AUTOMATIC TEST EQUIPMENT

Automatic test equipment (ATE) embraces a very broad array of test equipment ranging from hand-held meters to multiple racks of sophisticated instrumentation and *unit under test* (UUT) handling equipment all controlled by test programs. Automatic test equipment is applicable to the full spectrum from invisible light to direct current (dc). It involves an approach to testing rather than any particular configuration. Thus this treatise addresses the principles and philosophies of automatic testing rather than specific designs of ATE. Various ATEs have incorporated just about every imaginable variety of instrument in existence. Design parameters for instruments used in ATE vary little from those of stand-alone instruments other than in their method of control. They follow industry developments, becoming smaller, more complex, and requiring less power as the years pass. Thus, examining the attributes of such instruments would be far beyond the scope of this chapter. Specifics of instrument design are covered in many other sources, including other chapters of this encyclopedia.

EARLY HISTORY OF AUTOMATIC TEST EQUIPMENT

The decade of the 1960s spawned a new class of test equipment that had its roots in the prototypes of the late 1950s. Known as ATE, these testers used the programmable feature of digital computers to store "intelligence" pertaining to a testing procedure or process. This stored program, coupled to a variety of programmable instruments, provided a capability to perform a wide variety of complex tests on a device or UUT with a minimum of operator knowledge or intervention. Automatic test equipment was first developed and introduced to meet field maintenance needs of the military.

By the mid-1960s, military electronics equipment and systems had increased in complexity to a point where well-trained technicians were required to maintain them. In the Navy, for example, the Class A electronics school course was 42 weeks long. Additional schooling was required to qualify a technician in specialized equipment. By the time a technician was adequately trained and gained a year or two of field experience, the enlistment was over. Most chose to accept a discharge rather than reenlist, because civilian work was less demanding and paid better. Thus the military faced a real problem in meeting the maintenance needs of ever increasingly complex electronics equipment and weapons.

Automatic test equipment offered a solution to the military maintenance problem. Accordingly, the army pioneered the development of ATE by funding prototype designs of ATE. One of the first testers was designed to test Atlas missiles. Other testers followed for communications equipment and advanced missile system components and subassemblies. By the late 1960s military applications of ATE abounded. The thrust then was to build very capable and therefore large, complex ATE with a very wide variety of testing devices (often referred to as building blocks) capable of testing almost anything in the current electronics inventory and even provide for testing systems yet to be developed. This multipurpose test equipment became systems with up to a dozen 5 ft racks of electronics and costing several million dollars per copy. Still they were called simply, ATE.

The promises of military ATE technology were fantastic. Suppliers claimed that an unskilled person with only a couple of weeks of training could now do the work of a skilled technician with several years of training and experience. This, of course, greatly reduced the cost of maintenance over manual testing procedures.

What vendors failed to mention was the enormous cost of developing a set of test programs and interface devices required to test that wide variety of UUTs. A test program had to incorporate a complete qualification testing process (referred to as the *go chain*) as well as a complex network of fault isolation tests to diagnose the UUT when any go-chain test failed. This is analogous to anticipating all probability of failures during the lifecycle of a product and incorporating corrective actions even though in reality many of the failure modes might never be experienced. It is like buying a very expensive insurance policy and paying for it all at once rather than by modest premiums. It soon became evident that the claim of reduced cost of testing, at least during the early life of a product, was a myth; ATE was not nearly as cheap as believed. Yet, while not being as simple to use as claimed, it did greatly expand the capability of a trained technician to perform tests much more complicated than typical skills and time permitted. Thus ATE, in spite of its cost and impact on logistics requirements, became a necessary evil for maintaining complex military systems—ATE remains the backbone of modern weapon systems support.

In the 1970s ATE found a niche in the commercial manufacturing world. Here the challenge was not so much to test a wide diversity of complex assemblies, but rather to test large quantities of electronic components such as transistors and memory devices very rapidly and cheaply by minimizing costs and increasing manufacturing yield. As components increased in complexity ATE became an essential part of the manufacturing process.

For many years maintenance testing remained in the military domain, while manufacturing testing dominated the commercial world. Maintenance testing approaches the problem *top down,* starting at the system level and progressing to major assemblies, subassemblies, modules, and finally to the component level. Manufacturing testing generally works from the *bottom up,* beginning with basic materials and progressing to the finished product. Because of these basic differences and very different testing philosophies, there has been little cross-pollination between maintenance and manufacturing testing. Yet there is much to be gained from the common experiences of these seemingly independent areas of interest, because, broadly speaking, they do share a single technology. The basic principles apply to both maintenance testing and manufacturing testing. Differences and similarities are pointed out as applicable.

MAINTENANCE VERSUS MANUFACTURING TESTING

The basic principles of ATE and functional elements as described apply generally to both maintenance and manufactur-

Table 1. Comparison of Manufacturing and Maintenance Testing

	Manufacturing Test	Maintenance Test
First applications	Late 1960s	Late 1950s
Product perspectives	Speed	Integrity
	Cost to test	Life cycle cost
	Cost to produce	Cost of ownership
	Marketability	Reliability
Product mix	Large quantities	Small quantities
	Small variety	Large variety
	Inexpensive	Very expensive
Operating environment	Factory	Field
Test philosophy	Test to sell	Test to repair
	Discard	Salvage and repair
	Stand alone	Hierarchical
	Structural	Functional
Test logic	Bottom up	Top down
	Component to unit	Unit to component
Failure modes	Multiple	Single
	Out of tolerance	Catastrophic
Test techniques	Boundary scan	Logic tree
	Iddq	Guided probe
	Pseudorandom inputs	Truth table
ATE hardware	Standard commercial	Custom
Test languages	C, C++, custom	ATLAS, BASIC
Compilers	Incremental	Full pass
Test specification	Performance parameters original	Test requirements traceable
Operators	Engineers	Technicians
Documentation	Listings	Flow charts
		User instructions
Validation	Compare golden unit	Insert faults
UUT handlers	Automated	Manual
Common needs:	Verification, validation	
Common tools:	Test design tools, compilers, simulators, automatic test generators, software support stations	

ing testing. The greatest differences lie in the testing philosophies and the design of the ATE hardware. Maintenance testing assumes that the UUT has been properly manufactured, and it once operated satisfactorily so the objective is to return the product to its original operating condition. Manufacturing testing is designed to assure that the product is properly assembled using tested components and that it is ready for packaging and sale. The failure modes of these differing processes are very different. The organization of the ATE hardware is also very different. Maintenance ATEs are durably constructed and fairly simple to operate in a field environment. Manufacturing ATEs are designed to be fast, flexible, reconfigurable, fully automatic, and suitable for factory environments. The most salient differences between maintenance and manufacturing ATE are identified in Table 1.

BASIC PRINCIPLES OF AUTOMATIC TEST EQUIPMENT

Fundamentally any combination of instrumentation controllable by a programmed sequence of events or algorithm can be considered to be an ATE. This sequence is commonly referred to as a test program. Once the system has been properly energized and initially set up, control of the instruments and the testing process is vested in the test program. In the early days of ATE, the control logic that executed the programmed instructions was of a design unique to the special needs of a given ATE. Soon, however, digital computers were generally utilized as the control system. They offer both improved flexibility and reduced cost over specially designed controllers. Today computers find multiple usage in ATEs, not only as the central control unit but also in many "building blocks" or instruments embodied in a system.

In its basic form, an ATE consists functionally of five subsystems: control, stimulus, measurement, switching, and system software. An ATE also contains various peripheral devices to facilitate user communication with the system. They include but are not limited to maintenance panels, input keyboard, printers, and output panel for connecting to the UUT interface device (ID). A simplified block diagram is shown in Fig. 1.

Control is generally provided by one or more general purpose digital computer(s) with associated memory. Switching consists of a network of electrically actuated paths interconnecting all of the internal elements of the ATE and the interface to the outside world. It is through the switching subsystem that the control subsystem *instructs* and actuates each ATE function as to when and how to execute the programmed instructions. It also provides paths for the ATE and its associated input/output devices, such as control panels and bulk memory storage devices. The stimulus subsystems comprises all of the power and signal sources required to energize the instruments internal to the ATE and the device or unit being

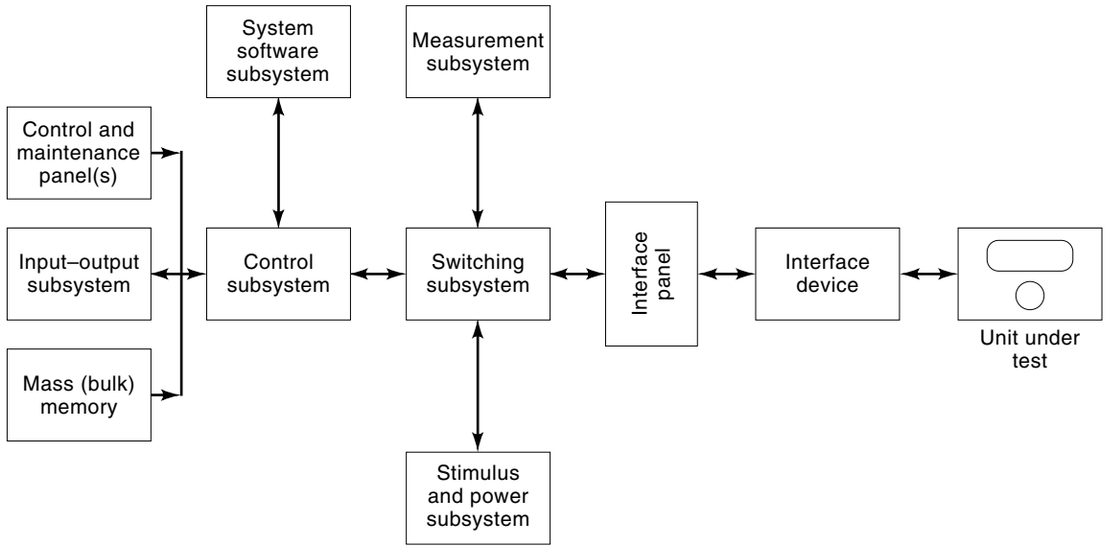

Figure 1. Simplified block diagram of an ATE.

tested. The measurement subsystem comprises all of the instrumentation required to measure both internal signal sources and responses from the UUT. The interface subsystem consists of all control panels, operator interface devices and linkages to the outside world, to which the UUT is connected directly or through an ID.

COMPARISON OF MANUAL AND AUTOMATIC TESTING

Manual and automatic testing have much in common. In both instances, the objective is to evaluate the item to be tested to verify proper performance or to detect and isolate a fault. Both approaches require several elements:

1. A way to energize and stimulate the UUT
2. Signal routing devices such as wire, switches, or relays
3. One or more devices or meters to condition signals and measure the signals involved
4. Controls and displays to allow interaction between the test technician and the hardware
5. A procedure, written or intuitive, that the technician follows in the testing process
6. The item or unit being tested

In the manual testing process these elements tend to be discrete instruments and components tied together by cables or wires. In an ATE most of these elements are embodied in the tester. If a good match exists between testing requirements and tester capability (referred to as assets), the only external elements are the interfacing connectors or cables connecting the tester to the UUT. The most significant difference is that with the ATE the procedure used in testing is neither dependent on intrinsic knowledge or sequence of actions by a technician nor on a written procedure. It is embodied in the ATE memory and is called a test program.

CONTROL SUBSYSTEM

The control subsystem is a functional entity rather than a single physical unit. In the early days of ATE it consisted of the system input and output devices such as a perforated tape or punched card reader, an operator control and display panel, a printer, and a uniquely designed logic network for interpreting and executing instructions fed into the system through the input device. Some of the earliest testers did not contain an internal memory but simply executed instructions directly from the input device. Later, specially designed computers with bulk and random access memories were developed for controlling the ATE. By the late 1960s commercial, general-purpose computers became popular because they offered greater versatility and reduced design and manufacturing costs. Some military ATEs used rather sophisticated, medium-scale computers, whereas those used in commercial manufacturing applications tended to use basic, 8-bit computers.

Once loaded with necessary software and a test program the role of the control subsystem is to execute all programs, both system level and application (test) programs. It works very much like any general-purpose computer, but instead of simply doing clerical work it actuates the various paths and instruments of the system as it interprets programmed instructions.

STIMULUS SUBSYSTEM

The most elaborate and costly subsystem of an ATE is the stimulus subsystem. It consists of all the power supplies, signal generators, and signal conditioners required to energize the UUT during dynamic testing, that is, operating the UUT in as close to its normal operational modes as practicable. In elaborate, general-purpose ATEs the stimulus subsystem can occupy several racks of signal generators, depending on the range of and complexity of the family of UUTs it is intended

to test. Special-purpose testers require only a few signal sources, because the application is limited to a small variety of UUTs.

In early maintenance testers, the signal generators consisted of building bricks or chasses of standard test equipment with programmable controls. In recent years many of the signal sources have been reduced in size to plug in cards. These signal sources (building bricks) are often coupled together to generate complex signals from basic circuits. For example, instead of using three signal generators to produce square, triangular, and sawtooth waves, a single signal synthesizer is utilized, thus saving space and cost. Some building bricks, such as a frequency multiplier, can serve for either signal generation or as a measurement resource. Hence, the actual role of building bricks can vary depending on how they are programmed and for what purpose.

MEASUREMENT SUBSYSTEM

The measurement subsystem provides the capability to measure the responses from a UUT during the testing process. Collectively the elements of the measurement subsystems serve as a programmable multimeter. The measured responses from the UUT are compared to the programmed upper and lower limit stored in the test program. The control subsystem then directs the testing sequence according to the results of the measurement.

SWITCHING SUBSYSTEM

The switching subsystem consists of all the programmable paths in the ATE required to connect the tester to and from the UUT through the interface panel, select the proper stimuli, choose the measurement instrumentation, and switch all of the tester resources as required for each test in sequence. Some testers incorporate a network of relays for internal signal routing, some use solid-state switching networks, and some have patch panels with wires connected as appropriate for the given UUT. Some testers use combinations of all of these switching mechanisms. Some utilize a *universal* routing system for maximum flexibility by allowing any tester resource to be connected to any other and to any pin in the ATE/UUT interface panel.

Some type of connecting mechanism is required to connect the ATE to the UUT. It can be as simple as a cable or as complex as a network of components within an enclosure. The general term for the interface hardware is the ID. In maintenance applications the ID typically consists of an assembly that mounts onto a patch panel. In manufacturing applications the ID can actually be a complex electromechanical unit for automatically feeding UUTs such as integrated circuit (IC) chips into test receptacles and sorting the tested chips into bins based on test results.

SYSTEM SOFTWARE SUBSYSTEM

As with general-purpose computers, an ATE requires several software subsystems or modules to effect its operation. The operating system provides all of the facilities required for operator interface, communicating with input–output devices,

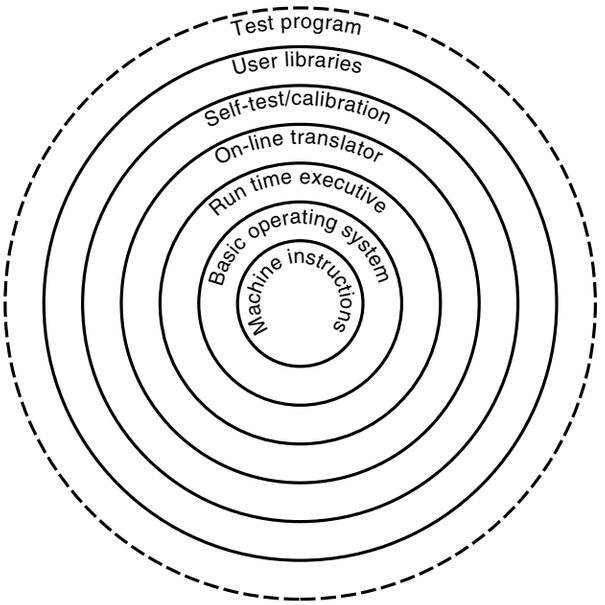

Figure 2. System software hierarchy.

allocation of the system resources, and execution of various mathematical subroutines commonly used during the testing process.

No ATE is complete without its software. The software consists of a hierarchy of programs (or software models), most of which are intrinsic to the ATE design and not modifiable by the user. Among the software programs are the operating system, the self-test system, and the user libraries and test program. The operating system controls all of the basic operations such as loading the test program, calling and executing subroutines, executing the testing sequence, and driving the peripheral equipment, such as operator displays and printers.

System software is built up in hierarchical layers, much like the cross-section of an onion, as illustrated in Fig. 2. The core of the system consists of the basic machine instructions of the computer that is the nerve center of the control subsystem. The next innermost layer is the operating system followed by the utilities that facilitate *booting* up the system and input–output operations. Next is the on-line interpreter that converts input code to executable code during program execution. Next is the set of self-test programs that consist of a hierarchy of programs as described later. Next is a library of user subroutines and/or macros that can be initiated by the test program during run time to perform various, standard testing and/or to test set-up functions such as power on sequence of combining several stimuli outputs to generate a complex signal or waveform. Finally there is the specific test program for the UUT to be tested.

MACHINE INSTRUCTIONS

Computers are designed to execute a set of basic instructions such as `add`, `subtract`, `store`, and `get`, that provide the capability to sequentially perform arithmetic functions and data manipulations. These functions are performed by logic circuitry that interprets *words* composed of different combina-

tions of ones and zeros. It is only through the addition of various software programs that ascribe specific meanings to these words that the user is able to communicate with the computer in an English-like language.

OPERATING SYSTEM AND RUN TIME EXECUTIVE

The operating system is a program that is fundamental to controlling the activity of the computer. It is a fixed part of the ATE design that the user cannot modify. As in most computer systems the operating system is the *traffic controller* for the various activities needed to allocate the system resources such as the arithmetic unit, executable and bulk memory, and input and output devices. In an ATE there often is also a *run-time* executive program that can be considered part of the operating system. It interprets programmed test statements and sends appropriate signals to the ATE resources.

TRANSLATORS

Translators simply interpret the source language statements, check them for proper format, and generate equivalent object language statements without attempting to evaluate the validity of the statements to perform the appropriate action in the ATE. Usually translators are used only to make changes to a test program during program validation rather than to generate the original object program. Hence they are often resident on the ATE as a auxiliary tool called into service through the maintenance panel when needed. Some ATEs utilize an on-line translator that converts the object program to executable code during test time. Its use is automatic and invisible to the user. It is one of the functions performed by the run time executive program.

COMPILERS

Interpretation of English-like (source language) test statements are accomplished by either compilers or translators. Compilers are not generally resident in the ATE. Rather they are hosted on a powerful, off-line data processing system. Hence no compiler is shown to be part of the ATE system software in Fig. 2. Compilers do much more than translate source language to ATE language. They generally screen test procedures to assure that the target ATE resources are available to perform the functions called out, that none of the tests programmed violate safe rules for using the ATE, and sometimes even automatically allocates ATE resources such as signal sources and generates the linkages to apply the signal to a specific set of pins on the ATE external connection panel.

SELF-TEST

One of the features of an ATE is the ability to test itself. To accomplish self-testing, however, certain features must be incorporated in the stimulus and measurement building bricks, and the diagnostic programs must be designed and incorporated in the ATE. The stimulus signal sources must have self-test outputs that can be internally routed to the ATE measurement subsystem. These outputs can be used to evaluate stimulus assets during self-test or used to measure current

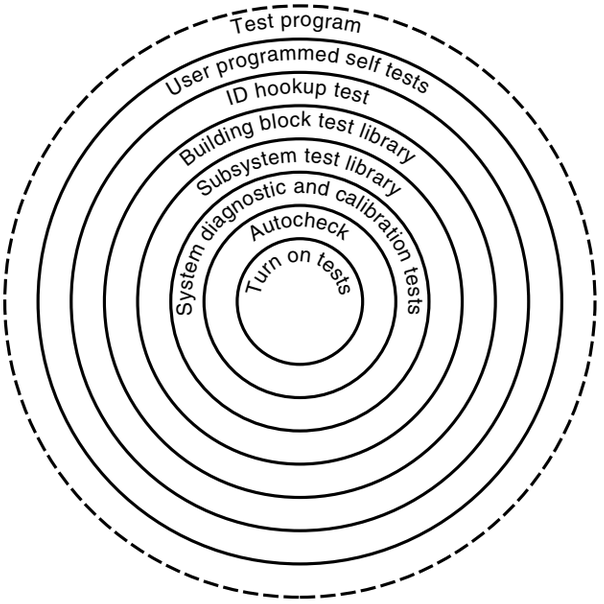

Figure 3. Self-test software hierarchy.

signal settings prior to be being applied to the UUT. Likewise, the measurement subsystem must have entry points for the selected stimulus signals to be routed to the proper measurement circuitry. These features are intrinsic to the ATE system hardware design. They cannot be added on later or implemented effectively with software only unless a network of programmable switches are provided to route signals internally. That is a poor alternative because of the signal losses or contamination that is introduced by long signal paths and switching devices.

When the required hardware features are provided to internally link the stimulus and measurement subsystems, a set of self-test programs can be incorporated in the ATE. Self-test capabilities range from a simple set of system *health checks* to a multilevel hierarchy of self-test programs, some of which are automatically executed during UUT test time or as more detailed tests selected at the user's option. A comprehensive set of self-tests are described in conjunction with the illustration of Fig. 3. Not all ATEs will provide such a complete set of test programs but they are desirable features to look for in selecting an ATE. Although these features add to the design cost of an ATE, that cost is amortized across all of the ATEs of a given design and not borne by a single customer.

SELF-TEST HIERARCHY

Just as system software is built up from a hierarchy of interdependent software modules, self-testing is also a hierarchical set of software modules. Figure 3 shows an expansion of the self-test function indicated in Fig. 2. The software modules shown in Fig. 3 are typical but not always organized in the fashion shown.

TURN-ON TESTS

Automatic test equipment is designed to provide maximum versatility to the user. This versatility can lead to misuse and

possible damage to the hardware or even harm to the operator. Accordingly, a well-designed ATE incorporates safeguards that preclude damage to the hardware or danger to the user. These are sometimes called *safe to turn on* tests. For example, they check to see that the hook up between the ATE and UUT will not draw excessive current or that high voltages are not connected to low-voltage terminals or where they are exposed to operator controls.

AUTOCHECK

After a safe turn on, the system automatically performs a set of basic tests of the ATE subsystems to assure that the system is *healthy* enough to be used. These tests, called *autocheck* continually monitor the health of the ATE during normal testing of a UUT. They perform routine measurements of stimuli and check for proper operation of measurement circuits. If any test fails, testing of the UUT is halted and the operator is advised that there is a problem. Generally the fault is not isolated by autocheck, but it does instruct the operator as to which set of diagnostic tests should be performed.

SYSTEM DIAGNOSTIC AND CALIBRATION TESTS

Some ATEs include secondary standards, such as standard voltage cells. These, together with the measurement subsystem, can be used to calibrate many of the system resources. Tests can be incorporated in the basic system design to continuously or periodically check the calibration of the most sensitive circuits of the ATE. These tests would automatically be executed without the need for the UUT test program being impacted unless a capability required falls below acceptable tolerance. Hence calibration tests are at the very core of the self-test hierarchy.

These consist of a library of tests generally resident in the ATE but initiated by the operator in the event of an autocheck indication of failure. They check out the system in more detail to automatically adjust calibration of the ATE resources or, when not corrected by calibration, to indicate which subsystem has failed. The operator or maintenance person can then initiate testing with the indicated subsystem diagnostic tests from the subsystem test library.

SUBSYSTEM TEST LIBRARY

This is a library of test routines grouped according to the ATE subsystems. Because the system diagnostic tests indicate which subsystem tests should be performed a logical linkage exists. It may be necessary to load in the appropriate subsystem diagnostic test program or it could be resident in the ATE so that it is selected from a maintenance panel. Normally the operator must call in an authorized maintenance person, because the maintenance panel can only be activated by a key. Once loaded the subsystem test program identifies the faulty assembly (or ATE building brick) requiring repair. Ideally the faulty assembly can be replaced with a spare, and the ATE can be returned to service. The diagnostic test must be repeated to assure that the maintenance action solved the problem. The faulty assembly can then be tested as a UUT using building-brick tests.

BUILDING-BRICK TEST LIBRARY

The building-brick test library is a set of test programs comprised of individual test programs for each ATE building brick. They are hooked up to the ATE through the standard interface panel and the appropriate ID just as any UUT would be tested.

USER LIBRARIES

As experience is gained through significant use of an ATE, commonly required tests of specific elements of the system are developed into standard test sequences. They can be implemented as subroutines or macros. The purposes vary but can include self-tests for specific building blocks known to be critical to a given test procedure or simply sets of instructions often required to perform complex measurements such as for gain measurement. This library is generally developed by the test design team and developed over a period of time as new ways are found to reduce test design time and/or increase standardization. Such a library is limited only by the ingenuity of the users.

THE TEST PROGRAM

At the top of the software hierarchy is the test program. It is the real workhorse of the ATE. All other software in the lower rings of the system are there to facilitate testing. The effectiveness of an ATE is dependent on a good test program design.

TEST PROGRAM SET DESIGN AND PRODUCTION

A test program set (TPS) consists of a test program, an ID, and a test program instruction. Producing a TPS for other than a trivial UUT is a complex, time-consuming, tedious process. It often involves a team effort with a variety of skills and a whole host of specialized hardware and software tools as well as the target ATE. However, it is divided into five fundamental processes performed more or less in sequence, but with multiple feedback loops when things don't go right the first time—and they rarely do. The five processes are test design, program production, ID design and fabrication, validation, and demonstration and acceptance.

TEST DESIGN

Test design is the most challenging part of the test program development process, requiring a skilled, technically competent individual with both design and maintenance experience. It requires years of training and experience to become adept in the science and art of test design. Neither this document nor any other can compensate for a lack of testing experience. However, an overview is offered here with some valuable tips and recommendations based on many years of TPS development experience applicable to maintenance testing.

Test design is the process of developing the testing requirements of the UUT and the logical sequence or flow of the intended process of testing. For maintenance applications the testing information required is derived from many sources.

The performance tests, referred to as the go-chain tests, are usually derived from the factory acceptance test (FAT) procedure. It should provide basic information on the acceptable performance requirements for a properly operating UUT. However, the FAT is most likely performed with test equipment other than the target ATE and the test process is designed to quickly check if the UUT has been properly assembled and tries to detect typical manufacturing errors such as solder splashes, open connections, or improperly mounted components. So although the go-chain tests might be usable in maintenance testing, the fault detection and isolation tests are not. And the go-chain tests most likely will have to be modified to take into account signal deterioration due to signal routing in the ATE as well as making the test limits less demanding to account for acceptable performance degradation or UUTs in service.

Sometimes a test requirements document (TRD) is provided by the UUT manufacturer. It identifies input and output signal requirements for normal testing of the UUT as well as some fault mode information. It may also recommend the proper sequence of testing. Test requirements documents are not cheap and often are hastily produced and contain numerous errors and voids. Making the producer of a TRD accountable for its completeness and accuracy is not easy. Ultimately the burden for proper test design rests on the test design engineer, regardless of how good the data package may be.

Software tools are sometimes available to assist in designing some tests. Such a tool is often called an automatic test program generator (ATPG). It consists of a powerful, special simulation program operable on a general-purpose computer. The program automatically generates test patterns based on a defined digital logic design. These patterns must then be integrated into the remainder of the test program. Some ATPGs can handle limited hybrid circuit simulation, but to date none provide a comprehensive analog circuit simulation capability.

The test designer need not know exactly how the ATPG works, but must become adept at interacting with it because ATPGs are not really as automatic as they profess to be. Specialized training is required to become adept at using an ATPG, and the availability of expert consultants may also be required. Even with the aid of an ATPG, much of the test-design process depends on the ingenuity of the test designer.

The objective of the test-design process is to develop a detailed, unambiguous, step-by-step procedure that becomes the controlling program used by the ATE to test a UUT. It is up to the test designer to "think as a machine thinks," while generating tests consisting of a series of interrelated tests pertinent to the UUT involved. Here, an individual test is defined as a set of instructions needed to set up the ATE instrumentation (sometimes referred to as ATE assets) as required to energize the UUT, route the ATE signals to the proper ATE output panel and interface device connectors, provide a signal response path from the UUT to the selected ATE measurement instrument, compare the response to stipulated upper and lower value limits, and direct the ATE to the next test to be performed based on the results of the current evaluation. The sequence of tests and all alternative branch paths between tests is referred to as the test-flow process. It is best recorded as a test-flow diagram showing all go-chain tests with appropriate branching to fault isolation tests. A typical test involves proper setting of all the parameters needed for a single measurement of a UUT response followed by a single decision in a three way branch in the program logic, based on a *high*, *low* or *go* test result. If the measured value is within the programmed limits the go branch is taken. If the measured value exceeds the upper limit, the high branch is followed. If less than the lower limit, the low branch is followed.

ESTABLISHING TEST LIMITS

One of the more difficult yet critical tasks of the test engineer is establishing upper and lower limits for each test in the program. The recommended approach is to begin the calculation with the *when-new* or manufacturer's specified limits for normal operation. These are the ideal limits which must be relaxed or made less demanding based on anticipated adverse impact due to expected environmental conditions of UUT usage, expected aging due to component deterioration while in use, and measurement inaccuracies of the ATE (including distortion and crosstalk). The final upper and lower limit values should be as broad as allowable for acceptable operation of the UUT in its normal environment but never beyond.

The sequence of tests begins with checks to assure that the UUT has been properly connected to the ATE via the proper ID. These tests are followed by several safe-to-turn-on tests that assure there is nothing in the setup that could damage the tester. The UUT tests begin with static tests to assure that there are no undesired open or short circuits in the UUT. Static tests are conducted without power being applied to the UUT. Having safely passed these tests power and signals are applied to the UUT to approximate normal operating conditions. Tests executed while the UUT is energized are called dynamic tests. They comprise the bulk of the testing process and require the greatest design skill.

Dynamic tests are generally grouped according to similar input signal requirements to minimize setup and signal settling times. It is also good practice to provide *break points* in the testing sequence. These are places in the testing process where it is safe to enter the process without having to begin with the first test. Break points assure that all ATE parameters are properly set. They are very handy during test design validation and also when attempting to repeat selected tests during a UUT retest.

Sometimes the test operator must interact with the testing procedure by performing manual actions. For example, the UUT may have operator controls or mode-setting switches. The test is not considered complete until each position of the switches or other UUT controls have been tested. The test designer must anticipate the need for manual intervention and program a stop or *pause* instruction and send a message to the operator regarding the action to be taken. When the manual action is completed, the operator initiates a *continue* action and the automatic testing sequence resumes.

When all parameters of a normally operating UUT have been programmed in a sequence of tests, the performance tests (referred to as go-chain tests) are considered complete. It is often necessary to redesign some of the tests based on actual results observed during the validation process as is discussed later. The entire go-chain should be designed before attempting to develop the fault isolation tests (referred to as the *no-go tests*). Results of each test are generally printed out together with a final judgment as if the UUT is okay or faulty.

Fault detection and isolation tests can be partially derived from deductive logic based on which go-chain tests fail. However, most no-go tests must be determined from anticipated field failures. Rarely is such information provided to the test designer, so developing it becomes one of the greatest challenges for the test design engineer. For this task there is no substitute for practical experience and, if available, help from the UUT designer.

When all anticipated failure modes (ways in which the UUT is anticipated to fail) have been accounted for in the test design the no-go chain is completed. Often the information concerning realistic field failures is not given in the source documentation on which the testing is to be based. The preferred solution is to hook up the UUT in a laboratory environment and induce failures in the UUT one at a time to determine the results of each anticipated failure. Without such bench testing the fault isolation tests are theoretical and are often found to be defective during validation when the UUT is attempted to be tested on the ATE. At the end of the fault isolation tests the ATE indicates any failed tests as well as notifying the operator what the most probable component in the UUT has failed. Once repaired, the UUT must again be tested to assure that it is now operating correctly. Usually only one fault is found at a time so it may take several passes in the ATE before the UUT is *ready for issue.*

TEST PROGRAM SET PRODUCTION

Test program production involves generating the object language test program (the language acceptable as an input to the ATE) that dictates the testing process, assembling the ID from existing hardware or building a prototype (called a brass board), and generating the TPS documentation. Generating the object language statements can involve two steps: (1) Translating an English-language test procedure, such as a TRD, into test programming language (referred to as the source language). (2) Translating the source language statements into the object language. Many programming languages have been used for source code. The IEEE standard language ATLAS (automatic test language for all systems) has been used most for military applications. The airlines maintenance community generally uses a specialized version of the ATLAS language. Commercial manufacturing applications tend to utilize more general-purpose and flexible programming languages such as Fortran or C+ but they are not test-oriented languages so they require development of a specialized test statement dictionary developed by the user so they require a greater depth of programming skill. A good test programming language allows the test engineer to compose a program in engineering English statements, such as set supply A to 15 V, dc or connect supply A to pin 3 of J24. ATLAS provides such a capability but many versions or dialects of ATLAS have been used so that there are special instructions required for each dialect. The proper dialect for any given tester is generally defined in a test programming guide.

Once encoded in source language the source program must be converted to the object language peculiar to the ATE. Conversion is usually performed on a software generation station consisting of a general-purpose computer containing a compiler. A compiler is a program that not only translates source code to object code, but also performs a number of validity checks on the programming logic, allocates memory spaces, and performs numerous other *housekeeping* chores needed to handle test program execution during testing (referred to as run time).

Compiler design is a highly specialized technology relegated to skilled programmers and not the domain of the test engineer. However, the test engineer must become adept at using the compiler and resolving problems *flagged* in the compiler printout or what is called the *object code*. The program is not considered completely compiled until all flagged statements have been corrected or resolved. It may take several passes through the compiler to get a clean object code listing that can be used during program debug and validation.

INTERFACE DEVICE DESIGN AND FABRICATION

The nomenclature of ID may be misleading because interfacing a UUT to an ATE is generally a complicated process. The term *interface device* was coined many years ago when a simple adaptive device such as a multipin connector or cable was all that was required to hook up a UUT to one of the early ATEs. Today a fairly large box is often required that contains a group of connectors, cabling, and some electrical components. Such an ID is typically used to interface a related group of circuit modules. In military terms the modules are generally called *shop replaceable assemblies* (SRAs). These consist of quite complex circuit boards or assemblies taken from larger UUTs called *weapon repairable assemblies* (WRAs) or *line replaceable units* (LRUs). Typically in aviation maintenance applications WRAs are 30 to 40 lb assemblies removable from an aircraft at the flight line, whereas SRAs are removed for testing in a maintenance shop. A WRA can contain 10 or more SRAs and a typical military aircraft can contain 80 or more WRAs.

Interface requirements linking UUT to the ATE are best determined and recorded concurrently with the test design process. A standard ID wiring form is recommended with ATE panel connections at the left side and UUT connection points on the right. Adequate space should be provided for drawing wiring and component schematic symbols. Because an ID is often shared among several UUTs, it is important that there be no interference between diverse connections required by each UUT. Unique connections required by UUTs can usually be accommodated by programmable switches in the ATE or, if unavailable, relays may have to be incorporated in the ID design. If specialized signals are required that are not available from the ATE, signal conditioning circuitry may have to be incorporated in the ID. But every attempt should be made to avoid the use of active components in the ID, because they become likely candidates for failure.

Once the ID requirements are documented, they are submitted to a shop for manufacturing. A "brass board" preliminary design is recommended until the TPS has been validated, since changes are likely. After validation the ID should be manufactured as a deliverable item using good design and manufacturing practice.

The ID design is normally represented by a schematic drawing generated by the test designer at the same time as the test procedure is programmed in source language. It shows the connections between the ATE and the UUT with any intervening parts. Ideally only standard cables or patch

panel connectors are required when all signals from and to the ATE are compatible with the ATE. Often, however, there are some components that must be included in the ID. They might include load resistors or special signal conditioning circuits. It is up to the test designer to specify the parts and the design of any nonstandard circuits. Assembly and testing of the ID is usually done by a technician.

In maintenance testing applications IDs are custom designed by the test design engineer to meet unique needs of a given UUT or set of UUTs. In manufacturing applications of ATE, interfacing the item to be tested to the tester is accomplished with a standard design *handler* that is part of the ATE system. Interfacing the tester to the items being tested typically involves a complex and expensive handler that automatically accepts each item to be tested from a *hopper* containing hundreds or thousands of items such as computer chips and feeds them in proper orientation to a standard test jig. There each is tested and fed to separate collecting chambers according to how it passed the tests. For testing circuit boards the interfacing is often accomplished by a *bed-of-nails* set of probes that make all contacts with the circuit board in a single operation. The tests are then performed and the next board is automatically loaded to the test position. Test results are automatically printed out and keyed to the pertinent board by serial number.

TEST PROGRAM INSTRUCTIONS

Test program set documentation is referred to as the test program instructions (TPI). Documentation requirements can vary widely depending on the needs and skill of the user. As a minimum it consists of source and object program listings, ID schematic, and instructions for the ATE operator. It may also include a flow diagram of the testing process and general operating procedures for the ATE operator.

During test design and ID design the test designer should also prepare notes on instructions that must be given to the test operator for use while testing the UUT. Generally, this includes initial hook-up instructions for the ID, any precautions to be taken, and associated manual actions required of the operator to start the testing process. This information when properly edited becomes the TPI. Traditionally this is treated as a formal, hard-copy publication, deliverable as an element of the TPS. A better approach is to incorporate all instructions to the test operator within the test program. Initialization of the standard testing process can be built into the ATE in a resident *boot strap* program that is automatically executed when a start button is pressed on the ATE control panel. Information to be passed to the operator during testing, whether requiring operator intervention or simply informing the operator as to what is happening or relaying test results, should also be incorporated in the test program so that there is a constant communication and a symbiotic interaction between the operator and the ATE. With current technology and equipment there is no need to depend on hard-copy documentation for any part of the UUT testing process or even the trouble-shooting and repair of the ATE.

Once the test program listings, the ID with its schematic, and the draft copy of the TPI are available the test designer can begin TPS validation on the ATE.

VALIDATION

Validation is the process of testing a TPS on the target ATE to confirm that the testing process is valid and accomplishes the intended purpose. This process should be distinguished from the process called verification, which simply assures that the program is a faithful representation of what the test designer stipulated in source language statements during test design; not that the test design and ID are doing what is really required to test or fault-isolate the UUT. Verification can be performed by a good compiler but usually requires some *desk checking* of the compiler printout by the test designer to be sure that the program is ready to be tried on the ATE with the UUT hooked up and activated.

Once the compilation has been verified and the ID brass board is assembled and bench tested, the TPS is ready to be validated on the ATE. A known good (properly operable) UUT is required for the process of validating the go-chain. At least one additional known good UUT should be tested on the ATE to be sure that the testing process and operating limits of the program are generally valid and not unique to only one UUT of the type being tested. Go-chain validation should be completed before any attempt is made to validate the fault isolation (no-go) tests. Once the go-chain has been validated a sample number of anticipated field failures should be induced in the UUT, one at a time. Careful analysis of the UUT design is required to assure that induced faults will not damage the UUT. What failures are considered reasonable field type or operational failures and what percentage of anticipated failures should be induced during the validation process is a tough judgment call. It depends on the degree of confidence that the customer is willing to pay for. Formulas have been developed to assist in the process of computing the degree of confidence and the level of fault isolation that is desired. Some guidance along these lines is provided in future chapters.

It is important that the test designer has a clean compiler listing, tested ID and its schematic, and UUT documentation prior to attempting to validate a TPS. Auxiliary external test equipment may also be needed to examine signals at interface points when the signals programmed or received from the UUT are in doubt. If a lot of validating is planned, it is a good idea to have an experienced trouble-shooter to assist in the validation process. Time on the ATE is very precious, so validation should proceed as rapidly as possible. That is not the time to design or redesign tests. One should note each problem and move on to the remaining tests unless precluded by the impact of previous failures. Problem solving and test redesign should be done off line. Validation can then be resumed at a later time when machine time can be scheduled. Any problems experienced with the ATE should be recorded in a validation logbook.

A TPS cannot be considered fully validated until a complete printout is obtained indicating all values measured with *go* results and printouts showing measured values for UUT failures with appropriate results and instructions printed for the test operator.

Demonstration and Acceptance. Once the TPS has been debugged and validated, some sort of demonstration is required to *sell off* the TPS to a customer. In the military sell off is a formal, well-documented, and explicit process. In a factory

application it may be simply demonstrating to the production manager that the test program and ID operate as required. In a formal sell off, the documentation package describing the testing process is reviewed and accepted. Next the TPS operation on the ATE is demonstrated. The go-chain is executed to show that an operating UUT can be properly tested as operational. Then an agreed upon set of faults are inserted into a UUT, one at a time, to show that faults are identified by the testing process. Once an acceptable number of faults have been shown to be covered by the process the customer's representative signs the acceptance document.

Built-in Test (or Built-in Self Test)

In recent years, as components and systems have become much more complex, considerable attention has been given to designing devices, modules, and systems so that they are more readily testable. Since the cost of testing logic chips in the manufacturing process represents as much as 30% of the production cost, it has become cost effective to include built-in-test (BIT) features, which reduce manufacturing test time and the overall cost of producing the product. Typically approximately 10% of the *real estate* (surface area) of a chip is devoted to testing circuitry. A popular technique is called boundary scan. This methodology provides program-controlled access to the internal circuits of a chip to allow sampling internal circuits without the need for an external probe.

In maintenance applications, the BIT features introduced in the manufacturing process is utilized to isolate faults in the product's components. In large systems such as aircraft, system-level BIT is usually incorporated in the design to assist in maintenance testing. BIT becomes a subsystem of the design and provides in-flight health monitoring as well as indications of particular failures. Identification of a fault allows the pilot to select alternate modes of operation during flight. Once landed the BIT subsystem can be queried to determine which subassemblies need to be replaced to restore the aircraft to full operation. In the maintenance shop the information gained from the BIT circuitry can be used to help isolate faulty components and thus facilitate the repair of the failed units removed from the aircraft. Automobiles are also being equipped with some BIT features.

Simulation as a Test Tool

Simulation provides a means for modeling a circuit, network of circuits, or even an entire system on a computer. Simulation finds greatest utility in the design process but can also help in test design. For many years software programs have been used to simulate digital circuits and networks and automatically generate test patterns needed for testing. More recently some simulators have incorporated test generation features for hybrid networks (a mix of analog and digital circuitry). Simulation of complex analog networks has not been very successful due to the complexity of developing effective models. Simulators used in circuit design generally do not provide for simulating fault modes as required for maintenance testing, so they offer only limited utility for test design. Future design simulators very likely will begin to address the needs of the test designer as well as the product designer.

Golden Card as a Testing Standard

Some manufacturing applications have utilized a circuit card or subassembly known to be operating properly as a standard against which newly manufactured cards of the same design can be tested by comparing input and output signals. The standard card is referred to as the *golden card.* This approach works well in some applications and is easier to use than a software simulator. The danger lies in the possibility that the golden card could fail and result in erroneous testing of the new cards. A more effective technique is to develop a software model of the golden card to use as a standard since once software is properly designed and tested it does not fail.

Guided Probe

Some commercial ATE includes a feature called *guided probe.* It is a system software module that helps in trouble-shooting a circuit card or assembly. In the probe mode of operation the ATE directs the operator where to put a signal probe in each of a series of points while the tester evaluates the signal from each point probed. Thus by use of a trouble-shooting algorithm the ATE leads the person to identify the faulty component. This technique can be more effective than relying only on access to the circuitry from input and output connectors, as is the case with many testers.

BIBLIOGRAPHY

Although ATE is utilized extensively in both manufacturing and maintenance applications, surprisingly few textbooks have been published on the subject. A search through the Library of Congress index of published books reveals that by 1997 only four have been published that are devoted to ATE since the first was published in 1974 by IEEE Press. The three that could be classified as textbooks are entirely devoted to manufacturing applications of ATE with no mention of the multibillion dollar market for maintenance testing utilized principally by the military and the airline industry.

The principal sources of documentation are the periodic symposia, conferences, and workshops dedicated to various ATE specialties. They provide both invited and volunteered papers which are subject to peer review. Copies of recent issues are generally sold directly by the technical societies sponsoring the events. Some conferences offer tutorial sessions that provide valuable prerequisite information covering basic principles or advanced techniques. These meetings, with their respective publications, are the best source of technical information, because they are directed at a wide variety of applications and tend to contain the latest developments in ATE technology. The International Test Conference is devoted almost exclusively to manufacturing test and AUTOTESTCON is almost exclusively devoted to maintenance testing.

There are many periodicals, generally published monthly, that provide a medium for publishing volunteered papers of variable quality. Industry studies are common but they stress marketing data rather than providing a source of technical information. Other publications, such as user guides, deal with the specifics of particular ATEs so they are of limited value for general education since all ATEs are different.

Some useful books are listed below.

K. Brindley, *Automatic Test Equipment,* London: Butterworth-Heinemann, 1991.

R. Knowles, *Automatic Testing Systems and Applications,* New York: McGraw-Hill, 1979.

F. Liguori (ed.), *Automatic Test Equipment: Hardware, Software and Management,* New York: IEEE Press, 1974.

A. C. Stover, *ATE: Automatic Test Equipment,* New York: McGraw-Hill, 1984.

Some useful conference records are as follows.

Automatic Support Systems Symposium for Advanced Supportability, New York: IEEE, published annually between 1965 and 1975 except for 1971.

AUTOTESTCON, New York: IEEE, published annually from 1976 to the present.

International Test Conference (successor to the Cherry Hill Conference), New York: IEEE, published annually from 1970 to the present.

FRED LIGUORI
ATE Consulting Services