4051 Applications Library Program Abstracts

Order

Documentation and program listings of each program are available for a nominal charge. Programs will be put on tape for a small recording fee per program plus the charge for the tape cartridge. One tape will hold several programs. (The program material contained herein is supplied without warranty or representation of any kind. Tektronix, Inc. assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.)

Domestic U.S. Prices:

Documentation and listings	\$15 per program
Recording Fee	2 per program
Tape Cartridge	26 per tape

Contribute

Contribute one program to the Library and receive three in exchange. Send in the membership card from your 4051 Graphic System Reference Manual to get the details. Or call us at (503) 682-3411, ext. 2618.

Forms

Please use the Applications Library Order Form. Order forms are included in the Membership Packet and are available from your local Tektronix Sales Engineer.

Outside U.S.

Program contributions or orders outside the U.S. must be processed through the local Tektronix sales office or sent to one of the Libraries serving your area. See Library Addresses section.

ABSTRACT NUMBER: 51/00-5202/0

Title: **NMR Calculation for a Three Spin System**

Authors: Tom Price and Dr. Jack Reid

Lorillard Div. of Loews

Greensboro, NC

Memory Requirement: 16K

Peripherals: Optional—4631

Optional—4641

Statements: 267

This program calculates the spin-spin transition frequencies and relative intensities for a three spin Nuclear Magnetic Resonance (NMR) system. The energies of the eight possible energy states are calculated using the chemical shifts, in Hz, and coupling constants in Hz, entered from the keyboard. In the three spin system there are four possible energy levels. The first and fourth each contain one spin state; however, the second and third each contain three spin states. The energies of these states are used as the diagonals of two 3 x 3 matrices. The off diagonal elements are calculated from the respective coupling constants. The two matrices are diagonalized, by the method of Jacobi ^{①②} and the intensities calculated by identical diagonalization operations on an identity matrix. The final transition frequencies are calculated by the difference in the energies between adjacent energy levels.

The output is displayed as either a listing and an NMR line spectrum on the graphic terminal or a listing on the printer.

ABSTRACT NUMBER: 51/00-8018/0

Title: **Text FORMATting**

Author: Bruce Clarkson

Science Applications, Inc.

Raleigh, NC

Memory Requirement: 16K (8K w/o comments)

Peripherals: Option I Data Communications Interface
or

Option 10 RS-232 Printer Interface

4051R06 Editor ROM

Hard Copy Terminal

Statements: 211

FORMAT provides formatted text output similar to the University of Waterloo's SCRIPT program for System 360. Commands imbedded in the text file (typically created with the TEKTRONIX Editor ROM) together with execution time parameters gives complete control over the format of the printed document. Options include right margin justification (padding with blanks), page size control, page numbering, indentation, spacing, centering, etc. Any standard ASCII RS-232 compatible output device may be used (example: DEC-WRITER terminal).

ABSTRACT NUMBER: 51/07-8019/0

Title: **4907 File Manager Based Mailing List**

Author: Brian Diehm
Tektronix, Inc.

Memory Requirement: 16K
Peripherals: 4907 File Manager
Optional—4641 Printer (Recommended)
Statements: 927

This program uses the 4907 FILE MANAGER to maintain and print mailing lists for up to 50 different publications. Subscribers to all the lists are drawn from a single master subscriber file. Provision is included to add, delete, edit and list subscribers or subscriber groups by a last name key. Up to four lines of address information may be used.

Output files may be created for any of the publications, and this file is in zip code order. This file may be used by the program to drive the 4641 Printer to print a mailing list of stick-on labels.

Two versions of the program are included. Only the minimum configuration version can be used on 16K machines. This version may also be best suited to very large output files, as the only limitation to their size is available disc space. The second version runs faster when creating output files by keeping a copy of the file's link pointers in memory. This nets a 40% reduction of time required to create output files. However, the length of output file that may be created is limited by the memory available to store the links. For a 32K 4051, up to 1000 entries may be filed; for a 24K machine, up to 500 entries may be filed.

In both versions, the size of the subscriber file is limited only by the available disc space.

ABSTRACT NUMBER: 51/00-0501/0

Title: **Inventory/Production Modeling I**

Author: Dennis R. Heckman
Tektronix, Inc.

Memory Requirement: 16K
Peripherals: Optional—4662 Plotter
Optional—4641 Printer
Statements: 333

This program is designed to naively represent a Manufacturing Operation producing one product. As inputs the program requires:

Ratios and Constants

- An average production rate per worker
- Total facility capacity
- Desired backlog time

An average material waste ratio

An average lost order rate

Desired inventory risk factor

Initial Settings

Initial inventory

Initial order backlog

Initial production

Initial shipments

Order Entry

The number of iterations

The initial order level

Either:

The growth rate
or

The iteration number for a step change

The amount of the change

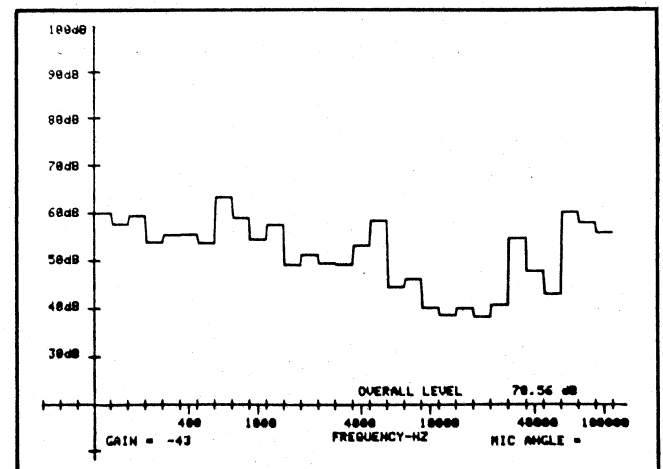
A series of equations are then used to model the various relationships while the order quantities are derived from a random number generator. The Work Force and Materials Inventory required for production at each iteration may be output.

ABSTRACT NUMBER: 51/00-5403/0

Title: **Sound Spectrum PRESENTATION**

Author: Norman D. Taylor
Memory Requirement: 16K
Peripherals: 4631 Hard Copy Unit
Rockland FFT 512/S
Statements: 126

This program reads the values for a one-third octave sound spectrum from a Rockland Analyzer, calculates Overall Sound Pressure Level (OASPL), plots this data on the 4051 and generates a hard copy.



The program interrogates the Analyzer to determine the frequency range that has been analyzed and the gain settings, both coarse and fine. Then the coordinate axes are drawn accordingly. The Y-axis is plotted in decibels (dB) and the X-axis is frequency in Hertz (Hz). Next, the spectrum is read in from the analyzer. As it is read and plotted, the OASPL is calculated. Finally, a hard copy of the finished plot is generated.

An option is included to allow a series of OASPLs to be calculated (for instance, a line of microphones) and a listing displayed.

ABSTRACT NUMBER: 51/00-9508/1

Title: **Dashed Lines**
Author: Bob Ross
Memory Requirement: 8K
Peripherals: Any Plotting Device
Statements: 154

This is a consolidation of programs 51 00-9508 0,
51 00-9509 0 and 51 00-9510 0.

Three subroutines draw dashed lines for:

- 1) a Y array with X values stepped linearly from a starting to an ending value;
- 2) points stored in X and Y arrays;
- 3) a sequence of X and Y values.

The dashes are a constant length regardless of the viewport and window chosen. The dash length and ratio of dash to dash plus space are selectable. The line can start and end on a full dash or full space.

ABSTRACT NUMBER: 51/07-9522/0

Title: **4907 U.S. Map and Segmented Data Base Windowing Routine**
Author: Leslie L. Brabetz
Tektronix, Inc.
Memory Requirement: 32K, Level 5 Firmware
Peripherals: 4907 FILE MANAGER
4952 Joystick
Statements: 956 plus 99 binary data files*

Two programs illustrate and employ the unique capabilities of the Tektronix 4907 FILE MANAGER.

U.S. Map

The U.S. Map program demonstrates accessing and displaying a segmented data base existing on the 4907 FILE MANAGER, as opposed to reading and clipping an entire serial data base in order to view a smaller

portion. The master data base is read and a highly detailed map of the continental United States, with grid, is displayed on the 4051 graphic screen.

The user is then requested to select a lower left corner and upper right corner using the 4952 Joystick. Only those segment files containing the selected portions of the map are read, permitting rapid display.

The routines included in the U.S. Map program are: disc formatting, transfer from tape to disc of the 96 segment data files, master data base, and routine to display the map and its segments.

Windowing Routine

This program creates a segmented data base from a master data base (the U.S. Map in this case) to demonstrate the 4907 FILE MANAGER capabilities. It performs random access of files from the disc. A series of articles in TEKniques (Vol. 1 No. 10, and Vol. 2 Nos. 1 and 2) describe the theory and operation of the demonstration.

The program allows definition of a rectangular data window. A master file may be read in, and the vectors which begin and end or intersect the data window are stored in a segment file. The coordinates of intersection with the boundaries are calculated and stored in the segment file. On a graphics display, this operation is called "windowing"; hence, this program windows on data bases. The master data file must be in the form of arrays, with the number of coordinate pairs N, then the coordinate arrays X, Y.

$$N, X_1, X_2, \dots, X_n, Y_1, Y_2, \dots, Y_n$$

Output segment files are created with the same format.

*Two cartridge tapes required.

ABSTRACT NUMBER: 51/00-9523/0

Title: **Data Graphing**
Author: Chuck Eng
Tektronix, Inc.
Memory Requirement: 32K
Peripherals: Optional—4631 Hard Copy Unit
Optional—4662 Plotter
Optional—4051R05 Binary ROM
Statements: 926

Create graphs quickly and easily with Data Graphing. Computer expertise is not required since the user is prompted for the minimal inputs. Up to six curves may be created from:

Keyboard data input

The sum of all previous curves

- The average of all previous curves
- The cumulative sum of the previous curve
- The least squares fit of the previous curve
- Data on a data file

X-axis values may be user-input or auto-sequence. Graph design allows selection of:

The type of curve (bar, solid, phantom, dash) with variations of each

Grid


Labels

Hidden line removal

Scaling of Y-axis

Saving data for curves and/or graph.

Other functions allow changes in curve data or graph parameters, listing of data, drawing to the 4051 screen or 4662 Plotter and examples.

The program includes the menu file. Binary and ASCII programs and four sample data files. Additional data files must be pre-MARKed for storage of curve and graph data. 

4051 Applications Libraries

Africa, Europe, Middle East

4051 Applications Library
Tektronix Datatek N.V.
P.O. Box 159
Badhoevedorp, The Netherlands

Australia

4051 Applications Library
Tektronix Australia Pty. Limited
Sydney
80 Waterloo Road
North Ryde, N.S.W. 2113

Canada

4051 Applications Library
Tektronix Canada Ltd.
P.O. Box 6500
Barrie, Ontario
Canada L4M 4V3

Caribbean, Latin America and Far East (excl. Japan)

Ms. Bev Brandon, 73-312
Export Marketing
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077
U.S.A.

Japan

4051 Applications Library
Sony Tektronix Corporation
9-31 Kitashinagawa-5
Tokyo 141 Japan

United States

4051 Applications Library
Tektronix, Inc.
Group 451
P.O. Box 500
Beaverton, Oregon 97077

TEKniques, the 4051 Applications Library Newsletter, is published by the Information Display Group of Tektronix, Inc., Group 451, P.O. Box 500, Beaverton, Oregon 97077. It is distributed to TEKTRONIX 4051 users and members of the 4051 Applications Library.

Publisher Ken Cramer
Managing Editor Patricia Kelley
Editor Terence Davis
Graphic Designer Mark Woods
Circulation Rory Gugliotta