

# HANDSHAKE

NEWSLETTER OF SIGNAL PROCESSING AND INSTRUMENT CONTROL



**Winning the component test race**



## Table of contents

<i>A case history for quality— Motorola uses TEK 7912AD for 100% testing of crystal oscillators</i> .....	3
<i>Programming hint... Use RENAME for TEK SPS BASIC program security</i> .....	7
<i>Semiconductor switch testing Part I—Acquiring the waveforms</i> .....	8
<i>Semiconductor switch testing Part II—Processing the waveforms</i> .....	14
<i>Getting the most out of TEK BASIC graphics</i> .....	20
<i>7912AD assists in measuring temperature of fluctuations in cold turbulent flows</i> .....	22
<i>New ROM packs enhance GPIB capabilities of 4050-Series Controllers</i> .....	23
<i>Tektronix Programmable Systems Workshops</i> .....	24

**Return the reply card in this issue for a free subscription to HANDSHAKE.**

**Managing Editor:** A. Dale Aufrecht

**Editing and Writing Staff:**

Bob Ramirez

Mark Tilden

**Graphics:** Joann Crook

**Production:** Brenda Smethers

HANDSHAKE is published quarterly by the HANDSHAKE Group. Permission to reprint material in this publication may be obtained by writing to:

HANDSHAKE Editor  
Group 157 (54-016)  
Tektronix, Inc.  
P.O. Box 500  
Beaverton, Oregon 97077

HANDSHAKE is provided free of charge by Tektronix, Inc., as a forum for people interested in programmable instrumentation and digital signal processing. As a free forum, statements and opinions of the authors should be taken as just that—statements and opinions of the individual authors. Material published in HANDSHAKE should not be taken as or interpreted as statement of Tektronix policy or opinion unless specifically stated to be such.

Also, neither HANDSHAKE nor Tektronix, Inc., can be held responsible for errors in HANDSHAKE or the effects of these errors. The material in HANDSHAKE comes from a wide variety of sources, and, although the material is edited and believed to be correct, the accuracy of the source material cannot be fully guaranteed. Nor can HANDSHAKE or Tektronix, Inc., make guarantees against typographical or human errors, and accordingly no responsibility is assumed to any person using the material published in HANDSHAKE.

## Apologies and corrections


Although we carefully proof HANDSHAKE at several stages before it is printed, some things do slip by us. Mostly it's just innocuous typos that get by. But we really goofed in the Winter 1981/82 issue. For that we apologize and offer the following corrections.

On page 15 of the Winter 1981/82 issue in the listing in Fig. 10, there's a typo in the first line. EVICE should be DEVICE. But more importantly, lines 30 through 60 should be corrected to read as follows:


```
30 ERROR$=ERROR$&"66,82:OPERATION COMPLETE;97,113:COMMAND ERROR;"
40 ERROR$=ERROR$&"98,114:EXECUTION ERROR;99,115:INTERNAL ERROR;"
50 ERROR$=ERROR$&"100,116:POWER-FAIL;101,117:EXECUTION WARNING;"
60 ERROR$=ERROR$&"102,118:INTERNAL WARNING;"
```

Without this correction, the complete ERROR\$ string variable cannot be constructed.

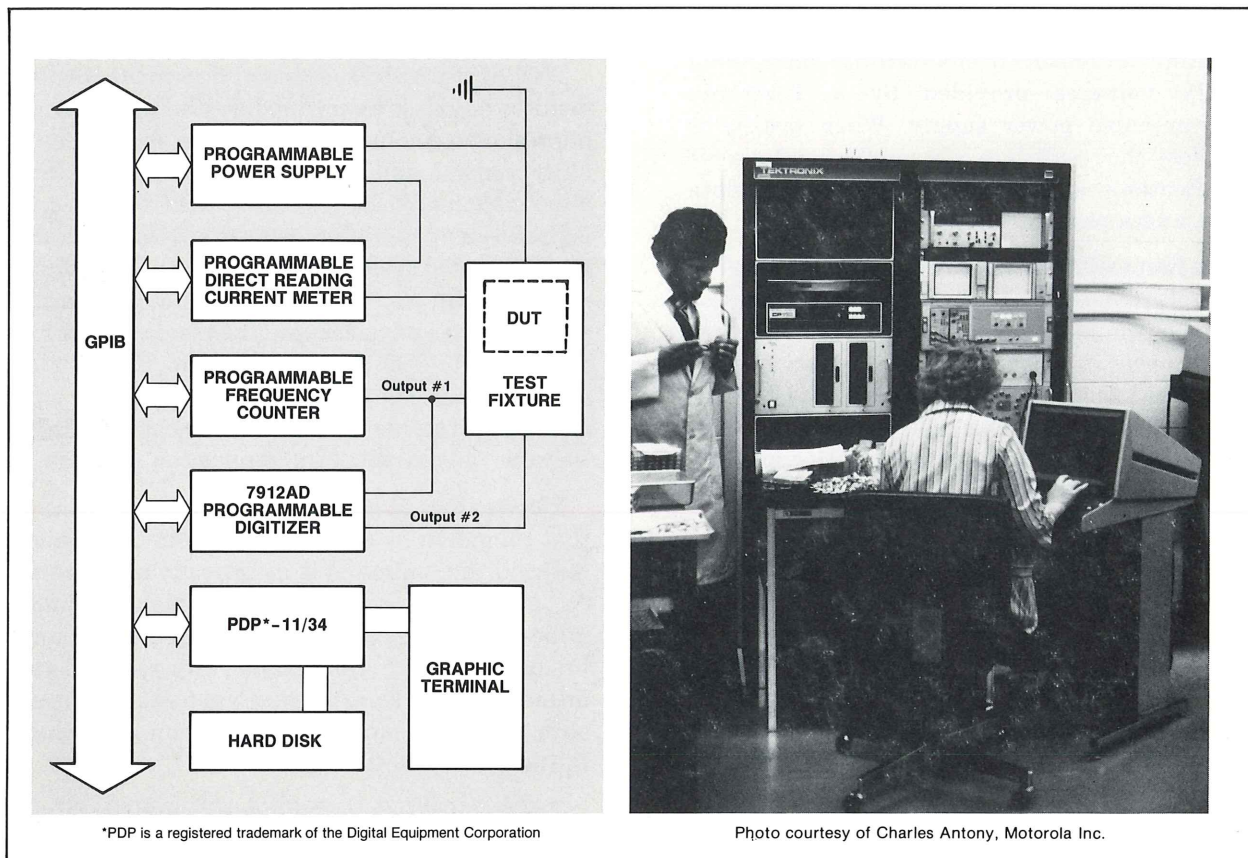
Another oversight concerns Fig. 1 on page 17. The photos in that figure were inadvertently switched. Also, in the same article, there is some command confusion. On page 18, where starting the program with 0 GOTO START is discussed, the 0 GOTO is redundant since START automatically begins program execution at line 0. And, on the next page, where 34 GOTO START and 5 GOTO START are used to access different program segments, RUN should be used instead of START.

If you keep HANDSHAKE on file for future reference, please be sure to go back to your Winter 1981/82 copy and note these corrections. 

## New address?

If you are a subscriber to HANDSHAKE and your address has changed, please be sure to use the HANDSHAKE reply card to notify us of the change. Print your new address on the reply card and attach the mailing label from your latest copy of HANDSHAKE. We'll update our mail list and make sure that you continue to receive HANDSHAKE at your new location. 

# A case history for quality— Motorola uses TEK 7912AD for 100% testing of crystal oscillators



**Fig. 1.** Basic test system for 100% quality testing of crystal oscillators.

Using oscilloscopes, how many crystal oscillators can be manually tested in a single day? How extensive is the evaluation?

Not enough was the answer to both questions for the Engineering department at Motorola. So they began looking for a better way, a faster way, a more complete way of evaluating their line of crystal oscillators.

---

**...with the speed offered by digital signal processing, Motorola is able to fully test each and every component coming off the line. For Motorola, that means high product reliability and high customer satisfaction.**

---

## Better testing in less time

The better way turned out to be a Tektronix 7912AD Programmable Digitizer operating under TEK SPS BASIC software. "The primary considerations for production test are the system has to be fast, it has to be reliable, and it has to be repeatable," states Charles Antony, a Motorola engineer involved in developing crystal oscillator test programs with TEK SPS BASIC. Mr. Antony went on to state that they are pleased with the results they are getting from Tektronix instruments and software.

Previously, testing had been done with trained operators using oscilloscopes to inspect crystal



## A case history for quality...

oscillator waveforms. The process was slow and subject to human error.

Now, however, with the speed offered by digital signal processing, Motorola is able to fully test each and every component coming off the line. For Motorola, that means high product reliability and high customer satisfaction.

Testing is 100%. All pieces are tested for all operating parameters over a 4.5 to 5.5 volt range of supply voltages provided by a Tektronix programmable power supply. When testing is complete, the results are compared by software to specifications stored from the component data sheet, and a pass-fail decision is made.

The first testing programs put into operation by Motorola did the full testing job in about five seconds per component. That, in itself, was a considerable improvement over manual testing, not to mention the bonus of complete testing. However, since testing speed is a primary consideration, program development continued until the test time per crystal oscillator was reduced.

Currently test time per component has been reduced to about 620 milliseconds. Waveform capture and transfer is included in that 620 milliseconds. And it was all done without sacrificing anything in the quality or extent of testing. Every crystal oscillator is still fully functionally tested. The test programs were just made more efficient.

Such efficiencies can be gained in a variety of ways. However, there are two key aspects of TEK SPS BASIC that can be of considerable assistance. One aspect is the extensive set of commands that provides the flexibility needed for implementing complete testing programs. Having the specifically needed command available for the job means programming efficiency as well as programs that run efficiently. The other aspect is the memory management features of TEK SPS BASIC V02XM and their role in reducing test program run times by providing access to more memory at semiconductor speeds.

### Flexibility through modularity

TEK SPS BASIC was designed as a general purpose waveform processing and instrument control software package. As a result, its commands and features are extensive, too extensive for all to be stored at one time in PDP-11 memory. So, to provide equal access to all of the

---

**...modularity provides an operating system that changes dynamically to meet the needs of individual application programs.**

---

software's capabilities, a scheme of modularity was devised.

TEK SPS BASIC consists of a resident BASIC monitor as the primary module and a variety of nonresident commands, peripheral drivers, and other control and processing packages as the other modules. When TEK SPS BASIC is booted, it is the resident monitor module that is loaded from the designated system device (floppy or hard disk) and brought up. During subsequent program development and execution, the resident monitor remains in PDP-11 memory and supervises, among other things, program execution and loading and storage of nonresident commands as they are referenced by an application program.

Referencing a nonresident command causes that command to be autoloading from the system device if the command is not already in memory. As a program executes, the resident monitor tallies the usage of each autoloading command. Frequently used commands are retained in memory for quick access, and less frequently used ones are autoloading from the system disk when the occasional need arises.

The result of this modular design and operation is that you have access to a wide variety of commands and capabilities via autoloading from the system disk while not burdening computer memory with seldom or never used commands. In short, TEK SPS BASIC modularity provides an operating system that changes dynamically to meet the needs of individual application programs.

### LOADed for speed

For production purposes, once programs are written to do the required tests, making the programs run faster becomes the next concern. Time consuming operations must be reduced or eliminated.

Some time can be saved beforehand with proper system specification. For example, a system controller with fast execution times should be selected. Also, a hard disk provides faster access time than a floppy disk. Therefore, a hard disk is



preferred as the device from which programs and drivers are loaded and nonresident commands autoloading. This type of system selection is indicated in the block diagram of the Motorola crystal oscillator test system (Fig. 1). The Motorola system also included cache memory in the PDP-11 to further speed processor operations.

Beyond providing fast peripheral storage, another step is to limit the number of times peripheral storage needs to be accessed. Keeping peripheral accesses few, as well as fast, markedly improves program execution times.

One approach taken by Motorola is to preload the necessary nonresident commands and drivers before loading and executing the test program. An example of this approach is shown in Fig. 2, where an initialization program loads commands and then deletes itself by OLDing in the test program.

(OLD deletes all defined string and numeric variables, string and numeric arrays, waveforms, and existing program lines before loading in the new program.)

### Keeping peripheral accesses few, as well as fast, markedly improves program execution times.

The time saving comes about because any module that is loaded with the LOAD command becomes part of Resident BASIC. The module remains resident until it's specifically released with the RELEASE command or until TEK SPS BASIC is rebooted. As part of Resident BASIC, the command or module is available for immediate program execution—no time is spent fetching it from peripheral storage.

#### Start up— an atypical test

Sometimes component qualification for a customer requires tests other than those typically performed. Oscillator start-up time is an example of such an atypical test.

Start-up time is the time it takes from applying power to an oscillator until it reaches full output amplitude. The programs listed here show the approach used by Motorola to accommodate a special customer requirement.

The first segment of the listed programs (lines 10-40) are initialization steps. The necessary commands are explicitly loaded in this segment. Then the start-up routine, STAR.BAS, is loaded.

The second segment, lines 5-140, is STAR.BAS. It's initial activity is instrument set up over the GPIB. Line 45 makes sure the oscillator supply is set to zero volts, and lines 50-85 prepare the 7912AD for waveform acquisition. Power is applied to the oscillator under test at line 90, and the digitized start-up waveform is read into array QQ at line 100. Processing to find start-up time, S1, occurs in lines 115 and 120 with the results output in 125. The last step, then, is turning off the supply, line 130, in preparation for the next device to be tested.

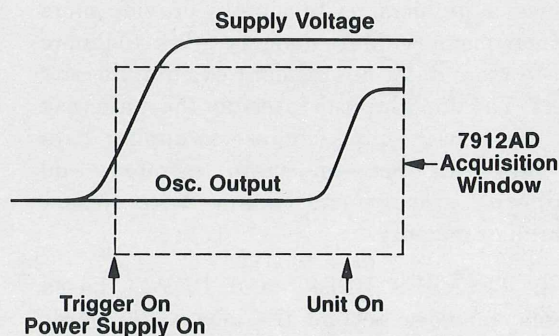
```

10 LOAD "PUT", "WAIT", "PRINT", "IFDTM", "GPI", "READBI"
20 LOAD "PAGE", "SIFLIN", "SIFCOM", "SIFTO"
30 OLD "STAR.BAS"
40 END

5 PAGE
10 SIFLIN @0, "IFC"
15 SIFCOM @0, 32, 96, "DCL"
20 SIFCOM @0, 32, 96, "GTL"
25 DELETE QQ, F, WF
30 DIM QQ(511)
35 WAIT 50
40 SIFTO @0, 300
45 PUT "U0$" INTO @0, 39
50 PUT "MAI 252" INTO @0, 32, 96
55 PUT "POS -1" INTO @0, 32, 97
60 PUT "RIN LOW" INTO @0, 32, 97
65 PUT "POS .5" INTO @0, 32, 98
70 PUT "MOD SSW; SRC EXT; T/D 1E-03; CPL DC" INTO @0, 32, 98
75 PUT "U/D 100E-03" INTO @0, 32, 96
80 PUT "DIG SSW" INTO @0, 32, 96
85 PUT "ATC; READ ATC" INTO @0, 32, 96
90 PUT "U1+4.5$UX$" INTO @0, 39
95 IFDTM @0, "PAK"
100 READBI QQ FROM @0, 64, 96
105 WAVEFORM WF IS F(511), DF, HFS, UFS
110 F=(QQ/2)*(1/64)-4
115 P1=CRS(WF, .9)
120 S1=P1*1E-03/51.2
125 PRINT "START-UP TIME =", S1
130 PUT "U0$" INTO @0, 39
135 WAIT
140 GOTO 5

```

a. Start-up test program.



b. Start-up waveforms.

Fig. 2. Testing crystal oscillators for start-up time.



## A case history for quality...

In the Motorola program (Fig. 2), notice that all nonresident commands and drivers referenced by the test programs are preloaded by the initialization program. That means that when the test program is run, it never has to access peripheral storage. However, it should also be noted that nine nonresident commands and a GPIB driver (GPI) were preloaded. To put more than six nonresident commands in memory at one time, either by autoloading or LOAD, the default limit (six) for nonresident commands must be changed.

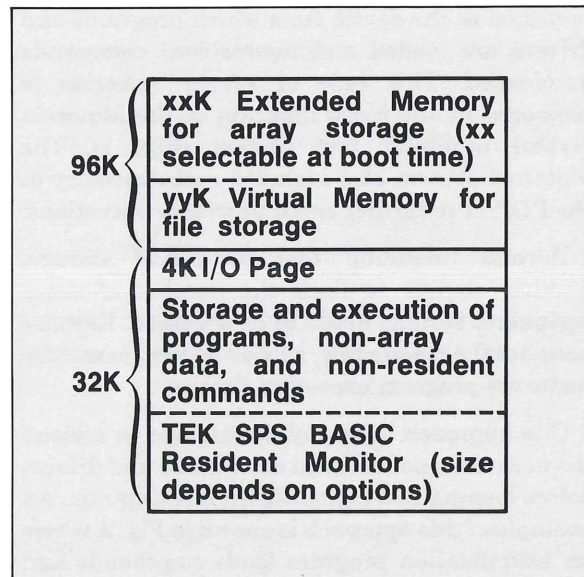
Changing any of the operating system parameters is done at system load time by executing the SYSBLD command after TEK SPS BASIC is booted. A number of system parameter prompts are displayed when SYSBLD executes. You can change the parameters according to the prompts or retain the default parameters by entering a carriage return. In the case of nonresident commands, simply specify the number of commands you wish to have in memory at one time when that prompt comes up.

In loading additional nonresident commands, there are some memory constraints that need to be considered. The more commands you load, the less space available for waveform storage, program storage, and program execution. More memory can be freed up by releasing commands or drivers after the need for them has passed. This does, of course, take some additional program steps that are housekeeping rather than measurement task related.

### More fast memory helps

Another approach, when memory housekeeping becomes a problem, is to simply provide more computer memory. More memory gives you more room to store data, nonresident commands, and drivers. You don't have to interrupt the main task with the housekeeping chores—swapping data and programs between main memory and peripheral storage—necessary with lesser amounts of memory.

With TEK SPS BASIC and PDP-11 based systems, memory beyond the lower 32K words means going with an extended memory system. In a PDP-11 minicomputer and operating with TEK SPS BASIC V02XM, you have an additional 96K words of extended memory available. With the 32K of lower memory that's 128K total. The



**Fig. 3.** Memory map for TEK SPS BASIC V02XM operating with 128K words of installed memory.

memory map in Fig. 3 shows how this 128K can be allocated.

Referring to the memory map in Fig. 3, when the operating system is booted, the default size of extended memory is the full 96K words. This memory area is defined by TEK SPS BASIC as being exclusively for storing numeric arrays and waveform arrays.

The additional array storage provided by extended memory can affect program execution speed in several ways. Primarily, areas in the lower 32K of memory that might otherwise be used for waveform storage are freed up for storing more program lines or nonresident commands. Another aspect is the number of waveforms that can be stored in the upper 96K. This can be an important time savings in applications requiring access to more than just one or two waveforms. Without extended memory it may be necessary to temporarily store some arrays on a peripheral and read them in from the peripheral as they are needed by the analysis program. However, if the arrays are stored in extended memory, the steps of accessing peripheral storage become unnecessary.

Also indicated in Fig. 3 is another type of memory area. It's referred to as virtual memory. Virtual memory exists only if it has been requested at load time. Requesting is done via SYSBLD and is somewhat indirect since it



depends on the answer to the following SYSBLD question—

“How many 1K-word segments do you want to reserve for array storage in extended memory (Default is 96 or maximum extended memory in system)?”

Specifying less than the maximum extended memory results in the remaining portion of memory being reserved as virtual memory.

Virtual memory is treated as a file-structured peripheral. In appearance, it is just like a small but extremely fast disk. As such, it provides an access speed equivalent to the installed semiconductor memory. What this means in terms of production test is that various programs or program segments can be loaded or overlaid from virtual memory at speeds in excess of those offered by standard peripherals.


Actual implementation of virtual memory is a matter of booting the operating system, including a SYSBLD to set up virtual memory size, and then copying all needed test programs from the system disk into virtual memory (referred to as VM: by TEK SPS BASIC). From there on, test programs can be loaded or overlaid at high speed directly

from VM into the lower 32K of operating system memory.

### New directions

In the end, no single approach suffices for a complete testing program. Explicitly loading nonresident commands, overlaying from virtual memory, and availability of more processing modules all become important as test programs are expanded to provide more services.

Motorola's crystal oscillator testing program is a good example. For Motorola, high-speed, 100% testing of their product is reality. But now they're looking at doing more. Recent conversations with Mr. Antony of Motorola have centered on his interest in adding graphics routines to provide more detailed operator prompts. That way, they can reduce operator training time.

And once that's done...well, it's nice to have the flexibility of modular software for making future additions. 

*By Bob Ramirez, HANDSHAKE Staff, with grateful acknowledgement to Charles Antony, Motorola, Inc., for contributing programming examples and application information on testing crystal oscillators.*

*HANDSHAKE also thanks Bob Beatty of the Tektronix St. Paul Field Office for putting us in contact with Mr. Antony.*

---

## Programming hint... Use RENAME for TEK SPS BASIC program security

There may come a time when you'd rather not have people accessing your TEK SPS BASIC programs. This is particularly true if you've developed the programs for use on the production floor. It's nice to be able to keep system operators from listing programs, thus discouraging them from making unauthorized changes.

One way to do this is to cancel the LIST.SPS file on the system disk used on the production floor. Make sure you have a backup disk, though, with LIST.SPS on it.


The problem with canceling LIST.SPS on the working disk is that no one can list the programs, not even you. That makes it difficult to check bugs or make authorized program changes on the production floor.

A more convenient means of program security, one that allows you to still list and debug programs, is to simply rename LIST.SPS to

something else, a code name that's released only to authorized software people. For example, entering

RENAME "LIST.SPS" TO "CODE.SPS"

renames the LIST command to CODE. When the working disk is rebooted on the production floor, entering LIST results in a FATAL P-9 error. The only way to get a listing is to enter the new name, CODE in this example.

Of course, some system operators may be astute enough to do a disk directory (DIR) and figure out by process of elimination what LIST has been renamed to. If you think this is a possibility, give yourself another level of program security. Rename DIR.SPS to a confidential code name as well. 

*By Bob Ramirez,  
HANDSHAKE Staff*

# Semiconductor switch testing

## Part I—Acquiring the waveforms

Fast switching devices are playing an ever increasing role in reducing the size of electronic products and improving their efficiency. The benefits are tremendous, but fast switching techniques also pose some testing problems.

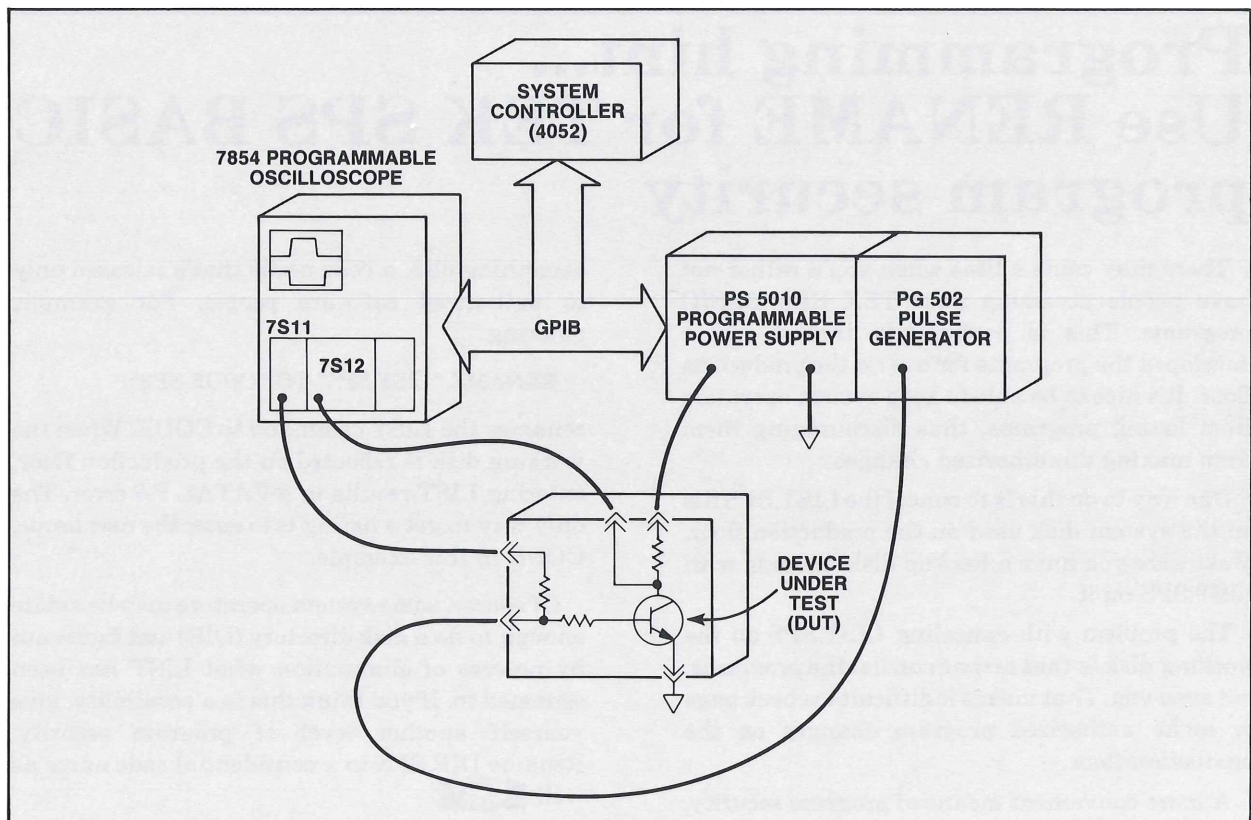
Today's switching devices demand faster testing. Testing has to be faster in terms of bandwidth simply because switching times are faster. And testing has to be faster in terms of devices tested per unit time simply because of economic demands.

A system of programmable instruments coordinated by a powerful GPIB system controller makes it possible to perform the high-speed tests required. And since the tests require little or no human interaction, repeatability is significantly improved. Such a system is shown in Fig. 1.

The system in Fig. 1 is designed to test switching characteristics of discrete transistors.

The device under test (DUT) is inserted in a test circuit and stimulated with a fast pulse. A Tektronix PS 5010 Programmable Power Supply provides the supply and bias voltages and a Tektronix 7854 Oscilloscope acquires and processes the test signals. The Tektronix 4052 Graphic Computing System manages the system, setting the power supply voltages, initiating acquisitions, and processing, displaying, and logging the results.

The same type of system can be applied to testing just about any kind of semiconductor switching device. Though the voltage levels and timing values may change with different types of devices, the basic tests are usually the same. For example, in discrete transistor testing, measurements are made based on 10% and 90% points of the waveforms. But, the tests could just as well be made at 20% and 80% points for ECL devices.



**Fig. 1.** A typical system designed to test discrete transistors for switching parameters.



Figure 2 shows typical switching waveforms acquired by the system. For switching transistors, the parameters of interest are:

Rise time—from 10% to 90% points of the output rising edge.

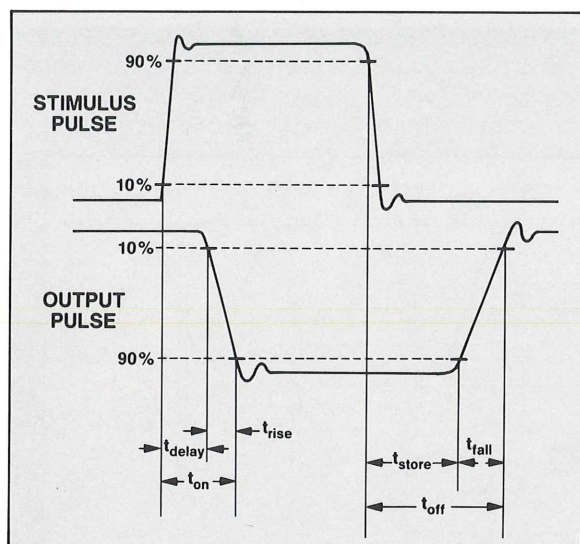
Fall time—from 90% to 10% points of the output falling edge.

Delay time—from 10% point of input pulse leading edge to 10% point of the output leading edge.

Storage time—from 90% point of the input pulse falling edge to 90% point of the output falling edge.

On time—the sum of the rise time and delay time.

Off time—the sum of the fall time and storage time.



**Fig. 2.** Typical waveforms acquired from the system shown in Fig. 1. For discrete transistors, measurements are all based on 10% and 90% levels.

## Acquiring the waveforms

Accurate, repeatable results from any automated test depend on proper data acquisition. The results can be no better than the acquisition, so it's important to take a careful look at the acquisition process.

In the example system (Fig. 1), acquisition is handled by the 7854 Oscilloscope. The 7854 can acquire waveforms in one of three modes—Single-Shot (AQS), Repetitive (AQR), and Signal Averaging (AVG). The single-shot acquisition speed is limited to about 100 kHz, so it is not useful in this application. Since switching waveforms are repetitive, the AQR mode can be used. To

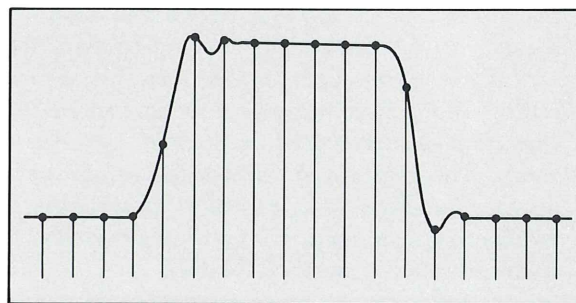
acquire and store a waveform, just press AQR or send the AQR command over the bus. Up to two waveforms can be acquired at once using the ALT mode on the mainframe or plug-in.

If noise is a problem, the AVG mode can be used to reduce random noise. To initiate an averaging acquisition, just enter the number of averages to be performed on the waveform calculator keyboard and press AVG or press the 10 AVERAGE, 100 AVERAGE, or 1000 AVERAGE buttons on the 7854 front panel. Over the bus, just send the number of averages followed by the AVG command. The 7854 will only allow averaging of one waveform at a time, so the mainframe and plug-ins must not be in ALT or CHOP modes.

In repetitive or signal averaging mode, the 7854 can acquire waveforms up to 400 MHz with real-time plug-ins such as the 7B87 Time Base and 7A19 Vertical Amplifier. To push the bandwidth beyond 400 MHz, sampling plug-ins, such as the 7S12 TDR/Sampler, can be used. There's an added advantage in using sampling plug-ins with the 7854—they can significantly reduce acquisition time. The faster acquisition is a result of the way the 7854 digitizes with the 7S12. The box titled "Sampling a Sampler" describes the 7854/7S12 operation in more detail.

## Choosing a sweep rate

Much of the information in switching parameters testing is contained in the fast rising and falling portions of the acquired waveforms, so it's especially important to capture sufficient samples on these portions of the waveform. Figure 3 shows how a typical waveform is sampled. Remember that the samples are equally spaced horizontally. As a result, fewer samples are captured on the transitions than along the flat base and top of the pulse.



**Fig. 3.** Samples are equally spaced horizontally. As a result, fewer samples are acquired on the fast transitions of the waveform than along its base or top.



It's important to use a fast enough sweep speed to acquire several samples on the rising and falling edges. Higher sweep speed means more samples per unit time, and better time resolution. However, if all the measurements must be made without manual intervention, the sweep speed must be low enough to capture the entire waveform, including the rise and fall of both input and output pulses in one acquisition.

Figure 4 illustrates the effects of different sweep speeds on the acquisition. In part a of the figure, the high speed provides excellent resolution on the rising edge, but the falling portion of the response pulse is not captured. This makes measuring the fall time, storage time and off time impossible without another acquisition.

Part b of Fig. 4 shows an acquisition with too slow a sweep speed. The problem here is insufficient resolution on the rising and falling edges of the waveform. Insufficient resolution makes computation of the 10% and 90% points inaccurate and, as a result, reduces the accuracy of the time measurements.

The best sweep speed is usually the fastest one that will allow the entire input and output pulse to be captured in one acquisition (part c). In the case of the transistor switching system shown in Fig. 1, a sweep speed of 20 nanoseconds/division provides sufficient resolution on the rising and falling edges and still captures the entire

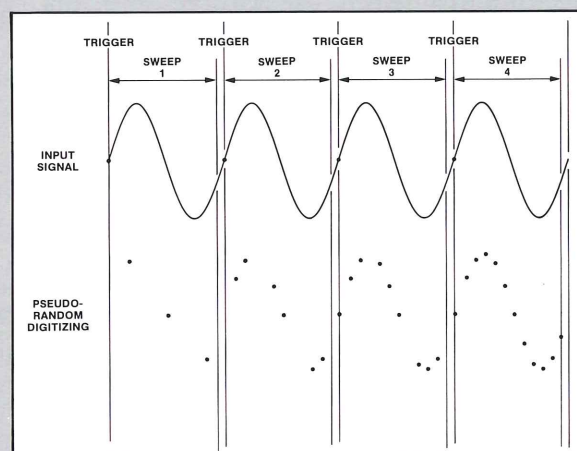
*continued on page 12*

### Sampling a sampler

The 7854 uses pseudo-random digitizing (also called equivalent-time sampling) in all cases except single-shot acquisition with the 7B87. In pseudo-random digitizing, the digital representation of the input waveform is constructed from several successive sweeps of the waveform (Fig. 1). A few samples are captured on each sweep until the complete waveform is sampled. The faster the sweep speed, the fewer samples are taken each sweep, so more sweeps are required to capture the entire waveform. As a result, waveforms captured at very high sweep speeds can take many sweeps and several seconds to acquire. If signal averaging is used, many acquisitions are required, so the total acquisition time can be even longer.

Sampling plug-ins can reduce the acquisition time as well as extend the useful bandwidth of the instrument. They take a high-speed repetitive waveform and convert it to a lower-frequency copy of the waveform. To visualize how this is done, refer to Fig. 2. Two ramp signals are generated in the 7S12—a high-speed ramp and a low-speed ramp. The high-speed ramp runs at the speed determined by the TIME/DIV control. For 20 nanoseconds/division, the fast ramp period is 200 nanoseconds (20 ns/div x 10 div).

The slow ramp runs at a much slower rate set by the SCAN control. Its rate does not affect the horizontal accuracy of the 7S12, just the refresh rate of the display. The two ramps are fed into a

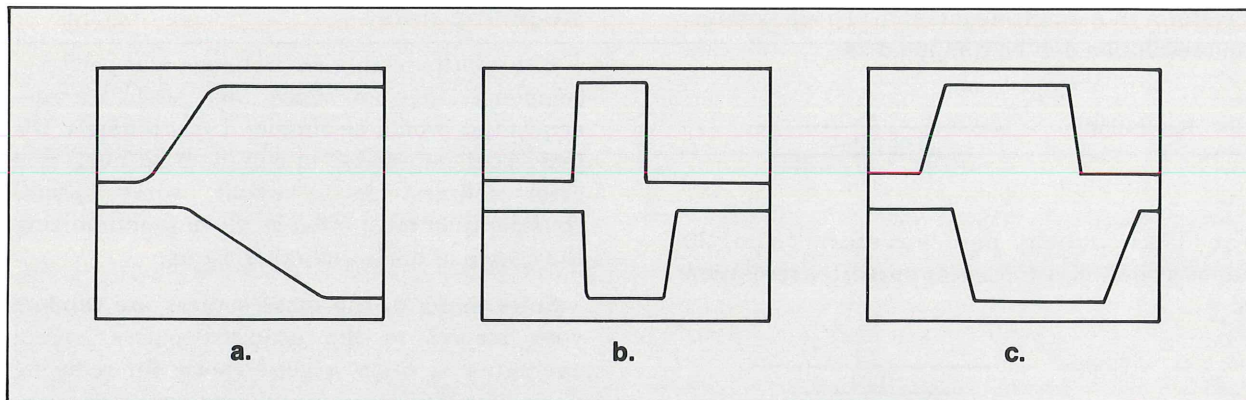


**Fig. 1.** Pseudo-random digitizers sample a repetitive signal on several successive sweeps, gradually collecting enough samples to fully define the waveform.

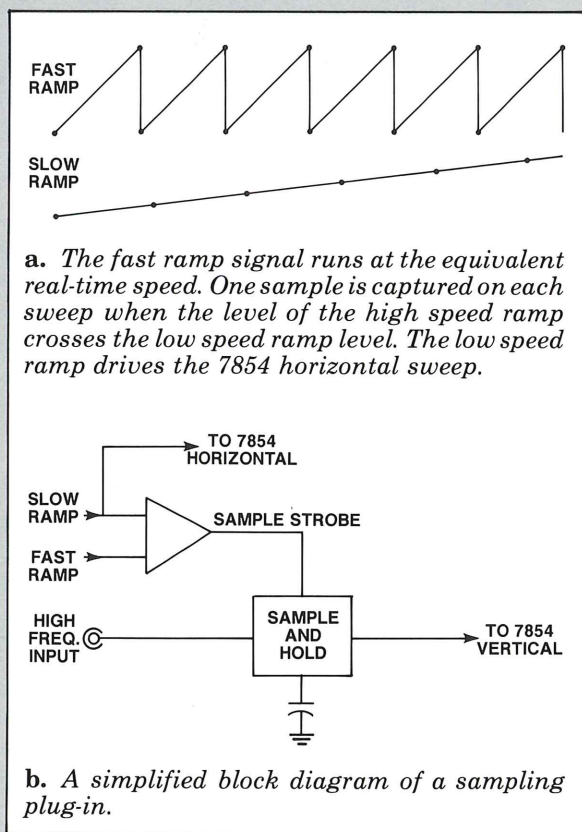
comparator that produces a pulse whenever the high-speed ramp crosses the level of the low-speed ramp. On each pulse a sample of the analog input voltage is stored and displayed on the CRT. Samples are stored at a different point on each fast ramp, since the level of the slow ramp changes as shown in the figure.

The 7854 horizontal circuits are driven by the slow ramp and the vertical circuits are driven by the output of the sampling memory. The Z-axis of the 7854 is turned on for a short time to display the sample. Then, the Z-axis is turned off until the



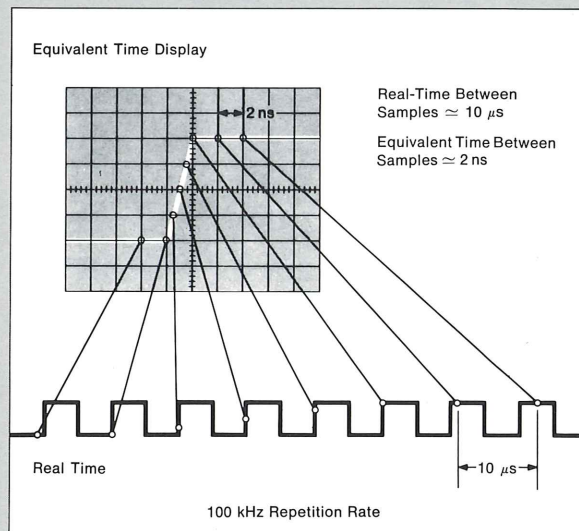


**Fig. 4.** Choosing the right sweep speed is a compromise between resolution and capturing the waveform in a single acquisition. **a.** Too fast a sweep speed provides good resolution on the rise, but won't capture the entire waveform. **b.** Too slow a sweep speed provides insufficient resolution on the rise and fall. **c.** A good compromise is the fastest sweep speed that will capture the entire waveform.




**Fig. 2.** Sampling plug-ins like the 7S12 convert high-frequency input signals to a lower frequency so that they can be displayed (and digitized) on a lower bandwidth oscilloscope.

next sample is captured. The result is a much slower copy of the analog input waveform as shown in Fig. 3.



**Fig. 3.** Sampling plug-ins capture a single sample on each high speed sweep. Each sample is captured at a point slightly later on the sweep so the result is a low-speed copy of the real-time waveform.

The 7854 samples and digitizes this low-frequency copy of the input waveform. Since the sweep speed the 7854 sees is quite slow, it can capture many samples on each sweep. As a result, fewer sweeps are required to capture the waveform than if it is acquired with real-time plug-ins, such as the 7A19 and 7B87.

In addition, the 7S12 with an S-6 sampling head has a 35 picosecond rise-time. This extends the useful bandwidth of the 7854 far beyond its capability with real-time plug-ins. 

waveform in a single acquisition. The maximum time resolution can be computed as:

$$\text{Resolution} = \frac{1}{\frac{\# \text{ of points}}{10 \text{ div} * 20 \text{ ns/div}}}$$

For 1024 points per waveform and 20 nanoseconds/division sweep speed, the resolution is:

$$195 \text{ ps} = \frac{1}{\frac{1024}{(10 * 20 \text{ ns/div})}}$$

Thus, the maximum time resolution (without interpolation) is about 200 picoseconds.

If additional resolution is required, the acquisition can be broken into three parts with some manual intervention. First, the complete waveform can be acquired to determine the pulse top and base amplitude. Then, the sweep speed can be increased and the waveform repositioned with the 7S12 TIME/DISTANCE knob. The fast rising edges can be acquired at this higher sweep speed to measure the rise time, delay time, and on time. The top and base amplitudes measured on the first acquisition are used to compute the 10% and 90% points. Finally, the falling edges are acquired at a high sweep speed to measure the fall time, storage time, and off time.

### Acquisition restrictions

Some signal analysis programs put additional restrictions on the waveform acquisition. For example, the processing routine may assume that some minimum amount of baseline is acquired before the rising edge of the pulse. This base is often used to determine the pulse base amplitude. The processing routine may also put other restrictions on the acquisition. These restrictions must be observed when you are acquiring the waveforms to assure accurate, repeatable results.

The processing routine described in Part II of this article assumes one division of leading baseline before the rising edge of the pulse. It also assumes that the pulse rising edge occurs within the first half (5 divisions) of the waveform and that the falling edge occurs in the second half.

In addition, the program in Part II assumes a positive going input pulse is acquired in the left channel and a negative going output pulse is acquired in the right channel.

### Reducing noise

If the entire testing environment was perfect—completely free of noise and other errors—acquisition would be simpler. Unfortunately, it's just about impossible to eliminate all noise and error sources. As a result, some special consideration must often be given to minimizing the effects of noise and other errors.

Since many of the noise sources are random with respect to the acquired pulses, signal averaging is often a good choice for reducing noise. The 7854 can signal average on-board using the AVG command. On-board averaging eliminates the time required to transfer many waveforms to the controller. It also relieves the controller's processing burden. As a result, on-board averaging is usually significantly faster than averaging the waveforms in the controller.

Signal averaging depends on the "mean zero" nature of random noise. This simply means that the average of the noise is zero. By repeatedly acquiring the waveform and averaging the acquisitions, the noise component is significantly reduced. For every power of two averages (2,4,8,16...), the noise is reduced by about 3dB.

This noise reduction does not come without cost, however. The first cost is time. If a single acquisition takes 200 milliseconds, for example, acquiring the signal 32 times for averaging takes about 6.5 seconds. At some point, an acceptable trade-off must be reached between the amount of noise in the acquired data and the time required to perform the averaging.

Another possible problem is errors resulting from pulse generator drift. Since the 7854 averages signals from one channel at a time, the averaging process must complete on one channel before starting on the next. The more averaging done, the more time between acquiring the input waveform and output waveform, and the more time there is for drift difference to accumulate between the acquired input and output waveforms.

If the generator does not drift significantly during the time required to average both signals, drift error will not be a problem. However, if the generator drift can't be easily reduced, one solution is to reduce the number of averages.

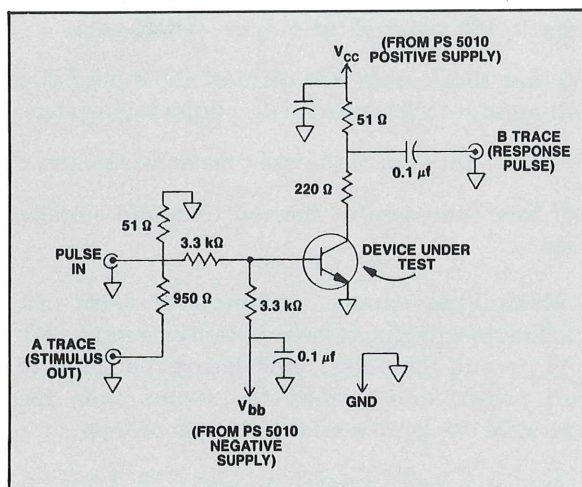
The drift error problem also means that it is usually best to acquire both waveforms before



processing either waveform. Acquiring and processing one waveform at a time saves some memory in the 7854, but it means that the acquisitions are separated by more time, and that can cause more drift error. Acquiring both waveforms before processing either one minimizes the drift error. However, for 1024-point waveforms, the 7854 requires option 2D to average and store both waveforms. Option 2D adds 4096 words of memory to expand program storage, waveform storage, and the number of constant registers.

### Test conditions

The results of the tests depend as much on the test conditions as they do on the DUT itself. Transistor manufacturers usually specify the test circuit and test conditions along with the switching parameters. A typical test circuit is shown in Fig. 5.



**Fig. 5.** A typical test circuit for testing discrete transistors.  $V_{cc}$  and  $V_{bb}$  supply voltages are programmable.


There are a number of variables in the test conditions, but the major ones are:

- Collector current
- Base current
- Input pulse width
- Base bias voltage ( $V_{bb}$ )

With a programmable system, most of these variables can be controlled in the test program so that changing the test conditions is simple. A device can even be tested under a variety of conditions with minimal operator intervention. If test results are logged, the test conditions can be included with the logged test results.

In the example system used here, the PS 5010 Programmable Power Supply provides the  $V_{cc}$  power supply, which regulates collector current. It also supplies the base bias voltage ( $V_{bb}$ ). The base current is affected by the amplitude of the stimulus pulse (provided by a PG 502 Pulse Generator) and the base bias voltage. The input pulse width is set with a control on the PG 502.


### In summary...

Accurate results from any automated test depend on proper data acquisition. The guidelines provided in this article will help you acquire data reliably. Part II of this article takes a closer look at the processing task. 

By Mark Tilden,  
HANDSHAKE Staff

## Take a few minutes to stay up to date

HANDSHAKE is published quarterly by Tektronix to help keep you up to date on signal processing and instrument control. If you are not already a subscriber, you can become one simply by filling out and mailing the postpaid reply card bound into the center of this issue. It costs you nothing but the time to fill out the card!

Be sure to fill out the reply card completely and clearly. By doing so, you'll help us to make sure that your name and address is entered correctly on the HANDSHAKE mail list. Also, we'll be able to make sure that you receive any additional literature that you might request via the HANDSHAKE reply card. 

---

# Semiconductor switch testing

## Part II—Processing the waveforms

Part I of this article discussed the first task in automatically testing semiconductor switching parameters—acquiring the switching waveforms. Once the waveforms are acquired, some processing is usually required to extract useful information such as delay time, rise time, etc. This article, Part II of the series, describes the processing task.

### On-board vs. controller processing

There are two basic tasks that can be taken to process the acquired waveforms. The digitizer or digitizing oscilloscope can transfer raw acquired waveforms to the controller for processing. New intelligent acquisition instruments like the 7854 Oscilloscope from Tektronix offer another option—processing the acquired waveforms in the instrument and sending partially processed data or final results to the controller.

This second technique, processing by the instrument, offers some significant advantages in a semiconductor switch testing system. Since the tests are often performed in a production environment, speed is important. One of the ways to improve the throughput of a system is to reduce the amount of data that must be transferred between the instruments and the controller.

Waveforms are the biggest transfer burden. A single waveform contains between 128 and 1024 points. If the waveforms are signal averaged in the controller, several waveforms must be transferred for a single acquisition. These bus transfers are byte serial and time consuming for large amounts of data.

In contrast, averaging and processing the waveforms in the acquisition instrument means that data transfers are limited to results instead of complete waveforms. For example, if a waveform is averaged 32 times and the rise time, fall time, delay time, storage time, on time, and off time are computed in the 7854, the data transfer is reduced to the six results instead of 32,768 values (thirty-two 1024-point waveforms).

On-board processing also frees the system controller for other tasks. It can analyze or log test results while the 7854 is acquiring and processing

the next test waveforms. This “distributed processing” also means that the controller could handle more than one 7854. It could process and log results from several 7854 test stations.

### Finding the answers

Remember from Part I that the parameters of interest for switching transistors are:

Rise time—from 10% to 90% points of the output rising edge.

Fall time—from 90% to 10% points of the output falling edge.

Delay time—from 10% point of input pulse leading edge to 10% point of the output leading edge.

Storage time—from 90% point of the input pulse falling edge to 90% point of the output falling edge.

On time—the sum of the rise time and delay time.

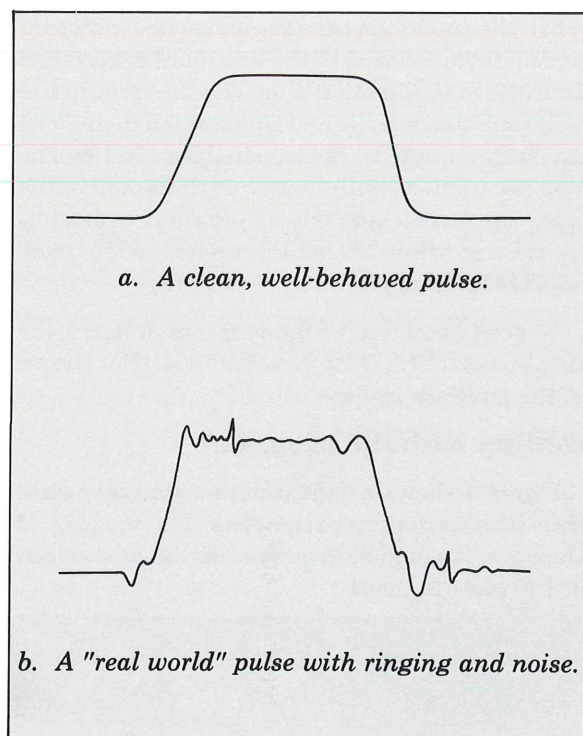
Off time—the sum of the fall time and storage time.

Measuring these parameters from an oscilloscope display amounts to locating the top and base of the pulses, calculating the 10% and 90% points, and reading the results from the graticule marks. It's a fairly simple process.

One reason it's simple is that human operators tend to be forgiving when the signals jitter or are noisy. However, automated acquisition and analysis can be considerably more accurate and repeatable. But, to achieve this accuracy and repeatability, the measurement routines must be carefully written to handle the real world—noise, jitter, and drift—that skilled operators mentally compensate for.

To illustrate this concept, consider the process of locating the 10% and 90% points on a waveform. The key is finding the top and base amplitudes of the pulse from which the 10% and 90% amplitudes are computed. If the pulse is clean with flat base and top portions displayed on screen as in Fig. 1a, the task is simple. But if the pulse has significant ringing or noise, things get a little more complex. Where are the base and top of the pulse in Fig. 1b?



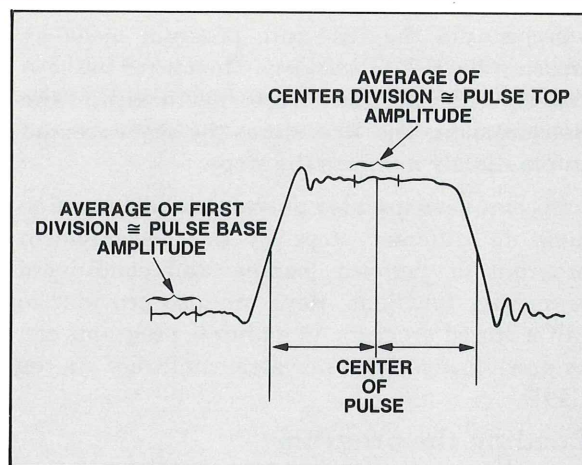


**Fig. 1.** Finding the top and base of an ideal pulse is easy. Unfortunately, noisy and ringing pulses are more common in the real world. Writing a program to find the top and base of a noisy pulse with ringing is more complicated.

In the simple case of the clean pulse shown in Fig. 1a, a program can simply find the maximum and minimum levels on the pulse. From these two points, the 10% and 90% points can be easily computed. However, if this technique is applied to the pulse in Fig. 1b, the maximum and minimum functions will return values on the noise spikes or ringing parts of the pulse which don't accurately represent the amplitude of the pulse. As a result, the 10% and 90% points will be inaccurate and the time measurements will be incorrect.

There are a variety of techniques for reliably finding the top and base of a digitized pulse with a computer. One common technique uses a histogram. The histogram routine counts how often a particular amplitude value occurs. During the flat top and base parts of a pulse, the same amplitude value recurs many times. The most prevalent values usually represent the top and base amplitudes of the pulse. This technique was described in more detail in a previous issue of HANDSHAKE (See "Some useful approaches to pulse analysis" Spring/Summer 1979 HANDSHAKE, available via the HANDSHAKE Reply Card.)

Another technique involves finding the average of a segment from the base of the pulse and an average from a segment of the pulse top. Figure 2 shows how this technique is applied to a pulse with ringing.



**Fig. 2.** One technique for finding the base and top amplitudes of the pulse involves averaging the first division and the center division of the pulse.

The position of the segments must be chosen carefully so that they contain only the top and base of the pulse—not part of the rise or fall. If the acquisition is restricted so that the waveforms have at least one division of leading base before the start of the pulse, the first division can be averaged for the base amplitude. Choosing the segment for the top amplitude is a little more complicated. A reasonably good choice for the top segment is one division centered around a point half way between rise and fall of the pulse. An average of this segment usually provides a good estimate of the pulse top amplitude. Figure 2 illustrates this.

Once the base and top amplitudes are determined, the rest is simple. The 10% and 90% points can be computed from the base and top amplitudes. Then the horizontal positions where the waveform crosses the 10% and 90% points can be found using the 7854 cursors. The time difference between these 10% and 90% points on the input and output waveforms yields the various parameters of interest.

### Implementing the 7854 program

It all sounds simple in theory, but what about actually making it work in a program?

The 7854 Programmable Oscilloscope provides an extensive set of measurement capabilities

preprogrammed into its measurement keyboard. In addition, the 7854 can store up to 920 commands (2000 with option 2D) and up to 1000 program lines in its internal program memory. And writing a 7854 program is as simple as performing the measurement from the keyboard. You just put the 7854 into program mode by pressing the PROGRAM key. Then press the keys exactly as if you were performing the measurement. The 7854 stores the sequence and automatically numbers the steps.

Special keys are also provided to allow you to jump to different steps in the measurement program or perform looping and conditional branching functions. Keys are also provided to edit a stored program. In addition, programs can be down-loaded from a system controller via the GPIB.

### Loading the program

Loading a measurement program from a GPIB controller like the Tektronix 4052 Graphic Computing System is simple, too. The controller just sends the PROGRAM command to put the 7854 in program mode and sends the program statements. The program lines are stored in a format that duplicates keyboard input. Each keyboard command is delimited by a space. For example, to set the number of points per waveform to 512, the command is transmitted to the 7854 as 5 1 2 P/W. Each statement is followed by a NEXT command to end the program line. The extra spaces and the NEXT are removed for clarity when the program is displayed on the 7854 screen. The program shown in Fig. 3 transfers a 7854 program from a tape file on the 4052 to the 7854.

```
100 PRINT "ENTER THE FILE NUMBER: ";
110 INPUT F
120 PRINT "ENTER THE GPIB ADDRESS OF THE 7854: ";
130 INPUT A
140 PRINT @A:"PROGRAM CLP NEXT"
150 FIND F
160 ON EOF (0) THEN 200
170 INPUT @33:L$
180 PRINT @A:L$
190 GO TO 170
200 PRINT @A:"EXECUTE"
210 END
```

**Fig. 3.** Loading a program into the 7854 from the 4052 tape drive requires only a simple program.

First, lines 100-130 get the GPIB address of the 7854 and the file number where the program is stored. Then, line 140 puts the 7854 in PROGRAM mode and clears any program left in the 7854 with the CLP (Clear Program) command. Line 150 finds the file on the 4052 magnetic tape drive and line 160 sets up an EOF (End Of File) condition.

When the end of the tape file is reached, program control is transferred to the line number specified (line 200) in the ON EOF line. The 7854 program is read from the tape file and transmitted to the 7854 one line at a time by the loop in lines 170-190. The loop is executed continuously until the end of the tape file is reached. Then, program control is transferred to line 200. This line puts the 7854 back in EXECUTE mode.

To start execution of the program at line 0, the 4052 sends 0 GOTO RUN to the 7854. (Not shown in the program in Fig. 3.)

### Getting with the program

Figure 4 shows a 7854 program that measures transistor switching parameters. The program is shown as it would be displayed on the 7854 screen in PROGRAM mode.

```
000 STORED TIME VECT
001 UNDL HMDA
002 1024 >P/W
003 16 AUG
004 3 >WFM
005 UNDR
006 16 AUG
007 2 >WFM
008 3 WFM 0 >WFM
009 1 LBL GSB
010 36 CNS 20 >CNS
011 37 CNS 21 >CNS
012 38 CNS 22 >CNS
013 39 CNS 23 >CNS
014 33 CNS 24 >CNS
015 2 WFM 0 >WFM 1CNS ENTER 0 WFM *
016 1 LBL GSB
017 36 CNS 25 >CNS
018 37 CNS 26 >CNS
019 38 CNS 27 >CNS
020 39 CNS 28 >CNS
021 33 CNS 29 >CNS
022 UMDALT SCOPE
023 STOP
024 L01
025 0 WFM 1 >WFM
026 0 WFM .5 SMOOTH
027 DIFF
028 CRS1 0 >HCRD
029 MAX >UCRD
030 HCRD 41 >CNS
031 MIN >UCRD
032 HCRD 43 >CNS
033 43 CNS 41 CNS - 2 / 41 CNS + >HCRD
034 46 >CNS
035 1 WFM 0 >WFM
036 HSCL .5 * 46 CNS X<Y - >HCRD
037 CRS2-1 HSCL >HCRD
038 MEAN 49 >CNS
039 CRS1 0 >HCRD CRS2-1 HSCL >HCRD
040 MEAN 31 >CNS
041 49 CNS 31 CNS - 33 >CNS
042 33 CNS .1 * 31 CNS + 34 >CNS
043 33 CNS .9 * 31 CNS + 35 >CNS
044 CRS1 0 >HCRD
045 34 CNS >UCRD
046 HCRD 36 >CNS
047 35 CNS >UCRD
048 HCRD 37 >CNS
049 46 CNS >HCRD
050 35 CNS >UCRD
051 HCRD 38 >CNS
052 34 CNS >UCRD
053 HCRD 39 >CNS
054 RTN
```

**Fig. 4.** This 7854 program acquires, averages, and processes the semiconductor switch waveforms. A positive going input pulse and negative going output pulse are assumed. Results are stored in constant registers 20-29.



This program acquires and averages the stimulus (input) waveform from the left channel and the response (output) waveform from the right channel. It assumes a positive going input pulse and a negative going output pulse. After acquiring both waveforms, the program computes the 10% and 90% points on both the rising and falling edges of the waveforms. The results are stored in constant registers for later transfer to the 4052 system controller.

The program is divided into two major parts. The first part (lines 0 through 23) acquires and averages both the stimulus and response waveforms and stores the measurement results in constant registers. The second part (lines 25 through 54) is a subroutine that is called twice, once to analyze the stimulus waveform and once to analyze the response waveform.

Line 0 begins by putting the 7854 in STORED, TIME, and VECTOR mode. STORED mode causes the 7854 to display the contents of its waveform memory instead of the real-time display. TIME means that the horizontal axis is time, instead of one waveform plotted against another. VECTOR mode causes the display to draw straight lines between the acquired waveform points. This mode **does not** add extra points to the stored waveform or store any interpolated points, it only affects the way the waveform is displayed.

The next two lines set the 7854 to Left Vertical Mode (VMDL) and Horizontal Mode A (HMDA) and set the number of points per waveform to 1024.

Line 3 begins averaging the left vertical channel (stimulus waveform) 16 times. Program execution is delayed until the average is complete. Then the averaged waveform is copied to waveform memory number 3. With the stimulus waveform acquired and stored, the vertical mode is changed to right (VMDR) in line 5 and the response waveform is averaged 16 times. The averaged response waveform is stored in waveform number 2.

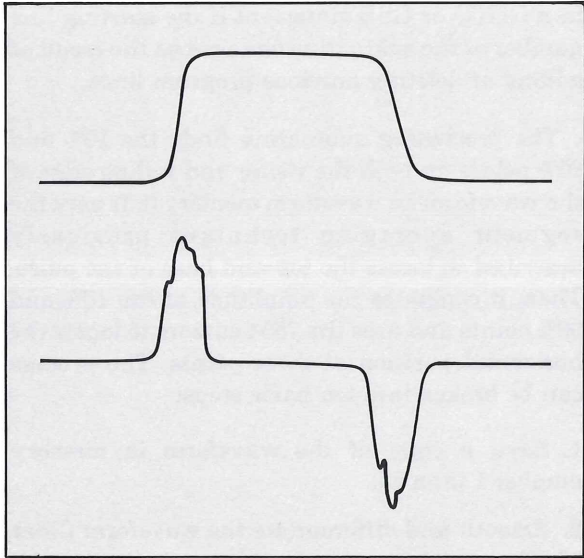
Next, the stimulus waveform is copied back into waveform 0 and the processing subroutine is called. The GSB (Go to Subroutine) statement in line 9 calls the routine by label number, not by line number. The label (L01) is shown in line 24. The label statement doesn't do anything except mark a spot for a GOTO or GSB statement. This eliminates the need for changing the line number

in a GOTO or GSB statement if the starting line number of the subroutine changes as the result of adding or deleting previous program lines.

The processing subroutine finds the 10% and 90% points on both the rising and falling edge of the waveform in waveform memory 0. It uses the segment averaging technique previously described to locate the top and base of the pulse. Then, it computes the amplitude of the 10% and 90% points and uses the 7854 cursors to locate the horizontal position of these points. The process can be broken into ten basic steps:

1. Save a copy of the waveform in memory number 1 (line 25).
2. Smooth and differentiate the waveform (lines 26-27).
3. Find the point of maximum slope on the rising edge (lines 28-30).
4. Find the point of maximum slope on the falling edge (lines 31-32).
5. Find the point half way between the rise and fall (lines 33-34).
6. Get the original waveform from memory 1 (line 35).
7. Average one division centered around the center of the pulse (lines 36-38).
8. Average the first division of the waveform (lines 39-40).
9. Compute the 10% and 90% points from the results of steps 7 and 8 (lines 41-43).
10. Find the horizontal position of the 10% and 90% points (lines 44-53).

Line 25 saves a copy of the original waveform, since the program differentiates the waveform to find the maximum slope points. Next, the program smooths the waveform to reduce high-frequency noise. This noise would show up as large spikes in the differentiation and could cause errors in finding the point of maximum slope. The smoothed waveform is differentiated to produce two spikes—one positive spike and one negative spike as shown in Fig. 5. The positive spike corresponds to the maximum slope of the original waveform. By finding the horizontal position of the maximum points on this waveform, the points of greatest slope on the rise and fall of the original waveform can be found.



**Fig. 5.** Differentiating the smoothed waveform produces two spikes—a positive one corresponding to the rising edge of the pulse and a negative one corresponding to the falling edge.

From the position of the maximums, line 35 computes the center of the pulse by finding half the difference between position of the two maximums and adding the result to the position of the first maximum.

With the position of the pulse center determined, the original waveform that was saved in memory 1 is copied back into waveform 0. Line 36 moves the first cursor to one half division before the center of the pulse. It does this by getting the horizontal scale factor (HSCL), multiplying it by 0.5 and subtracting the result from the position of the pulse center. Then, the second cursor is turned on and moved to one division beyond the position of the first cursor. The result is that the cursors delimit a segment of one division around the center of the pulse.

The MEAN command in line 38 computes an average of the points between the cursors. This value is usually a reasonably accurate representation of the pulse top amplitude.

Next, cursor 1 is moved to the beginning of the waveform and cursor 2 is moved to one division beyond the first cursor. The MEAN command is used again in line 40 to compute the average of the points delimited by the cursors. The result is the base amplitude of the pulse. Notice that this technique assumes at least one division of leading baseline is acquired before the rising edge of the pulse.

Line 41 computes the amplitude of the pulse by subtracting the base amplitude from the top amplitude. The result is stored in constant register 33. From this amplitude value, lines 42 and 43 compute the 10% and 90% amplitude values. This computation yields the amplitude of the 10% and 90% points, but it doesn't indicate where the 10% and 90% points are on the waveform. Lines 44-53 use the 7854 cursors to find the position of the points in time.

Line 44 moves cursor 1 to zero (the beginning of the waveform). Then line 45 moves the cursor to the 10% point stored in constant register 34. The >VCRD command tells the 7854 to scan the waveform starting at the current cursor position looking for the vertical value of the waveform to cross the value in the X register. In this case, the cursor is moved to the point closest to the 10% value. Line 46 gets the horizontal coordinate of this point and stores it in constant register 36.

Lines 47 and 48 do the same thing for the 90% point on the leading edge.

To find the 10% and 90% points on the falling edge, line 49 moves the cursor to the center of the waveform and repeats the process just described for the 10% and 90% points. This limits the search for the 10% and 90% levels to the last half of the waveform. The routine assumes that the falling edge occurs somewhere after the center of the screen.

The horizontal and vertical location of the 10% and 90% points on both edges are stored in constant registers. Finally, the RTN statement in line 54 ends the subroutine and returns to the line after the subroutine call (line 10).

Lines 11-14 move the results returned by the processing subroutine to constant registers 20-24 to allow the routine to be called again without destroying the results from the first call.

Line 15 gets the output waveform stored in waveform memory 2. Since the output waveform is a negative going pulse, line 15 inverts it by multiplying the waveform by -1. Then the processing subroutine is called again in line 16. The results returned from the subroutine are stored in constant registers 25-29.

At this point, the 7854 stops. Its part of the job is complete and it asserts SRQ to tell the system controller that it is done. The 7854 waits for the controller to read the results from the constant registers.



## Transferring the results

When the 7854 finishes acquiring and processing both waveforms, it asserts the SRQ line to notify the controller that it is done. The 7854 reports Operation Complete status (66 decimal) when polled by the controller. If the controller is handling several 7854 test stations, it polls each station, looking for the operation complete status. The one that reports a 66 in its status byte is the station that has results ready.

Once the controller has identified which instrument is reporting operation complete, it gets the test results by reading data from the appropriate constant registers in the 7854. The results are stored as follows:

Constant Register	Value Stored in this Register	Waveform
20	Location of 10% point on leading edge	Input
21	Location of 90% point on leading edge	Input
22	Location of 90% point on falling edge	Input
23	Location of 10% point on falling edge	Input
24	Amplitude of input pulse	Input
25	Location of 10% point on leading edge	Output
26	Location of 90% point on leading edge	Output
27	Location of 90% point on falling edge	Output
28	Location of 10% point on falling edge	Output
29	Amplitude of output pulse	Output

The 4052 program shown in Fig. 6 gets the constant register values from the 7854. First the program sets up an ON SRQ condition to handle SRQs from the 7854. When an SRQ occurs, control is passed to line 230 to poll the 7854 and get its status. Line 110 initializes the status byte to zero.

The destination array for the values read from the 7854 is dimensioned in line 120. Line 130 begins a loop that gets the values. First, the READX command is sent to the 7854 in line 140. This tells the 7854 to put the next value it receives in the X register. Line 150 delays execution of the program until the READX command is initiated and the 7854 is ready to receive the data. When the READX is initiated, the 7854 asserts SRQ and reports a status byte of 211. Until the 211 status byte is received, the program loops at line 150.

When the 7854 is ready to receive the X register value, line 160 sends the value of the index variable (I). Notice that for the first pass through the loop, I is equal to 20. Next, the program sends the CNS and SENDX command. The CNS command gets the value stored in the constant register pointed to by the value in the X register. As a result, the first pass gets the value in constant register 20. The value is placed in the X register,

```
100 ON SRQ THEN 230
110 S=0
120 DIM T(10)
130 FOR I=20 TO 29
140 PRINT @A:"READX"
150 IF S<>211 THEN 150
160 PRINT @A:I
170 PRINT @A:"CNS SENDX"
180 IF S<>210 THEN 180
190 INPUT @A:T(I-19)
200 NEXT I
210 IF S<>66 THEN 210
220 END
230 POLL D,S;A
240 RETURN
```

Fig. 6. A simple 4052 program reads the results of the processing routine from 7854 constant registers 20-29 into array T in the 4052.

and the SENDX command tells the 7854 to send the value over the GPIB.

When the SENDX command is initiated, the INPUT command in line 190 reads the value from the 7854 and stores it in the first element of array T. Then, the loop index is incremented and the loop is repeated to read constant register 21. The loop repeats until all ten values have been read and stored in array T. Then line 210 waits for the operation complete status byte (66) from the 7854 before ending the program.

## Now that you have the answer...

The 7854 program computes the position in time of the 10% and 90% points on both waveforms. It could go a step farther and compute the rise time, fall time, on time, etc. by subtracting the appropriate values. In this case, however, the 7854 just sends the points to the 4052 and the 4052 computes the final answers. For example, the rise time is computed by subtracting 10% point on the rising edge of the output waveform from the 90% point on the output waveform; the storage time is the 90% point on the falling edge of the output waveform minus the 90% point on the input falling edge.

Once the 4052 has computed the answers, there are a multitude of things you can do with the results. They can be logged on the magnetic tape drive or printed on the 4052 screen or a line printer. With a programmable power supply in the system, such as the PS 5010, the complete test conditions—Vcc, Vbb, base drive, pulse width, and collector current—can be printed with the test results to completely define the test. Figure 7 shows typical output from such a program.

Device type: 2N5769			
Ib:	1.03ma	Ic:	11.38ma
Vbb:	-1.00v	Vcc:	+3.00v
Twidth:	101.17ns		
Tdelay:	10.55ns		
Tstore:	1.6ns		
Ton:	22.08ns		
Trise:	11.53ns		
Toff:	10.6ns		
Tfall:	9ns		

Fig. 7. Typical output from the 4052/7854 semiconductor switch testing system.

The 4052 could compare the results to pre-defined test limits and produce a pass/fail result. It could even be interfaced to a parts handler to automatically sort the parts into two or more categories.

In some applications, it may be important to characterize a device under varying test

conditions. With programmable instruments, the tests can be repeatedly performed with varying test conditions and a minimum of human interaction. The results can be graphed to produce curves of switching times versus collector current or base drive, for example. In addition, the test results for a batch of parts could be statistically analyzed to determine failure rates or changes in quality control.

These are just a few examples of what could be done by building on the basic test system described in this article. The possibilities are nearly limitless. Your local Tektronix representative can get you started on the path toward an automated measurement solution.



By Mark Tilden, HANDSHAKE Staff  
with grateful acknowledgement  
to Clive Gilder of Tektronix  
for some of the groundwork that  
made this article possible.

## Getting the most out of TEK BASIC graphics—

### New ROM Pack produces high quality characters and symbols on 4052 and 4054 Graphic Systems and Plotters



Fig. 1. The 4052R11 Character and Symbol ROM Pack.

The 4052R11 Character and Symbol ROM Pack (Fig. 1) brings high quality stroked characters and symbols to the 4052 and 4054 Graphic Systems and 4662 and 4663 Plotters. It adds 27 commands to 4050-Series BASIC to display alphanumeric characters and symbols, draw smoothed arcs and circles, generate custom symbols, and provide additional graphics enhancements. Character and symbol sizing, spacing, proportioning, rotation angle, degree of slant, and smoothness are easily programmed.

Over 100 different characters and symbols are contained in the 4052R11 ROM Pack (Fig. 2) including upper- and lower-case letters for both the English and Greek alphabets, mathematical and special symbols, and characters for Swedish, German, Spanish and Danish languages.



UPPER and lower Case: VERTICAL LABELS

English: ABCD...xyz

Greek: ΨΦΣΤ...ψμπα

Numbers: 0 1 2 3 4 5 6 7 8 9

Centered Symbols: □ ◊ ▲ + ◊ ✕ × ◊ = +

Math Symbols: ∫ ∴ ∃ < > ∞ ° ≠ ≤ ≥

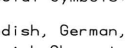
Special Symbols: {} [] ! # \$ % & © ø

Swedish, German, Spanish and Danish Characters: Å Ö Å ø ä Ü Ñ & Æ

Slanting: *Left and Right*


Over 100 Symbols, ANY SIZE.

Adds Dynamic Graphics Functions



Rotated to Any Angle

Arcs



Circles

the center of the arc and the angle from the current cursor position.

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

COMPOUND NAME: *p*-METHOXY- $\alpha$ -METHYLBENZYL ALCOHOL  
 MOLECULAR FORMULA:  $C_9H_{12}O_2$   
 MOLECULAR WEIGHT: 152.20

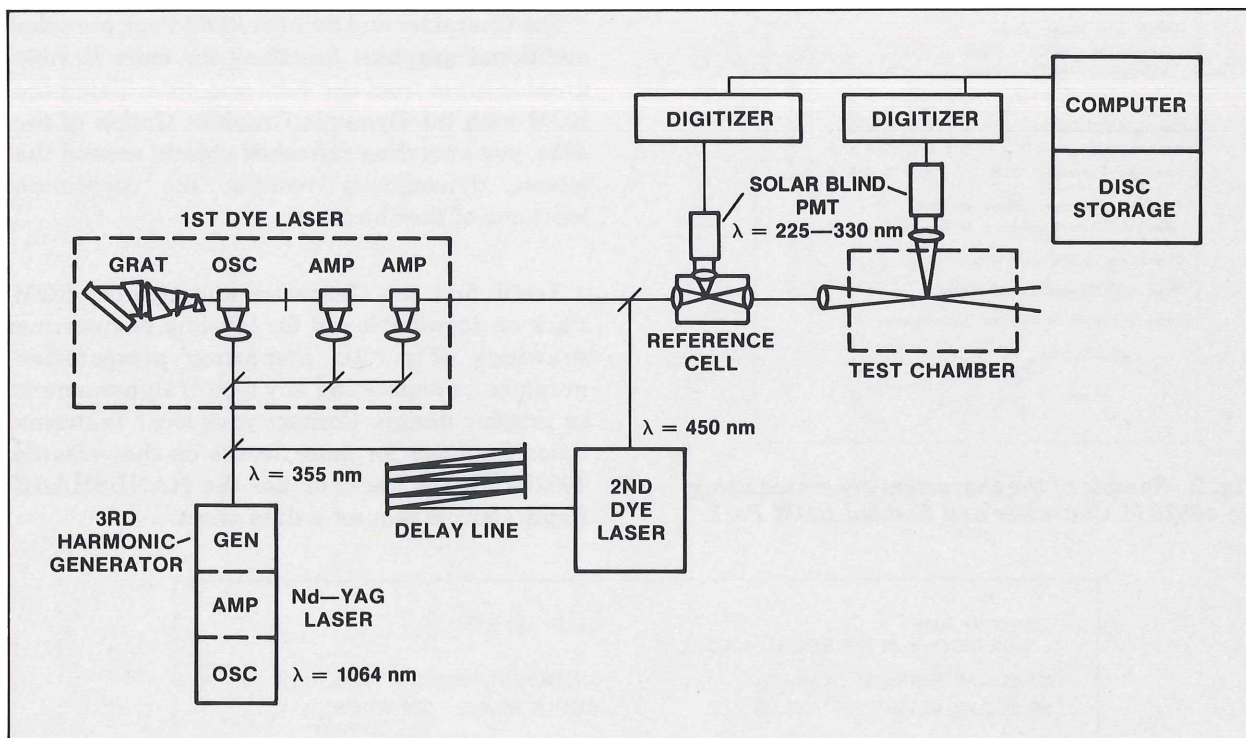
DATE: JULY 31  
 INSTRUMENT: BRAND X MODEL 123  
 SAMPLE SOURCE: DEPARTMENT A  
 XYZ COMPANY

INFRARED SPECTROSCOPY



page 21

# 7912AD assists in measuring temperature fluctuations in cold turbulent flows



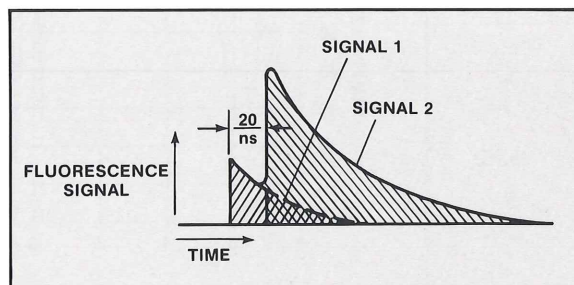
**Fig. 1.** Dual-excitation laser system using two Tektronix 7912AD Programmable Digitizers for waveform capture. (Reproduced by permission from *Applied Optics*, Vol. 20, No. 12, 15 June 1981, page 2156.)

In their paper entitled "Two-photon excitation of nitric oxide fluorescence as a temperature indicator in unsteady gasdynamic processes" (*Applied Optics*, Vol. 20, No. 12, 15 June 1981, page 2153), R.L. McKenzie and K.P. Gross describe a technique for resolving temperatures below 300 Kelvin with signal-to-noise ratios greater than 50 to 1.

The data for analysis is acquired via the test set up shown in Fig. 1. Two waveform digitizers are used to digitize the fluorescence signals. Since the fluorescence signals are fast transients, digitizing must be done single shot and at a very fast sampling rate. To meet the speed requirements, Tektronix 7912AD Programmable Digitizers are used.

Once the data is digitized and transferred into the computer, the basic analysis technique begins with fitting a synthetic curve to the dual-excitation fluorescence waveform (Fig. 2) using a least squares technique. The two curves are then

deconvolved for the areas under each, these areas being a measure of the energy contained. Ratioing the two areas gives the temperature of the gas.




**Fig. 2.** Dual-excitation fluorescence signal. (Reproduced by permission from *Applied Optics*, Vol. 20, No. 12, 15 June 1981, page 2155.)

McKenzie and Gross present the technique in their paper as an effective and practical means of measuring relative rotational state populations using two-photon excitation of nitric oxide and its



subsequent fluorescence. Nitric oxide is favored because it is rotationally simple and is present in many combustion processes. It may also be used at very low concentrations as a seed gas in cold flows and, as documented in the paper, still provide adequate signals for temperature measurements.

Further information on the technique is available from the authors. Robert L. McKenzie is with NASA Ames Research Center, Moffett Field, California 94035. K.P. Gross is with Stanford University, Department of Aeronautical and Astronomical Engineering, Stanford, California 94035. 

---

## New ROM packs enhance GPIB capabilities of 4050-Series Controllers



Thirty-nine new commands are now available for more extensive and simpler GPIB instrument control with the Tektronix 4050-Series Controllers. The 4051R14 GPIB Enhancement ROM Pack for 4051 Controllers and the 4052R14 Enhancement ROM Pack for the 4052 and 4054 Controllers allow easy access to standard IEEE-488 interface functions and provide general purpose utility routines for easier, more efficient GPIB programming.


With the new GPIB enhancement ROM Packs, CALLs for GPIB functions can now use common names such as 'IFC' for Interface Clear and 'SDC' for Selected Device Clear. Instruments may also now be assigned as talkers or listeners by use of the CALL 'TALK' or CALL 'LISTEN' commands. These and other functions previously had to be

implemented by sending ASCII-decimal equivalents with the write byte (WB) statement. The new GPIB Enhancement ROM Packs have simplified all of that.

Also, the polling features of the 4050-Series Controllers have been enhanced by the new ROM packs. A time-out parameter has been added to the serial poll routine. This prevents unexpected bus conflicts caused by a device not responding to a poll request. In addition, SRQ interrupt recognition can be disabled and enabled under program control. And a parallel poll capability is also now available via the GPIB Enhancement ROMs.

Another new routine, CONFIG, automatically determines what devices are on the bus and returns their addresses (by primary address, with secondary addresses being returned as an option). With this capability, programs can now be written to check system configuration before going into data acquisition or transmission. If the configuration isn't right, the program can turn to operator prompts for correcting the configuration.

Other general utility functions provided by the 4051R14 and 4052R14 allow increased control of tape file processing (including named tape files), bit manipulation, and binary data transmission. The variables used in a BASIC program can now also be listed.

To find out more about the 4051R14/4052R14 GPIB Enhancement ROM Packs, contact your local Tektronix Sales Engineer or Sales Representative. Data sheets are available simply by checking the appropriate box on the HANDSHAKE Reply Card in this issue. 

---

# Tektronix Programmable Systems Workshops

A series of workshops on programmable measurement systems is being made available by Tektronix at key locations throughout the continental United States. Through a balanced program of classroom lecture and hands-on laboratory exercises, participants will be able to quickly grasp basic system concepts and build system operating skills.


Included in the Programmable System series are the—

Tektronix SPS BASIC Workshop,  
5 days in length

Tektronix Waveform Processing Workshop,  
5 days in length

Tektronix 7854 Level I Workshop,  
2 days in length


Since specific systems are used in each workshop, the training is particularly valuable to new owners of Tektronix systems. But each workshop is also a quick and economical means of gaining practical experience and knowledge prior to making an equipment purchase decision.

To find out more about workshop content, schedules, and tuition, use the reply card in this issue of HANDSHAKE to request a catalog for the Tektronix Programmable Systems Workshops. Outside of the United States, please contact a Tektronix Sales Representative for information on systems training and workshops available in your country. 

---

## Program library offers measurement solutions, programming examples

A variety of programs are available at no cost from the HANDSHAKE Applications Program Library. The programs are written in various Tektronix signal processing and instrument control languages, which are mostly enhanced BASIC. Application areas covered vary widely but include waveform graphics, automated calibration, frequency domain analysis, pulse analysis, and many others. All programs are provided free of charge as hard copy listings.

Available programs are described in the HANDSHAKE Applications Program Library Catalog. The catalog also includes a convenient program order form. The June edition of the HANDSHAKE Applications Program Library Catalog contains 20 new programs. For a copy, check the appropriate box on the HANDSHAKE reply card in this issue. 

---

**HANDSHAKE**  
Group 157 (54-016)  
Tektronix, Inc.  
P.O. Box 500  
Beaverton, Oregon 97077

BULK RATE U.S. POSTAGE PAID Tektronix, Inc.
------------------------------------------------------

**Tektronix**  
COMMITTED TO EXCELLENCE

58-741  
MORRIS ENGELSON  
HANDSHAKE

33A-5039