

# Tekniques

The 4050 Series Applications Library Newsletter September 17, 1979 Vol.3 No.6



The Water Quality Branch of Environment Canada is concerned with monitoring and preserving the quality of Canada's water resources.

## 4051 HELPS STUDY WATER QUALITY PROBLEMS

by Simon Whitlow and Maureen Lamb  
Fisheries and Environment Canada  
Ottawa, Ontario, Canada

The Water Quality Branch of Environment Canada monitors the quality of waters in Canada to establish baseline information and to identify water quality trends on a national and regional scale. Analytical results, which are measured in five regional laboratories across Canada, are stored on NAQUADAT, the NATIONAL water QUALity DATA bank. a TEKTRONIX 4051 Graphic System is helping scientists utilize the data by providing effective ways of visualizing relationships between parameters, as well as facilitating the exchange of data among different computers.

### DATA DISPLAY

Ionic proportion diagrams such as the one in Figure 1 graphically illustrate the principal constituents in water chemistry. In this diagram, the relative proportion of each of the major ions are plotted on the anion (left) or cation (right) side of the "rose-type" figure. The concentration in

milliequivalents per litre of the individual ions are calculated as percentages of either the anionic or cationic field, and drawn such that the length of the bisecting radius in each of the sectors is proportional to the equivalent concentration of that ion. Tic marks are used to indicate 5% intervals, and since there are four ions in each field, the circumference of the circle represents 25 equivalent percent. The predominately  $\text{CaHCO}_3$  water type illustrated in Figure 1 is typical of much of Canada's surface waters.

### In This Issue

4052/4054 Processor Enhancements.....	3
Business Planning Graphics at Boise Cascade	7
Editor's Note.....	9
Programming Tips .....	10
Basic Bits .....	14
New Abstracts.....	16
Library Addresses .....	20

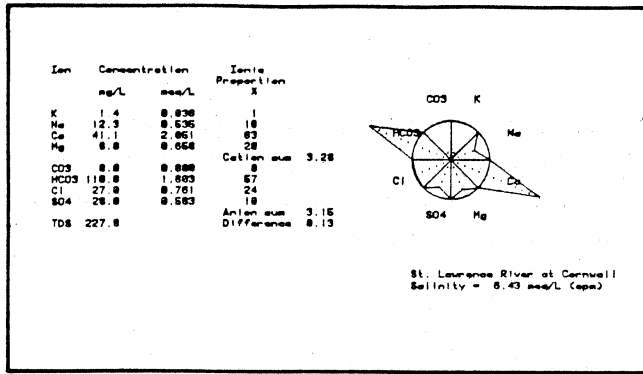


Fig. 1. An ionic proportion diagram showing the major ion chemistry (median values) of the St. Lawrence River at Cornwall, Ontario.

An alternative visual display is provided by the point density triangular diagram shown in Figure 2. In this form of display, the relative proportion of each ionic species is suggested by its location with respect to the three apices. Diagrams such as this are useful in characterizing large water basins and in picking out anomalous water types.

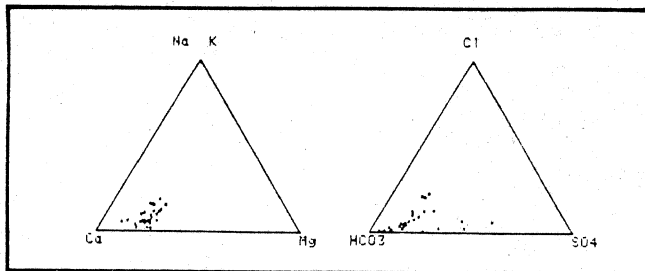


Fig. 2. A point density triangular diagram showing predominant water types at 30 sites in Eastern Canada.

The 4051 can be used to perform a "mini-search" and plot of a parameter versus time for a subset of the NAQUADAT data base. In Figure 3 a file on the 4051 internal tape unit is searched and an appropriately scaled plot of conductivity as a function of time is produced.

All of these display routines can be output either to the hard copy unit or the 4662 plotter. The ability to change the plot dimensions using the SCALE command, or to define the plot size using the plotter joystick, has proved very useful in producing output for publications or technical presentations.

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  
Managing Editor  
Editor  
Technical Editor  
Graphic Design  
Circulation

Ken Cramer  
Patricia Kelley  
Terence Davis  
Dan Taylor  
John Ellis  
Rory Gugliotta

Copyright © 1979, Tektronix, Inc.  
All rights reserved.

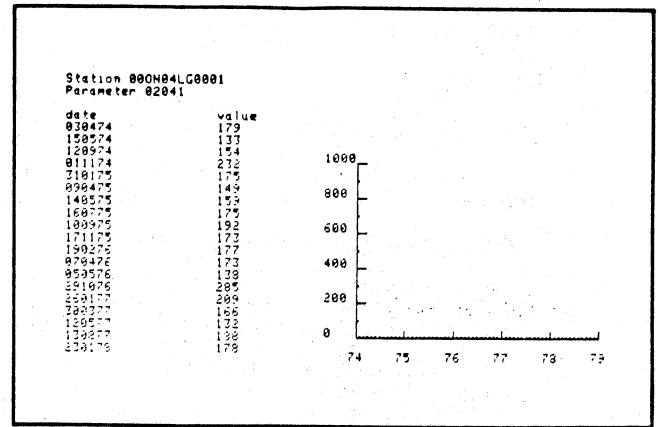
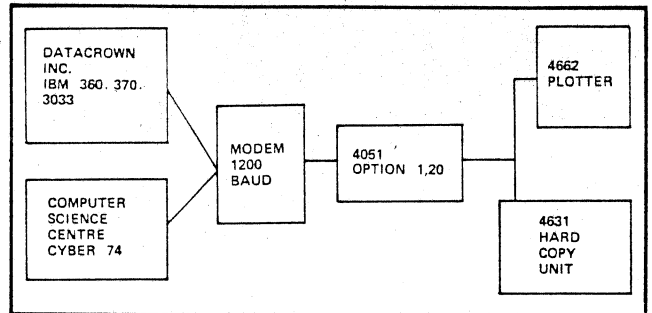


Fig. 3. A "mini-search" showing sampling dates and conductivity values ( $\mu$ S/cm) on the Moose River in Northern Ontario.

## DATA EXCHANGE AMONG COMPUTERS

The large-scale computing requirements of the Water Quality Branch are satisfied by two time-sharing service bureaus. Datacrown Inc., the largest time-sharing facility in Canada, is accessed to manipulate the NAQUADAT database while the Computer Science Centre, a government facility, is used to produce extensive graphic output. The way in which the 4051 fits into our computing environment is shown in the following Configuration Diagram:



Using the Option 1 data communications interface with a modem and telephone line, a number of useful and practical operations are being employed:

- Simple Communications:** With the 4051 it is possible to communicate at 1200 baud with either of the service bureaus. Advantages over our previous electrostatic printer-type terminals are increased speed and a saving of paper during routine data transmission.
- Plot Previews:** Associated with our NAQUADAT database are a number of relatively sophisticated, FORTRAN-based, plotting programs. These are used to produce high-quality plots on a large Calcomp plotter at the Computer Science Centre facility. With the 4051 it is now possible, using TEKTRONIX Plot 10 and Calcomp preview software, to scale the plot output to our screen and see it before it has been plotted. This permits verifying that a plot has been correctly created, or cancelling an incorrect plot before it has been drawn.
- Saving Plots:** We have found it useful to save certain

types of plots on the 4051 internal tape drive for subsequent recall. In this application, a plot created at the Computer Science Centre, is previewed and saved on a 4051 tape file. It is then possible to subsequently retransmit the plot from the file back to the screen or to the 4662 plotter as often as required. An advantage of this technique is the additional flexibility permitted in producing the final plot. The plot can be slightly rescaled, drawn in different colours or reversed for preparing transparencies. Figure 4 shows a hydrograph prepared "off-line" in this way.

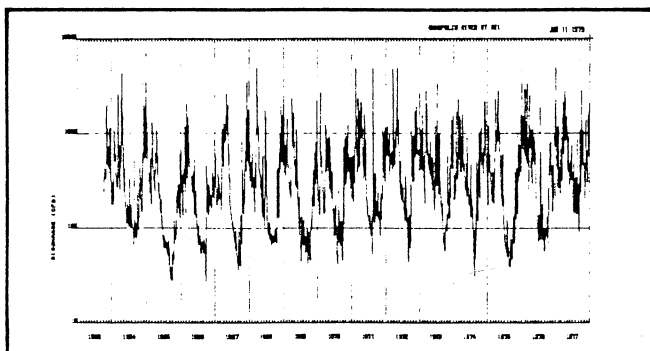



Fig. 4. A 15-year hydrograph showing the flow (cfs) of the Annapolis River in Nova Scotia.

d. **Transferring Data:** As can be seen from the configuration diagram, we frequently need to get water quality

data stored at one service bureau to another facility for plotting. Our traditional method has been to transfer cards or magnetic tapes back and forth although this has meant delivery costs and time delays. Recently, we have begun to use the 4051 to assist in this data transfer between the IBM and CDC computers at the different service bureaus. We have written 4051 BASIC programs to set up the communication protocols required by the different systems, so that now it is a relatively simple procedure to sign on to the first computer and store the data into a working file for plotting. We have found this to be an expedient way of transferring data and producing our plots.

The statistical packages available with the 4051 are very useful in our work. In particular, the Analysis of Variance routines have been helpful to us in studying and quantifying cross-stream versus down-stream variations in water quality conditions. Until now, data has been entered each time from the 4051 keyboard. In the future, however, we hope to work out a procedure whereby we can make use of the statistical routines without having to manually reenter data which is already stored at the service bureau. Such a procedure would involve data selection and formatting at the service bureau, followed by data transfer to the 4051 for analysis and result presentation. 

## 4052/4054 Processor Enhancements — The Inside Story

by Dave Barnard  
Tektronix, Inc.

Articles in the last few issues of TEKniques have paid passing tribute to the new processor in the 4052 and 4054. These earlier articles explained **what** the results are in terms of performance, but stopped short of explaining how or why these results came about. This article starts where those previous articles left off.

Any discussion of how the 4052 is faster than the 4051, yet compatible with 4051 programs or data, leads inevitably to the choices of technology employed and the philosophy of design. The very accomplishment of compatibility of function, computation, and family appearance seems at times to disguise the 4052, for example, as a 4051 look-alike, or the 4054 as a bigger 4051. Instead, this compatibility has been a consideration at the heart of engineering—intensive redesign of the entire inside of the products.

The first operational feature that you might notice on the 4052 is in the tape drive. Pick your favorite program tape, turn on the 4052 and load your program. If the program tape was not positioned with the file you wanted at the beginning, the 4052 found the file without returning to the beginning of the tape. It also loaded it noticeably faster.

Try running your program; you'll wonder where we put the racing stripes. If you looked inside, you wouldn't find them either, but you would find a new processor board, and memory board.

### Going Beneath the Surface of the 4052

If you heard somewhere that the 4051 processor uses an LSI 8-bit microprocessor, you would notice something else startling about the 4052 and 4054: they don't use a microprocessor.

The new processor of the 4052 and 4054 is significantly different from the processor of the 4051. Comparing the organization of the 4052 and 4054 with that of the 4051 will clarify the differences.

### Vis-a-vis the 4051

The 4051 Graphic System (Figure 1) is organized around a central address, data and control bus emanating from an 8-bit, 6800-type microprocessor. PIAs (peripheral interface adaptors) link main peripherals to the processor in a manner similar to the way terminals are linked to a central computer.

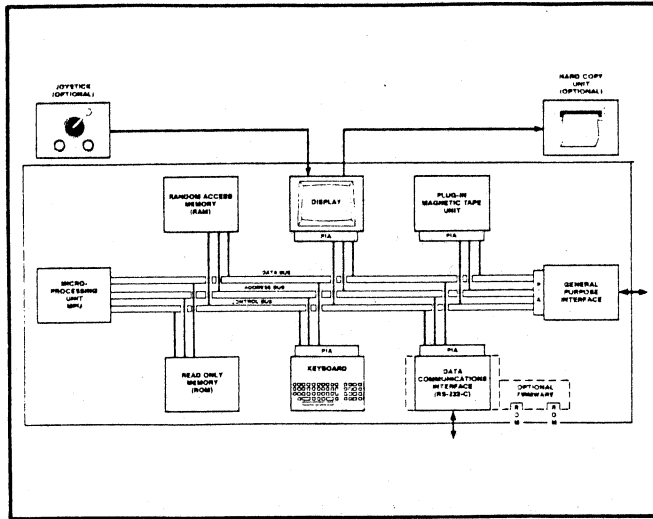


Fig. 1. 4051 Block Diagram.

The important features of the 4051 system are:

- 4050 BASIC program and data reside in Random Access Memory (RAM) expandable to 32K bytes.
- System operating program (firmware), which interprets your BASIC programs and contains constants such as the value of Pi, reside in Read Only Memory (ROM).
- The microprocessor gathers its instructions, data, or BASIC program information one 8-bit byte (character) at a time.
- Arithmetic operations are performed by the microprocessor 8 bits at a time.
- Extensions to BASIC are permitted through an area in ROM referred to as the 8K byte bank switched area. (If you plug in an Editor ROM Pack — 4051R06 — it acts as if it virtually resides here when you use its commands.)
- The display module, keyboard, tape cartridge drive, communications option, and GPIB (IEEE-488 Std) bus are processor peripherals.

### Not Just Another Microcomputer

The 4052 and 4054 share some of the same fundamental structures of the 4051. However, they also differ significantly (Figure 2).

- The RAM memory is larger (32K bytes standard). An option expands memory to 64K bytes (56K bytes user-accessible).
- System operating program (firmware) is larger because it contains the Matrix Functions and Binary Program Loader Function equivalent to the 4051R01 and 4051R05 ROM Packs, respectively.

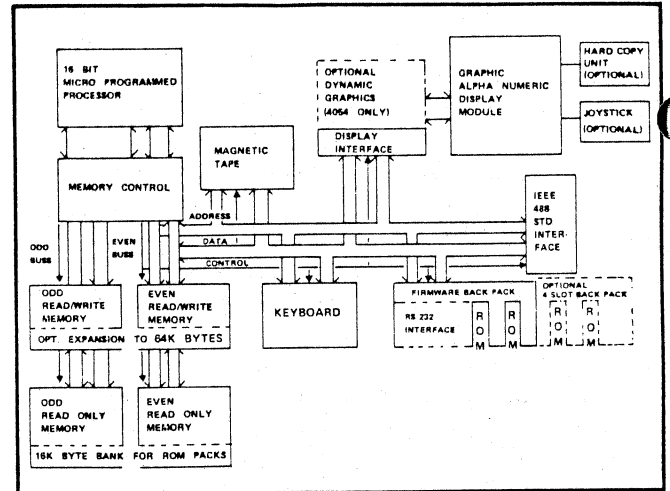


Fig. 2. Simplified 4054 Block Diagram.

- The entire memory is split into independent even and odd halves, which join together in the "Memory Control" section to provide the processor with twice as many bits as the 4051. This means: the processor is free to gather its instructions, data, or BASIC program information two 8-bit bytes at a time.
- The processor is a 16-bit processor able to perform 16-bit arithmetic (more on this later).
- Extensions to BASIC are added by ROM Packs as they are with the 4051. The added optional 4-slot backpack and the larger ROM Pack bank switched area (16K bytes) allows more extensive ROM Pack extensions to product performance than with the 4051.

Some of the differences may seem, at first, unrelated to improved features that a 4052 or 4054 user could enjoy.

For example, the structure of the new memory may sound like an insignificant detail. However, from the point of view of someone writing or running a program, splitting the memory into even and odd halves allows it to look like 4051 memory. The MEM and SPA commands return quantities of **bytes** just as with the 4051. Data is READ, INPUT, PRINTed; programs are OLDed, APPENDED, and SAVED in 8-bit bytes compatible with 4051 operations. But the result of the split memory and faster odd/even busses is to present to the processor the ability to gather twice the number of bits simultaneously. The memory and bus organization are designed to match the 16-bit processor. Thus, compatible operation is provided without compromising performance enhancement.

Although program expansions provided by ROM Packs must be accessed through a path only 8 bits wide, the path is faster in the 4052 and 4054 processor. The new processor pulls in the external program words in half the time required in the 4051. However, this does not imply that functions in ROM Packs will run only twice as fast on a 4052 or 4054. Usually, ROM Packs make use of strings of code in the main processor (floating point calculations,



putationally equivalent to the 6800 single chip processor even though it's a totally different processor.

Level 4: Processor Microprogram (resides in fast ROM). Written in a language meaningful to the bit-slice processor, this program directs execution of instructions in parallel and in stacked (pipeline) fashion.

#### Enter the Bit-Slice Processor

The combination of four 4-bit (Figure 4) processors receives its instruction information from a 56-bit pipeline register. While the current micro-instruction is held during execution on the processor side of the register, the next instruction is being found and placed in line. The remaining bits (of the 56) not used by the bit-slice processor send instructions to other portions of the processor system to direct operations **concurrent** with bit-slice processor operation. For each firmware instruction (obtained from ROM), one or more bit-slice instructions stream through the pipeline system. The firmware instruction location is determined by the **program counter** now located on the memory control section of the processor system. The memory controller houses other devices which perform partial decoding of firmware instructions to anticipate memory needs during execution. Thus, the fast bit-slice processor quickly handles only the details of program execution, while at the same time the surrounding system carries out memory operations and readies the next instruction.

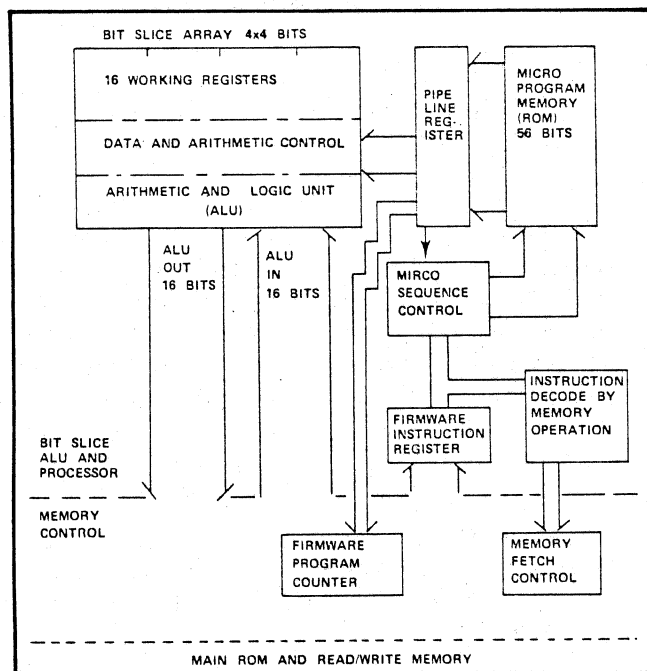


Fig. 4. Bit-Slice Microprogrammed Processor.

The bit-slice processor provides more temporary storage for arithmetic calculations (sixteen 16-bit registers rather than two 8-bit registers). Thus the registers can be loaded with a complete pair of 4050 BASIC floating point

variables. Once in working registers they may be operated on, using new floating point instructions at top speed.

Keep in mind that although there are more levels of hierarchy than before, the new processor works at several levels simultaneously! The microprogram words (level 4) are stored in a fast memory whose speed approaches that of cache memory used in large computers.

Unlike the 6800-based 4051, the 4052/4054 processor crunches its way through programs several bytes at a time and queues them up while partially executing them. Meanwhile a 6800 would be picking up a single 8-bit byte at a time.

#### What Is Bit-Slice Technology; What Does It Do For Me?

Probably you've asked, "What is bit-slice technology?" The words "bit-slice" refer to the way that the computing functions have been diced up into convenient levels of complexity to be fitted onto a single piece of silicon by the manufacturer.

Many computing products use devices divided along purely functional lines. For example, numerical calculations require something which adds and subtracts, performs shifting, comparing, and logical operations on data. They also require temporary storage registers to hold intermediate results or to store subroutine return addresses, or offset addresses. Functional digital circuits exist which contain 4 or 8 or 16 bits of arithmetic and logic functions, for the former, and also register functions for the latter purpose. Typically, minicomputers employ this functional division using many parts, although many of the manufacturers are switching to the flexible bit-slice processor.

At the other extreme is the fully integrated technology that places all of the processor functions on a single piece of silicon. The problem here is how to make it all fit. Use of high circuit density fabrication techniques lead to single chip computers that require using circuitry which has difficulty sending signals fast enough to the rest of the system. It's a speed-density trade-off. The 4051 processor uses the fully integrated technology to achieve computing power in a smaller space than was possible using the functional components. Further comparisons are shown in Table 1.


Technology Type	How Sliced	Parts Required for Typical Product (Excluding Memory)	Instruction Set Flexibility/Who Designs	Relative Speed	Word Width Modularity	Example
Bipolar MSI LSI	By Function	100-500	Complete flexibility hardware designer	Fast	Any No. of bits	Large mainframe or minicomputers
Bipolar bit-slice	By bits (4 slice)	40-60	Completely flexible system designer	Fast	Multiples of 4 bits	Some minicomputers 4052, 4054
Single Chip MOS LSI	Not sliced, fully integrated	3-10	Fixed cast in silicon by chip designer	Medium to slow	Not Modular	4051, home computers, calculators

Advantages of the technology choice to users of the 4052 and 4054 are:

- Processing speed with a simpler more reliable design than would have otherwise been possible.
- The modularity of the technology provided the desired processor word width without resorting to another 16-bit fixed instruction set single chip microprocessor, with its attendant differences from the 6800 8-bit processor, and potential product compatibility problems.
- The instruction set flexibility allowed Tektronix to custom tailor the processor and its repertoire of actions (instructions) to suit problem solving with 4050 BASIC and graphics. In other words, the flexibility

permitted an efficient processor (does things fast and well) to be made **effective** (does only the things needed and desired) from a user's point of view.

### Products Don't Design Themselves

The 4052 and 4054 are the result of many man-years of well executed efforts by a design team. Stringent goals were set and met. The well managed, team effort was squarely aimed at accomplishing a clear, single purpose: to bring greater computation and graphics power to within an arm's reach of individuals whose everyday work requires computation and meaningful graphic products to raise their productivity in decision making, analysis, design and instruction in their offices, laboratories, and classrooms around the world. 

---

## Business Planning Graphics at Boise Cascade

by Terry Davis  
TEKniques Staff



Gretchen Pierce, Manager of Planning Development at Boise Cascade Corporation, watches Susan Sandstrom, Planning Analyst, compute and plot Boise Cascade's annualized dividend rate using the Tektronix 4051.

The 4051 Graphic System has become an invaluable aid to the Planning Development staff at Boise Cascade Corporation, where graphics aid in planning the business future. And the commitment to graphics didn't start with the 4051. Rather, the benefits of graphics to ease and clarify financial communication became apparent to planners and managers. Subsequently, the 4051 was chosen as a natural tool to allow more graphics planning at all levels, and more conveniently.

Planners at Boise Cascade were faced with the same problem that all growing companies run into: financial reports were becoming too large to be easily used. There were more and more pages, each packed with business data, but these thicker volumes made business trends harder to see. Communicating those trends was even harder, and communication is an important key to the success of a business plan.

When graphics came into the pages, communication of management data quickly improved. At first, all of the graphics were prepared by hand, from manually calculated trends and growth rates. Preparing one financial report sometimes required the efforts of seven people to prepare the graphics — and these individuals were not professional illustrators, but management and staff. It was a waste of manpower.

Accuracy was an additional concern, especially at the end of eight hours preparing bar charts and graphs. Most of the graphics were done by cut-and-paste, with a typesetter providing the chart labels. But in spite of the difficulties, graphics really caught on; charts and graphs began to appear in all sorts of planning and management reports. The 4051 was chosen as an aid to this effort. Not only could this machine compute the financial trends, but could provide the graphic representations more easily, more quickly, and with more assurance of accuracy.

### A Little About Boise Cascade

Boise Cascade is an integrated forest products company that employs over 37,000 people. The corporate headquarters are in Boise, Idaho.

More than a lumber or paper company, Boise Cascade is organized into four operating groups. The Paper Group and the Timber and Wood Products Group are two of these. A third, the Building Materials Group, includes building materials distribution, cabinets and housing. The Packaging and Office Products Group includes corrugated containers, composite cans, envelopes, and the distribution of office equipment and supplies. In order to manage these diverse industries, it is important not only to generate plans, but to communicate them in an understandable fashion.

## What Kinds of Reports?

The Planning Development Department at Boise Cascade, managed by Gretchen Pierce, maintains the "interactive corporate strategy process," one of the processes by which the company is managed. Individual business units within each division have their own planners and do their own plans. The Planning Development Department, among other things, reviews, analyzes, and develops a point-of-view about each of these plans.

## Comprehensive Business Review and Strategic Plans

Two documents developed in the planning process are a comprehensive business review (an analysis of the attractiveness of an industry), and a strategic plan (a long-term business plan that is revised periodically). The Comprehensive Business Review analyzes the current and projected operating environment of all business units. It examines the company's position in the industry, and where the industry is going. This helps the business units plan the position in which each might want to be, and the role of that business unit within the company. Graphics are used extensively in communicating these positions, trends, and projections throughout company management. From this comprehensive plan, each business unit develops its strategic plan to get to its desired position.

## Five-Year Plans and Economic Scenario

Once the business unit's future strategy is written and approved, then every year it submits a five-year business plan. This plan is tactical in nature, in support of the strategic plan. However, turning out this business plan often requires many reports at a sub-unit level, for each business unit has many components. Additionally, to provide a basis against which the business units may develop their business plans, each year the Corporate Planning Department with the concurrence of corporate management prepares an economic scenario taking into consideration current legislation affecting the industry. The business plans then provide the foundation for detailed key budgets.

## Final Reports

The Planning Development Department reviews the individual plans to make sure they are consistent with the individual strategies and with the corporate objectives. The Department consolidates all of the financial information to give an overview of where the corporation is going, the capital requests are merged and tabulated, and a point-of-view is developed on each of the plans. The result is a document which makes extensive use of graphs to identify the corporate and business unit highlights.

In addition, a report for the board of directors is prepared. Other documents contain reports of major studies, for example, the operating trends and results of Boise Cascade and their major competitors. 90% of that document is graphic communication.

## Other Uses of the 4051

The 4051 is not only used in preparing formal reports. The Planning Development Department and business units headquartered in Boise test theories using the 4051. They review their capital budgets, keying data into the 4051 to look at it graphically. It gives them a flavor for what's happening; it helps them analyze where they are. They can graphically see where it fits.

Many of the divisional people use 4051-produced graphics in their internal, informal communication.

The corporate financial arm communicates to the board of directors five times a year. Critical ongoing information is gathered together and put into a book. Certain standard slides bring the board up to date — all in graphics.

## The Program Tape

The 4051 Data Graphing program was the first one used, and it worked well for quite awhile. However, as proficiency in using the 4051 and understanding its capabilities grew, more flexibility and variety was desired. Since Boise Cascade people had neither the time nor the inclination to do their own programming, they looked for outside help. Dr. Dick Reimann, a physics professor at Boise State University (and contributor to the 4050-Series Applications Library) was asked to write some programs that would provide business plans and forecasts in a flexible graphic format. Dr. Reimann developed the entire program that they're using now.

A real advantage is the program's ability to calculate growth rates in a variety of ways. For instance, the sales income growth rate may be automatically calculated using a compound growth rate off a linear regression line. Or it may just as easily be computed with an inflation factor automatically included. The graphic outputs of the program are many and varied. It will produce negative bars — something most programs lack. Scatter plots like the one in figure 1 are used often by Boise Cascade as are horizontal bar graphs (figure 2). Vertical bar and line charts (figure 3) are also part of the program's repertoire.

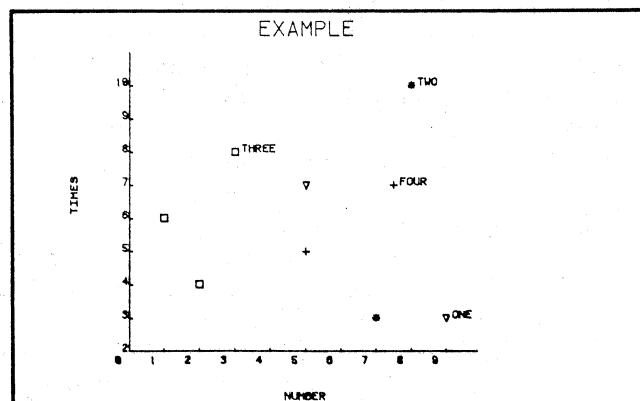


Fig. 1. A choice of many symbols is one of the features of Boise Cascade's 4051 scatter plot program.



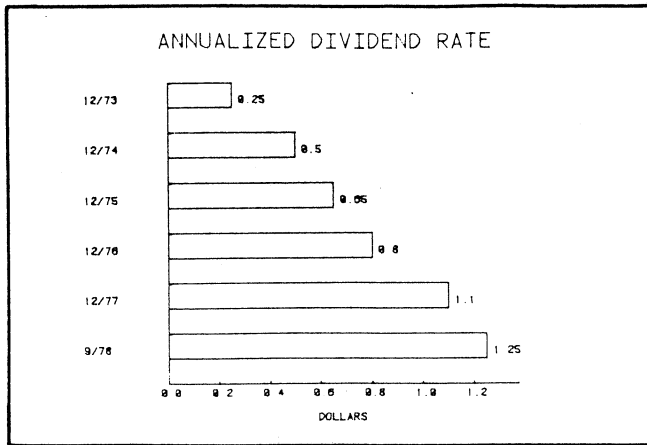



Fig. 2. Boise Cascade incorporates 4051-produced horizontal bar graphs into many of its reports.

### Results

Results are always the measurement of success. The graphics that the 4051 can provide are more flexible than anything that was previously available. The operator can alter size and position to make the graphs more readable

and presentable. Productivity is on the increase, and part of that is due to the 4051. It easily provides computation and output immediately when needed, and the output is meaningful because it's in the form that Boise Cascade believes in: graphics. 

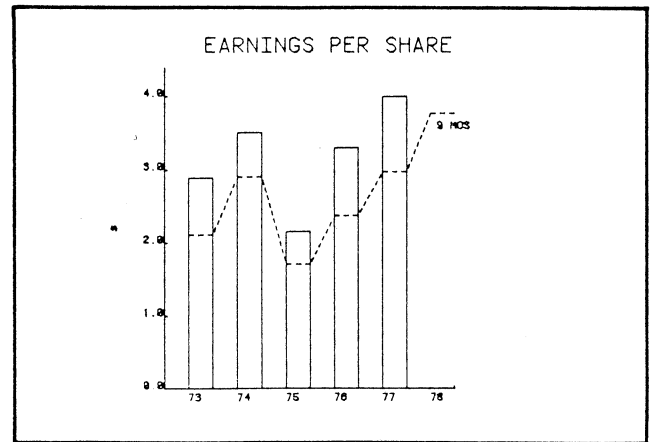


Fig. 3. Using the 4051 to calculate trends and graph them ensures accuracy.

## \*Editor's Note

### About Back Issues

Although we make it a practice to keep back issues of TEKniques available, in some cases they are not original stock. As our stock of initial printings runs out, we are turning to our local reproduction press to replenish our back issue inventory. This allows us to keep the content of back issues available to all.

### New Catalog Available

There is a new 4050 Series Applications Library Catalog available for the asking. If you haven't received your copy, drop us a note at the appropriate applications library address on the back of this issue. The new catalog contains 126 programs, with hard copy examples of most programs. The new catalog also contains a photo section of peripheral devices that can be used to extend system capabilities. Your own library isn't complete without one.

### The Questionnaire

In the last issue, we sent out a questionnaire on TEKniques and the Applications Library. If you haven't sent yours back, please do. We're anxious to hear from you, so we can fill your needs. We really are listening!

### Late Issues

As you've probably noticed, TEKniques grew like a garden during this summer. 28 pages put us a little behind schedule, and we're still running to catch up. Please bear

with us; we plan to be back on schedule with Vol. 3 No. 7 (November 1).

### A Contest

We have another contest in the offing. We're going to give you all winter to get your entries together, so you'll have something to do during those gloomy, gray, cold days. Watch for details in Vol. 3 No. 7.


### Program Tip Exchange

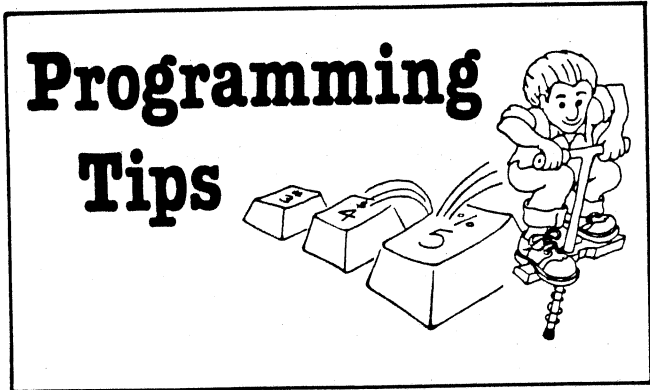
Send in your programming tip. Any one of the following 4051 Applications Library programs\* will be yours when it's published. Simply jot down a brief description of the function, the code, and your choice of program. Mail it to the 4050 Series Applications Library serving you: Library addresses are listed at the back of each TEKniques issue.

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

\*Documentation and listing only.

### Africa, Europe, Middle East Libraries Available Locally

Customers in Africa, Europe and the Middle East should address 4050 Series Library orders, and requests for issues of TEKniques, to their local sales office rather than the European Marketing Centre. 



## Following the UDK Path

by John Carter  
 Tektronix, Inc.  
 Santa Clara Annex  
 and Dan Taylor  
 Tektronix, Inc.  
 Wilsonville

The User-Definable Keys (UDK) are powerful tools for controlling program flow. However, newcomers to the 4050 Series may not understand all of the operational characteristics, so let's trace the path of the User-Definable Key operations.

### Pressing User-Definable Key = GOSUB

Pressing a User-Definable Key is the same as executing a GOSUB statement from wherever you are in a program. When a UDK is pressed, program control is transferred to the line number which is four times the UDK number. For example, pressing UDK 1 transfers program control to line number 4; pressing UDK 2 transfers program control to line number 8, and so on up to UDK 20 which transfers control to line number 80. (The line numbers associated with the UDK are fixed and cannot be changed.)

Note that each of the 10 User-Definable Keys represents two functions. That is, if the first UDK in the upper left corner is pressed, user-definable function 1 (beginning at line 4) is executed. If this same key is pressed in combination with the SHIFT key, user-definable function number 11 (beginning at line 44) is executed.

### Two Environments

The User-Definable Keys may operate in either a SET NOKEY or SET KEY environment. SET NOKEY is the default environment; power up, or the commands SET NOKEY or INIT put the system in this mode. SET KEY, the alternate environment, may be specified directly from the keyboard or under program control.

These two environments allow you to have a program which may be interrupted, but in which critical sections of

code can be protected from interrupts.

If a program is running while the 4050 system is in SET NOKEY mode, the BASIC interpreter will not respond when a UDK is pressed. However, in the SET KEY environment, when a UDK is pressed while a program is running, the BASIC interpreter completes the current instruction<sup>1</sup>, and does an implicit GOSUB from that point in the program to line 4, 8, ..., or 80. Now, let's take a closer look at the operating characteristics of each environment.

### SET KEY: The Interrupt Environment

Although a program is running, pressing a UDK causes the interpreter to complete the current instruction and do an implicit GOSUB from that point in the program to the statement associated with that UDK. The 4050 System stores the return address (the statement in the main program following the interrupt) in its memory. The subroutine execution continues until a RETURN statement is encountered, which returns execution to the main program, or until an END statement is reached, or, of course, a STOP<sup>2</sup>.

Pressing a UDK often leads to another subroutine. For example, pressing UDK 1 directs program execution to statement 4. Statement 4 might be a GOSUB to a larger subroutine. Once this larger subroutine is executed and a RETURN encountered, program execution is returned to the statement following statement 4. Therefore, another RETURN must be included in a statement following 4 in order to return to the main program. This follows normal programming rules, however, because pressing UDK 1 is an implicit GOSUB and not specifically coded, coding this second RETURN is often overlooked.

If in the previous example statement 4 had transferred control to another subroutine through a GOTO or IF...THEN statement, when a RETURN was encountered, execution would have been returned to the main program, again following normal programming rules.

Therefore, pressing a UDK is the same as executing a GOSUB; some memory is taken to store the return address, and a RETURN statement must be included for this UDK-GOSUB in order to return to the interrupt point. Figures 1 through 4 are examples that illustrate the paths we've been talking about.

<sup>1</sup>If the instruction is an INPUT statement (from the keyboard), the INPUT is considered to be completed. If any characters/digits had been keyed in before the UDK was pressed, they are used to satisfy as much of the input-list as possible.

<sup>2</sup>A STOP will not clear the return address from memory; either RETURN or END will release the memory reserved for the return address.

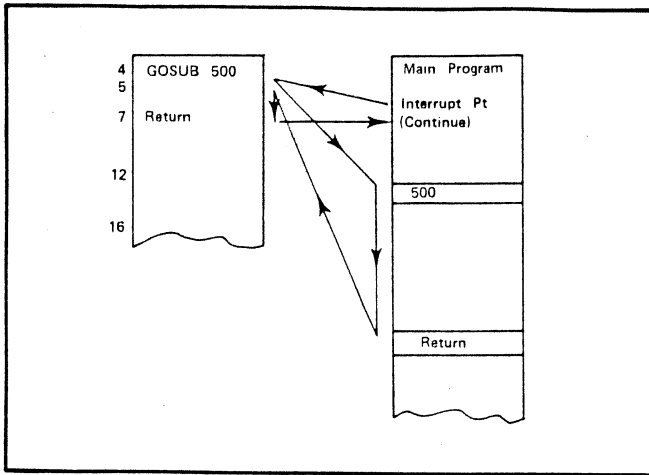


Fig. 1. In SET KEY environment, the program is interrupted by pressing UDK 1. This causes an implicit GOSUB to statement 4. Another GOSUB at statement 4 directs execution to statement 500. When the routine is completed a RETURN directs the interpreter back to statement 5, execution continues there until the second RETURN at statement 7 returns the interpreter to the main program.

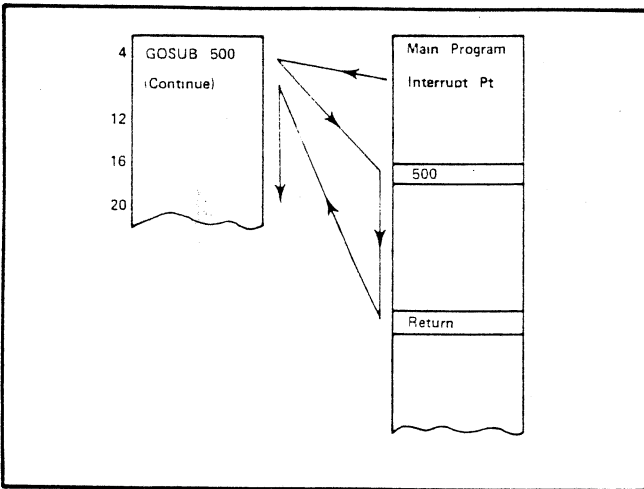


Fig. 2. In SET KEY environment, the program is interrupted by pressing UDK 1. This causes an implicit GOSUB to statement 4. Another GOSUB at statement 4 directs execution to statement 500. When this routine is completed, a RETURN directs the interpreter back to statement 5. However, no RETURN is encountered so program execution continues with statement 5 and those following.

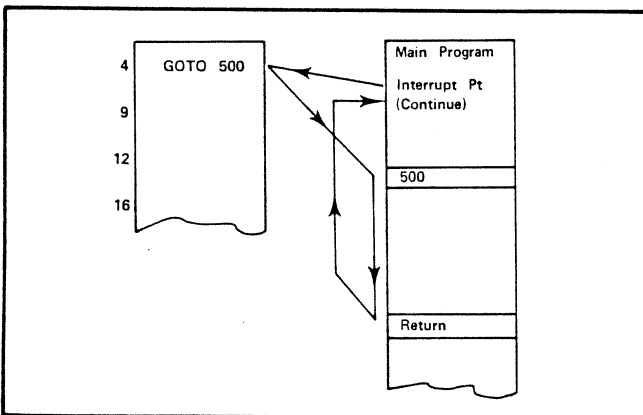


Fig. 3. In SET KEY environment, the program is interrupted by pressing UDK 1. This causes an implicit GOSUB to statement 4. A GOTO directs execution to statement 500. When this routine is completed a RETURN directs the interpreter back to the main program.

```

1 REM An experiment with SET KEY and SET NOKEY environments
2 REM using USER DEFINABLE KEY #1 (statement 4)
3 GO TO 100
4 GOSUB 500
5 RETURN
6 INIT
7 GO TO 120

100 REM ----- MAIN PROGRAM -----
101 REM The USER DEFINABLE KEYS work only after all routines
102 REM are completed unless SET KEY is performed. Therefore,
103 REM without statement 110 USER DEFINABLE KEY #1 could not
104 REM interrupt any operation.
110 SET KEY
120 FOR J=1 TO 20
130 REM The I loop is for a pause only! press UDK #1 during this loop
140 FOR I=1 TO 100
150 NEXT I
160 PRINT J
170 NEXT J
180 END

500 REM ----- THIS IS THE INTERRUPT ROUTINE. -----
510 PRINT "INTERUPT, I=";I
520 REM without the RETURN statements we could not resume the program.
530 RETURN
540 END

```

Fig. 4. If line 5 had not been a RETURN, the program would have continued execution at statement 5 rather than returning to the main program. Statement 6 would have initialized the program, putting it into SET NOKEY environment. Statement 7 would begin the loop a second time; but this time around, pressing UDK 1 would not interrupt the loop.

### SET NOKEY: The No-Interrupt Environment

The BASIC interpreter cannot respond to a UDK interrupt while the system is operating under program control in SET NOKEY mode. However, the system will set a flag indicating that a UDK was pressed, even though the interpreter couldn't respond. Each UDK has its own corresponding flag.

### Using The Two Environments

Should a SET KEY statement be included in a program (executing in a SET NOKEY mode), the moment that SET KEY command is executed, the interpreter will recognize the UDK flag(s). It will immediately branch (implicit GOSUB) to the statement associated with the first UDK pressed and execute that statement. If this statement is not a SET NOKEY, the interpreter will recognize the next UDK flag in the queue, branch to the statement associated with that flag's corresponding UDK, and execute that statement. If the interpreter does not encounter a SET NOKEY statement, it will continue to recognize the interrupt flags, and continue to execute these implicit GOSUBs until it reaches the last routine. Following normal programming rules, it will execute this last routine, return (provided a RETURN statement has been included at the end of the subroutine) to the second statement of the previous routine, execute that routine, and so on until all of the subroutines have been executed. This procedure is no different than if you had coded a set of nested subroutines, but you have the advantage of branching to the subroutines when you want to, by simply pressing the User-Definable Keys.

If a routine contains code which cannot be interrupted, the statement associated with the UDK (UDK number X 4 = statement number) should be a SET NOKEY command, followed with a SET KEY command at the end of

the critical code. The following example illustrates proper use of SET KEY and SET NOKEY.

```

1 GO TO 100
4 SET NOKEY
5 GOSUB 500
6 SET KEY
7 RETURN
8 SET NOKEY
9 GOSUB 600
10 SET KEY
11 RETURN
12 SET NOKEY
13 GOSUB 700
14 SET KEY
15 RETURN

100 INIT
110 PRINT "GBEGIN PROGRAM"
120 FOR I=1 TO 125
130 PRINT I;
140 NEXT I
150 SET KEY
160 PRINT "JJEND ROUTINE"
170 END

500 PRINT " SUB 500 - CRITICAL CODE - "
510 PRINT "2ND LINE OF 500"
520 PRINT "3RD LINE OF 500"
530 RETURN

600 PRINT " SUB 600 - CRITICAL CODE - "
610 PRINT "2ND LINE OF 600"
620 PRINT "3RD LINE OF 600"
630 RETURN

700 PRINT " SUB 700 - CRITICAL CODE - "
710 PRINT "2ND LINE OF 700"
720 PRINT "3RD LINE OF 700"
730 RETURN

RUN
BEGIN PROGRAM
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125
SUB 500 - CRITICAL CODE -
2ND LINE OF 500
3RD LINE OF 500

SUB 600 - CRITICAL CODE -
2ND LINE OF 600
3RD LINE OF 600

SUB 700 - CRITICAL CODE -
2ND LINE OF 700
3RD LINE OF 700

END ROUTINE

```

Fig. 5. The environment defaults to SET NOKEY with the INIT command at statement 100. As the loop at statements 120-140 executed, we pressed UDK 1, 2, and 3 respectively. The BASIC interpreter ignored these interrupts until it reached statement 150. When it executed the SET KEY command, it immediately recognized the UDK 1 flag and branched to line 4. Here it executed a SET NOKEY which prevented it from recognizing the other flags. Therefore, it continued to execute the routine associated with UDK 1. Returning to line 6, it encountered a SET KEY which enabled it to recognize the UDK 2 flag. It branched to statement 8, executed a SET NOKEY, and executed the routine associated with UDK 2. The SET KEY at statement 10 allowed it to go through the same procedure for UDK 3. Statement 15 returned the interpreter to statement 11, which sent it to statement 7, which directed it back to the main program.

```

1 GO TO 100
4 SET NOKEY
5 GOSUB 500
6 SET KEY
7 RETURN
8 GOSUB 600
9 RETURN
12 SET NOKEY
13 GOSUB 700
14 SET KEY
15 RETURN

100 INIT
110 PRINT "GBEGIN PROGRAM"
120 FOR I=1 TO 125
130 PRINT I;
140 NEXT I
150 SET KEY
160 PRINT "JJEND ROUTINE"
170 END

500 PRINT " SUB 500 - CRITICAL CODE - "
510 PRINT "2ND LINE OF 500"
520 PRINT "3RD LINE OF 500"
530 RETURN

600 PRINT " SUB 600 - NON CRITICAL CODE - "
610 PRINT "2ND LINE OF 600"
620 PRINT "3RD LINE OF 600"
630 RETURN

700 PRINT " SUB 700 - CRITICAL CODE - "
710 PRINT "2ND LINE OF 700"
720 PRINT "3RD LINE OF 700"
730 RETURN

BEGIN PROGRAM
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125
SUB 500 - CRITICAL CODE -
2ND LINE OF 500
3RD LINE OF 500

SUB 600 - NON CRITICAL CODE -
2ND LINE OF 600
3RD LINE OF 600

SUB 700 - CRITICAL CODE -
2ND LINE OF 700
3RD LINE OF 700

END ROUTINE

```

Fig. 6. The same steps were followed as in Figure 5 through the UDK 1 routine. However, upon branching to statement 8, a SET NOKEY was not encountered.

Consequently, the interpreter recognized the next flag and branched to statement 12, executed that routine, and returned to statement 15. Statement 15 directed it to statement 600 where it executed the UDK 2 routine and returned to statement 9. This statement returned it to statement 7 which directed it to the main program.

Regardless of the SET KEY/SET NOKEY environment, if the 4050 system is idle (no program is being executed), pressing a UDK will place the system under program control beginning at the statement number which is four times the UDK number. Because the program was initiated by an implicit GOSUB from the idle mode, when the BASIC interpreter encounters the RETURN statement associated with this implicit GOSUB, the system is returned to idle mode.

### SET KEY/SET NOKEY = Highly Flexible Programs

Thus, with a basic understanding of the User-Definable Keys and their two operating environments, highly flexible, user-oriented programs may be created.

# Recover Data After Magnetic Tape Read Errors

by Dale Grace and Bob Passeri  
 Combustion Power Company  
 Menlo Park, CA

We use the 4051 Graphic System as a controller and recorder in a data acquisition system (Figure 1). Over 50 parameters are sampled, converted to engineering units, and stored on magnetic tape for data reduction at a later time. In addition, we can use the 4051's graphic capabilities to display values of certain parameters over time.

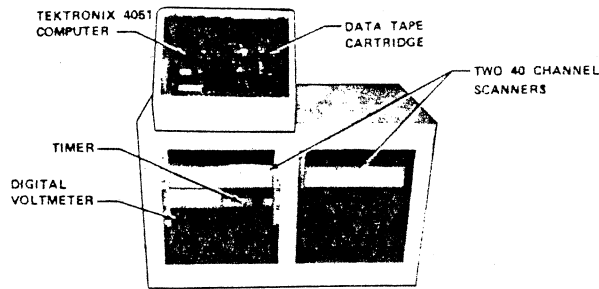


Figure 2-9 Data Acquisition System

Fig. 1. Data Acquisition system at Combustion Power Company, a division of Weyerhaeuser Company, at Menlo Park, CA.

Unknown environmental causes occasionally induce magnetic tape read errors (message 53 or 63). In order to increase the reliability of our data acquisition system, we have developed a simple means to recover nearly all of the information stored beyond the error. An external tape or disc device is required. We use the following procedure:

1. Read the tape up to the error and store the data on the external device.
2. Write over the initial data through the point of error with a placeholder number. Keep the format the same as the original data. Halt program execution with a STOP command.
3. Remove the data tape from the internal drive and re-insert. This procedure prevents a logical EOF from being written.
4. Read the data tape again and store the data on the external device, beginning with the actual data after the placeholder numbers. If an external tape drive is used, store the data in a different file than that used in step 1. If a random access disc file is used, store the data in sequential records.
5. Reconstruct the data tape by reading the data from the external device file(s) and record sequentially on the original tape.

The following software code demonstrates this general technique for data recovery. In this case 400 sets of binary data (each data set consisting of 62 parameters) were recovered after a magnetic tape error.

```

1 SET KEY
2 GO TO 100
4 GO TO 320
3 GO TO 440

99 REM ***** FILE "RECOVER" ***** 4 12/79
99 REM ***** READ IN INITIAL DATA *****
100 INIT
110 PRINT "IF MAG TAPE ERROR OCCURS, USE UDK #1"
120 ON EOF (0) THEN 230
130 DIM F(62)
140 PRINT #3:26:2
150 KILL "DATA"
160 CREATE "DATA", "U"1500,560
170 OPEN "DATA"1:"F",A#
180 FIND 1
190 REM ***** READ AND WRITE HEADER *****
200 READ #3:AS,BS,C,D
210 PRINT #4:BS,C,D
320 WRITE #1:1:AS,BS,C,D
370 REM ***** READ AND WRITE DATA *****
340 FOR I=2 TO 500
350 READ #3:F
260 PRINT I,F(1)
270 WRITE #1:I:F
280 NEXT I
290 PRINT "NO ERRORS FOUND"
300 END

310 REM ***** START FIX *****
320 FIND 1
330 ON EOF (0) THEN 560
340 F=10
350 REM ***** WRITE OVERLAY DATA *****
360 WRITE #3:AS,BS,C,D
370 FOR J=1 TO I
380 WRITE #3:F
390 NEXT J
400 J=J-1
410 PRINT "REMOVE AND REPLACE TAPE THEN PRESS UDK#2"
420 STOP

430 REM ***** CONTINUE PENDING TAPE *****
440 FIND 1
450 READ #3:AS,BS,C,D
460 PRINT #4:BS,C,D
470 FOR I=1 TO J
480 READ #3:F
490 PRINT I,F(1)
500 NEXT I
510 FOR I=J TO 500
520 READ #3:F
530 WRITE #1:I:F
540 PRINT I,F(1)
550 NEXT I
560 PRINT "FINISH TAPE"
570 FIND 1
580 ON EOF (1) THEN 600
590 REM ***** TRANSFER (DATA--DISC TO TAPE *****
600 READ #1:1:AS,BS,C,D
610 WRITE #3:AS,BS,C,D
620 PRINT #4:BS,C,D
630 FOR I=2 TO 500
640 READ #1:BS:F
650 WRITE #3:F
660 PRINT I,F(1)
670 NEXT I
680 PRINT "FINISHED"
690 END

```

## Polygon Design and Placement

by Bernard M. Gunn  
 W. & K. McLean Ltd.  
 Glenn Innes, Auckland, New Zealand

To draw a regular polygon anywhere on the 4050 Series Graphic System screen, or on a plotter, it is a bit hit or miss to enter the length of a side and the angle of rotation; we don't know what the size of the resulting figure will be.

The following routine allows the user to specify the radius, the number of sides, and the center of the figure for complete control of its size and placement. The center (X,Y in statements 130 and 140) could just as well be input from a digitizer, the joystick or the keyboard.

In this routine statements 24—27 allow the radius and number of sides to be input from the keyboard without disfiguring the main display (Control ↑ returns the cursor to the top left of the screen).

```

1 SET DEGREE$
2 GO TO 100

24 REM INP RADIUS & NUMBER OF SIDES OF POLYGON
25 PRINT "R= "
26 INPUT R,N
27 GO TO 2010

```

```

100 WINDOW 0,130,0,100
110 VIEWPORT 0,130,0,100
120 T=32
130 X=65
140 Y=50
150 RETURN

```

```

2000 REM POLYGON SUB
2010 ROTATE 0
2020 MOVE PT:X,Y
2030 L=RYTHN(180/N)
2040 PMOVE PT:-L,-R
2050 L=L+L2
2060 C=360/N
2070 FOR J=1 TO N
2080 PDPHM L,0
2090 ROTATE ZIJ
2100 NEXT J
2110 RETURN

```

You can store the coordinates of the polygon for replotting by adding three statements to your routine:

```

2001 DELETE X0,Y0
2005 DIM X0(N),Y0(N)
2005 GIN X0(J),Y0(J)

```

A two line routine will redraw the polygon:

```

MOVE X0(1),Y0(1)
DRAW X0,Y0

```

## Tracking Input From the 4956 Graphic Tablet

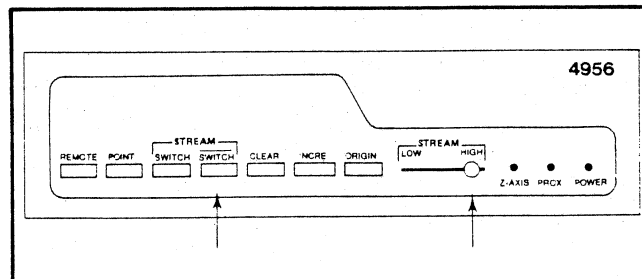
by Jan Erkelens

Grontmij Nederland B.V.  
Zeist, The Netherlands

When digitizing from the 4956 Graphic Tablet, it's often advantageous to have your position on the tablet

displayed on the 4050 graphic screen. The following routine achieves this.

First, set the tablet controller to **STREAM** mode and set the frequency to **HIGH** (Figure 1).



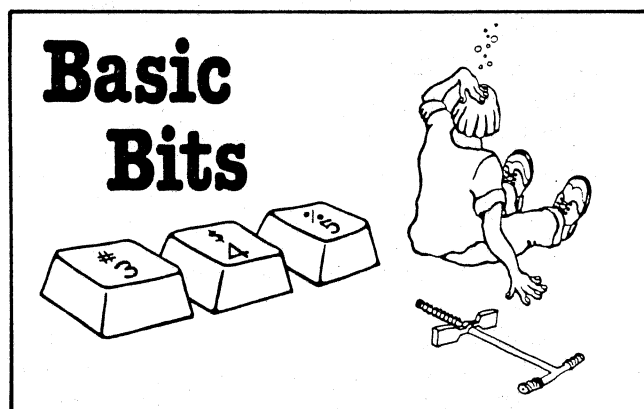
The position of the cursor or pen\* on the tablet will be displayed on your 4050 System as a blinking █; as you move the cursor or pen over the tablet surface, the █ tracks its position on the graphic screen. When a button on the cursor (or pen) is pushed, the 4050 System responds with a peep. A second peep follows if the digitized point is outside the window boundaries (X1,Y1,X2,Y2).

```

1000 REM SUBROUTINE FOR ASKING INPUT FROM TABLET
1010 PRINT @32,18:5
1020 INPUT @81,X,Y,Z$
1030 IF Z$="0" THEN 1070
1040 MOVE X,Y
1050 PRINT @32,24:"*"
1060 GO TO 1020
1070 PRINT @32,18:0
1080 PRINT "IG"
1090 IF X<X2 AND X>X1 AND Y<Y2 AND Y>Y1 THEN 1110
1100 PRINT "IG"
1110 RETURN

```

\*We have obtained the best results using the cursor.



## Interrupting a Program

by Pat Kelley

TEKniques Staff

To interrupt a program, press the **BREAK** key **once**. The program is halted but may be resumed by executing a

**RUN** command beginning at the line number printed in the interrupt message on the screen. For example, a program was halted and the following message was printed on the graphic screen:

PROGRAM INTERRUPTED PRIOR TO LINE 170

To resume this program, type **RUN 170**.

You may also use the **STEP PROGRAM** key to resume your program. It will begin execution at the line number printed in the message on the screen and execute one line at a time with each depression of the key. This is a useful debugging tool—key in **SET TRACE** directly from the keyboard while the program is halted and watch the flow of your program by line number.

If the program is waiting for input from the keyboard, you must satisfy the input before it will halt execution. If the tape is running, the system won't halt the program until the tape drive is finished; however, it won't execute

the next instruction before halting. Therefore, don't panic in these cases and abort a program unnecessarily by pressing the BREAK key twice. For pressing BREAK twice is the same as executing an END command; the line counter is reset, the execution stack is cleared, and you may not continue your program. You'll have to start at the beginning.

## Install ROM Packs Before Power Up

by Pat Kelley

TEKniques Staff

Install ROM Packs into your 4050 System before power up. Plugging a ROM Pack into or removing it from your machine with the power on can cause permanent damage to the ROM Pack or to the Central Processing Unit. Your ROM Pack is an extension of your 4050 System's circuitry. When the circuits are energized (power turned on), fiddling with this circuitry by any means—whether through ROM Packs or by taking a screwdriver to the inside—can cause damage. (The latter could cause damage to yourself, also!)

A second reason for installing ROM Packs before power up is so the 4050 System will know they are there. Upon power up the system checks which ROM Packs are in place: it won't repeat this survey until a power down and another power up. Consequently, should you manage to install a ROM Pack after power up without damaging the machine, the system won't be aware of it. Therefore, any call to the extended functions of that ROM Pack will result in an error: and program execution will halt.

## One Reason for a Tape File Directory(ies)

by Ray Holland

University of Manitoba  
Department of Medicine  
Winnipeg, Manitoba

What do you do when working on a program and the additions cause the ASCII program's length to exceed the file size on the tape reserved for that program? The usual procedure would be:

```
TLIST
FIND (Last)
MARK 1,SPACE
```

But very often there is an empty file, or an old unused program file, that you could put your now enlarged program into. If only you could find it or remember what file number it was in! TLIST will not help unless you have modified the tape file headers.

An alternate solution is to maintain a tape directory that contains the information available from TLIST as well as

a description of the contents of each file. A logical place for such a program is File #1. Now to find a place to put your program, execute the following commands from the keyboard:


```
1000 REM
FIND 1
APPEND 10000
RUN 10000
```

The directory will be printed. Once you located the file you want, let's suppose it is File #7, then:

```
DELETE 10000,20000
FIND 7
SAVE
```

This procedure is much faster than using TLIST, except on very short tapes. On very long tapes with a lot of small files, a number of copies of the directory could be included on the tape at easy-to-remember locations: e.g., FILE 1, 20, 40, etc., to make the process even faster.

```
1 INIT
2 IMAGE -8D,19T,2D,* - *,49A
3 GO TO 210
100 DATA 9
110 DATA 3072,'A PLOTTING DEMO'
120 DATA 10240,'TEK FFT'
130 DATA 20224,'DATA FILE FOR TEK FFT'
140 DATA 3072,'PROG FOR SYNTHESIZING FOR FFT'
150 DATA 4096,'DATA ACQUISITION PROG FOR FFT'
160 DATA 6144,'EMPTY FILE'
170 DATA 3072,'LEAST SQUARES LINEAR FIT'
180 REM
190 REM THIS SPACE RESERVED FOR MORE FILE LABELS
200 REM
210 PRINT 'LFILE SIZEITAPE FILE DIRECTORY'
220 PRINT '-----I-----JJ'
230 RESTORE
240 READ N
250 FOR I=2 TO N-1
260 READ A,D$
270 PRINT USING 2:A,I,D$
280 NEXT I
290 PRINT 'I ;N;' - LAST'
```

To make additions it is important to update line 100 as well. The number stored there is defined as N in line 240, where N is the number of files on the tape including the LAST file. The physical size of the file can be obtained from TLIST or can be calculated when MARKed. 

```
MARK 1,4000
```

```
MARK 1,(INT(-4000/256))*256
```

File Size = INT (-X/256) \* 256

Ed. Note: Previous issues of TEKniques have carried other methods for coding a tape directory. Ray Holland's discussion of *why* a tape directory lends additional weight to a good programming procedure. However, if named tape headers are desired in addition to a directory, the PLOT 50 4050A08 Utilities program contains a program to do this. Also, TEKniques Vol. 3 No 3 carried a tip from Ed Sawicki on extending tape file headers.

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

## Domestic U.S. Prices:

Documentation and listings	\$20 per program
Recording Fee	5 per program
Tape Cartridge	30 per tape

## 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. 2618.

## Forms

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

## Outside U.S.

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

## ABSTRACT NUMBER: 51/07-0717/0

Title: **4907 MIPS — A Management Information Processing System**

Author: Jim Dillon  
 Tektronix, Inc.  
 Santa Clara Field Office  
 Memory Requirement: 32K  
 Peripherals: 4907 File Manager  
 4631 Hard Copy Unit

Statements: 938

Files: 2 ASCII

Requires data disc

Available on disc only.

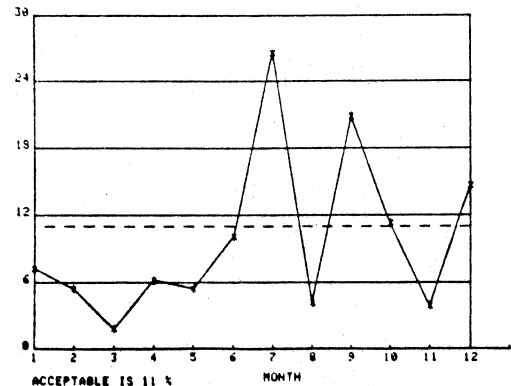
The program has the following capabilities and characteristics:

1. General purpose comparison reports (tabular and graphic).
2. Multicurve and ratio/percentage graphs.
3. Organization and straightforward editing of data based on either a 12 or 13 period fiscal year.
4. Ease of use through menus and prompting questions or through the use of function keys.
5. Progressive sophistication ranging from straightforward editing and graphing to utility level functions and subroutines. Newcomers to the 4051 should be able to manipulate and graph data as naturally as a sophisticated user should be able to modify and add code to perform special functions.

6. All necessary data (graph titles, curve titles, fiscal year indicators, and the 12 or 13 accounting periods of information) are held on the data disc. The program is general purpose in nature and can be used on any type of data (\$s, man-years, %, expenses order, etc.).
7. Format data disc and convert tape data files to disc.

SUBFILE #	1	2	3	4	5	6	7
TITLE	ORDERS	TARGET	CUST#1	CUST#2	PEOPLE	EXPENS	BUDGET
1	351234	420000	15000	30000	5	25400	31250
2	451025	420000	10900	2500	5	24600	31250
3	1205412	420000	14520	155000	5	22515	31500
4	456235	420000	17500	24500	5	28415	32540
5	265235	420000	27545	0	6	30100	32540
6	290505	420000	13450	36200	6	30100	32540
7	95865	450000	550	10995	6	25425	32540
8	546213	450000	4595	5000	6	23050	32540
9	156235	450000	15630	0	6	32545	32540
10	254654	450000	25995	5595	6	20790	32540
11	852346	450000	17569	30400	5	33600	35500
12	265045	450000	15655	3995	5	30940	35500

TITLE FOR FILE # 1 IS :  
 MYSTERY DISTRICT -- INFORMATION DISPLAY GROUP SALES ORGANIZATION  
 PERCENTAGE GRAPH -- EXPENSELY 600 AS A % OF ORDERSLY 600  
 MYSTERY DISTRICT - IDG SALES ORGANIZATION





Title: MC6800 Disassembler Program

Authors: Ed Sawicki  
 Joe Boim  
 Tektronix, Inc.  
 Long Island Field Office  
 Memory Requirement: 8K  
 Peripherals: Optional — 4641 Printer  
 4051R06 Editor ROM Pack

Statements: 157  
 Files: 1 ASCII

The program produces a source code-like listing from HEX ASCII object code. The object code must reside on a 4051 tape file. The source code listing can be directed to the 4051 screen or internal tape unit, a 4641 Printer or a GPIB device.

The 4051R06 Editor ROM Pack may be used to insert comments into the listing, change absolute addresses and operands to labels and symbols and to insert equate statements.

The object code must be in HEX ASCII. Only the hexadecimal characters 0 through 9 and A through F are allowed. The object file must contain only the HEX ASCII representation of machine-executable code.

The source listing is divided into five fields:

1. The byte count in decimal.
2. The value of the program counter in HEX.
3. The op-code and operand in HEX/ASCII.
4. The mnemonic.
5. The operand.

```

MC6800 DISASSEMBLER V1.1

TEST PROGRAM

1      1000  8602      LDAA #02  |
3      1002  B78798    STAA  8798 |
6      1005  7F8798    CLR   8798 |
9      1008  B687AA    LDAA  87AA |
12     100B  8508      BITA  #08  |
14     100D  2705      BEQ   1014 |
16     100F  8639      LDAA  #39  |
18     1011  974B      STAA  4B,D |
20     1013  39        RTS         |
21     1014  BDE00C    JSR   E00C |
24     1017  8623      LDAA  #23  |
26     1019  BDFFD1    JSR   FFD1 |
29     101C  BDFFB5    JSR   FFB5 |
32     101F  B68798    LDAA  8798 |
35     1022  2807      BMI   102B |
37     1024  B687AA    LDAA  87AA |
40     1027  8508      BITA  #08  |
42     1029  27F4      BEQ   101F |
44     102B  BDFBD7    JSR   FBD7 |
47     102E  BDFFE3    JSR   FFE3 |
50     1031  BDFC24    JSR   FC24 |
53     1034  7EC8BF    JMP   C8BF |

FINISHED !
    
```

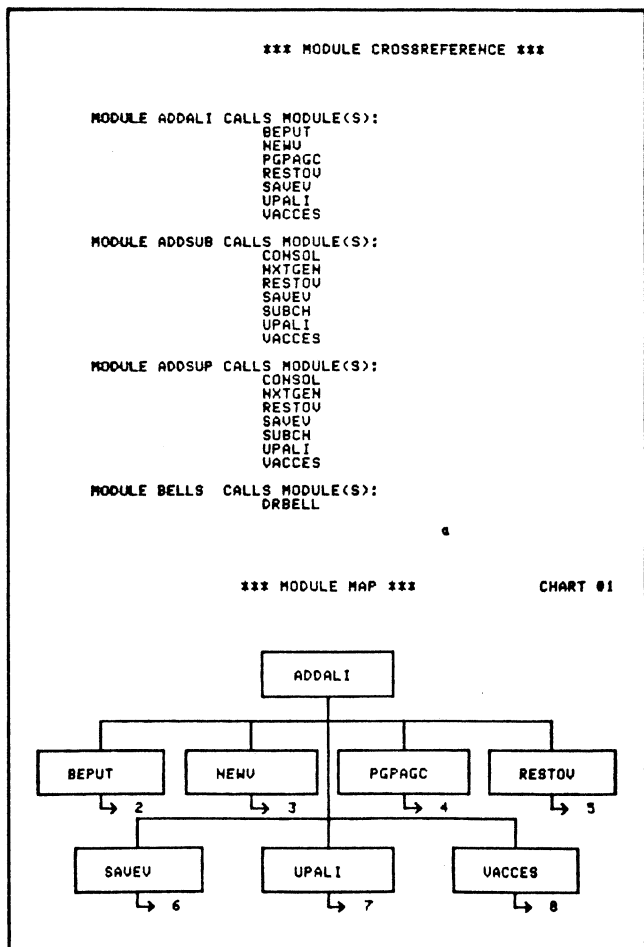
Title: Program Module Cross Reference & Map

Author: Captain S. K. Sanford  
 Aberdeen Proving Ground, MD  
 Memory Requirement: 16K  
 Peripherals: Optional — 4631 Hard Copy Unit  
 4924 Digital Cartridge Tape Drive  
 4051R06 Editor ROM Pack

Statements: 286  
 Files: 2 ASCII

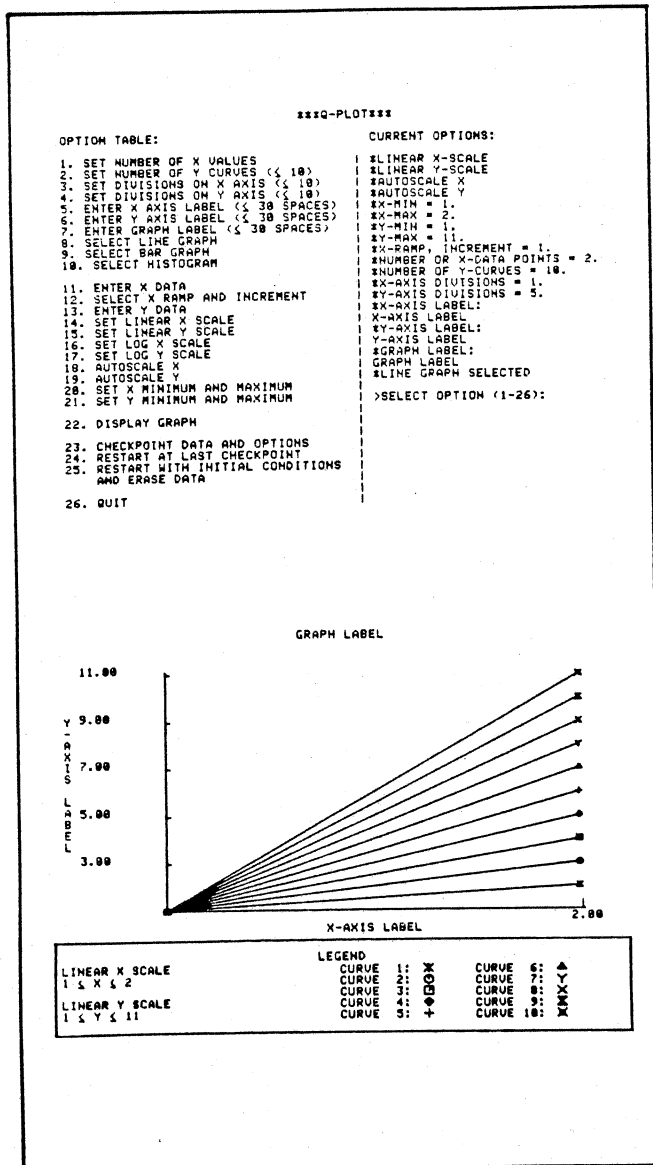
The cross reference program creates two files of calling and called subprogram names using the 4051R06 Editor ROM Pack or a simple BASIC program. The first file is sorted in calling program sequence (alphabetically), while the second, which is identical, is sorted in called program sequence.

The module map program reads the data files from the 4051 or the 4924 Digital Cartridge Tape Drive one at a time, and produces a listing of the calling programs with their called programs, then a listing of the called programs with their calling programs. The pages of output are numbered alphabetically from "a" to "zz", and may be automatically copied by the 4631 Hard Copy Unit.



Title: **Q-Plot**  
 Author: Captain S. K. Sanford  
 Aberdeen Proving Ground, MD  
 Memory Requirement: 32K  
 Peripherals: Optional — 4631 Hard Copy Unit  
 4924 Digital Cartridge Tape Drive  
 Statements: 871  
 Files: 1 ASCII

The program is a quick plotting utility which prompts the user for data for up to ten curves (to be plotted on the same set of axes), and displays these curves on the 4051 screen using either linear or log scales (or a combination of both), with labeled axes, graph, and line markers. A legend is printed at the bottom of the graph, including the data ranges. A checkpoint restart feature may be optionally employed which will save the data and graphing options selected automatically to allow the graph to be redisplayed at a later time.



Title: **4924 Mass Tape Duplication**  
 Author: Ed Mitchell  
 Tektronix, Inc.  
 Memory Requirement: 16K  
 Peripherals: 2 — 4924 Digital Cartridge Tape Drives  
 Optional — Up to 15 — 4924's  
 Statements: 205  
 Files: 1 ASCII

Use the 4924 Mass Tape Duplication program to duplicate up to 14 copies of a tape at a time.

The program uses one 4924 for the Master copy, and from one to fourteen 4924's for slaves.

Read after write mode is used on the slave 4924's. Slaves detecting errors in duplication "drop out" so as not to stop total duplication. A status report is printed at the end of duplication informing the operator which tapes are good and which are bad. Files are marked the same length as those on the master and the header records are copied, to give an exact copy of the master tape. WBYTE on the 4051 is used to operate all drives simultaneously. ASCII SECRET programs cannot be duplicated.

Title: **Slidemaker II**  
 Author: John R. Carter  
 Tektronix, Inc.  
 Santa Clara Annex  
 Memory Requirement: 32K  
 Peripherals: 4662 Plotter  
 Optional — 4907 File Manager  
 Statements: 918  
 Files: 2 ASCII Program  
 1 ASCII Text

This is a revised version of 51/07-9531/0; the purpose of the revision was to facilitate understanding and use of the program. Two new functions were added and several functions were renamed to give a more logical flow and greater flexibility to the use of the program. A program to convert plots saved by the original program is also provided.

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

The main features are:

1. Standard type sizes selected with a single variable.
2. Tab selections which operate like a typewriter.
3. Variable type sizes and color changes possible on the same line of type including choice of bold or normal type on the same line.





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

STELLA MCMORRAN  
60-307

IF YOU HAVE MOVED  
PLEASE CALL 3618

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)**

Ms. Bev Brandon, 73-312  
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