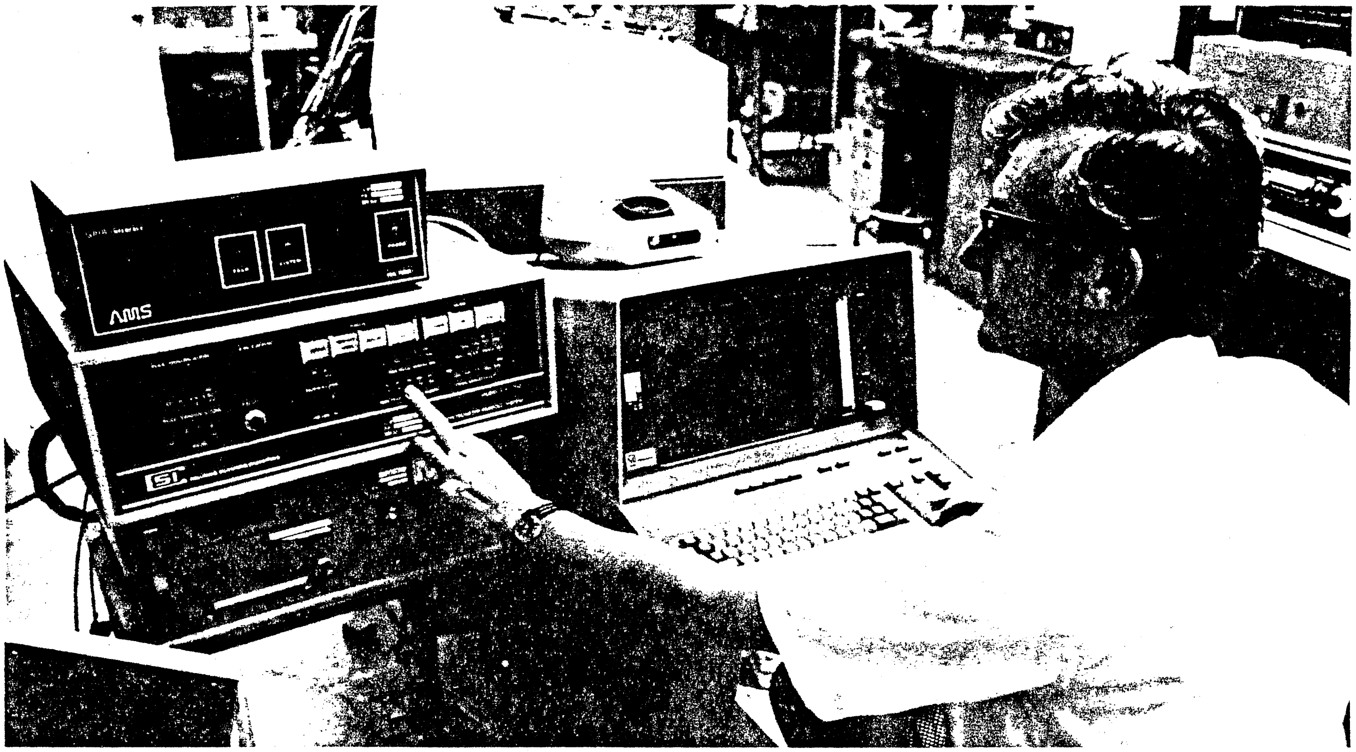
TEKniques

The 4051 Applications Library Newsletter March 15, 1979 Vol. 3 No. 2



Increased speed and reduced cost in the identification of unknown chemical compounds resulted when General Electric at Valley Forge, Pa., put together this system. A TEKTRONIX 4051 and 4907, A.M.S. Solver One interface and software. Perkin-Elmer Gas Chromatograph/Mass Spectrometer and a Columbia Scientific digital readout comprise the powerful analysis tool. *(Courtesy of General Electric, Valley Forge, Pa.)*

A 4051-Based Gas Chromatograph/Mass Spectrometer Data System

by Nick Bazil
A.M.S. Inc.

Gas Chromatograph/Mass Spectrometer systems are powerful tools for identifying and verifying unknown chemical compounds. They can be used to identify a pollutant in a water system, for instance, or any chemical sample. At General Electric Co. in Valley Forge, PA, a Gas Chromatograph/Mass Spectrometer system is in place in their analytical chemistry lab; its heart is a 4051 Graphic System with a 4907 File Manager. The 4051 handles the data reduction and analysis for the system, while the 4907 is the key to quickly matching the unknown sample against the library of known compounds.

In the past, data reduction and analysis for a Gas Chromatograph/Mass Spectrometer combination required a minicomputer, a requirement that could add fifty to eighty thousand dollars to the cost of a system. The high cost made it difficult to justify such data systems. But the advent of the powerful 4907 File Manager, coupled with the A.M.S. Solver One interface and application software package, has considerably reduced the cost of such systems.

The Gas Chromatograph/Mass Spectrometer system began with a 4051 Graphic System and an A.M.S. interface; the interface collected data from a Perkin-Elmer Gas Chromatograph/Mass Spectrometer system, through a Columbia Scientific digital readout connected to them. The digital readout unit reads the peaks from the

In This Issue

Gas Chromatograph and Mass Spectrometer, sorts the data, and performs an intensity count (Fig. 1).

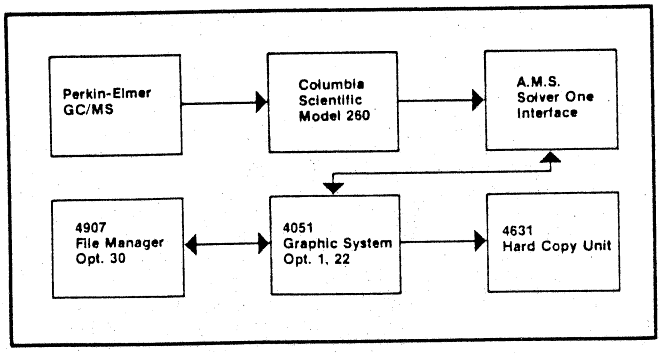


Fig. 1. System Configuration.

In this original system, G.E.'s Bob Ross intended to simply collect, normalize, graph, and tabulate the data locally, taking advantage of the 4051's high resolution display. (Refer to Figs. 2 and 3.) A time-share system could then be used to identify the unknown compounds from the mass spectrum run. A given sample in G.E.'s analytical chemistry lab could have 50 Gas Chromatograph peaks, denoting different compounds and different intensities or concentrations.

Visual interpretation and hand data entry with such

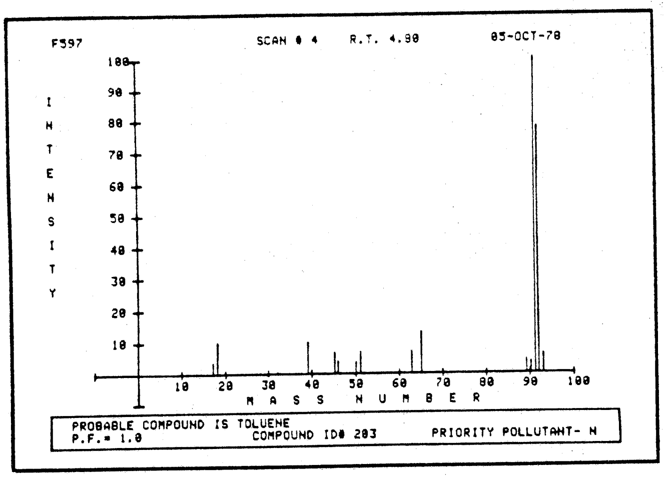


Fig. 2. Graphic Output of a mass spectra scan.

F597 SCAN # 4 R.T. 4.80 05-OCT-78

#	MASS#	INTEN	ACTUAL	#	MASS#	INTEN	ACTUAL
1	91.2	100.00	2914				
2	92.1	77.63	2262				
3	65.1	12.77	372				
4	39.0	9.81	286				
5	18.0	9.78	285				
6	63.0	6.76	197				
7	51.0	6.69	195				
8	45.0	6.55	191				
9	93.1	5.90	172				
10	89.2	4.19	122				
11	46.0	3.71	108				
12	90.2	3.64	106				
13	50.0	3.60	105				
14	17.0	3.47	101				
15	32.0	0.27	8				
16	28.0	0.00	-37				

PROBABLE COMPOUND IS TOLUENE
P.F.= 1.0 COMPOUND ID# 203 PRIORITY POLLUTANT- N

Fig. 3. Normalized and sorted data from the graph in Figure 2.

complex possibilities restricted productivity in the system. Two or three days were typically required to obtain a sample identification. The 4907 File Manager was then added to the system, and A.M.S. added a software package to identify compounds locally on the 4051, using the 4907 as the standard compound library. The result is a successful and pleasing system with local control and vastly increased speed and productivity.

System Capabilities

The system has two primary capabilities. One is its ability to collect multiple scans from the Mass Spectrometer, then search for known compounds to match each scan. The other is positive compound identification obtained by plotting retention time versus ion current.

To search for compound matches requires a library of standard compounds. The library for this system was created by storing the scans from a standard sample kit, on the 4907 File Manager. (The fact that this method of locally storing a standard library which reflects the characteristics of a given system is an added benefit.)

Searching for compound matches, in its purest sense, is quite simple. It requires taking the five most intense mass numbers from a scan, then searching through the standards library for a match of those numbers. This search can require thousands of iterations, but can be performed locally, with the 4907 in place, in a fraction of the time previously required.

During a search, the system will display the hit factor (5 out of 5, 4 out of 5, etc.), probable compound name, number, and priority pollutant for each of the run scans (Fig. 4). Optionally, the system will attach this information to the scan, so the scan can be replotted with the found compound information appearing on the graph.

Plotting retention time versus ion current for any three mass numbers is a powerful tool for positively verifying the compound identification. With this routine, any three

```
ENTER YOUR FILE NAME FOR THE SEARCH F597
THERE ARE 10 SCANS IN THIS FILE
  SCAN# 3 FOUND 5 OUT OF 5 : COMPOUND-  202N BENZENE
  SCAN# 4 FOUND 5 OUT OF 5 : COMPOUND-  203N TOLUENE
  SCAN# 5 FOUND 5 OUT OF 5 : COMPOUND-  204N XYLENE
  SCAN# 6 FOUND 5 OUT OF 5 : COMPOUND-  205N PHENOL
  SCAN# 7 FOUND 5 OUT OF 5 : COMPOUND-  206N OCRESOL
  SCAN# 7 FOUND 3 OUT OF 5 : COMPOUND-  207N PCRESOL
  SCAN# 8 FOUND 5 OUT OF 5 : COMPOUND-  207N PCRESOL
```

Fig. 4. Hard Copy output of the search for compounds on a run file.

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

Publishing Manager	Ken Cramer
Managing Editor	Patricia Kelley
Editor	Terence Davis
Graphic Design	John Ellis
Circulation	Rory Gugliotta

mass numbers can be selected for plotting during the entire run. The desired retention time can then be set to tabulate the ratio of intensities between those three mass numbers at that retention time (Fig. 5). The three mass ratios and the associated retention time verifies the probable compound identification.

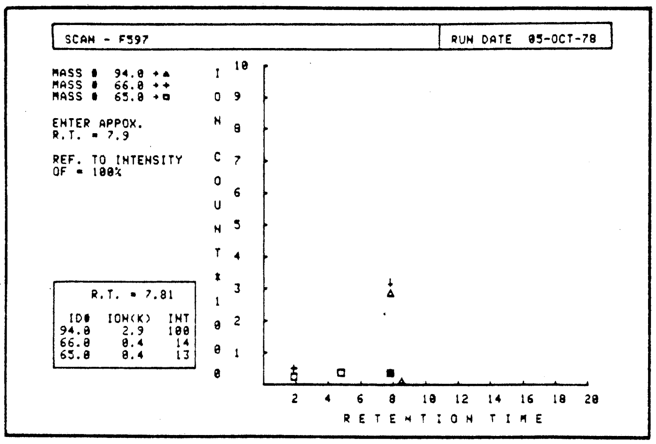


Fig. 5. Graph of retention time versus ion current for three mass numbers.

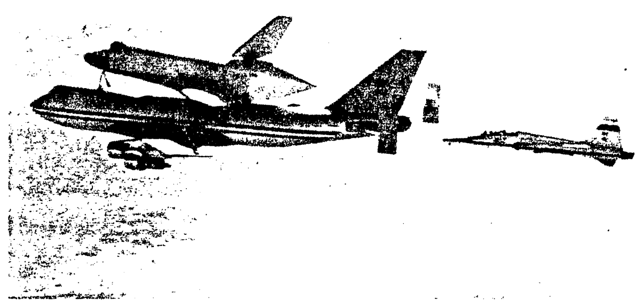
There are many other useful capabilities in the system, such as multiple background correction to compensate for column conditions. Also, any amount of mass numbers can be deleted from a run before any calculations are initiated, to clean up any bad samples that may have occurred. Data for each run can be stored on the disc for a run library and can be changed or deleted at any time. There are also utility commands for directories, automatic hard copy and disc to disc transfers.

The key to the successful operation of this system resides in the power of the 4907 File Manager coupled with the versatility of the Solver One system interface. They have made possible a desk-top Gas Chromatograph/Mass Spectrometer system, with data reduction and analysis, that produces desirable results at one-third the cost of a mini-based system. And the two or three days have been reduced to about twenty minutes, with a 4051 and a 4907 at the system's heart. (Any inquiries concerning the GC/MS data system should be directed to A.M.S. Inc., Box 873, Lake Elmo, MN 55042, (612)-439-0022.)

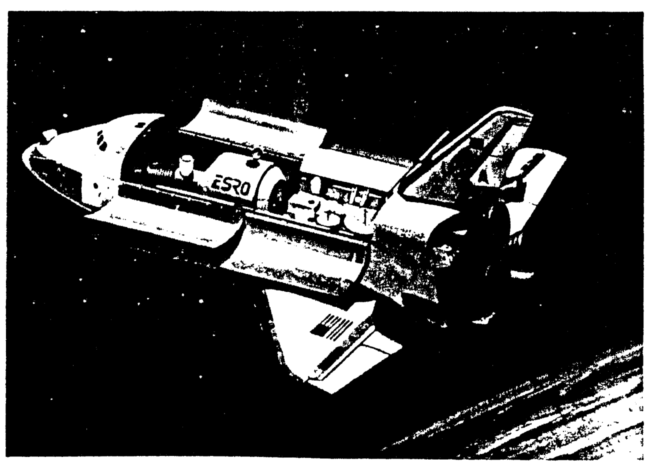
Science Fair Goes to Space: 4051 Helps

by Terry Davis

NASA's space shuttle flights, beginning early in the 1980's, will transform costly and complex space missions into routine, economical operations. The hybrid orbiter portion of the space shuttle lands like an airplane after its orbiting missions, to be checked out and launched again. The orbiter's huge cargo bay can carry heavy loads into space and bring them safely back again, landing on a three-mile strip. The many possible uses of such facilities stretch the average imagination. One early entry into the program, however, is the project of a 16-year old high school junior whose scientific interest and imagination made him a natural participant in the space shuttle project. And the 4051 is helping him on both ends of the project.



Space Shuttle Orbiter and 747 Carrier Aircraft. (Reprinted from NASA JSCL-157 *(U.S. Government Printing Office: 1977 774-457)*)



Shuttle Orbiter Carrying Spacelab. (Reprinted from NASA JSCL-133 *(U.S. Government Printing Office: 1975 671-191/6)*)

Beginnings

Bob Wheeler has long been interested in science. The sixth grade found him programming simple computers. In junior high his teachers allowed him an extra period to do science research, and by the ninth grade he was teaching science to seventh grade students. During this time, Bob did lots of self-imposed research, along with science fair projects. His last science fair project was measurement of lunar mountains—which placed fourth in the General Motors International Science and Engineering Fair at Anaheim, Calif. So research beyond the bounds of earth isn't new to him.

In the tenth grade, three busloads of high school students from the Ogden, Utah, area went to Edwards Air Force

Base in Southern California to watch space shuttle tests. Bob was in one of those buses and was immediately fascinated by the shuttle project, from the expanding tile heat shields to the computer-assisted flight. Now Bob is sixteen years old and a junior at Weber High School in Pleasant View, Utah. He's well on his way to producing a programmable microprocessor-controlled experiment for one of the early space shuttle orbital flights.

A parallel factor in Bob's space shuttle involvement is a North Ogden neighbor, R. Gilbert Moore. He works at Thiokol, the manufacturer of the two solid-fuel space shuttle motors. When engineers at NASA decided to offer for sale portions of the so-called "get-away space," the empty test flight payload area, Moore took an immediate interest. He reserved several of these cannisters, and offered one of them to Weber High School, where his son is also a student. Dr. James West, Science Coordinator, and Earl H. Heninger, Environmental Center Director, of Weber School District, teach advanced placement science courses at Weber High School. They immediately saw the opportunity for Bob to exercise his science and programming skills, and realize some of his fascination with the space shuttle program. With their encouragement, Bob started thinking about a project.

Getting Started

During June of last year, Bob attended a Space Shuttle Mini-Conference at Utah State University. There he and the other attendees were told of the requisites for a space shuttle project. Bob took particular note of NASA's Ernie Ott, who explained, "One of the biggest problems will be designing a controller to coordinate the timing and logic

Bob Wheeler, North Ogden, Utah, hopes to find the origin of cosmic rays. To accomplish this mission, he's sending a 4051-developed project along on one of the early space shuttle flights. At the left is John Hess, Tektronix Sales Engineer, who aided Bob in getting the kit to build the controller.

of the experiments." Having some experience in digital design and in programming, Bob began to think of a hardware timer as a project controller.

Two weeks after the Space Shuttle Mini-Conference, Bob was back at Utah State University for a National Science Foundation Summer Science Training Program. There Bob found a 4051 Graphic System in the math department, and got permission to use it. (Having previously read 4051 pamphlets, he was anxious to try his hand.) He and a friend, Dale Sather, spent all of their spare time during the six-week conference developing programs on the 4051. Bob says, "As with most people who are introduced to the 4051, we were convinced of its capabilities after using its powerful BASIC and excellent graphics." Bob and Dale even worked their class projects on the 4051, so they'd have more time for their own 4051 application projects.

One of their projects was a simulation of Apollo from lift-off to splash-down; this was a natural with their mutual interest in computing and Bob's love of astronomy. Working closely, Bob and Dale read up on Apollo, researching facts about the mission. Then they designed a model, a simulation of an entire Apollo flight. This effort taught the two programmers every 4051 BASIC command. They even looked at the System Software tape code to see how operations were performed, and used what they learned to speed up their own graphics operations.

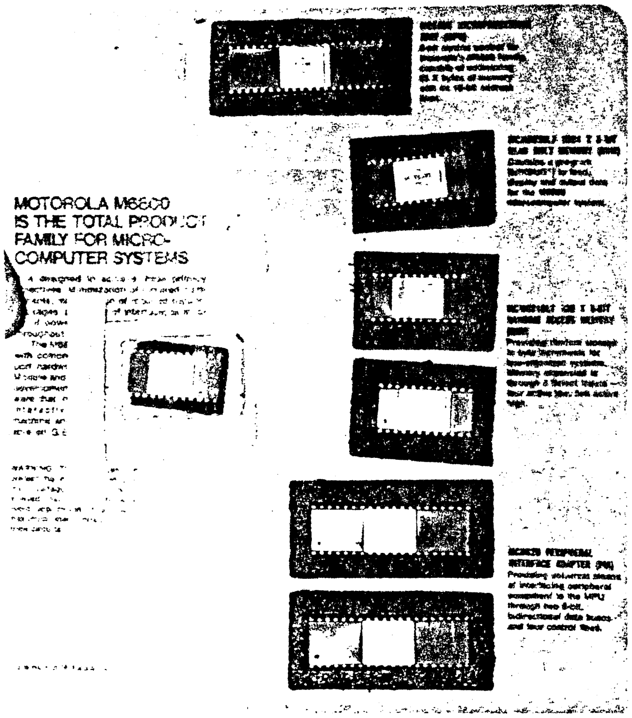
After learning that the 4051 was designed around the M6800 microprocessor, Bob wanted to know more about the 6800. For the next two months he read about the 6800, and studied 6800 Assembly Language. He soon decided that the flexibility of the 6800 made it a much better candidate for the heart of his project, and it replaced his hardware-timer idea. Then John Hess, a friend who works at the Tektronix Field Office in Salt Lake City, got together with Transera Corporation to donate a Motorola 6800 Evaluation Kit. Bob suddenly found himself with the parts he needed to build a 6800-based controller. His project was really under way!

More for the 4051

Bob was disenchanted with the time necessary to program the 6800 using its own hex code, so his next step was developing an assembler for the 6800. Again obtaining the use of a 4051, Bob developed his own assembler; most of it was developed in his room at home over Christmas vacation. The assembler runs on the 4051 and, through a special ROM, loads the 6800 controller memory with assembly-language programs. This lets Bob use the power of the 4051 **and** the small size of his 6800 processor for his space shuttle project.

Refining the Project

Bob soon began to feel that simply flying a working controller would be an opportunity missed. He began to think of a project that the processor could control, to examine some astronomical phenomena. Bob talked to the Electrodynamics Laboratory at Utah State University and found them willing to help design a cosmic ray detector for the project. So he began to plan a method of using his 6800-based controller to gather cosmic ray data. In the meantime, he thought there must be a way to reduce the size of the 6800 board. He called Motorola who put him in touch with Erwin Carroll at the Motorola plant in Houston. Carroll is now working with Bob to refine his controller for use with the cosmic ray detector.



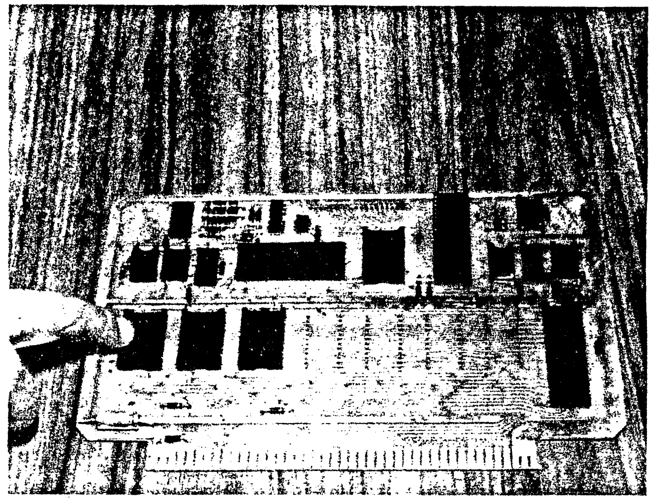
The Motorola 6800 Evaluation Kit has provided Bob with the parts for his project's controller. The MIKBUG ROM loads his microprocessor with hex code sent over the RS-232 from the 4051.

The detector will fly with the experiment, delivering an analog signal proportionate to the number of cosmic rays striking its surface. This signal will be fed to an analog-to-digital converter; the digital equivalent is then relayed to the 6800. The 6800 will record the cosmic ray intensity data on a small magnetic tape unit, along with the time of the sensor readings. This data will then be analyzed, after the orbiter returns from its mission.

```
*** TEKTRONIX 4051 / M6800 ASSEMBLER OBJECT PROGRAM ***

MEMORY ADDRESS      MEMORY CONTENTS      INSTRUCTION
    (HEX)               (HEX)            (MNEMONIC)

    0000                97 34              STAA $34
    0002                86 23              LDAA #$23
    0004                1B                 ABA
    0005                1B                 ABA
    0006                3F                 SWI

READY
```

To overcome the frustrations of programming in hex code, Bob developed his own assembler on the 4051. He inputs to the 4051 in mnemonic code which the 4051 segments into hex, and transmits to the 6800 microprocessor in the controller.

While the experiment is in operation, the space shuttle orbiter will be pointed in several directions. Thus readings will relate to the many directions, or points in space. After the mission, NASA will publish a report showing where the orbiter was pointed at any given time. Since the tape recorder will have intensity data recorded along with reading time, the position information can be easily extrapolated. The time and intensity data will be used as input to the 4051, which will analyze the data and provide graphic analysis. The hope is that the graph will show the general direction of cosmic ray origin in space.



Bob indicates where his first ROM will plug into his controller. Following this will be other ROMs and some RAMs. When NASA flips the switch on his controller, it will jump to the first address in this ROM and begin the program controlling his experiment.

The 6800 can handle several different sensors at once. Therefore, Bob plans to fly five or six experiments: he's designing additional projects with the help of the Electrodynamics Laboratory. The versatility of the 4051 Graphic System will allow Bob to look at the data in several different ways.

"So," Bob says, "the 4051 is really the main show." It will help develop the programs easily and quickly, and will convert them to machine language for the 6800. The 6800 microprocessor, also the 4051's heart, will control the experiment in flight. (Bob notes that the 6800 will not only control his experiment, but will branch to subroutines to service others' experiments.) And after the Space Shuttle flight returns, the 4051 will analyze the data and provide a detailed graphic analysis.

Scheduled for Space Shuttle flight number seven, the Weber High cannister, with Bob's controller, will fly early in the 1980's. When it returns, TEKniques will be there to cover the conclusion of this ambitious project.

Editor's Note: *John Hess, Tektronix Sales Engineer at Salt Lake City, brought the achievements of Bob Wheeler to the attention of TEKniques' staff, and arranged for an interview in Salt Lake City with Bob and his father, Dr. Bob Wheeler.*

Modeling and Reporting as Process Control Monitor

by Bob Cook

Previous issues of TEKniques have described the new TEKTRONIX Modeling and Reporting software. A specific application was described, along with a description of the capabilities and uses of the software package. The following is an example of a typical Modeling and Reporting installation in a process control environment, in this case, a sewerage treatment facility.

The agency in question resulted from the consolidation of nine city and 26 independent sanitary districts, and is responsible for all sewers and sewerage treatment within a growing county. The facility handles half of the waste in the county, and currently processes an average of 10 million gallons of wastewater daily. Plant capacity is 20 million gallons per day, and is expected to grow to 60 million gallons by the year 2000.

Applying the Modeling and Reporting System

When the wastewater (sewerage) comes to the plant, it is full of suspended solids that must be removed; several steps are used to do this. After initial filtering and settling, the wastewater is placed in aeration basins, where microorganisms are introduced. These organisms digest the suspended organic solids, producing solids that will settle. The settled solids are referred to as sludge.

After the biological treatment stages, the wastewater passes to the chemical purification stages. There, chemicals such as lime and aluminum sulfate are introduced. These chemicals combine with chemical pollutants like phosphorus, causing them also to settle out as sludge.

In the next stage, the sludge by-products are thickened, dewatered, and incinerated. The remaining processed water passes through a final filter, and is then chlorinated and discharged into a nearby river, cleaner than the river itself.

The key aspect of this operation is sludge production and processing; it must be carefully monitored and controlled. The Modeling and Reporting system analyzes sludge and phosphorus removal throughout the several processing stages.

Readings for flow in the tanks are taken every midnight. The percentage of solids in the flow is checked every two hours. Without Modeling and Reporting, these readings were jotted down by hand on a form. Analysis was done manually when it was deemed necessary. Besides being subject to manual calculation errors, these analyses took as long as 30 hours to produce. As a result, they were done only when absolutely necessary.

At the same time, the treatment process is slow to respond to corrections. This makes it difficult to maintain stable operation **without** readily-available analysis of current status and trends. Modeling and Reporting provides an easy-to-use method of obtaining the needed analysis whenever desired, to help maintain the stable process. Graphics provides a superior way to visualize and track process trends.

With Modeling and Reporting, readings are easily entered into the system daily; analysis is available in a matter of hours. Reports are presented in neat hard copy output, instead of the old handwritten form. Modeling and Reporting improves the operation by providing timely decision data, with great time savings and without manual error.

Material savings from the stabilized process have not yet been quantified, but the time savings is easily calculated. The analysis that takes 30 hours by hand can be done in four with Modeling and Reporting. Using this calculation alone, at a standard hourly labor rate, the Modeling and Reporting System could pay for itself in less than 18 months. Plus, the graphic reports are quickly produced and easily interpreted.

The same kind of process control could be used in new water processing plants, or such heavy industries as electroplating, casting, chemical laboratories, and pulp and paper plants. They could also benefit from the time and money savings, coupled with enhanced performance. They're all made possible by the 4051 Graphic System, with the powerful Modeling and Reporting Software package.

TEKniques a Winner!

TEKniques recently took part in the annual Communications Competition sponsored by the Society for Technical Communications. We entered the regional competition sponsored by the Willamette Valley Chapter of the society, and came back with the first place position in our category. Our entry has been forwarded to the national competition, where it will face other entries from electronics, medicine, forestry, and other industries.

We're very pleased with this achievement. Our winning entry would not have been possible without your interesting applications and tips, so we all share in this success. Thank you for the help!

*Editor's Note

Catalogs and Back Issues

Did you miss an issue of TEKniques, from Volume 1 or 2? Or perhaps you haven't yet gotten your copy of the 4051 Applications Library Program Catalog. We have catalogs and back issues of TEKniques on hand in the TEKniques office. If you'd like to receive a catalog, or have been wanting to find a copy of a previous issue of TEKniques, drop a note to the Applications Library serving you; Library addresses are located at the back of each TEKniques issue.

Program Tip Exchange

Send in your programming tip. Any one of the following 4051 Applications Library programs* will be yours when it's published. Simply jot down a brief description of the function, the code and your choice of program. Mail it to the 4051 Applications Library serving you; Library addresses are listed at the back of each TEKniques issue.

51/00-0101/0	51/00-5503/0
51/00-0702/0	51/00-7002/0
51/00-0715/0	51/00-8006/0
51/00-1401/0	51/00-9505/0
51/00-1402/0	51/00-9511/0
51/00-5401/0	51/00-9521/0

*Documentation and listing only.

Business Applications Contest Winners

The winners have been selected for the TEKniques business applications contest. The five winning entries are contained in the new abstracts section of this issue by TEKniques; the winners are listed below.

First Place: Dr. P.C. Holman has contributed programs and applications to TEKniques in previous issues. His winning entry is a collection of algorithms for business and accounting. See Abstract No. 51/00-0904/0.

Second Place: Robert S. Pilkington, of A.T. & T. Long Lines, Bedminster, N.J. contributed a program entitled "Y" Axis Graphs—12 Month Format. The program is very well documented and flowcharted, and makes excellent use of negative numbers. See Abstract No. 51/00-0905/0.

Third Place: Mallory M. Green, a computer specialist with the U.S. Department of Housing and Urban Development, Washington, D.C. has also contributed to previous issues of TEKniques. His third place winning entry is BARGRAPH II, which generates professional horizontal or vertical bar graphs in a simple interactive manner. It is further described in Abstract No. 51/00-0907/0.

Fourth Place: Mallory M. Green also took the fourth place honors with TIME SERIES II. It, like BARGRAPH II, is useful for business and general applications; both use the same data format, so they work interchangeably. Refer to Abstract No. 51/00-0906/0.

Fifth Place: R.J. Reimann is an Assistant Professor in the Department of Physics and Engineering as Boise State University, Boise, Idaho. His entry, Regular Plot, is a general graphics program developed for industrial market analysis. Previous contributions to TEKniques include the Graphics Demonstration for Astronomy and Physics. His prize winner is described in Abstract No. 51/00-0908/0.

Congratulations to all, and a big "Thank You!" to all who entered.

Programming Tips

Off-Line Plotting

by Carl Dawson

Tektronix, Inc.

Off-line plotting speeds program execution and frees the 4051 for other tasks. During program run time, it's quicker to send the graphics commands to tape instead of sending them to the plotter with the attendant waiting while they are plotted. After the graphics data is generated and sent to tape, you can plan the plot files back from a TEKTRONIX 4924 tape unit to the 4662 Plotter. This technique is especially useful when producing multiple plots, or when several users share a single plotter.

DAB and GDU

Off-line 4662 plot files require Data Byte Commands (DAB) and Graphic Display Units (GDU). DAB commands program the plotter by passing the command as part of the data. As a quick reference, the following is an abridged list of DAB commands.:

MOVE	M
DRAW	D
PRINT	P
HOME	H
ALPHAROTATE	R
ALPHASCALE	S
ALPHARESET	A
PROMPT LIGHT	T

User Data Units are defined by the WINDOW, VIEWPORT, ROTATE and SCALE commands. Four BASIC keywords **transform** User Data Units into device dependent GDU's: MOVE DRAW, RMOVE and RDRAW. Thus a MOVE @1:X,Y will send the **transformed** X,Y coordinate to device 1.

To send transformed coordinates to tape requires specifying the secondary address (MSA).[1] If a secondary address is omitted, as it is in MOVE@1:X,Y, the 4051 automatically generates a secondary address that corresponds to the keyword. The previous command, therefore, internally looks like MOVE @1,21:X,Y. This default secondary addressing can be overridden.[2] MOVE @1,20:X,Y will actually execute a DRAW since the secondary address of 20 tells the peripheral device to DRAW.

If the default secondary address for MOVE (MSA=21) is overridden with the secondary address for the PRINT operation (MSA=12), then MOVE@1,12:X,Y will **print** the **transformed** coordinates on the plotter surface beginning at the current pen position. To carry this example one step further, a MOVE@33,12:X,Y will **print** the **transformed** coordinates to the current file on the internal tape drive.

There is one limiting factor. The User Data Units specified as X,Y coordinates of the MOVE and DRAW statements **must lie within the boundaries of the WINDOW** coordinates. When a draw is clipped, either none or two X,Y coordinate pairs will be issued by the 4051. Consequently, the coordinates of points outside the window will not be consistent with the character commands being printed by the user on the tape.

Plot File Format

Plot files, therefore, contain the appropriate plotter DAB commands and their associated data. Each value associated with a particular DAB command must be separated by a valid delimiter (space or comma) and each command must be followed by a valid terminator which may be one of the following:

1. An ETX byte (Control C)
2. A CR, comma, or space when the data sent to the Plotter is **numeric** and all expected data values have been received.

Two other terminators are valid in other modes of operation, but we'll ignore them since they aren't usable in the "off-line" plotting described here.

Building the Plot Files

For off-line plotting, all plotter commands must be in the DAB format and all graphic coordinates in GDU's. A sequence of two BASIC statements specify first the DAB command followed by the transformed coordinate:

---

[1]A complete list of secondary addresses is on page B-18 of the 4051 Graphic System Reference Manual.

[2]Secondary addresses can be suppressed. PRINT @1,32:"H" sends the character "H" and a carriage return to the plotter without sending a secondary address. In this case the plotter treats it as a command (HOME). Complete information on I/O addressing can be found in the 4051 Graphic System Reference Manual beginning on page 7-9.

```
200 PRINT @33:"M";
210 MOVE @33,12:X,Y
```

Line 200 prints the DAB character for MOVE to the tape file; the automatic carriage return (CR) is suppressed. If the CR were allowed, it would terminate the DAB command before the coordinates were specified. Line 210 transforms the X,Y coordinate into GDU's and **prints** these values to the tape file. In addition, a CR is generated by the print operation in line 210 thereby properly terminating the DAB command.

The following example builds a plot file containing a sine wave and a title. Note that all coordinates of the MOVE and DRAW commands lie within the WINDOW boundaries.

```
100 REM - PROGRAM TO BUILD PLOT FILE
110 INIT
120 WINDOW 0,2*PI,-1,1
130 VIEWPORT 0,150,0,100
140 FIND 1
150 REM - "MOVE" command sequence
160 PRINT @33:"M";
170 MOVE @33,12:0,0
180 FOR X=0 TO 2*PI STEP 2*PI/100
190 REM - "DRAW" command sequence
200 PRINT @33:"D";
210 DRAW @33,12:X,SIN(X)
220 NEXT X
230 REM - "MOVE" command sequence with GDU coordinate
240 PRINT @33:"M50,90"
250 REM - "PRINT" - command sequence
260 PRINT @33:"PSINE WAVEC"
270 REM - "HOME"
280 PRINT @33:"H"
290 CLOSE
300 END
```

Since a CR is considered valid data by the print "P" command, note that the ETX ("C") character was used as the terminator in line 260.

Creating Axes

The AXIS command cannot be used to generate axes in a plot file. The AXIS command causes the 4051 to generate a series of moves and draws, sending this information to the plotting device with the appropriate secondary addresses. Therefore, the AXIS command has to be interpreted by the 4051 as it is being executed.

However, axes can be generated in a plot file by writing a short subroutine which performs the same functions as the BASIC keyword "AXIS". This code is discussed in the **PLOT 50 Introduction To Graphic Programming in BASIC** in the section entitled "AXIS: Without Axis Command."

Off-Line Plotting

The plot file tape can now generate a graph on the plotter without the 4051. Connect a 4924 tape unit to the Plotter through the GPIB. Set the Plotter for "LISTEN ONLY" and "DAB" modes. That is, the four hexadecimal switches on the rear panel of the Plotter labeled "A B C D" should be set to "1 C 0 1." Put the 4924 tape unit in manual mode by releasing the ON LINE switch (ON LINE button should be up, not depressed). Rewind the plot file tape by pressing the REWIND button and locate the desired file(s) with the SKIP FORWARD or SKIP BACK buttons. Start the plot transfer by pressing the TALK button on the 4924. When the end of file is encountered, the 4924 will stop and the plot can be removed from the 4662 Plotter.

Off-Line Plotting Monitored

You may wish to prompt the operator to change paper on the plotter. The following small program will enable the 4924 to signal the 4051 when it has finished transmitting to the plotter. The 4924 talks directly to the plotter which frees the 4051 to execute any program not requiring the GPIB. When the end of file is detected by the 4924, it generates a service request (SRQ) and sets its error number to 12. The 4051 detects the SRQ and signals the operator. (ON SRQ THEN ...).

For this method to work, the plotter must be set for both MSA and DAB operations by setting the hex switches on the back panel to "1 0 0 1." The 4924 must be placed ON LINE by depressing the ON LINE button. The program sets up the transfer from 4924 to plotter, waits for the SRQ, then prompts the operator to remove the plot.

```
100 REM MONITORED OFF-LINE PLOTTING
110 INIT
120 ON SRQ THEN 500
130 REM 4924 is device 2, plot on file 1
140 FIND @2:1
150 REM Set up "peripheral to peripheral" transfer
160 REM Note "%" gets 4051 off the GPIB
170 WBYTE %66,122,33:
180 REM Wait (or execute a BASIC program)
190 WAIT
200 REM Continue, end of file detected
210 PRINT "GGG Please remove plot GGG"
220 END
500 REM SRQ service routine
510 POLL I,J;2
520 IF I=1 THEN 550
530 PRINT "GGPOLL ERROR"
540 STOP
550 REM Get tape unit error number
560 INPUT @2,30:E
570 IF E<>12 THEN 610
580 REM Untalk & unlisten peripherals
590 WBYTE @95,63:
600 RETURN
610 PRINT "GGGTAPE ERROR NUMBER ";E
620 STOP
```

Keyed File Access on the 4907

by Jack Gilmore
Tektronix, Inc.

The 4907 brings a random access file and record capability to the 4051. Although 4907 files are accessed by name, random access of records still requires the use of cardinal numbers (1, 2, ..., number of records allocated).

Sometimes it would be more desirable to access a record based upon an alpha string such as a name. Nonsequential groups of numbers, such as a social security number, could make a desirable access key. Some combination of the two, such as street address, might also be useful.

A method that constructs and searches symbol tables in compilers and assemblers is "hashing." The hashing method maps the set of keys into a restricted range of numbers or "hash codes." In our case, hashing transforms the key field such as a last name into a numeric value representing 1, ..., number of records.

Linear Hashing

The hashing function is chosen so that:

1. The computation of the hashing function is rapid;
2. The mapping of all keys to hash codes is uniformly distributed over the range of hash codes.

These codes are then used as record numbers to access the data file in a random fashion.

To enter a new record, the following procedure is used:

1. The hashing routine computes a record number in the file from the key.
2. If the key-field (first) in that record has the initialized value (alpha-blank or numerical-zero), the key, followed by the rest of the data for that record, is written at that position.
3. If the key-field is the same as the key for the record being inserted, the record already exists.
4. If the key-field is different than the key for the new record, the hash code (record number) is incremented by one. If it's greater than the number of records in the file, it is reset to one. A check is made to see if the record numbers have completely wrapped around once. If so, the file is full and error action is taken. If not, steps 2 through 4 are repeated.

To recover a record takes a similar operation:

1. The hashing routine computes a record number.
2. If the key-field (first) is blank or zero, then the record doesn't exist.
3. If the key-field is the same as the key for the record for which you are searching—Eureka! —it's been found.
4. If the key-field is different, the record number is increased and tested in the same manner as inserting a new record. Steps 2 through 4 are repeated and an error return is taken if the record does not exist (exhaustive search).

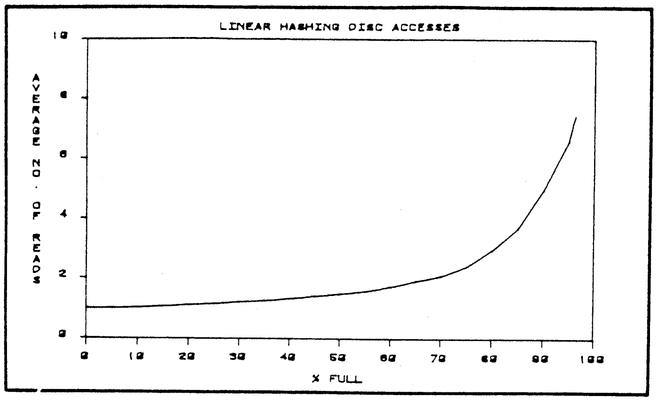
The following routines implement keyed file access for an alpha key.

```
100 KILL 'DATA'                              ┐
110 N0=0                                     │
120 N1=0                                     │
130 K0=200                                   │
140 K8=10                                    │
150 GOSUB 4000                               │
160 OPEN 'DATA';1,'U',A$                     │   Sample program to test
170 FOR J=1 TO 10                            │   key subroutines
180 K$=''                                    │
190 FOR I=1 TO 4                             │
200 C$=CHR(INT(RND(1)*26)+65)                │
210 K$=K$&C$                                 │
220 NEXT I                                   │
230 GOSUB 4230                               │
240 WRITE #1,K:K$                            │
250 NEXT J                                   │
260 GOSUB 4500                               │
270 GO TO 170                                ┘
4000 REM INITIALIZE FILE STRUCTURE
4010 REM K0 IS THE NUMBER OF RECORDS
4020 REM K8 IS THE RECORD LENGTH
4030 CREATE 'DATA';K0,K8
4040 OPEN 'DATA';1,'F',A$
4050 FOR K9=1 TO K0
4060 WRITE #1,K9:' '                             Set all keys to blank so we can
4070 NEXT K9                                     tell that a record is unused
4080 CLOSE 1
4090 RETURN
4100 REM HASH ROUTIENE                           For best results modify and test to
4110 REM K  = HASH CODE                          make sure this routine produces uniformly
4120 REM K$ = INPUT KEY                          distributed numbers for your particular
4130 K=LEN(K$)                                   keys.
4140 FOR K2=1 TO LEN(K$) MIN 4
4150 C$=SEG$(K$,K2,1)                            Key = Length + Σ(I=1 to 4) ASC(K$(I)*2^(I-1)
4160 K3=ASC(C$)
4170 K=K+K3*2^(K2-1)
4180 NEXT K2
4190 IF K<=K0 THEN 4210
4200 K=K-INT((K-1)/K0)*K0
4210 RETURN
4220 REM FIND KEY NO. FOR NEW RECORD
4230 GOSUB 4130                                  Calculate hash
4240 K9=K                                        Save it to compare for exhaustive search
4250 N0=N0+1                                     Number of record entered
4260 READ #1,K:F$                                Read key
4270 IF F$<>' ' THEN 4290                        Check for empty record
4280 RETURN                                      Success, found empty slot
4290 K=K+1                                       Nope, so try next record
4300 N1=N1+1                                     Number of extra reads
4310 IF K<>K9 THEN 4340                          Is key wrapped around once?
4320 PRINT 'FILE FULL'  ┐
4330 STOP               ┘ More sophisticated recovery technique could be used
4340 IF K<=K0 THEN 4260                          Check for key overflow
4350 K=1                                         If so, reset key
4360 GO TO 4260                                  Get next record
4370 REM FIND RECORD FROM KEY
4380 GOSUB 4130                                  Hash key
4390 K9=K                                        Save key for exhaustive search
4400 READ #1,K:F$                                Read key from file
4410 IF F$<>K$ THEN 4430                         Compare keys
4420 RETURN                                      Found it!
4430 K=K+1                                       Check next record
4440 IF K<>K9 AND F$<>' ' THEN 4470              Check for blank record or exhaustive search
4450 PRINT 'RECORD DOES NOT EXIST'┐
4460 STOP                         ┘ More sophisticated recovery could be used
4470 IF K<=K0 THEN 4400                          Check for key overflow
4480 K=1                                         Reset key
4490 GO TO 4400                                  Get next record
4500 REM SUMMARIZE STATISTICS
4510 PRINT 'FILE IS ';INT(N0/K0*100+0.5);'% FULL. ';
4520 PRINT 'AVERAGE READS PER ENTRY ';INT((N1+N0)/N0*100+0.5)/100
4530 RETURN
```

The following are advantages of the linear hashing method: quick access to a record, not having to read in the entire record to check if it's the right one, no tables to keep in memory, the routines to implement the technique are small.

However, once the file is created and initialized, it cannot be expanded; to do so would change the mapping of keys-to-record placement. This could be a major disadvantage in some applications although a program could be written to copy a file to a larger one, recomputing record placement in the process.

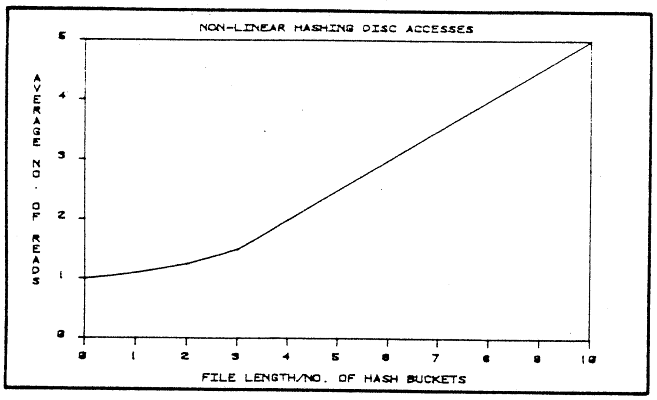
Another drawback is this: as the file fills up, the average number of reads to locate the desired record goes up dramatically, as shown in the following graph:



Non-Linear Hashing

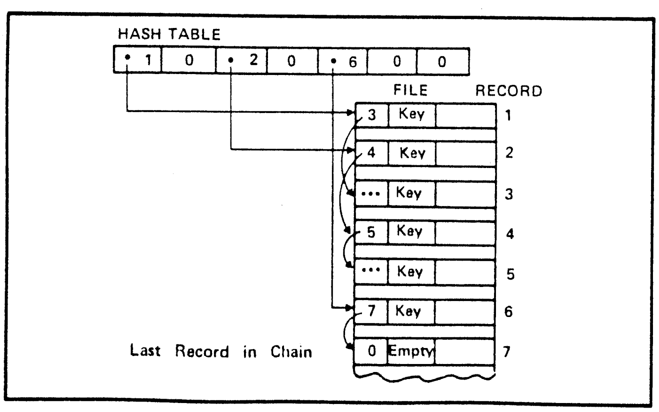
Non-linear hashing is another method of implementing keyed file access. It overcomes some of the linear hashing constraints, although there are tradeoffs. Non-linear hashing allows file expansion, and the records within the file do not have to be pre-written with null keys.

However, the routines are more complicated. A table containing record numbers must be retained in memory when the file is open and saved on the disc when the file is closed. When adding to a file, an entire record has to be read and modified. This technique has another performance characteristic demonstrated in the following graph:



If the file contains a large number of records, the hash table length could become unreasonably large to decrease the average number of accesses per read.

The following diagram illustrates the data structure of non-linear hashing.



The following routines implement non-linear hashing. Although the hash tables are stored as the first record in the file, they could be stored in a separate file. If stored separately, the hash table should be chosen independent of the record length. Each entry could also be in a separate record, so the entire hash table needn't be in memory at one time. This would require one additional disc access.

```
[illegible]                                  Program to test non-linear hashing
[illegible]                                  Initialize file
[illegible]                                  Number of bytes used
[illegible]                                  Record length
[illegible]                                  [illegible]
[illegible]                                  Input hash table
[illegible]
[illegible]                                  [illegible] new record
   WRITE [illegible]                         Write it
[illegible]                                  Increment record # for next available record
[illegible]
4000 REM INITIALIZE FILE [illegible]
4010 REM [illegible] = THE NUMBER OF HASH [illegible] STORABLE IN ONE RECORD
4020 [illegible]                             [illegible] # of hash buckets from record length
4030 REM [illegible] IS THE RECORD LENGTH
4040 CREATE "DATA"[illegible]
4050 OPEN "DATA"[illegible]
4060 DIM [illegible]
4070 [illegible]                             Initialize hash buckets to empty
4080 WRITE #1[illegible]
4090 CLOSE [illegible]
5000 RETURN
5010 REM HASH ROUTINE                        Same as linear hashing
5020 REM [illegible] = HASH CODE
5030 REM [illegible] = INPUT KEY
5040 [illegible]=LEN(N$)
5050 FOR [illegible]=1 TO LEN(N$) MIN 4
5060 C$=SEG(N$[illegible])
5070 N3=ASC(C$)
5080 [illegible]
5090 NEXT [illegible]
5100 IF [illegible] THEN [illegible]
5110 [illegible]
5120 RETURN
6000 REM FIND REC NO. FOR NEW RECORD
6010 GOSUB 5040                              Hash the key
6020 N1=0
6030 IF [illegible] THEN [illegible]         Check for empty hash bucket
6040 [illegible]                             Store record #
6050 WRITE #1[illegible]                     Write it to disc
6060 RETURN                                  Main program is responsible for writing entire
6070 REM N6 = NEXT AVAILABLE RECORD          record and incrementing K5
6080 [illegible]                             Hash bucket is full. [illegible] link to end of chain
6090 READ #1[illegible]                      K5 is link to the next record
6100 IF A$ [illegible] THEN [illegible]      Do keys match?
6110 PRINT "RECORD ALREADY EXISTS"
6120 STOP                                    Error procedure
6130 IF K5=0 THEN [illegible]                Have we found the end of chain?
6140 N=K5                                    No, chain down to next record
6150 N1=N1+1
6160 GO TO 6090                              Warning! This must be expanded to statement that reads
6170 WRITE #1[illegible]                     and writes the entire record for each record structure
6180 [illegible]                             Statistic variable (may be deleted)
6190 RETURN
6200 REM FIND RECORD FROM KEY
6210 GOSUB 5040                              Hash the key
6220 IF [illegible] THEN [illegible]         Check the hash bucket
6230 PRINT "RECORD DOES NOT EXIST"
6240 STOP
6250 [illegible]                             Index into the hash table
6260 READ #1[illegible]                      Read the record
6270 IF [illegible] THEN 6290                Is it the right one?
6280 RETURN                                  Yes
6290 IF K5=0 THEN 6230                       No, check for end of the chain
6300 [illegible]                             Chain to next record
6310 GO TO 6260                              And read it
```

With the above information, tradeoffs can be made between linear and non-linear hashing for any number of records. To calculate the number of hash buckets for any size file as a function of the number of disc accesses, the following empirical equation is given:

$$\frac{S}{2*N} = B$$

where B is number of hash buckets
S is number of records
N is number of reads

For linear hashing, the following empirical equation seems to work:

$$\frac{2(1+N)S}{N} = A$$

where A is allocated size
S is number of records used
N is number of reads

Packing Integers

by Ted Webber
Laurie, Montgomerie & Pettit Pty. Ltd.
Sydney, Australia

The following program saves memory when you're storing integers consisting of six digits or less. The integers are packed into one real number, complete with signs, in the format:

C = ±A.nB where A and B are each 6 (or less) digit integers and n = 0, 1 or 2 to indicate the sign of B

The sign of B is stored as the first digit after the decimal point in C so that 0 = –, 1 = 0 and 2 = +. FNB encodes N1 into the integer part of N. FNC encodes N2 into the decimal part of N. FND decodes N2 and FNE decodes N1.

The first routine generates two arrays of 500 random integers, which are then merged into one array of 500 real numbers. Partial output is illustrated in Fig. 1.

```
100 INIT
110 DEF FNA(C)=ABS(C)-INT(ABS(C))+1.0E-8
120 DEF FNB(C)=SGN(C+0.1)*(ABS(C)+FNA(D))
130 DEF FNC(C)=SGN(D+0.1)*(ABS(C)*1.0E-7+0.1*(SGN(C)+1)+INT(ABS(D)))
140 DEF FND(C)=SGN(10*FNA(C)-1)*INT(0.5+1000000*FNA(10*FNA(C)))
150 DEF FNE(C)=INT(ABS(C))*SGN(C)
160 DIM N(200),N1(200),N2(200)
170 N=0
180 R=RND(-1)
190 Z=1000000
200 FOR I=1 TO 200
210 Z=-Z
220 N1(I)=INT(RND(1)*Z)
230 N2(I)=INT(RND(1)*Z)
240 NEXT I
250 FOR I=1 TO 200
260 D=N(I)
270 N(I)=FNB(N1(I))
280 D=N(I)
290 N(I)=FNC(N2(I))
300 NEXT I
310 PRINT "GGG"
320 DELETE N1,N2
325 DIM N1(200),N2(200)
330 FOR I=1 TO 200
340 N1(I)=FNE(N(I))
350 N2(I)=FND(N(I))
360 NEXT I
370 PRINT "GGG"
380 END
```

The second routine packs the integers as they're input, first N1 (statements 230-250) and then N2 (statements 260-280). Once the real number array is completed, either N1 or N2 may be changed without re-computing the other. Figure 2 shows random generation of N1 and N2 using the second routine, and the consequent output by changing N1 only through deletion of statement 260-280 and **running from line 210**. Figure 3 repeats this process, but changes N2 only.

```
N1
-148669        492183         -969948        643160
-840569        68575          -955443        148942
-534638        920940         -516749        402061
-887424        480469         -952989        985830
-69920         143861         -631351        317737
-464459        49393          -117675        403301
-661820        95754          -379642        353021
-796499        888369         -571236        824686
N2
-87777         409083         -475883        68715
-405666        97823          -204454        752745
-436177        57070          -29368         423703
-535559        462180         -626662        415676
-368811        572686         -15186         708173
-654777        960105         -338661        559363
-268461        710938         -948991        66773
-241950        217464         -60729         230988
N
-148669.000778 492183.2409    -969940.047588 643160.206072
-840569.040567 68575.2097823  -955443.020445 148942.275275
-534638.043618 920940.205707  -516749.002937 402061.24237
-887424.053556 480469.246218  -952989.062666 985830.241568
-69920.0368811 143861.257269  -631351.001519 317737.270817
-464459.065478 49393.2960105  -117675.033866 403301.255936
-661820.026846 95754.2710938  -379642.094899 353021.206677
-796499.024195 888369.221746  -571236.006073 824686.223099
DECODE N1
-148669        492183         -969948        643160
-840569        68575          -955443        148942
-534638        920940         -516749        402061
-887424        488469         -952989        985830
-69920         143861         -631351        317737
-464459        49393          -117675        403301
-661820        95754          -379642        353021
-796499        888369         -571236        824686
DECODE N2
-87777         409083         -475883        68715
-485666        97823          -204454        752745
-436177        57870          -29368         423783
-535559        462180         -626662        415676
-368811        572686         -15186         788173
-654777        960105         -338661        559363
-268461        710938         -948991        66773
-241950        217464         -68729         230988
```

Fig. 1. N1 and N2 are shown before being merged into one array (N). N illustrates the packing routine result. The decoded arrays depict the integrity of the routine.

```
100 INIT
110 DEF FNA(C)=ABS(C)-INT(ABS(C))+1.0E-8
120 DEF FNB(C)=SGN(C+0.1)*(ABS(C)+FNA(D))
130 DEF FNC(C)=SGN(D+0.1)*(ABS(C)*1.0E-7+0.1*(SGN(C)+1)+INT(ABS(D)))
140 DEF FND(C)=SGN(10*FNA(C)-1)*INT(0.5+1000000*FNA(10*FNA(C)))
150 DEF FNE(C)=INT(ABS(C))*SGN(C)
160 DIM N(5)
170 N=0
180 R=RND(-1)
190 Z=1000000
200 PRI #32: USI "5X2(9A)17A2(9A)":"N1","N2","N(I)","N1 DECODE"," N2"
210 FOR I=1 TO 5
220 Z=-Z
230 N1=INT(RND(1)*Z)
240 D=N(I)
250 N(I)=FNB(N1)
260 N2=INT(RND(1)*Z)
270 D=N(I)
280 N(I)=FNC(N2)
290 N3=FNE(N(I))
300 N4=FND(N(I))
310 PRINT #32: USING "2(9D)9D.7D2(9D)":N1,N2,N(I),N3,N4
320 NEXT I
330 END
```

```
      N1       N2          N(I)          N1 DECODE N2
  -636801  -663989  -636801.0663990  -636801   -663989
   390792    28762   390792.2028768   390792     28762
  -511026   -37583  -511026.0037580  -511026    -37583
   928230   548080   928230.2548080   928230    548080
  -177612  -470969  -177612.0470970  -177612   -470969

DEL260,280

RUN210
   543970  -470969   543970.0663990   543970   -663989
  -436839  -470969  -436839.2028760  -436839     28762
   711657  -470969   711657.0037580   711657    -37583
  -997196  -470969  -997196.2548080  -997196    548080
   698221  -470969   698221.0470970   698221   -470969
```

Fig. 2. The first run shows the integers as they've been input, packed and decoded. In the second run, N1 array has been changed, packed and decoded.

```
      N1       N2          N(I)          N1 DECODE N2
  -581295  -503007  -581295.0503010  -581295   -503007
   271969   806404   271969.2806400   271969    806404
  -942176  -849438  -942176.0849440  -942176   -849438
   617385   979883   617385.2979880   617385    979883
  -264712  -128915  -264712.0128920  -264712   -128915

DEL230,250
RUN210
  -264712   924541  -581295.2924540  -581295    924541
  -264712  -451348   271969.0451350   271969   -451348
  -264712   551745  -942176.2551750  -942176    551745
  -264712  -950616   617385.0950620   617385   -950616
  -264712   553110  -264712.2553110  -264712    553110
```

Fig. 3. Deleting statements 230 through 250 allows N2 only to be changed.

4907 Data File Storage

by Chuck Eng and Ed Mitchell
Tektronix, Inc.

The 4907 has four data file structures:

Binary random access
Binary sequential access
ASCII random access
ASCII sequential access

(This discussion does not include ASCII and binary program files.)

Both the random and sequential files have the same physical structure (256-byte blocks), but random access files are broken down into logical records. These logical records have pointers to the beginning of each record.

ASCII Random Access Files

ASCII random access files must be intialized. That is, each record in the file must be completely filled before the next record can be accessed. A quick routine to do this is shown on page F-2 of the "4907 File Manager Operator's Manual." Notice that one byte of each record is allocated for the CR; therefore, ASCII random records are initialized with a string of record length minus 1 byte (the carriage return is placed in the last byte).

Printing of ASCII random records must be done accurately. Record boundaries are not detected when ASCII records are printed to or input from. If there is more data than record space, the data will print over the next record. And, when a file is opened, and an input operation is executed that brings in more data items than contained in one record, the following record(s) will be input.

The 4907 dynamically extends files. As ASCII random files are extended, the records must be completely filled or subsequent records cannot be accessed. If you are letting the 4907 automatically extend your ASCII random file, be sure the data string length equals record length minus 1; pad the data string with blanks if necessary.

Binary Random Files

While Binary random access file records don't have to be initialized in the same manner as ASCII (completely filled), something must be written to a binary record before the subsequent record can be accessed. The above mentioned routine on page F-2 writes 3 blanks to each binary record.

Record boundaries are detected on binary writes and reads; therefore, if too much data is written or read for the record size, an error message is generated.

Storage Space

Data cannot be added to random **records**; updating a record requires rewriting the entire record. Also individual random **records** cannot be enlarged. So knowing data storage space requirements is important.

Both sequential and random access **files** are expanded automatically as space is required. Random expansion records are the same length as those originally created. Sequential file expansion is in units of 256 byte blocks.

The following table capsulizes the storage information:

4907 Data File Storage	Binary		ASCII	
	Random	Sequential	Random	Sequential
Record Initialization	No, but records filled in order	N/A	Yes, (record length minus 1)	N/A
Strings: Overhead:	4 bytes per string plus 1 byte per record	5 bytes per string	1 byte per record	1 byte per string
Per character:	1 byte	1 byte	1 byte	1 byte
Numeric Data: Overhead:	1 byte per record	1 byte per record	1 byte per record	1 byte per record
Per value:	9 bytes	9 bytes	1 byte	1 byte
End of Record Separator Detection	Yes	N/A	No	N/A
End of File Marks	Yes	Yes	Yes	Yes

Coordinate Transformation

by James R. Umdenstock
City of Tulsa, Oklahoma

The City of Tulsa is presently assembling a graphic data system that includes all major buildings, streets, water and sewer lines, address centroids (building centers) and the like. This will be a major data system for use by all city departments.

One of the problems associated with constructing such a data base is that separate and detached surveys and maps will meet harmoniously, without gaps or overlaps. Therefore, we have developed a routine which converts points on the digitizer to state plane coordinates.

A map may be placed anywhere on the Summagraphics digitizer table interfaced to the 4051 Graphic System. As the ground points are digitized, the 4051 quickly converts them to state plane coordinates and stores them on tape. The data is then transmitted to the host computer over the Option 1 Data Communications Interface.

Since state plane coordinates are an established and understood system, this is a very practical program for land mapping use.

```
10  REM PTYOU1 SPCS READER & TRANSFORMATION PROGRAM
11  REM ***
12  REM AUTHOR  J. JMDENSTOCK CITY OF TULSA 4 JAN 79
13  REM ***
14  REM ABSTRACT:
15  REM PROMPTS FOR ANY TWO KNOWN STATE PLANE COORDINATE
16  REM BENCHMARKS ON THE MAP, PERFORMS A TRANSFORMATION
17  REM BETWEEN THE MAP AND THE DIGITIZED COORDINATES, WRITES
18  REM OUT THE POINTS IN THE X,Y STATE PLANE SYSTEM, WITH
19  REM THE PEN UP/DOWN MODE FOR CALCOMP, PLUS THE SYMBOL
20  REM NUMBER.
21  REM ***
22  REM BEGIN MAIN PROGRAM
23  REM PROMPT FOR THE TWO BENCHMARKS
24  REM ***
34  PRINT "INPUT THE STATE PLANE X-VALUE OF THE LOWER LEFT MARK ";
35  INPUT X1
36  PRINT "NOW, THE STATE PLANE Y-VALUE OF THE LOWER LEFT MARK ";
37  INPUT Y1
38  PRINT "NOW, DIGITIZE THAT MARK(POINT MODE)"
39  INPUT @8:...
41  PRINT "INPUT STATE PLANE X-VALUE OF THE 2ND MARK ";
42  INPUT X2
43  PRINT "NOW, THE STATE PLANE Y-VALUE OF THE 2ND MARK ";
44  INPUT Y2
45  PRINT "NOW, DIGITIZE THAT MARK(POINT MODE)"
46  INPUT @8:...
[illegible]
```

The logical unit number for the digitizer is 8.

This printout is from the host computer, using the Option 1.

Data Transfer Through the RS-232

by Mark Mehall

Tektronix, Inc.

Chicago

Two small programs allow one user (User 1) to control tape-to-tape transfer of data to or from another 4051. This is especially helpful if the other 4051 user (User 2) is unfamiliar with data communication procedures.

Preparation

Each user connects their Option 1 to their modem, through the RS-232 cable supplied with the 4051. Both modems are set either to full duplex or to half duplex; the 4051s and the modems are then powered up.

```
100 REM PROGRAM TO SET 4051 OPT. 1 PARAMETERS
110 REM   FOR TAPE TRANSFER TO ANOTHER 4051
120 INIT
130 REM SET L$ TO CTRL-L (NEWPAGE)
140 L$=CHR(12)
150 REM SET J$ TO CTRL-J (LINEFEED)
160 J$=CHR(10)
170 REM SET G$ TO CTRL-G (BELL)
180 G$="GGGG"
190 REM SET FIRST FILE TO 0
200 F1=0
210 PRINT L$;"                4051 TO 4051 TAPE TRANSFER";J$;J$
220 PRINT "THE OTHER 4051 USER SHOULD USE THE DEFAULT PARAMETERS,"
230 PRINT "AND SHOULD BE IN TERMINAL MODE.";J$
240 REM INITIALIZE THE COMM. INTERFACE
250 CALL "CMINIT"
260 REM SET TSTRIN TO MATCH DEFAULT RSTRIN PARAMETERS
270 REM SET R$ TO CTRL-R (DC2)
280 R$=CHR(18)
290 REM SET T$ TO CTRL-T (DC4)
300 T$=CHR(20)
310 REM SET D$ TO CTRL-D (EOT)
320 D$=CHR(4)
330 CALL "TSTRIN",R$,T$,D$
340 REM SET TCRLF TO SUPPRESS CARRIAGE RETURNS
350 CALL "TCRLF",0,2,0
360 REM SET RCRLF TO PRINTED CONTROL CHARACTERS
370 CALL "RCRLF",1,2,1
380 REM INITIALIZE THE INTERFACE
390 PRINT @40,30:
400 REM INITIALIZE FILE NUMBER
410 F1=0
420 PRINT "HOW MANY FILES ARE BEING TRANSMITTED? ";G$;
430 INPUT F
440 FOR I=1 TO F
450 REM FIND THE TAPE FILE TO BE TRANSFERRED
460 PRINT "WHAT TAPE FILE IS TO BE TRANSMITTED? "
470 PRINT "THE LAST FILE WAS ";F1;" ";G$;
480 INPUT F1
490 FIND F1
500 PRINT
510 REM SEND INSTRUCTIONS TO THE OTHER USER
520 REM SET E$ TO CTRL-E (ESC)
530 E$=CHR(27)
540 PRINT @40:E$;L$;"                4051 TO 4051 TAPE TRANSFER"
550 PRINT @40:J$;J$;"Use Function Key 13 (shift 3) to find the file"
560 PRINT @40:J$;"to store the data.  Enter the file number. "
570 PRINT @40:J$;"Press RETURN. Then press Function Key 3.",G$
580 PRINT @40:J$;J$;"The information on your screen will overprint,"
590 PRINT @40:J$;"that's normal.";J$;J$
600 REM WAIT FOR A TRANSMISSION FROM THE OTHER 4051 (SNDSTG)
610 INPUT @40:A$
620 REM DO THE TRANSFER
630 CALL "DTSEND"
640 PRINT "FILE TRANSMITTEDGG."
650 PRINT @40:J$;J$;J$;"File received. ";G$;J$
660 NEXT I
670 PRINT @40:J$;"Final file received.  ";
680 PRINT @40:J$;"Pick up your headset. ",G$
690 END
```

Fig. 1. Routine to control transfer to another 4051 over the Option 1.

```
100 REM PROGRAM TO SET 4051 OPT. 1 PARAMETERS
110 REM   FOR TAPE TRANSFER FROM ANOTHER 4051
120 REM SET L$ TO CTRL-L (NEWPAGE)
130 L$=CHR(12)
140 REM SET J$ TO CTRL-J (LINEFEED)
150 J$=CHR(10)
160 REM SET G$ TO CTRL-G (BELL)
170 G$=CHR(7)
180 REM SET FIRST FILE TO 0
190 F1=0
200 PRINT L$;"                4051 TO 4051 TAPE TRANSFER";J$;J$
210 PRINT "THE OTHER 4051 USER SHOULD USE THE DEFAULT PARAMETERS,"
220 PRINT "AND SHOULD BE IN TERMINAL MODE.";J$
230 REM INITIALIZE THE COMM. INTERFACE
240 CALL "CMINIT"
250 REM SET RSTRIN TO MATCH DEFAULT TSTRIN PARAMETERS
260 REM SET N$ TO CTRL-@ (NUL)
270 N$=CHR(0)
280 REM SET S$ TO CTRL-S (DC3)
290 S$=CHR(19)
300 REM SET D$ TO CTRL-D (EOT)
310 D$=CHR(4)
320 CALL "RSTRIN",N$,S$,D$
330 REM SET EOLCHR TO SEND NOTHING AS SNDSTRG
340 CALL "EOLCHR",13,"",0
350 REM SET RCRLF TO PRINTED CONTROL CHARACTERS (FOR LINE FEEDS)
360 CALL "RCRLF",1,2,1
370 REM SET FRCR (FORCE CARRIAGE RETURN) TO 0 (NO)
380 CALL "DELAYS",0,0,0
390 REM INITIALIZE THE INTERFACE
400 PRINT @40,30:
410 PRINT "HOW MANY FILES ARE BEING RECEIVED? ";G$;
420 INPUT F
430 FOR I=1 TO F
440 REM FIND THE TAPE FILE TO STORE IT
450 PRINT L$,"WHAT TAPE FILE IS TO RECEIVE DATA? "
460 PRINT "THE LAST FILE WAS ";F1;" ";G$;
470 INPUT F1
480 FIND F1
490 PRINT
500 REM SEND INSTRUCTIONS TO THE OTHER USER
510 REM SET E$ TO CTRL-E (ESC)
520 E$=CHR(27)
530 PRINT @40:E$;L$;"                4051 TO 4051 TAPE TRANSFER"
540 PRINT @40:J$;J$;"Use Function Key 13 (shift 3) to find the file"
550 PRINT @40:J$;"to be transferred. Enter the file number. "
560 PRINT @40:J$;"Press RETURN. Then press Function Key 4."
570 PRINT @40:J$;"The last file transferred was ";F-1;G$
580 PRINT @40:J$;J$;"The information on your screen will overprint,"
590 PRINT @40:J$;"that's normal.";J$;J$
600 REM DO THE TRANSFER
610 CALL "DTRECV"
620 PRINT "FILE RECEIVEDGG."
630 PRINT @40:J$;J$;J$;"File transmitted. ";G$;J$
640 NEXT I
650 PRINT @40:J$;J$;J$;"Final file transmitted.  ";
660 PRINT @40:J$;"Pick up your headset. ",G$
670 END
```

Fig. 2. Routine to control transfer from another 4051 over the Option 1.

The user who is receiving the data marks a tape for the number and size of files required for the incoming data. User 1 loads the applicable program below (sending or receiving) into 4051 memory. User 2 types in CALL "CMINIT" on the second 4051 and presses the RETURN key, then types in CALL "TERMIN" followed by a RETURN.

Data Transfer

Establish telephone contact (either one can dial the other). One user sets their modem to answer (ANS) mode and lays their headset into the modem cradle (note where the cord end is placed). The other user sets their modem to originate (ORIG) and listens for the frequency signal; then sets their headset into the modem cradle. When contact is established, the carrier light on the modem will come on; the 4051 will stop flashing its lights.

Now the program in User 1 4051 memory is run. Users each respond to the prompts on their 4051 screen, and the data transfer is carried out. When transfer is completed, load one of the files into the receiving 4051 to verify.

A related and informative article is contained in TEKniques Vol. 1 No. 4, "4051s Talk To Each Other."

Changing 4051 Parameters to Read Different Tape Formats

by Aaron Eisenbach
Tektronix, Inc.
Baltimore, MD

The 4051 can easily adapt its ASCII input format requirements to the ASCII output formats used by different peripheral devices. A customer recently logged data on the Tektronix 4923, and then read it into the 4051 using the following routine.

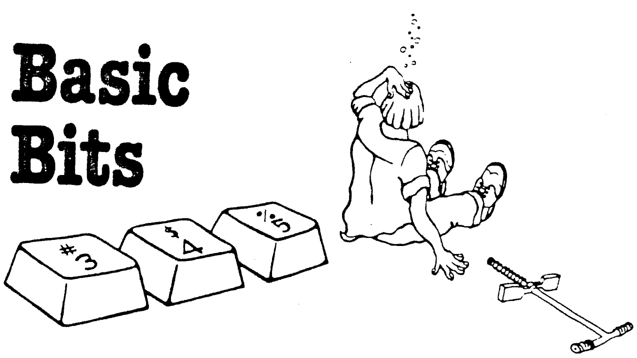
Statement 120 changes the status byte for the internal magnetic tape unit so it can read the 128 byte length, no checksum, no header format of the 4923. The next steps determine the record separator, end of file mark, and incoming character(s) to be deleted. After the alternate parameters are input, statement 230 instructs the microprocessor to use these when the "%" sign is used in place of the "@" sign on INPUT, OLD or APPEND commands.

The rest of the routine reads the tape and resets the status bytes.

```
100 PRINT "LG*  INSERT DATA TAPE AND PRESS [RETURN]";
110 INPUT T$
120 PRINT @33,0:1,1,1
130 PRINT "JJG*  USE THE DECIMAL EQUIVALENTS OF THE"
140 PRINT "    ASCII CHARACTERS FOR THE FOLLOWING INPUTS"
150 PRINT "JJJG*  RECORD SEPARATOR ? :  ";
160 INPUT A1
170 PRINT "JG*  END OF FILE MARK ? : ";
180 INPUT A2
190 PRINT "JG*  CHARACTER TO BE DELETED ? (ENTER 128 IF NONE) : ";
200 INPUT A3
210 PRINT "JG*  DATA FILE # ? : ";
220 INPUT F
230 PRINT @37,0:A1,A2,A3
240 PAGE
250 ON EOF (0) THEN 300
260 FIND F
270 INPUT %33:A$
280 PRINT A$
290 GO TO 270
300 PRINT @37,26:0
310 PRINT @33,0:0,0,0
320 PRINT "JJGDONE"
```

Caution should be used when specifying an alternate record separator. If an ASCII data string contains both Carriage Return characters and the alternate record separator character, logical records could be lost. To avoid this, specify the character to be deleted (statement 190) as "13." This will delete all incoming Carriage Returns but will read all logical records.

For a complete discussion on magnetic tape status parameters, see the "4051 Graphic System Reference Manual," pages 2-19 to 2-21. Check pages 2-25 to 2-30 for detailed information on processor status parameters.



Program Byte Counter

by Leo J. LaFrance
New Mexico State University
Las Cruces, NM

This routine quickly determines the minimum space required by an ASCII program file. It analyzes a file already on tape and, if requested, will re-mark the file and re-store the program. Note that the file should be the last one on the tape.

```
LIS
100 INIT
110 C$=CHR(13)
120 DIM A$(MEMORY-300)
130 A$=""
140 ON EOF (0) THEN 230
150 PRINT "LInsert tape and input number of file "
160 PRINT "to be analyzed. ";
170 INPUT F
180 FIND F
190 INPUT @33:R$
200 R$=R$&C$
210 A$=REP(R$,1+LEN(A$),0)
220 GO TO 190
230 PRINT "JJASCII file requires ";LEN(A$);" bytes."
240 PRINT "Minimum is 768 bytes."
250 PRINT "If you want the file re-sized, enter "
260 PRINT "'YES.'  CAUTION:  YOUR TAPE WILL BE ";
270 PRINT "RE-MARKED. ";
280 INPUT G$
290 IF G$<>"YES" THEN 340
300 FIND F
310 MARK 1,LEN(A$)+1
320 FIND F
330 PRINT @33:A$;
340 END
```

4051 Applications Library Program Abstracts

Order

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

Domestic U.S. Prices:

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

Contribute

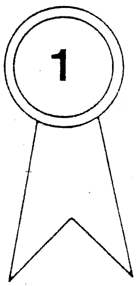
Contribute one program to the Library and receive three in exchange. Send in the membership card from your 4051 Graphic System Reference Manual to get the details. Or call us (503) 682-3411, ext. 2618.

Forms

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

Outside U.S.

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



ABSTRACT NUMBER: 51/00-0904/0

Title: **Business & Accounting Formulas**
Author: Dr. P.C. Holman
University of Wisconsin
Stevens Point, WI
Memory Requirement: 32K
Statements: 7698
Files: 19 ASCII Program

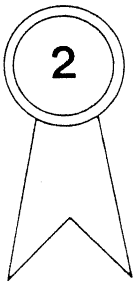
A turorial program containing formulas, statistics and tables in the following areas:

1. Simple Interest Formulas: A set of eight formulas used to compute simple interest.
2. Compound Amount Formulas: A set of programs determining the dollar amount that an account or loan is worth now, or would be worth at some future time.
3. Annuities: A set of programs determining the value of an annuity. They handle any kind of annuity, as well as determining what one would have to pay if one set up an annuity.
4. Statistics; Averages and Variation Formulas: A set of programs for solving statistics most commonly used in business.
5. Statictics; Sampling: A set of programs determining statistical sampling parameters.
6. Statistics: Correlation: A set of programs determining the statistical correlation between sets of data.
7. Statistics: Index Number Formulas: A set of programs determining the index values for several sets of data.
8. Statistics: Time Series Formulas: A set of programs determining secular trends, seasonal changes, and cycles. Used to determine what quantity a firm should produce during given periods of time.
9. Inventory Formulas: A set of programs used for inventory assessment.
10. Depreciation Formulas: A set of programs allowing calculation of depreciation.
11. Finance Section: A set of formulas used for financial analysis.
12. Price Level Adjustments: A set of programs used to convert long term liabilities (i.e. depreciation, A P) into current dollar values.
13. Marketing Formulas: A set of programs used to calculate the selling price a retailer or wholesaler should use to obtain the profit desired.
14. Cost and Production Formulas: A set of programs used to determine the cost and production relationships of various products and payroll billings.
15. Ratio Analysis Formulas: A set of programs presenting basic accounting ratios as well as time periods

for such things as production and inventory turnover, and collection periods for various accounts.

16. Single Entry Formulas: A set of programs used to determine income statement entries for sales, purchases, expenses, and inventories.

17. Miscellaneous Formulas: A set of programs that are useful in business, but that do not belong in the major groupings previously described.



ABSTRACT NUMBER: 51/00-0905/0

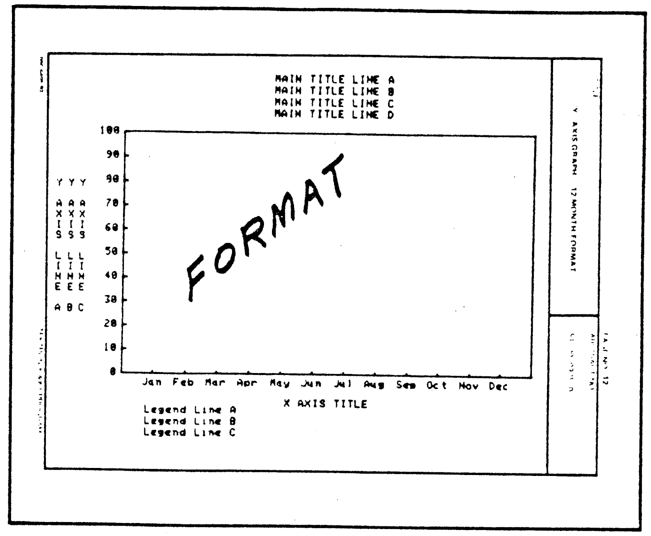
Title: **"Y" Axis Graph—12 Month Format**
Author: Robert Pilkington
AT&T Long Lines
Bedminster, NJ
Memory Requirement: 24K
Peripherals: Optional—4610/4631 Hard Copy Unit
Statements: 568
Files: 1 Program
4 Data

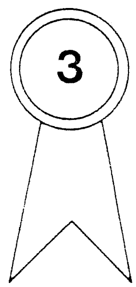
A solid line curve, a dashed line curve, shaded bars, or any combination of the three can be created easily on a graph with titles and labels by using this monthly scale adjustable "Y" axis graphing program.

All titles and labels are automatically spaced and centered around the axis. Up to four main title lines can be entered with one "X" axis title, a three position "Y" axis title, and a legend label for each plot curve. The user can select bar shadings as well (empty bars, solid fill or cross hatch). A parameter and data listing can be accessed before or after the graph is diplayed.

Data for all three plot modes can be added, deleted or changed. All data and parameters can be saved or retrieved from pre-marked binary files on the same 4051 data cartridge.

Limitations: "Y" axis tic labels up to six characters using whole numbers and seven characters using decimal numbers including decimal point and dollar sign. The scales must be positive numbers. The size and position of the graph on the screen is constant and stationary.





ABSTRACT NUMBER: 51/00-0907/0

Title: **Bargraph II**
Author: Mallory M. Green
U.S. Dept. of H.U.D.
Washington, D.C.
Memory Requirement: 32K
Peripherals: 4662 Plotter
Optional—4907 File Manager
Statements: 1493
Files: 4 ASCII Program
4 Binary Data

A program designed to generate professional horizontal or vertical formatted bar graphs in a simple interactive manner.

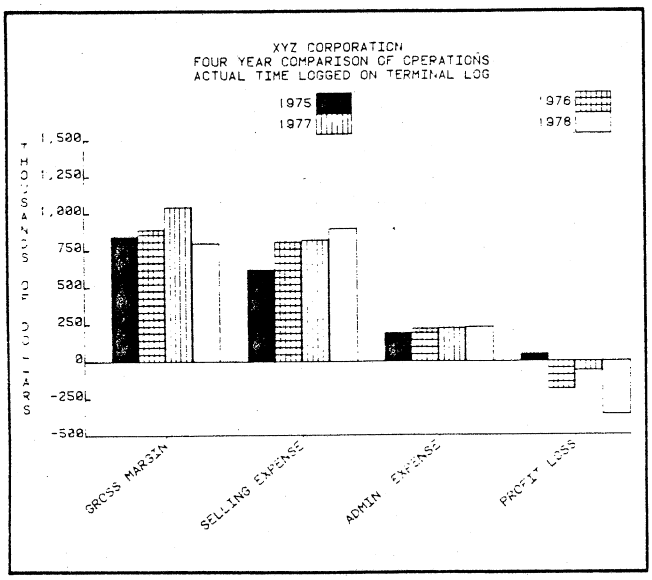
Features of the program are:

1. Horizontal and vertical bargraph formats.
2. User-prompted keyboard entry of all titles, labels and data.
3. Interactive modification of titles, labels, or data via user keys.
4. Storage of complete chart descriptions on either tape or disc files for future modification and plotting.
5. Output drawn on either the screen or 4662 Plotter.

6. 4662 Plotter outputs in one or more colors.

7. Automatic layout of centered and proportional charts.

8. TIMESERIES II and BARGRAPH II are fully compatible.



4

ABSTRACT NUMBER: 51/00-0906/0

Title: **Timeseries II**
Author: Mallory M. Green
U.S. Dept. of H.U.D.
Washington, D.C.
Memory Requirement: 32K
Peripherals: 4952 Joystick or 4662 Plotter
Optional—4907 File Manager
Statements: 1157
Files: 3 ASCII Program
4 Binary Data

A program designed to generate professional timeseries charts in a simple interactive manner.

Features of the program are:

1. Can draw up to 6 lines for up to 35 time periods.

2. Time periods such as seconds, minutes, hours, days, weeks, months or years can be used.

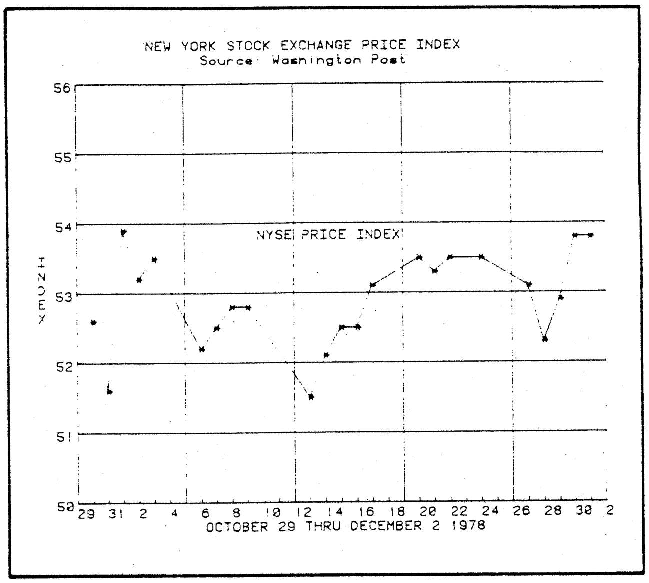
3. User-prompted keyboard entry of titles, labels, and data.

4. Allows interactive modification of titles, lables, and data via user keys.

5. Chart descriptions can be stored on either tape or disc.

6. Utilizes all 20 user-definable keys for maximum flexibility.

7. TIMESERIES II and BARGRAPH II are fully compatible.



5

ABSTRACT NUMBER: 51/00-0908/0

Title: **Regular Plot**
Author: R.J. Reimann
Memory Requirement: 32K
Peripherals: 4662 Plotter
Optional—4051R05
Statements: 862
Files: 1 Binary Program
1 Binary Data

A general graphics program developed for industrial market analysis. It provides line and bar plots with regularly spaced X-axis entries. Features include:

1. Auto-scaling of the Y-axis.

2. Line addition/deletion capability.

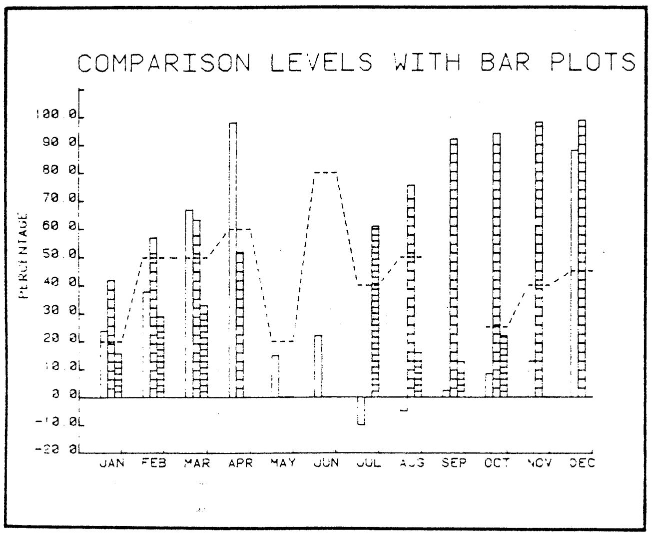
3. Labels and data correction.

4. Allowance for negative data.

5. Aligned data tables.

6. Data storage/recall.

7. Linear regression.



ABSTRACT NUMBER 51/00-9531/0

Title: **Slidemaker II**
Author: John R. Carter
Tektronix, Santa Clara Annex
Memory Requirement: 32K
Peripherals: 4662 Plotter
Optional—4907 File Manager
Statements: 679
Files: 1 ASCII Program
1 ASCII Text

Slidemaker II offers a highly versatile tool for creating professional and sophisticated presentation aids.

The main features are:

1. Standard type sizes selected with a single variable.

2. Tab selections that operate like a typewriter.

3. Variable type sizes and color changes possible on the same line of type, including choice of bold or normal type on the same line.

4. Fast line centering.

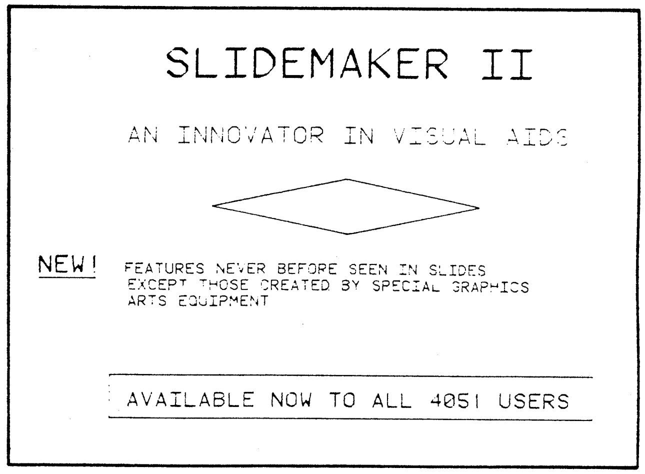
5. Input a whole page of text in one operation, with only one pen setting at the beginning.

6. Save and retrieve plots on either cartridge tape or disc in the same program.

7. Edit text at any time.

8. Alter retrieved plot and save new plot in same or separate file.

9. Graphics symbols available: Line, Box, or Diamond. Draw anywhere on plotter surface.



TEKTRONIX Announces 4050 Catalog: Vendor Supplied Software and Services

TEKTRONIX will be producing a catalog of software and programming services for the 4051 Graphic Computing System available from companies and consultants not associated with TEKTRONIX. The catalog will be sent directly to current 4051 users and to our worldwide sales organization.

Software products will be listed in the catalog by application area. Programming services for the 4051 will be listed by geographic location. Customers will contact the advertiser directly for additional information. *TEKTRONIX does not endorse or warrant any products or services listed.*

Listings are restricted to Software Packages written exclusively in 4051 BASIC.

For a direct avenue to the market for your 4051 software products and programming services, complete the enclosed form and return it to: Catalog Listing, Group 452, Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077. More forms may be acquired by writing to this address.

Tektronix
COMMITTED TO EXCELLENCE

TEKTRONIX, INC.
Information Display Group
Applications Library
Group 451
P.O. Box 500
Beaverton, Oregon 97005

STELLA MCMORRAN
60-307
IF YOU HAVE MOVED
PLEASE CALL 2618

MAR 26 1979

ADDRESS CORRECTION REQUESTED

4051 Applications Libraries

Africa, Europe, Middle East

4051 Applications Library
Tektronix International, Inc.
European Marketing Centre
"Bavinckstaete"
Prof. Bavincklaan 5
1183 at Amstelveen, The Netherlands

Australia

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

Canada

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

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

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

Japan

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

United States

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