

AN OPERATING SYSTEM FOR COMPUTER CONTROLLED
LOGIC PRINTED CIRCUIT BOARD TESTING

PETER H. TRAU, Eng.

A THESIS
in
The Faculty
of
Engineering

Presented in Partial Fulfillment of the Requirements for
the Degree of Master of Engineering at
Sir George Williams University
Montreal, Canada
March, 1972

ACKNOWLEDGMENTS

The author wishes to thank the following individuals for their invaluable help:

Dr. W. M. Jaworski of Sir George Williams University consented to act as the author's advisor and provided many helpful suggestions on the scope and subjectmatter of this thesis.

Mr. R. E. Enos, Eng., is the author's department manager at the Northern Electric Co., and gave the author a task which ultimately resulted in this paper. He also provided valuable guidance.

Mr. F. Benedetti, Eng., of the Northern Electric Co., designed the interface and testset of the logic test system, and offered much help on hardware considerations.

Mr. E. Beneteau of the Northern Electric Co. is the author of the translator program and cooperated closely with the author in implementing the software system.

CONTENTS

ACKNOWLEDGMENTS	ii
FIGURES	vii
FLOWCHARTS	viii
CHARTS	ix
TABLES	ix
1. INTRODUCTION	1
2. TESTING LOGIC CIRCUITS	4
2.1 LOGIC DEVICES	4
2.1.1 Example: SN7407 HEX BUFFER/DRIVERS with open collector high-voltage outputs	5
2.1.2 Example: SL-6-4025 MTNS QUAD 25-BIT STATIC SHIFT REGISTER	6
2.1.3 Testing TTL Integrated Circuits	8
2.2 LOGIC CIRCUIT BOARDS	9
2.2.1 Physical Characteristics	9
2.2.2 Possible Failures on a Printed Circuit Board ..	10
2.3 LOGIC CIRCUIT ANALYSIS	12
2.3.1 Analysis From a Logic Function Point of View ..	12
2.3.2 Analysis From a Logic Device Point of View ...	13
2.3.3 Analysis From a Random Error Point of View ...	13
2.4 TRUTHTABLE GENERATION	14
2.5 SOFTWARE IMPLEMENTATION	16
2.6 ERROR DETECTION AND CORRECTION	17
3. TEST SYSTEM HARDWARE	19
3.1 THE PDP-15/20 COMPUTER	19
3.1.1 Central Processor Description	20
3.1.2 Memory Organization	23

3.1.3	The Input/Output Processor	23
3.2	THE INTERFACE	25
3.3	THE TEST STATION	26
4.	TEST SYSTEM SOFTWARE	29
4.1	THE SOURCE TEST PROGRAM	29
4.2	THE TRANSLATOR	30
4.3	THE OPERATING SYSTEM	30
4.4	THE UTILITY PROGRAMS	31
5.	OPERATING SYSTEM SOFTWARE	33
5.1	OPERATING SYSTEM ORGANIZATION	34
5.2	OPERATING SYSTEM INITIALIZATION AND START	36
5.3	KEYBOARD MONITOR AND COMMAND REPERTOIRE	39
5.3.1	Keyboard Wait Loop	40
5.3.2	Data Mode Commands	42
5.3.3	Biasing Mode Command	43
5.3.4	Testing Modes	44
5.3.5	Test Initiation Command "Start"	45
5.3.6	Test Initiation Command "Continue"	47
5.3.7	"Partial Test" and "Looping"	47
5.3.8	"Repeat Current Line" and "Repeat Sequence" ..	50
5.3.9	The Step Mode	50
5.3.10	Test Program Verification	52
5.3.11	Housekeeping Commands	57
5.4	ELEMENTARY FUNCTIONS AND FUNCTIONAL TEST ROUTINES	57
5.4.1	FTR 4: Full Address Word	59
5.4.2	FTR 5: Partial Address Word	64

- 5.4.3 FTR 8: Manual Operation 64
- 5.4.4 FTR 9: Wait 68
- 5.4.5 FTR 11: General Release; End 69
- 5.4.6 FTR 12: Pin Connection Table; Truthtable
 Number 71
- 5.4.7 FTR 13: Octal Format; Truthtable Line Number . 74
- 5.4.8 FTR 14: Input Pin Toggle; Sequential Input
 Setting 74
- 5.5 PROGRAM INTERRUPT HANDLER 79
- 5.6 UTILITY SUBROUTINES 83
- 5.6.1 Biasing Group Subroutines 84
- 5.6.2 Output Levels Checking 87
- 5.6.3 Input Levels Modifications 91
- 5.6.4 Partial Testing Boundaries Location 92
- 5.6.5 Next EC 96
- 5.6.6 Keyboard Reading and Character Packing Group . 96
- 5.6.7 Transmission to Test Station 98
- 5.6.8 Input Pin Short Test 102
- 5.6.9 Station Status Check 105
- 5.6.10 Type-out Group 109
- 6. OPERATOR'S MANUAL 111
- 6.1 OPERATING SYSTEM INITIALIZATION AND START 112
- 6.2 AVAILABLE OPERATOR COMMANDS 117
- 6.2.1 Mode Commands 117
- 6.2.2 Test Initiation Commands 121
- 6.2.3 Miscellaneous Commands 128
- 6.2.4 Command Combinations 128

6.3 TESTING AND TROUBLESHOOTING TECHNIQUES	130
6.3.1 Test Program Prove-in	130
6.3.2 Sorting of PCB's	131
6.3.3 Troubleshooting	131
7. A SAMPLE TEST PROGRAM	133
7.1 CIRCUIT UNDER TEST	133
7.2 TESTING TRUTHTABLES	136
7.3 SOURCE TEST PROGRAM	138
7.4 TRANSLATOR OUTPUT	142
7.5 TEST RESULTS	146
7.5.1 Translation; Initial Run	146
7.5.2 Diagnostic Output Listings	148
7.5.3 Partial Testing	151
7.5.4 Short Test and Miscellaneous Error Diagnostics	154
7.5.5 Test Program Verification	157
8. CONCLUSIONS	159
APPENDIX A: OPERATING SYSTEM SOURCE PROGRAM LISTINGS .	160
APPENDIX B: 8-BIT AND TRIMMED 6-BIT ASCII CODES	209
LIST OF ABBREVIATIONS	212
REFERENCES	213

FIGURES

2.1	SN7407 Circuit Schematic (each buffer/driver) ...	5
2.2	SN7407 Dual-In-Line package (Top View)	6
2.3	SL-6-4025 Logic Diagram and Input/Output Relationship	7
3.1	Simplified Block Diagram of PDP- 15 CPU	22
3.2	Block Diagram of the Logic Test System	28
5.1	Operating System Core Allocation	35
5.2	EC 4 and FAW Bit Assignments	60
5.3	Input and Output Levels Tables	63
5.4	EC 5, Wordfill and PAW Bit Assignments	65
5.5	EC 8 and Wordfills Bit Assignments	66
5.6	EC 9 and Delay Wordfill Bit Assignments	68
5.7	EC 11 Bit Assignments	69
5.8	EC 12, Pin Connection Table and Definition Word	73
5.9	EC 13 and Octal Format Word Bit Assignments	75
5.10	EC 14 and Toggles Word Bit Assignments	77
5.11	Biasing Table "PWRWRD"	86
5.12	Status Word 0 (Station Status) Bit Assignments	87
5.13	Input Format to Subroutine SEND	99
5.14	Input Pins Table "INPINS" and Input Pin Short Test Table "SHTAB" Bit Assignments	105
5.15	Status Word 2 Bit Assignments	108
6.1	IC Pin Definition for Test Program Verification	124
	Circuit Under Test Schematic	134
	Testing Truthtable	137

FLOWCHARTS

5.1	OS Initialization and Start	38
5.2	Keyboard Wait Loop	41
5.3	Testing Under The "Start" Command	46
5.4	Testing Under "Loop" or "Partial Test"	49
5.5	Step Mode	51
5.6	Test Program Verification	54
5.7	EC Handler	58
5.8	FTR 4 (Full Address Word)	61
5.9	FTR 8 (Manual Operation)	67
5.10	FTR 11 (General Release; End)	70
5.11	FTR 12 (Pin Connection Table; Truthtable Number)	72
5.12	FTR 13 (Truthtable Line Number; Octal Format) ..	76
5.13	FTR 14 (Input Pin Toggle; Sequential Input Setting)	78
5.14	Program Interrupt Handler	81
5.15	Subroutine "CHKOUT"	89
5.16	Subroutine "LOCATE"	93
5.17	Subroutine "NEXTEC"	97
5.18	Subroutine "SEND"	100
5.19	Subroutine "SHORT"	103
5.20	Subroutine "STACHK"	106

CHARTS

6.1	Loading the Operating System	114
6.2	OS Start; Test Program Loading	115
6.3	Test Station Status Check; Keyboard Monitor ...	116
6.4	"Start" Command	125
6.5	"Partial Test" Command	126
6.6	Test Program Verification	127

TABLES

6.1	Data Mode Commands	119
6.2	Testing Mode Commands	120
6.3	Test Initiation Commands	122
6.4	Miscellaneous Commands	128
6.5	Summary of Commands and their Combinations	129

SIR GEORGE WILLIAMS UNIVERSITY
FACULTY OF ENGINEERING
GRADUATE STUDIES RESEARCH THESIS

by

PETER H. TRAU, Eng.

entitled

AN OPERATING SYSTEM FOR COMPUTER CONTROLLED
LOGIC PRINTED CIRCUIT BOARD TESTING

ABSTRACT

An Operating System for a computer controlled logic printed circuit board test system is presented. This Assembler language program was developed by the author for a Northern Electric Co. designed manufacturing test facility, consisting of a Digital Equipment Corporation PDP-15/20 computer, and an interface and testset built by the Northern Electric Co. It tests logic printed circuit boards containing up to 45 integrated circuits, according to stored test programs. The Operating System decodes the test programs and controls the testset. An extensive command repertoire is provided to facilitate error location and correction. This test system has been successfully used to test over 25 different logic printed circuit board codes.

CHAPTER 1
INTRODUCTION

Northern Electric Co. Ltd., the author's employer, conducts a significant part of its business in the design and manufacture of electronic switching systems. All recent designs employ DTL and TTL Integrated Circuits, currently up to medium scale integration on devices such as encoders, decoders, converters and shift registers.

The logic devices and associated circuit components are mounted on printed circuit boards, of double-sided design, with 86 access terminals (43 to a side) on one edge of the board, and containing up to 45 IC's. This quantity allows for major logic functions, or multiple identical functions, to appear on a single circuit board. It also results in high printed circuit path densities with minimum path-to-path separations of one ten-thousands of an inch.

It is of course necessary to test such logic circuit boards. Whereas binary logic is simpler to check than an analog electronic circuit of similar device density - that is if one omits rise, fall, and delay time measurements for the logic circuit - the aforementioned logic device densities result in extremely large numbers of tests. Hence arises the need for automatic testing facilities. Furthermore, since there exist a large number of different circuits, the best solution to the testing problem lies in a digital computer controlled automatic test system. This can operate under stored-program control, and can thus check different circuits according to different test programs.

Employing this philosophy the Northern Electric Co. designed such an automatic test system. It consists of a Digital Equipment Corporation PDP-15/20 Computer, which was purchased, and a logic testset connected to the computer via an interface, both of the latter designed and built by the Northern Electric Co. The author's task was to develop an Operating System for this facility. This Operating System should load and decode test programs, and control the testset accordingly. It should also facilitate the on-line debugging of faulty logic circuits.

The following chapters of this paper, which deals primarily with software, will include some hardware description of the test system. Due to confidentiality considerations these will necessarily be sketchy since the author was granted permission to publish only the Operating System. The philosophy employed in testing logic circuits will be discussed to give the reader more perspective over the requirements of the Operating System, but it should be stated that this philosophy was well established in the Company when this testing system was designed. Hence any of the Author's ingenuity was restricted to maximizing the versatility of the Operating System within the constraints expressed in the test system hardware.

While this system was specifically designed to test logic circuits for SG-1 EPBX, an electronic PBX (Private Branch Exchange) system newly developed by Bell-Northern Research, and the Northern Electric Co., as a stand-alone unit, its hardware was designed to be compatible with other

existing testsets. This was done with the intent of an eventual time-shared test system, capable of operating concurrently up to seven separate testsets via one computer. This concept, which is now in the software design state, determined the overall software philosophy of the test system, adding further constraint on the Operating System. It will be possible, if so desired, to include this test system in a time-sharing configuration, without hardware changes, and with relatively minor software modifications.

Along with this introduction, chapters 2, 3, and 4 are meant essentially as preambles to the Operating System. Chapter 2 will deal with logic devices and circuit boards, and why they have to be tested. It presents the 'raison d'être' for the test system. Chapter 3 presents a brief description of the test system hardware, consisting of the computer, the interface, and the actual testset. Chapter 4 deals with the entire test system software, of which the Operating System is but a part. The Operating System software is finally discussed in Chapter 5. Chapter 6 is titled Operator's Manual and describes the Operating System usage from an operator's point of view. Part of an actual test program is presented in Chapter 7, along with a representative sample of computer input and output during the testing process.

CHAPTER 2

TESTING LOGIC CIRCUITS

The test system described in this paper was designed to test and trouble-shoot logic circuit boards. The following sections will describe the logic devices and circuit boards, give some of the reasons for testing them, and detail the steps necessary to generate an automatic testing procedure for a logic circuit.

2.1 LOGIC DEVICES

We are dealing here exclusively with binary logic devices existing, as the name implies, in one or the other of two possible states. Furthermore these devices operate in a positive logic mode. Hence their "level 0", "low", or "off" state is characterized by an output voltage near ground (0 VDC), whereas their "level 1", "high", or "on" state results in an output voltage typically above +2 VDC.

While this test system is capable of testing any positive logic device or circuit, given certain restrictions explained in the following chapters, it was specifically designed for logic circuits comprised of TTL (transistor-transistor logic) or TTL-compatible IC's, of which two examples follow. These IC's require a +5 VDC positive bias and in certain cases an additional -12 VDC negative bias. One IC pin supplies a return to ground. The devices have 14 or 16 access pins generally in a dual-in-line package.

2.1.1 Example: SN7407 HEX BUFFER/DRIVERS with open-collector high-voltage outputs (1)

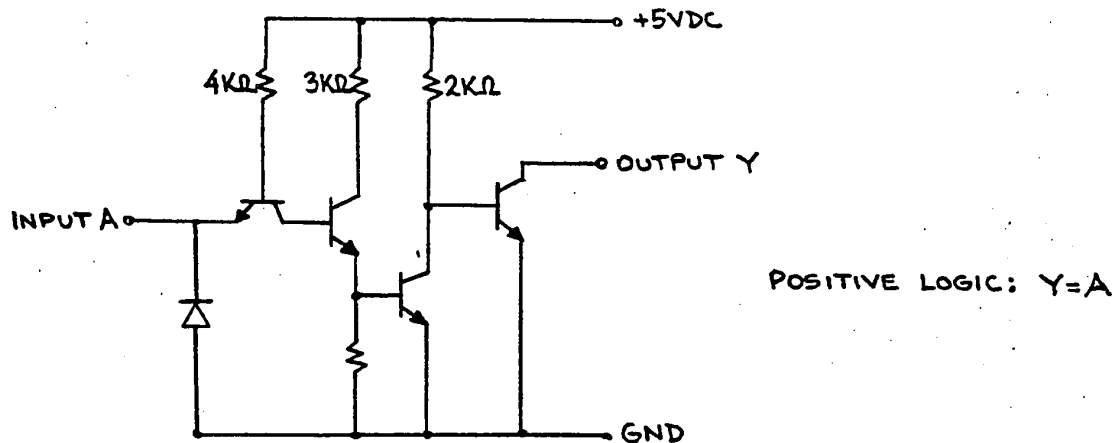


FIGURE 2.1: SN7407 Circuit Schematic (each buffer/driver)

These monolithic TTL hex buffers/drivers feature high voltage open collector outputs for interfacing with high-level circuits (such as MOS), or for driving high-current loads (such as lamps or relays), and are also characterized for use as buffers for driving TTL inputs. For increased fan-out, several buffers in a single package may be paralleled.

Some of the electrical characteristics of this device are:

Supply voltage V_{cc} : +15 VDC max.

Low level output voltage V_{ol} : 0.4 - 0.7 VDC according to load.

Low level sinking current I_{ol} : 40 mA max.

Propagation delay time, low-to-high level output, at

nominal bias, 110 ohm load: 17 - 26 ns (Turn-off time).
 Similarly, turn-on time: 10 - 15 ns.

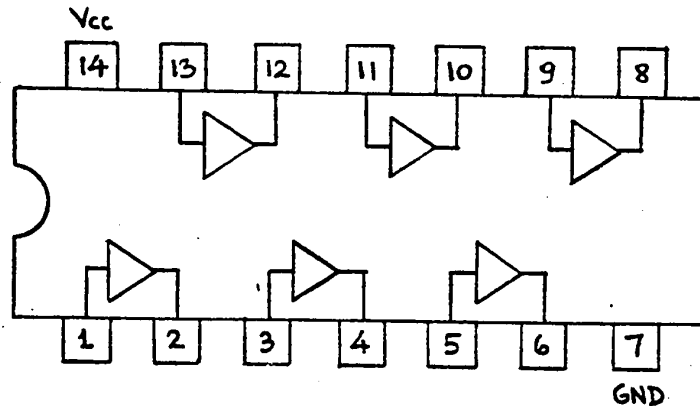


FIGURE 2.2: SN7407 Dual-In-Line Package (Top View)

2.1.2 Example: SL-6-4025 MTNS QUAD 25-BIT STATIC SHIFT REGISTER (2)

This is a static shift register belonging to a standard family of DTL/TTL/MOS compatible registers which are constructed using low threshold silicon nitride passivated P-channel enhancement mode Field Effect Transistors.

Each register has one serial input and one serial output, and the clock input is common to the four registers. all inputs, including clock, can be driven directly from DTL/TTL logic levels, and each output can directly drive DTL/TTL without external interfacing components.

Data is read into each register when the clock input is at a logic 0 and is read out following the positive going (0 - 1) clock transition. Data can be stored indefinitely

in the register with the clock held in either logic state.

Since this device is TTL compatible from a "black box" point of view, its biasing and input/output level characteristics are similar to the TTL device described in the previous section. The only major difference with TTL logic is the additional requirement of a -12 VDC bias, necessitated by the Metal Thick Oxide Nitride Silicon (MTNS) process employed in the device's manufacture.

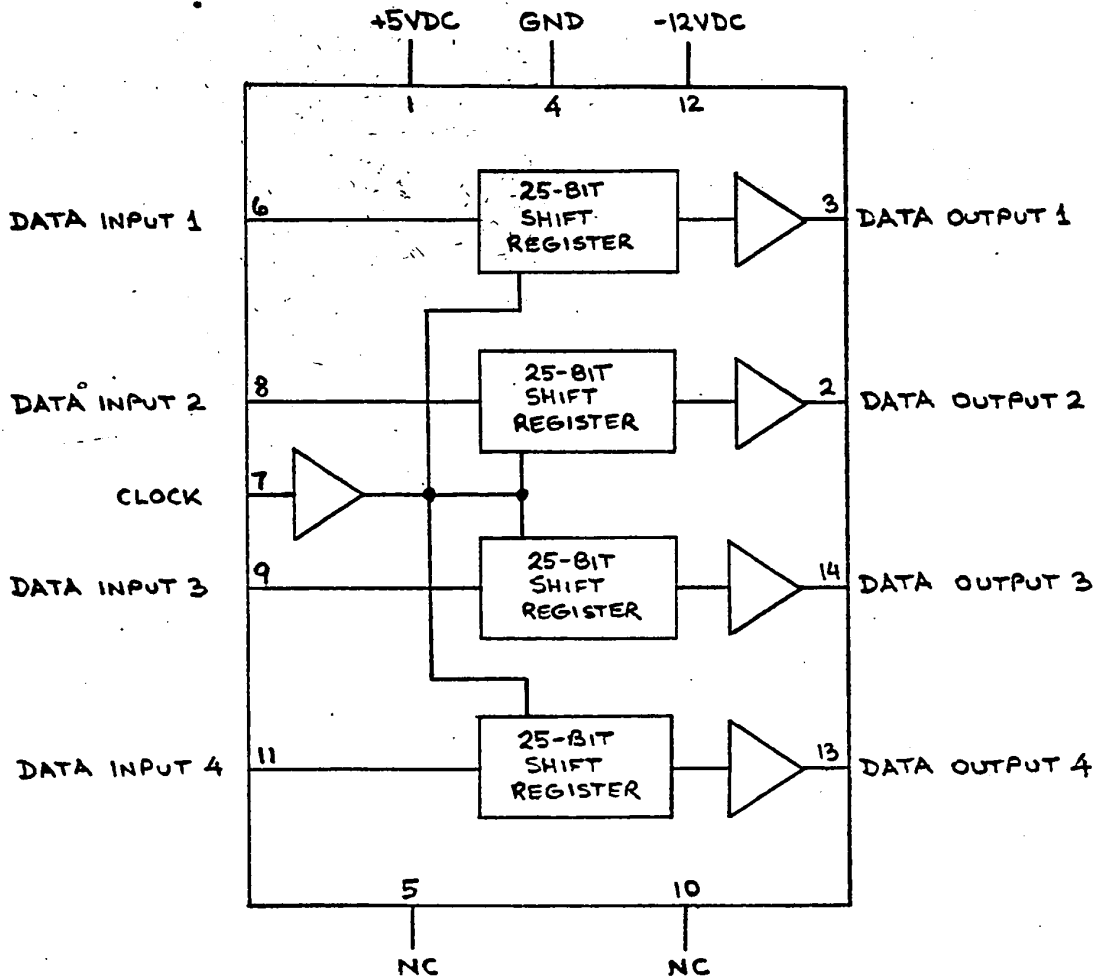


FIGURE 2.3: SL-6-4025 Logic Diagram and Input/Output Relationship

2.1.3 Testing TTL Integrated Circuits

A manufacturer selling TTL IC's provides his customers with specification sheets detailing the device's parameters which should be met under specific test set-ups. Broadly speaking, these are:

- (a) Switching speeds
- (b) Noise immunity
- (c) Steady-state input/output characteristics as affected by temperature and DC-bias variations, and loading variations.

The manufacturer then guarantees a certain maximum percentage of faulty devices, say 1%.

To guarantee perfectly functioning components before inserting them into printed circuit boards would, because of large volume, require an automatic test set-up for IC's to measure the above parameters. This would necessitate a large capital expense due to the sophisticated hardware required to automatically check switching speeds and noise immunity. Furthermore, since assembled printed circuit boards will have to be tested also, due to reasons which will be outlined in section 2.2.2, and since it is impractical to test switching speeds and noise immunity on a circuit board basis, two different and expensive testsets are required. Considering then the IC manufacturer's guarantee of 1% rejects, it is sufficient to concentrate on testing circuit boards, and only check IC's on a sample basis which can make do with manually operated facilities.

When checked as part of a logic function on a circuit board individual IC's are checked for correct steady-state input/output characteristics under fixed loads, current ambient temperature, but, if required, under varying DC-bias and input level conditions. This corresponds to the margin testing feature which will be described in the following chapters.

2.2 LOGIC CIRCUIT BOARDS

The term "logic circuit boards" describes printed circuit boards containing logic devices (although there may also be some devices performing analog functions), which form part of, or entire, logic functions. As a logic system is partitioned onto individual circuit boards it may not always be possible to include entire discrete logic functions on a single board. Hence such boards contain incomplete parts of logic functions which may or may not result in separate identifiable functions. Also, as was already stated, there may be analog circuit components on part of the circuit board, resulting in hybrid circuits. The varied portions may feed each other, resulting in logic inputs and analog outputs, or vice-versa.

2.2.1 Physical Characteristics

The type of printed circuit board (PCB) employed in the SG-1 EPBX system is double-sided, with printed circuit paths running north-south on one side, and east-west on the other

side of the board. There are 86 access pins on one edge, 43 to a side. Circuit components are mounted on only one side, the component or "A" side. The other side is denoted the wiring or "B" side. Certain access pins are identically defined on most of these logic boards - B1 is ground, B3 and B43 carry the -12 VDC and +5 VDC bias respectively.

There is provision for up to 45 IC's plus certain load resistors, filter capacitors, etc. on a board. This can result in high printed circuit path densities, with minimum path-to-path spacings of one ten-thousands of an inch. Circuit connections to devices may be made on either side, at device pins inserted into plated-through holes. These also serve to connect circuit paths on the A and B sides.

2.2.2 Possible Failures on a Printed Circuit Board

To give a better idea of the reasons for spending large sums in designing or purchasing automated test equipment for logic PCB's, the following list of failure types is presented.

(a) Component failure - This may be caused by defective or marginal components, components improperly inserted, or incorrect component types or values. As was already stated, the manufacturer may guarantee less than 1% failures in IC's. With up to 45 IC's per PCB this fact alone could result in one failure per two or three boards. While dropout due to this cause is happily much less, errors made during PCB assembly, which accounts for the other types of errors of component failure, largely offsets any gains made here.

(b) Solder bridges between circuit paths - This results from poor wave soldering of closely spaced circuit paths and may cause failures on 10% of the PCB's tested. Visual inspection after wave soldering and before testing catches a large percentage of this error type.

(c) Matrix errors - This includes printed circuit path routing errors, improper etching resulting in hair-thin bridges between adjacent paths, improperly plated-through holes which causes open circuits, or hairline cracks on circuit paths, which has the same result. Circuit paths routing errors are usually caused by improper checking of artwork when a new circuit is laid out, or when an old design is changed. This type of error will be picked up when the corresponding PCB test program is being proved in, and will thus seldom enter into manufacturing testing. All of the other types of matrix errors mentioned above may occur during manufacturing testing.

(d) Poor connector contacts - Grease or dirt on the gold-plated PCB access pins can result in an open-circuit condition between the PCB and the test set connector. This is enhanced by non-wiping connectors which make contact by being clamped down onto the goldfingers, and thus do not wipe them clean before contact is made as is the case when the PCB is plugged into an ordinary wiping connector. Non-wiping connectors are used during testing to insure that the PCB goldfinger contacts are not scratched (they might otherwise corrode).

These failures may each result in a single test failing,

or a catastrophic failure ^h were devices are destroyed when power is applied to the PCB. Considering also that thousands of individual tests may be necessary to completely check a logic PCB, it becomes clear that automatic test facilities are necessary to insure proper PCB performance.

2.3 LOGIC CIRCUIT ANALYSIS

As was already mentioned in section 2.2 individual PCB's do not necessarily contain entire logic functions from a system point of view. Rather they may contain a seemingly random collection of interconnected logic devices with no apparent identifiable function. The system designer, when partitioning his circuits onto individual PCB's, rarely bothers to rewrite his logic equations to fit the bits and pieces of circuitry ending up on a particular board.

All logic circuit analysis as far as this test system is concerned is done from a steady-state DC level point of view. Switching speeds and possible hazards caused by critical races are considered only so far as to avoid simultaneous changes to several direct inputs to sequential circuits.

2.3.1 Analysis From a Logic Function Point of View

Whenever an identifiable logic function appears on a PCB all of its design-required input/output relationships should be checked. If they are all correct, and one assumes that don't cares are indeed meaningless, nothing else need be tested for this particular circuit. This is strictly a

functional test as far as the steady-state input/output conditions are concerned. It must be remembered that switching speeds and associated hazards are not checked. This is based on the assumption that the designer has taken care of this. The function of the PCB tester is not to verify a design, but rather to make certain that all required devices are functioning correctly, and are mounted and interconnected as specified. In the case of an identifiable logic function this purpose is achieved also by checking the function.

2.3.2 Analysis From a Logic Device Point of View

Some PCB's contain odds and ends in logic devices which are mounted with their input/output pins wired directly to the PCB access terminals. Or the devices may be interconnected in small groups. These must be analyzed from a device point of view - that is each device must be checked for its particular function.

2.3.3 Analysis From a Random Error Point of View

Random errors are those caused by solder bridges or matrix errors resulting in open and short circuits. Certain PCB's contain multiple identical logic circuits and functions. Whereas these circuits are separate as far as their inputs and outputs are concerned, a solder bridge for example could short part of one circuit to part of another. This may not be detected when these circuits are checked individually and separately. It may result in faults when both circuits operate concurrently.

It is extremely difficult - if not impossible - to try and analyze a circuit from the point of view of including sufficient additional tests to try and pick up every conceivable short and open circuit that might occur between it and other separate circuits. There are certain procedures followed to minimize the possibility of overlooking such an error. These include exercising separate circuits concurrently and checking all outputs all of the time, testing twice - once with all unused inputs set low, once set high, and employing Test Program Verification (described in subsequent chapters) which assures that the specified tests will pick up any open or short circuit condition at any IC access pin.

2.4 TRUTHTABLE GENERATION

Truthtables, or tables of combination, describe the steady-state relationship between the input and output levels of a logic circuit. Complete truthtables consist of 2^n lines of input/output relationships, where n is the number of discrete inputs to the circuit. Testing truthtables are usually subsets of complete truthtables since it is often unnecessary to subject a logic circuit to 2^n tests in order to check it.

Some work has been done to computerize the generation of testing truthtables for PCB's. This usually involves entering a device interconnection map into memory, along with the device codes. Each device code corresponds to a subroutine. The analysis program then tries to determine PCB input pin setups so as to exercise each device input and output pin at

least once per logic level. It then gives the expected output levels at the PCB access terminals. As the number of separate devices grows, so must the number of defining subroutines. Such analysis programs require large computer memories because of their large size. Their greatest disadvantage lies in their inability to generate trouble-shooting information. If certain ones of hundreds of thousands of tests fail it becomes imperative to provide exact information as to the input/output conditions of the PCB at the time of failure, as well as the actual circuits currently under test.

In any case, the first step in generating a logic PCB test program is to analyze the logic circuits and write testing truthtables. These must include sufficient tests to exercise each logic device enough to insure correct functioning. In the case of combinational circuits it is sufficient to set each device input pin at least once low, and include any additional tests necessary to have the output change state at least once. In the case of sequential circuits the capability of memory must also be checked. Furthermore, additional tests may have to be added to check for possible interaction between separate circuits. The testing truthtables will clearly state PCB input and output terminal numbers and possible intermediate testpoints exercised and checked at any time. Coordinate information must be provided to the test program to indicate the exact location within any truthtable as a test is performed. These truthtables are then used to both encode the testprogram, and to provide trouble-shooting information.

2.5 SOFTWARE IMPLEMENTATION

As the name implies this is the encoding of a testing truthable into some form of computer language to permit the computer to test the PCB via a logic testset.

There are of course several computerized logic PCB testsets manufactured by various companies, and each has its unique accompanying software package. Most of these are "high level" to facilitate the encoding of testing truthables without requiring much knowledge of computer programming. There is little to choose in this respect between Hewlett-Packard's 2060A Logic Test System, Computer Automation Inc. Capable Tester, or General Radio's 1790 Logic Circuit Analyzer. This is of course understandable when one considers that there are only so many things one can do with logic circuits, given the constraints of steady-state level testing only. What separates the men from the boys in this business is the power of the error detection and correction tools provided by the tester's operating software and hardware.

If a "high level" language is used to encode testing truthables in readable "English" statements, this must then be translated into an input acceptable to the operating system which communicates with the testset. This can be either a one or two pass affair. In a one pass system translator and operating system are part of the same program and hence co-resident in memory as the test program is being executed. This has the advantage of usually permitting tests to be modified,

or additional tests to be entered when a PCB is under test. (This corresponds to on-line editing). Its disadvantages are large core requirements for the software system and hence less available storage for test programs, and increased test execution times as each statement must be translated before being executed each time the test is to be performed.

A two-pass system separates the translating and execution functions. Here the "English" test program is converted by a translator program into a format usable as direct input to an operating system. When a PCB is to be tested this translated test program is then loaded by the operating system and executed. Here the advantages are lesser core requirements for the operating system which can hence be more sophisticated, increased test program storage availability for more tests as the test program statements are "condensed" by the translator, and improved execution speeds. The disadvantages are the usual lack of direct translation capability by the operating system, although some manufacturers have provided some translation capability for their operating system either directly, or as a separate translator-editor-operating system package. Also, the translator output has to be stored in addition to the original source test program.

2.6 ERROR DETECTION AND CORRECTION

The purpose of a computer controlled logic testset is not only to sort out faulty circuits but also to provide on-line debugging facilities. Considering the complexity

of logic circuits such debugging facilities are a combination of hardware and software devices intended to minimize the time required to locate and correct a failure.

Hardware devices include manual measuring instruments such as logic probes and volt-ohmmeters, or automatic checking of testpoints within a circuit under test. Software devices comprise extensive command repertoires for an operator to initiate partial or modified testing in order to zero in on the fault. There is also the possibility of including error directories which store fault sequences for single errors and can hence identify and locate them. This becomes impractical for multiple errors.

CHAPTER 3

TEST SYSTEM HARDWARE

This chapter gives a general description of the test system hardware. This comprises the computer, the logic PCB testset, and the interface linking the two. Both of the latter were designed and built by the Northern Electric Co.

Since this dissertation concentrates on software, the following is intended only to place the Operating System into the context of its hardware constraints. It should be noted that once it has been decided which computer to purchase as a controller, certain limitations are placed on peripheral hardware design. Similarly the testing hardware of necessity governs the design and function of the software. For optimum efficiency the various hardware items as well as the software must be matched to exclude any inconsistencies.

3.1 THE PDP-15/20 COMPUTER

The computer is a Digital Equipment Corporation PDP-15 data processor with 8192 18-bit words of memory and an 800 ns cycle time. Its input/output devices comprise a KSR-35 Teletype (keyboard and printer only), a high-speed paper tape reader (300 characters/second) and punch (50 characters per second), as well as two Dectape (Digital Equipment Corp. magnetic tape) drives, each of which can store 150,000 words per reel.

For purposes of comparison, the General Radio 1790

Logic Circuit Analyzer standard system⁽³⁾ employs a computer with 4096 12-bit words of 1.6 micro-second cycle core memory. It includes a Teletype including paper tape reader and punch, and a high-speed photoelectric paper tape reader.

Computer Automation Inc. Capable Test System⁽⁴⁾ in its standard configuration includes its 208 minicomputer, which has a memory of 8192 16-bit words. Its input/output equipment includes an ASR-33 Teletype with paper tape reader and punch, and a high-speed (300 characters/second) photoelectric paper tape reader.

The basic Hewlett-Packard 2060 Digital Logic Module Test System⁽⁵⁾ is furnished with an HP 2116B computer with 8192 16-bit words, a modified ASR-35 Teletype with paper tape reader and punch, as well as a high-speed paper tape reader.

None of the three systems include mass storage devices such as magnetic tape in their basic configurations. All have expandable memories. There is no attempt to make value judgments on the relative merits of one system over another, since the author was not part of the decision-making process in the test system hardware configuration. The following sections give brief descriptions of the PDP-15 organization.

3.1.1 Central Processor Description

The Central Processor (CPU) is the nerve centre for control and execution of stored programs. By coordinating its own operation with that of other subsystems, it provides supervisory control over the PDP-15 System.⁽⁶⁾ It contains arithmetic and control logic hardware, several registers

for processor-memory communications, a program counter, instruction register, accumulator (as well as index and limit registers not used in the Bank mode operation).

The CPU performs calculations and data processing in a parallel binary mode through step-by-step execution of individual instructions. Both the instructions and the data on which the instructions work are stored in the core memory of the PDP-15. The arithmetic and logical operations necessary for the execution of all instructions are performed by the arithmetic unit operating in conjunction with the central processor registers. Figure 3.1 gives a simplified block diagram of the CPU. Some of its registers are briefly described below.

- Arithmetic unit: Handles all Boolean functions and contains an 18-bit, 85 ns adder. It acts as the transfer path for inter-register transfers and shift operations.

- Instruction register: It accepts the six most significant bits of each instruction word fetched from memory, which include the operation code, indirect addressing, and indexing.

- Accumulator: This 18-bit register retains the result of arithmetic-logical operations for the interim between instructions. For all program-controlled input/output transfers, information is transferred between core memory and an external device through the accumulator. Its contents can be manipulated, and can be added to from memory.

- Link: This 1-bit register is used to extend the arithmetic capability of the accumulator. In 1's complement arith-

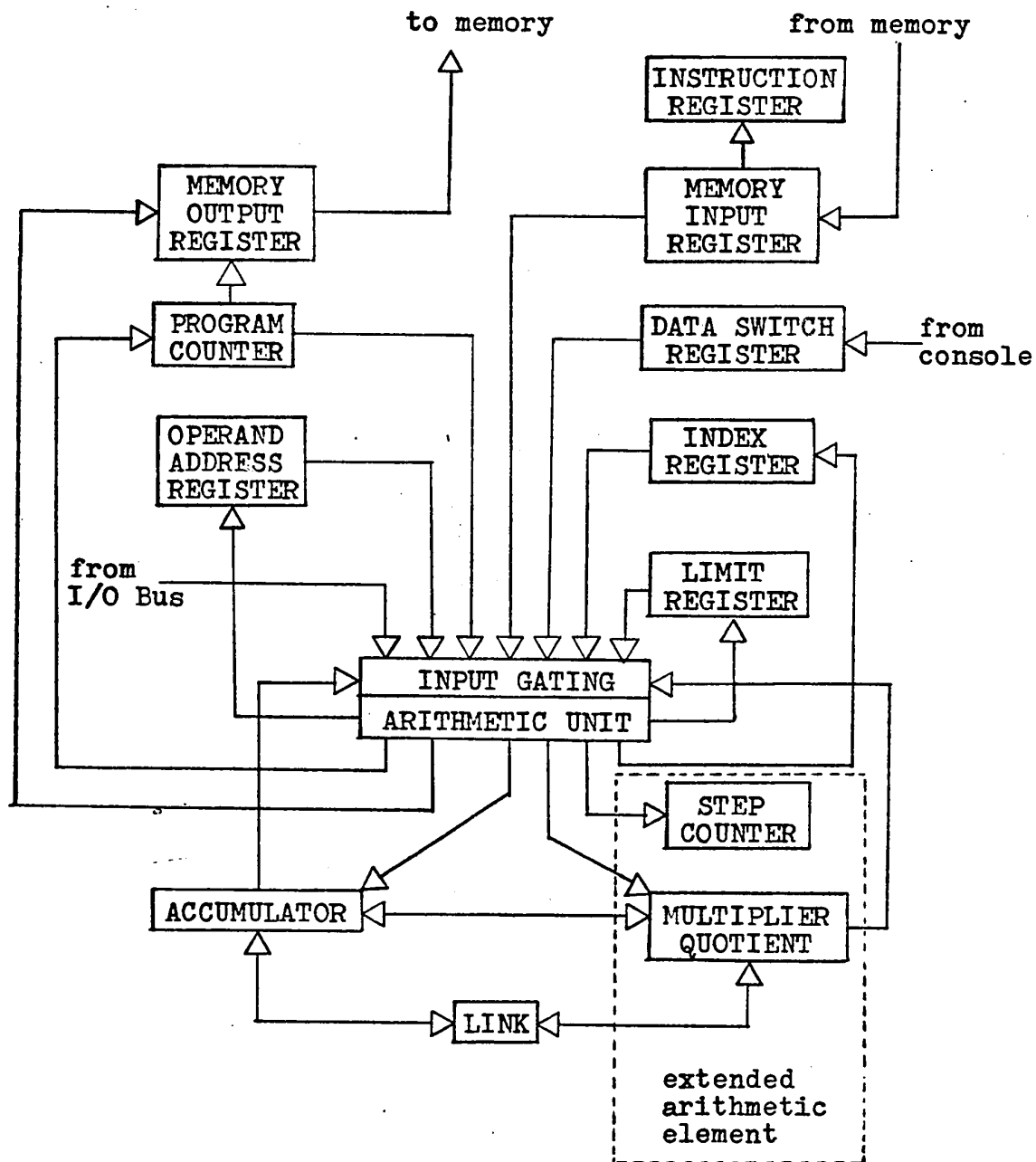


FIGURE 3.1: SIMPLIFIED BLOCK DIAGRAM OF PDP-15 CPU⁽⁶⁾

metic it is an overflow indicator, while in 2's complement arithmetic it logically extends the accumulator to 19 bits and functions as a carry register. The contents of the link can be checked and manipulated.

- Program counter: It determines the order in which instructions are performed. It contains the address from which the next instruction is to be taken.

3.1.2 Memory Organization

The primary storage facility for the PDP-15 is an 8192 word magnetic core memory. It provides rapid (400 ns) random access data instruction storage for both the CPU and the input/output processor. The basic subsystem of memory is the bank, which contains 2 pages of 4096 words each for a total of 8192 words per bank. Every bank contains a data buffer, an address buffer, and all the necessary read/write control and timing circuitry to make it an autonomous unit.

Memory communicates directly with the CPU and the input/output processor through the memory bus and the memory data buffer. A 13-bit memory address register allows direct addressing of 8192 words in each bank.

Reading from and writing into memory are performed by read and write memory half cycles requiring 400 ns each for an effective cycle time of 800 ns.

3.1.3 The Input/Output Processor

The I/O Processor contains 2 sub-units, the data channel controller and the addressable I/O bus. It contains the control

logic and registers necessary to transfer up to 18 bits of parallel data on a common bi-directional I/O bus. Data may be transferred directly between the I/O Processor and the accumulator of the CPU. While transfers are being made between memory and the I/O Processor the CPU is free to operate independently.

Peripheral devices may transfer data in any one of three modes: single-cycle block transfers, multi-cycle block transfers, and program-controlled transfers. The first two are implemented by the data channel controller, to which the TC15 Dectape controller is connected. The addressable I/O bus implements the program-controlled transfers. It also contains the program interrupt.

Program-controlled transfers, implemented by input/output transfer (IOT) instructions can move up to 18 bits of data between a selected device and the accumulator of the CPU. The microcoding of the IOT instruction includes the issuing of both a unique device selection code and the appropriate processor-generated input/output pulses to initiate a specific operation. For an "out" transfer the program reads a data word from memory into the accumulator. A subsequent IOT instruction places the data on the I/O bus, selects the device, and transfers the data to the device. For an "in" transfer the IOT instruction selects the device and transfers data into the accumulator.

The program interrupt (PI) facility, when enabled, allows the ready status of I/O device flags to automatically cause a program interrupt. At that time the contents of the

program counter and other information is stored in location 0, and the instruction in location 1 is then executed, transferring control to an I/O service routine for IOT instructions to identify and service the interrupt. When completed, the routine restores the system to the status prior to the interrupt via a single instruction, allowing the interrupted program segment to continue.

3.2 THE INTERFACE

Entirely designed and built by the Northern Electric Co. this device is connected to the PDP-15 I/O bus as an interface to up to seven separate testsets. Since the physical length of the I/O bus is limited by timing considerations, and since test stations are not necessarily or practically placed in close proximity to the computer, an interface becomes necessary. In this case the interface communicates with the test stations via balanced transmission lines independent of the timing requirements of the computer's I/O bus. All communication between computer and interface are IOT initiated. The interface decodes the IOT's and transmits a corresponding command to the test stations. Whereas all test stations see the instruction, only the actually addressed station will decode it and take action. Output from the test stations is requested via an IOT from the computer. The interface receives 18 status bits from the station, which are then strobed into the accumulator via the I/O bus.

The interface was designed to handle up to seven separate

test stations with a view towards future expansion in a time-sharing testing environment. The only hardware modifications necessary will be the addition of the Automatic Priority Interrupt option to the computer, to handle interrupts on a priority basis rather than via a skip-chain as is the case in the current program interrupt system. An additional 8K of memory will also be necessary. It is then possible to operate up to seven test stations concurrently, with a corresponding vast increase in tested PCB output. The test stations need not be of similar intent - a mix of logic and analog testsets is possible - as long as they correspond to the general hardware philosophy employed in the current system.

3.3 THE TEST STATION

The test station applies all programmed inputs to the PCB under test, and checks requested output levels. It contains all the necessary hardware to test a logic PCB. This includes programmable bias power supplies, stimuli and detectors for 86 PCB access pins and an additional 10 active testpoints (where stimuli may be applied) and 48 passive testpoints containing detectors only. The DC voltage levels of the stimuli are variable for both logic 0 and 1 so as to permit marginal testing of logic devices. Output levels are checked via programmable detector "windows".

The test station console contains switches to select desired testing and output data modes, and to permit manual modifications of stimuli application to the PCB under test.

It also displays the current input/output levels on the PCB, the latest command received via the interface, and the current status of either test station or PCB according to which status display has been pre-selected.

A separate connectable unit contains the test program verification hardware. It consists essentially of programmable relays through which IC access pins may be isolated from the rest of the circuit on the PCB, and grounded. Hence open circuits and shorts to ground may be simulated to check if the test program will pick them up.

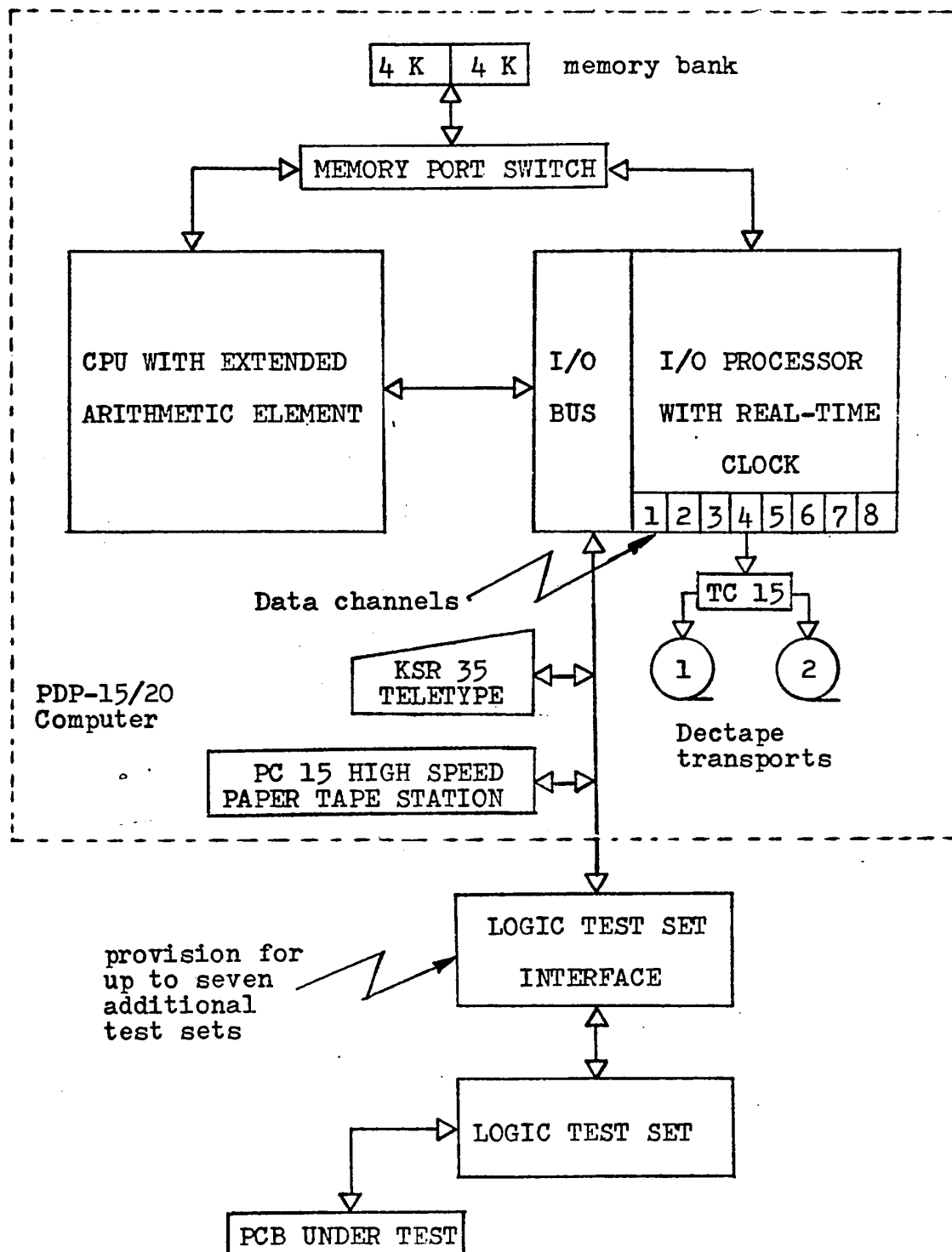


FIGURE 3.2: BLOCK DIAGRAM OF THE LOGIC TEST SYSTEM

CHAPTER 4

TEST SYSTEM SOFTWARE

The test system software is an integrated package of programs necessary for the efficient utilization of the logic test system. It consists of source test programs for the PCB's and a translator to convert them into Operating System input, and utility programs to perform storing, editing, swapping and debugging operations. Again, as in chapter 3 the following descriptions of the various programs are mainly intended to place the Operating System in context. All software, unless otherwise noted, was written by the Northern Electric Co.

4.1 THE SOURCE TEST PROGRAM

This consists of the encoded testing truthables and other necessary information in "English" statements. This type of coding permits easy interpretation of the source program listings and does not require skilled programmers for the encoding of the testing instructions. For an example of a source test program listing see chapter 7.

The source test program is listed and an 8-level ASCII paper tape is punched offline on a KSR-35 Teletype. The paper tape becomes the original input medium for either an editing or translating operation.

4.2 THE TRANSLATOR

The Operating System cannot decode the source test program statements directly. They must be translated (or compiled) into series of command bit-strings (18-bit words) which can be recognized and implemented by the various handlers of the Operating System. This is done by the translator program.

The translator accepts its source program input either directly from paper tape, or from a directory-oriented Dectape (Digital Equipment Corp. magnetic tape). It outputs the translated test program onto another directory-oriented Dectape. The translator includes diagnostics for test program encoding errors. These must be corrected before a translation is completed.

4.3 THE OPERATING SYSTEM

The Operating System exercises the test station according to the instructions in the translated test program. The various PCB testing and output data modes are under operator control and may be initiated either from the test system Teletype or directly from the test station control switches.

The Operating System loads the entire translated test program from Dectape into core following the area occupied by itself. The maximum size of a translated test program is roughly 5.5K words. Considering the encoding format of the

translator output, which will be detailed in the following chapter, this could be from 5K to upwards of 50K individual tests per test program.

4.4 THE UTILITY PROGRAMS

These comprise the various programs performing editing, debugging, storage, loading, listing, etc. functions.

A group of programs called the FACTS DECTRIEVE SYSTEM was written by the author to store source programs, assembled binary programs, and translated test programs on Dectape under a directory system. It can also swap programs from one tape to another, list them, punch them on paper tape, etc. Using this system all necessary software for the operation of the test system is stored on a single Dectape in the following order:

- (a) FACTS DECTRIEVE SYSTEM programs
- (b) Operating System
- (c) Translator
- (d) Translated test programs

By loading a bootstrap paper tape the FACTS DECTRIEVE SYSTEM keyboard monitor program is loaded from the Dectape. The Operating System or the translator may then be requested and loaded into memory. Control is then transferred to the loaded program.

Editing and debugging operations are performed using the Digital Equipment Corporation's Advanced Editor and Dynamic Debugging Technique programs. The Editor operates

on source programs only and is used to correct errors in the source test program before translation. Editing is done from Dectape to Dectape, although initial paper tape input is possible. The translator obtains its input from the same Dectape containing the edited source test program. Hence the entire editing-translating-testing process is mass-storage oriented with resulting advantages in speed and versatility.

The Dynamic Debugging Technique is a co-resident program in upper core that allows the operation of an assembled binary program in lower core to be examined and controlled via the debugging program's command repertoire. It was used to prove in the Operating System.

CHAPTER 5
OPERATING SYSTEM SOFTWARE

This chapter deals with the actual Operating System (hereafter abbreviated by OS) software. The OS is an assembled binary program. Its source program was written in the assembler language of the PDP-7,-9, -15 family of computers, using mnemonics. The source program was then assembled using the Digital Equipment Corp. Basic Assembler for the PDP-7,-9 series of computers. The resulting object program was stored on Dectape for subsequent loading into memory and execution.

The question will probably arise from those familiar with PDP-9,-15 machines as to why the program was written in basic assembler language rather than using the more sophisticated Macro language and Assembler of the PDP-15 Advanced Software System. The reason for this was the desire to minimize core requirements by writing our own I/O device handlers, rather than using the device handlers of the Advanced Software System. It was also desirable to retain absolute control of the disposition in core of the various parts of the program. Lastly, this OS is one of a family of more or less compatible operating systems controlling various digital and analog test stations. These will all eventually be incorporated in a time-shared test system which of necessity will have all its controlling software written in the basic assembler language for purposes of control.

There are several basic differences between the PDP-7, PDP-9, and PDP-15 computers. This OS includes several instruction types which may only be executed on a PDP-15. Otherwise the program could be run on a PDP-9, which was actually used to prove in much of the software. The program could be executed on a PDP-7 only if the machine has been hardware updated to a PDP-9 level, but again only if several aforementioned instruction types are changed.

It should also be mentioned that the PDP-15 must be in the PDP-9/15 (bank) mode which allows the direct addressing of 8192 words of memory.

5.1 OPERATING SYSTEM ORGANIZATION

The OS software can be divided into six sections for ease of analysis. These are:

- (a) OS initialization and start
- (b) Keyboard monitor and command repertoire
- (c) Elementary function decode and functional test routines
- (d) Program interrupt handler
- (e) Utility subroutines
- (f) OS tables, variables and elementary function storage

The following sections of this chapter will deal with these parts in detail.

The OS is normally stored on Dectape and is loaded into memory via the FACTS DECTRIEVE bootstrap paper tape and a keyboard command. It then occupies lower core as shown

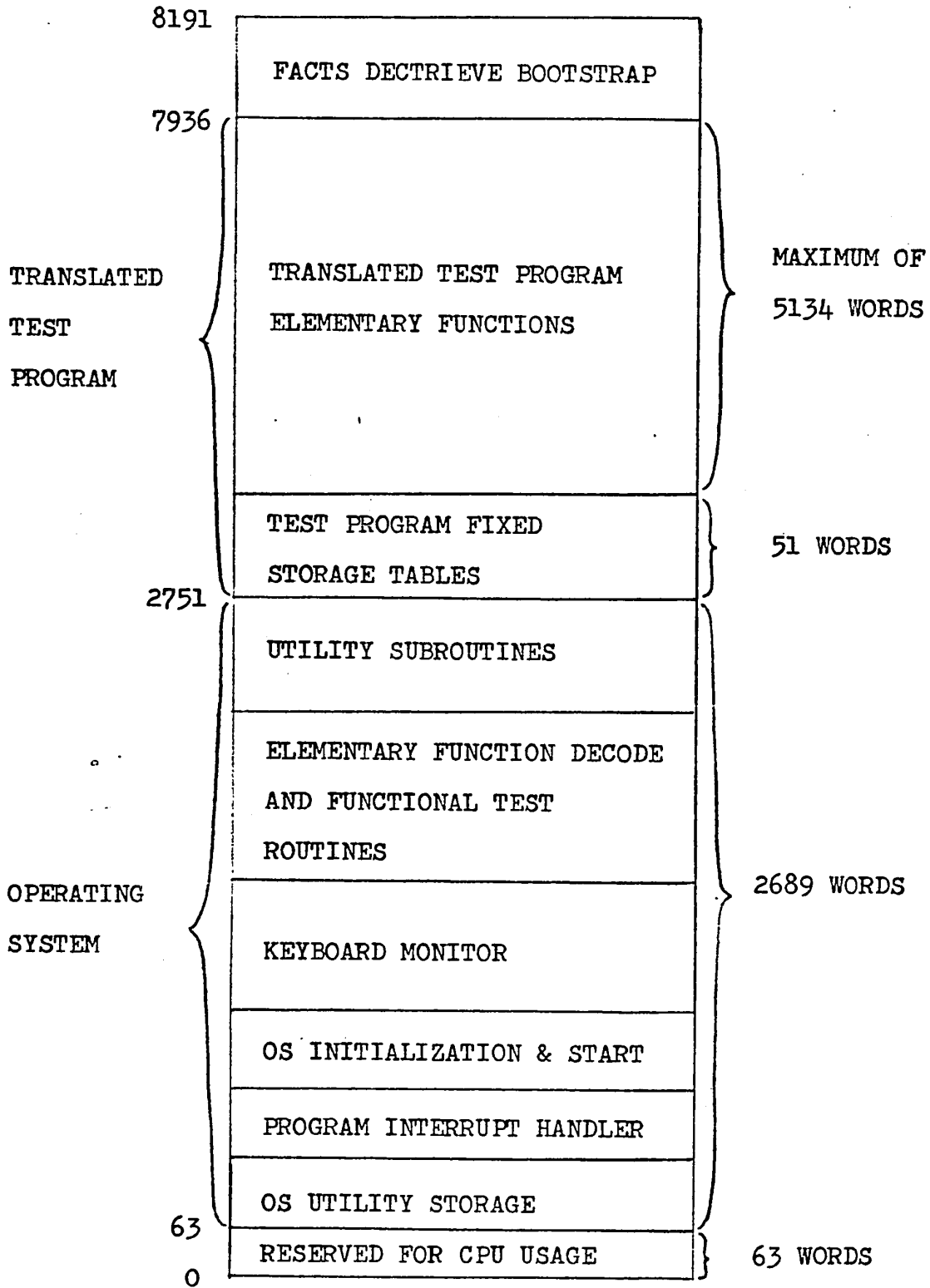


FIGURE 5.1: OPERATING SYSTEM CORE ALLOCATION

in Figure 5.1. Various registers are then reset to their initial state and the OS requests the name of a test program, which is subsequently loaded into memory. Test program execution occurs as specified by various commands given through the OS keyboard monitor. This execution involves the decoding of the test program's elementary functions via the functional test routines, sending appropriate instructions to the test station, and receiving output via the status lines.

I/O operations are usually performed through the program interrupt handler which functions in a skip-chain mode. Common subroutines are stored among the OS utility subroutines. Tables and variables which are modified by the OS during execution are stored at the beginning of the OS. Various tables of constants and fixed parameters of a test program are loaded into an area immediately following the OS, along with the test program's elementary functions.

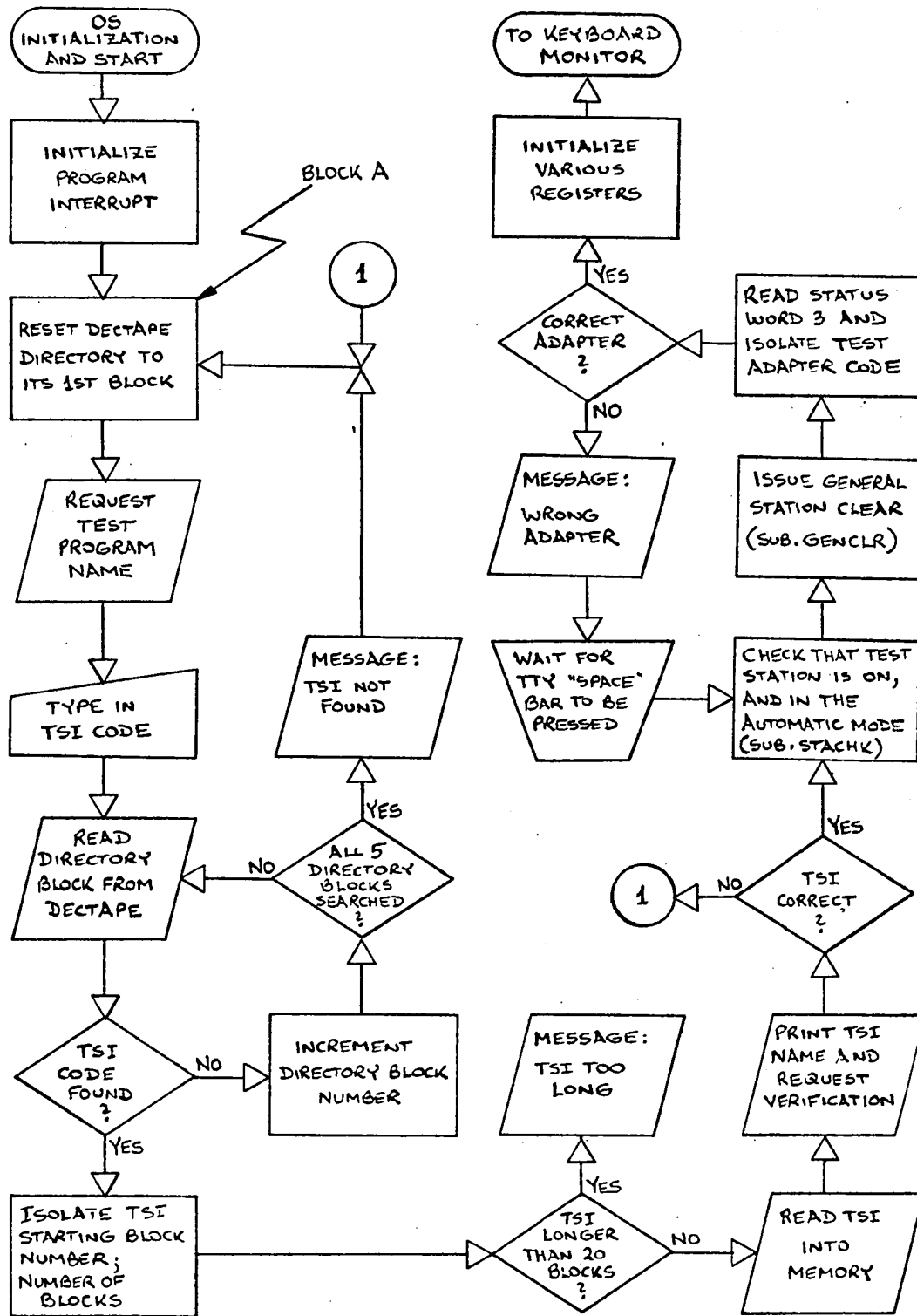
5.2 OPERATING SYSTEM INITIALIZATION AND START

OS execution commences at this program section. Its first action is to initialize the program interrupt handling to enable the interrupt skip chain. The Dectape directory is reset to its first directory block. There are a total of five directory blocks per Dectape, each block containing a maximum of 63 discrete program names, for a total of up to 315 programs per Dectape.

The OS now requests a test program (abbreviated by TSI

which stands for testset instructions) name, and waits for characters to be typed on the Teletype keyboard. The end-of-message is indicated by striking the space bar. Up to 8 characters may be entered as the TSI code. When entered, the code is checked against all directory entries and a diagnostic message is typed if no match occurs. Otherwise the TSI's tables and elementary functions are read into memory providing the total length of the TSI does not exceed 20 Dectape blocks (5120 words). Otherwise the permissible TSI storage of 5134 words for the elementary functions plus 51 words for the storage tables would be exceeded.

One of these storage tables contains the full TSI name in packed 6-bit ASCII. This is now typed to make sure that the correct TSI has been loaded. Upon verification via the Teletype keyboard a first transmission is initiated to the test station to check that it is turned on, and in the automatic mode of operation. If not, appropriate diagnostic messages are typed. A general clear is then issued to reset the station. A status word is next requested and read to check that the correct external adapter (if any) is plugged in. Various additional software registers are then initialized before the OS enters the keyboard monitor to wait for a testing initiation command.



FLOWCHART 5.1: OS INITIALIZATION & START

5.3 KEYBOARD MONITOR AND COMMAND REPERTOIRE

The entire testing process is under operator control and is performed according to his commands. These may be entered from either the Teletype keyboard or various push-buttons and switches on the test station console. The keyboard monitor section of the OS consists essentially of a wait-loop when commands are accepted, and corresponding decoding and action initiating routines.

This is one of the key sections of the OS. It was the intention of the author to provide a sufficiently powerful and versatile command library to allow rapid trouble shooting of faulty logic circuits. This was attempted by permitting the operator to control the extent and sequence of test program execution, and provide him with various types of printed output data. This will become more evident in the following sections.

The command repertoire may be divided into six classes of commands:

- (a) Data mode commands
- (b) Biasing mode command
- (c) Testing modes
- (d) Test initiation commands
- (e) Test program verification
- (f) Housekeeping commands

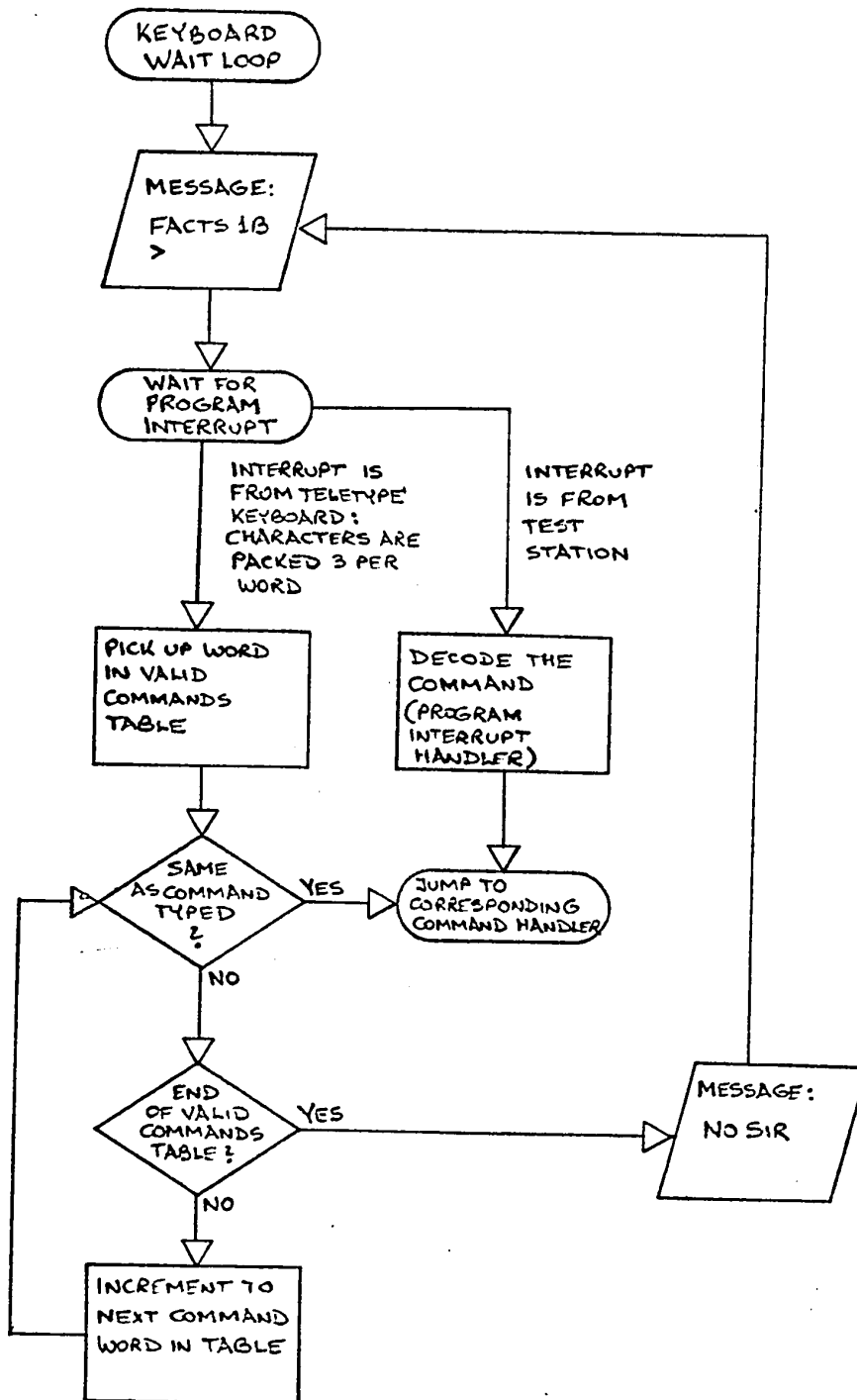
The first three types are preconditioners or qualifiers for the test initiation commands which specify the extent and sequence of test program execution. The test program verifi-

cation command is used to check on the ability of a new or modified TSI to pick up simulated faults on the PCB. The last type groups commands of miscellaneous intent, not included in the other classes.

5.3.1 Keyboard Wait Loop

Whenever a command is required to initiate or continue test program execution the OS enters the keyboard wait loop. This is indicated by the message FACTS 1B appearing on the teleprinter. The OS then waits for an interrupt to occur from either the Teletype keyboard or the test station. In the first case the typed characters are trimmed to 6-bit ASCII and packed 3 per word. The first word is then compared to the entries in a table of valid keyboard command character combinations. If a match occurs a "jump" is executed to the corresponding command handler. If the character sequence was invalid, this fact is indicated to the operator.

If the command was initiated from the test station it is decoded by the program interrupt handler which also transfers control to the appropriate command handler. When a test initiation command was given from the test station, the data mode is set according to the test station's data mode switch setting. Conversely, if such a command was given via the Teletype keyboard, the data mode must also be specified from the Teletype. Otherwise the last data mode used remains enabled. In other words, qualifying commands and test initiation commands must be issued from the same device - either all from the Teletype, or all from the test station.



FLOWCHART 5.2: KEYBOARD WAIT LOOP

This was done to avoid confusion as to which data mode or testing mode etc. is enabled for the current test run.

The operation of the keyboard wait loop can be summarized via Flowchart 5.2.

5.3.2 Data Mode Commands

These specify the format of the output data on the teleprinter. Four separate commands allow the operator to specify from a minimum of print-out in the case of the "No Data" command, where only the total number of test failures is printed, to a maximum in the case of the "All Data" command, where all input and output pins and associated levels on the PCB are printed for each test performed. The two intermediate stages comprise the "Reject Data" and the "Reject Coordinates" commands. The first results in the same output as the "All Data" command but only for those truth-table lines containing one or more failures. The second causes only those PCB access pins that had incorrect output levels to be printed, along with the corresponding truth-table numbers and truth-table line numbers.

The handlers for this type of command set a software switch and then return control to the keyboard wait loop to enable a test initiation command to be typed. The data mode remains unchanged until either another data mode command is recognized, or the reset command "Normal Run" is given.

5.3.3 Biasing Mode Command

This command allows the setting of a specific biasing mode before a subsequent test initiation command. A test program may specify up to five groups of biasing conditions. Each group consists of a setting for the +5 VDC PCB bias supply, the logic level 0 and 1 input voltage level specifications, and the voltage ranges (windows) for the output level detectors for logic levels 0 and 1. The first biasing group is called "Nominal Bias", and the other groups are denoted "Margin 1" to "Margin 4". The biasing mode command may specify any of these groups as long as they were included in the TSI. This command cannot create a new biasing group since this would involve translating operations in decoding voltage levels into wordfills for transmission to the test station.

This command may also be used to specify the input level mode for the current test. This refers to the logic level which is applied to those PCB input pins which are not exercised in a current truthtable. Normally, at the first pass of a test program, these pins are set low. If so specified in the TSI, a second pass will repeat all specified tests with these pins set high. This provides another level of assurance of picking up possible shorts between printed circuit paths.

The biasing mode command is intended as a trouble-shooting tool only in that it permits the partial execution of a test program - say only for that biasing condition where failures occurred. It is obviously of no use if only nominal

biasing was specified. Marginal biasing levels are intended to check if a circuit performs within the maximum and minimum voltage levels specified for the logic devices. They can also be used to detect marginally operating devices which cause intermittent failures under nominal biasing conditions.

This command may only be specified via the Teletype keyboard since it causes questions to be printed, the answers being the required bias and input level modes respectively.

5.3.4 Testing Modes

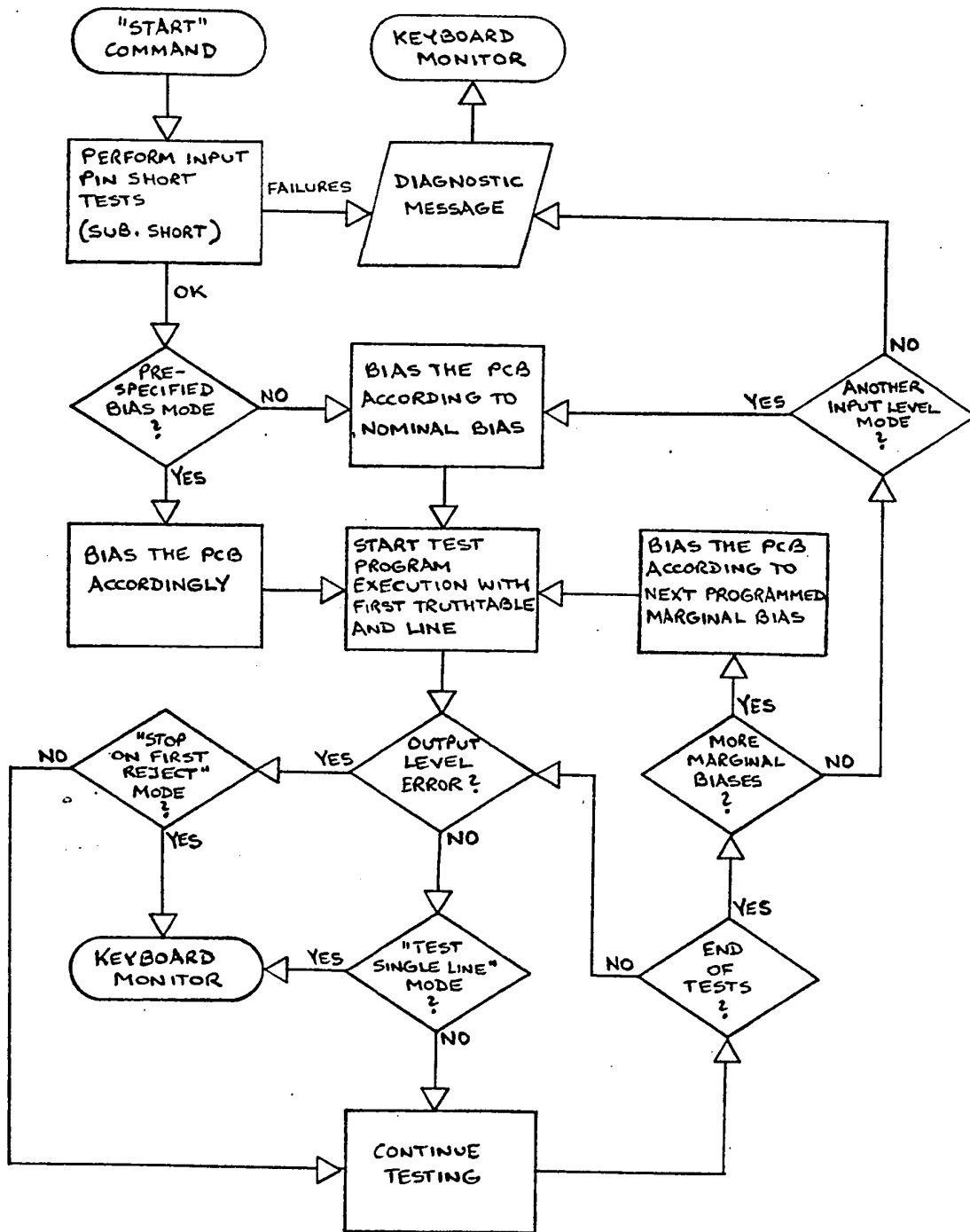
These specify the sequence and extent of test program execution upon a subsequent test initiation command. The "Stop First Reject" command causes testing to halt at the end of the next truthtable line containing one or more incorrect output levels. The "Test Single Line" instruction will result in an unconditional halt at the end of the next truthtable line. In both cases all current input levels remain applied to the PCB under test at the time of the halt. When these commands are given, specific software flags are set and the OS returns to the keyboard wait loop to await further commands. They are intended as troubleshooting tools. Once specified they remain in force unless cancelled either by resetting their respective toggle switches on the test station and initiating testing from there, or by specifying the "Normal Run" command on the Teletype. The latter also resets the data and biasing modes to "No Data" and nominal bias respectively.

5.3.5 Test Initiation Command "Start"

This command causes the execution of test program instructions to take place. It first performs an input pin short test on the PCB under test. This is explained in section 5.6.8. If there were no failures during the input pin short test, the PCB is biased according to either a preceding "Bias Mode" command, or according to the nominal bias mode specified in the TSI. The tests are then executed from the beginning of the test program, according to any pre-specified testing mode, with output data being printed according to any pre-specified data mode. Testing will normally continue until all truthtables have been executed once per programmed biasing mode with all unused inputs set low, and again once per biasing mode with all unused inputs set high, if so specified. It is thus possible to have the programmed truthtables executed ten times per PCB if all five biasing modes and both input level modes were specified in the TSI.

If the "Start" command was given via a push-button on the test station, an interrupt is generated which, when decoded, will cause data and testing modes to be assumed according to switch settings at the test station.

Flowchart 5.3 indicates test program execution under the "Start" command. It does not include data modes since these govern print-out only and do not influence the testing sequence.



FLOWCHART 5.3: TESTING UNDER THE "START" COMMAND

5.3.6 Test Initiation Command "Continue"

As the name implies this is intended to continue test program execution after a valid intermediate halt. Valid intermediate halts are those which are either programmed in the TSI, such as a manual operation request (see section 5.4.3), those which occur due to preset testing modes, or a halt caused by the operator requesting an interrupt via the Teletype keyboard (typing a semi-colon) or the test station push-button. In such cases a return address is generated by the OS to allow subsequent program continuation.

The "Continue" command may be given from either the Teletype or the test station. When issued, the test station is first interrogated to check that it is turned on, whether this command originated there, and, if so, its switches are scanned for testing and data modes. The command handler then checks if a return address has been set, and control is transferred to that address. Otherwise a diagnostic message is printed and control is returned to the keyboard monitor.

5.3.7 "Partial Test" and "Looping"

These testing initiation commands are trouble-shooting tools only. They allow a specified section of a test program to be executed, singly in the case of "Partial Test", and continuously in the case of "Looping". Since the operator must specify the boundaries of the sectional test in terms of truthtable and truthtable line numbers, as well as the

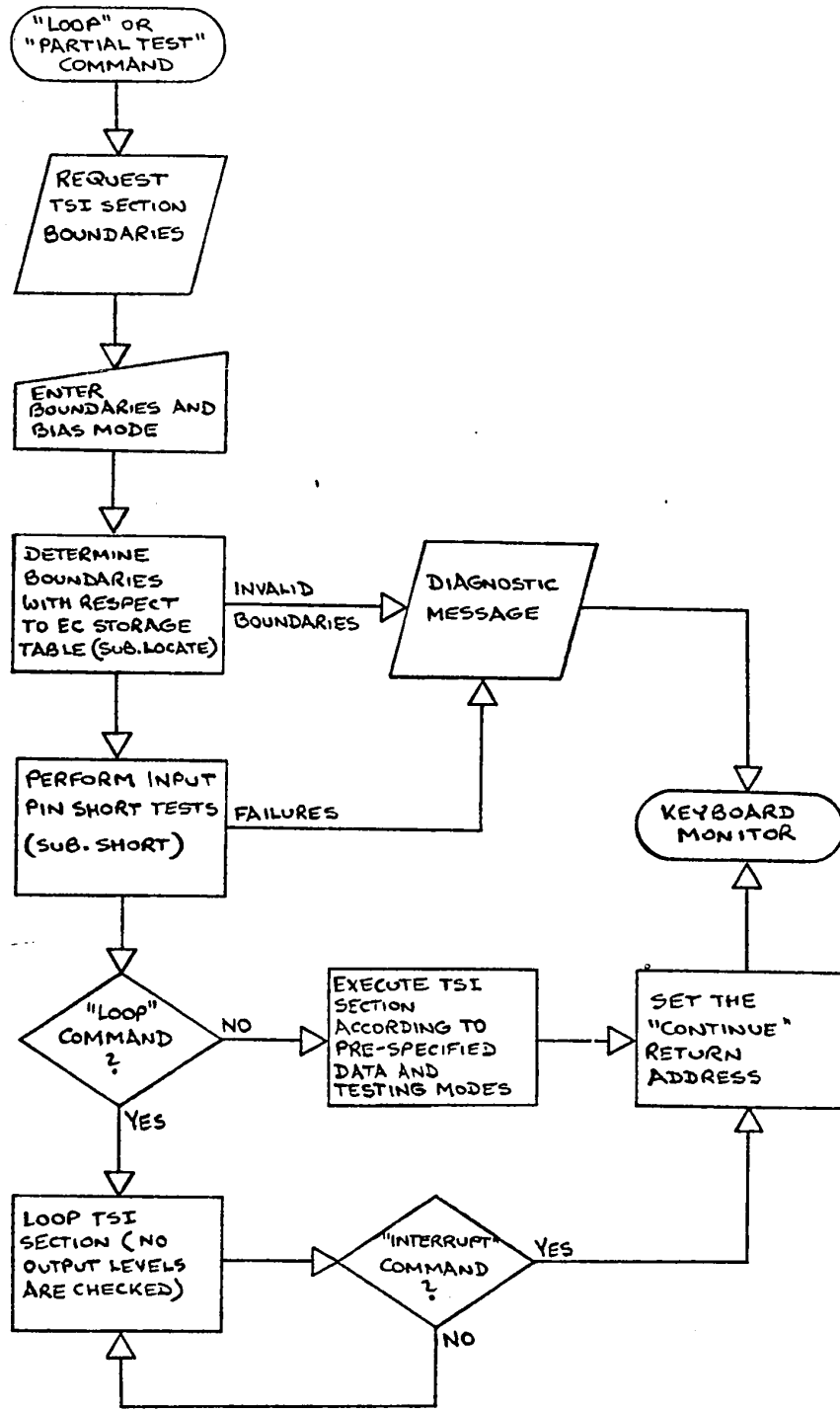
desired biasing mode, these commands can only be entered via the Teletype.

In both cases the handler first tries to locate the boundaries of the specified section in the elemental commands storage. If any of the specified table or line numbers cannot be found an appropriate diagnostic message is printed. Otherwise test execution commences with an input pin short test, followed by the test program section.

For "Partial Test" any of the available data and testing modes may be pre-specified. Testing normally halts at the end of the specified section. Test program execution can then be completed via the "Continue" command.

The "Looping" command causes the execution of the specified test program section in a continuous loop. Outputs are not checked, so that testing and data modes now have no meaning. This is intended to allow waveform monitoring with an oscilloscope by the rapid repetition (at computer speed) of programmed input level changes. The loop can only be stopped with the "Interrupt" command, at which time the keyboard monitor is re-entered. The loop can be restarted via a subsequent "Continue" command.

Flowchart 5.4 shows the testing sequence under the "Loop" and "Partial Test" commands. The action of the testing modes during "Partial Test" is identical to that shown for the "Start" command.



FLOWCHART 5.4: TESTING UNDER "LOOP" OR "PARTIAL TEST"

5.3.8 "Repeat Current Line" and "Repeat Sequence"

These further two test initiation commands are useful in re-testing after having performed some manual operation on the PCB - such as the replacing of a faulty device. The "Repeat" command will repeat the current truthable line's output level checks. Testing will then continue according to the preset testing modes. The "Repeat" command does not change any input conditions.

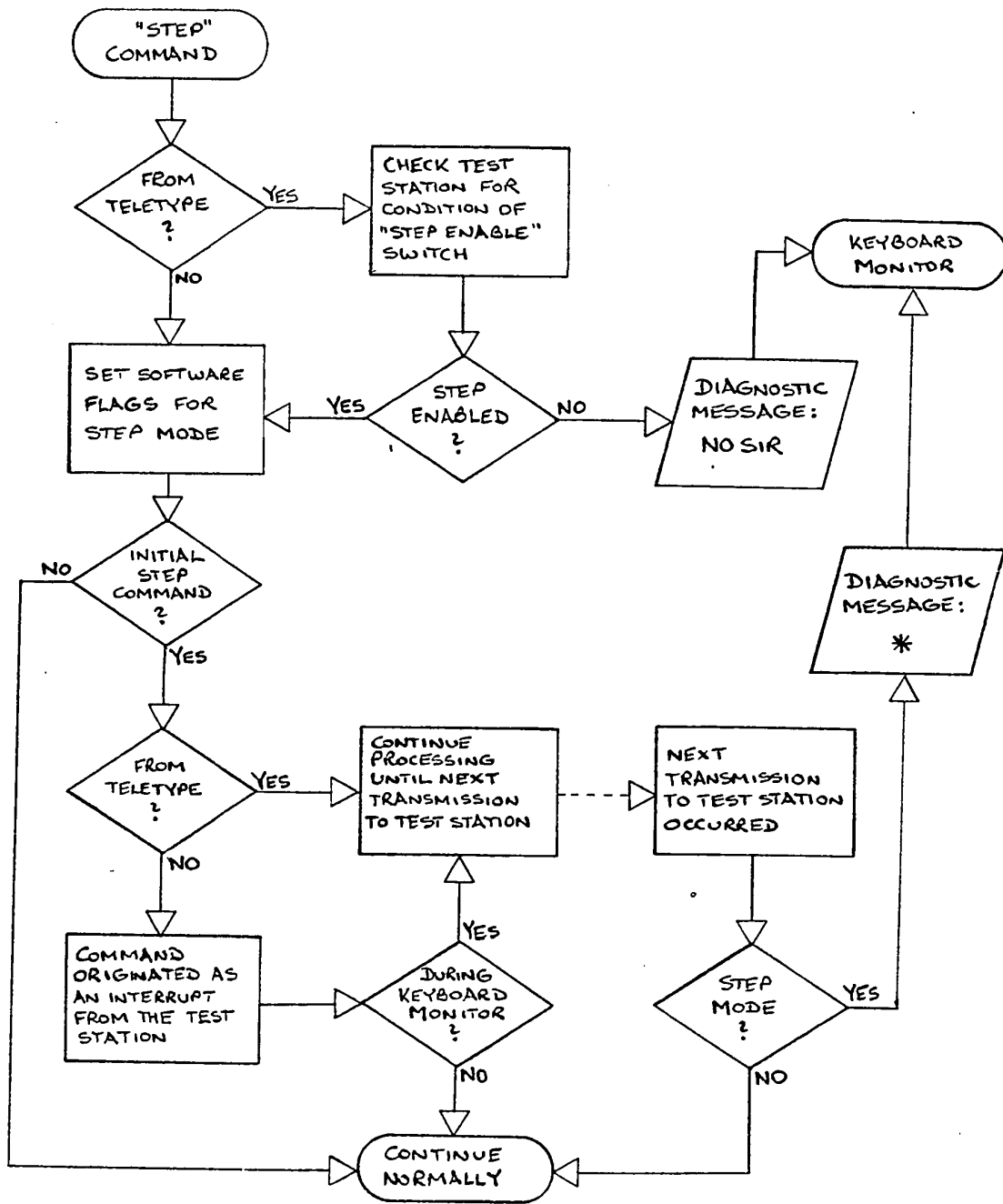
The "Repeat Sequence" command restarts a pre-defined "Partial Test" within the same pre-specified boundaries. Again, testing occurs according to pre-specified testing modes.

Both these commands may be implemented from either the Teletype or the test station.

5.3.9 The Step Mode

Under this mode the OS halts after each complete transmission to the test station (except for those transmissions which are necessary for step mode implementation). This allows a check on OS software operation by examining the sequence of communications with the testset. It is also useful in hardware maintenance by placing the speed of hardware updating via OS commands under the control of the operator.

The "Step" command may be implemented either from the Teletype or the test station, but in either case a switch must be set on the testset to enable this mode. This is a precautionary measure as well as a means of disabling the step mode to continue normal processing.



FLOWCHART 5.5: STEP MODE

5.3.10 Test Program Verification

Since testing truth tables are generated manually through the analysis of logic circuits, it is very likely that individual logic devices may not be actually tested. This is due to the complexity of the logic circuits and the fact that only a relatively small number of possible input level combinations are exercised. Otherwise a PCB with only ten inputs would require 10^{24} individual tests. The actual PCB's tested generally have many more input pins.

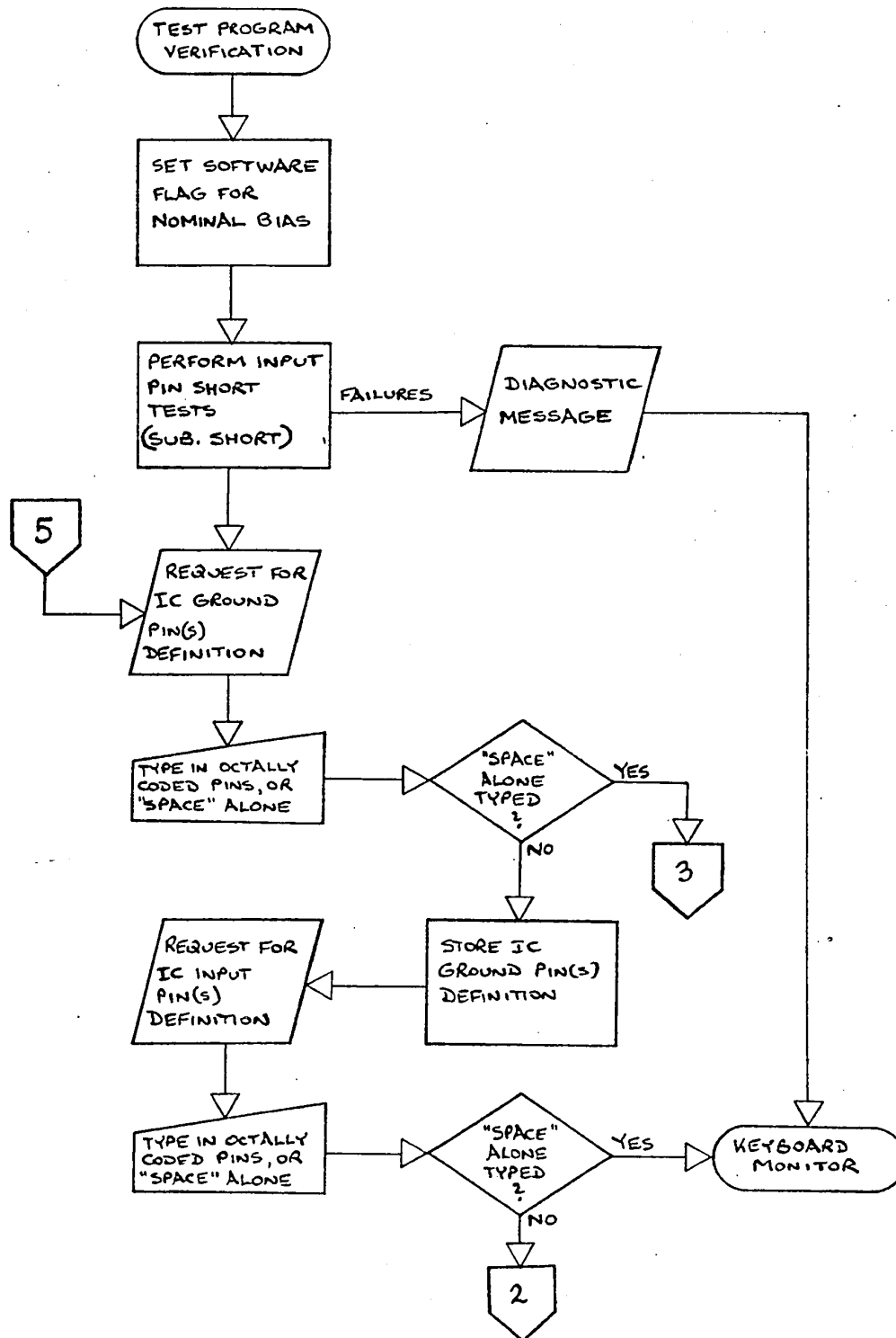
Test Program Verification simulates faults on each IC access pin. If the entire test program is then executed, and no error is detected, the test program is hence incomplete in that it does obviously not exercise that particular IC pin. The necessary hardware consists of a copy of the PCB having all its IC's mounted removably in sockets, and a set of programmable relays which isolate the pins of an IC socket on a testset fixture from corresponding pins on a cable-mounted connector. This connector is inserted into a vacated IC socket on the PCB, whose corresponding IC is inserted into the socket on the fixture. By exercising the aforementioned relays open-circuits and shorts to ground can be simulated on each IC input and output pin. Faults are introduced one IC pin at a time, with the other pins remaining connected to the circuit on the PCB via the cable-mounted connector.

The software implementation of the "Test Program Verification" command is as follows:

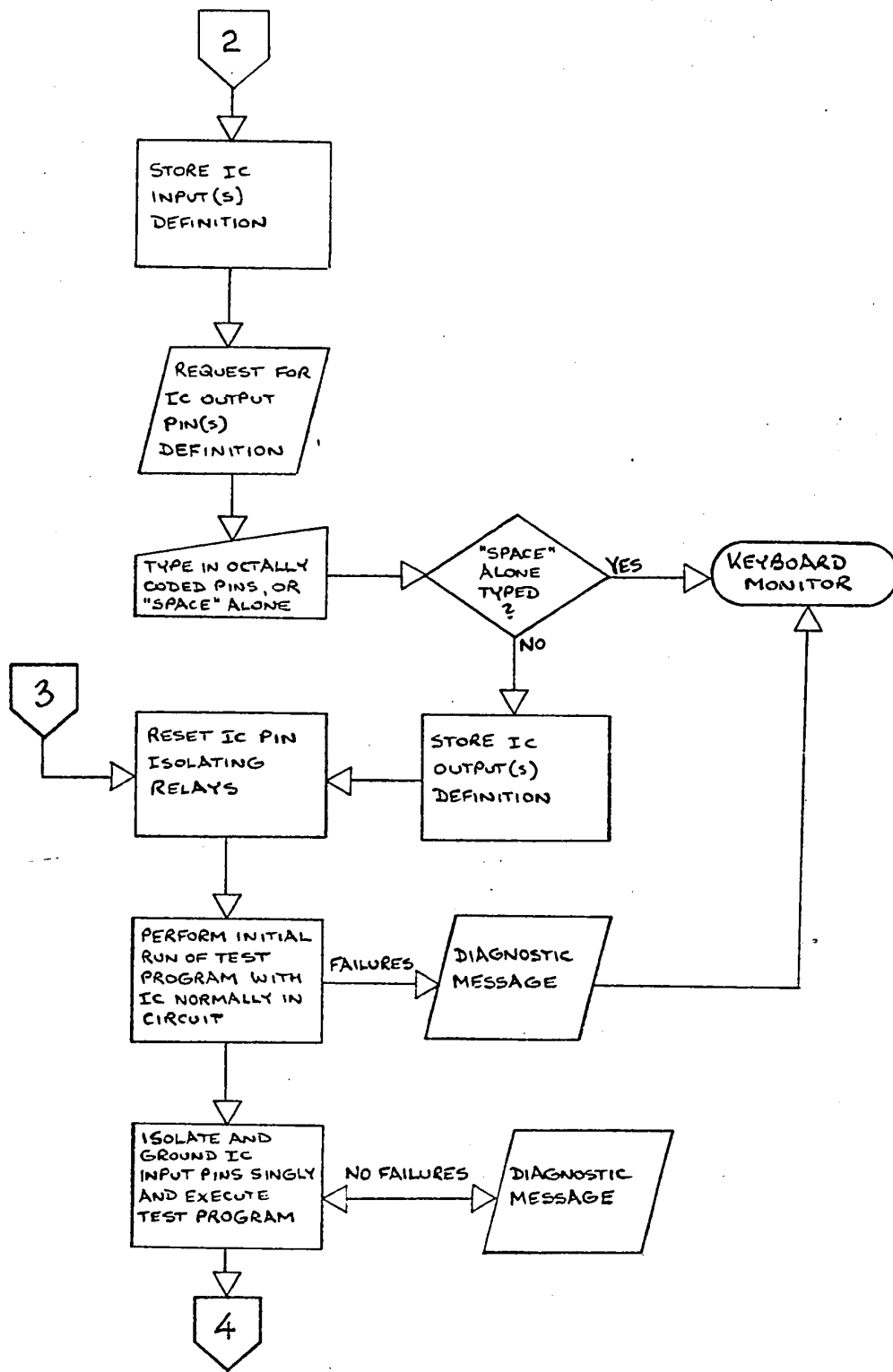
After an IC has been selected and transplanted, its

ground pin number, input and output pin numbers are defined via the Teletype. The testprogram is then executed with all fixture relays in their normal state such that the IC appears in the circuit as normal. This is to prevent misleading failures due to poor contacts. For this and subsequent runs of the test program only the nominal biasing mode is assumed, but both input level modes are used, if so specified in the TSI. After the successful completion of the first run, the first IC input pin is isolated from the PCB, and grounded. The entire test program is executed. If there were no failures a diagnostic is printed, identifying the current IC pin. In any case, this input pin is reconnected to the PCB, and the process is repeated for all subsequent IC input pins. The output pins are handled next, but are only isolated from the PCB, and not grounded, as otherwise certain types of IC's could be destroyed.

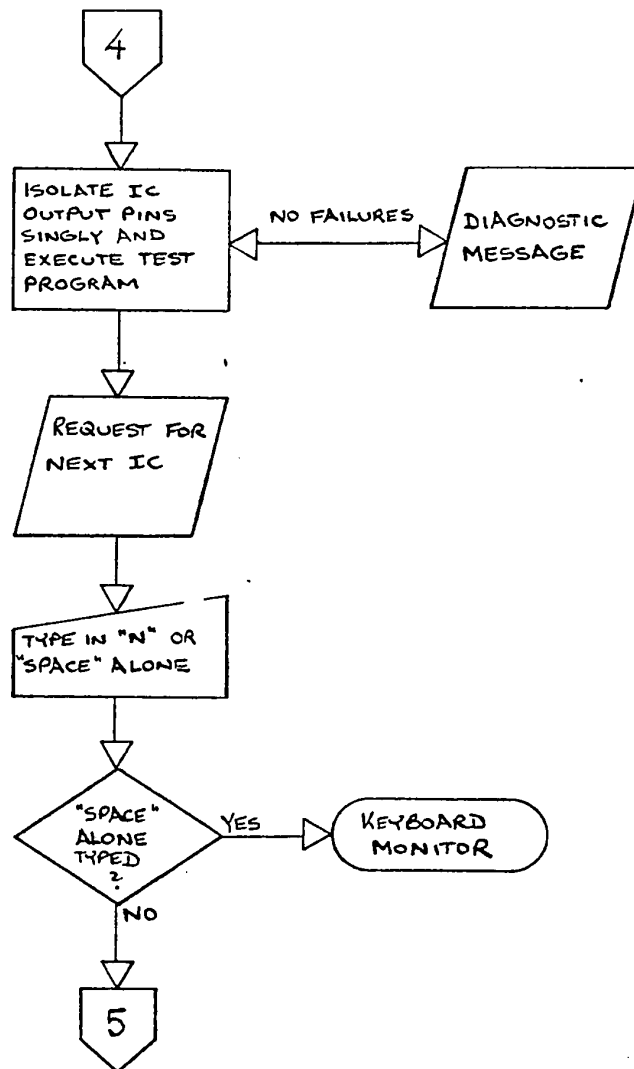
With the diagnostic output from the Test Program Verification the programmer can then add sufficient truth tables to test the previously missed devices. It is of course possible in certain circuit configurations that "don't care" conditions surround a device such that no externally applied input level configurations will change its output level. In such cases this command must be used judiciously to avoid needless frustration in trying to create impossible test setups. Flowchart 5.6 shows the logic of this command.



FLOWCHART 5.6: TEST PROGRAM VERIFICATION (page 1 of 3)



FLOWCHART 5.6: TEST PROGRAM VERIFICATION (page 2 of 3)



FLOWCHART 5.6: TEST PROGRAM VERIFICATION (page 3 of 3)

5.3.11 Housekeeping Commands

These miscellaneous commands serve to load a new test program in the case of the "New" instruction, clear all operated conditions on the test station in the case of "General Release", and return to the FACTS DECTRIEVE bootstrap. All of these commands can only be entered from the Teletype.

The "New" command transfers control to the OS initialization and start routine (section 5.2), specifically to block A in Flowchart 5.1.

The "General Release" command clears the test station by resetting all logic, relays, power supplies, etc., and then returns control to the OS keyboard monitor.

The "Return to FACTS DECTRIEVE" command transfers control to the starting address 7936 of the FACTS DECTRIEVE bootstrap (see Figure 5.1), which in turn enables the usage of the FACTS DECTRIEVE system.

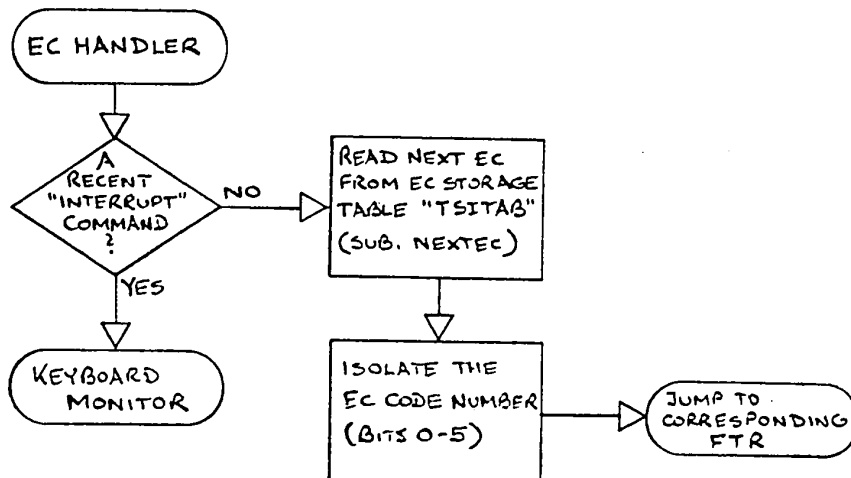
5.4 ELEMENTARY FUNCTIONS AND FUNCTIONAL TEST ROUTINES

The translator output of a test program's truthtables is in the form of 18-bit words which, when decoded by the OS, result in manipulation of the test station in order to test a logic PCB. There are eight discrete types of these words, which are called elementary functions or commands. The OS decodes them with eight corresponding functional test routines.

These elementary commands may be considered to be the basic building blocks of the testing system's performance repertoire insofar as they define exclusively all possible functions that the test system may be required to perform by the TSI.

The following sections will define the eight elementary functions. It should be noted that the reason for their code numbers not being in numerical order from 1 to 8 is due to other existing test stations having already used certain code numbers. To avoid decoding ambiguities in a future time-shared system, the code numbers for this OS had to be unique.

The correct functional test routine (FTR) is selected by a dispatch routine denoted the EC (elementary command) handler, which operates as shown in Flowchart 5.7.



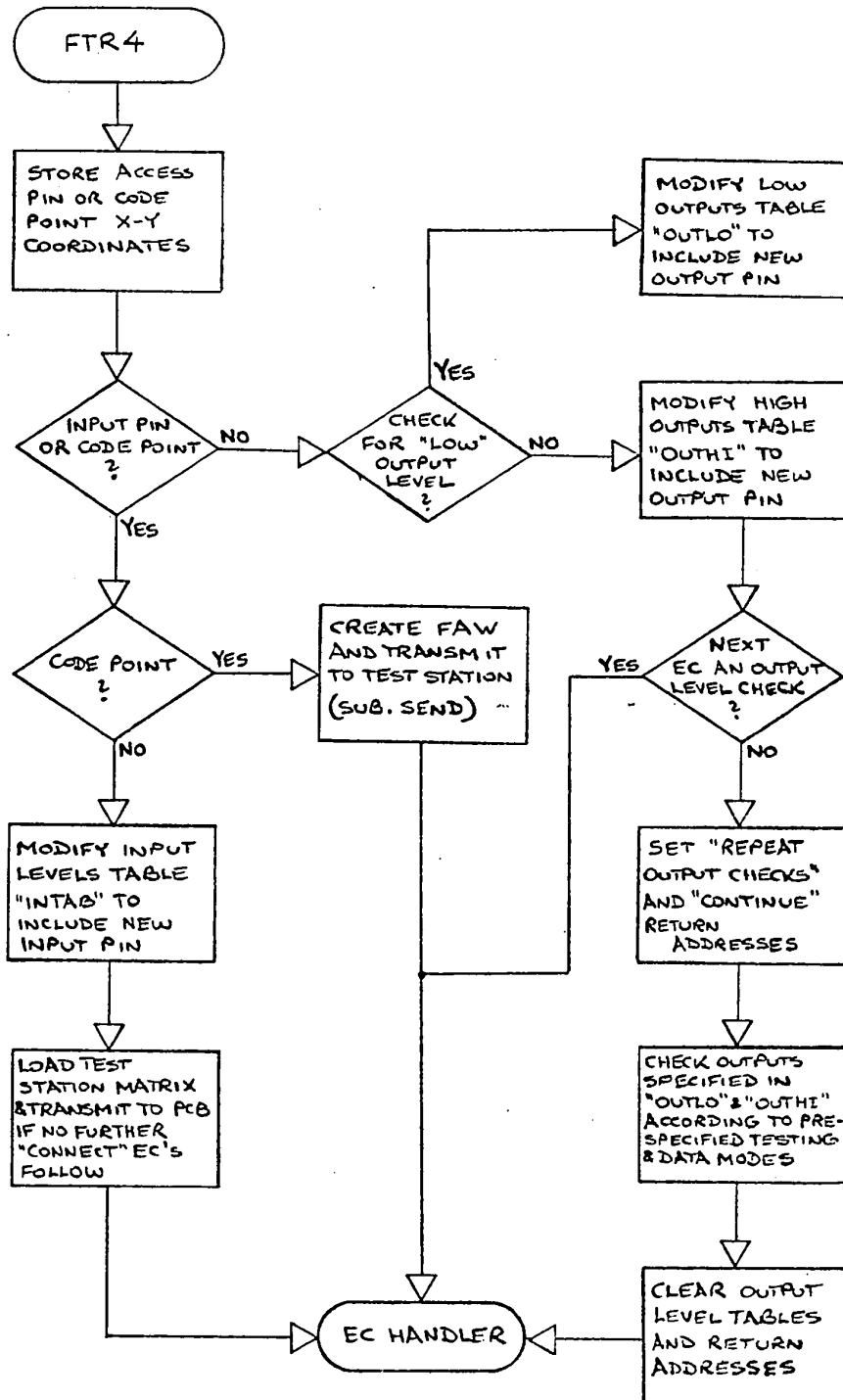
FLOWCHART 5.7: EC HANDLER

5.4.1 FTR 4: Full Address Word

Instructions from the OS to the test station may be transmitted in two separate formats. One of these is the Full Address Word (FAW) which uniquely defines the X and Y coordinates of an individual code point in the test station. Every action that can be performed by the test station is controlled through one or more of these code points. The other format is that of the Partial Address Word (PAW) and accompanying Wordfill, which will be discussed in section 5.4.2.

The EC 4 is created by the translator whenever it is desired to change the level of a single PCB input pin, to check the output level of a single PCB output, or to exercise a single code point in the test station, say to modify a power supply setting. The FTR 4 then creates a corresponding FAW for transmission to the testset. The individual bit assignments for both the EC 4 and a FAW are shown in Figure 5.2.

The actions of the FTR 4 are depicted in Flowchart 5.8. The following explanation will aid in understanding it. The differentiation between PCB input pins or ordinary test station code points is made from the fact that input pins must fall within Y-coordinate levels 0 to 5. Each Y-level has 16 X-coordinates. This results in a total of 96 random accessible input pins. Since a logic PCB has 86 access terminals, the additional 10 points are utilized as active test-points which may be externally connected to the PCB.



FLOWCHART 5.8: FTR 4 (FULL ADDRESS WORD)

The test station incorporates a memory matrix which stores the current PCB input levels until they are simultaneously strobed onto the PCB. This is necessary to avoid race conditions which could occur if input level changes arrive in a random order. To update this memory the OS employs a table called the input levels table "INTAB" which is shown in Figure 5.3. Whenever the table is completely updated, say at the end of all input level changes for a truth table line, the memory matrix is loaded according to the table and is then strobed onto the PCB.

PCB output levels are read back into the computer from the test station via 18-bit status words. The specific status word desired is pre-selected by transmitting a FAW for a corresponding code point. It is then read into the accumulator with an IOT instruction. 16 bits per output level status word correspond to 16 pins or test points. There are 9 of these status words for a total of 144 output level detectors corresponding to 86 PCB access terminals, 10 active and 48 passive testpoints. The output pins or testpoints which are to be checked are stored by the OS in two tables - one for low output levels, the other for high output levels - called "OUTLO" and "OUTH1" respectively. These are also shown in Figure 5.3. The output status word selection is then determined by the contents of these tables.

Input Levels Table "INTAB"

	BIT	0	1	2	14	15	16	17
WORD	1	A1	A2	A3	A15	A16		
	2	A17	A32		
	3	A33	A48		
	4	B1	B2	B16		
	5	B17	B32		
	6	B33	B48		

A1 - A43 = PCB component side access pins

B1 - B43 = PCB wiring side access pins

A44 - A48, B44 - B48 = active test points

Any bit = 0: Corresponding access pin or test point = 0

= 1: Corresponding access pin or test point = 1

Output Levels Tables "OUTLO" and "OUTH1"

	BIT	0	1	2	14	15	16	17
WORD	1	A1	A2	A3	A15	A16		
	2	A17	A32		
	3	A33	A48		
	4	B1	B2	B16		
	5	B17	B32		
	6	B33	B48		
	7	T1	T2	T16		
	8	T17	T32		
	9	T33	T48		

A1 - B48 are defined as for table "INTAB"

T1 - T48 = passive test points (detectors only)

Any bit = 0: Do not check the corresponding pin or
test point.

Any bit = 1: Table "OUTLO": check corresponding pin or
test point for a logic 0

Table "OUTH1": check corresponding pin or
test point for a logic 1

FIGURE 5.3: INPUT AND OUTPUT LEVELS TABLES

5.4.2 FTR 5: Partial Address Word

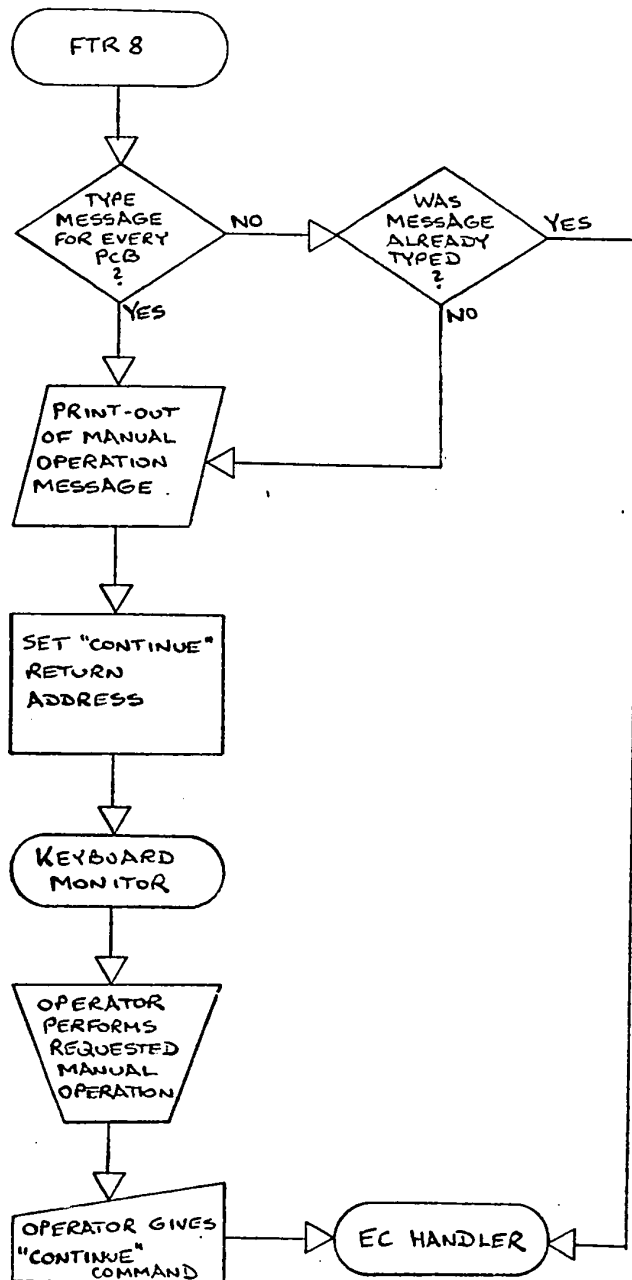
The PAW defines the Y-coordinate level, and the accompanying Wordfill defines all X-coordinates for this level. Each Y-level consists of 16 X-coordinates. Hence, if for example it is desired to set five input pins high, and all these pins are enabled via code points on the same Y-level, the transmission to the test station of a PAW for this level, along with a wordfill defining the five X's will accomplish this.

The translator creates an EC 5 plus a wordfill whenever several code points for a particular Y-level are to be exercised. The FTR 5 creates the corresponding PAW, and transmits it along with the Wordfill to the station. Its actions in all other respects are identical to that of the FTR 4 (see Flowchart 5.8). The bit assignments for the EC 5, the PAW and Wordfill are shown in Figure 5.4.

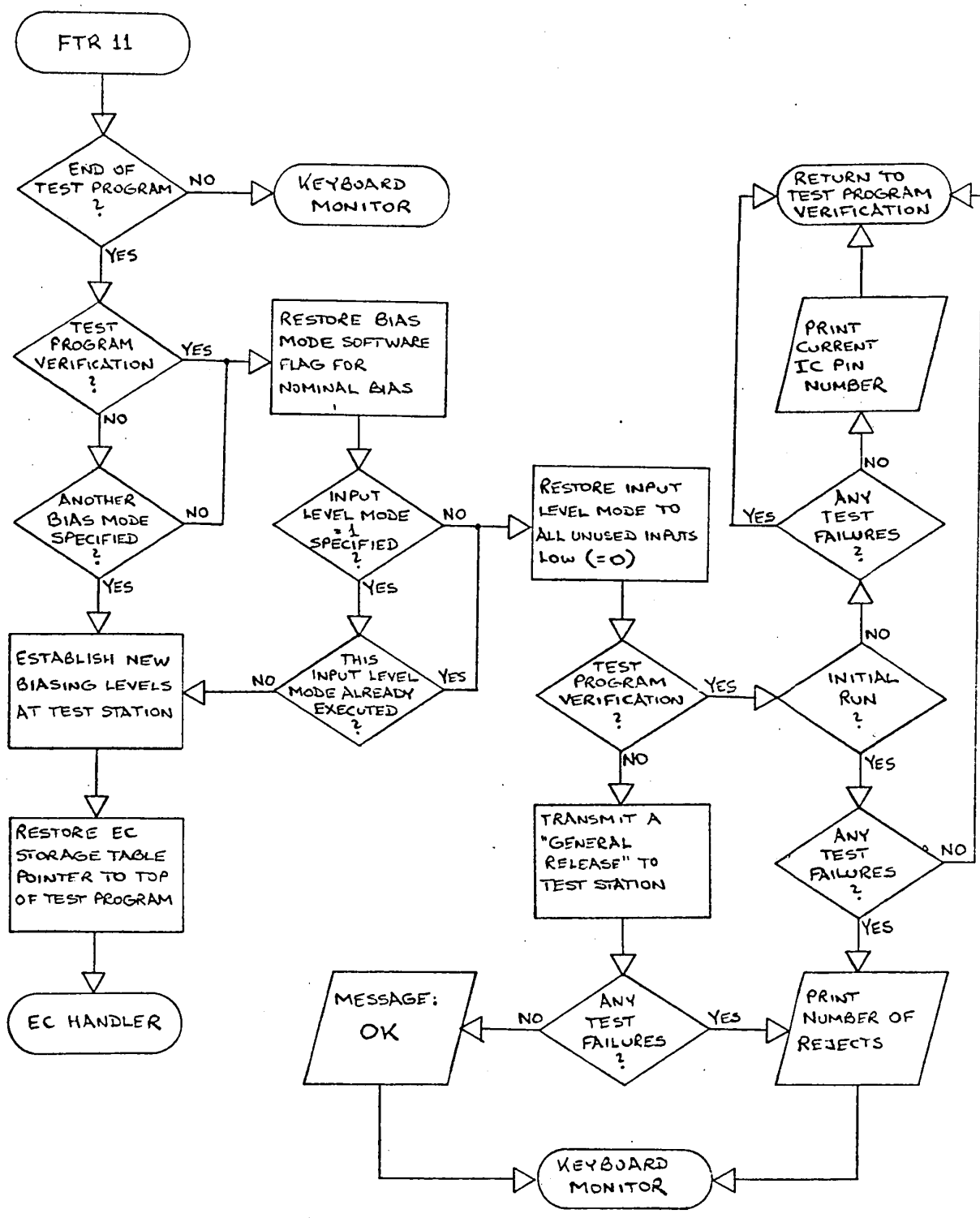
5.4.3 FTR 8: Manual Operation

Manual operations imply certain actions required by the operator at pre-specified times during a test. An example would be the connecting of sensing wires to testpoints on the PCB. The message to be printed is specified in the TSI. The translator generates an identifying EC 8 along with wordfills containing the message packed as 3 6-bit ASCII characters per word. This is shown in Figure 5.5.

The FTR 8, in decoding the EC 8, decides whether or not to type the message, turns on appropriate indicator lights



FLOWCHART 5.9: FTR 8 (MANUAL OPERATION)



FLOWCHART 5.10: FTR 11 (GENERAL RELEASE; END)

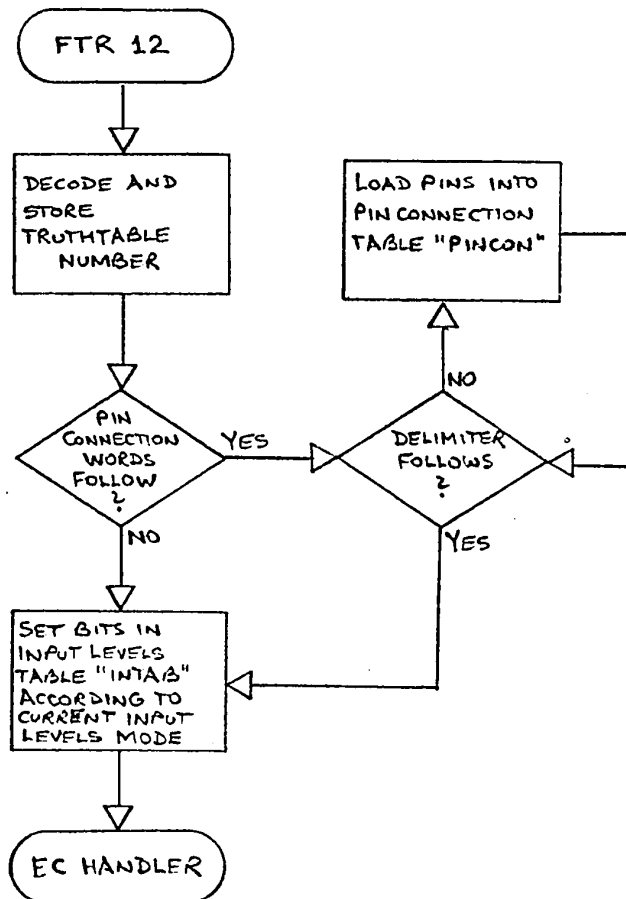
Flowchart 5.10 shows the performance of the FTR. It can be seen that the entire test program is executed once for each programmed bias condition per input level mode. The FTR also causes the total number of failures to be printed. In the case of test program verification the FTR disables additional biasing modes, and identifies the IC pin where an error has been simulated without being picked up by the test program.

5.4.6 FTR 12: Pin Connection Table; Truthable Number

EC 12 serves the dual function of indicating the start of a new testing truthable, and heading a series of words identifying PCB input/output pins and testpoints corresponding to bit positions in octal-coded truthable lines. The latter requires some further explanation:

It may be recalled that EC's 3 and 4 are used to set or check PCB pins and testpoints singly or in groups per Y-level. An alternate format is made available through octal coding. This means that the pin and testpoint conditions of a truthable line are treated as binary bits which, taken orderly in groups of three from left (inputs) to right (outputs) result in a series of octal digits. The pin connection table following the EC 12 serves to correlate input/output pins and testpoints with the octally grouped binary bits. Octal coding of truthable lines vastly reduces the size of a translated test program. The EC 13 described in the next section defines the octal format.

The FTR 12 decodes this EC whose bit assignments are shown in Figure 5.8. It stores the pins, which are packed two per word following the EC, in a table "PINCON" which is also shown in Figure 5.8. Since the EC always defines the start of a new truthtable, the FTR also resets the input levels table "INTAB" according to the current input levels mode to all inputs either low or high. Flowchart 5.11 indicates the actions of this FTR.

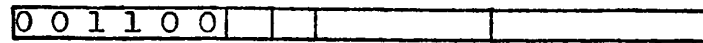


FLOWCHART 5.11: FTR 12 (PIN CONNECTION TABLE;

TRUTHTABLE NUMBER

EC 12 Bit Assignments

BIT 0 5 7 12 17



Bit: 0-5: EC code = 12

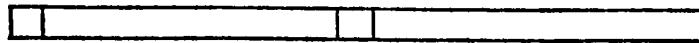
7: 0 = Pin connection table follows

1 = Truthtable number only

12-17: Truthtable number

Pin Connection Definition Word

BIT 0 1 8 9 10 17



Bit: 0,9: 0 = Input, 1 = Output

1-8,10-17: Pin or testpoint number:

0-42 = PCB pins A1 - A43

43-47 = active testpoints A44 - A48

48-90 = PCB pins B1 - B43

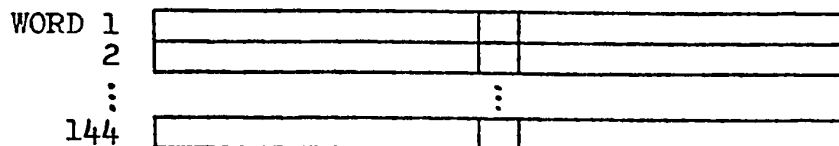
91-95 = active testpoints B44 - B48

96-143 = passive testpoints T1 - T48

Delimiter = 1's in bits 0 - 17

Pin Connection Table PINCON

BIT 0 8 9 10 17



Bit: 9: 0 = Input, 1 = Output

10-17: Pin or testpoint number

FIGURE 5.8: EC 12, PIN CONNECTION TABLE AND DEFINITION WORD

5.4.7 FTR 13: Octal Format; Truthtable Line Number

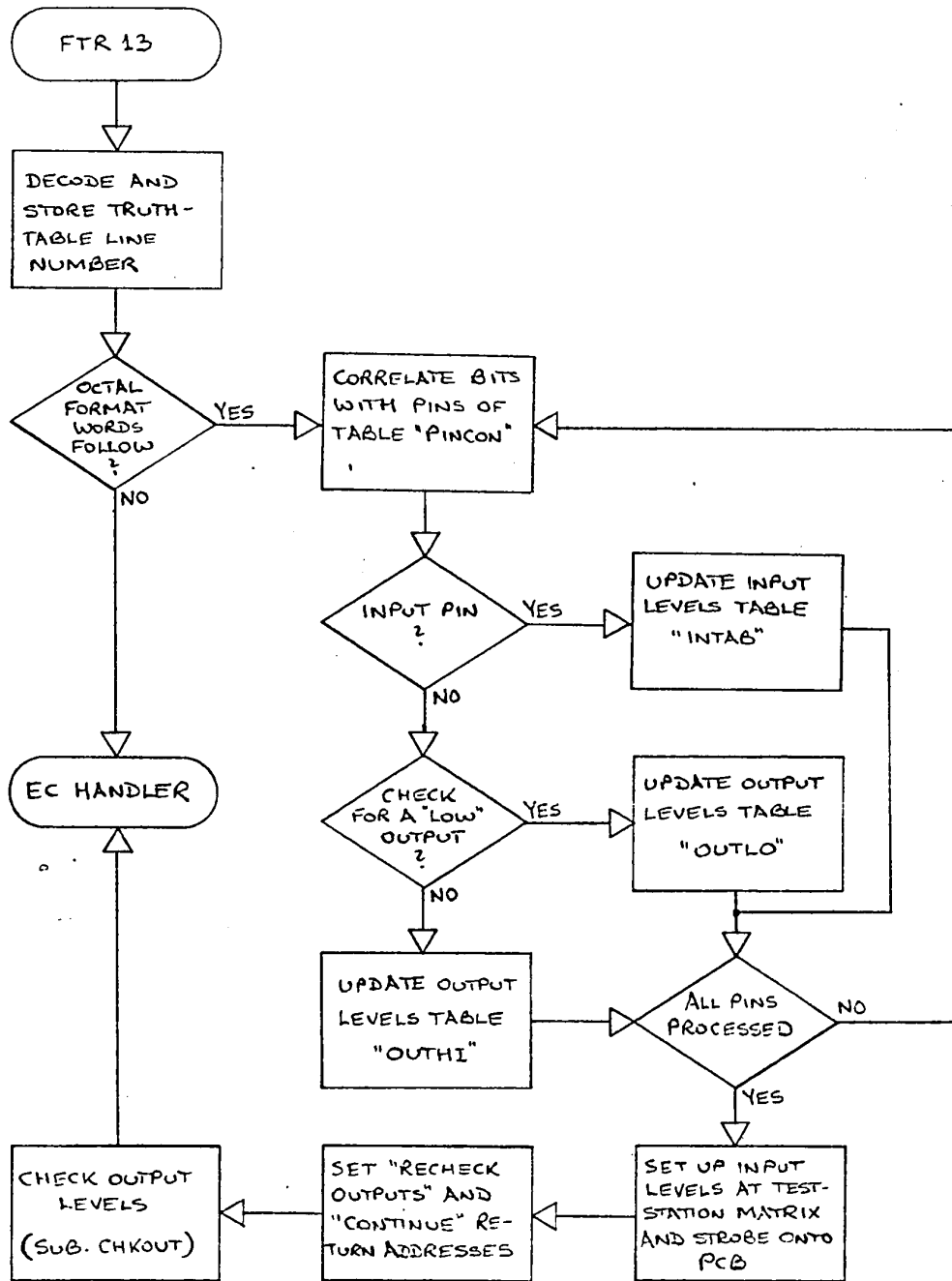
Each testing truthtable line must be identified since it, along with the truthtable number and specific output pins uniquely locates failures. If PCB access pins and testpoints are accessed via EC's 4 and 5 then the preceeding EC 13 is a truthtable line indicator only. But if the truthtable line is coded octally, a number of 18-bit words follow the EC 13 according to how many groups of 18 bits each are necessary to define the entire line.

The FTR 13 stores the truthtable line number. In the case of octal coding it also matches the bits with the corresponding pins or testpoints stored in the pin connection table "PINCON", and updates the input and output levels tables. When all bits have been decoded and the tables completely updated the new input levels are strobed onto the PCB, and the requested outputs checked. Flowchart 5.12 shows the actions of this FTR. The EC 13 and octal format word bit assignments are shown in Figure 5.9.

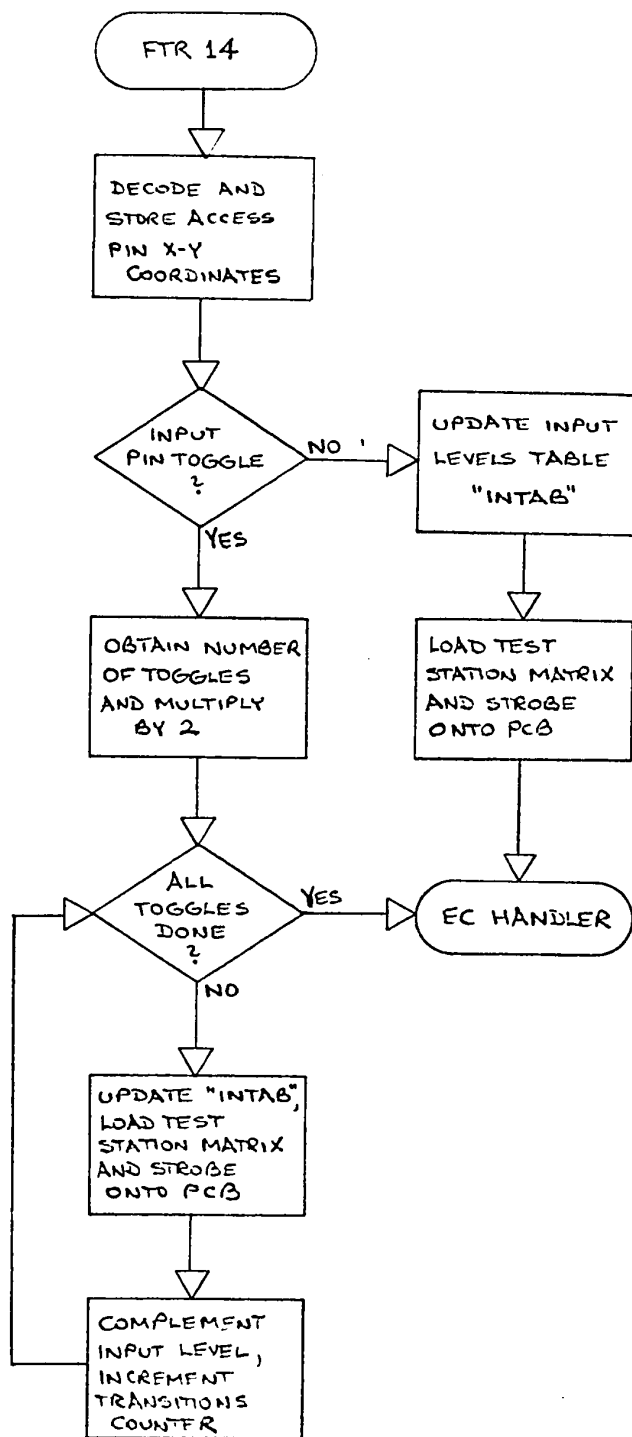
5.4.8 FTR 14: Input Pin Toggle; Sequential Input Setting

EC 14 allows the toggling of a single PCB input pin or active testpoint used as a clock input, for either positive or negative going pulses. A single toggle consists of a 0 - 1 - 0 or 1 - 0 - 1 logic level sequence, with the level always being returned to its original state. A word following the EC specifies the number of toggles to be executed.

Another function of the EC is the specification of sequential level setting. This means that the pin or active



FLOWCHART 5.12: FTR 13 (TRUTH TABLE LINE NUMBER; OCTAL FORMAT)



FLOWCHART 5.13: FTR 14 (INPUT PIN TOGGLE;
SEQUENTIAL INPUT SETTING)

testpoint specified via X-Y coordinates will be set to its specified level individually and separately from any other pin. This differs from the usual case where the test station matrix is preset for all level changes per truth table line, which are then simultaneously strobed onto the PCB. This might under certain circumstances result in race conditions, which can be avoided by changing inputs sequentially.

The FTR 14 is shown in Flowchart 5.13, while the bit assignments of the EC 14 and the toggles wordfill appear in Figure 5.10.

5.5 PROGRAM INTERRUPT HANDLER

The program interrupt facility, when enabled, relieves the OS of the need for repeated I/O flag searching by allowing the ready status of the I/O device flags to automatically cause a program interrupt. The CPU can continue with OS execution until a previously selected device signals that it is ready to transfer data. At that time program execution is interrupted, and the program counter contents (among others) are stored in memory location 0. The instruction in location 1 is then executed, transferring control to an I/O service routine - the interrupt handler. When completed, the routine restores the system to the status prior to the interrupt, enabling the interrupted program segment to continue.

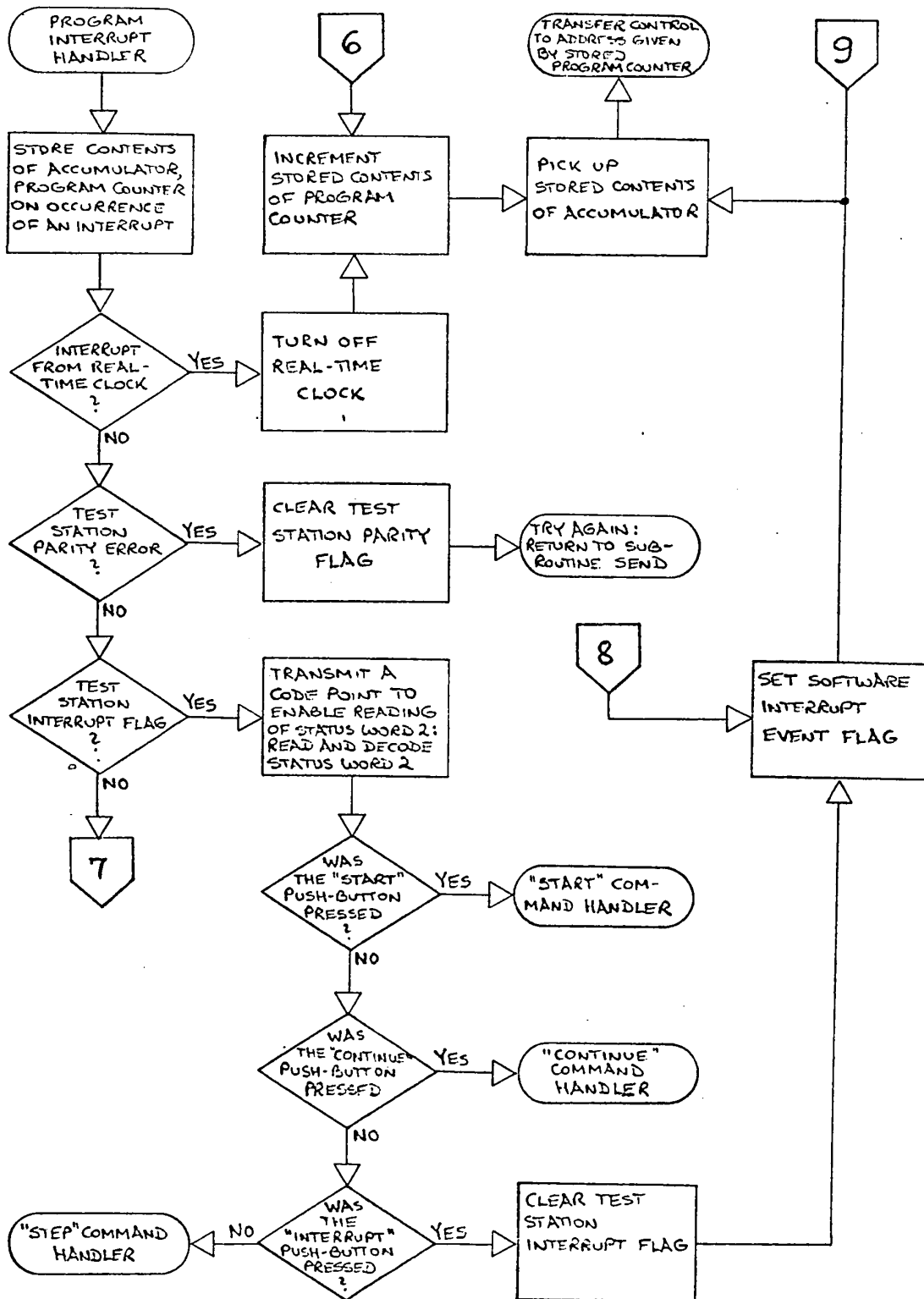
The I/O devices connected to the interrupt line are:

- (a) Real-time 10 KHz clock
- (b) Transmission to test station parity flag
- (c) Test station interrupt flag: "Interrupt", "Start",
"Continue", or "Step" pushbutton
- (d) Teletype keyboard and printer

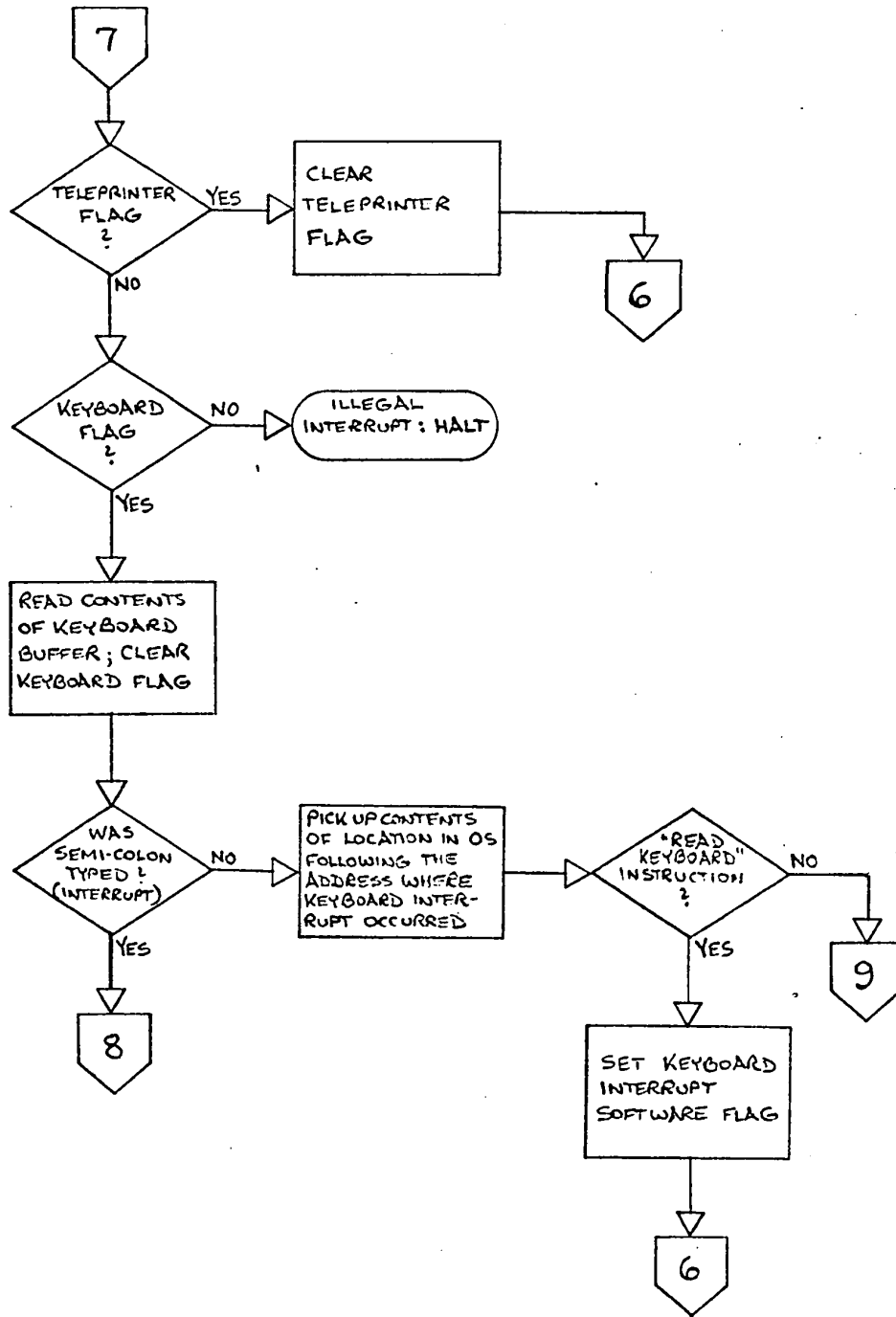
When an interrupt occurs these devices are checked via a skip chain by interrogating associated device-status testing (skipping) instructions. Since the program interrupt facility then is disabled, it must be re-enabled at the end of the service routine.

As has already been explained in section 5.4.4, the real-time clock interrupt occurs whenever a preset time interval has elapsed. The second device on the interrupt line, the test station parity flag, causes an interrupt whenever a FAW, PAW or Wordfill currently being transmitted is of incorrect parity. This check is automatically done by the test station's hardware. The test station interrupt flag comes up whenever one of the associated pushbuttons is pressed. To determine the one that has been pressed a status word must be read from the station. Finally, the Teletype keyboard and printer are connected to the interrupt line as two independent devices. The Teletype is operated in a full duplex mode, so that it echoes automatically on the teleprinter what is being typed on the keyboard.

The Dectape controller has not been included in the interrupt skip chain since Dectape usage occurs only at



FLOWCHART 5.14: PROGRAM INTERRUPT HANDLER (page 1 of 2)



FLOWCHART 5.14: PROGRAM INTERRUPT HANDLER (page 2 of 2)

specific known instances in the OS. Dectape-associated delays are handled in program wait loops rather than through the program interrupt facility.

The activities of the OS's program interrupt handler are depicted in Flowchart 5.14.

5.6 UTILITY SUBROUTINES

To limit the core requirements of the OS it was attempted to gather any similar program sections into subroutines. Whereas some of these consist of only a few programming statements, others are quite complex. The first will not be discussed, while the second will be treated in some detail. This will further show the implementation of the OS's concepts.

Certain subroutines fall into functional groupings, while others perform more or less independent tasks. The following single subroutines, or groups, will be discussed:

- (a) Biasing group
- (b) Output levels checking
- (c) Input levels modifications
- (d) Partial testing boundaries location
- (e) Next EC
- (f) Keyboard reading and character packing group
- (g) Transmission to test station
- (h) Input pin short test
- (i) Station status check

(j) Type-out group

Most of the subroutines of the OS are nested, that is they call other subroutines in turn. All subroutines appear in alphabetical order in the OS program listings, except in the case of the "PRINT" routine, which is part of the type-out group.

5.6.1 Biasing Group Subroutines

This group consists of the three subroutines BIAREQ, BIAS and BIASUB. The first of these is used in communication with the operator whenever it is necessary to specify the biasing mode, such as in the case of "Partial Test", "Looping" or the biasing mode command itself. It requests both bias and input level mode, and sets software registers accordingly, which will be utilized by the BIAS subroutine. If a bias is requested which is not programmed in the TSI, an appropriate response is printed.

The BIAS subroutine sets the programmed bias voltage levels at the station's power supplies. These consist of the following:

- (a) A voltage and current programmable +5 VDC nominal supply which applies positive bias to the PCB's logic devices through access terminal B43.
- (b) The logic input levels 0 and 1, both variable from 0 to +5 VDC, which will be applied to the PCB's input terminals according to the test program's truth tables

- (c) The two logic output level "windows". These are programmable lower and upper limits for PCB output levels 0 and 1.
- (d) A fixed -12 VDC supply which provides negative bias to certain types of logic devices via PCB access terminal B3.

The subroutine first clears (resets) the variable supplies. It next sets the input levels table "INTAB" according to the input level mode to all 0 or 1. The -12 VDC supply is then enabled to the PCB only if required. This is requested in the TSI by specifying circuit type A or B (for the latter pin B3 may be used as an ordinary input or output terminal). Finally appropriate PAW's and Wordfills are transmitted to the test station to set up the variable supplies' levels. A delay occurs to allow relays to operate and power supplies to settle down, and control is transferred to subroutine BIASUB.

The variable supplies are programmed via code points. They each comprise separate Y-levels of 16 X-coordinates, and are thus easily set via a PAW for the Y-level and a Wordfill giving the X's. The translator output for the TSI's biasing specifications consists of these Wordfills, which are stored by the OS in a table called "PWRWRD". There are 4 entries each per biasing mode as shown in Figure 5.11.

Subroutine BIASUB, the last in this group, checks the station power supply status and identifies any failures. Towards this purpose it transmits a FAW for a code point

that enables Status Word 0 to be interrogated. The status word's bit assignments are shown in Figure 5.12. If any of the programmable and fixed value supplies are overloaded, the appropriate bit will be set in the status word. The subroutine decodes them and types an identifying message. On detecting a power supply failure the station is immediately cleared, the "Power Fail" lamp turned on, and control transferred to the keyboard monitor.

WORD	BIT 0	15	16	17	
1	+5VDC (nominal) X-coord.				NOMINAL BIAS
2	Input levels 0 & 1 X-coord.				
3	Output window 0 X-coord.				
4	Output window 1 X-coord.				
5					MARGIN 1
6					
7					
8					
9					MARGIN 4
16					
17					
18					
19	Output window 1 X-coord.				
20					

Each word contains all the X-coordinates necessary to set the programmed level. The Y-level for the code points is inherent in the word number of the table

FIGURE 5.11: BIASING TABLE "PWRWRD"

BIT

0	3	6	7	9	10	11	12	13	14	15	16	17
---	---	---	---	---	----	----	----	----	----	----	----	----

- Bit: 0: Y-coordinate status (a PAW was last transmitted)
- 3: "Step" toggle switch is turned on
- 6: Transmission parity flag (Tied to program interrupt)
- 7: Station enable flag
- 9: Station power supply AC fuse blown
- 10: +5 VDC bias supply current overload (not used by OS)
- 11: +5 VDC bias supply voltage overload (denoted PS1-V)
- 12: -12 VDC bias supply current overload (not used by OS)
- 13: -12 VDC bias supply voltage overload (denoted PS2-V)
- 14: +12 VDC station supply failure (denoted PS3)
- 15: +15 VDC station supply failure (denoted PS8)
- 16: +12 VDC station supply failure (denoted PS6/7)
- 17: Station in manual mode

Note: Subroutine BIASUB checks only bits 9, 11, 13-16.

FIGURE 5.12: STATUS WORD 0 (STATION STATUS) BIT ASSIGNMENTS

5.6.2 Output Levels Checking

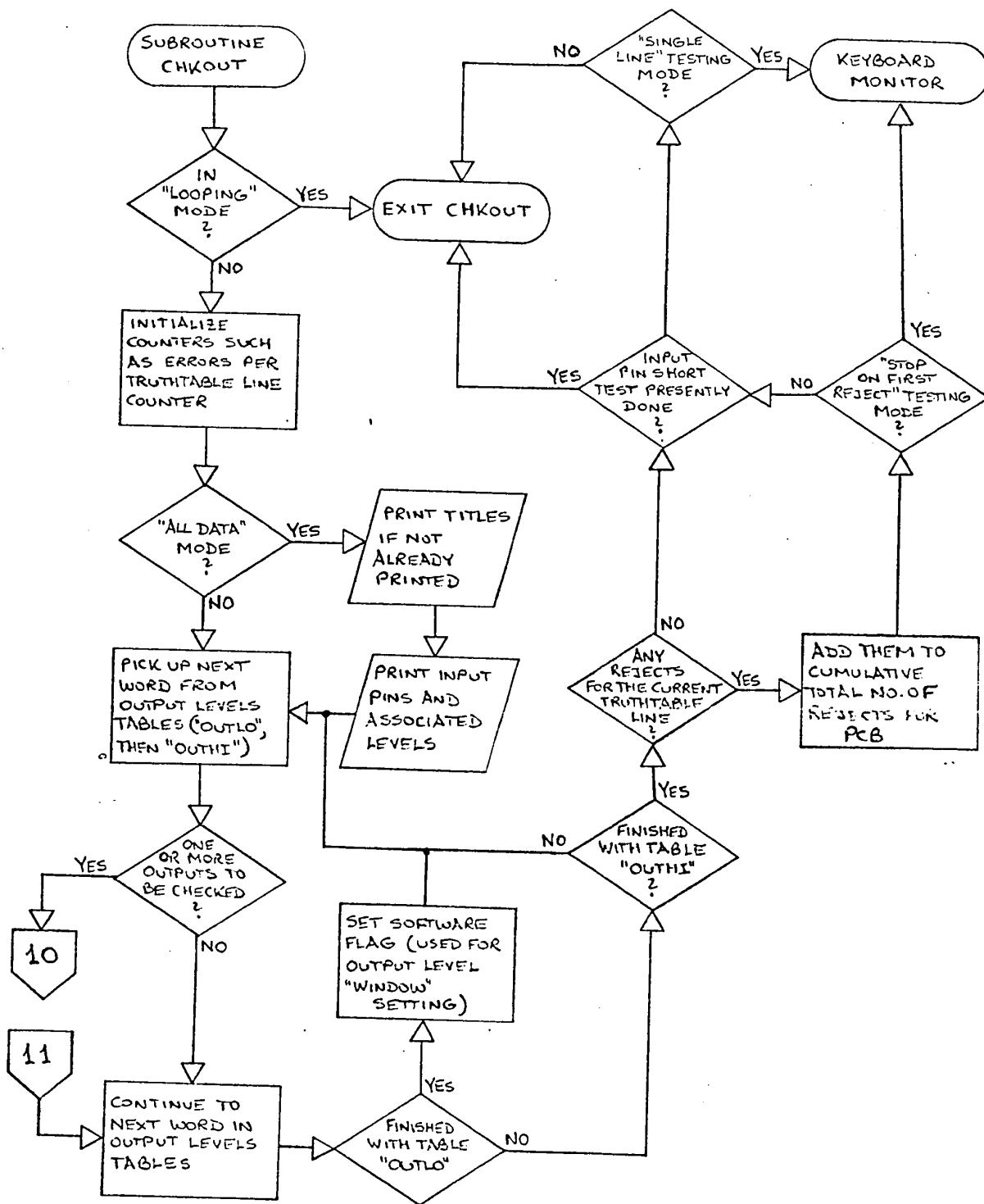
This subroutine, called CHKOUT, checks output levels per truthtable line, and presents output data according to the pre-specified data mode. It causes control to be transferred to the keyboard monitor if the "Single Line" testing mode is specified, or if "Stop On First Reject" is requested, and there are no failures.

The test station hardware is arranged such that output levels are checked with comparators (detectors) which are biased between an upper and lower limit or window. All comparators are simultaneously set for either a low window, to check for a logic 0, or a high window, to check for a logic 1. Hence if both types of output levels need be checked on a truthtable line, the low outputs are checked first, and then the detectors are reset for a high window.

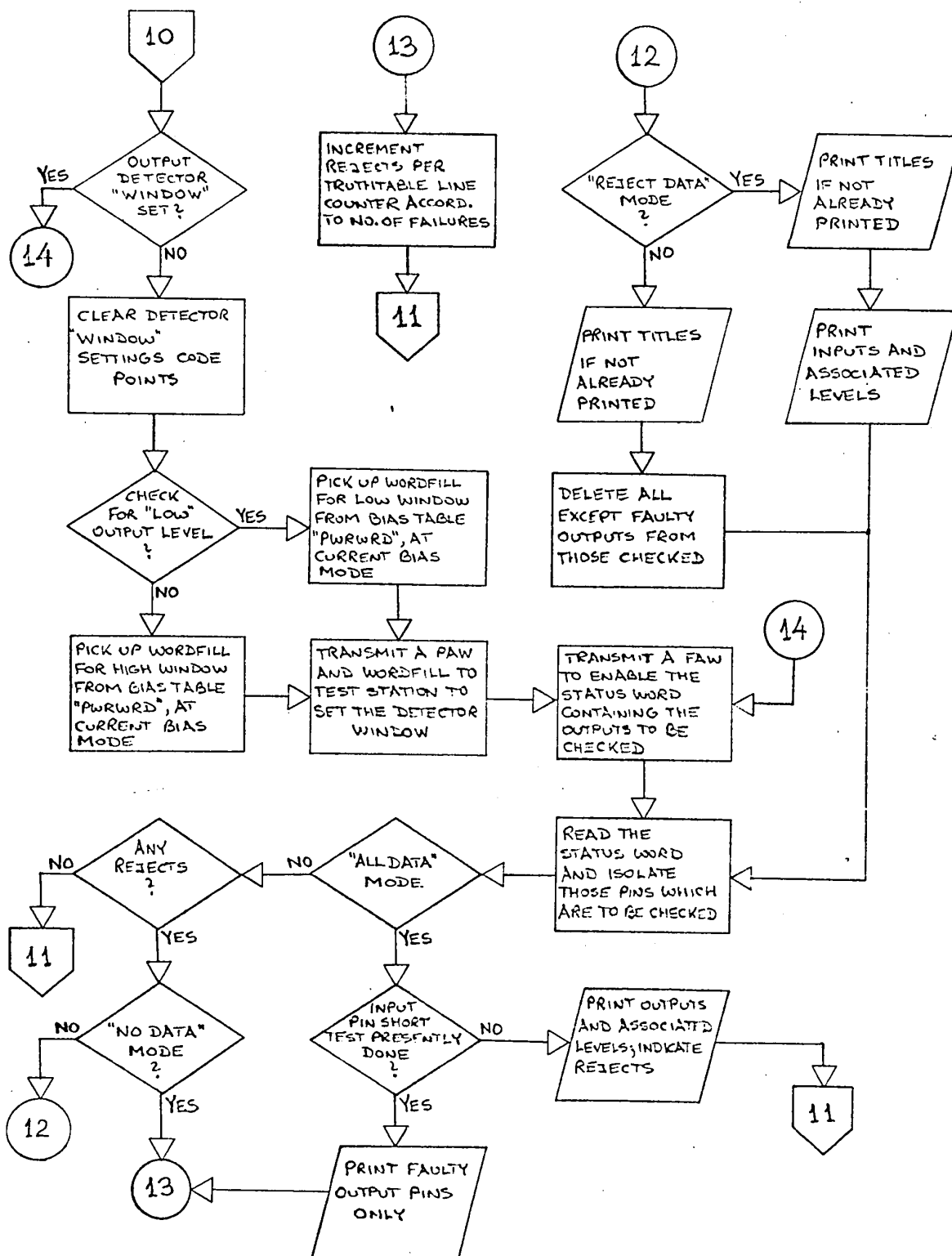
Outputs are read via status words. If an output level does not fall within the detector window, a corresponding bit is set in the status word. There are 9 status words, checking 16 pins or testpoints each, with the same bit assignments as the output level tables "OUTLO" and "OUTH1". The status words are enabled via code points.

The function of this subroutine is hence to obtain the output pins to be checked from the tables "OUTLO" and "OUTH1", one word at a time. It must then set the detector window and read the correct status word. By masking any fault indications in the status word with the current output table word, errors are isolated and can be related to specific pins or testpoints.

Flowchart 5.15 shows the actions of this subroutine. In it, the printing of titles and headings refers to the test program name, the truthtable and line numbers, the current input level and bias modes. Some of these, such as the test program name, are printed only once per complete test run. All of this is performed by the subroutine TITLES.



FLOWCHART 5.15: SUBROUTINE "CHKOUT" (page 1 of 2)



FLOWCHART 5.15: SUBROUTINE "CHKOUT" (page 2 of 2)

5.6.3 Input Levels Modifications

A group of three subroutines is used to modify the contents of the input levels table "INTAB", transmit its contents to the test station, and finally to strobe the updated input levels onto the PCB.

Subroutine INLOAD is utilized to preset the input levels table to all bits either low or high according to the current input levels mode.

Subroutine INMOD in conjunction with a previously called subroutine UPDATE, which has modified a bit or word in the input levels table, serves to first clear the test station PCB inputs matrix, and then reset it according to the current word of the input levels table. Thus when one or more input pin levels are to be changed via an FTR 4 or 5, the input levels table is first modified for the specific input pins by subroutine UPDATE. Subroutine INMOD will then take the updated word in "INTAB" and, using it as a Wordfill, transmit it to the test station matrix.

Subroutine INMOD1 is used to both update the input levels table, and to transmit the "latch enable" code point that causes the contents of the test station matrix to be strobed onto the PCB. Hence all input level changes are applied simultaneously, and unchanged inputs remain applied to the PCB. This code point is only transmitted if the next EC is not an input level modification instruction. Control is then transferred to subroutine BIASUB to check the power supply status. This is done in case an internal short circuit in the PCB might have overloaded a biasing supply.

5.6.4 Partial Testing Boundaries Location

For "Partial Testing" or "Looping" the OS must determine the section in the EC storage table "TSITAB" to be executed. The boundaries are specified by the operator in terms of truthtable and truthtable line numbers. The subroutine LOCATE generates four addresses with respect to the "TSITAB", referring to the following sections:

Address 1 points to the EC that defines the starting truthtable number.

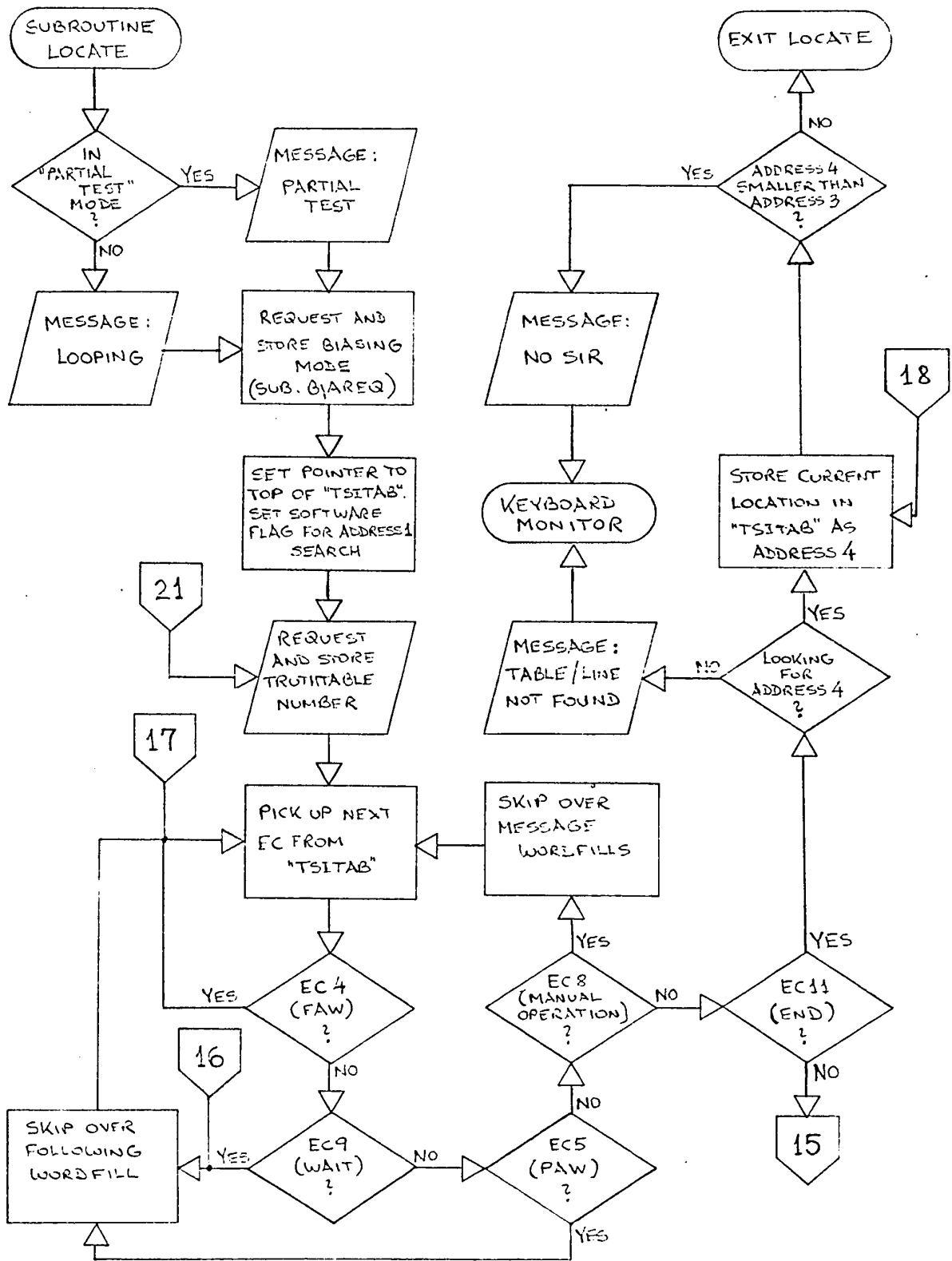
Address 2 identifies the EC defining truthtable line 1 of the starting truthtable.

Address 3 locates the EC which contains the starting truthtable line number.

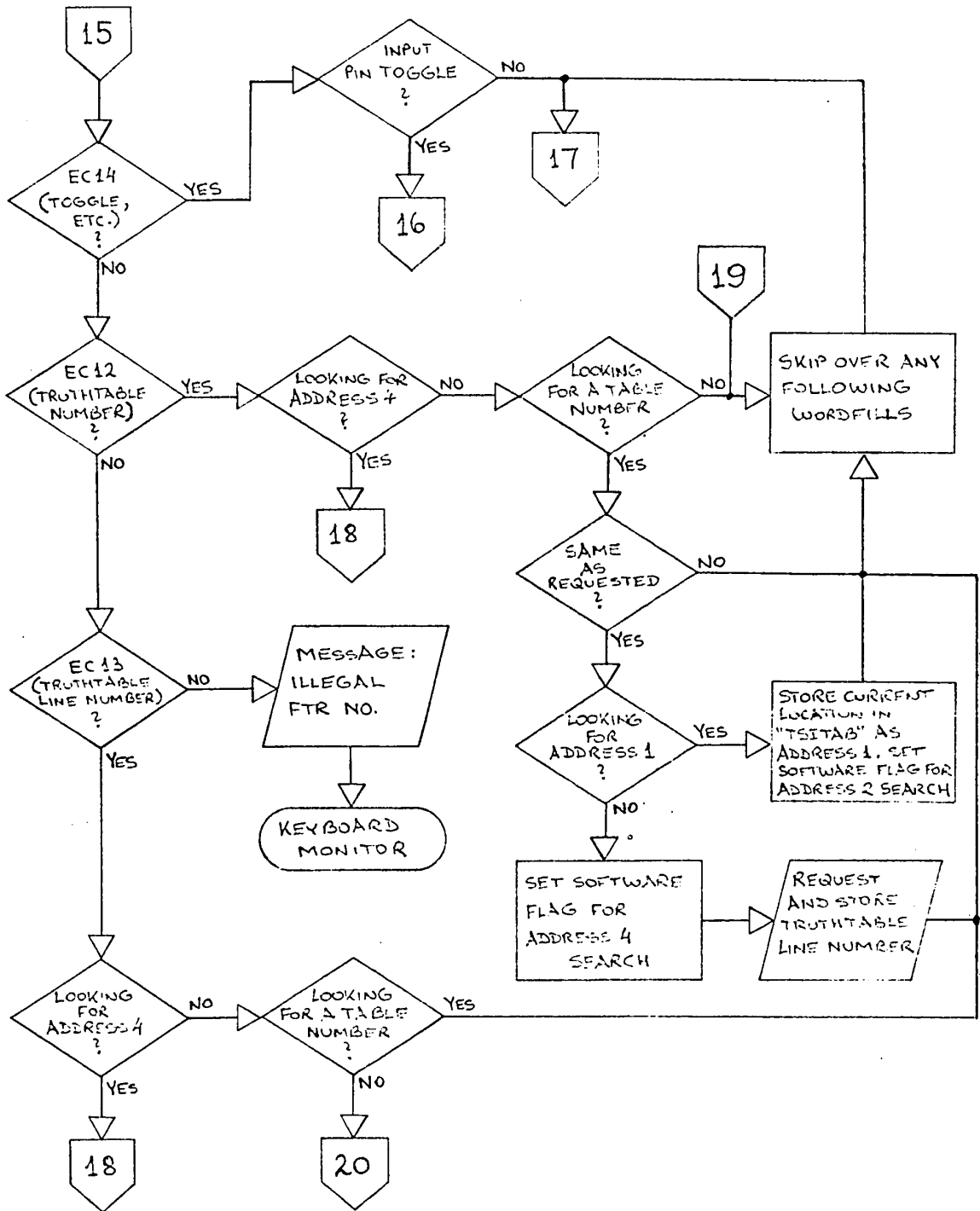
Address 4 refers to the first EC after the end of the specified section.

The reason for four rather than two boundary locations is due to the fact that the initial conditions of the starting truthtable must be executed first. Initial conditions refer to certain input level settings which must be applied in order for the testing truthtable to be valid. These input settings usually disable circuitry which might otherwise influence those devices checked by the current truthtable. Hence a partial test must include any initial conditions of the starting truthtable. These are identified via truthtable line number 0.

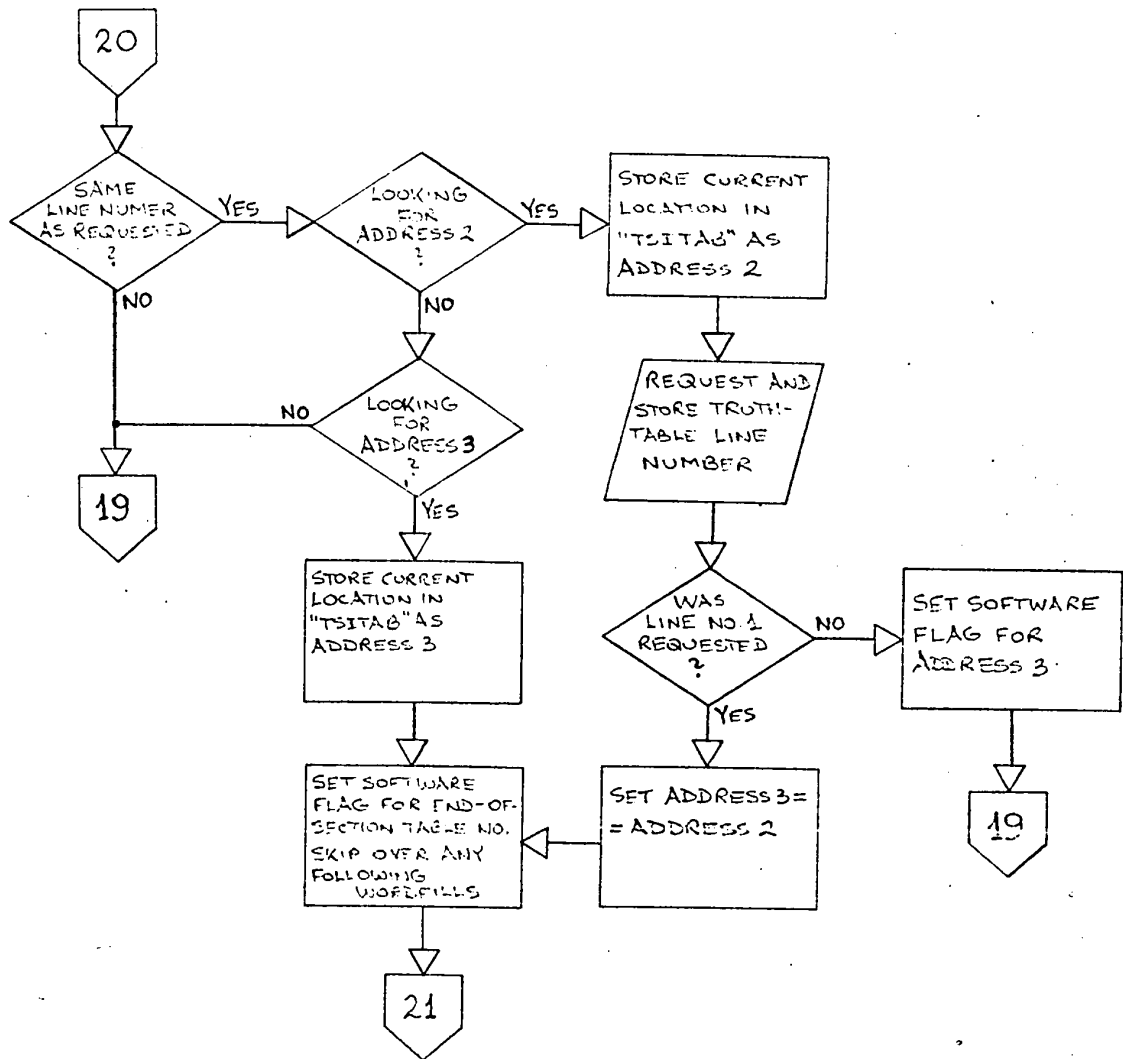
The logic of this subroutine is shown in Flowchart 5.16. As part of the communication with the operator, the biasing mode required for the partial test is also requested.



FLOWCHART 5.16: SUBROUTINE "LOCATE" (page 1 of 3)



FLOWCHART 5.16: SUBROUTINE "LOCATE" (page 2 of 3)



FLOWCHART 5.16: SUBROUTINE "LOCATE" (page 3 of 3)

5.6.5 Next EC

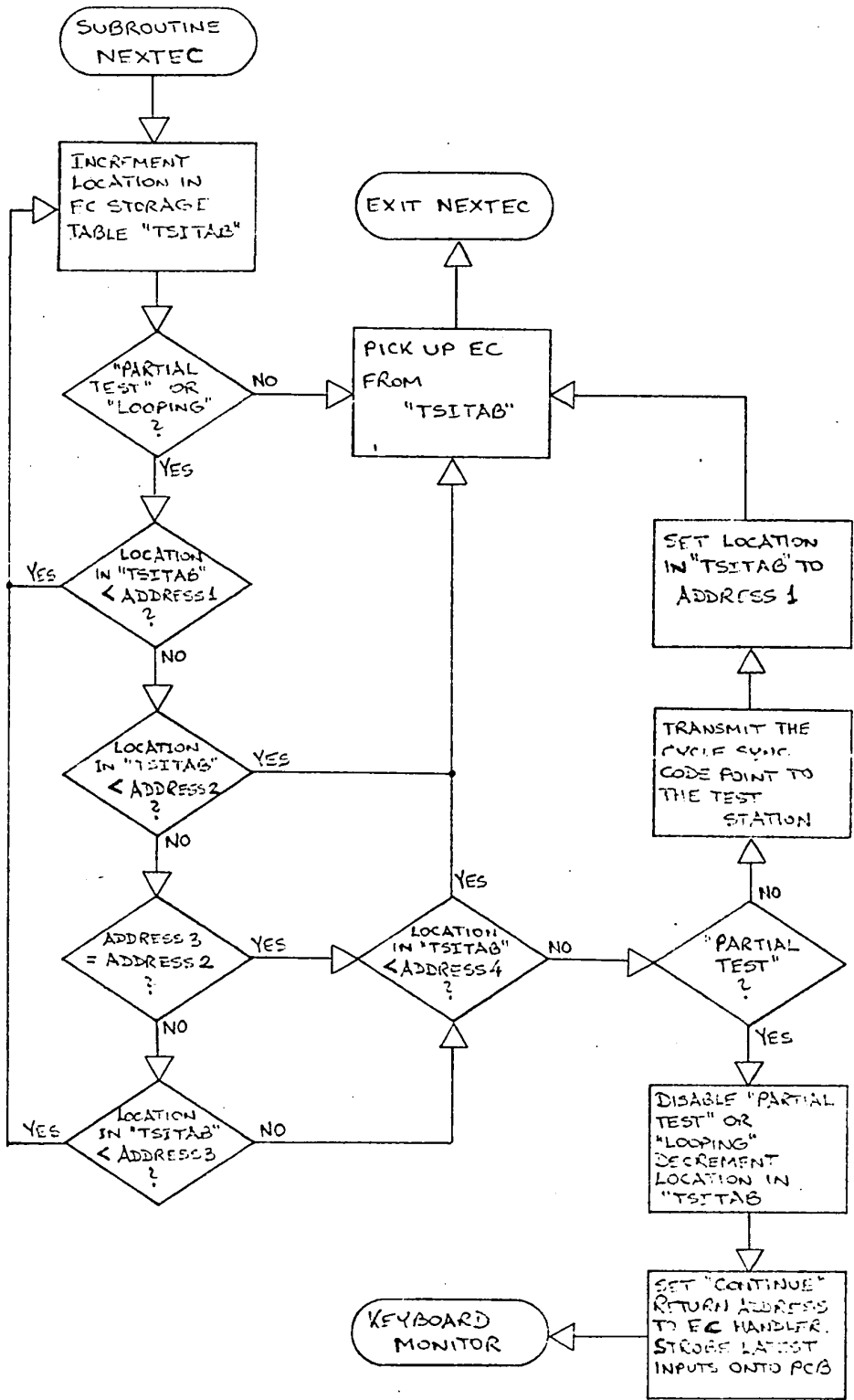
This subroutine NEXTEC picks up the next elemental command from the EC storage table "TSITAB". In the case of "Partial Test" or "Looping" it handles only those EC's falling within the specified section, defined by addresses 1-4. For "Partial Test" a "Continue" return address is set at the end of the section. For "Looping" this subroutine transmits a cycle synchronization code point at the end of each loop. This results in a pulse appearing at a BNC connector at the test station which can be used to externally trigger an oscilloscope. NEXTEC is shown in Flowchart 5.17.

5.6.6 Keyboard Reading and Character Packing Group

Since the OS is a self-contained assembler language program it must include all I/O device handlers. Among them are the two Teletype handlers, one to read the Teletype keyboard, and one to activate the teleprinter. This group of subroutines deals with the former.

When characters are typed on the Teletype keyboard they are stored in the keyboard buffer in 8-bit ASCII code. An I/O transfer instruction must load the contents of this buffer into the CPU's accumulator before another character may be typed. A program interrupt is generated whenever a character is typed.

To conserve valuable storage space the 8-bit ASCII code is trimmed to 6 bits to allow three characters to be stored per 18-bit word. Since this reduces the total character



FLOWCHART 5.17: SUBROUTINE "NEXTEC"

set to 64, it was necessary to redefine certain character codes, and delete others. The codes used appear in Appendix B.

Subroutine READ accepts up to 9 characters from the Teletype keyboard, converts them to 6-bit ASCII, and stores them in three successive registers. If a "Rub-out" is typed all preceding characters are deleted, and a "back-slash" is echoed on the teleprinter. (A "Rub-out" is a non-printing character.) This subroutine is used to read test program names and accept keyboard commands.

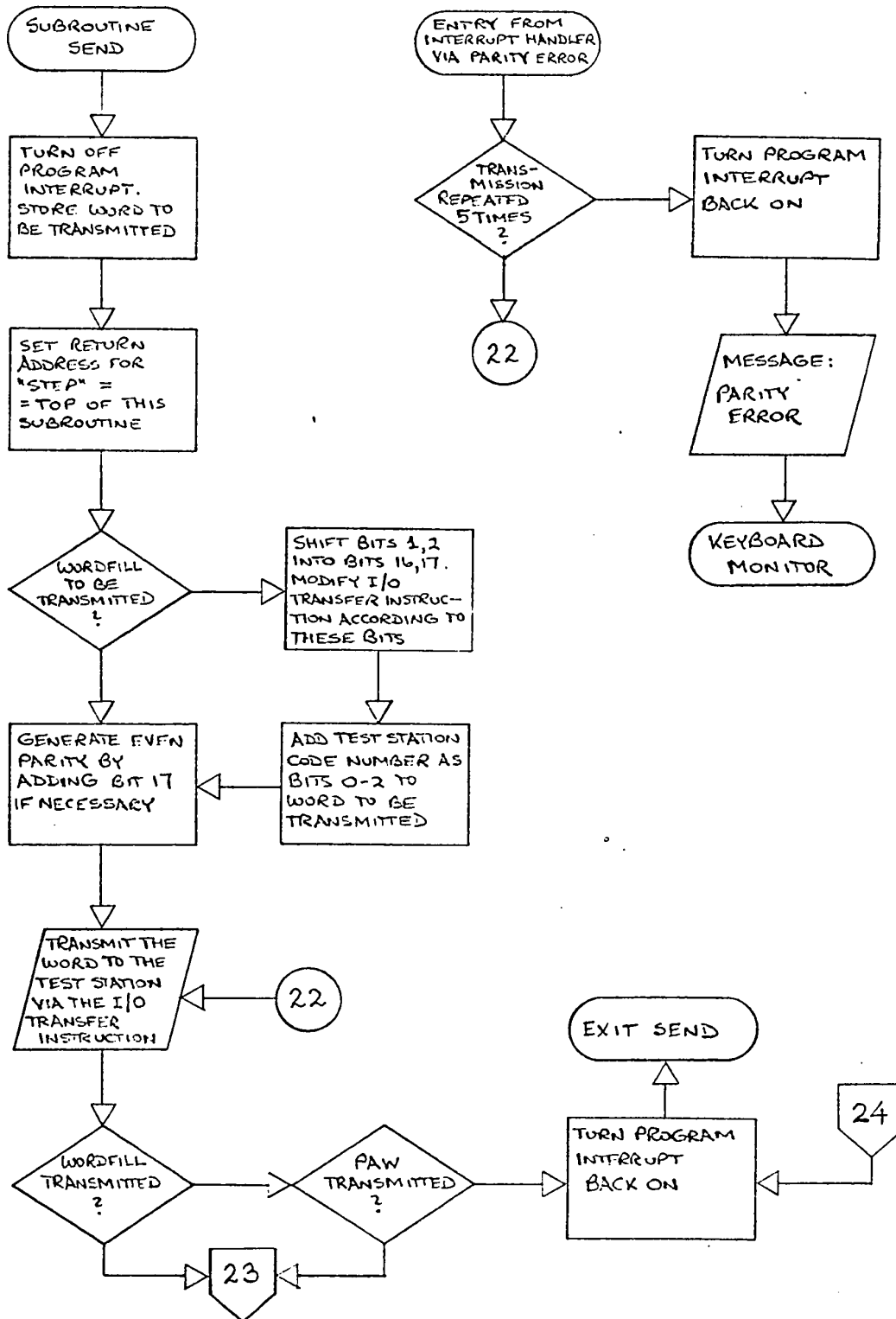
Subroutine READNO reads a multi-digit decimal number from the keyboard, converts it to octal, and stores it. Again, a "Rub-out" will delete all hitherto typed digits. The term octal is used since the 18 binary bits are grouped conveniently into 6 octal digits. All data is of course handled and stored in binary form.

Subroutine READOC reads a multi-digit octal number from the keyboard and stores it.

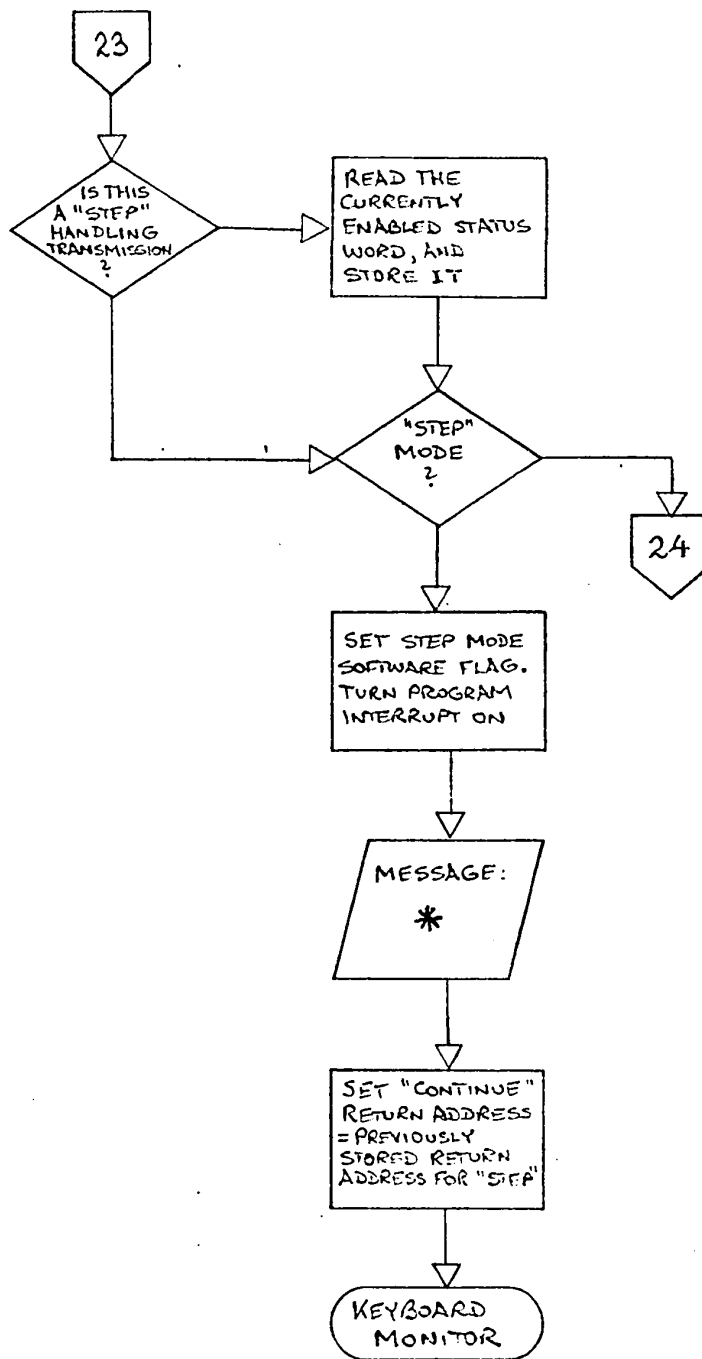
Finally, subroutine READSR is used by READNO and READOC to handle the "Rub-out" and to ignore all characters other than digits.

5.6.7 Transmission to Test Station

Subroutine SEND executes the actual transmissions to the test station of FAW, PAW and Wordfill. It generates the even parity added before transmission, which is checked by hardware at the testset. If there is a parity error the word is re-transmitted up to five times before a diagnostic message



FLOWCHART 5.18: SUBROUTINE "SEND" (page 1 of 2)



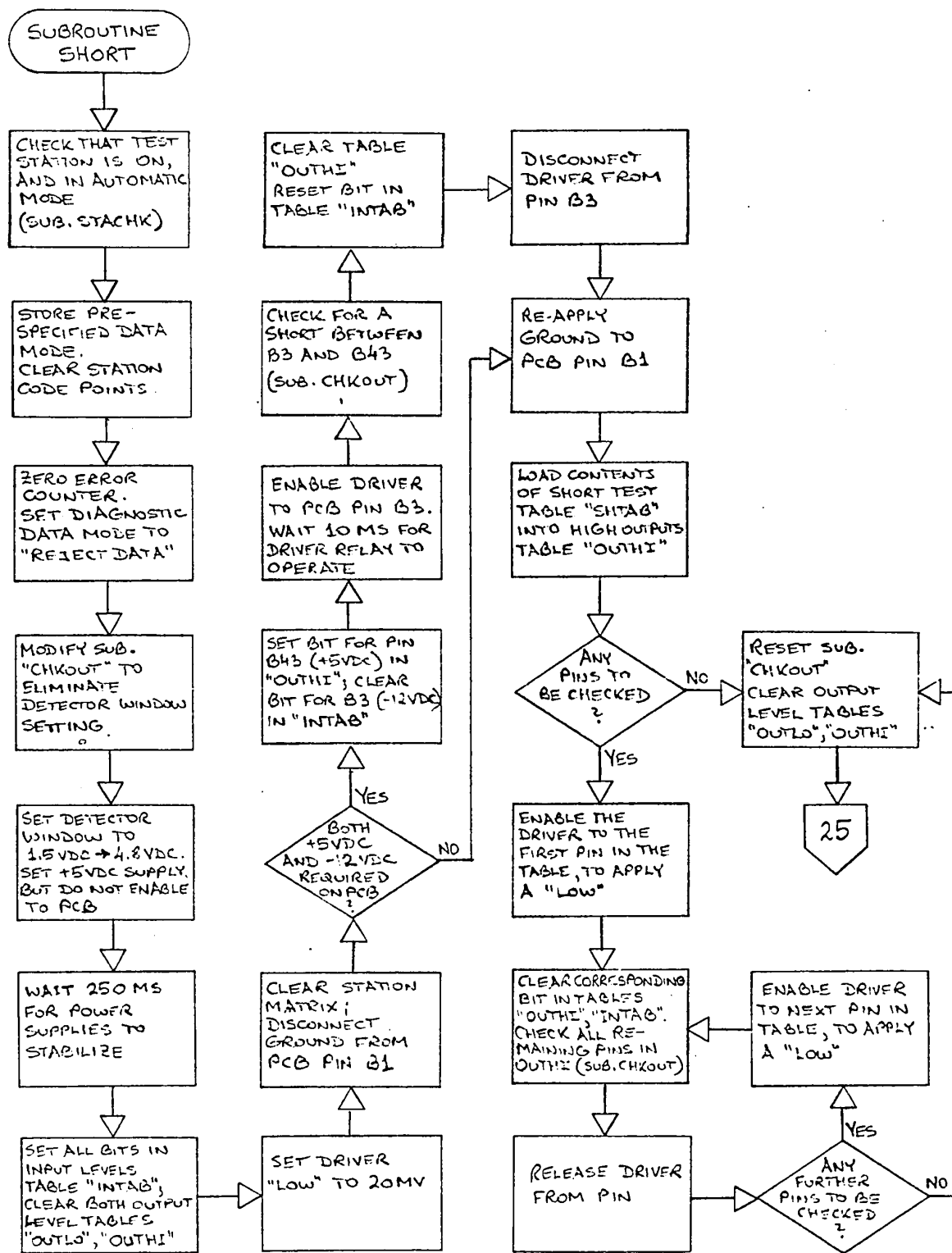
FLOWCHART 5.18: SUBROUTINE "SEND" (page 2 of 2)

5.6.8 Input Pin Short Test

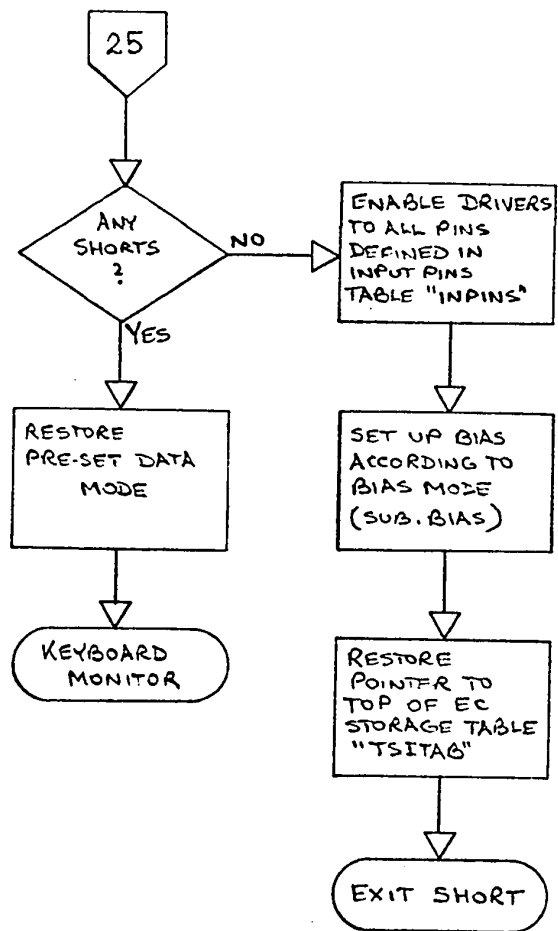
This subroutine SHORT performs the important functions of checking for inter power access pin shorts, as well as for short circuits between specified access pins on the PCB. The first applies to those PCB's requiring both positive and negative bias. Should there be a short circuit on the PCB between these two power paths the resulting excessive current flow could damage an expensive board.

The programmer may specify any or all of the PCB's access terminals for the purpose of having a pin short test performed. This consists of having each of the specified pins checked against all other pins. Diagnostic messages result for each failure and further test program execution is inhibited. These pins are stored as binary 1's in a 6-word input pin short test table "SHTAB", shown in Figure 5.14. This table has the identical format to the input pins table "INPINS", which defines those PCB access pins and active test points which will be used as inputs. This is because input level drivers are connected to PCB inputs via relays, which will be operated after the successful completion of the input pin short test. Hence all input pins must be defined at the beginning of the TSI.

Basically the input pin short test is performed by enabling a low to the first specified pin, and checking all remaining pins for a high. The driver is then released from the first pin, connected to the next pin, and the process is repeated until all pins were set once low. This is indicated in Flowchart 5.19.



FLOWCHART 5.19: SUBROUTINE "SHORT" (page 1 of 2)



FLOWCHART 5.19: SUBROUTINE "SHORT" (page 2 of 2)

	BIT	0	1	2	3...	...	14	15	16	17
WORD	1	A1	A2	A3...		...	A15	A16		
	2	A17...				...	A32			
	3	A33...				...	A48			
	4	B1	B2...			...	B15	B16		
	5	B17...				...	B32			
	6	B33...				...	B48			

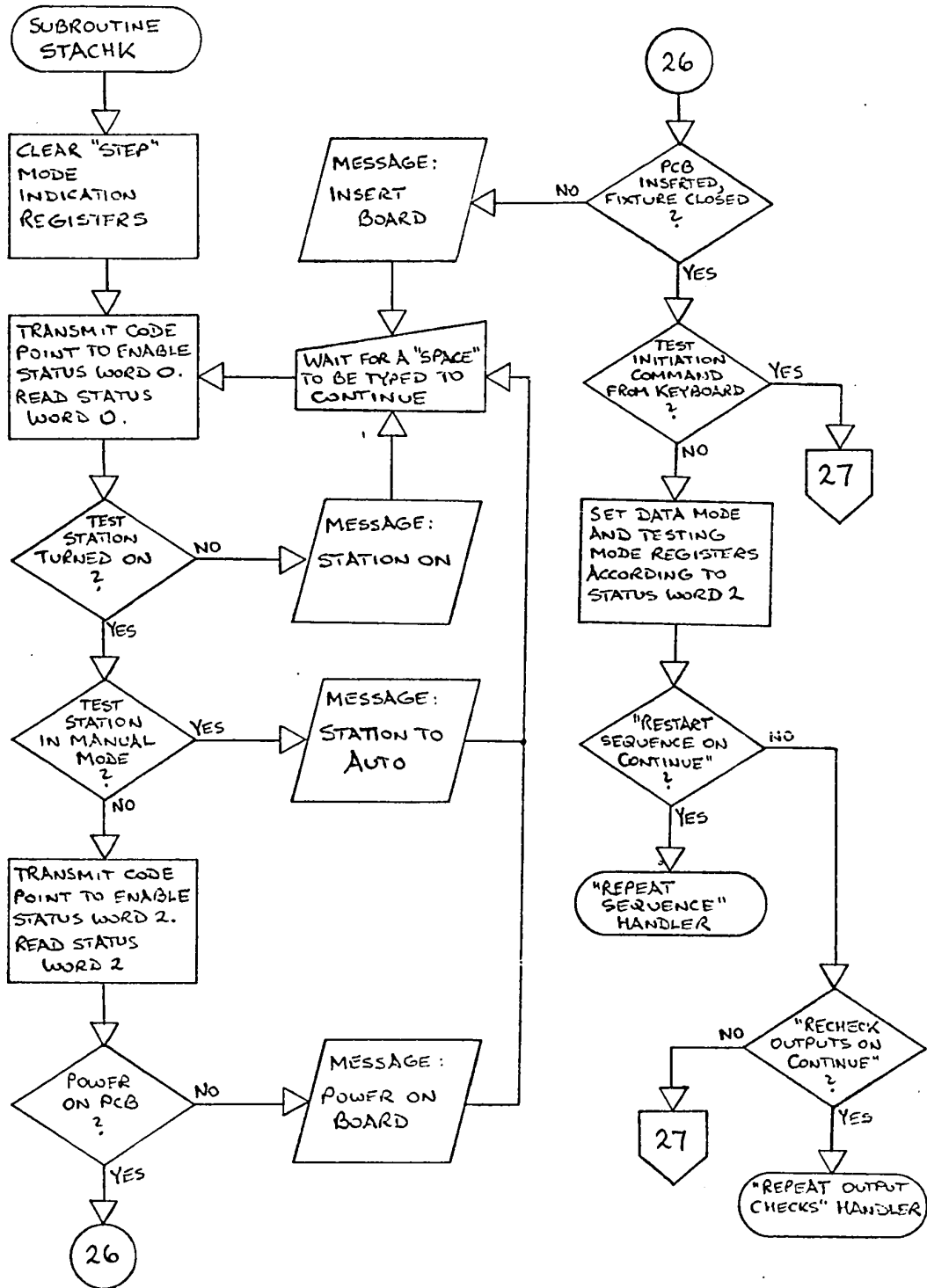
A "1" in any bit position indicates that the corresponding pin is an input pin in the case of the input pin table "INPINS", or is to be checked against all other indicated pins for inter-pin shorts in the case of the input pin short test table "SHTAB"

FIGURE 5.14: INPUT PINS TABLE "INPINS" AND INPUT PIN SHORT TEST TABLE "SHTAB" BIT ASSIGNMENTS

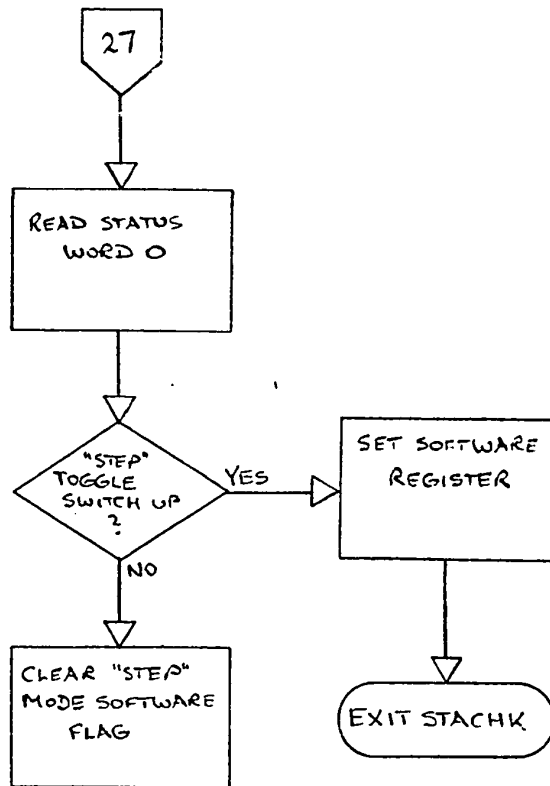
5.6.9 Station Status Check

This subroutine STACHK is used to determine if the test station is set for automatic testing. Furthermore, if a test initiation command originated from the station, the subroutine interrogates station switch settings to determine the data and testing modes.

Status words 0 and 2 are used to obtain the required information. As has been previously mentioned, these status words are selected by transmitting specific code points. They are then read into the CPU's accumulator via an I/O transfer instruction. Status word 0 has been defined by Figure 5.12.



FLOWCHART 5.20: SUBROUTINE "STACHK" (page 1 of 2)



FLOWCHART 5.20: SUBROUTINE "STACHK" (page 2 of 2)

BIT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	17
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

- Bit: 0: Data mode "All Data"
- 1: Data mode "No Data"
- 2: Data mode "Reject Data"
- 3: Data mode "Reject Coordinates"
- 4: Testing mode "Single Line"
- 5: Testing mode "Stop on First Reject"
- 6: Test initiation command "Repeat Sequence on Continue"
- 7: PCB status - power on PCB
- 8: Prove-in mode (not used by the OS)
- 9: Test initiation command "Recheck Outputs on Continue"
- 10: PCB status - PCB inserted into fixture
- 11: PCB status - fixture closed
- 12: "Interrupt" push-button*
- 13: "Continue" push-button*
- 14: "Start" push-button*
- 15: "Step" push-button*
- 17: Station in manual mode

*Test initiation command - tied to program interrupt

FIGURE 5.15: STATUS WORD 2 BIT ASSIGNMENTS

Here it is used to check that the test station is turned on, and in the automatic mode of operation. (If the station is turned off, the attempt to read a status word results in 0's loaded into the accumulator.)

Status word 2 defines the logic console status, which refers to testing and data mode switch settings on the test station console. Its bit assignments are shown in Figure 5.15. In addition this status word also furnishes information whether DC biasing power is applied to the PCB (power may be removed from the PCB via a switch to avoid contact damage due to sparking when the PCB is removed or connected from its fixture). The status of the fixture which receives the PCB and makes contact with its access pins may also be examined through this status word.

This subroutine is further described by Flowchart 5.20.

5.6.10 Type-out Group

This group of subroutines deals with all manner of typing done by the OS. In alphabetical order they include:

- (a) Subroutine TITLES types test program titles, identifies truth tables and lines, biasing and input level modes, according to the currently specified data mode.
- (b) Subroutine TYPDEC types signed decimal numbers.
- (c) Subroutine TYPE types character strings, with the symbols packed in 6-bit ASCII.
- (d) Subroutines TYPTB, TYPSP, and TYPGR are primarily used for formatting, and cause a tab, space, and

carriage return - line feed to be typed.

- (e) Subroutine TYPIN decodes a number between 1 and 154 into PCB access pin or testpoint, and types it as such.

As in the case of keyboard reading subroutines, typing must also be done via I/O transfer instructions, which load a teleprinter buffer, and cause printing. The program interrupt tells whenever a character has been completely typed.

CHAPTER 6
OPERATOR'S MANUAL

This chapter deals with the usage of the OS from an operator's point of view. It describes in detail all communication between OS and operator when testing and troubleshooting logic PCB's. In addition it goes into certain fault location techniques.

To facilitate differentiation between operator input via the Teletype, and OS print-out, without having to identify them each time, the former is here indicated by underlined capital letters, whereas OS messages are capitalized, and always start on a new line. For example, when the operator requires a new test program to be loaded, he types NEW, and the OS will respond by requesting
TSI

All operator input via the Teletype must be terminated by pressing the "space" bar. This acts as a delimiter and indicates end-of-message to the OS. Occasionally, as in the case of a "yes" response to an OS query, this delimiter itself is the answer. This will be denoted here by SPACE, which means that the operator need press the "space" bar only.

Whenever a command may be given either from the Teletype or by activating certain push-buttons and switches on the test station console, this will be indicated by identifying the switches by a capitalized name between quotation marks. Hence the "Start" command may be given by typing S or

by pressing the "START" push-button.

If, at any time, an incorrect character is inadvertently typed, pressing the "Rub-out" key will delete the entire character string, which may then be retyped.

6.1 OPERATING SYSTEM INITIALIZATION AND START

The following steps are necessary to load the OS, assuming a cold start:

The Dectape containing the FACTS DECTRIEVE system and the logic testing system programs must be mounted on drive 1.

The FACTS DECTRIEVE bootstrap paper tape is read into memory starting at location 17400 (octal). This will cause the FACTS DECTRIEVE monitor program to be loaded from the Dectape. It will initialize itself and request

F-D >

The operator responds with the program name, in this case XOS for the OS. The OS will now be loaded into core from the Dectape, initialize itself, and ask

TSI >

which is a request for a test program name. The translated test programs are also stored on the same Dectape under names consisting of up to 8 alphameric. The operator will type the required program's name, and the TSI will be loaded into memory, as shown in Figure 5.1.

The OS responds by typing the test program title, as specified in the original source test program, followed by

OK?

If the correct program was loaded the operator responds with SPACE. The OS then enters its keyboard monitor, indicated by

FACTS 1B

>

and is now ready to receive commands specifying modes and extent of testing.

The above sequence, which is the error-free flow of events, as well as possible faults and associated diagnostics are summarized in Charts 6.1 to 6.3 incl. These and subsequent charts are divided into four columns. The first, headed "Manual Operation", indicates necessary setups and/or checks to be done by the operator. The second column contains commands to be typed by the operator. The "TTY Output" column consists of messages typed by the program currently being executed. In Chart 6.1 this would be the FACTS DECTRIEVE monitor. In all subsequent charts this refers to the OS. Finally, the fourth column shows the operations performed by the program of column three.

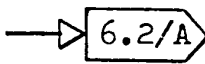
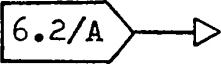
The arrows linking the blocks within the columns show the sequence of events. In the case of bi-directional arrows the events are synonymous. Blocks indicated by  show that the next event occurs in Chart 6.2, terminal A, indicated by  somewhere in Chart 6.2.

CHART 6.1: LOADING THE OPERATING SYSTEM

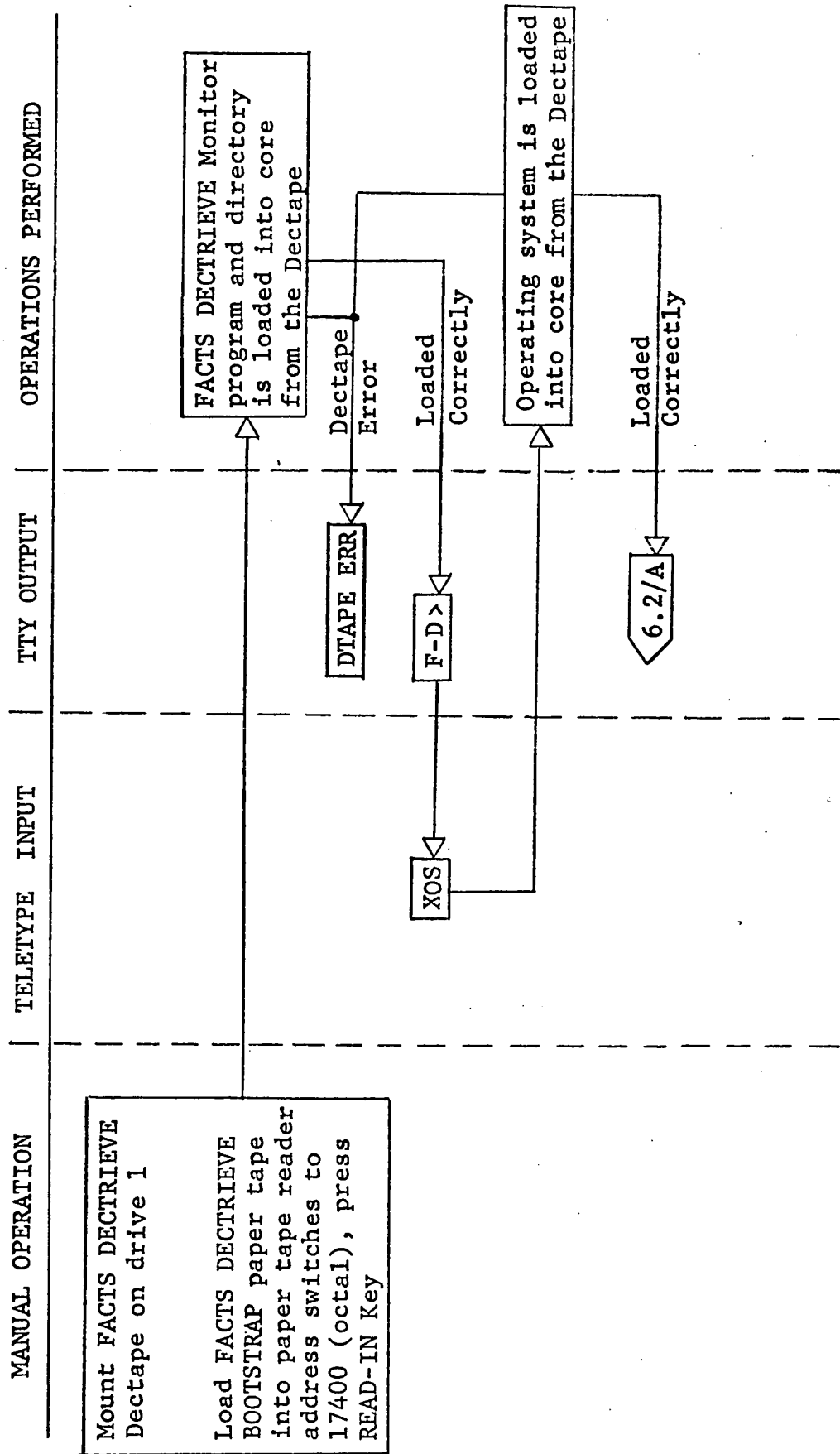


CHART 6.2: O.S. START; TEST PROGRAM LOADING

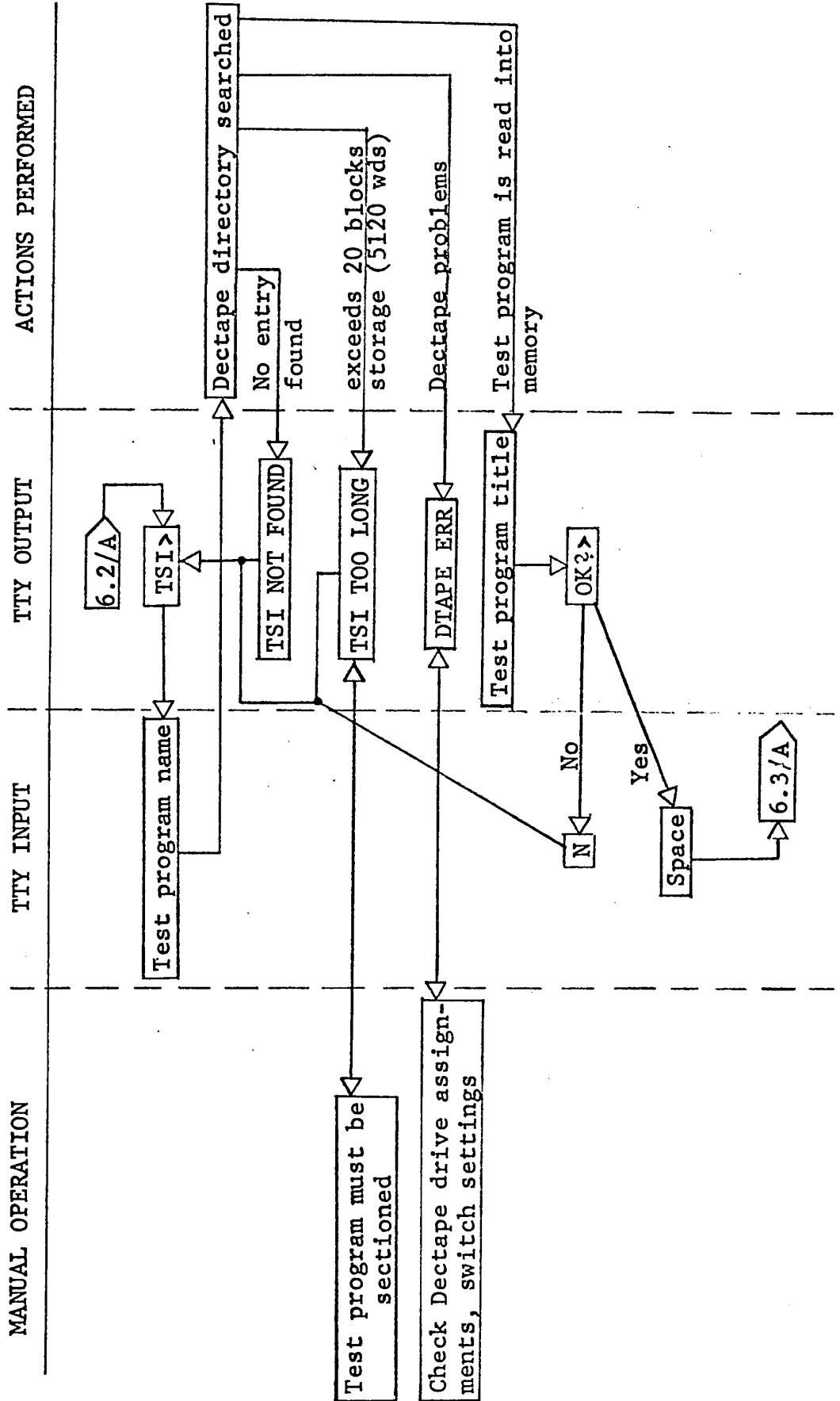
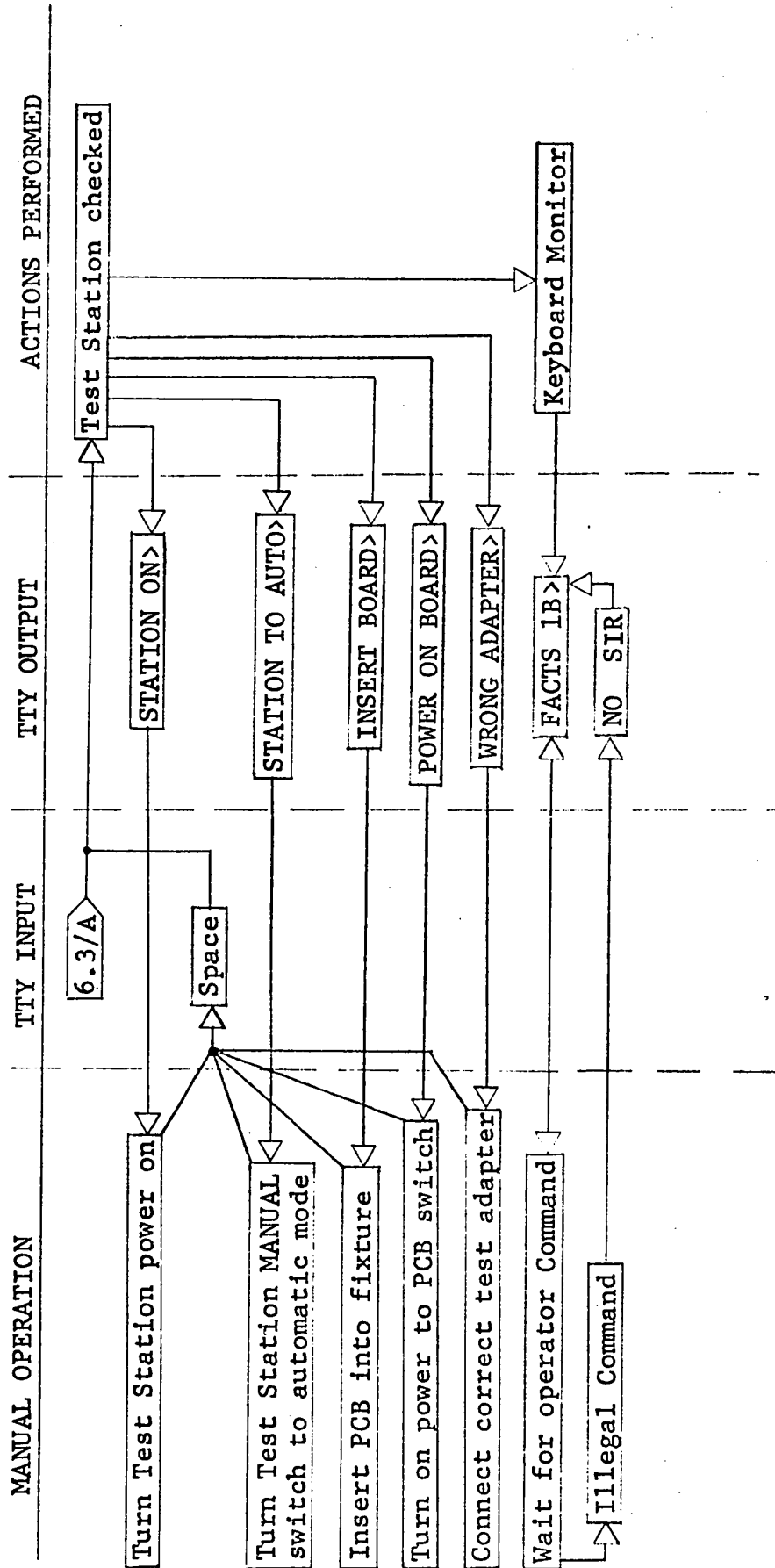


CHART 6.3: TEST STATION STATUS CHECK; KEYBOARD MONITOR



6.2 AVAILABLE OPERATOR COMMANDS

There are three groups of commands available to the operator. The first includes all mode commands, which specify data, biasing, and testing modes, the second consists of the test initiation commands, while the last group includes all miscellaneous or housekeeping commands.

6.2.1 Mode Commands

If subsequent test initiation commands are given via the Teletype, the mode commands must precede them and must also be entered on the Teletype. Mode commands may be given in any order and number. In cases of conflict, the later mode command will override the earlier one.

If the test initiation command is given via push-button from the test station console, those modes are assumed which are set on the test station mode switches. Mode commands do not initiate test program execution.

Table 6.1 lists the data mode commands. They may all be specified from either the Teletype or the test stations "DATA MODE SELECT" switch. When the OS is initially loaded the "No Data" mode is assumed as the default option.

The biasing mode command sets up the biasing and input level modes for subsequent testing. It can only be specified on the Teletype. To call it, the operator types BM. The OS responds with

BIAS>

The operator types 0 (zero) or SPACE for nominal bias, or the digits 1 to 4 for margins 1 to 4 respectively. The bias levels specified must be existent in the TSI. If the operator specifies a margin that was not programmed the OS will return

NO SIR

and return to the keyboard monitor. Otherwise it will request I.L.M.>

If the operator returns 0 (zero) or SPACE a low input level mode (all unused inputs low) is assumed. If he types 1, a high input level mode results. The OS then returns to the keyboard monitor to await further commands.

The default option for the biasing mode is nominal bias; all unused inputs low. This is automatically assumed when the OS is initially loaded, and after a test program has been completely executed.

The testing mode commands are shown in Table 6.2. These may also be specified from either the Teletype or the indicated test station switches. The "Normal Run" mode has no distinct test station switch, but is set by resetting the other mode switches for the equivalent result. Note that the biasing mode cannot be reset from the test station in this manner. The "Normal Run" mode is the default option originally assumed when the OS is loaded.

COMMAND NAME	TTY INPUT	TEST STATION SWITCH	COMMAND FUNCTION
All Data	AD	ALL DATA	Types all input/output pins and associated levels for all testing truthables & lines. Indicates failures.
Reject Data	RD	REJECT DATA	Types all input/output pins and associated levels for those truthable lines containing one or more failures
Reject Coordinates	RC	REJECT COORDINATES	Types only those output pins and associated levels that failed per truthable line
No Data	ND	NO DATA	Types only OK, or the total number of failures at end of test program execution. (Default option.)

Note: In all cases except "No Data" the current biasing mode and input level mode are identified, as well as truthable and truthable line numbers. "OK" or the total number of rejects are always typed at then end of test program execution.

TABLE 6.1: DATA MODE COMMANDS

COMMAND NAME	TTY INPUT	TEST STATION SWITCH	COMMAND FUNCTION
Stop on First Reject	FR	FIRST REJECT	Testing will be stopped at the end of the next truthable line containing one or more failures. All input levels will remain applied.
Test Single Line	SL	SINGLE LINE	Testing will unconditionally stop at the end of the next truthable line, with all input levels remaining applied.
Normal Run	N	Data Modes switch to NO DATA, above switches off	The data mode will be reset to "No Data", the biasing mode will be reset for nominal bias, and the testing mode will be reset for no intermediate halts. (Default option.)

TABLE 6.2: TESTING MODE COMMANDS

6.2.2 Test Initiation Commands

All commands listed in this section cause partial or complete execution of the test program, with output data presented according to pre-specified data mode, initial biasing according to pre-specified biasing mode, and extent of test program execution according to pre-specified testing mode.

These commands are listed in Table 6.3. The operation sequence of the "Start", "Partial Test" and "Test Program Verification" commands is also shown in Charts 6.4 to 6.6 respectively. The following require further information:

An input pin short test is done unconditionally preceding test program execution, whenever "Start", "Partial Test", "Loop" or "Test Program Verification" commands are given. On those PCB's requiring both +5 VDC and -12 VDC bias, it first checks for a short between the corresponding two access pins. If such a short exists the OS types

B3 TO B43 SHORT TEST

OUTH1 B43 REJECT

(indicating that access pin B43 which carries the +5 VDC supply was pulled down to the level 0 applied to access pin B3). The OS then ceases any further testing and returns to the keyboard monitor section. If, on the other hand, there was no failure, or in any case for those PCB's which do not require the above test, an input pin short test follows for those access pins specified in the TSI. In the case of any failure the title

INPUT PIN SHORT TEST

COMMAND NAME	TTY INPUT	TEST STATION SWITCH	COMMAND FUNCTION
Start	S	START push-button	Test program execution starts at the beginning, following an input pin short test.
Continue	C	CONTINUE push-button	Test program execution continues from the intermediate halt.
Repeat Output Checks	REP	RECHECK OUTPUTS ON CONTINUE toggle + CONTINUE push-button	The outputs specified in the current truth table line at the time of the intermediate halt are rechecked.
Partial Test	P	None	The specified test program section is executed once.
Repeat Sequence	RS	RESTART SEQUENCE ON CONTINUE toggle + CONTINUE push-button	The previously specified "Partial Test" is repeated.

TABLE 6.3: TEST INITIATION COMMANDS (page 1 of 2)

COMMAND NAME	TTY INPUT	TEST STATION SWITCH	COMMAND FUNCTION
Loop	L	None	The specified test program section is executed in a continuous loop. Outputs are not checked.
Test Program Verification	TVO	None	Faults are simulated on specified IC input and output pins to check if the TSI will detect these.
Step	ST	STEP toggle-switch and push-button	A single transmission to the test station occurs (except for those transmissions necessary to handle the command itself).
Interrupt	;	INTERRUPT push-button	Test program execution is unconditionally halted before the OS picks up the next EC. If typing is in progress it will continue until the end of the current message.

TABLE 6.3: TEST INITIATION COMMANDS (page 2 of 2)

is typed, followed by the pins which are shorted. Only when all short tests pass is test program execution started.

The test program boundaries for the "Loop" command are specified in the same manner as those for "Partial Test" which is shown in Chart 6.5. For both types of test the initial conditions of the truth table containing the start of the section are executed first. Once a loop is started it continues until the "Interrupt" command is given. A synchronization pulse appears at the "CYCLE SYNC" BNC connector at the test station console per pass.

For "Test Program Verification" the IC access pins are defined as octal numbers, shown in Figure 6.1 below.

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
IC PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
OCTAL DIGIT	1			2			3			4			5			6		

IC pin definitions are entered as octal numbers - one octal digit per 3 IC pins. If an IC pin is used, the equivalent bit is set and counted in the corresponding octal digit. Hence, if IC pins 2, 6, 10, 11 and 13 are to be defined (say as inputs), the equivalent octal number will be 210640

FIGURE 6.1: IC PIN DEFINITION FOR TEST PROGRAM VERIFICATION

CHART 6.4: "START" COMMAND

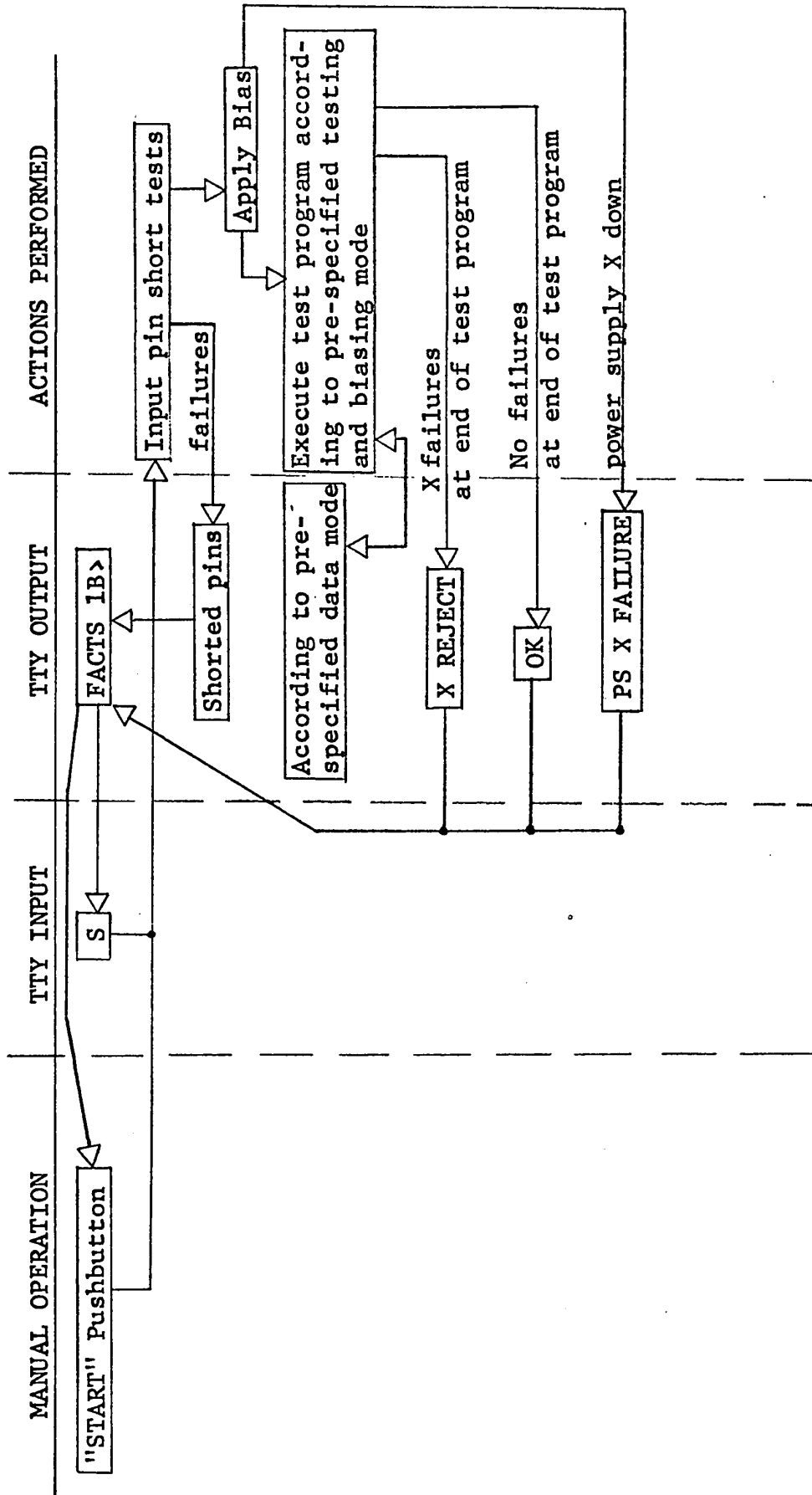


CHART 6.5: "PARTIAL TEST" COMMAND

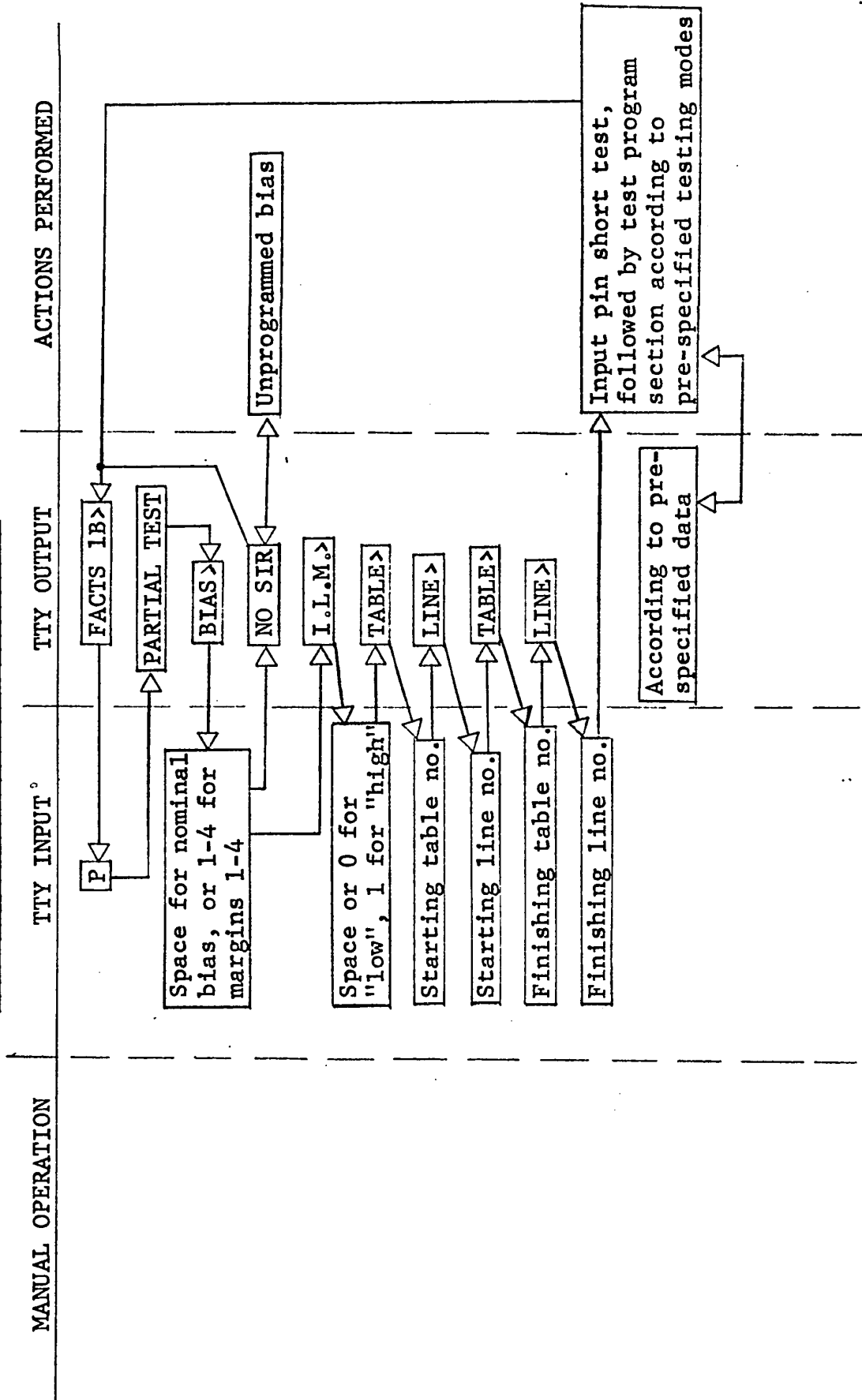
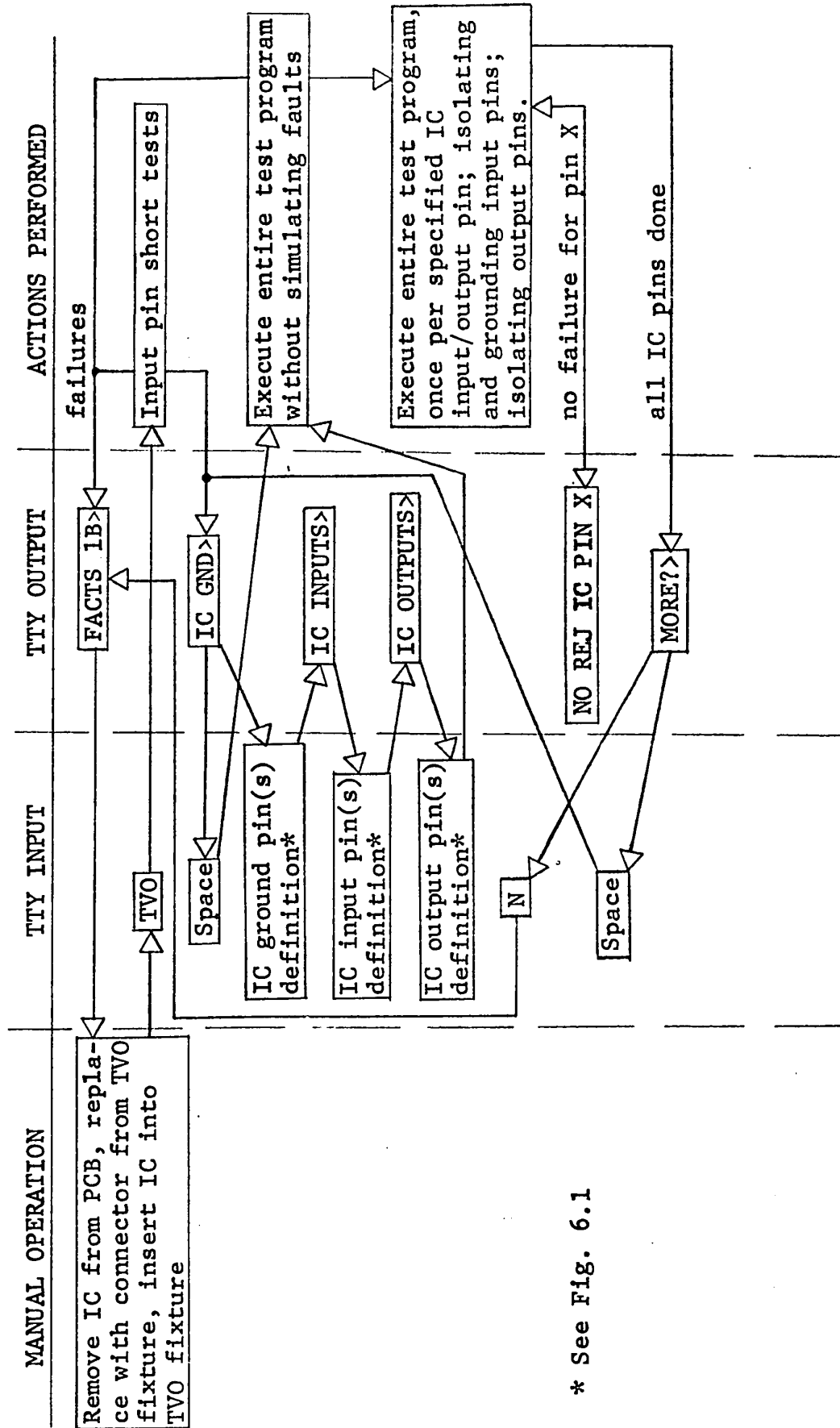


CHART 6.6: TEST PROGRAM VERIFICATION



* See Fig. 6.1

6.2.3 Miscellaneous Commands

The three remaining available commands are summarized in Table 6.4 below.

COMMAND NAME	TTY INPUT	TEST STATION SWITCH	COMMAND FUNCTION
General Release	GR	STATION CLEAR	Resets and clears all levels and relays at the test station
New Test Program	NEW	None	Requests a new test program name to load a new TSI
FACTS DECTRIEVE		None	Control is transferred to the FACTS DECTRIEVE bootstrap program

TABLE 6.4: MISCELLANEOUS COMMANDS

6.2.4 Command Combinations

Table 6.5 gives both a summary of all available commands, and indicates which combinations of mode and test initiation commands are possible.

A "Y" means that the combination is available, while an "N" indicates that the combination cannot be used. A "-" signifies that the combination is meaningless. An "*" means that while the combination is possible, it is not recommended.

6.3 TESTING AND TROUBLESHOOTING TECHNIQUES

A primary requirement for facilitating troubleshooting of faulty PCB's are of course simple but comprehensive testing truthtables and accompanying circuit schematics. It must be quite clear to the operator which part of the PCB is being tested at any time. If possible, testing truthtables should also include logic levels at intermediate points which, while not being checked automatically by the test program, may be used by the operator when probing the circuit manually. An attempt should be made to divide the PCB into small circuit sections to be checked independently by separate truthtables. This can usually be accomplished by disabling peripheral circuitry. Interaction between these sections can then be checked via one or more final and larger truthtables.

6.3.1 Test Program Prove-in

The desirable situation for a prove-in of a new test program is to have a PCB available which is known to be functioning. It then becomes a relatively simple problem to detect logical errors in the TSI. If, on the other hand, the prototype PCB is of questionable performance, both test program and PCB must be debugged concurrently, vastly increasing the difficulties. In this case it is adviseable to manually check the signal flow at each reject, and thus determine whether PCB or test program is at fault. This is at

best a tedious and time consuming process.

Once a test program is known to be correct, test program verification should be done to make sure that it is also sufficient. Additional tests may then be added to make certain that every device on the PCB is being checked.

6.3.2 Sorting of PCB's

To separate functioning from faulty PCB's the test program should be executed as rapidly as possible. This is done by specifying the "Normal Run" mode, and using the "Start" command. Faulty PCB's will have their total number of failures printed and can be isolated.

6.3.3 Troubleshooting

If the total number of rejects was small, say less than 10, they should all be listed using the "Reject Coordinates" data mode. This helps to identify any pattern which may serve to identify the error. Otherwise a small number of failures should be listed. After a number of reject coordinates has been obtained the listing process may be stopped via the "Interrupt" command.

Using the "Stop at First Reject" and "Reject Coordinates" modes combination testing is halted at the failing truth table line, with all inputs applied. The PCB may now be probed with the test station's Hewlett Packard Logic Probe to check the signal flow through the circuit.

Rejects for which the solution has been found may be skipped by using the "Partial Test" command. Care must be taken to avoid starting a "Partial Test" or "Loop" within a memory retention sequence, which might destroy the programmed test sequence of the sequential circuit.

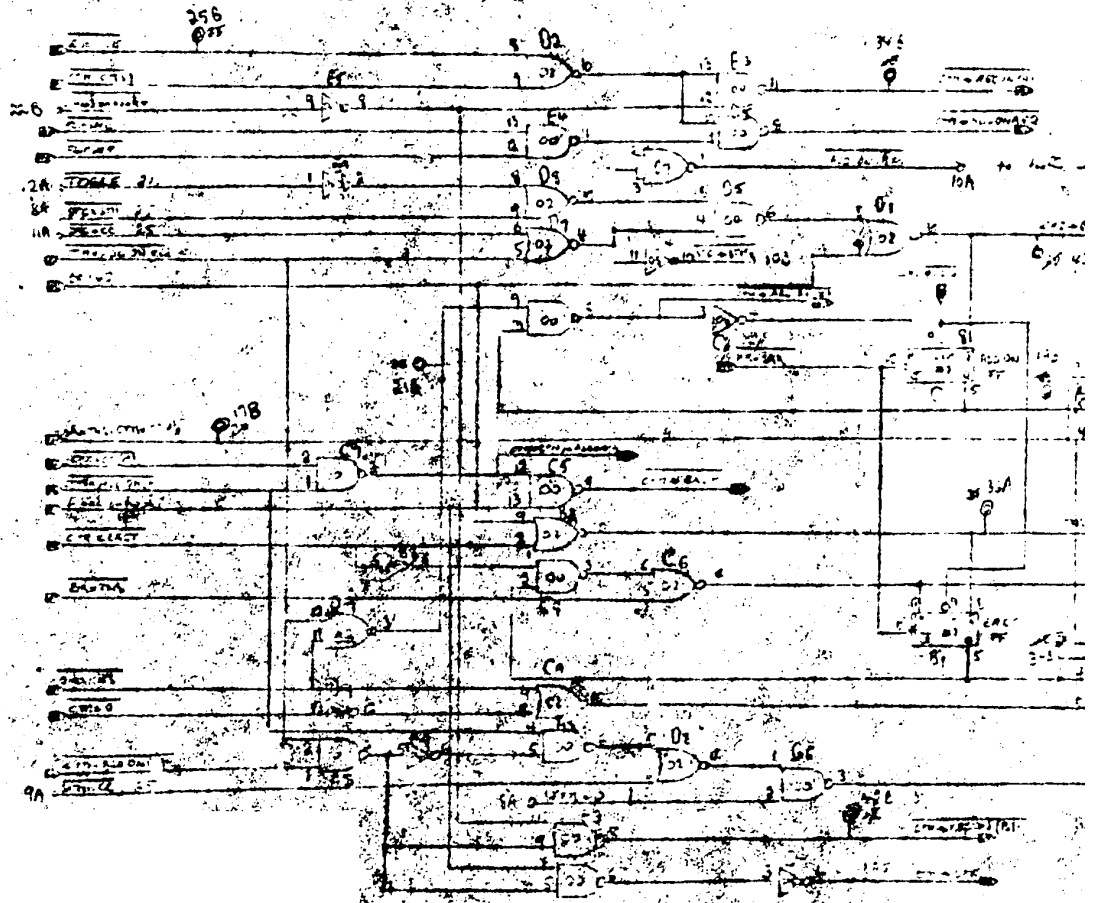
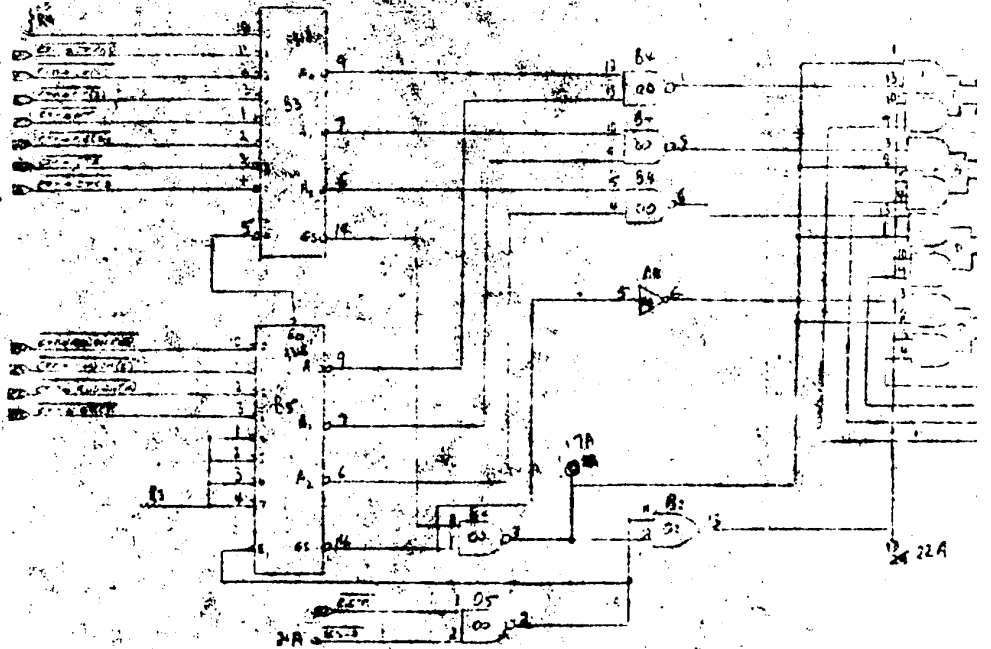
CHAPTER 7

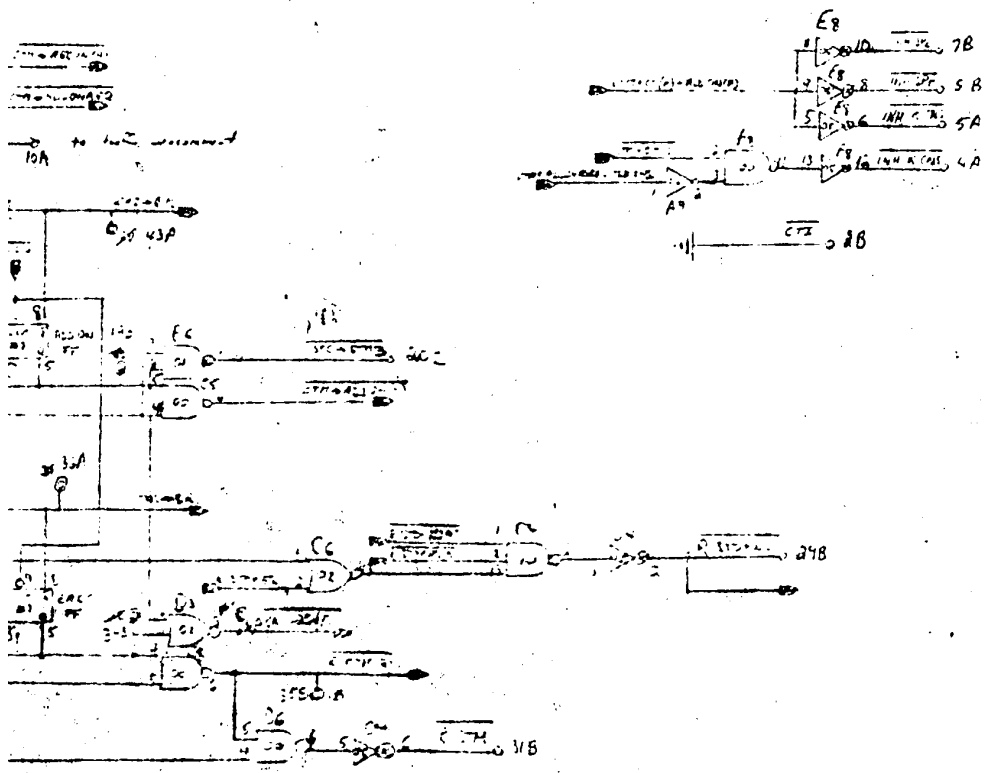
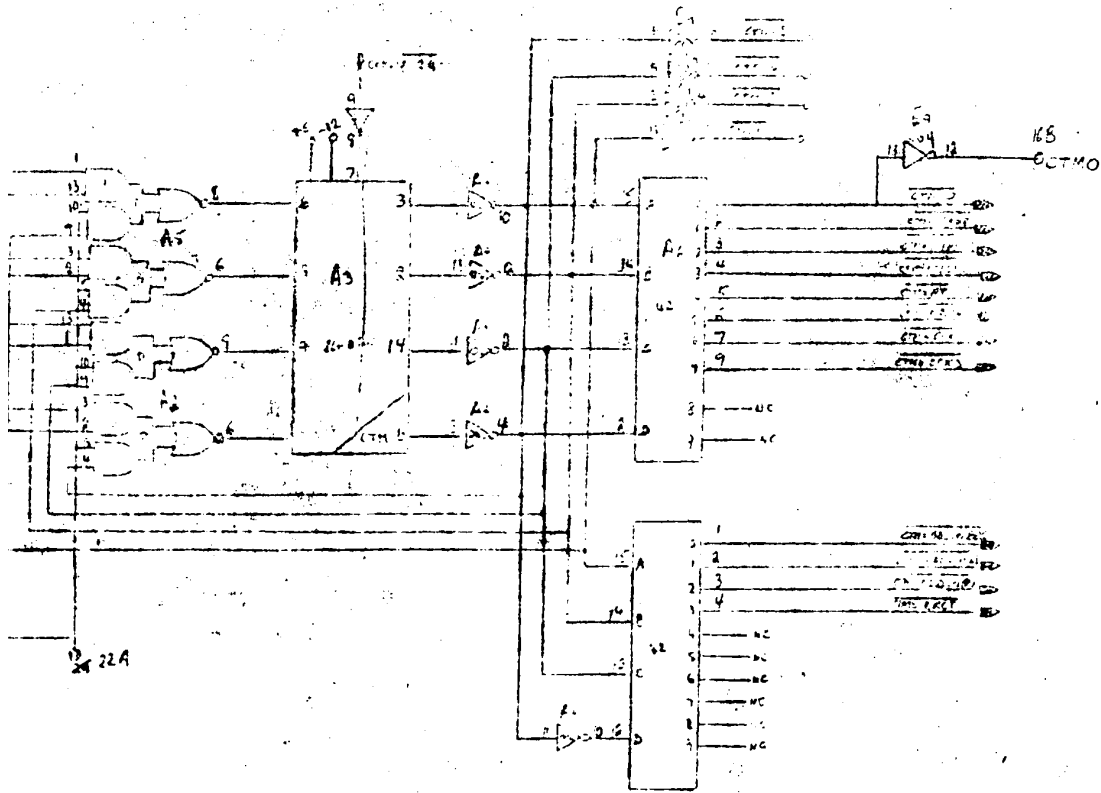
A SAMPLE TEST PROGRAM

This chapter presents an example of part of an actual test program which was written by the author for an SG-1 EPBX logic PCB. The steps from circuit analysis and testing truth-table generation to diagnostic data print-out during testing are shown. Since the complete test program comprises eight truth tables and a very lengthy source program, only the first truth table is given, and should suffice as an example.

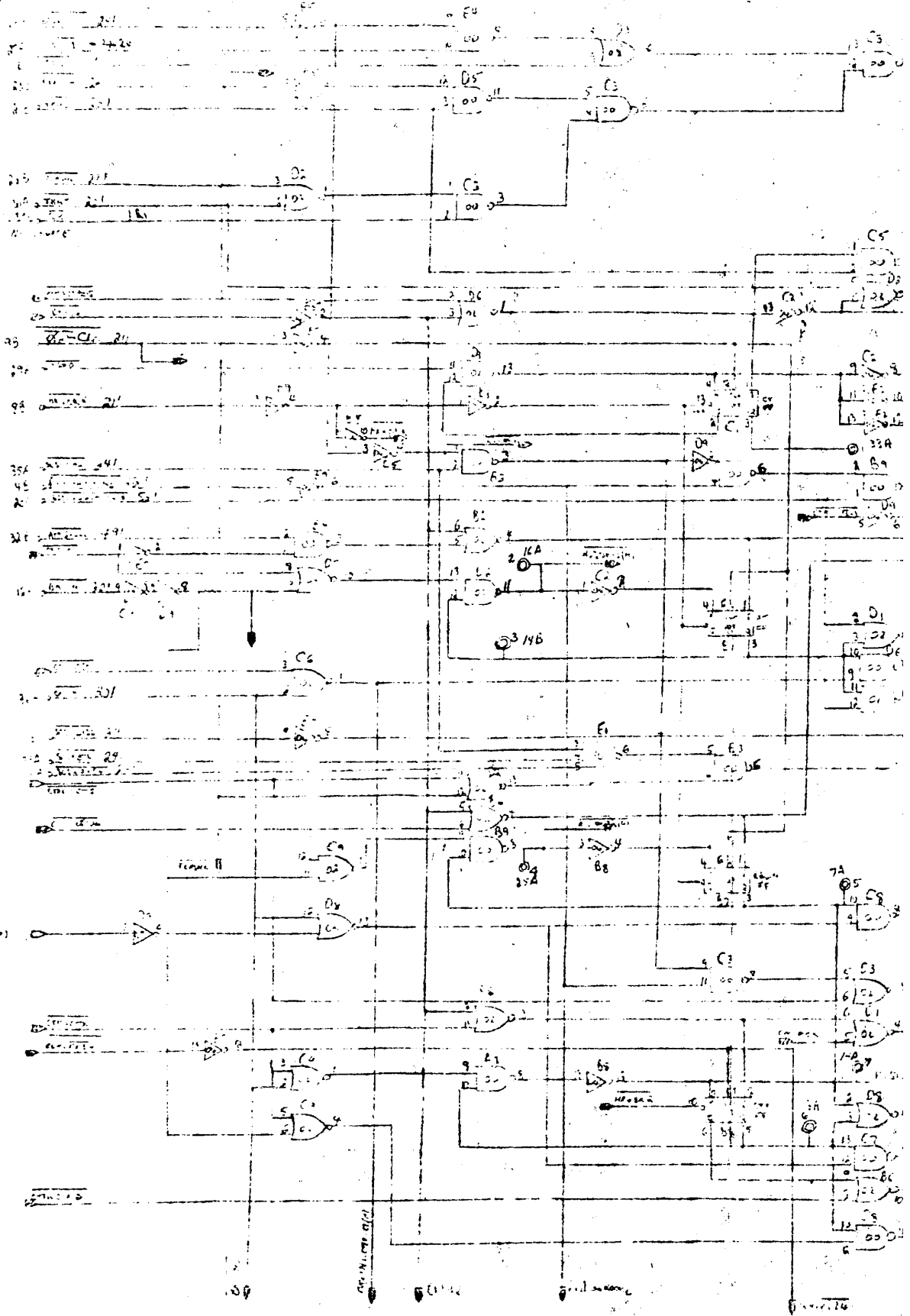
7.1 CIRCUIT UNDER TEST

The circuit schematic for the PCB appears on the next two pages. The key section of the PCB is a re-circulating memory built around a 25-bit shift register fed through a priority encoder. Since the inputs of the memory portion are not readily accessible via access pins, and cannot thus be easily manipulated, it was necessary to treat the entire PCB as a unit. This means that the circuit could not be sectioned for testing. The obvious implications were first of all great difficulties in generating logical testing sequences, and secondly problems in debugging of faulty PCB's. The first is due to the fact that the circuit designer did not furnish adequate functional descriptions of this prototype PCB. It was left to the author to analyze it and try to discover meaningful test sequences. The second





THIS PORTION TO BE CUT OFF IF USED IN BOOK FORM OR BINDER



implication, that of trouble shooting difficulties, was a direct result of the impossibility of sectioning the circuit to produce small testing truthtables.

7.2 TESTING TRUTHTABLES

The truthtable on the following page is the first of eight for the PCB. It is a typical example, since all eight truthtables include all the PCB's input and output pins. The following will aid in understanding the truthtable's format:

The access pins listed under INPUTS serve to apply input levels to the circuit via the test station's drivers. The logic levels at these pins are never checked.

The INTERMEDIATE OUTPUTS are access pins leading into the central portions of the circuit. They are used to check logic levels only. Since these pins are connected to both inputs and outputs of TTL logic devices they cannot be used as drivers, as this would destroy certain types of devices. These outputs are essentially test points and aid in error location.

The access pins in the OUTPUTS column are true circuit outputs in that they have no device inputs connected to them. Again only output levels are checked here.

The column TIME SLOT gives a cumulative account of the number of clock pulses. One or more clock pulses on input pin B9 is necessary to enable progression from one truthtable

line to the next. These are negative-going pulses, involving a 1 - 0 - 1 logic level transition each. They are also identified by T x in the B9 input pin column, where T stands for toggle, and x gives the number of pulses.

The arrows at the bottom of the truth table identify the input and output pins included in the octal format coding of the truth table lines. Those input pins not included are set individually.

7.3 SOURCE TEST PROGRAM

The source test program listing to the end of the first testing truth table appears on the next three pages.

TITLE identifies the test program title which is printed by the OS at various times.

CONFIG A will be translated to tell the OS that this PCB requires both +5 VDC and -12 VDC bias.

ADAPTR 1 means that adapter no. 1 must be connected to the test station.

BIAS gives the settings for the various variable supplies, as identified by the comments following the asterisks. These comments are not translated.

INPUTS lists all the PCB access pins which will have drivers connected to them - i.e. the input pins.

PINSHT identifies the pins which will have an input pin short test performed on them.

INPINS and OUTPINS form the pin connection table for the

```

TITLE   SGI-141631 (ISSUE OF 15.DEC.71)  FEB. 22/72  PHT
CONFIG  A
ADAPTR  1
BIAS    5000 MV   2500 MA           * +5 V SUPPLY
        3500 MV   200 MV           * DRIVER HIGH AND LOW
        800 MV    0 MV            * DETECTOR LOW WINDOW
        4900 MV   2000 MV          * DETECTOR HIGH WINDOW
INPUTS  B9 B8 A24 B24 B27 B23 B21 B28 A30 B39 A29 A35 B4 A20 B32 B12
        B36 B11 A39 A40 B22 A12 A8 A11 A9 A18
PINSHT  B24 B27 B23 B21 B28 A30 B39 B9 A29 B8 A35 B4 A20
        B32 B12 B36 B11 A39 A40 A24 B22 A12 A8 A11 A9 A18
        A42 B40 A41 B38 B37 A2 B10 B13 B15 B18 B26 A23 B33
        A1 A15 A19 A13 B16 B7 B5 A5 A4 A10 B30 B20 B29
        B31
        A33 A32 A38 A16 B14 A37 A25 A7 A14 A3 A26 A17 A22
        B25 B17 A21 B34 A43 B19 A36 B42 A34 B35 A31
TABLE 1 * EXERCISING THE CT FF
INPINS  B9 B8 A24 B24 B27 B23 B21 B28 A30 B39 A29
OUTPINS A33 A16 B14 A25 A7 A3 A14 A26 A32 A37 A38
        B42 A31 B35 A17 A22 B25 A21 B17 B34 A43 B19
        A36 A34 A13 A19 A15 A1 B16 A42 B40 A41 B38
        B37 A2 B10 B13 B15 B18 B26 A23 B33 B7 B5
        A5 A4 A10 B30 B20 B29
.L
SET      +B9 -B8 -A24      * RESET AND CLEAR
CHECK   -A17 -A22
TOGGLE  25      LOW      B9      * INITIAL 25 CLOCK PULSES
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  71445077504177777774 * TIME SLOT 26
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73405077504177777774
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73645077504177777774 * TIME SLOT 28
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73545077504177777774
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  72745077504177777774 * TIME SLOT 30
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73345077504177777774
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  72345077604177777774 * TIME SLOT 32
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  72445077604177777774
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  72245077604177777774
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73445077604177777774 * TIME SLOT 35
.L
TOGGLE  15     LOW      B9
.L
TOGGLE  73445077604177777774 * TIME SLOT 50
.L
TOGGLE  1      LOW      B9
.L
TOGGLE  73445077604177777774

```


TOGGLE 1 LOW B9
 .L 73445077604177777774
 TOGGLE 1 LOW B9
 .L 73445077604177777774 * TIME SLOT 53
 TOGGLE 1 LOW B9
 .L 73445077604177777774
 TOGGLE 1 LOW B9
 .L 73445077604177777774
 TOGGLE 1 LOW B9
 .19 73445077604177777774
 TOGGLE 1 LOW B9
 .20 73445077604070377774 * TIME SLOT 57
 TOGGLE 1 LOW B9
 .L 73455057604073476774
 TOGGLE 1 LOW B9
 .L 73465077604070377774
 TOGGLE 1 LOW B9
 .23 73475057504073776774 * TIME SLOT 60
 TOGGLE 15 LOW B9
 .24 73475057504073776774 * TIME SLOT 75
 TOGGLE 1 LOW B9
 .25 73455057604073476774
 TOGGLE 1 LOW B9
 .26 730450376040703775774
 TOGGLE 1 LOW B9
 .27 73475057504073776774
 TOGGLE 1 LOW B9
 .28 73475057504073776774
 TOGGLE 1 LOW B9
 .29 73475057504073776774 * TIME SLOT 80
 TOGGLE 1 LOW B9
 SET -B36 +B4
 .30 73475057504073776774
 TOGGLE 1 LOW B9
 SET +B22
 .31 734550576141334766174
 TOGGLE 1 LOW B9
 SET -A20 -B4
 .32 734450776040337377674
 TOGGLE 1 LOW B9
 .33 734650776141337776174
 TOGGLE 1 LOW B9
 .34 73465077604070377774 * TIME SLOT 85
 .35
 TOGGLE 1 LOW B9
 CHECK +A33 -A32 +B42 -A17 +A22 -B25 -A21 -B17 +B34 * TIME SLOT 86
 -A13 +A19 +A15 +A1 -B16 +A42 +B40
 +A41 +B38 +B37 +A2 +B13 +B13 +B15
 +B18 +B7 +B5 +A5 +A4 +A10 +B30 -B26
 .36
 TOGGLE 14 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34 * TIME SLOT 100
 -A13 +A19 +A15 +A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30
 .37
 SET -B22

TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 -B34
 -A13 -A19 +A15 +A1 -B16 -A2
 +B7 +B5 +A5 -A4 +A10 +B30

.38
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 +B17 +B34
 +A13 -A19 +A15 +A1 -B16 +A2
 -B7 -B5 -A5 +A4 +A10 +B30

.39
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34
 -A13 +A19 +A15 +A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30

.40
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34
 -A13 +A19 +A15 +A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30

.41
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34
 -A13 +A19 +A15 +A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30

* TIME SLOT 105

.42
 SET +A9 +A18
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34
 -A13 +A19 +A15 +A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30 +B31

.43
 SET -A11 -A9
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 -B34
 -A13 -A19 +A15 +A1 -B16 -A2
 +B7 +B5 +A5 -A4 +A10 +B30 -B31

.44
 SET -B12
 TOGGLE 1 LOW B9
 CHECK -B42 -B25 +A21 -B17 +B34
 +A13 +A19 +A15 -A1 -B16 +A2
 +B7 +B5 +A5 -A4 -A10 -B30 +B31

.45
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 -B17 +B34
 -A13 -A19 +A15 -A1 -B16 +A2
 +B7 +B5 +A5 +A4 +A10 +B30 -B31

.46
 SET -A18
 TOGGLE 1 LOW B9
 CHECK +B42 -B25 -A21 +B17 +B34
 +A13 -A19 +A15 +A1 -B16 +A2
 -B7 -B5 -A5 +A4 +A10 +B30 +B31

* TIME SLOT 110

following octal coding of the truthtable lines.

.L or .Number identifies a new truthtable line number. They start with 1, and are automatically incremented by the translator.

SET means that the following pins are to be set to their indicated level - to a logic 1 in the case of a +, and to a logic 0 in the case of a -.

CHECK is similar to the above except that here the pins are checked for the indicated output level.

TOGGLE 25 LOW B9 means that the pin B9 is to be pulsed 25 times with a negative going pulse.

7.4 TRANSLATOR OUTPUT

The translator output for the preamble and first twelve lines of the first truthtable are listed on the following three pages. This output is presented in the form of octal digits. Certain EC's such as those for table and line numbers have been decoded by this output listing program (which is part of the FACTS DECTRIEVE system) into actual titles such as TABLE x, and LINE y. Furthermore, those EC's which are directly decoded by FTR's (i.e. those which are not word-fills) are indicated in this listing through FTR preceeding them.

The long sequence of octal numbers preceeding TABLE 1 represent the encoded TSI title, input and short test pins, biasing information, configuration, and adapter number.

027107
 112106
 104113
 035126
 034440
 417475
 430406
 402415
 431437
 444445
 531436
 522420
 425510
 424500
 521452
 502443
 441414
 422416
 400477
 451527
 450525
 524401
 471474
 476501
 511426
 520466
 464404
 403411
 515503
 514777
 777777

LINE 1

FTR 042070
 FTR 040027
 FTR 040067
 FTR 054001
 410000
 FTR 160070
 000031

LINE 2

774450
 775441
 777777
 774000
 FTR 160070
 000001

LINE 3

714450
 775041
 777777
 774000
 FTR 160070
 000001

LINE 4

734050
 775041
 777777
 774000

	FTR	160070
LINE 5		000001
		736450
		775041
		777777
		774000
	FTR	160070
LINE 6		000001
		735450
		775041
		777777
		774000
	FTR	160070
LINE 7		000001
		727450
		775041
		777777
		774000
	FTR	160070
LINE 8		000001
		733450
		775041
		777777
		774000
	FTR	160070
LINE 9		000001
		723450
		776041
		777777
		774000
	FTR	160070
LINE 10		000001
		724450
		776041
		777777
		774000
	FTR	160070
LINE 11		000001
		722450
		776041
		777777
		774000
	FTR	160070
LINE 12		000001
		734450
		776041
		777777
		774000
	FTR	160070
		000017

7.5 TEST RESULTS

This section presents a sample of OS output on the Teletype, starting with the original translation of the source test program, followed by diagnostic output according to various data and testing modes, and finally indicating Test Program Verification.

Using the various command tables of chapter 6 it is hoped that the reader will understand the presented listings without requiring much further explanation.

7.5.1 Translation; Initial Run

The listing on the following page indicates the computer-operator communication during the translation of a source TSI.

F-D> indicates that the FACTS DECTRIEVE system is in core. The operator types XTR to request the loading of the translator program. When loaded, the translator requests
TRANSLATION FILE

NAME >

The operator returns the source TSI code; in this case SG6001. Since an older translated version of this TSI is already on the FACTS DECTRIEVE Dectape, this is indicated by asking if it should be deleted. By responding with a SPACE alone the operator indicates the affirmative. The translator next asks if the source TSI is on Dectape. Again, the operator responds in the affirmative. Lastly, the code of the translated TSI is requested and is given as the same code as the source TSI.

F-D> XTR
TRANSLATION FILE
NAME > SG6001
SAME NAME
DELETE? >
DECTAPE INPUT ? >
SOURCE TSI NAME > SG6001
TRANSLATION OK

TRANSLATION FILE
NAME >
F-D> XOS
TSI > SG6001
SGI-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

OK?>
FACTS IB
>S
41 REJECT

The translator indicates that there were no mechanical errors during the translation process, and requests if another TSI is to be translated. This time a response of a SPACE alone indicates the negative, and control is returned to FACTS DECTRIEVE. The operator now requests the loading of the OS by typing XOS. The subsequent loading and running of the TSI in the "Normal Run" mode (the default option) indicates a total of 41 failures.

7.5.2 Diagnostic Output Listings

Since the initial run of the test program indicated a relatively low number of rejects, these are all listed on the following two pages, using the "Reject Coordinates" data mode.

TABLE 7 -0- NOM indicates that the following failures occurred in truthtable 7, with the input level mode 0 (all unused inputs low), and under nominal bias. -1- indicates input level mode 1.

The bottom half of the second page following shows output data under the "Reject Data" mode. The testing mode here is "Stop at First Reject", and hence only the first truthtable line containing failures is presented. Every output pin followed by REJECT has failed.

FACTS 1B
>RC
>S

SGI-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

TABLE 7 -0- NOM

LINE 46 REJECT

OUTLO A2 A4
OUTH1 A1

LINE 50 REJECT

OUTLO A5 B5 B7
OUTH1 A13 B31

LINE 56 REJECT

OUTLO A5 A36 B5 B7
OUTH1 A13 B33

LINE 60 REJECT

OUTLO A17
OUTH1 A22

TABLE 8 -0- NOM

LINE 37 REJECT

OUTH1 A17 A37

LINE 39 REJECT

OUTH1 A22 A37

TABLE 2 -1- NOM

LINE 24 REJECT

OUTH1 A21

TABLE 7 -1- NOM

LINE 46 REJECT

OUTLO A2 A4
OUTH1 A1

LINE 50 REJECT

OUTLO A5 B5 B7
OUTH1 A13 B31

LINE 56 REJECT

OUTLO A5 A36 B5 B7
OUTH1 A13 B33

LINE 60 REJECT

OUTLO A17
OUTH1 A22

TABLE 8 -1- NOM

LINE 37 REJECT

OUTH1 A17 A37

LINE 39 REJECT

OUTH1 A22 A37
41 REJECT
FACTS 1B
>RD
>FR
>S

SG1-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

TABLE 7 -0- NOM

LINE 46

INLO A8 A9 A11 A29 A30 A35 A39 A40 B11 B12 B21 B23 B24 B32
B36 B39
INH1 A12 A18 A20 A24 B4 B8 B9 B22 B27 B28
OUTLO A2 REJECT A3 A4 REJECT A7 A13 A14 A19 A21 A22 A32
A34 A36 A43 B14 B16 B17 B19 B25 B26 B31 B37 B38
OUTH1 A1 REJECT A5 A10 A15 A16 A17 A23 A25 A26 A31 A33 A37
A38 A41 A42 B5 B7 B10 B13 B15 B18 B20 B29 B30 B33 B34
B35 B40 B42

7.5.3 Partial Testing

The next two pages of output listings show the execution of a "Partial Test", with output data presented in the "All Data" mode. Both pages show a partial test of the first three lines of truthtable 1, except that the input level mode differs. No failures occurred for these truthtable lines.

FACTS 1B
 >AD
 >P
 PARTIAL TEST

BIAS > I.L.M.>
 TABLE >1 LINE >1
 TABLE >1 LINE >3

SGI-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

TABLE 1 -0- NOM

LINE 1

INLO A8 A9 A11 A12 A18 A20 A24 A29 A30 A35 A39 A40 B4 B8
 B11 B12 B21 B22 B23 B24 B27 B28 B32 B36 B39
 INHI B9
 OUTLO A17 A22

LINE 2

INLO A8 A9 A11 A12 A18 A20 A29 A30 A35 A39 A40 B4 B11 B12
 B22 B28 B32 B36
 INHI A24 B8 B9 B21 B23 B24 B27 B39
 OUTLO A3 A7 A14 A17 A21 A33 A34 A36 A43 B14 B17 B19
 OUTHI A1 A2 A4 A5 A10 A13 A15 A16 A19 A22 A23 A25 A26 A31
 A32 A37 A38 A41 A42 B5 B7 B10 B13 B15 B16 B18 B20 B25
 B26 B29 B30 B33 B34 B35 B37 B38 B40 B42

LINE 3

INLO A8 A9 A11 A12 A18 A20 A29 A30 A35 A39 A40 B4 B11 B12
 B22 B24 B27 B28 B32 B36
 INHI A24 B8 B9 B21 B23 B39
 OUTLO A3 A7 A14 A17 A21 A33 A34 A36 A43 B14 B17 B19 B25
 OUTHI A1 A2 A4 A5 A10 A13 A15 A16 A19 A22 A23 A25 A26 A31
 A32 A37 A38 A41 A42 B5 B7 B10 B13 B15 B16 B18 B20 B26
 B29 B30 B33 B34 B35 B37 B38 B40 B42

FACTS 1B
>P
PARTIAL TEST

BIAS > I.L.M.>1
TABLE >1 LINE >1
TABLE >1 LINE >3

SGI-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

TABLE 1 -1- NOM

LINE 1

INLO A24 B8
INHI A8 A9 A11 A12 A18 A20 A29 A30 A35 A39 A40 B4 B9 B11
 B12 B21 B22 B23 B24 B27 B28 B32 B36 B39
OUTLO A17 A22

LINE 2

INLO A29 A30 B28
INHI A8 A9 A11 A12 A18 A20 A24 A35 A39 A40 B4 B8 B9 B11
 B12 B21 B22 B23 B24 B27 B32 B36 B39
OUTLO A3 A7 A14 A17 A21 A33 A34 A36 A43 B14 B17 B19
OUTH1 A1 A2 A4 A5 A10 A13 A15 A16 A19 A22 A23 A25 A26 A31
 A32 A37 A38 A41 A42 B5 B7 B10 B13 B15 B16 B18 B20 B25
 B26 B29 B30 B33 B34 B35 B37 B38 B40 B42

LINE 3

INLO A29 A30 B24 B27 B28
INHI A8 A9 A11 A12 A18 A20 A24 A35 A39 A40 B4 B8 B9 B11
 B12 B21 B22 B23 B32 B36 B39
OUTLO A3 A7 A14 A17 A21 A33 A34 A36 A43 B14 B17 B19 B25
OUTH1 A1 A2 A4 A5 A10 A13 A15 A16 A19 A22 A23 A25 A26 A31
 A32 A37 A38 A41 A42 B5 B7 B10 B13 B15 B16 B18 B20 B26
 B29 B30 B33 B34 B35 B37 B38 B40 B42

FACTS 1B
>

7.5.4 Short Test and Miscellaneous Error Diagnostics

The first page following shows diagnostic print-out in the case of input pin short tests. The first indication is for a +5 VDC to -12 VDC access pin short, while the output under INPUT-PIN SHORT TEST indicates various pins seemingly shorted either to ground, or to pin A1.

A subsequent run of the TSI indicates no failures via the diagnostic

OK

The second page following shows set-up error diagnostics. The first resulted from an incorrect TSI code being requested. The next four were caused by the indicated station set-up errors. The last diagnostic was caused by the +5 VDC bias supply being overloaded. Here a "General Release" was automatically issued by the OS, which then returned to the keyboard monitor. After all these errors were cleared, a subsequent run of the TSI passed.

S

SG1-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

B3-TO-B43 SHORT TEST

INLO

OUTH1 B43 REJECT

INPUT-PIN SHORT TEST

INLO A1

OUTH1 A2 REJECT A3 REJECT A4 REJECT A5 REJECT A7 REJECT
A8 REJECT A9 REJECT A10 REJECT A11 REJECT A12 REJECT
A13 REJECT A14 REJECT A15 REJECT A16 REJECT

FACTS 1B

>S

OK

FACTS 1B

>

F-D> XOS
TSI > SG6002
TSI NOT FOUND
TSI > SG6001
SGI-141601 (ISSUE OF 15.DEC.71) FEB. 22/72 PHT

OK?>
STATION ON>
STATION TO AUTO>
PWR ON BOARD>
WRONG ADAPTER>
FACTS 1B
>S
PSI-V FAILURE

FACTS 1B
>S
OK
FACTS 1B
>

7.5.5 Test Program Verification

The next page presents the results of simulating failures on two IC's of the PCB tested. Recalling the pin definitions for test program verification presented in Figure 6.1 it is seen that the particular IC checked has ground on pin 7, inputs on pins 1, 3, 5, 9, 11, and 13. Its outputs are on pins 2, 4, 6, 8, 10, 12. This IC is a hex inverter similar to the hex buffer shown in Figures 2.1 and 2.2.

As can be seen from the listings, the first IC checked resulted in failures for each simulated fault, and the TSI was hence sufficient. But in the case of the next IC, which had the same access pin utilization, it was found that three of the six inverters on the chip were not checked.

A third IC of a different type, but with the same access pin assignments was checked next, and also showed the TSI to be insufficient.

```
FACTS IB
>TVO
IC GND> 4000
IC INPUTS> 521240
IC OUTPUTS> 252500
MORE?>
IC GND>
NO REJ IC PIN 5
NO REJ IC PIN 11
NO REJ IC PIN 13
NO REJ IC PIN 6
NO REJ IC PIN 10
NO REJ IC PIN 12
MORE?>
IC GND>
MORE?>
IC GND>
NO REJ IC PIN 5
NO REJ IC PIN 6
MORE?>N
FACTS IB
>
```

CHAPTER 8
CONCLUSIONS

At the time of this writing in excess of 25 different codes of logic PCB's have been successfully tested and debugged on this test system. It has been found that the software system in general, and the OS in particular are sufficiently comprehensive for logic testing, and still simple enough to use in an industrial environment.

It is hoped that this paper has given some insight into a logic PCB testing system from an operating system software point of view. In spite of the rapidly advancing technology in IC's it is felt that this type of test system philosophy is adequate for the next several years. This can be justified through the versatility of the software system which can easily be expanded to include any type of testing hardware necessary. Furthermore, in a future time-shared environment, radically different pieces of hardware may be operated concurrently in a real-time mode, using the same basic OS philosophy in the software implementation.

The creation of software requires perseverance and a certain degree of artistry. For those seriously investigating the software listings of Appendix A, undoubtedly certain parts of the OS could have been written in another way; simpler perhaps, or to do more things. No apology is offered. If it had to be written again, this author would also change many a thing. Yet it works, and works well.

APPENDIX AOPERATING SYSTEM SOURCE PROGRAM LISTINGS

The source program of the OS was written in Digital Equipment Corporation PDP-9/15 Assembler language, using mnemonics. The interested reader should consult references 6 to 10 inclusive for the PDP-9/15 instruction repertoire.

/SG1-EPBX FACTS 18 OPERATING SYSTEM - FEB. 17/72 - PETER H. TRAU
 /NORTHERN ELECTRIC CO.LTD. - DEPT. K412 - SHEARER ST. MONTREAL
 /*****

/OPERATING SYSTEM STORAGE -- ISS. 11 SEPTEMBER 28/71 PHT
 120/

JMP STRTOP /OPERATING SYSTEM STARTING ADDRESS
 INTAB, \$INTABX /INPUT LEVELS TABLE
 BAR 5
 VARIABLES
 OUTLO, \$OUTLOX /LOW-OUTPUT TABLE
 BAR DECIMAL 8 OCTAL
 VARIABLES
 OUTHI, \$OUTHIX /HIGH-OUTPUT TABLE
 BAR DECIMAL 8 OCTAL
 VARIABLES
 PINCON, \$PINCOY /PIN CONNECTION TABLE
 \$PINCOX
 BAR DECIMAL 72 OCTAL
 VARIABLES
 ACSAVE, 0 /STORES CONTENTS OF AC WHEN INTERRUPT OCCURS
 ADDR.1, 0 /ADDRESSES IN TSITAB FOR PARTIAL TEST OR LOOPING
 ADDR.2, 0
 ADDR.3, 0
 ADDR.4, 0
 CHECK, 0 /CHECK RETURN ADDRESS
 COSTEP, 0 /CONTINUE, STEP RETURN ADDRESS
 DATMOD, 0 /DATA MODE
 ECPNTR, 0 /CURRENT ADDRESS IN TSITAB
 ECSTOR, 0 /CURRENT E.C.
 FSTREJ, 0 /"STOP ON FIRST REJECT" TEST MODE
 FTRNO, 0 /CURRENT FTR NUMBER
 INSET, 0 /INPUT LEVEL MODE
 KBDMOD, 0 /KEYBOARD MODE FLAG
 LINE, 0 /CURRENT TRUTH TABLE LINE NUMBER
 LINERR, 0 /ERRORS PER TRUTH TABLE LINE COUNTER
 LINTIT, 0 /TYPE LINE NUMBER
 LNKC.1, 0 /SUBROUTINE LNKCHK UTILITY REGISTER
 MOTYPE, 0 /TYPE MANUAL OPERATION (MO) MESSAGE
 PARFLG, 0 /PARTIAL TEST OR LOOPING MODE FLAG

```

PINNO, 0 /NUMBER OF ACCESS PINS PER TRUTH TABLE LINE
PWRSET, 0 /BIAS MODE
PWRIT, 0 /TYPE BIAS MODE
RDSTOR, 0 /KEYBOARD READ SUBROUTINES STORAGE REGISTERS
0
0
REQINT, 0 /SEMI-COLON OR INTERRUPT PUSH-BUTTON EVENT FLAG
ROTA.1, 0 /SUBROUTINE ROTATE UTILITY REGISTERS
ROTA.2, 0
SEND.A, 0 /TRANSMISSION SUBROUTINE "SEND" UTILITY REGISTERS
SEND.B, 0
SEND.C, 0
SEND.D, 0
SEND.E, 0
SHTIT, 0 /TYPE SHORT TEST
SINLIN, 0 /SINGLE LINE TESTING MODE FLAG
STATUS, 0 /CONTENTS OF CURRENT STATUS WORD
STPFLG, 0 /STEP MODE FLAG
STPHAN, 0 /STEP HANDLING TRANSMISSION FLAG
STPTOG, 0 /STEP TOGGLE FLAG
TABLE, 0 /TRUTH TABLE NUMBER
TABIT, 0 /TYPE TABLE NUMBER
TEMP.1, 0 /GENERAL PURPOSE UTILITY REGISTERS
TEMP.2, 0
TEMP.3, 0
TEMP.4, 0
TEMP.5, 0
TEMP.6, 0
TEMP.7, 0
TEMP.8, 0
TEMP.9, 0
TEMP10, 0
TEMP11, 0
TEMP12, 0
TEMP13, 0
TEMP14, 0
TEMP15, 0
TEMP16, 0
TEMP17, 0
TEMP18, 0
TEMP19, 0
TEMP20, 0
TOTERR, 0 /TOTAL NUMBER OF ERRORS
TPSW, 0 /SUBROUTINE CHKOUT OUTPUT PINS STATUS STORAGE REGISTER
0 /TYPE TSI TITLE
TSITIT, 0 /TSI VERIFICATION OPTION INDICATOR FLAG
TVOFLG, 0 /TSI VERIFICATION OPTION UTILITY REGISTERS
TVO.S1, 0
TVO.S2, 0
TVO.S3, 0
TVO.SG, 0
TVO.SI, 0
TVO.SO, 0
YCOORD, 0 /ACCESS PIN Y-LEVEL
TSINAM=5300
FACTS=TSINAM+24

```

```

CCTYPE=TSINAM+25
ADPNO=TSINAM+26
INPINS=TSINAM+27
SHTAB=TSINAM+35
PWRWRD=TSINAM+43
TSITAB=TSINAM+63
/*****
/PROGRAM INTERRUPT HANDLER -- ISS.9 SEPTEMBER 20/71 PHT
INT,   DAC ACSAVE      /STORE CONTENTS OF ACCUMULATOR
        CLSF           /SKIP IF R.T.CLOCK FLAG = 1
        JMP .+3
        CLOF           /R.T.CLOCK OFF
        DISMIS-1
        PARSKP        /SKIP IF PARITY FLAG = 1
        JMP .+3
        PARCLR        /CLEAR PARITY FLAG
        JMP SEND.3    /RETURN TO SUBROUTINE SEND
        INTSKP        /SKIP ON TEST SET INTERRUPT
        JMP INT.1
        INTCLR        /CLEAR TEST SET INTERRUPT FLAG
        DZM STPTOG
        ION
        LAM
        DAC STPHAN    /SET STEP HANDLING TRANSMISSION FLAG
        LAC (177500)  /STATUS WORD 2
        JMS SEND
        DZM KBDMOD    /INTERRUPT IS FROM TEST SET CONSOLE
        CHFACT        /READ STATUS WORD 2
        CLLIRTR
        RTR           /BIT 14 INTO LINK:
        SZL           /START?
        JMP STARTS    /YES
        RAR           /NO - BIT 13 INTO LINK:
        SZL           /CONTINUE?
        JMP CONTIN    /YES
        RAR           /NO - BIT 12 INTO LINK:
        SNL           /INTERRUPT P-B?
        JMP STEP+3    /NO - MUST BE STEP
        JMS FLGCLR    /CLEAR FLAG
        LAM           /YES
        DAC REQINT    /FOR USE IN SUBROUTINE SEMCOL
        DZM STPHAN    /CLEAR STEP HANDLING TRANSMISSION FLAG
        DISMIS
INT.1, TSF           /SKIP ON TTY FLAG
        JMP .+3
        TCF           /CLEAR TTY FLAG
        DISMIS-1
        KSF           /SKIP ON KEYBOARD FLAG
        HLT           /INVALID INTERRUPT
        KRB           /READ KEYBOARD BUFFER
        SAD (273)     /SEMI-COLON?
        JMP INT.1-4   /YES
        LAC 0
        IAC
        DAC KBDMOD    /CONTAINS ADDRESS+1 WHERE KEYBOARD ...

```



```

LAC I KBDMOD      /... INTERRUPT OCCURRED
SAD (KRB)         /PICK UP CONTENTS OF THAT ADDRESS
JMP .+3           /WAS IT A "READ KEYBOARD"?
DZM KBDMOD        /NO - HENCE IGNORE THE INTERRUPT
DISMIS            /YES
LAM              /INTERRUPT IS FROM KEYBOARD
DAC KBDMOD
ISZ 0
DISMIS=JMP .      /RETURN ADDRESS "DISMIS"
LAC ACSAVE
ION
DBR
JMP I 0
PAWIOT, 705001
DIV=640323
LACQ=641002
LMQ=652000
IAC=740030
SWHA=742030
CHFACT=705032
PARSKP=703521
PARCLR=703522
INTSKP=703541
INTCLR=703542
/*****
/OPERATING SYSTEM START -- ISS. 11 JAN. 25/72 PHT
STRTOP, CAF      /OPERATING SYSTEM INITIALIZATION
LAC (JMP INT)    /INITIALIZING INTERRUPT HANDLING
DAC I
LAC (HLT)
DAC 21
ION
DZM STPHAN      /CLEAR STEP HANDLING TRANSMISSION FLAG
STRT.1, LAC (5)
DAC STRT.1+10   /INITIAL DIRECTORY BLOCK NUMBER
LAC (STRTT1)    /LF-CR; TSI>; TAB
TYPE
JMS READ        /WAIT FOR TSI CODE (UP TO 8 CHARACTERS)
JMS DTHAND      /READ DIRECTORY BLOCK
JMP STRT.4      /DECTAPE ERROR
TSINAM          /FIRST WORD LOCATION
5              /BLOCK NUMBER
1              /READ 1 BLOCK
LAC (TSINAM)
DAC 10
LAM -76         /DIRECTORY ENTRY COUNTER (63 DIRECTORY ...
DAC TEMP.1     /... ENTRIES PER DIRECTORY BLOCK)
STRT.5, LAC I 10 /PICK UP FIRST 3-CHAR. DIRECTORY ENTRY
SNA            /ANY MORE DIRECTORY ENTRIES?
JMP STRT.4+3   /NO
SAD RDSTOR     /YES - SAME AS FIRST 3 CHARACTERS TYPED?
JMP STRT.7     /YES
ISZ 10        /NO
STRT.6, ISZ 10
ISZ 10

```

```

-----
ISZ TEMP.1      /ALL 63 ENTRIES SEARCHED FOR THIS BLOCK?
JMP STRT.5      /NO - CONTINUE
ISZ STRT.1+10   /YES - INCREMENT DIRECTORY BLOCK NUMBER
LAC STRT.1+10
SAD (12)        /ALL 5 DIRECTORY BLOCKS SEARCHED?
JMP STRT.4+3    /YES
JMP STRT.1+5    /NO - READ NEXT DIRECTORY BLOCK
STRT.7, LAC I 10 /NEXT 3-CHARACTER DIRECTORY ENTRY
SAD RDSTOR+1    /SAME AS TYPED?
SKP            /YES
JMP STRT.6      /NO
LAC RDSTOR+2    /PICK UP LAST 2(OR 3)CHARACTERS TYPED
AND (777700)   /REMOVE POSSIBLE 9TH CHARACTER
SAD I 10       /SAME AS IN 3RD DIRECTORY ENTRY WORD?
SKP           /YES
JMP STRT.6+1    /NO
LAC I 10       /4TH DIRECTORY ENTRY WORD
DAC TEMP.1
AND (1777)     /ISOLATE TSI STARTING BLOCK NUMBER
DAC STRT.2-1
LAC TEMP.1
SWHA
RAR
AND (377)      /ISOLATE NUMBER OF BLOCKS
DAC STRT.2
TAD (-24)     /PERMISSIBLE TSI STORAGE IS 20 BLOCKS
SPA          /STORAGE EXCEEDED?
JMP .+4       /NO
LAC (STRTT6)  /YES - TAB; TSI TOO LONG
TYPE
JMP STRT.1
JMS DTHAND    /READ TSI
JMP STRT.4
TSINAM
0
STRT.2, 0     /NUMBER OF BLOCKS
TYPGR
LAC (TSINAM)
TYPE         /TSI IDENTIFICATION
LAC (STRTT2) /LF-CR; OK? >
TYPE
DZM STPTOG   /CLEAR "STEP" MODE FLAG
JMS READ     /WAIT FOR CONFIRMATION
LAC RDSTOR   /SPACE ALONE:TSI OK; CHAR+SPACE:WRONG TSI
SZA         /SPACE ALONE TYPED?
JMP STRT.1   /NO
JMS STACHK   /CHECK STATION FOR ON; AUTO MODE
JMS GENCLR   /YES - ISSUE GENERAL STATION CLEAR
LAC (177540) /STATUS WORD 3
JMS SEND
LAC STATUS
AND (176)    /BITS 11-16: ADAPTER NUMBER
SAD ADPNO    /CORRECT ADAPTER?
JMP STRT.3   /YES
LAC (STRTT3) /NO - LF-CR: WRONG ADAPTER: LF-CR: >

```

```

TYPE
JMS READ /WAIT FOR A SPACE TO CONTINUE
JMP .-13
STRT.3, DZM COSTEP /CONTINUE, STEP RETURN ADDRESS
DZM CHECK /CHECK RETURN ADDRESS
DZM REQINT /SEMI-COLON OR INTERRUPT PUSH-BUTTON EVENT FLAG
DZM MOTYPE /MO MESSAGE TYPING INDICATOR
DZM DATMOD /"NO DATA" MODE
DZM SINLIN /CLEAR "SINGLE LINE" MODE FLAG
DZM FSTREJ /CLEAR "STOP FIRST REJECT" MODE FLAG
LAM
DAC TSITIT /FOR TSI NAME PRINTING
JMP EXWAIT /TO KEYBOARD WAIT
STRT.4, DAC DTERR /DECTAPE ERROR REGISTER STORAGE
LAC (STRTT4) /LF-CR; DTAPE ERR>
SKP
LAC (STRTT5) /LF-CR; TSI NOT FOUND
TYPE
JMP STRT.1
DTERR, 0
STRTT1, 472423 /LF-CR; TSI>; TAB
114076
440000
STRTT2, 471713 /LF-CR; OK? >
777600
STRTT3, 472722 /LF-CR; WRONG ADAPTER>
171607
400104
012024
052276
0
STRTT4, 470424 /LF-CR; DTAPE ERR>
012005
400522
227600
STRTT5, 472423 /LF-CR; TSI NOT FOUND
114016
172440
061725
160440
0
STRTT6, 442423 /TAB; TSI TOO LONG
114024
171740
141716
070000
/*****
/EXWAIT -- ISS. 9 OCTOBER 4/71 PHT
/WAIT FOR A KEYBOARD COMMAND OR INTERRUPT FROM TEST SET CONSOLE
EXWAIT, LAC (EXTAB-4) /LF-CR; FACTS IB; LF-CR; >
TYPE
JMS READ
LAC (EXTAB-1) /VALID COMMANDS TABLE
DAC I0
LAC I I0
SAD RDSTOR /SAME CHARACTER SEQUENCE?
XCT I I0 /YES - EXECUTE THE FOLLOWING JUMP

```

```

SAD ALLDAT-1 /NO - END OF TABLE?
JMP NOSIR /YES
ISZ 10 /NO
JMP .-6
470601 /LF-CR; FACTS IB; LF-CR; >
32423
406102
477600
EXTAB, 010400 JMP ALLDAT /ALL DATA MODE - "AD"
021500 JMP BIAMOD /BIAS MODE - "BM"
030000 JMP CONTIN /CONTINUE - "C"
062200 JMP FIRSTR /STOP FIRST REJECT - "FR"
072200 JMP GENRLS /GENERAL RELEASE - "GR"
140000 JMP LOOP /LOOP - "L"
160400 JMP NODAT /NO DATA MODE - "ND"
160000 JMP NORMAL /NORMAL RUN - "N"
200000 JMP PARTES /PARTIAL TEST - "P"
220300 JMP REJCOO /REJECT-COORDINATES MODE - "RC"
220400 JMP REJDAT /REJECT-DATA MODE - "RD"
220520 JMP REPEAT /REPEAT - "REP"
222300 JMP REPSEQ /REPEAT SEQUENCE - "RS"
230000 JMP STARTS /START - "S"
232400 JMP STEP /STEP MODE - "ST"
231400 JMP SINGLE /TEST SINGLE LINE - "SL"
160527 JMP STRT.1 /NEW TSI - "NEW"
242617 JMP TVO /TSI VERIFICATION OPTION - "TVO"
360000 JMP 17400 /RETURN TO FACTS DECTRIEVE - "I"
777777 /FIXED ADDRESS FOR CONSTANT "-0"

/*****
/ALLDAT -- ISS.1 FEB.23/71 PHT
/ALL-DATA MODE - "AD"
ALLDAT, LAM
DAC DATMOD
LAC (EXTAB-1)
JMP EXWAIT+1

/*****
/BIAMOD -- ISS.1 FEB.25/71 PHT
/BIAS MODE - "BM"
BIAMOD, JMS BIAREQ /REQUEST BIAS MODE
JMS BIAS /SET UP BIASING
JMP EXWAIT

/*****
/CONTIN -- ISS.2 SEPTEMBER 2/71 PHT
/EXECUTES CONTINUE COMMAND - "C"
CONTIN, JMS STACHK /CHECK STATION FOR "ON", TEST MODES
JMS FLGCLR /CLEAR INTERRUPT FLAGS
LAC COSTEP
SZA /CONTINUE RETURN ADDRESS?
JMP I COSTEP /YES
NOSIR, LAC (NOSIRT) /LF-CR; NO SIR; LF CR
TYPE
JMP EXWAIT
NOSIRT, 471617
402311
224700

/*****
/FIRSTR -- ISS.1 FEB.23/71 PHT

```

```

/SETS STOP-ON-FIRST-REJECT MODE FLAG - "FR"
FIRSTR, ISZ FSTREJ
      JMP ALLDAT+2
/*****
/GENRLS -- ISS.1 FEB.22/71 PHT
/EXECUTES A GENERAL RELEASE COMMAND - "GR"
GENRLS, JMS GENCLR      /CLEAR STATION
      JMP EXWAIT
/*****
/LOOP -- ISS.1 FEB.25/71 PHT
/LOOPING - "L"
LOOP,   DZM PARFLG
      ISZ PARFLG      /SET FOR LOOPING
      JMP PARIES+1
/*****
/NODAT -- ISS.1 FEB.23/71 PHT
/NO-DATA MODE - "ND"
NODAT,  DZM DATMOD
      JMP ALLDAT+2
/*****
/NORMAL -- ISS.2 SEPT. 2/71 PHT
/NORMAL-RUN MODE (NO-DATA, NO STOPS) - "N"
NORMAL, DZM SINLIN
      DZM FSTREJ
      DZM PWRSET
      DZM STPTOG
      JMP NODAT
/*****
/PARTES -- ISS.1 FEB.25/71 PHT
/PARTIAL TEST - "P"
PARTES, DZM PARFLG      /SET FOR PARTIAL TEST
      JMS LOCATE      /DETERMINE BIASING MODE, ADDRESSES
      JMS SHORT      /PERFORM SHORT TESTS, SET UP BIASING
      JMP ECHAND
/*****
/REJCOO -- ISS.1 FEB.23/71 PHT
/REJECT-COORDINATES MODE - "RC"
REJCOO, LAC (2)
      DAC DATMOD
      JMP ALLDAT+2
/*****
/REJDAT -- ISS.1 FEB.23/71 PHT
/REJECT-DATA MODE - "RD"
REJDAT, DZM DATMOD
      ISZ DATMOD
      JMP ALLDAT+2
/*****
/REPEAT -- ISS.1 FEB.22/71 PHT
/REPEATS CURRENT TRUTH-TABLE LINE OUTPUTS CHECK - "REP"
REPEAT, JMS STACHK      /CHECK STATION FOR MODES
      LAC CHECK
      SNA              /CHECK RETURN ADDRESS?
      JMP NOSIR      /NO
      JMP I CHECK
/*****
/REPSEQ -- ISS.1 OCT.4/71 PHT

```

```

/REPEAT SEQUENCE - "RS"
REPSEQ, JMS STACHK      /CHECK STATION FOR MODES
        LAC (TSITAB-1)
        DAC ECPNTR      /RESET TO TOP OF TSI
        DZM PARFLG     /RESET FOR PARTIAL TEST
        JMP CONTIN+1

```

```

/STARTS -- ISS.2 MARCH 21/71 PHT
/EXECUTES START COMMAND - "S"
STARTS, JMS SHORT      /PERFORMS INPUT SHORT TESTS, ...
        LAM            /... SETS UP BIASING
        DAC PARFLG
        JMP ECHAND

```

```

/STEP -- ISS.4 SEPTEMBER 20/71 PHT
/STEP (SINGLE TRANSMISSION) MODE - "ST"
STEP,   JMS STACHK     /ENTRY VIA TELETYPE:CHECK STEP TOGGLE
        ISZ STPTOG     /STEP TOGGLE UP?
        JMP NOSIR      /NO
        JMS FLGCLR     /CLEAR STEP INTERRUPT FLAG
        DZM STPHAN     /CLEAR STEP HANDLING TRANSMISSION FLAG
        LAM
        DAC STPTOG
        ISZ STPFLG     /INITIAL STEP ACTION?
        SKP            /YES
        JMP I COSTEP   /NO-CONTINUE TO NEXT TRANSMISSION (SUB.SEND)
        LAC KBDMOD     /INITIAL STEP REQUEST FROM TELETYPE?
        SZA
        JMP ECHAND     /YES - CONTINUE PROCESSING
        LAC 0          /NO - REQUEST FROM STEP P-B
        SAD (LAC READWT+1) /WERE WE IN THE KEYBOARD WAIT LOOP?
        JMP ECHAND     /YES - CONTINUE PROCESSING
        DISMIS        /NO - HANDLE INTERRUPT NORMALLY

```

```

/SINGLE -- ISS.1 FEB.23/71 PHT
/SINGLE LINE MODE - "SL"
SINGLE, ISZ SINLIN
        JMP ALLDAT+2

```

```

/TVO -- ISS.3 SEPTEMBER 30/71 PHT
/TSI VERIFICATION OPTION
/IC GROUND, INPUT AND OUTPUT PINS ARE ENTERED AS OCTAL NUMBERS FROM
/THE TELETYPE. PIN NUMBERS ARE DEFINED FROM 1-16 AND CORRESPOND DIRECTLY
/TO AC0-15. ONLY THOSE PINS WHICH ARE USED IN THE CIRCUIT SHOULD BE
/DEFINED. POWER PINS MUST NOT BE DEFINED, AND GROUND PINS MUST ALSO
/NOT BE REDEFINED AS INPUT OR OUTPUT PINS.
/EVERY DEFINED IC ACCESS PIN IS ISOLATED SINGLY, AND THE ENTIRE
/TSI IS EXECUTED FOR THIS CONDITION, WITH NOMINAL BIAS ONLY, BUT WITH
/UNUSED INPUTS ONCE LOW AND ONCE HIGH IF SO SPECIFIED IN THE TSI
/BEFORE THE FAULT SIMULATION THE TSI IS EXECUTED ONCE TO MAKE SURE THAT
/THAT THERE ARE NO REJECTS, FOR EACH IC BEING HANDLED.
TVO,   DZM PVRSET     /NOMINAL BIAS ONLY
        JMS SHORT     /PERFORM INPUT PIN SHORT TESTS
        LAM
        DAC PARFLG
        RCL

```

```

DAC TVOFLG           /SET TVO FLAG FOR INITIAL RUN
LAC (TVOT00)        /CR-LF; IC GND>; TAB
TYPE
JMS READOC          /READ IC GROUND PIN(S) DEFINITION (OCTAL)
LAC RDSTOR
SNA                 /SPACE ALONE TYPED?
JMP TVO.01-5        /YES - SAME DEFINITIONS AS BEFORE.
DAC TVO.SG          /STORE IC GROUND PIN(S) DEFINITION
LAC (TVOT01)        /CR-LF; IC INPUTS>; TAB
TYPE
JMS READOC          /READ IC INPUTS DEFINITION (OCTAL)
LAC RDSTOR
SNA                 /SPACE OR 0 TYPED?
JMP EXWAIT          /YES - RETURN TO KEYBOARD WAIT
DAC TVO.SI          /STORE INPUTS DEFINITION
LAC (TVOT02)        /CR-LF; IC OUTPUTS>;TAB
TYPE
JMS READOC          /READ IC OUTPUTS DEFINITION (OCTAL)
LAC RDSTOR
SNA                 /SPACE OR 0 TYPED?
JMP EXWAIT          /YES
DAC TVO.S0          /STORE OUTPUTS DEFINITION
LAC (176540)        /"A" AND "B" RELAYS CLEAR
JMS SEND
DZM TOTERR          /CLEAR ERROR REGISTER
JMP ECHAND          /PERFORM INITIAL RUN
DZM TVO.S3          /SET FLAG FOR IC INPUT PINS
TVO.01, LAC TVO.SI   /PICK UP INPUT PIN NUMBERS WORD
DZM TVO.S2          /PIN COUNTER
DAC TVO.SI
LAC (TSITAB-1)
DAC ECPNTR          /RESET TO TOP OF TSI
LAC (176540)        /"A" AND "B" RELAYS CLEAR
JMS SEND
DZM TOTERR          /CLEAR ERROR REGISTER
LAC TVO.SG          /GND PIN(S) DEFINITION
JMS LNKCHK          /GET PIN NUMBER
JMP .+5             /DONE
LAC TEMP16          /PIN NUMBER
TAD (20)
JMS TVOSUB          /OPERATE IC GND PIN "B" RELAY
JMP LNKCH1          /REPEAT FOR ANY OTHER GROUND PIN
ISZ TVO.S2
LAC TVO.S2
SAD (21)            /ALL PINS HANDLED?
JMP TVO.02          /YES
LAC TVO.S1          /NO
RAL
SNL                 /THIS PIN USED?
JMP TVO.01+2        /NO
DAC TVO.S1          /YES
LAC TVO.S2
DAC TEMP.1
JMS TVOSUB          /OPERATE CORRESPONDING "A" RELAY
LAC TVO.S3
SZA                 /IC INPUT PIN?
JMP ECHAND          /NO - EXECUTE TSI

```

```

LAC TEMP.1
TAD (20)
JMS TVOSUB          /YES - OPERATE CORRESPONDING "B" RELAY
JMP ECHAND          /EXECUTE TSI
TV0.02, LAC TV0.S3
SZA                /FINISHED WITH ALL IC INPUTS?
JMP TV0.03         /NO - FINISHED WITH OUTPUTS AS WELL!
ISZ TV0.S3        /YES - SET FLAG FOR IC OUTPUTS
LAC TV0.S0        /PICK UP OUTPUT PINS DEFINITION
JMP TV0.01+1
TV0.03, LAC (TVOT03) /CR-LF; MORE?>
TYPE
JMS READ
LAC RDSTOR
SNA                /FINISHED WITH "TVO"?
JMP TV0+2         /SPACE ALONE TYPED: NOT FINISHED!
DZM TVOFLG       /CHARACTER+SPACE TYPED: DISABLE TVO FLAG
JMP EXWAIT
TVOSUB, 0
CLLROT            /PIN COUNTER SHIFTED TO BECOME X-COORDINATE
-5
TAD (117740)     /ADD Y-COORDINATE TO CREATE CODE POINT
JMS SEND         /TRANSMIT "A" OR "B" RELAY CODE POINT
LAM -62
JMS WAIT         /WAIT 5 MS
JMP I TVOSUB
TVOT00, 471103   /CR-LF; IC GND>; TAB
400716
047644
0
TVOT01, 471103   /CR-LF; IC INPUTS>; TAB
401116
202524
237644
0
TVOT02, 471103   /CR-LF; IC OUTPUTS>; TAB
401725
242025
242376
440000
TVOT03, 471517   /CR-LF; MORE?>
220577
760000
/*****
/ECHAND -- ISS.3 MARCH 2/71 PHT
/E.C. HANDLER
ECHAND, JMS SEMCOL /ENABLE SEMI-COLON INTERRUPT
JMS NEXTEC       /GET NEXT E.C.
DAC ECSTOR      /STORE E.C.
CLLROT         /ROTATE FIR NO. INTO BITS 12-17
-7
AND (77)
DAC FIRNO      /STORE FIR CODE NO
TAD (XCT .+3)
DAC .+2
LAC ECSTOR
0

```



```

JMP LOCA.C-7      /IN SUBROUTINE LOCATE
JMP LOCA.C-7
JMP LOCA.C-7
JMP FTR4          /F.A.W.
JMP FTR5          /P.A.W.
JMP LOCA.C-7
JMP LOCA.C-7
JMP FTR8          /MANUAL OPERATION
JMP FTR9          /WAIT
JMP LOCA.C-7
JMP FTR11         /GENERAL RELEASE AND END, ALL INPUTS HIGH
JMP FTR12         /PIN CONNECTION TABLE, TRUTH TABLE NO.
JMP FTR13         /OCTAL FORMAT HANDLER, TRUTH TABLE LINE NO
JMP FTR14         /INPUT PIN TOGGLE OR SEQUENCING
JMP LOCA.C-7

/*****
/FTR4 HANDLER -- ISS. 3 APRIL 7/71 PHT
/F.A.W.
FTR4,    JMS XYSTOR      /STORE X,Y COORDINATES
          AND (2000)     /ISOLATE BIT 7: CONNECT OR RELEASE
          DAC TEMP.9
          LAC ECSTOR
          AND (4000)     /ISOLATE BIT 6: INPUT OR OUTPUT
          SZA           /INPUT OR CODE POINT?
          JMP FTR4.2     /NO: OUTPUT!
          LAC TEMP.4     /X-Y COORDINATES
          TAD (-137)
          SMA           /CODE POINT?
          JMP FTR4.1     /YES
FTR4.3,  LAC (INTAB)    /NO-ACCESS POINT
          JMS UPDATE    /MODIFIES TABLE INTAB
          LAC TEMP.9
          JMS INMOD1    /LOAD MATRIX AND TRANSMIT IF NO MORE...
          LAC I ECPNTR  /... CONNECTS FOLLOW
          JMP ECHAND+2
FTR4.1,  LAC TEMP.4     /CODE POINT: PICK UP X-Y COORDINATES
          CLLROT        /SHIFT INTO BITS 3-12
          -5            /FIXED LOCATION FOR CONSTANT "-5"
          XOR (100000)  /ADD BIT 2 (F.A.W.)
FTR4.4,  DAC TEMP.4
          LAC TEMP.9
          SZA           /RELEASE?
          JMP .+4        /NO-CONNECT
          LAC TEMP.4     /YES - ADD BIT 15 FOR RELEASE
          XOR (4)
          SKP
          LAC TEMP.4
          JMS SEND      /TRANSMIT F.A.W. FOR CODE POINT
          LAC FTRNO
          SAD (5)
          JMP FTR5.1
          JMP ECHAND
FTR4.2,  LAC TEMP.9     /OUTPUT CHECK: OUTLO OR OUTHI
          SZA           /OUTLO?
          JMP .+3        /NO
          LAC (OUTLO)   /YES
          SKP

```

```

LAC (OUTH1)
JMS UPDATE
ISZ ECPNTR
LAC I ECPNTR /PICK UP NEXT E.C.
AND (774000) /ISOLATE BITS 0-6
SAD (44000) /OUTPUT FTR4?
JMP FTR4.3+4 /YES
SAD (54000) /OUTPUT FTR 5?
JMP FTR4.3+4 /YES
LAC (FTR4.5)
DAC CHECK /CHECK RETURN ADDRESS
IAC
DAC COSTEP /CONTINUE, STEP RETURN ADDRESS
FTR4.5, JMS CHKOUT
JMS OUTCLR /CLEAR OUTPUT TABLES OUTLO AND OUTHI
DZM COSTEP
DZM CHECK
LAM
TAD ECPNTR
DAC ECPNTR /DECREMENTS ECPNTR
JMS NEXTEC /DONE THIS WAY TO ALLOW HANDLING ...
JMP FTR4.3+4 /... OF PARTIAL TESTING
/*****
/FTR5 HANDLER -- ISS. 2 AUGUST 30/71 PHT
/P.A.W.
FTR5, AND (77) /ISOLATE Y-COORDINATE
DAC YCOORD
LAC ECSTOR
AND (2000) /ISOLATE RELEASE (CONNECT) BIT 7
DAC TEMP.9
LAC ECSTOR
AND (4000) /ISOLATE INPUT (CODE POINT) OR OUTPUT BIT 6
SZA /INPUT OR CODE POINT?
JMP FTR4.2 /NO-OUTPUT (FTR4 HANDLER)
LAC YCOORD
TAD FTR4.1+2 /-5
SPA /CODE POINT?
JMP FTR4.3 /NO (FTR4 HANDLER)
LAC YCOORD /YES
SWHA /SHIFT Y-COORDINATE INTO BITS 3-8
JMP FTR4.4 /FTR4 HANDLER
FTR5.1, JMS NEXTEC /PICK UP WORDFILL (CODE POINT HANDLING)
XOR (2) /ADD BIT 16 - WORDFILL
JMS SEND /TRANSMIT UNCONDITIONALLY
JMP ECHAND
/*****
/FTR8 HANDLER -- ISS.5 AUGUST 30/71 PHT
/MANUAL OPERATION
FTR8, AND (2000) /ISOLATE BIT 7
SZA /MESSAGE TO BE TYPED FOR EVERY CCT?
JMP FTR8.3 /NO
LAC (FTR8T1)
TYPE /2LF-CR; MO; TAB
ISZ ECPNTR
LAC ECPNTR
TYPE /TYPE THE MESSAGE
LAC TEMP.1

```

```

IAC
DAC ECPNTR          /RESTORE ECPNTR TO LAST WORDFILL IN MESSAGE
TYP CR
FTR8.1, LAC (117100) /TURN ON THE MO LIGHT
JMS SEND
LAC (FTR8.2)
DAC COSTEP         /CONTINUE, STEP RETURN ADDRESS
JMP EXWAIT        /GO TO KEYBOARD WAIT
FTR8.2, LAC (117104) /TURN OFF THE MO LIGHT
JMS SEND
JMP ECHAND
FTR8.3, LAC ECSTOR  /MESSAGE TO BE TYPED ONLY ONCE
AND (1777)        /ISOLATE BITS 8-17
DAC TEMP.4        /CONTAINS MESSAGE IDENTIFICATION BIT
LAC MOTYPE        /MO TYPING INDICATOR
AND TEMP.4
SZA               /WAS MESSAGE ALREADY TYPED?
JMP FTR8.4        /YES
LAC MOTYPE        /NO
XOR TEMP.4
DAC MOTYPE        /SET BIT IN MOTYPE
JMP FTR8+3
FTR8.4, JMS NEXTEC /MESSAGE ALREADY TYPED: SKIP WORDFILLS
SZA              /END-OF-MESSAGE WORDFILL?
JMP FTR8.4        /NO
JMP ECHAND        /YES
FTR8T1, 474715
          174400
/*****
/FTR9 HANDLER -- ISS.1 FEB.5/71 PHT
/WAIT
/FOLLOWED UNCONDITIONALLY BY A WORDFILL GIVING THE COMPLEMENT OF THE..
/..NUMBER OF TIME UNITS OF WAIT (1 T.U. = 0.1 MS)
FTR9,   JMS NEXTEC  /PICK UP WORDFILL
        JMS WAIT    /WAIT SPECIFIED TIME
        JMP ECHAND
/*****
/FTR11 HANDLER -- ISS.9 OCTOBER 7/71 PHT
/GENERAL RELEASE; END; REPEAT WITH ALL INPUTS HIGH
FTR11,  AND (2000)  /BIT 7: GENERAL RELEASE; END
        SZA        /END?
        JMP F11..2  /NO - GR
        LAC TVOFLG
        SZA        /TSI VERIFICATION?
        JMP F11..1-12 /YES - DO ONLY NOMINAL BIAS
        ISZ PWRSET  /BIAS MODE INDICATOR
        LAC PWRSET
        SAD (5)    /5 BIASING MODES HANDLED?
        JMP .+7    /YES
        CLL!RTL    /X 4
        TAD (PWRWRD) /ADDRESS OF BIAS TABLE
        DAC TEMP.1
        LAC I TEMP.1
        SZA        /IS THERE ANOTHER BIAS MODE?
        JMP F11..1  /YES
        DZM PWRSET  /NO - RESTORE FOR NOMINAL BIAS

```

```

LAC ECSTOR           /PICK UP E.C.
AND (4000)           /BIT 6: ALL INPUTS LOW (OR HIGH)
SNA                   /ALL INPUTS HIGH?
JMP F11..1+5         /NO - ALL INPUTS LOW
LAC INSET             /YES ...
SZA                   /... DID WE ALREADY DO ALL INPUTS HIGH?
JMP F11..1+5         /YES
LAM                   /NO
DAC INSET
F11..1, JMS BIAS      /SET NEW BIASING MODE
LAC (TSITAB-1)
DAC ECPNTR           /RESET TO TOP OF TSI
DZM TABTIT           /FOR TABLE NUMBER PRINTING
JMP ECHAND
DZM INSET             /RESET FOR ALL UNUSED INPUTS LOW
LAC TVOFLG
SZA                   /TSI VERIFICATION?
JMP OK+1             /YES
TYPGR                /FINISHED!
JMS GENCLR           /GENERAL CLEAR
LAC TOTERR
SZA                   /ANY ERRORS?
JMP .+6              /YES
LAC (OK)
TYPE                 /NO - "OK"
LAC (117400)         /TURN-ON "PASS" LIGHT
JMS SEND
JMP F11..2
JMS TYPDEC
TYPSP
LAC (CHKO.T)        /REJECT
TYPE
LAC (117340)         /TURN-ON "REJECT" LIGHT
JMS SEND
F11..2, LAC (117140) /TURN ON "END OF TEST" LIGHT
JMS SEND
OK, JMP EXWAIT
171300
LAC TVOFLG
SAD (-1)             /INITIAL RUN?
JMP F11..3           /YES
LAC TOTERR           /NO - FAULT SIMULATION RUN
SZA                   /ANY ERRORS?
JMP TVO.01+3        /YES - OK
LAC (F11..T)        /NO: CR-LF; NO REJ IC PIN
TYPE
LAC TVO.S2           /IC PIN NO.
JMS TYPDEC
JMP TVO.01+3
F11..3, LAC TOTERR   /INITIAL RUN:
SZA                   /ANY REJECTS?
JMP F11..2-6        /YES - ABORT FAULT SIMULATION RUNS
ISZ TVOFLG           /NO - OK - SET FLAG FOR FAULT SIMULATION RUNS
JMP TVO.01-1
F11..T, 471617       /CR-LF; NO REJ IC PIN; SPACE
402205
124011

```

034020
111640
0

/*****

/FTR12 HANDLER -- ISS. 4 SEPTEMBER 22/71 PHT

/PIN CONNECTION TABLE; TRUTH TABLE NUMBER

FTR12, AND (77) /STORE TRUTH TABLE NO
 DAC TABLE
 LAC ECSTOR
 AND (2000)
 SZA /TRUTH TABLE NO.INDICATOR ONLY?
 JMP F12..2+1 /YES
 DZM PINNO /PIN COUNTER REGISTER
 LAC (PINCON)
 DAC TEMP10 /ADDRESS IN PIN CONNECTION TABLE
 F12..1, JMS NEXTEC /PICK UP NEXT WORDFILL
 SAD ALLDAT-1 /END OF PIN-CONNECTION WORDFILLS?
 JMP F12..2 /YES
 SWHA /SWAP HALVES OF WORDFILL
 AND (777) /ISOLATE BITS 9-17
 DAC I TEMP10 /LOAD INTO TABLE PINCON
 ISZ TEMP10
 ISZ PINNO
 LAC I ECPNTR /PICK UP SAME WORDFILL
 AND (777)
 SAD (777) /BITS 9-17 = 1?
 JMP F12..1 /YES
 DAC I TEMP10
 ISZ TEMP10
 ISZ PINNO
 JMP F12..1
 F12..2, DAC I TEMP10
 LAC (INTAB-1)
 DAC 10
 LAC INSET /INPUT LEVEL SETTING MODE
 AND (777774) /REMOVE BITS 16 AND 17
 DAC I 10 /SET ALL INPUTS EITHER LOW OR HIGH
 LAC 10
 SAD (INTAB+5) /ENTIRE INPUT LEVEL TABLE HANDLED?
 JMP ECHAND /YES
 JMP F12..2+3 /NO

/*****

/FTR13 HANDLER -- ISS. 2 MARCH 15/71 PHT

/OCTAL FORMAT; TRUTH TABLE LINE NUMBER

FTR13, AND (377) /ISOLATE TRUTH TABLE LINE NO
 DAC LINE#
 LAC ECSTOR
 AND (7400) /ISOLATE WORDFILL COUNTER
 SNA
 JMP ECHAND /NO WORDFILLS FOLLOWING THIS FTR 13
 LAC (PINCON)
 DAC TEMP11
 DZM TEMP12 /COUNTS NO. OF PINS PROCESSED
 F13..1, LAM -21
 DAC TEMP13 /PINS PROCESSED PER WORDFILL
 JMS NEXTEC
 DAC TEMP14 /STORE WORDFILL

```

F13..2, LAC I TEMP11 /PICK UP CONTENTS OF PINCON
        AND (377) /ISOLATE THE PIN'S X-Y COORDINATES
        JMS XYSTOR
        LAC I TEMP11
        AND (400) /ISOLATE BIT 9
        SZA /INPUT PIN?
        JMP F13..3 /NO-OUTPUT PIN
        LAC (INTAB)
        JMS UPDATE /UPDATE INPUT LEVELS TABLE
        LAC TEMP14 /PICK UP WORDFILL
        RAL
        DAC TEMP14
        SNL /SETLO?
        JMS UPDAT1 /YES
F13..4, ISZ TEMP11 /INCREMENT ADDRESS IN PINCON
        ISZ TEMP12
        LAC TEMP12
        SAD PINNO /ALL PINS PROCESSED?
        JMP F13..5 /YES
        ISZ TEMP13 /ENTIRE WORDFILL HANDLED?
        JMP F13..2 /NO
        JMP F13..1 /YES-GET NEXT WORDFILL
F13..5, DZM YCOORD
        LAC (INTAB)
        TAD YCOORD
        DAC TEMP.7 /SET UP INPUT LEVELS AT MATRIX
        JMS INMOD
        ISZ YCOORD
        LAC YCOORD
        SAD (6) /ALL 6 Y-LEVELS OF INPUTS COVERED?
        SKP
        JMP F13..5+1 /NO
        LAC (176140) /YES
        JMS SEND /TRANSMIT LATCH ENABLE CODE POINT
        LAC (F13..6)
        DAC CHECK /CHECK RETURN ADDRESS
        IAC
        DAC COSTEP /CONTINUE, STEP RETURN ADDRESS
F13..6, JMS CHKOUT /CHECK OUTPUTS
        JMS OUTCLR
        DZM COSTEP
        DZM CHECK
        JMP ECHAND
F13..3, LAC TEMP14
        RAL
        DAC TEMP14
        SZL /OUTLO
        JMP .+3 /NO
        LAC (OUTLO) /YES
        SKP
        LAC (OUTH1)
        JMS UPDATE /UPDATE OUTPUT LEVEL TABLES
        JMP F13..4

```

/*****

/FTR14 HANDLER -- ISS. 4 SEPTEMBER 28/71 PHT
 /INPUT PIN TOGGLE; SEQUENTIAL INPUT SETTING

```

FTR14,  JMS XYSTOR      /X=Y COORD. INTO TEMP.4, Y-COORD. INTO YCOORD
        ROTATE          /BIT 7 INTO BIT 17
        -11             /FIXED ADDRESS FOR CONSTANT "-11"
        DAC TEMP.9      /SETLO (SETHI) INDICATOR
        AND (2)         /ISOLATE BIT 6 (NOW IN BIT POSITION 16)
        SZA             /INPUT PIN TOGGLE?
        JMP F14..2      /NO-SEQUENTIAL INPUT SWITCHING
        JMS NEXTEC      /YES-PICK UP FOLLOWING WORDFILL
        CLLIRAL        /MULTIPLY BY 2
        CMA
        DAC TEMP11      /STORES NO. OF TRANSITIONS
F14..1,  ISZ TEMP11      /ALL TOGGLES DONE?
        SKP
        JMP ECHAND      /YES
        LAC (INTAB)     /NO
        JMS UPDATE      /UPDATE INPUT LEVEL TABLE
        LAC TEMP.9
        AND (1)
        JMS INMOD1      /TRANSMIT
        ISZ TEMP.9
        JMP F14..1
F14..2,  DZM FTRNO      /TO ASSURE SEQUENTIAL SETTING IN FTR4
        LAC TEMP.9
        AND (1)         /ISOLATE SETHI-SETLO BIT
        DAC TEMP.9
        JMP FTR4.3

```

***** UTILITY SUBROUTINES *****

/BIAREQ -- ISS. 3 SEPTEMBER 3/71 PHT

/USAGE: JMS BIAREQ

/SETS REG. PWRSET FOR BIASING MODE AS SPECIFIED ON KEYBOARD:

/0 (OR SPACE) = NOMINAL

/1-4 = MAR-1 TO MAR-4 RESPECTIVELY

/SETS REGISTER INSET FOR INPUT LEVEL MODE AS SPECIFIED ON KEYBOARD:

/0 (OR SPACE) = ALL UNUSED INPUTS LOW

/1 = ALL UNUSED INPUTS HIGH

BIAREQ, 0

LAC (BIARET) /LF-CR; BIAS >

TYPE

JMS READNO /WAIT FOR BIAS SPEC. NO.

LAC RDSTOR

DAC PWRSET

CLL!RTL /X4

TAD (PWRWRD)

DAC TEMP.1

LAC I TEMP.1

SNA /IS THE SPECIFIED BIAS AVAILABLE?

JMP NOSIR /NO

LAM

DAC INSET /ASSUME ALL UNUSED INPUTS HIGH

LAC (BIARET+3) /TAB; I.L.M.>

TYPE

JMS READNO /WAIT FOR INPUT LEVEL MODE NUMBER

LAC RDSTOR

SNA /ALL LOW SPECIFIED?

DZM INSET /YES

```

      JMP I BIAREQ
BIARET, 470211
      12340
      760000
      441156
      145615
      567600

```

```

/*****

```

```

/BIAS -- ISS. 7 OCTOBER 7/71 PHT
/USAGE: JMS BIAS
/SETS THE BIASING SUPPLIES ACCORDING TO THE BIAS MODE SPECIFIED ...
/... IN REGISTER PWRSET.

```

```

BIAS, 0
      LAC (76000)   /PARTIAL ADDRESS WORD AND WORDFILL ...
      JMS SEND      /TO CLEAR PSI, DRIVER VOLTAGES ...
      LAC (16002)   /AND DETECTOR VOLTAGES
      JMS SEND
      LAC PWRSET    /BIAS MODE
      CLLIRTL      /X 4
      TAD (PWRWRD)
      DAC TEMP.1
      LAC INSET
      DAC .+2
      JMS INLOAD    /SET INPUT LEVELS TABLE ACCORDING ...
      0             /... TO INPUT LEVELS FLAG INSET
      LAC CCTYPE
      SZA           /NORMAL PWR, GROUND PINS CIRCUIT?
      JMP .+3       /NO - DO NOT ENABLE PS2
      LAC (117600)  /PS2 ENABLE
      JMS SEND
      LAC (16000)   /P.A.W. FOR PSI
      JMS SEND
      LAC I TEMP.1  /WORDFILL FOR PSI
      TAD (2)
      JMS SEND
      ISZ TEMP.1
      LAC (14000)   /P.A.W. FOR DRIVER VOLTAGES
      JMS SEND
      LAC I TEMP.1  /WORDFILL FOR DRIVER VOLTAGES
      TAD (2)
      JMS SEND
      LAM -4704
      JMS WAIT      /250 MS DELAY
      JMS BIASUB    /CHECK STATION FOR POWER SUPPLY FAILURE
      JMP I BIAS

```

```

/*****

```

```

/BIASUB -- ISS.4 OCTOBER 7/71 PHT
/USAGE: JMS BIASUB
/CHECKS STATION POWER SUPPLIES STATUS AND IDENTIFIES FAILURES
BIASUB, 0
      LAC (177000)  /STATION STATUS WORD CODE POINT
      JMS SEND
      LAC STATUS
      AND (536)     /NOTE: PSI-C AND PS2-C CURRENTLY NOT CHECKED!
      SNA          /ANY STATION POWER SUPPLIES DOWN?
      JMP I BIASUB  /NO
      DAC TEMP16   /YES

```



```

JMS GENCLR          /GENERAL RELEASE
TYP CR
LAC (117300)       /TURN ON "POWER FAIL" LAMP
JMS SEND
LAC TEMP16
JMS LNKCHK         /CHECK FOR HIGH BITS IN STATUS WORD
JMP EXWAIT        /ENTIRE STATUS WORD CHECKED
LAC TEMP16        /CONTAINS NUMBER OF HIGH BIT
TAD FTR14+2       /-11
CLLIRAL           /X 2
TAD (BIATAB)
TYPE              /POWER SUPPLY NAME (2 REGISTERS)
LAC (BIAS.T)
TYPE              /SPACE; "FAILURE"
TYP CR
JMP LNKCHI
BIATAB, 010300     /AC
0
202361            /PS1-C
550300
202361            /PS1-V
552600
202362            /PS2-C
550300
202362            /PS2-V
552600
202363            /PS3
0
202370            /PS8
0
202366            /PS6/7
576700
BIAS.T, 400601
111425
220500

/*****
/CHKOUT -- ISS. 14 FEBRUARY 17/72 PHT
/USAGE: JMS CHKOUT
/CHECKS OUTPUT LEVELS AND TYPES ACCORDING TO DATA MODE
CHKOUT, 0
LAC PARFLG
SAD (1)           /LOOPING?
JMP I CHKOUT     /YES
LAC DATMOD
DAC DTHAND       /TEMPORARY STORAGE
LAC (3761)       /TPSW0 COORDINATES + F.A.W. INDICATOR BIT
DAC TEMP10       /TPSW CODE POINTS
DZM LINERR       /ERRORS PER TRUHTABLE LINE
DZM TEMP.4       /USED IN SUB. INPUTS
LAM
DAC TEMP14       /USED IN SETTING DETECTOR WINDOW
DAC TEMP15       /USED IN SUB. CHKSUB
DZM TEMP11       /OUTPUT TABLE WORD COUNTER
LAC (OUTLO)
DAC TEMP12       /ADDRESS IN OUTPUT TABLES
LAC DATMOD
SMA              /ALL DATA MODE?

```

```

      JMP .+3           /NO
      JMS TITLES       /YES - PRINT TITLES IF NECESSARY
      JMS INPUTS      /TYPE INPUTS
CHKO.A, LAC I TEMP12  /PICK UP CONTENTS OF OUTPUT TABLE WORD
      SZA             /IS THERE AN OUTPUT PIN IN THIS WORD?
      JMP CHKO.B      /YES
      ISZ TEMP10      /NO - CONTINUE TO NEXT OUTPUT WORD
      ISZ TEMP11
      ISZ TEMP12
      LAC TEMP11
      SAD (11)        /FINISHED WITH TABLE OUTLO?
      JMP CHKO.B-2    /YES
      SAD (22)        /FINISHED WITH TABLE OUTHI?
      SKP             /YES
      JMP CHKO.A
      LAC DTHAND
      DAC DATMOD      /RESTORE DATA MODE
      LAC LINERR
      SNA
      JMP .+6
      TAD TOTERR
      DAC TOTERR
      LAC FSTREJ
      SZA             /STOP ON FIRST REJECT?
      JMP EXWAIT      /YES
      LAC SHTTIT
      SMA             /SHORT TESTING?
      JMP I CHKOUT    /YES
      LAC SINLIN      /NO
      SZA             /SINGLE LINE MODE?
      JMP EXWAIT      /YES
      JMP I CHKOUT
      DZM TEMP14
      JMP CHKO.A
CHKO.B, LAC TEMP14
      SAD (1)         /DETECTOR WINDOW ALREADY SET?
      JMP CHKO.G      /YES
      LAC (176300)
      JMS SEND        /DETECTOR UPPER AND LOWER THRESHOLD CLEAR
      LAC (15000)
      JMS SEND        /P.A.W. FOR DETECTOR WINDOW
      LAC PWRSET      /MARGIN
      CLL!RTL        /X4
      TAD (PWRWRD+2)
      DAC TEMP16      /ADDRESS OF LOW WINDOW
      LAC TEMP14
      SMA             /LOW WINDOW?
      ISZ TEMP16      /NO - ADDRESS OF HIGH WINDOW
      LAC I TEMP16    /WORDFILL FOR DETECTOR WINDOW
      XOR (2)         /ADD BIT 16 (WORDFILL)
      JMS SEND
      LAC (1)
      DAC TEMP14
CHKO.G, LAC TEMP10
      TAD (-3771)
      SPA
      JMP .+4

```

```

TAD (3761)
DAC TEMP10
DZM TEMP15
LAC TEMP10
CLLROT                                /CREATE F.A.W. FOR TPSW CODE POINT
-5
JMS SEND
LAC STATUS                            /READ TPSW
AND I TEMP12                          /ISOLATE PINS WHICH ARE TO BE CHECKED
DAC TPSW
LAC DATMOD
SMA                                    /ALL DATA MODE?
JMP CHKO.D                             /NO
CHKO.C, JMS CHKSUB                      /TYPE OUTLO (OR OUTHI)
LAC I TEMP12                          /CONTENTS OF OUTPUT TABLE WORD
JMS LNKCHK                             /DETERMINE OUTPUT PINS IN THIS WORD
JMP CHKO.A+3                          /ENTIRE WORD CHECKED
LAC TEMP11                             /OUTPUT PIN FOUND:
TAD (-10)
SPA                                    /ARE WE DOING OUTHI?
LAC TEMP11                             /NO
CLLRTL
RTL                                    /X16
TAD TEMP16                             /ADD PIN POSITION NO.(FROM LNKCHK)
DAC TEMP19                             /OUTPUT PIN NUMBER
LAC SHTTIT
SMA                                    /SHORT TESTING?
JMP CHKO.H                             /YES
LAC TEMP19                             /NO
JMS TYPPIN                             /PRINT OUTPUT PIN
LAC DATMOD
SAD (2)                                /REJECT COORDINATES, MODE?
CHKO.H, JMP CHKO.E-2                  /YES - RETURN TO SUB. LNKCHK
LAC TEMP16
CMA
DAC .+3
LAC TPSW
ROTATE                                /SHIFT TPSW LEFT = NUMBER OF PIN ...
0                                       /... POSITIONS IN OUTPUT WORD
SNL                                    /INCORRECT OUTPUT ON THIS PIN?
JMP LNKCHI                             /NO - OUTPUT IS OK
LAC SHTTIT
SPA
JMP .+3
LAC TEMP19
JMS TYPPIN
LAC (CHKO.T)
TYPE                                  /REJECT; 2 SPACES
ISZ TEMP17
ISZ TEMP17
JMP CHKO.E-2
CHKO.D, LAC TPSW
SNA                                    /ANY REJECTS FOR THIS LINE OF OUTPUTS?
JMP CHKO.A+3                          /NO
LAC DATMOD
SZA                                    /NO-DATA MODE?
JMP CHKO.E                             /NO

```

```

LAC TPSW /YES
JMS LNKCHK /PICK UP THE ERROR INDICATION BITS
JMP CHKO.A+3 /ALL 16 BITS CHECKED
ISZ LINERR
JMP LNKCHI
CHKO.E, SAD (1) /REJECT DATA MODE?
JMP CHKO.F /YES
JMS TITLES /PRINT TITLES IF NECESSARY
JMS CHKSUB /TYPE OUTLO (OR OUTHI)
LAC I TEMP12 /CONTENTS OF OUTPUT TABLE WORD
AND TPSW /ISOLATE ERROR PINS
JMP CHKO.C+2
CHKO.F, LAM /REJECT DATA MODE:
DAC DATMOD /FORCE INTO ALL DATA MODE FOR CURRENT LINE
JMP CHKOUT+6
CHKO.T, 220512
50324
404000
CHKSUB, 0
LAC TEMP15
SAD (1) /OUTLO OR OUTHI TO BE TYPED?
JMP I CHKSUB /NO
SZA /OUTH1?
JMP .+3 /NO - OUTLO
LAC (CHKST1) /LF-CR; SPACE; OUTHI; TAB
JMP .+2
LAC (CHKST2) /LF-CR; SPACE; OUTLO; TAB
TYPE
LAC (1)
DAC TEMP15
DZM TEMP17
JMP I CHKSUB
CHKST1, 474017
252410
114400
CHKST2, 474017
252414
174400
/*****
/DTHAND -- ISS.3 JULY 7/71 PHT
/USAGE: JMS DTHAND
/ JMP ERROR /1ST ARG.: ERROR HANDLER
/ 0 /2ND ARG.: CORE LOC.OF 1ST WORD(OCTAL)
/ 0 /3RD ARG.: STARTING BLOCK NO.(OCTAL)
/ 0 /4TH ARG.: NO. OF BLOCKS (OCTAL)
/READS SPECIFIED NO. OF BLOCKS FROM DECTAPE DRIVE NO.1
DTHAND, 0
LAM -4
DAC TEMP.3 /REPEAT COUNTER
LAC (10000) /TO ENABLE READ(WRITE) IN CONTINUOUS MODE
DAC TEMP.4
LAC (TEMP.1) /TEMPORARY BLOCK STORAGE
DAC 31 /CURRENT ADDRESS REGISTER
DZM 30 /WORD COUNT REGISTER
LAC DTHAND
DAC 10
LAC I 10 /CORE LOCATION OF 1ST WORD

```

```

DAC TEMP.2
LAC (121000)
DTLA
DTHA.1, JMS DTFLAG /STATUS REG.A:UNIT 1;FWD;GO;NM;SEARCH
LAC I 10 /WAIT UNTIL EF OR DTF IS SET
CMA /STARTING BLOCK NO.
TAD TEMP.1 /ADD BLOCK NO. ENCOUNTERED IN SEARCH
SMA /REQ'D BLOCK NO > CURRENT ONE?
JMP DTHA.3 /NO - REVERSE MOTION
IAC /YES
SNA /REQ'D BLOCK NO = CURRENT ONE?
JMP DTHA.2 /YES
DAC 30 /NO - SET WC FOR NO. OF BLOCKS TO ...
DZM TEMP.4
LAC (10000) /... SKIP IN CONTINUOUS MODE (CM)
DTXA /STATUS REG.A:UNIT 1;FWD;GO;CM;SEARCH
DTHA.2, JMS DTFLAG
LAM
TAD TEMP.2
DAC 31 /CA = CORE LOC.OF 1ST WORD - 1
LAC I 10 /NO. OF BLOCKS
CMA
DAC TEMP.1
DZM 30
ISZ TEMP.1 /256 WORDS ADDED TO WC PER BLOCK?
SKP /NOT YET
JMP .+5 /YES
LAC 30
TAD (DECIMAL -255 OCTAL)
DAC 30 /WC IN 2'S COMPLEMENT FORM
JMP .-6
LAC DTHA.4 /3000 FOR READ DATA, 5000 FOR WRITE DATA
TAD TEMP.4
DTXA /STATUS REGISTER A: UNIT 1; FWD; GO; ...
JMS DTFLAG /... CM; READ (WRITE) DATA
LAC (20000)
DTXA /STOP DECTAPE
DTHA.3, JMP I 10 /EXIT SUBROUTINE
TAD (2) /REVERSE MOTION
CMA
DAC 30 /NO.OF BLOCKS TO SKIP IN REVERSE;CM
LAC (50000)
DTXA /STA.REG.A:UNIT 1;REV;GO;CM;SEARCH
JMS DTFLAG
LAC (50000)
DTXA /REVERT TO FWD; NM
LAM
TAD 10
DAC 10 /RESTORE TO PICK UP 3RD ARG.
JMP DTHA.1
DTHA.4, 3000
/*****
/DTFLAG -- ISS.3 AUGUST 6/71 PHT
/USAGE: JMS DTFLAG
/INTERROGATES DECTAPE FLAG STATUS FOR USE BY SUBROUTINE DTHAND
DTFLAG, 0
DTDF /DECTAPE FLAG UP?

```

```

SKP                               /NO
JMP I DTFLAG                       /YES
DTEF                               /ERROR FLAG UP?
JMP .-4                            /NEITHER - WAIT
ISZ TEMP.3                         /YES - REPEATED 5 TIMES?
JMP .+3                             /NOT YET
DTRB                               /YES - READ STATUS REGISTER B
JMP I DTHAND                       /ERROR HANDLER - 1ST ARG. IN DTHAND
DTRB
AND (100000)                       /BIT 2: END OF TAPE
SNA                               /END OF TAPE?
JMP DTHAND+3                       /NO - TRY AGAIN
DTRA                               /YES - READ STATUS REGISTER A
AND (40000)                       /ISOLATE DIRECTION BIT
SZA                               /TAPE WAS GOING FORWARD?
JMP DTHAND+3                       /NO - REVERSE - TRY AGAIN
LAC (160000)                       /YES - UNIT 1; REVERSE; GO; MOVE
JMP DTHA.1-1

```

```

/FLGCLR -- ISS.1 FEB.26/71 PHT
/USAGE: JMS FLGCLR
/CLEARs STEP, INTERRUPT P-B, CONTINUE, AND START CODE POINTS
FLGCLR, 0
LAC (76000)                        /P.A.W. FOR Y = 76
JMS SEND
LAC (76)
JMS SEND
JMP I FLGCLR

```

```

/GENCLR -- ISS.2 SEPT. 16/71 PHT
/USAGE: JMS GENCLR
/CLEARs STATION
GENCLR, 0
LAC (76000)                        /P.A.W. FOR Y=76
JMS SEND
LAC (737776)                       /WORDFILL
JMS SEND
JMP I GENCLR

```

```

/INLOAD -- ISS.2 SEPTEMBER 2/71 PHT
/USAGE: JMS INLOAD
/      0 /BIT CONFIGURATION
/LOADS EACH WORD IN INPUT LEVEL TABLE INTAB WITH SPECIFIED BIT CONFIG.
INLOAD, 0
LAC (INTAB-1)
DAC I 0
LAC I INLOAD
AND (777774)                       /REMOVE BITS 16 AND 17
DAC I 0
LAC I 0
SAD (INTAB+5)
SKP
JMP .-6
ISZ INLOAD
JMP I INLOAD

```

```

/INMOD -- ISS. 1 JAN. 25/71 PHT
/USAGE: JMS INMOD
/PRE-SET REGISTERS: YCOORD - Y COORDINATE OF ACCESS POINT
/                      TEMP.7 - INPUT TABLE ADDRESS CORRESPONDING TO YCOORD
/SETS UP NEW INPUT Y-LEVEL AT THE MATRIX, AND TRANSMITS
INMOD, 0
    LAC YCOORD          /ACCESS POINT Y-LEVEL
    SWHA                /SHIFT INTO BITS 3-8
    XOR (4)             /ADD RELEASE BIT 15
    DAC TEMP10
    JMS SEND            /TRANSMIT P.A.W. - RELEASE
    LAM -1
    JMS SEND            /CLEAR ALL BITS ON THIS Y-LEVEL
    LAC TEMP10
    XOR (4)             /REMOVE RELEASE BIT
    JMS SEND            /TRANSMIT P.A.W. - CONNECT
    LAC I TEMP.7        /PICK UP WORDFILL FROM INPUT TABLE
    XOR (2)             /ADD WORDFILL INDICATOR BIT 16
    JMS SEND
    JMP I INMOD

/*****
/INMOD1 -- ISS. 2 MARCH 24/71 PHT
/USAGE: JMS INMOD1
/IF THIS SUBROUTINE IS ENTERED WITH A ZERO IN THE AC, IT WILL...
/... GO TO SUBROUTINE UPDAT1 TO MODIFY THE INTAB
/IT WILL TRANSMIT THE LATCH ENABLE CODE POINT CONDITIONALLY
/...IF THE CURRENT FTR IS AN FTR 14, OR NO FURTHER CONNECT ACCESS POINT
/...FTR'S FOLLOW.
INMOD1, 0
    SNA
    JMS UPDAT1
    JMS INMOD
    LAC FTRNO
    SAD (16)            /FTR 14?
    JMP INM1.1         /YES
    JMS NEXTEC         /PICK UP NEXT E.C.
    AND (774000)       /ISOLATE BITS 0-6
    SAD (400000)       /INPUT FTR4?
    JMP INM1.2         /YES
    SAD (500000)       /INPUT FTR5?
    JMP INM1.3         /YES
INM1.1, LAC (176140)   /TRANSMIT LATCH ENABLE CODE POINT
    JMS SEND           /CHECK POWER SUPPLIES STATUS
    JMS BIASUB
    JMP I INMOD1
INM1.2, LAC I ECPNTR  /PICK UP E.C.
    AND (1777)         /ISOLATE BITS 8-17
    TAD (-137)
INM1.4, SMA           /CODE POINT?
    JMP INM1.1         /YES
    JMP I INMOD1
INM1.3, LAC I ECPNTR
    AND (77)           /ISOLATE Y-COORDINATE
    TAD FTR4.1+2      /-5
    JMP INM1.4

/*****

```

```

/INPUTS -- ISS. 3 MARCH 21/71 PHT
/USAGE: JMS INPUTS
/PRE-SET REGISTERS: TEMP.4=0 - TYPE INPUT PIN
/                               =1 - DO NOT TYPE
INPUTS, 0
  LAC TEMP.4
  SZA                               /INPUT PINS TO BE TYPED?
  JMP I INPUTS                       /NO
  LAC (INPUT1)
  TYPE                               /LF-CR; 2 SPACES; INLO; TAB
INPUT3, DZM TEMP17                   /PINS TYPED PER LINE: 14 MAX.
  DZM TEMP16                         /PINS PER WORD
  DZM TEMP18                         /TABLE WORD COUNTER
  LAC SHTTIT
  SPA                               /SHORT TESTING?
  JMP .+3                             /NO
  LAC (SHTAB)                         /YES
  SKP
  LAC (INPINS)
  DAC TEMP.7                         /INPUT PIN TABLE ADDRESS
  LAC (INTAB)
  DAC TEMP.8                         /INPUT LEVELS TABLE ADDRESS
INPUT4, LAC I TEMP.8                 /CONTENTS OF INTAB
  AND I TEMP.7                       /CONTENTS OF INPINS
  DAC TEMP.9                         /INHI'S
  LAC TEMP.4
  SZA                               /ARE WE HANDLING INLO'S?
  JMP .+4                             /NO
  LAC TEMP.9                         /YES
  XOR I TEMP.7
  DAC TEMP.9
  LAC TEMP.9
  SZA                               /ANY LOW (OR HIGH) PIN IN THIS WORD?
  JMP INPUT6                         /YES
INPUT5, ISZ TEMP.7                   /NO
  ISZ TEMP.8
  ISZ TEMP18
  LAC TEMP18
  SAD (6)                            /ALL TABLE WORD HANDLED?
  SKP
  JMP INPUT4                         /NO - CONTINUE WITH NEXT WORD
  LAC TEMP.4                         /YES
  SZA                               /DID WE DO INLO?
  JMP I INPUTS                       /NO - FINISHED
  ISZ TEMP.4
  LAC (INPUT2)
  TYPE                               /LF-CR; 2 SPACES; INHI; TAB
  JMP INPUT3
INPUT6, JMS LNKCHK
  JMP INPUT5                         /16 PINS HANDLED FOR CURRENT WORD
  LAC TEMP18
  CLL IRTL
  RTL                               /WORD COUNTER X 16
  TAD TEMP16                         /ADD PIN COUNTER
  JMS TYPPIN                         /TYPE THE PIN
  JMP LNKCHI

```


INPUT1, 474040
 111614
 174400
 INPUT2, 474040
 111610
 114400

/*****

/LNKCHK -- ISS.2 FEB. 17 /71 PHT

/USAGE: JMS LNKCHK

/ RETURNS HERE IF ALL 16 BITS WERE CHECKED

/ RETURNS HERE IF CURRENT BIT IS HIGH

/OUTPUT REGISTER: TEMP16 - CONTAINS POSITION OF HIGH BIT

/RE-ENTER WITH JMP LNKCHI TO CONTINUE CHECKING BITS

LNKCHK, 0

DAC LKNC.1

LAC LNKCHK

IAC

DAC LNKCH2

DZM TEMP16 /BIT COUNTER

LAC LKNC.1

CLL!RAL

DAC LKNC.1

ISZ TEMP16

SZL /BIT IS HIGH?

JMP I LNKCH2 /YES

LNKCHI, LAC TEMP16 /NO

SAD (20)

JMP I LNKCHK

JMP LNKCHK+6

LNKCH2, 0

/*****

/LOCATE -- ISS. 3 APRIL 1/71 PHT

/USAGE: JMS LOCATE

/PRE-SET REGISTER: PARFLG - 0 = PARTIAL TEST; 1 = LOOPING

/OUTPUT REGISTERS: ADDR.1 - STARTING TABLE ADDRESS

/ ADDR.2 - ADDRESS OF LINE 1 IN STARTING TABLE

/ ADDR.3 - ADDRESS OF STARTING LINE IN START. TABLE

/ ADDR.4 - ADDRESS AFTER END OF FINISHING LINE

/DETERMINES ADDRESSES IN TSITAB FOR THOSE E.C.'S TO BE EXECUTED...

/...FOR PARTIAL TESTING OR LOOPING. IN EACH CASE TESTING WILL...

/...INCLUDE ANY INITIAL CONDITIONS IN THE STARTING TABLE.

LOCATE, 0

LAC PARFLG

SZA /PARTIAL TEST?

JMP .+3 /NO - LOOPING

LAC (LOCAT1) /YES: LF-CR; PARTIAL TEST; LF-CR

SKP

LAC (LOCAT2) /LF-CR; LOOPING; LF-CR

TYPE

JMS BIAREQ /REQUEST BIASING MODE

DZM TEMP.5 /FOR TABLE SEARCH

LAM

DAC TEMP.6 /FOR LINE SEARCH

LOCA.A, LAC (TSITAB-1)

DAC 10

LAC (LOCAT3) /LF-CR; TABLE >

	TYPE	
LOCA.B,	JMS READNO	/WAIT FOR TABLE NUMBER
	LAC I 10	/PICK UP E.C.
	DAC ECSTOR	
	AND (770000)	/ISOLATE FTR NO.
	SAD (40000)	/FTR 4?
	JMP LOCA.B	/YES
	SAD (110000)	/FTR9?
	JMP LOCA.C	/YES
	SAD (50000)	/FTR5?
	JMP LOCA.C	/YES
	SAD (100000)	/FTR8?
	JMP LOCA.C+2	/YES
	SAD (130000)	/FTR11?
	JMP LOCA.D	/YES
	SAD (160000)	/FTR14?
	JMP LOCA.C-4	/YES
	SAD (140000)	/FTR12? (TABLE NUMBER)
	JMP LOCA.L	/YES
	SAD (150000)	/FTR13? (LINE NUMBER)
	JMP LOCA.H	/YES
	LAC (LOCAT4)	/ILLEGAL FTR NO
	TYPE	
	JMP EXWAIT	
	LAC ECSTOR	
	AND (4000)	/BIT 6
	SZA	/INPUT PIN TOGGLE?
	SKP	/NO
LOCA.C,	ISZ 10	
	JMP LOCA.B	
	LAC I 10	/FTR8: PICK UP MESSAGE WORDFILL
	SZA	/END OF MESSAGE ?
	JMP .-2	/NO
	JMP LOCA.B	
LOCA.D,	LAC TEMP.6	/FTR11:
	TAD (-1)	
	SMA	/LOOKING FOR ADDR.4?
	JMP LOCA.E	/YES
	LAC (LOCAT5)	/LF-CR; TABLE/LINE NOT FOUND
	JMP LOCA.C-6	
LOCA.E,	LAC 10	
	DAC ADDR.4	
	CMA	
	TAD ADDR.3	
	IAC	
	SPA	/LAST ADDRESS SMALLER THAN PREVIOUS ONE?
	JMP I LOCATE	/NO
	JMP NOSIR	/YES
LOCA.L,	LAC TEMP.6	/FTR12:
	TAD (-1)	
	SMA	/LOOKING FOR ADDR.4?
	JMP LOCA.E	/YES
	LAC TEMP.5	
	SPA	/LOOKING FOR A TABLE NO.?
	JMP LOCA.F	/NO
	LAC ECSTOR	/YES

	AND (77)	/ISOLATE TABLE NO.
	SAD RDSTOR	/SAME AS SPECIFIED?
	SKP	/YES
	JMP LOCA.F	/NO-CONTINUE LOOKING
	LAC TEMP.5	
	SZA	/LOOKING FOR ADDR.1?
	JMP LOCA.G	/NO
	LAC I0	/YES
	DAC ADDR.1	
	DZM TEMP.6	/SET FOR ADDR.2
	DZM RDSTOR	
	ISZ RDSTOR	/SET FOR LINE NO. 1
	JMP LOCA.G+5	
LOCA.F,	LAC ECSTOR	
	AND (2000)	/ISOLATE BIT 7
	SZA	/WORDFILLS FOLLOW?
	JMP LOCA.B	/NO
	LAC I I0	/YES - PICK UP WORDFILL
	SAD ALLDAT-1	/ANY MORE WORDFILLS?
	JMP LOCA.B	/NO
	JMP .-3	/YES
LOCA.G,	LAC (2)	
	DAC TEMP.6	/SET FOR ADDR.4
	LAC (LOCAT6)	/TAB; LINE >
	TYPE	
	JMS READNO	/WAIT FOR LINE NO
	LAM	
	DAC TEMP.5	
	JMP LOCA.F	
LOCA.H,	LAC TEMP.6	/FTR13:
	YAD (-2)	
	SPA	/LOOKING FOR ADDR.4?
	SKP	
	JMP LOCA.E	/YES
	LAC TEMP.6	
	SMA	/LOOKING FOR TABLE NO RATHER THAN LINE NO?
	JMP LOCA.I+2	/NO
LOCA.I,	JMS LOCA.S	/YES - SKIP OVER ANY WORDFILLS
	JMP LOCA.B	
	LAC ECSTOR	
	AND (377)	/ISOLATE TRUTH TABLE LINE NO.
	SAD RDSTOR	/IS THIS THE REQUESTED LINE NO.?
	SKP	
	JMP LOCA.I	/NO
	LAC TEMP.6	/YES
	SZA	/LOOKING FOR ADDR.2?
	JMP LOCA.K	/NO
	LAC I0	/YES
	DAC ADDR.2	
	LAC (LOCAT6)	/TAB; LINE >
	TYPE	
	JMS READNO	/WAIT FOR LINE NO
	LAC RDSTOR	
	SAD (1)	/WAS LINE NO. 1 REQUESTED?
	JMP LOCA.J-2	/YES
	DZM TEMP.6	/NO-SET TEMP.6 FOR ADDR.3
	ISZ TEMP.6	

```

        JMP LOCA.I
        LAC ADDR.2
        DAC ADDR.3
LOCA.J, LAM
        DAC TEMP.6
        DZM TEMP.5
        ISZ TEMP.5
        JMS LOCA.S
        JMP LOCA.A
LOCA.K, SAD (1)
        JMP .+3
        ISZ TEMP.6
        JMP LOCA.I
        LAC I0
        DAC ADDR.3
        JMP LOCA.J
LOCA.S, 0
        LAC ECSTOR
        AND (7400)
        SNA
        JMP I LOCA.S
        CLLROT
        I0
        TAD I0
        DAC I0
        JMP I LOCA.S
LOCAT1, 472001
        222411
        11440
        240523
        244700
LOCAT2, 471417
        172011
        160747
        0
LOCAT3, 472401
        21405
        407600
LOCAT4, 111414
        400624
        224016
        170000
LOCAT5, 472401
        21405
        571411
        160540
        161724
        400617
        251604
        470000
LOCAT6, 441411
        160540
        760000
/*****

```

/SET FOR END-OF-SECTION TABLE NO.
 /SKIP OVER ANY WORDFILLS

/LOOKING FOR ADDR.3?
 /YES
 /NO-MUST BE ADDR.4

/UTILITY SUBROUTINE TO SKIP ANY...
 /...WORDFILLS FOLLOWING THE FTR13
 /ISOLATE BITS 6-9
 /ANY WORDFILLS FOLLOWING THE FTR13?
 /NO
 /SHIFT INTO BITS 14-17

/ADD NO. OF WORDFILLS TO ADDRESS IN I0

/LF-CR; PARTIAL TEST; LF-CR

/LF-CR; LOOPING; LF-CR

/LF-CR; TABLE >

/ILL FTR NO

/LF-CR; TABLE/LINE NOT FOUND; LF-CR

/TAB; LINE >

```

/NEXTEC -- ISS. 5 NOV. 26/71 PHT
/USAGE: JMS NEXTEC
/PICKS UP NEXT E.C. FROM STORAGE TABLE TSITAB VIA THE ADDRESS...
/...POINTER ECPNTR. IN THE CASE OF PARTIAL TEST OR LOOPING IT...
/...HANDLES ONLY THOSE E.C.'S FALLING WITHIN SPECIFIED ADDRESSES...
/...ADDR.1-ADDR.4
NEXTEC, 0
    ISZ ECPNTR
    LAC PARFLG
    SPA                                /PARTIAL TEST OR LOOPING?
    JMP NEXT.A                          /NO
    LAC ADDR.1
    JMS NEXT.S
    JMP NEXTEC+1                        /ECPNTR < ADDR.1
    LAC ADDR.2                          /ECPNTR = (>) ADDR.1
    JMS NEXT.S
    JMP NEXT.A                          /ECPNTR < ADDR.2
    LAC ADDR.3                          /ECPNTR = (>) ADDR.2
    SAD ADDR.2                          /ADDR.3 = ADDR.2?
    JMP .+3                              /YES
    JMS NEXT.S
    JMP NEXTEC+1                        /ECPNTR < ADDR.3
    LAC ADDR.4                          /ECPNTR = (>) ADDR.3
    JMS NEXT.S
    JMP NEXT.A                          /ECPNTR < ADDR.4
    LAC PARFLG                          /ECPNTR = (>) ADDR.4
    SZA                                /PARTIAL TEST?
    JMP NEXT.A-4                        /NO - LOOPING
    LAM
    DAC PARFLG                          /DISABLE PARTIAL TEST OR LOOPING
    TAD ECPNTR
    DAC ECPNTR                          /DECREMENT ECPNTR
    LAC (ECHAND)
    DAC COSTEP                          /SET CONTINUE RETURN ADDRESS
    LAC (176140)                        /LATCH ENABLE CODE POINT
    JMS SEND
    JMP EXWAIT
    LAC (176400)                        /CYCLE SYNC CODE POINT
    JMS SEND
    LAC ADDR.1
    DAC ECPNTR
NEXT.A, LAC I ECPNTR
NEXT.S, JMP I NEXTEC
    /UTILITY SUBROUTINE
    TCA                                /2'S COMPLEMENT ACCUMULATOR
    TAD ECPNTR
    SPA
    JMP I NEXT.S
    ISZ NEXT.S
    JMP I NEXT.S
TCA=740031
/*****
/OUTCLR - ISS. 1 JAN. 25/71 PHT
/USAGE: JMS OUTCLR
/DEPOSITS ZERO IN ALL LOCATIONS OF OUTPUT TABLES OUTLO AND OUTHI
OUTCLR, 0
    LAC (OUTLO-1)

```

```

DAC 10
DZM I 10
LAC 10
SAD (OUTH1+10)
JMP I OUTCLR
JMP .-4
/*****
/READ -- ISS.1 JULY 7/71 PH:
/USAGE: JMS READ
/READS UP TO 9 CHARACTERS FROM THE TELETYPE KEYBOARD, CONVERTS ...
/... TO 6-BIT ASCII AND PACKS THEM INTO REGISTERS RDSTOR TO ...
/... RDSTOR+2. IF A RUB-OUT IS TYPED, ALL PRECEDING CHARACTERS ...
/... ARE DELETED, A BACK-SLASH IS ECHOED, AND THE SUBROUTINE ...
/... RE-STARTED. TO EXIT THIS SUBROUTINE A SPACE MUST BE TYPED.
READ, 0
      DZM RDSTOR
      DZM RDSTOR+1
      DZM RDSTOR+2      /ZERO CHARACTER STORAGE REGISTERS
      LAC (RDSTOR)
      DAC TEMP.3        /STORAGE WORD ADDRESS
READ.1, LAM -2
      DAC TEMP.1        /CHARACTER PER WORD COUNTER
      LAC TEMP.1
      DAC TEMP.2        /SHIFTING COUNTER
      JMS READWT        /WAIT FOR KEYBOARD INTERRUPT
      AND (77)          /TRIM TO 6-BIT ASCII
      ISZ TEMP.2        /3RD CHARACTER FOR CURRENT STORAGE WORD?
      SKP                /NO
      JMP READ.2        /YES
      CLLROT            /SHIFT 1ST CHAR.INTO AC (0-5), 2ND CHAR. ...
      -6                /... INTO AC (6-11), 3RD CHAR. INTO AC (12-17)
      JMP .-5
READ.2, TAD I TEMP.3
      DAC I TEMP.3      /ADD CHARACTER TO CHARACTER STORAGE WORD
      ISZ TEMP.1        /3RD CHARACTER PACKED?
      JMP READ.1+2      /NO
      ISZ TEMP.3        /YES
      LAC TEMP.3
      SAD (RDSTOR+3)    /ALL 3 STORAGE WORDS FILLED?
      SKP                /YES
      JMP READ.1        /NO
      JMS READWT        /WAIT FOR A SPACE TO EXIT SUBROUTINE
      JMP .-1           /NO SPACE YET
READ.3, LAC (334)      /BACK-SLASH
      PRINT
      JMP READ+1
READWT, 0
      JMP .              /WAIT FOR KEYBOARD INTERRUPT
      KRB                /READ KEYBOARD BUFFER
      SAD (240)          /SPACE?
      JMP I READ        /YES - EXIT
      SAD (377)        /RUB-OUT?
      JMP READ.3        /YES
      JMP I READWT      /NEITHER - DO NORMAL
/*****

```

```

/READNO -- ISS.2 FEB.27/71 PHT
/USAGE: JMS READNO
/READS A MULTY-DIGIT DECIMAL NUMBER FROM KEYBOARD, CONVERTS TO ...
/... OCTAL AND STORES IN REGISTER RDSTOR. TO EXIT SUBROUTINE MUST ...
/... READ A "SPACE". ALL SYMBOLS EXCEPT DIGITS ARE IGNORED. IF A RUB-OUT
/... IS TYPED, A BACK-SLASH IS ECHOED, AND ALL HITHERTO TYPED NUMBERS..
/... DELETED.

```

```

READNO, 0
    DZM RDSTOR
    JMS READSR          /OBTAIN DIGIT
    DAC TEMP.2
    LAC RDSTOR
    CLLIRAL            /MULTIPLY BY 2
    TAD TEMP.2         /ADD PREVIOUS MULTIPLICATION RESULT
    TAD TEMP.1         /ADD LATEST DIGIT
    DAC RDSTOR         /NUMBER NOW IN OCTAL
    JMP READNO+2       /PICK UP NEXT DIGIT

```

```

/*****

```

```

/READOC -- ISS.1 AUGUST 6/71 PHT
/USAGE: JMS READOC
/READS A MULTY DIGIT OCTAL NUMBER FROM KEYBOARD AND STORES IT IN ...
/... RDSTOR. TO EXIT SUBROUTINE MUST READ A "SPACE". ALL SYMBOLS ...
/... EXCEPT DIGITS ARE IGNORED. IF A RUB-OUT IS TYPED, A BACK-SLASH.
/... IS ECHOED, AND ALL HITHERTO TYPED DIGITS ARE DELETED.

```

```

READOC, 0
    DZM RDSTOR
    LAC READOC
    DAC READNO
    JMS READSR          /OBTAIN DIGIT
    TAD TEMP.1
    DAC RDSTOR
    JMP READOC+4

```

```

/*****

```

```

/READSR -- ISS.1 AUGUST 6/71 PHT
/USAGE: JMS READSR
/UTILITY SUBROUTINE FOR READNO AND READOC
READSR, 0
    JMP .              /WAIT FOR KEYBOARD INTERRUPT
    KRB                /READ KEYBOARD BUFFER
    SAD (240)          /SPACE?
    JMP I READNO       /YES - EXIT
    SAD (377)          /RUB-OUT?
    JMP READS1         /YES
    TAD (-257)
    SPA                /ILLEGAL CHARACTER?
    JMP READSR+1       /YES - IGNORE
    TAD FTR14+2        /-11
    SMA                /ILLEGAL CHARACTER?
    JMP READSR+1       /YES - IGNORE
    TAD (12)           /NO - RESTORE DIGIT VALUE
    DAC TEMP.1
    LAC RDSTOR
    CLLIRTL
    RAL                /X 8
    JMP I READSR
READS1, LAC (334)     /BACK-SLASH

```

```

PRINT
DZM RDSTOR
JMP READSR+1
/*****
/ROTATE -- ISS.2 FEB.17/71 PHT
/USAGE: ROTATE
/ +(-)X /SHIFTING COUNTER(OCTAL): + = RIGHT, - = LEFT
/SHIFTS CONTENTS OF THE AC RIGHT OR LEFT ACCORDING TO THE COUNTER
ROTATE=JMS+.
Ø
DAC ROTA.1
LAC (RAL) /ASSUME LEFT SHIFTING
DAC .+16
LAC I ROTATE-JMS /PICK UP SHIFTING COUNTER
DAC ROTA.2
SPA /RIGHT SHIFTING?
JMP .+5 /NO - LEFT
CMA
DAC ROTA.2
LAC (RAR)
DAC .+6
LAC ROTA.1
ISZ ROTA.2 /FINISHED?
JMP .+3 /NO
ISZ ROTATE-JMS /YES - OUT
JMP I ROTATE-JMS
Ø /RAL OR RAR
JMP .-5
/*****
/CLLROT -- ISS.2 FEB.17/71 PHT
/USAGE: CLLROT
/ +(-)X SHIFTING COUNTER(OCTAL): + = RIGHT, - = LEFT
/CLEARs LINK, THEN SHIFTS CONTENTS OF THE AC RIGHT OR LEFT ...
/... ACCORDING TO THE SHIFTING COUNTER
CLLROT=JMS+.
Ø
DAC ROTA.1
LAC CLLROT-JMS
DAC ROTATE-JMS
CLL
JMP ROTATE-JMS+2
/*****
/SEMCOL -- ISS.2 APRIL 20 /71 PHT
/USAGE: JMS SEMCOL
/PRE-SET REGISTER: REQINT - INTERRUPT VALIDITY FLAG
/WHENEVER, DURING TSI EXECUTION, A SEMI-COLON IS TYPED, OR THE ...
/... INTERRUPT P-B IS PRESSED, THE INTERRUPT HANDLER WILL INITIALIZE ...
/... THE REGISTER REQINT WITH -Ø, AND SET THE CONTINUE-STEP RETURN ...
/... ADDRESS REGISTER COSTEP FOR LOCATION SEMICO. PROGRAM EXECUTION ...
/... CONTINUES UNTIL THE NEXT OCCURRENCE OF SUBROUTINE SEMCOL.
SEMCOL, Ø
ISZ REQINT /SEMI-COLON OR INTERRUPT P-B?
JMP .+4 /NO
LAC (ECHAN+1) /YES - RETURN ADDRESS
DAC COSTEP
JMP EXWAIT

```



```

DZM REQINT
JMP I SEMCOL
/*****
/SEND -- ISS. 8 SEPTEMBER 22/71 PHT
/USAGE: JMS SEND
/TRANSMITS F.A.W., P.A.W. OR WORDFILL ACCORDING TO CONTENTS...
/...OF AC WHEN SUBROUTINE IS CALLED
SEND, 0
      IOF /TURN INTERRUPT OFF
      DAC SEND.A /STORE WORD TO BE TRANSMITTED
      LAC SEND
      DAC DTERR /STORE RETURN ADDRESS FOR STEP IN THIS REGISTER
      LAM -4
      DAC SEND.B /TRANSMISSION REPEAT COUNTER
      LAM -21
      DAC SEND.C /BIT COUNTER
      DZM SEND.D /1'S COUNTER
SEND.0, LAC SEND.A
      AND (2) /ISOLATE BIT 16
      SZA /WORDFILL?
      JMP SEND.1 /YES
      LAC SEND.A
      AND (300000) /ISOLATE BITS 1,2...
      CLLIRTL /...AND SHIFT THEM INTO BITS 16,17
      RTL
      DAC SEND.D /INCLUDE IN PARITY COUNT
      RAL
      XOR PAWIOT /P.A.W. IOT
      DAC SEND.E /MODIFIED IOT
      LAC SEND.A
      AND (77777) /ISOLATE BITS 3-17
      TAD FACTS /ADD STATION NO. IN BITS 0-2
SEND.1, DAC SEND.A
      LAC SEND.A
      RAL
      SZL
      ISZ SEND.D /INCREMENT 1'S COUNTER FOR EACH 1
      ISZ SEND.C /ALL 18 BITS HANDLED?
      JMP .-4 /NO
      LAC SEND.D
      AND (1) /ISOLATE BIT 17
      XOR SEND.A
      DAC SEND.A /WORD HAS EVEN PARITY NOW
SEND.2, LAC SEND.A
      XCT SEND.E /EXECUTE THE IOT
      AND (2)
      SZA /WAS A WORDFILL TRANSMITTED?
      JMP .+5 /YES
      LAC SEND.E /NO - P.A.W. TRANSMITTED?
      AND (2)
      SNA
      JMP SEND.4 /YES - EXIT NORMALLY ALWAYS
      LAC STPHAN
      SZA /IS THIS A STEP HANDLING TRANSMISSION?
      JMP .+3 /YES
      CHFACT /NO - READ THE LAST STATUS WORD THAT WAS...
      DAC STATUS /...SET UP, AND STORE IT IN THIS REGISTER

```

```

LAC STPTOG
SNA /STEP MODE?
JMP SEND.4 /NO
LAM
DAC STPFLG /YES - SET STEP FLAG
ION /TURN INTERRUPT BACK ON
LAC (SEND.6) /LF-CR; *
TYPE
LAC DTERR
DAC COSTEP /SET STEP RETURN ADDRESS
JMP ALLDAT+2
SEND.3, ISZ SEND.B /ENTRY POINT FROM SKIP CHAIN
JMP SEND.2
ION
LAC (SEND.5) /PERSISTENT PARITY INTERRUPT
TYPE /TYPE: PARITY ERROR
JMP EXWAIT /KEYBOARD WAIT
SEND.4, ION /TURN INTERRUPT BACK ON
JMP I SEND
SEND.5, 472001 /LF CR, "PARITY ERROR"
221124
314005
222217
220000
SEND.6, 475200 /LF-CR; *
/*****
/SHORT -- ISS. 17 JAN. 21/72 PHT
/USAGE: JMS SHORT
/PERFORMS INPUT PIN SHORT TESTS FOR ALL PINS SPECIFIED IN TABLE SHTAB,..
/... AND +5V TO -12V POWER PIN TO POWER PIN SHORT TESTS ON BOARDS OF ..
/... CCTYPE=0 ONLY (WHERE +5V=PIN B43, -12V=PIN B3, GROUND=PIN B1)
SHORT, 0
JMS STACHK /CHECK STATION STATUS
LAC DATMOD
DAC STRT.2-1 /TEMPORARY STORAGE USING REG. IN STRTOP
JMS GENCLR /CLEAR STATION
SHOR.1, LAC (117040) /TURN-ON "UNDER TEST" LIGHT
JMS SEND
DZM SHTTIT /USED IN SUBROUTINE TITLE
DZM TOTERR /TOTAL NO. OF REJECTS
DZM DATMOD
ISZ DATMOD /REJECT DATA MODE
LAC (JMP CHKO.G) /MODIFY SUB. CHKOUT TO ELIMINATE ...
DAC CHKO.B /... DETECTOR LEVEL SETTING
LAC (JMP I INPUTS) /MODIFY SUB. INPUTS TO DISABLE ...
DAC INPUT6-2 /... TYPING INHI PINS
LAC (15000)
JMS SEND /P.A.W. FOR DETECTOR THRESHOLDS
LAC (740456) /UTV=4.8V, LTV=1.5V
JMS SEND /WORDFILL
LAC (16000) /P.A.W. FOR PSI
JMS SEND
LAC (310102) /+5V,800MA, NOT ENABLED TO UUT
JMS SEND
LAM -4704 /250 MS WAIT
JMS WAIT

```

```

JMS INLOAD
-0 /SET ALL BITS IN TABLE INTAB HIGH
JMS OUTCLR /CLEAR TABLES OUTLO, OUTHI
LAC (114740) /ENABLE DRIVER LOW VOLTAGE TO 20 MV
JMS SEND
LAC (176200) /LATCH CLEAR CODE POINT
JMS SEND
LAC (117700) /REMOVE GROUND FROM B1
JMS SEND
LAC CCTYPE /CIRCUIT TYPE
SZA
JMP SHOR.3
SHOR.2, LAC (200)
DAC OUTHI+5 /SET BIT FOR B43 (+5V)
LAC (677777)
DAC INTAB+3 /CLEAR BIT FOR B3 (-12V)
LAC (111100) /B3 DRIVER ENABLE RELAY
JMS SEND
LAM -144 /10 MS WAIT
JMS WAIT
JMS CHKOUT /CHECK FOR A SHORT BETWEEN B3 AND B43
JMS OUTCLR /CLEAR OUTHI
LAM
SHOR.3, DAC INTAB+3 /RESET BIT FOR B3
LAC (176100) /DRIVER RELAY ENABLE CLEAR
JMS SEND
LAC (117704) /RE-APPLY GROUND TO B1
JMS SEND
DZM SHTTIT
ISZ SHTTIT
LAC (SHTAB-1)
DAC I0
LAC (OUTH1-1)
DAC I1
LAC I I0 /LOAD CONTENTS OF INPUT PIN ...
DAC I I1 /... SHORT TEST TABLE INTO ...
LAC I0 /... HIGH-OUTPUT LEVELS TABLE
SAD (SHTAB+5)
SKP
JMP .-5
LAC (INTAB)
DAC TEMP.5 /ADDRESS IN INPUT LEVELS TABLE
LAC (OUTH1)
DAC TEMP.6 /ADDRESS IN HIGH-OUTPUT LEVELS TABLE
SHOR.4, LAC I TEMP.6 /PICK UP CONTENTS OF WORD IN OUTH1
DAC STRT.2 /TEMPORARY STORAGE USING REG. IN STRTOP
SNA /ANY OUTPUTS TO BE CHECKED?
JMP SHOR.5 /NO OUTPUTS; OR 16 BITS HANDLED
DZM SHOR.8 /BIT COUNTER
LAC STRT.2
CLLIRAL
DAC STRT.2
ISZ SHOR.8
SZL /BIT IS HIGH?
JMP .+5 /YES
SHOR.9, LAC SHOR.8 /NO
SAD (20) /ALL 16 BITS CHECKED?

```

```

JMP SHOR.5           /YES
JMP SHOR.4+5        /NO - CONTINUE
LAC TEMP.5
TAD (-INTAB+1)      /RESULTS IN THE TABLE WORD NUMBER
CLL!RTL
RTL                 /SHIFT INTO BITS 11-13
TAD (137)           /X-Y COORD. FOR Y=5, X=17
TAD SHOR.8          /PIN NUMBER
CLLROT              /SHIFT RESULTING DRIVER RELAY ENABLE...
-5                 /...CODE POINT X-Y COORD. INTO BITS 6-12
TAD (1000000)       /ADD BIT 2 FOR F.A.W.
DAC TEMP13          /F.A.W. FOR DRE CODE POINT
JMS SEND
LAM -62
JMS WAIT            /5 MS WAIT
CLA
STL
ROTATE
SHOR.8, 0
DAC TEMP20          /CONTAINS A "1" IN THE POSITION OF ...
XOR I TEMP.6        /... THE HIGH BIT DETERMINED IN LNKCHK ...
DAC I TEMP.6        /... AND IS USED TO CLEAR THIS BIT IN ...
LAC TEMP20          /... TABLE OUTHI
XOR I TEMP.5
DAC I TEMP.5        /CLEAR SAME BIT IN TABLE INTAB
JMS CHKOUT          /CHECK ALL REMAINING PINS GIVEN IN ...
LAC TEMP20          /... SHTAB AGAINST SUCCESSIVELY SET ...
XOR I TEMP.5        /... LOW PINS
DAC I TEMP.5        /RESET BIT IN INTAB
LAC TEMP13
XOR (4)             /RELEASE BIT 15
JMS SEND            /RELEASE DRIVER RELAY FOR LOW PIN
JMP SHOR.9          /CONTINUE WITH NEXT BIT
SHOR.5, ISZ TEMP.5   /ADDRESS IN INTAB
ISZ TEMP.6          /ADDRESS IN OUTHI
LAC TEMP.5
SAD (INTAB+6)       /ALL WORDS IN TABLE HANDLED?
SKP                 /YES
JMP SHOR.4          /NO - CONTINUE
LAC (LAC TEMP14)    /RESET SUBROUTINE CHKOUT
DAC CHKO.B
LAC (TYPE)
DAC INPUT6-2        /RESET SUBROUTINE INPUTS
LAM
DAC SHTTIT
JMS OUTCLR          /CLEAR OUTPUT LEVEL TABLES
LAC TOTERR          /TOTAL NO. OF REJECTS
SZA                 /ANY SHORTS?
JMP SHOR.7          /YES
LAC (INPINS-1)     /NO
DAC I0
LAC (5000)
SHOR.6, TAD (1000)
DAC TEMP.5
JMS SEND            /P.A.W. FOR DRIVER RELAYS ...
LAC I 10            /... CORRESPONDING TO WORD IN INPINS
TAD (2)

```

```

JMS SEND          /WORDFILL FOR DRIVER RELAYS
LAC I0
SAD (INPINS+5)   /FINISHED?
JMP .+3          /YES
LAC TEMP.5       /NO
JMP SHOR.6
JMS BIAS         /SET UP BIASING, ETC.
LAM
DAC PVRTIT       /FOR BIAS MODE PRINTING
DAC TSITIT       /FOR TSI NAME PRINTING
DZM TABTIT       /TRUTH TABLE NO. TYPING INDICATOR
DZM LINTIT       /LINE NO. TYPING INDICATOR
DZM TVOFLG       /CLEAR TSI VERIFICATION FLAG
LAC STRT.2-1
DAC DATMOD       /RESTORE DATA MODE
LAC (TSITAB-1)
DAC ECPNTR       /RESTORE TO TOP OF TSI
JMP I SHORT
SHOR.7, LAC (117044) /CLEAR "UNDER TEST" LIGHT
JMS SEND
LAC (117340)     /TURN-ON "REJECT" LIGHT
JMS SEND
LAC STRT.2-1
DAC DATMOD       /RESTORE DATA MODE
JMP EXWAIT

/*****
/STACHK -- ISS.13 OCTOBER 14/71 PHT
/USAGE: JMS STACHK
/CHECKS THAT STATION IS TURNED ON AND IS IN THE AUTO MODE
/CHECKS CONSOLE STATUS AND SETS MODE REGISTERS
STACHK, 0
DZM STPHAN       /CLEAR STEP HANDLING TRANSMISSION FLAG
DZM STPTOG       /CLEAR STEP TOGGLE FLAG
LAC (177000)     /STATUS WORD 0: STATION STATUS
JMS SEND
LAC STATUS
SNA              /STATION ON?
JMP STAC.1-2     /NO
AND (1)          /BIT 17: MANUAL FLAG
SNA              /STATION IN MANUAL MODE?
JMP STAC.1       /NO
LAC (STACT1)     /YES: LF-CR; STATION TO AUTO>
TYPE
JMS READ         /WAIT FOR A "SPACE" TO CONTINUE
JMP STACHK+2
LAC (STACT2)     /LF-CR; STATION ON>
JMP .-4
STAC.1, LAC (177500) /STATUS WORD 2: CONSOLE STATUS
JMS SEND
LAC (117304)     /CLEAR "POWER FAIL" LAMP
JMS SEND
LAC STATUS       /STATUS WORD 2
CLLROT          /BIT 6 INTO LINK, BIT 7 INTO AC 0
-7
SPA              /POWER ON BOARD?
JMP .+3          /YES
LAC (STACT3)     /NO - LF-CR; POWER ON BOARD>

```

```

JMP STAC.1-5
LAC STATUS /STATUS WORD 2
AND (300) /BITS 10,11
SAD (300) /BOARD INSERTED, FIXTURE CLOSED?
JMP .+5 /YES
LAC (117000) /NO - TURN ON "INSERT BOARD" LIGHT
JMS SEND
LAC (STACT4) /CR-LF; INSERT BOARD>
JMP STAC.1-5
LAC (117004) /CLEAR "INSERT BOARD" LIGHT
JMS SEND
LAC KBDMOD
SZA /LATEST COMMAND FROM KEYBOARD?
JMP STAC.2 /YES
LAM -3 /NO
DAC TEMP.5
DZM TEMP.6
LAC STATUS /STATUS WORD 2
CLL!RAL
SZL /BIT UP?
JMP .+4 /YES
ISZ TEMP.6 /NO
ISZ TEMP.5 /FINISHED CHECKING FOR DATA MODES?
JMP .-5 /NO
LAC TEMP.6
TAD ALLDAT-1 /-0
DAC DATMOD
DZM SINLIN /SINGLE LINE FLAG
DZM FSTREJ /STOP ON FIRST REJECT FLAG
LAC STATUS /STATUS WORD 2
CLLROT
-5 /BIT 4 INTO LINK, BIT 5 INTO AC0
SZL /SINGLE LINE MODE?
ISZ SINLIN /YES
SPA /STOP ON FIRST REJECT MODE?
ISZ FSTREJ /YES
RTL /BIT 6 INTO LINK
SZL /"RESTART SEQUENCE ON CONTINUE"?
JMP REPSEQ+1 /YES
RTL /BIT 9 INTO AC 0
SPA /"RECHECK OUTPUTS ON CONTINUE"?
JMP REPEAT+1 /YES
STAC.2, LAC (177000) /STATUS WORD 0
JMS SEND
LAC STATUS /STATUS WORD 0
AND (40000) /BIT 3
SNA /STEP TOGGLE UP?
JMP .+4 /NO
LAM /YES
DAC STPTOG
JMP I STACHK
DZM STPFLG /CLEAR STEP MODE FLAG
JMP I STACHK
STACT1, 472324 /CR-LF; STATION TO AUTO>
12411
171640
241740

```

```

12524
177600
STACT2, 472324 /CR-LF; STATION ON>
12411
171640
171676
0
STACT3, 472027 /CR-LF; PWR ON BOARD>
224017
164002
170122
047600
STACT4, 471116 /CR-LF; INSERT BOARD>
230522
244002
170122
047600

```

/******

/TITLES -- ISS. 9 JAN.10/72 PHT

/USAGE: JMS TITLES

/PRE-SET REGISTERS: PWRTIT = -0 /BIASING MODE

/ TSITIT = -0 /TSI NAME

/ TABTIT = 0 /TABLE NUMBER

/ LINTIT = 0 /LINE NUMBER

/ SHTTIT = 0 /SHORT TESTS

/CONDITIONALLY TYPES TSI NAME, TRUHTABLE NO. AND MARGIN, AND ...

/... TRUHTABLE LINE NO.

TITLES, 0

ISZ TSITIT /TSI NAME PRINTED?

JMP .+6 /YES

TYPCR

TYPCR

LAC (TSINAM)

TYPE

TYPCR

LAC SHTTIT /SHORT TESTS FLAG

SMA /PIN SHORT TEST?

JMP TITL.D /YES

LAC TABTIT

SAD TABLE /TABLE NO. PRINTED?

JMP TITL.B /YES

TITL.C, LAC (TITLT1)

TYPE /2 LF-CR; 2 TABS; TABLE; SPACE

LAC TABLE

DAC TABTIT

JMS TYPDEC /TYPE TABLE NO

LAC INSET /INPUT LEVELS FLAG

SZA

JMP .+3

LAC (TITLT6) /SPACE; -0-

SKP

LAC (TITLT7) /SPACE; -1-

TYPE

TYPTB

LAC PWRSET

DAC PWRTIT

CLLIRAL /X2

	TAD (TITLT2)	
	TYPE	/MARGIN
TITL.A,	DZM LINTIT	
	LAC LINTIT	
	SAD LINE	/LINE NO. PRINTED?
	JMP I TITLES	/YES - OUT
	LAC (TITLT3)	
	TYPE	/2 LF-CR; LINE; SPACE
	LAC LINE	
	DAC LINTIT	
	JMS TYPDEC	/TYPE LINE NO.
	LAC DATMOD	
	SAD (2)	/REJECT COORDINATES MODE?
	SKP	/YES
	JMP .+4	
	TYPTB	
	LAC (CHKO.T)	/"REJECT" - SEE SUB. CHKOUT
	TYPE	
	TYPCR	
TITL.B,	JMP I TITLES	
	LAC PWRIT	
	SAD PWRSET	/MARGIN TYPED?
	JMP TITL.A	/YES
	JMP TITL.C	/NO
TITL.D,	SAD (2)	/TITLE TYPED?
	JMP I TITLES	/YES
	SNA	/NO-INPUT PIN SHORT TEST?
	JMP .+7	/NO - B3 TO B43 SHORT TEST
	LAC (TITLT5)	/YES- 2LF-CR; "INPUT-PIN"
	TYPE	
	LAC (TITLT8)	/SPACE; "SHORT TEST"
	TYPE	
	ISZ SHTTIT	
	JMP I TITLES	
	ISZ SHTTIT	
	LAC (TITLT4)	/LF-CR; "B3-TO-B43"
	JMP .-7	
TITLT1,	474744	
	442401	
	021405	
	400000	
TITLT2,	161715	/NOM
	0	
	150122	/MAR-1
	556100	
	150122	/MAR-2
	556200	
	150122	/MAR-3
	556300	
	150122	/MAR-4
	556400	
TITLT3,	474714	
	111605	
	400000	
TITLT4,	470263	
	552417	
	550264	

630000
 TITLT5, 474711
 162025
 245520
 111600
 TITLT6, 405560
 550000
 TITLT7, 405561
 550000
 TITLT8, 402310
 172224
 402405
 232400

/******

/TYPDEC - ISS. 2 JULY 9/71 PHT

/USAGE: JMS TYPDEC

/TYPES THE CONTENTS OF THE AC AS A DECIMAL NUMBER; PRECEDED BY...

/... A MINUS SIGN IF THE AC HAD BIT 0 ON A 1

TYPDEC, 0

SPAICLL

/POSITIVE NO: CLEAR LINK

CMAISTL

/NEGATIVE NO: COMPLEMENT AND SET LINK

DAC TEMP.1

SNL

/NEGATIVE NO?

JMP .+3

/NO

LAC (255)

/YES - TYPE A "--"

PRINT

LAC (DECIMAL 120000 OCTAL)

DAC TYPD.1

/DIVISOR

LAM -5

DAC TEMP.2

EAC TEMP.1

LMQ

/LOAD DIVIDEND

CLAICLL

DIV

TYPD.1, 0

/OBTAIN INDIVIDUAL DIGITS BY DIVIDING THE ...

DAC TEMP.1

/NO. BY 100K, 10K, 1K, 100, 10 AND 1

LACQ

/PICK UP QUOTIENT

TYPD.2, SNA

/MODIFIED TO SKP AT FIRST NON-ZERO DIGIT

JMP TYPD.3

/IGNORE LEADING ZEROS

TAD (260)

PRINT

/TYPE THE DIGIT

LAC (SKP)

DAC TYPD.2

/MODIFY THE INSTRUCTION TO TYPE ALL DIGITS

TYPD.3, LAC TYPD.1

/MODIFY THE DIVISOR

LMQ

CLAICLL

DIV

12

LACQ

DAC TYPD.1

ISZ TEMP.2

/FINISHED?

JMP TYPD.1-4

/NO

LAC (SNA)

SAD TYPD.2

/WAS INSTRUCTION MODIFIED?

JMP .+3

/NO

DAC TYPD.2

/YES - RESTORE TO SNA

JMP 1 TYPDEC

```

LAC (260)      /TYPE 0
PRINT
JMP I TYPDEC
/*****
/TYPE -- ISS.1 JAN 13/71  PHT
/USAGE:  LAC (ADDRESS)
/        TYPE
/WHERE ADDRESS IS THE FIRST LOCATION OF THE CHARACTER STRING...
/...PACKED AS 3 6-BIT ASCII CHARACTERS PER WORD.
/SPECIAL CHARACTERS: 0 = END OF FILLD (NON-PRINTING)
/                  44 = TAB
/                  47 = CARRIAGE RETURN - LINE FEED (=54)
/NOTE: A FIELD (6 BITS) OF 0'S MUST ALWAYS FOLLOW THE LAST...
/...CHARACTER TO BE PRINTED
TYPE=JMS+.
    0
DAC TEMP.1     /ADDRESS OF CHARACTER WORD
LAC (+14)
DAC TYPE.2     /INITIALIZE SHIFTING COUNTER
LAC (-2)
DAC TEMP.2     /INITIALIZE CHARACTER COUNTER
LAC I TEMP.1   /PICK UP 3 PACKED CHARACTERS
ROTATE        /ROTATE LEFTMOST CHARACTER INTO 12-17
TYPE.2, 0     /SHIFTING COUNTER
AND (77)
DAC TEMP.3     /STORE BITS 12-17 (CHARACTER TO BE DECODED)
SNA           /NON-PRINTING CHARACTER?
JMP TYPE.1-3  /YES
SAD (44)      /TAB?
JMP TYPE.1-6  /YES
SAD (47)      /LF CR?
JMP TYPE.1-4  /YES
SAD (54)      /LINE FEED - IGNORE
JMP TYPE.1-3
TAD (-37)     /START OF CHARACTER CODE CONVERSION
SMA
JMP .+4
TAD (340)
PRINT        /PRINT THE DECODED CHARACTER
JMP TYPE.1-3  /CONTINUE TO NEXT CHARACTER
TAD (240)
JMP .-3      /END OF CHARACTER CODE CONVERSION
TYPTB       /PRINT A TAB
SKP
TYP CR      /PRINT A LF CR
LAC TYPE.2
TAD FTR4.1+2 /-5
DAC TYPE.2   /MODIFY THE SHIFTING COUNTER
TYPE.1, ISZ TEMP.2 /ALL 3 CHARACTERS DECODED?
JMP TYPE.2-2 /NO
LAC TEMP.3  /YES - WAS LAST CHARACTER...
SNA         /...AN END-OF-FIELD INDICATOR?
JMP I TYPE-JMS /YES - EXIT
ISZ TEMP.1  /NO - SET ADDRESS COUNTER FOR NEXT CHAR.WD.
JMP TYPE.2-6
/*****

```

/TYPTB -- ISS.3 SEPTEMBER 2/71 PHT

/USAGE: TYPTB

/TYPES A TAB

TYPTB=JMS+.

Ø

LAC (211)

PRINT

LAC (377)

PRINT

LAC (377)

PRINT

JMP I TYPTB-JMS

/*****

/TYPSP -- ISS.1 JAN 13/71 PHT

/USAGE: TYPSP

/TYPES A SPACE

TYPSP=JMS+.

Ø

LAC (240)

PRINT

/TYPE THE SPACE

JMP I TYPSP-JMS

/*****

/TYPCR -- ISS.2 JULY 7/71 PHT

/USAGE: TYPCR

/TYPES A CARRIAGE RETURN - LINE FEED

TYPCR=JMS+.

Ø

LAC (215)

PRINT

/TYPE THE CR

LAC (212)

PRINT

/TYPE THE LF

JMP I TYPCR-JMS

/*****

/PRINT -- ISS.2 JULY 7/71 PHT

/USAGE: PRINT

/TYPES AN 8-BIT ASCII CHARACTER LOCATED IN AC BITS 10-17

PRINT=JMS+.

Ø

TLS

/TYPE THE CHARACTER

JMP .

/WAIT FOR END-OF-TYPING INTERRUPT

JMP I PRINT-JMS

/*****

/TYPPIN -- ISS.1 FEB.3 /71 PHT

/USAGE: JMS TYPINI

/WITH A NUMBER BETWEEN 1 AND 154 (DECIMAL) IN AC UPON ENTERING...

/...TYPES A + PIN NO. FOR NO'S BETWEEN 1 AND 48, B + PIN NO...

/...FOR NO'S BETWEEN 49 AND 96, AND T + PIN NO. FOR NO'S...

/...BETWEEN 97 AND 154 (THESE LAST ONES REPRESENTING TEST POINTS)

TYPPIN, Ø

DAC TEMP19

/STORE CONT. OF AC

LAC TEMP17

TAD (-15)

SPA

/14 PINS TYPED PER LINE?

JMP TYPP.A

/NO

TYPCR

/YES

TYPTB

DZM TEMP17

```

TYPP.A, ISZ TEMP17
        LAC TEMP19
        TAD (-60)
        SPA /NO. >48?
        JMP TYPP.C /NO
        IAC
        DAC TEMP19 /YES
        TAD (-60)
        SPA /NO. >96?
        JMP TYPP.C+2
        IAC /NO
        DAC TEMP19
        LAC (324)
TYPP.B, PRINT /TYPE T
        LAC TEMP19
        JMS TYPDEC /TYPE THE DECIMAL NUMBER
        TYPSP /TYPE A SPACE
        JMP I TYPPIN
TYPP.C, LAC (301)
        JMP TYPP.B
        LAC (302)
        JMP TYPP.B
/*****
/UPDATE -- ISS. 2 MARCH 15/71 PHT
/USAGE: LAC (TABLE ADDRESS)
/ JMS UPDATE
/PRE-SET REGISTERS: YCOORD - Y COORDINATE OF ACCESS POINT
/ TEMP.4 - X-Y COORDINATES OF ACCESS POINT
/ FTRNO - FTR CODE NO
/UPDATES INPUT OR OUTPUT TABLE REGISTER CORRESPONDING TO THE...
/... CONTENTS OF YCOORD. IN THE CASE OF AN FTR4 (F.A.W.) THE...
/... PIN TO BE UPDATED IS DETERMINED FROM THE X-COORD. IN TEMP.4.
/OTHERWISE THE WORDFILL CONTAINING THE UPDATES IS PICKED UP
/NOTE: THIS SUBROUTINE WILL PLACE A 1 IN EACH BIT POSITION...
/... CORRESPONDING TO EACH PIN TO BE UPDATED IN THE SPECIFIED...
/... TABLE. HENCE IN THE CASE OF AN INPUT SETLO ONLY, THE ...
/... RESULT MUST AGAIN BE XOR-ED WITH THE CONTENTS OF TEMP.8...
/... TO PUT A 0 IN INTAB FOR EACH SETLO PIN.
/OUTPUT REGISTERS: TEMP.8 - 1'S FOR EACH PIN TO BE CHANGED
/ TEMP.7 - ADDRESS IN TABLE
UPDATE, 0
        TAD YCOORD /ADD Y-COORDINATE TO TABLE ADDRESS
        DAC TEMP.7
        LAC FTRNO /FTR CODE NUMBER
        SAD (5) /P.A.W. FTR?
        JMP UPDA.1 /YES
        LAC TEMP.4 /X-Y COORDINATES
        AND (17) /ISOLATE X-COORDINATE
        DAC .+3
        LAC (400000) /1 IN BIT 0
        CLLROT /ROTATE ACCORDING TO X-COORDINATE
        0
UPDA.2, DAC TEMP.8 /CONTAINS A 1 IN THE PIN POSITION
        CMA
        DAC TEMP10
        LAC I TEMP.7 /PICK UP CONTENTS OF TABLE

```

```

        AND TEMP10      /CLEAR PIN POSITION
        XOR TEMP.8      /SET BIT IN PIN POSITION
        DAC I TEMP.7    /LOAD INTO TABLE
        JMP I UPDATE
UPDA.1, JMS NEXTEC      /PICK UP WORDFILL OF UPDATES
        JMP UPDA.2
/*****
/UPDAT1 -- ISS. 1 JAN. 28/71 PHT
/USAGE: JMS UPDAT1
/PRE-SET REGISTERS: TEMP.7 - ADDRESS IN INPUT LEVEL TABLE INTAB
/                   TEMP.8 - 1'S FOR EACH PIN TO BE CHANGED
/RESETS INPUT PINS TO 0 - SEE NOTE IN SUBROUTINE UPDATE
UPDAT1, 0
        LAC I TEMP.7
        XOR TEMP.8
        DAC I TEMP.7
        JMP I UPDAT1
/*****
/WAIT -- ISS.1 FEB.5/71 PHT
/USAGE: LOAD AC WITH COMPLEMENT OF NO.OF TIME UNITS OF WAIT
/       JMS WAIT
/I TIME UNIT = 0.1 MS
WAIT, 0
        DAC 7
        CLON
        JMP .           /WAIT FOR CLOCK INTERRUPT
        JMP I WAIT
/*****
/XYSTOR -- ISS. 2 AUGUST 30/71 PHT
/USAGE : JMS XYSTOR
/ISOLATES THE X-Y COORDINATES AND STORES IN TEMP.4
/SHIFTS Y-COORDINATE INTO BITS 12-17, ISOLATES AND STORES IN Y-COORD
XYSTOR, 0
        AND (177)      /ISOLATE X-Y COORDINATES
        DAC TEMP.4
        RTR
        RTR
        AND (77)       /ISOLATE Y-COORDINATE
        DAC YCOORD
        LAC ECSTOR     /PICK UP E.C.
        JMP I XYSTOR
START

```

APPENDIX B8-BIT AND TRIMMED 6-BIT ASCII CODES

The 8-bit code and resulting 6-bit code are shown with their corresponding symbols. Certain symbol definitions had to be changed in the case of the 6-bit code to include all required symbols within the 64-symbol set.

8-BIT CODE (TELETYPE INPUT/OUTPUT)		6-BIT CODE (CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
A	301	01	A
B	302	02	B
C	303	03	C
D	304	04	D
E	305	05	E
F	306	06	F
G	307	07	G
H	310	10	H
I	311	11	I
J	312	12	J
K	313	13	K
L	314	14	L
M	315	15	M
N	316	16	N
O	317	17	O
P	320	20	P

8-BIT CODE (TELETYPE INPUT/OUTPUT)		6-BIT CODE (CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
Q	321	21	Q
R	322	22	R
S	323	23	S
T	324	24	T
U	325	25	U
V	326	26	V
W	327	27	W
X	330	30	X
Y	331	31	Y
Z	332	32	Z
0	260	60	0
1	261	61	1
2	262	62	2
3	263	63	3
4	264	64	4
5	265	65	5
6	266	66	6
7	267	67	7
8	270	70	8
9	271	71	9
!	241	41	!
"	242	42	?
#	243	43	#
\$	244	44	Tab
%	245	45	%

8-BIT CODE (TELETYPE INPUT/OUTPUT)		6-BIT CODE (CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
&	246	46	&
'	247	47	carriage return
(250	50	(
)	251	51)
*	252	52	*
+	253	53	+
,	254	54	line-feed
-	255	55	-
.	256	56	.
/	257	57	/
:	272	72	:
;	273	73	;
<	274	74	<
=	275	75	=
>	276	76	>
?	277	77	Rub-out
@	300	00	end-of-field
[333	33	[
\	334	34	\
]	335	35	form feed
↑	336	36	↑
←	337	37	←
space	240	40	space

LIST OF ABBREVIATIONS

Bank	PDP-15 operates in a PDP-9/15 mode
CPU	Central Processor Unit of a computer
Dectape	Digital Equipment Corporation magnetic tape
DTL	Diode-Transistor Logic
EC	Elementary Command
FAW	Full Address Word
FTR	Functional Test Routine
HEX	Six
IC	Integrated Circuit
I.L.M.	Input Level Mode
IOT	Input/Output Transfer instruction
I/O	Input/Output
OS	Operating System
PAW	Partial Address Word
PCB	Printed Circuit Board
PI	Program Interrupt
TSI	Test Set Instructions (test program)
TTL	Transistor-Transistor Logic
VDC	DC voltage level

REFERENCES

- (1) Texas Instruments Incorporated: "TTL Integrated Circuits Catalog Supplement from Texas Instruments" 15. March 1970
- (2) General Instrument Corporation, Microelectronics Division: Data Sheets (Advance, November 1969) on the SL-6-4025 MTNS Quad 25-bit Static Shift Register
- (3) General Radio Corporation: "1790 Logic Circuit Analyzer" Brochure, July 1969
- (4) Computer Automation Incorporated: "CAPABLE Product Description", "CAPABLE FOUL Programming Language Description" November 1970
"Engineering Specification CAPABLE System" December 1970
- (5) Hewlett Packard Corporation: "2060 Digital Logic Module Tester" Brochure, 1970
"AUTEST - 2060A Software" Brochure
- (6) Digital Equipment Corporation: "PDP-15 Systems Reference Manual" DEC-15-BRZB-D 3rd Printing (Rev) December 1970
- (7) Digital Equipment Corporation: "PDP-15 Systems Interface Manual" DEC-15-HOAB-D 2nd Printing (Rev) February 1970
- (8) Digital Equipment Corporation: "PDP-15 Systems Operator's Guide" DEC-15-H2CB-D
- (9) Digital Equipment Corporation: "PDP-15 Systems User's Handbook Vol. 1 Processor" DEC-15-H2DB-D 2nd Edition (Rev) November 1970
- (10) Digital Equipment Corporation: "PDP-15 Systems User's Handbook Vol. 2 Peripherals" DEC-15-H2DB-D 2nd Edition (Rev) November 1970

- (11) BOYCE, A. H., EMMERSON, R. C., STRINGER, D. V., and WEST, B. G., "Simulation of Binary Logic Circuits by Digital Computers" The Marconi Review, Second Quarter, 1971
- (12) CALDWELL, S. H., "Switching Circuits and Logical Design" John Wiley & Sons 6th Printing, Feb. 1965
- (13) MALEY, G. A., EARLE, J., "The Logic Design of Transistor Digital Computers" Prentice Hall, Copyright 1963
- (14) CHANG, H. Y., MANNING, E. G., METZE, G., "Fault Diagnosis of Digital Systems" John Wiley & Sons (1970)