I

DESIGN AND IMPLEMENTATION OF THE TEST PACKAGE FOR THE SERIAL OUTPUT CONTROLLER BOARD

by

Spyridon Karapetsas, B. Eng.

A PROJECT REPORT
SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES

in Partial Fulfilment of the Requirements
for the Degree

Master of Engineering
McMaster University

December, 1981

MASTER OF ENGINEERING (1981) McMaster University
(Engineering Physics) Hamilton, Ontario

TITLE: The Design and Implementation of the Test Package for the Serial Output Controller Board

AUTHOR: Spyridon Karapetsas, B. Eng. (McMaster University)

SUPERVISOR: Mr. Garry Mansfield, Litton Systems (Canada) Limited

NUMBER OF PAGES:

ACKNOWLEDGEMENTS

I would like to express my appreciation to Litton Systems (Canada) Limited for its participation in the Industrial Internship Program of the Department of Engineering Physics, under whose auspices this project was conducted. I would also like to thank the National Research Council of Canada for awarding me with a scholarship to pursue Graduate Studies.

This report would not have been possible without the assistance of my industrial supervisor, G. Mansfield. Many enlightening conversations with G. Mansfield provided me with invaluable insight into the field of Test Equipment Engineering.

ABSTRACT

The Serial Output Controller is one of the component boards of a Marine Navigation System. Litton Systems (Canada) Limited required the development of a test package for this board to be implemented on the DIGIPACT test station. This report introduces the subject of electronic component board testing with an overview of the test equipment and underlying philosophies used by Litton Systems for fault detection and fault diagnosis. The four stages of the test package development process, test plan definition, programming, validation and evaluation are described. The architecture of the DIGIPACT test system is presented as background information.

TABLE OF CONTENTS

CHAPTER I

CHAPTER II - THE ARCHITECTURE OF THE DIGIPACT TEST STATION

APPENDICES

LIST OF ILLUSTRATIONS

CHAPTER I

AN OVERVIEW OF ELECTRONIC TESTING

The purpose of this chapter is to introduce the fundamental concepts, philosophies and methods of electronic testing. It does that by describing the following:

- o Manufacturing testing
- o In-Line Testing
- o Test Equipment Used for Fault Detection
- o Test Equipment Used for Fault Diagnosis

1.1 Production Testing

There are three levels to production testing of electronic systems: evaluation of incoming components, testing of printed circuit component boards and final system inspections. The purpose of any level of testing is to verify that the parameters of the items involved conforms to design specifications. Some of the parameters which are usually checked are in the following list:

a) mechanical dimensions

b) electrical parameters

c) functional requirements

d) chemical composition

e) mechanical properties

f) special processes

g) physical parameters

The objectives of production testing is to achieve a high turn on rate of the final assembly at optimum cost.

Under these constraints and considering the distributed nature of the testing process, test engineering personnel are faced with the problems of determining the extent of and the type of test philosophy most suitable for each production stage.

Experience has proven that 100% incoming inspection and card testing tuned to the type of manufacturing faults peculiar to the particular production environment are two general guidelines making production testing cost effective. To understand the significance of these guidelines consider

the probability $Ps = (Pp)^n$ that an assembly consisting of n components each having a probability Pp of being good, will itself be good. Table I summarizes some calculations.

Probability of good Component Pn	Number of Components	Probability of A Good Assembly Ps
.99	50	0.60
.60	30	2.8×10^{-7}
.90	30	4.2×10^{-2}
.95	30	0.21

TABLE I: This table lists values for the probability of a good assembly, calculated according to $Ps = (Pp)^n$

The first calculation in the table demonstrates the necessity of 100% component evaluation. The second calcualtion illustrates the reason for having card testing. The next two calculations narrow the extent of card testing to desirable levels. Although these calculations are very rough estimates of actual production statistics, they provide valuable insight to test planners.

Thorough incoming inspection of components does not solve all production problems. Various processes and workmanship faults can be inadvertently introduced into system parts during assembly. For example, solder splashes and solder bridges are common connectivity faults found on printed circuit (P.C.) component boards. Also, missed connections resulting from cold solder joints and unsatisfactorily plated through-holes is another common class of process faults. Finally, workmanship faults such as misplaced, misloaded wrong value and even damaged components make their way to the assembly.

Testing at the card and system level is aimed primarily at the detection and the repair of assemblies with process faults. To put the entire process in perspective, I should quote the following two estimates:

First, the cost of finding a fault escalates by a factor of 10 each time it advances to higher levels of assembly undetected. Second, the effort required to locate multiple faults increases in proportion to the square of the number of faults. Therefore, there is a high incentive to catch faults as early in the production as possible.

The next section deals with testing at the card level.

1.2 In-Line Testing

At Litton Systems P.C. component boards are tested at In-Line Testing. Fig. 3 illustrates the process. Loaded boards arriving from the assembly area are first subjected to Go/No-Go testing. Go/No-Go testing simply sorts boards to bad and good according to whether they have production faults or not. Good boards are then allowed to advance to higher levels of assembly whereas bad ones are sent for fault diagnosis and repair.

Fault diagnosis is the process of pinpointing the exact causes of process faults. At this stage the objective is to generate detailed instructions for the repair station personnel. A technician finds the faults using a combination of automatic and manual diagnostic procedures and makes an entry on the Manufacturing Control Tag (M.C.T.) of the board describing the fault and the method of repair. He then sends it to the repair station.

Quality assurance checks whether the repair was performed according to the practices and standards of Litton Systems. Properly repaired boards are circulated through the process again whereas all other boards are sent back to the repair station.

The Test Equipment Engineering (T.E.E.) department is in control of Manufacturing Testing. For the purpose of this report, it is sufficient to identify three of the many responsibilities of the T.E.E. department. First, it provides In-Line Testing with the necessary hardware and procedures to perform fault detection and diagnosis. Second, it monitors the effectiveness of testing using data collected by quality assurance. Finally, it recommends changes to aspects of the product design in order to improve its testability.

In light of the importance of testing to the overall success of a manufacturing facility management has uplifted its respect for T.E.E. to levels which have been traditionally enjoyed by more respectable disciplines such as design and production.

Since the level and quality of testing is a function of the hardware and test philosophies adopted by T.E.E., the remaining of the chapter is an overview of the testers and philosophies used for card testing at Litton Systems.

1.3 Card Testers

In this section I will describe the various types of card testing hardware and the underlying test philosophies used at Litton Systems. Before classifying the hardware, I should introduce the concepts of a test station and a test procedure.

A test station is a collection of stimulus sources, measurement devices and a means of connecting them to a unit under test (UUT). Stimulus sources include such instruments as power supplies, signal generators and digital drivers. Measurement devices include multimeters, frequency counters and comparator circuits. The means of connecting these devices to the UUT are diverse. Things like interconnecting cables, test boxes and switching matrices are commonly used.

A test procedure is a set of instructions to a human or a machine detailing the use of a test station for the purpose of testing a UUT. The test procedure specifies the level and type of signals applied to indicate UUT inputs as well as the expected response and the allowable error at the indiciated UUT outputs.

Card testers are classified as follows:

- Manual

- Automatic
  - Card Edge tester
    - stored sequence
    - automatic sequence
  - In-Circuit tester

1.3.1 Manual Test Station

In a Manual Test Station test set up, control of the instruments and evaluation of the test results are performed by an operator. He normally follows instructions from a written manual test procedure.

The test procedure indicates how to switch the test equipment to the UUT by means of reconnecting cables, test probes, or through manually operated prewired switch boxes. The operator exercises control of the test instruments by setting knobs on panels of the test equipment at readings specified in the test procedure. Finally, he evaluates the test results by comparing measurements against pre-established limits. In this manner he classifies boards as good and bad according to whether they pass successfully all the tests included in the test procedure.

1.3.2 Automatic Test Station

In an automatic test station human intervention is minimized although it is not completely eliminated. The operator loads the board into the testing fixture and invokes the appropriate test program. During testing, the system optionally instructs the operator to preset switches, set pots and jumpers or to verify that they have already been preset. After completing the test, the system informs the operator whether the UUT is 'bad' or 'good'.

In such a system, most test functions are done automatically under the control of a processor executing the instructions of the test program. Test set up is implemented by calls to special routines driving the programmable instruments of the station. Signal routing is usually accomplished by an electrically switched relay matrix. Evaluation of the test results is performed by the processor who compares the automatically gathered test measurements against established limits.

The advantages of an automatic tester over a manual tester are: speed, repeatability and improved fault coverage. Computers can be programmed to do more tests, do them at electronic switching speeds and do them always in the same manner.

The disadvantage is the increase cost of the test equipment. However, with the increased complexity of today's electronic circuits, there is no other alternative to automatic testers.

There are two types of automatic testers in use at Litton Systems: the Card Edge Tester and the In-Circuit Tester. The next section describes the test philosophy, the types of testing possible and the programming techniques required by these two types of automatic testers.

1.3.2.1 Card Edge Testers

This type of tester relies on the application of signals at the input pins of the UUT to propagate the effects of manufacturing faults to the output pins where they can be measured and detected. The type of signals required to prove a component board is determined by the nature of the circuits which are loaded on the board. Card Edge Testers could be designed to have both digital and analog testing capabilities and they could be used to implement static as well as dynamic tests.

Card Edge Testing simply refers to the method of accessing the UUT. The programming methodology and the limitations of the system is determined by the test philosophy underlying the operation of the tester. The following subsections describe the different types of Edge Card Testing systems: the Stored Sequence and the Automatic Sequence Testers.

1.3.2.1.1 The Stored Sequence Tester

The stored sequence tester essentially automates manual testing. In developing a test program for such a system, the test engineer first finds out how to test the board manually and then he translates his plan into the language of the tester.

The test program is all encompassing. It contains all the UUT tests. Each test is made up of instructions performing test set up, measurement and evaluation. The system controller sequence through the test program and executes only the tests encoded in it.

Most popular commercial stored sequence test systems fall into one of two categories: digital only or digital with custom configured analog test capabilities. The second category is known as the HYBRID tester. At Litton Systems, almost all of the stored sequence testers are also HYBRID testers. Examples of such systems are DIGIPACT, LATS and VAT. Furthermore, the majority of the stored sequence testers perform static rather than dynamic testing. The next section describes static and dynamic testers.

1.3.2.1.1.1 Static-Dynamic Testers

A static tester basically answers the following question: Does the UUT work at a speed much lower than its rated operating speed? This type of a tester is aimed at detecting primarily process and workmanship faults rather than faults caused by marginal components. With 100% incoming inspection of components, static testing is sufficient for most P.C. component boards.

Dynamic testers answer the question: Does the UUT work at its rated speed? This type of tester is used to verify the operation of certain types of digital boards containing dynamic memories, microprocessors and other LSI and VLSI circuits.

Currently, the area of dynamic testing is one of the most controversial subjects in the Test Equipment Engineering field. There is widespread disagreement as to what speed is adequate to testthese complex circuit boards. Some experts[12]

claim that running a board at 10% of its system speed is sufficient to detect 99% of the dynamic faults. Others insist on performing a limited functional test at full operating speeds. Yet others find dynamic testing a cost ineffective approach and reject it altogether. The important point here is that the industry is going through the pains of change as it tries to adapt to the challenges presented by the enormous complexity of today's micro circuits.

The architecture of a dynamic tester is considerably different from that of a static tester. Dynamic testers require either a special purpose processor or a very fast cache memory. With a special purpose processor the instruction set of the machine is optimized for pattern generation. In this manner, pattern sequences can be generated at a very high speed. With fast cache memory,a small number of test patterns are stored in it and are applied at high speed to the UUT. The outputs of the UUT may be loaded into a similar high speed memory for later verification or alternately the expected UUT responses could be included with the input patterns and the comparison is done by hardware in real time.

The only dynamic stored sequence card tester in use at Litton Systems is the ADAR memory card tester.

1.3.2.1.1.2 The Elimination of Marginal Components

If component boards were to be tested statically, then how would marginal components be eliminated from the final product? To answer this question, we must take a global view of the testing process. In fact, testing starts at the manufacturer of electronic components. He normally specifies components conservatively and tests them on a statistical basis thus eliminating most marginal components at their source.

The manufacturer of electronic systems does not incorporate components in a design at their absolute maximum specifications. He allows a safety margin to guarantee operation of the system outside ambient temperature conditions. In addition, he tests components at the very least on a statistical basis but more likely on 100% basis before allowing them to enter into the production lines.

At the board and higher assembly levels, he normally uses temperature cycling to detect marginal components. The idea here involves the operation of an assembly way outside its normal temperature in order to induce failures caused by underrated components. Temperature cycling is the most important technique for tuning a production line to other than process faults.

1.3.2.1.1.3 Programming on a Stored Sequence Tester

Programming of a stored sequence tester will be considered in more detail in Chapter III. In introducing the subject here, it is sufficient to identify and briefly explain the steps undertaken by a test engineer during the test program development process. Fig. 7 identifies the basic stages of the programming effort. They are: test plan development, coding, program validation and program evaluation.

In the first stage the test engineer analyses the function and the structure of the component board. His objective is to determine the acquired input stimuli which will exercise the circuit on the board in a desired manner. To derive test patterns for digital circuits, he considers either the function that the board performs in the end product or alternatively assumes the existence of faults within the board and selects appropriate stimuli making these faults visible from the output pins. Analog circuits require considerably more specialized testing because they use a wide variety of input signals ranging

from steady dc voltages to complex ac waveforms. However, the methodology of developing a test plan for them is similar.

A simulator could be used at this stage as an aid in the test pattern selection process. A similator is a software tool allowing the test engineer to model the operation of a circuit board. As a result, it can be used to calculate the response of the board to a set of selected inputs. However, simulators are currently limited in use to simple digital boards employing MSI and LSI circuits.

In the second stage, the test plan is coded in the programming language of the test system. Software tools such as editors, compilers and loaders are used in a black box fashion to generate the object code. Only in this form the test program can be executed by the processor.

In the third stage, the test program is validated using a known good board. A validate program is repeatable. It always fails bad boards and consistently passes good ones. Some of the reasons why programs do not behave in a repeatable manner are improper initialization of the UUT, analysis errors during the test plan development stage and old board specification documents. To correct these errors, the test engineer makes changes to the test program and repeats the test.

In the fourth stage, the fault catching ability of the test program is measured. This is not necessarily the final programming stage because, as Fig. 7 indicates, programs having inacceptably low fault coverage are improved by changing the original test plan and repeating the whole process over again.

The fault coverage of a program is defined as:

$$\text{FAULT COVERAGE} = \frac{\text{FAULTS DETECTED}}{\text{ALL POSSIBLE FAULTS}} \% \quad (1)$$

This quantity can only be estimated by manual fault insertion of a limited class of faults in a known good board or by software simulation.

1.3.2.2 Automatic Sequence Tester

Automatic sequence testers employ a combination of two techniques for the detection of faults in digital logic boards: the automatic generation of input sequences and a known good board for the comparison of output responses.

This type of a tester generates a large number of input patterns and applies them at a very rapid rate to a reference board and the UUT. The assumption here is that if these boards receive lengthy enough random input sequences, then their logic circuits will be exercised thoroughly so that faults embedded in the UUT will cause incorrect states to appear at the outputs where they can be detected by direct comparison to the reference board outputs.

Automatic sequence testers perform dynamic testing. This is so because the input patterns are generated by special purpose processors adapted to the alogorithmic generation of these patterns. For example, Litton Systems employs the 3040A logic tester by Fluke in the maintenance of other in house built test equipment. This sytem can test boards at clock rates up to 10MHz.

1.3.2.2.1 Input Test Patterns

A typical commercially available automatic sequence tester generates a variety of input test sequences with different statistical properties such as average duty cycle, average frequency and relative phase relationship. With this type of a tester, it is important that the available input signals approximate those experienced by the UUT in ordinary service.

Signal generators usually comply with the following two restrictions. First, they generate repeatable sequences. Repeatability becomes especially important during fault isolation. Then it is desirable to have a fault detected at exactly the same test step in consecutive test runs. Second, in a group of related signals, only one transition is allowed at any time. This restriction is important to avoid combinations of input patterns leading to indeterminate board states. Fig. 9 illustrates some of the test patterns available by the 3040 logic tester.

1.3.2.2.2 Programming on an Automatic Sequence Tester

As in the case of a stored sequence tester, programming an automatic sequence tester consists of the development of a test plan, coding, program validation and evaluation. However, the programming tasks are considerably simpler.

The test engineer begins by examining the circuits loaded on a board to determine the kind of activity which will be required by each input. At this stage, his task consists of assigning test pattern generators to each input pin. His objective is to make this selection in such a manner as to exercise thoroughly all the circuits on the board without inducing any indeterminate states.

There are some general guidelines to follow for the initial assignment. For example, high frequency test sequences are assigned to clock lines whereas reset lines are either tied to constant logic levels or pulsed only once during board initialization.

Next, the test engineer codes his test plan. In this stage he informs the system of the signal generator assignments: Coding here does not imply the development of a program in a high level language. It simply means the pin to pin connection of the system to the UUT and reference boards. In simple systems, this involves hardwiring the UUT to the tester. In more flexible systems, this is done by software.

Now, the test engineer can proceed with program validation. Again, his objective is to produce a repeatable program. Lack of repeatability is usually attributed to improper board initialization. To assist in this task, some automatic sequence testers have limited stored sequence testing capability.

Finally, the test engineer evaluates the peformance of the selected input patterns by manual fault insertion. A fault coverage of 95% is the usual target for medium size boards loaded with SSI, MSI and LSI circuits. Fault coverage is normally improved by extending the length of the test or by reassignment of the input test patterns.

1.3.2.2 In-Circuit Tester

In-circuit testers rely on direct measurement to verify component boards structurally. Structural verification means that the tester ensures that every component on the UUT is correct and functional and that it is properly connected to other components. A correctly assembled circuit board with good parts and no shorts or opens has a very high probability of working in the end product (the advertised figure is 95%).

Historically, structural testing was first used to automate the visual inspection of bare boards. Bare board testers use an interconnection fixture called the bed-of-nails to detect such defects as shorts and opens. A bed-of-nails consists of an array of many spring loaded probes which are mounted on a mechanical or vacuum operated holddown fixture. During testing, the board is mounted on the bed-of-nails so that the probes contact the many board traces and the tester performs continuity and dielectric strength tests.

Later on, capabilities of the simple bare board tester were extended so that it could verify discreet analog component boards. This was made possible with the introduction of the 'guarded measurement' technique. Fig. 11 illustrates the technique. Guard points, placed at selected nodes at an equipotential, isolate or guard out the component under test from the effects of adjacent circuitry. With this technique, measurements could be performed in-circuit using a high accuracy programable voltage source on one side of the component along with measurement devices on the other side. Passive components such as resistors, capacitors, and inductors could be measured in-situ. Also, the orientation of semiconductors such as diodes and transistors could be verified for correctness.

Currently, in-circuit testers can evaluate complex hybrid boards loaded with almost any kind of digital and analog circuits including microprocessors, ROM, RAM, and Op-Amps. Litton Systems uses two in-circuit testers. They are Fairchild, 303S and AFIT 1500/3000. Fig. 10 illustrates the architecture of a typical In-Circuit tester. The control value V connects the vacuum system to the holddown fixture. Loading the UUT on the bed-of-nails activates the valve and generates a vacuum which is necessary to pull the unit down on the pins. The switching array allows access to the board nodes by any stimulus or measurement device on the system. The in-circuit analyser

is the device used for guard measurements. Other instruments which may be part of the system are signal generators, signal analysers, counters and digital driver/sensor electronics.

1.3.2.3.1 Programming on an In-Circuit Tester

The program preparation process begins by describing the topology of the component board to the system. The test engineer indicates the type of components used on the board, their interconnections and tolerances if required. He enters this information into mass storage via an editor thus creating a model for the board. In one in-circuit tester, the GENRAD 2270, this completes the program development effort. In the 2270 tester, automatic test generation software makes use of the model to select an optimum test plan and develop the actual test program.

In other less sophisticated testers, the programmer himself develops the test plan. But even so, programming is reduced to testing a single component at a time. Furthermore, In-Circuit tester manufacturers usually supply with the systems a library of software routines to perform such basic tests, as a 'check for shorts', 'check for resistance value', or 'perform functional testing for an IC'. So, test engineers could construct lengthy component test sequences by adapting these routines to fit their particular test needs.

1.4 Fault Diagnosis

Fault diagnosis is complementary to fault detection in the testing process. The manner in which it is performed may be crucial to the success of a test station.

The purpose of fault diagnosis is to generate detailed repair instructions for bad boards. The following sections describe how fault diagnosis is performed manually and automatically on the types of test systems considered so far.

1.4.1 Manual Fault Diagnosis

A skilled technician performs fault diagnosis on a bench top manual tester as follows. He begins by tracing the path of a faulty signal. Starting at a faulty output and using the schematic and assembly diagrams of the circuit board, he methodically follows the signal until he finds a component with proper inputs and at least one incorrect output. In this manner, he isolates a node suspect of a manufacturing fault. Then he applies many trial and error measurements aimed at pinpointing the exact nature of the fault.

1.4.2 The Guided Probe Technique

The automated version of the manual diagnostic procedure is called 'GUIDED PROBING'. To implement guided probing, a system requires the following:

1. A network description of the component board.

2. A means of determining correct board responses at internal nodes.

3. The help of an unskilled operator who follows instructions to move a probe to nodes selected by the system.

The network description contains the interconnection between board components so that the system can select the next node to probe.

Systems with simulators can calculate the expected board response at any internal node on-line. If a simulator is not available, then other techniques such as signature analysis are used to characterize the correct behaviour of each node off-line.

During automatic fault diagnosis, there is a dialogue between the test system and the operator. The system issues messages to the operator instructing him to probe a node. The operator attaches the probe on the node and informs the system that he has done so by pushing a button. This interaction continues until the system decides that it has located a faulty node. Finally, the operator must examine the identified node to pinpoint the exact causes of the fault.

1.4.3 Fault Diagnosis in Automatic Sequence Testers

In an automatic sequence tester, the reference board simplifies fault diagnosis considerably. It can be used as a ROM supplying the correct board responses at all nodes. As a result, an operator can perform fault diagnosis on the tester by simply following non-comparing signals. Furthermore, if the system could accept a disruption of the circuit, then it is possible to perform 'dual guided probing' with one probe for each board.

1.4.4 Diagnosis on In-Circuit Testers

In-Circuit testers are especially suited for automatic diagnosis because they nicely integrate testing and diagnosis into the same activity. The bed-of-nails fixture provides sufficient visibility so that the tester can pinpoint faults to the exact node, accurately and directly.

SUMMARY

Table II summarizes the various types of test equipment used by Litton Systems at the component board level.

TABLE II

	TYPE OF TESTER					TYPE OF ASSEMBLIES TESTED				TYPE OF INTERFACE			PROGRAMMING EFFORT			DIAGNOSTIC METHOD			
	FUNCTIONAL				IN-CIRCUIT	DIGITAL	ANALOG	HYBRID	OTHER	EDGE CARD CONNECTOR	BED_OF_NAILS	OTHER	SIMPLE	MODERATE	COMPLEX	GUIDED PROBE		K.G.B. COMPARISON	IN-SITU MEASUREMENT
	STORED SEQUENCE		AUTOMATIC SEQUENCE													SIMULATOR	SIGNATURE ANALYSIS		
	STATIC	DYNAMIC	STATIC	DYNAMIC															
DIGIPACT	x					x	x	x		x					x		2		
LATS	x					x	x	x		x					x				
VAT	x					x	x	x		x					x				
CAPABLE	x					x				x					x				
AFIT					x	x						x	x						x
3040A				x		x				x				x				x	
303S					x	x	x	x			x		x						x
ADAR		x							1	x					x				

NOTE 1 - Memory boards

NOTE 2 - Digipact uses a form of signature analysis
(a signature is a digitized waveform stored
on disk)

CHAPTER II

THE ARCHITECTURE OF DIGIPACT

The purpose of this chapter is to describe the architecture of the system for which the test program for the Serial Output Controller Card was developed. The objective here is to provide the reader with sufficient background information to understand the code. The chapter is divided into the following two sections:

1. System hardware

2. System software

2.1 System Hardware

DIGIPACT is an acronym for DIGITAL PROGRAMMABLE ASSEMBLY AND CARD TESTER. It is a hybrid Card Edge tester capable of performing functional static tests. Fig. 12 is a block diagram of the tester. It is composed of the following subsystems:

- central processor and mass storage
- operator interface
- digital subsystem
- analog subsystem
- UUT interface

2.1.1 The Control Section and Mass Storage

The system controller is an HP 2100 general purpose minicomputer. The processor has a 16 bit wide word and 32K of core memory. This particular hardware determines the basic characteristics of the system and defines its limitations. For example, the access time of core memory makes DIGIPACT a static tester. It is approximately 1 usec and determines the average instruction execution time to be 2 usec. If we in turn assume that the processor executes about 50 instructions in applying a digital pattern to a board and testing for its

response, then we can estimate the maximum rate of generating digital patterns to be 10 KHz. At this rate, the system can only perform static testing.

DIGIPACT has two disk drives for storing all program and data files. Disk storage facilitates both production testing as well as the test program development process. Loading of programs for execution takes no longer than a few seconds and is done by simply issuing a command on the system console.

2.1.2 The Operator Interface

Automatic testers are designed so that they can be used by humans. In DIGIPACT, the peripheral devices utilized for operator interaction are:

- system console
- x-y display
- tape reader and punch
- line printer
- switches on the front panel of the processor.

The system console is a KSR 35 teletype consisting of a keyboard and a teleprinter. The keyboard is the prime means of controlling the system. For example, production personnel initiate go/no-go testing by entering the part number of an assembly through the keyboard. Similarly, test engineering personnel use the keyboard for on-line program development.

The system uses the teleprinter and the x-y display to communicate with people. During testing, it may issue messages to the operator instructing him to set switches, pots or jumpers on the UUT. At the same time, it could present real time measurements and graphs on the x-y display. During program development, the system informs test engineers about the result of their command with messages on the teleprinter.

Time on production equipment is expensive. As a result, a great deal of the programming effort takes place off-line on other more suitable systems. The taper reader and punch punch are two peripherals facilitating the transportation of program files between systems.

Program listings can be obtained from the line printer. Listings are used for documentation and are especially handy during program debugging.

The settings of the switches on the front panel of the processor are used to select the various trouble shooting nodes of the system. All test programs read and interpret the setting of these switches to implement such modes as 'continuous looping of the complete program', 'looping of only a selected test sequence', 'supression of diagnostic messages' and 'stop on first failure'.

2.1.3 The Digital Subsystem

The function of the digital subsystem is to provide the stimulus and sensing capabilities required for testing logic circuit boards. The digital subsystem includes seven I/O channels and a programmable clock and frequency measuring device.

2.1.3.1 Output Channels

Fig. 13 is a block diagram of an output channel. It indicates that the output driver is an open collector gate. An advantage of using open collector gates at the UUT end of the channel is that it allows the selection of voltage levels compatible to the logic families of the circuits under test.

Behind each driver there is a bank of memory elements one bit deep. They minimize the variance in propagation delays of signals from the processor. Fig. 14 illustrates how a register is clocked into an indesirable state as a result of gross propagation delays in the input signals.

2.1.3.2 Input Channels

Fig. 15 is a block diagram of an input digital channel. It has the dual purpose of sensing the response of the UUT and isolating the tester from the affects of damaging faults on the UUT. The input circuits are compatible to TTL and DTL logic families. Other circuit families are handled by providing signal conditioning in the UUT interface. The UUT interface is described in a following section.

2.1.3.3 Programmable Frequency Measurement Device and Clock

This device has two modes of operation: it provides frequency measurement and clocking capabilities. When operated in the first mode it can measure an unknown frequency using a technique in which the incoming signal is gated with a standard time interval. Then.

$$F.\ \text{unknown} = \frac{\text{No. of pulses counted}}{\text{Standard time interval}}$$

An unknown time interval can similarly be measured by gating it with a standard frequency. Fig. 16 illustrates these methods of measurement.

In the second mode of operation, it is a programmable clock supplying 4 different frequencies by dividing an oscillator generated 4 MHz frequency by 8, four times.

2.1.4 Analog Subsystem

The analog subsystem does not only expand the capabilities of the tester to handle hybrid boards but also helps to improve the thoroughness of testing digital boards. Common analog measurements for logic circuit boards involve DC voltages and frequencies. Additional more complex analog measurements are

required by hybrid boards loaded with transducers and other electro mechanical devices. The stimulus and measurement devices used for analog testing on DIGIPACT are:

- o Power supplies
- o DMM (Digital multimeter)
- o Synchro and resolver angle simulator
- o Synchro and resolver angle indicator
- o Relay matrix interface

There are two groups of complementary instruments in the above list. The first group includes the power supplies together with the DMM. These instruments are used to supply and characterize DC and AC signals. The second group includes the synchro simulator standard together with the synchro angle indicator. These are two specialized instruments used for testing boards loaded with synchros. The simulator provides the excitation signal to the synchro and the angle converter measures the response.

The function of the relay matrix interface is to connect on-line an analog device to the appropriate UUT pin. Analog instruments are expensive and cannot be dedicated to a single UUT pin. The relay matrix interface provides 3 sets of 8 DPDT relays.

2.1.5 UUT Interface

The UUT interface has a three-fold purpose. First, it provides the mechanical support for the UUT during testing. Second, it removes any electrical or connector related incompatibilities from the UUT. Third, it supplements the instrumentation of the system.

Boards tested on DIGIPACT presented a wide range of different interfacing problems. As a result, each board has its own custom built interface. Test programs protect the UUT against accidental use of the wrong interface by testing for a hardwired interface ID before turning the power on.

2.2 System Software

DIGIPACT was developed around a completely integrated minicomputer system. The HP2100 processor and the DOS III disk based operating system are the two components of a general purpose system adapted to a production testing application. This system includes many interacting subsystems and a host of utility programs. For example, such subsystems are the executive, the filing system and the console monitor. Among the utilities are included the language processors, the system loader and the on-line debugging tool. However, only a few of the many available software tools have any direct bearing to testing and the test program development process. These are the Editor, Algol compiler and the test language of the system.

2.2.1 Editor

The editor is an interactive program facilitating the preparation of test programs. The test engineer uses the editor to enter a new program into the mass storage of the system. Later on, he can use the editor again to correct it, expand it, and in general, modify it.

DIGIPACT has a line editor. It references program lines by number. The type of operations it performs on a line are insertions, replacement, deletion and display. Furthermore, it performs some of these operations on groups of lines.

2.2.2 ALGOL Compiler and Loader

Language processors and loaders are used to translate programs written to people to a form acceptable by machines. The compilation and loading processes are relevant to testing in so far as they are essential steps in the program development process. The test engineer must compile and load his program before he can execute it.

2.2.3 Test Language

The programming language used on DIGIPACT is HP-ALGOL. This language is a subset of ALGOL-60 developed by E. DIJKSTRA for the efficient implementation of computer algorithms.

HP - ALGOL is block structured and is compiled. It uses constructs such as IF - THEN - ELSE, DO WHILE, BEGIN - END which improve considerably the readability of programs. The result is that programs are developed fast and that a person other than the original programmer can maintain them at a later time.

The benefit of a compiled language is more efficient run time code. This shortens the time required for the execution of a test program and improves the throughput of the system.

Control of the system instruments was implemented by ENGLISH-LIKE subroutine calls such as SET (DRIVER - #, STATE) and IN (CHANNEL - #). These routines are stored in a common programming library for general use. The concept of programming libraries has been proven to further reduce the program development effort. A summary of DIGIPACT utility routines used in the test program for the SERIAL OUTPUT CONTROLLER CARD is found in Appendix I.

CHAPTER III

DESIGN OF THE TEST PACKAGE

The purpose of this chapter is to describe the engineering assignment of developing a test package for the Serial Output Controller Card. It is intended to expand those aspects of the development effort wich were not covered in Chapter I. It has two sections. The first section describes the components of the test package. The other section elaborates on the design process.

3.1 The Test Package

Programming a component circuit board on an automatic tester is primarily an engineering task which can be described as follows:

Given the Engineering Specification of a component circuit board and an automatic test system, develop a test program for the board with a predefined level of fault coverage. In parallel, construct the required mechanical and electrical fixtures interfacing the board to the test system and also supply the necessary documentation for the test program, the interface and the method of conducting the test.

My industrial intership project with Litton Systems was to develop such a test package for the Serial Output Controller Board of the Marine Navigational System. The test package was to be implemented on DIGIPACT, a static Functional Card Edge tester.

At Litton Systems, the minimum accptable level of fault coverage is set by the TEE department to 90%. Furthermore, standard documentation requirements involve a narrative description of the Test Program, the Interface Drawings and the Acceptance Test Procedure. (A.T.P.) With this information , the project requirements are more clearly defined as follows:

Develop a functional test procedure for the Serial Output Controller Card with a fault coverage level better than 90%. Implement this test procedure on the DIGIPACT system and construct the necessary interface. Also, supply the following documentation:

- o Narrative description of the test program
- o Program listing
- o Interface Drawings
  - schematic
  - assembly
  - part list
- o ATP

The above documentation is found in appendix II.

3.3 The Test Package Design Process

The development process of the test package was briefly outlined in section of Chapter I. As Fig. 17 indicates, it involves four stages. In the first stage, a test plan is developed and the interface requirements are defined. In the second stage, the test plan is translated into the programming language of the system and the interface is constructed. In the third stage, the program and interface are validated. In the final stage, the test package is evaluated for comprehensiveness and the documentation is completed. This process does not progress in a strictly linear fashion. It has many iterative loops requiring the retracing of some or even all stages.

without repeating much of the material already covered in Chapter I, the tasks in the test plan development and validation stages will be reconsidered here in the context of programming a logic circuit board on DIGIPACT.

3.2.1 Selecting Input Test Patterns

The function of logic circuits is to process information encoded in binary format. In simple terms, they perform logic, arithmetic and store operations on the digital signals appearing at their inputs. Logic circuit boards either extend or add complexity to these basic operations.

In selecting input test patterns for a logic circuit board, the test engineer first considers the functions performed by the board in the end product. His ojbective is to identify the exact sequence of control signals which will sequence the board through its various operating states.

The next task is to calculate the output responses to selected sequences of signals applied to the data inputs. This is done manually with the aid of information contained in the Engineering Specifications and schematic diagrams.

The end result is a truth table, in narrative form, describing how the board operates. This constitutes the test plan.

3.2.2 Defining Interface Requirements

The function of the interface is to adapt a general purpose tester to the specific electrical and mechanical requirements of a given board.

Most logic circuit boards have electrically simple interfaces consisting of hardwire connections between the system and the UUT mating connectors. Occasionally, some boards might require voltage translation, buffering circuits or other complementary circuity for additional tests. The mechanical fixtures on the interface assist the insertion and extraction of UUTs during testing.

3.2.3 Validating the Test Package

The purpose of test package validation is to ensure that it behaves in a repeatable manner. Lack of repeatability is attributed to one of the following reasons. First, the process of generating input patterns and calculating the output responses are prone to errors. Analysis errors, injected into the truth table of the board, are usually discovered during validation.

Second, the board might not be initialized properly. Initialization involves the application of input patterns for the purpose of setting the memory elements on the UUT to a known state. If this is done incorrectly or inadequately, the test program will not produce repeatable results.

The third reason is the use of input patterns leading to race conditions. A race condition occurs when a logic circuit cannot respond to input signals in a predetermined manner according to its truth table. For example, applying simultaneously a logic 0 to the set and reset inputs of a Flip-Flop circuit causes such a race condition. This type of problem is hard to identify and correct.

The fourth reason is design or wiring errors in the interface. Interfaces are subject to workmanship errors the same way as component boards are.

The last reason involves the use for validation of a 'known good board' built to a different configuration than that of the test package. Test packages are usually developed for new products during a time when many changes are introduced. Delays in informing all concerned departments of all the modifications could lead to a situation in which the product is not built as designed.

APPENDIX I

The purpose of this appendix is to describe the use of the DIGIPACT routines forming the Test Language of the System. Only those routines used in the test program for the Serial Output Controller Card are explained.

POWER SUPPLY ROUTINE

The power supply routine is used to set the programmable power supplies to specified voltage and current limits. The routine is called as follows:

POWER (PWR - SUPPLY - NO, VOLTAGE, CURRENT)

where

PWR - SUPPLY - NO - is the power supply address. Legal address values are 1 and 2

VOLTAGE - is the selected voltage level in volts

$-20V \leq VOLTAGE \leq 20V$

CURRENT - is the selected current limit in amperes

$|CURRENT| \leq$ 1 amp.

DMM ROUTINE

The DMM routine is used to select a desired functional mode on the DMM. This routine can be called as follows:

VOLTS - sets the function to DC volts and reads the DMM

MILLI - sets the function to Millivolts and reads the DMM

ACV - sets the function to AC volts and reads the DMM

F VOLT - sets the sample rate to fast, selects the DC volts function and outputs a start command to the DMM

F MILL - same as F VOLT except it selects the DC millivolt function

FACV - same as F VOLT except it selects the AC volts function

PULSE GENERATOR AND PULSE MEASUREMENT ROUTINE

This routine drives the pulse generator and pulse measurement device. The routine could be called as follows:

PNUM (FREQ-RANGE, NUMBER)

- This call outputs the specified NUMBER of pulses at the selected frequency.

$1 \leq \text{FREQ - RANGE} \leq 4$

$976\ \text{Hz} \leq \text{FREQUENCY} \leq 4\ \text{MHz}$

$0 \leq \text{NUMBER} \leq 32767$

PGEN (PERIOD) - This call outputs continuous pulses at the specified period.

$250\ \mu\text{sec} \leq \text{PERIOD} \leq 4.19\ \text{sec}.$

PER - This call returns the period of a signal in seconds connected to the reference gate of the device.

P MEAS (GATE) - This call returns the number of pulses generated by the signal connected to the pulse width in input during gate time.

POWER ROUTINE

This routine drives the power sequencer. This is a device which controls the order in which the AC and DC power supplies of the system are applied on the UUT.

The routine is called as follows:

ON - initiates power up sequence
OFF - initiates power down sequence.

GENERAL PURPOSE I/O ROUTINE

The general purpose I/O routine is used to transfer data between the processor ahnd the DIGITAL SUBSYSTEM. The routine is called as follows:

INGPR - This call initializes all input and output buffers to logic 0.

It also links the test program to the GPR buffers.

IN (CHANNEL) - This call inputs data from a channel $0 \leq \text{CHANNEL} \leq 6$

BIT (SENSOR - #) - This call reads in a single input sensor. To calculate sensor -#, use the following formula:

sensor -# = CHANNEL -# *100 plus sensor offset in channel

OUT - This call outputs the current value of the GPR buffers to the output channels.

SET (DRIVER -#, STATE) - This call sets the specified driver to the desired state. To calculate DRIVER -#, use the following formula:

DRIVER -# = CHANNEL -# *100 plus CHANNEL OFFSET

STATE - ON, OFF

PULSE (DRIVER -#, NUMBER) - This call pulses the specified driver a NUMBER of times.

RELAY (RELAY -#) - This call operates or releases the specified relay

$700 \leq$ RELAY -# ≤ 715

$800 \leq$ RELAY -# ≤ 815.

APPENDIX II

TEST PACKAGE DOCUMENTATION

LITTON SYSTEMS CANADA LIMITED TORONTO OFFICE INTER-OFFICE CORRESPONDENCE

TO: S. Karapetsas

FROM: G. Mansfield

DATE: January 14, 1982

SUBJECT: LSL DOCUMENTATION

In reference to your memo of January 12, 1982 you may accept this memo as authority to use the requested LSL documentation in the preparation of the project report you are preparing in partial fulfilment of the requirements for the Master of Engineering Degree.

GM:as

G. Mansfield
Test Equipment Engineering
Section Manager

```
SCTR  T=00003 IS ON DR0165 USING 00239 BLKS R=000

0001  !PAL,B,"SCTRL"
0002  BEGIN
0003
0004  COMMENT
0005
0006        18-JULY-1980
0007  THIS PROGRAM TESTS THE FOLLOWING ASSEMBLIES:
0008         855075-03
0009  PROGRAM UPDATED TO SCH. 855917 REV. A
0010
0011  SWITCH 15 UP ENABLES LOOPING OF PROGRAM
0012  SWITCH 0 UP SUPPRESSES ERROR TYPEOUT
0013  SWITCH 14 UP SELECTS THE LINE PINTER
0014
0015  &**********************************************************
0016  &
0017  &           DRIVERS, RECEIVERS & RELAY PROCEDURES
0018  &
0019  &**********************************************************;
0020
0021  PROCEDURE INGPR(GPR0); INTEGER GPR0; CODE;
0022  PROCEDURE IN(A); VALUE A; INTEGER A; CODE;
0023  PROCEDURE SET(A,B); VALUE A,B; INTEGER A,B; CODE;
0024  PROCEDURE OUT; CODE;
0025  PROCEDURE PULSE(A,B); VALUE A,B; INTEGER A,B; CODE;
0026  PROCEDURE BIT(A); VALUE A; INTEGER A; CODE;
0027  PROCEDURE RELAY(A); VALUE A; INTEGER A; CODE;
0028  PROCEDURE XOR(A,B); VALUE A,B; INTEGER A,B; CODE;
0029
0030  &**********************************************************
0031  &
0032  &           POWER & MISCELLENEOUS PROCEDURES
0033  &
0034  &**********************************************************
0035
0036  PROCEDURE PON; CODE;
0037  PROCEDURE POFF; CODE;
0038  PROCEDURE POWER(A,B,C); VALUE A; INTEGER A; REAL B,C; CODE;
0039  PROCEDURE ABIT(A); VALUE A; INTEGER A; CODE;
0040  PROCEDURE FETCH(A,B); VALUE B; INTEGER A,B; CODE;
0041  PROCEDURE TBOFF; CODE;
0042  PROCEDURE TBON; CODE;
0043
0044  &**********************************************************
0045  &
0046  &           DISPLAY PROCEDURES
0047  &
0048  &**********************************************************
0049
0050  PROCEDURE INIT; CODE;
0051  PROCEDURE ERASE; CODE;
0052  PROCEDURE CLEAR; CODE;
0053  PROCEDURE MARK(A); VALUE A; INTEGER A; CODE;
0054  PROCEDURE RESET(A); VALUE A; INTEGER A; CODE;
0055  PROCEDURE CGEN(X,Y,S,R,F);VALUE X,Y,S,R;INTEGER X,Y,S,R;FORMAT F; CODE;
0056  PROCEDURE CHAR(X,Y,S,R,F); VALUE X,Y,S,R; INTEGER X,Y,S,R,F; CODE;
0057
0058  &**********************************************************
```

```
0059  &
0060  &           DVM PROCEDURES
0061  &
0062  &**********************************************************;
0063
0064  REAL PROCEDURE VOLTS; CODE;
0065  REAL PROCEDURE MILLI; CODE;
0066  REAL PROCEDURE ACV; CODE;
0067  PROCEDURE FVOLT; CODE;
0068  PROCEDURE FMILLI; CODE;
0069  PROCEDURE FACV; CODE;
0070
0071  &**********************************************************
0072  &
0073  &           PULSE GENERATOR & COUNTER PROCEDURES
0074  &
0075  &**********************************************************;
0076
0077  REAL PROCEDURE PER; CODE;
0078  INTEGER PROCEDURE PNUM(A,B); VALUE A,B; INTEGER A,B; CODE;
0079  INTEGER PROCEDURE PMEAS(A); VALUE A; INTEGER A; CODE;
0080  PROCEDURE PGEN(A); VALUE A; REAL A; CODE;
0081  INTEGER PROCEDURE RDLU; CODE;
0082  PROCEDURE WAIT(T); VALUE T; INTEGER T; CODE;
0083
0084  &**********************************************************
0085  &
0086  &           VARIABLES, LABELS & BOOLEANS
0087  &
0088  &**********************************************************
0089  INTEGER GPR0,GPR1,GPR2,GPR3,GPR4,GPR5,GPR6;
0090  INTEGER I,J,K,L,M,N,P,Q;
0091  INTEGER TEST,E,X,Y,Z;
0092  REAL A,B,C,D,F,G;
0093  REAL V1,V2,V3,V4,CUR←1;
0094
0095  LABEL MAIN,LOOP,EN,ENDK;
0096  LABEL L10;
0097  LABEL LBL1;
0098
0099  BOOLEAN ERROR,SUSPEND,SUPPRESS;
0100  BOOLEAN INFLAG;
0101  BOOLEAN ARRAY B[NO:7];
0102  INTEGER EXPBIT,ACTBIT,TESTWD5,TESTWD6,STATENO,WD5,WD6,
0103          MASK5,MASK6,HWADD,HWADDIN,LU,NMADD,NMADDIN,
0104          STWD,STWDIN,SDB,SDBIN,LBIT;
0105  INTEGER CLK20←392,CLK10←388;
0106  INTEGER SHTENB←501,PORI←515,PBUSY←535;
0107  INTEGER PARITY←204;
0108  INTEGER T0←514;
0109  INTEGER WDCNT;
0110  INTEGER BIT13,BIT16;
0111  INTEGER FTOUT,TOUT←602;
0112  INTEGER CH5,CH6;
0113  INTEGER TEMP;
0114  INTEGER SCB2;
0115  INTEGER FPORI,NOPULSES;
0116  INTEGER NOTIMES;
0117  INTEGER BITTOUT,LU1;
0118  INTEGER ARRAY SERNO[1:3];
```

```
0119  &*****************************************************
0120  &
0121  &          ARRAYS
0122  &
0123  &*****************************************************
0124
0125  INTEGER ARRAY PNC1:5];
0126  INTEGER ARRAY DTC1:3]←@252,@125,@377;
0127  INTEGER ARRAY STARTC1:3]←11,6,1;
0128  INTEGER ARRAY DTOUTC1:3]←@5252,@2525,@377;
0129
0130
0131
0132  &*****************************************************
0133  &
0134  &          FORMAT
0135  &
0136  &*****************************************************
0137
0138  FORMAT  D0(/,"*ERRORS *",/),
0139          D1("E",I4),
0140          D2(/,"ERROR TYPEOUT WAS SUPPRESSED",/),
0141          D3(/,"BOARD TESTED",I3,"TIMES"),
0142          D4(/,"TEST DISCONTINUED"),
0143          D5(/,"NO PROGRAM FOR REQUESTED DASH # "),
0144          D6(/,"WRONG INTERFACE INSERTED"),
0145          D7("SERIAL NO. ←"),
0146          DUMMY(10/),
0147          PF1(/,"POWER SUPPLY MULFUNCTION.CALL MAINTENANCE."),
0148          PF2(/,"EXP. V1=5.00+/-.2V  ACT. V=",F4.2,"V"),
0149          PF3(/,"EXP. PULLUP 1(J1-82)=1, ACT.=0"),
0150          PF4(/,"EXP. PULLDN 1(J1-81)=0, ACT.=1"),
0151          FIN5(/,'EXP. SHTENB=0 ACT. SHTENB=1"),
0152          FIN6(/,'EXP. PORI=0 ACT. PORI=1"),
0153          HW1(/,"U20,U27"),
0154          HW2(/,"EXP. NMAD(1-8)=@",@3," ACT. NMAD=@",@3),
0155          FST(/,"U5"),
0156          FST1(/,'EXP. NDAT(1-3)=",I2," ACT. NDAT=",I2),
0157          FSDB(/,'U26,U27,U40"),
0158          FSDB1(/,"EXP. SDB(1-16)=@",@6," ACT. SDB=@",@6),
0159          SDB2(/,'U13,U14,U41"),
0160          SDB3(/,'EXP. SDB(17-32)=@",@6," ACT. SDB=@",@6),
0161          FEXM1(/,"CH5=@",@6," WD5=@",@6),
0162          FEXM2(/,"CH6=@",@6," WD6=@",@6),
0163          FIN1(/,'STATE OF INPUTS"),
0164          FIN2(/,'EIE=",I2,2X,"IDE=",I2,2X,"IECF=",I2,2X,'REBLX*=",I2),
0165          FIN3(/,'PARITY=",I2,2X,"POR*=",I2,2X,"DSTRB=",I2),
0166          FIN4(/,'STATE COUNTER IS IN STATE=",I2),
0167          FOT1(/,'EXP. EOTWD*(P1-96)=',I2," ACT. EOTWD*=",I2),
0168          FOT2(/,'EXP. SHTENB(P1-98)=',I2," ACT. SHTENB=",I2),
0169          FOT3(/,'EXP. BTCTLD(P1-17)=',I2," ACT. BTCTLD=",I2),
0170          FOT4(/,'EXP. STCNENB(P1-115)=",I2," ACT. STCNENB=",I2),
0171          FOT5(/,'EXP. TP12(P1-118)=",I2," ACT. TP12=",I2),
0172          FOT6(/,'EXP. ENBSTR(P1-109)=",I2," ACT. ENBSTR=',,I2),
0173          FOT7(/,'EXP. PBUSY(P1-108)=',I2," ACT. PBUSY=",I2),
0174          FOT8(/,'EXP. T9(P1-119)=",I2," ACT. T9=",I2),
0175          FOT9(/,'EXP. T8(P1-83)=",I2," ACT. T8=",I2),
0176          FOT10(/,"EXP. T7*(P1-113)=",I2," ACT. T7*=",I2),
0177          FOT11(/,"EXP. T4(P1-117)=",I2," ACT. T4=",I2),
0178          FOT12(/,"EXP. T3*(P1-47)=",I2," ACT. T3*=",I2),
```

```
0179          FOT13(/,"EXP. T2*(P1-107)=",I2," ACT. T2*=",I2),
0180          FOT14(/,"EXP. T1*(P1-18)=",I2," ACT. T1=",I2),
0181          FOT15(/,"EXP. T0*(P1-105)=",I2," ACT. T0*=",I2),
0182          FOT16(/,"EXP. PORI(P1-36)=",I2," ACT. PORI=",I2),
0183          FOT17(/,"EXP. LDADCN*(P1-58)=",I2," ACT. LDADCN*=",I2),
0184          FOT18(/,"EXP. LDDAT(P1-44)=",I2," ACT. LDDAT=",I2),
0185          FOT19(/,"EXP. GO*(NC)=",I2," ACT. GO*(NC)=",I2),
0186          FOT20(/,"EXP. RQSTX*(P1-23)=",I2," ACT. RQSTX*=",I2),
0187          FOT21(/,"EXP. NR/W*(P1-69)=",I2," ACT. NR/W*=",I2),
0188          FOT22(/,"EXP. SCB2*(P1-37)=",I2," ACT. SCB2=",I2),
0189          FOT23(/,"EXP. SCB1*(P1-34)=",I2," ACT. SCB1*=",I2),
0190          FOT24(/,"EXP. ADCNDEC(P1-60)=",I2," ACT.ADCNDEC=",I2),
0191          FOT25(/,"EXP. DBENB*(P1-120)=",I2," ACT. DBENB=",I2),
0192          FOT26(/,"EXP. MAX A AD(P1-97)=",I2," ACT. MUX A ADD=",I2),
0193          FOT27(/,"EXP. ABENB*(P1-104)=",I2," ACT. ABENB*=",I2),
0194          FOT28(/,"EXP. STCNCLR(P1-106)=",I2," ACT. STCNCLR=",I2),
0195          FOT29(/,"EXP. STCNLD(P1-103)=",I2," ACT. STCNLD=",I2),
0196          FOT30(/,"EXP. WDCNO*(P1-89)=",I2," ACT. WDCNO*=",I2),
0197          FOT31(/,"EXP. PARSTB(P1-116)=",I2," ACT. PARSTB=",I2),
0198          FOT32(/,"EXP. 20MSEL(P1-101)=",I2," ACT. 20MSEL=",I2),
0199          FIDL1(/,"EXP. PBUSY(P1-103)=0 ACT. PBUSY=1"),
0200          MSG1(/,"CHECK WAVEFORM ON PBUSY"),
0201          MSG2(/,"TOGGLE SWITCH 2 TO CONTINUE"),
0202          FBT1(/,"U48,U46"),
0203          FBT2(/,"EXP. BTCNLD(P1-17)=0 ACT. BTCNLD=1"),
0204          FCH(/,"CH5",@5,"WD5",@6,"CH6",@6,"WD6",@5,"T5",@6,"T6",@6),
0205          FPM(/,"TIMING ERROR",/,
0206                "20 MSEL NOT PULSING CORRECTLY"),
0207          FPM1(/,"BOARD FREE RUNNING WITH 000 AT IECF,IDE,EIE"),
0208          FPM2(/,"U46,U23,U10,U31,U47,U35,U75",/,
0209                 "R7,R8,C9,C10"),
0210          FPM3(/,"TIMING ERROR",/,
0211                 "SHTENB NOT PULSING"),
0212          FPM4(/,"U8,U43,U36,U9"),
0213          FPM5(/,"EXP. PULSE WIDTH",I3," MICROSEC+/-300 NSEC"),
0214          FPM6(/,"ACT PULSE WIDTH",I4,"X 100 NSEC"),
0215          FPM7(/,"BOARD FREE RUNNING WITH 001 AT IECF,IDE,EIE"),
0216          FPM9(/,"BOARD FREE RUNNING WITH 010 AT IECF,IDE,EIE"),
0217          FPM10(/,"BOARD FREE RUNNING WITH 011 AT IECF,IDE,EIE"),
0218          FPM8(/,"BOARD FREE RUNNING WITH 101 AT IECF,IDE,EIE"),
0219          FCNT1(/,"U12,U25,U42,U2,PORI",/,
0220                  "U25 CARRY WAS SET AFTER ",I4," PULSES",/,
0221                  "COUNTERS WERE INITIALLY SET TO 1"),
0222          FCNT2(/,"U12,U25,U42,U2,PORI",/,
0223                  "COUNTERS DID NOT TIME OUT ON TIME",/,
0224                  "U25 CARRY WAS NOT SET AFTER 254 PULSES",/,
0225                  "COUNTERS WERE INITIALLY SET TO 1"),
0226          FCNT3(/,"U12,U25,U42,U2,PORI",/,
0227                  "U25 CARRY WAS NOT RESET AFTER 256 PULSES"),
0228          FCNT4(/,"U12,U25",/,
0229                  "COUNTERS TIMED OUT PREMATURELY"),
0230          FCNT5(/,"SHTENB NOT SET(P1-98)",/,
0231                  "U12,U25 CANNOT BE INITIALIZED"),
0232          FCNT6(/,"SHTENB CANNOT BE RESET",/,
0233                  "U12,U25 CANNOT BE TESTED"),
0234          FCNT7(/,"EXP. PORI=0 ACT. PORI=1",/,
0235                  "U12,U25 CANNOT BE TESTED"),
0236          FCN8(/,"EXP. TOUT=0 ACT. TOUT=1",/,
0237                 "U12,U25 ARE INITIALISED",/,
0238                 "CHECK U12,U25,U42"),
```

```
0239          FBT3(/,'EXP. BTCNLD(P1-17)=1 ACT. BTCNLD=0");
0240
0241
0242  &*****************************************************
0243  &
0244  &           USER PROCEDURES
0245  &
0246  &*****************************************************
0247
0248
0249  BOOLEAN PROCEDURE SW(A);
0250  VALUE A; INTEGER A;
0251  BEGIN
0252        A←ABIT(A);
0253        IF (KEYS AND A)=A THEN SW←TRUE ELSE SW←FALSE;
0254  END OF SW;
0255
0256  BOOLEAN PROCEDURE HEAD;
0257  BEGIN
0258       IF SW(0) THEN BEGIN SUPPRESS←TRUE; HEAD←FALSE; END
0259       ELSE BEGIN HEAD←TRUE;
0260                  IF NOT ERROR THEN BEGIN WRITE(LU,D0); ERROR←TRUE;END;
0261                  WRITE(LU,D1,E);
0262             END;
0263
0264  END OF HEAD;
0265
0266  PROCEDURE HD(F); FORMAT F;
0267  BEGIN
0268          IF SW(14) THEN LU←8 ELSE LU←6;
0269          IF HEAD THEN WRITE(LU,F);
0270
0271  END OF HD;
0272
0273
0274  PROCEDURE INPUTS;
0275  BEGIN
0276  &
0277  &UUT INPUTS ARE PRINTED ONLY ONCE
0278  &FOR EVERY TEST
0279  &
0280    INFLAG←TRUE;
0281    FOR N←0 TO 7 DO
0282    IF (ABIT(N) AND GPR2)#0 THEN BIN[N]←TRUE ELSE BIN[N]←FALSE;
0283    BIN[3]←NOT BIN[3];
0284    BIN[6]←NOT BIN[6];
0285    FOR N←0 TO 7 DO BIN[N]←(-1)*BIN[N];
0286    HD(FIN1);
0287    IF NOT SUPPRESS THEN
0288    BEGIN
0289    WRITE(LU,FIN2,BIN[0],BIN[1],BIN[2],BIN[3]);
0290    WRITE(LU,FIN3,BIN[4],BIN[6],BIN[7]);
0291    WRITE(LU,FIN4,STATENO);
0292    END;
0293  END;
0294  PROCEDURE MSG5;
0295  BEGIN
0296  &
0297  &PRINTS OUT EXPECTED AND ACTUAL
0298  &STATES OF UUT OUTPUTS
```

```
0299  &
0300    IF NOT INFLAG THEN INPUTS;
0301    IF (LBIT AND WD5)#0 THEN EXPBIT←1 ELSE EXPBIT←0;
0302    IF (LBIT AND CH5)#0 THEN ACTBIT←1 ELSE ACTBIT←0;
0303    M←L+1;
0304    CASE M BEGIN
0305    WRITE(LU,FOT1,EXPBIT,ACTBIT);
0306    WRITE(LU,FOT2,EXPBIT,ACTBIT);
0307    WRITE(LU,FOT3,EXPBIT,ACTBIT);
0308    WRITE(LU,FOT4,EXPBIT,ACTBIT);
0309    WRITE(LU,FOT5,EXPBIT,ACTBIT);
0310    WRITE(LU,FOT6,EXPBIT,ACTBIT);
0311    WRITE(LU,FOT7,EXPBIT,ACTBIT);
0312    WRITE(LU,FOT8,EXPBIT,ACTBIT);
0313    WRITE(LU,FOT9,EXPBIT,ACTBIT);
0314    WRITE(LU,FOT10,EXPBIT,ACTBIT);
0315    WRITE(LU,FOT11,EXPBIT,ACTBIT);
0316    WRITE(LU,FOT12,EXPBIT,ACTBIT);
0317    WRITE(LU,FOT13,EXPBIT,ACTBIT);
0318    WRITE(LU,FOT14,EXPBIT,ACTBIT);
0319    WRITE(LU,FOT15,EXPBIT,ACTBIT);
0320    WRITE(LU,FOT16,EXPBIT,ACTBIT);
0321    END OF CASE M;
0322  END OF MSG5;
0323  PROCEDURE MSG6;
0324  BEGIN
0325  &
0326  &PRINTS OUT EXPECTED AND ACTUAL STATES
0327  &OF UUT OUTPUTS
0328  &
0329    IF NOT INFLAG THEN INPUTS;
0330    IF (LBIT AND WD6)#0 THEN EXPBIT←1 ELSE EXPBIT←0;
0331    IF (LBIT AND CH6)#0 THEN ACTBIT←1 ELSE ACTBIT←0;
0332    M←L+1;
0333    CASE M BEGIN
0334    WRITE(LU,FOT17,EXPBIT,ACTBIT);
0335    WRITE(LU,FOT18,EXPBIT,ACTBIT);
0336    WRITE(LU,FOT19,EXPBIT,ACTBIT);
0337    WRITE(LU,FOT20,EXPBIT,ACTBIT);
0338    WRITE(LU,FOT21,EXPBIT,ACTBIT);
0339    WRITE(LU,FOT22,EXPBIT,ACTBIT);
0340    WRITE(LU,FOT23,EXPBIT,ACTBIT);
0341    WRITE(LU,FOT24,EXPBIT,ACTBIT);
0342    WRITE(LU,FOT25,EXPBIT,ACTBIT);
0343    WRITE(LU,FOT26,EXPBIT,ACTBIT);
0344    WRITE(LU,FOT27,EXPBIT,ACTBIT);
0345    WRITE(LU,FOT28,EXPBIT,ACTBIT);
0346    WRITE(LU,FOT29,EXPBIT,ACTBIT);
0347    WRITE(LU,FOT30,EXPBIT,ACTBIT);
0348    WRITE(LU,FOT31,EXPBIT,ACTBIT);
0349    WRITE(LU,FOT32,EXPBIT,ACTBIT);
0350    END OF CASE M;
0351  END OF MSG6;
0352  PROCEDURE EXMNDATA;
0353  BEGIN
0354  &
0355  &EXAMINES THE STATE OF SINGLE OUTPUTS
0356  &AND CALLS APROPRIATE MESSAGE ROUTINES
0357  &WHEN FAULTY
0358    INFLAG←FALSE;
```

```
0359    FOR L←0 TO 15 DO
0360    BEGIN
0361      LBIT←ABIT(L);
0362      IF (TESTWD5 AND LBIT)#0 THEN MSG5;
0363      IF (TESTWD6 AND LBIT)#0 THEN MSG6;
0364    END;
0365  END;
0366  PROCEDURE EXMNIN;
0367  BEGIN
0368  &
0369  &EXAMINES WHETHER BOARD OUTPUTS AGREE WITH
0370  &EXPECTED RESPONSES.IF NOT IT CALLS ROUTINE TO
0371  &IDENTIFY FAULTY BITS.
0372    CH5←IN(5) AND MASK5;
0373    CH6←IN(6) AND MASK6;
0374    TESTWD5←XOR(CH5,WD5);
0375    TESTWD6←XOR(CH6,WD6);
0376    IF (TESTWD5#0 OR TESTWD6#0) THEN
0377    BEGIN
0378      IF NOT SW(0) THEN EXMNDATA;
0379    END;
0380  END;
0381
0382  PROCEDURE RTOUT;
0383  BEGIN
0384    BITTOUT←BIT(602);
0385    IF BITTOUT=0 THEN
0386    BEGIN
0387      PULSE(CLK10,1); BITTOUT←BIT(602);
0388      IF BITTOUT=0 THEN
0389      BEGIN
0390        IF HEAD THEN BEGIN
0391        WRITE(LU,#("U25,U12 CANNOT BE INITIALIZED"));
0392        SUSPEND←TRUE;
0393        GO TO ENCK; END;
0394      END;
0395      GO TO LBL1;
0396    END;
0397  END;
0398
0399  &*********************************************************
0400  &
0401  &           MAIN PROGRAM
0402  &
0403  &*********************************************************
0404  MAIN:
0405        FETCH(PNC1],5);
0406         LU1←RDLU;
0407         IF PNC3]#3
0408         THEN BEGIN WRITE(LU1,D5); GO TO EN; END;
0409        WRITE(LU1,D7); READ(LU1,#(3A2),FOR I←1 TO 3 DO SERNOCI]);
0410        WRITE(14,#("SER.NO ",3A2),FOR I←1 TO 3 DO SERNOCI]);
0411
0412         IF IN(0)#
0413         (NOT @111)
0414         THEN BEGIN WRITE(LU1,D6); GO TO EN; END;
0415
0416         INGPR(GPR0);
0417         SUPPRESS←FALSE;
0418         PON;                          & POWER ON
```

```
0419
0420  LOOP:
0421        IF SW(14) THEN LU←8 ELSE LU←6;
0422         TEST←TEST+1;
0423         ERROR←FALSE;
0424         GPR0←0;
0425         GPR1←GPR2←GPR3←GPR4←GPR5←GPR6←0; OUT;
0426         RELAY(0);
0427         V1←VOLTS;          &CHECK POWER SUPPLY LEVEL
0428  E←1;   IF ABS(V1-5.00)>.25 THEN
0429         BEGIN
0430           POFF;
0431           HD(PF1);
0432           IF NOT SUPPRESS THEN
0433           WRITE(LU,PF2,V1);
0434           SUSPEND←TRUE;
0435           GO TO ENCK;
0436         END;
0437         IF BIT(308)#1 THEN HD(PF3);
0438         IF BIT(309)#0 THEN HD(PF4);
0439  &
0440  &TEST IF CARD CAN BE INITIATED FROM
0441  &THE NTDS RST LINE
0442  &
0443         FOR NOTIMES←1 TO 3 DO
0444         BEGIN
0445         GPR2←@100; OUT;
0446         PULSE(CLK20,1);
0447         STATEN0←0;
0448         WD5←@135004; MASK5←@177774;
0449         WD6←@142411; MASK6←@157753;
0450  E←2;    EXMNIN;
0451         GPR2←@110; OUT;       &SET NTDS RST TO 1
0452                               &AND REBLX TO 1
0453         WD5←@135004;
0454         WD6←@142411;
0455  E←3;    EXMNIN;
0456                               &SET CONTROL INPUTS EIE,IDE,IECF
0457                               &ONE AT A TIME TO 1
0458         GPR2←@101; OUT;
0459         WD5←@135104;
0460         WD6←@142411;
0461  E←4;    EXMNIN;
0462         GPR2←@102; OUT;
0463  E←5;    EXMNIN;
0464         GPR2←@104; OUT;
0465  E←6;     EXMNIN;
0466  LBL1:  GPR2←4; OUT; WAIT(1);        &INITIALIZE U25,U12
0467         RTOUT;
0468         WD5←@035100; MASK5←@177770;
0469         WD6←@146411; MASK6←@157613;
0470  E←601; EXMNIN;
0471         PULSE(CLK10,1); RTOUT; PULSE(CLK10,1); RTOUT;
0472         WD5←@035100; MASK5←@177774;    &SET BTCNLD TO 0
0473  E←602; EXMNIN;
0474         GPR2←0; OUT; GPR2←2; OUT;
0475         PULSE(CLK10,1);
0476         WD5←@035106; MASK5←@177776;    &SET SHTENB TO 1 AND
0477  E←603; EXMNIN;                       &RESET BTCNLD TO 1
0478         PULSE(CLK10,3);
```

```
0479          WD5←@035104; MASK5←@177777;     &SET EOTWD TO 0 AND
0480  E←604;  EXMNIN;                         &RESET SHTENB TO 0
0481          PULSE(CLK10,1);
0482          WD5←@035105;                    &RESET EOTWD TO 1
0483  E←605;  EXMNIN;
0484          PULSE(CLK10,1);                  &SET NDAT2* TO 1
0485  &
0486  &COMMNAD TO BEGIN TRANSMITION
0487  &
0488          GPR2←@5; OUT;
0489          WD5←@035105; MASK5←@177777;
0490          WD6←@156411; MASK6←@157753;
0491  E←9;     EXMNIN;
0492          PULSE(CLK10,1);                  &ADVANCE TO T1;
0493          STATENO←1;
0494          WD5←@055125;
0495          WD6←@146501; MASK6←@157713;
0496  E←10;     EXMNIN;
0497          PULSE(CLK10,1);                  &SET BUSY
0498  E←11;     EXMNIN;
0499          GPR2←@15; OUT;                   &RESEET MC RQST FF
0500          WD5←@055125; MASK5←@177777;
0501          WD6←@145501; MASK6←@157713;
0502  E←12;     EXMNIN;
0503  &
0504  &TEST SEL/MUX U7,U20
0505  &
0506          FOR I←0 TO 7 DO
0507          BEGIN
0508            GPR1←HWADD←BIT(I); OUT;
0509            HWADDIN←IN(3) AND @377;
0510  E←13;       IF HWADD#HWADDIN THEN
0511            BEGIN
0512              HD(HW1);
0513              IF NOT SUPPRESS THEN
0514              WRITE(LU,HW2,HWADD,HWADDIN);
0515            END;
0516          END;
0517          PULSE(CLK10,1);                   &RESET MC RQST FF
0518          WD5←@055125;
0519          WD6←@145511; MASK6←@157713;
0520  E←14;     EXMNIN;
0521          GPR4←DTOUTENOTIMES]; OUT;            &SET ADDRESS CNTR TO @252
0522                                              &OR @125 OR @377
0523                                              &AND WDCNO TO @12 OR 5
0524          GPR2←@215; OUT;                     &SET DSTRB TO 1
0525          WD5←@055125; MASK5←@177777;
0526          WD6←@165512; MASK6←@177713;
0527  E←15;     EXMNIN;
0528          PULSE(CLK20,3);
0529          PULSE(CLK10,1);              &SET STCNENB TO 1
0530          WD5←@055175;
0531          WD6←@165512;
0532  E←16;     EXMNIN;
0533          GPR2←@15; OUT;               &SET DSTRB TO 0
0534          WD5←@055175;
0535          WD6←@165511;
0536  E←17;     EXMNIN;
0537          PULSE(CLK20,3);              &SET ENBSTR TO 0
0538                                       &SET STCNENB TO 0
```

```
0539          WD5←@055125;
0540          WD6←@165511;
0541  E←18;     EXMNIN;
0542          GPR2←@5; OUT;                  &SET REBLX TO 0
0543          PULSE(CLK10,1);                &AND ADVANCE TO T2
0544          STATENO←2;
0545          WD5←@065125;
0546          WD6←@166501;
0547  E←19;     EXMNIN;
0548          GPR2←@15; OUT;                 &SET REBLX TO 1
0549          WD5←@065125;
0550          WD6←@164101;
0551  E←20;     EXMNIN;
0552          NMADD←DTI(NOTIMES);
0553          NMADDIN←IN(3) AND @377;        &CHECK NMADD
0554  E←21;     IF NMADD#NMADDIN THEN
0555          BEGIN
0556            HD(HW1);
0557            IF NOT SUPPRESS THEN
0558            WRITE(LU,HW2,NMADD,NMADDIN);
0559          END;
0560          STWD←2;                        &CHECK THE STATUS WD
0561          TEMP←GPR4;
0562          GPR4←@177777; OUT;
0563  E←22;   STWDIN←IN(4) AND 7;
0564          IF STWDIN#STWD THEN
0565          BEGIN
0566            HD(FST);
0567            IF NOT SUPPRESS THEN
0568            WRITE(LU,FST1,STWD,STWDIN);
0569          END;
0570          STWD←6;
0571          SET(PARITY,1);
0572  E←221;  STWDIN←IN(4) AND 7;
0573          IF STWD#STWDIN THEN
0574          BEGIN
0575            HD(FST);
0576            IF NOT SUPPRESS THEN
0577            WRITE(LU,FST1,STWD,STWDIN);
0578          END;
0579          SET(PARITY,0);
0580          GPR4←TEMP; OUT;
0581          GPR2←@215; OUT;                          &SET DSTRB TO 1
0582          PULSE(CLK10,1);                          &RESET MC RQST FF
0583          WD5←@065125;
0584          WD6←@164111;
0585  E←23;     EXMNIN;
0586          PULSE(CLK20,3);                          &SET ENBSTRB
0587          WD5←@065175;
0588          WD6←@164111;
0589  E←24;     EXMNIN;
0590          PULSE(CLK10,1);                          &SET P TO 1
0591  E←25;     EXMNIN;
0592          GPR2←5; OUT;                             &SET REBLX TO 0
0593                                                   &AND DSTRB TO 0
0594          PULSE(CLK20,3);
0595          WD5←@065125;
0596          WD6←@166511;
0597  E←26;     EXMNIN;
0598            PULSE(CLK10,1);
```

```
0599          STATENO←3;                              &ADVANCE TO T3
0600          WD5←@071125;
0601          WD6←@166501;
0602  E←27;     EXMNIN;
0603            GPR4←@125252; OUT;                    &SDB1-16 ARE SET
0604                                                  &@125252. SDB1 IS AT 0
0605                                                  &CAPTURE THIS DATA ON NDAT BUS
0606          GPR2←@15; OUT;                          &SET REBLX TO 1
0607          WD5←@071125;
0608          WD6←@164701;
0609  E←28;     EXMNIN;
0610          PULSE(CLK10,1);                         &RESET MC RQST FF
0611          WD5←@071125;
0612          WD6←@164711;
0613  E←29;     EXMNIN;
0614          NMADD←NMADD-1;
0615  E←30;   NMADDIN←IN(3) AND @377;
0616          IF NMADDIN#NMADD THEN
0617          BEGIN
0618            HD(HW1);
0619            IF NOT SUPPRESS THEN
0620            WRITE(LU,HW2,NMADD,NMADDIN);
0621          END;
0622          GPR2←@215; OUT;
0623          PULSE(CLK20,3);                       &SET ENBSTR TO 1
0624          WD5←@071175;
0625          WD6←@164711;
0626  E←31;     EXMNIN;
0627          PULSE(CLK10,1);                       &SET P TO  1
0628  E←32;     EXMNIN;
0629          GPR2←5; OUT;                          &SET DSTRB AND REBLX TO 0
0630                                                &SCB1 IS SET TO 0
0631          PULSE(CLK20,3);
0632          PULSE(CLK10,1);                       &ADVANCE TO T4
0633          STATENO←4;
0634          WD5←@077105;
0635          WD6←@166401; MASK6←@177753;
0636  E←33;     EXMNIN;
0637          SDB←@125252;                          &CHECK SDB1-16
0638          SDBIN←IN(1);
0639  E←3301;IF SDB#SDBIN THEN
0640          BEGIN
0641            HD(FSDB);
0642            IF NOT SUPPRESS THEN
0643            WRITE(LU,FSDB1,SDB,SDBIN);
0644          END;
0645          GPR2←@15; OUT;                        &SET REBLX TO 1
0646          WD5←@077105;
0647          WD6←@164501;
0648  E←34;     EXMNIN;
0649          NMADD←NMADD-1;
0650          NMADDIN←IN(3) AND @377;
0651  E←35;   IF NMADD#NMADDIN THEN
0652          BEGIN
0653            HD(HW1);
0654            IF NOT SUPPRESS THEN
0655            WRITE(LU,HW2,NMADD,NMADDIN);
0656          END;
0657          PULSE(CLK10,1);                       &RESET MC FF TO 1
0658          WD5←@077105;
```

```
0659          WD6←@164511;
0660    E←36;   EXMNIN;
0661          GPR2←@215; OUT;                 &STROBE DATA IN
0662          PULSE(CLK20,3);
0663          WD5←@077145;
0664          WD6←@164511;
0665    E←37;   EXMNIN;
0666          PULSE(CLK10,1);                 &SET BTCNLD TO 1
0667          WD5←@077141;
0668          WD6←@164511;
0669    E←38;   EXMNIN;
0670          GPR2←@5; OUT;                   &SET DSTRB TO 0
0671          PULSE(CLK20,3);                 &AND REBLX TO 0
0672          WD5←@077101;
0673          WD6←@166411;
0674    E←39;   EXMNIN;
0675          PULSE(CLK10,1);                 &SET SHTENB TO 1
0676          WD5←@077107;
0677          WD6←@166411;
0678    E←40;   EXMNIN;
0679          PULSE(CLK10,33);                &TIME OUT BIT CNTR
0680    E←41;   EXMNIN;
0681          PULSE(CLK10,1);                 &SET EOTWD TO 1
0682                                          &AND RESET SHTENB TO 0
0683          WD5←@077104;
0684          WD6←@122411;
0685    E←42;   EXMNIN;
0686          PULSE(CLK10,1);                 &ADVANCE TO T0
0687          STATENO←0;
0688          WD5←@035105;
0689          WD6←@166411;
0690    E←43;   EXMNIN;
0691          PULSE(CLK10,1);                 &LOAD BIT COUNTER
0692          WAIT(1);
0693          WD5←@035101;
0694          WD6←@166411;
0695  E←4301;   EXMNIN;
0696          PULSE(CLK10,1);                 &SET SHTENB
0697          WD5←@035107;
0698          WD6←@166411;
0699    E←44;   EXMNIN;
0700          PULSE(CLK10,2);                 &TIME OUT BIT CNTR
0701    E←45;   EXMNIN;
0702          PULSE(CLK10,1);                 &RESET SHTENB TO 0 AND
0703          WD5←@035104;                    &SET EOTWD TO 0
0704          WD6←@166411;
0705    E←46;   EXMNIN;
0706          PULSE(CLK10,1);                 &RESET EOTWD* TO 1
0707          WD5←@035105;
0708          WD6←@166411;
0709  E←461;  EXMNIN;
0710          GPR2←2; OUT;
0711          PULSE(CLK10,1);                &CLEAR U23
0712          SDB←@125252;                   &CHECK SDB17-32
0713          SDBIN←IN(2);
0714    E←47;   IF SDB≠SDBIN THEN
0715            BEGIN
0716              HD(SDB2);
0717              IF NOT SUPPRESS THEN
0718              WRITE(LU,SDB3,SD9,SDBIN);
```

```
0719          END;
0720        GPR2←6; OUT;                    &SEND COMMAND TO BEGIN
0721                                        &TRNSMITION
0722        WD5←@035105;
0723        WD6←@176411;
0724  E←48;   EXMNIN;
0725        PULSE(CLK10,1):                       &ADVANCE TO T7
0726        STATENO←7;
0727        GPR2←2;
0728        GPR4←@52525; OUT;                     &SET NDAT1 TO 1
0729                                              &ADDRESS CNTR IS AT @250
0730                                              &OR @123
0731        WD5←@074125;
0732        WD6←@166501;
0733  E←49;   EXMNIN;
0734        FOR WDCNT←STARTENOTIMES] TO 14 DO
0735        BEGIN
0736          IF WDCNT=STARTENOTIMES] THEN SCB2←0 ELSE SCB2←@40;
0737          GPR2←@12; OUT;                       &MEMORY ACKNOWLEDGE
0738          WD5←@074125;
0739          WD6←@164701 OR SCB2;
0740  E←50;     EXMNIN;
0741          NMADD←NMADD-1;
0742          NMADDIN←IN(3) AND @377;               &NMADD IS DECREMENTED
0743                                                &FROM @250 TO @237
0744  E←5001; IF NMADD#NMADDIN THEN
0745          BEGIN
0746            HD(HW1);
0747            IF NOT SUPPRESS THEN
0748            WRITE(LU,HW2,NMADD,NMADDIN);
0749          END;
0750          PULSE(CLK10,1);                      &RESET MC FF
0751          WD5←@074125;
0752          WD6←@164711 OR SCB2;
0753  E←51;   EXMNIN;
0754          GPR2←@212; OUT;                     &SET DSTRB TO 1
0755          PULSE(CLK20,3);                     &SDB1 IS SET TO 1
0756          WD5←@074175;
0757          WD6←@164711 OR SCB2;
0758  E←52;   EXMNIN;
0759          PULSE(CLK10,1);                     &SET P
0760  E←53;   EXMNIN;
0761          GPR2←2; OUT;                         &SET REBLX AND DSTRB
0762                                               &TO 0
0763          PULSE(CLK20,3);
0764          WD5←@074125;
0765          WD6←@166551;
0766  E←54;   EXMNIN;
0767          SDB←@052525;
0768          SDBIN←IN(1);                        &CHECK SDB1-16
0769  E←55;   IF SDB#SDBIN THEN
0770          BEGIN
0771            HD(FSDB);
0772            IF NOT SUPPRESS THEN
0773            WRITE(LU,FSDB1,SDB,SDBIN);
0774          END;
0775          PULSE(CLK10,1);
0776          STATENO←8;                          &AD VANCE TO T8
0777          WD5←@075505;                        &AND SET MC REQ FF
0778          WD6←@166541;
```

```
0779  E←56;    EXMNIN;
0780            GPR2←@12; OUT;                    &MEMORY ACKNOWLEDGE
0781            WD5←@075505;                      &AND INCREMENT WORD CNT
0782            WD6←@164741;
0783  E←57;    EXMNIN;
0784            PULSE(CLK10,1);                   &RESET MC REQ FF
0785            WD5←@075505;
0786            WD6←@164751;
0787  E←58;    EXMNIN;
0788  E←5801;  NMADD←NMADD-1;
0789            NMADDIN←IN(3) AND @377;
0790            IF NMADD#NMADDIN THEN
0791            BEGIN
0792              HD(HW1);
0793              IF NOT SUPPRESS THEN
0794              WRITE(LU,HW2,NMADD,NMADDIN);
0795            END;
0796            GPR2←@212; OUT;                   &SET DSTRB TO 1
0797            PULSE(CLK20,3);                   &ND CAAPTURE DATA
0798            WD5←@075545;                      &FROM DATA BUS
0799            WD6←@164751;
0800  E←59;    EXMNIN;
0801            PULSE(CLK10,1);
0802                                              &SET BIT CNTR TO 0
0803            WD5←@075541;
0804            WD6←@164751;
0805  E←60;    EXMNIN;
0806            GPR2←@2; OUT;                     &SET DSTRB TO 0
0807            PULSE(CLK20,3);                   &CARRY OF WORD CNTR
0808            WD5←@075501;                      &IS SET AT THIS POINT
0809            WD6←@166551;
0810  E←61;    EXMNIN;
0811            PULSE(CLK10,1);                    &SET SHTENB TO 1
0812            WD5←@075507;
0813            WD6←@166551;
0814  E←62;    EXMNIN;
0815            SDB←@052525;                       &CHECK SD317-SDB32
0816            SDBIN←IN(2);
0817  E←621;   IF SDB#SDBIN THEN
0818            BEGIN
0819              HD(SDB2);
0820              IF NOT SUPPRESS THEN
0821              WRITE(LU,SDB3,SDB,SDBIN);
0822            END;
0823            PULSE(CLK10,33);
0824  E←63;    EXMNIN;
0825            PULSE(CLK10,1);                    &SET ETOWD TO 1
0826            WD5←@075504;
0827            WD6←@122551;
0828  E←65;    EXMNIN;
0829            PULSE(CLK10,1);                   &CLEAR STATE CNTR
0830            STATEND←0;
0831            WD5←@035105;
0832            WD6←@166411;
0833  E←66;    EXMNIN;
0834            PULSE(CLK10,1);                   &SET BITCNLD TO 1
0835            WAIT(1);
0836            WD5←@035101;
0837            WD6←@166411;
0838  E←67;    EXMNIN;
```

```
0839            PULSE(CLK10,1);                  &SET SHTENB TO 1
0840            WD5←@035107;
0841            WD6←@166411;
0842  E←68;     EXMNIN;
0843            PULSE(CLK10,2);                  &TIME OUT BIT CNTR
0844  E←69;     EXMNIN;
0845            PULSE(CLK10,1);                  &SET EOTWD TO 1
0846            WD5←@035104;
0847            WD6←@166411;
0848  E←70;     EXMNIN;
0849            PULSE(CLK10,1);                  &SET EOTWD TO 0
0850            WD5←@035105;
0851            WD6←@166411;
0852  E←71;     EXMNIN;
0853            GPR2←6; OUT;
0854            WD5←@035105;
0855            WD6←@176411;
0856  E←72;     EXMNIN;
0857            PULSE(CLK10,1);                  &ADVANCE TO T7
0858            STATENO←7;
0859            GPR2←2; OUT;
0860            WD5←@074125;
0861            WD6←@166541;
0862  E←73;     EXMNIN;
0863          END;
0864                                             &UUT IS IN T7 ON
0865                                             &COMPLETION OF LOOP
0866          GPR2←@12; OUT;                     &SET REBL X TO 1
0867          WD5←@074125;
0868          WD6←@164741;
0869  E←74;     EXMNIN;
0870          NMADD←NMADD-1;
0871          NMADDIN←IN(3) AND @377;
0872  E←75;   IF NMADD≠NMADDIN THEN
0873          BEGIN
0874            HD(HW1);
0875            IF NOT SUPPRESS THEN
0876            WRITE(LU,HW2,NMADD,NMADDIN);
0877          END;
0878          PULSE(CLK10,1);                    &RESET MC FF
0879          WD5←@074125;
0880          WD6←@164751;
0881  E←76;     EXMNIN;
0882          GPR2←@212; OUT;                    &SET DSTRB TO 1
0883          PULSE(CLK20,3);
0884          WD5←@074175;
0885          WD6←@164751;
0886  E←77;     EXMNIN;
0887          PULSE(CLK10,1);                    &SET P TO 1
0888  E←78;     EXMNIN;
0889          GPR2←2; OUT;                       &SET DSTRB TO 0
0890          PULSE(CLK20,3);
0891          WD5←@074125;
0892          WD6←@166551;
0893  E←79;     EXMNIN;
0894          SDB←@052525;
0895          SDBIN←IN(1);
0896  E←60;   IF SDB≠SDBIN THEN
0897          BEGIN
0898            HD(FSDB);
```

```
0899            IF NOT SUPPRESS THEN
0900            WRITE(LU,FSDB1,SDB,SDBIN);
0901          END;
0902          PULSE(CLK10,1);                          &ADVANCE TO T8
0903          STATENO←8;
0904          WD5←@075505;
0905          WD6←@166541;
0906  E←81;     EXMNIN;
0907          GPR2←@12; OUT;                           &MEMORY ACKOWLEDGE
0908          WD5←@075505;
0909          WD6←@164741;
0910  E←82;     EXMNIN;
0911          NMADD←NMADD-1;
0912          NMADDIN←IN(3) AND @377;
0913  E←83;   IF NMADD≠NMADDIN THEN
0914          BEGIN
0915            HD(HW1);
0916            IF NOT SUPPRESS THEN
0917            WRITE(LU,HW2,NMADD,NMADDIN)
0918          END;
0919          PULSE(CLK10,1);                          &RESET MC FF
0920          WD5←@075505;
0921          WD6←@164751;
0922  E←831; EXMNIN;
0923          GPR2←@212; OUT;
0924          PULSE(CLK20,3);                          &SET ENBSTR TO 1
0925          WD5←@075545;
0926          WD6←@164751;
0927  E←84;     EXMNIN;
0928          PULSE(CLK10,1);                          &SET BITCNTLD TO 1
0929          WAIT(1);
0930          WD5←@075541;
0931          WD6←@164751;
0932  E←85;     EXMNIN;
0933          GPR2←2; OUT;                             &SET DSTRB TO 0
0934          PULSE(CLK20,3);                          &SET REBLX TO 0
0935          WD5←@075501;                             &WDCNO* SET TO 0
0936          WD6←@146551;
0937  E←86;     EXMNIN;
0938  E←861;    SDB←@052525;
0939            SDBIN←IN(2);
0940            IF SDB≠SDBIN THEN
0941            BEGIN
0942              HD(FSDB);
0943              IF NOT SUPPRESS THEN
0944              WRITE(LU,FSDB1,SDB,SDBIN);
0945            END;
0946          PULSE(CLK10,1);                          &SET SHTENB TO 1
0947          WD5←@075507;
0948          WD6←@146551;
0949  E←87;     EXMNIN;
0950          PULSE(CLK10,33);
0951  E←88;     EXMNIN;
0952          PULSE(CLK10,1);                          &SET EOTWD TO 1
0953          WD5←@075504;
0954          WD6←@116551;
0955  E←89;     EXMNIN;
0956          PULSE(CLK10,1);                          &ADVANCE TO T9
0957          STATENO←9;
0958          WD5←@075305;
```

```
1019          WD6←@146411;
1020  E←102;    EXMNIN;
1021            END OF NOTIMES;
1022  &
1023  &IDLE ROUTINE
1024  &
1025          GPR2←@103; OUT;
1026          GPR2←0; OUT;
1027          WAIT(1);
1028  E←103; IF  BIT(PBUSY)#0 THEN
1029          BEGIN
1030            INPUTS; INFLAG←FALSE;
1031            IF NOT SUPPRESS THEN
1032            WRITE(LU,FIDL1);
1033            GO TO L10;
1034          END;
1035          GPR3←2; OUT;            &ENABLE CLOCK
1036          RELAY(814);   WAIT(2);
1037          NOPULSES←PMEAS(0);
1038          RELAY(-814);
1039  E←1031; IF ABS(NOPULSES-200)>35 THEN
1040           BEGIN
1041             HD(FPM);
1042             IF NOT SUPPRESS THEN
1043             BEGIN
1044               WRITE(LU,FPM1); WRITE(LU,FPM2);
1045               WRITE(LU,FPM5,20); WRITE(LU,FPM6,NOPULSES);
1046             END;
1047           END;
1048           RELAY(815); WAIT(2);
1049           NOPULSES←PMEAS(0);
1050           RELAY(-815);
1051  E←1032; IF ABS(NOPULSES-200)>35 THEN
1052           BEGIN
1053             HD(FPM3);
1054             IF NOT SUPPRESS THEN
1055             BEGIN
1056               WRITE(LU,FPM4);
1057               WRITE(LU,FPM5,20);
1058               WRITE(LU,FPM6,NOPULSES);
1059             END;
1060           END;
1061           RELAY(814); WAIT(2);
1062           FOR J←1 TO 3 DO
1063           BEGIN
1064           GPR2←J; OUT;                &ISSUE NO RESPONSE COMMAND
1065           NOPULSES←PMEAS(0);
1066  E←1003; IF ABS(NOPULSES-200)>35 THEN
1067          BEGIN
1068            HD(FPM);
1069            IF NOT SUPPRESS THEN
1070            BEGIN
1071              CASE J BEGIN
1072              WRITE(LU,FPM7);
1073              WRITE(LU,FPM9);
1074              WRITE(LU,FPM10);
1075              END OF CASE J;
1076              WRITE(LU,FPM5,20);
1077              WRITE(LU,FPM6,NOPULSES);
1078            END;
```

```
1079          END;
1080          END;
1081          GPR3←0; OUT;              &STOP CLOCK
1082  &
1083  &TEST OUT U12,U25
1084  &
1085  L10:   GPR2←@100; OUT;
1086         GPR2←0; OUT;
1087         GPR2←4; OUT;
1088         PULSE(CLK10,3);
1089  E←104;IF BIT(SHTENB)≠1 THEN HD(FCNT5);
1090         GPR2←0; OUT; GPR2←2; OUT;
1091         PULSE(CLK10,3);
1092  E←105;IF BIT(SHTENB)≠0 THEN HD(FCNT6);
1093  E←106;IF BIT(PORI)≠0 THEN HD(FCNT7);
1094  E←1061; IF BIT(TOUT)≠1 THEN HD(FCN8);
1095         FOR I←1 TO 254 DO
1096         BEGIN
1097           PULSE(CLK10,1);
1098           FPORI←BIT(PORI);
1099           FTOUT←BIT(TOUT);
1100           IF NOT SUPPRESS THEN
1101           BEGIN
1102  E←107;   IF (I<254 AND (FPORI=1 OR FTOUT=0)) THEN
1103           BEGIN
1104             HD(FCNT4);
1105             WRITE(LU,FCNT1,I);
1106           END;
1107  E←108;   IF (I=254 AND (FPORI=0 OR FTOUT=1)) THEN HD(FCNT2);
1108           END;
1109         END;
1110         PULSE(CLK10,1);
1111         FPORI←BIT(PORI);
1112         FTOUT←BIT(TOUT);
1113  E←109;IF (BIT(PORI)≠0 OR BIT(TOUT)≠1) THEN HD(FCNT3);
1114  &**********************************************************
1115  &
1116  &            END OF PROGRAM
1117  &
1118  &**********************************************************
1119
1120  ENCK:
1121           IF SW(15) THEN GO TO LOOP;
1122           POFF;
1123         IF SUPPRESS THEN WRITE(LU,D2);
1124         IF TEST>1 THEN WRITE(LU,D3,TEST);
1125         IF SUSPEND THEN WRITE(LU,D4);
1126         WRITE(LU,#("SERIAL OUTPUT CONTROLLER TEST COMPLETE"));
1127
1128
1129  EN:
1130         WRITE(LU,DUMMY);
1131
1132  END$
1133  COMENT
1134
1135  SERIAL I/O CONTROL NARRATIVE
1136
1137  NOTES:
1138       THIS NARRATIVE TRNSLATES THE TEST PROGRAM FOR
```

```
1139        THE SERIAL I/O CONTROL CARD INTO ENGLISH.REFERENCE
1140        TO FIRMWARE OF THE TEST SYSTEM IS AVOIDED AS FAR AS
1141        PRACTICAL.
1142        IN THE CONTENT OF THIS DOCUMENT THE FOLLOWING
1143        EXPRESSIONS SHALL BE INTERPRETED AS OUTLINED
1144        BELOW:
1145        UUT=UNIT UNDER TEST
1146        @=OCTAL NUMBER
1147        0=TTL LO
1148        1=TTL HI
1149        XA6=16 BIT WORD. THE BITS IN WORD REPRESENT THE
1150            STATE OF THE FOLLOWING UUT OUTPUTS
1151            1. LDADCN*    (J1-58)   (LSB)
1152            2. LDDAT      (J1-44)
1153            3. TOUT*      (J1-21)
1154            4. RQSTX*     (J1-23)
1155            5. NR/W       (J1-69)
1156            6. SCB2*      (J1-37)
1157            7. SCB2*      (J1-34)
1158            8. ADCNDEC    (J1-60)
1159            9. DBENB*     (J1-120)
1160           10. MAX "A" AD (J1-97)
1161           11. ABENB*     (J1-104)
1162           12. STCNCLR    (J1-106)
1163           13. STCNLD     (J1-103)
1164           14. WDCNNO*    (J1-89)
1165           15. PARSTB     (J1-116)
1166           16. 20 MSEL    (J1-101)  (MSB)
1167
1168        XA5=16 BIT WORD. THE BITS IN WORD REPRESENT THE
1169            STATE OF THE FOLLOWING UUT OUTPUTS
1170            1. EOTWD*     (J1-96)   (LSB)
1171            2. SHTENB     (J1-98)
1172            3. BTCTLD*    (J1-17)
1173            4. STCNENB    (J1-115)
1174            5. TP12       (J1-118)
1175            6. ENBSTR     (J1-109)
1176            7. PBUSY      (J1-118)
1177            8. T9         (J1-119)
1178            9. T8         (J1-83)
1179           10. T7*        (J1-113)
1180           11. T4         (J1-117)
1181           12. T3*        (J1-47)
1182           13. T2*        (J1-107)
1183           14. T1*        (J1-13)
1184           15. T0*        (J1-105)
1185           16. PORI       (J1-36)   (MSB)
1186        MASK6=16 BIT WORD USED TO MASK OUT XA6 BITS
1187              THAT ARE IN INDETEMINATE STATE.IF A BIT
1188              IN MASK6 IS SET TO 0 THEN BIT OF SAME SIGNIFICANT
1189              POSITIOM IN XA6 IS MASKED OUT.
1190        MASK5=SAME AS MASK6 BUT FOR XA5
1191        XA2=8 BIT WORD. BITS REPRESENT STATE OF FOLLOWING
1192            UUT INPUTS
1193            1. EIE        (J1-91)    (LSB)
1194            2. IDE        (J1-88)
1195            3. IECF       (J1-95)
1196            4. REBLX*     (J1-31)
1197            5. PARITY     (J1-99)
1198            7. NDTS RST   (J1-13)
```

```
1199          8. DSTRB     (J1-70)     (MSB)
1200        CLK10=IS THE UUT INPUT FOR THE 10 MHZ CLOCK
1201              (SR 10 MHZ A/B* AND SR 10 MHZ A/B)
1202        CLK20=IS THE UUT INPUT FOR THE 20 MHZ CLOCK
1203        DT[1:3]=IS AN ARRAY.THE ELEMENTS OF THE ARRAY
1204                HAVE THE FOLLOWING VALUES:
1205                1. @252
1206                2. @125
1207                3. @377
1208        DTOUT[1:3]=ARRAY.THE ELEMENTS OF THE ARRAY HAVE
1209                THE FOLLOWING VALUES:
1210                1. @5252
1211                2. @2525
1212                3. @377
1213        START[1:3]=ARRAY.THE ELEMENTS OF THE ARRAY HAVE THE
1214                FOLLOWING VALUES:
1215                1. 11
1216                2. 6
1217                3. 1
1218
1219  START:
1220
1221
1222  1.      CHECK VALIDITY OF PART NO. AND INTERFACE ID.
1223          (PART NO=855075-01,INTERFACE ID=@111)
1224  2.      SET UP TEST CONDITIONS AND TURN POWER ON.
1225  3.      VERIFY THAT VOLTAGE TO UUT IS 5.00+/-.25 V
1226  3.1     TERMINATE PROGRAM IF VOLTAGE IS FOUND OUT OF
1227          TOLERANCE.
1228  3.2     CHECK STATE OF PULLUP 1(J1-82)
1229  3.3     UUT IS FAULTY IF PULLUP1 IS SET TO 0
1230  3.4     CHECK STATE OF PULLDN 1(J1-81)
1231  3.5     UUT IS FAULTY IF PULLDN 1 IS SET TO 1
1232                                        ERROR=1
1233  4.      CONSECUTIVELY FOR NOTIMES 1 TO 3
1234          EXECUTE THE FOLLOWING
1235  4.1     SET XA2 TO @100.
1236  4.1.1   PULSE CLK20 ONCE.
1237  4.1.2   VERIFY THAT XA5=@135004 WITH MASK5=@17774 AND
1238          XA6=@142411 WITH MASK6=@157753
1239                                        ERROR=2
1240  4.2     SET XA2 TO @110.
1241  4.2.1   VERIFY THAT XA5=@135004 WITH MASK5=@177774
1242          AND XA6=@142411 WITH MASK6=@157753.
1243                                           ERROR=3
1244  4.3     SET XA2=@101.
1245  4.3.1   VERIFY THAT XA5=@135104 WITH MASK5=@177774 AND
1246          XA6=@142411 WITH MASK6=@157753.
1247                                        ERROR=4
1248  4.4     SET XA2 TO @102
1249  4.4.1   VERIFY THAT XA5=@135104 WITH MASK5=17774 AND
1250          XA6=@142411 WITH MASK6=@157753.
1251                                        ERROR=5
1252  4.5   SET XA2 TO @104.
1253  4.5.1 VERIFY THAT XA5=@135104 WITH MASK5=@17774 AND
1254        XA6=@142411 WITH MASK6=@157753.
1255                                        ERROR=6
1256  4.6     SET XA2 TO 4.
1257  4.6.1   WAIT FOR 1 MSEC.
1258  4.6.2   VERIFY THAT XA5=@035100 WITH MASK5=@177770
```

```
1259          AND XA6=@146411 WITH MASK6=@157613.
1260                                              ERROR=601
1261    4.7   PULSE CLK10 TWICE.
1262    4.7.1 VERIFY THAT XA5=@035100 WITH MASK5=@177774
1263          AND XA6=@146411 WITH MASK6=@157613.
1264                                                ERROR=602
1265    4.8   SET XA2 TO 0.
1266    4.8.1 PULSE CLK10 ONCE.
1267    4.8.2 VERIFY THAT XA5=@035106 WITH MASK5=@177776
1268          AND XA6=@146411 WITH MASK6=@157613.
1269                                                 ERROR=603
1270    4.9   PULSE CLK10 3 TIMES.
1271    4.9.1 VERIFY THAT XA5=@035104 WITH MASK5=@177777
1272          AND XA6=@146411 WITH MASK6=@157613.
1273                                                ERROR=604
1274    4.10  UNTL THE END OF THE NARRATIVE
1275          THE VALUE OF MASK5 REMAINS UNCHANGED TO @177777
1276    4.11  PULSE CLK10 ONCE.
1277    4.11.1 VERIFY THAT XA5=@035105 AND
1278          XA6=@146411 WITH MASK6=@157613.
1279                                              ERROR=605
1280    4.12  PULSE CLK10 ONCE.
1281    4.13  SET XA2 TO 5.
1282    4.13.1 VERIFY THAT XA5=@035105 AND
1283          XA6=@156411 WITH MASK6=@157753.
1284                                              ERROR=9
1285    4.14  PULSE CLK10 ONCE.
1286    4.14.1 VERIFY THAT XA5=@055125 AND
1287          XA6=@146501 WITH MASK6=@157713.
1288                                              ERROR=10
1289    4.15  PULSE CLK10 ONCE.
1290    4.15.1 VERIFY THAT XA5=@055125 AND
1291          XA6=@146501 WITH MASK6=@157713.
1292                                              ERROR=11
1293    4.16  SET XA2 TO @15.
1294    4.16.1 VERIFY THAT XA5=055125 AND
1295          XA6=@145501 WITH MASK6=@157713.
1296                                              ERROR=12
1297    4.17    FOR I=1 TO 8
1298    4.17.1  SET HWADD8*-HWADD1* TO I**2
1299    4.17.2  VERIFY THAT NMADD8*-NMADD1* IS SET TO I**2
1300                                              ERROR=13
1301    4.18  PULSE CLK10 ONCE.
1302    4.18.1 VERIFY THAT XA5=@055125 AND
1303          XA6=@145511 WITH MASK6=@157713.
1304                                              ERROR=14
1305    4.19  OUTPUT DTOUT[NOTIMES] ON NDAT16*-NDAT1*
1306    4.20  SET XA2 TO @215.
1307    4.20.1 VERIFY THAT XA5=@055125 AND
1308          XA6=@165512 WITH MASK6=@177713.
1309                                              ERROR=15
1310    4.21  PULSE CLK20 3 TIMES.
1311    4.21.1 PULSE CLK10 ONCE.
1312    4.21.2 VERIFY THAT XA5=@055175 AND
1313          XA6=@165512 WITH MASK6=@177713.
1314                                              ERROR=16
1315    4.22  SET XA2 TO @15.
1316    4.22.1 VERIFY THAT XA5=@055175 AND
1317          XA6=@165511 WITH MASK6=@177713.
1318                                              ERROR=17
```

```
4.23   PULSE CLK20 3 TIMES.
4.23.1 VERIFY THAT XA5=@055125 AND
       XA6=@165511 WITH MASK6=@177713
                                          ERROR=18
4.24   SET XA2 TO 5.
4.24.1 PULSE CLK10 ONCE.
4.24.2 VERIFY THAT XA5=@065125 AND
       XA6=@166501 WITH MASK6=@177713.
                                          ERROR=19
4.25   SET XA2 TO @15.
4.25.1 VERIFY THAT XA5=@065125 AND
       XA6=@164101 WITH MASK6=@177713.
                                          ERROR=20
4.26   VERIFY THAT OUTPUTS NMADD8* TO NMADD1*
       ARE SET TO DT(NOTIMES)
                                          ERROR=21
4.27   OUTPUT @177777 ON THE NDAT1*-NDAT16*
4.27.1 VERIFY THAT NDAT1* AND NDAT3+ IS AT 0.
                                          ERROR=22
4.28   SET PARITY TO 1.
P.28   VERIFY THAT TNDAT1* IS SET TO 0 AND
       NDAT2*,NDAT3* IS AT 1.
                                          ERROR=221
4.29   SET PARITY TO 0.
4.29.2 SET XA2 TO @215.
4.29.2 PULSE CLCK10 ONCE.
4.29.3 VERIFY THAT XA5=@065125 AND
       XA6=@164111 WITH MASK6=@177713.
                                          ERROR=23
4.30   PULSE CLK20 3 TIMES.
4.30.1 VERIFY THAT XA5=@065175 AND
       XA6=@164111 WITH MASK6=@177713.
                                          ERROR=24
4.31   PULSE CLK10 ONCE.
4.31.1 VERIFY THAT XA5=@065175 AND
       XA6=@164111 WITH MASK6=@177713.
                                          ERROR=25
4.32   SET XA2 TO 5.
4.32.1 PULSE CLK20 3 TIMES.
4.32.2 VERIFY THAT XA5=@065125 AND
       XA6=@166511 WITH MASK6=@177713.
                                          ERROR=26
4.33   PULSE CLK10 ONCE.
4.33.1 VERIFY THAT XA5=@071125 AND
       XA6=@166501 WITH MASK6=@177713
                                          ERROR=27
4.34   OUTPUT @125252 ON NDAT16*-NDAT1*.
4.34.1 SET XA2 TO @15.
4.34.2 VERIFY THAT XA5=@071125 AND
       XA6=@164701 WITH MASK6=@177713.
                                          ERROR=28
4.35   PULSE CLK10 ONCE.
4.35.1 VERIFY THAT XA5=@071125 AND
       XA6=@164711 WITH MASK6=@177713.
                                          ERROR=29
4.36   VERIFY THAT THE CONTENTS OF THE
       NDAT1*-16* ARE DECREMENTED BY ONE.
                                          ERROR=30
4.37   SET XA2 TO @215.
4.37.1 PULSE CLK20 3 TIMES.
```

```
1379  4.37.2 VERIFY THAT XA5=@071175 AND
1380         XA6=@164711 WITH MASK6=@177713.
1381                                        ERROR=31
1382  4.38   PULSE CLK10 ONCE.
1383  4.38.1 VERIFY THAT XA5=@071175 AND
1384         XA6=@164711 WITH MASK6=@177713.
1385                                        ERROR=32
1386  4.39   SET XA2 TO 5.
1387  4.39.1 PULSE CLK20 3 TIMES.
1388  4.39.2 PULSE CLK10 ONCE.
1389  4.39.3 VERIFY THAT XA5=@077105 AND
1390         XA6=@166401 WITH MASK6=@177753.
1391                                        ERROR=33
1392  4.40   UNTIL THE END OF THE NARRATIVE THE VALUE
1393         OF MASK6 REMAINS UNCHANGED AT @177753.
1394  4.41   VERIFY THAT OUTPUTS  SDB16*-SDB1* ARE SET TO
1395         @125252.
1396                                        ERROR=3301
1397  4.42   SET XA2 TO @15.
1398  4.42.1 VERIFY THAT XA5=@077105 AND
1399         XA6=@164601.
1400                                        ERROR=34
1401  4.43   VERIFY THAT THE CONTENTS OF THE NMADD16*-NMADD1*
1402         BUS HAVE BEEEN DECREMENTED BY ONE.
1403                                        ERROR=35
1404  4.44   PULSE CLK10 ONCE.
1405  4.44.1 VERIFY THAT XA5=@077105 AND
1406         XA6=@164511.
1407                                        ERROR=36
1408  4.45   SET XA2 TO @215.
1409  4.45.1 PULSE CLK20 3 TIMES.
1410  4.45.2 VERIFY THAT XA5=@077145 AND
1411         XA6=@164511.
1412                                      ERROR=37
1413  4.46   PULSE CLK10 ONCE.
1414  4.46.1 VERIFY THAT XA5=@077141 AND
1415         XA6=@164511.
1416                                      ERROR=38
1417  4.47   SET XA2 TO 5.
1418  4.47.1 PULSE CLK20 3 TIMES.
1419  4.47.2 VERIFY THAT XA5=@077101 AND
1420         XA6=@166411.
1421                                    ERROR=39
1422  4.48   PULSE CLK10 ONCE.
1423  4.48.1 VERIFY THAT XA5=@077107 AND
1424         XA6=@166411.
1425                                    ERROR=40
1426  4.49   PULSE CLK10 33 TIMES.
1427  4.49.1 VERIFY THAT XA5=@077107 AND
1428         XA6=@166411.
1429                                     ERROR=41
1430  4.50   PULSE CLK10 ONCE.
1431  4.50.1 VERIFY THAT XA5=@077104 AND
1432         XA6=@122411.
1433                                    ERROR=42
1434  4.51   PULSE CLK10 ONCE.
1435  4.51.1 VERIFY THAT XA5=@035105 AND
1436         XA6=@166411.
1437                                    ERROR=43
1438  4.52   PULSE CLK10 ONCE.
```

```
4.52.1 WAIT FOR 1 MSEC.
4.52.2 VERIFY THAT XA5=@035101 AND
       XA6=@166411.
                                    ERROR=4301
4.53   PULSE CLK10 ONCE.
4.53.1 VERIFY THAT XA5=@035107 AND
       XA6=@166411.
                                    ERROR=44
4.54   PULSE CLK20 TWICE.
4.54.1 VERIFY THAT XA5=@035107 AND
       XA6=@166411.
                                    ERROR=45
4.55   PULSE CLK10 ONCE.
4.55.1 VERIFY THAT XA5=@035104 AND
       XA6=@166411.
                                    ERROR=46
4.56   PULSE CLK10 ONCE.
4.56.1 VERIFY THAT XA5=@035105 AND
       XA6=@166411.
                                    ERROR=461
4.57   SET XA2 TO 2.
4.57.1 PULSE CLK10 ONCE.
4.57.2 VERIFY THAT OUTPUTS SDB16*-SDB1* IS .
       SET TO @125252
                                    ERROR=47
4.58   SET XA2 TO 6.
4.58.1 VERIFY THAT XA5=@035105 AND
       XA6=@176411.
                                    ERROR=48
4.59   PULSE CLK10 ONCE.
4.59.1 SET XA2 TO 2.
4.59.2 OUTPUT @52525 ON THE NDAT16*-NDAT1*
       INPUTS.
4.59.3 VERIFY THAT XA5=@074125 AND
       XA6=@164701.
                                    ERROR=49
4.60     FOR WDCNT START[NOTIMES] TO 14
         EXECUTE THE FOLLOWING
4.60.1   IF WDCNT=START[NOTIMES] THEN SET
         SCB2 TO 0 ELSE SET SCB2 TO @40
4.60.2   SET XA2 TO @12.
4.60.2.1 VERIFY THAT XA5=@074125 AND
         XA6=@164701 OR SCB2.
                                    ERROR=50
4.60.3   VERIFY THAT THE CONTENTS AT THE
         NMADD8*-NMADD1* OUTPUTS HAS BEEN
         DECREMENTED BY ONE.
4.60.4   PULSE CLK10 ONCE.
4.60.4.1 VERIFY THAT XA5=@074125 AND
         XA6=@164711 OR SCB2.
                                    ERROR=51
4.60.5   SET XA2 TO @212.
4.60.5.1 PULSE CLK20 3 TIMES.
4.60.5.2 VERIFY THAT XA5=@074175 AND
         XA6=@164711 OR SCB2.
                                    ERROR=52
4.60.6   PULSE CLK10 ONCE.
4.60.6.1 VERIFY THAT XA5=@074175 AND
         XA6=@164711 OR SCB2.
                                    ERROR=53
```

```
4.60.7   SET XA2 TO 2.
4.60.7.1 PULSE CLK20 3 TIMES.
4.60.7.2 VERIFY THAT XA5=@074125 AND
         XA6=@166551.
                                    ERROR=54
4.60.8   VERIFY THAT OUTPUTS SDB16*-SDB1*
         SET TO @052525.
                                    ERROR=55
4.60.9   PULSE CLK10 ONCE.
4.60.9.1 VERIFY THAT XA5=@075505 AND
         XA6=@166541.
                                    ERROR=56
4.60.10   SET XA2 TO @12.
4.60.10.1 VERIFY THAT XA5=@075505 AND
          XA6=@164741.
                                     ERROR=57
4.60.11   PULSE CLK10 ONCE.
4.60.11.1 VERIFY THAT XA5=@075505 AND
          XA6=@164751.
                                     ERROR=58
4.60.12   VERIFY THAT THE CONTENTS AT THE OUTPUTS
          NMADD8*-NMADD1* HAVE BEEN DECREMENTED BY
          ONE.
                                     ERROR=5901
4.60.13   SET XA2 TO @212.
4.60.13.1 PULSE CLK20 3 TIMES.
4.60.13.2 VERIFY THAT XA5=@075545 AND
          XA6=@164751.
                                     ERROR=59
4.60.14   PULSE CLK10 ONCE.
4.60.14.1 VERIFY THAT XA5=@075541 AND
          XA6=@164751.
                                     ERROR=60
4.60.15   SET XA2 TO 2.
4.60.15.1 PULSE CLK20 3 TIMES.
4.60.15.2 VERIFY THAT XA5=@075501 AND
          XA6=@166551.
                                     ERROR=61
4.60.16   PULSE CLK10 ONCE.
4.60.16.1 VERIFY THAT XA5=@075507 AND
          XA6=@166551.
                                     ERROR=62
4.60.17   VERIFY THAT OUTPUTS SDB16*-SDB1*
          ARE SET TO @052525.
                                     ERROR=63
4.60.18   PULSE CLK10 ONCE.
4.60.18.1 VERIFY THAT XA5=@075507 AND
          XA6=@166551.
                                     ERROR=621
4.60.19   PULSE CLK10 ONCE.
4.60.19.1 VERIFY THAT XA5=@075504 AND
          XA6=@122551.
                                     ERROR=65
4.60.20   PULSE CLK10 ONCE.
4.60.20.1 VERIFY THAT XA5=@035105 AND
          AND XA6=@166411.
                                     ERROR=66
4.60.21   PULSE CLK10 ONCE.
4.60.21.1 WAIT FOR 1 MSEC.
4.60.21.2 VERIFY THAT XA5=@035101 AND
```

```
1559            XA6=@166411.
1560                                        ERROR=67
1561  4.60.22   PULSE CLK10 ONCE.
1562  4.60.22.1 VERIFY THAT XA5=@035107 AND
1563            XA6=@166411.
1564                                        ERROR=68
1565  4.60.23   PULSE CLK10 TWICE.
1566  4.60.23.1 VERIFY THAT XA5=@035107 AND
1567            XA6=@166411.
1568                                        ERROR=69
1569  4.60.24   PULSE CLK10 ONCE.
1570  4.60.24.1 VERIFY THAT XA5=@035104 AND
1571            XA6=@166411.
1572                                        EROOR=70
1573  4.60.25   PULSE CLK10 ONCE.
1574  4.60.25.1 VERIFY THAT XA5=@035105 AND
1575            AND XA6=@166411.
1576                                        ERROR=71
1577  4.60.26   SET XA2 TO 6.
1578  4.60.26.1 VERIFY THAT XA5=@035105 AND
1579            XA6=@176411.
1580                                        ERROR=72
1581  4.60.27   PULSE CLK10 ONCE.
1582  4.60.27.1 VERIFY THAT XA4=@074125 AND
1583            XA6=@166541.
1584                                        ERROR=73
1585  4.61    SET XA2 TO @12.
1586  4.61.1 VERIFY THAT XA5=@074125 AND
1587         XA6=@164741.
1588                                        ERROR=74
1589  4.62    VERIFY THAT THE CONTENTS OF THE
1590         OUTPUTS NMADD8*-NMADD1* HAVE BEEN
1591         DECREMENTED BY ONE.
1592  4.63    PULSE CLK10 ONCE.
1593  4.63.1 VERIFY THAT XA5=@074125 AND
1594         XA6=@164751.
1595                                        ERROR=76
1596  4.64    SET XA2 TO @212.
1597  4.64.1 PULSE CLK20 3 TIMES.
1598  4.64.2 VERIFY THAT XA5=@074175 AND
1599         XA6=@164751.
1600                                        ERROR=77
1601  4.65    PULSE CNK10 ONCE.
1602  4.65.1 VERIFY THAT XA5=@074175 AND
1603         XA6=@164751.
1604                                        ERROR=78
1605  4.66    SET XAT TO 2.
1606  4.66.1 PULSE CLK20 3 TIMES.
1607  4.66.2 VERIFY THAT XA5=@074125 AND
1608         XA6=@166551.
1609                                       ERROR=79
1610  4.67    VERIFY THAT OUTPUTS SDB16*-SDB1*
1611         ARE SET TO @052525.
1612                                       ERROR=80
1613  4.68    SET XA2 TO @12.E.
1614  4.68.1 VERIFY THAT XA5=@075505 AND
1615         XA6=@164751.
1616                                       ERROR=81
1617  4.69    SET XA2 TO @12.
1618  4.69.1 VERIFY THAT XA5=@075505 AND
```

```
1619          XA6=@164741.
1620                                        ERROR=82
1621    4.70  VERIFY THAT THE CONTENTS AT THE
1622          OUTPUTS NMADD8*-NMADD1* HAVE BEEN
1623          DECREMENTED BY ONE.
1624                                        ERROR=83
1625    4.71  PULSE CLK10 ONCE.
1626    4.71.1 VERIFY THAT XA5=@075505 AND
1627          XA6=@164751.
1628                                        ERROR=831
1629    4.72  SET XA2 TO @212.
1630    4.72.1 PULSE CLK20 3 TIMES.
1631    4.72.2 VERIFY THAT XA5=@075545 AND
1632          XA6=@164751.
1633                                        ERROR=84
1634    4.73  PULSE CLK10 ONCE.
1635    4.73.1 WAIT FOR 1 MSEC.
1636    4.73.2 VERIFY THAT XA5=@075541 AND
1637          XA6=@164751.
1638                                        ERROR=85
1639    4.74  SET XA2 TO 2.
1640    4.74.1 PULSE CLK20 3 TIMES.
1641    4.74.2 VERIFY THAT XA5=@075501 AND
1642          XA6=@146551.
1643                                        ERROR=86
1644    4.75  VERIFY THAT OUTPUTS SDB16*-SDB1*
1645          ARE SET TO @52525.
1646                                        ERROR=861
1647    4.76  PULSE CLK10 ONCE.
1648    4.76.1 VERIFY THAT XA5=@075507 AND
1649          XA6=@146551.
1650                                        ERROR=87
1651    4.77  PULSE CLK10 33 TIMES.
1652    4.77.1 VERIFY THAT XA5=@075507 AND
1653          XA6=@146551.
1654                                        ERROR=88
1655    4.78  PULSE CLK20 ONCE.
1656    4.78.1 VERIFY THAT XA5=@075504 AND
1657          XA6=@116551.
1658                                        ERROR=89
1659    4.79  PULSE CLK10 ONCE.
1660    4.79.1 VERIFY THAT XA5=@075305 AND
1661          XA6=@146540.
1662                                        ERROR=90
1663    4.80  PULSE CLK10 ONCE.
1664    4.80.1 VERIFY THAT XA5=@075305 AND
1665          XA6=@146540.
1666                                        ERROR=91
1667    4.81  SET XA2 TO @32.
1668    4.81.1 VERIFY THAT XA5=@075305 AND
1669          XA6=@144140.
1670                                        ERROR=92
1671    4.82  VERIFY THAT NDAT1*,NDAT3* IS SET TO 0
1672          AND THAT NDAT2* IS SET TO 1.
1673                                        ERROR=93
1674    4.83  VERIFY THAT OUTPUTS
1675          NMADD8*-NMADD1* ARE SET TO
1676          DTI[NOTIMES].
1677                                        ERROR=94
1678    4.84  PULSE CLK10 ONCE.
```

```
4.83.1 VERIFY THAT XA5=@075305 AND
       XA6=@144150.
                                    ERROR=95
4.84   SET XA2 TO @212.
4.84.1 PULSE CLK20 3 TIMES.
4.84.2 VERIFY THAT XA5=@075345 AND
       XA6=@140150.
                                    ERROR=96
4.85   PULSE CLK10 ONCE.
4.85.1 VERIFY THAT XA5=@025145 AND
       XA6=@146411.
                                    ERROR=97
4.86   SET XA2 TO 2.
4.86.1 PULSE CLK20 3 TIMES.
4.86.2 PULSE CLK10 ONCE.
4.86.3 WAIT FOR 1 MSEC.
4.86.4 VERIFY THAT XA5=@035101 AND
       XA6=@146411.
                                    ERROR=98
4.87   PULSE CLK10 ONCE.
4.87.1 VERIFY THAT XA5=@035107 AND
       XA6=@146411.
                                    ERROR=99
4.88   PULSE CLK10 TWICE.
4.88.1 VERIFY THAT XA5=@035107 AND
       XA6=@146411.
                                    ERROR=100
4.89   PULSE CLK10 ONCE.
4.89.1 VERIFY THAT XA5=@035104 AND
       XA6=@146411.
                                    ERROR=101
4.90   PULSE CLK10 ONCE.
4.90.1 VERIFY THAT XA5=@035105 AND
       XA6=@146411.
                                    ERROR=102
5.     IDLE ROUTINE
5.1   SET XA2 TO @100.
5.2   SET XA2 TO 0.
5.3   WAIT FOR 1 MSEC.
5.4   CHECK PBUSY.
5.4.1 IF PBUSY IS SET TO THE SKIP THE REST
      OF THE IDLE ROUTINE AND CONTINUE AT 6.1
                                     ERROR=103
5.4.2 ENABLE A FREE RUNNING 10 MHZ CLOCK AT
      THE SR 10MHZ A/B* AND SR 10 MHZ A/B
      INPUTS.
5.4.3 VERIFY THAT THE OUTPUT 20 MSEL IS
      PULSING AND THAT THE PULSE WIDTH IS
      20 MICROSECONDS.
                                    ERROR=1031
5.4.4 VERIFY THAT THE OUTPUT SHTENB IS
      PULSING AND THAT THE THE PULSE WIDTH
      IS 20 MICROSECONDS.
                                    ERROR=1032
5.4.5   CONSECUTIVELY FOR 3 TIMES
5.4.5.1 SET XA2 TO 1,2,3
5.4.5.2 VERIFY THAT OUTPUT 20 MSEL IS
        PULSING AND THAT THE PULSE WIDTH
        ID 20 MICROSECONDS.
                                    ERROR=1033
```

```
6.    U12,U25 TEST
6.1   SET XA2 TO @100.
6.2   SET XA2 TO 0.
6.3   SET XA2 TO 4.
6.4   PULSE CLK10 3 TIMES.
6.5   CHECK BIT SHTENB.
6.5.1 UUT IS FAULTY IF SHTENB IS SET TO 1
                                        ERROR=104
6.5.2 SET XA2 TO 0.
6.5.3 SET XA2 TO 2.
6.5.4 PULSE CLK10 3 TIMES.
6.6   CHECK BIT SHTENB.
6.6.1 UUT IS FAULTY IF SHTENB IS SET TO 1
                                        ERROR=105
6.7   CHECK BIT PORI.
6.7.1 UUT IS FAULTY IF PORI IS SET TO 0
                                        ERROR=106
6.7.2 CHECK BIT TOUT*
6.7.3 UUT IS FAULTY IF BIT TOUT* IS SET TO 0
                                        ERROR=1061
6.8   FOR 254 TIMES EXECUTE THE FOLLOWING
6.8.1 PULSE CLK10 ONCE.
6.8.2 CHECK BIT PORI
6.8.3 CHECK BIT TOUT*
6.8.4 UUT IS FAULTY IF PORI IS SET TO 1
      OR BIT TOUT* IS SET TO 0 AND CLOCK
      IS PULSED LESS THAN 254 TIMES
                                          ERROR=107
6.8.5 UUT IS FAULTY IF PORI IS SET TO 0
      OR BIT TOUT* IS SET TO 0 AND THE
      CLOCK10 IS PULSED EXACTLY 254 PULSES
                                          ERROR=108
6.9   PULSE CLK10 1  TIMES.
6.9.1 CHECK BIT PORI
6.9.2 TERMINATE TEST IF PORI IS SET TO 0.
                                          ERROR=109
;
```

LITTON SYSTEMS CANADA LIMITED
Litton 25 Cityview Drive, Rexdale, Ontario, Canada M9W 5A7

NO. CD 540-93

DOCUMENT TYPE	
ACCEPTANCE TEST PROCEDURE	☒ SHIPBORNE ☐ AGE ☐ FTE

UNIT UNDER TEST	MAJOR TEST EQUIPMENT	
NAME CIRCUIT CARD ASSEMBLY, SERIAL OUTPUT CONTROLLER	NAME DIGIPACT	NO. T11700

PROGRAM		PRODUCT PART NUMBER	
CEI	SYSTEM	BASIC	DASH

PROGRAM		PRODUCT PART NUMBER	
CEI	SYSTEM	BASIC	DASH

REVISIONS

REV	ECO	CHANGE DATE	CHANGE DESCRIPTION	APPROVAL

LSL RELEASE: DRN 47255 DATE 2-9-82

APPROVALS

PREPARED		PROJECT ENGINEER
S. Karapetsas S. KARAPETSAS		M. GEORGI M. Georgi
		RELEASE APPROVAL S. MASLOW

LITTON SYSTEMS (CANADA) LIMITED
25 Cityview Drive, Rexdale, Ontario, Canada

LE

CIRCUIT CARD ASSEMBLY,
SERIAL OUTPUT CONTROLLER

PAGE	
2	
NUMBER CD 540-93	REV.

1. SCOPE

1.1 This document presents instructions which enable automatic testing of the Circuit Card Assembly, Serial Output Controller in the configuration identified on the document cover sheet.

1.2 This document exhibits test record data sheet(s) which establish proof of successful completion of functional tests.

1.3 This document defines only those test requirements which are unique to the Unit Under Test (UUT). Digipact Operating Procedure CD 540 shall be used for the operation of the automatic test system.

2. APPLICABLE DOCUMENTS

Litton

855075	Assembly Drawing
CD-540	Operating Procedure, Digipact
38028	Tape Data
38027	Program

3. REQUIREMENTS

3.1 Test Equipment Requirements.

3.1.1 Automatic Test System, Litton Part No. T11700 (Digipact).

3.1.2 Interface, Serial Controller, Litton Part No. T16868.

3.1.3 Test Tape, Litton Part No. 38028 (Functional Test).

3.2 Power Requirements.

3.2.1 Three Phase, 60Hz power as specified below:

FREQUENCY	60 ± 3Hz
VOLTAGE	115 ± 7V RMS
CURRENT	20 AMPERE PHASE
PHASE	ABC, WYE CONNECTED
CREST FACTOR	1.41 ± 0.1
HARMONIC CONTENT	5 PERCENT MAX. OF FUNDAMENTAL

LITTON SYSTEMS CANADA LIMITED
25 Cityview Drive, Rexdale, Ontario, Canada M9W 5A7

TLE

CIRCUIT CARD ASSEMBLY,
SERIAL OUTPUT CONTROLLER

3.2.2 Single Phase, 400Hz power as specified below:

FREQUENCY	400 ± 20Hz
VOLTAGE	115 ± 7V RMS
CURRENT	1 AMPERE
PHASE	SINGLE
CREST FACTOR	1.41 ± 0.1
HARMONIC CONTENT	5 PERCENT MAX. OF FUNDAMENTAL

3.3 Acceptance Requirements.

3.3.1 The assembly tested shall meet all requirements specified in this document.

3.3.2 Where plus or minus (±) tolerances, as specified in Functional Test Program No. 38027, control test result acceptability any test result within the specified tolerance shall be automatically accepted by the test system.

3.4 Environmental Requirements.

Conduct all tests under room ambient pressure, temperature and humidity.

3.5 Pre-Test Requirements.

3.5.1 Insert the Serial/Controller interface No. 73 into the test system I/O connector interface. Care shall be exercised to prevent damage to the connectors.

3.5.2 Inspect UUT for any obvious physical damage. If applicable, corrective action shall be taken.

3.6 Functional Test.

3.6.1 Insert UUT into the Serial/Controller interface. Care shall be exercised to prevent damage to the connectors.

3.6.2 Refer to operating procedure CD 540, for initialization of the automatic testing.

3.6.3 Switch 14 up selects the line printer as the output device.

LITTON SYSTEMS CANADA LIMITED
25 Cityview Drive, Rexdale, Ontario, Canada M9W 5A7

TLE	PAGE
CIRCUIT CARD ASSEMBLY, SERIAL OUTPUT CONTROLLER	4
	NUMBER: CD 540-93 / REV.

3.6.4 Remove UUT from the interface after completion of test.

3.7 Post Test Requirements.

3.7.1 Remove the Serial/Controller interface from the test system I/O connector interface.

4. QUALITY ASSURANCE PROVISIONS

4.1 Cognizant departments at the test facility shall:

4.1.1 Ensure that the physical condition and the accompanying records establish the UUT ready for testing.

4.1.2 Ensure that the test equipment is operationally useable and that the applicable equipment is in valid calibration status.

5. PREPARATION FOR DELIVERY

5.1 Not applicable.

6. NOTES

6.1 None.

LITTON SYSTEMS CANADA LIMITED
25 Cityview Drive, Rexdale, Ontario, Canada M9W 5A7

TITLE	PAGE 5
CIRCUIT CARD ASSEMBLY, SERIAL OUTPUT CONTROLLER	NUMBER CD 540-93 / REV.

TEST DATA SHEET D1 OF D1

DATE _____ TEST TECHNICIAN _____

INSPECTOR _____

REF. PARA.

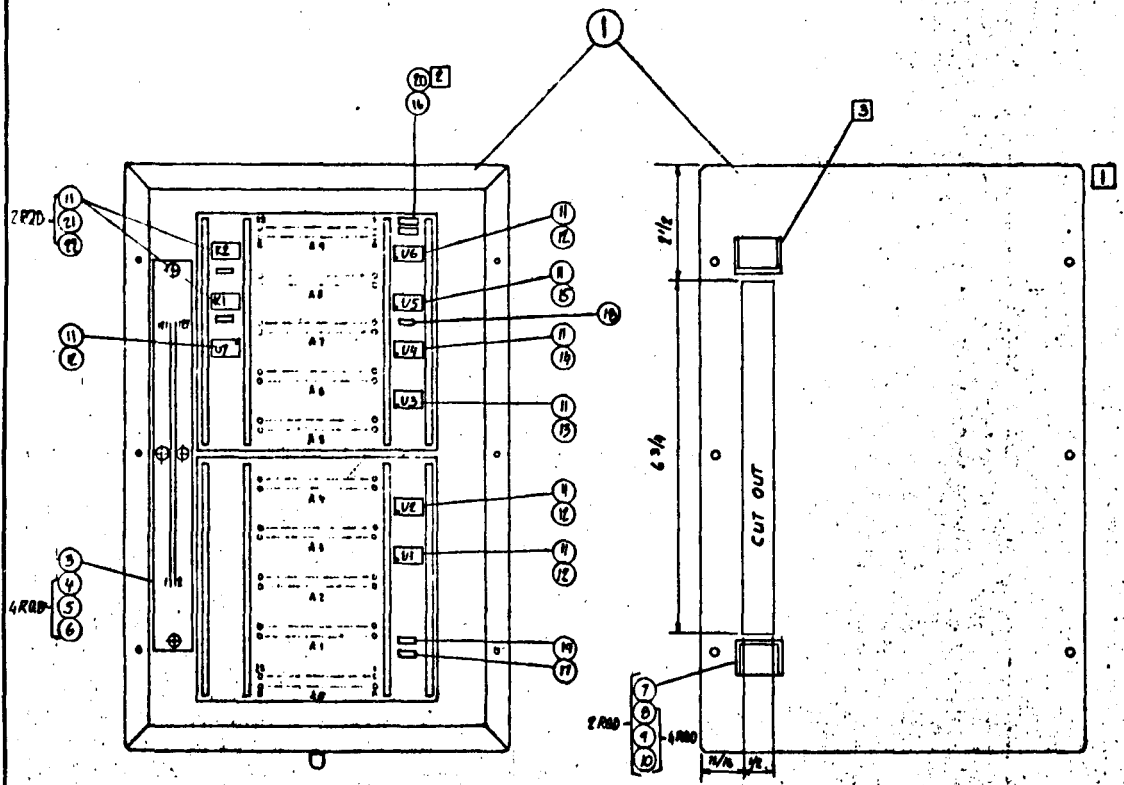
3.6 Functional Test

Functional test successfully completed _____ Yes

Attach teletype printout below

A WHOLLY-OWNED SUBSIDIARY. ANY DISCLOSURE OR USE OF THIS INFORMATION OR ANY REPRODUCTION OF THIS DOCUMENT FOR OTHER THAN THE SPECIFIC PURPOSE FOR WHICH IT IS INTENDED IS EXPRESSLY PROHIBITED EXCEPT AS LITTON SYSTEMS (CANADA) LIMITED MAY OTHERWISE AGREE IN WRITING.

LTR	DESCRIPTION	DATE	APPROVED



ITEM	QTY.	PART NO.	NOMENCLATURE OR DESCRIPTION
33			
32			
31			
30			
29			
28			
27			
26			
25			
24			
23			
22	2		DIODE. IN914
21	2		RELAY, PRME 1A05 K1, K2
20	A/R		CAPACITOR, CERAMIC, .1µF. 50V
19	1		CAPACITOR, TANTALUM, 10µF, 50V
18	1		CAPACITOR, CERAMIC, 20 PF, 50V
17	1		RESISTOR 2.2K, 1/8W
16	1		TRIM POT 5K. 1/8W
15	1		IC, DIP, MC4024 U5
14	1		IC, DIP, SN7432 U4
13	1		IC, DIP, SN7408 U3
12	4		IC, DIP. SN7404 U1, U2, U6, U7
11	9		SOCKET, IC, 14 PIN
10	8		WASHER LOCK #4
9	8		NUT HEX 4-40
8	8		SCREW FLAT HEAD 4-40 x .30
7	2	T15210-1	SUPPORT BRACKET
6	4		SCREW ROUND HEAD 6-32 x .38
5	4		SCREW FLAT HEAD 6-32 x .38
4	4		STANDOFF HEX 6-32 x 2
3	1	T15212-1	CONNECTOR BOARD (182 PINS)
2	REF	T16 867	SCHEMATIC
1	1	T15237	BASIC INTERFACE

LIST OF MATERIAL

[5] DRILL AND FIT DURING ASSEMBLY USING UUT

[2] DISTRIBUTE ON POWER LINES

[1] FINISH: PAINT LITTON GRAY.

NOTES: UNLESS OTHERWISE SPECIFIED.

UNLESS OTHERWISE SPECIFIED	DR S. KARAPETSAS	DATES 11.8.78
DIMENSIONS ARE IN INCHES TOLERANCES ON .X ± .XX ± .XXX ± ANGLES ± ✓	CHK	
MAT'L	DESIGN	
SPEC	PROJ S. Karapetsas	11.8.78
FINISH	APP M. Gruzi	9.10.79
	OTHER	

LITTON SYSTEMS (CANADA) LIMITED
Litton Toronto, Canada

TITLE ASSEMBLY - INTERFACE, SERIAL OUTPUT CONTROLLER

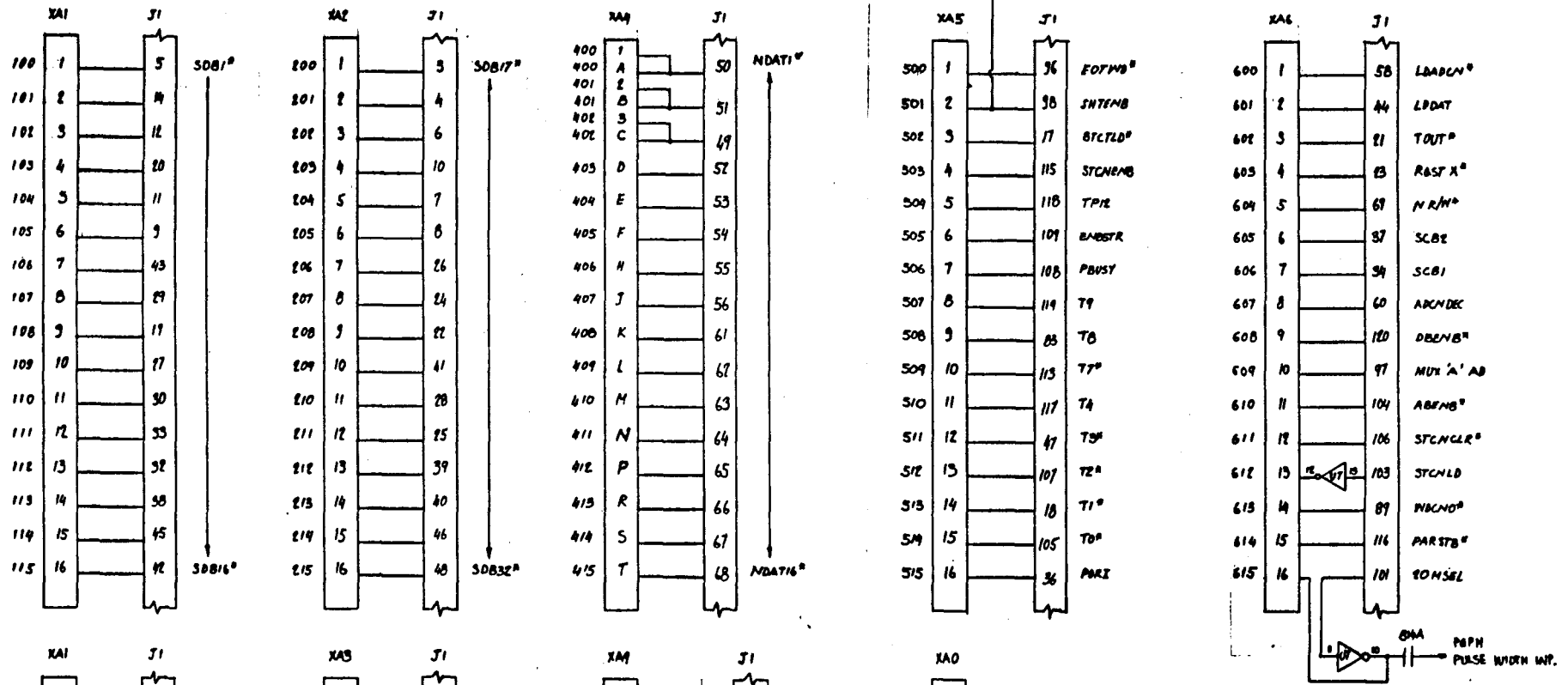
ENGINEERING SKETCH SHEET DO NOT SCALE

NUMBER T16868 REV A

SCALE SHEET 1 OF 1

...INFORMATION OR ANY REPRODUCTION OF THIS DOCUMENT FOR OTHER THAN THE SPECIFIC PURPOSE FOR WHICH IT IS INTENDED IS EXPRESSLY PROHIBITED EXCEPT AS LITTON SYSTEMS CANADA LIMITED MAY OTHERWISE AGREE IN WRITING.

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED



NOTES:
1. ASSEMBLY DRAWING IS T16868.

ITEM	QTY.	PART NO.	NOMENCLATURE OR DESCRIPTION

LIST OF MATERIAL

ECL RELEASE DR... Ref only DATE 26 2.79

UNLESS OTHERWISE SPECIFIED	DR S.KARAPETSAS	DATE 11.8.78
DIMENSIONS ARE IN INCHES TOLERANCES ON X ± .XX ± .XXX ± ANGLES ± ✓	CHK	
MAT'L	DESIGN	
SPEC	ENGR S.Karapetsas	11.8.78
FINISH	PROJ M. Pongi	9.10.79
	OTHER	

LITTON SYSTEMS CANADA LIMITED
Litton Toronto, Canada

TITLE
SCHEMATIC
INTERFACE, SERIAL OUTPUT CONTROLLER

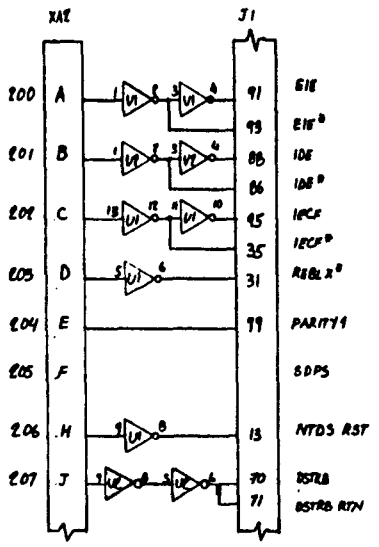
ENGINEERING SKETCH SHEET DO NOT SCALE

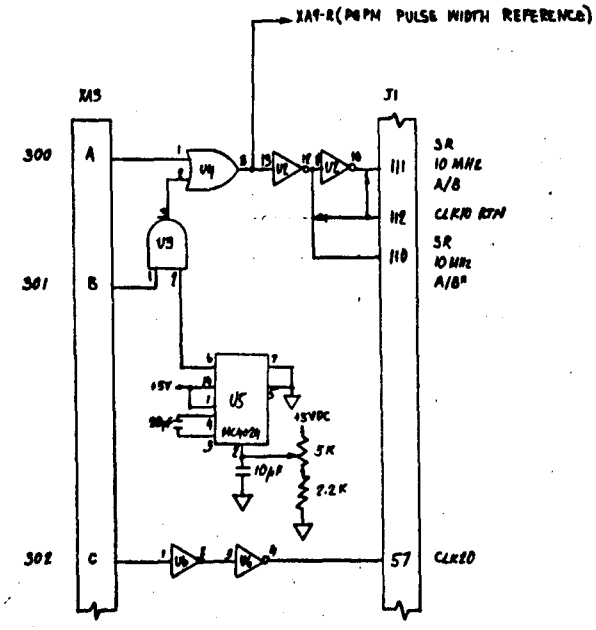
NUMBER T16867 REV B

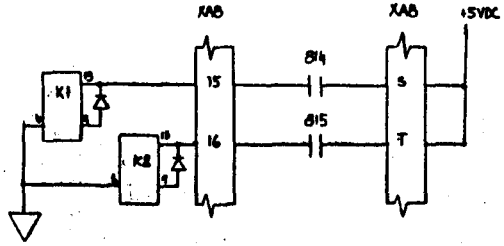
SCALE SHEET 1 OF 2

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO LITTON INDUSTRIES, INC. OF WHICH LITTON SYSTEMS CANADA LIMITED IS A WHOLLY-OWNED SUBSIDIARY. ANY DISCLOSURE OR USE OF THIS INFORMATION OR ANY REPRODUCTION OF THIS DOCUMENT FOR OTHER THAN THE SPECIFIC PURPOSE FOR WHICH IT IS INTENDED IS EXPRESSLY PROHIBITED EXCEPT AS LITTON SYSTEMS CANADA LIMITED MAY OTHERWISE AGREE IN WRITING.

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED







ITEM	QTY.	PART NO.	NOMENCLATURE OR DESCRIPTION

LIST OF MATERIAL

RELEASED FOR Ref only DATE 26.2.80

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES ON
.X ±
.XX ±
.XXX ±
ANGLES ±

	BY	DATE
CHK		
DESIGN		
ENGR		
PROJ		
OTHER		

LITTON SYSTEMS CANADA LIMITED
Litton Toronto, Canada

TITLE
SCHEMATIC
INTERFACE, SERIAL OUTPUT CONTROLLER

ENGINEERING SKETCH SHEET DO NOT SCALE

NUMBER T16867 REV B

SCALE SHEET 2 OF 2

APPENDIX III

ATE TERMINOLOGY

ATE TERMINOLOGY

ADAPTER, UUT:	Converts the output connector interface of a test system to mate with the connector(s) of a unit under test. An adapter may include circuitry to further enhance the test system capability.
ALGORITHMICALLY GENERATED PATTERN:	An array of digital data automatically generated by a predetermined software routine or program.
ANALOG SECTION:	That portion of ATE that includes stimuli and measurement devices required to perform tests on analog units.
AUTOMATIC TEST SYSTEM:	Provices stimuli to and makes measurements on a unit under test with minimal or no human intervention.
BLOCK DIAGRAM:	A simplified functional diagram of a system or assembly.
BIDIRECTIONAL BUS:	Transmits and receives digital data on a single line.
BUFFER:	An isolating circuit used to avoid distortion of the input signal by the driven circuit. Generally used in digital data transmission when driving through long cables.
BURN-IN:	The operation of items prior to their end application with the objective of stabilizing their characteristics and identifying early failures.

CARD TESTER:	Tests and diagnoses printed circuit boards with or without analog and digital components.
CONSTANT CURRENT POWER SUPPLY:	Maintains a fixed current through a variable load resistance.
CONSTANT VOLTAGE POWER SUPPLY:	Maintains a fixed voltage through a variable load resistance.
CONTINUITY TESTER:	Determines the presence and/or location of broken or open connections within a circuit.
COUNTER/TIMER:	Performs timing measurements on AC waveforms in addition to determining number of events.
COMPILER:	Convers a source programming language (e.g. ATLAS) to machine language (object code).
COMPUTER:	The controlling device in ATE that performs all data processing and evaluation
CONTROL SECTION:	The portion of ATE that includes the control computer and memory in addition to peripherals such as printers, magtapes, discs, card/tape readers and video terminals.
CONTROLLER:	A hardware interface that accepts instructions from a computer and reformats them to program an instrument or peripheral.

CONTROL PANEL: That part of ATE that includes push-buttons for the operator to supply instructions to the computer for test system control.

DATA LOGGING: Recording of test data generated by a UUT for future reference or calculation.

DATA RATE: The speed at which digital information is transmitted, expressed in Hertz/second or bits/second.

DATA REDUCTION: The conversion of raw data into a more useful form such as graphic representation.

DEBUG: Detect and remove errors from a test program by comparing it against a known good unit.

DEDICATED SWITCHING: Allows each device in a test system to connect to one point on the output interface.

DIAGNOSTICS: Methods used for detecting and isolating faults in a unit under test.

DIGITAL CLOCK: A series of synchronized pulses that determine the bit times (data rate) of a digital pattern.

DIGITAL DRIVER: The output stage of a digital data generator.

DIGITAL SECTION: That portion of ATE that includes all of the digital circuitry required for testing digital units

DIRECT MEMORY ACCESS (DMA):	The transfer of data directly from computer memory to a peripheral.
DISC:	A memory device that stores data on a magnetic disc.
EDITOR:	An interactive software subsystem that allows users to modify test programs directly on an automatic test system.
EXECUTIVE SOFTWARE:	The portion of ATE software that has supervisory control over all other software routines.
FAULT ISOLATION:	The process of identifying and locating failures in a unit under test.
FAULT SIGNATURE.	Data that represents the outputs of a known good unit and used to compare against outputs of a unit under test.
FAULT SIMULATION:	The process of injecting faults during a simulation run to determine the test comprehensiveness of an input pattern.
FIXED LOGIC LEVELS:	Digital data with high and low levels that are not programmable or adjustable.
FLEXIBLE SWITCHING:	Allows each device in a test system to connect to any pin on the output interface.
FOREGROUND/BACKGROUND:	A test system capable of performing some primary function (foreground) simultaneously with a secondary function (background), where the foreground function has priority. (e.g. testing a UUT while programming).

FUNCTIONAL ATE: Performs dynamic tests that tend to simulate the actual performance of the unit under test in its system environment.

GENERATOR (GENERAL): Supplies signals to a unit-under-test.

GENERATOR, ARBITRARY WAVEFORM: Supplies any shape waveform within its frequency and amplitude constraints.

GENERATOR, AC SIGNAL: Produces sine waves whose frequency and amplitude can usually be varied.

GENERATOR, DIGITAL DATA: Supplies arrays of binary bits for exercising the logic circuitry of a unit-under-test.

GENERATOR, DC SIGNAL: Sets and maintains DC levels primarily used in testing items such as A/D converters and operational amplifiers.

GO/NO-GO TEST: Determines whether a unit under test is functioning in accordance withspecifications but does not perform any diagnostic tests to determine the cause of an incorrect output.

GUIDED PROBE: A hand held probing device (single point or multipin clip) guided by an operator with instructions from a computer controlled algorithm.

HIGH LEVEL LANGUAGE: Uses English-like statements for programming instructions.

HIGH SPEED DIGITAL: Generating and receiving digital data at a specific rate, independent of the computer recycle time.

IN-CIRCUIT ATE: Tests individual components within a circuit while "guarding" out the effects of surrounding components.

INITIAL VALUE: The value of the time response of a system or element just prior to the time a stimulus is applied.

INITIALIZE: To establish an initial condition or starting state; for example, to set logic elements in a digital circuit or the contents of a storage location to a known state so that subsequent application of digital test patterns will drive the logic elements to another known state.

INTERFACE CONTROLLER: Enables a computer to transmit and receive digital information for programming an instrument or peripheral.

KNOWN GOOD BOARD: A printed circuit board assembly that satisfies all the conditions of its test specification; used in ATE for debugging test programs.

LABEL PRINTER: Prints data on small labels (about 2" x 3"); used for repair information in an ATE facility.

LIBRARY: Digital components modeled for use with a specific simulator.

LIMITED SWITCHING: Allows each devide in a test system to connect to more than one pin (but not all) on the output interface.

LINE PRINTER: Prints a complete line of information in one mechanical operation.

LOCAL PIN STORAGE: Digital data stored in a memory element such as a shift register or RAM for each pin in a digital devide; used for clocking out data at a predetermined rate.

LOGIC FAMILY: The values of the "1" and "0" levels for a specific type of logic such as TTL or CMOS.

LOGIC DIAGRAM: A diagram representing the logic elements and their interconnections.

MASK: Used to ignore specific bits in a digital pattern.

MASTER/SLAVE: A configuration where one system, the master, always has control over another system, the slave.

MEMORY: The portion of a computer that stores information for future use.

MODEL: Characterization of a digital component's functions for use in a simulator. A listing of models makes up the library for a simulator

MULTIPLEXER: Routes analog and digital devices to the same field of pins on the output interface of a test system.

MUTLI-STATION: A master system controlling more than one test system.

MULTI-TASKING: Performing more than one task simultaneously (e.g. testing and programming).

MULTIMETER: Measures AC volts, DC volts, resistance and current.

NODE: A terminal of any branch of a network or common to two or more branches of a network.

OBJECT CODE: A series of "1"sand "0"s that can be used by a computer directly.

OFF-LINE: Any operation performed away from the ATE.

ON-LINE: Any operation performed exclusively on the ATE.

OUTPUT INTERFACE: The output connection device in ATE that has all analog and digital signals transmitted through it for testing UUTs.

PATTERN: A defined set of digital data with more than one bit in depth and more than one line in width.

PERFORMANCE BOARD: Same as adpater.

PRESET: To establish an initial condition or starting state.

PRINTED CIRCUIT BOARD (PCB): For mounting of electrical components on which most connections are made by conductive circuit paths printed on the board.

PROPAGATION: The travel of electric waves through or along a medium.

PROGRAMMABLE LOGIC LEVELS: Digital data with high and low levels capable of being varied under computer control.

PROGRAMMING LANGUAGE: The system language used to generate test sequences for a UUT.

PROGRAMMING STATION: A system configured to perform programming tasks only.

PSEUDO RANDOM PATTERNS: A repeatable sequence of digital patterns that has the appearance of being random.

REAL-TIME DIGITAL: Digital tests conducted at the same speed and time that events actually occur on a UUT.

SIMULATOR: A software program capable of imitating and evaluating the functions of a digital unit without the physical unit available.

STIMULUS:	A signal input to a UUT.
SUBROUTINE:	A software program that performs a specific function, such as sequencing through a series of AC volt measurements. A subroutine usually resides in mass storage and can be called by the computer whenever that specific function must be performed.
TEST COMPREHENSIVENESS:	A numerical evaluation (expressed in %) of the effectiveness of a pattern used for testing a digital unit.
TEST PROGRAM:	A software routine that describes the power, stimulus, measurement and routing required to test a UUT.
TEST SPECIFICATION:	The governing document that defines the input and output parameters of a UUT.

APPENDIX IV

ILLUSTRATIONS

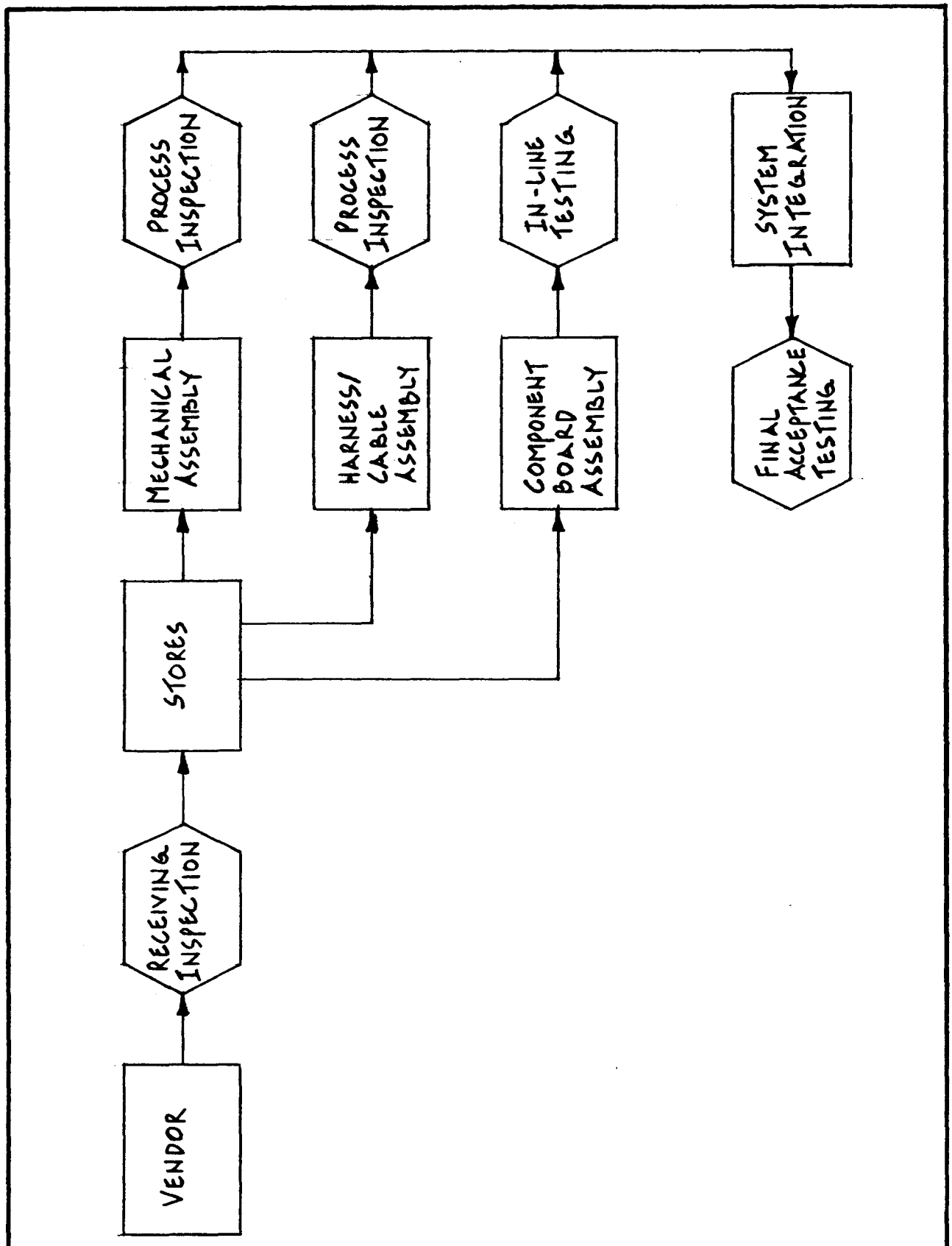


FIG. 1 THE PRODUCTION TESTING PROCESS.

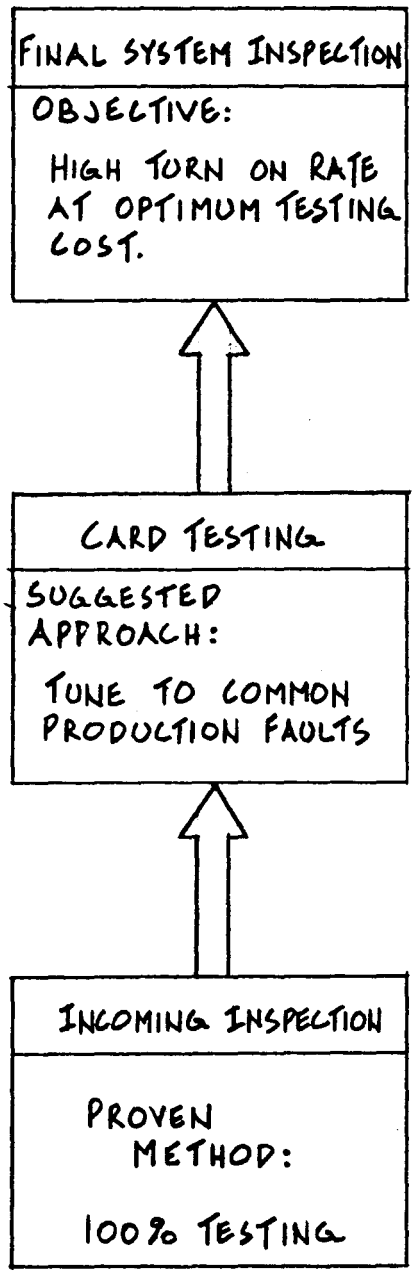


FIG. 2 TEST PLAN FOR A SUCCESSFUL PRODUCTION LINE.

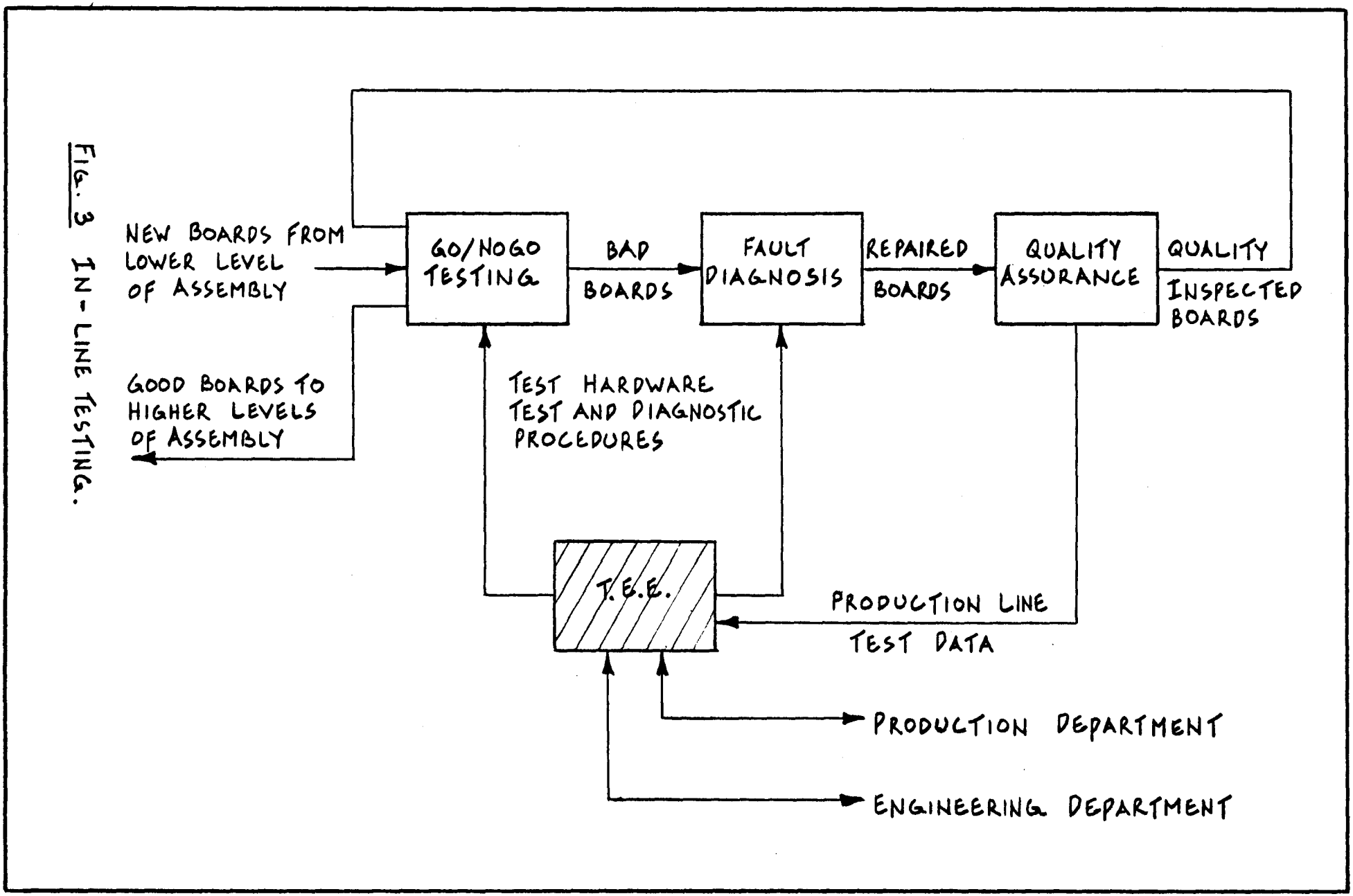


FIG. 3 IN-LINE TESTING.

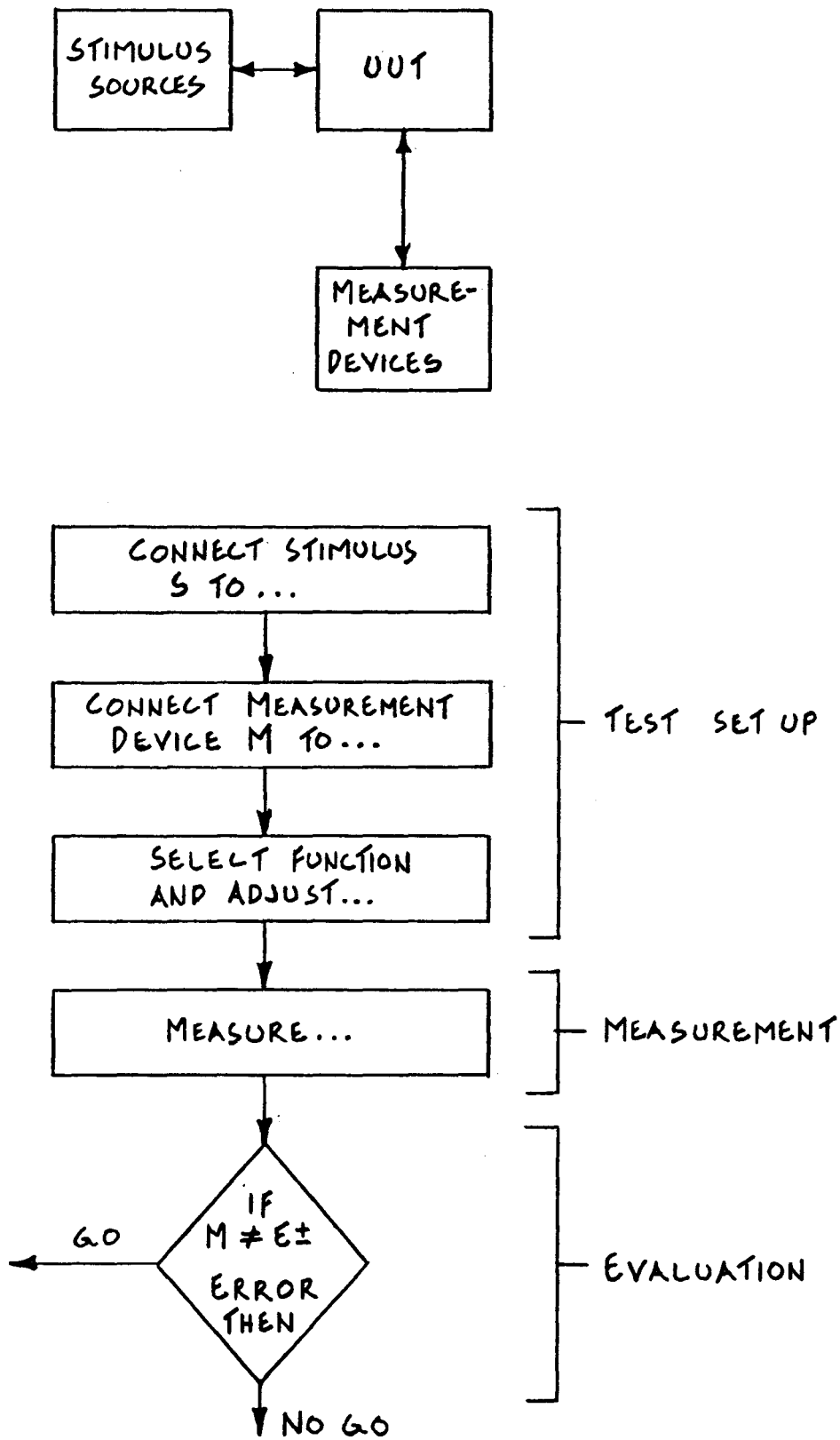


FIG. 4 A GENERALISED TEST STATION AND TEST PROCEDURE.

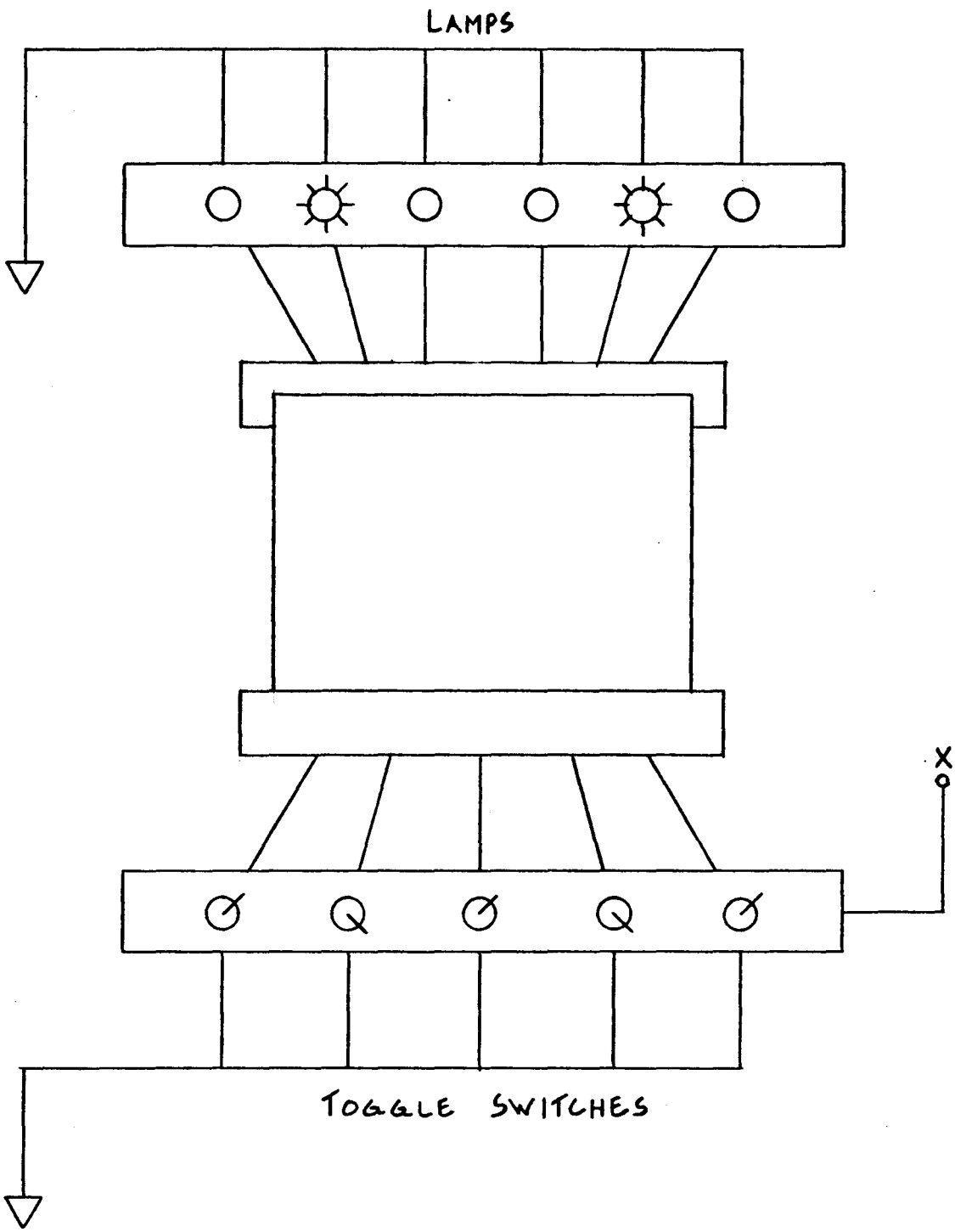
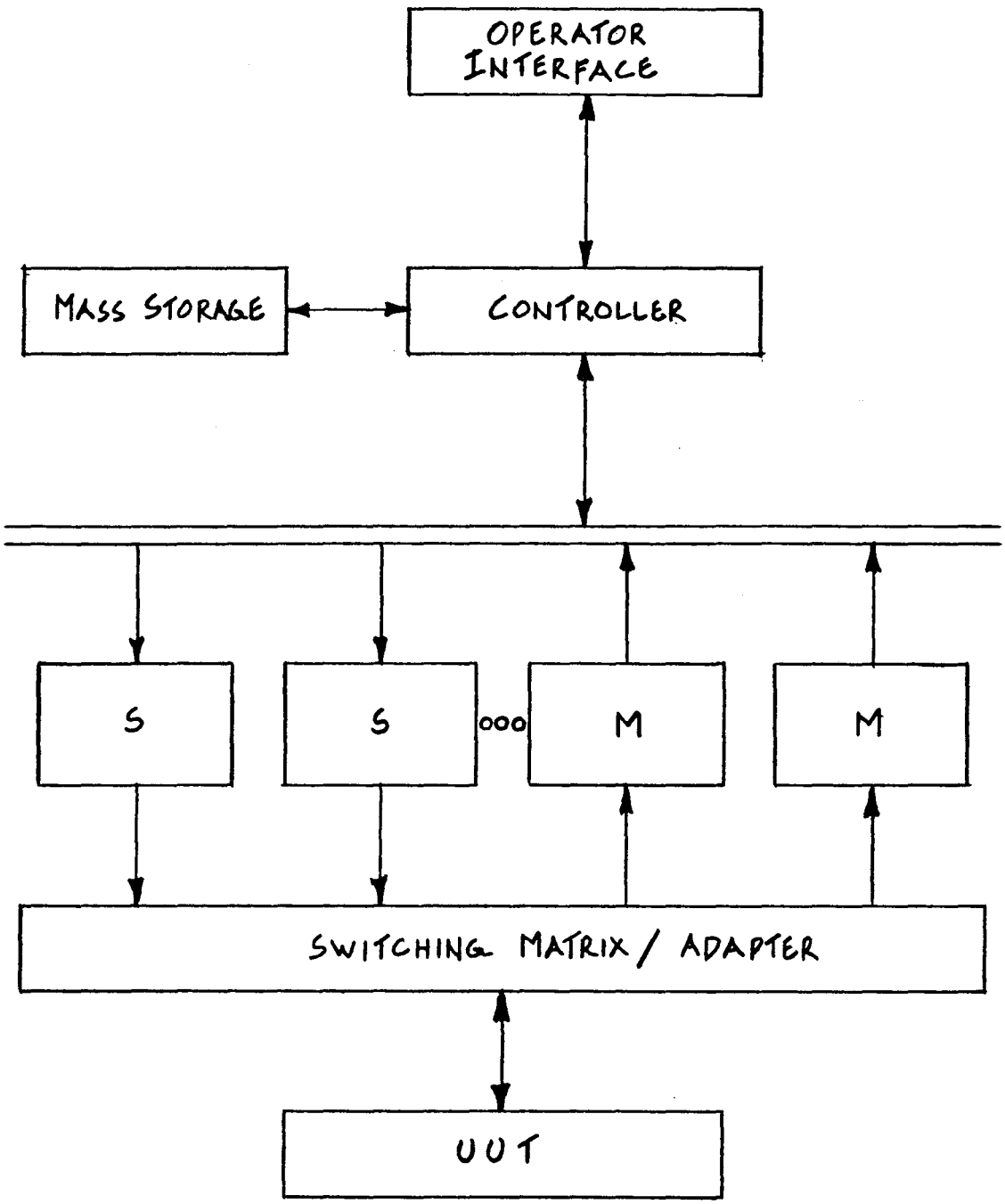


FIG. 5 A MANUAL DIGITAL TESTER.



S - STIMULUS DEVICE

M - MEASUREMENT DEVICE

FIG. 6 AN AUTOMATIC TEST STATION.

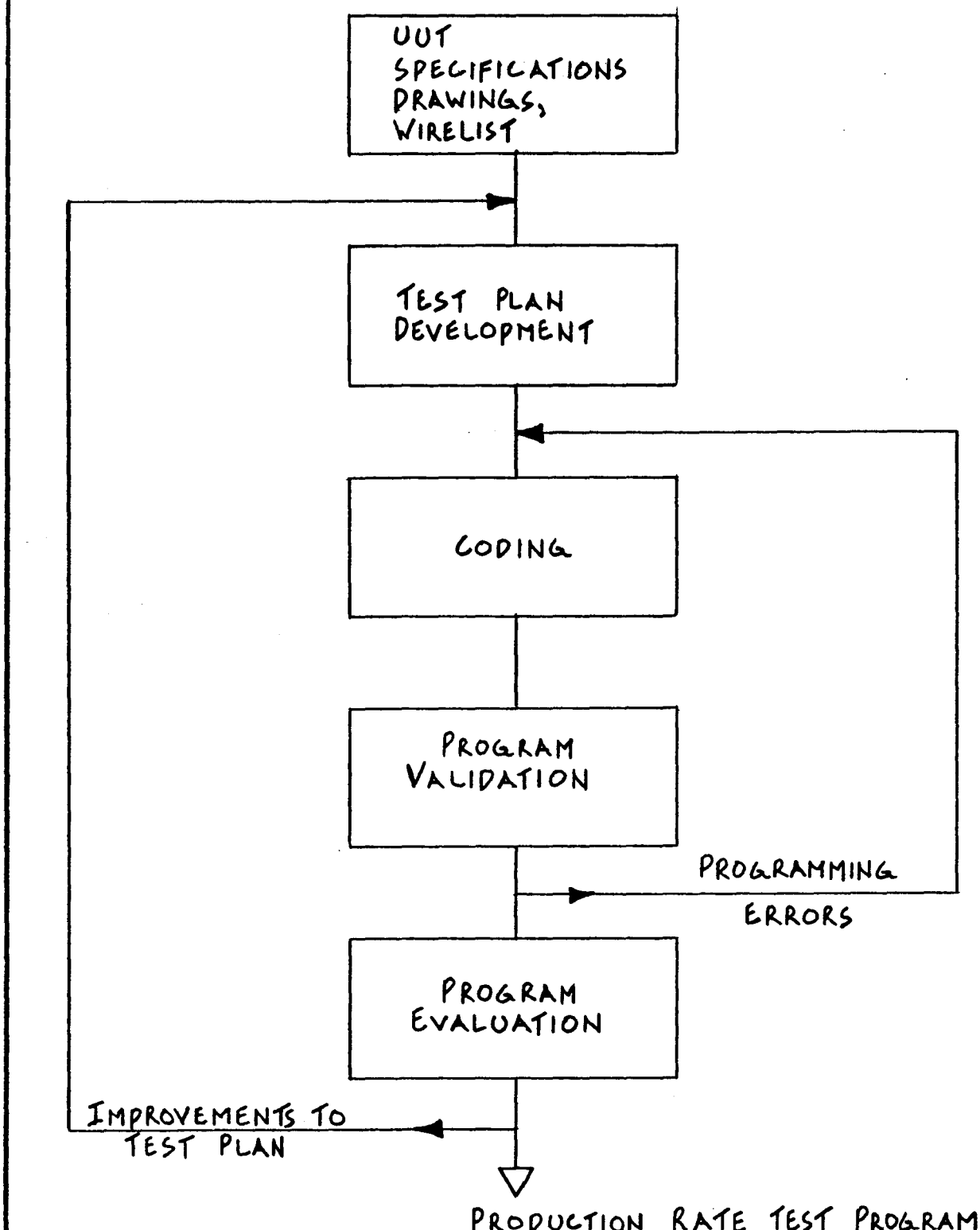


FIG. 7 THE TEST PROGRAM DEVELOPMENT PROCESS.

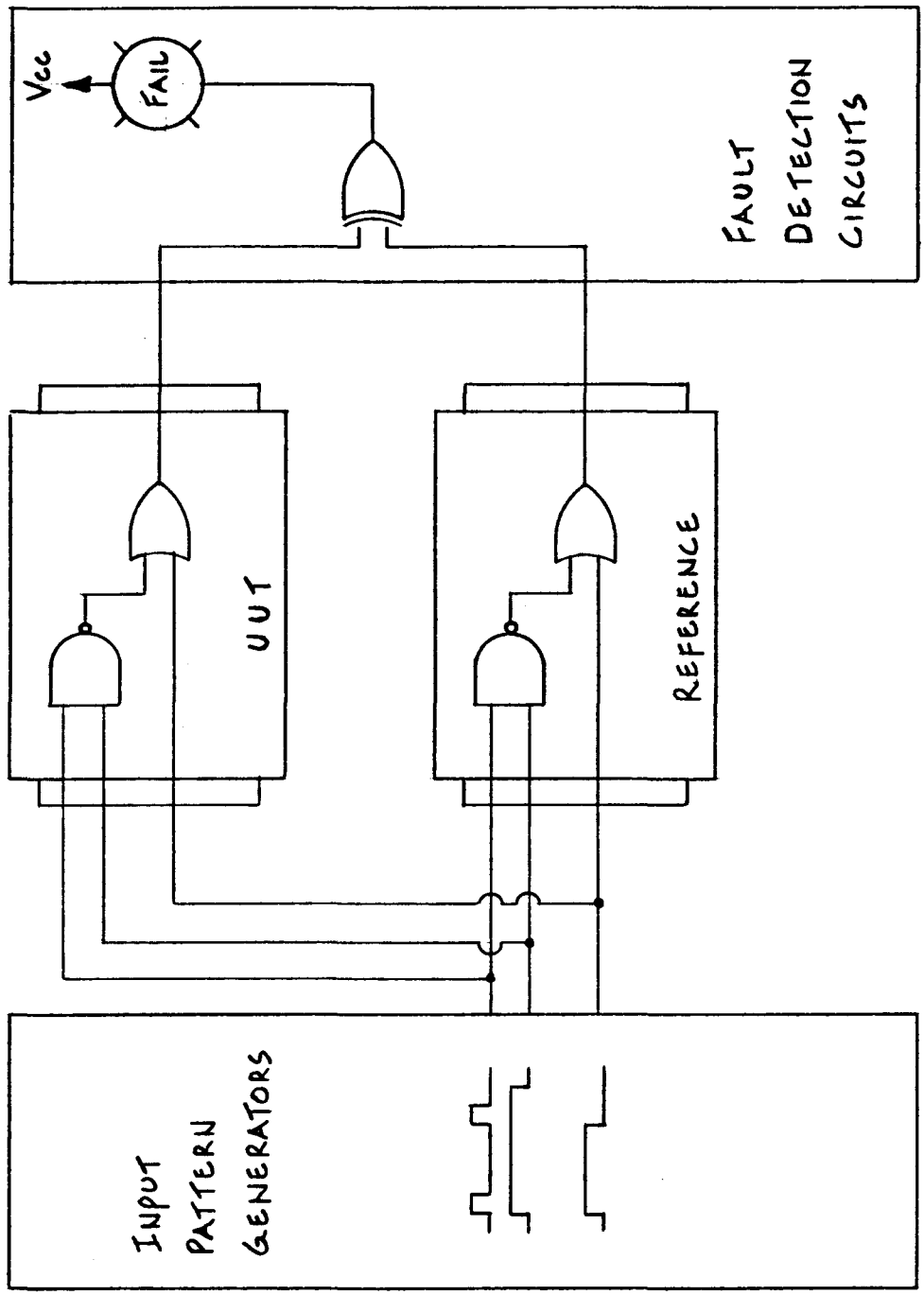
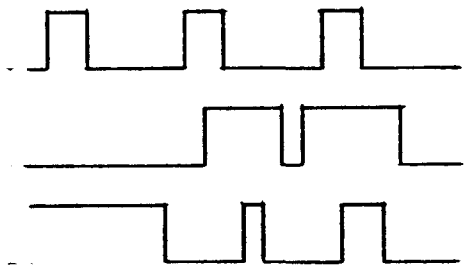
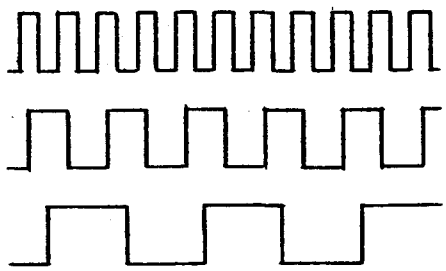


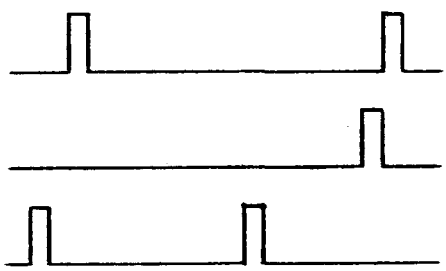
FIG. 8 AN AUTOMATIC SEQUENCE TESTOR.



9.A PSEUDO - GRAY CODE



9.B SQUARE - WAVE CODE



9.C 1 OF N CODE

FIG. 9 AUTOMATICALLY GENERATE INPUT TEST PATTERNS.

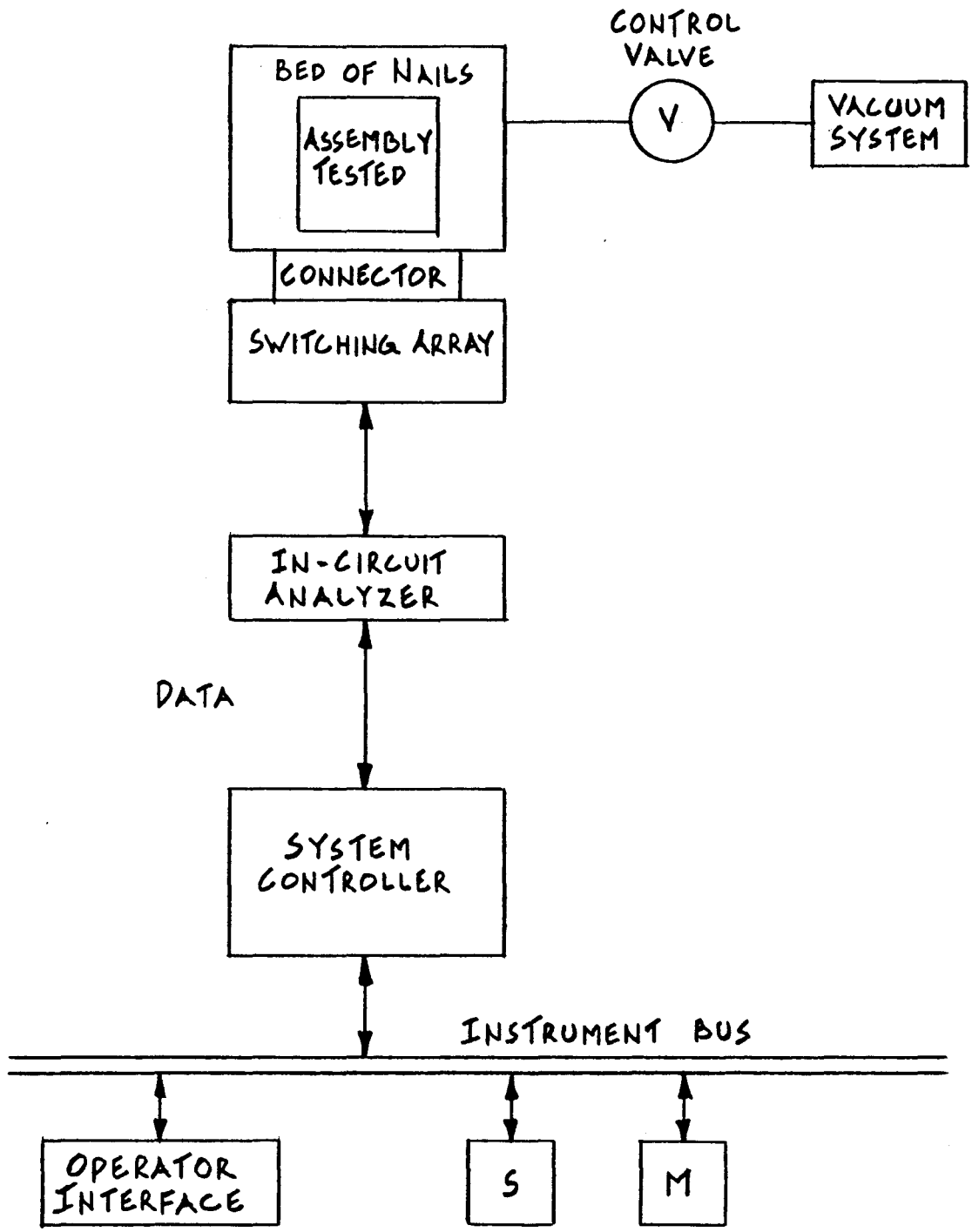


FIG. 10 THE ARCHITECTURE OF AN IN-CIRCUIT TESTER.

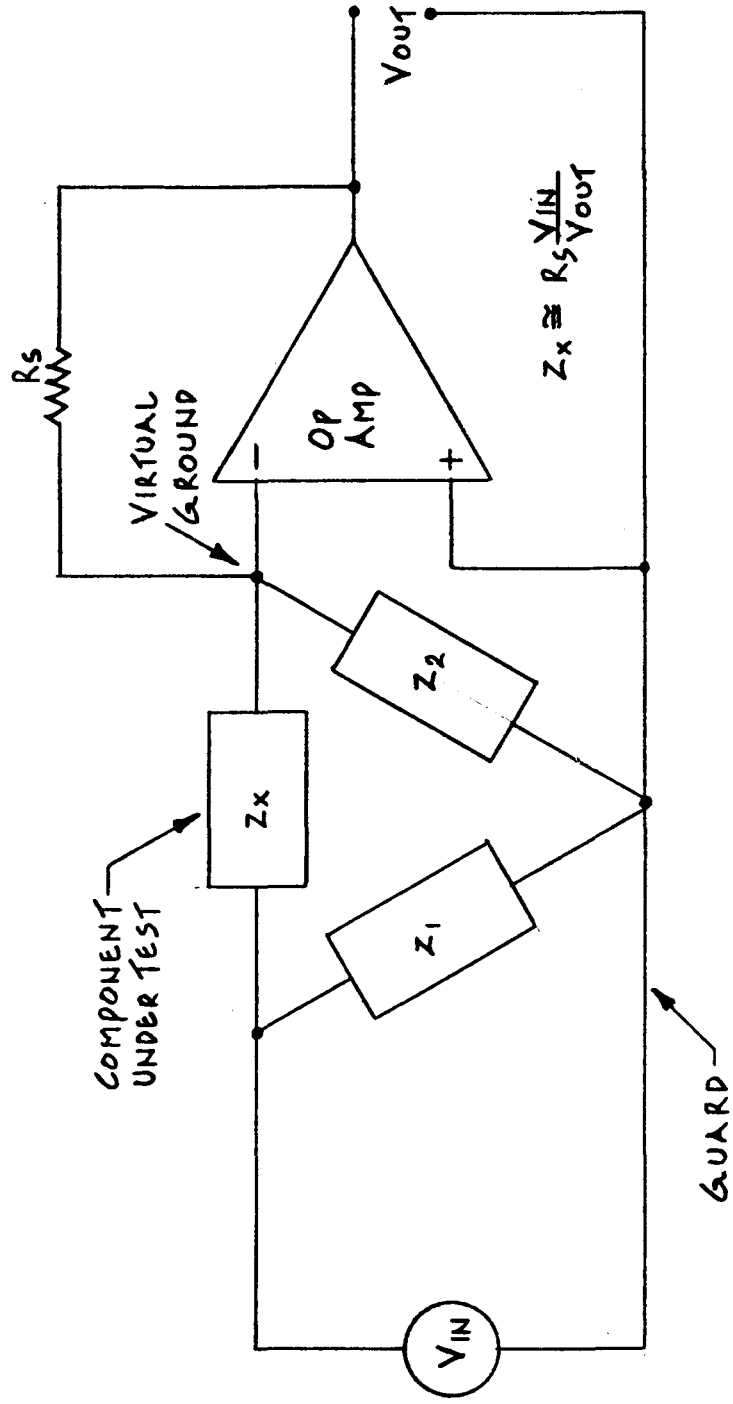


FIG. 11 GUARED MEASUREMENT TECHNIQUE.

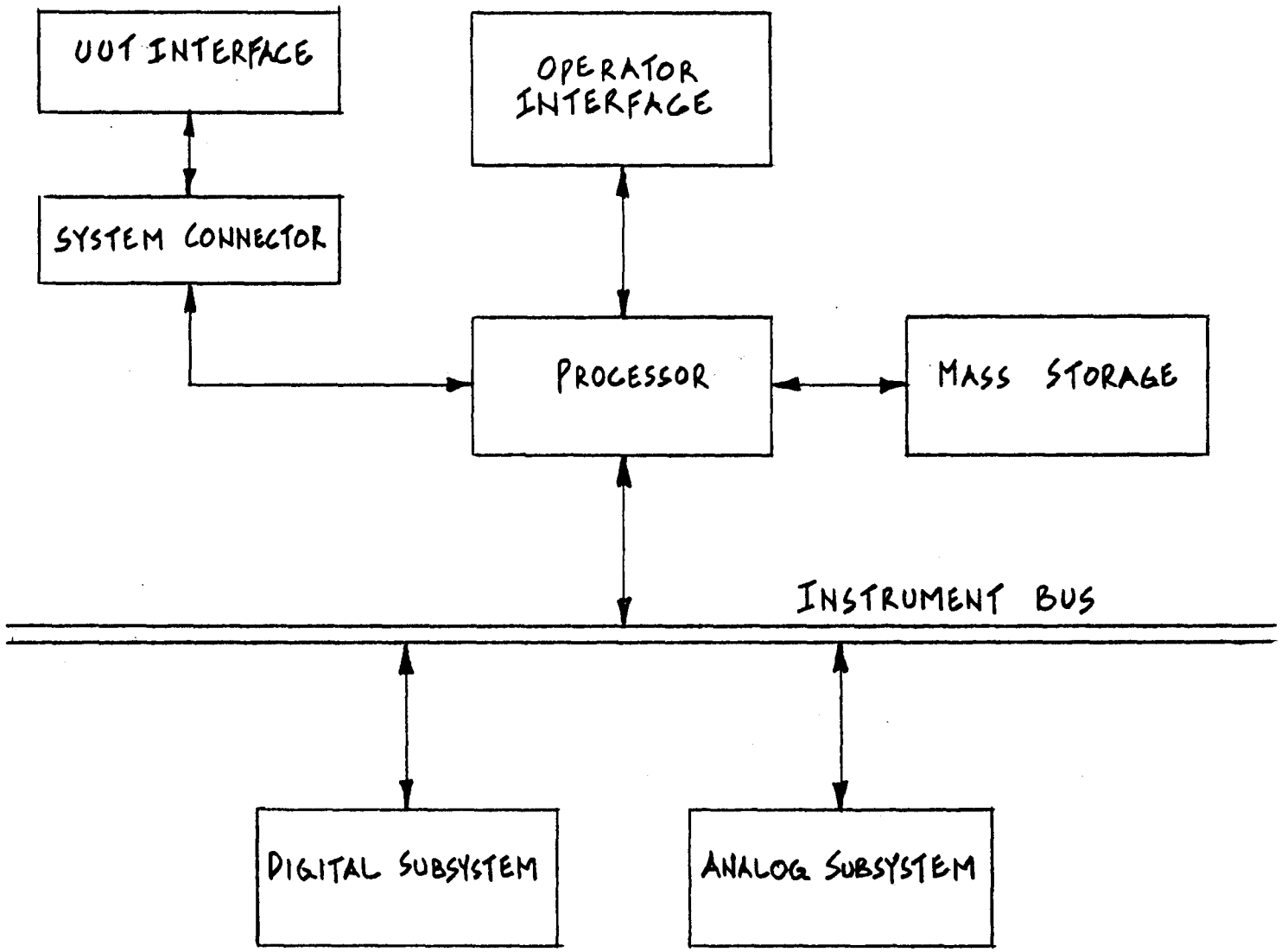


FIG. 12 DIGIPACT ARCHITECTURE.

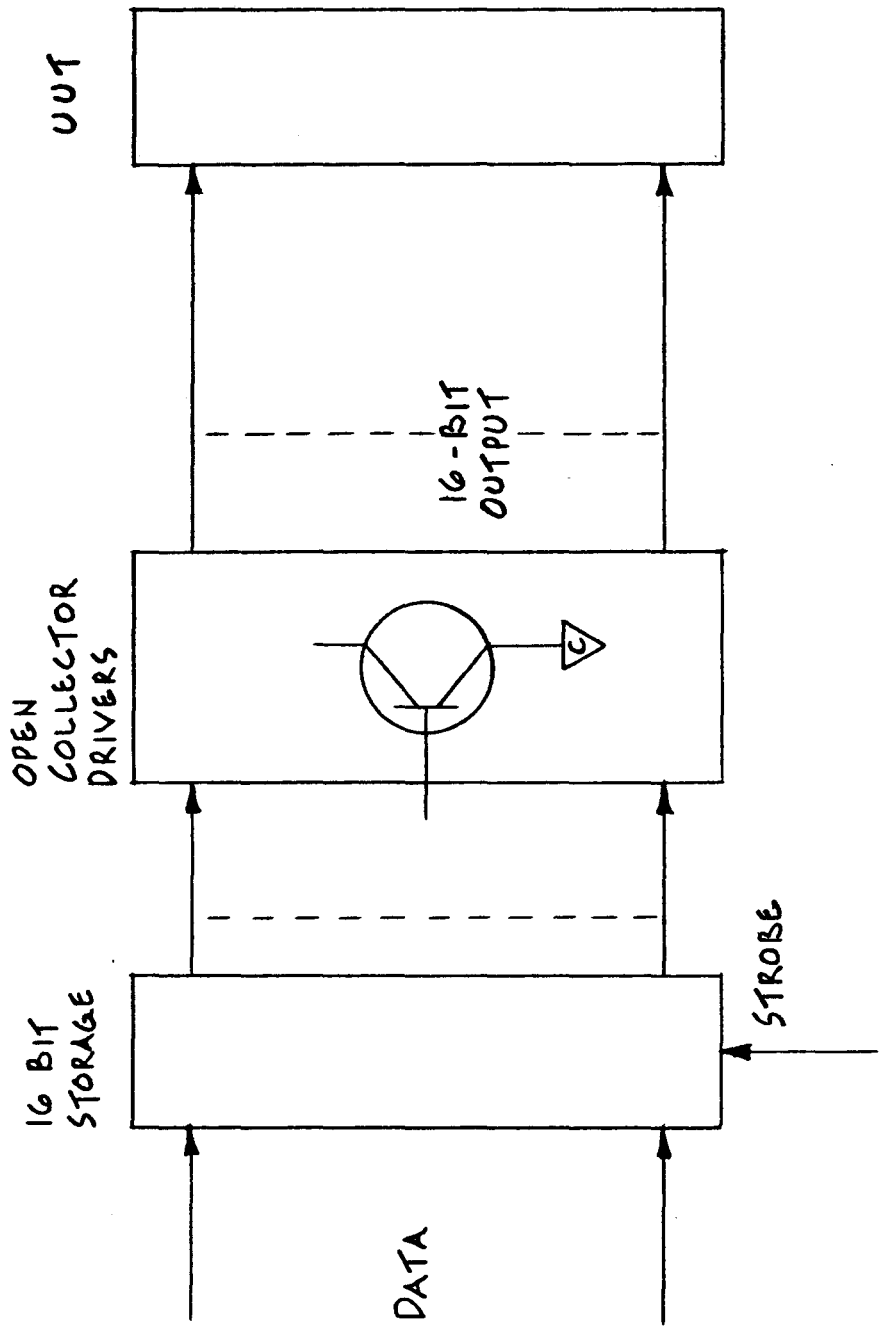


FIG. 13 OUTPUT CHANNEL

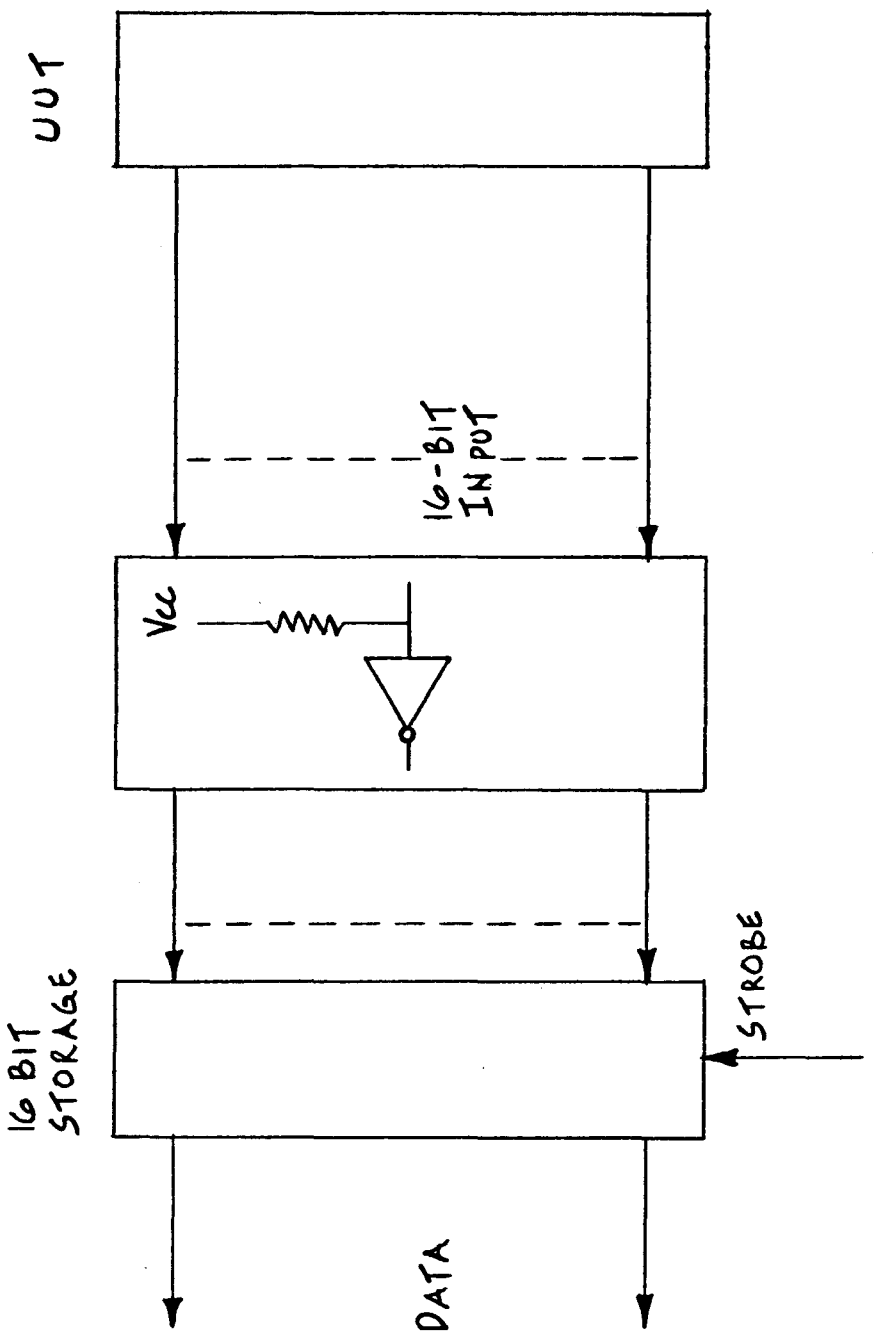


FIG. 14 INPUT CHANNEL

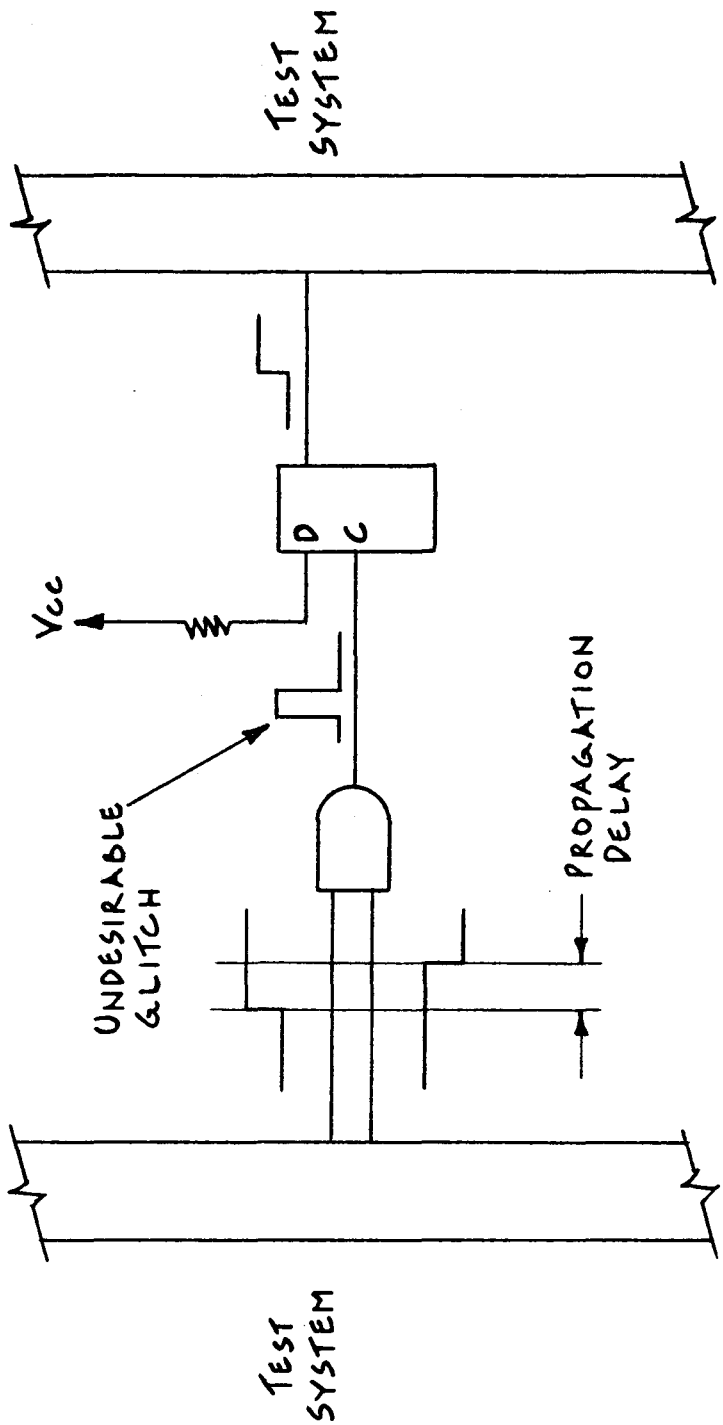


FIG. 15 A PROBLEM CAUSED BY GROSS PROPAGATION DELAYS OF THE INPUT SIGNALS.

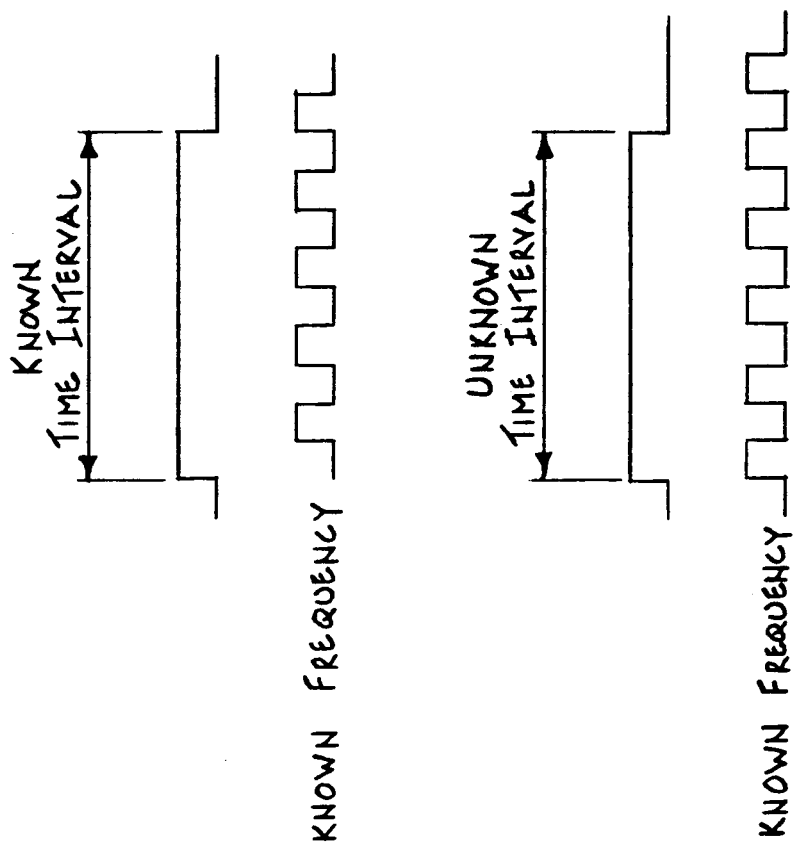


FIG. 16 TIME MEASUREMENT TECHNIQUES.

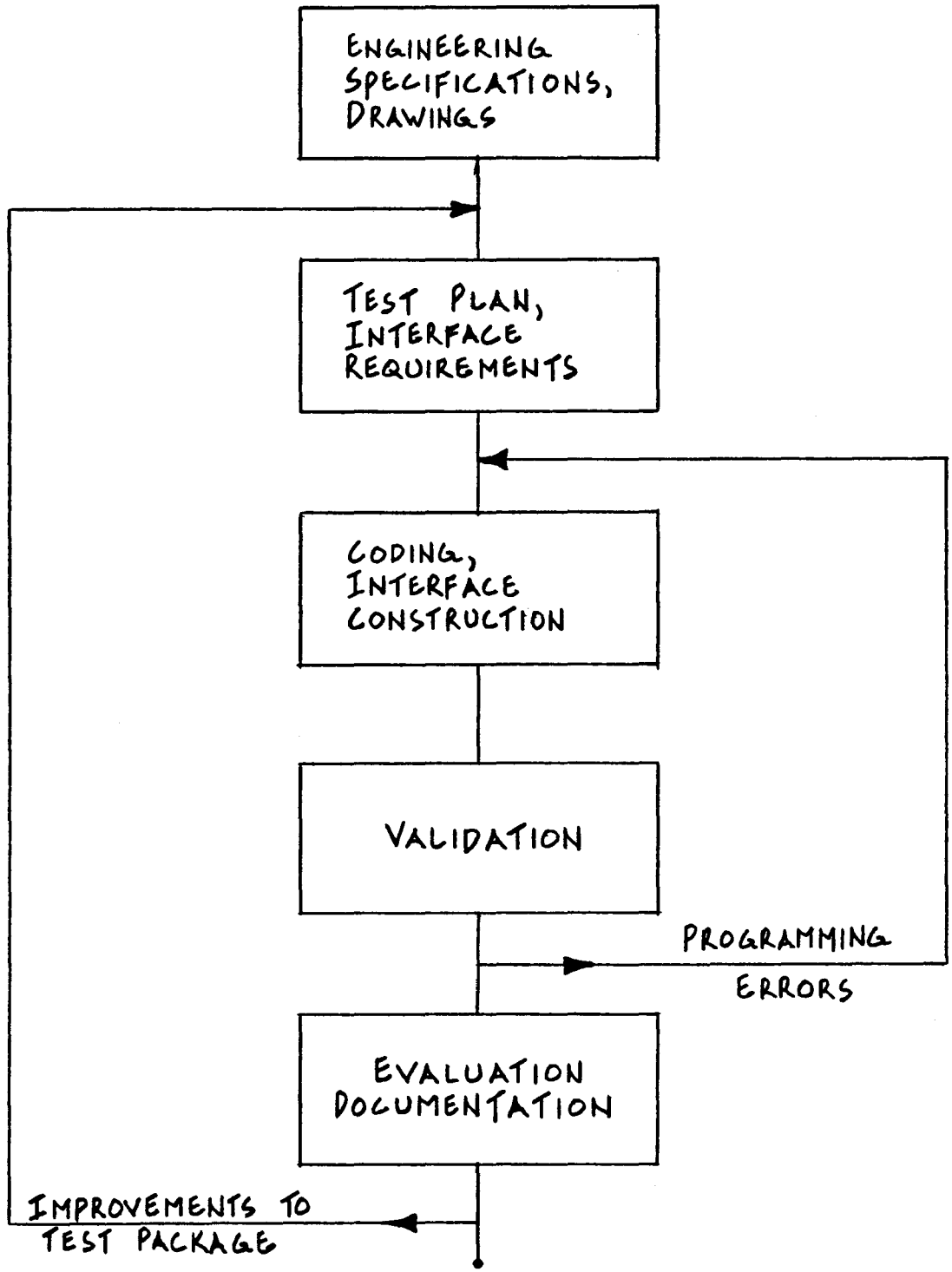


FIG. 17 TEST PACKAGE DEVELOPMENT PROCESS.

BIBLIOGRAPHY

1. Quality Control Instruction -13
   Litton Systems
   Quality Assurance
   In-Line Test/Final Test
   and acceptance.

2. Q.C.I. - 103
   Litton Systems
   Inspection instructions
   for P.C.B.s component
   boards and small assemblies

3. JN 4168 - 879
   GENRAD
   2270 in-circuit test
   system (product description)

4. 5952 - 8712
   Hewlet-Packard
   3052 Automatic Data
   Acquisition System
   (product description)

5. 5952 - 8502
   5952 - 8777
   Hewlet-Packard
   DTS - 70 Digital Test
   System - HP (Product
   Description)

6. Bulletin No. 101
   Fluke/Trendar
   Series 3000
   The Push Button Logic
   Tester (Product description)

7. Bulletin no. 103
   Fluke/Trendar
   3040A Logic Tester
   Questions and Answers
   (Product Description)

8. Bulletin No. 102A
   Fluke/Trendar
   The Economics of Logic
   Testing

9. Cork, Randall C., and Christopher H. Salzmann
   'Devising Patterns to Test Complex Logic Circuits',
   Electronics, August 14, 1972

10. McAller, Harold T., 'A Look at Automatic Testing' 'EEE SPECTRUM, May, 1974.

11. McHenry, Tom, 'In-Circuitry Vs. Functional Testers', Electronics Test, July, 1978.

12. DeSena, Tom, 'In-Circuit Test Systems Configurations, Techniques, Applications Electronics Test' November, 1979.

13. OminiComp., 'Handbook of Logic - Circuit Testing Volume I, Techniques and Implementations'.