TEKTEST™ III

AUTOMATED TEST SYSTEM CONTROL THROUGH SOFTWARE

by W. E. Kehret Automated Systems Engineering

TEKTRONIX AND AUTOMATED TEST SYSTEMS

A FAMILY HERITAGE: The S-3260 Automated Test System is the most recent addition to the TEKTRONIX family of test systems which began in 1964. The first system, and much of the later development was based on programmable sampling oscilloscopes and digital readouts. In contrast, the S-3260 is a real-time test system which uses single-shot measurement techniques to replace the sequential sampling techniques of the earlier systems. System operations are performed under software control of a dedicated computer. A graphic computer terminal handles communication with the system.

As is true of most technologies, automated test systems have made rather sweeping changes since their inception a little more than a decade ago. Although these early systems used all of the technologies available to them, their performance appears very limited as we look back from the vantage point of more than ten years of technological advancement. However limited they may appear in hindsight, these early systems adequately made the measurements required of them, and only development of more sophisticated devices to be tested placed the capabilities of these systems in question. Most tests were fixed by the original system design; if test parameters could be changed, it was usually through internal wiring changes or by front-panel switching.

As test systems advanced to include stored program control and the capability to test more complex components such as integrated circuits, the major criterion of system performance was still the hardware—what instruments were used and how they were physically connected together. Major changes in test characteristics often required reconfiguration of the system.

Implementation of large-scale-integration techniques by semiconductor manufacturers brought about the need for a much more flexible test system. The vast number of checks required on each device made testing with existing systems unfeasible, both in view of limitations of the equipment and limitations in the ability to physically program all of these tests. For relief the systems designers looked to the computer's ability to control many operations in real time (as they occur) and to ease the burden of programming.

Cover: *Sixty-four Input/Output sector cards form the heart of the S-3260 Automated Test System. The Device Under Test mounts on a test board in the center of this array, placing it very close to the measurement subsystems.*

Addition of the computer to the test system shifted the focal point of system flexibility from the hardware configuration to the software operating system. In order to achieve a flexible system, the test language or software must also be flexible. Designers of the new TEKTRONIX S-3260 Automated Test System sought a test language which could be easily understood by engineers relatively untrained in computer programming methods, yet was powerful enough to control the full range of hardware capability in the system.

A New Language

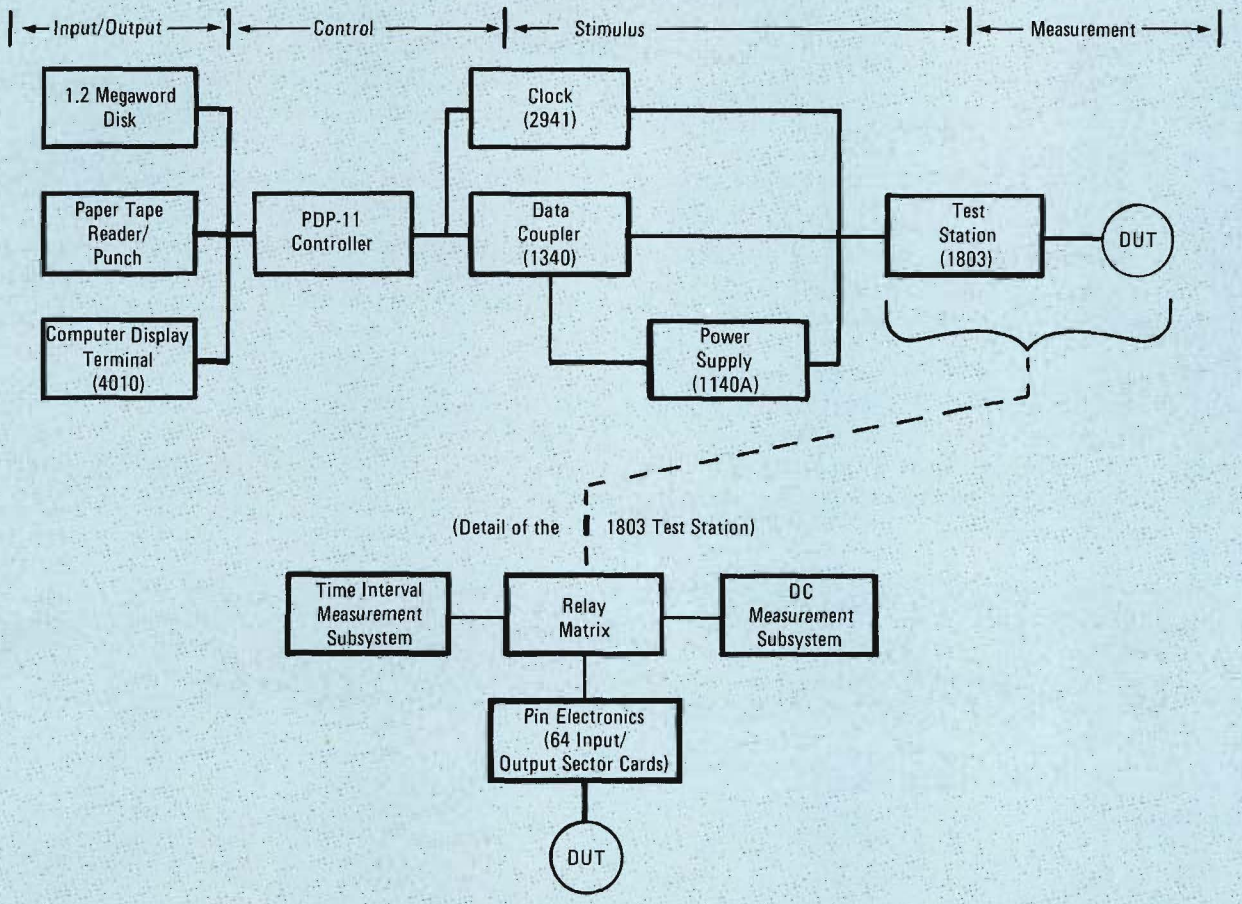
Existing test languages proved inadequate to meet the goals set forth for the S-3260 System. As a result, a totally new test language called TEKTEST III was developed.

For those familiar with computer operating systems, it is appropriate to distinguish TEKTEST III from existing compilers such as FORTRAN, ALGOL, or BASIC. While TEKTEST III is a higher level language with capabilities in some ways similar to the above mentioned compilers, this new software language for the S-3260 System is not equivalent to, or a subset of, any previously defined language. This is also true of other software components including the editors and the disk operating system. This totally new software system was developed to allow ease of operation for users other than systems specialists, and for speed of operation. Although it would appear that adaptation of existing software systems would be the easiest, this could not be done and still fully meet the objectives.

Software + Hardware = S-3260 System

While we will be dealing mainly with the Software Operating System, a basic understanding of the hardware used in the S-3260 Automated Test System is helpful. Please refer to the S-3260 System Hardware Organization discussion and Fig. 1 which accompanies this article.

S-3260 SYSTEM HARDWARE ORGANIZATION



The hardware which makes up the S-3260 System is shown in the block diagram of Fig. 1. Although a variety of options are available for this system, we will only look at those items which are a standard part of the system.

Input and output for the system is shown at the left side of the block diagram. The 1.2 Megaword Disk provides storage for the test program library, system programs, test results, files, etc. The disk system provides fast retrieval of stored material when addressed by the computer. Program material can be entered into the system through either the Paper Tape Reader Punch or the Computer Display Terminal. These devices can also be used for output. The computer display terminal is of particular importance when used to generate test programs in conjunction with the interactive programming capabilities of the Software Operating System. Since the terminal has both alphanumeric and graphic output capabilities, it can provide output in either form.

The entire system is under control of a dedicated computer referred to as the Controller. The Data Coupler interfaces between the Controller and other instruments in the system. Timing relationships for system operations are established by the Clock. Reference voltage levels are provided by the programmable Power Supply.

Interfacing for input or output to as many as 64 pins of the device under test (DUT) is provided by the Test Station. The Test Station contains 64 Input/Output Sector Cards which can be connected to the pins of the DUT through software control. Each card contains the forcing and sensing circuitry required for Functional measurements. Timing for these measurements is controlled by the Clock. The Sector Cards also contain a 1032-bit high-speed buffer memory. During functional tests, data is available from this buffer at Clock rates up to 20 MHz. Errors may also be stored in buffer memory at Clock rates.

In addition to measurements using the Sector Cards, the Relay Matrix allows each pin of the DUT to be connected to the parametric measurement subsystem. This subsystem consists of two separate measurement systems: the Time Interval Measurement Subsystem for timing measurements and the DC Measurement Subsystem for current or voltage measurements.

***Fig. 1.** Basic block diagram showing hardware organization of the S-3260 System.*

Putting It All Together

To complement the very effective hardware organization, the S-3260 has an equally effective Software Operating System (see Fig. 2). This system can be broken down into three levels: Editor Level, Translator Level, and Machine Level. The Editor Level provides three separate editors to aid in the preparation of the test program (see The Editors for further information). This test program is not in the language most readily understood by the machine environment of the test system. The process of converting the source language to a machine language is called "translation" and is handled at the Translator Level. The translated code consists of macro-instructions which are processed under software control of a Digital Equipment Corporation PDP-11/40 and its floating-point hardware and software service routines. These macro-instructions relate to the S-3260 hardware functions at the Machine Level. These include clock generator instructions, power supply instructions, data coupler instructions, and parametric subsystem instructions.

System operations are classified by priority: background operations and foreground operations. Fig. 2 shows this breakdown. Foreground operations deal with the actual running of the test program. These operations occur at the Machine Level and are given priority by the Controller. Normally, pauses occur in the foreground operations as they are being run. These pauses are caused by reed switch settling time (normally 1 to 2 milliseconds), power supply slewing, etc. Rather than remaining idle during these pauses, the computer processes background operations if they have been scheduled by the operator. Background operations can consist of test program editing, data logging, graphing, etc. This relationship between foreground and background operations allows a form of time-sharing of the computers capabilities for more effective use of the total system. For example, a new program can be generated from the input terminal at the same time that devices are being tested. Even though the test receives priority scheduling, the programmer will probably not realize that he is not receiving the full attention of the computer.

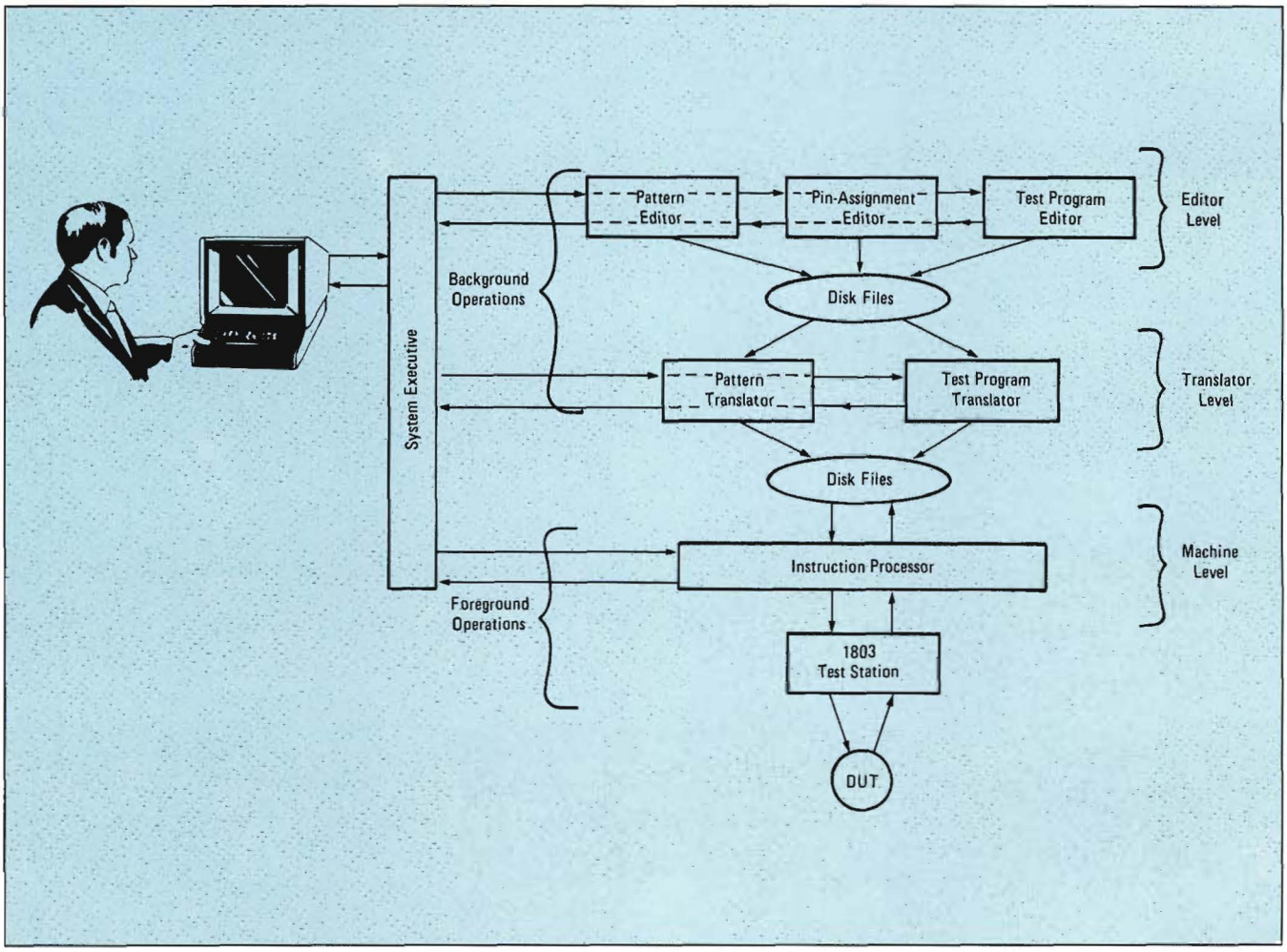
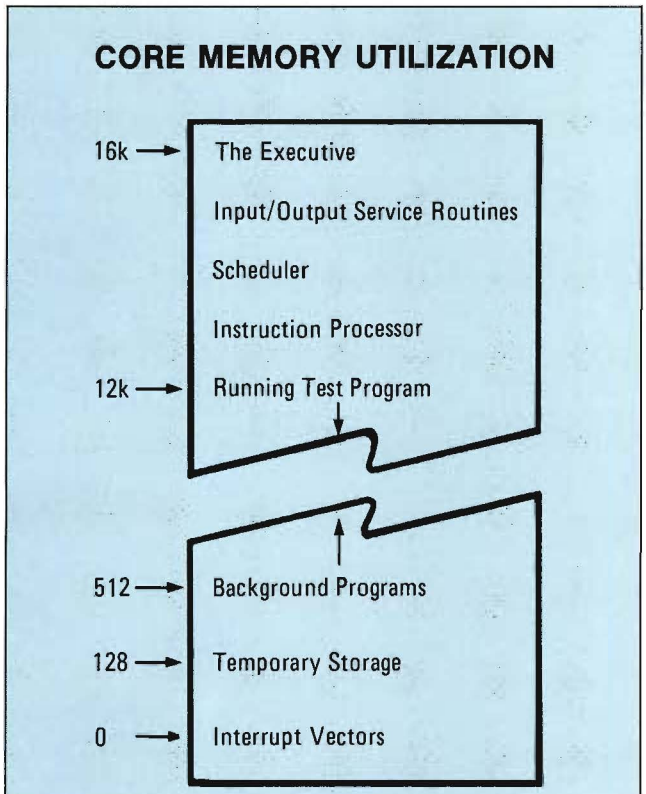


Fig. 2. *Organization of the Software Operating System for the S-3260 System.*

By this time you may be concerned about getting it all together: editors, translators, background, foreground, and who knows what else? Fortunately, there's help built into the Software Operating System. This comes from the Executive. The Executive is a disk operating system which does file handling, program loading, background scheduling, and accepts data logging directives at test-program run time (see special description on Core Memory Utilization and Fig. 3).



Although the user does not need to be concerned about core-memory utilization when operating this system, an understanding of how the software operating system and the core memory inter-relate is helpful in understanding the overall system.

Fig. 3 is a graphic representation of how the 16K (16,000) words of core memory are used in running a program. The top 4000 words of memory between memory location 16K and 12K are occupied by the Executive, input/output service routines, the scheduler, and the instruction processor, in that order. The running test program is loaded downward from the instruction processor.

From the bottom of core, the first 128 words are occupied by interrupt vectors. The temporary storage occupies the area between the 128th and 512th words. Background programs are loaded upwards from the 512th word. The point at which the downward loaded test program meets the upward loaded background programs is controlled by the scheduler and the loader.

Fig. 3. *Graphic representation of core memory utilization by the computer.*

The Editors

The Editors consist of three separate editors to aid in test program preparation.

Test Program Editor. The Test Program Editor aids in the preparation of the test statements. This editor is line number oriented. To enter a statement, type a line number and the test command. For example:

```
 99.00 WHEN ERROR     110
100.00 MOVE REGISTER (1,16) TO SECTOR
       ON   PINLIST
```

Additional statements can be added between two sequential statements as follows:

```
 99.00 WHEN ERROR     110
100.00 MOVE REGISTER (1,16) TO SECTOR
       ON   PINLIST
 99.01 LOOP 100   IND=1,60
 99.02 HIDRIVE=5.0 V—50 mV* IND ON DPINS
```

Statements 99.01 and 99.02 would appear in the program between 99.00 and 100.00. If the Editor was asked to list the program, it would appear in the correct line number sequence.

The Test Program Editor also allows statements to be erased either singly or in blocks, and replacement of individual lines by typing the original line number and the new command statement.

Certain programming errors are detected and may be corrected at the time the program is entered. Other program errors are not detected until the program is actually translated or run. An example of the latter would be a command to transfer program control to a non-existent line location in the test program. When this is encountered in translation, an error statement is printed along with the line number of the command statement where the error occurred. Then the programmer can enter the correct information from the terminal.

Another feature of the Test Program Editor is to list either the entire program or individual lines within the program. This allows the programmer to inspect or modify the program.

Pattern Editor. Patterns refer to the sequence of addresses and/or control data applied for functional testing of a device. A variety of patterns are in common usage, each one best suited to test particular types of devices.[1] The purpose of the Pattern Editor is to format the test pattern data into a two-dimensional matrix or table. A pattern-table row corresponds to data sent to the DUT; one row for each cycle of a clock sequence. A pattern-table column corresponds to the sequence of data at a particular pin of the DUT; one column for each active pin.

[1]For further information on patterns, refer to TEKTRONIX Automated Systems Application Note No. 2, "Test Patterns and Their Use In LSI Memory Diagnostics."

For simple devices, the test pattern can be developed manually. However, as the device becomes more complex, the difficulties in generation of the test pattern increase similarly. Through the use of the Pattern Editor and the interactive programming features of TEKTEST III, the Controller can be used to help develop the pattern according to a given algorithm or established logic sequence. If a pattern can be described by a closed algorithm, the test pattern can be generated under software control with the Algorithmic Program Generator.

Pin Assignment Editor. The pins of the device under test are referred to by a reference pin name and coordinated with an Input/Output Sector Card of the tester by program statements. The Pin Assignment Editor assists the programmer in preparing this list. Basic operation of this editor is similar to the Test Program Editor.

The Man/Machine Interface

The S-3260 System includes a Graphic Display Terminal both for input of programs or test data and for output of test results. The Executive portion of the software operating system serves as a coordinator between the input from the terminal and the rest of the system. Since the Executive is a disk-file based system, its operation is very fast. This speed becomes important in the interactive program preparation capability of the Software Operating System.

A major feature of interactive program preparation is on-line editing. Through the Executive, the operator communicates with the software system and receives quick feedback in the form of prompts and error messages. Several interactive loops are provided in the system for on-line editing (see Fig. 4). For example, assume that the programmer enters a program statement. If this statement contains certain errors, an error message is printed on the terminal (Loop 1) even before the programmer can move his fingers to type the next part of the program. This quick feedback allows the operator to make the correction immediately rather than repeating this same error throughout a program. As a result, considerable time is saved in test program preparation compared to off-line program preparation.

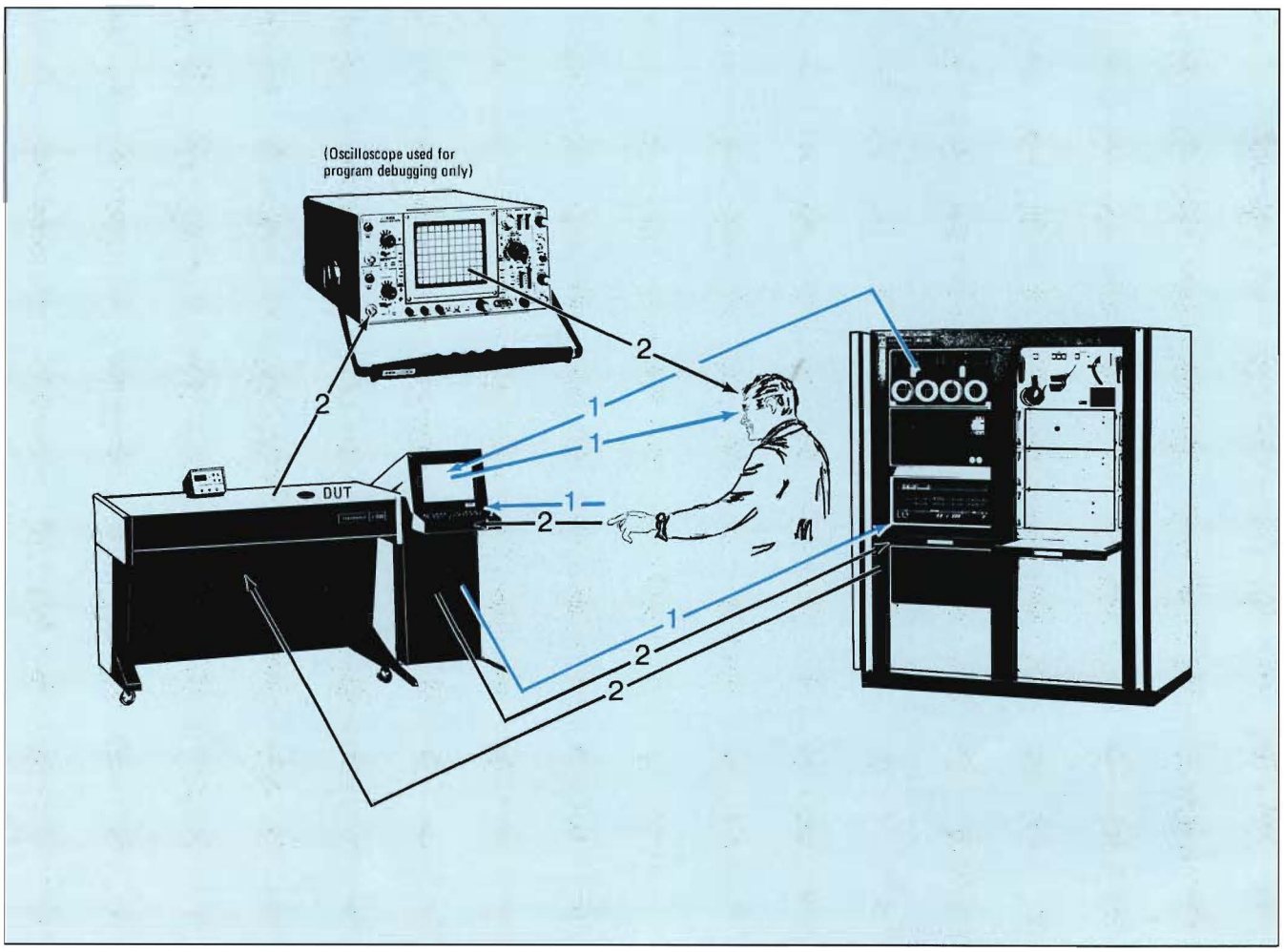


Fig. 4. The Man/Machine interface provided by the interactive program capabilities of TEKTEST III.

A second kind of interaction involves the operator more directly in test program debugging (Loop 2). As part of the system hardware, a buffer channel is provided so that any pin of the device under test (DUT) can be monitored with a test oscilloscope. A trigger output is also provided by the clock generator. This trigger can be programmed to allow examination of an individual data pattern within a functional sequence by the oscilloscope. In this way, the operator is actively involved in the process of program development and debugging.

Command Statements

Command statements consist of three elements; line no., command name, and command argument. A typical example is:

39.00	COMPARE	PINLST WITH TABL99
Line No.	*Command name*	*Command argument*

The line no. identifies the correct sequence of a command statement in the program.

The command name defines the type of operation to be performed.

The command argument can consist of symbols of up to six characters, constants, or arithmetic expressions. The type of command used determines the format of the command argument.

For other examples of typical command statements, see the discussion on the Test Program Editor.

Functional Testing

The concept of functional testing with an automated test system is similar to testing with discrete instrumentation (see Fig. 5). The functional test system consists of three parts: the generator or stimulus, the unit or device under test, and an observing or measuring instrument.

It should be noted that for functional testing with the S-3260 System, the device under test is measured each time the stimulus is changed. This form of testing is often called real-time testing or single-shot testing and should be distinguished from repetitive stimulation required by sampling oscilloscope based systems. Let's examine the three parts of the functional test system for the S-3260 as they relate to the software operating system.

Parameters of the Stimulus. The stimulus source must be characterized by the software in time and amplitude. When sequential or memory devices are tested, the time/amplitude must also be specified relative to a clock cycle. When the device under test contains memory, the input patterns must be applied in specific order to carry the device from one state to another. If memoryless, the input patterns can be applied in any order.

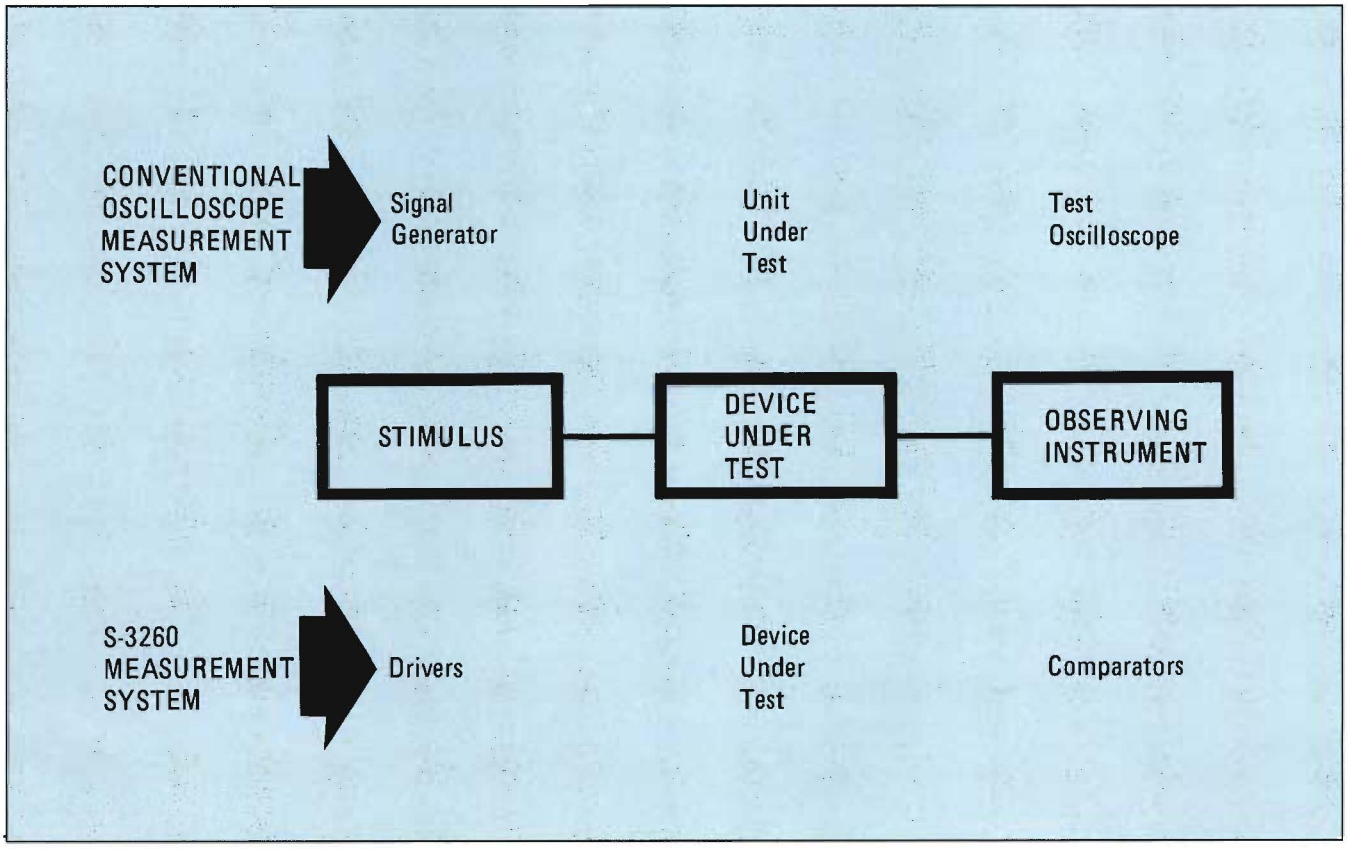


Fig. 5. *Comparison between S-3260 functional testing system and a conventional oscilloscope measurement system.*

For logic devices the purpose of functional testing is to determine that all meaningful logic states can be reached according to the specified transition rules and that all meaningful inputs produce the desired results.

Environment of the Device Under Test. Interface to the device under test (DUT), can be conveniently broken down into two categories: input and output. The input functions can all be operated through the drivers located in the test station in close proximity to the DUT. This includes input data, address, control, and supply pins.

For functional tests, the output pins are connected to the comparators.

Nature of the Measurement Instrument. The measurement instruments for functional testing are comparators at each Sector Card which make a decision on the basis of expected data and time and amplitude references. The result of the functional test is a binary decision. This decision is communicated to the system Controller which determines the resultant action to be taken in the test program.

Parametric Testing

In parametric testing, current or voltage values or a time interval are measured. Current and voltage measurements can be made separately or while forcing, respectively, a voltage or a current.

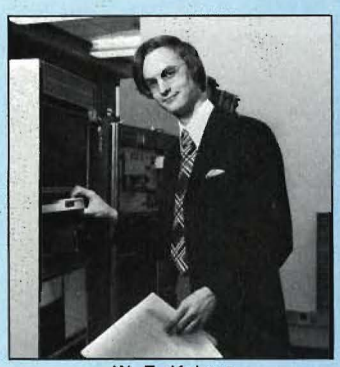
The program for a voltage measurement must specify two pins of the device under test, one of which may be ground. Current measurements must specify the pin of the DUT and the voltage supply between which the current is measured.

Time intervals are measured with the aid of the comparators. When the voltage at a designated start pin passes through a specified reference level, the time-interval measurement begins. It is stopped by a specified transition through a reference level at the stop comparator pin.

The result of a parametric measurement does not have a direct binary decision value. The result of the measurement is an analog quantity which must be compared against specified limits of the test program to diagnose the condition of the DUT. Measured values may be used as arguments in other commands, entered into arithmetic expressions, and be printed or logged as specified by the program.

SUMMARY

In this article, we have seen only a brief over-view of TEKTEST III—the Software Operating System for the TEKTRONIX S-3260 Automated Test System. The real power and flexibility of this new Software Operating System and automated test system cannot be presented within the context of these few pages.[2] What we have seen is that TEKTEST III is a new programming language, powerful enough to control the full range of hardware capability in the S-3260 System, yet easily understood by the systems engineer who may be relatively untrained in computer programming methods. As such, TEKTEST III can become a useful, flexible tool for your automated testing applications.



W. E. Kehret

OUR AUTHOR

Bill joined TEK in 1966 after receiving a B.A. in Physics from The College of Wooster. He has continued his education with graduate study in Electrical Engineering at Oregon State University and is currently participating in a Systems Science Ph. D. program at Portland State University. Prior to his present work in the Automated Systems Group, Bill worked as design engineer in the Sampling and Digital Instruments group and Advanced Product Development.

As leisure-time activities, Bill and his wife Bonnie enjoy sailing, skiing, and tennis.

[2]For a complete discussion of TEKTEST III, request a copy of "S-3260 Automated Test System Control Through TEKTEST III Software" from Automated Systems, Tektronix, Inc., Beaverton, OR.