Tekniques

4051 Used in Embryonic Nerve Development Study

Tektronix®
COMMITTED TO EXCELLENCE
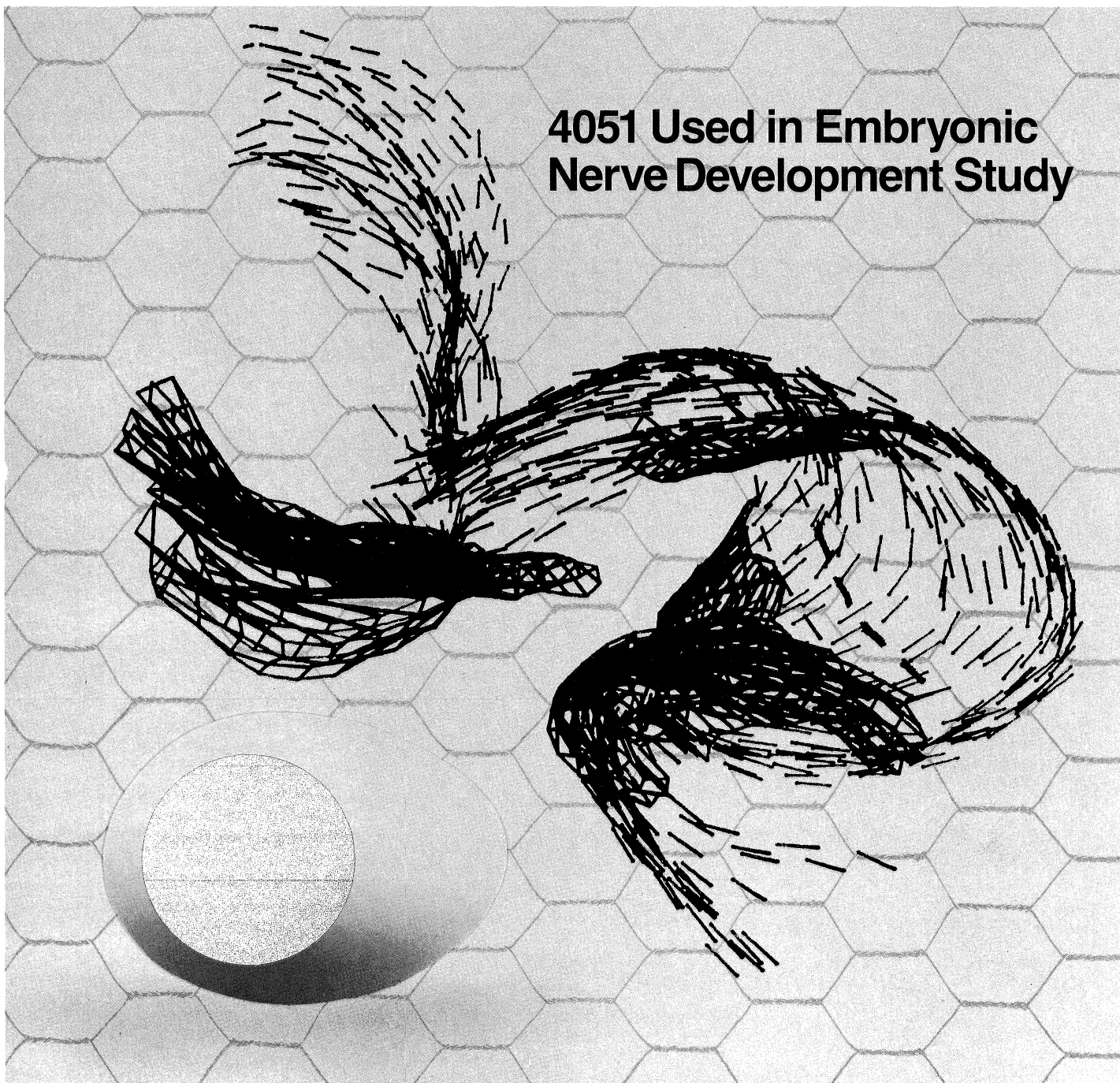
Tekniques

In This Issue

Nerve Development:

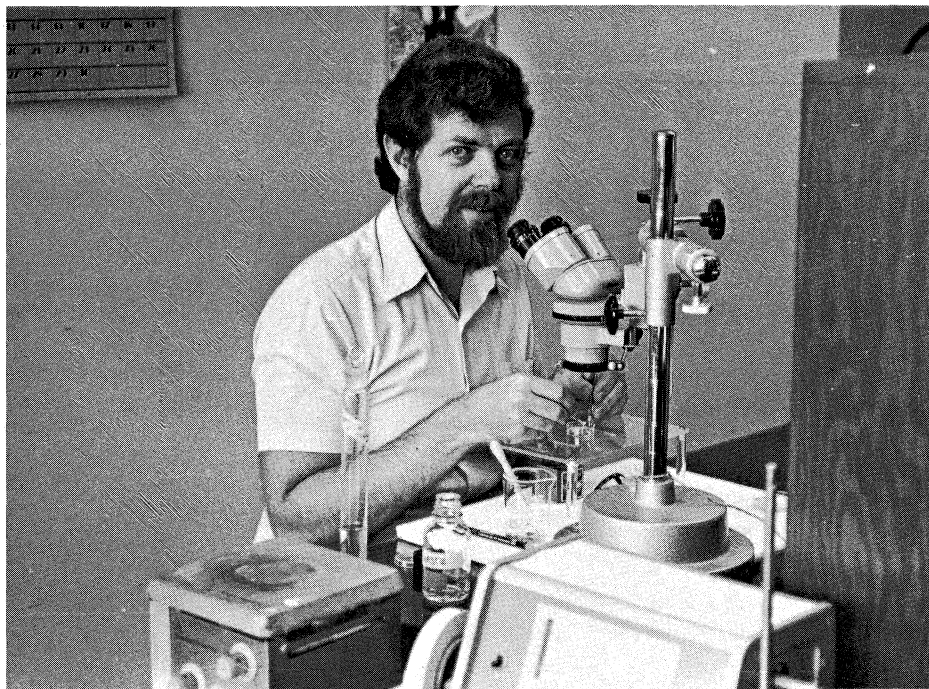# Digitization and Three-Dimensional Reconstruction from Serial Sections

Fig. 1. Selecting fertile frog's eggs from a newly-laid batch at McGill University in Montreal, where the author is on a special studies program. The embryos are used to obtain nerve and muscle cells for tissue culture studies of development of connections between them.

by David F.Davey
Neurobiology Laboratory
University of Sydney
N.S.W., Australia

As a part of the Neurobiology Laboratory's continuing investigation of embryonic nerve development, Max Bennett, Kerrie Uebel and I began a detailed study of the growth of nerves into the wing of the chick embryo. Our initial interest centered on the establishment of the first two nerve trunks into the limb, a process which occurs between about 3 1/2 days of embryonic age when the limb is simply a tiny bulge on the embryo, and about 5 days when it has grown to a length of about 2 mm. (The chick hatches at 21 days.)

The size was small enough to require microscopy, yet large enough to require serial sectioning to analyze a whole wing. We quickly found that constructing a view of the rapidly developing limb from such sections, often exceeding a thousand in number, was a very difficult task. We dreamed of computer techniques, but were discouraged by attempts of a colleague to make use of some 3-D architectural programs on a large computer. These methods proved too compromised towards rectilinear structures; finding out that much proved expensive.

Just at the right time, another colleague, Professor J.A. Young, obtained a 4051 Graphics Computer and 4662 Interactive digital plotter. We are indebted to him for allowing me to investigate the usefulness of these machines in our study. It is a credit to

the simplicity of the Tektronix BASIC graphics commands, that with only little Fortran experience, the 4051 and 4662 manuals, and a session with the graphics tutorial program, I had a crude digitization and display program working within about 20 hours of access to the machines.

The general approach I adopted was to digitize outlines of the chick wing, and structures within it that we were interested in, by placing light micrographs directly on the 4662 and tracing these structures using the cross-hair sight and joystick control. The "CALL" button was used to signal the start and end of lines, and the coordinates were obtained within a loop containing a GIN command. The X-Y data, together with information on distance separating adjacent sections enabled three dimensional projections of the traced information to be displayed.

Several problems associated with the data acquisition quickly became evident, and these turned out to have a common solution. First, it was possible to digitize far too many points per unit length of line traced; too many both in terms of the resolution we needed and in terms of the data storage and handling. Second, the number of points per unit length of line traced varied with the speed of tracing. Third, with even a fair amount of practice, it was difficult to avoid straying from the intended line using the joystick, especially if a large amount of material was being processed.

The solution to these problems lay in the following strategy. First, I predetermined how many points were to be permanently stored for each outline to be traced (small structures such as the nerve trunks did not require as many points as the outline of the wing itself), and from the data acquired from the digitization run with the joystick) calculated the positions of these points such that these calculated points would be equidistant. Second the position of these points were verified by the joystick operator so that when the initial tracing inevitably strayed from the desired line, the operator could reposition the points using the joystick.

In the meantime we found that comparing structure outlines from one section to the next was difficult working with micrographs themselves, and began to draw our outlines onto overhead projector transparent sheets. Groups of these can be easily compared, and if water soluble pens are used for drawing outlines, alterations are easy and quick. These transparencies are held well by the 4662 electrostatic hold-down, so we began to digitize from these transparencies.
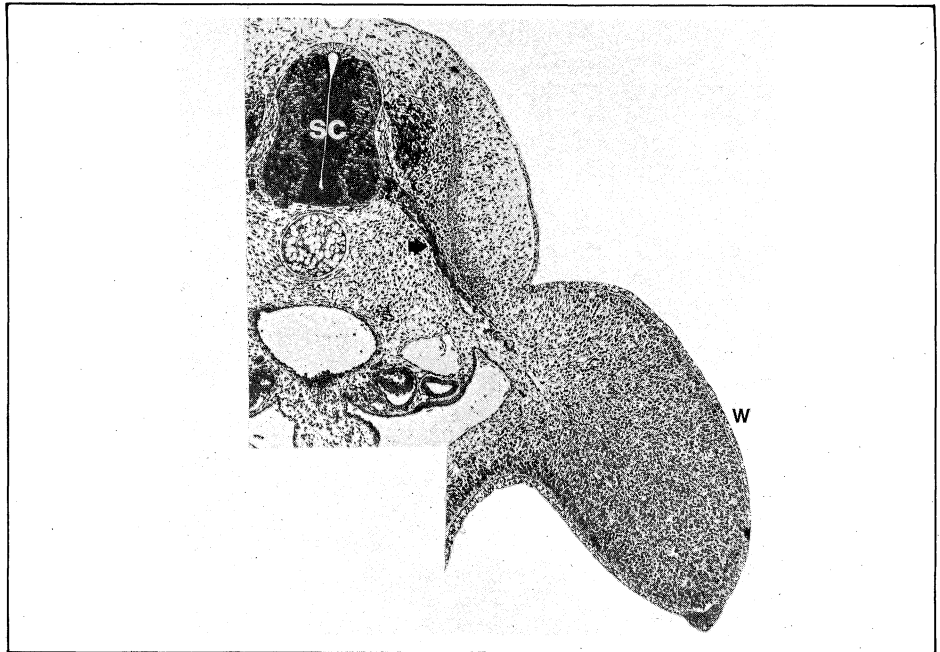
Fig. 2. Light micrograph of a section through trunk and wing of a chick embryo after 3 1/2 days incubation. One nerve trunk (arrow) can be seen growing from the spinal cord (SC) toward the wing-bud (W), which is about 1/2 mm long. The section is only about 1 μm thick; 500 to 1000 such sections are required to follow all the nerves into the wing. They are obtained by embedding a chemically preserved embryo in an epoxy resin which is cut with a glass knife using a microtome.
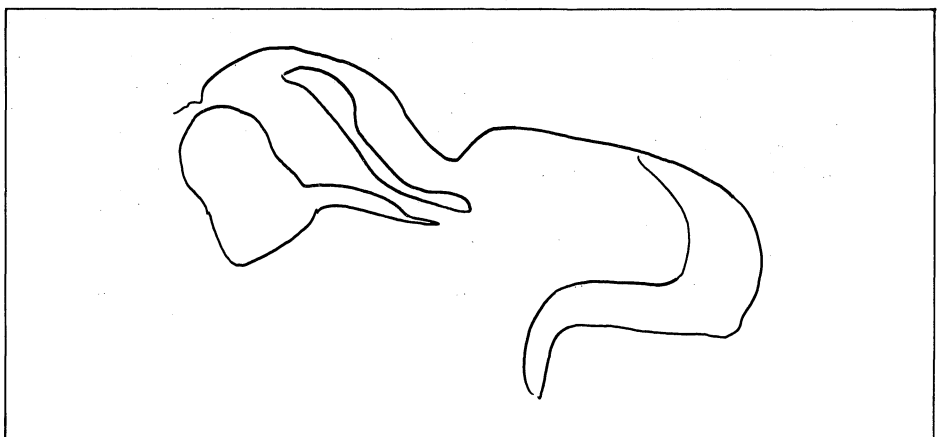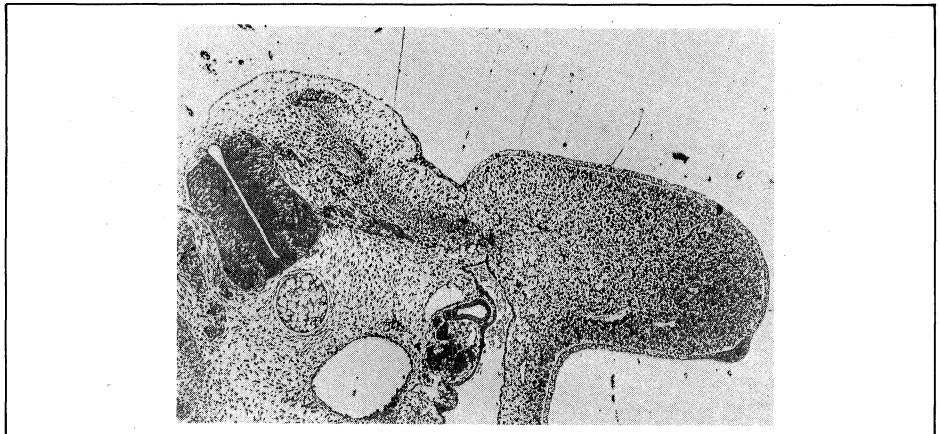
Fig. 3a & 3b. Key structures are outlined on overhead projector transparencies placed over the micrographs. These are handy for comparing adjacent sections. The original sections, which are stained with two different coloured dyes, are referrred to when making these outlines. In addition, thinner sections cut periodically from the same specimen are examined in the electron-microscope to positively identify small structures. The denser area outlined in the wing bud is the first sign of muscle development.

Digitization Procedures

Data input begins by initializing file names, number of structures, points per structure and number of sections to be digitized. The program then requests the operator to trace each line in turn for each section. For each such line the method is:

1. In response to a request on the 4051 screen, and the plotter prompt light being illuminated, the operator moves the plotter cursor to the start of the line and signals readiness to begin tracing by pressing the plotter's "CALL" button. The program puts the pen down and the operator then traces the line with reasonable (but not extreme) accuracy, using the joystick control. The end of the line is signalled by pressing "CALL" again. During the trace, X,Y coordinates of the pen position are repetitively determined by the program executing a GIN request to the plotter and transferring the result to a disk scratch file. The length of line traced is summated. At this stage the path of the cursor is digitally approximated by the series of coordinates in the disk file. The length of the series of straight lines joining these points is divided into n − 1 equal length segments (where n is the predetermined number of points to be used finally to represent the line), and the coordinates of the points at the end of each segment and lying on the approximation formed by the series of lines is calculated.

2. Under program control, the plotter cursor is moved to each of these points in turn, and the operator checks that the calculated point lies appropriately on the original desired outline. It may not, either due to tracing inaccuracy during step 1 or because sharp contours have been represented by too few points in step 1. If necessary the operator repositions the cursor with the joystick. The operator accepts the point by pressing "CALL"; the cursor position is then saved as the permanent coordinate for that point.

The overall result is the storage of the predetermined number of points lying on the original outline with an accuracy under the control of the operator. The points will be separated by equal distances unless gross repositioning has occurred during step 2.

These steps are repeated for each structure traced from the section, and finally the program requests the distance separating the section from the previous one, then the data for the current section are written to disk.

Although the scheme was devised as a means of coping with the difficulty of tracing with the joystick, as well as with the data handling problems mentioned above, it has an additional payoff. Since a structure

Fig. 4a. Stage 22. Cut 30.
1-Ectoderm
5,9-Spinal Cord
7-Nerve

Fig. 4b. Cut 31.
1-Ectoderm
5,9-Spinal Cord
6,8-Ventral Root
7-Spinal Nerve
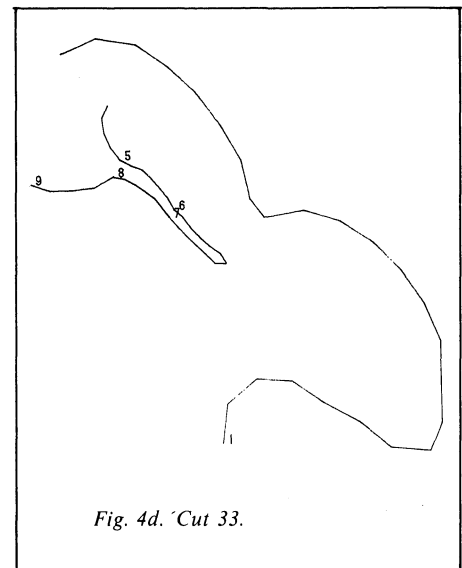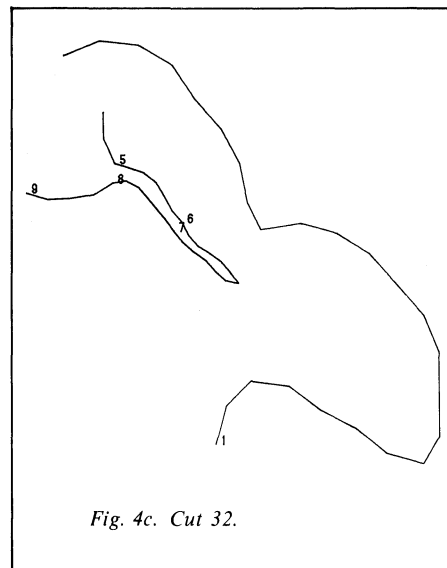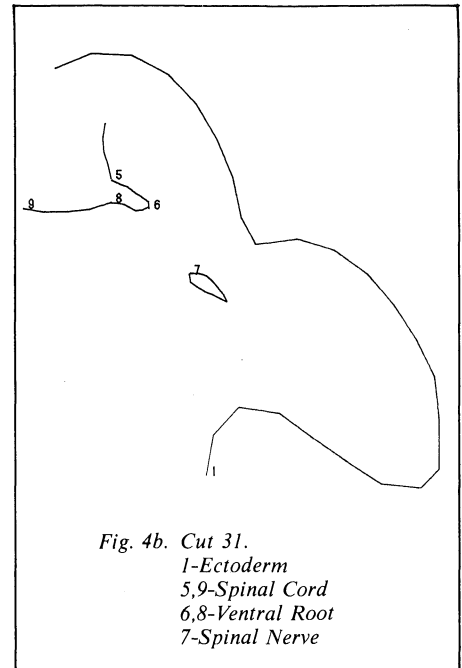
Fig. 4c. Cut 32.

Fig. 4d. Cut 33.
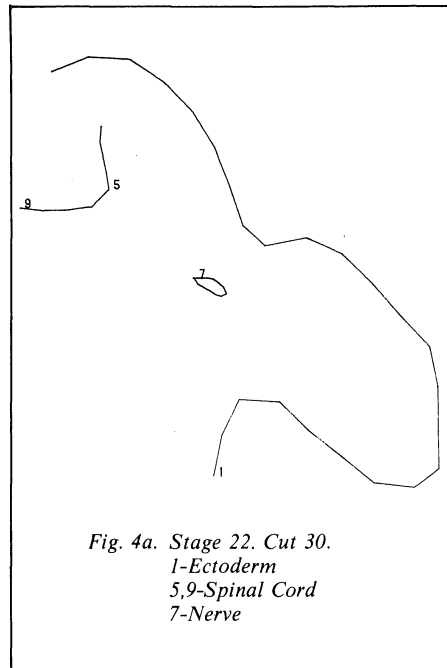
Fig. 4. Examples of digitized outlines such as shown in Fig. 3. A-C from 4 adjacent sections from the digital file. Since only about every fifteenth section is digitized, these represent about 60 µm of the limb. Note that one of the nerves emanating from the spinal cord in C (see arrow Fig. 2) is not present in A.

is represented by an equal number of points in each digital section, corresponding points in adjacent sections can be connected in drawing the three dimensional projections of the data without any further calculations being necessary. Such connecting lines are very useful in representing continuous structures, such as nerve trunks, in the reconstructions. Clearly just having an equal number of points in each section does not guarantee a reasonable point-to-point correspondence without ensuring a relation between the start and end of each outline from one section to the next. This must be borne in mind when drawing the outline from the micrographs in the first instance.

Principle of Data Output

The aim of the programs used to handle the digitized structural information is to depict three-dimensional representations of the structures traced on the transparencies and to represent these data in two dimensional projections. Furthermore these representations can be examined from any viewpoint, taking perspective into account. To this end the programs enable rotation of the data in the X-Y plane about the Z-axis and tilting of the data in the Y-Z plane about the X-axis. Drawings of the data are built up from many drawings of single sections (apart from joining corresponding points in adja-

cent sections) and the technique may be described on the basis of a single section.

The simplest representation of a section is without rotation or tilting. If the Z co-ordinate of the section is zero, the representation of the section is identical to the original tracing; if greater than zero it is "similar" (in the Euclidean geometrical sense) to the original tracing but slightly larger to account for it being closer to the viewpoint (i.e., the distance between the section and the viewpoint, used to determine perspective would be different).

Rotation of the projections is achieved simply by a transposition of the XY data, recalculating the values of the co-ordinates using the origin as the center of rotation. Tilting of the projections involved certain approximations, related to perspective, considered expedient in terms of the time required to generate drawings. If the Z coordinate of a section is zero, tilting its projection involves a rescaling of the Y axis to reduce the magnitude of each Y coordinate to $Y\cos\Theta$ where Θ is the angle of tilt. Perspective is not accounted for, leading to a slight distortion in the Y direction. Perspective calculations are also not applied to the X co-ordinates, again leading to a slight distortion. The point of these approximations is that they enable scaling simply through a "SCALE" command. No trigometric calculations in BASIC are required.

If the Z co-ordinate of a section is greater than zero, calculation of new Y scaling follows recalculation of the XY data to account for perspective, as described above for a case where no tilting is involved. The fact that the Z co-ordinate differs from zero means that the distance to the viewpoint also differs from that if no tilting is introduced, and this is taken into account in the calculation since this proved to be of considerable importance in providing three-dimensional visual cues. These calculations can be applied simply with a "MOVE" and "SCALE" sequence.

Production of Full Reconstructions

Some of our reconstructions have been fairly complex, both in terms of the numbers of structures being represented and the fact that we have structures within structures within structures, e.g. nerves within muscles within the wing. Because the method does not involve any routines to remove hidden lines, the final representation of the reconstructions presents some problems. At the same time we have not wanted to remove hidden lines, since we have been concerned
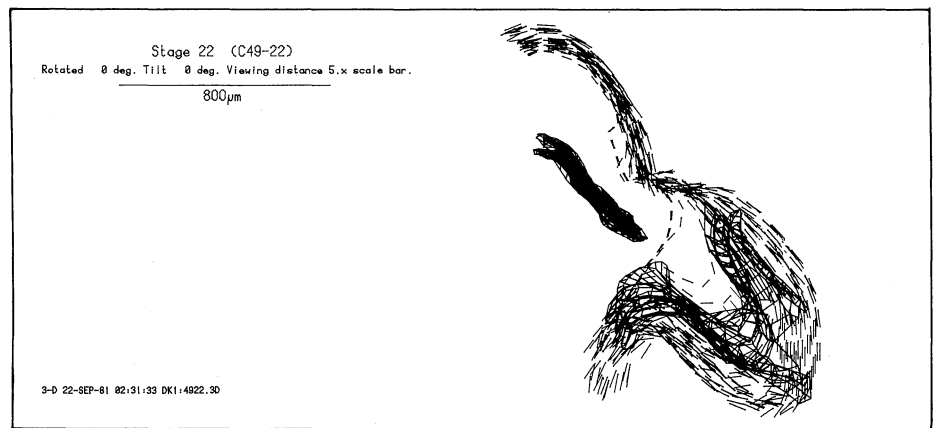
Fig. 5a. Reconstruction of limb sections such as shown in the micrograph of Fig. 2 and the digital outlines of Fig. 4. In this case no rotation or tilt has been used.
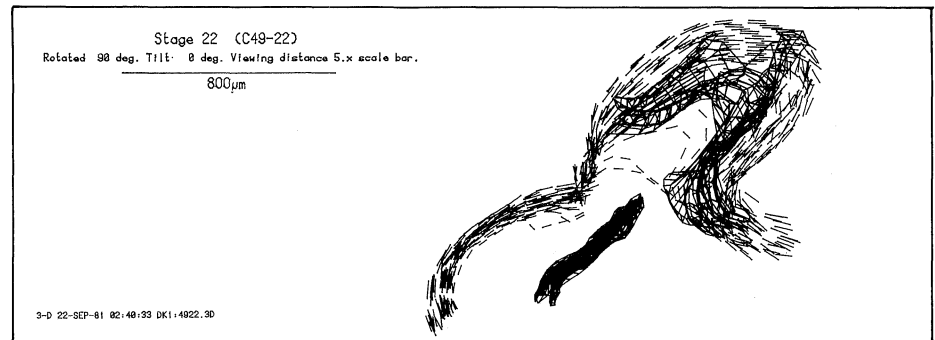
Fig. 5b. Rotation alone does not change the information content.
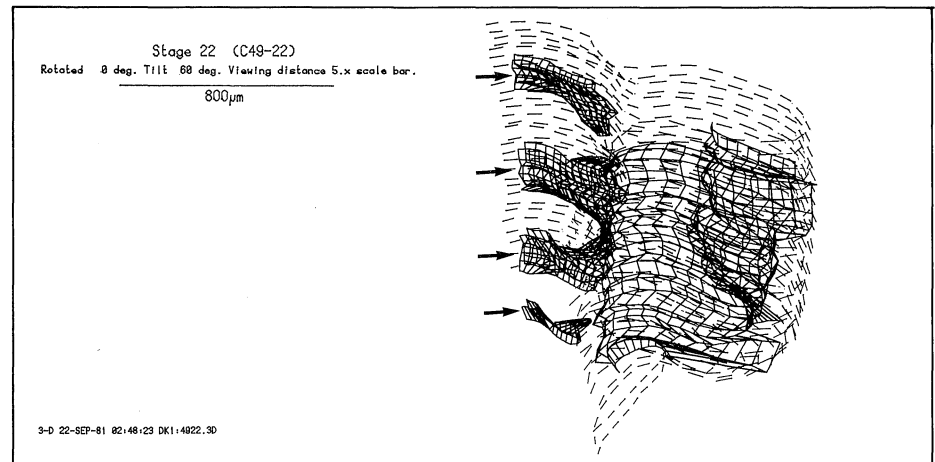
Fig. 5c. Tilt alone reveals four nerves (arrows) entering the limb, but their relationship is confused.
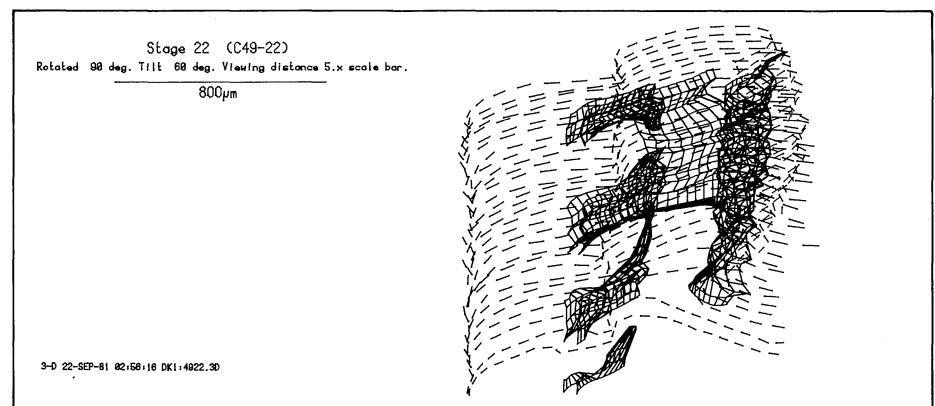
Fig. 5d. Both rotation as in B and tilt as in C enables the convergence of the four nerves towards the muscle density (see Fig. 3) to be seen. This information is exceedingly difficult to convey using only serial micrographs such as shown in Fig. 2. This view looks into the limb bud from inside the chick.

with the way nerves grow inside the other structures. The use of different colors aids in our own analysis, but the cost of color printing prohibits this as a solution to producing figures for publication. Two additional developments have helped with this problem. First I have provided for drawing the reconstructed structures in a variety of ways based on combinations of dashed lines, solid lines, drawing the outlines contained in each digitized section and drawing lines connecting sections. Structures drawn with solid lines with sections connected are easily seen within larger structures drawn with dashed lines without sections being connected.

The second development proved even more powerful, that of producing stereoscopic figures. Since stereopsis depends upon the slightly different viewpoints of the two eyes, and the slightly different visual information therefore received by each eye, and since the variation of viewpoint is easily achieved by a three-dimensional rotation, the production of stereo pairs of illustrations is a very simple extension of the reconstruction technique.

The methods have enabled members of the Neurobiology Laboratory to complete analysis of the development of the chick wing innervation, and a number of related studies have been completed and are in progress. Quite a number of computer-drawn illustrations have been published. Most of the work would have been impossible without such a powerful tool.

References: Bennett, M.R., D.F. Davey, and K.E. Uebel, "The Growth of Segmental Nerves From the Brachial Myotomes Into the Proximal Muscles of the Chick Forelimb During Development," J. Comparative Neurology, 189, pp. 335-357.
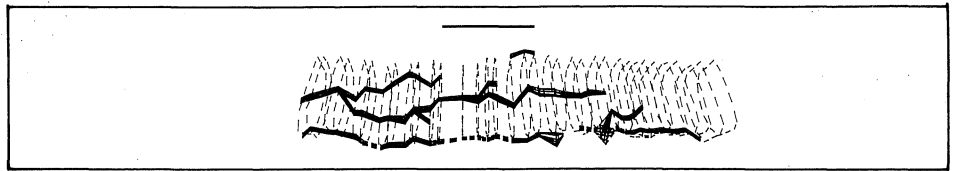
Fig. 6. Example of a full reconstruction of serial sections through a muscle cell (dashed outlines) through the region where nerve cell branches (solid lines connected from section to section) contact the muscle. Parts of one of nerve branches was found in the electron microscope not to make functional contact. This is indicated by broken connecting lines. Scale bar represents 0.1 mm.
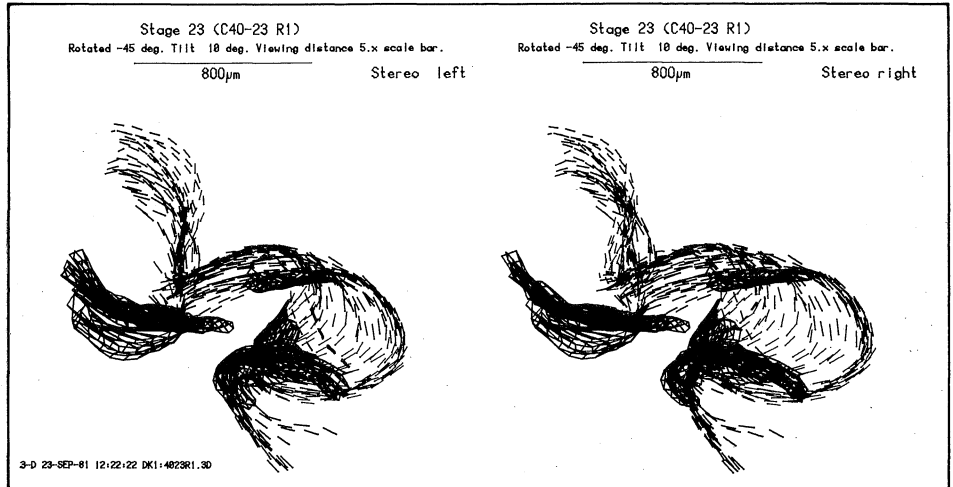
Fig. 7a. Chick wing about 1/2 day older than the one of Figs. 2 through 5. The nerves have entered the wing and are headed for the lower muscle density. The upper density is newly formed, and would have been reached by a nerve branch in another few hours.
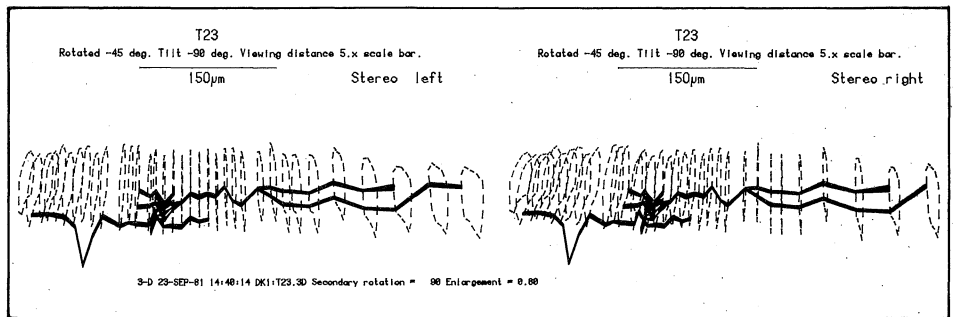
Fig. 7b. An unusual toad-muscle cell contacted by two different nerve cells. Stereoscopic examination reveals the right-hand (branched) nerve to be on the far side of the muscle cell. Its apparent overlap with the left hand nerve can be resolved.

Fig. 7. These pairs of figures are calculated so that if the left figure is viewed with the left eye and fused with the right figure viewed with the right eye, the three dimensional structure can be seen.
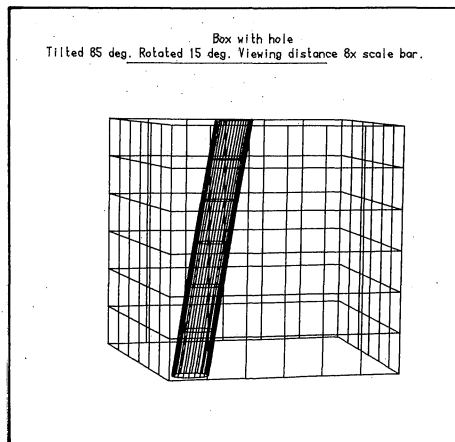
Fig. 8a. Test data consisting of a cube with an angled hole was used in the programme development.
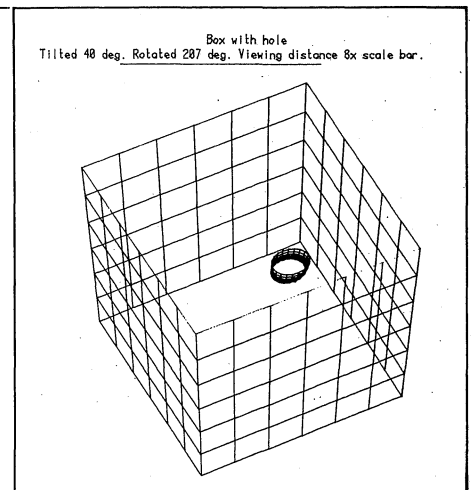
Fig. 8b. Looking through the hole.

4050 Users Share Data Storage and Retrieval in New Tektronix Hard Disk

by Chuck Smith
and Dave Watts
   Tektronix, Inc.
   Wilsonville, OR

... 1 to 10 simultaneous users ... 768 million bytes of disk storage ... removable disk cartridge ... dynamically allocated files ... variable length records ... indexed files ... file security ... continuous time/date clock ...

These functions characterize the advanced technology of the TEKTRONIX 4909 Multi-User File Management System. The 4909's smart approach to file management and its cost effective data storage and sharing bring convenient economical mass storage to 4050 Series users.

Flexible configurations allow easy expansion of users or storage. The modular design of the 4909's controller connects up to 11 GPIB or special purpose interfaces through simple plug-in slots.

Each GPIB interface supports one 4050 desktop computer; each disk interface handles two disk drives.

With full plug-in configuration up to 10 4050 Systems may be interfaced to the 4909–10 simultaneous users! It solves the transportability problem for users who share data; it also provides large local storage for applications such as drafting, design, mapping, data acquisition, data analysis, research and development.

When adding more users to a system, no software or operating system change are required. The modular plug in approach to the controller allows you to add or reduce users easily.

As part of the standard 4909 package, a GPIB interface is included which will transfer data in burst mode at 240,000 bytes per second. (Of course, the actual data transfer rate depends on your system and application.) Additional GPIB interfaces are available as users are added.

Maximum storage capacity is 768 megabytes; the standard 4909 configuration includes the controller and one disk drive of 32 megabytes; 16 megabytes of which are

Supports one or two disk drives per interface

Supports one GPIB device per interface
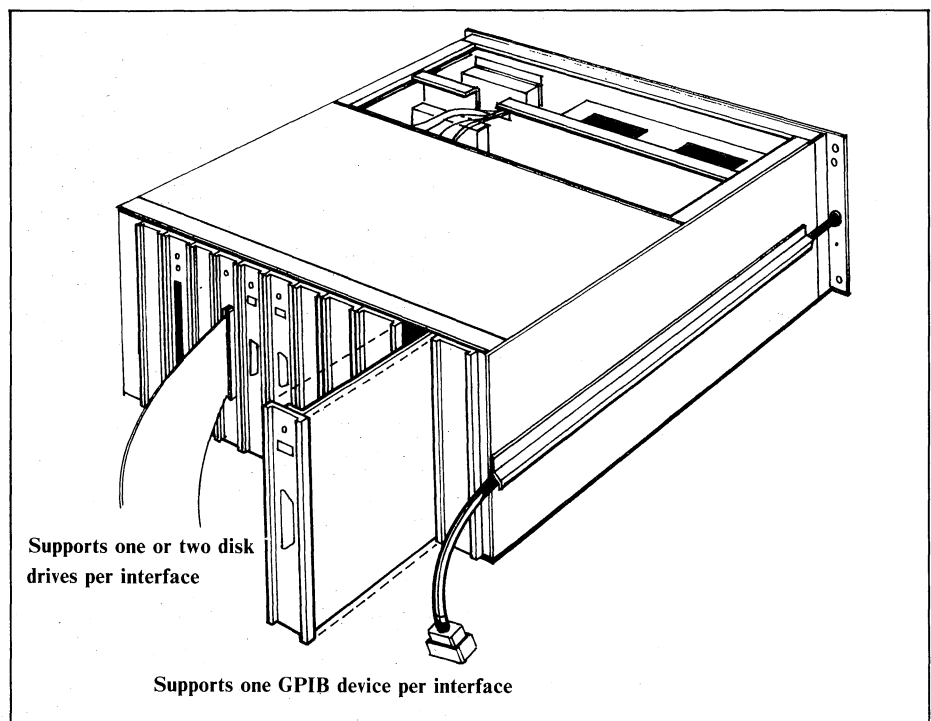
Fig. 1. Easy plug-in interfacing supports up to 11 interfaces.

on a removable cartridge. An optional 4909 disk drive contains 96 megabytes of storage (16 megabytes removable).
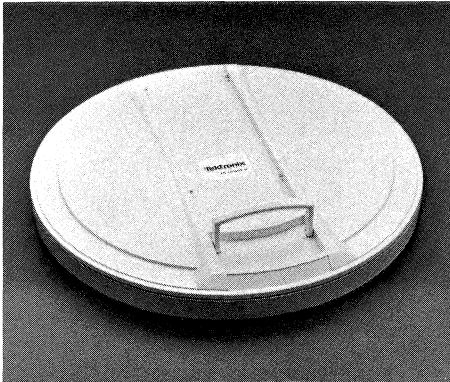
Fig. 2a. 16 Megabyte removable disk pack provides backup and transportability of data.
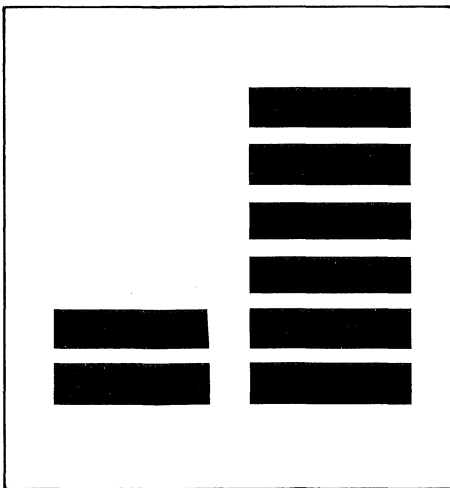
Fig. 2b. Disk capacity is either 32 megabytes or 96 megabytes.

A total of eight drives may be used with one 4909 controller, resulting in maximum storage capacity of 768 megabytes (8 x 96). Additional drives are housed in attractive auxiliary cabinets, each of which supports one or two disk drives.
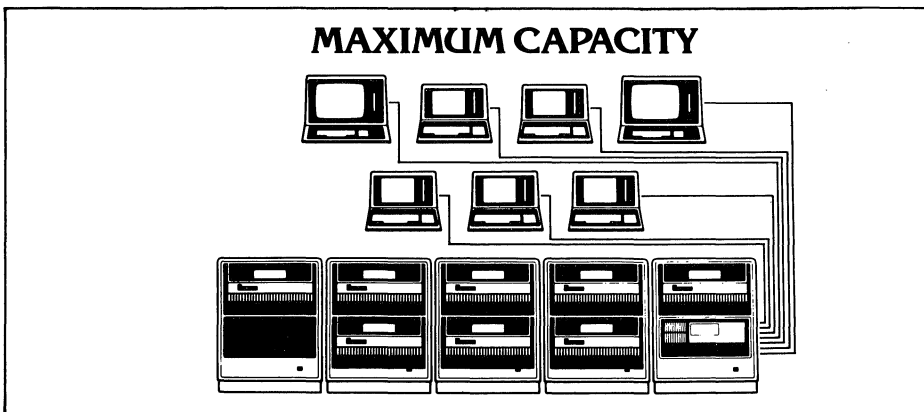
English (ASCII) commands which operate the 4909 may be directly transmitted over the GPIB. Therefore, any computer or instrument controller which supports IEEE-488 1978 may access the 4909. For 4050 Series users, a ROM pack is available which allows easy access to the 4909's file handling in 4050 BASIC, without having to use GPIB print and input statements for file operation.

Through its intelligent controller, advanced file management functions formerly available only on mini or mainframe systems have been implemented in the 4909. Because the file management system resides within the 4909, very little 4050 memory is required and commands are executed from the 4909 not from 4050 software. With more of the file handling chores built into the 4909, it takes fewer commands to get the job done; and default parameters make the commands easy to use.

File Space Allocated Dynamically

For instance, the space required for data is calculated by the 4909. It expands or contracts the file size as data is stored or modified. No longer does the user need to worry about estimating the correct file size or what to do when he reaches the end of the file.
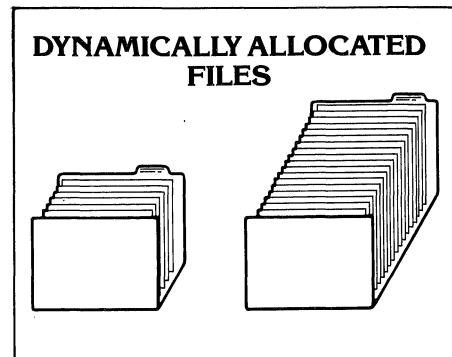
Fig. 4. 4909 handles file management chores.

Records May Be Variable Lengths

Records within a file may be different lengths. For example, within a binary file a record holding x,y,z coordinate position data might contain 28 bytes while in the same file another record holding a text string might contain 100 bytes. And the storage space for each record is calculated and allocated as needed by the 4909.

Concatenated Volumes Link Disk

Concatenation allows two or more disk to be treated as if they were one. Files on one disk may overflow to another, yet appear as one logical unit. Thus, the remaining capacity on a disk is not a limiting factor when storing data. Large data bases may be continued from one disk to another. And, information may be retrieved without specifying which disk it is on.
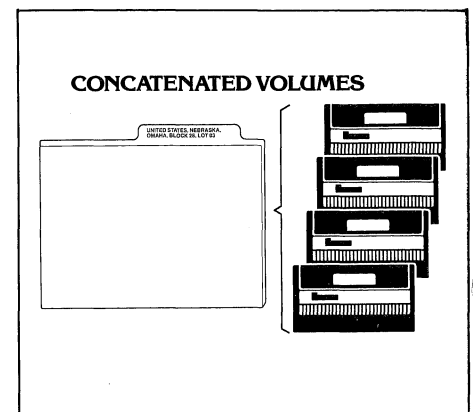
Fig. 5 Files may occupy more than one disk yet be accessed as though on one.

Multiple Level Storage Structure

There is no limit to the number of libraries permitted in the 4909 storage structure and file names may be up to 100 characters long. These features result in meaningful file storage.
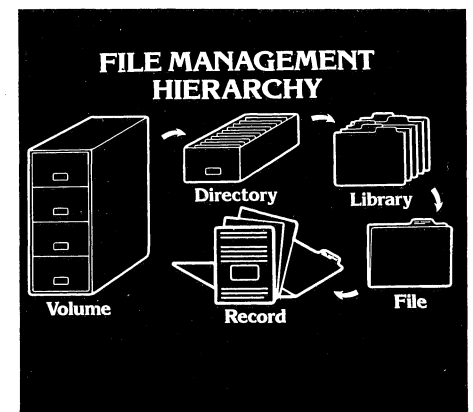
Fig. 6a. An overview of 4909 file management hierarchy.

Fig. 3. A fully configured system would include 8 disk drives and 7 users.
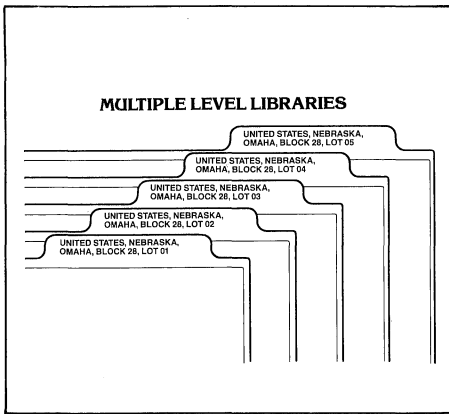
Fig. 6b. Multiple level libraries result in coherent data bases.

INDEXED FILES

FLOOR PLAN

UNITED STATES, NEBRASKA,
OMAHA, BLOCK 28, LOT 03

UNITED STATES, NEBRASKA,
OMAHA, BLOCK 28, LOT 03, FLOOR PLAN

Fig. 7a. ASCII indexed files are a natural method for storing and retrieving data.

FILE NAMES

UNITED STATES, NEBRASKA,
OMAHA, BLOCK 28, LOT 03

Fig. 6c. File names may be up to 100 characters long.

KEY

Index: "FLOORPLAN"
or
Approximate: "FLOOR"
or
Prefix: "FL"
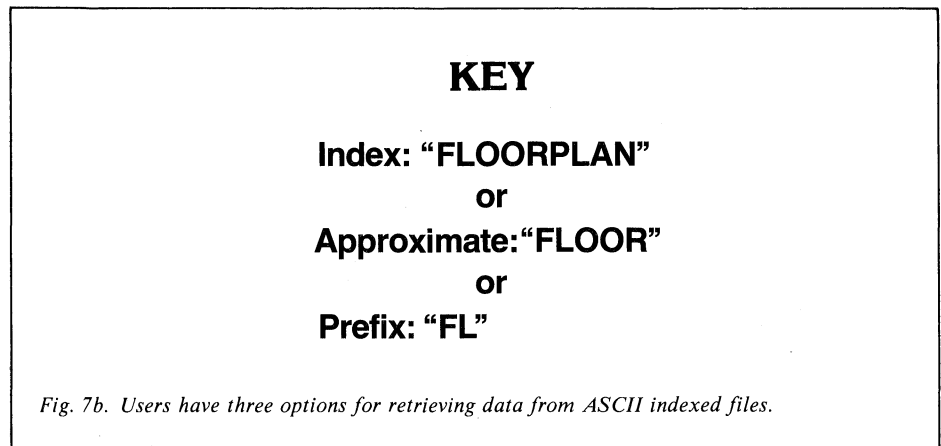
Fig. 7b. Users have three options for retrieving data from ASCII indexed files.

Fig. 7. Schematic of an ASCII indexed file. The combination of file types offers a blend of efficient data storage and flexibility.

Three File Types Provide Flexible Data Manipulation

The 4909 offers six types of data files: binary or ASCII direct, binary or ASCII numeric indexed, or ASCII indexed binary or ASCII.

A **direct** file contains a stream of bytes usually accessed in a serial fashion. However, an individual byte may be addressed by specifying the byte number relative to the beginning of the file.

Indexed files are similar to direct files but are divided into records. Each record may be accessed by a unique key. If the file is a **numeric indexed** file, the keys are integers and a record is retrieved by number, like most random access files.

If the file is an **ASCII indexed** file, the keys are alphanumeric, i.e., employee names, payroll codes. Because names are easier for most of us to remember, ASCII indexed files provide a natural method for storing aand retrieving data. And the 4909 can be directed to locate a record in an ASCII indexed file without stipulating the record's exact key. By specifying an approximate key, the 4909 will locate that index and then step through other records having a key equal to or greater than the specified key. A prefix search will locate an index that starts with the specified prefix. This is many times faster than searching numeric indexed records from beginning to end, or faster even than a binary search of numeric indexed records.

Files Carry Time and Date

Battery backup on the 4909's clock means it's set at the time of installation only. It doesn't have to be reset at each power up. The current time and date will automatically be stamped on files as operations on them occur.

Disk Security

Each disk drive consists of a fixed and a removable portion, each of which is considered a separate physical volume. However, the term "volume" may be extended to include several physical volumes.

When a disk is initially formatted, volume configuration is stipulated and names are assigned to each logical volume. At the same time the user may provide a master password for each logical volume. If included, other users must supply this password when attempting to reformat the disk(s) or when performing restricted operations.

Volume security also comes in the form of the removable cartridge disk which allows files to be backed up and removed from the system for archiving.

Public and Private Space

Depending upon the number of hard disks interfaced to it, the 4909 can support from one to 10 users at one time. To accommodate the users, public and private file workspaces may be declared.

Private file workspaces are declared in advance for individuals and each is assigned their own password. No one may access a private space without the correct password. Public workspace provides users common access to the same storage space. Thus casual users of the 4909 are not required to remember passwords or special commands. Files need not be copied to a users space in order to access them.

For each file, an access list may be assigned which specifies which users have access to the file and what type of access is allowed. The five levels of access are: full access, write access, append access, read access and no access. Several users may access the same file simultaneously.

Another type of file access works much like an airline reservation system. A user can obtain exclusive use of a file for updating. Operations on the file by others are not permitted until the file is released. Requests to read, write, etc., are queued until the file is free, then are handled on a first come, first served basis. (Users have the option of not remaining in the queue should a file be reserved when they wish to access it.)

4909 Functionally Compatible with 4907

Although the 4909 offers more features than the TEKTRONIX 4907 File Manager and has differing commands, it is functionally compatible. For 4907 users wishing to convert their programs and data files to the 4909, a utility is available through the 4050 Series Applications Library. The utility programs aid in the conversion and identify the changes required in the 4907 program to complete the conversion. The 4907-to-4909 conversion utility will be described in more detail in a future issue of TEKniques.

4909 Not Just Another Hard Disk

The 4909 allows users to store their own individual programs and data, but also allows them access to common programs and data. The latest program version will easily be available to users; the latest data will be at their fingertips. And sharing achieves more value per byte of storage.

4909 storage capacity permits users to download and store a very large amount of data, thus alleviating the interaction problems with a host and increasing user accessibility and control.

While easily slipping into any 4050 Series environment, the 4909 also provides mass storage for data acquired from TEKTRONIX Test and Measurement instruments, or any instrument controller supporting GPIB. The 4909 Supports Tektronix Codes and Formats for GPIB instruments.*

Advanced file management and easy expandability ensure the 4909 a place as a

---

*Tektronix Codes and Formats information is available in Tektronix publication part number 99AX-4607. The standard was also discussed in Electronics, March 24, 1981: "In-house standards fill gaps in instrument-computer interface," by Maris Graube, Tektronix, Inc.

long term companion to the 4050 Systems and other GPIB compatible system.

For those interested in details, the following table lists some of the features of the 4909.
  14" hard disk
  600 bits per inch
  384 tracks per inch
  30 millisec seek time
  8.33 millisec latency

Your TEKTRONIX Sales Engineer will be happy to supply you with more details on the 4909 Multi-User File Management System.

Computers Making Computers

by Carter West
   Prime Metals
   San Carlos, CA

Take a look at your 4050 system. The outside is wrapped in sheet metal; the inside contains many metal parts enclosing electrical components, mechanical components and so forth. Prime Metals is a precision sheet metal firm manufacturing these types of metal components for computer systems in use today. One of our best " workers" in the process from design to final product is the 4051 desktop computer.

Using a series of programs, the 4051's capabilities, and our U.S. Amada punch press, we can turn out parts efficiently and accurately. A human operator could possibly make one accurate part, but not 3,000 and not at 4:30 in the afternoon or 8:00 on a Monday morning.

Using a 4051-to-paper tape-to-punch press system, Prime Metals has increased productivity and accuracy in the sheet metal fabricating shop.

Fig. 1. A punch press works very much like a sewing machine. For one punch, the head comes down, punches, then raises and moves to the next punch position. For contouring, the head comes down and up rapidly, continuously moving a short distance to make a smooth cut; this is called nibbling. Note that the 4051 calculates user time of each part, a big help in costing or quoting.

Machine operator mounts the sheet metal " blanks" and clamps them. The rest is automatic. (Photos courtesy of Prime Metals.)

As the first step, we define the part on the 4051. The code produced will eventually be transmitted to a paper tape drive which punches out a tape to instruct the punch press.

Program functions are invoked through the 4051's User-Definable Keys (UDK). Press a UDK and the program prompts for the new part. Referring to a book of codes, and reading from a blueprint, the operator keys in the move or draw and the X, Y dimensions of a part.

Once the cut dimensions and punch positions are keyed in, another UDK is pressed and the operator keys in tool instructions. Our punch press has a turret tool holder with 72 tool stations.

Therefore, we must tell it which tool to put in which station and which station to use at which cut. For instance we might use a 1" by 1" punch at station 1, a 1/4" round hole punch at station 3, and so on.

The sheet size is figured so the shearing department can cut the metal blank to those
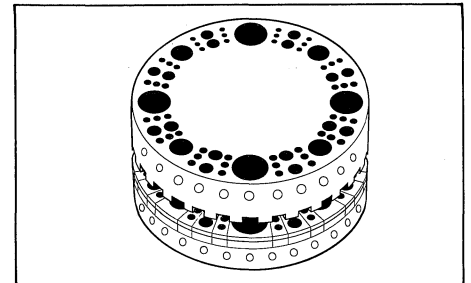
Fig. 2. Instructions from 4051 to punch press include which tool to put in which of the 72 stations in the turret tool holder. (Diagram from U.S. Amada Ltd.)

dimensions. Clamp positions guide the machine operator in clamping the sheet metal on the punch press.

Before sending the code to the 4907 disk, the operator plots on the screen exactly what the punch press will do. If desired, we can take calipers, scale the plot, and tell if a hole or cut is within at least 1/16 of an inch to the blueprint callout.
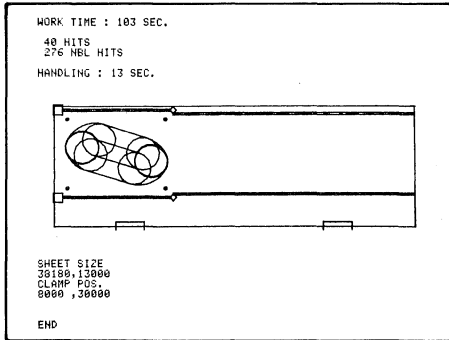
Fig. 3. Previewing the plot on the 4051 display prevents mistakes. Notice the elliptical cutout automatically calculated by the 4051.

Once the operator is happy with the input, it's stored on the disk. What we have at this point is a series of codes that the paper tape drive understands.
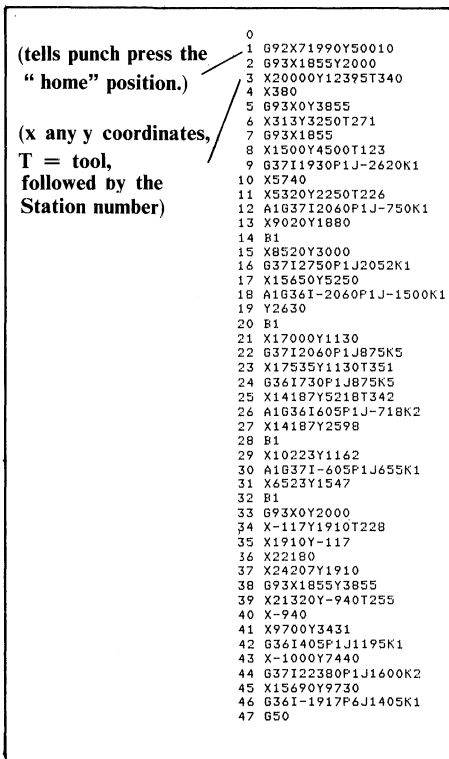
Fig. 4. Each line of code instructs the machine to move or cut or punch, specifies the coordinates and the tool station.

When we're ready to punch the tape, the data is transmitted over the Option 1 to the tape drive. We can also read the code back in from the paper tape punch.

At production time the tape is sent to the machine operator on the fabricating floor. Just before the tape goes downstairs another 4051 program enters the name of the customer, tape number, job number and so forth. We also input the tools used, the location of the tools and the shear sizes. The hard copy of the plot is put in plastic with this information slipped into the other side. By tagging this to the sheet metal the shop foreman knows exactly what's being worked on and for whom.

After the machine operator inserts the tape into the punch press controller, an optical scanner reads the tape and transmits the instructions to the machine. All the operator has to do is mount the sheet metal on the punch press, clamp it, then remove it and repeat the process. High level skills are not required, and allows the operator to perform other duties until the part is completed.

From feedback signals, the control unit verifies the machine is cutting and punching where it should be. Accuracy is within 1/5000 of an inch.

The 4051 has created a whole new approach for us. It can do anything. For instance, to calculate a 13° arc across a 12° shape manually would require days; the 4051 can do it in 20 seconds, and it's accurate!

It's also very easy to define identical parts for the same sheet. We key in the dimensions one time. The 4051 calculates the placement of the like parts, where they will be cut or punched, and optimizes the moves between parts or cutting areas and tool changes.
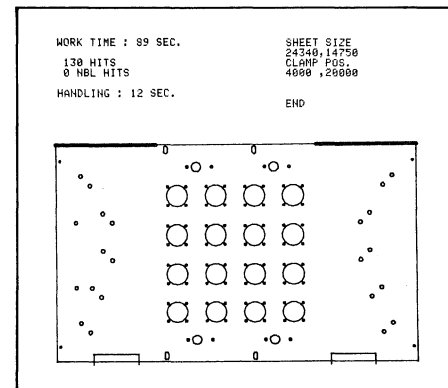
Fig. 5. The 4051 will memorize and repeat a pattern. The design was programmed only once, then the 4051 told to repeat it 15 more times.

However, the 4051 is not limited to coding sheet metal fabrication. We use it for payroll, job storage and inventory. One of our immediate goals is using it for status reporting. The secretary would input the status of all jobs based on purchase orders received and jobs finished. We predict a reduction in

time spent updating this report from two hours to 15 minutes; and this is a daily report. Thus, the information for each customer's would be available first thing in the morning–very critical since customers have a daily need to know when their parts will be available.

We work on a tight schedule and with close tolerances; the 4051 is helping us do a better job for our customers. 🖅
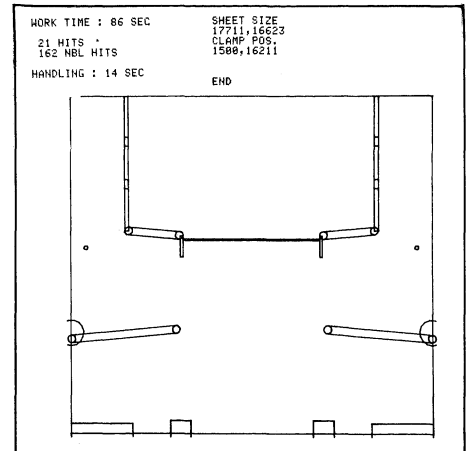
Fig. 6. The 4051 will compute any angle (or arcs) asked of it.
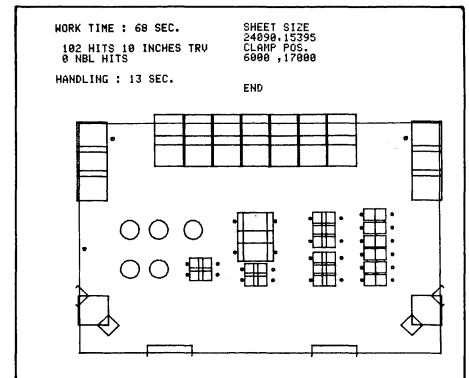
Fig. 7. Hard copy specifies sheet size for the shearing department and clamp positions for the machine operator. It also carries the work time for the punch press and handling time for the machine operator.
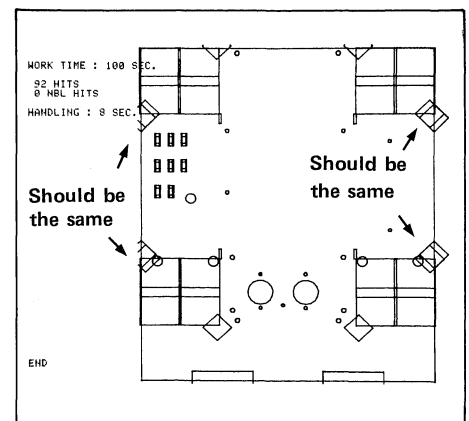
Fig. 8. The use of "plotting" instantly shows errors to be corrected.

New ROM Pack Produces High Quality Characters and Symbols on 4052 and 4054 Graphic Systems and Plotters
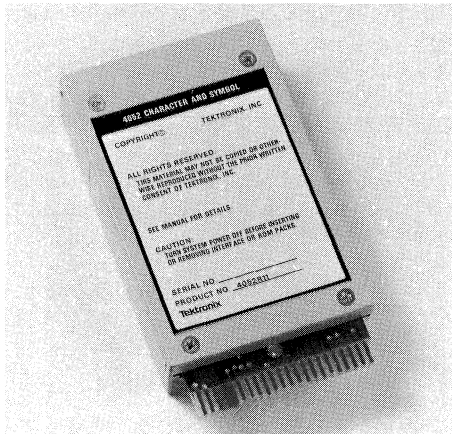
by Mark Mehall
Tektronix, Inc.
Beaverton, OR

The 4052R11 Character and Symbol ROM pack brings high quality stroked characters or symbols to the 4052 or 4054 Graphic System and the 4662 or 4663 Plotter. It adds 27 commands to 4050 Series BASIC to display alphanumeric characters and symbols, draw smoothed arcs and circles, generate custom symbols or provide additional graphics enhancements. Character/symbol sizing, spacing, proportioning, rotation angle, degree of slant, and smoothness are easily programmed.
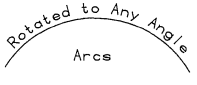
Over 100 different characters/symbols are contained in the 4052R11 ROM pack including upper and lower case letters for both the English and Greek alphabets, mathematical and special symbols, and characters for Swedish, German, Spanish and Danish languages.

Using the Character and Symbol ROM, you may easily draw arcs and circles. Circles are drawn around the cursor position at a given radius. Arcs can be specified by giving either the radius and starting and ending angles of the arc, or by giving the center of the arc and the angle from the current cursor position.

You may also design your own symbols which can be manipulated in the same manner as the ROM pack characters/symbols.

The Character and Symbol ROM Pack provides additional graphics functions for more flexible graphic input from the 4052 and 4054. Using the ROM with the Dynamic Graphics Option of the 4054, you may drag refreshed objects around the screen, dynamically reading the coordinate locations of the object.

You'll find the Character and Symbol ROM pack an invaluable aid for labeling engineering drawings, preparing presentation graphics, or annotating any type of alphanumeric or graphic design. Contact your local Tektronix Sales Engineer for more details about the versatile 4052R11 ROM pack.

Sample of the characters generated using the 4052R11 Character and Symbol ROM pack.
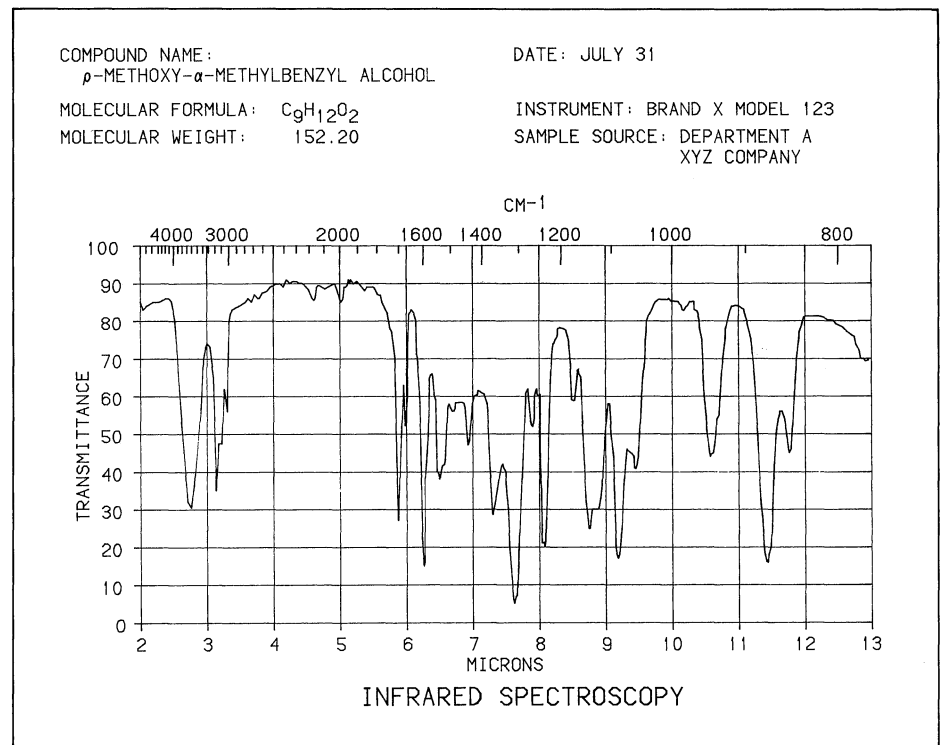
The 4052R11 allows High Quality Annotation of Graphs and Drawings.

Input/ Output

File Names for the 4907 File Manager

W. Budde from the Optics Section of the National Research Council in Ottawa, Canada, writes: Could you please tell me what the character requirements are for file names for the 4907 File Manager? I can't seem to locate them in the 4907 Operator's manual (October, 1980).

The requirements for file names are:

> 1 to 10 characters.
> First character must be alpha, the rest may be alpha or numeric.
> No special characters allowed.
> No spaces or delimiters.

Internal Representation of Numbers

Several readers have requested the internal format for numeric data in the 4050 system. Much of the following information has been extracted from the 4051 GPIB Hardware Support manual (part #070-2270-00).

All numeric values that enter the 4050 Random Access Memory (RAM) are stored in a special eight-byte floating point format.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

Status and Exponent Information

The first byte and the second byte contain status and exponent information. The format for these two bytes is shown below:
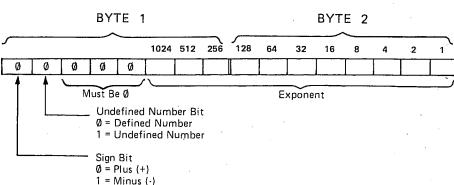
The bit on the far left (bit 8 in byte 1) is the sign bit. This bit tells the 4050 processor whether the numeric value is positive or negative. If the bit is 0, the number is positive or 0. If the bit is 1, the number is negative.

The second bit from the left (bit 7 in byte 1) is the undefined number bit. The 4050 uses this bit in arrays to detect undefined elements. Binary numbers coming into the 4050 will always have this bit set to 0.

Three bits in byte 1 must be set to zero: bit 6, bit 5, and bit 4.

The three least significant bits in byte 1 and all the bits in byte 2 form a binary number that serves as the exponent. The exponent range for 4050 numbers is 2^{-1024} to 2^{1023}. To keep the exponent representation positive, 1024 is added to each exponent to make the range 0-2047. This means that if bit 3 in byte 1 (the most significant in the exponent) is set to 1, the exponent is 0 or positive. If bit 3 in byte 1 is set to 0, the exponent is negative.

Example 1: Bit 3 in byte 1 and bit 2 in byte 2 are set to 1. The rest of the exponent bits are set to 0. What is the true exponent of the floating-point number?
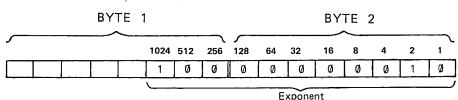
This binary bit pattern represents 1026. The true exponent is found by subtracting 1024 from this number, therefore, the exponent is 2 (i.e., 2^2).

Example 2: Bits 1 and 2 in byte 1 and bits 1 through 8 in byte 2 are set to 1. The rest of the exponent bits are set to 0. The true exponent is found by subtracting 1024 from this number, therefore, it is $1023 - 1024$ or -1 (i.e., 2^{-1}).
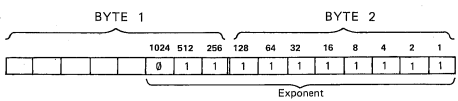
Binary Mantissa Represents Numbers

The remaining bytes in the floating point number form a binary mantissa which allows numbers to be represented with 48 bits of precision. This representation is shown below:
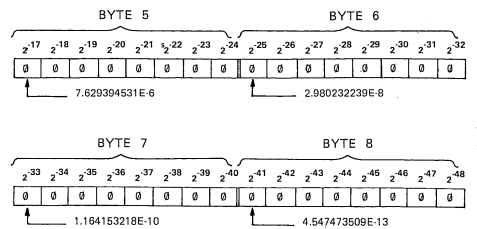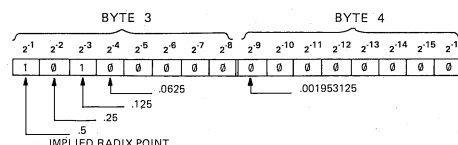
If bit 6 and bit 8 in byte 3 are set to 1 as shown above, and the rest of the bits are 0, the binary mantissa is equal to $2^{-1} + 2^{-3}$ which is equal to .625 in base 10.

In another case, if bit 8 in byte 4 and bit 8 in byte 7 are set to 1 and the rest of the bits are zero, the binary mantissa is equal to $2^{-9} + 2^{-33}$. This number is equivalent to .00195312511642 in base 10.

This representation method allows numbers to be stored in the 4050 memory with a great deal of precision.

Floating Point Numbers are Normalized

ASCII numbers coming into the 4050 are converted to floating point and normalized; that is, the binary bits in the mantissa are shifted left as far as possible and the difference made up in the exponent. The following examples illustrate the difference between a floating point number that is not normalized and a number that is normalized. Both numbers represent the same decimal value.

For instance A = .0625 is entered into the 4050 system. Converting this decimal fraction to its binary equivalent we have .0001.
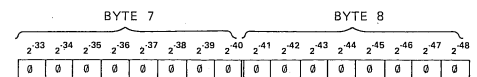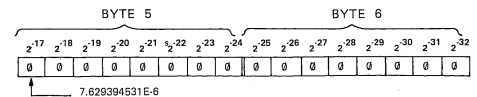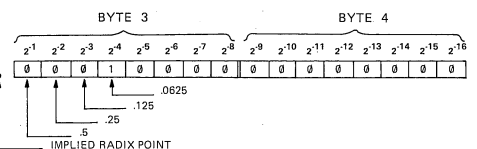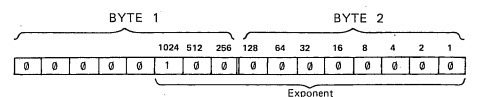
Example 3: The floating point number is not normalized because there are leading zeros in the mantissa.

BYTE 1 | BYTE 2

					1024	512	256	128	64	32	16	8	4	2	1
Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	1	1	1	1	Ø	1

Exponent

BYTE 3 | BYTE 4

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

.0625
.125
.25
.5
IMPLIED RADIX POINT

BYTE 5 | BYTE 6

2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}	2^{-31}	2^{-32}
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

BYTE 7 | BYTE 8

2^{-33}	2^{-34}	2^{-35}	2^{-36}	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}	2^{-45}	2^{-46}	2^{-47}	2^{-48}
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Example 4: The floating point number has been normalized by shifting the mantissa bits three places to the left. To make up for the increase in the value of the mantissa, the exponent is decreased by 3.
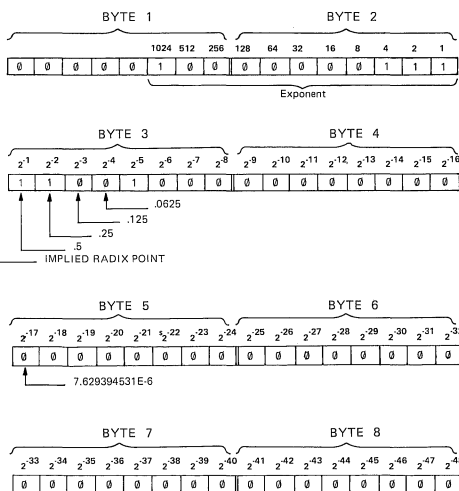
We will illustrate in the next section that the decimal value of this normalized binary number is the same as the decimal value of the binary number shown in example 3.

All binary numbers coming into the 4050 must be normalized because the 4050 firmware math routines assume that all floating point numbers in memory are normalized.

Putting It Together

The complete floating point representation is expressed in the following formula:

$$N = (-1)^S \times M_{10} \times 2^{(E-1024)}$$

where N = the decimal number entered
into the 4050

M_{10} = the decimal equivalent of
the binary mantissa (last
six bytes).

s = the sign bit (1 or 0).

E = the decimal equivalent of the
binary exponent.

Referring to example 3, let's compute the decimal equivalent of the floating point number.

The sign bit S = 0

The exponent E = 1024

The mantissa $M_{10} = 2^{-4} = .0625$

The decimal number $N = (-1)^0 \times .0625 \times 2^{(1024-1024)}$

$$N = 1 \times .0625 \times 1$$
$$N = .0625$$

Now let's compute the decimal equivalent of the normalized floating point number in example 4.

The sign bit S = 0

The exponent E = 1021

The mantissa $M_{10} = 2^{-1} = .5$

The decimal number $N = (-1)^0 \times .5 \times 2^{(1021-1024)}$

$$N = 1 \times .5 \times 2^{-3}$$
$$N = .0625$$

If A = 100 is entered into the 4050 system, its floating point representation is:

BYTE 1 | BYTE 2

					1024	512	256	128	64	32	16	8	4	2	1
Ø	Ø	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1

Exponent

BYTE 3 | BYTE 4

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
1	1	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

.0625
.125
.25
.5
IMPLIED RADIX POINT

BYTE 5 | BYTE 6

2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}	2^{-31}	2^{-32}
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

7.629394531E-6

BYTE 7 | BYTE 8

2^{-33}	2^{-34}	2^{-35}	2^{-36}	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}	2^{-45}	2^{-46}	2^{-47}	2^{-48}
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

First, the decimal 100 is converted to its binary equivalent of 1100100. This value is normalized by shifting it right seven places. The change in value is reflected by increasing the exponent by 7, or to 1031.

Now, applying the formula

The sign bit S = 0

The exponent E = 1031

The mantissa $M_{10} = 2^{-1} + 2^{-2} + 2^{-5} = .5 + .25 + .03125 = .78125$

The decimal number $N = (-1)^0 \times .78125 \times 2^{(1031-1024)}$

$$N = 1 \times .78125 \times 2^7$$
$$N = 100$$

To represent the number 0.0, bit 3 in byte 1 is set to 1 (exponent = 1024), the remaining bits in the 8 bytes are all set to 0.

Getting Larger Images from the 4611 Hard Copy Unit

by Cathy Cramer
Tektronix, Inc.
Wilsonville, OR

We're pleased to announce a new option for the 4611 electrostatic Hard Copy Unit*. Called Option 1 for Landscale Image Format, it provides a larger image on the paper, for easier reading and better detail.

Option 1 is factory or field-installable, and creates images in a "landscape" rather than "portrait" orientation, with an enlarged image size. The landscape format provides a 10" by 7.5" image on legal-size paper (see Figure 1). That's 30 extra square inches of image area! The legal-size paper can be hand-trimmed to 8 1/2" X 11" with one cut, leaving a half-inch border on all sides of the image.

The landscape format provides a more readable image, and improved copy quality when used with 19" or larger storage tube displays. Dot size, line width, dot density (dots per square inch), and toner concentration all remain unchanged. However, the total number of dots per image increases, providing finer detail for small-size alphanumerics and dense graphics.

A sample copy is printed on the following page. Since it's reproduced by a printing process, it does not represent the copy quality of an actual sample. However, it does show you the new image size and orientation.

For more information, contact your local Tektronix representative about the 4611 Option 1 Hard Copy Unit. 📠

TEKniques Vol. 5 No 1 introduced the 4611 Hard Copy Unit.

DATA ANALYSIS OF SAMPLE 123.6-ASEWQ (37)

RELATIVE CONCENTRATION

PARTICLE COUNT

Temperature = 123 Degrees Celsius
Pressure = 70mm
Magnetic Flux = 220

PROCESS WSX892.129

Standard portrait format

Option 1 landscape format
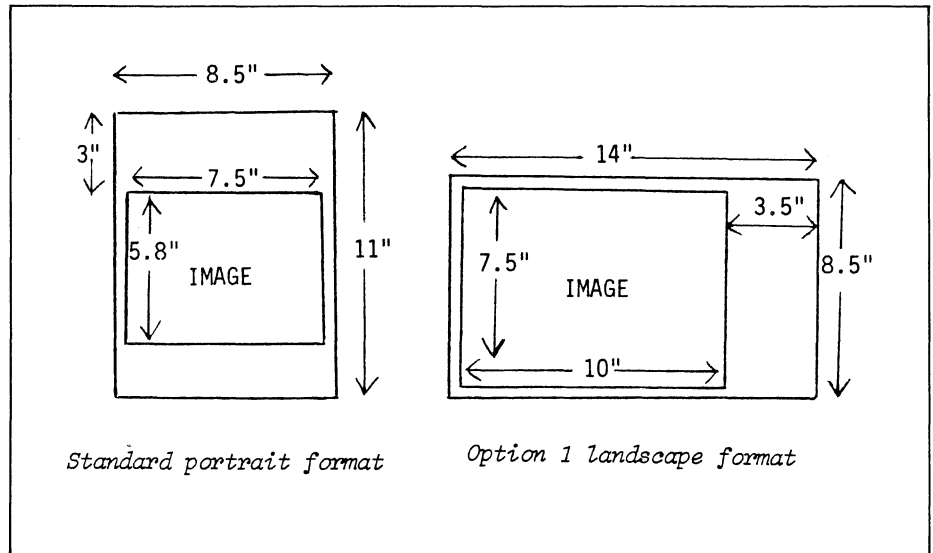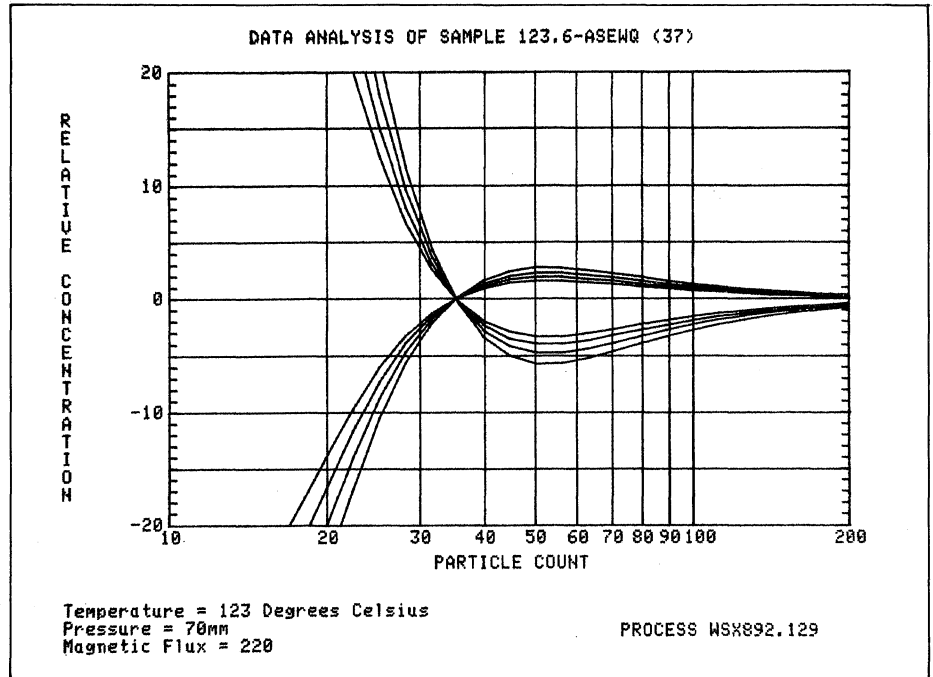
4611 Standard and Optional Image Formats

Belgium, United States Entrants
Win Firsts in Graphing Contest

Dony Robert, mathematics instructor from Brussels, Belgium, walked away with first prize in the 3-D Graphing contest sponsored by TEKniques. His "General Function Z= F(X,Y) Plot" draws two variable functions with hidden lines removed. The program, user interface and documentation are outstanding. 4050 Applications Library software users may remember Mr. Robert as having contributed "3D Function Plot" to the Applications Library a couple of years ago.

From Whirlpool Corporation in Benton Harbor, Michigan, Steven Salisbury sent in his unique "Pipechart." His original approach to data representation garnered Steve first place in the 2-D Graphing with Shading contest. He is a marketing data analyst with Whirlpool.

Second place winner in 3-D Graphing is Dr. P.R. Tregenza from the Dept. of Architecture at the University of Notting-

ham in England. Hidden line removal and perspective are included in his function plot. Optional output as a stereo pair enhances his program and provides the name "Stereo Surface."

Jerry Anderson, a quality assurance engineer from Bartlesville, Oklahoma, grabbed second and third prizes in 2-D Graphing with "Contour Plots" and "Hierarchal Clustering." He also entered two additional programs, "Nonlinear Mapping" and "Vertical Plot." Jerry indicates the four programs will aid in applications of statistics, operations research, pattern recognition, cluster analysis, quality assurance, and multivariate analyses.

A 4050 user from the University of Bundesweher in Munich, Germany, captured third prize in the 3-D Graphing. Lothar Tschimpke contributed "3-D Plot with or without Hidden Lines." His code is extremely easy to read.

Honorable mention went to Robert Kennedy and Elliot Noma from Philadelphia, Pennsylvania. "PLOT3D" functions as an appended subroutine to a main program. Their code is well RE-Marked.

All programs are describedin the "New Abstracts" section of TEKniques.

TEKniques thanks all of the contributors who took time out of busy schedules to document and send in their programs. Their programs will provide many 4050 users with unique and useful methods for graphing.

New GPIB Guide Features 4050
as Instrument Controller

GPIB instruments designed by Tektronix go beyond simply conforming to the IEEE Standard 488-1978. Tektronix GPIB instruments use the Tektronix Standard for Codes and Formats* providing a common message structure for all system instruments. The result is easier, faster, and more efficient system integration and application programming between the controller and the instruments.

The GPIB Programming Guide (part # 070-3985-00 aids the user of 4050 Desktop Computers and TEKTRONIX TM5000 series instruments in making the software

connection. The Guide introduces the 4050 as a controller for TM5000 instruments, with programming information specific to instrument control. Major topics in this section with coded examples where applicable are:

    4050 Desktop Computer IEEE controller capabilities
    GPIB Input/Output
    Interrupt handling
    Interrupt handling statements
    Utility routines
    4052/GPIB send and receive program

TEKTRONIX TM5000 series instruments specifically covered in the Guide include:

    D C 5 0 0 9 and D C 5 0 1 0 — Programmable Universal Counter/Timers

    DM5010 — Programmable Digital Multimeter
    FG5010 — Programmable Function Generator
    PS5010 — Programmable Power Supply

A final section lists programs that show two or more Tektronix TM5000 instruments working together in a system with the 4050 as the controller. The guide includes source code for 47 routines. Machine readable copies of the routines are included in the 4050 Applications Library as part of TEKniques Vol. 5 No. 4 T1 tape, part number 062-5981-01.

---

*The Tektronix Standard for Codes and Formats is discussed in Tektronix publication "A Vital Link in Instrument Systems: The General-Purpose Interface Bus (GPIB)," part #AX-4524-1.

Programming Tips
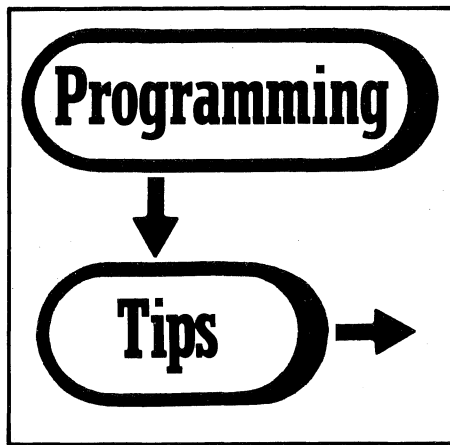
TYPE Function and 4051

by Tom Price
   Lorillard
   Greensboro, NC

Calling the TYPE function before writing to an empty file on a 4051 will overwrite the file header. Issuing a second FIND command after the TYP function corrects the problem.

Overwrites Header:

```
150 FIND F
160 IF TYP(0)<>0 THEN 200
170 PRINT @33:A$
200 REM GET ANOTHER FILE
```

Preserves Header:

```
150 FIND F
160 IF TYP(0)<>0 THEN 200
170 FIND F
180 PRINT @33:A$
```

Conditional GOSUB

by Tom Price
   Lorillard
   Greensboro, NC

A useful programming construct is the conditional GOSUB. How many times in 4051 BASIC have you wanted to write IF A<=B THEN GOSUB 500? It would probably have come out coded like this:

```
150 IF A>B THEN 170
160 GOSUB 500
170 REM ---
```

Conditional GOSUBs make programs much easier to read and cut out lots of "dead" GOTOs. The above example becomes:

```
150 GOSUB A<=B OF 500
```

This will work for any conditional except one containing an equals preceded by one variable name. The 4051 parser will not accept it unless it's enclosed in parentheses. It then removes the parentheses and is unable to reparse it if the line is edited or the program is saved in ASCII format.

Won't work:

```
GOSUB A=B OF 500
GOSUB L=50 OF 1000
```

Works well:

```
GOSUB NOT (A<>B) OF 500
GOSUB 50=L OF 1000
```

This affects a very small percentage of the cases encountered during normal programming. No extra memory is required when the branch is not taken. I think you will find this novel and quite useful.

Dynamic Programming Aids

by Scott E. Lee
   Tektronix, Inc.
   Indianapolis, IN

Several techniques I have found useful are incorporated in the following program. The main function of the program (lines 100 to 230) is to dynamically create and evaluate expressions which might be useful in several cases:

```
4 GO TO 100
40 PAGE
41 LIST 100,230
42 RETURN
80 KILL "EVAL.BAK"
81 CALL "RENAME",0,"EVAL","EVAL.BAK"
82 SAVE "EVAL"
83 RETURN
100 INIT
110 PRINT "TYPE EXPRESSION: ";
120 INPUT A$
130 A$="10000 PRINT "&A$
140 CALL "FILE",0,"@EXPEVAL.LINE",S$
150 IF S$<>"" THEN 170
160 CREATE "@EXPEVAL.LINE","A";1,0
170 OPEN "@EXPEVAL.LINE";1,"F",S$
180 PRINT #1:A$
190 PRINT #1:
200 CLOSE 1
210 APPEND "@EXPEVAL.LINE","A";220
220 REM This line will be replaced with a PRINT
230 GO TO 110
```

1. Using "Calculator Mode" without exiting from a program.

2. Examining variables for program debugging without pressing BREAK (this mode could be invoked with a UDK).

3. Creating functions on the fly by concatenating together things like ``"+SIN(X)"``, ``"+COS(X)"``, and ``"+LOG(X)"``. In this case an assignment statement should be created instead of a PRINT.

This same technique could also be used to create multi-line routines to be APPENDed.

Note that the extra PRINT at line 190 is needed to generate a blank line which means end-of-file to the APPEND command.

Also in this program (lines 80 to 83) is a UDK definition that will SAVE the program and make a backup of the previous version.

Lines 40 to 42 allow you to list a section of the program (for example the section you are currently working on) with only one keystroke. Sometimes I will define several of the UDK listing keys, one for each subroutine that I am debugging.

Rotation About an Arbitrary Axis

by **Javad Farjami**
Tektronix, Inc.
Wilsonville, OR
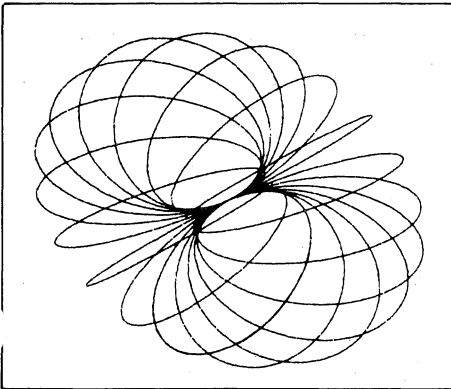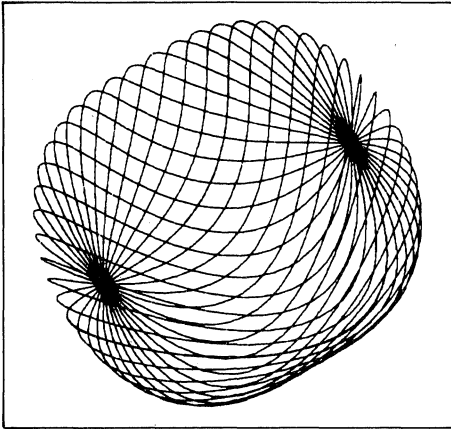
The steps to rotate an object through an angle θ about an arbitrary axis are:

1. Translate the object into a new coordinate system (Matrix T):

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_o & -y_o & -z_o & 1 \end{bmatrix}$$

where (x_o, y_o, z_o) is a point through which the rotation axis passes.

2. Map the unit vector which lies along the axis of rotation into the unit vector along the Z axis (Matrices R_1, R_2):

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \dfrac{c}{\sqrt{b^2+c^2}} & \dfrac{b}{\sqrt{b^2+c^2}} & 0 \\ 0 & \dfrac{-b}{\sqrt{b^2+c^2}} & \dfrac{c}{\sqrt{b^2+c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \sqrt{b^2+c^2} & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & \sqrt{b^2+c^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $<a,b,c>$ is a unit vector along the axis of rotation.

3. Perform the desired rotation θ about the Z axis of the new coordinate system (Matrix R_θ):

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Apply the inverse of step 2 (Matrices R_2^{-1}, R_1^{-1}):

$$R_2^{-1} = \begin{bmatrix} \dfrac{\sqrt{b^2+c^2}}{a^2+b^2+c^2} & 0 & \dfrac{-a}{a^2+b^2+c^2} & 0 \\ 0 & 1 & 0 & 0 \\ \dfrac{a}{a^2+b^2+c^2} & 0 & \dfrac{\sqrt{b^2+c^2}}{a^2+b^2+c^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \dfrac{c}{\sqrt{b^2+c^2}} & \dfrac{-b}{\sqrt{b^2+c^2}} & 0 \\ 0 & \dfrac{b}{\sqrt{b^2+c^2}} & \dfrac{c}{\sqrt{b^2+c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: $a^2+b^2+c^2 = 1$; since $<a,b,c>$ is assumed to be a unit vector.

5. Apply the inverse of step 1 (Matrix T^{-1}):

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_o & y_o & z_o & 1 \end{bmatrix}$$

Note: the purpose of step 4 and 5 is to return the object to its original coordinate system.

6. The complete transformation is thus:

$$Q = T \cdot R_1 \cdot R_2 \cdot R_\theta \cdot R_2^{-1} \cdot R_1^{-1} \cdot T^{-1}$$

or:

$$[X,Y,Z,1] = [x,y,z,1] \cdot Q$$

or:

$X = (x-x_o)(a^2 + (1-a^2)\cos\theta)$
$+ (y-y_o)(ab(1-\cos\theta)) + c\sin\theta)$
$+ (z-z_o)(ac(1-\cos\theta) - b\sin\theta) + x_o$

$Y = (x-x_o)(ab(1-\cos\theta) - c\sin\theta)$
$+ (y-y_o)(b^2 + (1-b^2)\cos\theta)$
$+ (z-z_o)(bc(1-\cos\theta) + a\sin\theta) + y_o$

$Z = (x-x_o)(ac(1-\cos\theta) + b\sin\theta)$
$+ (y-y_o)(bc(1-\cos\theta) - a\sin\theta)$
$+ (z-z_o)(c^2+(1-c^2)\cos\theta) + z_o$

```
100 INIT
110 PAGE
120 WINDOW -260,260,-200,200
130 PRINT "ENTER A POINT ON THE AXIS OF ROTATION:";
140 INPUT X0,Y0,Z0
150 SET DEGREES
160 PRINT "ENTER THE DIRECTION OF THE AXIS OF ROTATION:";
170 INPUT A,B,C
180 REM    COMPUTE THE COMPONENTS OF THE UNIT VECTOR.
190 D=(A↑2+B↑2+C↑2)↑0.5
200 A=A/D
210 B=B/D
220 C=C/D
230 A2=1-A↑2
240 B2=1-B↑2
250 C2=1-C↑2
260 REM    B5 REPRESENTS THE TANGENT OF THE ANGLE THAT EACH
270 REM    AXIS (X,Y, AND Z) MAKES WITH THE SCREEN.
280 B5=(2/3)↑0.5
290 A5=50
300 M=180
310 PAGE
320 REM    O IS THE ROTATIONAL ANGLE.
330 FOR O=0 TO 360 STEP 20
340 O1=1-COS(O)
350 REM    THE FOLLOWING LOOP DRAWS THE OBJECT (CIRCLE).
360 FOR J=-M TO M STEP 5
370 X1=A5*SIN(J)-X0
380 Y1=A5-Y0
390 Z1=A5*COS(J)-Z0
400 U1=X1*(A↑2+A2*COS(O))
410 V1=Y1*(B↑2+B2*COS(O))
420 W1=Z1*(C↑2+C2*COS(O))
430 U2=Y1*(A*B*O1+C*SIN(O))
440 V2=Z1*(B*C*O1+A*SIN(O))
450 W2=X1*(A*C*O1+B*SIN(O))
460 U3=Z1*(A*C*O1-B*SIN(O))
470 V3=X1*(A*B*O1-C*SIN(O))
480 W3=Y1*(B*C*O1-A*SIN(O))
490 U=(U1+U2+U3+X0)*B5
500 V=(V1+V2+V3+Y0)*B5
510 W=(W1+W2+W3+Z0)*B5
520 X9=(V-U)*COS(30)
530 Y9=W-(U+V)*SIN(30)
540 IF J=-M THEN 570
550 DRAW X9,Y9
560 GO TO 580
570 MOVE X9,Y9
580 NEXT J
590 NEXT O
600 END
```

Function Plot with Auto Scaling

by Dr. J.L. Aubel
    Department of Physics
    University of South Florida
    Tampa, FL

The following program will automatically scale and plot a function — f(x) — which may include user-defined parameters. The function is described at statement 110. Parameters may be set in statements 900 and 990.

```
100 Y=SIN(X)/X

Origin is at 9, 0
Tic intervals are 1 for x and 0.02 for y
```

```
110 Y=TAN(X)
Tic intervals are 20 for x and 2 for y
```
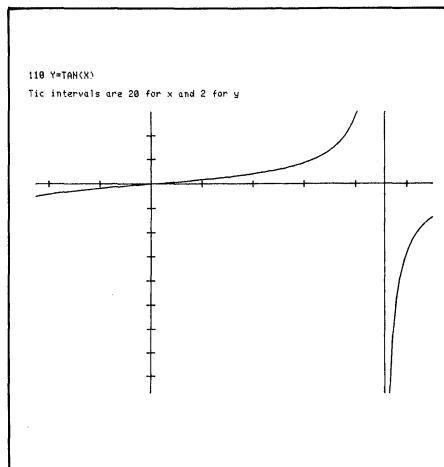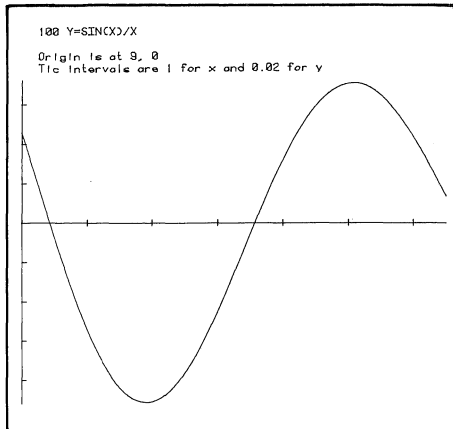
```
1 REM Function plot
2 N=100
3 ON SIZE THEN 400
4 PRINT "LEnter function at line 110.  Then press UDK2"
5 PRINT "Parameters may be entered in lines 900 to 990"
6 D=32
7 END
8 GO TO 890
100 Y=ASN(X)
190 RETURN
200 IF Y=0 THEN 290
210 T5=10*INT(LGT(Y))
220 Y=0.2*INT(5*Y/T5+1)*T5
290 RETURN
300 T5=10*INT(LGT(2*S))
310 T=(2+3*(S/T5)2.5)-(S/T5(1))*T5/10
390 RETURN
400 IF X>X0 THEN 190
410 X=X+X5/10
420 GOSUB 100
430 RETURN
890 REM Set Parameters
1000 PRINT "LEnter Xmin, Xmax"
1010 Y0=0
1020 Y9=0
1030 INPUT X0,X9
1040 X5=(X9-X0)/10
1050 IF X5<=0 THEN 1000
1060 X=X0-X5
1070 X=X+X5
1080 GOSUB 100
1090 Y9=Y9 MAX Y
1100 Y0=Y0 MIN Y
1110 IF X<X9 THEN 1070
1120 Y=Y9
1130 GOSUB 206
1140 Y9=Y
1150 Y=-Y0
1160 GOSUB 200
1170 Y0=-Y
1180 S=Y9-Y0
1190 GOSUB 300
1200 T2=T
1210 S=X9-X0
1220 GOSUB 300
1230 T1=T
1240 PAGE
1250 LIST @D:100
1255 IF X0<=0 AND X9>0 THEN 1270
1260 PRINT @D:"Origin is at ";X0;", 0"
1270 PRINT @D:"Tic intervals are ";T1;" for x and ";T2;" for y"
1280 VIEWPORT 2,130,0,88
1290 WINDOW X0,X9,Y0,Y9
1300 AXIS @D:T1,T2
1310 X=X0
1320 X5=S/N
1330 GOSUB 100
1340 MOVE @D:X,Y
1350 X=X+X5
1360 GOSUB 100
1370 DRAW @D:X,Y
1380 IF X<X9 THEN 1350
1390 HOME
1400 END
```

Message Prompting

by J. Eltabet
    University Paris Val
    de Marne
    Creteil, France

Two subroutines aid in general message prompting and answer input. The first one displays the message using two statements followed with the subroutine to prompt for input.

```
200 W$="QUESTION......"
210 GOSUB 5000
220 REM Here you perform answer treatment
230 END

5000 REM Question Subroutine
5010 FOR W9=1 TO 8
5020 PRINT W$;" : K"
5030 NEXT W9
5040 PRINT W$;" : G";
5050 INPUT W$
5060 RETURN
```

The second one performs a YES/NO question input.

```
200 W$="QUESTION......"
210 GOSUB 5000
220 GOSUB 5070
230 END

5000 REM Question Subroutine
5010 FOR W9=1 TO 8
5020 PRINT W$;" : K"
5030 NEXT W9
5040 PRINT W$;" : G";
5050 INPUT W$
5060 RETURN
5070 REM Yes/No Subroutine
5080 U=POS("YESNO",W$,1)
5090 IF U THEN 5130
5100 W$="Y(ES) OR N(O) PLEASE"
5110 GOSUB 5000
5120 GO TO 5080
5130 U=U=1
5140 RETURN
```

Changing a 4052/4054 File Header

by F.G.G. Voigts
    National Physical Re-
    search Laboratory
    Pretoria, South Africa

The program given on page 7-42 (Rev. A, Mar 1979) of the Reference Manual works well on the 4051. The 4052 and 4054 on the other hand require a CLOSE statement (or implied CLOSE, e.g.

FIND) before the PRINT @33,0:0,0,0. Therefore, the program should read:

```
100 INIT
120 PRINT @33,0:0,0,1
130 FIND 2
140 INPUT @33:A$
150 A$=REP("MATH",28,4)
160 FIND 2
170 PRINT @33:A$
180 PRINT @33:"S"
185 CLOSE
190 PRINT @33,0:0,0,0
200 TLIST
210 END
```

Modifications to Data Graphing to Redraw Lines

by Victoria Skwiertz
   Union National Bank
   Pittsburgh, PA

The following modifications to the program, "Data Graphing" will redraw the lines/dashes on the plotter in line graph charts. These darker, heavier lines stand out more and are easier to read, which is particularly useful for graphs that are reproduced or drawn for overhead transparencies.

The lines are drawn darker and heavier under the control of two variables:

V2 counts the number of times the line is redrawn. It is increased to 4 but can be changed to a higher number for even darker lines.

V3 is the symbol for the change in the pen position. It is incremented by 0.1 with each iteration. A heavier line could be drawn by changing this value.

I have worked with several different versions of "Data Graphing" and the section of the program that draws the lines is similar so these modifications could be added if your version resembles the listing.

In this particular version, "Data Graphing for the 4051/2/4" the following instructions are inserted after the 'CHANGE PEN COLOR IF DESIRED' routine. The line numbers reference this version.

```
                                          4090 IF Z=32 THEN 4140
                                          4100 IF F$="N" THEN 4140

(initializes pen moving increment)        4140 REM MODIFICATIONS-REDRAWS LINES ON PLOTTER
                                          4141 V3=0

(skips redrawing on screen)               4142 IF Z=32 THEN 4150

                                          4143 REM TEST FOR BAR CHART
                                          4144 IF B2(K)=1 OR B2(K)=2 THEN 4150

                                          4146 REM V2=INCREMENT TO MOVE PEN;V3=COUNTER FOR #REDRAWS
                                          4148 FOR V2=1 TO 4

(add pen moving increment to pen position) 4232 R=R+V3

                                              After instruction 4850 insert

(skip repeating if screen)                4852 IF Z=32 THEN 4860

(skip if bar charts)                      4853 IF B2(K)=1 OR B2(K)=2 THEN 4860

(adds 0.1 to move pen position up)        4854 V3=V3+0.1

(goes back to redraw lines)               4856 NEXT V2
```

```
4060 VIEWPORT 20,120,20,90
4070 WINDOW 20,120,20,90
4080 FOR K=1 TO N1
4090 IF Z=32 THEN 4140
4100 IF F$="N" THEN 4140
4110 PRINT "J CHANGE PEN COLOR IF DESIRED, THEN PRESS CR";
4120 PRINT @Z:"GGGG";
4130 INPUT X$
4140 REM MODIFICATIONS-REDRAWS LINES ON PLOTTER (4232,4852)5/13/81 US
4141 V3=0
4142 IF Z=32 THEN 4150
4143 REM TEST FOR BAR CHART
4144 IF B2(K)=1 OR B2(K)=2 THEN 4150
4146 REM V2=INCREMENT TO MOVE PEN;V3=COUNTER FOR # REDRAWS-US
4148 FOR V2=1 TO 4
4150 X5=0
4152 Y5=0
4160 M4=1
4170 E0=0
4180 FOR I=1 TO H
4190 IF Y(I,K)<>8.988465674E+307 THEN 4220
4200 E0=1
4210 GO TO 4850
4220 R=20+70*(Y(I,K)-L2)/S4
4230 C=S3+S2*(I-1)
4232 R=R+V3
4240 IF D4(K)=0 THEN 4850
4250 IF B2(K)=1 OR B2(K)=2 THEN 4390
4260 IF E0=1 THEN 4200
4270 IF I(>1 THEN 4320
4280 PRINT @Z,21:C,R
4290 E0=0
4300 C1=C
4310 R1=R
4320 IF D4(K)=0 THEN 4370
4330 GOSUB 6050
4340 C1=C
4350 R1=R
4360 GO TO 4850
4370 PRINT @Z,21:C,R
4380 GO TO 4850
4390 IF E0=0 THEN 4420
4400 E0=0
4410 GO TO 4850
4420 IF N9=0 THEN 4490
4430 W2=(W1*2/N9-0.5)/2
4440 C2=C-W1+U1+W8*U1*2
4450 IF B2(K)=1 THEN 4510
4460 L3(I)=L9
4470 H7(I)=L9
4480 GO TO 4510
4490 C2=C
4500 W2=W1
4510 IF R<L9 THEN 4590
4520 R=R-L9
4530 PRINT @Z,21:C2-W2,L3(I)
4540 PRINT @Z,20:C2-W2,R+L3(I),C2+W2,R+L3(I),C2+W2,L3(I),C2-W2,L3(I)
4550 L6(1)=L3(I)
4560 L3(I)=L3(I)+R
4570 L6(2)=L3(I)
4590 GO TO 4650
4590 R=L9-R
4600 PRINT @Z,21:C2-W2,H7(I)
4610 PRINT @Z,20:C2-W2,H7(I)-R,C2+W2,H7(I)-R,C2+W2,H7(I),C2-W2,H7(I)
4620 L6(1)=H7(I)
4630 H7(I)=H7(I)-R
4640 L6(2)=H7(I)
4650 IF D(K)<0 THEN 4760
4660 IF D(K)=0 THEN 4850
4670 D1=W2/INT(D(K)/2)
4680 J=D1
4690 PRINT @Z,21:C2-W2+J,L6(1)
4700 FOR J=D1 TO 2*W2 STEP 2*D1
4710 PRINT @Z,20:C2-W2+J,L6(1),C2-W2+J,L6(2)
4720 IF C2-W2+J+D1>C2+W2 THEN 4820
4730 PRINT @Z,20:C2-W2+J+D1,L6(2),C2-W2+J+D1,L6(1)
4740 NEXT J
4750 GO TO 4850
4760 D1=10/-D(K)
4770 J=D1
4780 PRINT @Z,21:C2-W2,L6(1)+J
4790 FOR J=L6(1)+D1 TO L6(2) STEP 2*D1
4800 IF J>L6(2) THEN 4840
4810 PRINT @Z,20:C2-W2,J,C2+W2,J
4820 IF D1+J>L6(2) THEN 4840
4830 PRINT @Z,20:C2+W2,D1+J,C2-W2,D1+J
4840 NEXT J
4850 NEXT I
4852 IF Z=32 THEN 4860
4853 IF B2(K)=1 OR B2(K)=2 THEN 4860
4854 V3=V3+0.1
4856 NEXT V2
4860 IF K>7 THEN 4890
4870 IF B2(K+1)<>2 THEN 4890
4880 H8=H8+1
```

Editor's Note: The current version of "Data Graphing" being distributed by the 4050 Applications Library requires that the following statements be changed:

```
                        3250 VIEWPORT 20,120,20,90
                        3260 WINDOW 20,90,20,90
                        3280 FOR K=1 TO N1
                        3285 GOSUB 18000
Change Statement ——     3290 IF Z=32 THEN 3321
Change Statement ——     3295 IF F$="N" THEN 3321
                        3300 PRINT "J CHANGE PEN COLOR IF DESIRED, THEN PRESS CR";
                        3310 PRINT @Z:"GGGG";
                        3320 INPUT X$
Change Statement ——     3321 REM *** Mods for redrawing lines on plotter
Add Statement ——        3322 V3=0
Add Statement ——        3323 IF Z=32 THEN 3330
Add Statement ——        3324 REM Test for bar chart
Add Statement ——        3325 IF B2(K)=1 OR B2(K)=2 THEN 3330
Add Statement ——        3326 REM V2=Increment to move pen; V3=Counter for # Redraws
Add Statement ——        3327 FOR V2=1 TO 4
                        3330 X5=0
                        3340 Y5=0
                        3350 M4=1
                        3370 IF N4(K)=1 OR N4(K)=6 THEN 3430
                        3380 GOSUB N4(K)-1 OF 9000,9100,9220,10500
                        3430 FOR I=X1 TO X2
                        3450 IF Y(I,K)<-1.0E+306 THEN 4050
                        3460 R=19.6+70*(Y(I,K)-L1)/(U-L1)
                        3470 C=S3+S2*(I-X1)
Add Statement ——        3471 R=R+V3
                        3490 IF D4(K)=7 THEN 4050
                        3500 IF B2(K)=1 OR B2(K)=2 THEN 3640

                        4000 PRINT @Z,20:C2+W2,V9+D1+J,C2-W2,V9+D1+J
                        4020 NEXT J
                        4050 NEXT I
Add Statement ——        4051 IF Z=32 THEN 4060
Add Statement ——        4052 IF B2(K)=1 OR B2(K)=2 THEN 4860
Add Statement ——        4053 V3=V3+0.1
Add Statement ——        4054 NEXT V2
                        4060 IF K>5 THEN 4090
                        4070 IF B2(K+1)<>2 THEN 4090
                        4080 N8=N8+1
                        4090 GOSUB 8000
                        4100 NEXT K
```
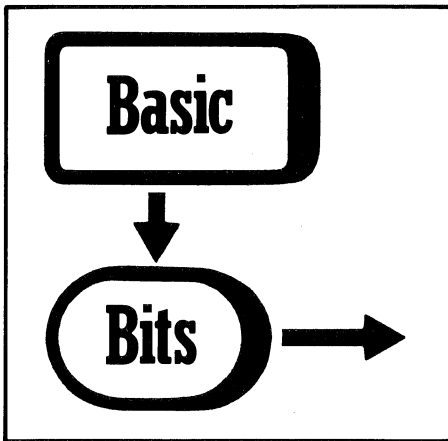
Determining Dimensions of an Unknown Array

by Dick Browne
  Tektronix, Inc.
  Philadelphia, PA

Did you ever dimension an array and then forget how large it was? Here is a way to find out without changing the contents of the array. Assuming A is the unknown array, write this program using some unused program steps:

```
5000 FOR I=1 TO 10000
5010 FOR J=1 TO 10000
5020 IF A(I,J)=0 THEN 5030
5030 NEXT J
5040 NEXT I
```

Running the program will give you the following error:

DIMENSION-SUBSCRIPT ERROR IN LINE 5020 — MESSAGE NUMBER 10

Enter J-1. The value displayed is the second dimension. Change step 5010 as follows:

```
5010 FOR J=1 TO 1
```

Now run the program again. When the error message appears, enter I-1. The number displayed will be the first dimension.

If the array only has one dimension, use the following program. The value of I-1 is the dimension.

```
5000 FOR I=1 TO 10000
5010 IF A(I)=0 THEN 5020
5020 NEXT I
```

Common Input Variable

by Bryan Burma
  Tektronix, Inc.
  Kansas City, KS

When writing a program with a lot of interactive input, it is a good idea to always use the same input variable name and to use a string variable. Using a variable such as Q$ to denote an input variable or a query to the keyboard has several benefits:

Data validity checking is much easier since the string variable can easily be checked for either alpha, numeric, or alphanumeric validity. After the validity checks have been performed, the common input variable can always be reassigned to the appropriate variable type.

Using a common input variable, you can reduce the number of variable used in a program.

Using an alpha variable (string) will aid in preventing program crashes. A string variable will accept any keyboard input. If the data is not correct, the program can flag the operator.

Editor's Note: See the programming tip ''PRINT USING and Forms Design'' in this issue, which discusses input error checking.

Tape Index

by Ron Robinder
  Tektronix, Inc.
  Beaverton, OR

Following is the program I place on file 1 on all my personal tapes. It lists the names of all programs on the tape quickly, then locates and reads in the one desired.

As given, the program will read up to 275 files in pages of 25 each. In practice, you seldom require more than two or three pages for most practical tapes. The first line in each program file must be a REM statement which gives the title of the file. Fifty characters starting after the space will be printed if a two, three, or four digit line number is used. If the first line is not a REM statement, the entire line will be printed.

For non-program files, the program uses the header information to list the file type in place of the title. The only characters acceptable in position 9 of a header are: A, B, D, L, N, P or T. Any other character will cause an error message to be printed.

The principle advantages of this program over using a TLIST program are:

1. No need to modify the header and risk damaging it.

2. On a 4051, the tape does not auto rewind at the end, greatly speeding location of files near the end of the tape.

The output option sends the tape index to a plotter to make tape labels, or to a printer to list rapidly the contents. Note that a carriage return is needed in line 460 to print the next page.

```
1 REM INDEX FOR TAPE #1, DATA HANDLING
4 RUN 100
100 INIT
110 PRINT "LINPUT NUMBER FOR OUTPUT DEVICE (32=SCREEN): ";
120 INPUT D
130 I$=" "
140 DIM A$(50)
150 FOR J=1 TO 1
160 FOR I0=1 TO 5
170 PAGE @D:
180 PRINT @D:"INDEX OF PROGRAMS ON TAPE."
190 PRINT @D:"FILE      EXTENDED TITLE(LINE 1)"
200 FOR I=1 TO 25
205 PRINT @33;0:0,0,1
210 FIND I+5%(I0-1)
220 INPUT @33:I$
225 PRINT @33;0:0,0,0
227 FIND I+25%(I0-1)
230 H$=SEG(I$,9,1)
240 IF H$="A" THEN 280
250 IF H$="B" THEN 400
260 IF H$="L" THEN 480
270 IF H$="N" THEN 420
280 H$=SEG(I$,17,1)
290 IF H$="P" THEN 350
300 IF H$<>"T" AND H$<>"D" THEN 330
310 PRINT @D:I+25%(I0-1),"DATA OR TEST FILE"
320 GO TO 450
330 PRINT @D:"ERROR IN FILE ";I+25%(I0-1)
340 GO TO 450
350 INPUT @33:A$
360 A=POS(A$,"REM ",1)
370 A$=SEG(A$,A+4,LEN(A$))
380 PRINT @D:I+25%(I0-1),A$

390 GO TO 450
400 PRINT @D:I+25%(I0-1),"BINARY FILE"
410 GO TO 450
420 H$=SEG(I$,35,6)
430 H=VAL(H$)
440 PRINT @D:I+25%(I0-1),"NEW FILE; LENGTH = ";H; " BYTES"
450 NEXT I
460 INPUT T$
470 NEXT I0
480 PRINT @D:I+25%(I0-1),"LAST FILE"
490 NEXT J
500 PRINT "INPUT FILE YOU WANT ";
510 INPUT F
520 IF F=0 THEN 550
530 FIND F
540 OLD
550 END
```

PRINT USING and Forms Design

by Dick Browne
    Tektronix, Inc.
    Philadelphia, PA

The PRINT USING command is one of the fastest and most efficient ways to draw multiple symbols and forms. Here is a review of the proper use of this command.

The command:

```
100 PRINT USING "P35(72""8"")""±""S";
```

prints a page full of 8's. Let's analyze the format.

The "P" causes the screen to page. The "35" prints 35 lines, and the "72" prints 72 symbols per line. Inside the quotation marks is the symbol to be printed. The next symbol to be printed is the HOME symbol (Control-T), which returns the cursor to the home position. Next the "S" suppresses the carriage return. Without the "S", the cursor would move down to the second line.

```
100 PRINT USING "P(72""*"")/33(""**""71T""**""/)(72""*"")""±""S";
```

Again, the "P" pages the screen. The contents of the first parenthesis draw 72 *'s on the first line. Next a CR/LF (/), moves the cursor to the next line before continuing. The contents of the next parenthesis will be repeated 33 times. First two *'s will be printed; then the cursor will tab to space 71 and print two more *'s, then it generates a line feed. The contents of the last parenthesis prints the last line. The cursor then returns to the HOME position.

```
¦ TECHNICIAN- JOE TECH          A/P: 101         ¦
¦                                                ¦
¦PRODUCT GROUP ¦A/C¦ACTUAL HRS¦BILLED HRS¦TAT¦OB?¦PWF TOTAL ¦
¦--------------+---+----------+----------+---+---+----------¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦              ¦   ¦          ¦          ¦   ¦   ¦          ¦
¦TO BACKSPACE, PRESS RETURN WITH NO ENTRY, TYPE 'END' AFTER LAST ENTRY,¦
¦TO DELETE A LINE, ENTER 'DELETE' AND RETYPE INCORRECT LINE.           ¦
```

The above form can be drawn with the following program:

```
100 INIT
110 O$="JOE TECH"
120 S$="101"
130 X=0
140 IMAGE P2T,11A,14T,FA,46T,5A,52T,3A,3/,T,14A,17T,3A,S
150 PRINT USING 140:"TECHNICIAN-",O$,"A/P:",S$,"PRODUCT GROUP","A/C"
160 IMAGE 2T,11A,13T,12A,26T,3A,30T,3A,34T,9A
170 PRINT USING 160:"ACTUAL HRS","BILLED HOURS","TAT","OB?","PWF TOTAL"
180 PRINT USING """±""*/4/71(""-""/)/"KK"30(16T""±""/)""±""2/";
190 PRINT USING "30(20T""±""31T""±""44T""±""48T""±""52T""±""/)S";
200 PRINT USING "FAS";"TO BACKSPACE, PRESS RETURN WITH NO "
210 PRINT "ENTRY, TYPE "END" AFTER LAST ENTRY."
220 PRINT USING "FAS";"TO DELETE A LINE, ENTER ""DELETE"" AND "
230 PRINT USING "FA""±""4/";"RETYPE INCORRECT LINE."
240 X=X+1
350 PRINT USING "17TAS";:"K"
360 INPUT B$
450 PRINT USING "21TAS";:"K"
460 INPUT C$
550 PRINT USING "32TAS";:"K"
560 INPUT D$
650 PRINT USING "45TAS";:"K"
660 INPUT E$
750 PRINT USING "49TAS";:"K"
760 INPUT F$
850 PRINT USING "53TAS";:"K"
860 INPUT G$
950 IF X=>28 THEN 130
960 GO TO 240
```

Steps 140-170 position and print the words. Step 180 homes the cursor, line feeds down four spaces, prints 71 dashes, carriage returns, then prints a column of 30 vertical lines, tabbed in 16 spaces.

The cursor is then returned to the HOME position and line fed down two lines to prepare for the next line.

Line 190 prints five vertical columns of 30 vertical lines, tabbed in 20, 31, 44, 48, and 52 spaces, respectively. After the last (30th) line is printed, there would actually be two line feeds, one included in the command, and the other which is automatically generated at the end of a print command unless suppressed. The "S" suppresses the extra carriage return.

Lines 200-230 print the instructions at the bottom of the form. Line 230, after finishing the printing, homes the cursor and line feeds four lines to place the cursor in position for the first input.

Line 240 is a line counter. After every 28 lines, the form is redrawn.

String variables are used instead of numeric variables in the input command to allow checking for legitimate input. For instance, let's say you want to limit the entries in the product group column to three categories: "car," "truck," and "bus." Elsewhere in the program you could define a string:

```
H$="CAR  TRUCKBUS "
```

Then when A$ is entered (line 250), the next step would be:

```
260 IF LEN(A$)<3 THEN 1120
270 P=POS(H$,A$,1)
```

The routine starting at line 1120 would ring a bell, increment the line counter and return you to the first column:

```
1120 PRINT "GG";
1130 X=X+1
1140 GO TO 950
```

If the correct category were entered, P would be equal to 1, 6, or 11. If not, it would be 0, or some other number between 1 and 15. To determine if the category is correct, we'll add 4 to P and see if the resulting number divided by 5 is a whole number:

```
280 P=(P+4)/5
290 IF P=INT(P) AND P<>0 THEN 350
300 PRINT "GGINCORRECT CATEGORY"
310 X=X+1
320 GO TO 250
```

B$ is the activity code, (line 360). If you want to make sure it is a number you could add:

```
370 IF B$="" OR B$<"0" OR B$>"9" THEN 1120
```

Using similar techniques, you can check for appropriate entries in any column, whether the entry should be a number, a letter, one or two particular letters, or whatever.

If fewer columns are required, a form can be drawn with instructions down the right side:

```
¦-----------------------------------------------------------------------¦
¦       TURNAROUND TIME REPORT FOR A/P   101                            ¦
¦                                                                       ¦
¦CATEGORY  ¦QTY ¦DAYS   ¦PARTS QTY OR 'N'                               ¦
¦----------+----+-------+------------    ENTER CATEGORIES               ¦
¦          ¦    ¦       ¦                AS FOLLOWS:                     ¦
¦          ¦    ¦       ¦                                               ¦
¦          ¦    ¦       ¦                W = WARRANTY                    ¦
¦          ¦    ¦       ¦                C = CONTRACT                    ¦
¦          ¦    ¦       ¦                M = MEDICAL                     ¦
¦          ¦    ¦       ¦                F = FDI                         ¦
¦          ¦    ¦       ¦                B = CODE 08'S                   ¦
¦          ¦    ¦       ¦                A = ALL OTHERS                  ¦
¦          ¦    ¦       ¦                                               ¦
¦          ¦    ¦       ¦                ENTER THE NO. OF INSTS,THAT     ¦
¦          ¦    ¦       ¦                PARTS WERE ORDERED FOR OR       ¦
¦          ¦    ¦       ¦                'N' FOR 'NO' IN THE PARTS       ¦
¦          ¦    ¦       ¦                ORDERED COLUMN.                 ¦
¦          ¦    ¦       ¦                                               ¦
¦          ¦    ¦       ¦                IF YOU HEAR THE BELL RING,      ¦
¦          ¦    ¦       ¦                AN INAPPROPRIATE ENTRY WAS      ¦
¦          ¦    ¦       ¦                MADE AND THE LINE WILL BE       ¦
¦          ¦    ¦       ¦                IGNORED.  START AGAIN AT THE    ¦
¦          ¦    ¦       ¦                BEGINNING OF THE LINE.          ¦
¦          ¦    ¦       ¦                                               ¦
¦          ¦    ¦       ¦                ENTER '0' TO END PROGRAM.       ¦
¦          ¦    ¦       ¦                                               ¦
¦          ¦    ¦       ¦                                               ¦
¦-----------------------------------------------------------------------¦
```

This form is generated using the following program:

```
100 INIT
120 H$="101"
130 U=0
140 IMAGE P10TFAFA
150 IMAGE 2/T8A12T3A17T4A25T16A
160 IMAGE 43TFA
170 PRINT USING 140:"TURN AROUND TIME REPORT FOR A/P: "H$
180 PRINT USING 150:"CATEGORY","QTY","DAYS","PARTS QTY OR ""N"""
190 PRINT USING 160:"ENTER CATEGORIES"
200 PRINT USING 160:"AS FOLLOWS:"
210 PRINT USING 160:"W = WARRANTY"
220 PRINT USING 160:"C = CONTRACTS"
230 PRINT USING 160:"M = MEDICAL"
240 PRINT USING 160:"F = FDI"
250 PRINT USING 160:"B = CODE 08'S"
260 PRINT USING 160:"A = ALL OTHERS"
270 PRINT USING 160:"ENTER THE NO. OF INSTS. THAT"
280 PRINT USING 160:"PARTS WERE ORDERED FOR OR"
290 PRINT USING 160:""N"" FOR 'NO' IN THE PARTS"
300 PRINT USING 160:"ORDERED COLUMN."
310 PRINT USING 160:"IF YOU HEAR THE BELL RING,"
320 PRINT USING 160:"AN INAPPROPRIATE ENTRY WAS"
330 PRINT USING 160:"MADE AND THE LINE WILL BE"
340 PRINT USING 160:"IGNORED. START AGAIN AT THE"
350 PRINT USING 160:"BEGINNING OF THE LINE."
360 PRINT USING 160:"ENTER '0' TO END PROGRAM."
370 PRINT USING """±""4/40(""-"")""±""2/";
380 PRINT USING "30(11T""±""16T""±""24T""±""/)""±""4/";
390 IF U=>28 THEN 130
400 U=U+1
410 INPUT A$
540 PRINT USING "12TAS";:"K"
550 INPUT X$
610 PRINT USING "17TAS";:"K"
620 INPUT D$
700 PRINT USING "25TAS";:"K"
710 INPUT P$
860 GO TO 430
```

The main difference between this program and the previous one is the IMAGE 43TFA statement. It can be used repeatedly to print a column of any length starting at a point other than the left margin. Changing the number before the "T" changes the starting point of the column. 🖾

4050 Series Applications Library Programs

Ordering

Programs included in the Applications Library prior to September 1981 are packaged and nomenclated by function. Those programs accepted into the Library after September 1981 are packaged and nomenclated with the Volume and Number of the corresponding issue of TEKniques in which the package was announced.

Each package includes the source code on tape or disk (T=tape ; D=disk) together with the supporting documentation; listings are not included in the documentation. Documentation may be purchased separately.

The 4050 Series Applications Library Programs catalog (September 1981) contains the abstracts describing the programs in each package along with representative output in most cases. The catalog part number is 062-6343-00.

To receive a copy of the catalog, or to order a package, contact your local Tektronix field office. The field office has the current prices.

Program Contributions

Contribute one program to the Applications Library and receive the package of your choice in exchange. Send in the membership card from your 4050 Series Graphic System Reference Manual to get the details. Or call us at (503) 685-3618.

Outside U.S.

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

TEKniques Vol. 5 No. 4 T1

TEKniques Vol. 5 No. 4 T1 tape consists of 67 programs: 10 programming aid routines, four 3-D graphing programs, five 2-D graphing programs, one index program for literature references, and 47 programs using the 4050 as a controller for TM5000 instruments.

The programming aid routines maintain a tape index, label file headers, cross reference ASCII programs and include code for a variety of routines useful in programs.

The graphing programs are the entries in the Graphing Contest sponsored by TEKniques in the Spring of 1981.

The Reference Index program creates and maintains a list of literature references. It must be transferred to a separate dedicated tape.

As a supplement to the GPIB Programming Guide (see related article in this issue of TEKniques), 47 4050/TM5000 instrument controller examples and utilities have been included in this tape, along with their index. Having the machine code available through the Applications Library saves users from keying in the code printed in the Guide. These programs must be transferred to a tape where they occupy the first 48 files.

The following individual abstracts describe each program's function. Documentation for programs affected contain complete instructions for accomplishing file transfers.

Program 1

Title: TAPEMENU
Author: Bob Manthey
County of San Diego
San Diego, CA
Memory Requirement: 16K
Peripherals: Optional-4924 Tape Drive
Files: 1 ASCII Program
1 Binary Data
Statements: 188

TAPEMENU will establish, maintain and display a file of the names and file numbers of all ASCII Program files on tape. When a program is selected, TAPEMENU will load and run it.

You may also number and label a tape cartridge using TAPEMENU. The filenames are previously created with FMARKER (Program 4 below).

Storage capacity of file names is currently 100, but may be easily changed.

```
     T B A S I C   P R O G R A M   L I B R A R Y   O F
      = TEKTRONIX APPLICATIONS LIBRARY PROGRAMS =
                      09-22-81
        Tape CARTRIDGE:  #VOL5H04T1  @ Device #33

=================================================================
  1- TAPEMENU   2- ReadNote   3- WritNote   4- FMarker    5- LHeader
  6- PHeader    7- Stripper   8- CrossRef   9- TapeDupe  10- CompSort
 11- TAPEME:S  12- ReadNo:S  13- WritNo:S  14- FMarke:S  15- LHeade:S
 16- PHeade:S  17- Stripp:S  18- CrossR:S  19- TapeDu:S  20- CompSo:S
 21- TAPEME:R  22- ReadNo:R  23- WritNo:R  24- FMarke:R  25- LHeade:R
 26- PHeade:R  27- Stripp:R  28- CrossR:R  29- TapeDu:R  30- CompSo:R
 31- Z=F(X,Y)  32- StereoPt  33- H/WoHdnL  34- PlotThrD  35- Append
 36- Append   '37- PipeChrt  38- Contours  39- Hierarcl  40- Nonlinear
 41- Vertical  42- RefIndex
=================================================================
              Input 'PROGRAM' Option  >
```

Program 2

Title: READNOTE

Author: Bob Manthey
     County of San Diego
     San Diego, CA
Memory Requirement: 8K
Files: 1 ASCII Program
Statements: 37

READNOTE is a short utility routine that displays a data file created by WRITNOTE (see abstract below).

However, the core of this routine and its methods may be used as a subroutine within a complex program, allowing it to display instructions that would be prohibitively large to code into the program.

```
          G E N E R A L   P R O G R A M   N O T E S :

1. TAPEMENU -
    This program will establish, maintain, and display a binary data-
    file containing the names and file numbers of all ASCII encoded
    programs contained on the tape.  (The names are those previously
    created with the 'FMARKER' utility.)  Upon selecting a program by its
    relative display-number, the program will load-and-run it.

                  (Press 'RETURN' to Continue!)
```

Program 3

Title: WRITNOTE

Author: Bob Manthey
     County of San Diego
     San Diego, CA
Memory Requirement: 8K
Peripherals: Optional-4924 Tape Drive
Files: 1 ASCII Program
Statements: 37

This is a quick utility which creates and edits notes in a data file. It is line oriented.

Once the notes are created and stored, they may be modified a line at a time. Modifications include:
    Leaving a line as is
    Changing a line
    Inserting a line
    Deleting a line

The program will automatically MARK a file large enough for the newly created or modified text and store it.

```
          G E N E R A L   P R O G R A M   N O T E S :

             LINE-EDIT MODE:
             1- Modify, 2- Insert, 3- Delete, 4- End  >
1. TAPEMENU -
             LINE-EDIT MODE:
             1- Modify, 2- Insert, 3- Delete, 4- End  >
  This program will establish, maintain, and display a binary data-
             LINE-EDIT MODE:
             1- Modify, 2- Insert, 3- Delete, 4- End  >
  file containing the names and file numbers of all ASCII encoded
             LINE-EDIT MODE:
             1- Modify, 2- Insert, 3- Delete, 4- End  >1
    Enter LINE   (Start LAST line with 'Control-C')
-------------------------------------------------------------------
file containing the names and file numbers of all ASCII
                   (Is this CORRECT!) yes
  programs contained on the tape.  (The names are those previously
             LINE-EDIT MODE:
             1- Modify, 2- Insert, 3- Delete, 4- End  >
```

Program 4

Title: FMARKER

Author: Bob Manthey
     County of San Diego
     San Diego, CA
Memory Requirement: 8K
Peripherals: Optional-4924 Tape Drive
Files: 1 ASCII Program
Statements: 255

Use this routine to mark a tape file header with an 8 character name and also a 30 character Remarks-Extender.

In addition, the tape cartridge may be given a 10 character alpha-numeric identifier and a 48 character name.

A listing facility included in this routine will produce an index of the tape files, and compute and display the file space — used and free, in both a byte and block format, with their respective percentages. (This is for a standard DC300A cartridge only!)

```
                    TEKTRONIX APPLICATIONS LIBRARY PROGRAMS
                      TAPEFILE INDEX   VOL5H04T: Page- 1
No.    File Type      Marker      Size          Remarks
========================================================================
 1    ASCII  PROGRAM  TAPEMENU  40     ProgramFile Menu with Chaining
 2    BINARY DATA     MenuDat   5      ProgramFile Numbers-&-Names
 3    ASCII  PROGRAM  ReadNote  6      ASCII Data-Note Reading Routin
 4    ASCII  PROGRAM  WritNote  12     ASCII Data-Note Writing Routin
 5    ASCII  PROGRAM  FMarker   33     Tape-File Header Marker-Extndr
 6    ASCII  PROGRAM  LHeader   12     File-Header Library SubRoutine
 7    ASCII  PROGRAM  PHeader   6      Standardized Program Header
 8    ASCII  PROGRAM  Stripper  36     Program 'REMARK's Stripper
 9    ASCII  PROGRAM  CrossRef  52     Cross-Reference ASCII Programs
10    ASCII  PROGRAM  TapeDupe  45     4924 Tape-to-Tape Copy Utility
11    ASCII  PROGRAM  CompSort  32     Sample Bubl-Sort & Shel-Sort
12    ASCII  DATA     PROGNOTE  15     General Library-Programs Notes
13    ASCII  PROGRAM  TAPEME:S  15     STRP: ProgramFile Menu with Ch
14    ASCII  PROGRAM  ReadNo:S  4      STRP: ASCII Data-Note Reading
15    ASCII  PROGRAM  WritNo:S  8      STRP: ASCII Data-Note Writing
16    ASCII  PROGRAM  FMarke:S  22     STRP: Tape-File Header Marker-
17    ASCII  PROGRAM  LHeade:S  10     STRP: File-Header Library SubR
18    ASCII  PROGRAM  PHeade:S  6      STRP: Standardized Program Hea
19    ASCII  PROGRAM  Stripp:S  19     STRP: Program 'REMARK's Stripp
20    ASCII  PROGRAM  CrossR:S  28     STRP: Cross-Reference ASCII Pr
21    ASCII  PROGRAM  TapeDu:S  32     STRP: 4924 Tape-to-Tape Copy U
22    ASCII  PROGRAM  CompSo:S  17     STRP: Sample Bubl-Sort & Shel-
23    ASCII  PROGRAM  TAPEME:R  3      XREF: ProgramFile Menu with Ch
24    ASCII  PROGRAM  ReadNo:R  3      XREF: ASCII Data-Note Reading
25    ASCII  PROGRAM  WritNo:R  3      XREF: ASCII Data-Note Writing
========================================================================
     File Space USED:   510 blocks    (130,309 bytes) 42.6%
     File Space FREE:   685 blocks    (175,611 bytes) 57.4%
```

Program 5

Title: LHEADER

Author: Bob Manthey
     County of San Diego
     San Diego, CA
Memory Requirement: 8K
Files: 1 ASCII Program
Statements: 77

The FMARKER utility writes an eight character marker beginning at column 25 of the file header. Unfortunately whenever you write/save data to a file, the 4051 writes over columns 25 and 26 with spaces.

You may alleviate this problem by incorporating the two routines contained in this file into your write/save programs.

Program 6

Title: PHEADER

Author: Bob Manthey
     County of San Diego
     San Diego, CA
Memory Requirement: 8K
Files: 1 ASCII Program
Statements: 38

This file contains an example of how the tape-cartridge identifer created with FMARKER might be used as part of the BASIC program identifier line.

Program 7

Title: STRIPPER

Author: Bob Manthey
      County of San Diego
      San Diego, CA
Memory Requirement: 32K
Peripherals: Optional-4924 Tape Drive
Files: 1 ASCII Program
Statements: 260

This program strips the 'REMarks' from an ASCII Program file. One of three stripping modes is available:
1. Delete REMark commentary
2. Delete REMark line
3. Delete REMark line and RENumber the following line

The stripped program is saved and labeled as a separate ASCII Program file. The original program remains intact.

```
100 REM ====== INTRINSIC INITIALIZATION & GLOSSARY ====================
110 INIT
120 REM ------ Numeric Variables/Constants -----------------------------
122 REM D  = Device-Drive Number  (2= 4924, 33= 4051)
124 LET D=33
126 REM F  = File Number
128 LET F=2
130 REM P  = Program Index Number
132 LET P=0
134 REM T  = Total Number of Programs
136 LET T=100
140 REM ------ String Variables/Constants ------------------------------
142 REM C$ = Tape-File Header-Delimiter  (Carriage Return)
144 LET C$=CHR(13)
146 REM D$ = Date of Latest File Activity
148 DIM D$(8)
149 LET D$=" "
150 REM E$ = Tape-File Header-Extender
152 DIM E$(210)
154 REM F$ = Composite File-Number-&-Name String
156 DIM F$(T&11)
158 REM H$ = Tape-File Header
160 DIM H$(42)
162 REM M$ = Tape-File Header-Marker
164 DIM M$(8)
166 REM N$ = Tape-File Number
168 DIM N$(3)
170 REM P$ = Program-File [Number-&-]Name
172 DIM P$(11)
174 REM Q$ = Tape-Cartridge Identifier
176 DIM Q$(10)
177 LET Q$=" "
178 REM R$ = Tape-Cartridge Name
180 DIM R$(48)
```

```
100 INIT
120 D=33
126 F=2
130 P=0
134 T=100
140 C$=CHR(13)
146 DIM D$(8)
149 D$=" "
150 DIM E$(210)
154 DIM F$(T&11)
158 DIM H$(42)
162 DIM M$(8)
166 DIM N$(3)
170 DIM P$(11)
174 DIM Q$(10)
177 Q$=" "
178 DIM R$(48)
```

Program 8

Title: CROSSREF
Author: Bob Manthey
      County of San Diego
      San Diego, CA
Memory Requirement: 32K
Peripherals: Optional-4924 Tape Drive
Files: 1 ASCII Program
Statements: 404

This program will sequentially index a cross-referenced ASCII program. The cross-referencing may include six or 13 commands:

   OF, GO TO, GOSUB, THEN, RESTORE, & USING only

or all of the above plus

   DELETE, LIST, SAVE, APPEND, RENUMBER, CALL and RUN

The index is saved and labeled as an ASCII program or REMark statements. This file may be loaded later for display, or appended to the source file from which it is derived.

```
7000 REM ====== TAPEMENU CROSS-REFERENCE INDEX =====================
7010 REM Label:   Reference:
7020 REM   270    <-  240
7030 REM   300    <-  260, 650
7040 REM   320    <-  310
7050 REM   470    <-  450
7060 REM   500    <-  630
7070 REM   600    <-  530
7080 REM   700    <-  550
7090 REM   800    <-  220, 710, 870
7100 REM   850    <-  830
7110 REM   870    <-  850
7120 REM   900    <-  430, 720
7130 REM   970    <-  950
7140 REM  1000    <-  640, 2400
7150 REM  1300    <-  1140
7160 REM  1400    <-  1130
7170 REM  2000    <-  250
7180 REM  2400    <-  2100
7190 REM  3000    <-  1510
7200 REM  3700    <-  3500
7210 REM  4000    <-  270
7220 REM  5000    <-  2800, 3600
7230 REM  6000    <-  230, 1120, 2500, 3200, 5600
7240 REM  6500    <-  6100, 6150, 6420
```

Program 9

Title: TAPEDDUPE.

Author: Bob Manthey
      County of San Diego
      San Diego, CA
Memory Requirement: 16K
Peripherals: 4924 Tape Drive
Files: 1 ASCII Program
Statements: 320

This program transfers any type of program or data files from the 4050 internal tape drive to the 4924 tape drive. The duplicated file will retain the original header and extender.

The program will automatically MARK the new files, if desired.

Program 10

Title: COMPSORT
Author: Bob Manthey
      County of San Diego
      San Diego, CA
Memory Requirement: 16K
Files: 1 ASCII Program
Statements: 172

This sample program generates a list of random and ordered numbers. These numbers are then sorted by a Bubble-Sort routine which uses a floating window; followed by a Shell-Metzner Sort for comparison. Both routines may be used, as is, with only the array name and element variables changed to ones appropriate to your program.

The Bubble-Window sort is my transcription of a North Star BASIC routine written by Paul T. Brady, and published in the September 1980 issue of "BYTE" magazine.

The Shell-Metzner sort is my transcription of a routine originally transcribed into North Star Basic by Steven Fisher, La Mesa, CA, and made available by him to the San Diego North Star User Group Library.

The Bubble-Window routine is designed specifically to sort lists with only a few entries out of order, or to check a list quickly to ensure that all entries are ordered.

The Shell-Metzner technique, on the other hand, is reasonably efficient when used with moderate amounts of random sequential data need to be sorted.

Either routine should be easy to adapt to specific sorting configurations in a variety of programming languages, or dialects.

Program 11

Title: General Function Z= F(X,Y) Plot
Author: Dony Robert
      Brussels, Belgium
Memory Requirement: 32K
Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 452

This program draws two variable functions, $z = f(x,y)$, with hidden lines removed. The draw is made in a rectangular region so that $X1 \leqslant X \leqslant X2$ and $Y1 \leqslant Y \leqslant Y2$.

The function may be drawn on a block representing axes parallel to the three real axes X,Y,Z as well as the study intervals.

The user writes the function to be studied, for instance in the form:

2500 Z= $-8*EXP(-X*X-Y*Y)*(X+Y)$

As soon as the user types RUN (RETURN), the program asks the inputs necessary to do the calculations and prompts for the following options.

1. Any seeing angle. The eye always looks at the origin of the axis. The azimuth angle 01 can vary from 0° to 180°, and from 0° to −180°. The dip angle 02 can vary from 0° to .90° and from 0° to −90°.

2. The number of slices cutting the surface.

3. The number of joints on each slice.

4. The eventual file number to record the drawing in G.D.U.s that allows a fast redraw of the function later.

5. The true scale or a uniform scale.

6. One or two directions of cutting.

7. The drawing of the function with or without the axis block.
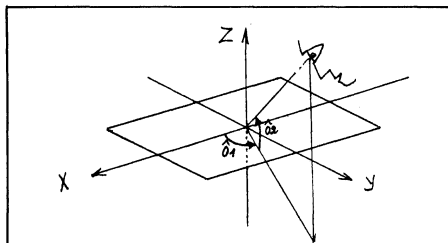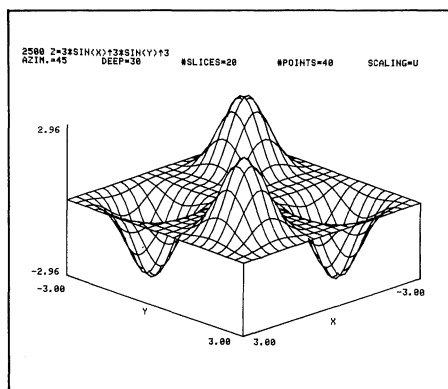
Projection of the eye on the XY plane

Program 12

Title: Stereo Surface
Author: Dr. P.R. Tregenza
      Department of Architecture
      University of Nottingham
      Nottingham, England
Memory Requirement: 4052/54 32K
Peripherals: Optional-4662/4663 Plotter
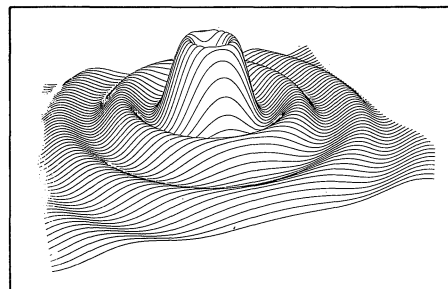Files: 1 ASCII Program
Statements: 200

An equation of the form $Z = F(X,Y)$ is plotted as a surface drawn in perspective with optional plotter output as a stereo pair.

The equation is contained in a subroutine which is listed initially on the screen.

The user may change the perspective viewpoint and specify the number of lines drawn. Hidden lines are removed from the image.

Functions must be continuous and injective.

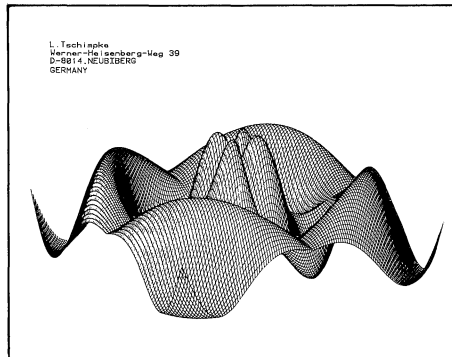Surfaces can be drawn from underneath.

When using the Hidden-Line Algorithm, the Z-axis has to be vertical.

The program automatically begins drawing in front and generates a pattern, which gives the information, whether next line can be drawn or not.

Program 13

Title: 3-D Plot w/wo Hidden Lines

Author: Lothar Tschimpke
       Munich, Germany
Memory Requirement: 16K
Peripherals: Optional-4662/4663 Plotter
Files: 1 ASCII Program
Statements: 272

This program plots a net or lines of a mathematical or programmable function of two variables, e.g., Z=F(X,Y). The functions can be viewed under any angle.


```
L.Tschimpke
Werner-Heisenberg-Weg 39
D-8014.NEUBIBERG
GERMANY
```

Program 14

Title: PLOT3D
Author: Robert Kennedy
       Elliot Noma
       Philadelphia, PA
Memory Requirement: 16K
Peripherals: Optional-4662 Plotter
Files: 3 ASCII Program
Statements: 226

PLOT3D is a plotting routine which is to be run as an appended subroutine to a main program.

PLOT3D receives vertical Z values for points on a line through the X-Y plane. The X,Y,Z coordinates for these points are rotated around the Z-axis and then around the X-axis. PLOT 3D then plots the line segments joining the rotated points, provided the segment appears above the hidden line horizon stored as a mask.

The mask is an array which spans the screen's horizontal axis containing the values of the maximum vertical values (in screen coordinates) for the particular horizontal value. The plotted point is checked against the mask by comparing vertical values for the point and the mask given by the horizontal values.

Once plotted or attempted to plot, the mask values for the horizontal values affected is updated.

After the line is plotted, control returns to the calling program and PLOT3D waits for the next line of Z values.

Program 15

Title: Pipechart
Author: Steven Salisbury
       Whirlpool Corporation
       Benton Harbor, MI
Memory Requirement: 16K
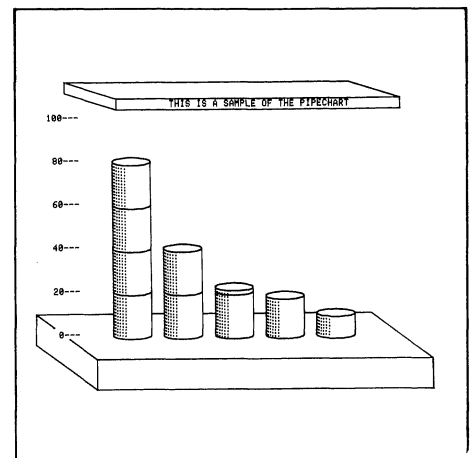Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 160

This program quickly provides a three dimensional bar chart with shading. Up to 12 bars may be displayed on the 4050 screen or the 4662 Plotter.

Data entered from the keyboard is displayed in the form of N cylindrical bars, where N is the number of bars selected by the user. The bars are drawn on a stand, and a title block is placed above the bars for a one line, 52 character title. Due to the intensity of the shading, the program does not shade bars when drawing on the plotter.

The Y-axis minimum is always set to zero. Only positive values may be plotted.

Once the data is entered, the width of the bars, maximum Y-value and plotting device may be changed through the User-Definable Keys and the plot redrawn.


```
THIS IS A SAMPLE OF THE PIPECHART
100---
80---
60---
40---
20---
0---
```

②

Program 16

Title: Contour Plots
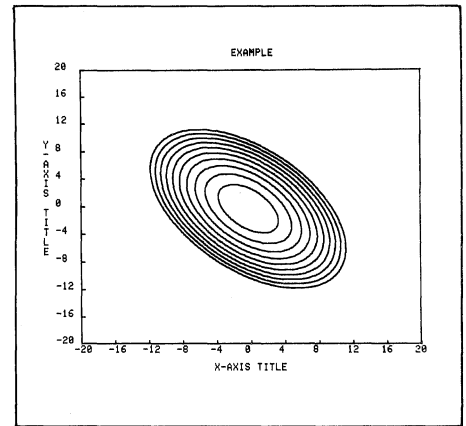Author: Jerry W. Anderson
      Bartlesville, OK
Memory Requirement: 8K
Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 149

This program plots response surfaces (or contour plots) in two dimensions. It uses the 3-dimensional full-quadratic model (or subsets thereof) as a function:

$$Z = Y0 + A*X + B*Y + (AB)*X*Y + (ASQ)*X{\uparrow}2 + (BSQ)*Y{\uparrow}2$$

where Z, X, Y are variables and Y0, A, B, AB, ASC, & BSQ are constants which may be known or determined by regression. Very high resolution plots may be generated. The user may determine the number of contours desired and their value.

In effect, we have displayed a three-dimensional data space in two dimensions, much as a topographical map displays geographical features.

---

③

Program 17

Title: Hierarchal Clustering
Author: Jerry W. Anderson
      Bartlesville, OK
Memory Requirement: 32K
Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 244

Consider a data set consisting of N vectors (or data points) each in L dimensions.

A sequence of classifications in which larger clusters (or groupings) are obtained through a merger of smaller ones is called a nested or hierarchal classification. Basically, it shows the relationship between each data point and every other point in the data set by forming a tree or dendograph (similar to a family tree). The similarity of two data points is inversely related to the distance between them along the branches of the tree.

There are basically three types of hierarchal clustering: single linkage, average linkage, and complete linkage. Among these, single linkage is the most simple. It begins by finding the link between the two closest data points. The succeeding stages consist of finding the shortest remaining link which directly joins together data points not already joined indirectly through a chain of links or branches. In essence, it searches through the distance matrix seeking the shortest distance that will contribute to further linkage or branching of the clusters.

Data may be entered from the keyboard or reside on a data file.

Example: Eleven measurements were made on each of 8 products. Products B and Y were actually different lots of the same product, as were I & J, L & M, and C & E. We desire to know something about the relative similarities and differences among there eight products. The data matrix is as follows:

We may conclude that I & J are the most similar products made, followed by L & M, C & E, then B & Y. The grouping into pairs was as expected. (The smaller the distance with which two products "join up," the greater the similarity.) We may then conclude that among all pairs, pair B & Y and pair I & J are most similar. We then see that the group B, Y, I, J, L, & M are more similar among themselves than among the pair of C & E.
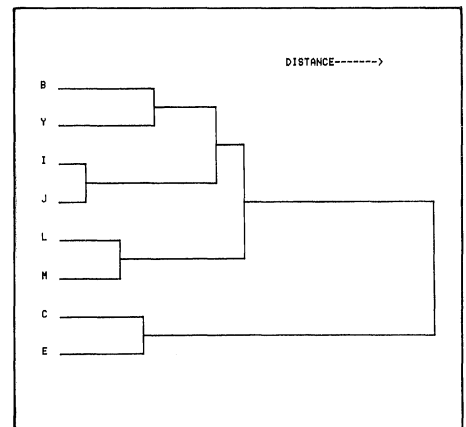
B: 2022170	25607	21199	46.5	4.1	4.0	0.0034	0.0025	460	160	10
Y: 2070000	22300	16356	47.0	4.0	4.0	0.0016	0.0015	520	159	42
C: 2414685	13007	9729	66.6	5.3	4.6	0.0470	0.0320	450	270	182
E: 2261855	16254	11731	66.6	5.3	4.5	0.0600	0.0330	430	270	182
I: 2251560	16037	11774	55.5	5.5	5.4	0.0064	0.0043	310	240	15
J: 2227055	16617	12310	50.4	5.5	5.4	0.0070	0.0045	280	240	7
L: 2442670	19619	14601	54.4	5.1	5.0	0.0079	0.0049	500	250	111
M: 2456880	16530	13195	53.3	5.2	5.1	0.0063	0.0036	470	250	133

Program 18

Title: Nonlinear Mapping
Author: Jerry W. Anderson
        Bartlesville, OK
Memory Requirement: 32K
Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 244

Nonlinear mapping reduces the dimensions of a data set (i.e., N vectors with L dimensions) to one of lower dimensions, usually to two so a cross plot may be used to observe the groups (or clustering) of the data units. The mapping preserves the data structure by obtaining N points in lower-dimensional space so their interpoint distances approximate the corresponding interpoint distances in L space. In other words, we seek a set of N two-dimensional vectors whose distance matrix (N x N) is very near the distance matrix (N x N) of the original data set, i.e., N vectors in L dimensions.

Designate the N vectors in L-space by X_i, i = 1, . . ., N. Designate the N vectors in d-space (where d = 2 usually) by Y_i, i= 1, . . ., N. Let the distance (according to any desired distance metric) between X_i and X_j in L-space be defined by $d_{ij}* = \text{dist } (X_i, X_j)$. Similarly, let the distance between the corresponding vectors Y_i and Y_j in d-space be defined by $d_{ij} = \text{dist } (Y_i, Y_j)$.

We shall choose starting values for the Y values in d-space. Using these starting values, we shall calculate the d-space distance, d_{ij}, from which we define an error E. This error tells us how well the present configuration of N points in d-space fits the N points in L-space:

$$E = \frac{1}{\Sigma\{d_{ij}*\}} \overset{N}{\underset{i<j}{\Sigma}} \frac{\{d_{ij}*-d_{ij}\}^2}{d_{ij}*}$$

The next step in this iterative algorithm is to adjust the Y variables so as to decrease the error E. A steepest descent procedure is used in order to minimize the error.

The data matrix must be stored rowwise on a previously-created file.

Considering the same data used in Program 17, use nonlinear mapping as a data reduction technique to reduce the data set from (8,11) to (8,2) while preserving interpoint distances as much as possible, and then crossplot the two resulting columns. Closeness on the crossplot indicates relative similarity.

Products B & Y are very similar, as are C & E, I & J and L & M. However, the pair B & Y is relatively dissimilar to the pair C & E. The mapping error of .014 is good.
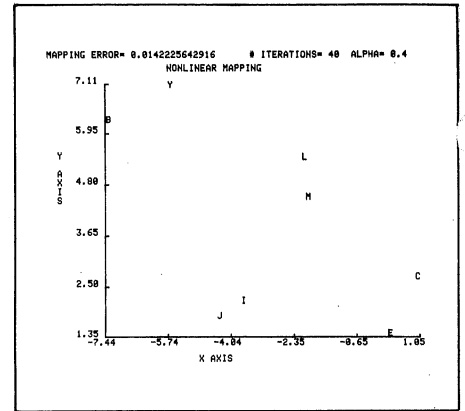
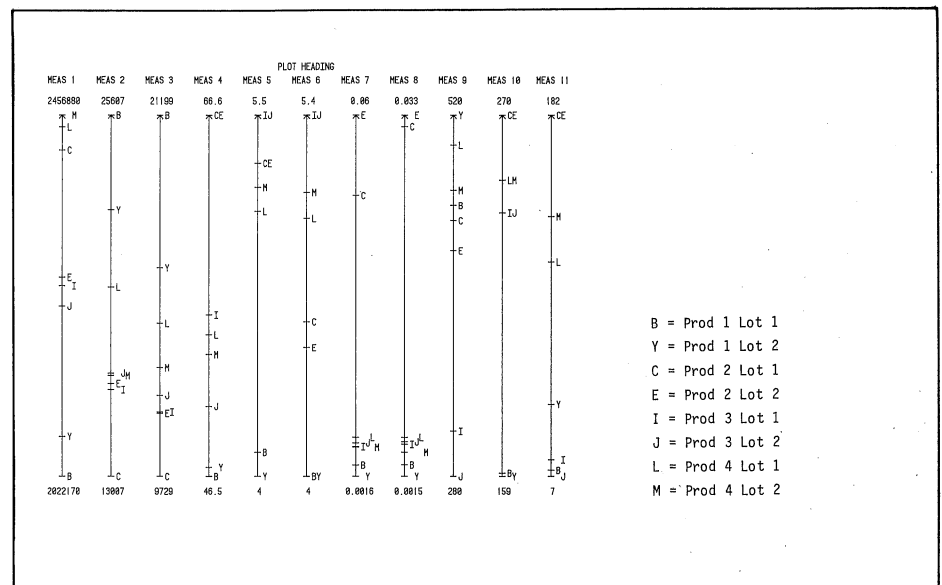
Program 19

Title: Vertical Plot
Author: Jerry W. Anderson
        Bartlesville, OK
Memory Requirement: 8K
Peripherals: Optional-4662 Plotter
Files: 1 ASCII Program
Statements: 137

This program graphically presents on one page a data set consisting of M rows in N dimensions. Each of the M observations is represented by a 1-character code, which is plotted on each of N vertical scales across the page.

A typical use of the program would be in an examination of M products, each with N measurements or properties (missing values are allowed). Relative product differences, with respect to each of the N measurements, are easily observed on the scales. Product and measurement labeling is allowed.

Thus, we have depicted in two dimensions an N dimensional space. We have consolidated many plots on one page.

Program 20

Title: Reference Index
Author: F.G.G. Voigts
     National Physical Research Laboratory
     Pretoria, South Africa
Memory Requirement: 32K
Files: 1 ASCII Program
     Requires dedicated tape
Statements: 385

This program creates, stores on tape, updates and lists an index of literature references. The following data may be stored:

    Author
    Journal
    Volume
    Date
    Page
    Status (or type) Symbol
    Title

The minimum information required is author, year of publication and the status symbol. The rest is optional and the fields could be used for other information.

The program assigns a code to each reference, comprised of the first three letters of the author's surname, the year of publication and another letter to distinguish between otherwise equal combinations. The references are arranged alphabetically in 26 data files according to these codes. New references are sorted and inserted in the appropriate file. Existing references within a file may be deleted or changed.

```
COU78AC  Cougar, J. Daniel &   Datamation         24.9(Sep 78)116-123
         WHAT MOTIVATES DP PROFESSIONALS

HUG80AC  Hug, Richard & Press  Mini-Micro Syste  XIII.5(May 80)128-138
         DESIGNING TRANSPORTABLE SOFTWARE

IVA81A#  Ivancevich, John M '  Ergonomics         24.5(May 81)351-363
         JOB DIFFICULTY DIMENSIONS

KIM80AD  Kimmel E.             Machine Design     52.6(Mar 6 80)58-53
         TEMPERATURE-INDICATING MATERIALS

LEW78AB  Lewontin, Richard C.  Scientific Ameri  239.3(Sep 78)212-230
         ADAPTATION

MIN80AW  Minor, Dennis E.      Technical Writin   18.3(1980)201-212
         COHERENT DEFORMATION IN SCI & TECH WRITING

SON80AC  Sonsco, David C. et   Spectrum           17.5(May 80)34-38
         HOW COMPUTERS TALK TO THE BLIND

SPA79AC  Spaniol, O.           Computer Network   (1979)315-326
         MODELLING OF LOCAL COMPUTER NETWORKS
```

Program 21

Title: 4050/TM5000 GPIB Routines
Authors: Jim Kimball
     Barbara Malin
     Steve Peterson
     Tektronix, Inc.
     Beaverton, OR
Memory Requirement: 8K
Peripherals: TM5000 Mainframe and
           Plug-ins
Files: 48
Statements: 2,498
Requires dedicated tape

These 47 routines, following a directory on file 1, match listings in the GPIB Programming Guide (available separately under part number 070-3985-00). They are provided on this tape to save the user the task of keying in examples from the listings printed in the Guide.

The GPIB Programming Guide aids the users of 4050 Desktop Computers and TEKTRONIX TM5000 series instruments in making the software connection. The Guide introduces the 4050 as a controller for TM5000 instruments, with programming information specific to instrument control. Major topics are:

    GPIB Input/Output
    Interrupt handling
    Interrupt handling statements
    Utility routines
    4052/GPIB send and receive program

TM5000 series instruments specifically covered in this Guide include:

    DC5009 and DC5010 — Programmable Universal Counter/Timers
    DM5010 — Programmable Digital Multimeter
    FG5010 — Programmable Function Generator
    PS5010 — Programmable Power Supply

Corrections to New Packages

As happens to any new product, we are findng some errors crept into some of the documentation or programs for the new 4050 Applications Library packages. If these errors impact the use of a program, they will be published in TEKniques as they are uncovered. If they are "typos" or cosmetic problems, they will be corrected in the next printing of the affected documentation or duplication of the master tape/disk, but won't be reported in this column.

Documentation Correction

Title
TEXT PROCESSING D1
Part Number
062-5970-01
Program Title
$Edit.Dos/$Format.Dos
Page Number
88

Original Copy
Step 3.
  COPY "$FORMAT.DOS",0TO"$#",1
  COPY "$EDIT.DOS",0TO"$# ",1

  COPY "EDIT#",0TO"@#",1
  COPY "@TEXT#",0TO"@# ,1

Change

Insert @ before EDIT on third line so it will read:

COPY "@EDIT#",0TO"@#,1

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

COMMITTED TO EXCELLENCE

Tektronix®