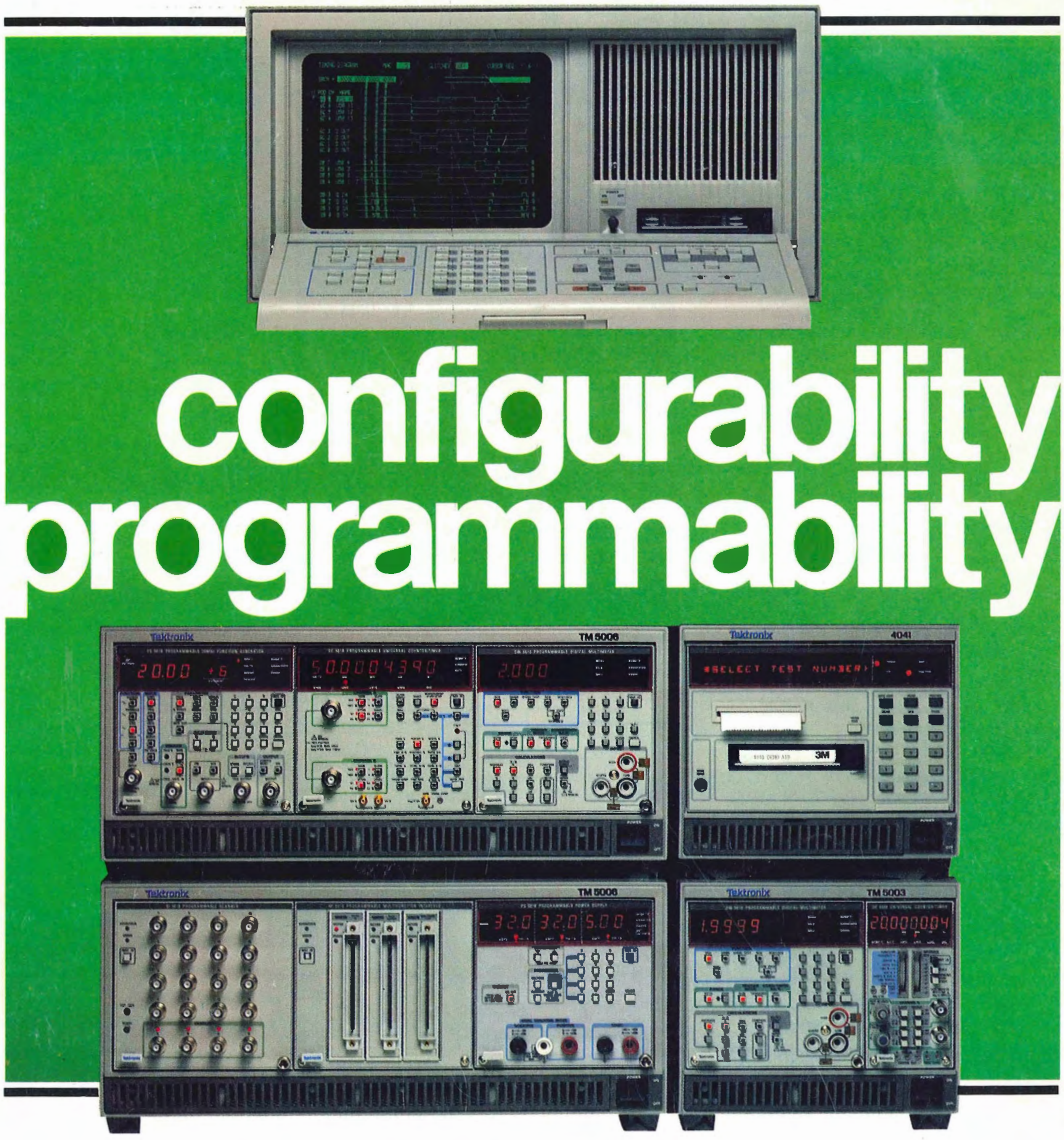
TECHNICAL INFORMATION
FOR TEKTRONIX CUSTOMERS

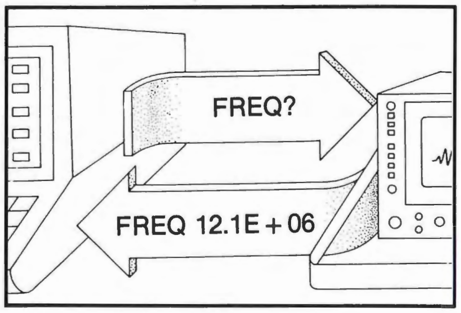
VOLUME 13 NUMBER 3
SEPTEMBER 1981

Tekscope



Tektronix®
COMMITTED TO EXCELLENCE

Tekscope CONTENTS



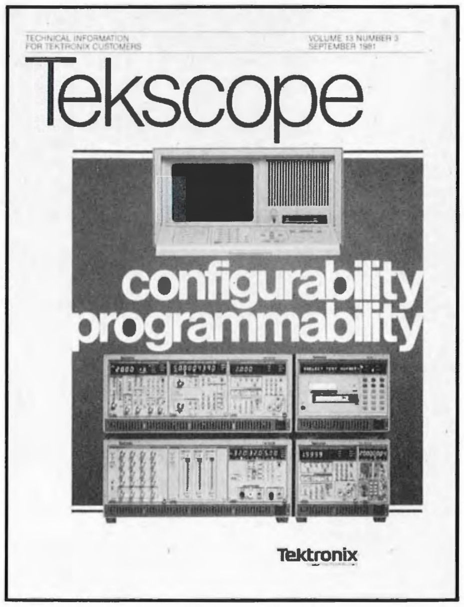
Customer information from Tektronix, Inc.
Beaverton, Oregon 97077

Tekscope is a quarterly publication of Tektronix, Inc. In it you will find articles covering the entire scope of Tektronix' products. Technical articles discuss what's new in circuit and component design, measurement capability, and measurement technique.

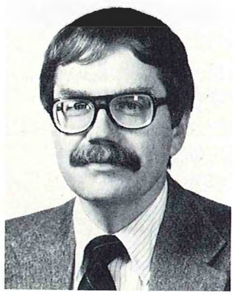
Editor: Gordon Allison
Graphic Designer: Michael Satterwhite

Cover

Configurability and programmability characterize both the DAS 9100 Digital Analysis System (top) and the TM 5000 Series general purpose instruments (bottom). Plug-in card modules allow you to configure the DAS for today's logic analysis needs and update the system as your needs change. Plug-in instrument modules provide the same flexibility for your automated test and measurement needs. Both systems are programmable via the IEEE-488 bus or an RS-232 interface and conform to the Tektronix Codes and Formats Standard.


TECHNICAL INFORMATION FOR TEKTRONIX CUSTOMERS
VOLUME 13 NUMBER 3 SEPTEMBER 1981
Tekscope
configurability
programmability
Tektronix

TM 5000—A New Family of IEEE-488 Programmable Instruments



Bob Metzler is Marketing Manager for the General Purpose Instruments business unit, which includes both TM 500 and TM 5000 Series products. Bob joined Tek in 1973 after several years of design work in the communications field. He received his B.S. in physics from the University of Louisville in 1960 and has an M.B.A. from the University of Portland. For recreation Bob enjoys backpacking with his family, listening to jazz and classical music, and dabbling in photography.

Many of today's "systems" resemble what we used to call conglomerations. Adoption of the IEEE-488 interface standard has made it easier to mechanically and electrically interconnect instruments from multiple vendors. However, other elements (not defined by IEEE-488) differ from instrument to instrument making it difficult to integrate the pieces into an effective operating system.

As discussions for a proposed new family of programmable instruments progressed, it became apparent that much of the compatibility problem centered around the codes and formats used to encode and transmit data within a system. Accordingly, a set of standard codes and formats for IEEE-488 instruments was formulated and adopted at Tektronix.

With the establishment of the codes and formats standard the task of developing a powerful, yet simple, IEEE-488 control software came into sharper focus. It seemed feasible to develop a set of software that would make instrument programming straightforward and easy to understand.

The goal for the overall program was to provide a versatile, modular, programmable instrumentation system that could be programmed and used by a diverse group of people—those involved in research, design, manufacturing, servicing, and other areas. The system should accommodate users having little or no programming experience as well as those capable of tackling complex measurement problems.

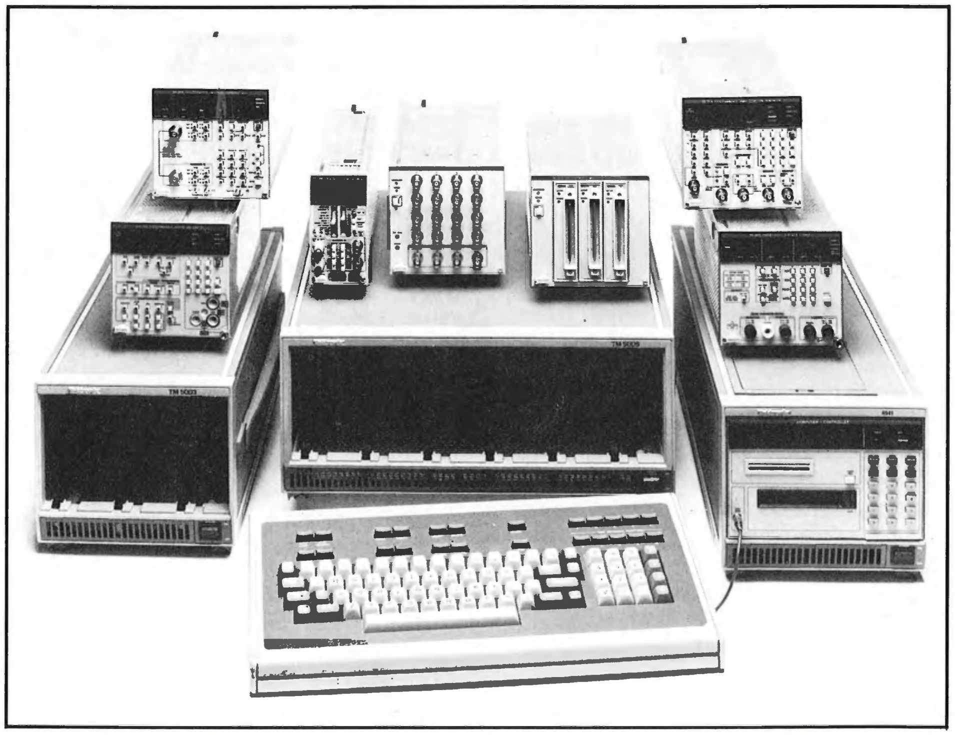


Fig. 1. The TM 5000 Series—a programmable test and measurement system that is easy to configure, easy to program, and occupies a minimum of space.

An instrument-oriented BASIC

Because of its wide use in industry, BASIC was selected as the basis for the new software. BASIC can be quickly grasped and applied to a wide variety of problems and tasks, and can be put to work with a minimum understanding of the language as a whole.

Several enhancements to BASIC adapt it to the instrumentation task. One extends the maximum length of variable names from two characters out to eight, thus increasing the programmer's ability to identify variables with their English equivalents, or close approximations. As an example, the variable "channel" could be written CH, CHA, CHAN, or CHANNEL.

Another enhancement permits variables to be defined exclusively within the confines of a subroutine. This feature lets programmers break down a complex program into subroutine modules that can be approached independently.

Other enhancements allow any program line to be identified with a label, and provide a flexible format for number entry.

Front-panel programming

For the instrument-oriented user, a "learn" mode provides, essentially, a "nonprogramming" method of setting up instruments on the bus. The user sets up the instrument from the front panel and by a single keystroke executes a command that causes the controller to query the instrument and store the control settings in memory. This is a fast, error-free way of programming an instrument.

A new controller

To fully implement the new software, a new instrument oriented controller—the 4041—was developed. The 4041 is based on the powerful 16-bit 68000 microprocessor and accommodates up to 160 kilobytes of user RAM. The standard configuration includes 32 kilobytes of RAM and a DC 100 mag-tape cartridge drive that provides up to 160 kilobytes of storage per tape. The DC 100 is used to store programs during program development, to load programs into system RAM during

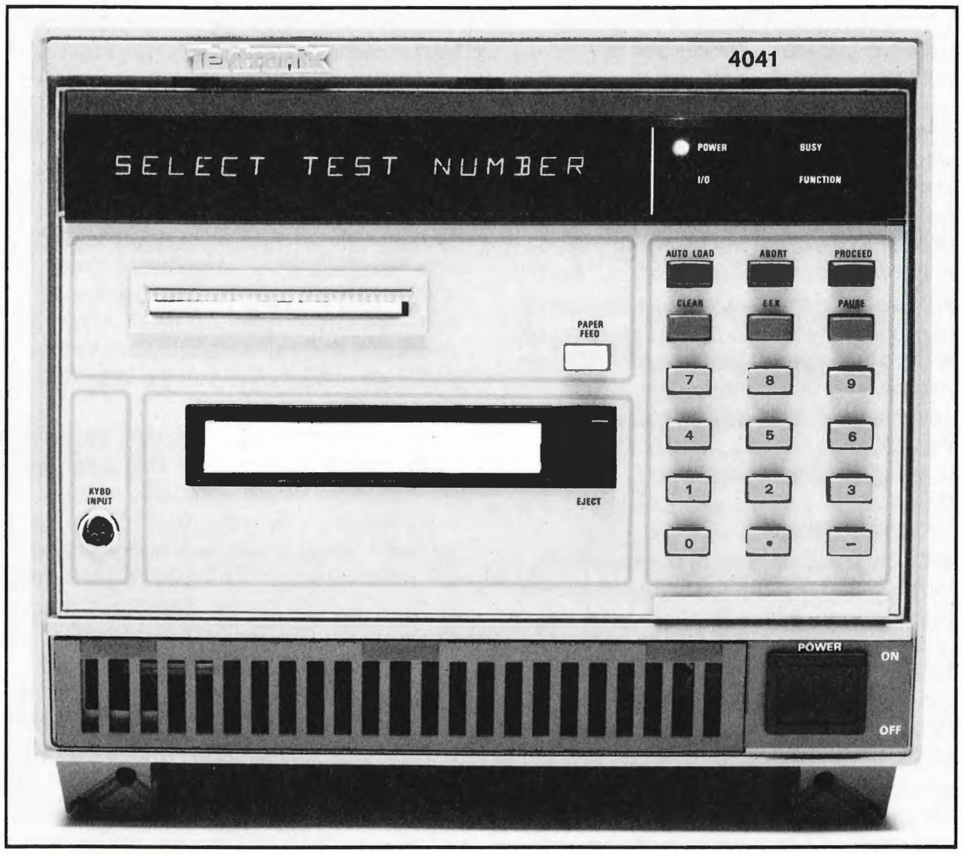


Fig. 2. The 4041 instrument-oriented controller is designed to serve needs in both engineering and production test. It can be used for program development, using an optional keyboard or any RS232-equipped terminal, or set up for "execute only" operation with programs loaded via mag-tape cartridges.

"execute only" operation, and for unattended, long-term data logging.

The 4041's front panel contains an 18-key operator keypad. Ten numeric keys, useful for entering numeric information requested by an INPUT statement, are user-definable and can be assigned subroutines by the applications program. A 20-character alphanumeric LED display communicates test procedures, operator prompts, and program results. Hardcopy is provided by a 20-column thermal strip printer.

The basic 4041 is configured for "execute only" operation. When using the 4041 for program development, an optional ROM set in a special carrier is inserted via the front panel. This firmware completes the operating system and permits users to generate and debug original programs. Source code can be written using an optional keyboard that plugs into the 4041 or any CRT terminal connected via the RS-232C port. A hardcopy printer can also be connected to the 4041 if an optional second serial interface has been installed. The 4041 is compatible in appearance and mechanical design with the mainframe of the new TM 5000 Series of programmable instruments.

A new family of programmables

While work was proceeding on the software and the controller, a parallel program was underway to develop a whole family of programmable instruments. Designated the TM 5000 Series, the family includes nine new products at introduction: a triple power supply, 135 MHz and 350 MHz universal counter/timers, 20 MHz function generator, 4-½ digit digital multimeter, 16-channel rf scanner, multi-function interface, and two power mainframes.

All of the instruments comply with the Tektronix Codes and Formats Standard and have individual IEEE-488 interface controllers. Each has its own microprocessor to perform processing locally and keep bus traffic and demands on the system controller at a minimum.

An important design decision was to make all of the functions and parameters settable from the front panel, as well as being programmable. This provides a simple, straightforward means of programming an instrument. Lighted controls and digital displays indicate the status of the instruments at all times (except for the DC 5009). A major operating convenience is front-panel display of an instrument's primary address, available at the touch of a button.

At power-up, each instrument performs a self-test diagnostic routine to verify its readiness for use. If no internal error is found, the instrument enters the local state with its current front-panel settings and default remote settings, and signals the controller it is ready for service. If an internal error is found, an error code is immediately displayed on the instrument's front panel.

TM 5000 Series mainframes

Two power mainframes: a three-compartment wide (TM 5003) and a six-compartment wide (TM 5006), house and provide power to the plug-in modules. Both TM 500 and TM 5000 plug-in modules can be plugged into either mainframe. This gives users the option to use both manual and programmable instruments in the same enclosure to optimize system performance and cost.

A key-and-slot arrangement on the plug-in interface connectors prevents inadvertent insertion of alien modules. This scheme also provides a convenient means of reserving compartments for a particular class of instrument. For example, the right-hand compartment of each mainframe is a high-power location. A PS 5010 Programmable Power Supply inserted in this position can supply twice as much power as from another location.

Separate 20-pin connectors in each compartment interface with the programmable modules. These connectors are paralleled and brought out to a single IEEE-488 connector at the rear of the mainframe.

A programmable triple power supply

The PS 5010 Programmable Power Supply is designed to furnish commonly-used positive, negative, and logic supply voltages, from a single unit. The 0 to +32 and 0 to −32 volt floating supplies (referenced to a common front-panel terminal) can each supply up to 750 milliamps. In a mainframe high-power slot, up to 1.6 amps at 15 volts is available. Programming increments are 10 millivolts from 0 to 10 volts and 100 millivolts from 10 to 32 volts. Current limiting is also programmable in 50-milliamp increments from 50 milliamps to 1.6 amps. The two supplies can be operated independently or set in a dual tracking mode.

The logic supply is ground-referenced and programmable over a range from 4.5 to 5.5 volts in 10 millivolt increments. Here, too, current limits can be set in increments of 100 milliamps over a 100 milliamp to 3-amp range.

The supplies have three operating states—voltage regulated, current limited, or undefined. Should a supply change from one state to another, for example, from voltage regulated to current limited, the PS 5010 will indicate the change on its digital display and send an interrupt to the system controller. The controller can then query the status of the supply and take appropriate action.

Unique function generator control

The FG 5010 Programmable Function Generator provides the usual capabilities plus some unique ones. For example, users can control waveform symmetry in one-percent increments from 10 to 90 percent, and can program phase from +90 degrees to −90 degrees in one-degree steps when operating in the trigger and gate modes. In the phase-locked mode, phase with reference to the locking signal can also be controlled with the same resolution and range.

Frequencies from 2 millihertz to 20 megahertz, amplitudes from 20 millivolts to 20 volts, and offsets from 0 to 7.5 volts are all programmable. An error-correction circuit maintains frequency accuracy within 0.1 percent. All of the programmable functions, parameters, operating modes, and so forth can be set up manually via front-panel pushbuttons. The FG 5010 can store up to ten setups, thereby reducing bus programming time during system operation.

Counter/Timer 1-picosecond resolution

The 350 megahertz DC 5010 Universal Timer/Counter also contributes some unique capabilities to the system. A proprietary, front-end, integrated circuit, which was designed and fabricated at Tektronix, makes possible one-picosecond averaging resolution. Single-shot time resolution is 3.125 nanoseconds.

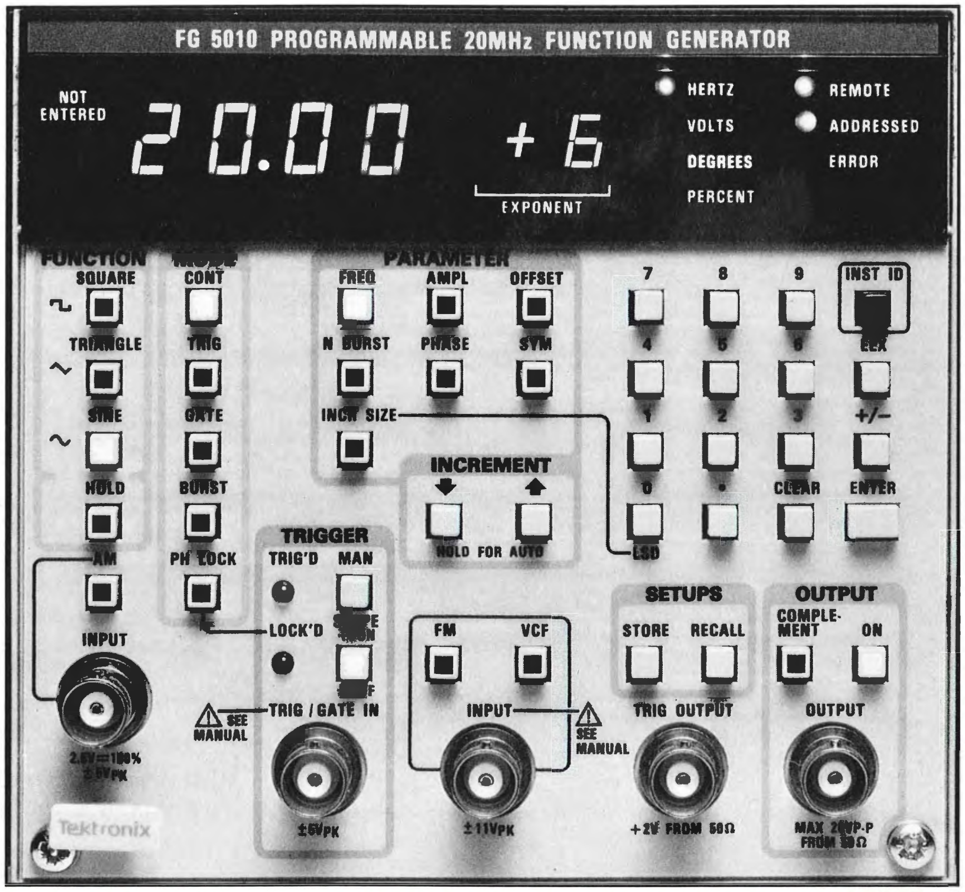


Fig. 3. The FG 5010 Programmable 20-MHz Function Generator can store 10 complete panel setups and has counted-burst and phase lock capabilities as well as programmable symmetry and phase.

Both the DC 5010 and the 135 megahertz DC 5009 feature a dual-input ratio architecture, with several modes for adding, subtracting, and calculating ratios of events on the two channels.

An autotrigger function measures the peak-to-peak input signal amplitude and sets the trigger level to the 50 percent point. The DC 5010 uses this same peak-to-peak information to calculate the 10 percent to 90 percent rise and fall times of the input signal. A null function subtracts an initial reading from subsequent readings, to automatically compensate for differences in cable lengths, and so forth.

All of the DC 5009 and DC 5010 functions can be controlled from the front panel or the IEEE-488 bus.

A smart 4-½ digit multimeter

The DM 5010 Programmable Digital Multimeter is a 0.015 percent instrument with extensive math capability. Besides measuring resistance and AC or DC volts, the DM 5010 calculates averages, handles offset and scaling, performs decibel conversions, has a nulling function, and makes comparison readings for a high-low-pass sorting mode. This extensive math capability makes the DM 5010 a valuable stand-alone instrument, and greatly reduces time on the bus when used in systems applications.

The DM 5010 measures up to 1000 volts DC, 700 volts true RMS AC, and 20 megohms of resistance. A user can step through the ranges or use autoranging in both manual and programming modes of operation. A dual sample rate provides 4-½ digit resolution at 3 measurements per second, or 3-½ digit resolution at 26 measurements per second. As with other TM 5000 instruments, the DM 5010 is fully programmable from the front panel or the IEEE-488 bus.

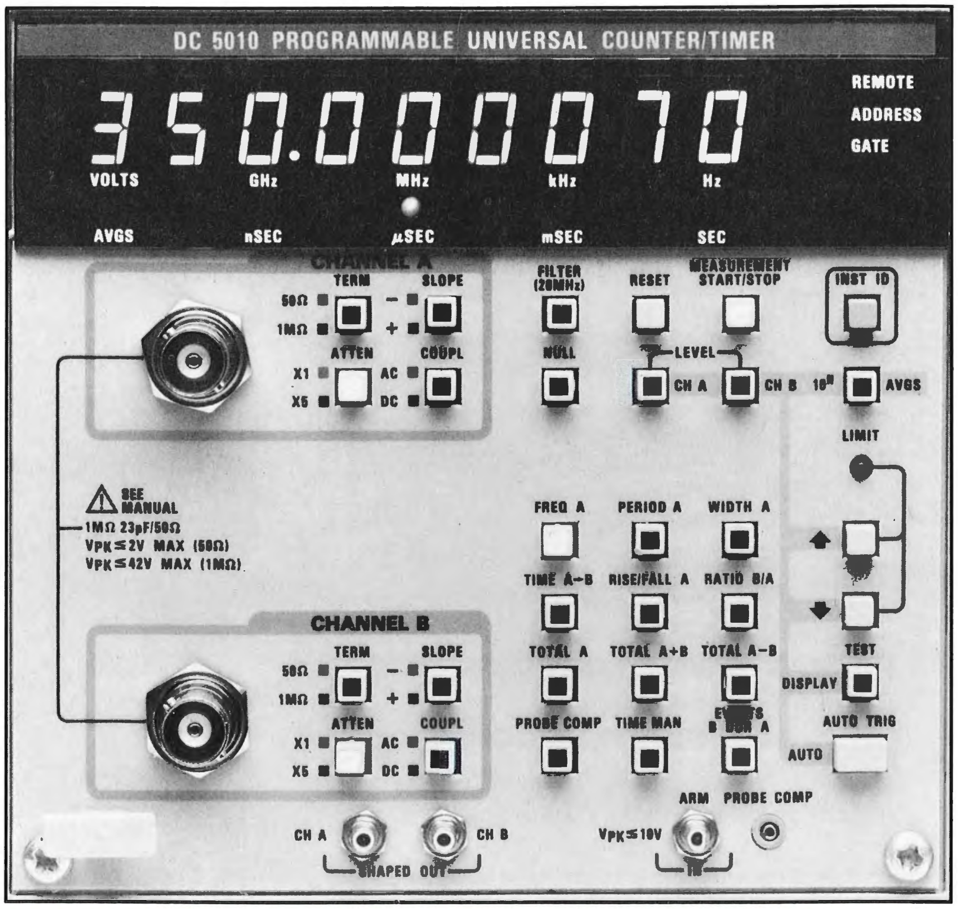


Fig. 4. The DC 5010 Programmable Universal Counter/Timer features reciprocal frequency, auto-trigger, RISE/FALL mode, and one-picosecond time interval averaging resolution.

Rounding out the system

An important part of any system activity is interfacing the test instruments to the device under test. Two products in the TM 5000 family—the MI 5010 Programmable Multifunction Interface and the SI 5010 R.F. Scanner—make the task relatively easy.

The MI 5010 and its companion MX 5010 Multifunction Interface Extender module each provide space for three plug-in cards. The purpose of the modules and their cards is to provide a programmable interface between the TM 5000 instruments and the device under test, and other non-IEEE-488 programmable instruments.

Three cards are available at introduction:

The 50M30 Programmable Digital I/O Card features 16 channels each of digital input and digital output. Both input and output are TTL compatible. Typical applications for this card include interfacing with relay drivers, switches or custom keyboards, BCD data from non-IEEE-488 instruments, and static binary words.

The 50M40 Programmable Relay Scanner Card serves as a low-frequency scanner accommodating up to 16 inputs or outputs. The card contains 16 mercury-wetted relays that can be configured 16-to-1, 8-to-1, or 4-to-1 by positioning jumpers. Typical uses include control of larger relays, scanning a multimeter input over several signal points, turning on lights or audible indicators, and so forth.

The 50M70 Programmable Development Card allows users to design their own special functions, such as digital-to-analog and analog-to-digital converters, special word recognizers, or any other function needed to complete an automated system, without concern for interfacing to the TM 5000 system. The rear portion of the card contains interface firmware, with the remainder of the card (about 20 square inches) reserved for the user's customized circuitry.

For switching high-frequency signals, the SI 5010 R.F. Scanner provides 16 wide-bandwidth channels that can be software-configured into a 16-to-1, two 8-to-1, or four 4-to-1 configurations. Optimum bandwidth is 350 MHz in the 4-to-1 configuration.

Both the SI 5010 and the MI 5010 have a real-time clock and buffer for sequence storage and execution, allowing the system controller to attend to other activities while switching commands are being implemented.

Summary

The TM 5000 Series Programmables are designed to extend the advantages of automated test systems to users previously denied access because of limited interfacing capability or programming expertise. A new instrument-oriented controller and extended-BASIC language provides versatility adequate for even the most sophisticated applications. And the modular design provides configurability and compactness unique in programmable systems.

Acknowledgements

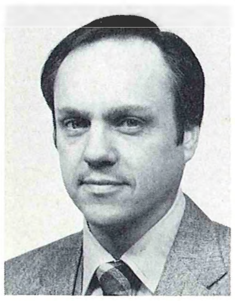
The TM 5000 Series consists of several products, each developed as an individual project. R. Michael Johnson is engineering manager and provides overall direction for the TM 5000 program. Bob Bretl serves as overall software manager, and Bud Deibele coordinates mechanical design.

Don Hall was project manager for the FG 5010 Function Generator, Bruce Hofer for the PS 5010 Power Supply, Ken Panck for the DMM and counters, and Dave Wert for the mainframes and MI 5010 Multifunction Interface and SI 5010 R.F. Scanner.

Dave Heinen was program manager for the 4041 Controller, with Richard Crall heading up the electrical design team and Tom Hamilton the software team. Charles Brown coordinated the mechanical design.

Our thanks to these and the many others who made substantial contributions to the TM 5000 and 4041 programs. ■

Configurable State-of-the-Art Logic Analyzer Gives Choice of Performance



Bill Allen is currently logic analyzer product marketing manager. In the seven years since joining Tektronix, Bill has been a software systems analyst for computer graphics software and product line manager for the 4010 Series of computer display terminals. Bill received his B.A. in math and physics from Willamette University in 1964, an M.B.A. from Portland State University in 1976, and is presently working on his M.S.E.E. at the University of Portland.

John Huber is product line manager for the DAS 9100. Since joining Tek in 1973, he has performed diverse functions from quality control of large-screen storage display monitors to on-site service of complex semiconductor test systems. John is presently completing work on his B.S.E.E. at the University of Portland.

Tektronix customers are well aware of the benefits plug-in modularity affords oscilloscope users. Now, these same benefits are available to logic analyzer users. Selectable, plug-in card modules and a powerful mainframe let you configure the new DAS 9100 digital analysis system to meet your specific needs. You can have synchronous or asynchronous data acquisition at speeds up to 330 MHz, timing resolution to 1.5 nanoseconds on 8 channels, and data widths from 16 channels at 330 MHz to 104 channels at 25 MHz.

And the DAS can be more than just a logic analyzer. Two plug-in pattern generation modules let you generate word widths up to 80 bits at rates to 25 MHz. Combining pattern generation and data acquisition in a single modular instrument provides greater performance versatility and operating ease than is possible with separate units.

The DAS mainframe houses up to eight plug-in card modules. Two slots are reserved for the controller and time base/trigger modules, with the remaining six slots available for configuring the DAS to your specific task.

For data acquisition, there is a choice of four different modules:

- The 91A32—32 channels and sampling rates to 25 MHz; well-suited for analyzing bus transactions.
- The 91A08—an eight-channel, 100-MHz unit with 5-ns glitch capturing capability.
- The 91A04—with four channels capable of 330-MHz acquisition and operation in a high-speed mode providing two channels of 1.5 nanosecond resolution.
- The 91AE04—a four channel, 330 MHz expander module.

Up to four acquisition modules can function in the DAS mainframe simultaneously. Memory depth is 2048 bits per channel with the 91A04 (4096 bits in the two-channel mode) and 512 bits per channel with the 91A08 and 91A32.

Two card modules are available for the pattern generator. The 91P16 basic unit provides 16 data output channels, two strobes, and a clock. An expander module, the 91P32, adds 32 data channels and four strobes. Three pattern-generator modules can function in the DAS simultaneously, to generate up to 80 channels of data and 10 programmable strobes at rates up to 25 MHz.

A common controller

The DAS uses a single Z80 microprocessor to control both data acquisition and pattern generation. You can start either function separately or both at the same time. The controller card contains 32 kilobytes of system RAM, with firmware occupying up to 128 kilobytes of ROM, part of which resides on the various modules. As the Z80 address bus is not open-ended, a way of mapping a module's firmware into the Z80 address space had to be implemented. This is accomplished by two 4-bit registers that are programmed to select the module and the particular ROM on the module to be addressed.

Communication between modules is via two busses: the controller bus—used for internal communications, and the high-speed bus, dedicated to control applications. The controller bus goes to all of the modules and carries address and data information. Faster signals, such as clocks, qualifiers, and word recognition travel the high-speed bus. Communication between the pattern generator and its extension modules is also via the high-speed bus.

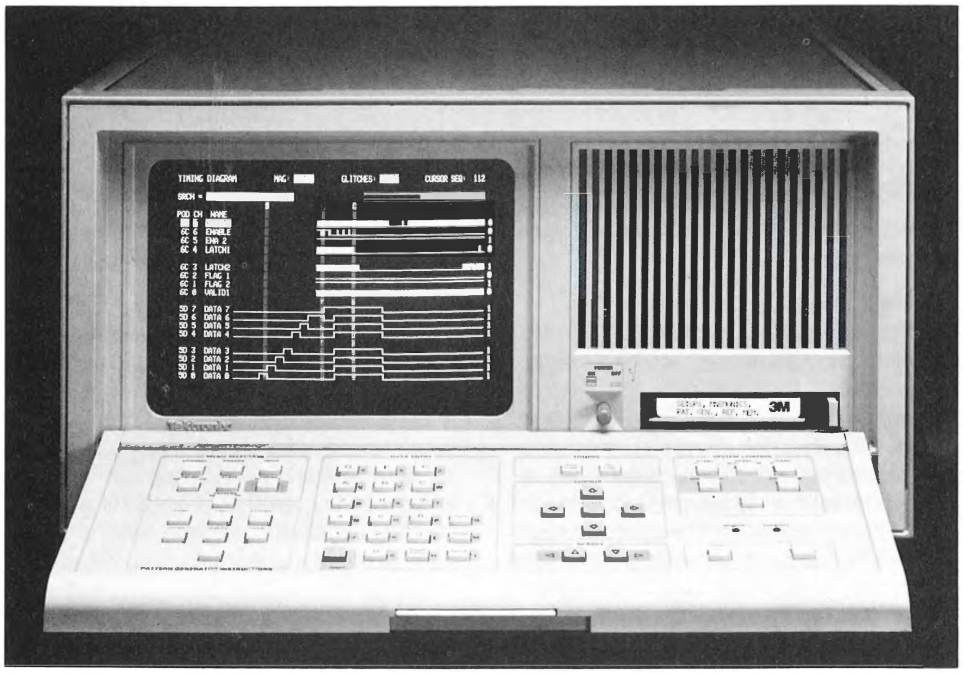


Fig. 1. The DAS 9100 Digital Analysis System provides a choice of performance via plug-in card modules. An easy-to-use keyboard, selectable menus, and a large, bright display facilitates operation.

Menus for manipulation

Extensive menus provide a high degree of versatility, yet make the DAS setup straight-

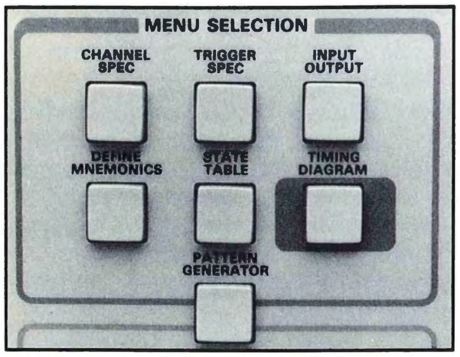
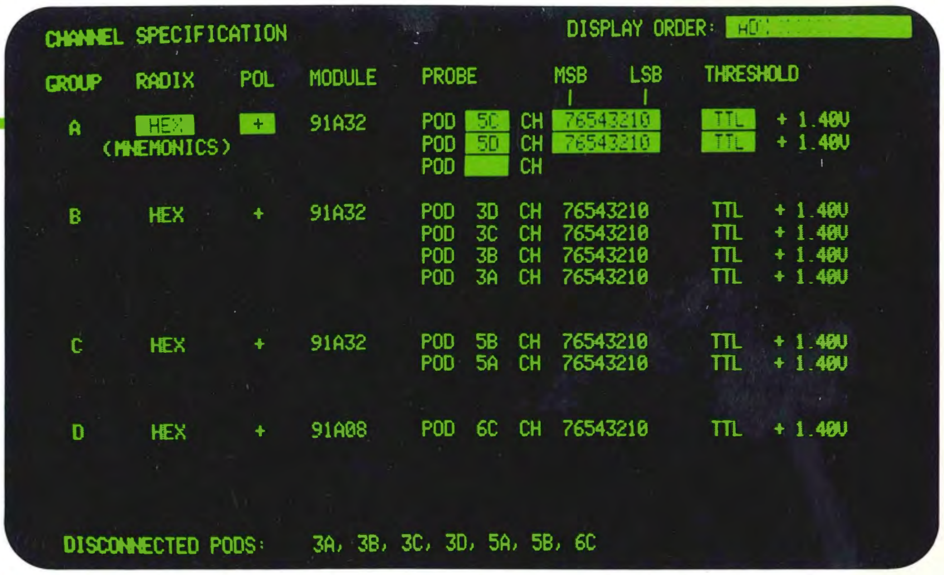


Fig. 2. The menu selection section of the DAS keyboard.

forward and easy to understand. Figure 2 shows the menu section of the DAS keyboard. There are seven general menus: five control the data acquisition function, one the pattern generator function, and one the input/output function. Several sub-menus can be selected from fields displayed in the general menus. At power up, all menus are automatically programmed with default values allowing immediate use of the DAS capabilities.

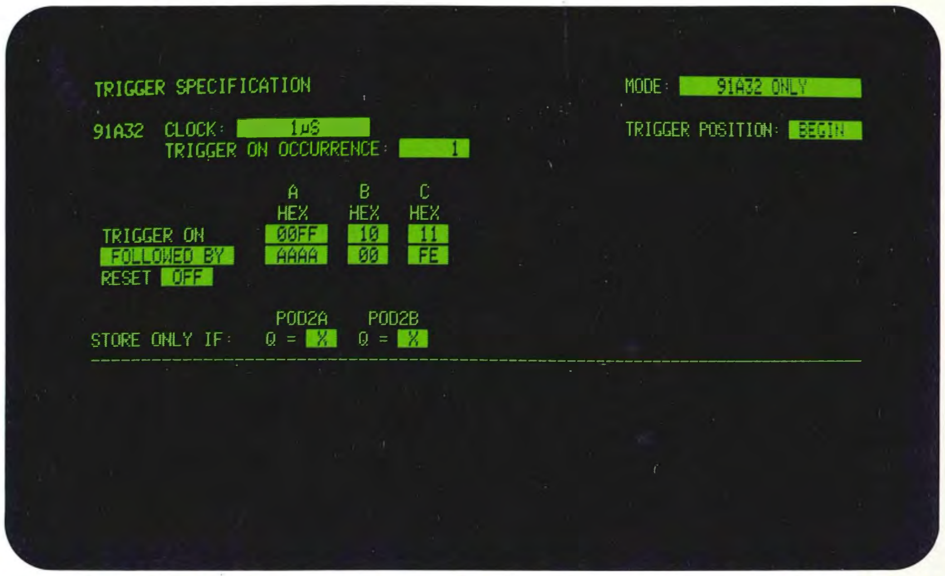
Also at power up, a configuration display lists the mainframe bus slots and identifies the installed modules. This display tells the user when power-up self-tests have been completed and indicates which modules pass or fail. If all elements pass, the screen displays a message instructing the user to select a channel specification, trigger specification, or pattern generation menu to program the desired function.

The Channel Specification menu (figure 3) controls the way in which incoming data is formatted and used by the trigger specification, state table, and mnemonic menus. However, it does not control how data is acquired. Once probes have been connected to the system under test, you can use this menu to change the display of data channels without manually reconnecting the probes or acquiring new data. Probe pods and channels can be grouped into logical display blocks, such as address, data, or control lines, and the order in which the groups will be displayed can be defined. A group may have from one to 96 channels. The Channel Specification menu is also used to select the radix and logic polarity of the displayed data, and for setting the probe threshold voltage for each group.



```
CHANNEL SPECIFICATION                                     DISPLAY ORDER: HD:

GROUP    RADIX     POL    MODULE    PROBE       MSB      LSB    THRESHOLD
                                                 |        |
  A       HEX       +     91A32     POD  5C  CH  76543210        TTL    + 1.40V
      (MNEMONICS)                   POD  5D  CH  76543210        TTL    + 1.40V
                                    POD      CH

  B       HEX       +     91A32     POD  3D  CH  76543210        TTL    + 1.40V
                                    POD  3C  CH  76543210        TTL    + 1.40V
                                    POD  3B  CH  76543210        TTL    + 1.40V
                                    POD  3A  CH  76543210        TTL    + 1.40V

  C       HEX       +     91A32     POD  5B  CH  76543210        TTL    + 1.40V
                                    POD  5A  CH  76543210        TTL    + 1.40V

  D       HEX       +     91A08     POD  6C  CH  76543210        TTL    + 1.40V


 DISCONNECTED PODS:   3A, 3B, 3C, 3D, 5A, 5B, 6C
```

Fig. 3. The Channel Specification menu allows you to group the data input channels and arrange their display order, set probe threshold voltages, and choose the display radix (hex, binary, or octal).



```
TRIGGER SPECIFICATION                                  MODE:  91A32 ONLY

91A32  CLOCK:      1µS                                 TRIGGER POSITION: BEGIN
       TRIGGER ON OCCURRENCE:     1

                         A       B      C
                        HEX     HEX    HEX
 TRIGGER ON             00FF    10     11
 FOLLOWED BY            AAAA    00     FE
 RESET  OFF

                       POD2A      POD2B
STORE ONLY IF:         Q = X      Q = X
--------------------------------------------------------------------------
```

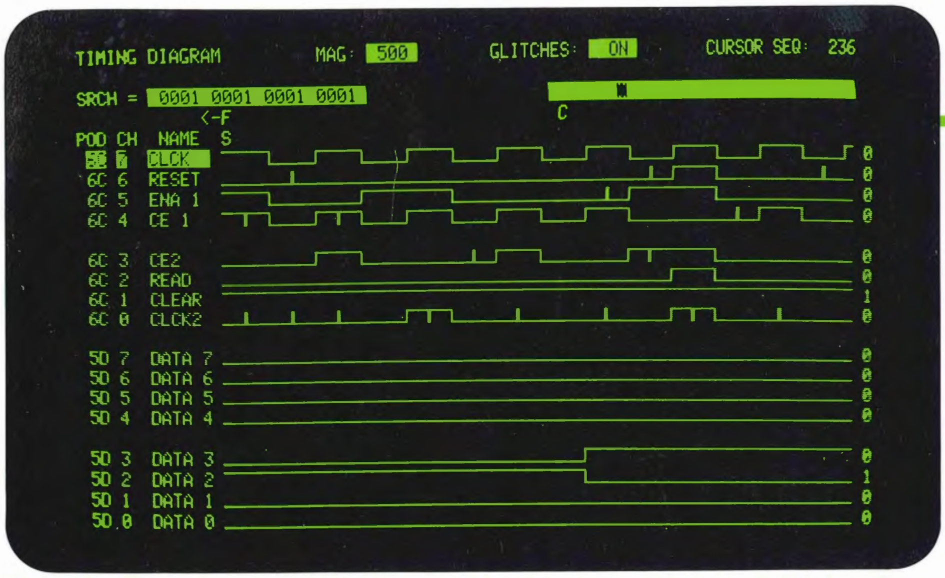
Fig. 4. The Trigger Specification menu is used to set up triggering parameters including clocks and qualifiers.

Versatile triggering provides split acquisition

The DAS provides a wide choice of trigger setups. Each DAS module has its own parameters, which are selected using the Trigger Specification menu (figure 4). With the 32-channel acquisition module installed, three-level sequential triggering (with various logical combinations), multiple trigger occurrences, arming, and delay times can be selected. Two clock qualifiers also can be set. Installation of a 91A08 module adds a 5-nanosecond glitch triggering capability and the trigger-arming capability when used with the 91A32 module.

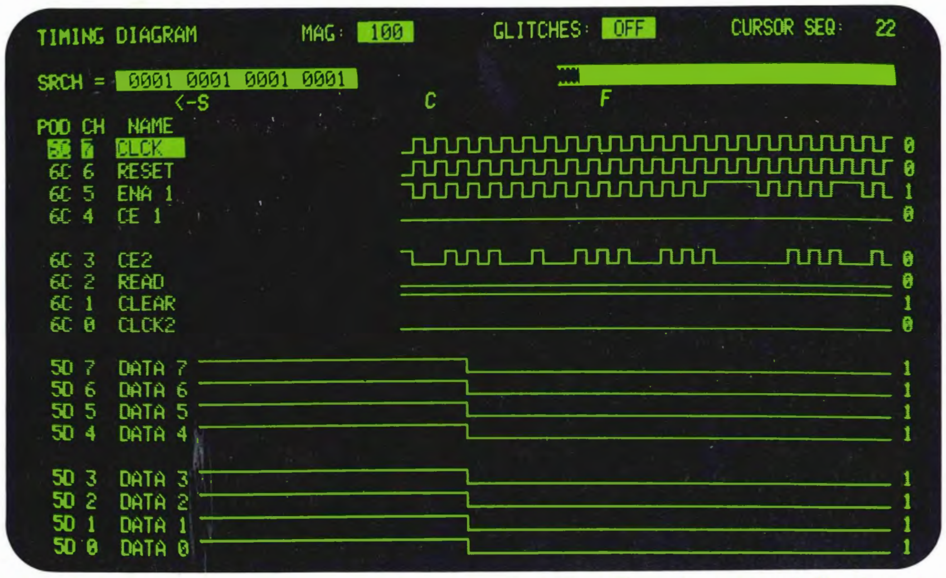
The DAS also can acquire and display information from multiplexed microprocessor busses. With two 91A32 modules installed, a trigger specification sub-menu allows you to configure the DAS for external-split clock operation. By adding modules, up to three separate external clocks may be tracked, in turn, arming another module for asynchronous acquisition of high-speed data.

As a timing analyzer, the DAS provides state-of-the-art performance with resolution selectable from 40 nanoseconds to 1.5 nanoseconds. The timing display is enabled by selecting the Timing Diagram menu. A typical timing display is shown in figure 5. The display includes logic events stored in the acquisition and glitch memo-



```
TIMING DIAGRAM        MAG: 500      GLITCHES: ON      CURSOR SEQ: 236
SRCH = 0001 0001 0001 0001
        <-F                          C
POD CH  NAME   S
5C 7    CLCK                                          0
6C 6    RESET                                         0
6C 5    ENA 1                                         0
6C 4    CE 1                                          0

6C 3    CE2                                           0
6C 2    READ                                          0
6C 1    CLEAR                                         1
6C 0    CLCK2                                         0

5D 7    DATA 7                                        0
5D 6    DATA 6                                        0
5D 5    DATA 5                                        0
5D 4    DATA 4                                        0

5D 3    DATA 3                                        0
5D 2    DATA 2                                        1
5D 1    DATA 1                                        0
5D 0    DATA 0                                        0
```

Fig. 5. The timing diagram provides glitch viewing and magnification up to 10,000 times for making high-resolution measurements. The portion of the timing window being displayed is indicated by the dark square in the horizontal green bar at upper right.



```
TIMING DIAGRAM        MAG: 100      GLITCHES: OFF      CURSOR SEQ: 22
SRCH = 0001 0001 0001 0001
        <-S                     C          F
POD CH  NAME
5C 7    CLCK                                          0
6C 6    RESET                                         0
6C 5    ENA 1                                         1
6C 4    CE 1                                          0

6C 3    CE2                                           0
6C 2    READ                                          0
6C 1    CLEAR                                         1
6C 0    CLCK2                                         0

5D 7    DATA 7                                        1
5D 6    DATA 6                                        1
5D 5    DATA 5                                        1
5D 4    DATA 4                                        1

5D 3    DATA 3                                        1
5D 2    DATA 2                                        1
5D 1    DATA 1                                        1
5D 0    DATA 0                                        1
```

Fig. 6. This timing diagram shows simultaneous acquisition of synchronous data (low-speed acquisition) and asynchronous data (high-speed acquisition), using the trigger arming mode. The low and high-speed displays are displayed in correct time relationship.

ries, and several other elements to help you in analyzing the data.

When using multiple 330-MHz data acquisition modules, the transition time down the backplane between modules in the mainframe can be an appreciable part of a sample interval. Provision is made to automatically deskew, or balance out, the data paths in each high-speed module to assure correct time relationships. Deskewing is of particular importance in the 1.5 nanosecond resolution mode.

The DAS even allows split acquisition of data, in which slow and fast acquisition modules operate as linked logic analyzers. This operation is called the "arms mode."

Although many analyzers have an arming capability, there is an important difference in the DAS arming mode—both slow and fast data are displayed simultaneously and in correct time relationship.

To accomplish coincidence of the two data displays in the arms mode, the fast data module samples the slow data's clock and stores the result in a clock array. The DAS also records the slow and fast modules' trigger points, and the number of fast clocks occurring between the two triggers. Using these bits of information, the DAS firmware calculates what the relative on-screen positions of the data should be and displays them accordingly. The positions of the cursor and the fast and slow triggers are also displayed.

Of particular importance in split acquisition is the MAG (magnification) field, which allows you to increase the display resolution. In default, no magnification is used and the entire memory content of each channel trace is shown. You can select magnifications of up to 10,000 times in a 1-2-5 sequence (see figure 6). A memory window in the upper-right section of the screen shows the portion of memory being displayed.

When a 91A08 acquisition module is in use, glitches are always acquired. You can quickly determine whether a displayed pulse is data or a glitch by setting the GLITCH field to OFF.

The SEARCH field in the menu allows you to do two types of searches. You can search for the next change in data values (from the memory cursor location) or you can search for a specific data word. When data has been acquired using a fast internal clock rate, the search for next change of data mode allows you to quickly move to the next area of interest without continual scrolling. The logic levels present at the cursor position are displayed in binary at the right-hand edge of the screen.

The POD and CH (channel) fields at the left of the screen identify which data channels are being displayed. You can use these fields to display channel traces in any order you choose and can label each trace with a name consisting of up to six characters. The trigger cursor indicates the position of the trigger word on-screen. Both fast and slow triggers are indicated in split acquisition timing displays.

Mnemonics enhance state table displays

The Channel Specification menu provides great flexibility in formatting data for the state table display. Once data is acquired, it can be moved around easily and grouped for fast, efficient analysis. The State Table menu (figure 7) gives you a choice of displaying data residing in acquisition memory, or displaying data from reference memory separately or simultaneously with acquisition memory. You can also choose to mask out columns of data so that they are ignored during memory comparisons, or select a portion of the data in reference memory, for comparison.

The state table display can be enhanced by use of the Define Mnemonics menu (figure 8). With this menu you can identify events with a word and more easily follow data transactions acquired by the DAS. Mnemonics can be linked to individual probe inputs for easy separation of data. Up to 256 mnemonics can be assigned.

When the Define Mnemonic menu is selected, the menu "reads" the Channel Specification menu to find out how the channels are grouped together. It then provides a mnemonic table for each valid group. All 256 mnemonic values can be entered under one table or divided into as many as 16 separate groups. The group table can be used to disassemble data displayed in the state table.

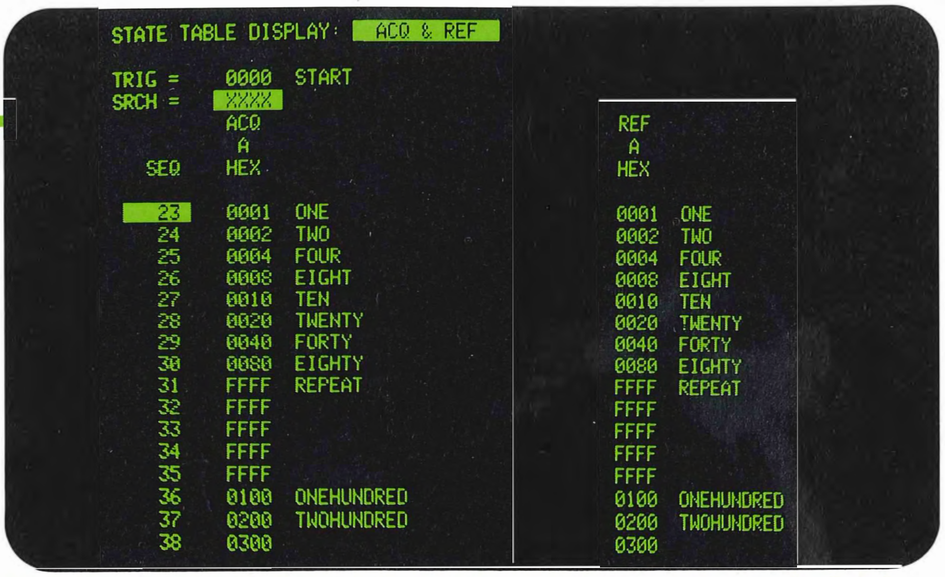
A trailing words field lets you specify whether a given mnemonic applies to more than one data value. Many assembly languages use a series of data words for one instruction. You may want to specify a mnemonic on the first word and disable disassembly on the remaining words. The trailing words feature gives you this capability for up to nine trailing words. In addition, you can use mnemonics for labeling calls to subroutines or for defining your own custom labels.

The pattern generation function

In the introductory paragraphs of this article we discussed briefly the pattern generation capabilities of the DAS—word widths up to 80 channels, clock rates to 25 MHz, and up to 10 programmable strobes. Now let's consider this function in greater detail.

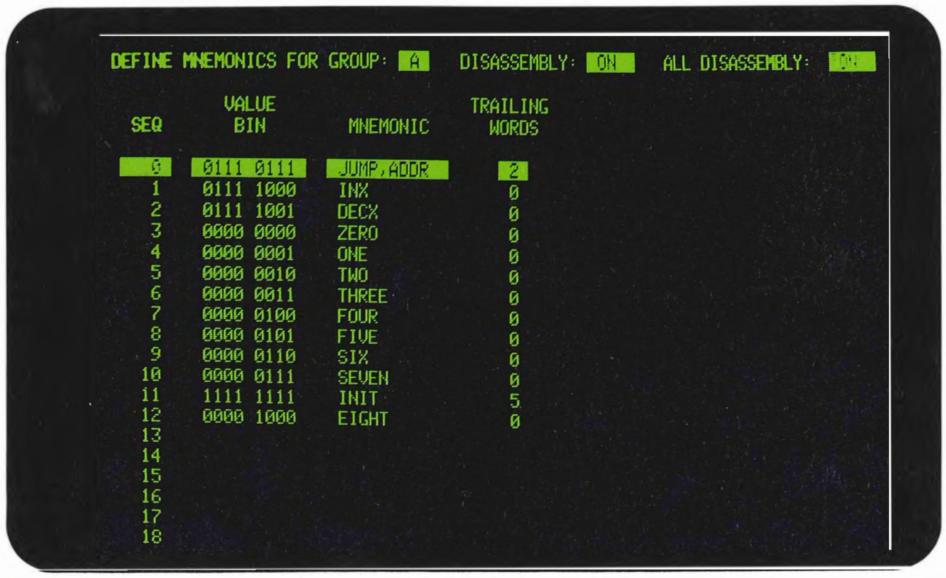
The basic pattern generator module provides 16 channels of data output, plus two strobe outputs and a clock output. An extension module adds 32 data outputs and four more strobes. The basic module and two expansion modules can be plugged into the DAS to provide a total of 80 data channels and 10 strobe outputs. Up to 254 different words can be defined in a single program.

Like the logic analyzer function, the pattern generator is menu driven. Two submenus direct input from the keyboard to set up the pattern generator for the specific application. The Program sub-menu (figure 9) allows you to specify clock source and rate, response to input signals from the system under test, and to program the data output. Seven instruction keys (figure 10) are used with this menu. You can:



```
STATE TABLE DISPLAY: ACQ & REF

TRIG =    0000  START
SRCH =    XXXX
          ACQ                                REF
           A                                  A
  SEQ     HEX                                HEX

   23     0001  ONE                          0001  ONE
   24     0002  TWO                          0002  TWO
   25     0004  FOUR                         0004  FOUR
   26     0008  EIGHT                        0008  EIGHT
   27     0010  TEN                          0010  TEN
   28     0020  TWENTY                       0020  TWENTY
   29     0040  FORTY                        0040  FORTY
   30     0080  EIGHTY                       0080  EIGHTY
   31     FFFF  REPEAT                       FFFF  REPEAT
   32     FFFF                               FFFF
   33     FFFF                               FFFF
   34     FFFF                               FFFF
   35     FFFF                               FFFF
   36     0100  ONEHUNDRED                   0100  ONEHUNDRED
   37     0200  TWOHUNDRED                   0200  TWOHUNDRED
   38     0300                               0300
```

Fig. 7. Using a state table display, both acquisition and reference memory can be viewed simultaneously for easy comparison of newly acquired data with known data.



```
DEFINE MNEMONICS FOR GROUP: A   DISASSEMBLY: ON   ALL DISASSEMBLY: ON

            VALUE                     TRAILING
  SEQ        BIN        MNEMONIC       WORDS

   0      0111 0111    JUMP,ADDR         2
   1      0111 1000    INX               0
   2      0111 1001    DECX              0
   3      0000 0000    ZERO              0
   4      0000 0001    ONE               0
   5      0000 0010    TWO               0
   6      0000 0011    THREE             0
   7      0000 0100    FOUR              0
   8      0000 0101    FIVE              0
   9      0000 0110    SIX               0
  10      0000 0111    SEVEN             0
  11      1111 1111    INIT              5
  12      0000 1000    EIGHT             0
  13
  14
  15
  16
  17
  18
```

Fig. 8. The Define Mnemonics menu allows you to label numerous values with a custom word of up to 10 characters. The trailing-words feature allows you to apply a given mnemonic to more than one data value.

- Specify each word in the pattern generation sequence
- Loop to labeled program lines
- Call labeled subroutines and return from them
- Program incremental counting
- Repeat a word n number of times
- Hold the same data output for n clock cycles (with clock output disabled)
- Halt data output
- And select the number of active strobes.

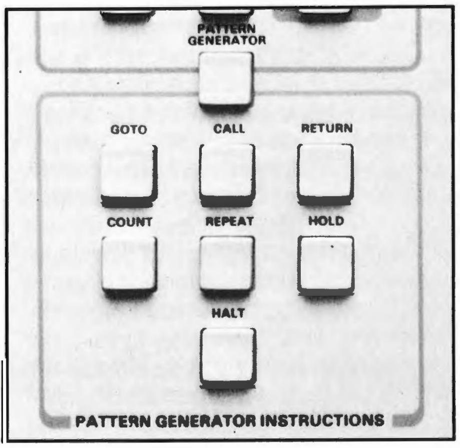
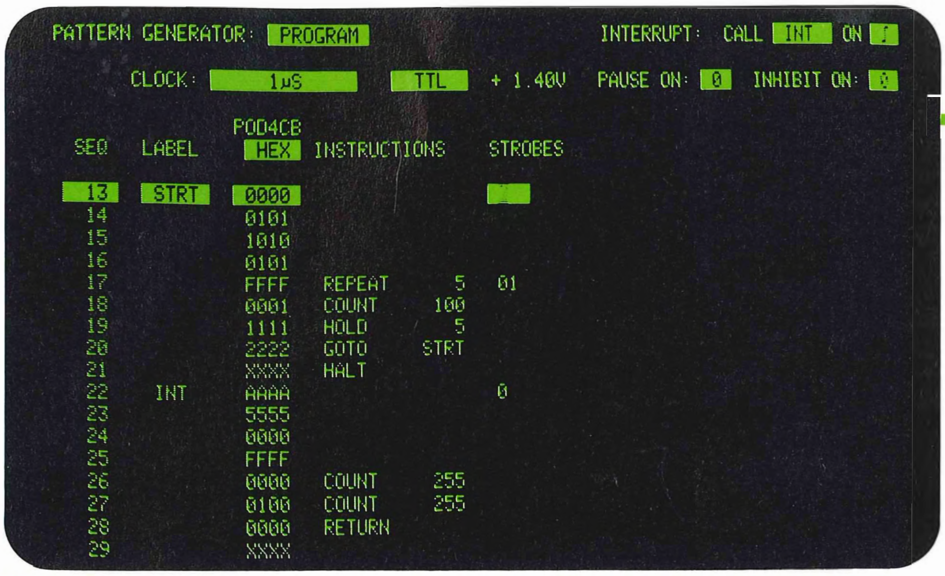
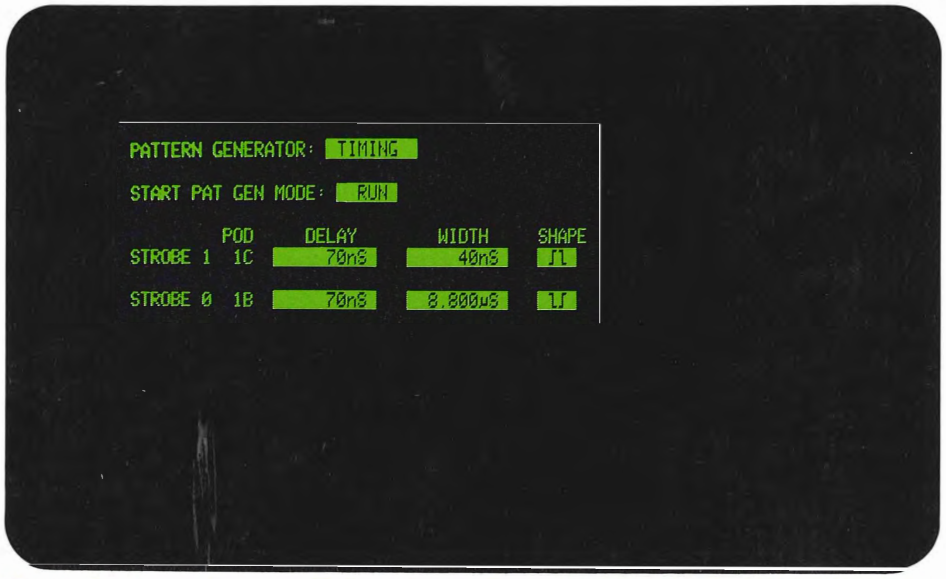


Fig. 10. The pattern generator instructions keyboard simplifies entering instructions in the program.



```
PATTERN GENERATOR: PROGRAM                        INTERRUPT:  CALL INT  ON ∫

         CLOCK:      1μS        TTL   + 1.40V   PAUSE ON: 0   INHIBIT ON: 0

                 POD4CB
 SEQ    LABEL     HEX   INSTRUCTIONS      STROBES

  13    STRT     0000
  14             0101
  15             1010
  16             0101
  17             FFFF   REPEAT      5      01
  18             0001   COUNT     100
  19             1111   HOLD        5
  20             2222   GOTO     STRT
  21             XXXX   HALT
  22     INT     AAAA                        0
  23             5555
  24             0000
  25             FFFF
  26             0000   COUNT     255
  27             0100   COUNT     255
  28             0000   RETURN
  29             XXXX
```

Fig. 9. The Pattern Generator Program sub-menu is used to set up the pattern generator's output program and select the programmed strobes.



```
PATTERN GENERATOR: TIMING

START PAT GEN MODE: RUN

              POD     DELAY        WIDTH     SHAPE
STROBE 1      1C       70nS         40nS      ⎍
STROBE 0      1B       70nS      8.800μS      ⊔
```

Fig. 11. The Pattern Generator Timing sub-menu is used to define the characteristics of the programmable strobes and the pattern generator's start mode (either run or step).

The pattern generation Timing menu (figure 11) provides control of two pattern generation functions: the mode in which data will be output—continuous or step by step, and the characteristics of each strobe output. Three characteristics can be defined—delay of the strobe output with respect to the output clock edge, duration of the strobe, and the polarity of the strobe. The strobe outputs can be programmed and combined in a manner to simulate bus management activity and thus control the input of a programmed pattern sequence to a prototype.

Interaction with the prototype is enabled by programming the pattern generator to respond in specific ways to active inputs on the DAS's pause, inhibit, and interrupt lines. An active "pause" signal from the prototype will cause the pattern generator to stop at its most recent step in programmed data output until the prototype releases the "pause." The pattern generator will then resume output at the next step in the pattern program. If the prototype initiates an "inhibit" signal, the pattern generator's data output lines go tri-state, effectively disconnecting them from the bus. If the prototype asserts the "interrupt" line, the pattern program will jump to a specified subroutine, execute it, and then return to the main pattern program flow. It is possible to have subroutines nested inside the interrupt routine. The system permits up to 16 levels of nesting.

Interfacing the pattern generator

Two different pattern generator probes were developed to provide interfacing with the various logic families. The TTL/MOS version has active pull-up and pull-down and accommodates large swings and relatively slow rise and fall times.

Each probe provides ten output channels: eight data, one strobe, and one clock. Additional lines connect to voltage rails and establish the high- and low-level thresholds.

The external clock, and pause, inhibit, and interrupt signals are input to the pattern generator via a clock probe plugged into the trigger/timebase module. The same probe provides a master-external-clock input and two additional external-clock lines for external-split-clock operation of the 32-channel data acquisition module.

DAS applications expanded

Several optional features allow you to apply the DAS to applications outside the usual design realm. The DC 100 Magnetic Tape Drive Option provides over 160 kilobytes of permanent storage that can be used for saving test patterns used in designing and troubleshooting a product. These saved patterns are quite useful in testing and evaluating a product in production. Instrument setups and mnemonic tables also can be stored and used to automatically restore the DAS to a particular setup.

Another option provides the DAS with RS-232, IEEE-488 GPIB, and hard copy unit interfaces. The RS-232 and IEEE-488 interfaces allow you to operate the DAS from a controller in networked or automated test applications. The remote programming language supports all operations which can be performed from the instrument keyboard and some additional commands that are useful for automated test applications.

The RS-232 interface also allows one DAS to be linked with another in a master/slave arrangement for remote operation. The master DAS serves as the controller for setting up and operating the remote slave DAS, and does not have to have the same complement of modules as the slave.

An optional composite video output allows documentation of test results and operating parameters through the use of a hard copy unit such as the Tektronix 4612 or 4632. This port also may be used for connecting to a remote monitor for group viewing.

Self-test diagnostics

When an instrument performs as many functions as the DAS, it is important to have self-test at power up. Each time the DAS is powered up, internal diagnostic tests automatically check out the major mainframe components and operating firmware.

During the first phase of the self-test, the DAS tests the major blocks of RAM and ROM and initializes I/O ports. After initialization is complete, a screen display (figure 12) lists the installed modules, which slot they reside in, and whether each passes or fails the self-test. Procedural options are listed at the bottom of the display. An extended diagnostic menu can be called up to run tests that fully check out any module or the entire system.

To allow you to check out the probes used with the DAS, a diagnostic lead set is provided that facilitates connection between a pattern generation probe and an acquisition probe. A preset walking-ones pattern allows you to test the complete path from the pattern generator, through the pattern generator and acquisition probes, to the acquisition module. This setup is also a convenient vehicle for learning how to operate the instrument.

Summary

The DAS 9100 is a unique combination of state-of-the-art logic analyzer and pattern generator. It is designed to allow you to efficiently simulate, stimulate, and analyze the response of digital circuitry. Modular design lets you configure the DAS for today's tasks and expand it to meet future needs. It is easy to operate, yet versatile enough to handle even the most demanding application.

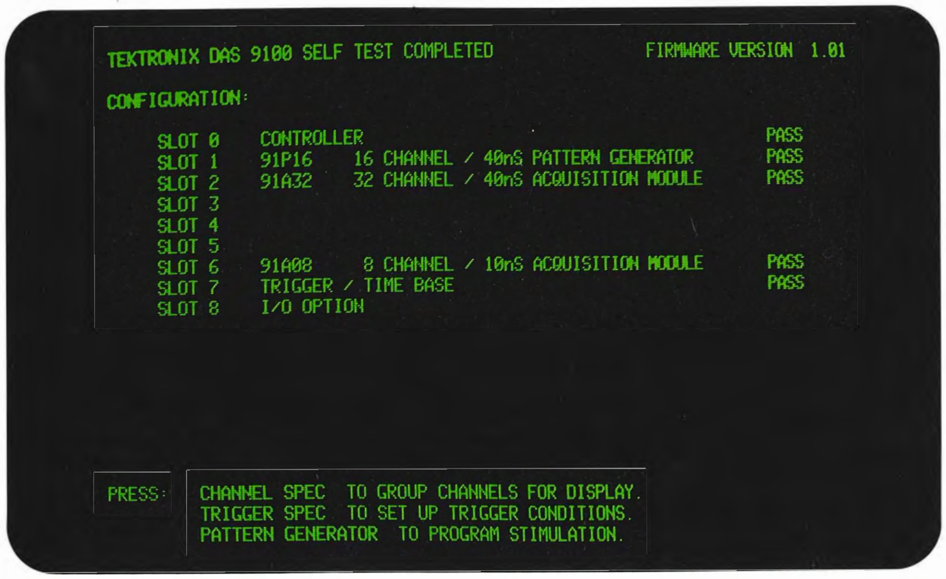
Acknowledgements

Steve Palmquist headed up the DAS design team, which included analog and digital circuit designers, software designers, and mechanical designers. Dave Chapman designed the controller and trigger/time base modules, the 91A08, and the I/O card; Steve Palmquist designed the 91A32 and the 91P16 and 91P32 pattern generator modules; and Glenn Gombert designed the 91A04 high-speed module. Robin Larson did the analog design including probes and power supplies.

Gerd Hoeren was the software project leader, ably assisted by Mark Pettet, who designed the interpreter and the trigger and time base menus. Steve Sutton designed the state table and pattern generator menus; Dennis Lamb designed the Define Mnemonics and I/O menus; and Jerrs Sewell defined and designed the remote programming capability (IEEE-488 and RS-232).

Ben Demise did an outstanding mechanical design job fitting the DAS into Tek's new standard mechanical package.

Many others were involved in prototype construction, evaluation, and putting the DAS into production. Our thanks to all for their valuable contributions to the DAS project. ■



```
TEKTRONIX DAS 9100 SELF TEST COMPLETED                FIRMWARE VERSION  1.01

CONFIGURATION:

   SLOT 0   CONTROLLER                                          PASS
   SLOT 1   91P16    16 CHANNEL / 40nS PATTERN GENERATOR        PASS
   SLOT 2   91A32    32 CHANNEL / 40nS ACQUISITION MODULE       PASS
   SLOT 3
   SLOT 4
   SLOT 5
   SLOT 6   91A08     8 CHANNEL / 10nS ACQUISITION MODULE       PASS
   SLOT 7   TRIGGER / TIME BASE                                 PASS
   SLOT 8   I/O OPTION

PRESS:  CHANNEL SPEC  TO GROUP CHANNELS FOR DISPLAY.
        TRIGGER SPEC  TO SET UP TRIGGER CONDITIONS.
        PATTERN GENERATOR  TO PROGRAM STIMULATION.
```

Fig. 12. The Configuration menu is automatically displayed at power-up. It lists the installed modules, the firmware version resident in the DAS, and whether each module passes or fails the power-up self-tests.

Codes and Formats Standard Adds Compatibility and Capability to IEEE-488 Instruments

Bob Cram is manager of the Automated Instrument Compatibility Evaluation group. He started with Tek in 1973 and joined the evaluation group following five years as Metrologist with the Electrical Standards group. While a part of the Standards group he did extensive development work on automated calibration procedures. Bob received his BA in physics from Linfield College in 1967 and performed graduate work at the University of Colorado in Boulder. His leasure time activities include ham radio and astronomy.

Exciting new measurement capabilities are announced almost daily in the form of microprocessor-based products designed to operate as stand-alone instruments or as part of an automated measurement system.

The IEEE Standard 488, introduced in 1975, makes the task of assembling a system much easier by defining a standard interface for programmable instruments. The standard covers three basic aspects:

Mechanical—the connector and the cable.

Electrical—the voltage levels for logic signals and how the signals are sent and received.

Functional—the tasks that an instrument's interface is to perform, such as sending data, receiving data, and triggering the instrument.

Using the interface standard, instruments can be designed for basic compatibility. However, this standard is only the first step in ensuring compatibility. Instruments from various manufacturers still differ in the way in which data is encoded and transmitted. The situation can be likened to two persons trying to converse over the telephone; they have a physical connection, but unless both speak and understand the same language, little communication takes place.

To resolve this dilemma for instrumentation, Tektronix has developed a codes and formats standard. This standard establishes a common message structure, describes communication elements and how they will be combined, defines control protocol, and standardizes features that are particularly important to test, measurement, and analysis systems.

Because nearly all of today's IEEE-488 instruments use ASCII-coded characters to send and receive data, Tektronix has chosen ASCII coding as standard. In addition, nearly all instruments that send or receive numbers use the ANSI X3.42 standard format. This format states, in effect, that there are three types of numbers—integers, reals, and reals with exponents—and that they should be sent with the most significant character first. Table 1 shows examples of these formats.

Note that while a number has been defined, its use has not. The codes and formats standard places no restrictions on the use of a number. It does not matter whether a number is from a multimeter, counter, or spectrum analyzer. In all cases, the syntax, or structure, of the number is the same.

Instruments that incorporate microprocessors can perform complex functions. To have a computer or instrument controller fully interact with such devices requires code and format conventions more comprehensive than those that simply define numbers.

For instance, consider a device that makes a group of measurements and is asked to report them. This requires a group of numbers to be sent. In the Tektronix standard, a comma is used to separate one number from another. As an example, a digital multimeter might send measured output voltage limits as −2.32, 1.65.

Further, suppose that another device makes two different types of measurements, such as frequency and phase, and is asked to report them. There should be a means for identifying each type. This is done by sending a "header" that is a description of the number. If headers and

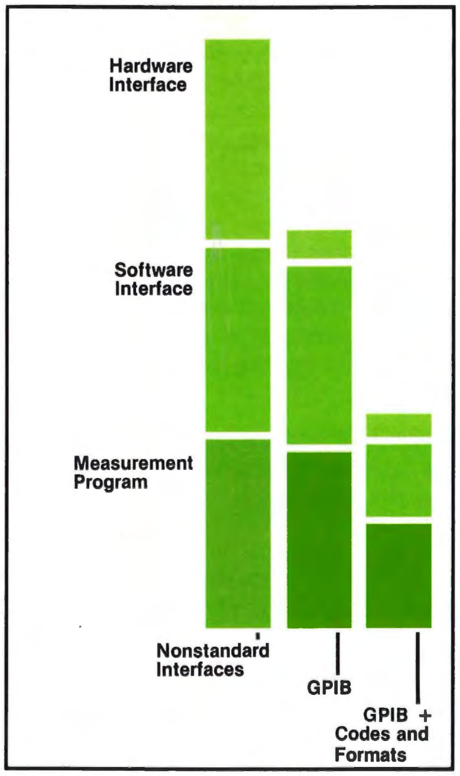


Fig. 1. Set-up times are decreased and ease-of-use is increased (improved productivity) when instruments use GPIB combined with standard Codes and Formats and user-oriented operating conventions.

Table 1: Number Formats (ANSI X3.42)

NR1	375 +8960 −328 +0000	Value of "0" must not contain a minus sign.
NR2	+12.589 1.37592 −00037.5 0.000	Radix point should be preceded by at least one digit. Value of "0" must not contain a minus sign.
NR3	−1.51E + 03 +51.2E − 07 +00.0E + 00	Value of "0" must contain an NR2 zero followed by a zero exponent.

numbers are sent sequentially, they must be separated from each other. A semicolon is used for this purpose; for example, FREQ 3570; PHASE 72.

These well-defined formats significantly enhance data communication compatibility over the IEEE-488 bus.

The human interface

The rapid growth in the numbers of instruments that include an IEEE-488 interface means more people will be involved in connecting these instruments together to perform some measurement function. Typically, the people will have widely divergent skill levels. The manner in which an instrument can be programmed will determine the skill level required to effectively use it. For instance, consider an IEEE-488 programmable power supply. It can be designed in one of two ways. The first is with minimal intelligence that allows the instrument to accept some code that can be conveniently interpreted and executed. As an example, some power supplies require the sequence 0 8 E 3 to put out 20 volts. Here, the "0" stands for the 0-to-36 volt range, and the "8E3" is the ASCII representation of the hexadecimal commands required.

The second method is to design the power supply with a microprocessor and intelligence to accept easily understood numbers. In this instance, to put out 20 volts from the positive supply of a multiple-supply instrument, the programmer simply sends the character sequence "VPOS 20". This method of interacting is obviously easier for the person writing the original program and also for someone who later has to figure out what the program is supposed to do.

Numbers in easily read formats are easily handled by both computers and people. It is also necessary to send instructions to an instrument in a format other than numbers. For example, in setting up a measurement, you may want to specify: trigger external, peak auto, and function sine. In these instances, we can treat the first word as a header and the second word as a data type that is different from a number.

Other data types, called arguments, are useful for various purposes:

Character arguments—for sending information relevant to a header but not expressable as a number; for example, GRATICULE ON or COUPLING DC.

String arguments—for sending text to a display or printer.

Binary block arguments—for sending binary data of known length.

Link arguments—for sending certain types of instrument commands.

End block arguments—for sending binary data of unknown length or format.

The same general format can be used for all types of communications on the IEEE-488 bus. Table 2 defines the message structure used for Tektronix instruments.

Message conventions

While the use of standard codes and formats enhances compatibility between devices using the IEEE-488 bus, it does not solve all of the compatibility problems. Well-defined operational conventions are also needed.

A good example is the need for a standard way to terminate messages. Two methods are currently in use: the first sends printer format characters such as CR or CR LF; and the other asserts the EOI line during the time the last data byte of the message is sent.

The first method provides opportunity for confusion. For example, it is possible for a message to contain a sequence of binary coded bytes which, if perceived as ASCII, will appear to be a CR LF and thus be misinterpreted.

The second termination method—asserting EOI concurrently with the last byte—unambiguously terminates the message. This is the method specified in the Tektronix Codes and Formats standard.

Other problems can be created by the lack of good message handling conventions. A classic example is a programmable power supply which executes each individual command as received. If the programmer neglects to set the current limit first and programs a substantial increase in output voltage, the device under test could be damaged. A power supply designed to execute a command only after receiving the entire message terminated by asserting EOI would not present this problem.

This same convention prevents misunderstandings between a computer and a measurement instrument. When instructed to send a measurement message, an instrument sends EOI only when the message is completed, and no more data is sent unless directed by the computer to do so. Thus, the computer knows the message is complete, and the instrument has not been stopped in the middle of talking.

The Tektronix Codes and Formats standard defines a message to be a complete block of information. It begins when a device starts sending a message and ends when EOI is sent concurrently with the last data byte.

Obviously, it is important to also clarify the beginning of a message. An instrument sending a message may be interrupted by the computer taking control, perhaps to conduct a serial poll. When the instrument becomes a talker again, it should resume sending the message. Thus, the message beginning is defined as the time when a device enters the talker active state for the first time following a reset or a previously sent EOI.

There are other elements in the message convention. When a device is made a talker, it should always say something. If it has nothing to say, it should send a byte of all ones concurrent with EOI. This tells the listening device that no meaningful data is forthcoming and prevents tying up the IEEE-488 bus while the computer waits for a nonexistent message.

A listening device should always handshake, even when it does not understand or cannot execute a particular message. Under no circumstances should a device execute a message it does not understand. After EOI is received, if the listening device is confused, it should send out a service request and, on a serial poll, notify the controller that the message was unclear.

Status byte and query convention

The Tektronix Codes and Formats standard also includes a status byte convention to augment that provided by IEEE-488. The IEEE-488 standard defines a facility to send a byte of status data to the computer. However, it assigns only one bit—bit 7—which shows whether a device is, or is not, requesting service.

There is a need for instruments to report other kinds of status and/or errors to the computer. One common need is for instruments to report if they are busy or ready (bit 5 is used for this purpose). Another need is to report abnormal conditions being encountered (bit 6 is selected for this). More complex conditions are reported by other status byte configura-

Table 2: Device-Dependent Message Structure

A message represents a given amount of information whose beginning and end is defined. It is communicated between a device functioning as a talker and one or more devices functioning as listeners.

A message begins when the transmitting device is initially addressed to talk and the receiving device is addressed as listener.

A message is composed of one or more message units separated by message unit delimiters. A message unit delimiter is a semicolon.

The message ends when the talking device asserts EOI.

There are two message unit types:

1. Mixed Data Message Unit.
   Two acceptable formats are:
   a. Header (in character argument format followed by a space and optional arguments of any type separated by commas).
   b. Noncharacter argument followed by optional arguments of any type separated by commas.
2. Query Message Unit.
   Consists of a character argument such as "SET", "ID", or "FREQ" followed by a question mark.

Argument Types and Examples	Definition
Character Argument: TRIGGER	One alphabetic ASCII character optionally followed by any number of ASCII characters excluding space, comma, semicolon, question mark, the control characters, and rubout.
Noncharacter Arguments: Number Argument −12.3	A numeric value in any of the formats shown in Table 1.
String Argument "Remove Probe"	Opening delimiter (single or double quote) followed by a series of any ASCII characters except for the opening delimiter, and a closing delimiter identical to the opening delimiter.
Binary Block Argument % | **2 bytes** 16 bit binary value | **Binary Data** wave-form values | **8 bits** checksum	"%" followed by a 2-byte (16 bit) binary integer specifying the number of data bytes, plus a checksum byte which follows the data bytes. The checksum is a twos complement of the modulo 256 sum of the preceding binary data bytes. This includes the two bytes comprising the 16-bit integer specification.
Link Argument NT.PT:1024	Character argument (label) followed by ":" and a value represented in any of the above argument types.
End Block Argument @ABCDEFGHIJKL E O I	"@" followed by a block of data with EOI set concurrent with the last data byte. End block can only be the last argument in a message and cannot be followed by a message unit delimiter.

tions as shown in Table 3. These status bytes are useful for most purposes; however, some instruments may have conditions that are peculiar to them. Bit 8 is used to indicate that a status byte is particular to an instrument.

A standard coding for the status byte is a convenience when programming a system, especially if all instruments use the same coding. This allows a common status byte handling routine to be written for all instruments.

To supply even more detailed information than the status byte can convey, a set of queries are used. Queries take the form of a header followed by a question mark.

A typical example is shown in figure 2. Here, the computer has asked the instrument to state its frequency setting.

A SET? query makes it possible to develop a program using an instrument's front panel as an input to the computer. The programmer using this feature may never need to know the instrument's IEEE-488 bus commands. Implementing a SET command restores the instrument to the state it was in when the SET? query was invoked.

Some fine touches

As instruments get more complex and possess computer-like capabilities, more operational conventions will be needed. These conventions should make the instrument easier to use if the intelligence is used properly. Here are some examples of conventions adopted by Tektronix to enhance both computer/instrument compatibility and human/system compatibility:

- While an instrument should always send numbers in the correct format described earlier, it should receive numbers forgivingly. Specifically:
  - Negative zero numbers should never be sent, but they should be accepted.
  - Any number sent in scientific notation should be sent exactly as defined in ANSI X3.42 standard; that is, with the decimal point included. Some computers violate this standard by omitting the decimal point. This "illegal" number should be received with an implied decimal point following the least significant digit.
  - If an instrument receives a number whose precision is greater than the instrument can handle internally, the number should be rounded off, not truncated.
- An instrument should recognize both spaces and commas as argument delimiters. Multiple spaces or commas should not be construed as delimiters for null arguments.
- An instrument should receive headers and character arguments in both upper and lower case and equate them (a = A, b = B). Some desktop instrument controllers have a problem sending upper case alpha characters.

Table 3. Status Byte Convention

Name	Code	Function
Normal conditions:		
Normal status	000X 0000	Response to serial poll when condition of instrument is normal.
SRQ query request	010X 0000	Tells controller to query device for service needed.
Power on	010X 0001	Reported after every power on. Tells controller that device is on the bus.
Operation complete	010X 0010	Tells controller that a task is completed.
Abnormal conditions:		
ERR query requested	011X 0000	Reports error but does not identify it. Controller should send query to identify errors.
Command error	011X 0001	Reports that message received cannot be parsed.
Execution error	011X 0010	States that message cannot be executed.
Internal error	011X 0011	States that device is out of calibration or is malfunctioning.
Power fail	011X 0100	Notifies controller that power failure is occurring. Controller may take action to save data or flag suspect operation.
Execution error warning	011X 0101	Warns that device has received and is executing a command but that a potential problem exists.
Internal error warning	011X 0110	Warns that device has an internal failure but is continuing to function.

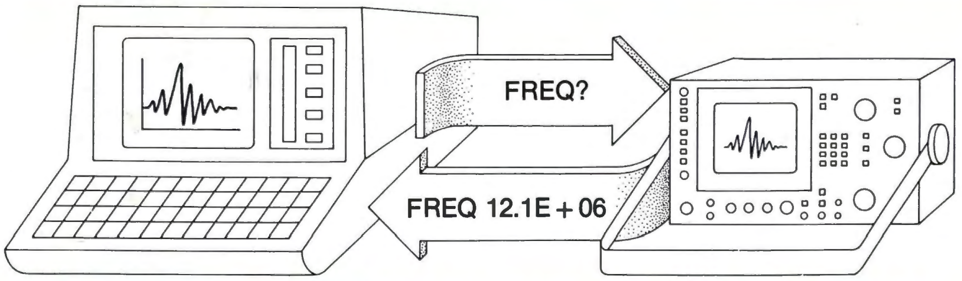


Fig. 2. The use of easy-to-remember queries is an important feature of Tektronix GPIB instruments. Many query commands are formed by adding a question mark to the mnemonic for the setting to be queried.

- An instrument sending data about its front-panel status should use headers and character arguments that correspond to the front-panel nomenclature.

Compliance with these conventions make Tektronix instruments "friendly" to a casual or inexperienced programmer and compatible with most computers.

There are other features built into Tektronix intelligent instruments that enhance their usability. Here are two examples:

The Service Request (SR) function and corresponding status byte are very important. They can alert the instrument controller to new events or possible malfunctions. For example, sometimes a computer or instrument does not want to be interrupted. For these cases, an RQS OFF message can be sent to disable any service requests. To turn the service request capability back on, an RQS ON is sent.

Another useful convention relates to the Device Trigger (DT) function. Sometimes a command message sent to an instrument should be executed immediately. At other times, the command should only set up the instrument, and the desired action should be executed when the Group Execute Trigger interface message is sent. To make the instrument execute commands immediately, the message DT OFF is sent. Conversely, to make the instrument defer execution of commands, the message DT followed by a descriptive character argument such as TRIG or GATE is sent.

Summary

The Tektronix Codes and Formats standard is designed to extend the compatibility of programmable instruments. This standard should reduce the cost and time required to develop system and applications software by making it easier to generate and understand the necessary device-dependent coding.

Operational conventions are an important part of the standard. They ensure that communications over the bus take place in a logical, consistent manner and avoid ambiguities between the transmitted and received message. ■

9/81 99AX-4896

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

BULK RATE
U.S. POSTAGE PAID
TEKTRONIX, INC.

0421867 1337
MR THOMAS P DOWNEY LAB TECH
GEOLOGY DEPT EPS-J106
CITY UNIVERSITY OF NEW YORK
138TH ST AT CONVENT AVE
NEW YORK NY 10031