Tekniques

The 4050 Series Applications Library Newsletter | May 1, 1980 | Vol. 4 No. 3

Sgt. Bob Wilson says the Criminal Justice Data System is helping detectives pool their data on crimes, to aid in predicting the likely times and locations of future crimes. (Photo Courtesy of Eugene Register-Guard.)

4050 DRAWS FOR LANE COUNTY LAWMEN

by Tom Sawyer
Director, Criminal Justice Data System
Eugene, OR
with Terry Davis
TEKniques staff

Consider the following, with respect to modern law enforcement: Inflation continually increases the cost of manpower and equipment. Pressure is on legislative bodies everywhere to hold back taxation, bringing a corresponding reduction in services. Yet criminal activity is on the increase, from cities to suburbs.

How, then, are modern law enforcement agencies to operate, and continue to do their job. In Lane County, Oregon, computing power and graphic output were chosen as a means to increase the efficiency of law enforcement throughout the county. And the new tools that they're using include 4052 and 4054 Graphic Computing Systems. Let's take a look at what's being done.

Some History

In Lane County, local governments have historically cooperated for the benefit of county residents. One area of cooperation is the Lane County Regional Information System, formed in the mid-sixties as a cooperative effort of Lane County and the cities of Eugene and Springfield, along with other local government agencies. The

Tekniques

In This Issue

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

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

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

Regional Information System contains several kinds of data systems, and serves as a central coordination and management point.

The Criminal Justice Data System (CJDS) is one of the major projects of the Regional Information System, and is of central importance here. The CJDS was started in 1963 to meet the needs of local police, and evolved by 1976 into an integrated information system. The CJDS was linked with Geographic Data System (discussed later) to provide a major graphic benefit to the law enforcement personnel. This link has allowed the system to analyze and display public safety data by area.

The Criminal Justice Data System is jointly funded and maintained by all of the law enforcement and criminal justice agencies throughout Lane County. All official acts of this law enforcement community are recorded in the data base of the Criminal Justice Data System. And that data base is used daily by officers from the cities of Eugene, Springfield, and Cottage Grove, as well as Lane County Sheriff's deputies.

All of this information is recorded in the CJDS IBM computer, and is available through any of the law enforcement agencies in the system, helping personnel to manage their daily activities quickly and accurately. Their access is through more than 90 terminals that operate 24 hours a day, every day of the year. Most of these terminals provide readout in alphanumeric form, but interest in graphics is growing daily as they use the graphic systems.

In fact, you might view graphics a major component of the Operations/Crime Analysis section of the Eugene Police Department. This section, funded since 1978 by the Law Enforcement Assistance Administration (LEAA), is part of the very successful Integrated Criminal Apprehension Program (ICAP). Through a cooperative effort, the Eugene Police and the CJDS' have used the resources of the criminal justice data base and the geographic data system, to supply an enhanced graphic information system to local law enforcement agencies. But before we look at the crime analysis graphics tool, let's take a look at the geographic data system that provides the base for ICAP's mapping output.

The Geographic Data System

The Geographic Data System is jointly funded and developed by Eugene, Springfield, Lane County, and the Lane Council of Governments. The system provides a standardized collection of computerized locational information, with data identified and interrelated by (x,y) coordinates derived from the Oregon State Plane Coordinate System. The system also includes a set of techniques and procedures that let the user manipulate and display the geographic information, creating specialized maps.

The key to the Geographic Data System is the system address file, which provides a complete standardized set of site addresses in Lane County. For each of those addresses in the County, there is an address record that contains the state plane coordinates for that address. So, any data that contains addresses can be entered into the system and analyzed geographically.

The basis for graphics in the system is the parcel file. The parcel file contains the digitized parameters of all land ownership parcels (homes, businesses) and land use parcels (parks, public lands), along with streets, rivers, and other geographic features, to build a complete map base for the system. This file has two main uses in the system: It serves as the source for other special purpose files, and it is used for drawing maps on a plotter or on a graphics terminal. In addition, the parcel file supplies land use information that allows users to separate, for example, the addresses of service stations from those of grocery stores. This also turns out to be of benefit to the law enforcement officers that use the CJDS facilities.

Mapping Crime

Look at the possibilities that the address file holds for law enforcement use: crime incident addresses, for instance, can be matched to the data in the address files, attaching geographic coordinates. Crime incidents can thus be analyzed by location, and plotted or displayed on the Geographic Data System base map. Compare this with the pin map of "crime in the city," so prevalent in detective dramas. Except this map resides in the computer, to be called upon as needed. The map can be "blown up," to zero in on a particular area. And each "pin" represents a lot more information than a map pin ever could, and can be sorted and displayed on the basis of any one piece of that information, or a combination of those pieces. (See Fig. 1.)

Now, using the combined power of the Criminal Justice Data and the Geographic Data Systems, officers can produce maps and reports showing where different types of

crimes occur. And Ken Hanfland, a crime analyst with the Eugene Police Department's operations/crime analysis section, uses the interactive graphics capabilities of the system to select an area and display criminal activity within that area. Or to look for a particular type of criminal activity, and produce a map showing where those activities have occurred, then zero in on those areas. Figures 2 and 3 are examples of this technique.

Officers also select from the information in the files to produce specialized maps. For example, by searching for type of crime and time of day, an officer can quickly produce a map that shows the location of nighttime residential burglaries during the last month. In addition to these mapping activities, area law enforcement agencies are experimenting with the Rand Corporation's Patrol Car Allocation Model (PCAM) and a locally-developed simulation model. A combination of these tools will be used with the graphics system to present agency personnel with usable tactical data and strategic decision-making alternatives.

Some Benefits

The mapping activities just described provide a very real benefit in the form of improved tactical deployment. Commanders can call up a map, or a series of maps, to isolate those areas that show increased criminal activity, or otherwise show a need for increased police patrol service. Analysis may reveal specific types of suppressible crime, as well as pinpointing areas where patrol crime can be seen as well as areas pinpointed where patrol saturation methods would likely be effective. And later, they can use the same analysis methods to monitor and evaluate their effectiveness in addressing area crime problems.

Graphics and the Geographic Data Base add a new dimension to Investigative Support as well. For instance, area detectives and investigators can become quickly familiar with the nature of the area surrounding a crime. They can look into the data base for specific types of information, and come up with all reported activity in the area, in map form as well as tabular form. And investigators can also confirm the presence of known offenders who reside or were contacted in the proximity of a crime, to aid in identifying suspects.

Crime analysts like Ken Hanfland find that their productivity is also on the rise. Since the crime analysts have had the power of computer graphics at their disposal, the clearance rate (cases solved) for robbery,

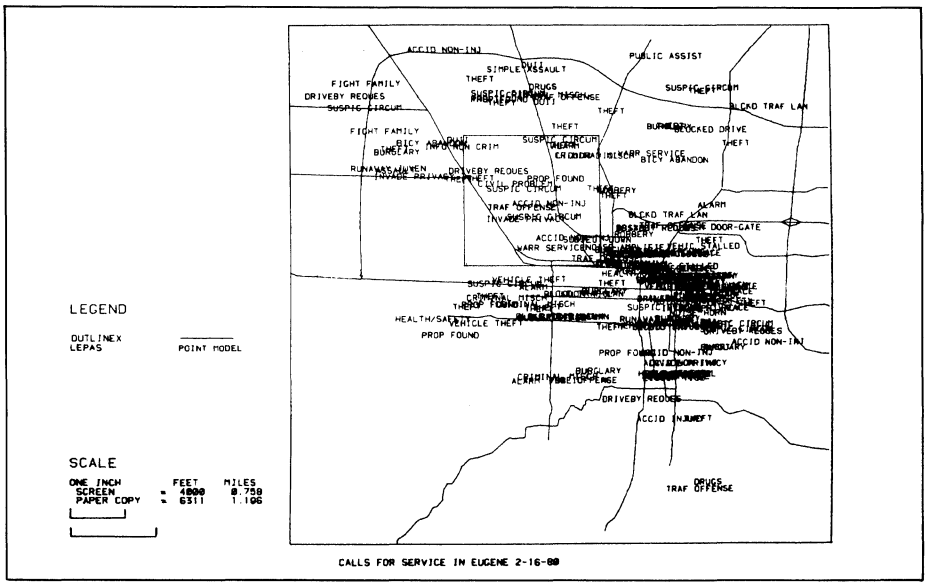


Fig. 1a. Eugene city map displayed with calls for service.

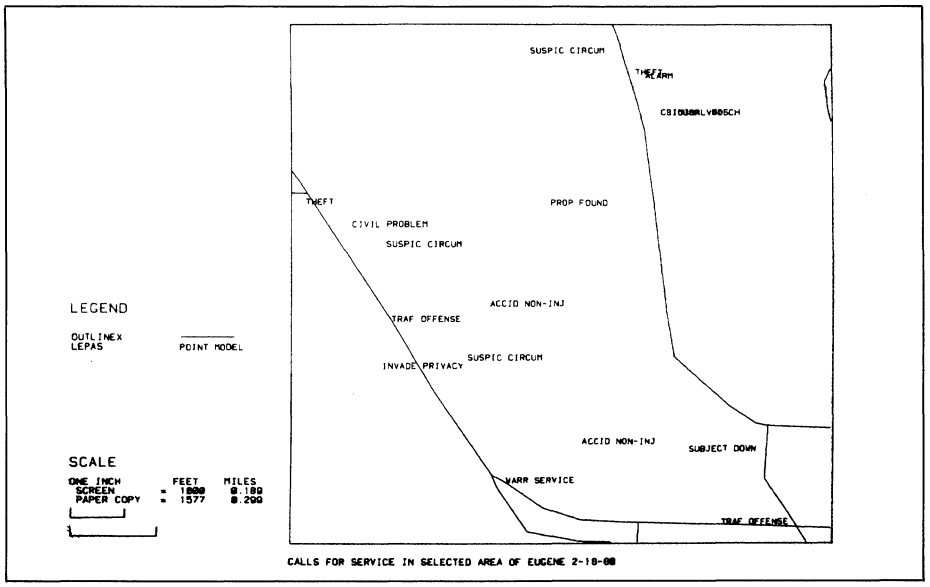


Fig. 1b. Zooming in to look at calls for service in one city area.

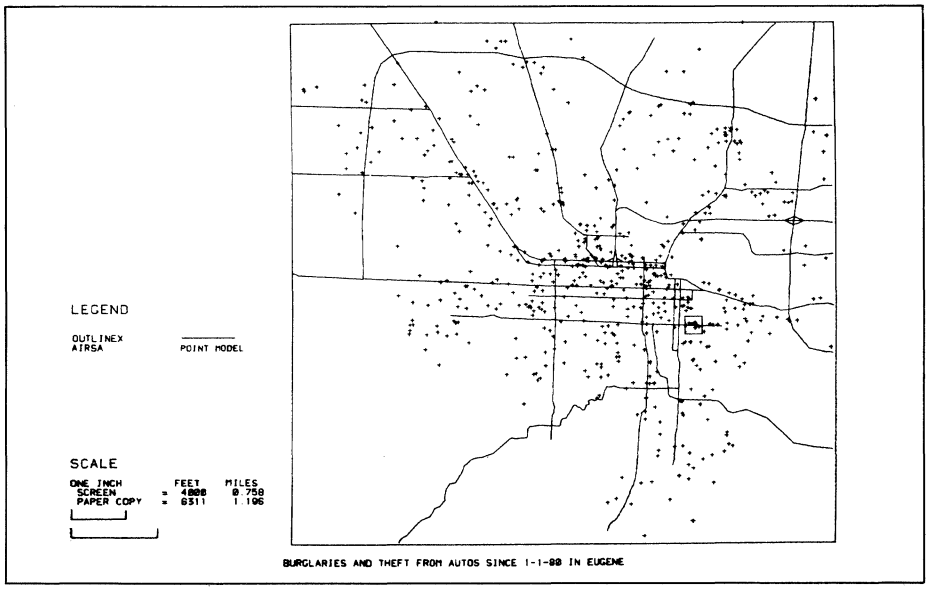


Fig. 2. A map showing one crime type over a period of time.

burglary and rape in area police agencies has increased 18 to 30 percent. It's this kind of productive activity that CJDS looks for when they talk about increased efficiency through the use of the computing system; increased clearance rates are one of the law enforcement community's equivalents to added sales or increased profits.

It's the crime analysts who use the system to identify and relate those elements valuable for solving a particular criminal problem; their analyses are distributed in meaningful form to the appropriate commanders. "The neat thing for us," Sgt. Bob Wilson reports, "is that we can tap however many files we need, and analyze and break the information down more and more until we get what we're looking for." It's that ability to interact with the system and go through several iterations quickly that has made the present system so useful. Previous systems sometimes took days for one graphic interation; several iterations took several times as long.

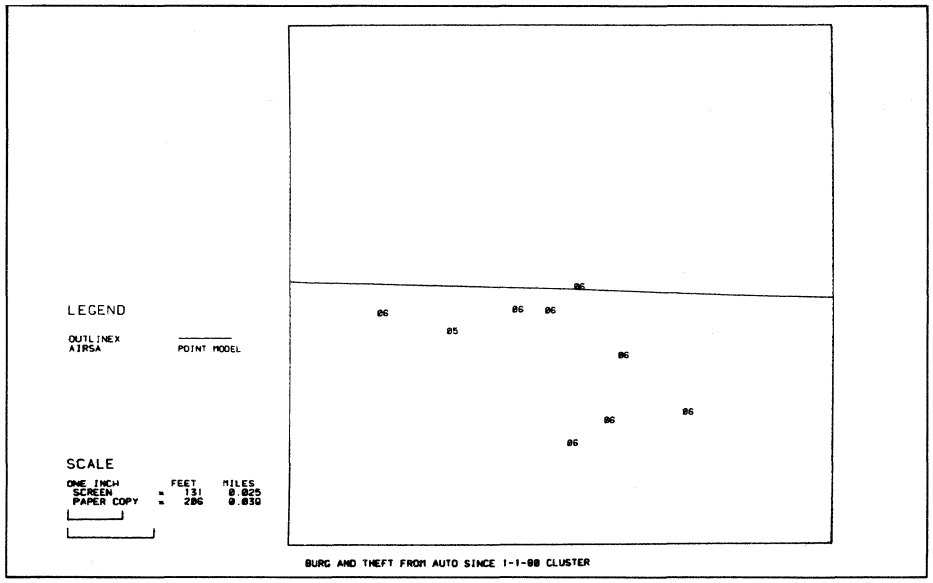


Fig. 3. Zooming in on one cluster of criminal activity. The numbers are codes for the type of call: burglary or theft.

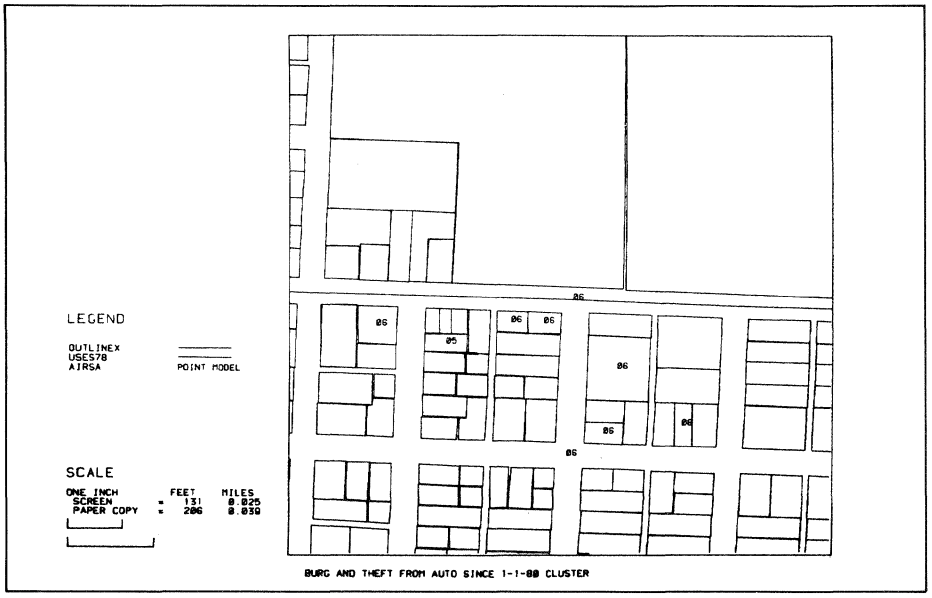


Fig. 4. The same map shown in Figure 3, with land parcels and streets shown also.

More Uses, More Benefits

The graphic computing power of the system makes resource allocation a tangible, dynamic function. And it's a function that provides real benefits in the form of additional increases in efficiency. A series of graphic "snapshots" of the CJDS maps shows the locations and types of police calls for service over some set time periods; commanders use these maps to assist them in allocating resources and providing "directed patrols." This may well save a significant portion of a recent budget cutback, in the form of gasoline saved through more efficient, directed use of resources.

The ability to interact easily and quickly with the 4052 and 4054 Graphic Computing Systems lets the commanders look at the locations of calls for service based on any number of parameters. Among these parameters are time of day, day of week, type of call, how swiftly a response was made, which calls required additional backup, and which units responded in areas outside their district. In addition, they like to use the Data Graphing and Presentation Aids software to prepare bar charts, pie charts, and graphs. Ken Hanfland notes that Patrol Commanders now request specific graphs and management information, to be included as a part of their regular reports.

The output of the Graphic Computing System proved a benefit in the citizen awareness and involvement program instituted not long ago. Maps and charts showing specific crime problems in neighborhoods helped increase citizen understanding of community problems, and helped them help prevent crime through involvement actions such as neighborhood watch, or improving their home security devices.

Output from the same 4050 Series Graphic Computing Systems helps law enforcement officials communicate with elected officials, executive officers, and the media as well. A recent study of jail "crowding" helped to quckly resolve a dispute, through the clear communication with graphics, as "it's hard to argue with graphics." Line graphs showing arrest rate trends, and bar charts comparing crime types with clearances, have become a regular part of management reporting in Lane County law enforcement agencies. All are prepared on the 4050 Series Graphic Computing Systems in the CJDS offices.

Projected Uses

Fueled by the success of graphic applications thus far, the CJDS personnel are looking at more ways to put graphic computing power to work for them. Byron Vanderpool, a programmer with the department, has a potential use that he hopes may soon be refined into regular use. Figuring that "the more you understand what you have, the more you want to do with it," he began looking at a way to combine the analysis power of the Graphic Computing System with another police graphic tool, the composite drawing.

The composite drawing is prepared by a police specialist from "standard" facial features, with the aid of a victim or an eyewitness. Although there are always

variations in such drawings, it was determined that witnesses are usually consistent about some details, such as eye position, the shape and position of the mouth, and the spatial relationship between them.

So a program is under development to use those consistencies to the advantage of the Lane County law enforcement officers. Four points on composite drawings, and photographs of known offenders, are digitized and entered into the data base: the pupils of both eyes, and both corners of the mouth, as shown in Figure 5. The system then performs calculations of several angles between these points and stores the information in the system data base. When a crime occurs, and a new composite sketch is made, the sketch can be digitized and compared against those in the data base. The file numbers of those that match, within a specified margin for error (3-5% typically seems to work well) are listed on the display, so that their "mug shots" and files can be pulled.

And the Future

Real improvements are occuring in the activities of Lane County Law Enforcement personnel, as a result of the interactive graphics of the ICAP system. These are improvements that make the involved organizations operate in a more efficient, streamlined manner, and that can be measured in units from clearance rates (productivity) to fuel savings (efficiency). The system's usefulness as a tactical and strategic tool, as well as an informational tool, is undisputed. The effectiveness of programs can be evaluated with the same tools that implement them; those that work well can be expanded, and tools that are not productive can be abondoned. This system, using both the 4052 and 4054 Graphic Computing Systems, brings the future into the present for Lane County law enforcement personnel.

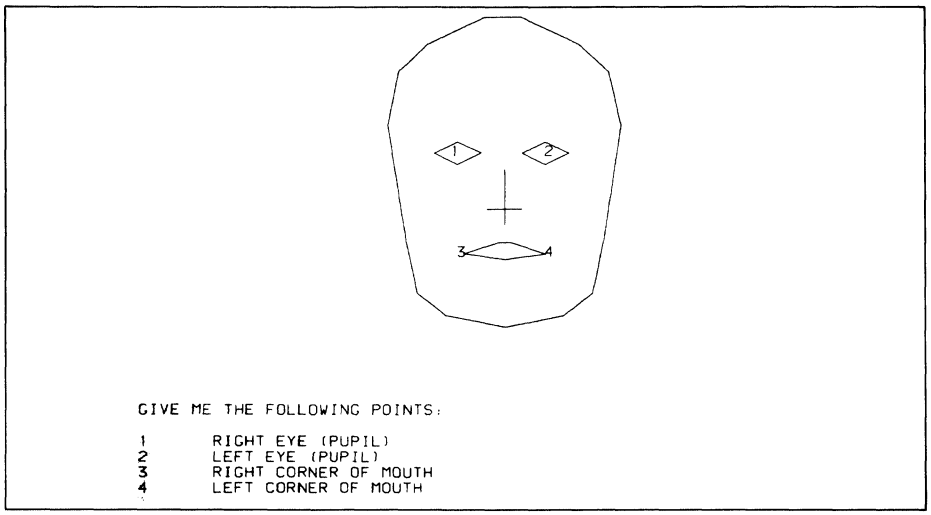


Fig. 5. Digitized eye and mouth positions from composite drawings can be compared against a data bank of the same digitized points from photos of known offenders.

PLOT 50 Easy Graphing: You Describe the Task; It Handles The Details

by Patricia Kelley
TEKniques Staff

with Howard Mozeico
Tektronix, Inc.
Wilsonville, OR

The PLOT 50 Easy Graphing Vol. 1 software package was introduced in the March 15, 1980, issue of TEKniques. That article took an in-depth look at its five unique resource modules: EASY GRAPH, TUTORIAL, HELP, UDK Syntax, LIST.

We talked briefly about Easy Graphing's command processor and touched on the Easy Graphing command language. Now let's take a closer look at the command language and see why it's so versatile.

When you load Easy Graphing into your 4050 System, you have 41 commands with which to prepare and display graphs. These commands allow you to enter **your** graph parameters for almost unlimited graph design. And the commands and parameters are entered directly from the keyboard—no waiting for questions, menus, or other prompts.

First, let's review a list of the 41 commands. Notice, these are special purpose Easy Graphing commands, not BASIC language commands. (BASIC language commands are disabled while Easy Graphing is in your system.)

Easy Graphing Commands

Data entry and manipulation
- ATTACH
- CHANGE
- DEFINE
- DELETE
- ENTER

Output device specification
- PERIPHERAL
- PLOTTER
- HCOPY (hardcopy)

Graphing
- ADD
- GRAPH
- LIST (summarizes current graph)
- PIE

Graph formatting
- BAR
- EXPLODE
- LINE
- PVALUE
- SYMBOL
- XGRID
- XLABEL
- XLOG
- XRANGE
- XTICS
- YGRID
- YLABEL
- YLOG
- YRANGE
- YTICS

Titling
- DATE
- LEGEND
- TITLE
- XTITLE
- YTITLE

Command files
- BELL
- CONTINUE
- KEYBOARD
- PAUSE
- PRINT
- RUN
- SAVE

Exit
- BYE
- RESTART

Choosing Printer, Plotter or Screen

Easy Graphing directs all output to the 4050 Screen unless told otherwise. But it can direct certain output to a plotter or printer.

When the command PLOTTER ON is given, all **graphic** output will be sent to the plotter until the command PLOTTER OFF is given. This command may be extended to prompt for pen changes between curve plots; or, if your plotter has multiple pen holders, you can specify a curve's color by pen number.

```
PLOTTER ON PEN 1 2
```

Here output is directed to the plotter; pen holder number 1 will draw curve 1 and pen number 2 will draw curve 2. (Pen 1 will always do the labeling, axes, etc.)

By preceding certain commands with the hard copy device command, **alphanumeric** output can be directed to a printer or plotter. For example,

```
HCO PRINTER;LIST
```

directs the output generated by the LIST command to the printer, rather than the screen. (The Easy Graphing LIST command was covered in TEKniques Vol. 4 No. 2.)

Designing a Graph

By using Easy Graphing's pre-set (default) graph parameters, you need only know two commands to produce a graph: ENTER and GRAPH. Each ENTER command requires a variable name followed by the numeric data.

```
ENTER MAIN 2419,4245,6030,7183
ENTER NEBR 2009,3657,5882,6720
ENTER IDHO 1812,3243,5179,5980
ENTER YEAR 1960,1970,1975,1977
```

The first parameter of the GRAPH command must be the X-axis variable name, followed by the names of the variables you want to graph.

```
GRAPH YEAR MAIN NEBR IDHO
```

The graph will be sparse but informative; a quick way to preview your data.

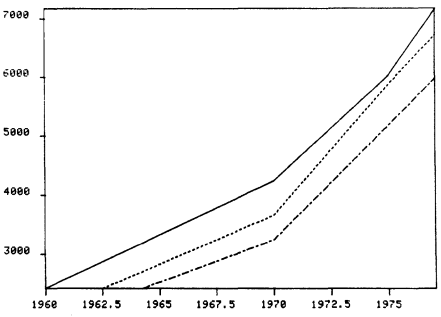


Fig. 1. A graph created using only two Easy Graphing commands: ENTER and GRAPH. Default parameters set the X and Y ranges, tic intervals and labeling, and line types.

Tailoring a Graph

After you preview your data you can tailor your graph by overriding the default

parameters with **your** parameters. For example, you can change both axes' ranges and the X-axis tic spacing on the above graph by entering:

```
YRANGE 1500,7500
XRANGE 1955 1980;XTIC 1955 5
GRAPH
```

The RANGE command changes lower and upper values for the respective axis. TIC commands re-position the tics; the parameters specify the location of the first tic and the increment for succeeding tics, both in axis units. (Note, two commands may be on the same line by using a semicolon as a delimiter.)

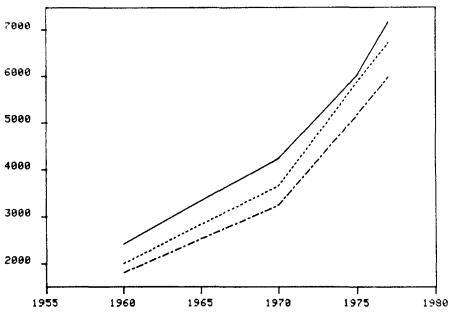


Fig. 2. Notice the entire Y-axis data are now displayed because of the YRANGE change. The default RANGEs (Fig. 1) are the ranges of the first Y-axis curve named in the GRAPH command (MAIN in the first graph).

Axis units are the default labels, but alternate X and Y tic labels can be set through the XLABEL and YLABEL commands. You may assign the number of tic marks and specify a label for each, convert integer-labeled tics to their equivalent three-lettered month abbreviations, or impose your own periodic labeling.

Six line types, bar shadings and symbols provide a wide choice for customizing your graph:

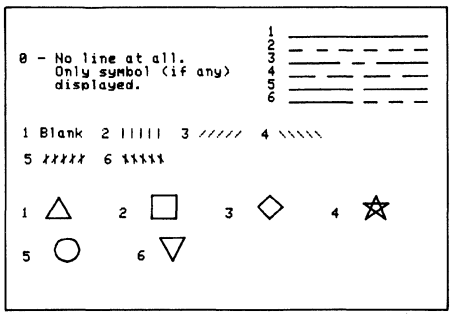


Fig. 3. You have a choice of line types, bar shadings and symbols in Easy Graphing.

Choosing from the styles in Figure 3, a few commands reformat the curves in the previous graph. By default the first curve is graphed as a solid line, so we'll just add a symbol to it. The line for the second variable will be changed from dashed to solid, and symbol type 5 added. The third curve will be graphed in bar form with shading pattern 3. Notice the parameters for LINE, SYMBOL and BAR commands are: the number of the curve you wish changed, followed by your choice of pattern. The default bar width can also be changed. BAR WIDTH is specified in **X-axis units**.

```
SYMBOL 1 4;LINE 2 1;SYMBOL 2 5
BAR 3 3;BAR WIDTH 1
```

Several commands can be used to annotate the graph. LEGEND parameters are the curve number followed by the legend title (in quotes). And, by specifying screen coordinates, you can relocate the legends.

```
LEGEND 1 "Maine";LEGEND 2 "Nebraska"
LEGEND 3 "Idaho"
LEGEND 25,90
```

A main title (heading), subtitle, X-axis title and Y-axis title may also be specified.

```
TITLE "PERSONAL INCOME PER CAPITA"
DATE "Data for Years 1960, 1970, 1975, 1977"
XTITLE "From STATISTICAL ABSTRACT OF THE UNITED STATES"
YTITLE "Dollars"
```

Thus, a few simple commands incorporate your graph parameters to produce a custom-designed graph.

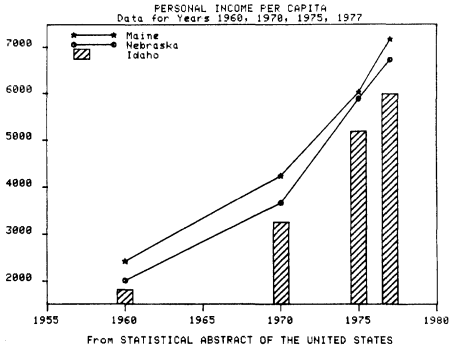


Fig. 4. Compare this graph to the one in Figure 1. They both contain the same data.

You'll note other commands under **Graph formatting** in the list of Easy Graphic commands. EXPLODE and PVALUE were illustrated in the last article in TEKniques; both format the pie chart style graph. YLOG or XLOG assigns a logarithm scale along the respective axis; XGRID and YGRID create grids extending from the respective axis.

In this article you have become acquainted with one of the **Graphing** function commands: GRAPH. Two others, LIST and PIE, were covered in the last issue of TEKniques. Another command in this category is ADD. ADD allows you to graph additional variables without re-defining your initial graph.

Data for the curve(s) to be ADDed can be ENTERed any time. For example, they could have been entered when data for the variables MAIN NEBR IDHO and YEAR were entered above. Remember that the first time you issue a GRAPH command, you designate which variables you want to graph; therefore, all variables residing in memory need not be displayed. However, you can also ENTER additional curve data after defining your initial graph. That's what we'll do here.

```
ENTER YR2 1960 1970 1975 1976 1977
ENTER WASH 2608 4317 6518 6772 7820
```

An additional data point on the X-axis was desired for this curve so a second X-axis was entered (YR2), along with the variable and its data.

The ADDed curve is immediately drawn, so the ADD command must always be preceded by a GRAPH command. (The semicolon allows another command(s) to be issued before execution; the ampersand allows the command(s) to be entered on the next line.)

```
GRAPH;&
ADD YR2 WASH;LINE 4 1;SYMBOL 4 3;LEGEND
```

Compare the resulting graph with that in Figure 4. Different data is compared within the same graph parameters.

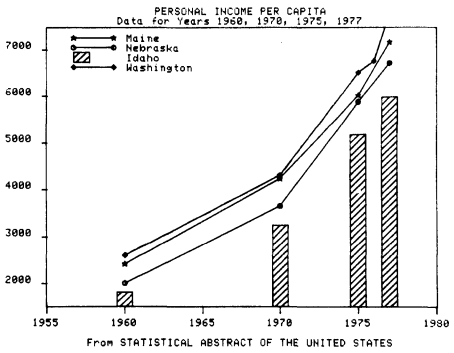


Fig. 5. The ADDed curve is graphed against a different X-axis: notice the extra data point for 1976.

Once you are satisfied with your graph, give it a filename (up to 10 characters) and store it. (If another file exists by the same name, it will be overwritten.)

```
SAVE PERSINCOME
```

To retrieve a graph just key in the command RUN followed by the filename. Easy Graphing will bring all your data and parameters for that graph back into 4050 memory and display the graph.

Editing a Graph

To change, add or delete **data**, first use the LIST resource to examine your variables and their elements.

```
NAME  TYPE          MIN     MAX          COUNT

MAIN  Long          2419    7183           4
NEBR  Long          2009    6720           4
IDHO  Long          1812    5980           4
YEAR  Long          1960    1977           4

LEGEND  25,90
XRANGE  1955,1980
YRANGE  1500,7500
BAR WIDTH 1

CURVE  XVAR  YVAR  SYMBOL  BAR  LINE

  1    YEAR  MAIN     4          1
  2    YEAR  NEBR     5          1
  3    YEAR  IDHO           3    3
```

Fig. 6. Typing LIST produces a listing of variables, along with a summary of data and graph parameters residing in Easy Graphing's memory.

```
          MAIN      NEBR      IDHO

  1       2419      2009      1812
  2       4245      3657      3243
  3       6030      5992      5179
  4       7183      6720      5980

TOTAL     19877     18268     16214
```

Fig. 7. Typing LIST followed by the name of the variable(s) produces a listing of all elements in that variable. The command which produced the above display was: LIST MAIN, NEBR, IDHO.

The CHANGE command changes, inserts, or deletes an **element** within the named variable. Below, the first command would change element 3 of the variable named NEBR to 5884. The second command would delete the same element entirely. By specifying a value between the third and fourth element in the variable IDHO, the third command would insert the value 5800 after the 3rd element and move current element 4 to position 5.

```
CHANGE NEBR 3 5884
CHANGE NEBR 3
CHANGE IDHO 3.4 5800
```

DELETE deletes a specified **variable** from memory. All data and graph parameters pertaining to that variable are removed. The following command would delete the variable NEBR and all data and graph formatting associated with it from future graphs.

```
DELETE NEBR
```

With these commands at your disposal, it's easy to design and SAVE your graph, retrieve it, update it, and SAVE it again. But other commands increase versatility and flexibility.

Data Manipulation

Entering data from the keyboard may not be enough to suit all your needs. Therefore, Easy Graphing allows you to manipulate your data, and to bring in data from PLOT 50 Standard Files.

DEFINE

As an alternative to ENTERing data from the keyboard, you may use the DEFINE command. DEFINE lets you create data points from other variables or functions.

For example, you have two variables—TER1 and TER2—consisting of 12 months of sales data for two territories. You may DEFINE another variable depicting average sales. Easy Graphing automatically performs the calculation for each data point and graphs the third curve.

```
DEFINE AVRG = (TER1 + TER2)/2
```

Or, you could use DEFINE to compare experimental results with some theoretical values. For example, say you ENTERed data that had been observed experimentally. Theoretical values may be known only by some equation, e.g., hyperbolic cosine, or the normal probability function. You can use DEFINE to create the data values for this theoretical data:

```
DEFINE COSH = (EXP(X) + EXP(-X))/2

DEFINE NORM = EXP(-(X↑2)/2)/6.28
```

In each case you would have to ENTER data for the independent variable X. This could simply be done by:

```
ENTER X SHORT 26 -3 .25
```

where 26 values are to be created for X, starting at –3, with an increment of .25 between successive elements.

DEFINE will accept arithmetic or algebraic operations. Eight functions may also be used:

ABS	Absolute Value
EXP	Raises e to a power
MOD	Modulo arithmetic
SUM	Sums specified variable
COS	Cosine
SIN	Sine
LOGN	Natural Logarithm
LOG10	Base 10 Logarithm

ATTACH

PLOT 50 Standard Files* communicate data quickly and easily between applications; the Easy Graphing ATTACH command implements this philosophy. ATTACH allows Easy Graphing to read and graph data from a variable within a PLOT 50 Standard File. Since the number of observations (data points) per variable in a PLOT 50 Standard file are limited only by the amount of disc space available, the ATTACH command allows you to graph large amounts of data that would otherwise surpass Easy Graphing's memory.

To use ATTACH, assign a variable name (just as you do in the ENTER command), specify the name of the PLOT 50 Standard File to be accessed (i.e., the username associated with your PLOT 50 HEADER/DATA file), and the number of the variable within the Standard File. The following example would ATTACH the third variable in the PLOT 50 Standard File named BPA2 to the Easy Graphing variable called CRV1.

```
ATTACH CRV1 BPA2 3
```

When Easy Graphing ATTACHes data from a PLOT 50 Standard File, it doesn't retain the data points in its memory but brings them in at graph time. Therefore, any PLOT 50 Standard File in which a variable is being ATTACHed must reside on the Easy Graphing system disc when you run the graph.

This has been an overview of commands which create and change graph characteristics. As you use the package, you'll begin to realize how you can manipulate commands and parameters to produce an infinite variety of graphs tailored to your specific needs. In the next issue of TEKniques we'll look at a powerful capability of Easy Graphing, the ability to create **Command Files.** We'll also take a look at the UTILITIES routines which create and retrieve PLOT 50 Standard Files and manipulate Easy Graphing files.

***PLOT 50 Standard Files were discussed in TEKniques, Vol. 4, No. 1.**

Educators Update CAD Knowledge

by Tom Sutherlin
Cameron University
Lawton, OK

During their recent break between semesters, some Oklahoma educators got together to learn more about Computer Aided Drafting (CAD). Digitizing techniques taught at a two-day seminar were derived from programs developed and taught at Cameron University at Lawton, Okla.

Cameron CAD History

Cameron University has been producing graduates with expertise in Computer Aided Drafting since 1976. The first digitizing system was developed and taught using a 4662 Interactive Digital Plotter and a 4051 Desktop Computer. Cameron's system now includes a 4956 Graphic Tablet also.

Many graduates of Cameron's Design Drafting Technology program have gone to work programming and operating CAD systems. These technicians have been able to make a fast transition to other systems because of their training in the digitizing system. One graduate, Andy Van Hoy, is developing specialized digitizing programs for small companies. A number of the students have taken special project courses at Cameron after their training in computer graphics, and have aided in the growth of the Cameron system.

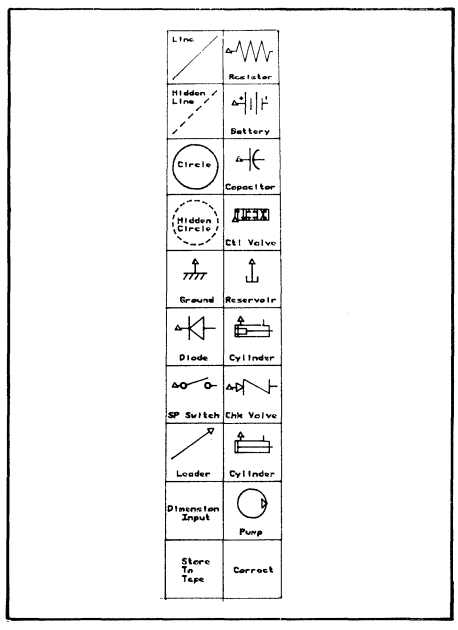


Fig. 1. Andy Van Hoy's composite menu for the 4662 or 4956. (Courtesy of Cameron University, Lawton, OK)

The Seminar Instruction

The seminar, hosted by Mike Wilkins, Tektronix Sales Engineer, provided 4051 systems, 4662 Plotters and 4956 Tablets for the participants. Programming techniques for digitizing a menu to achieve a finished drawing were extensively explored. These techniques were derived from the Cameron System which allows the user to select graphic symbols from a menu (see the Composite Menu in Figure 1) and complete a **finished** drawing on the 4051 screen or on the 4662 Plotter. The method greatly reduces drafting time by using a rough sketch on grid paper, with a menu, for the actual input document. The educators also learned programming techniques to accommodate either the 4662 or the 4956 as input mediums, and to store data to tape.

The Future

Seminar participants expressed the hope of forming an industry/education standards group, which could take on national scope. This would allow educators to teach accepted industry standards, with great potential benefits for both areas. Meanwhile, seminars like these allow a common meeting ground for users interested in computer aided design, where information can be shared so that all can get the most out of their system.

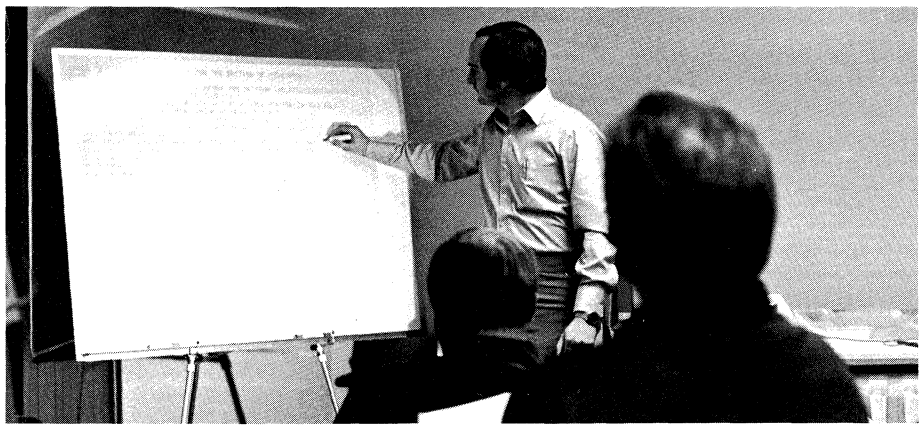
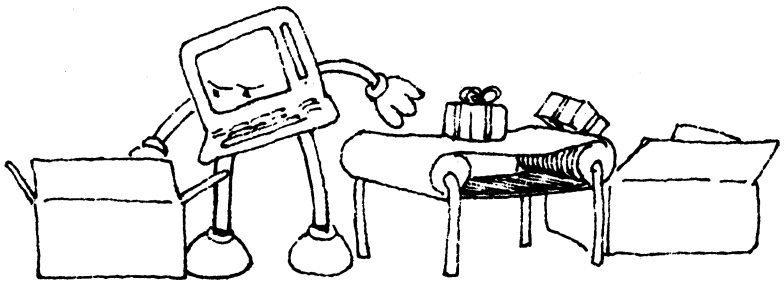


Photo 1: Tom Sutherlin, Cameron University, explains the digitizing program. (Courtesy of Cameron University, Lawton, OK)

Photo 2: CAD seminar participants, left to right: David Kersey, Northern Oklahoma State College, Tonkawa, OK; Gerald Anderson, Connors State College, Werner, OK; Leonard Bachman, Northeastern Oklahoma A&M College, Miami, OK; Gerald McClain, Oklahoma State University, Stillwater, OK; Jim Hysaw, Seminole Junior College, Seminole, OK; Mike Wilkins, Tektronix, Inc., Oklahoma City. (Courtesy of Cameron University, Lawton, OK)

INPUT / OUTPUT



Ed Mallett, AD/ACC, Eglin Air Force Base, Fla, sent in the following questions. **Howard Sanders, Technical Support Specialist at Tektronix, Wilsonville,** responds.

Here is my first problem: I have a BASIC program in my 4050 Series System. I am in **Terminal Mode** using the Data Communications Interface. What commands may the host computer send so the 4050 System will go back to BASIC and continue the program?

ESC ESC will return the 4050 System to the mode it was in before CALL "TERMIN" was executed. If it was under program control, i.e., running a program, and one of the statements was call "TERMIN", when this is executed it enters terminal mode. When it receives the ESC ESC from the host, it will return to program control and continue executing the program at the statement following the CALL "TERMIN" statement.

If it was in idle mode and CALL "TERMIN" entered from the keyboard, when it receives ESC ESC, it will return to idle mode, even though a program may be resident in memory.

Here is the second: In the following code, where is the command RUN stored during the OLD operation? If it isn't stored, why does the program then RUN?

```
100 INIT
110 FIND 1
120 OLD
130 RUN
```

The RUN statement in the above code is unnecessary. In fact the 4050 System automatically does a DELETE ALL before it OLDS in a program. However, there are two ways to OLD in a program: under program control and from the keyboard. When the 4050 System is directed from the keyboard to OLD in a program it is in idle mode. It therefore executes the keyboard command (OLD in this case) and returns to idle mode.

But, when it OLDS in a program in program control mode, it automatically begins executing that program at the lowest statement number in the new program.

Editor's Note

Contest Winners to be Announced

As this issue of TEKniques went to press, our judges were reviewing the entries in the Interfacing Contest. We received several entries in each category, and want to thank all who participated in the contest for submitting their fine applications. The winners of the contest will be announced in the next issue of TEKniques (Vol. 4 No. 4).

A New Applications Library Catalog Coming

Along with the next issue of TEKniques, June will also bring the next Applications Library Catalog. This catalog will contain lots of new programs: the abstracts from the last year's TEKniques, as well as the programs entered in the recent contest. Watch for it!

(Remember, programs added to the library since the 1979 catalog have been documented in your most recent issues of TEKniques.)

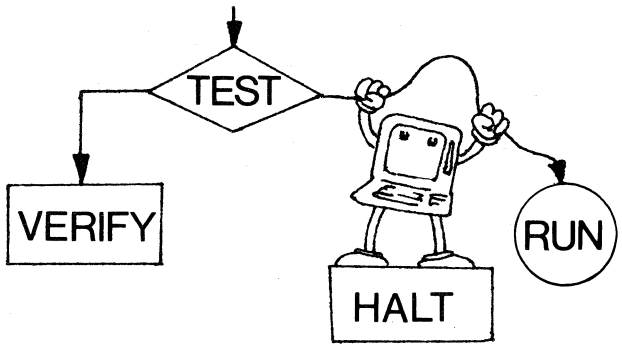
Wanted: Your Programs

We're always looking for more programs to add to the 4050 Series Applications Library. This helps us keep the library growing to better serve more of you, and provides a medium for you to share your programs with others who may be able to use them too. And you get three free programs for each program you send in to the library, so everybody wins.

Programming Tips Too

The applications library staff would also like to remind you that we're looking for those valuable Programming Tips and BASIC Bits. You'll receive any one of 12 programs from the library in exchange for your tip. Details are listed at the end of the BASIC Bits column in each issue.

PROGRAMMING TIPS



Computed GOSUB and GO TO Statements

by Dan Taylor
Tektronix, Inc.
Wilsonville, OR

The computed GOTO and GOSUB statements are used when program control must be passed to different routines, depending on the circumstances.

```
1000 GO TO J OF 2000,3000,4000
1000 GOSUB J OF 2000,3000,4000
```

(J contains the numeric value reflecting the circumstance.)

J is rounded to an integer. If that integer is less than one or greater than the number of statements listed following OF (i.e., if J were 4 or greater in the above case), the branch wouldn't occur and execution would continue with the next statement. This next statement would, therefore, contain some default action (or error message). This is fine for the GOTO...OF... command. However, when the GOSUB...OF... command executes (i.e., J is within the proper range), command is transferred to the routine; and at the end of the routine a RETURN statement would transfer control back to the statement following the computed GOSUB statement. In this case the default or error action would not be appropriate.

```
1000 GO TO J OF 2000,3000,4000
1010 REM Default action or error if J is not in
1020 REM proper range



1000 GOSUB J OF 2000,3000,4000
1010 REM Can't do default here because
1020 REM each GOSUB RETURNs to here, too.
```

Therefore, when working with computed GOSUB statements, use both a GOSUB and a GO TO to accomplish your branching task.

```
1000 GOSUB 1020
1010 GO TO 1100
1020 GO TO J OF 2000,3000,4000
1030 REM Default action
|
|
1100 REM Continue
|
|
```

In statement 1000 an unconditional branch is made to line 1020. There the computed branch (if made) is done through a GO TO command. If it branches, the RETURN statement sends control back to the statement following the GOSUB, statement 1010 above. If J is out of range, the program continues at statement 1030. The conflict is resolved.

Overcoming Mag Tape Read Errors

by N.J.J. Ogbourne
Comalco Aluminum (Bell Bay) Limited
George Town, Tasmania

The following notes describe an alternative to that suggested by Grace and Passeri in the September 17th issue of TEKniques.

We use a Fluke 2240B logger with a 4923 Option 1 tape drive. We then use the 4051 to transfer data from the 4923 tape, through the internal tape drive (using the 128 byte no header format) to a disc file.

The alternative code recognizes that the 4051 places the tape head immediately following the bad read, ready to read the next item. Thus, the array F is initialized to some value, e.g., 9999.9999 (line 250) prior to reading each scan. The scan is read one item at a time (lines 251 to 253) and when a mag tape read error is encountered, UDK 2 is pressed, which will read the next item. Items 'missed' by the bad read then retain the value 9999.9999 and can thus be identified.

A similar procedure is used to recover from a header bad read, using UDK 1. The same approach can be used reading the data from the 4923 to a 4051 internal tape file if you don't have a disc unit.

```
1 INIT
2 SET KEY
3 GO TO 100
4 REM MAG TAPE READ ERROR IN HEADER
5 GOSUB 1000
6 GO TO 210
8 REM MAG TAPE READ ERROR IN DATA
9 REM SCAN NUMBER IS I-1
10 PRINT "BAD READ ITEM F(";J;")", SCAN "I-1
11 GO TO 253
98 REM                FILE "RECOVER" (2)        28/12/79
99 REM ************ READ IN INITIAL DATA ******
100 PRINT "IF MAG TAPE ERROR OCCURS IN LINE 200, PRESS UDK #1"
110 PRINT "IF MAG TAPE ERROR OCCURS IN LINE 252, PRESS UDK #2"
120 ON EOF (0) THEN 290
130 DIM F(62)
140 PRINT @32,26:2
150 KILL "DATA"
160 CREATE "DATA","U";500,560
170 OPEN "DATA";1,"F",A$
180 FIND 1
190 REM ***** READ AND WRITE HEADER **********
200 READ @33:A$,B$,C,D
210 PRINT A$,B$,C,D
220 WRITE #1,1:A$,B$,C,D
230 REM ***** READ AND WRITE DATA *****
240 FOR I=2 TO 500
250 F=9999.9999
251 FOR J=1 TO 62
252 READ @33:F(J)
253 NEXT J
260 PRINT I,F(1)
270 WRITE #1,I:F
280 NEXT I
290 PRINT "NO ERRORS FOUND."
300 END
1000 REM MAG TAPE READ ERROR IN HEADER
1010 A$="BAD READ"
1020 B$=A$
1030 C=9999.9999
1040 D=C
1050 RETURN
```

Compress Memory Before Appending

by Brian K. Kleber
Department of the Army
St. Louis, MO

A program was too big to run without exceeding memory; therefore, it was broken into two subprograms. The first subprogram was executed, then most of it deleted; then the second subprogram was appended to the end of the first and processing was completed.

Initially, when we attempted to APPEND the second subprogram from the 4907, the attempt failed under program control, but could be accomplished manually. The failure symptoms were similar to those when the buffer is overflowed on the data communications interface; the system also locked up under BUSY.

To solve this, we did a memory compress before executing the APPEND:

```
1590!
    |
    |
1600 X=MEMORY
1610 APPEND "@SYSLIB/GEO.BIG";1620
1620 REM
```

PRINTing Arrays to Tape or Disc

by Howard Sanders
Tektronix, Inc.
Wilsonville, OR

When a PRINT statement is used to output data, the 4050 System first converts the data into an ASCII character string, then sends that string to the specified device. If the format of the data string is not specified, default formatting applies. The default formatting can make quite a difference in storage space taken up when you PRINT arrays to tape or disc. And when arrays and strings are PRINTed to the **same** file, their storage and retrieval methods must be considered.

When an array is output using a PRINT statement, the 4050 System formats it the same, regardless of where the characters are being sent (e.g., 4050 screen, tape, printer, etc.). It places two carriage returns before the array, adds spaces so every four elements of the array are aligned, and adds two carriage returns to the end of the array. If the array is two dimensional, the data is formatted in row major order with two carriage returns separating the rows. As you will see, the extra spaces mean that far more tape or disc space is used than that required just for the data. And, the extra carriage returns must be considered when inputting a string preceded by an array.

The following routine generated two arrays, one linear and one two-dimensional, and a character string. The arrays and string were then stored in various files. We'll look at the methods of storing and what they accomplished:

```
100 INIT
110 DIM A(20),B(2,10)
120 FOR I=1 TO 20
130 A(I)=I
140 NEXT I
150 FOR I=1 TO 2
160 FOR J=1 TO 10
170 B(I,J)=(I-1)*10+J
180 NEXT J
190 NEXT I
200 A$="**********"
210 FIND 1
220 PRINT @33:A
230 PRINT @33:A$
240 CLOSE
250 FIND 2
260 PRINT @33:B
270 PRINT @33:A$
280 CLOSE
290 FIND 3
300 PRINT @33:A;
310 PRINT @33:A$
320 CLOSE
330 FIND 4
340 PRINT @33:B;
350 PRINT @33:A$
360 CLOSE
370 FIND 5
380 FOR I=1 TO 20
390 PRINT @33:A(I);
400 NEXT I
410 PRINT @33:
420 PRINT @33:A$
430 CLOSE
440 FIND 6
450 FOR I=1 TO 2
460 FOR J=1 TO 10
470 PRINT @33:B(I,J);
480 NEXT J
490 PRINT @33:
500 NEXT I
510 CLOSE
520 FIND 7
530 PRINT @33: USING 600:A
540 PRINT @33: USING 610:A$
550 CLOSE
560 FIND 8
570 PRINT @33: USING 620:B
580 PRINT @33: USING 610:A$
590 CLOSE
600 IMAGE 20(XFD)
610 IMAGE FA
620 IMAGE 2(/10(XFD))
630 IMAGE FA
640 END
```

Using the EDITOR ROM to input the data on the files and then the Search command, all occurrences of carriage returns were replaced with "(cr)". The following illustrates how the array and character string were stored when statements 210 through 280 were executed.

```
list
    :(cr)
    :(cr)
    : 1                 2       3       4(cr)
    : 5                 6       7       8(cr)
    : 9                 10      11      12(cr)
    : 13                14      15      16(cr)
    : 17                18      19      20(cr)
    :(cr)
    :(cr)
    :**********(cr)
list
    :(cr)
    :(cr)
    : 1                 2       3       4(cr)
    : 5                 6       7       8(cr)
    : 9                 10(cr)
    :(cr)
    : 11                12      13      14(cr)
    : 15                16      17      18(cr)
    : 19                20(cr)
    :(cr)
    :(cr)
    :**********(cr)
```

A semicolon after the array variable will compact the array, separating the elements by one space only (statements 290 through 360).

```
list
    :(cr)
    :(cr)
    : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20(cr)
    :(cr)
    :(cr)
    :**********(cr)
list
    :(cr)
    :(cr)
    : 1 2 3 4 5 6 7 8 9 10(cr)
    :(cr)
    : 11 12 13 14 15 16 17 18 19 20(cr)
    :(cr)
    :(cr)
    :**********(cr)
```

However, the two carriage returns will still be added to the beginning and end of the array. There is no problem unless the array precedes string data. When this happens, the string input handles the extra carriage returns as normal delimiters. Therefore, two extra string inputs must be used to get past the extra carriage returns. To input the data stored in File 1 or File 2, the following code would be necessary:

```
100 INIT
110 FIND 1
120 DIM A(20)
130 INPUT @33:A,A$,A$,A$
140 END
```

An alternative is to print the array, a variable at a time suppressing all but the final carriage return with a semicolon (statements 370 through 510). Strings following these arrays would then be accessible in the normal manner.

```
list
    : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20(cr)
    :**********(cr)
list
    :(cr)
    : 1 2 3 4 5 6 7 8 9 10(cr)
    : 11 12 13 14 15 16 17 18 19 20(cr)
    :**********(cr)
```

And, of course, you may always use the PRINT USING statement to specify the exact format. The following examples illustrate some possible formats contained in statements 520 through 630.

```
list
    : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20(cr)
    :**********(cr)
list
    : 1 2 3 4 5 6 7 8 9 10(cr)
    : 11 12 13 14 15 16 17 18 19 20(cr)
    :**********(cr)
```

Group OPEN on the 4907

by John Carter
Tektronix, Inc.
Santa Clara, CA

The group OPEN command for the 4907 File Manager paves the way for files in a selected group to be accessed consecutively, without having to issue CLOSE and OPEN commands as each is accessed. There is no limit (other than disc space) to the number of files in the group.

To implement a group OPEN, you need special characters within the file identifier. These special characters specify the extent of library and file [1] access you want. Although we will use only one of these special characters in the following routine (the #), you can check out the others by referring to the 4907 File Manager manual section: HOW TO WRITE A COMMAND: SPECIAL CHARACTERS IN FILE IDENTIFIERS.

The files to be accessed in this routine are grouped under the first level library DBMPROG in a user library.[2]

```
DIR0,"@DBMPROG#"
DBMPROG/INIT
DBMPROG/FILEINIT
DBMPROG/RECORDKEEP/OV1
DBMPROG/RECORDKEEP/OV4
DBMPROG/RECORDKEEP/OV7
DBMPROG/RECORDKEEP/OV3A
DBMPROG/RECORDKEEP/MAIN
DBMPROG/RECORDKEEP/TAPETODISC
DBMPROG/RECORDKEEP/OV3
DBMPROG/RECORDKEEP/OV6
DBMPROG/RECORDKEEP/OV7A
DBMPROG/RECORDKEEP/OV0
DBMPROG/RECORDKEEP/OV2
DBMPROG/RECORDKEEP/OV5
```

Now let's take a look at the routine.

```
100 INIT
110 REM * ASSUMES DRIVE 0 *
120 DIM S$(300)
130 PRINT "ENTER THE LIBRARY TO OPEN: ";
140 INPUT A$
150 REM * BUILD A$ WITH PROPER SYMBOLS *
160 REM * A$="@NAME#" *
170 A$="@"&A$
180 A$=A$&"#"
190 OPEN A$,"G"#1,"R",S$
200 ON EOF (1) THEN 410
210 REM * RESET EOF FLAG *
220 E=0
230 REM * GET THE FILENAME OPENED FROM STATUS *
240 A$=SEG(S$,189,LEN(S$)-189)
250 PRINT A$
260 REM * GET DATA FROM FILE *
270 INPUT #1:A$
280 REM * PRINT JUST A PORTION OF IT *
290 A$=SEG(A$,1,6)
300 REM * PLACE CURSOR BACK ON SAME LINE
310 REM FOR OVERWRITE TO SAVE SPACE *
320 PRINT A$;"C"
330 REM * IF EOF FLAG NOT SET THEN GET NEXT RECORD *
340 IF NOT(E) THEN 270
350 REM * OTHERWISE GET NEXT FILE *
360 CALL "NEXT",1,S$
370 REM * IF STATUS IS NULL THEN QUIT *
380 IF S$<>"" THEN 220
390 END
400 REM * SET EOF FLAG *
410 E=1
420 PRINT
430 RETURN
```

Initializing the system sets the disc drive to 0 and the default first level library to SCRATCH. Including @ in the file identifier[3] in statement 170 directs the File Manager to the user library specified in A$, (statement 140). Since the special character # directly follows the user library name (DBMPROG), this instructs the File Manager that all files in all levels below this library will take part in the group OPEN.

Statement 190 issues the OPEN command, names the file identifier (with special character included), specifies G for group open, identifies this group as logical file number 1, specifies a read only operation for the files, and sends the status message of the first accessed file to S$.

The variable which will denote End of File is set to 0 in statement 220.

The processing begins at statement 240 which takes a look at the filename (the first 189 characters in the status message are file parameters) of the just opened file, and then prints the name.

Statement 270 assumes that the accessed files contain ASCII programs or sequential ASCII data.[4] This simplified processing routine inputs an ASCII string, prints part of it and gets the next. When the end of file is reached, the CALL "NEXT" statement in 360 closes this file and opens the next. The parameters of CALL "NEXT" are: logical file number for the group and target string for the status message. Statement 380 checks for a null string which would signal that all files within the selected group have been processed.

The following output[5] is generated as a result of the above routine.

```
RUN
ENTER THE LIBRARY TO OPEN: DBMPROG
DBMPROG/INIT
■■0 ■■
DBMPROG/FILEINIT
■■0 ■■
DBMPROG/RECORDKEEP/OV1
1■0 ■■
DBMPROG/RECORDKEEP/OV4
1■0 ■■
DBMPROG/RECORDKEEP/OV7
1■0 ■■
DBMPROG/RECORDKEEP/OV3A
1■0 ■■
DBMPROG/RECORDKEEP/MAIN
■■0 ■■
DBMPROG/RECORDKEEP/TAPETODISC
■■0 ■■
DBMPROG/RECORDKEEP/OV3
■■0 ■■
DBMPROG/RECORDKEEP/OV6
■■0 ■■
DBMPROG/RECORDKEEP/OV7A
■■0 ■■
DBMPROG/RECORDKEEP/OV0
■■0 ■■
DBMPROG/RECORDKEEP/OV2
■■0 ■■
DBMPROG/RECORDKEEP/OV5
■■0 ■■
```

Regardless of the type of processing you're doing, you can see how the group OPEN command will save time and memory when you're going to be manipulating related files.

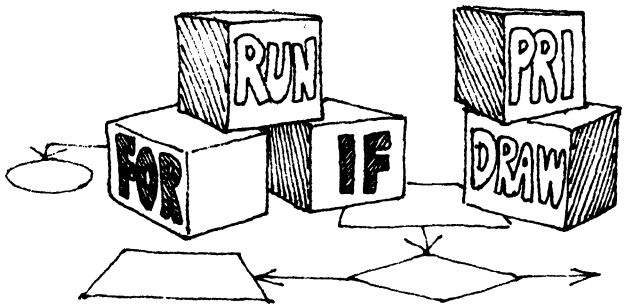
[1]For a review of library/file storage structure, see the 4907 File Manager Manual section: STORAGE STRUCTURE.

[2]There are three types of first level libraries: the SYSTEM library, the SCRATCH library, and any number of user libraries.

[3]If the @ sign had been omitted the File Manager would have looked for DBMPROG under the default first level library—SCRATCH in this case.

[4]Random ASCII files would require a record number in the command: INPUT #1,n:A$ where n is the record number. Binary files would require a READ statement and if random, a record number, followed by the variable to match the data in the file.

[5]Files are opened in the order they are stored on the disc. In cases where you have to access files in a particular order then the group OPEN probably won't suffice, and individual OPENs will be necessary.



BASIC BITS

Quotation Mark Display in PRINT USING Statements

by Dan Taylor and Patricia Kelley
Tektronix, Inc.
Wilsonville, OR

TEKniques Vol. 4 No. 2 told how to display quotation marks with a PRINT statement. Now let's look at quotation mark display in the PRINT USING form of the PRINT statement.

You use the PRINT USING statement when you want to tell the 4050 System exactly how you want some data formatted for output (either storage or display). This is done through a format string which contains special characters* to guide the format of your data.

There are three ways to set up this format string. And how you specify quotation marks to be displayed depends on which method you use.

You can include your format string in the PRINT USING statement; you can assign it to a string variable and then include the string variable in your PRINT USING statement; or, you can include the format string in an IMAGE statement and refer to the line number of the IMAGE statement in your PRINT USING statement.

In all of these methods quotation marks are used as delimiters: delimiters for the beginning and end of the format string itself, or delimiters for literal strings within the format string. So let's take the three format string methods and dissect them.

Format string included in the PRINT USING statement

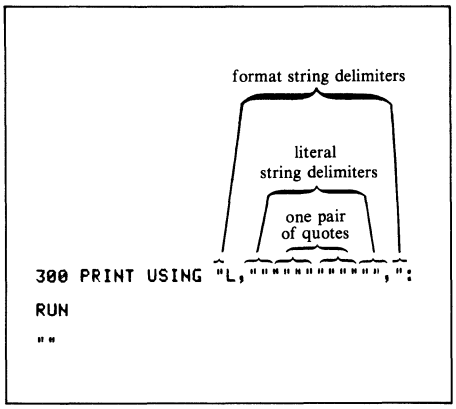
The format string begins with one quotation mark which signals the beginning of the string. The special characters and their modifiers are then inserted; commas can be used to separate a field and its modifiers from another field and modifiers. The format string is delimited by a quotation mark. The colon delimits the command; following the colon is the data to be formatted.

```
100 INIT
110 A$="ANSWER"
120 A=5800
130 PRINT USING "L,14A,6D,":A$,A
RUN
ANSWER          5800
```

If you include a literal string within the format string, it must be delimited by double quotation marks.

```
200 PRINT USING "L,""The answer is: "",6D,":A
RUN
The answer is:   5800
```

If you want to include quotation marks for display, you must have four for each one you want displayed. Since they are a literal string, they must be enclosed within double quotation marks. And, of course, the format string must be delimited.



However, you normally include quotation marks with other alphanumeric characters. And when building your format string to display quotes, be careful in using commas for separation of fields. The format strings below both contain the same alphanumeric characters but comma placement results in different literal strings being displayed.

```
400 PRINTUSING"L,""He said,""""The answer is:"",6D,""""""""",":A
RUN
He said, "The answer is:   5800"
```

versus

```
400 PRINTUSING"L,""He said,"","""The answeris:"",6d,""""",""""",":A
RUN
He said, The answer is:   5800
```

But as you can see, including the format string with the PRINT USING statement can result in quite a lengthy statement, perhaps overruning the 72 characters allowed. Therefore, a format string is often assigned to a string variable.

Format strings assigned to a variable

The following examples assign the first four format strings above to string variables which are used with the PRINT USING statement. The delimiters in the format string are the same. This method simply removes the format string from the PRINT USING statement line.

```
100 INIT
110 A$="ANSWER"
120 A=5800
130 F$="L,14A,6D,"
140 PRINT USING F$:A$,A
RUN
ANSWER          5800
200 F$="L,""The answer is: "",6D,"
210 PRINT USING F$:A
RUN
The answer is:   5800
300 F$="L,""""""""""""","
310 PRINT USING F$:
RUN
""
400 F$="L,""He said, """"The answer is: "",6D,""""""""","
410 PRINT USING F$:A
RUN
He said, "The answer is:   5800"
```

Format String in the IMAGE statement

But to assign all format strings to string variables could use up a lot of these

***For a complete list of the special characters and their modifiers, see the 4050 Series Graphic System Reference Manual, INPUT/OUTPUT Operations section, "The Image Statement: Field Operators and Field Modifiers."**

variables. Therefore, the IMAGE statement provides a method of specifying format strings. This statement doesn't require that the format string be delimited, thus it requires only one set of quotation marks to delimit any literal string, and two quotation marks for every one you want to display. The examples use the same format specifications contained in the string variables above.

As you can see, if you're going to be printing quotation marks, the best method is the IMAGE statement. But now you have an overview of how to display quotation marks in any of the three format specifications.

```
100 INIT
110 A$="ANSWER"
120 A=5800
130 IMAGE L,14A,6D
140 PRINT USING 130:A$,A
RUN
ANSWER          5800
200 IMAGE L,"The answer is:",6D
210 PRINT USING 200:A
RUN
The answer is:  5800
300 IMAGE L,""""""
310 PRINT USING 300:
RUN
""
400 IMAGE L,"He said, ""The answer is: ",6D,""""
410 PRINT USING 400:A
RUN
He said, "The answer is:   5800"
```

Sizing Viewport for 4662 Plotter or 4050 Screen.

by Martin Guiver
Tektronix, Inc.
European Marketing Centre
Amstelveen, The Netherlands

Try this for "size:"

```
9000 VIEWPORT 0,130+20*(D<>32),0,100
```

where D = device #

The statement replaces the two statements described in TEKniques Vol. 3 No. 1, and runs faster.

4050 Series Applications Library Program Abstracts

Order

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

(The program material contained herein is supplied without warranty or representation of any kind. Tektronix, Inc. assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.)

Domestic U.S. Prices:

Documentation and listings	*$20 per program*
Recording Fee	*5 per program*
Tape Cartridge	*30 per tape*
Flexible Disc	*15 per disc*

Contribute

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

Forms

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

Outside U.S.

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

ABSTRACT # 51/07-0102/0

Title: **Gantt Chart**
Author: Connie Breithaupt
Tektronix, Inc.
Rockville, MD
Memory Requirement: 32K
Peripherals: 4907 File Manager
(For Disk Version Only)
Optional—
4631 Hard Copy Unit
4662 Plotter
Statements: Tape Version—1625
Disc Version—1617
Files: Tape Version
4 ASCII Program
3 Binary Data (Directory and Example)
Requires pre-MARKed data files
Disc Version
4 Binary Program
10 Binary Data (Examples)

The program generates a chart and graph of activity vs. time.

The following is input:

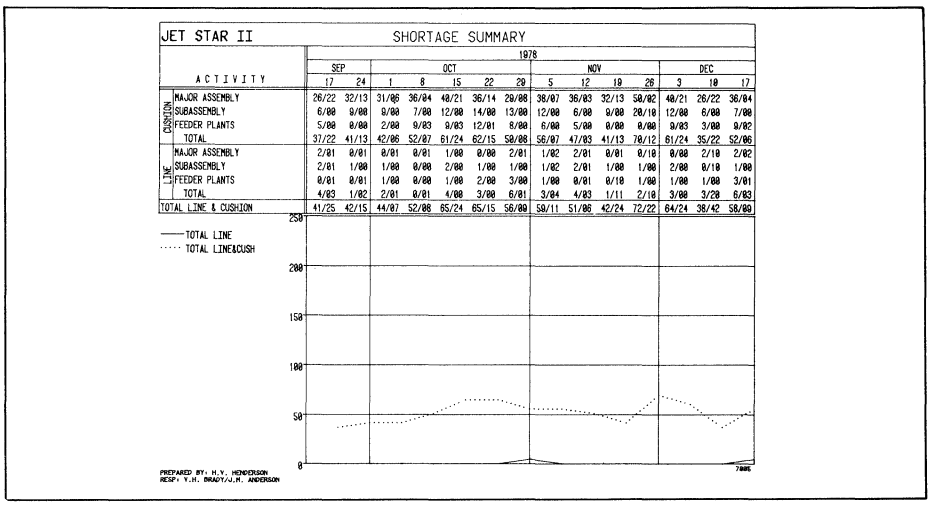
- Project name—2 lines—15 characters each
- Chart title—45 characters
- Code—16 characters
- Project Code—6 characters
- Signatures—2 lines—30 characters each
- Vertical row descriptions—single or double spaced—4 rows maximum
- Horizontal row descriptions—single or double spaced—16 rows maximum
- Weekly data for each row—28 weeks maximum—beginning with a Sunday

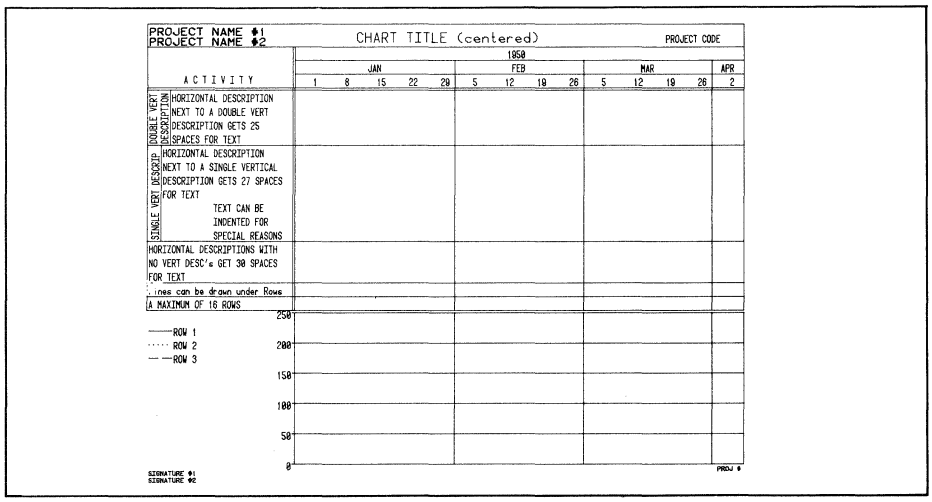
Charts are saved by name. File 5 of the tape version contains a directory of the charts by name.

Weekly data my be edited for each week or selected weeks.

Charts are generated for a 14 week period starting with a chosen beginning column date.

Three selected rows of data may be graphed on one chart. The first selected row will be graphed with a solid line, the second with a dotted line, and the third with a dashed line.





ABSTRACT # 51/00-0103/0

Title: **Capital Budget**
Author: David Mills
Tektronix, Inc.
Beaverton, OR
Memory Requirement: 32K
Peripherals: Optional—
4631 Hard Copy Unit
4662 Plotter
Statements: 712
Files: 1 ASCII Program
1 ASCII Data (Project Directory)
Automatically marks data files

The program will build and maintain files of current capital budget projects. Up to 50 projects may be stored on one tape, with standard information as:

Rank
Project description
Budget (by quarter)
Category
Originator
Actual cost
Cumulative capital

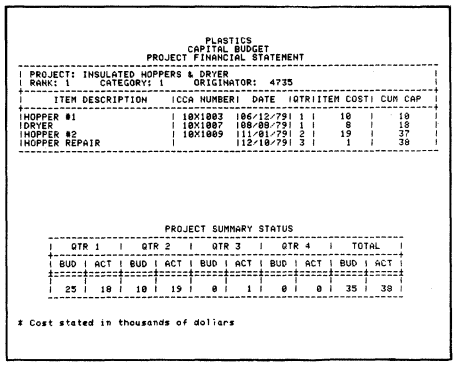
Up to 30 sub-items may contribute to a project including:

Item description
CCA number
Cost
Date
Quarter

All information may be examined or altered to correct errors or changes in budget, etc.

Charts may be produced for one, or all projects, giving total project cost, cumulative cost, budget vs. actual expenditures, etc.

Cumulative and quarterly graphs of budget vs. actual cost for any project, or the total of all projects may be drawn.



PLASTICS
CAPITAL BUDGET
PROJECT FINANCIAL STATEMENT

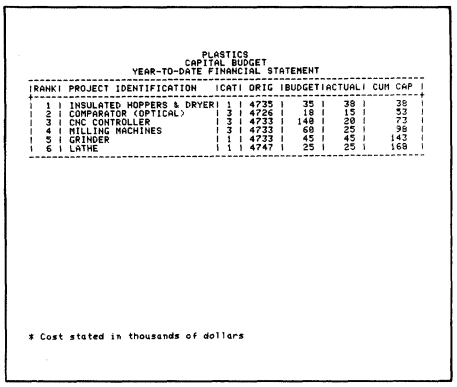
PROJECT: INSULATED HOPPERS & DRYER
RANK: 1 CATEGORY: 1 ORIGINATOR: 4735

ITEM DESCRIPTION	CCA NUMBER	DATE	QTR	ITEM COST	CUM CAP
HOPPER #1	10X1003	06/12/79	1	10	10
DRYER	10X1007	08/08/79	1	8	18
HOPPER #2	10X1009	11/01/79	2	19	37
HOPPER REPAIR		12/10/79	3	1	38

PROJECT SUMMARY STATUS

QTR 1		QTR 2		QTR 3		QTR 4		TOTAL	
BUD	ACT	BUD	ACT	BUD	ACT	BUD	ACT	BUD	ACT
25	18	10	19	0	1	0	0	35	38

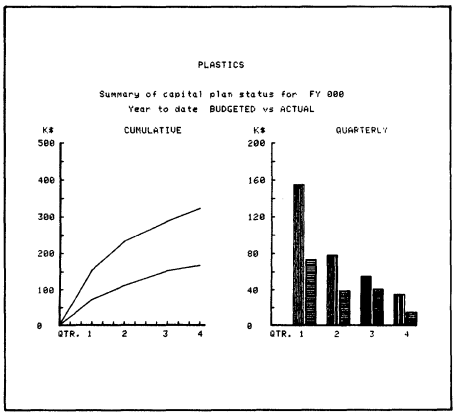
* Cost stated in thousands of dollars



PLASTICS
CAPITAL BUDGET
YEAR-TO-DATE FINANCIAL STATEMENT

RANK	PROJECT IDENTIFICATION	CAT	ORIG	BUDGET	ACTUAL	CUM CAP
1	INSULATED HOPPERS & DRYER	1	4735	35	38	38
2	COMPARATOR (OPTICAL)	3	4726	18	15	53
3	CNC CONTROLLER	3	4733	140	20	73
4	MILLING MACHINES	3	4733	60	25	98
5	GRINDER	1	4733	45	45	143
6	LATHE	1	4747	25	25	168

* Cost stated in thousands of dollars



ABSTRACT # 51/07-6108/1

Title: **RECORDKEEP II**
Author: Jim Gish
Tektronix, Inc.
Irvine, CA
Memory Requirement: 16K
Peripherals: 4907 File Manager
Optional—
4631 Hard Copy Unit
4641 Printer
Statements: 2098
Files: 13 ASCII Program

This is a revised version of 51/07-6108/0; the purpose of the revision was to make minor changes to the program and provide multiple character support for the 4054 Graphic System. Data created with the previous version is compatible with this revision.

Documentation has been added describing the process of adding additional tasks to RECORDKEEP II and descriptions of some of the utility subroutines contained in the program.

RECORDKEEP II allows tables and lists to be created and stored on the 4907 File Manager. Reports can then be quickly generated from this table (database) through the RECORDKEEP II commands that allow editing, sorting, and table listing.

Twelve commands are available. They are broken down into three functional groups. The first group provides the commands necessary to edit the table, which we will call a database or file. Commands in the second group include file management operations that allow saving databases and creating smaller tables from existing databases. These tables can also be sorted and listed to emphasize their content by the third functional group. RECORDKEEP II can be applied in areas where you need to store and manipulate small inventories, sales history records, personnel records, patient history, catalogues, and more.

ABSTRACT # 51/00-0719/0

Title: **Depreciation**
Author: P.C. Holman
University of Wisconsin
Stevens Point, WI
Memory Requirement: 16K
Peripherals: None
Statements: 459
Files: 5 ASCII Program

The program consists of five separate depreciation methods:

1. Straight line
2. Sum of digits
3. Sum of digits, switch to straight line
4. Double declining balance
5. Double declining balance, switch to straight line

Each program calculates a depreciation table using that particular method. Each contains examples and is tutorial.

ABSTRACT # 51/07-9538/0

Title: **4052/4 Drafting Program**
Author: Connie Breithaupt
Tektronix, Inc.
Rockville, MD
Memory Requirement: 64K
Peripherals: 4052 Graphic System,
Option 24
4907 File Manager
4952 Joystick, Option 2
or
4054 Graphic System,
Option 24
4907 File Manager
Optional—
4631 Hard Copy Unit
4662/4663 Plotter
4956 Graphics Tablet
Statements: 2321
Files: 2 Binary Program
1 Binary Data
Requires pre-MARKed data files

The program allows a drawing to be defined by creating, modifying and/or deleting its elements. The elements may be arrows, circles, lines, text, cross-hatching, and sub-drawings. The created drawing is called a Picture Data Base (PDB) and is stored on the 4907 File Manager. Each PDB may contain 100 different layers of display. For example, Layer 1 (the default layer) may show the PDB outline, Layer 2 the dimensions, Layer 3 the linework, etc. A frequently used symbol may be created as a PDB and then used as a Sub-PDB in other PDB's.

Program commands include:

CHANGE
- Dash line font
- Height of text

COMPRESS

DISPLAY
- All of current PDB
- Grid points
- Layer number

END

ERASE
- Arrow(s)
- Circle(s)
- Line(s)
- Sub-PDB(s)
- Text
- Cross-hatching(s)

INSERT
- Arrow—horizontal
- Arrow—vertical
- Circle
- Line
- Sub-PDB
- Text
- Cross-hatching

LIST
- Parameters—selectable
- PDB—contents

PAGE
- Down
- Left
- Right
- Up

PAUSE

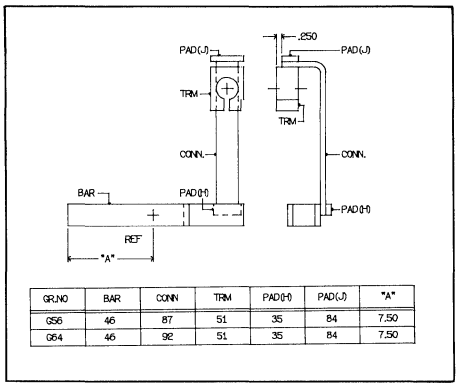
RESTART

SAVE

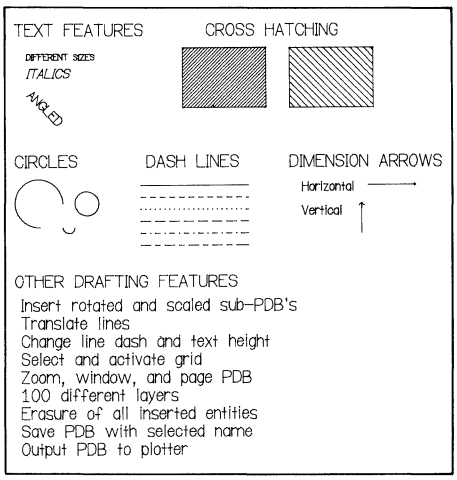
SELECT
- Angle—text
- Arrowhead size
- Dash code
- Figure (Sub-PDB)
- Grid interval
- Grid off
- Grid on
- Height—text
- Input device address
- Non-italic text
- Italic text
- Tolerance range
- Translation—amount
- Window
- Cross-hatching—angle, line displacement
- Layer number
- Output device address

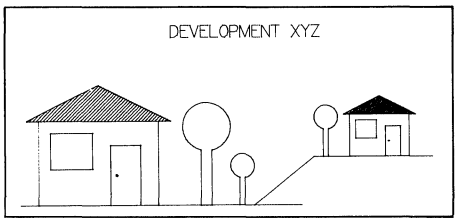
TRANSLATE
- Line—copies n times, increment, direction

ZOOM
- All
- Down
- Up
- Window



GR.NO	BAR	CONN	TRM	PAD(H)	PAD(J)	"A"
G56	46	87	51	35	84	7.50
G64	46	92	51	35	84	7.50





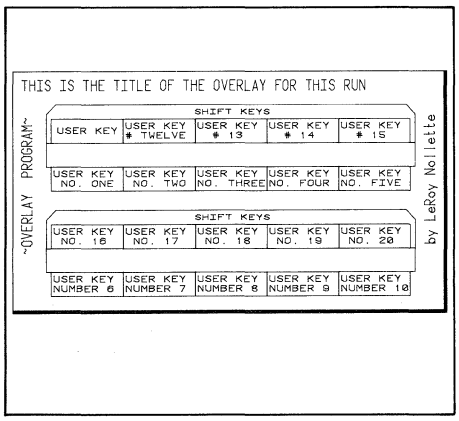
ABSTRACT # 51/00-9537/0

Title: **Overlay Drawing Program**
Author: LeRoy Nollette
Tektronix, Inc.
Wilsonville, OR
Memory Requirement: 16K
Peripherals: 4662 Plotter
Statements: 250
Files: 1 ASCII Program
2 ASCII Data (Sample Overlays)
Optional—Pre-MARKed data files

The program draws an overlay on the 4662 Plotter that can be cut out and placed over the User-Definable Keys. Key descriptions may be entered from the keyboard. Data may be saved on a pre-MARKed data file and redrawn at a later date.

The program may be modified (one line of code) to draw a large copy of the overlay and then reduce it on a copy machine having reductions capabilities.

By changing one line of code the user may preview the overlay on the screen.



Tektronix®
COMMITTED TO EXCELLENCE

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

ADDRESS CORRECTION REQUESTED

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