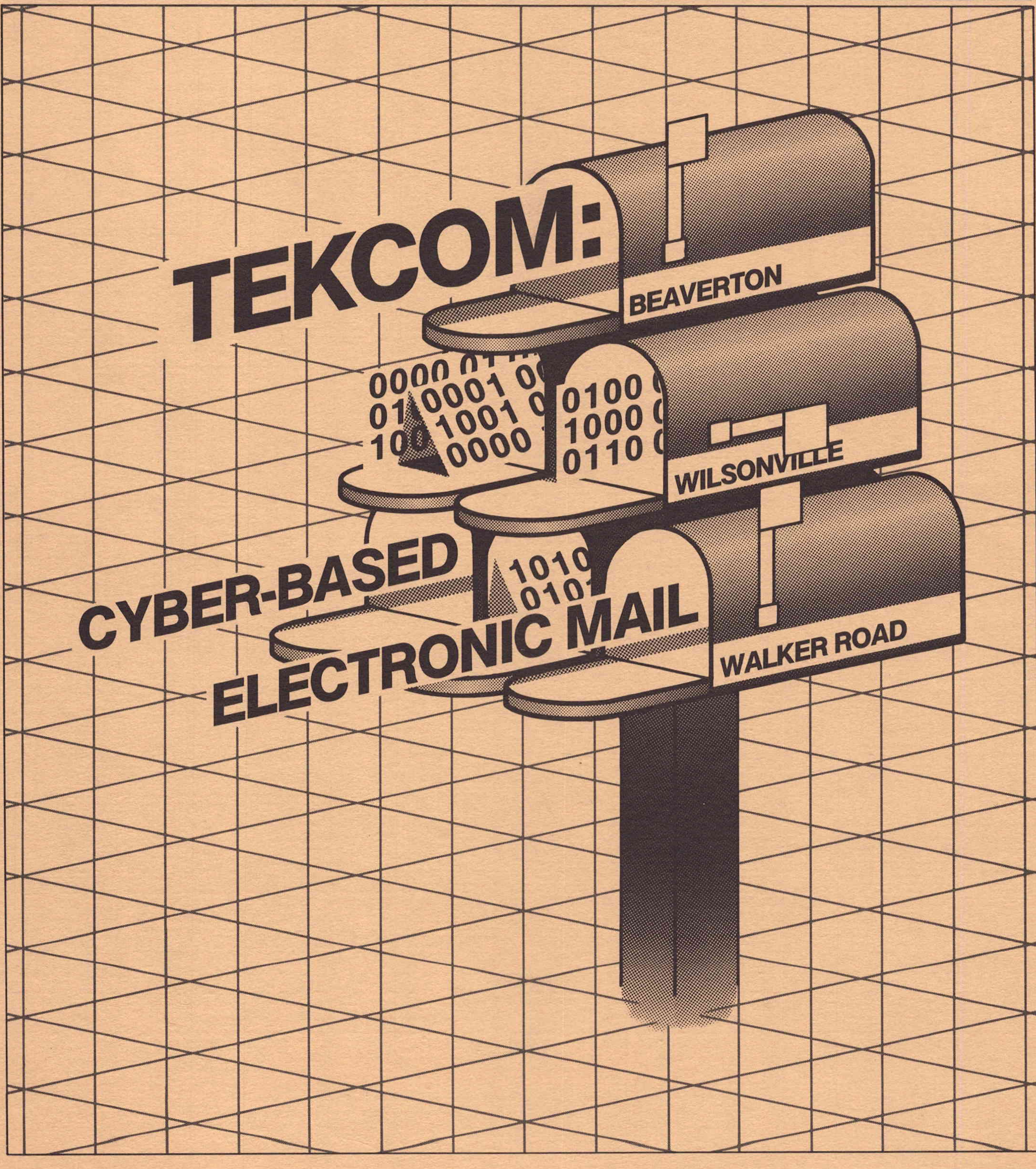
TEKTRONIX,INC. COMPANY CONFIDENTIAL NOVEMBER/DECEMBER,1979

TECHNOLOGY report

HARDWARE SOFTWARE FIRMWARE PROCESS ENGINEERING MATERIALS RESEARCH



CONTENTS

FORUMS: ENGINEERS TALK TO MANAGERS

In November 1976, Bill Walker (executive vice president) formed the Engineering Activities Council (EAC) to provide engineers with a forum in which to present directly, to multiple levels of management, what engineers themselves consider important in technology.

In the last three years, the EAC has sponsored 14 forums presenting engineers' views of the problems and progress of new technology at Tektronix. The forum presentations are described in **Forum Reports.**

For a copy of a **Forum Report** listed in the table below, call ext. 8920 (Merlo Road) or write or drop by d.s. 53-077. □

	FORUMS	CHAIRPERSONS
1.	General Purpose Interface Bus	Robert Chew, Paul Williams
2.	A-D and D-A Converters	Bob Nordstrom, Mike Boer
3.	Video Display Techniques	Steve Joy, Phil Crosby
4.	New Technologies: I	John Addis, Bob Burns
5.	New Crt Technologies	Bob Oswald, Cal Diller
6.	Creative Microprocessor Hobby Projects	Dave Chapman, Joyce Lekas
7.	Managers Talk to Engineers	Mike Boer, John Mutton
8.	Microprocessor Design Pitfalls	Robert Chew, Paul Williams
9.	New Technologies: II	Hock Leow, Binoy Rosario
10.	Packaging	Bob Burns, Cal Diller
11.	Creative Microprocessor Hobby Fair Projects: II	Steve Joy, Hock Leow
12.	Reliability	Tim Flegal, Mike McMahon
13.	Managing Firmware Throughout Its Life Cycle	Lynn Saunders, Jim Tallman, Dave Armstrong

Volume 1, No. 2 November/December 1979. Managing editor: Burgess Laughlin, ext. 8929 (Merlo Road), d.s. 53-077. Associate editor: Laura Lane. Cover: Michael Satterwhite. Graphic design: Joe Yoder. Graphic assistant: Jackie Miner. Typesetter: Jean Bunker. Published by Technical Publications (part of Technical Communications Services) for the benefit of the Tektronix engineering and scientific community. Copyright © 1979, Tektronix, Inc. All rights reserved.

Eleanor McElwee wrote the TEKCOM article on page 3.

Why TR?

Technology Report serves two purposes. Long-range, it promotes the flow of technical information among the diverse segments of the Tektronix engineering and scientific community. Short-range, it publicizes current events (new services available and notice of achievements by members of the technical community).

REVIEWERS NEEDED

Technology Report articles are becoming more technical and more detailed and that trend will continue. Although most articles are written and edited for a general engineering audience, only technical specialists can effectively review the content of very complex articles.

Technology Report publishes articles written by engineers and scientists in all the disciplines found at Tektronix. Examples include electrical engineering, mechanical engineering, chemistry, chemical engineering, materials research, human factors, and aspects of marketing and manufacturing of direct interest to the Tektronix engineering and scientific community.

If you are interested in reviewing a rough draft article in your specialty, call ext. 8929 (Merlo Road) or write to d.s. 53-077. □

TEKCOM: A CYBER-BASED ELECTRONIC MAIL SYSTEM

ELECTRONIC MAIL

Electronic mail provides a fast, reliable, paperless medium for sending messages to an individual or each member of a group. It allows the user to accept incoming messages at the user's convenience, to electronically dispose of them, and to file them or pass them along.

Electronic mail is rapidly gaining acceptance in corporations and institutions throughout the world. Many of the top 50 Fortune-500 companies now use commercially-available or in-house electronic mail systems.

TEKCOM (Tektronix Communications), an electronic mail system operating on the CYBER system, was developed by Stephen Heitmann and Paul Blattner (Tektronix Laboratories Computer Research Group) to provide a state-of-the-art communication tool for general use at Tektronix. Assistance from the Scientific Computer Center was provided by Bob Mainero, Mike Bonnain, Andy Davidson, Jeff Gilbert, Kurt Krueger, and Susan McCullough. Response from an early test group of 19 users was favorable and enthusiastic.

TEKCOM is an easy-to-use electronic mail system available on the Scientific Computer Center's CYBER. It requires no specialized user knowledge or programming. It accepts messages typed at a computer terminal, revises them as instructed, stores them, forwards them, or files them. TEKCOM effectively removes time and distance barriers, conserves resources by reducing paper usage, reduces secretarial work load, and facilitates better utilization of personal time.

WHAT TEKCOM CAN DO

Like other electronic mail systems, TEKCOM allows every user to both transmit and receive messages. When transmitting, you can send messages (as short or as long as you like) to one or more individuals. At any time while entering your message at the terminal, you can see the entire message and add, delete, change, or relocate material.

After entering and revising the message, you can instruct the computer to send it — or you can change your mind and not send it. You can send the message and also store it in a new or existing computer file. If you wish, you can get a hard copy of it, either from your own hard-copy unit or from the Scientific Computer Center printer.

When receiving, you can use your computer terminal as a private mailbox. TEKCOM messages are secured by the CYBER file-protection system and with TEKCOM passwords. Messages are inaccessible to anyone but the owner of the mailbox.

Most people who first use computer mail facilities are very enthusiastic about their initial experience. Reading mail becomes fun, not a chore.

If more than one message is addressed to you, TEKCOM can summarize the messages by listing the subject and writer of each. You then assign priorities to the messages and read them in the order preferred. After reading a message, you can decide what to do about it. You can delete it. You can leave it in the mailbox to be read again later. You can forward it to someone else. You can file it or copy it. Or you can reply to it right away.

ADVANTAGES FOR YOU

HOW TEKCOM WORKS (on page 4) indicates some of the things TEKCOM can do for you. TEKCOM's major advantages are that it does these things at your convenience and with very little effort on your part.

Everyone knows the frustrations of conducting business by either written mail or telephone. The conventional mail system introduces a time delay which is often unacceptable. The telephone can also introduce a time delay when the person called is not there. When you're on the receiving end, however, the telephone permits no delay; it insists on interrupting your work and your thoughts, usually at the most inconvenient and inappropriate times.

TEKCOM permits you to send messages easily and quickly to people throughout the CYBER network, and to respond easily and quickly to messages you receive. You receive and reply to messages at your own convenience, selecting periods in the work day which do not conflict with your other priorities. Your communications are instantly accessible to your addressees, or can be read by them at their convenience.

Perhaps the greatest advantage of TEKCOM is its ease of use. You need no special knowledge or training. TEKCOM leads you step-by-step through the message preparation (as described in HOW TEKCOM WORKS), tells you if you're doing something wrong, and even helps you out when you're not sure what to do.

TEKCOM generates two basic error messages. The first, UNDEFINED COMMAND - REENTER, indicates that you have used a command improperly, misspelled a command, or used a non-existent command. The other error message is 'partial command' NOT UNIQUE - REENTER. If you're not sure what you've done wrong, type HELP and TEKCOM will print out a list of acceptable commands. HELP is always available whenever you're not sure what to do in response to a prompt.

Unlike previous message systems, TEKCOM allows you to send messages to people by name rather than only by account number. Although TEKCOM stores names by last name/first name, you enter them by typing the first name followed by a space and the last name. The space tells TEKCOM where to split the name for inversion and is especially important when you use an abbreviated first name.

Continued on page 4

Continued from page 3

To make this sending-by-name capability flexible enough to accommodate many users, TEKCOM uses certain "wild-card" characters. This feature permits you to enter abbreviated names or spellings you're unsure of. To abbreviate a name, type the first part of the name followed by a period. The period is a wild-card character representing the longest string that allows a match. Thus, HOW. V. represents HOWARD VOLLUM. Abbreviations for one-word names don't need a period; TEK always matches TEKTRONIX.

If you're not sure how to spell a name, use the wild-card characters ? and * to represent, respectively, a single character or a string of characters. For example, FRED OLS?N will reach the right person whether his name is Olsen or Olson. BLA*NER will match either Blattner or Blatner.

If you try to send a message to someone who has not yet used TEKCOM, the system will respond with JOHN DOE NOT FOUND or JOHN DOE DOES NOT HAVE A TEKCOM MESSAGE FILE. You will then have to communicate with him by conventional means (or show him this article and hope he gets a TEKCOM mailbox).

ADVANTAGES FOR TEK

TEKCOM has many advantages for Tektronix besides those implicit in better engineer and manager time-utilization. As an alternative to interoffice memos, TEKCOM reduces secretarial work load and conserves resources by reducing paper usage. TEKCOM's convenience encourages more frequent and less formal interchange between engineers. TEKCOM can also facilitate better communication between managers and engineers by permitting a sender to contact many people simultaneously regardless of their schedules.

TEKCOM can lead to many benefits in product development and definition. Senders can expose new ideas to many people for evaluation without disrupting schedules. Senders can solicit and evaluate a wider range of opinions. A larger group of managers can evaluate diverse creative ideas. Dispersed task groups can be more easily managed with this tool, and managers may have a greater "span of control." And users can automatically document all communications.

TEKCOM could even eliminate the need for some face-to-face meetings, or facilitate better organized meetings by making preparatory communications easier. Computer conferencing (such as the system described in **Engineering News**, August-September 1979) is a logical extension of electronic mail, and TEKCOM could provide such a system to Tektronix. Other uses of TEKCOM will undoubtedly arise in areas not presently envisaged.

FOR MORE INFORMATION

TEKCOM development was a Tektronix Laboratories Computer Research Group project. For more information, call Paul Blattner on ext. 6056 (Beaverton). (Stephen Heitmann initiated research in electronic mail and computer conferencing, designed TEKCOM, and worked on the program development. He has left Tektronix to start his own business.)

HOW TEKCOM WORKS

TEKCOM is available on the HOST-A CYBER system only. If you don't have a CYBER account number, application forms are available from the Scientific Computer Center on the fourth floor in building 50 (50-454, ext. 6870, Beaverton). A detailed writeup on TEKCOM is also available from the Scientific Computer Center.

You then need a computer terminal with a telephone dial-up. After accessing CYBER, selecting Host A, and entering your account number and charge number, type TEKCOM.

The first time you use TEKCOM, the system prints out a one-time-only general-information message. You can then send a message to someone else (or to yourself), and other users will be able to send messages to you. To compose and send a message, type SEND when the initial prompt #:? appears.

To Compose and Send a Message . .

The TEKCOM message format is similar to that of an IOC (Interoffice Communication) and includes the same standard heading (To, From, Subject, Date). When you type the command SEND, TEKCOM prints the prompt TO:?. Respond with the first and last name of the addressee, followed by a comma if there are several names to be typed.

To send the message to yourself, type ME. If the names will not fit on one line, type a comma as the last character before the CR (carriage return). TEKCOM will then respond with another TO:? prompt.

When you have entered all the names, TEKCOM prints the next prompt, which is SUBJECT:?. Respond with the subject of your message. Your response must not exceed one line.

Because TEKCOM already knows who you are (from your account number) and what the date is, TEKCOM automatically enters the FROM and DATE information.

The next (and last) prompt for the SEND command is TEXT:?, a request for you to enter your message. Enter each line of text after the line-number prompts (that is, 1 ?, 2 ?, 3 ?, ...).

After entering your message (and editing it, if required), you send it by typing .DONE while in text mode. TEKCOM displays the verification MESSAGE DELIVERED TO (NAME) to indicate that the message has been sent to each individual on the distribution list. (All commands entered while in text mode must begin with a period, and the command must immediately follow the period. Otherwise, the command will be entered as part of the text and not processed.)

If you decide not to send the message (either when you're finished or at any time during the preparation), simply type .QUIT at the beginning of a line. TEKCOM then returns to the command input mode, and the initial prompt #:? appears. (QUIT is used to terminate all functions in TEKCOM.)

To Make Changes . . .

At any time while entering your text, you may see the complete message (header and text) by typing .REVIEW at the beginning of a line. While reviewing the message, you may decide to make some corrections or additions.

A simple editor in TEKCOM provides for manipulating the header and text. The commands .TO and .SUBJECT enable alteration of the heading (distribution list and subject); .EDIT enables text editing.

To add people to the distribution, simply type .TO followed by a sequence of names (separated by commas). To remove individual names, type .TO followed by REMOVE and the names to be deleted. To delete the entire distribution list, type .TO followed by REMOVE and a CR; the TO:? prompt will then be displayed and you will have to enter at least one name (or ME). TEKCOM will not let you exit the message construction mode without specifying at least one addressee.

To make changes in the subject, type .SUBJECT followed by the new text (one line only). The old subject text will be replaced completely by the new material.

To make changes in the text, type .EDIT and the edit prompt *? will appear. TEKCOM gives you a choice of three text editing commands: L (list line), K (kill line), and I (insert line). The L and K commands can specify one line number or consecutive line numbers. For example, L2 will list line two. K2-6 will delete lines 2, 3, 4, 5, and 6. The I command specifies the line before which new material is to be inserted.

After making your changes, type Q (QUIT) to exit from the editing mode. TEKCOM then returns to the text input mode, and a line-number prompt appears for the next line after the existing text.

To Read Messages . . .

The first time you use TEKCOM, you'll have no mail to read. Each succeeding time you access the system, TEKCOM tells you if there are any messages waiting for you and how many of them are new (received since you last used the system). To see a list of the messages, type SUMMARY when the prompt command #:? appears. SUMMARY lists the date, sender, and subject of each message. To read new messages only, type READ NEW. To read a particular message, type READ followed by its number. If you type READ without a number, TEKCOM will display the first message.

Continued on page 8

```
TEXT:
  1 ? .EDIT
*? HELP

YOU MAY USE THESE EDIT FUNCTIONS:

    IN        - INSERT TEXT PRIOR TO LINE N.  TEXT INSERTION IS TERMINATED BY TYPING A '.' AS
                  THE FIRST AND ONLY CHARACTER ON A LINE
    LN        - LIST LINE NUMBER N.  LN-M LISTS LINES N THROUGH M
    LN-*      - LIST EVERYTHING FROM LINE N TO END.  L* MEANS L1-*
    KN        - DELETE LINE N.  KN-M DELETES LINES N THROUGH M
    KN-*      - DELETE EVERYTHING FROM LINE N TO END.  K* MEANS K1-*
    SN        - COPIES LINE N ONTO A SCRATCH PAD.  SN-M COPIES LINES N THROUGH M
    SN-M,P    - COPIES LINES N THROUGH M ONTO A SCRATCH PAD, DELETES LINES N THROUGH M
                  AND INSERTS THESE LINES PRIOR TO LINE P
    UN        - INSERTS THE CONTENTS OF 'SCRATCH PAD' BEFORE LINE N

    L/STRING1/STRING2/  - LIST EVERYTHING FROM THE FIRST OCCURRENCE OF 'STRING1' TO
                            THE FIRST OCCURRENCE OF 'STRING2' AFTER 'STRING1'.  '/' CAN
                            BE ANY CHARACTER THAT DOES NOT APPEAR IN EITHER STRING

    G#/STRING/          - FINDS THE LINE NUMBER IN WHICH THE STRING OCCURS

    R/STRING1/STRING2/  - REPLACE 'STRING1' WITH 'STRING2'.  PRECEDING 'R' WITH 'N' WILL
                            CAUSE 'N' REPLACEMENTS OF 'STRING1'

    Q OR '.' - RETURNS TO TEXT INPUT MODE.  THE LINE NUMBER OF THE NEXT INPUT LINE IS
               PRINTED

    ALL TEXT '.' COMMANDS MAY BE USED WHILE IN THE EDITOR, BUT THE USE OF A '.' COM-
    MAND WILL ALSO RETURN TO TEXT INPUT

    A STRING IS A SEQUENCE OF ANY KEYBOARD CHARACTERS
*?
```

Figure 1. TEKCOM's menu of edit commands.

```
#:? HELP
 YOU MAY RESPOND WITH THE FOLLOWING COMMANDS:

    SEND      - USED TO COMPOSE AND TRANSMIT A MESSAGE
    READ      - USED TO READ MESSAGES IN THE MAILBOX
    SUMMARY   - PRODUCES A SUMMARY OF ALL MESSAGES IN THE MAILBOX
    DELETE    - REMOVES MESSAGES FROM THE MAILBOX BY MESSAGE NUMBER, I.E., DELETE
                  3,4-7
    NEW       - USED TO CREATE A NEW USER GROUP
    REMOVE    - DELETES AN ENTIRE GROUP OR REMOVES NAMES FROM A GROUP
    ADD       - ADDS NAMES TO A GROUP
    LOCATE    - USED TO DETERMINE WHETHER INDIVIDUALS HAVE A TEKCOM MAILBOX.  I.E.,
                  LOCATE NAME1, NAME2, ...
    TIME      - GIVES THE DATE & TIME
    QUIT      - TERMINATES TEKCOM
    HELP
```

Figure 2. TEKCOM's menu of action commands.

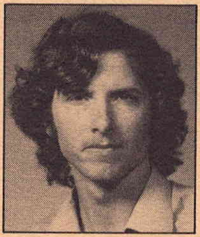
```
#:? HELP
 YOU MAY RESPOND WITH THESE COMMANDS:

    .EDIT      - ALLOWS USE OF THE EDITOR TO CORRECT ERRORS IN THE MESSAGE TEXT
    .REVIEW    - PRINTS THE ENTIRE MESSAGE
    .ERASE     - DELETES THE MESSAGE TEXT, BUT LEAVES THE HEADER INTACT
    .TO        - ALLOWS MODIFYING OR ADDING TO THE DISTRIBUTION LIST
    .SUBJECT   - ALLOWS REPLACEMENT OF THE SUBJECT
    .DONE      - TERMINATES TEXT COMPOSITION - SENDS THE MESSAGE
    .QUIT      - TERMINATES TEXT COMPOSITION - NO MESSAGE IS SENT
```

Figure 3. TEKCOM's menu of text-input mode commands.

```
DATE:     AUG 16, 1979; 10:59:04 AM
FROM:     STEVE HEITMANN
TO:       WILLIAM WEIR
SUBJECT:  FIRST SUCCESSFUL USE OF TEKCOM
TEXT:
  1 THANKS FOR THE COMMENT
  2 ?
```

Figure 4. As this example illustrates, when you type "REVIEW" while in text-input mode, TEKCOM shows you the entire message so you can decide whether you need to make changes.

FLOATING-POINT STANDARD IMPLEMENTED IN TESLA



Randy Klingenberg, Microprocessor Software Support, ext. 1905 (Walker Road).

In September 1978, the Microprocessor Floating-Point Standard Committee (an IEEE Computer Society committee) proposed a standard for floating-point formats for microprocessors.

In the August-September *Engineering News,* Randy published an article that summarizes the standard and reviews floating-point computer arithmetic basics. (For a copy, call Technical Communication Services, ext. 8920, Merlo Road.)

In this article, Randy describes an implementation of the floating-point standard in TESLA, a high-level language developed by the Tektronix Scientific Computer Center as a prototype for floating-point packages, and illustrates its use for the Motorola 6800 microprocessor.

Inevitably, floating-point systems will become widely available on the chip level. At that time, designers will want portable mathematical software.

For greater portability, all future floating-point systems should handle floating operations in the same way. The results and accuracy of these computations should not depend on each manufacturer's implementation. The industry must adopt a standardized approach that yields correct numeric results and known precision.

Hopefully, the floating-point standardization movement will produce a clearer model of practical mathematical computations and lead to a better "real" world than we now have. The Scientific Computer Center's implementation of the IEEECS Proposed Floating-Point Standard in TESLA is a step in this direction.

TESLA IMPLEMENTATION

In a floating-point system, TESLA's high-level constructs for controlling program flow allow structured implementation of numeric algorithms and also yield a highly readable source program. The TESLA compiler also provides portability across the Intel 8080, the Zilog Z80, and the Motorola 6800 microprocessors.

The prototype floating-point package described below implements, in TESLA, a subset of the IEEECS Proposed Standard; it is a base on which to build and improve. The TESLA floating-point representation yields the range and precision specified for the single-word format described in the previous article. Double precision real is not yet available.

Most floating-point operations use a floating-point accumulator and assume a floating-point number in this accumulator. The result of an operation usually stays in the accumulator. The second operand is in memory. This article uses the following notation:

FPN	: floating-point number
FPACC	: floating-point accumulator
FPADDR	: address of FPN in memory
(FPACC)	: value of FPN in FPACC
(FPADDR)	: value of FPN at FPADDR

The specific numeric representations are:

zero ______	exp = sign = fraction = 0
+ infinity ____	exp = fraction = 1's, sign = 0
− infinity ____	exp = sign = fraction = 1's
indefinite____	exp = 1's, sign = fraction = 0's

Table 1 lists the floating-point operations that Microprocessor Software Support has implemented so far.

DEFICIENCIES

The TESLA floating-point system does not implement transcendental functions (sin, cos, tan, and their inverses) or the SQRT function. Each user can tailor a floating-point system to eliminate excessive code inherent in many numerical algorithms (such as polynomial evaluation or numerical integration and differentiation). Users can also modify programs to handle error conditions by enabling traps for specific errors.

IMPLEMENTATION FOR THE 6800

The 6800 code generator for the TESLA compiler currently has a subset of the IEEECS Proposed Standard modeled after the prototype in this article. The floating-point package is written in the 6800 assembly language.

Since all floating-point numbers in their single-precision representation are 32 bits long, the programmer must declare them at this size and the numbers must be real. Real values are specified in the following format:

```
[<sign>]<digits> <.> [<digits>][<E>[<sign><exponent>]]
where <sign>       : = + | −
      <digits>     : = <digit>[<digits>]
      <digit>      : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
      <exponent>   : = <digit>[<digit>]
```

As an example of manipulation of expressions containing real numbers, consider a calculation to determine the force acting between the earth and its moon. First, the floating-point values are initialized in TESLA as follows:

```
VAR
  REAL * 32 : FORCE / 0.0 /
  REAL * 32 : PI / 3.141592 /
  REAL * 32 : GRAVCON / 6.67E-11 / # GRAVITATIONAL CONSTANT
  REAL * 32 : MOON.ORBIT / 3.84E8 / # MOON'S ORBIT RADIUS
  REAL * 32 : MOON.MASS / 7.34E22 / # MOON'S MASS (KILOGRAMS)
  REAL * 32 : EARTH.MASS / 5.98E24 / # EARTH'S MASS (KILOGRAMS)
  REAL * 32 : ELECTRON.CHARGE / −1.60210E-19 /
ENDV
```

Continued on page 8

Table 1. TESLA Floating-Point Operations

FORMAT	FUNCTION
LOD (<FPADDR>)	loads the FPN, stored at the location indicated, into the FPACC without disturbing the FPN in memory.
STO (<FPADDR>)	stores the FPN in the FPACC into the location indicated without disturbing the FPN in the FPACC.
NEG	changes the sign of the FPN in the FPACC and leaves the result in the FPACC. If the FPACC is zero, does nothing. If the FPACC has a special representation, produces the following actions: NEG (+ infinity) ______ FPACC = − infinity NEG (− infinity) ______ FPACC = + infinity NEG (indeterminate) __ FPACC = indeterminate
<Integer * 8 > = CMP (<FPADDR>)	compares the FPACC with the FPN stored at the location indicated and returns one of the following values: −1 if FPACC < FPN 0 if FPACC = FPN 1 if FPACC > FPN
ADD (<FPADDR>)	adds the FPN stored, at the location indicated to the FPN, in the FPACC. Leaves the result in the FPACC. Does not modify memory. Detects overflow and underflow if present.
SUB (<FPADDR>)	subtracts the FPN, stored at the location indicated, from the FPN in the FPACC. Leaves the result in the FPACC. Does not modify memory. Detects overflow and underflow if present.
MUL (<FPADDR>)	multiplies with rounding, the FPN in the FPACC by the FPN at the location indicated. Does not modify memory. Detects overflow and underflow if present.
DIV (<FPADDR>)	divides, with rounding, the FPN in the FPACC by the FPN at the location indicated. Leaves the result in the FPACC. Does not modify memory. Detects overflow, underflow, zero/zero, and division by zero if present.
ABS	returns the absolute value of the FPACC in the FPACC. Processes the following special cases: 0 ____________ returns 0 + or − infinity __ FPACC = + infinity, abort other __________ abort
TST	tests the FPACC against zero and returns: −1 if FPACC < 0 0 if FPACC = 0 1 if FPACC > 0 Processes the following special cases: 0 ___________ returns 0 other _______ aborts
FIX (<integer * 16>)	converts the FPACC into its 16-bit integer representation and stores the result in the argument.
FLT (<integer * 16>)	converts the argument integer into its floating-point representation. Leaves the result in the FPACC. Does not modify memory.
MOD (<FPADDR>)	performs modulo arithmetic. FPACC − ARG * [FPACC/ARG] where [X] yields the greatest integer < = x.
C2B(<Binary * 8>)	converts the base 10 floating-point number stored in the argument string variable into its corresponding floating-point representation. Allows leading blanks, but not imbedded blanks. Uses sign of + unless the first non-space character is a −. Allows optional E for the exponent and a range for any value following it of −37 to +38. Stores the resulting FPN in the FPACC.
B2C (<Binary * 8>)	converts the binary floating-point number stored in the FPACC into a character string representing the number.

Continued from page 6

The basic equation for force is:

```
Force = G (m1 * m2)(1/R ** 2])
```

Using the above variables, the TESLA equation is:

```
FORCE = GRAVCON * (EARTH.MASS * MOON.MASS)
                * (1.0/(MOON.ORBIT * MOON.ORBIT))
```

Floating-point expressions cannot mix variable types. The user must convert all variables to floating-point values before evaluating them in an expression containing other real values. This is accomplished by inserting the appropriate type conversion operators within a floating-point expression.

The 6800 implementation of TESLA type real provides the operations shown in table 2.

This implementation does not allow use of other TESLA functions such as the rotate and shifts. Real values cannot be used as FOR LOOP control variables.

Some other useful information in the 6800 implementation are declared as follows:

```
EXT
 REALSTR              # CONVERSION OF REAL NUMBERS TO ASCII
 REAL * 32 : REALVAL  # CONVERSION OF ASCII TO REAL NUMBERS
 REAL * 32 : REALINT  # TRUNCATES FRACTIONAL PORTION OF REAL
 REAL * 32 : LOG10    # COMMON LOGARITHMS
 REAL * 32 : LOGE     # NATURAL LOGARITHMS
 REAL * 32 : LOG2     # BASE 2 LOGARITHMS
ENDE
```

The procedure REALSTR converts a real expression into an ASCII string. The REALVAL function converts an ASCII representation of a real number into internal form. Function REALINT performs a truncation of the real number's fraction and returns another real number. Each log function returns the logarithm to the desired base of the passed argument, which must be a real number.

SYMBOL TABLE

+	addition
–	subtraction, unary minus
*	multiplication
/	division
MOD	modulus
ABS	absolute value
FIX	post decimal number to integer number
FLT	integer number to real number
=	assignment
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
<>	not equal
=	equal

Table 2. The 6800 implementation of TESLA type real provides the operations listed here.

FOR MORE INFORMATION

Current users of TESLA who would like to see type real implemented for the Intel 8085 or Zilog Z80 generators, and who would be able to help code the assembly language for a floating-point system, should contact Randy on ext. 1905 (Walker Road). □

Continued from page 5

When a message is displayed, it is followed by the request ACTION:?. In other words, now that you've read the message, what do you want to do about it? You can answer this request in a variety of ways.

```
YOU HAVE 5 MESSAGES
NO NEW MAIL
#:? SUMMARY
   DATE                  FROM                      SUBJECT

 1. AUG 16            WILLIAM WEIR              FIRST SUCCESSFUL USE OF TEKCOM
 2. AUG 16            RICHARD LEFAIVRE          LOWER CASE COMMANDS
 3. AUG 16            RICHARD LEFAIVRE          PROMPT CHARACTER
 4. AUG 16            RICHARD LEFAIVRE          AMBIGUITY IN ACTION COMMANDS
 5. AUG 16            RICHARD LEFAIVRE          "HELP"
```

Figure 5. As this example demonstrates, TEKCOM lets you see what messages are waiting for you so you can decide which to read first.

You may want to read the message again, especially if it is longer than the number of lines on the terminal screen. If so, type AGAIN.

You may decide the message should go to someone else for action. Type FORWARD. When the prompt TO:? appears, type the first and last name.

To get a hard copy, type PRINT. TEKCOM then lists the message on the line printer in the Scientific Computer Center, where you may pick up the copy.

To save the message in a private file, type FILE filename. FILE will create a new file if "filename" does not already exist. If the file does exist, TEKCOM will append the message to the file.

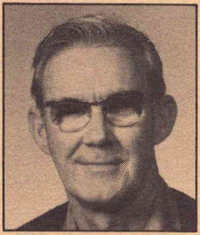
If you decide to answer the message right away, type REPLY. TEKCOM then puts you into the text input mode and you proceed as described above. However, you do not have to enter information for the heading; TEKCOM automatically inserts the proper information from the message to which you are replying.

Before you can read the next message, you must type either HOLD or DELETE. HOLD will keep the message in the mailbox to be read again at a later time. DELETE will remove the message from the mailbox.

When You're Finished . . .

When you are through with TEKCOM, type QUIT in response to the command prompt #:?. TEKCOM then ends and returns you to the operating system. □

GPIB INTERFACE FUNCTIONS AND MESSAGES



Arnold Farley, TM 500 Manuals, ext. 1552 (Walker Road).

Arnold Farley has been with Tektronix for ten years...the first three with Field Engineering Training, and seven writing manuals for TM500 products. In addition to his work as a writer, Arnold serves as a member of the GPIB Documentation Standards Committee which is seeking to standardize GPIB descriptions in Tektronix manuals.

The second of a two-part series, this article discusses interface functions and the protocol for transferring data between instruments on the General Purpose Interface Bus. The first article discussed signal-line definitions. For a copy of the first article, call Technical Communications Services on ext. 8929 (Merlo Road) and ask for a copy of the August-September 1979 issue of *Engineering News.*

INTRODUCTION

The ten interface functions of the GPIB (listed in table 1) provide a variety of capabilities and options for an instrumentation system. These functions may be implemented in, or for, any particular instrument with instrument hardware or with a programming routine (software).

INTERFACE FUNCTION	SYMBOL
Source Handshake	SH
Acceptor Handshake	AH
Talker or Extended Talker	T or TE
Listener or Extended Listener	L or LE
Service Request	SR
Remote-Local	RL
Parallel Poll	PP
Device Clear	DC
Device Trigger	DT
Controller	C

Table 1. The ten major interface functions for the GPIB.

Figure 1 illustrates the basic linkage between a GPIB controller and the interface functions implemented in another instrument on the bus. For a particular measurement or stimulus device, any or all of nine possible interface functions (SH through PP) may be selected to be linked to the tenth function (controller, C). Only those functions necessary for an instrument's purpose need be implemented by the instrument's designers; it is not likely that one instrument has all ten interface functions. For example, an instrument generally doesn't need to implement the Parallel Poll (PP) function if the instrument can respond to a serial polling sequence from the controller-in-charge (there may be more than one controller in a system).

The following is a discussion of the interface functions and their relationship to interface messages and commands.

The interface messages discussed in this article are shown in the ASCII and IEEE (GPIB) Code Chart in Table 2. All interface messages and commands, except IFC, discussed in this article are sent and received over the GPIB with the ATN line asserted low true.

TALKER AND LISTENER FUNCTIONS (T/TE AND L/LE)

Although discussed under one heading, the T/TE and L/LE functions are independent of each other.

The T and TE functions provide an instrument and its secondary devices, if any, with the capability of sending device-dependent data (or, in the case of a controller, the capability to send interface messages or device-dependent program data) over the GPIB. The T (Talker) function is a normal function for a talker and uses only a one-byte address code called **MTA** (My Talk Address); the TE (Talker Extended) function uses a two-byte address code: an **MTA** code followed by an **MSA** code (My Secondary Address).

Only one instrument in the system can be in the talker active state at any given time. A non-controller commences talking when **ATN** is released and continues its talker status until an Interface Clear (**IFC**) message occurs, an Untalk (**UNT**) command is received from the controller-in-charge, the instrument is addressed as a listener (receives My Listen Address, **MLA**), or another instrument is addressed as a talker when a data byte called **OTA,** Other Talk Address, appears on the bus.

One or more instruments on the bus (up to a maximum of 14) can be programmed for the L (Listener) function by use of their specific primary listen address (**MLA**). All or none of these instruments may be programmed for the LE (Listener Extended) function (if implemented). The LE function requires a secondary listen address (**MSA**).

An instrument on the bus may be a talker only, or a listener only, or have both functions (T/TE and L/LE). In any case, its address code has the form X10TTTTT for a talker and X01LLLLL for a listener. For instruments with both functions, the T-bit binary values are equal to the binary value of the L bits. The system operator sets these five bits by

Continued on page 11

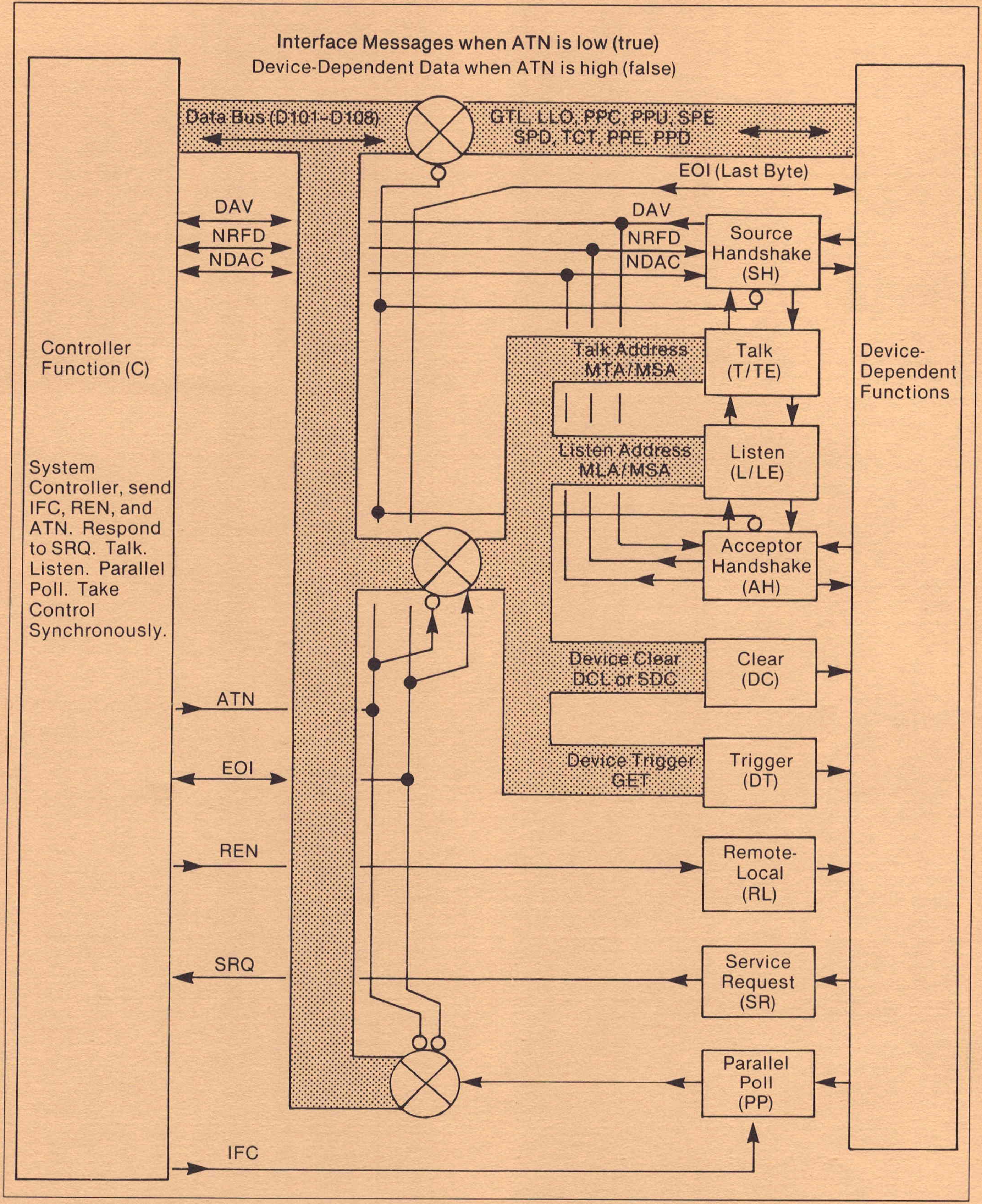


Figure 1. A maximum of ten interface functions can be interlinked between instruments on the GPIB.

Continued from page 9

means of address switches on each instrument before applying power to the system. The controller's address code may be implemented in software.

If an instrument has TE or LE functions, the five secondary address bits must not be set to the same value as the primary address bits. The system program, from the controller, designates the primary talker and primary listener status of the desired instruments by coding bits 6 and 7 (10 for talker and 01 for listener). Secondary listen addresses (or commands) are represented by the controller sending both bits (6 and 7) as a 1.

A talker may assert the **EOI** line with the last data byte or send a special "end of message terminator" code so that the assigned listeners will know that the talker has no more data transfer to send.

SOURCE AND ACCEPTOR HANDSHAKE FUNCTIONS (SH AND AH)

Like the T/TE and L/LE functions, SH and AH are totally independent of each other. The SH function guarantees proper transmission of data, while the AH function guarantees proper reception of data. The interlocked handshake sequence between these functions guarantees asynchronous transfer of each data byte.

Both functions utilize the **DAV, RFD,** and **DAC** messages to transfer each byte. (For details, see figures 2 and 5 and the discussion under "The Transfer Bus (Handshake)" in GPIB Systems Concepts, **Engineering News,** August 1979; call ext. 8920 (Merlo Road) for a copy.) The SH and AH functions in some instruments, but especially in controllers, allow the source (talker) to listen to itself, with or without **ATN** asserted. Both functions must respond to the **ATN** message within 200 nanoseconds.

DEVICE CLEAR FUNCTION (DC)

The Device Clear function allows a controller-in-charge to **clear** (initialize) an instrument, either individually or as part of a group of instruments. The group can be either a part or all of the addressed instruments in one system.

The controller (under program direction) asserts **ATN** and sends either the universal Device Clear command (**DCL**) or the Selected Device Clear command (**SDC**). When the **DCL** message is received, all instruments on the bus must clear or initialize their internal device functions. When the controller sends the **SDC** command, only those instruments which have previously been addressed to listen must clear or initialize their internal device functions. The IEEE 488 standard does not specify the state an instrument goes to as a result of the **DCL** or **SDC** command; it may be, but does not have to be, the power-up default setting.

DEVICE TRIGGER FUNCTION (DT)

The Device Trigger function allows the controller-in-charge to start the basic operation of an instrument, either by itself or as part of a group of instruments. The group may be either a part or all of the addressed instruments in one system. The IEEE 488 standard does not specify an instrument's basic operation when it receives the **GET** (Group Executive Trigger) command. To issue this command, the controller asserts **ATN,** sends the listen addresses of the instruments which are to respond to the trigger, and then sends the **GET** message.

Once an instrument starts its basic operation, the instrument must not respond to subsequent trigger-state transitions until the current operation is complete. Only after completing the operation can the instrument start the same operation in response to the next **GET** message; thus the basic operating time is the major factor that determines how fast the instrument(s) can be repeatedly "triggered."

REMOTE-LOCAL FUNCTION (RL)

The Remote-Local (RL) function provides an instrument with the capability to select between two sources of information input. The function indicates to the instrument that its internal device-dependent functions respond to information input from the front panel (Local) or corresponding information input from the GPIB (Remote). Only the system controller is permitted to assert the **REN** line, whether or not it is the controller-in-charge at the time.

When the system controller asserts the **REN** line, an instrument on the GPIB goes to the remote mode when it is addressed as a listener with its primary address, not before. In this case only, the primary listen address is sufficient to cause an instrument to go to the remote mode. For example, if several instruments have a different secondary, but a common primary address, they will all go to the remote mode when the primary address is received.

An instrument remains in the remote mode until the **REN** line is released, a front-panel switch on the instrument is activated to request the local mode, or a Go to Local (**GTL**) command is received while the instrument is enabled as a listener. However, the controller can disable the local mode function of an instrument by sending a Local Lockout (**LLO**) command, which applies to all instruments on the bus, addressed or not. The **UNL** (Unlisten) command does not return an instrument to a local mode.

All instruments must recognize when the **REN** line goes false (a high voltage level) and go to the local mode within 100 microseconds. If data bytes are still being placed on the data bus when **REN** goes false, the system program should assure that the data bytes are sent and received with the knowledge that the system is in a local mode, not remote.

CONTROLLER FUNCTION (C)

The Controller function provides the capability to send primary talk and listen addresses, secondary addresses, and universal commands to all instruments on the bus, and secondary commands to previously-addressed instruments. The controller function also provides the capability of responding to a service request (**SRQ**) message or conducting a parallel poll routine to determine the status of any or all instruments.

If an instrumentation system has more than one controller, only the system controller is allowed to assert the **IFC** and **REN** lines at any time during the system operation, whether or not it is the controller-in-charge at the time.

Continued on page 13

ASCII & IEEE 488 (GPIB) CODE CHART

B7 B6 B5 BITS	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
Ø Ø Ø Ø	0 NUL 0 0	20 DLE 10 16	40 SP 20 32	60 0 30 48	100 @ 40 64	120 P 50 80	140 ` 60 96	160 p 70 112
Ø Ø Ø 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 ! 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a 61 97	161 q 71 113
Ø Ø 1 Ø	2 STX 2 2	22 DC2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b 62 98	162 r 72 114
Ø Ø 1 1	3 ETX 3 3	23 DC3 13 19	43 # 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c 63 99	163 s 73 115
Ø 1 Ø Ø	4 SDC EOT 4 4	24 DCL DC4 14 20	44 $ 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 d 64 100	164 t 74 116
Ø 1 Ø 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e 65 101	165 u 75 117
Ø 1 1 Ø	6 ACK 6 6	26 SYN 16 22	46 & 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f 66 102	166 v 76 118
Ø 1 1 1	7 BEL 7 7	27 ETB 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g 67 103	167 w 77 119
1 Ø Ø Ø	10 GET BS 8 8	30 SPE CAN 18 24	50 (28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h 68 104	170 x 78 120
1 Ø Ø 1	11 TCT HT 9 9	31 SPD EM 19 25	51) 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i 69 105	171 y 79 121
1 Ø 1 Ø	12 LF A 10	32 SUB 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 j 6A 106	172 z 7A 122
1 Ø 1 1	13 VT B 11	33 ESC 1B 27	53 + 2B 43	73 ; 3B 59	113 K 4B 75	133 [5B 91	153 k 6B 107	173 { 7B 123
1 1 Ø Ø	14 FF C 12	34 FS 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \ 5C 92	154 l 6C 108	174 ¦ 7C 124
1 1 Ø 1	15 CR D 13	35 GS 1D 29	55 − 2D 45	75 = 3D 61	115 M 4D 77	135] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 Ø	16 SO E 14	36 RS 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 ∧ 5E 94	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47	77 UNL ? 3F 63	117 O 4F 79	137 UNT _ 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

Interface messages are sent with ATN asserted.

KEY

octal	25 PPU	GPIB code
	NAK	ASCII character
hex	15 21	decimal

Table 2. ASCII and IEEE 488 (GPIB) Code Chart.

Continued from page 11

The controller function has specified time intervals for certain operations. For example, the execution time for parallel polling instruments on the bus cannot be less than 2 microseconds. The **ATN** message must have a controller delay of at least 500 nanoseconds to allow a current talker to see the **ATN** line asserted before placing a new data byte on the bus. The **IFC** message must be asserted for at least 100 microseconds.

If a controller requests system control from another controller and receives an internal message to send the remote enable message (**REN**), the controller must verify that the **REN** line remains unasserted (false) for at least 100 microseconds before asserting **REN.** The time interval that **REN** is asserted depends on the remote programming sequence and will vary with the program.

If a controller is in the controller active wait state and does not receive an internal message to conduct a parallel poll, it must wait at least 1.5 microseconds before going to the controller active state to give the NRFD, NDAC, and EOI lines sufficient time to assume their valid states.

Taking Control (Asynchronous or Synchronous). All data bytes transmitted over the GPIB with **ATN** asserted are interpreted as system control information. Asserting **ATN** directly at any moment is an asynchronous operation with respect to the bus and may cause loss of data if a handshake cycle is in progress. To prevent loss of data, a controller can take control synchronously with the handshake cycle (if it is in progress) by first asserting the NRFD line to stop the next handshake cycle, and then automatically asserting **ATN** when the current talker releases DAV.

Taking control synchronously presents problems; the **ATN** line may not become asserted automatically if the data transfer has stopped for some reason. For example, (1) the talker may have finished talking (sent the last data byte), or (2) the talker may be very slow to send the next data byte, or (3) an instrument on the bus may not be functioning properly. Programmers can solve the first problem if they know that all talkers on the bus assert **EOI** with the last data byte. The second and third problems may require asserting **IFC** to clear the bus and then asserting **ATN** asynchronously.

Performing a Serial Poll. The controller may conduct a serial poll at any time, whether or not an instrument has asserted the **SRQ** line. Most, but not all, instruments have the Service Request (**SRQ**) function.

To perform a serial poll, the controller first asserts **ATN** and issues the Untalk (**UNT**) and Unlisten (**UNL**) commands. The controller then sends Serial Poll Enable (**SPE**) command, followed by the talk address of the first instrument to be polled. The controller then releases **ATN**, and the addressed talker responds by sending its status byte over the data bus. If the addressed talker has requested service, it must assert bit seven of the status byte and encode the remaining seven bits of the status byte to indicate the reason for asserting **SRQ.** Status bytes are device-dependent and are not specified in the IEEE 488 standard.

An addressed instrument will release its **SRQ** line when serial polled, but other instruments may still be holding it asserted. When the controller has read the status byte of an addressed instrument, it should send the **UNT** and Serial Poll Disable (**SPD**) commands before repeating the procedure to poll the remaining instruments. The routine should continue until the controller no longer detects **SRQ** asserted.

Performing a Parallel Poll. The Parallel Poll (PP) function provides an instrument with the capability to present one, and only one, bit of status information to the controller without being previously addressed to talk. The parallel polling capability requires a commitment by the system program to periodically conduct a parallel poll sequence.

When an instrument responds to a parallel poll, the single data bit presented to the controller may or may not indicate a need for service. If the data bit is used as a service request function, the controller should perform a serial poll in order to obtain a complete status byte.

Before an instrument can respond to a parallel poll, the GPIB system must first be configured. The controller asserts **ATN,** sends the **UNT** command followed by the listen addresses of all instruments to be included in the parallel poll, and then sends the Parallel Poll Configure (**PPC**) command. The **PPC** command is followed immediately with the Parallel Poll Enable (**PPE**) command to cause all listeners to go to the parallel poll standby state.

If the **EOI** line has been asserted along with **ATN,** all selected instruments have up to 200 nanoseconds to go to the parallel poll active state. **EOI** may be asserted along with the **PPE** command, or at any time after the **PPE** command. An instrument does not present its one bit of status information to the controller until it sees both **ATN** and **EOI** asserted.

The **PPE** message sent by the controller has the form X11OSPPP. Bit 4 (S) is called the sense bit, and PPP is an octal number (000=0 through 111=7) designating a specific data line (DI01 - DI08) that an instrument must assert if its internal status message has the same value as the sense bit (S may equal 1 or 0). If so designed, the controller can read the data lines while **ATN** is asserted to interpret the status of the instruments.

To conclude the parallel poll, the controller releases **EOI** and then sends the Parallel Poll Disable (**PPD**) command. If the system needs to be reconfigured, the Parallel Poll Unconfigure (**PPU**) command is sent, followed by the Unlisten (**UNL**) command.

Passing Control. As a controller-in-charge, the system controller (program) may relinquish control to any other instrument in the system capable of acting as a controller. The controller-in-charge first addresses the other controller as a talker, and then sends the Take Control (**TCT**) command and other desired control messages. The other controller becomes controller-in-charge when **ATN** is released. ☐

TECHNOLOGY REPORT'S EDITORIAL PROCESS

Laura Lane, Technical Publications (part of Technical Communications Services), ext. 8927 (Merlo Road).

Technology Report welcomes articles from Tektronix' technical community. As a member of that community and as a potential author, you may ask, "What happens after I submit a draft?" We'd like to answer that question and give you a clear picture of how we work.

YOUR FIRST DRAFT

As the author, you have the responsibility for producing a first draft...for putting your message down on paper. If you're not a writer, or if you feel unsure of your writing ability, you are no less qualified to submit material to us than if you find writing a welcome and easy task.

Your initial input can take one of two forms: a complete first draft, or (if you have trouble getting started, as many writers do) we can help you write an outline which you then fill in to produce the first draft.

Our job is to edit your information so that the end product conforms to our editorial criteria: accuracy, clarity, and conciseness — in that order of priority. Your job, as the expert on the subject, is to see that we correctly interpret your input. We strive to present your information in the most professional manner possible — we want the final article to be something that you are proud to have in print.

OUR FIRST DRAFT

When we receive your input, we assign it to a **Technology Report (TR)** editor. Using your draft, the editor produces the first of a series of edited drafts. In editing each draft, our goal is always the same: to ensure that the article meets our editorial criteria.

REVIEWS

The **TR** editor then submits the article to another editor in the group for review. In addition to reviewing for our editorial criteria, this second editor asks such questions as: Is the language usage proper? Does the article flow smoothly? Does it follow a logical progression?

After this review, the **TR** editor incorporates suggestions made by the reviewing editor. Depending on the complexity of the article and the extent of editorial criticism by the reviewing editor, a third editor may review the article.

YOUR REVIEW

When the draft is ready, the **TR** editor sends a copy to you, the author. At this time, we ask you to check for accuracy and clarity. This is the time to make sure the article says what you want it to say. We can easily make changes at this point.

When you return the article, the **TR** editor incorporates your changes. If the changes are extensive, another editor within our group again reviews the article.

OUTSIDE REVIEW

Depending on the subject's complexity or controversiality, the **TR** editor may next send a copy of the draft to a reviewer inside the company who has some expertise on the subject. If extensive revisions are required, the editor sends you the new draft again for review.

TYPESETTING

When the above steps are complete, we are ready to typeset the article. We turn the material over to the **TR** graphic designer who arranges to have the text typeset and begins work on illustrations, charts, schematics, and photos. When typesetting and artwork are complete, the **TR** editor sends a copy of the typeset text and the final artwork to the author for a final review.

THE LAYOUT

By this time, the article should meet **Technology Report's** editorial requirements and be satisfactory to all involved, especially to you, the author. When the typeset material is returned to us with your signed approval, we return the article to the graphic designer who makes the changes requested by you and the editor. The graphic designer then lays out the issue and, after the managing editor approves it, sends it to the printer.

THE PRINTER'S PROOF

The printer makes a photographic proof copy from the graphic designer's layout of the issue and sends it for our approval. This final check is for errors that slipped by us or were made by the printer. To ensure that the issue is as clean as possible, the graphic designer checks for uneven lines, broken type, spots on the paper, correct page numbers, and a long list of other details; the managing editor double-checks the proof copy.

FINALLY....

After both the managing editor and graphic designer approve the proof copy, the printer makes any final changes and starts the presses rolling. About a week later, the mailroom distributes **Technology Report** to Tektronix' technical community. □

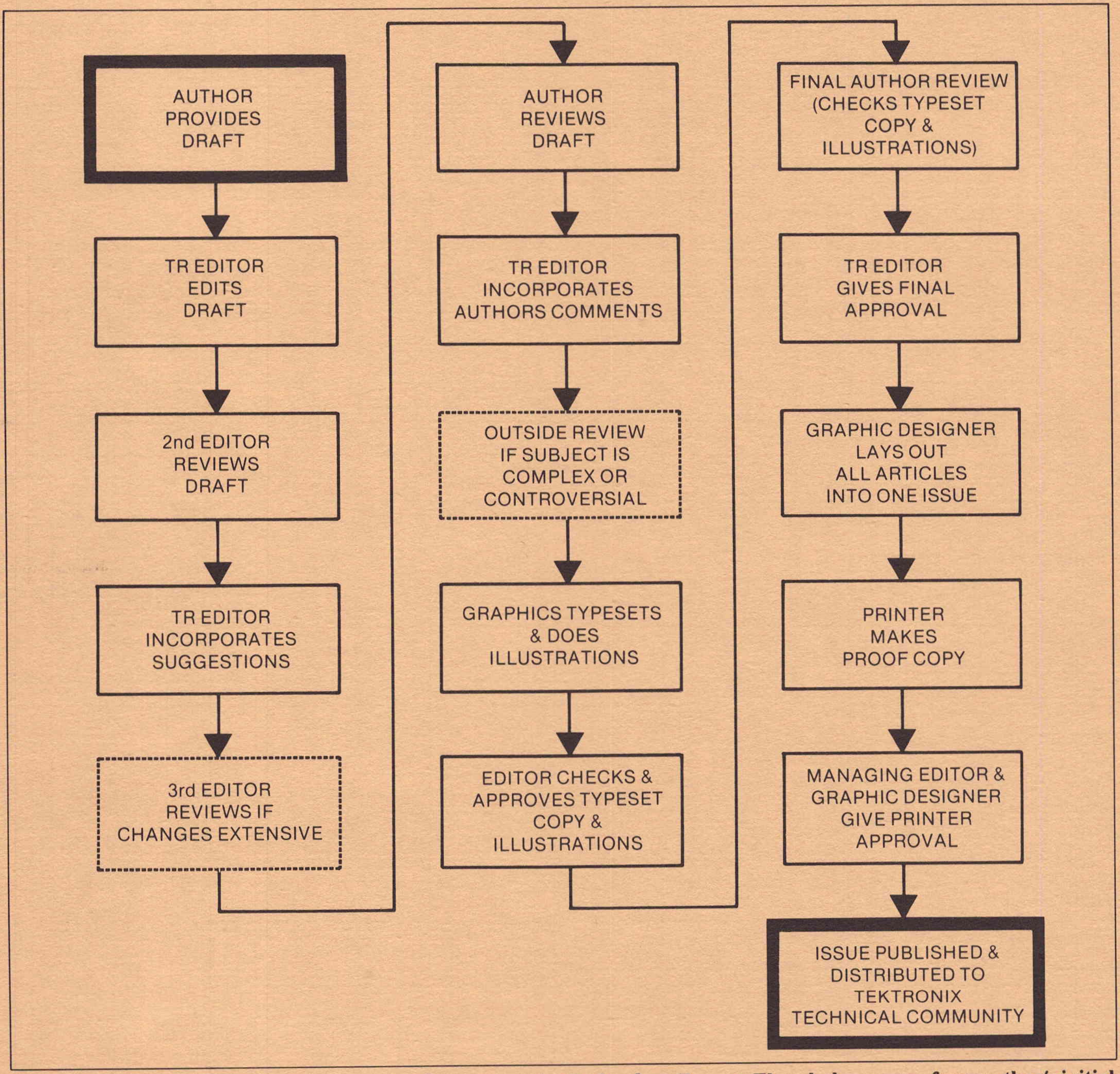
TECHNOLOGY REPORT'S EDITORIAL CRITERIA

Accuracy: Is the content accurate? Ultimately, this is the author's responsibility.

Clarity: Is the article clearly written for the publication's audience? TR's audience consists of people with strong but specialized technical backgrounds, but not necessarily the same specialization as the author.

Conciseness: Does the article express the ideas as succinctly as possible while still maintaining clarity for a wide audience?

Editing articles to meet these criteria helps us to present your information in the most professional manner possible. We want the final article to be something that you are proud to have in print.


AUTHOR PROVIDES DRAFT
TR EDITOR EDITS DRAFT
2nd EDITOR REVIEWS DRAFT
TR EDITOR INCORPORATES SUGGESTIONS
3rd EDITOR REVIEWS IF CHANGES EXTENSIVE
AUTHOR REVIEWS DRAFT
TR EDITOR INCORPORATES AUTHORS COMMENTS
OUTSIDE REVIEW IF SUBJECT IS COMPLEX OR CONTROVERSIAL
GRAPHICS TYPESETS & DOES ILLUSTRATIONS
EDITOR CHECKS & APPROVES TYPESET COPY & ILLUSTRATIONS
FINAL AUTHOR REVIEW (CHECKS TYPESET COPY & ILLUSTRATIONS)
TR EDITOR GIVES FINAL APPROVAL
GRAPHIC DESIGNER LAYS OUT ALL ARTICLES INTO ONE ISSUE
PRINTER MAKES PROOF COPY
MANAGING EDITOR & GRAPHIC DESIGNER GIVE PRINTER APPROVAL
ISSUE PUBLISHED & DISTRIBUTED TO TEKTRONIX TECHNICAL COMMUNITY

This flow chart shows the editing and production process for *Technology Report*. The whole process, from author's initial draft input through publication, takes about 3-1/2 months.

Why does it take so long to publish? Each editing and review cycle takes at least three days, and we may pass through as many as eight cycles. Once the author and editor agree on the final form of the article, the text and illustrations go into production. Production includes graphic design, technical illustration, specifying type, typesetting, photography, and layout. Production requires about four weeks. Printing and distribution require about three more weeks.

ARCHIVING *DOES* MAKE A DIFFERENCE

Linda Todd, Teknet Group in Electronic Data Processing Operations, (formerly with Logic Development Products Microlab Project), ext. 6123 (Beaverton).

Linda Todd worked as a programmer for the Commonwealth of Virginia from 1972 to 1976. After moving to Portland in 1976, she worked with the Georgia-Pacific Corporation until 1978. Both institutions carry strong emphasis on reliable archiving and use well-established procedures. Her work as a software engineer at Tektronix started in 1978.

***Archiving* means filing or collecting records or documents in a place in which records or historical documents are preserved.**

At Tektronix, there is a corporate archiving policy which states, "The minimum archival documentation will be on file with Reprographics prior to a Product Engineering Release (ER)." Some software people believe archiving should begin during product development and within each business unit long before archived material is sent to Reprographics. The following article describes a method proposed by one engineer for archiving software/firmware *prior* to release to Reprographics.

Giving archiving low priority often leads to improper archiving and archiving mistakes, which in turn can cause frustration, wasted time and effort, and an inability to readily track down correct information. Conversely, strong archiving support leads to ease in (1) troubleshooting, (2) maintenance, (3) long-term product support, and (4) future product design.

One of my first projects at Tektronix was to do maintenance on an existing software module. I got the source from the archiver and added instructions to correct a bug. When I tested the new version against the program currently in use, four or five unexpected bugs appeared. I reviewed my code changes and was unable to find any reason. The bugs were completely unrelated to the changes I had made.

Becoming suspicious, I tried assembling the source that I had been given by the archiver instead of using the production copy of the object module. I found that the errors were in the archived source, even though they were not in the object module that was distributed by Manufacturing.

The object module was clean because the source bugs had been fixed before the final pass of evaluation, but the old version of the source had been archived. The corrected version of the source had since been lost, and I had to fix the same bugs fixed by a previous programmer.

So, one small maintenance turned into one big waste of time, *due entirely to improper archiving.*

HOW CAN ARCHIVING-BASED PROBLEMS BE AVOIDED?

Take archiving seriously. All software-related items must be archived to permit reproduction and modification activities which support customer service, long-term product support, and future product design. Archived material must allow rebuilding any program or product with no resources other than those archived, and there must be ready access to those resources.

Archiving must proceed *in conjunction with* the progress of a project. Archived software and documentation are essential to later maintenance. Reconstructing work archived incorrectly or not archived at all can waste hours, days, or even months.

Archivers must be recognized as useful members of the product development team. A stable archiving group acquires experience over time resulting in efficient and accurate archiving. This also ensures faster turnaround time.

Know what needs to be archived. The design package handed over to the archiving team should, at minimum, include source and object modules, command files, in-house and customer documentation, non-standard software tools (with their instructions), and all manuals.

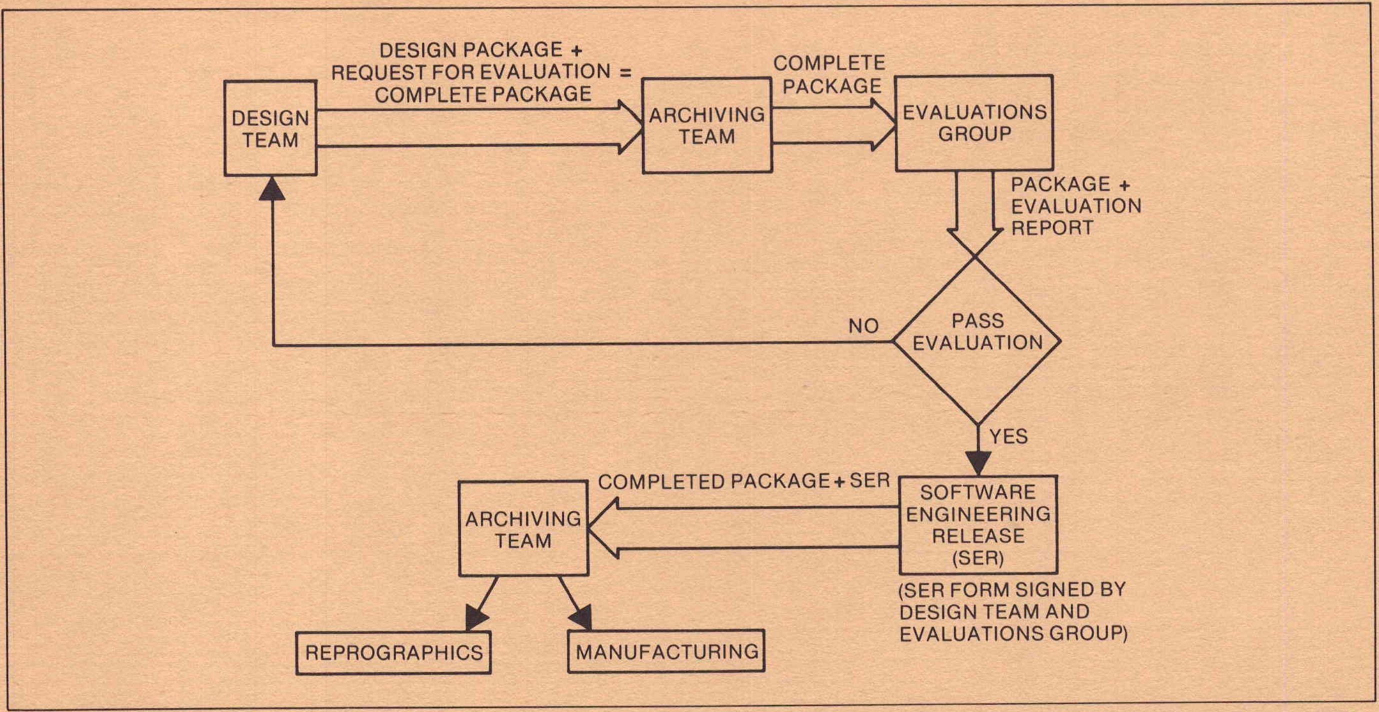
For example, an LDP program for exercising the Z80 assembler might take one week to design and three weeks to code and test. Of course we want to archive it and avoid redoing a man-month of work when the Z80 assembler requires maintenance.

Know when to archive. Software modules and documentation pass through many hands during development. Ensure that archived modules and released modules match and that archived modules are compatible with each other.

When translated, a source module should produce the object module being tested or archived. There are two common reasons why they might not match:

- An old copy of the source gets mixed up with the more recent version.
- Machine code "patches" are made to the object module, but the changes are never made to the source.

During software development, continuous informal communication between programmers and evaluators occurs. There are, however, two stages at which archivers should intervene, delaying the project only long enough for all material to be archived.


DESIGN PACKAGE + REQUEST FOR EVALUATION = COMPLETE PACKAGE
DESIGN TEAM
ARCHIVING TEAM
COMPLETE PACKAGE
EVALUATIONS GROUP
PACKAGE + EVALUATION REPORT
PASS EVALUATION
NO
YES
SOFTWARE ENGINEERING RELEASE (SER)
(SER FORM SIGNED BY DESIGN TEAM AND EVALUATIONS GROUP)
COMPLETED PACKAGE + SER
ARCHIVING TEAM
REPROGRAPHICS
MANUFACTURING

In the product development archiving system described in this article, the design team sends a completed Request For Evaluation form and the design package (source programs, source listings, and documentation) to the archiving team. The archiving team archives the package, then transfers it to the evaluations group. After the evaluations group completes their evaluation, they produce a report stating whether or not the design has passed evaluation. If the design doesn't pass evaluation, the cycle is repeated. If the design passes evaluation, the design team and the evaluations group clear it for Software Engineering Release informing the archiving team that this is the final package. The archiving team sends this final package to Reprographics for corporate archiving and to Manufacturing for production of the software/firmware and assembly of the documentation.

The first stage is the Request for Evaluation form. When the appropriate people complete and sign this form, all material should go directly to archiving. Archiving will pass the package to the evaluations group *after* archiving it. If a second or third pass is required, the same procedure should be followed.

The second stage is the Software Engineering Release (SER). When the package is cleared by both the project leader and by the evaluations group, it must again go through archiving. The project leader should inform the archiving team that the package archived for the latest evaluation is the official release of a particular program version and, after archiving, the team can then discard all passes previous to the SER. Only at this point should material be released to manufacturing and corporate archiving, and then only by the archiving team.

Define archiving procedures. Well-defined procedures are essential to proper archiving. The following list suggests procedures for well-organized and easy-flowing archiving.

- Have one person (or team) be responsible for archiving

Continued on page 18

ARCHIVING COROLLARIES TO MURPHY'S LAW

1. Essential programs are never archived.
2. Insignificant programs are always archived perfectly.
3. Maintenance is never required on a program if the source is archived correctly.
4. Rush maintenance is always required if the source was never archived.
5. The archived source will never match the archived object code. Neither will match the program that is released to the field.
6. If the correct source and object modules are inadvertently archived, an old version of the object that destroys every 47th byte on the system device will be released to the field.
7. If a program is well-written and easy to understand, detailed internal specs will be written and archived correctly.
8. If a program is obscure, convoluted, and essential, no internal specs will be written. The programmer will then leave to join a commune in New Mexico.
9. The user documentation will not match the released software. Neither will match the modules verified by evaluation.
10. If five copies are made of a valuable archive tape or disc, none will be verified and all will be unreadable past the third byte.
11. If no archiving is done, and the programmer is expected to store the latest copy of the software, the programmer will play with the program and add personal messages. The programmer will forget about the messages when doing a rush maintenance. Evaluation will miss them, but the sales rep will find them during the demo for a big sale.

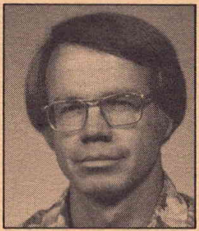
Continued from page 17

project software and documentation and for *all* the archiving, evaluating, and releasing.

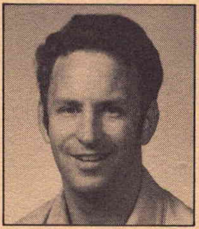
- Have software designers, coders, and writers submit their modules to the archiving team before submitting them for evaluation.
- Avoid the trap of saving time by reducing the archiving effort. A well-staffed and well-organized archiving team results in minimal delays. A 24-hour turnaround time is reasonable.
- Have the software designer furnish a command file and instructions for creating executable modules from the source. Have the archiving team assemble, compile, or link the source to create the executable modules.
- Submit documentation in machine-readable form along with complete instructions for creating the document from the input file (assuming a text formatting program processes the documentation). The archiving team then builds each document from the input file using the instructions and command files given. The team should never assume that the input and document files match. Archive text formatting programs written in-house if they have not been previously archived.
- Have the archiving team archive the modules and *then* supply them to the evaluations group. The evaluations group will review software performance, code maintainability, manuals, internal documentation, and external documentation as one unit.
- Have the archiving team process every phase of software and documentation prior to evaluation. The final set of archived modules is the one that passes evaluation. Manufacturing produces software and assembles documentation from the final archived set only.
- Start with the archived copy of the modules when maintaining software or changing documentation. Never ask a member of the design team for the source and risk getting a copy that may not be the latest and may contain "custom" modifications. The same evaluation and release procedures should be followed for maintenance as those followed for the initial software release. □

PATENT RECEIVED: No. 4,159,439

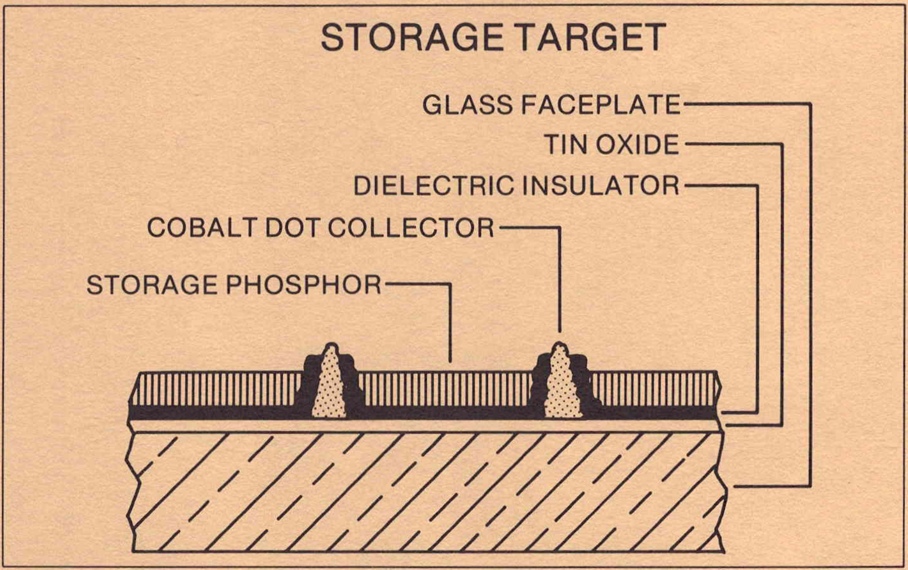
NEW CRT DISPLAY DECREASES BACKGROUND LUMINANCE



Duane Haven, Display Research (Tektronix Laboratories), ext. 6388 (Beaverton).



Bob Arneson, Component Development (Tektronix Laboratories), ext. 5607 (Beaverton).



The cross-sectional drawing shown here represents a direct-view storage tube (dvst) that, by decreasing background luminance, increases the contrast between trace and background. This increased contrast greatly improves viewability of a bistable-storage cathode ray tube (crt) display.

In other dvst's, there is a luminescent phosphor ring (sometimes called a "rim-lit" area of phosphor) around the collector electrode. Bombardment of the storage phosphor by primary flood electrons in the collector electrode's vicinity causes this rim-lit area. The resulting background luminance, approximately 10% to 20% as bright as the written trace luminance, causes low contrast in low ambient light.

This invention provides a collar of non-luminescent dielectric material around each collector electrode. The collar isolates the collector electrode from the storage phosphor, yet allows collection of secondary electrons at the exposed outer ends of the collector electrode.

Thus, the dielectric material prevents rim-lighting by intercepting primary flood electrons which would otherwise bombard the storage phosphor at its interface with the collector electrode.

This invention could be applied to any storage monitor or oscilloscope crt. □

REDUCING RISKS IN DESIGNING WITH LSI

William Broderick (Microprocessor Design Lab national program manager), Los Gatos, California field office.

This is the conclusion of a two-part article. Part one described some LSI data sheet errors and misleading assumptions that designers using LSI should be aware of. This part discusses techniques that reduce the risks of designing with LSI.

For a copy of part one of this article, call Technical Communications Services on ext. 8920 (Merlo Road) and ask for a copy of the April/May *Engineering News*.

Symbol	Parameter	Min.	Max.	Affected By	Used By	All Parameters Specified on Data Sheet
$I_{\ell\ell}$	ALE Pulse Width	400	500	$I_{a\ell}$	I_{aw} I_{ad}	Yes
$T_{a\ell}$		200		$T_{\ell\ell}$	T_{ad}	No
T_{mr}		200		$T_{\ell\ell}$	T_{ad}	Yes
T_{mr}		200		$T_{\ell\ell}$	T_{ad}	Yes
T_{mr}		200		$T_{\ell\ell}$	T_{ad}	No

Figure 1. This table of control and data signals for an LSI device will help a design team determine parts count and rom space.

ASSUME LSI LIMITS

A designer can create a versatile design "safety valve" by assuming that most program initialization and restart routines that support a maximum configuration of an end-user product must be implemented in software rather than hardware. For example, the Floppy Disk Controller Chip (FDCC) manufacturer claims the chip interfaces up to four drives. Assume that little or no extra housekeeping for track/sector is performed by the FDCC chip for the third and fourth drives. If the assumption is correct, the rom is used as planned. If, however, the FDCC chip proves to carry more program overhead for the system programmer, the extra rom can be used for more important benefits like executing mini-diagnostics upon power-up conditions.

LIST ALL DATA SIGNALS

Make a table, as in figure 1, which lists all LSI device and user data signals called for in the design (for both dynamic and static levels). Review the most recent data sheets. When chips from different vendors are used, carefully check minimum clock time. Record each signal, noting signal "to" and "from" locations along with bus control and data interface lines. Determine where each line is used and identify those signals and gates which create or affect each signal.

The table should also include minimum and maximum values from chip data sheets. When data is missing, or maximum values are not included, contact the vendor. The table should be completed at design start to determine parts count and rom space.

Also be aware that LSI designs do not always protect companion chips from spurious current during power-up operations. Particular care is necessary in power on/off conditions where separate power supplies are used for subsystems.

ADD PROTECTION HARDWARE

System designers seldom add hardware to reduce risks except in power/current protection. However, the benefits of hardware/software overlap provided by microprocessors offer some interesting risk-reducing opportunities. For example, a simple gate structure added to monitor bus address lines will verify that the software portion of the design is operating within the proper address space.

A hardware decode circuit added to the address lines will monitor invalid address, and jam any op code it finds into the program counter, forcing an interrupt to the monitor program to help the system under control.

Continued on page 22

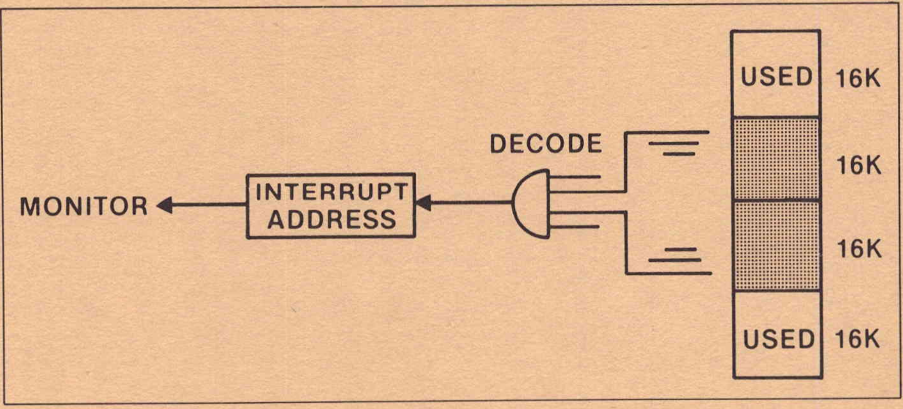


Figure 2. This additional circuit decodes unused address space to force the interrupt vector to system monitors.

PROFILE

DIGITAL PRODUCT COORDINATION

Digital Product Coordination members are (left to right): Hal Cobb, Vicki Powe, Al Zimmerman, John Sheppard, Tom Hamilton, and Bob Down. Not shown is Kathi Soder.

Digital Product Coordination (DPC) has a tough job. Chartered in 1977 by Bill Walker (executive vice president), DPC's objectives are (1) to unify development work being done in different business units for programmable digital products and (2) to ensure the resulting products' compatibility.

Compatibility is foremost on DPC group members' minds. To them, it means not only that products work together, but also that they appear to the customer to have come from the same company...through manuals, programming formats, mass-storage devices, system features, packaging, and overall design philosophy.

A large part of DPC's objectives are achieved through coordination groups, seminars, and reports.

Coordination groups, with members from all the business units, discuss subjects ranging from how messages are sent over the GPIB to human-factors considerations. Each group includes practicing specialists in the appropriate discipline and limits its deliberations to its area of expertise. These groups are defining recommended design practices for product designers. (See "Digital Product Coordination Committees" table on page 21.)

Every couple of months, DPC conducts a **Programmable Instrument Seminar** to bring product designers together and to raise issues affecting systems compatibility.

DPC provides **regular reports** for engineering management on the progress of all coordination groups and on any outstanding problems or opportunities.

In addition, DPC provides a number of services to instrument designers. For GPIB-compatible-product designers, for example, DPC offers consultation for problems with the GPIB interface, message protocol, and programming formats.

DPC also maintains an information service about programmable products being developed at Tektronix as well as programmable (GPIB) products available from other companies. DPC has a large file of such information (including operators' manuals) available for use at d.s. 50-428.

THE PEOPLE

Though the six DPC group members have specific specialties, they plan strategy and decide on approaches to coordination tasks together. These people are: Hal Cobb, product planning; Bob Down, product information; Tom Hamilton, systems software; John Sheppard, product documentation; Kathi Soder and Vicki Powe, secretaries; and Al Zimmerman, manager.

FOR MORE INFORMATION

For information about the coordination committees, contact the chairpersons listed in the table. For more information about the Digital Product Coordination group, call Al Zimmerman, ext. 7095 (Beaverton), or drop him a line at d.s. 50-428. □

DIGITAL PRODUCTS COORDINATION COMMITTEES

TOPIC	OBJECTIVE
Programming Manuals	Standardize a format for documenting programming instructions and GPIB-related information for programmable instrument users. **George Dunn, ext. 8922 (Merlo Road).**
Codes and Formats	a) Standardize procedures for assembling and sending over the GPIB such data as programming information, measurement results, waveforms, and status information; b) write and distribute a description of Tektronix GPIB Codes and Formats for novice users. (In July, 1979, Technical Standards released the committee's findings as the new Tektronix GPIB Codes and Formats Standard, 062-1780-01, Rev. C.) **Maris Graube, ext. 6234 (Beaverton).**
Waveform Peripherals	Define GPIB peripherals to support waveform-digitizing instruments. **Hal Cobb, ext. 5285 (Beaverton).**
Packaging	Establish standards and agreements for instrument packaging and for unified appearance, mounting, cooling, and cable handling for instrument systems. **Garry Burgess, ext. 5347 (Beaverton).**
GPIB Interface Components	Establish a standard interface module, including firmware, to handle all GPIB functions and Tektronix GPIB Codes and Formats operations. **Maris Graube, ext. 6324 (Beaverton).**
Firmware/Software Program Archiving	Establish corporate archives for programs which are resident in, supplied with, or used internally on Tektronix microprocessor-based products. **Tom Hamilton, ext. 5491 (Beaverton).**
Product Specification	Revise the EIS form, in use in all business units, and update it to include pertinent GPIB-related product specifications (such as functions implemented, control language, status-byte definition, and timing). **John Sheppard, ext. 5165 (Beaverton).**
GPIB Compatibility Evaluation	Provide evaluation and feedback to designers about operational compatibility of new Tektronix GPIB products with other products. **Bob Cram, ext. 5397 (Beaverton).**
Design for Testability	Coordinate a company-wide effort to improve the testability of new products. Four separate-but-related subcommittees are underway: **Internal Diagnostics.** Develop guidelines for a "core set" of internal diagnostics in microprocessor-based products. (Operations and Marketing Group Vice President Larry Mayhew's staff approved the committee's guidelines as ***recommended*** design practice on July 19, 1979.) **Mike Mihalik, ext. 1533 (Walker Road).** **Etched Circuit Layout.** Establish etched circuit layout testability guidelines. **Ted Churchill, ext. 5666 (Beaverton).** **Logic Design.** Establish logic-design testability guidelines. **Alan Winslow, ext. 1931 (Walker Road).** **Signature Analysis.** Develop design guidelines for products intended to use signature analysis as a troubleshooting strategy. **Todd Paulus, ext. 7092 (Beaverton).** Coordinator for the Design for Testability subcommittees is **Steve Swerling, ext. 5324 (Beaverton).**

A large part of the Digital Product Coordination group's effort is coordination of committees whose goal is to produce a body of recommended design practice for product designers. Some of the current topics being handled are listed above together with the respective objectives and leadership. Feel free to contact *any* of them for more information.

Continued from page 19

ADDITIONAL SOFTWARE

A few extra lines of code can often alert the end user to a malfunctioning LSI chip before the failure causes any damage. For example, a small subroutine executed as part of the power-up sequence can poll key parts and subsystems, compare the status results with data stored in rom, and either validate proper operation or flag breakdown. Borrowed from large system technology, this self-test approach is functionally proper for small microprocessor-based systems.

Software timing aids are valuable tools in determining precise elapsed times for critical program areas such as interrupt servicing. "Hand-timed" analysis of program traces can be risky, particularly with the presence of memory wait states or the frequent occurrence of other interrupts. A real-time prototype analyzer, an option in some microprocessor design systems, is one of the most valuable tools for reducing risk in software timing. It functions under operator-initiated control to report elapsed time between breakpoints in milliseconds, microseconds, or the number of cycles executed in the clock circuit of the prototype under design. (See Figure 4.) □

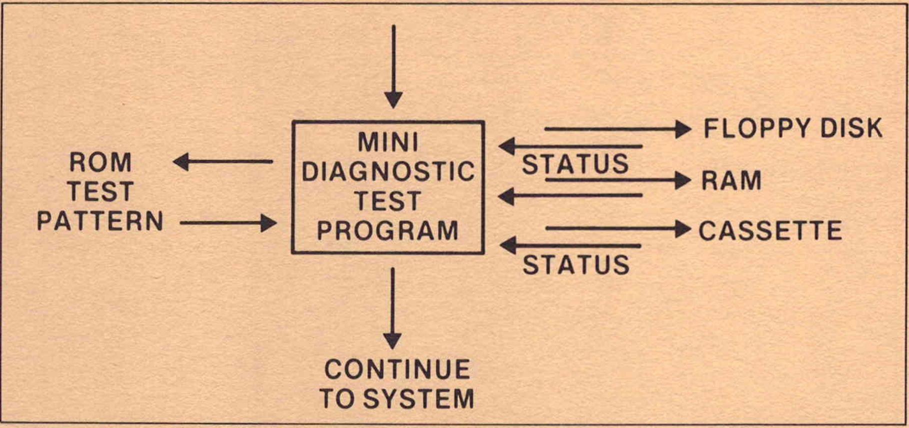
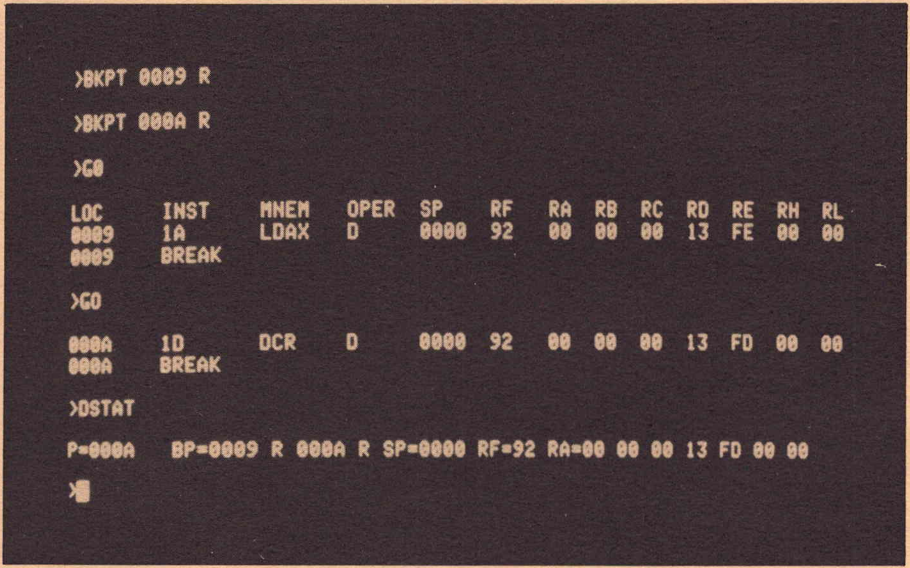


Figure 3. Addition of mini-diagnostic software prompts hardware subsystems after system power-up.



```
>BKPT 0009 R

>BKPT 000A R

>GO

LOC    INST   MNEM   OPER   SP    RF   RA  RB  RC  RD  RE  RH  RL
0009   1A     LDAX   D      0000  92   00  00  00  13  FE  00  00
0009   BREAK

>GO

000A   1D     DCR    D      0000  92   00  00  00  13  FD  00  00
000A   BREAK

>DSTAT

P=000A   BP=0009 R 000A R SP=0000 RF=92 RA=00 00 00 13 FD 00 00

>
```

Figure 4. Display of real-time prototype analyzer, part of the Tektronix 8002 Microprocessor Design Lab.

PUBLISHING OR PRESENTING A PAPER OUTSIDE OF TEK?

All papers and articles to be published or presented outside Tektronix must pass through Technical Communications Services (TCS) for confidentiality review. TCS helps Tektronix employees write, edit and present technical papers. Further, the department interfaces with Patents and Licensing to make sure that patent and copyright protection has been undertaken for all patentable and copyrightable material discussed in the paper or article.

For more information and for assistance in producing your paper, contact Eleanor McElwee, ext. 8924 (Merlo Road). □

TECHNOLOGY REPORT PAPER POLICY

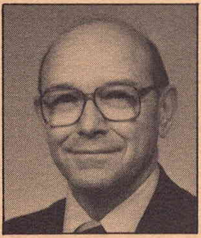
Technology Report and **Forum Reports** are printed on Exact Matte paper. We selected Exact Matte for its exceptional qualities and its low cost (15% less than the Howard Offset paper formerly used).

Sized (made water-resistant) with a particle coating, the paper surface allows photographs to be printed as clearly as those printed on more expensive, coated (slick) stock. Additional advantages are uniform texture and opacity, necessary qualities for printing charts, drawings, and technical equations.

Exact Matte paper is not recycled paper (recycled paper is too expensive) and is not WOWable, but it *can* be recycled. Send extra **Technology Report** and other non-WOWable paper, in quantity, to Material Evaluation at d.s. 02-001. Non-WOWable paper should be marked "Salvage." □

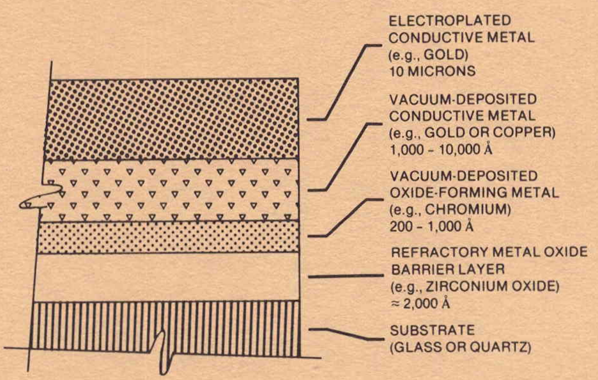
PATENT RECEIVED: No. 4,153,518

THIN-FILM BARRIER LAYER HELPS FABRICATE METALLIZED SUBSTRATES



Robert E. Holmes, Hybrid Circuits Engineering, ext. 5822 (Beaverton).

Robert R. Zimmerman, Thin-Film Lab (Hybrid Circuits Engineering), ext. 5813 (Beaverton).


ELECTROPLATED CONDUCTIVE METAL (e.g., GOLD) 10 MICRONS
VACUUM-DEPOSITED CONDUCTIVE METAL (e.g., GOLD OR COPPER) 1,000 - 10,000 Å
VACUUM-DEPOSITED OXIDE-FORMING METAL (e.g., CHROMIUM) 200 - 1,000 Å
REFRACTORY METAL OXIDE BARRIER LAYER (e.g., ZIRCONIUM OXIDE) ≈ 2,000 Å
SUBSTRATE (GLASS OR QUARTZ)

To make connections to active components in hybrid circuits, manufacturers must form metal conductors and pads on alumina, beryllia, glass, or fused-silica (quartz) substrate. Traditionally, manufacturers form such metal layers by first vacuum-depositing a thin layer of oxide-forming metal (such as chromium) which reacts with the substrate to form a reliable bond. The next step in production is vacuum-depositing gold, to the desired thickness, or other highly conductive metal, on the first metal layer. The chromium and gold are deposited in the same vacuum pumpdown cycle. This method requires an etching process to remove undesired portions of the metal, leaving only the conductors and pads. The etching is difficult to control, particularly in microcircuit applications where close tolerances are necessary in very thick metal films.

An alternative method is to vacuum-deposit a thin metal layer on the substrate, as before, but then electroplate the metal layer to the desired thickness. This process permits production of conductors and pads whose dimensions are within small tolerances, and is also simpler and less expensive. Tektronix uses a process known as "pattern plating" to define conductor geometries during the plating process. The plating is defined by a photoresist image that masks all areas except the desired conductors. In early attempts to use this method, however, the electroplating of conductor patterns degraded the adhesion of the thin metal film to glass and fused-silica substrates.

The method shown in the figure uses a thin-film barrier layer of zirconium oxide between the substrate and the vacuum-deposited metal layer to prevent reaction between them and subsequent loss of adhesion during electroplating. As a result, manufacturers can use standard thin-film processing techniques. For example, manufacturers can define, using a pattern mask, conductors and pads and electroplate them to the desired thickness. The mask permits tight control of the electroplated conductors' dimensions.

This zirconium-oxide method provides several benefits in the fabrication of high-resolution microwave integrated circuits. First, it provides for the close control of tolerances required for microcircuit applications. Second, it assures a reliable bond between the substrate and the thick metal conductors. Third, it is simple and inexpensive.

Manufacturers can also use this method to fabricate thin thermal printers on low-thermal-conductivity substrates using thin-film resistive elements and electroplated conductors. □

Technology Report

MAILING LIST COUPON

☐ ADD	**Name:** ____
☐ REMOVE	**Old Delivery Station:** ____
☐ CHANGE	**New Delivery Station:** ____
	Payroll Code: ____

(Required for the mailing list)

MAIL COUPON TO 53-077 **Allow four weeks for change**

technical standards

STANDARD IDENTIFIES MOUNTING-HOLE PATTERNS

The use of existing mounting-hole patterns for new components, when applicable, represents a definite cost saving for Tektronix.

Tektronix Standards 062-1732-00 through -08 describe the selection and use of mounting-hole patterns, and make it easy for designers to locate patterns for part-numbered components previously used.

Standard 062-1732-00 lists component part numbers in numerical order and identifies the corresponding detail numbers for mounting-hole patterns (a "detail" is a dimensioned hole pattern for a component). The standard also includes a computerized listing of mounting-detail numbers cross-referenced to applicable components. Standards 062-1732-01 through -08 list mounting-hole patterns according to the number of holes.

When a detail number exists, use it. When a design requires a new mounting-hole pattern, request a new detail from Technical Standards. Submit the component's part number, along with a sketch complete with dimensions and details. Technical Standards will then issue a new detail number for use on drawings. The drawing must include a reproduction of the detail, as explained in paragraph six of 062-1732-00.

For more information, call Pauline Whitmore (Technical Standards) on ext. 245 (Town Center). □

NEW STANDARDS

Drafting Standard, Datums, 062-3135-00. This standard explains the principle of referencing features of a part to an appropriate datum by means of a Datum Reference Frame.

Product Nomenclature Standard, New Product Introduction, 062-4323-00. This standard establishes procedures for assignment of Tektronix Product Nomenclature. It will form a part of the **New Product Introduction Guidebook,** but will also be issued separately.

Fabrication Standard, Checklist for Mold Designers and Mold Makers, Standard Mold Requirements, 062-1709-00.

For a copy of these or any other standards, call Technical Standards, ext. 241 (Town Center) or write d.s. 41-260. □

CONTRIBUTING TO TR

Do you have an article or paper to contribute or an announcement to make? Contact the editors on ext. 8929 (Merlo Road) or write to d.s. 53-077.

How long does it take to see an article appear in print? That is a function of many things (the completeness of the input, the review cycle, and the timeliness of the content). The *minimum* is six weeks for simple announcements and as much as 14 weeks for major technical articles.

The most important step for you, the contributor, is to put the message on paper so that we, as editors, will have something with which to work. Don't worry about organization, spelling, and grammar. We will take care of those when we put the article into shape for you.

Scratch Area

WANT TO REACH TR READERS?

A continuing feature of **Technology Report** is the *Scratch Area* column. This space is set aside for miscellaneous short notices such as new personnel introductions, calls for information, and calls for papers. To contribute to *Scratch Area,* send your input to the associate editor, Laura Lane, at d.s. 53-077 (ext. 8927-Merlo Road). □

COMPANY CONFIDENTIAL

S

58-121

PAUL E GRAY

TECHNOLOGY REPORT

mail label goes here