

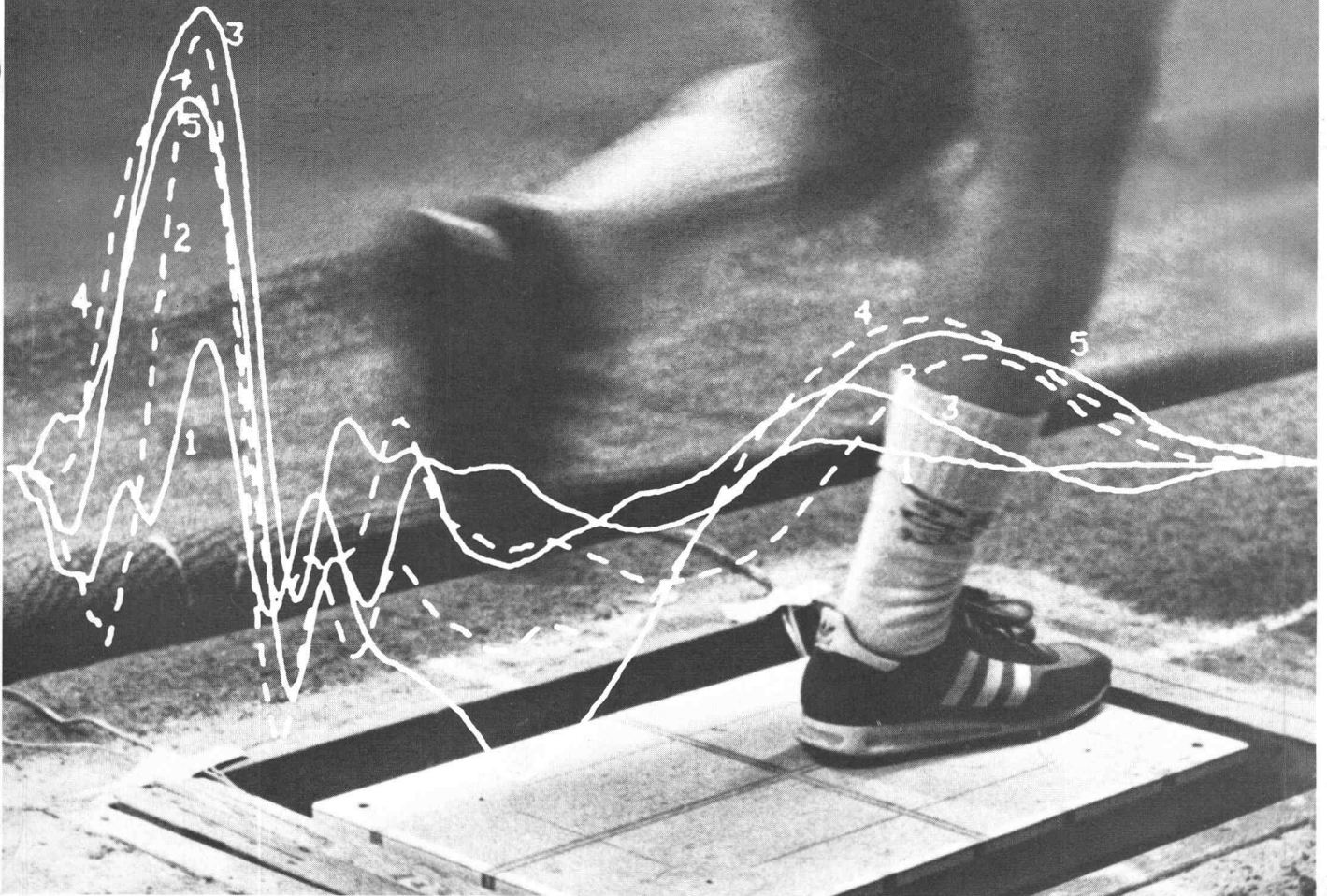
Tekniques

The 4050 Series Applications Library Newsletter

September 15, 1980

Vol. 4 No. 6

4051 Aids Biomechanics Research



Tekniques

In This Issue

4051 Aids Biomechanics Research	1
New ROM Pack Adds Time Functions to 4052 and 4054.....	6
Avionics Research Using the 4051	9
Editor's Note	13
Back Issues: Vol.4 Only	
Programming Tip Handbook	
New Catalog	
International Users	
New Contest Coming	
Programs Wanted	
Programming Tips Too	
Telephone Notes	
Input/Output	14
Graphic Systems Workshops	
Continue	16
Programming Tips.....	17
BASIC Bits.....	25
New Abstracts	27
Library Addresses.....	31

TEKniques, the 4050 Series Applications Library Newsletter, is published by the Information Display Division of Tektronix, Inc., Group 451, P.O. Box 500, Beaverton, Oregon 97077. It is distributed to TEKTRONIX 4050 Series users and members of the 4050 Series Applications Library.

Publishing Manager	Ben Buisman
Managing Editor	Patricia Kelley
Editor	Terence Davis
Technical Editor	Dan Taylor
Graphic Design	John Ellis
Circulation	Rory Gugliotta

Copyright © 1980, Tektronix, Inc.
All rights reserved.

To submit articles to TEKniques or for information on reprinting articles, write to the above address. Changes of address should be sent to the 4050 Series Library serving your area (see Library addresses).

4051 Aids Biomechanics Research at University of Oregon. Sports, Medicine, Industry, and Public Safety all Benefit.

by Terry Davis
TEKniques Staff

Biomechanics is a science that studies the human body in motion. By applying principles from mechanics and engineering, biomechanists study the forces that act on the body, along with the effects they produce. This involves detailed analysis of movement, to find new and meaningful information about movement and related forces. The ultimate goal is better understanding of movement and movement-related problems in skill performance, sports medicine, and equipment design.

At the University of Oregon, in the Physical Education Department's Gerlinger Hall, Dr. Barry Bates heads up the Biomechanics Sports Medicine Laboratory. The primary thrust of his research there is concerned with the use of this dynamic science in the design of human-related machines and equipment. He's applying biomechanics principles to the problems of industry and sports medicine; much of that effort is directed toward movement-related problems of the lower extremities. And to do so, he's built a system for Biomechanical Analysis, in which the 4051 Graphic System plays an important part.

One major area of research is in the biomechanics of jogging and distance running, especially in relation to the design and evaluation of shoes. In addition, Dr. Bates has consulted in the biomechanical aspects of injury-related civil cases, and investigated the causes (and possible methods of prevention) of injuries in the wood products industry.

Improving Running Shoe Designs

With the current surge in enthusiasm for distance running as sport and exercise, and the associated upswing in injuries, the design of running shoes is an excellent area for investigation. Dr. Bates and his colleagues are trying to answer a number of questions that relate to the design of running shoes. How, for instance, do different shoe materials and construction types affect the functioning of the lower extremity? And how do different subjects and shoes interact,

especially with respect to ground reaction forces—those forces that appear when the moving body mass meets solid ground. What happens during the support phase of running, that causes runners to become injured?

In addition, total body mechanics during the impact (breaking) and loading (propulsion) phase of running are being studied as a function of running speed and type of footfall. As the answers to these questions begin to appear, laboratory personnel are able to assist in the development of material and performance standards in conjunction with the American Society of Testing Materials (ASTM).

The key difference between Dr. Bates' approach to shoe design and evaluation compared to other methods, such as those used by the widely-read *Runner's World* survey, is the dynamic nature of the tests with the emphasis placed on the runner instead of the shoe. In the survey, the potential interactions between the shoe and the human foot and leg are ignored. Says Bates, "Their tests are mechanical, not biomechanical. In other words, they take a shoe, crank it into a machine, and test it. You can bend, twist, hammer, tear, and sand a shoe, but until you put a foot in it, it's not functional. Dynamic function must be evaluated, in relation to healthy runners as well as injured ones."

Reflecting on the problems that he's tackled, Dr. Bates says, "I think we're helping to make better shoes. Two primary factors contribute to shoe-related injuries: (1) lack of ability to absorb shock, and (2) inadequate rear foot control. We haven't found a shoe so far that provides both of these important functions. We're working hard on a shoe here at the lab that does combine both." To do that, Dr. Bates is drawing science and technology into an area that, until now, has been largely a guessing game, defaulting into the court of the fashion designer. In this process, the primary data collection and reduction equipment being used all functions around and/or in conjunction with a 4051 Graphic System.

The System

The system involves several pieces of equipment that gather two basic kinds of data for

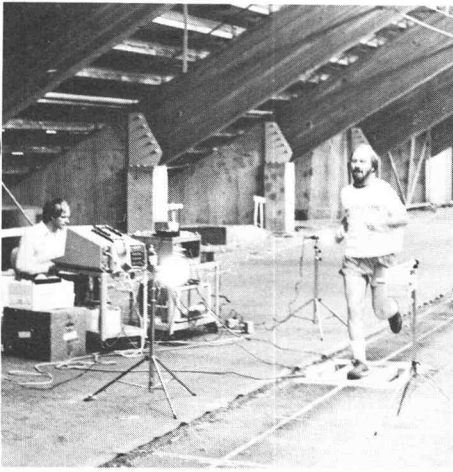


Fig. 1. Dr. Bates runs across the force plate during dynamic shoe testing, as a research assistant watches the system display.

the biomechanics studies. One is a Kistler Force Platform, which is used primarily for the running studies. It's set up during running tests under the grandstands at nearby Hayward Field, and is interfaced to the 4051 through a TransEra A-to-D converter. To test the interaction between shoes and runners, test subjects run on the platform wearing various types of shoes (Figure 1). The platform can read forces in several different directions, as the runners contact it. The analog signal generated is converted to digital form; the digital data is then processed by the 4051.

While running on the platform, the runners are also filmed at 200 frames per second with high-speed Super-8 mm cameras. These films provide additional data input for the system. They're played back on a projector that's capable of stop action, and projected on the surface of a Numonics digitizing tablet. This allows the researchers to digitize the positions of various key parts of the anatomy during the phases of running. (This method is used in other research projects as well, as we'll see later.) Running speed is monitored with a photoelectric timing system.

Digitized movie data is also input to the 4051, to be processed into a graphic representation of total body mechanics during running, or to provide additional data as required to supplement the force platform data. Figure 2 is a representation of the digitized film data of two views of the runner. Viewing running as a total body activity actually provided the starting point for research on running and associated injuries. That research led to increased emphasis on evaluating the functional aspect of the lower extremity, especially foot functions. Shoe design naturally evolved as a primary interest area.

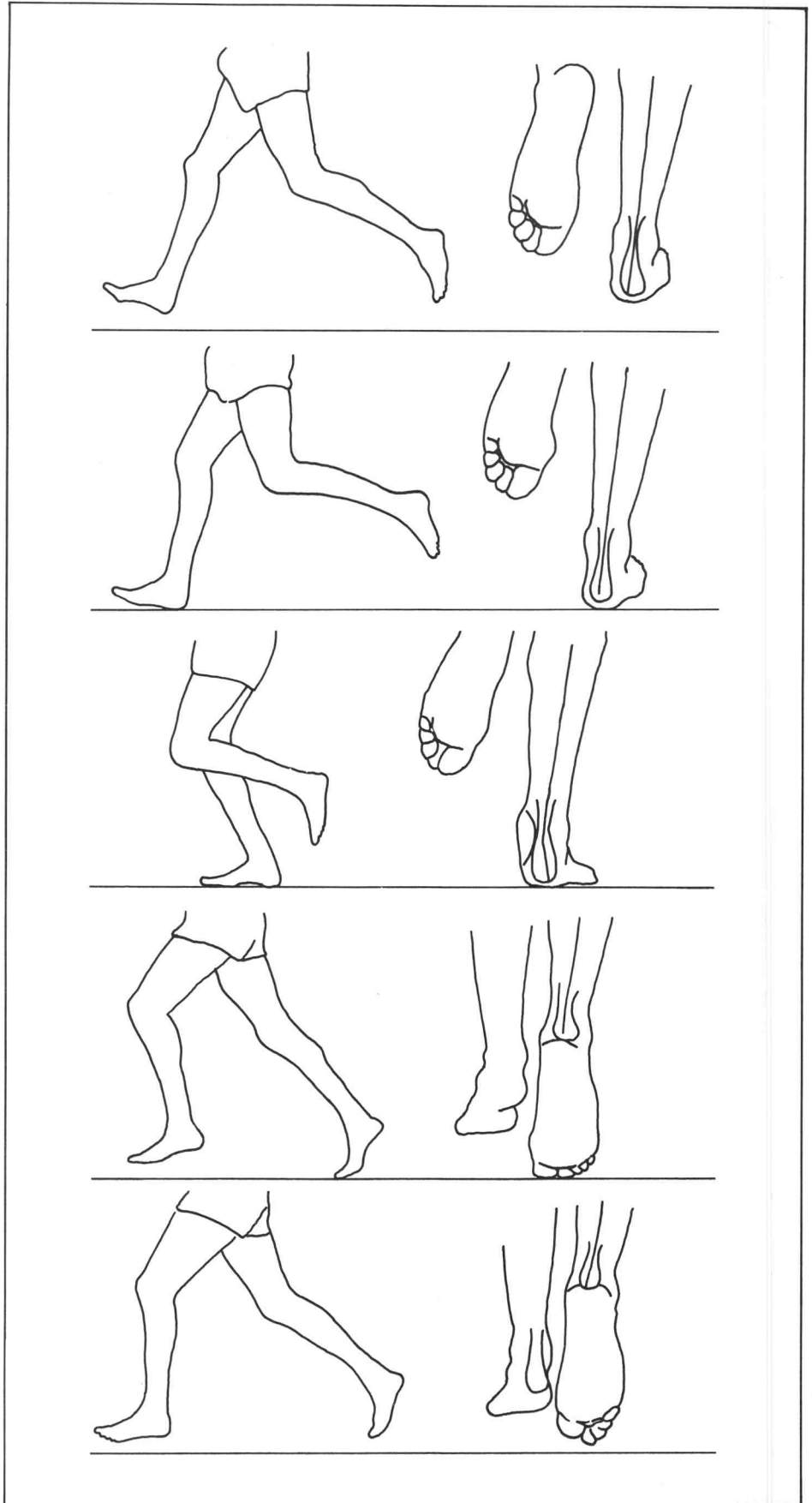


Fig. 2. Lateral and posterior views of the lower extremity, drawn from digitized film data. Positions shown are (a) prior to contact, (b) begin pronation, (c) maximum pronation, (d) end pronation, (e) after take-off.

Shoe Test Procedures

A typical session of collecting data might involve evaluating five different shoes worn by five different runners. Each runs across the force platform, under specified conditions, until a number of successful trials are recorded. During the tests, conditions are varied for each runner and between different runners, to provide systematic variation of the tests. Runners are also given as much time as they desire between trials and between conditions, to rest and to become accustomed to each new condition.

A key to accurate data from these trials is the definition of an acceptable trial. For this research, an acceptable trial is defined as one where the runner contacts the force platform in a normal stride pattern at a designated pace. Each runner will typically perform 10 successful trials under each condition, translating to about 1—1 1/2 hours to perform the 50 trials.

Running speed is monitored and controlled for the purposes of the study; the average pace is equivalent to a 5.5 to 6.5 minute mile. Data from the force plate is sampled at 1000-1200 Hz during the running trials. To prevent "abnormal" footplants (and associated invalid data) due to runners reaching or shortening their strides as they cross the force plate, or having only part of their foot on the plate, all data samples are visually inspected before storage. The A-D Converter ROM Pack has a graphing function that allows all channels of collected data to be graphed using a single function.

S3-T1-0N-0.96 2-FORCE														
TRIAL 1	1990	1227	313	1688	928	692	3739	1395	798	138	248	1478	1851	
TRIAL 2	1847	1116	281	1651	820	613	3685	1237	791	185	284	1192	1781	
TRIAL 3	1128	1438	328	1792	1011	684	3827	1658	923	143	285	1658	1788	
TRIAL 4	1952	1283	328	1738	842	788	3841	1381	782	126	258	1441	1843	
TRIAL 5	1868	1179	297	1785	866	661	3795	1249	712	189	221	1248	1741	
TRIAL 6	1863	1297	297	1697	928	645	3688	1472	848	123	245	1483	1749	
TRIAL 7	1114	1245	328	1769	872	661	3646	1265	717	126	252	1298	1812	
TRIAL 8	1118	1225	328	1776	845	684	3794	1336	746	127	249	1347	1884	
TRIAL 9	1899	1375	313	1771	1011	637	3618	1513	868	145	283	1518	1764	
TRIAL 10	1889	1425	385	1711	924	716	4822	1536	883	142	271	1572	1788	

MEAN VALUES	184	1887	1281	389	1738	984	678	3757	1483	797	128	252	1422	1783
STANDARD DEVIATIONS	8	29	185	14	46	68	33	128	138	79	14	25	148	47

178	1847	1116	281	1651	820	613	3685	1237	781	185	284	1192	1781	
282	1128	1438	328	1792	1011	684	3841	1658	923	143	285	1658	1851	
1	4	16	2	7	18	5	12	21	11	2	4	23	8	
2	8	32	4	14	19	9	24	41	22	4	8	46	15	

Fig. 3. An example of processed data from the force plate, tabulated by the 4051, to be used for later analysis.

The researchers are looking at three primary forces that are directly involved with foot and lower leg function during running. The vertical force is the force exerted downward by the body mass as a result of gravity. It is the largest force, and is related to the shock absorbing characteristics of the shoe. The other two forces that they are examining are

the anteroposterior and medial-lateral. The anteroposterior force is the force of propulsion and braking, exerted in the direction of travel. The medial-lateral force occurs across the foot, perpendicular to the direction of motion, and is related to the shoe's ability to provide rear foot control. This force occurs, for instance, as the foot rolls inward to absorb the vertical shock.

Once the researchers have gathered their test trial data (and the samples have passed the initial visual check), they must be evaluated by looking at the reactions of the various subjects to the different conditions. A preliminary step in evaluation is dividing the amplitude of the force data by the body mass of the runner, to normalize the data and allow comparison of data between different runners. The data samples are processed automatically by the 4051; copies are made with the Hard Copy Unit. Figure 3 shows a sample printout of processed data for 10 trials for one subject and one condition. Data curves are then prepared, graphing the force amplitude as a function of time.

Dr. Bates then examines the data statistically, using two methods of curve evaluation. The first method averages the curves and then evaluates the average curve. The second method identifies specific curve events on the original curves, and then computes averages for the events. These two techniques usually provide slightly different information.

As a part of the evaluation process, Dr. Bates uses the 4051 to compute 10 trial averages for all subjects under each condition. In addition, statistical analyses are performed using programs they've developed for repeated measures analysis of variance. The comparison information from the curves consist of a number of values:

- positive, negative, algebraic, and absolute cumulative impulse values,
- minimum, maximum, and average force values,
- total and average force and deviation values, and
- temporal characteristics of specific curve events.

All of these values can be obtained for the entire curve, for a portion of the curve, or for a series of successive time intervals.

Some Graphic Examples

Figure 4 is a graph showing the average vertical force curves of five key test conditions for one subject; Figure 5 shows the medial lateral force curves for the same

subject and conditions. For comparison, Figures 6 and 7 show the same curves for a single condition, for all subjects. A visual check of these curves reveals some interesting information. For instance, it appears that the differences between subjects for one given condition are greater than the differences between conditions for a single subject. Figures 6 and 7 provide additional detail, showing that different runners respond quite differently to the same shoe.

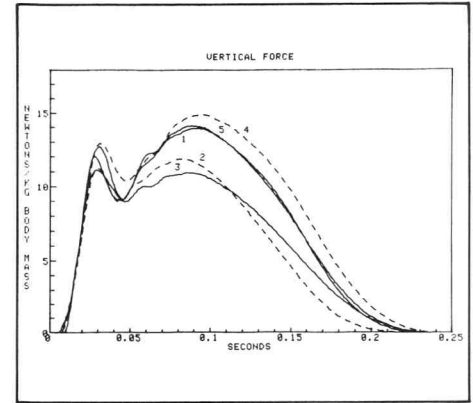


Fig. 4. Average Vertical Force curves for one subject, for all test conditions.

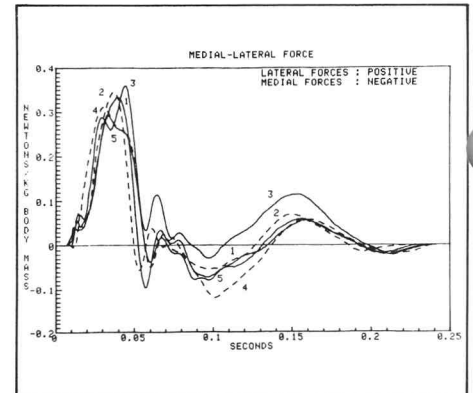


Fig. 5. Average Medial-Lateral Force curves for one subject, for all test conditions.

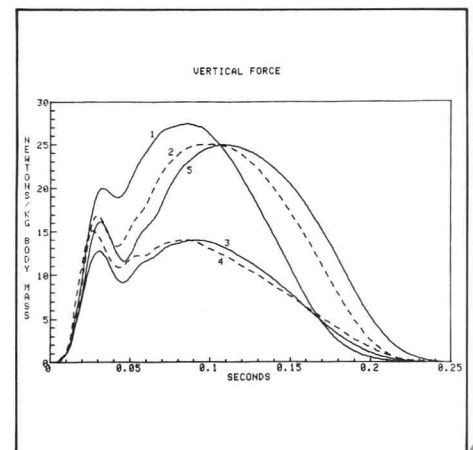


Fig. 6. Average Vertical Force Curves for one condition, for all test subjects.

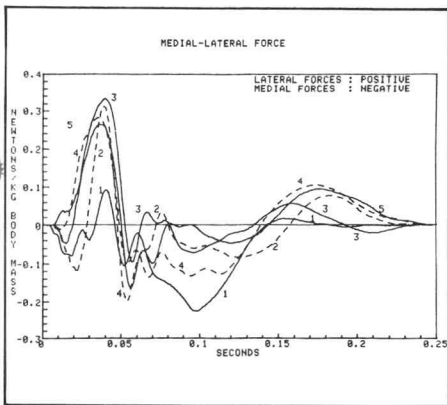


Fig. 7. Average Medial-Lateral Force curves for one test condition, for all test subjects.

Analysis of Variance statistical techniques were also applied to these curves. The analysis revealed a significant interaction between subjects and conditions, for each of the variables examined. This statistical check, performed with the ANOVA program written by Dr. Bates (see Applications Library Abstract #51/00-5702/00, *TEKniques* Vol. 4 No. 4), supported the initial observations.

Important Factors in Shoe Design

Some of the conclusions reached in this research have already been alluded to. In the vertical area, the first maximum vertical component relates to the shoe's ability to absorb shock at ground impact. The second maximum vertical component relates to forefoot loading. The average vertical force simply provides a composite rating of the shoe's ability to provide protection from this major force component. The vertical force component makes up about 85% of the total force impulse applied to the foot.

The average medial-lateral force is the factor that relates to stability and control during running. The average force for the period between 30—60% of the support period relates to foot control during that part of the support period when the foot is at or near maximum pronation.

From this research, one important fact has emerged, conflicting with previous shoe surveys: there is no one best shoe for all runners. Since all individuals are different, with varying anatomical and biomechanical performance characteristics, each interacts differently with a shoe. A best shoe, with this new data in hand, can only be designated on the basis of averages. So the one that has the highest average values for the important design factors identified might be considered best.

Even then, it must be realized that there will be individuals for which this "best average"

shoe will not be a good fit. But the characteristics that this research has identified can be used as a basis for fitting the correct shoe to each individual, on a scientific basis rather than a stylistic one. This will, hopefully, be especially important to those who find that injuries are interfering with their running performance and enjoyment.

Other Biomechanics Applications

Causes for a Fall. In a recent legal consultation, Dr. Bates was called upon to investigate the biomechanical aspects of a fall-related injury. In this incident, building construction had resulted in a temporary sidewalk ramp. Several falls happened at that location; one resulted in serious injury. The question raised was: Did the design of the ramp contribute to the fall? A ramp of identical angles was constructed in the Biomechanics Laboratory, and test subjects were filmed attempting to negotiate the ramp (Figure 8). The digitized film data was fed into the 4051 for analysis.



Fig. 8. A test subject attempting to negotiate the lab's ramp duplicate. The marks show key points to be digitized from film of the attempt.

The results supported the claims of the injured party. The ramp design was such that, as a person walked up the ramp, the force through the front foot at contact was directed upward in front of the body mass center, causing the body to be rotated backward and resulting in a fall. Figure 9 illustrates this unsuccessful attempt. In order to successfully negotiate the ramp, the subject had to make a conscious effort to lean into the ramp, so that the force would be directed through or behind the body mass center.

Green Chain Research. In the lumber industry, the job of handling green lumber is a common starting point in lumber mills. It is also a common source of back injuries. Dr. Bates has applied the same biomechanical

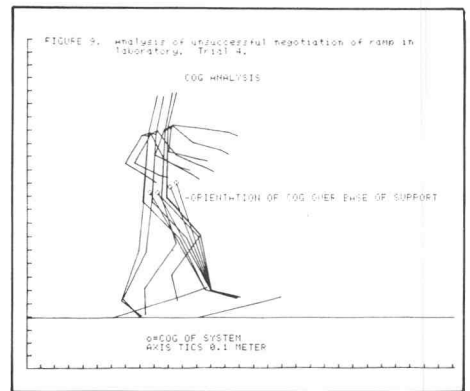


Fig. 9. A digitized representation of an unsuccessful attempt to negotiate the duplicate ramp. Note that the ramp throws the force ahead of the body mass, making it difficult to make it up the ramp.

analysis principles to the task of "pulling green chain" to define and better understand the demands of the job. Figure 10 shows initial analysis of the green chain task. The purpose of the evaluation was to design and manufacture a testing machine that would simulate the job performance requirements in order to identify, in advance, those candidates that possess the physical qualifications to perform the task successfully.

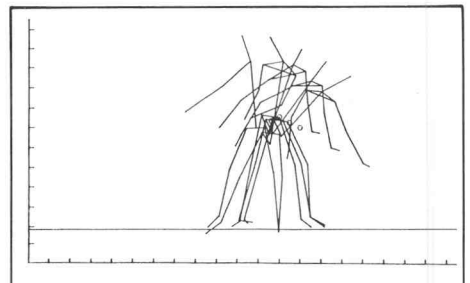



Fig. 10. A digitized representation of a worker "pulling green chain" in a lumber mill. This may later lead to testing equipment that will help prevent injuries in this job.

The Future

Biomechanics research is continuing at the University of Oregon and elsewhere, as an exciting field of research with down-to-earth applications. Each discovery of biomechanical relationships leads to new areas where those relationships can be applied. These applications lead to safer designs, increased productivity, and, in some cases, increased enjoyment. And Desktop Computer Graphics, through the 4050 Series Graphic Computing Systems, will continue to play an important part in graphing the data and graphically displaying the discoveries. 

Tekniques would like to thank Bob Bublitz, Tektronix Sales Engineer in Portland, for bringing this application to our attention. In addition, Bob put us in contact with the Eugene Criminal Justice Data System for the feature article in *Tekniques* Vol. 4 No. 3. And, of course, thanks also to Dr. Bates for his invaluable assistance in preparing this article.

New ROM Pack Adds Time Functions to 4052 and 4054

by **Dave Barnard**
Tektronix, Inc.
Wilsonville, OR

We've discussed new ROM Packs previously in *TEKniques*, but here's one that's different. For the most part, ROM Packs offer speed or convenience improvements over their BASIC program counterparts. But the new 4052R09 adds five new commands that have no equivalents in 4050 BASIC. The Real Time Clock ROM Pack combines hardware and firmware to add new timing possibilities to the 4052 and the 4054:

- **Time and Date**—set time and date after you power up the system, then forget it. Later you can ask, "What time is it?"; the ROM Pack will return the current time and date.
- **Elapsed Time**—set an elapsed time counter to zero, and start it counting in 0.1 second increments. You can then read its value. These steps let you measure the time between events in a program.
- **Vectored Time Interrupts**—set any desired time interval, with 0.1 second resolution. When the set time interval expires, your program can be interrupted (Like a User-Definable Key) by a branch to 4050 BASIC line number 84.

The new functions are accessed through the 4052 or 4054 processor with added commands, but the details are handled outside 4050 BASIC. Like other ROM Pack commands, these five new ones can be used from a program or directly from the keyboard.

Now let's look at the commands and see what they can do.

Time and Date Functions

Have you ever wanted to tag data with time and date of acquisition? Or programs with time and date of entry, or revision? Or have you ever wanted to annotate printer or plotter output with time and date, so you could later determine the most recent copies? Two of the five new commands answer these needs.

- **"SETIME"**—sets the clock time and date to initialize the ROM Pack.
- **"RDTIME"**—reads the current time and date from the ROM Pack, and assigns its value to a character (string) variable of your choice. Other 4050 BASIC commands can then manipulate the time and date string, or send the time and date to a peripheral.

Time is maintained in a 24 hour format in the ROM Pack, and is incremented every second. To set the time, you use a character string of up to 18 alphanumeric characters. For example, April 18, 1980 3:01:49 P.M. would be entered as follows:

```
100 CALL "SETIME", "18-APR-80 15:01:49"

100 AS="18-APR-80 15:01:49"
110 CALL "SETIME", AS
```

Fig. 1. CALL "SETIME" command initializes the 4052R09 ROM Pack.

If you prefer, the seconds can be omitted; setting the seconds is optional.

```
100 PRINT "ENTER TIME IN FORMAT DA-MON-YR HH:MM:SS"
110 INPUT AS
120 CALL "SETIME", AS
130 PRINT "The clock is set and counting. To verify use return."
140 INPUT AS
150 CALL "RDTIME", AS
160 PRINT "The time in the clock is now ", AS
170 END

RUN
ENTER TIME IN FORMAT DA-MON-YR HH:MM:SS
24-JUL-80 15:26:28
The clock is set and counting. To verify use return.
The time in the clock is now 24-JUL-80 15:26:26
```

Fig. 2. CALL "RDTIME" returns the current time once the 4052R09 ROM Pack has been initialized with the CALL "SETIME" command.

Figures 1 and 2 show the methods of setting and retrieving time data. The characters may be assigned to a string variable, or may be literally included as a string on the same line as the "SETIME" command. Either produces identical results. When the current time is read from the ROM Pack, the

information comes back in the same format that you use for input. This simplifies using the command in some situations, such as comparing time and date information in different files. And since the same conversion routine can be used for setting and reading the clock, it also simplifies the task of converting between time zones, or between AM/PM and 24 hour notation.

Elapsed Time Functions

In some situations, you might want to know how long your 4052 or 4054 is used, either as a stand-alone computer or as a terminal connected to a host computer. The 4052R09 lets you measure these times, and perform other timing tasks, simply. You can:

- Measure the time a peripheral instrument takes to provide data input, or handle data output, or
- Time computational routines to determine the comparative time each takes, or
- Measure the time a subject takes to respond to a problem or to a prompt displayed on the graphic screen.

The Elapsed Time commands were designed to answer these needs:

- **"STARTW"**—Sets the elapsed time counter to zero and starts it running, like a stopwatch, in 0.1 second steps.
- **"STOPIT"**—Reads the value of the elapsed time counter, assigns the value to any chosen numeric variable, and resets the counter.

You can try out the commands with the program shown in Figure 3. Just insert your favorite program as a subroutine, beginning at line 1100 and ending with a RETURN statement. The ROM Pack and the four program lines do the rest. Line 100 of the program resets the counter and starts it running. Using a GOSUB to the test program doesn't add to the total time by a measurable amount. When the test program is terminated by the RETURN, the program resumes running at line 120, where the clock is read into the variable. The time is then printed in line 130.

```

1 REM BENCHMARK PROGRAM
100 CALL "STARTW"
110 GOSUB 1100
120 CALL "STOPIT",T
130 PRINT "THE PROGRAM COMPLETED IN "T;" SECONDS"
140 END
1100 REM THE PROGRAM BEING TESTED STARTS HERE
1110 REM EXAMPLE SIN(X)/X
1120 FOR I=1 TO 1000
1130 X=I/1000
1140 S=SIN(X)/X
1150 NEXT I
1160 RETURN

```

Fig. 3. Elapsed time may be measured using CALL "STARTW" and CALL "STOPIT" commands.

More Than Connect Time Printouts. You can do more than just get a print-out or a hard copy of your connect time. With the 4052R09 and the fundamentals of the Data Communications Option (Opt. 1 or Opt. 3), you can make your 4052 or 4054 aware of how much of that connect time is actually being used. To some users of very busy computers, this can be important for several reasons. For instance, if a loss of communication occurs, your connect time may not agree with your monthly statement. In other situations, the problem is loss of communication due to a time limit on usage; you might have to try to limit your access to some number of minutes. The Real-Time Clock ROM Pack can answer the question: how much time did you really get that session?

The short program in Figure 4 takes care of everything except dialing the computer and setting the communications parameters. Line 110 starts the timer. Remember, when you enter Terminal Mode from a running program, returning to BASIC resumes program operation at the next program line. You can return to BASIC by pressing the "RETURN TO BASIC" function key, or your host computer can perform the task with the ESC ESC sequence, as described in TEKniques Vol. 4 No. 3.

```

100 REM DATA COMMUNICATIONS TIMER
110 CALL "STARTW"
120 CALL "TERMIN"
130 CALL "STOPIT",N
140 PRINT "CONNECT TIME WAS "N;" SECONDS"
150 END

```

Fig. 4. Using the 4052R09 with the Data Communications Interface enables you to determine connect time.

Vectored Timed Interrupt Functions

Perhaps your application requires you to:

- Divert the system periodically from a main task to a background task, or

- Scan instruments or other peripherals, or alert someone using the 4052 or 4054, at periodic intervals, or

- Restart the system automatically from idle after an error terminates the program.

If one of these needs is yours, this last command is for you. The command is:

- CALL "ONTIME",T where T is the time interval from the execution of the command to the program interrupt.

Executing this command sets the timer to the specified number of seconds, T in this case; time can be specified to the tenth of a second. When that program line is executed, either in a program or from the keyboard, the ROM Pack begins counting down the interval you set. There is no immediate effect on the program; subsequent lines will be executed as if nothing had happened. But when the time elapses, if the interrupt is enabled (through SET KEY), the program will perform a forced GOSUB to 4050 BASIC line number 84, the timed interrupt vector location.

This function works just like a User-Definable Key. Like a UDK, the interrupt is enabled by the SET KEY command and disabled by the SET NO KEY command. If the interrupt is disabled when it occurs, it, along with any UDK interrupts, will be remembered in the sequence in which they occurred. Further details on programming with the User-Definable Keys can be found in TEKniques Vol. 3 No. 6.

Dividing Up the 4052 or 4054. The beauty of the ROM Pack timing functions is that they require only the time necessary to start the function in the program. Thereafter, the ROM Pack handles the rest, with little attention from the desktop computer's processor. So now you can have timing loops without the waiting of the WAIT command. You can use this timing loop function to divide up the processor's attention between tasks. The program listing in Fig. 5 shows how the timed interrupt can divide the time your system spends between two different tasks.

Lines 84 through 90 are set up as an interrupt handling routine; it includes a branch to a secondary program every 45 seconds. The main task can be any program you might normally run, preceded by a CALL "ONTIME" to set the time interval allowed it. The sample time interval is arbitrarily set to 45 seconds, but could have been any other value sufficient to give the primary task the majority of the time.

```

1 INIT
2 GO TO 1000
84 SET NOKEY
85 REM GO DO THE OTHER TASK PRINTING A TAPE FILE
86 GOSUB 300
100 REM TASK MONITORING ROUTINE
110 IF S=1 AND M=1 THEN 220
120 IF S=1 OR M=1 THEN 170
130 SET KEY
140 CALL "ONTIME",45
150 RETURN
160 REM IF EITHER TASK IS DONE FIGURE OUT WHICH AND PRINT A MESSAGE
170 IF S=1 THEN 200
180 PRINT "MAIN TASK IS COMPLETE NOW SUB TASK GETS ALL THE TIME"
190 GOSUB 300
195 GO TO 220
200 PRINT "SUB TASK IS COMPLETE NOW MAIN TASK GETS ALL THE TIME"
210 RETURN
220 REM BOTH TASKS ARE DONE
230 PRINT "BOTH TASKS ARE COMPLETE"
240 END
300 REM SUB TASK PRINTING A TAPE FILE
310 IF TYP(0)=1 THEN 300
320 FOR I=1 TO 20
330 INPUT #33:AF
340 PRINT #51:AF
350 NEXT I
370 RETURN
380 S=1
390 RETURN
1000 REM MAIN PROGRAM FIRST TIME ENTRY POINT
1010 DIM X(1000)
1020 M=0
1030 S=0

1040 GOSUB 140
1050 REM FOR EXAMPLE, GENERATE A BUNCH OF NUMBERS AND PRINT THEM
1060 FOR N=1 TO 100
1070 X(N)=RND(1)
1080 X(N)=LOG(1+TAN(ACS(SIN(X(N))))))
1090 PRINT N,X(N)
1100 NEXT N
1110 REM TASK COMPLETE
1120 M=1
1130 GO TO 100

```

Fig. 5. Dividing up the 4050's time:

- (1) The programs begin at statement 1000.
- (2) Statement 1040 branches to the routine which enables 'ON TIME' interrupt and the countdown time, then returns to begin main task.
- (3) After 45 seconds the program is interrupted with an implied GOSUB to statement 84.
- (4) The processor is directed to the subtask beginning in statement 300, after disabling interrupt.
- (5) After 20 lines of code input, the program is directed back to the monitoring routine in statement 100
- (6) If neither task is complete, it will branch back to main task for another 45 seconds. If main task is complete, it will devote full time to subtask. If subtask is complete, but not the main task, it will devote full time to the main task.

If it looks simple, it is. It works just that easily! And you can use the same technique to sequentially scan instruments on a timed basis.

Restarting the System From Idle. Suppose you have a task that includes gathering data. If an error occurs that is "fatal" to the program, program execution will end. Unless you happen to notice, no further data will be collected until you take corrective action. This poses an added problem when the system must operate unattended, such as during overnight operation. Now if the error is an interface problem that hangs up the system, there is little you can do. But if it is the type described, that brings the program to an end unexpectedly, the Real Time Clock ROM Pack can bring the system back to life gracefully.

First you need to know how long it normally takes to perform a major portion of the program. (The other functions of the ROM Pack can help you measure this.) Then you add an occasional CALL "ONTIME" in the program. Since each CALL "ONTIME" resets the interrupt timer, the added statements continue to reset the interrupt timer before it can cause an interrupt (as long as the program continues to run). The time value set should be greater than the time it normally takes the program to reach the next CALL "ONTIME" statement.

So long as the program continues to run, and catches the interrupt timer before the interrupt occurs, operation will proceed normally. However, if a "fatal" error occurs, the program will stop; when it does, it will also fail to reset the interrupt timer. The resulting interrupt will cause the program to start running, beginning at line 84; it will keep running until the next time an error occurs.

You can also look at variables that serve as flags in the program. Then, when a restart from IDLE occurs, the program can detect how it got there and where it was before the error occurred; the restart process is even safer. The example in Fig. 6 illustrates this technique.

In the example, the program also sets a time limit for the statement that requires keyboard input to be complete. (This is something else now—the input can be interrupted.) The variable takes whatever it

```

1 INIT
2 SET KEY
3 GO TO 1000
84 REM USE THE SEMI-PHORE TO SEE WHERE THE PROGRAM QUIT
85 REM A= PROGRAM NUMBER
86 GOSUB R OF 1000,2000,3000
87 REM PRINT AN ERROR MESSAGE INDICATING WHERE THE ERROR WAS
88 REM HAVING HANDLED THE INTERRUPT GO TO THE NEXT SEGMENT
100 REM THIS PROGRAM SEGMENT HANDLES THE ERROR IF IN PROGRAM 1
110 PRINT "FATAL ERROR OCCURRED IN PROGRAM 1"
120 PRINT "CHECK INPUT " "M" "X"
130 RETURN
200 REM THIS PROGRAM SEGMENT HANDLES THE ERROR IF IN PROGRAM 2
210 PRINT "NO OPERATOR ENTRY IN PROGRAM 2"
220 RETURN
300 REM THIS PROGRAM SEGMENT HANDLES THE ERROR IF IN PROGRAM 3
310 PRINT "A FATAL ERROR OCCURRED IN PROGRAM 3"
320 RETURN

1000 REM PROGRAM 1 SET THE TIMER TO 10 SECONDS LETS MAKE AN ERROR
1010 A=1
1020 CALL "ONTIME",10
1030 GOTO 2000

2000 REM PROGRAM 2 SET THE TIMER TO 5 SECONDS KEYBOARD INPUT TIME-OUT
2010 A=3
2020 CALL "ONTIME",5
2030 INPUT Z$
2040 PRINT Z$
2050 GO TO 3000

3000 REM PROGRAM 3 SET THE TIMER TO 1
3010 A=3
3020 CALL "ONTIME",1
3030 PRINT @1:"INTERFACE CLEAR"
3040 PRINT "LAST SEGMENT COMPLETED"
3050 SET HKEY
3060 END
4000 PRINT "PROGRAM ENDED"
4010 END

```

Fig. 6. Restarting the system:

- (1) Routine 1 (1000-1030) generates a fatal error. After 10 seconds the Real Time Clock forces a GOSUB to statement 84 and starts the program.
- (2) Routine 2 (2000-2030) monitors the keyboard for 5 seconds, then forces a GOSUB to statement 84 and continues the program.
- (3) Routine 3 (3000-3030) tries to output to an off-line device but at the end of 1 second will be forced to statement 84 and program continued.

can get from the keyboard; interrupt processing then begins at line 84. This is important in order to let instruments be scanned in the normal operation of the system. This also shows how the operator can be made "optional" in some cases, providing the overriding input but not the pace for the system. So the operator doesn't have to be alerted just to provide a default response that makes no change to system operation.

More Information

The specific command formats are shown in Fig. 7. They may answer most other questions. The examples in this article were intended to show some of the totally new possibilities provided by 4052R09, the Real-Time Clock ROM Pack. For other information, contact your local Tektronix Sales Engineer.

```

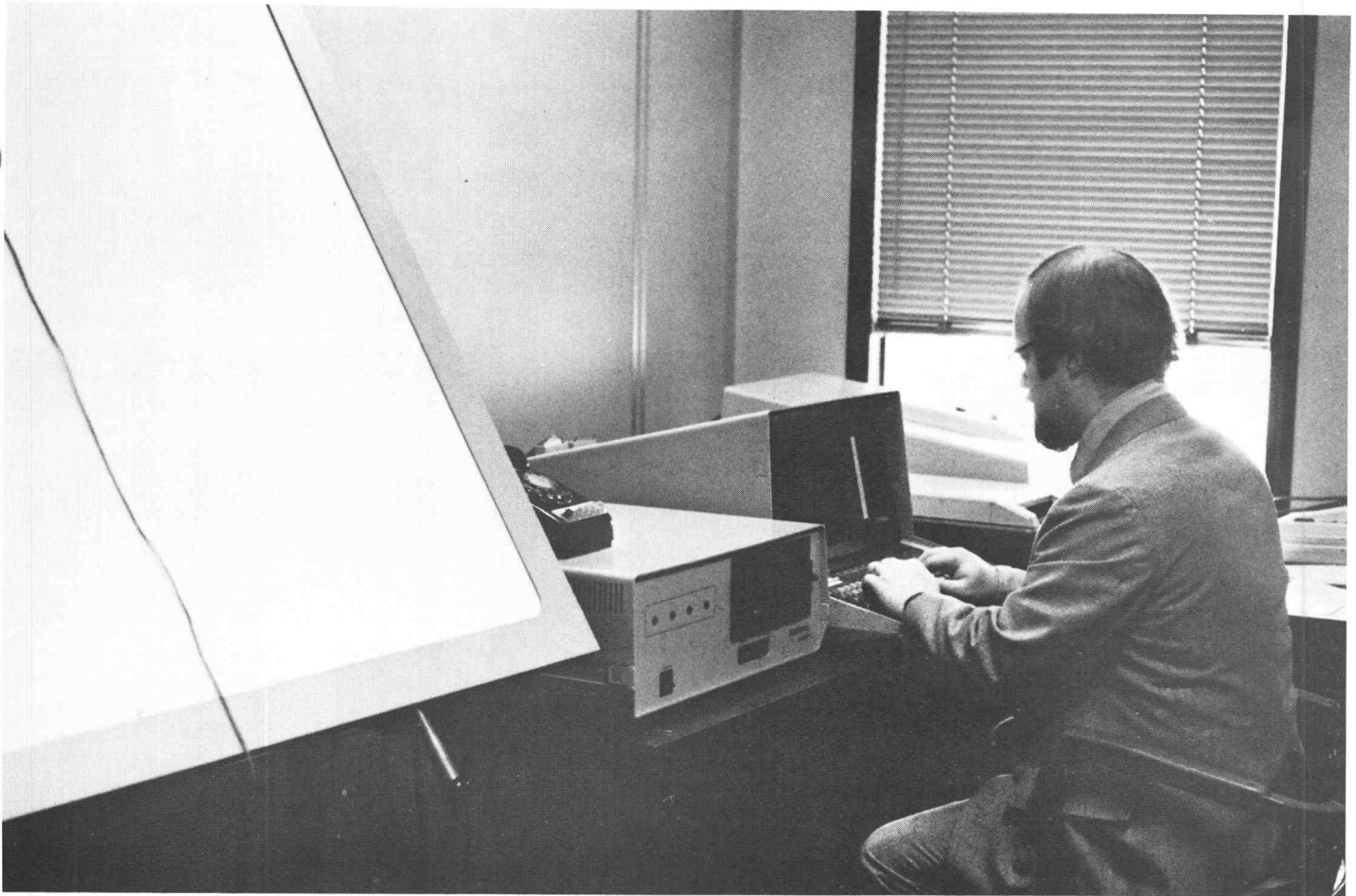
CALL "SETIME",A#
CALL "ROTIME",A#

CALL "STARTH"
CALL "STOPIT",N
CALL "ONTIME",N

A# IS A STRING VARIABLE
N IS A NUMERIC VARIABLE

```

Fig. 7. The 4052R09 adds five new commands to your 4052 or 4054 System.



Computer simulation to determine effects of wind shear on aircraft keeps a 4051 system busy at the Federal Aviation Agency in Washington, D.C. Herb Schlickemaier of the Agency's Systems Research and Development Service says the program has really used the 4051 to its best advantage.

Avionics Research Using the 4051

by Patricia Kelley
TEKniques Staff

TEKniques Vol. 4 No. 4 took a behind-the-scenes look at how the 4050 Series is used at the National Air and Space Museum of the Smithsonian Institution in Washington, D.C. While in the area we ventured down the street to the Federal Aviation Administration (FAA) to see how they were applying their 4050 System.

We talked with Herb Schlickemaier of the Systems Research and Development Service (SRDS) of FAA. His particular branch is the Airborne Guidance and Control Group where they are primarily engaged in research and development of avionics, i.e., aircraft electronic systems.

Wind Shear Analysis

One project which is nearing completion is the Wind Shear Analysis program, and is one in which Herb says the 4051 has really been used to its best advantage.

The research was an initial effort to define specific wind shear conditions that pose hazards to particular types of aircraft. From this research "hazard envelopes" could be developed and included as part of an aircraft's hazard evaluation and warning

¹A system including many components such as ground-based atmospheric sensors, ground-to-air data link, aircraft-based sensors and controllers and so on.

²Flaps, gear positions, and others.

³Flight phase, i.e., final approach, take-off, climb; true airspeed, flight path angle, etc.

system.¹ These envelopes would enable the system to correlate various configurations² and flight parameters³ for that aircraft with wind shear conditions which might constitute a hazard. In a fully automated system, this flight-derived data would actuate the automatic flight control system (AFCS).

But what is wind shear? Simply put, wind shear is a rapid change in either wind speed or direction. Wind shears normally are formed by thunderstorms and frontal zones. The most severe shears usually occur in gust fronts at the edge of mature thunderstorms. Strong shears also occur in the frontal zone between cool and warm air masses; low level jet shear can occur by warm airflow atop a low level temperature inversion.

Wind shear has been the culprit in a number of aircraft mishaps, such as those that

occurred in Tokyo in 1966, LaGuardia in 1971, Logan International in 1973, JFK in 1975, Philadelphia in 1976, and others. Figure 1 is a profile of a commonly occurring wind shear, called a logarithmic shear; Figures 2 — 6 are profiles that have been reconstructed from aircraft accidents. According to certain distinguishing qualities, wind shears have been broadly categorized into four types: linear, log, jet stream and reversal (Figure 7).

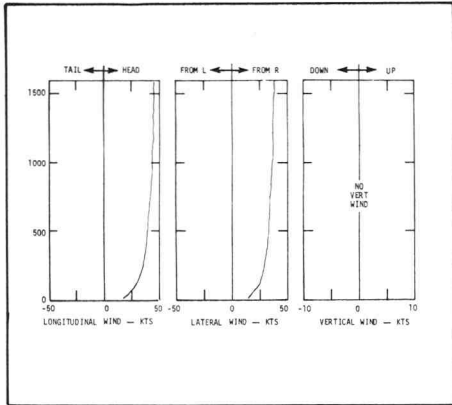


Fig. 1. Wind Profile 1 (Neutral)

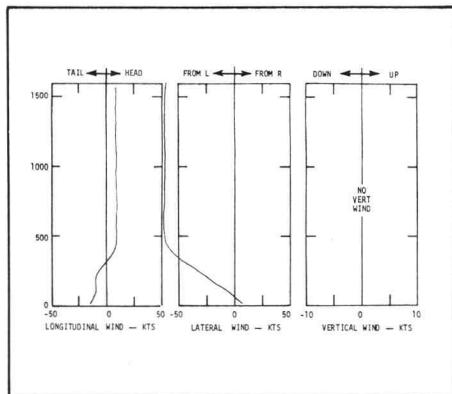


Fig. 2. Wind Profile 2 (Frontal). Similar to Tokyo (1966)

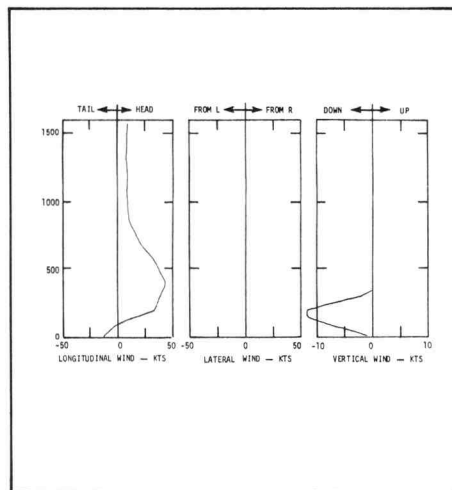


Fig. 3. Wind Profile 4 (Thunderstorm). Similar to Philadelphia Profile (1976)

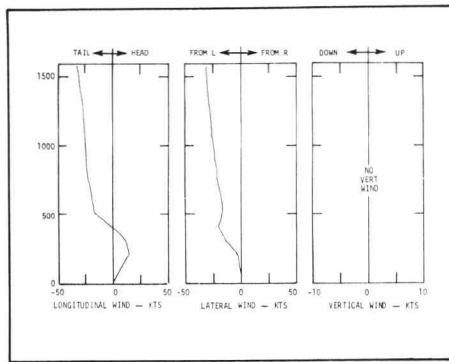


Fig. 4. Wind Profile 5 (Frontal)

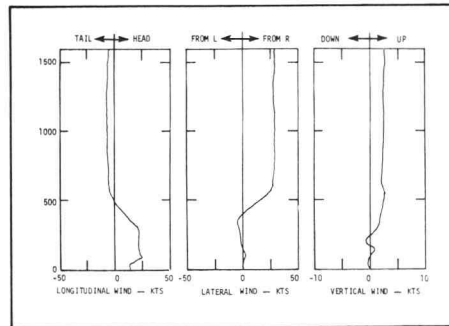


Fig. 5. Wind Profile 9 (Frontal)

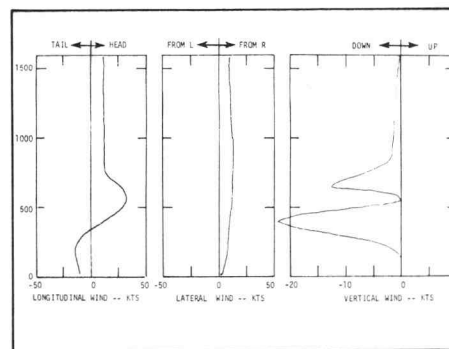


Fig. 6. Wind Profile 10 (Thunderstorm). Similar to Kennedy Profile (1975)

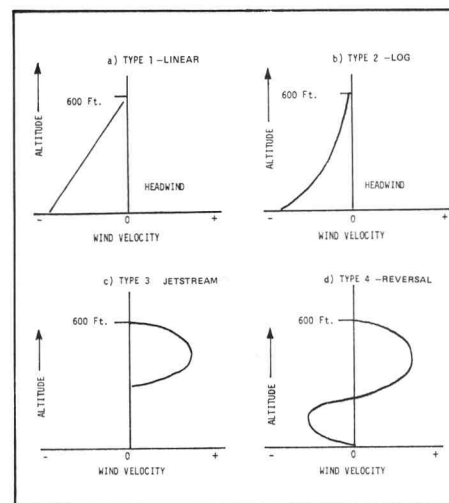


Fig. 7. Categorized Environments

Simulating Wind Shear Hazards

Since wind shears have different characteristics, and because a particular wind shear will have different effects on unlike aircraft, no one formula for what is hazardous can be applied to all. Therefore, defining which types of wind shears constitute a hazard requires a variety of parameters to be tested. This is where the 4051 comes in.

Computer simulation is the primary tool for hazard definition analysis. It allows for a variety of aircraft configurations, flight paths, and atmospheric disturbance models to be examined in various combinations. By taking National Transportation Safety Board reports describing aircraft configurations, flight paths, and wind conditions at the time of aircraft accidents, adverse combinations of actual conditions can serve as computer simulation check points.

For one analysis, an aircraft model of a wide-bodied three-engine jet was studied. The model was selected as being representative of a category that offers the most potential for catastrophe in a severe atmospheric disturbance, i.e., a commercial airliner.

The analysis was conducted with the computer-simulated model landing under simulated automatic control (autopilot and autothrottle). Wind shear profiles were selected that were representative of actual encounters with severe wind shears.

The aircraft model was considered successfully landed through the wind shear condition if the simulated control actions demanded to correct flight path deviations did not exceed the actual capabilities of the aircraft. Also, touchdown and airspeed must have been within limits. If these criteria were not met, a hazardous condition was presumed.

Many simulations were run on the 4051 for each of the four wind shear categories shown in Figure 7. Deviations from the programmed values of the flight parameters, whether caused by wind shear or other factors, were corrected by *elevator* and *throttle*. Results of some of the runs are displayed in Figures 8 through 11. Notice that each display shows the maximum and minimum values of:

- elevator deflection (DELE) in radians
- true airspeed (VA) in ft/sec
- normalized throttle position (TH)
- elevator rate (dDELE) in radians/sec. and
- deviation from the desired altitude (ZTD) just prior to landing
- deviation from the desired touchdown (XTD)

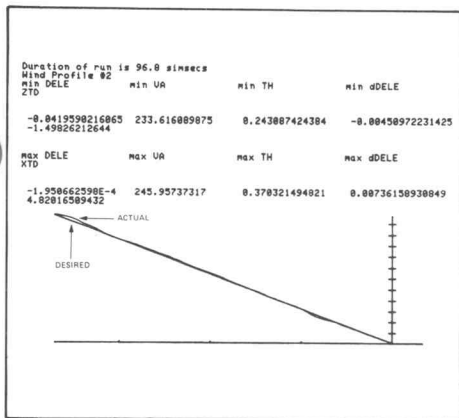


Fig. 8. Simulation Wind Profile No. 2.

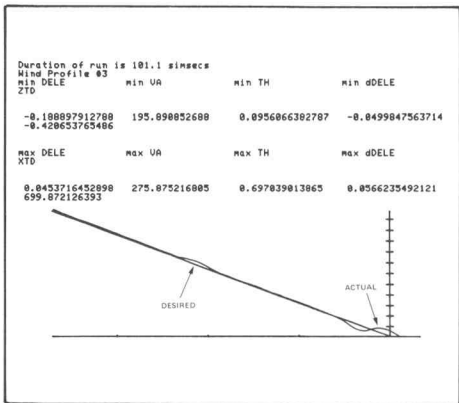


Fig. 9. Simulation Wind Profile No. 3.

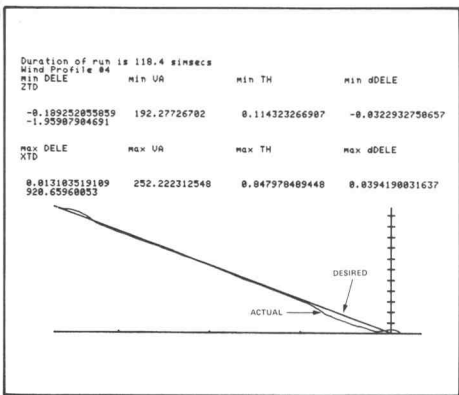


Fig. 10. Simulation Wind Profile No. 4.

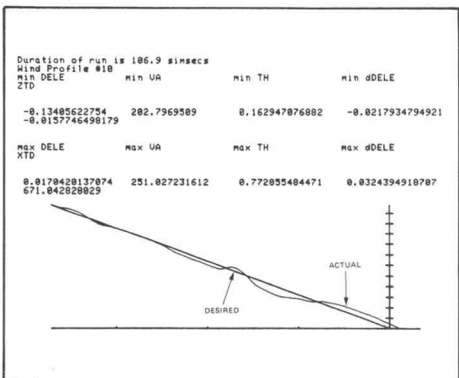


Fig. 11. Simulation Wind Profile No. 10.

Although the aircraft was considered safely landed in Figures 8 through 11, all landings weren't so successful. The results laid the basis for a proposed hazard definition.

Defining Wind Shear Hazards

From the simulations, Herb drew the following conclusions applicable to a jumbo tri-jet aircraft in automatic flight control system (AFCS) mode on a three-degree flight path.

The Type 1 (linear) shear illustrated in Figure 7a can be penetrated safely and does not pose a hazard during final approach and landing.

The Type 2 (log) shear illustrated in Figure 7b was penetrated safely during 89% of the simulated runs. When the surface wind in this log distribution was a 100 ft/sec tailwind, it constituted a hazard, and the defined conditions should be included within the hazard envelopes.

The Type 3 (jet stream) shear illustrated in Figure 7c was penetrated safely during only 36% of the attempts. The most severe shears that were safely penetrated were a 200 foot vertical segment having a center headwind of 40 ft/sec, and a 600 foot vertical segment having a center headwind of 50 ft/sec. Jet stream wind shears exceeding these speed magnitudes should be included within the hazard envelopes.

The Type 4 (reversal) shear illustrated in Figure 7d was penetrated safely during only 10% of the attempts. The most severe thunderstorm shear that was safely penetrated had a center headwind of 40 ft/sec in its 450 foot lower segment, and a center tailwind of 40 ft/sec in its 450 foot upper segment. Thunderstorm shears with intensity magnitudes in that vicinity should be included within the hazard envelopes.

Alleviating the Wind Shear Hazard

From these results, the SRDS group is working to develop systems which can warn the flight crew of an impending hazardous environment. For instance, they are currently working on a telemetry system that will automatically tell the pilot what the winds are at the touchdown zone.

They are also working on inertial velocity (groundspeed) sensors that will tell the pilot his aircraft's speed with respect to the ground as opposed to the air mass that it's flying through. For example, if a headwind increased slowly to some large speed, the pilot will change his inertial velocity unknowingly.

If this headwind suddenly disappeared, the aircraft wouldn't have enough lift to maintain flight. So, if he monitors his ground speed, it will tell him there's a certain limit beyond which he shouldn't go.

The group is also looking at the aerodynamic limits of the aircraft. Herb explained, "If you instantaneously remove 20 or 30 knots from the airspeed of the aircraft, or if the angle of attack increases by 15 or 20 degrees, you get into what is called a stall. The wings no longer can support the weight, in which case the aircraft is going to start to nose over and dive. Structurally the aircraft is still sound. For a short time the aircraft will still fly close to the same velocity with respect to the ground, since you don't stop a 200,000 pound aircraft instantly. But, aerodynamically, dramatic things have happened to the aircraft and we want to see how the atmosphere affects it."

To do this, Herb's group is looking at a variety of signals that the aircraft is detecting from a math model. Did the angle of attack start to oscillate as it approached the stall angle; what's the airspeed starting to do; is it starting to drop off; can you see a creep before it instantaneously drops; is there a pattern you can pick up?" From these signals they can find out how the environment is affecting the aircraft.

Then, by knowing which of these signals are important, or which seem to give the most information, they can determine how to build a "box" that looks at those signals and displays them or feeds them back into one of the flight computers on the aircraft. They have done some simulations of such a "box", called an acceleration margin system, which looks at about five different signals from the aircraft. They haven't built it in hardware, however, Herb says that simulations on the 4051, on the Eclipse⁴, and piloted simulations have proven it to the point where they can hand off the specifications to another group for manufacture.

Head-Up Display Program

Another simulation analysis is the head-up display. Flight information is displayed on the windscreen for the pilot so he doesn't have to look down; thus, he's always looking out. Because of the work in wind shear, when the head-up display program started, they were able to take the head-up display control laws and play them through the tri-engine jet model on the computer. They played it through the Eclipse onto a TEKTRONIX

⁴The Eclipse is a 16-bit minicomputer manufactured by Data General Corp.

4006 terminal screen and put up the symbology. Herb noted that although it was under automatic control, they were able to see where the symbology would have moved on the windscreen as the aircraft moved through the wind field. He wryly observed, "Unfortunately, it wasn't dynamic graphics; with a 4054 or 4081 everything would have been as it was in the real world." But it did give them a good feel as to what gains had to be turned around, and better insight as to what sensitivity had to be attached to certain control laws.

Other Tasks

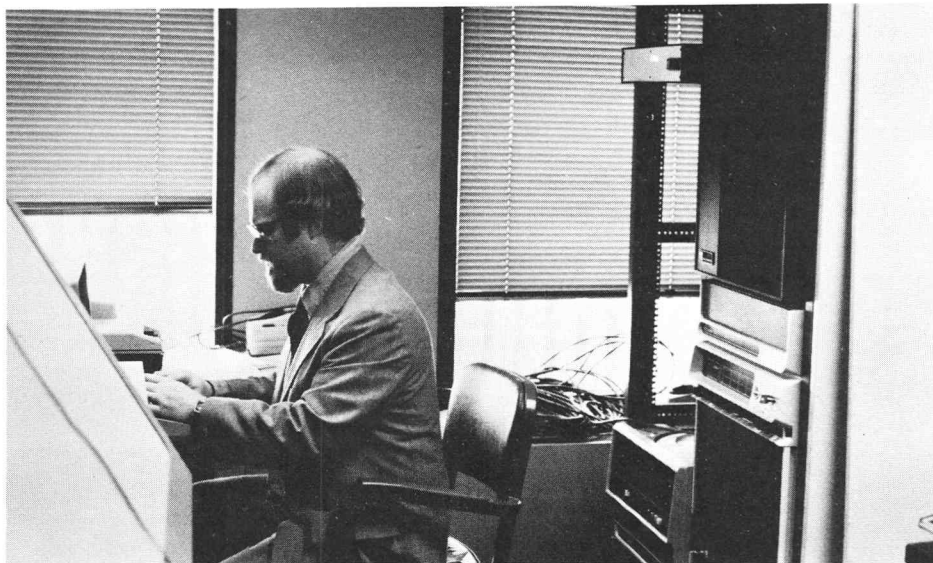
Although the 4051 has been doing a lot of work on how the environment affects aircraft, it is also put to other uses. Herb says they've developed a few programs which design active filters. They've also used the 4051 for printed circuitboard layout.

Another area in which they employ the 4051 is occasional interacting with contractor computers. A contractor may have a data bank that SRDS wants to examine. "It's nice to be able to retrieve a data bank and not be on-line and impact the contractor," noted Herb. They download from the host to the 4051. They can process it if they want to take a look at some of the work the contractor has been doing. Herb said it had given them a big advantage in a couple of contracts; it made the difference in whether a contract really recovered any intelligent data or not. They were actually able to tell the contractor specifically where problems existed.

Herb has also been using the text editing capability of the 4051. He saves all of his FORTRAN text on the 4051, using the Editor ROM, thus relieving the Eclipse. He has written a program that automatically compiles the FORTRAN program, performs the error checking, and all the rest, down to the map⁵; it even checks the map to see that it passed all the tests. At this point, it's interactive with the Eclipse. Herb commented, "It basically takes the operator out of the loop; you're there as a monitor, if you're interested." The 4051 takes the data files off the disk, dumps them onto the RS-232 and on into the Eclipse at the proper baud rates (the Eclipse requires its data to be spaced out in a certain fashion). The 4051 then checks to see what the return is from the Eclipse to determine whether the compilation was good or not.

SRDS has engineering students from the cooperative education program (work-

⁵A compiler translates higher level languages to a contiguous machine code. A map, sometimes called a SAVE file or absolute code, combines the machine language codes into one executable program.



For other tasks, Herb has the 4051 working interactively with their Eclipse minicomputer.

study) working with them. Herb said the students, as well as others in his group, are enthusiastic in using the 4051. "They don't have that big brother fear of the large main frame," he noted, "nor do we have the operating costs." One of the students, Joe W. Ascavage, built a wake vortex simulation for two aircraft on the 4051. This basically figured out what the vortex position shedding off the wing was for the lead aircraft. Then, depending on the speeds and the initial spacing of the two aircraft, you could see whether the follow aircraft ever interfered with the first aircraft's wake. If it did, you would tell what the intensity was and other parameters. This program has been used for some of the analysis the SRDS is putting into wake vortex programs. It's also used to a certain degree to determine what type of spacing is safe for various aircraft.

Herb observed that initially they simply started the engineering student with some basic concepts, as to what salient information was really important; after that he was essentially on his own. Later the student came out of the computer room with a very nice piece of software.

While attending the University of Maryland, Herb was a co-op program student himself. He worked through college as a graphic arts draftsman. He commented he had seen too much work done by hand that should have been run on the computer, thus, another reason he is so enchanted with the 4051.

In school he majored in Aeronautical Engineering but became very interested in computers so he took as many Electrical Engineering courses as his schedule permitted. He feels the experience he gained developing applications for the university

computer has really come in handy. In fact when SRDS picked up the Eclipse there were no plotting packages on it. Instead of pushing through a software contract, he built up a 4051 compatible plotting program in FORTRAN for the Eclipse which enables them to take any program in BASIC and with the proper syntactical changes (e.g., DRAW becomes CALL DRAW), the same plot will be displayed on the 4051 from the Eclipse.

This allows his group to write graphics programs initially on the 4051, de-bug them, put them back into FORTRAN, compile and execute. He said, "Now all you have to do is get your numbers straight."

Occasionally the SRDS group has to work on rush orders. Once a fellow came in who wanted information about the effects of a particular type of wind shear that they hadn't been looking at. So they took it to some rather ridiculous extremes and showed that it didn't significantly affect the aircraft. The data came off the Eclipse in this case; so they "scratched up a quick and dirty program on the 4051, dumped the data in, hit the copy button, and handed the plot to the guy." Herb notes. "That impressed a few people."

His final comment, "But usually it's the same old stuff. We keep the 4051 busy."

Ed. Note: The Wind Shear Analysis study is covered in Report No. FAA-RD-79-90, "Wind Shear Hazard Definition For a Wide Body Jet," Herbert W. Schlickemaier. The document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161.

We want to thank Herb Schlickemaier for taking time to discuss some of his applications with us. A special thanks goes to Mallory Green at the Department of Housing and Urban Development for suggesting we contact Herb to discuss his unusual applications.



Editor's Note



Back Issues: Vol. 4 Only

This is just a note to remind you that back issues of TEKniques are no longer available for Volumes 1, 2, and 3. The application articles from those volumes have been compiled into reprint sets, collected by application area. Currently there are five sets of reprints available, in the following application categories:

Engineering and Design AX-4449
Mapping AX-4460
Data Acquisition and Analysis .. AX-4450
Business Graphing and
Reporting AX-4451
Peripherals and ROM Packs ... AX-4452

If you need an article from one of these previous volumes, and don't have your copy, one of the reprint sets will likely fill your needs. To obtain a copy of one of the reprint volumes, just contact your local Tektronix Office or the Applications Library Office serving you.

And, of course, back issues of TEKniques Vol 4 (1980) will continue to be available from the 4050 Series Applications Library office that serves your area.

Programming Tip Handbook

We've got a new Programming Tip Handbook, too. This handbook contains all of the Programming Tips and BASIC Bits from the past three volumes of TEKniques. Since they've proven to be such a valuable reference to many of you, and since reprints of those volumes are no longer available, we put the handbook together to answer that need as well. And the Tips are all indexed so that you can find what you need in a number of ways.

The Programming Tip Handbook is available through the Applications Library office serving your area. It's listed in the Resource Materials section of the new 4050 Series Applications Library Catalog, as Abstract Number 51/00-7004-0. U.S. Domestic Price is \$10.

Like the New Catalog?

By now, you should have received your copy of the new 4050 Series Applications Library 1980 Catalog. We put a lot of hours into revising the old catalog entries and adding the 66 new abstracts to make up this catalog; our view may, therefore, be a little biased.

So now it's your turn to tell us what you think of the catalog. Drop us a note and share any of your views about the catalog with us. Do you think the choices of categories are appropriate, for instance? Or how do you like the new keyword index? Is it easy to use? Let us know; we're actively seeking your opinion.

International User Applications

We're also actively looking for application articles describing how users outside of the U.S. are applying 4050 Series Graphic Computing Systems. If you're one of those users, please drop us a line through your Applications Library office.

As usual, we're happy to help with editing or rewriting, as necessary. Just let us know what you're doing, and what you need to complete an article. We're waiting to hear from you.

New Contest

As we announced in the previous issue of TEKniques, we're going to have another contest in the near future. Details, including contest area, rules, and prizes, will be announced in the next issue of TEKniques. Watch for it.

Programs Wanted

We're always looking for more programs to add to the 4050 Series Applications Library. New programs keep the library growing, so we are able to share more programs with more users. And, of course, you who use the systems are the source of new programs.

If you have a program to contribute, check the information at the beginning of the Abstracts section for information. And remember, you get three programs for each one accepted by the library, so you can't lose.

Programming Tips Too

We'd also like to remind you that we like to publish your Programming Tips and BASIC Bits too. You get any one of 12 Programs from the library for each of your Tips or Bits. Details are listed below

Telephone Note

If you've tried to phone the Applications Library recently, you may have experienced some confusion over extension numbers. Recent changes in the Tektronix telephone system have resulted in duplicate extension numbers in Beaverton. To avoid any confusion just remember:

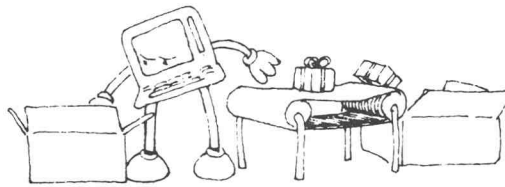
- If you're calling from outside Tektronix, call (503) 682-3411—then ask for the library extension you want.
- If you're an inside caller, or you've already called the Beaverton plant, just let the operator know that you want a Wilsonville extension. Ask for 3607—Wilsonville, for instance, to avoid confusion with other extensions.

Programming Tip Exchange

51/00-0501/0	51/00-6002/0
51/00-0901/0	51/00-8004/0
51/00-1403/0	51/00-8017/0
51/00-1603/0	51/00-8022/0
51/00-4002/0	51/00-9507/0
51/00-5204/0	51/00-9533/0



INPUT / OUTPUT



Digitizing on 4663

Reed Phillips, Systems Analyst at Tektronix, Raleigh, NC. writes:

I had a customer that was having difficulty understanding how to digitize from the 4663. The enclosed program will shed some light on the subject.

```

100 INIT
110 REM Set Response Format to 5
120 PRINT #1,32:"CC 5"
130 PAGE
140 ON SRQ THEN 200
150 WAIT
160 GO TO 150
170 REM z=0 means MOVE
180 REM z=1 means DRAW
190 REM z=2 means LAST POINT
200 INPUT #1,32:X,Y,Z
210 GO TO Z OF 240,270
220 MOVE X,Y
230 RETURN
240 DRAW X,Y
250 RETURN
260 REM Reset Response Format to 1
270 PRINT #1,32:"CC 1"
280 END
    
```

The POINT light will turn on indicating that you may begin. The first point digitized should obviously be a MOVE POINT. Depressing LAST POINT will terminate the program.

Option 1 (RS-232) Interfacing

Various users have had similar questions about using the Option 1 (RS-232) Data Communications Interface and their 4050 Systems. Howard Sanders, Technical Support Specialist, and Frank Lees, Communications Support Specialist, at Tektronix, Wilsonville, have provided the answers to some of the more frequent questions.

When I connect my 4050 System to a modem and key in CALL "TERMIN", the BUSY and I/O lights on the 4050 System continue to blink alternately but I can't transmit. Obviously something is happening, but what?

First, let's review RS-232 signal definitions...

From the chart you can see that Pin 4 is always high when the 4050 Series is transmitting. However, the 4050 System requires a Clear to Send signal from the modem. Perhaps the modem or RS-232 device to which you are connected simply doesn't provide the Clear to Send signal. To circum-

RS-232 Signal Definitions

Pin	Circuit	Signal Name	Definition	Direction
1	AA	Protective Ground	Electrically bonded to equipment frame. May be further connected to external grounds as required by applicable regulations.	Not applicable.
2	BA	Transmitted Data	Transmit data line to Terminal Modem from Terminal.	TO data communication equipment.
3	BB	Received Data	Receive data line from Terminal Modem to Terminal.	FROM data communication equipment.
4	CA	Request to Send	High to Terminal Modem when Terminal wants to transmit, telling the Terminal Modem to generate a carrier to Host Modem on which to transmit data from the Terminal.	TO data communication equipment
5	CB	Clear to Send	Response from Terminal Modem to Terminal's Request to Send. After Terminal Modem generates a carrier, it sets Clear to Send high to tell Terminal that it's okay to transmit.	FROM data communication equipment.
6	CC	Data Set Ready	Always high from Terminal Modem to Terminal when Terminal Modem is powered up and ready to operate.	FROM data communication equipment.
7	AB	Signal Ground or Common Return	Common return for all signal leads.	Not applicable.
8	CF	Received Line Signal Detector	High to terminal when Terminal Modem detects a usable carrier generated from distant Host Modem; Terminal can then receive data.	FROM data communication equipment.
11	SCA	Secondary Request to Send	Used with RS-232-A in Half-Duplex Supervisory mode only. ¹ High from Terminal to Terminal Modem in receive submode, telling Terminal Modem to generate a secondary carrier over which Terminal may transmit a break signal to Host.	TO data communication equipment.
12	SCF	Secondary Received Line Signal Detector	Used in Half-Duplex Supervisory mode only, to control line turn around. High from Terminal Modem to Terminal in transmit submode, telling Terminal that a secondary carrier is being generated by Host Modem.	FROM data communication equipment.
15	DB	Transmitter Signal Element Timing ³	Line to supply an external clock for Transmitted Data line from Terminal Modem to Terminal.	FROM data communication equipment.
17	DD	Receiver Signal Element Timing ³	Line to supply an external clock for Received Data line from Terminal Modem to Terminal.	FROM data communication equipment.
19	SCA	Secondary Request to Send ²	Used with RS-232-C in Half-Duplex Supervisory mode only. ¹ High from Terminal to Terminal Modem in receive submode, telling Terminal Modem to generate a secondary carrier over which Terminal may transmit a break signal to Host.	TO data communication equipment
20	CD	Data Terminal Ready	Always high from Terminal to Terminal Modem when Terminal is powered up and ready to operate.	TO data communication equipment.

¹ The 4050 Series supports both RS-232A and RS-232C.

² Some modems need to see this high all the time when in Half-Duplex Normal.

³ Supported by 4052/4054.

Notes for Option 1:

Terminal = 4050 Series
Low = -3 to -15 V
High = +3 to +15 V

vent this, an adapter, sometimes called a loopback connector, can be included between the 4050 Option 1 RS-232 interconnect cable⁴ and the modem or device. The following two diagrams illustrate such adapters you could build or buy depending on whether the 4050 is used as a terminal or host.

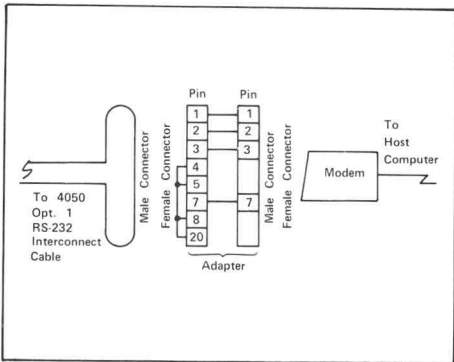


Fig. 1. 4050 Operating as a Terminal

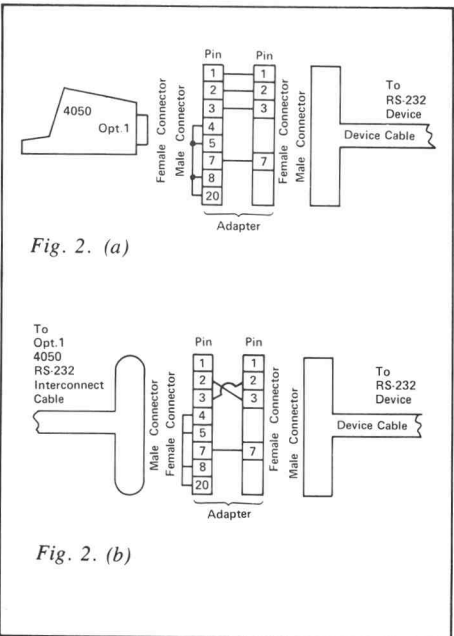


Fig. 2. (a)

Fig. 2. (b)

Fig. 2. 4050 Operating as a Host Computer
(a) Shows adapter plugged directly into 4050 Opt. 1 backpack.
(b) Uses the interconnect cable (for distance), thus the 4050 appears to be in a terminal (see footnote 4), so proper transmit and receive must be achieved by cross wiring pins 2 and 3 in the adapter.

Pins 4, 5, 8, and 20 are connected in the adapter. Since pin 4 is always high when the 4050 System is transmitting, this will provide positive (high) voltage to Pin 5. Pin 8 must also be high, which is provided by Pin 20, thus they may all be looped together.

Pin 6 doesn't matter—it could be "floating."

⁴ A special cable (Tektronix part no. 012-0689-00) provided with your Opt. 1. It reverses the wiring on the 4050 RS-232 connector so the 4050 appears to be a terminal to the modem and host computer.

All other pins not shown in the sketch could be grounded.

The pin numbers are imprinted on the male connectors of the Opt. 1 interconnect cable. Pin 1 is the Protective Ground. Refer to the note regarding this pin in the chart.

If the pin configurations on your RS-232 device are different than those described in the chart, refer to the device manual. Compare circuit definitions in the chart with those in the manual. This could help you attach the correct pins to your RS-232 device to make it work.

What's going wrong when garbled characters are received from the 4050 System?

In almost all cases, the baud rate on the 4050 Series doesn't match that of the RS-232 device receiving the characters. Both the 4050 System and the RS-232 device send and receive data at specified rates, known as the baud rate. If the baud rate on the 4050 System is different than the baud rate set on your RS-232 device, you'll get garbled characters. If this happens, take the following action:

Press User-Definable Key 14 to list the Data Communications parameters on the 4050 screen. Check the first parameter in the RATE routine to see if it matches the device's baud rate.

If not, press UDK 5 to return to BASIC mode.

Reset the baud rate on your 4050 System through the CALL "RATE" command (see the 4050 Option 1 Data Communications Manual for the parameters).

Re-enter **TERMINAL** mode and transmit. 

Graphic Systems Workshops Continue

TEKniques Vol. 4 No. 2 described the 4050 Series Graphic Computing Systems Workshops, which are designed to help you get the most out of your systems. These week-long workshops are held in four basic locations around the country. These workshops are continuing; new dates and locations are detailed here.

The workshops teach the application of BASIC and the concepts of desktop computer graphics. The classes emphasize integrating and using TEKTRONIX 4050 Series Graphic Computing Systems with their associated peripheral equipment. The full range of Graphic Computing System equipment is available in the workshops.

The workshops combine lectures and exercises to develop a working knowledge of system capabilities. Laboratory sessions complement and reinforce lecture information, through developing practical examples. When you complete a session, you'll have the skills necessary to apply the system capabilities to your own tasks.

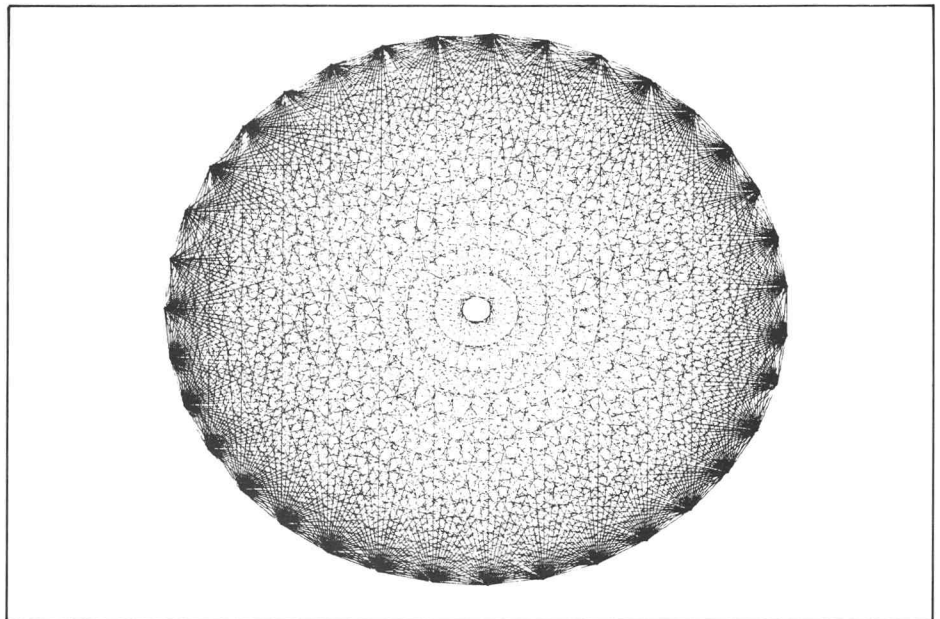
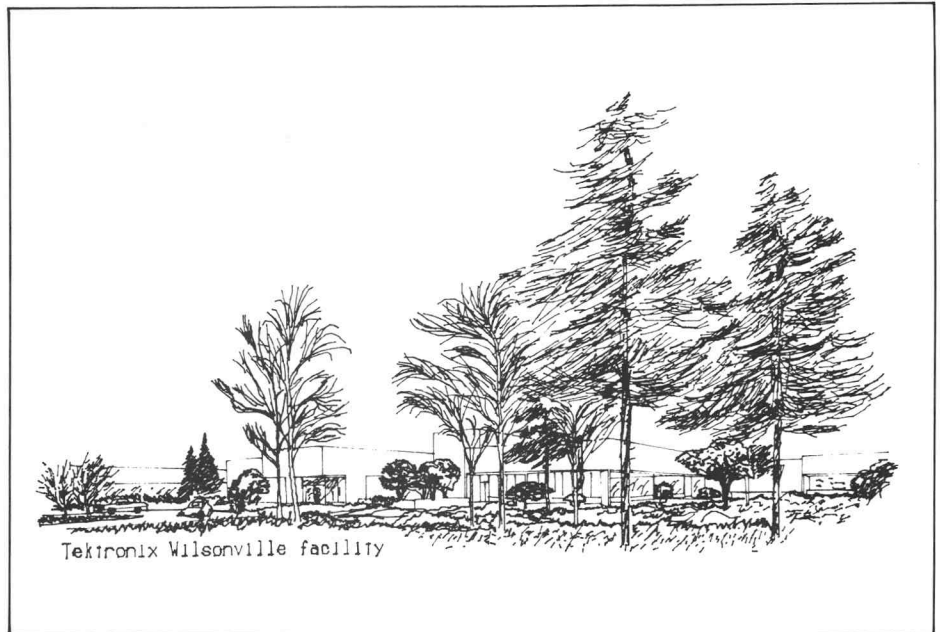
Locations and Schedules

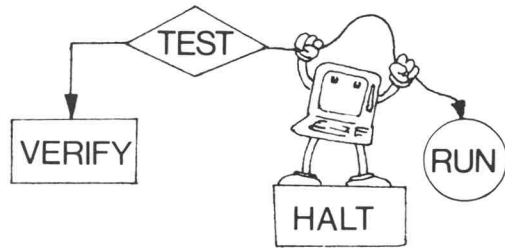
The workshops are scheduled through four basic locations, as shown on the schedule. But on-site classes and special locations can be arranged for groups of 10 or more. Currently, the following places and times are scheduled:

Santa Clara, CA	Oct. 6-10
Rockville, MD	Oct. 13-17
Santa Clara, CA	Oct. 20-24
Rockville, MD	Oct. 27-31

For Information

More information can be found in TEKniques, Vol. 4 No. 2. If you're interested in attending one of these sessions, or think you might be, you can get additional information by contacting your local Tektronix Sales Engineer. Or you can call Raynor Christianson at (503) 644-0161, ext. 8949. He can provide information on other workshops as well, such as the recently-established 4020 Series Terminal workshop.





PROGRAMMING TIPS

Home Insulation Break-Even Point

by Chris Avery
U.S. Dept. of Agriculture
Forest Service
Washington, D.C.

The following program calculates the number of years required to break even on home insulation. It takes into consideration the cost of insulation, fuel savings due to insulation, interest that could have been received on the money spent for insulation, etc.

The value of all variables for each year is displayed. You can determine if home insulation would be cost effective under various cost/savings assumptions. Generally, a break-even point greater than seven years is not considered cost effective.

```

100 INIT
110 PAGE
120 F#0
140 REM Enter Amount in Dollars, i.e., 450
150 PRINT "Enter current yearly cost of heating--summer & winter "
160 INPUT C
170 REM Percent of heating bill to be saved by insulation, i.e., 35
180 PRINT "Enter percent heating bill saved by insulation "
190 INPUT S
200 REM Convert all percentages to fractions
210 W=S/100
220 REM Cost is entered in dollars
230 PRINT "Enter cost of insulation "
240 INPUT I
250 Z=I
260 REM Percent cost of fuel will increase ea year, i.e., 20
270 PRINT "Enter percent of yearly increase of fuel cost "
280 INPUT P
290 W#I*W
300 REM Return on savings if money not spent on insulation
310 PRINT "Enter interest rate on savings "
320 INPUT R
330 W#P/100
340 REM Rate of increase of interest over time, i.e., .2
350 PRINT "Enter yearly increase of interest rates "
360 INPUT R
370 W#W*100
380 PRINT "Enter number of times interest is compounded yearly "
390 INPUT C
400 IMAGE C:"fuel cost",5x,"savings",4x,"interest",5x,"diff",8x,"total"
410 PAGE
420 PRINT #0: USING #00:
430 IMAGE "increase #",4x,"reduction #",13x,"on $",5d,2d

440 PRINT #0: USING #30:
450 IMAGE 2d,"% w/ no",5x,2d,"%",20x," ",2d,1d,"% w/"
460 PRINT #0: USING #08:PISIG
470 IMAGE "insulation",4x,"w/ insul.",13x,"inc. of ",1d,2d,"%"
490 IMAGE 72: ", "
510 REM #0: USING #90:
520 W#C#X
530 REM B=fuel cost with insulation installed
540 B=C-A
550 W#B
560 REM Following loop computes compound interest on savings
570 FOR I=1 TO C
580 COSUB 770
590 END I
600 REM e=net savings (gain or loss) due to insulation
610 E=A-D
620 REM P=accumulative savings due to insulation
630 P#E
640 IMAGE 6(6d,2d,3x)
650 PRINT #0: USING #40:CIBAI|D|E|F
660 REM C=next year's fuel cost w/o insulation
670 C=C#C#W
680 W#I#W#U
690 REM Test if savings are greater than initial cost
700 IF F<Z THEN 520
710 PRINT #0: USING #90:
720 IMAGE "interest is compounded ",2d," times per year"
730 PRINT #0: USING 720:G
740 PRINT #0: USING 750:
750 IMAGE 5("L")
760 REM W=interest on current savings amount
770 W#I#W#G
780 REM D=accumulative interest this year

790 D#W#H
800 T#T#H
810 REM Compute current total savings
820 RETURN

Enter current yearly cost of heating--summer & winter #00
Enter percent heating bill saved by insulation #0
Enter cost of insulation 2000
Enter percent of yearly increase of fuel cost #5
Enter interest rate on savings #0
Enter yearly increase of interest rates #0
Enter number of times interest is compounded yearly 4
  
```

fuel cost increase # 15% w/ no insulation	fuel cost reduction # 18%	savings w/ insul.	interest on \$ 2000.00 @ 8% w/ inc. of 0.00%	diff	total
880.00	720.00	80.00	164.00	-84.00	-84.00
920.00	820.00	82.00	178.45	-86.45	-171.32
1050.00	952.20	105.00	193.16	-87.36	-258.68
1216.70	1095.03	121.67	209.09	-87.42	-346.10
1399.21	1259.20	139.92	226.32	-86.40	-432.50
1609.09	1440.18	168.91	244.98	-84.87	-516.50
1858.45	1665.48	185.84	265.17	-80.13	-596.70
2149.82	1935.21	212.80	287.03	-74.23	-670.94
2484.30	2232.07	244.72	310.69	-65.97	-736.91
2864.28	2532.07	291.43	336.30	-54.67	-791.70
3292.45	2835.00	323.64	364.03	-40.38	-832.10
3771.91	3149.72	372.19	394.03	-21.84	-854.01
4298.20	3485.18	428.82	426.52	1.50	-852.50
4922.23	3838.01	492.22	461.67	30.55	-821.95
5650.90	4209.10	565.86	499.73	66.13	-755.63
6509.65	4609.68	650.96	540.92	110.04	-645.59
7489.10	5039.49	748.61	585.51	163.10	-482.49
8600.81	5509.11	860.98	633.70	227.12	-285.37
9860.36	6019.33	990.04	686.02	304.01	-48.64
11305.42	6569.00	1130.54	742.57	395.97	444.61
13003.23	7170.91	1309.32	803.79	505.54	950.15
15007.21	7851.49	1505.72	870.04	635.68	1505.92
17315.00	8584.22	1731.50	941.76	789.82	2375.64

Interest is compounded 4 times per year

Program Initialization

by Jon C. Mutton
Tektronix, Inc.
Wilsonville, OR

Many programs use an initialization routine to set constants and default parameters. Often it's deleted upon execution to avoid re-execution later. There are two serious drawbacks to this approach:

1. It only works when the entire program is present, i.e., when the program is initially loaded via an "OLD" or "Auto Load" command. Subsequently, initiating the system with an "INIT" command would require all the variables to be set manually or the program to be reloaded.

2. Saving the program with a "SAV" command without restoring the routine.

The following short routine solves these problems.

```

100 DIM R$(3)
110 R$=REP(" ",1,0)
120 IF LEN(R$)>0 THEN 500
130 REM BEGINNING OF INIT ROUTINE
140
150
160
170
180
190
200 REM END OF INITIALIZATION ROUTINE
210 R$=" "

500 REM PROGRAM STARTS HERE
  
```

The "DIM" command in line 100 is not necessary but does limit the amount of memory used by the routine. The REP function in line 110 defines string variable R\$ without changing it. The length of R\$ is checked in line 120 and if R\$ is 1 character or more long the main program is executed. If LEN(R\$)=0 the initialization routines are executed before the main program. A string variable contained in the main program may be used so long as it will always contain 1 or more characters. This variable may be dimensioned if necessary in line 100.

Packing Integers for Memory Saving/Efficiency

by Ted Webber

Laurie, Montgomerie and Pettit
Sydney, Australia

This program uses a base 100 counter and stores integers as strings. It compares with an article in TEKniques Vol. 1 No. 10,¹ where a base 128 counter is used.

The program is also a follow-on of one published in TEKniques Vol. 3, No. 2,²

which stored two six-digit integers in one real number. While that routine is considerably faster for larger data values, this program becomes more efficient for smaller numbers.

As you can see, approximately 4,000 bytes of memory are saved using the string method of storage.

```

1 REM ENCODE AND DECODE AN INTEGER VECTOR AS A STRING
2 REM TO BASE 100 FOR MAX 6 DIGITS
100 INIT
110 REM DIM A$(2+DIGITS),B$(1+DIGITS/2)
120 D=32
130 D1=1000
140 DIM A$(8),B$(4),C$(3),N(D1)
150 PRINT @D:"FOR 32-K RAM AND DIM N(";D1;) MEM= ";MEMORY;"J"
160 REM GENERATE A VECTOR
170 N=0
180 Z=1000000
190 R=RND(-1)
200 REM ILLUSTRATE EFFECT OF SIGN
210 FOR I=2 TO D1
220     Z=-Z
230     N(I)=INT(RND(1)*Z)
240 NEXT I
250 PRINT @D:N
260 REM ENCODE: DIM D$(N*DIM B$)
270 DIM D$(D1*4)
280 D$=""
290 FOR J=1 TO D1
300     A$=STR(ABS(N(J)))
310     REM 1ST CHAR OF A$ IS SPACE, 2ND CHAR MAY BE NEG SIGN
320     L=LEN(A$)
330     IF L/2=INT(L/2) THEN 360
340     REM ODD NUMBER OF DIGITS
350     A$=SEG(A$,2,L)
360     L=LEN(A$)/2
370     B$=""
380     IF SGN(N(J))=>0 THEN 400
390     B$="-"
400     FOR I=1 TO L
410         C$=SEG(A$,2*I-1,2)
420         N1=VAL(C$)
430         REM CHR(0) UNSUITABLE, SO INC N1 FOR ASCII CHRS 1-100
440         C$=CHR(N1+1)
450         B$=B$&C$
460     NEXT I
470     IF L=3 THEN 520
480     REM FILL TRAILING NULLS WITH DUMMY
490     FOR I=1 TO 3-L
500         B$=B$&"z"
510     NEXT I
520     D$=REP(B$,J*4-3,0)
530 NEXT J
540 REM *** WARNING *** D$ HAS MANY CONTROL CHRS
550 PRINT @D:D$
560 DELETE N
570 PRINT @D:"FOR 32K RAM & STRING CODING, MEM= ";MEMORY;"J"
580 REM DECODE
590 DIM N(D1)
600 FOR J=1 TO D1
610     A$=""
620     B$=SEG(D$,J*4-3,4)
630     FOR I=2 TO LEN(B$)
640         C$=SEG(B$,I,1)
650         IF C$="z" THEN 720
660         N1=ASC(C$)-1
670         C$=STR(N1)
680         IF N1>9 THEN 700
690         C$=REP("0",2,0)
700         C$=SEG(C$,2,2)
710         A$=A$&C$
720     NEXT I
730     B$=SEG(B$,1,1)
740     A$=B$&A$
750     N(J)=VAL(A$)
760 NEXT J
770 PRINT @D:N
780 END

```

FOR 32-K RAM AND DIM N(1000) MEM= 20629

0	-252901	596147	-814750
493581	-416834	458856	-142034
907604	-430393	269945	-150517
691099	-270820	591752	-610182
238893	-204131	141654	-246865
434129	-92608	169368	-607186
742116	-21104	639556	-549386
318785	-600703	815120	-252624
936187	-886799	413928	-210561
138855	-969763	490179	-894520
90622	-892717	184493	-453694
883523	-172721	794873	-218540
678429	-776535	232198	-580170
137942	-707162	514708	-162067
590419	-115344	889214	-647247
811832	-121166	493406	-177948
999025	-912606	931820	-347510
390795	-715398	153628	-334115
216702	-374597	706608	-27
549595	-962964	599367	-551710
726031	-502268	121244	-817138
986156	-636130	222116	-975711
765705	-549616	386785	-739735
729579	-351734	97069	-35500
438551	-819812	809345	-137776
945004	-444707	191836	-218197
264835	-285481	594671	-551699
402678	-751828	522665	-809994
436081	-329694	949993	-404296
298111	-35466	896833	-220144

FOR 32K RAM & STRING CODING, MEM= 24645

0	-252901	596147	-814750
493581	-416834	458856	-142034
907604	-430393	269945	-150517
691099	-270820	591752	-610182
238893	-204131	141654	-246865
434129	-92608	169368	-607186
742116	-21104	639556	-549386
318785	-600703	815120	-252624
936187	-886799	413928	-210561
138855	-969763	490179	-894520
90622	-2717	184493	-453694
883523	-172721	794873	-218540
678429	-776535	232198	-580170
137942	-707162	514708	-162067
590419	-115344	889214	-647247
811832	-121166	493406	-177948
999025	-912606	931820	-347510
390795	-715398	153628	-334115
216702	-374597	706608	-27
549595	-962964	599367	-551710

¹Page 23 in Programming Tip Handbook.

²Page 72 in Programming Tip Handbook.

Editor's Note: Mr. Webber contributed these routines to TEKniques in the hope of generating a discussion into similar techniques. A subsequent tip from Phil Somerset touches on the same subject and is included in this issue. Related routines are contained in TEKniques Vol. 1 No. 9 and Vol. 2 No. 1 (pages 19 and 32 respectively, in the Programming Tip Handbook).

Interfacing MIPS and MARs: General Procedure for Downloading Data to MARs

by Jim Dillon
Tektronix, Inc.
Santa Clara, CA

This program allows you to download a MIPS Data File to a Modeling and Reporting software (MARs) Data File automatically.

It also demonstrates the correct way to construct a MARs Data File outside of MARs. Thus, you can download data from any source, i.e., from a data base on a mainframe. Just follow a few simple procedures in constructing your files:

- A. First construct the MARs model that will be the basis for subsequent reports and graphs. The size of the maximum matrix in this model determines the size of the subsequent data matrix. Also determine the row and column names. This will be discussed further under (C).
- B. Construct a dummy data file referencing the model constructed in (A). This file will be the file that later is used for the downloading. There are two reasons for constructing this file beforehand. They are:
 1. MARs automatically will put the DATA file name in the directory;

2. The model referenced determines the row and column name strings and the size of the matrix. They **must** be equivalent.

```
10 DIM C$(113)
40 DIM T$(72),D$(14,13),D$(113),F$(14)
46 T$=" "
47 D$=" "
50 D9=0
55 PRINT "LOAD MIPS DATA TAPE INTO INTERNAL TAPE DRIVE***"
60 PRINT "WHAT MIPS FILE DO YOU WANT TRANSFERED?:"
70 INPUT N
80 FIND N
81 READ Q33:T$,D$,F$
82 C$=C$+R$ R2 R3 R4 R5 R6 R7 R14 " "
93 C$=C$+R$ R9 R3 R10 R11 R12 R13 R14 " "
92 D$=C1 C2 C3 C4 C11 C12 C6 C7 " "
93 D$=C$+C$ C9 C10 C11 C12 C13 " "
100 PRINT "REMOVE MIPS DATA TAPE AND LOAD MARs DATA TAPE***"
101 PRINT "WHEN READY PRESS RETURN*"
110 INPUT M
120 PRINT "WHICH MARs DATA FILE DO YOU WANT TO REPLACE?"
130 INPUT N
140 FIND N
141 D$="FINANCIAL MODELING AND REPORTING SYSTEM"
142 C1=103
143 PRINT "WHAT NAME DO YOU WANT TO GIVE THIS FILE?:"
144 INPUT L$
150 WRITE Q33:C$,M,L$
190 WRITE Q33:C$
200 WRITE Q33:D$
210 WRITE Q33:D9
215 PRINT "MARs DATA FILE IS SAVED."
220 END
```

- C. Run the enclosed program. The last step of the program writes 7 data items onto the designated MARs Data File. They are:

Q\$ = FINANCIAL MODELING
AND REPORTING SYSTEM

C1 = 103

M = The number of the MARs Data File

L\$ = Name of the MARs data file used in
(B)

C\$ = A string containing the row names used in the MARs model. Each name is 8 characters long, spaced with blanks to complete the string. The string is dimensioned 8* (number of rows) +1 (see listing). In the MIPS example it is dimensioned to 113.

D\$ = A string containing the column names structured similarly to D\$. It is dimensioned 8* (number of columns) +1.

D9 = The data matrix containing the numeric values to be used by MARs. In our MIPS example, it is dimensioned 14 rows by 13 columns.

The above elements comprise a total MARs Data File and can be used as if it had been constructed by MARs itself.

Editor's Note: MIPS is a popular program in the 4050 Series Applications Library (Abstracts # 51/00-0716/0 and # 51/07-0717/0). Modeling and Reporting software is a Tektronix Product (#4050B01) which was introduced approximately two years ago—see TEKniques Vol. 2 No. 8 or Business Graphing and Reporting reprint # AX-4451.

Compact Integer Storage

by Phil Somerset
Tektronix, Inc.
Rockville, MD

Suppose you have a large number of integer data values to store in a 4050 Series Graphic Computing System. When stored in a numeric array, each data value requires eight bytes of 4050 memory. The method shown below can save the same amount of data in one-fourth of the space if the data values fall within the range $0 \leq x \leq 9999$.

Examine the listing. Lines 100-280 encode the data and lines 290-380 decode the data. Each number is broken into two two-character segments, each of which is then turned into an ASCII character in the range 27-126 (lines 190-210). These characters are then concatenated into string A\$. Thus, a character string 1000 bytes long can accommodate 500 positive integer values.

The program is an interactive program, but the same method can be built into any

program to maximize storage capacity. The method could be effectively used for storing graphic coordinates from the 4956 Graphic Tablet which fall in the range 0 to 4000. This structure lends itself quite well to editing, too, since the string functions can be used. For example, the Nth number could be deleted by issuing the statement: A\$ = REP (" ", 2N-1, 2). Another advantage is that character string I/O is faster than numeric I/O.

If the requirement exists for storing large numbers of positive integers less than 10000, this method may save you a lot of time and trouble.

```
1 INIT
2 GO TO 110
4 GO TO 150
8 GO TO 300
24 GO TO 260
100 REM - INITIALIZE DATA STORAGE AREAS
110 DIM A$(1000)
120 I=0
130 A$=""
140 REM PUT NUMBERS INTO STORAGE STRING
150 PRINT "? "
160 INPUT N
170 IF N<0 THEN 260
180 I=I+1
190 R=N-INT(N/100)*100+27
200 N=(N-R)/100+27
210 B$=CHR(N)
220 A$=A$B$
230 B$=CHR(R)
240 A$=A$B$
250 GO TO 150
260 PRINT "A$ IS "LEN(A$)" BYTES LONG AND I"
270 PRINT "CONTAINS "I" NUMBERS"
280 END
290 REM - GET NUMBERS OUT OF STORAGE STRING
300 PRINT "NUMBER? "
310 INPUT N
320 IF N<1/2*LEN(A$) THEN 300
330 IF N<1 THEN 400
340 B$=SEG(A$,2*N-1,2)
350 C$=SEG(B$,1,1)
360 B$=REP(" ",1,1)
370 N=(ASC(C$)-27)*100+ASC(B$)-27
380 PRINT N
390 GO TO 300
400 END
```

Underlining Text

by Dan Taylor
Tektronix, Inc.
Wilsonville, OR

If you use control-H sequences followed by shift-rubout to underline your text, e.g.,

```
Example of underline#####
```

try the following routine.

```
LIS
100 DATA 1.792,0.17,0.5,0.4
110 RESTORE
120 READ C1,C2,C3,C4
130 PRINT "List a Text file, with control-H sequences converted to ";
140 PRINT "down underlines"
150 PRINT "I\What tape file is your text on? ";
160 INPUT J
170 FIND J
180 PAGE
190 ON EOF (0) THEN 460
200 DIM A$(255)
210 INPUT @33:A$
220 IF POS(A$,"H",1)>0 THEN 250
230 PRINT A$
240 GO TO 210
250 FOR J=1 TO LEN(A$)
260 B$=SEG(A$,J,1)
270 IF B$(">"H" THEN 420
280 REM -----
290 FOR J1=J+1 TO LEN(A$)
300 C$=SEG(A$,J1,1)
310 IF C$(">"H" THEN 340
320 NEXT J1
330 STOP
340 J2=J1-J
350 J=J1+J2-1
360 INPUT @32,24:X,Y
370 PRINT @32,21:X-J2*C1-C4,Y-C2
380 PRINT @32,20:X-C3,Y-C2
390 PRINT @32,21:X,Y
400 GO TO 430
410 REM -----
420 PRINT B$:
430 NEXT J
440 PRINT
450 GO TO 210
460 END
```

It will locate your text to be underlined, i.e., your control-H sequences, and replace it with a continuous underline. Also, note the "fudge" factor used in statements 370 and 380 which drops the line a bit, begins it a little to the left and ends it a little to the right of the text to produce a more eye-appealing underline.

```
A Test of show an Example_of_underline
A test to show an Example of underline
```

The first example uses the control-H|shift-rubout sequence; the second the above routine.

Appending

by Phillip Sivyer
Consolidated Fertilisers Limited
Brisbane, Queensland, Australia

If a program is to be APPENDED, and the beginning line number of the incoming program is the same as the target line number (and no increment is specified), the incoming program lines are not renumbered. In other words, the default increment of 10 does not operate in this case.

This is especially useful when lines 1 to 100 are involved, as they must stay at their existing line numbers to enable the User-Definable Keys to operate correctly. Other applications include the loading of programs containing syntax or tape errors. In both cases, after clearing the error, the rest of the program lines can be APPENDED to their original line numbers.

Editor's Note: The recovery for syntax and tape errors was included in the Input/Output column in TEKniques Vol. 4 No. 5.

```
LIS
4 INIT
5 SET KEY
6 GO TO 100
100 REM
130 REM
140 REM
200 REM THIS IS A TEST
400 GO TO 411
411 REM
500 REM

FIN17
SAV
DELETE ALL
```

Fig. 1. Original program saved on tape, then deleted from memory.

```
4 REM
FIND 17
APPEND 4
LIS
4 INIT
5 SET KEY
6 GO TO 100
100 REM
130 REM
140 REM
200 REM THIS IS A TEST
400 GO TO 411
411 REM
500 REM
```

Fig. 3. When the REM line number is the same as the incoming program and no increment is specified, the original line numbers are intact.

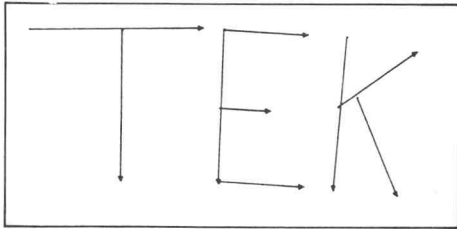
```
1 REM
FIN 17
APPEND 1
LIS
1 INIT
11 SET KEY
21 GO TO 31
31 REM
41 REM
51 REM
61 REM THIS IS A TEST
71 GO TO 81
81 REM
91 REM

DELETE ALL
```

Fig. 2. The program in Fig. 1 appended. Notice when the REM line number is different than the incoming line number, the 4050 automatically increments the statements by 10.

Omni-Directional Arrows

by Roger Chan
U.S.V. Pharmaceutical Corp.
Yonkers, NY

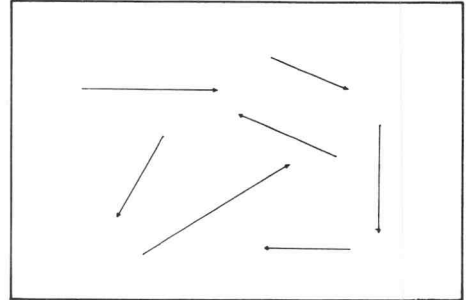


This routine draws any number of arrows on the 4050 screen. You use the 4662 Plotter joystick to position and orient the arrows. If any arrow slope is very close to vertical or horizontal, it will be changed to true vertical or true horizontal. The origin is considered the tail of the arrow.

Move the Plotter joystick until the arrow tail is positioned, then press User-Definable Key (UDK) #1. Next, position the arrow head and press UDK #1. The arrow will be drawn.

Continue until all arrows requested (line 170) have been positioned and drawn.

This routine could be incorporated as a subroutine in a graphic program.



Editor's Note: In place of statements 170 and 180, you could have an infinite loop with a UDK to terminate the loop.

```

1 INIT
2 SET KEY
3 GO TO 100
4 GO TO 120
5 REM ** KEY NO. 1 IS TO ENTER DATA*
90 REM ** THIS IS A SUBROUTINE TO DRAW ARROWS ON SCREEN IN ANY**
92 REM ** DIRECTION *****
100 PAGE
102 SET DEGREES
110 GO TO 160
120 GIN V,W
130 RDRAW 0,0
140 PRINT "GGG";
150 GO TO 310
160 HOME
170 PRINT "NO. OF ARROWS: ";
180 INPUT N1
185 PAGE
190 M3=0
200 B1=1
210 B3=0
220 DELETE G,L$
230 DIM G(1,10)
240 FOR I=1 TO N1
250 T=0
260 GIN @1:V,W
270 MOVE V,W
280 PRINT @32:18:5
290 PRINT @32:24:**
300 GO TO 260
310 PRINT @32:18:0
315 REM** CHECK FLAG TO SEE IF INPUT IS THE TAIL OF ARROW(1ST ENTRY)
320 IF T=1 THEN 370
330 G(I,1)=V
340 G(I,2)=W
350 T=1
360 GO TO 260
365 REM** FOUND DELTA X AND DELTA Y
370 M1=W-G(I,1)
380 M2=W-G(I,2)
385 REM** CHECK TO SEE IF IT IS A SAME POINT
390 IF M1=0 AND M2=0 THEN 690
395 REM** IF SLOPE IS CLOSE TO VERTICAL LINE OF HORIZONTAL LINE
396 REM** MAKE THEM INTO EITHER TRUE VERT. OR HORIZ. WITH ONE
397 REM** OF THE DIFFERENCE SET TO ZERO
400 IF -1<M1 AND M1<1 THEN 430
410 IF -1<M2 AND M2<1 THEN 450
420 GO TO 490
430 M1=0
440 GO TO 460
450 M2=0
460 B2=0
470 B1=1
480 GO TO 530
485 REM ** FOUND SLOPE AND THE PERCENTAGE DIFFERENCE IN TERM OF
486 REM ** THE LINE LENGTH
490 M3=M2/M1
500 B3=1/SQR(M1^2+M2^2)
510 B1=M1*B3
520 B2=M2*B3
530 MOVE G(I,1),G(I,2)
540 IF M3<0 THEN 630
550 IF M2=0 THEN 630
560 IF M1=0 AND W<G(I,2) THEN 590
570 IF G(I,1)>V OR G(I,2)>W THEN 610
580 IF M1=0 AND W>G(I,2) THEN 610
590 DRAW V-B1*(B2<0),W-B2+1*(B2=0)+B2*(M1=0 AND V<W)
600 GO TO 720
610 DRAW V-B1*(B2<0),W-B2-1*(B2=0)+B2*(M1=0 AND V<W)
620 GO TO 720
630 IF G(I,1)<V AND G(I,2)>W THEN 670
640 IF M2=0 AND V>G(I,1) THEN 670
650 DRAW V+B1*(B2=0)-B1*(B2<0),W+B2*(B2=0)-B2*(B2<0)
660 GO TO 720
670 DRAW V-B1,W-B2
680 GO TO 720
690 PRINT "NO ARROW"
700 I=I+1
710 NEXT I
720 REM ** ARROW HEAD SUBROUTINE
730 GIN G(I,3),G(I,4)
735 REM ** DETERMINE THE SLOPE AND ROTATE ARROW HEAD ACCORDINGLY
740 IF M1=0 THEN 790
750 IF M2=0 THEN 840
760 IF M1<0 AND M2>0 OR (M1<0 AND M2<0) THEN 870
770 A=ATN(M3)
780 GO TO 900
790 IF M2<0 THEN 820
800 A=90
810 GO TO 900
820 A=270
830 GO TO 900
840 IF M1<0 THEN 870
850 A=0
860 GO TO 900
870 A=180
880 GO TO 900
890 A=ATN(M3)+180
900 ROTATE A
910 RDRAW 0,0.5
920 GIN G(I,5),G(I,6)
930 RDRAW 1,-0.5
940 GIN G(I,7),G(I,8)
950 RDRAW -1,-0.5
960 GIN G(I,9),G(I,10)
970 RDRAW 0,0.5
980 A=0
990 NEXT I

```

Determining Data File Type on the 4907

by Kevin Talbot
Mannesmann-Tally
Kent, WA

Sometimes after writing a program that accessed various data files on our 4907, we found that a data file it used had been created with "PRINT" statements (ASCII file), but the program contained "READ" statements, thus wanted the data in binary format. Rather than rewriting all of the programs and data files so they were compatible, the following routine was added to most of our programs so either a READ (binary) or INPUT (ASCII) could be used to access the data from a disk file.

The routine uses the CALL "FILE" utility in the 4907 ROM pack. The CALL 'FILE' in statement 350 gets the current file status message of the file whose name is in A\$. Although the complete status message is 189 characters long plus the file identifier, we only need the first 13 characters since the "attributes" of the file (ASCII, public, etc.) are contained within these first characters. Thus, the default length of 72 characters of X\$ need not concern us.

If the file does not exist (due to a spelling error, or perhaps the wrong disk being in the 4907), the length of X\$ will be 0 and indicates an error condition; T is set to -1 (statements 360, 470-490).

Once the file status message is in X\$, we check it one character at a time to see if there is an "A" indicating the data file is in ASCII format (statements 370-390). If none of the characters is an "A", then the file is assumed to be binary and the variable T is set accordingly to +1 (statements 410-430). If one of the characters is an "A", then a jump out of the loop is done and T is set to 0 (statements 390-440-460).

After calling this subroutine, our programs then use the value of T to determine if a jump to a READ or WRITE statement (binary, T=1) is in order, or a jump to an INPUT or PRINT statement (ASCII, T=0) is in order. The error condition is trapped T=-1) and can be used to prompt the user to check spelling, etc. For our convenience, the routine returns with the file open for "full" access (read or write) in line 500.

A similar program can be written for the internal tape drive of the 4050. By using the TYP(0) statement, a jump to the appropriate statement (READ/INPUT or WRITE/PRINT) can be performed based on the value of TYP (0).

```
105 REM      This subroutine checks to see the type of a file
110 REM      (binary or ASCII) and open the file for "full" access
120 REM      (either read or write).
130 REM
140 REM
150 REM
160 REM      * If the file is BINARY ----- "T" is set to 1 *
170 REM      * If the file is ASCII ----- "T" is set to 0 *
180 REM      * If the file doesn't exist ----- "T" is set to -1 *
190 REM      * The program returns with the "ifn" set to 1 (logical
200 REM      * file number).
210 REM
220 REM
230 REM
240 REM      VARIABLES USED:   FILE NAME -----> AS
250 REM
260 REM      DEVICE NUMBER -----> X9
270 REM
280 REM      COUNTER INDEX -----> I
290 REM
300 REM      DUMMY STRING VARIABLES=> X$,Y$
310 REM
320 DELETE Y$
330 DIM Y$(1)
340 CALL "MOUNT",X9,X$
350 CALL "FILE",X9,AS,Y$
360 IF LEN(X$)=0 THEN X$="0"
370 FOR I=1 TO 13
380   Y$(I)=MID(X$,I,1)
390   IF Y$(I)="A" THEN GOTO 420
400 NEXT I
410 REM: none of the characters was an "A" so file is binary
420 T=1
430 GO TO 500
440 REM: one of the characters was an "A" so file is ASCII
450 T=0
460 GO TO 500
470 PRINT " ERROR!---- FILE DOES NOT EXIST "
480 T=-1
490 GO TO 510
500 OPEN AS:,"F",X$
510 RETURN
```

Editor's Note: The TYP command will also work on the disk, however, it would be TYP (1) through TYP(9) depending on the logical file number assigned.

Note that although statement 390 will cause the program to jump out of a loop, if the file is ASCII, the memory allocated to keep track of the loop is recovered when the RETURN command in statement 510 is executed.

Clearing the Stack

by Dan Taylor
Tektronix, Inc.
Wilsonville, OR

Several programming tips have been written¹ on how to exit a FOR/NEXT loop using a dummy loop, e.g.,

```
200 FOR dummy = 1 to 1
    FOR I = . . .
        GO TO 400
    NEXT I
400 NEXT dummy
```

This dummy FOR/NEXT loop has a much wider application. It can be used to "flush" part or all of the GOSUB/FOR stack²:

1. To flush the entire stack, execute
FOR dummy = 1 to 1
when the program starts.
2. Whenever you want the stack cut back, execute
NEXT dummy

This will get rid of all GOSUB/FOR on the stack since the most recent FOR dummy.

3. To re-arm it, do another FOR dummy = 1 to 1.
4. To flush part of the stack, use other dummy variables in your program and execute FOR dummy. . . when you want to establish a cut-back point.

The following examples illustrate what happens to the memory allocated for the stack.

```
100 INIT
110 PRINT "Program start, memory = "MEMORY
120 FOR I=1 TO 1
130 PRINT "Dummy loop set, memory = "MEMORY
140 GOSUB 200
200 PRINT "Gosub 200 here! memory = "MEMORY
210 GOSUB 300
300 PRINT "Gosub 300 here! memory = "MEMORY
310 NEXT I
320 PRINT "Complete flush! memory = "MEMORY
330 PRINT "A return will return to keyboard"
340 RETURN
RUN
Program start, memory = 30354
Dummy loop set, memory = 30328
Gosub 200 here! memory = 30322
Gosub 300 here! memory = 30316
Complete flush! memory = 30354
A return will return to keyboard
```

Fig. 1. Complete stack flush.

```
100 INIT
110 PRINT "Program start, memory = "MEMORY
120 GOSUB 200
130 PRINT "Program returned to line 130! memory = "MEMORY
140 STOP
200 PRINT "Gosub 200 here! memory = "MEMORY
210 FOR I=1 TO 1
220 PRINT "First dummy loop set! memory = "MEMORY
230 GOSUB 300
240 PRINT "Program returned to line 240! memory = "MEMORY
250 END
300 PRINT "Gosub 300 here! memory = "MEMORY
310 FOR J=1 TO 1
320 PRINT "Second dummy loop set! memory = "MEMORY
330 FOR K=1 TO 10
340 PRINT MEMORY
350 IF K=3 THEN 400
360 NEXT K
400 PRINT "Jumped out of loop! memory = "MEMORY
410 GOSUB 500
500 PRINT "Gosub 500 here! memory = "MEMORY
510 NEXT J
520 PRINT "Partial flush! memory = "MEMORY
530 RETURN
```

Fig. 2.

```
RUN
Program start, memory = 29993
Gosub 200 here! memory = 29993
First dummy loop set! memory = 29967
Gosub 300 here! memory = 29961
Second dummy loop set! memory = 29935
29915
29915
Jumped out of loop! memory = 29909
Gosub 500 here! memory = 29903
Partial flush! memory = 29961
Program returned to line 240! memory = 29967
```

Fig. 3. Partial flush. The first GOSUB (statement 120) is still pending, i.e., another RETURN would have returned the program to line 130. However, the unfinished loop (statement 330) and GOSUB in statement 410 were flushed from the stack when statement 510 was executed.

¹TEKniques Vol. 1 No. 9, Vol. 2 No. 2, Vol. 2 No. 4 and Vol. 3 No. 4 (pages 21, 37, 45, and 83 in the Programming Tip Handbook).

²The push-down/pop-up stack located in RAM used to track program flow.

Disk Transfers of Arrays

by Arthur Ungar
Planning Systems
Lafayette, CA

You can greatly speed up information transfer to and from the disk by array reads and writes, rather than transferring individual data items. However, sometimes the actual array size is not known at the start of the program. The following example shows how arrays can be redimensioned for efficient disk transfer.

Editor's Note: For related Programming Tips see TEKniques Vol. 3 No. 3 (Page 79 in Programming Tip handbook) "Dynamic Memory Management—Arrays" and Vol 4, No. 1, "Paging Using the 4907."

```
100 REM INPUT A GRAPHIC FIGURE FROM THE TABLET, DISPLAY IT,
110 REM WRITE IT TO DISK, READ IT BACK AND ADD TO IT.
120 INIT
130 REM CURSOR BUTTON: Z - DRAW, 1 - MOVE, 3 - STOP
140 REM TABLE UNITS IN THOUSANDTHS
150 WINDOW 0,20000,0,20000/1.2
160 REM DIMENSION ARRAY OF POINTS TO THE MAXIMUM SIZE
170 DIM Z(1000,3)
180 REM READ THE POINTS BACK AND DISPLAY
190 PAGE
200 CALL "FILE",0,"TIP",Z$
210 IF LEN(Z$)>0 THEN 250
220 I=0
230 CREATE "TIP",4000,0
240 GO TO 270
250 OPEN "TIP";1,"R",Z$
260 READ #1:I
270 DIM Z(1,3)
280 READ #1:Z
290 CLOSE 1
300 DIM Z(1000,3)
310 FOR K=1 TO I
320 IF Z(K,3)>0 THEN 350
330 MOVE Z(K,1),Z(K,2)
340 GO TO 360
350 DRAW Z(K,1),Z(K,2)
360 NEXT K
370 REM INPUT POINTS
380 PRINT "Q";
390 INPUT "X,Y,Z";Z$
400 IF Z$="0" THEN 390
410 INPUT "A,B,A,A";A$
420 IF A$<>"0" THEN 410
430 IF Z$="2" THEN 550
440 IF Z$="1" AND I>0 THEN 490
450 MOVE X,Y
460 I=I+1
470 Z(I,3)=0
480 GO TO 520
490 DRAW X,Y
500 I=I+1
510 Z(I,3)=1
520 Z(I,1)=X
530 Z(I,2)=Y
540 GO TO 380
550 REM WRITE THE FIGURE TO DISK
560 DIM Z(I,3)
570 OPEN "TIP";1,"W",Z$
580 WRITE #1:I,Z
590 CLOSE 1
600 GO TO 170
```

Modified Auto-Paging

by Raymond DeMers
Tektronix, Inc.
Rochester, NY

In programs where the operator has to answer many questions, the AUTO-PAGE feature (PRINT @ 32,26:2) may be used to clear the screen on a "PAGE FULL".

However, repetition of a question because of unacceptable operator input may cause the AUTO-PAGE to occur in the middle of a

multi-line print sequence with only the last part of the print statement appearing at the top of the new page. This may be avoided by a GOSUB to a routine which checks how far it is to the bottom of the page after each question.

By inserting a GOSUB before each sequence of print statements, a new page is started if there is not room to complete another sequence) The sample value of 20 in line 1010 may be changed to suit the program.

```
999 REM PAGING SUBROUTINE
1000 INPUT @32,24:X,Y
1010 IF Y>20 THEN 1030
1020 PAGE
1030 RETURN
```

Butterfly Sort Extended

by Dan Taylor
Tektronix, Inc.
Wilsonville, OR

The November 1979 issue of TEKniques (Vol. 3 No. 7, or page 94 in the Programming Tip Handbook), contained the Butterfly sort algorithm for strings. It used one 4050 string variable (A\$) to emulate a linear array of strings. The sorting was done based on a string comparison in line 1620 (IF B\$ ≥ D\$ THEN 1650).


If this simple comparison is not adequate, e.g., each string in the linear array may contain a number of subfields (such as last name, first name, social security number,

and so on), and you want the strings sorted based on one (or more) subfield, just do the following.

Expand line 1620 to a number of lines of code to determine, in some fashion, if B\$ "≥" D\$ (for whatever rule of "≥" you want). For example, if your string contained name and social security number with positions 1 to 20 allowed for the name and 21 to 31 for the social security number, the following would sort in ascending order of the social security number:

```
1620 Q$=SEG(B$,21,11)
1622 R$=SEG(D$,21,11)
1624 IF Q$>R$ THEN 1650
```


You could also use more scratch variables and the string functions, SEG, POS, LEN, etc., to segregate more subfields to compare.

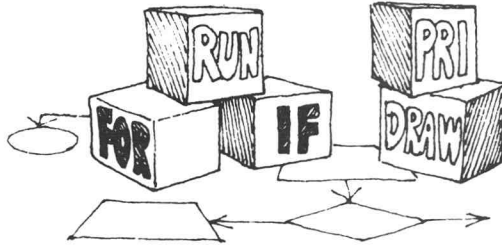
When you are adding code to this routine, be sure to change statement 1130 to allow for memory used for the extra code. If you don't, it will cause memory full to occur before your sort is complete. 

Please Correct Your Catalog

The 1980 4050 Series Applications Library Program Catalog recently distributed contained a few errors in part numbers. Please correct your catalog as follows:

Page	Item	Correct Part #
99	4050-4052-4054 Standard Accessories	
	System Software Tape	020-0160-03
	System Software Back-up Tape	020-0161-03
	4051-4052-4054 Optional Accessories	
	Tape Head Alignment Module Manual	070-3385-00
101	4050A10 Statistics Vol. 4	
	Overlay (nonlinear programs)	334-3035-01
	4050B01 Opt. 5 Modeling and Reporting (disc)	
	Reference Manual	070-2673-01
	Overlay	334-3377-00
	4050D01 Statistics-Test and Distributions	
	Reference Manual	070-3431-00
	Autoload Tape	delete
	System Disc	delete
103	4642 Optional Accessories	
	Ribbon	002-1451-01
	4662 Standard Accessories	
	RS-232 15 ft. I/F cable	012-0690-01
	4662 Optional Accessories	
	4662 Test Tape Manual	070-2366-00
104	4663 Options	
	Opt. 31	020-0355-01
	Opt. 32	020-0388-02
	4663 Optional Pens	
	Wet Ink—Pen Replacement Part Kit	006-2968-01

Please change the description on page 107 under 4924 Optional Accessories, GPIBI/F Cable, 3 meter, to read: GPIBI/F Cable, 2 meter (6.5'). 



BASIC BITS

Transferring UDK Overlay Legends

by Bill Detweiler
West Virginia University
Morgantown, WV

Persons who have ordered programs through the Applications Library are probably aware of the User-Definable Key (UDK) overlay figure included in some of the program documentation. Often, the user reproduces this UDK directory on a plastic overlay blank (Tektronix Part No. 334-2630-00), either by handprinting or through the use of transfer lettering. An alternative

method is to copy the figure to a Xerox transparency. The transparency is then mounted onto the blank. This technique is generally neater than hand lettering and faster than the transfer method.

Instructions are as follows:

1. Copy the overlay figure found in the program documentation to a Xerox transparency.
2. Mount the transparency on the plastic overlay blank using a spray adhesive such as Scotch Photo Mount.

3. Place the overlay face down and trim the transparency with a hobby knife, so that it conforms to the shape of the blank.

There are two drawbacks to this method. These are, the small degree of distortion inherent in the copy process, and the fact that some overlay figures are poorly typed.

The use of Xerox transparencies is, however, an attractive alternative to the other methods commonly employed.

Graphic Programming: Refresh on the 4051/4052

by Patricia Kelley
TEKniques Staff

If you want to display a character on the screen, but not store it, use `PRI @32,23: "c"` where `c` = the character. For instance, try the following routine.

```

300 DIM X(4),Y(4)
310 DATA 30,45,60,85,20,40,65,40
320 READ X,Y
330 FOR I=1 TO 4
340 MOVE X(I),Y(I)
350 FOR J=1 TO 10
360 PRINT @32,24: "*"
370 NEXT J
380 NEXT I

```

The character "*" is refreshed on the display for about 1/4 second each time statement 360 is executed. Only the first character of a designated string will be executed. For example, if statement 360 had read

```
360 PRINT @32,24: "TALK"
```

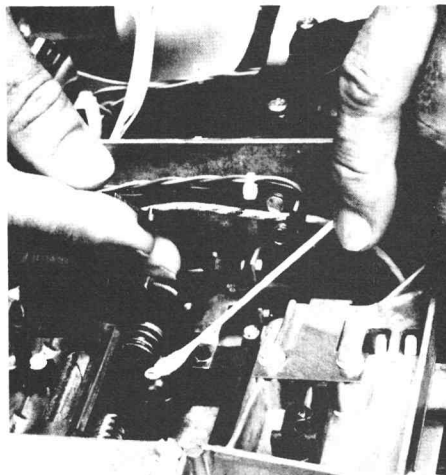
only the "T" would be refreshed.

Clean your Disk Read/Write Head

(From the 4907 Service Manual)

The Read/Write head of your 4907 Disk unit should be cleaned after each 12 months of normal use. The procedure is:

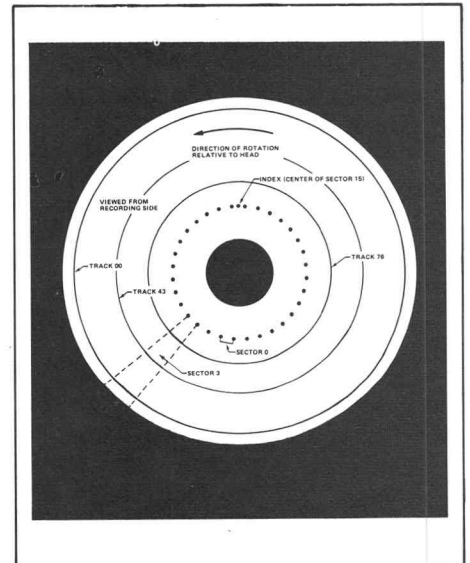
1. Remove power cable.
2. Remove top cover of cabinet (3 screws on each side).
3. Use a cotton swab and denatured alcohol to remove accumulated oxide from the head. Or, use one of the commercially prepared aerosol magnetic head cleaners.



Note: The head should not require frequent cleaning. Clean once a year, as necessary.

CAUTION: To prevent possible damage to the torsion spring, the load arm should never be opened over 90 degrees from the carriage assembly, or while at track 00.

Manually rotate the motor shaft to advance the read/write head away from track 00.



String Comparisons

by Patricia Kelley
TEKniques Staff

When you are checking for a numeric value in a string variable, don't inadvertently insert an extra step. For example.

```
IF ASC(A$)>57 OR ASC(A$)<48 THEN . . .
```

requires the 4050 System to convert the character in A\$ to its decimal equivalent and then compare. Whereas, if you had used

```
IF A$>"9" OR A$<"0" THEN . . .
```

A straightforward comparison would have accomplished the same end without the needless conversion.

The following simple routine carries the idea a bit further.

```
LIS
100 INIT
110 PRINT "INPUT A$ "
120 INPUT A$
130 IF A$<="9" AND A$>="0" THEN 200
140 PRINT "TRY AGAIN"
150 GO TO 110
160 END
200 A=VAL(A$)
210 PRINT A
220 END
```

STR Function

by Patricia Kelley
TEKniques Staff

The STRing Function converts a numeric expression to a character string and assigns the character string to the specified target string variable. If you dimension your target string to something less than the default of 72 characters, you should be aware of the conversion format (which is the default format when PRINTing numbers).

The number is printed as specified unless:

it's between -1 and +1 and has more than 14 character positions including the decimal point

it's between -1 and +1 and has 3 or more leading zeros

it's not between -1 and +1 and has more than 13 character positions including the decimal point

it's greater than 10 million and in standard notation

it's outside the numeric range of the 4050 System

In all cases, the STR Function always puts a space as the first character in the target string.

The following number is printed as specified, but with a leading space resulting in a string length of 3.

```
A$=STR(29)
A$
29
LEN A$
3
```

Below a 17 character number between -1 and 1 is reduced to 14 character positions including the leading zero and decimal, plus two positions for the leading space and negative sign, resulting in a string length of 16.

```
B=-.12345678901234567
B$=STR(B)
B$
-.123456789012
LEN B$
16
```

The following uses the same format as the preceding example, however, the positive sign isn't included, resulting in 15 character positions.

```
C=+.12345678901234567
C$=STR(C)
C$
.123456789012
LEN C$
15
```

A number between -1 and 1 with three or more leading zeroes will be converted to scientific notation.

```
C=-.0001
C$=STR(C)
C$
1.0E-4
LEN C$
7
```

A number not between -1 and 1 but less than 10 million will be printed as specified, up to 13 character positions, including the decimal point. In the following example, notice 13 characters plus the minus sign and leading space result in a length of 15.

```
D=-89.012345678912
D$=STR(D)
D$
-89.0123456789
LEN D$
15
```

Notice the rounding:

```
D=+9999999.1234555
D$=STR(D)
D$
9999999.12346
LEN D$
14
```

A number whose absolute value is greater than 10 million will be converted to scientific notation with up to 9 digits of accuracy to the right of the decimal.

```
E$=STR(12987765.345)
E$
1.298776535E+7
LEN E$
15
```

Nine digits to the right of the decimal, one to the left, plus the decimal, the scientific notation and the leading space result in a length of 15 characters.

```
E$=STR(-12987765)
E$
-1.2987765E+7
LEN E$
14
```

An eight character number results in a 14 character length.


```
E$=STR(89145)
E$
5.278983433E+87
LEN E$
16
```

The STR function is not restricted to constants. It will work for any numeric expression whose result is within the numeric range of the 4050 system.

Constants specified outside of the numeric range of the 4050 System will be converted to the number closest to the range boundary.

```
F$=STR(345.892E+345)
F$
8.988465674E+307
LEN F$
17

F$=STR(-1.234897654E499)
F$
-8.988465674E+307
LEN F$
18
```

For more information on the PRINT format, see the INPUT/OUTPUT section of your 4050 Graphic System Reference Manual. 

4050 Series Applications Library Program Abstracts

Order

Documentation and program listings of each program are available for a nominal charge. Programs will be put on tape or disc for a small recording fee per program plus the charge for the tape cartridge or flexible disc. One tape/disc will hold several programs. Programs will be recorded on like media only, i.e., programs on tape cannot be sent on disc and vice versa unless so noted in the abstract.

(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	\$20 per program
Recording Fee	5 per program
Tape Cartridge	30 per tape
Flexible Disc	15 per disc

Contribute

Contribute one program to the Library and receive three in exchange. Send in the membership card from your 4050 Series Graphic System Reference Manual to get the details. Or call us (503) 682-3411, ext. 3618.

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 #: 51/00-0603/0

Title: **Mortgage/Loan Payment Table**

Author: Tim Giesbers

Tektronix, Inc.

Beaverton, OR

Memory Requirement: 8K

Statements: 115

Files: 1 ASCII Program

The program will compute a table for a mortgage or loan. Each line of the table includes a month, year, payment for the month, interest for the month, principal paid for the month and the balance at the end of the month.

User Prompted Input:

Amount of the mortgage/loan

Interest rate (in percent)

Amount of the payment per month

When payment starts (month and year)

Number of months to be printed in the table

The program will output the number of months it will take to pay off the mortgage/loan and then print the table. The table will be printed with 21 lines per page and requires a carriage return to continue (allowing the user to make a hard copy of the screen if desired).

```
Enter the amount of the mortgage: 1000
Enter the interest rate (in percent): 10
Enter the amount of the payment per month: 100
Payments start when (i.e. '2,80' is February, 1980): 1,80
How many months do you want printed: 12
```

Your last payment will be in month no. 11

```
*****
Month      Payment      Interest      Principal      Balance
*****
JAN 80     $100.00       $8.33         $91.67         $908.33
FEB 80     $100.00       $7.57         $92.43         $815.90
MAR 80     $100.00       $6.80         $93.20         $722.70
APR 80     $100.00       $6.02         $93.98         $628.72
MAY 80     $100.00       $5.24         $94.76         $533.96
JUN 80     $100.00       $4.45         $95.55         $438.41
JUL 80     $100.00       $3.65         $96.35         $342.07
AUG 80     $100.00       $2.85         $97.15         $244.92
SEP 80     $100.00       $2.04         $97.96         $146.96
OCT 80     $100.00       $1.22         $98.78         $48.18
NOV 80     $100.00       $0.40         $99.60         $0.00
*****
```

Do you want to run this program again (Y/N)

ABSTRACT #: 51/00-1606/0

Title: **Performance Prediction of Sailcraft**

Author: Alex Gares
 University of South Florida
 Tampa, FL

Memory Requirement: 32K
 Peripherals: Optional—4641 Printer
 —4662 Plotter

Statements: 589
 Files: 1 ASCII Program

The program allows the user to predict the speed of any sailing craft with respect to the wind velocity and angle to the true or apparent wind direction. Polar diagrams are generated of the ratios of:

- Boat speed to true wind speed
- Boat speed to apparent wind speed
- Velocity made good to windward to true wind speed

The program permits the sailboat designer, handicapper or performance sailor to evaluate fully the effect of the various significant parameters on sailing craft velocities at all angles to the apparent and true wind and generate polar plots.

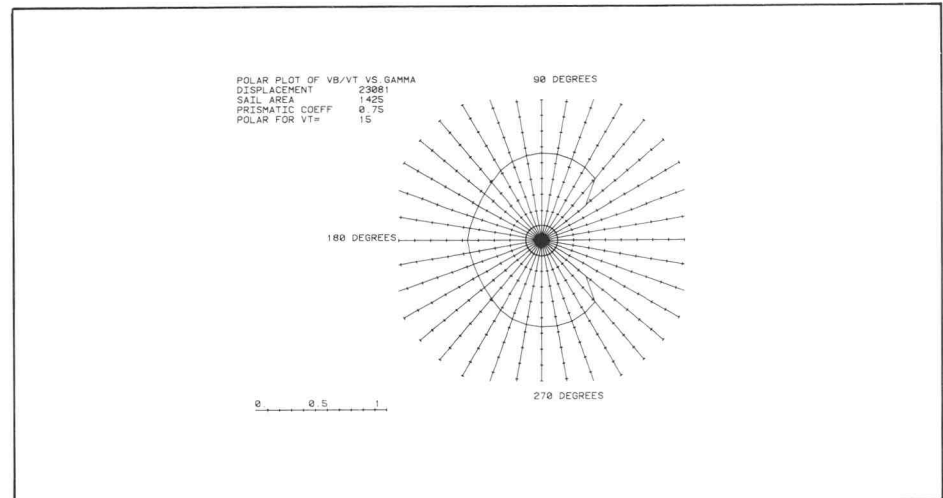
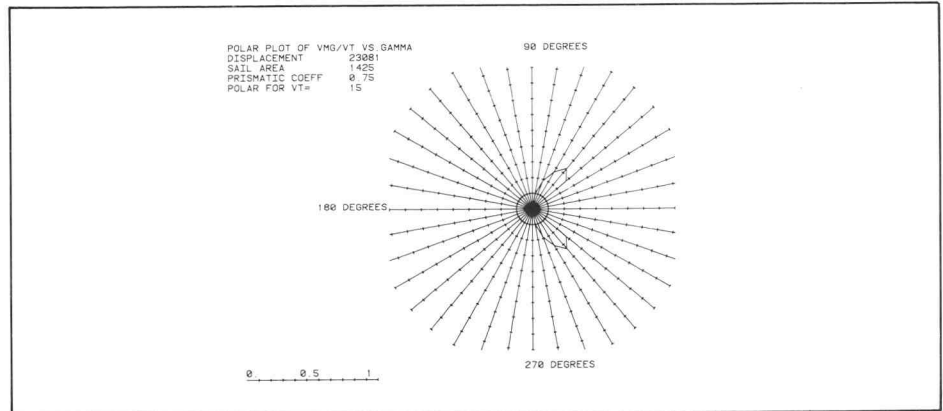
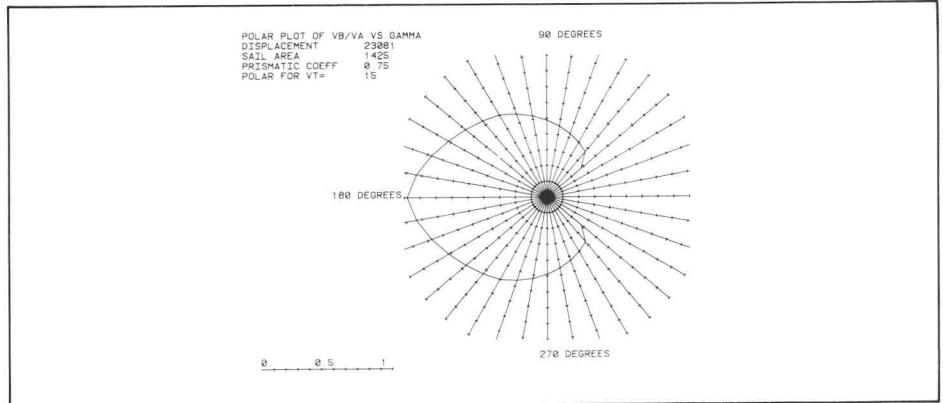
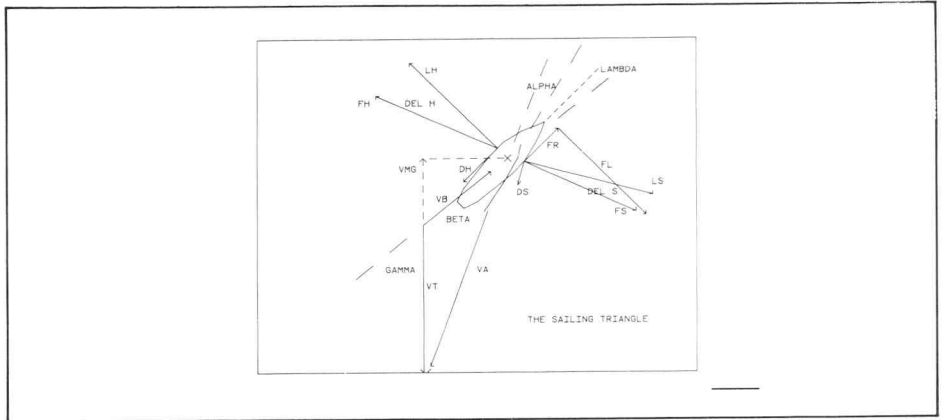
A test routine for a catamaran is included in the program to demonstrate the output. Also included in the program is a picture of the vectors involved.

User Prompted Input:

- Sail area in FT² 2
- Sail lift coefficient
- Sail drag coefficient
- Waterline length in FT
- Waterline length from bow to widest beam in FT
- Displacement in LBS
- Prismatic coefficient
- Hull drag force at 3.16 knots in LBS
- True wind velocity in knots
- Angle between true wind and boat velocity in degrees

The program is a result of a senior level engineering research project and stems from a study of over 20 references in the field.

TEST CASE FOR A CATAMARAN HULL
 THE FOLLOWING VALUES WILL BE USED FOR THE TEST CASE
 SAIL AREA=1425 FT²
 SAIL LIFT COEFFICIENT=1.0
 SAIL DRAG COEFFICIENT=0.2
 WATERLINE LENGTH=35 FT.
 WATERLINE LENGTH FROM BOW TO WIDEST BEAM =19 FT
 DISPLACEMENT =23001 LBS.
 PRISMATIC COEFFICIENT =0.75
 HULL DRAG AT 3.16 KNOTS =61 LBS.
 TRUE WIND VELOCITY =15 KNOTS
 HIT 1 FOR COMPLETE POLAR DIAGRAMS, HIT 2 IF NOT.



ABSTRACT #: 51/00-9544/0

Title: Pie Chart with Shading and Transparency Routines

Author: Fred Fachtet
Social Security Administration
Chicago, IL

Memory Requirement: 32K
Peripherals: 4662 Plotter
Statements: 655
Files: 1 ASCII Program

The program allows the user to create a pie chart, and optionally, alter data values, main title, subtitle and segment or slice labels. The pie is previewed on the 4050 screen, and can be plotted on the 4662 Plotter. In the Plotter mode, the user has the option of changing pen colors for the title (main and sub), one or more segments of the pie, and the border.

The user may also elect to offset one or more segments of the pie. For additional emphasis, apart from color options on the 4662 Plotter, the user may also use the shading routine to outline the segment and have it shaded.

The transparency routines allow the user to annotate the chart with standard horizontal or vertical text. The user may also select different character fonts to label the pie chart with non-standard characters for foreign languages or currencies.

The shade and transparency routines may be used independently to highlight and/or annotate, make geometric figures, produce text, overhead transparencies, report cover sheets and so on.

User-Definable Keys Provide:

- Pie Chart
- Change Data
- Screen
- Plotter
- Shade
- Transparency
- Fonts
- Menu/Restart

User Prompted Input:

- Title
- Sub-title
- Number of segments
- Labels (12 characters maximum)
- Data values

The shading program was modified from TEKniques Vol. 3, No. 1 Programming Tip. Special thanks go to Nathan Oxhandler (Tektronix, Inc., Santa Clara Field Office) for his conception of the exploding pie chart technique.

```

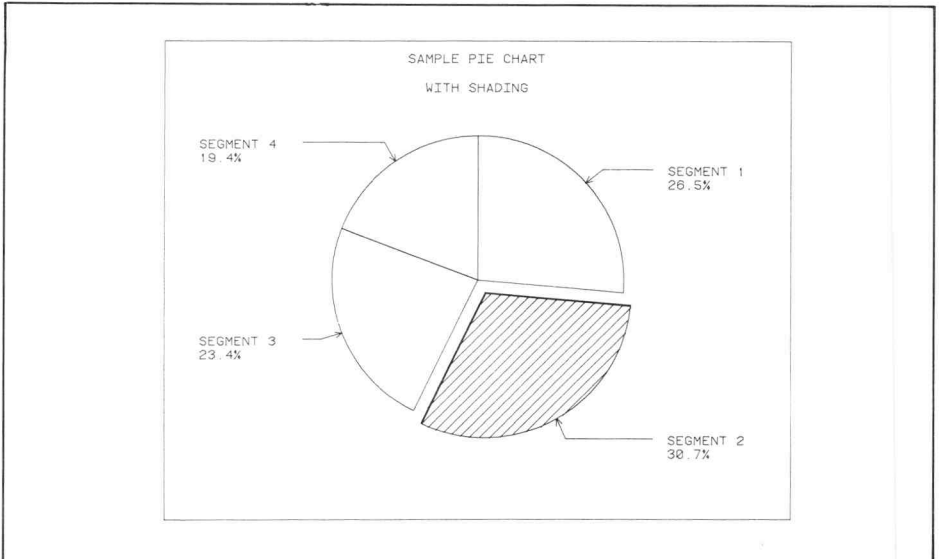
[ < ] > # $ \ | @           FONT 0
. . . . .                     FONT 1
A a A a # $ 0 0 e           FONT 2
A a u u £ $ 0 0 e           FONT 3
[ < ] > £ $ \ | @           FONT 4
i < > # $ n | @           FONT 5
[ + ] > # $ \ k £           FONT 5
SELECT FONT NUMBER FOR PLOTTER
DEFAULT IS 0 -- JUST PRESS RETURN:

```

```

*** MENU ***
LEFT JUSTIFY TEXT = T
RIGHT JUSTIFY TEXT = R
USER POSITION TEXT = P
VERTICAL TEXT = V
ARROW = A
QUAL = Q
CIRCLE = C
LINE = L
BOX = B
DIAMOND = D
REPEAT LAST FIGURE = I
SELECT FUNCTION:

```



ABSTRACT #: 51/00-5703/0

Title: **One Factor Repeated Measures Analysis of Variance**

Author: Richard M. Engeman
U.S. Fish and Wildlife Service
Denver, CO

Memory Requirements: 16K
Peripherals: Optional—4641 Printer
Statements: 193
Files: 1 ASCII Program

The program calculates the univariate analysis of variance for data from a one factor repeated measures experimental design (see Winer, *Statistical Principles in Experimental Design*, pps 261-308).

Data is input from the keyboard and allows the user to make any necessary corrections. The program then outputs the following:

- Analysis of variance table
- Subject means
- Means for treatment levels

All output, including the raw data, may be printed to the screen or to the 4641 Printer.

User Prompted Input:

- Number of treatments
- Number of subjects
- Each subject's data
- Output device (screen or printer)

```

THE DATA STRUCTURE AND NOTATION ARE AS FOLLOWS

      TREATMENT
-----
SUBJECT 1 2.....K
 1 x11 x12....x1K
 2 x21 x22....x2K
 . . . . .
 H xH1 xH2 .xHK
-----
    
```

```

SUBJECT MEANS
SUBJECT 1 27
SUBJECT 2 16
SUBJECT 3 23
SUBJECT 4 34
SUBJECT 5 24.5

PRESS G & RETURN AND THE SCREEN WILL PAGE
AND THE MEANS FOR TREATMENT LEVELS WILL PRINT OUT
    
```

```

THIS IS YOUR RAW DATA
30 28 16 34
14 18 10 22
24 20 18 30
38 34 20 44
26 28 14 30

DO YOU WANT THE DATA PRINTED ON ANOTHER DEVICE? Y OR N
    
```

```

MEANS FOR TREATMENT LEVELS
LEVEL 1 26.4
LEVEL 2 25.6
LEVEL 3 15.6
LEVEL 4 32

ANOTHER DATA SET? Y OR N
    
```

```

          ANOVA TABLE
-----
SOURCE      DF      SS      MS      F
-----
WITHIN SUBJECT  4.    680.000
BETWEEN SUBJECT 15.    811.000
TREATMENTS      3.    698.200
RESIDUAL       12.    112.800
-----
TOTAL         19.   1491.000

PRESS G & RETURN AND THE SCREEN WILL PAGE
AND THE PROGRAM WILL CONTINUE
    
```

4050 Series Applications Libraries

Africa, Europe, Middle East

Contact local sales office

Australia

4050 Series Applications Library
Tektronix Australia Pty. Limited
Sydney
80 Waterloo Road
North Ryde, N.S.W. 2113

Canada

4050 Series Applications Library
Tektronix Canada Ltd.
P.O. Box 6500
Barrie, Ontario
Canada L4M 4V3

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

IDD Group
Export Marketing
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077
U.S.A.

Japan

4050 Series Applications Library
Sony/Tektronix Corporation
9-31 Kitashinagawa-5
Tokyo 141 Japan

United States

4050 Series Applications Library
Tektronix, Inc.
Group 451
P.O. Box 500
Beaverton, Oregon 97077

Tektronix[®]
COMMITTED TO EXCELLENCE

TEKTRONIX, INC.
Information Display Division
Applications Library
Group 451
P.O. Box 500
Beaverton, Oregon 97077

BULK RATE
U.S. POSTAGE
PAID
TEKTRONIX, INC.

MICHAEL OLIVERI
LONG ISLAND FIELD OFFICE

IF YOU HAVE MOVED
PLEASE CALL 3618

Address Correction Requested — Forwarding and Return Postage Guaranteed.