CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

COMPUTER AIDED SOFTWARE DESIGN

FOR TEST PROGRAMS

A project submitted in partial satisfaction of the requirements for the degree of Master of Science in
ENGINEERING

by

John Carl Colgan

January, 1974
The project of John Carl Colgan is approved:

Committee Chairman

California State University, Northridge

January, 1974
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section/Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Section 1.</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td>Section 2.</td>
<td>3</td>
</tr>
<tr>
<td>Computer Aided Design Philosophy</td>
<td></td>
</tr>
<tr>
<td>Section 3.</td>
<td>6</td>
</tr>
<tr>
<td>Survey of CASD Systems</td>
<td></td>
</tr>
<tr>
<td>Section 4.</td>
<td>11</td>
</tr>
<tr>
<td>Man-Machine Interface</td>
<td></td>
</tr>
<tr>
<td>Section 5.</td>
<td>14</td>
</tr>
<tr>
<td>A Personal Example: APA</td>
<td></td>
</tr>
<tr>
<td>Section 6.</td>
<td>34</td>
</tr>
<tr>
<td>Conclusion</td>
<td></td>
</tr>
<tr>
<td>Bibliography</td>
<td>36</td>
</tr>
<tr>
<td>Appendix 1.</td>
<td>38</td>
</tr>
<tr>
<td>Typical Examples of First Programming Aid</td>
<td></td>
</tr>
<tr>
<td>Appendix 2.</td>
<td>44</td>
</tr>
<tr>
<td>Conversational Test Program Preparation</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>Appendix 3.</td>
<td>66</td>
</tr>
<tr>
<td>Typical Examples of Second Programming Aid</td>
<td></td>
</tr>
<tr>
<td>Appendix 4.</td>
<td>70</td>
</tr>
<tr>
<td>PTS-100 Intelligent Terminal Descriptive</td>
<td></td>
</tr>
<tr>
<td>Brochure</td>
<td></td>
</tr>
</tbody>
</table>
ABSTRACT

COMPUTER AIDED SOFTWARE DESIGN FOR TEST PROGRAMS

by

John Carl Colgan

Master of Science in Engineering

January, 1974

Computer aided design systems have been a useful tool in the design of hardware for several years. It is just recently that CAD systems have been made available for use in software design. The purpose of this project was to survey the state of the art in computer aided software design systems and to detail the characteristics of three systems which the author has developed to assist in the generation of application test programs. A variety of CASD systems are discussed, and it is found that most of them are interactive systems. This requires the operator to be included as an integral part of the system and that the man-machine interface be carefully considered.

The three CASD systems for the generation of test programs uncover many of the problems that have been encountered in using the computer to aid design and show the evaluation of the solutions. They introduce the concept of generalized test procedures or test techniques which simplifies the automatic approach. These systems have
been used to generate several hundred tests.

The report is concluded by briefly discussing the future of computer aided software design systems.
SECTION 1. INTRODUCTION

Computers have provided a highly useful computational tool to man for approximately twenty years. Although development began some ten years ago, it is only in the past two or three years that the computer has become a widely used design aid tool. The purpose of this project is to investigate one aspect of computer aided design (CAD) and to discuss a series of projects I have been involved in during the past three years which relate to this topic.

First this report outlines the philosophy of computer aided design which shows the importance of computers in our society. This background provides insight into the topic of Computer Aided Software Design (CASD) which the rest of the report details. The next section discusses several examples with a brief description of how each system performs a CASD task.

The third section describes the different types of hardware systems which can be used to support a CASD system. It also examines the significance of the man-machine interface through which the operator makes use of the CASD systems. The next section is the most important of the report since it outlines the original work to which I have contributed in the area of computer aided software design. This section is a summary of three generations of computer aided application test program generation systems. Two of these systems have been used to generate several hundred application tests, while the most recent system is presently in the development phase. These systems represent the realization of two significant principles: the first is the use of generalized test procedures called test techniques to construct
application test programs, and the second is the automation of this process including detailed syntax checking.

The last section is a summary of the information presented in the report and provides comment on the future areas of possible development.
SECTION 2. COMPUTER AIDED DESIGN PHILOSOPHY

One of the most important problems to resolve is what is the role of the computer in the design process. The design problem is characterized by the following parameters: the alternatives are not well defined and numerous, there are many types of undefined impact, and the models are varied and complex. This leads to the resolution that the computer should be an extension of the operator used to determine the impact and consequences of the man-made decisions during the evolution of a design.

Computer aided design of hardware is the most developed and widely used form of CAD. Practically all electrical design engineers have had contact with some form of CAD in the design, analysis, and/or documentation of hardware systems. In the digital system development these CAD programs have been extensively developed in the areas of: gate level simulation, synthesis, partitioning, interconnect, and fault test generation. A variety of programs are also available for the design of continuous circuits. Although in some limited areas these CAD programs have a high degree of automation, it is in general true that the CAD operation is an interactive man-machine process in which the computer correlates and analyzes the man-made decisions.

Computer aided design of software has not reached the level of acceptance that CAD for hardware has, but a considerable amount of work has been done. Development in the CASD area has been towards making the programmers efforts easier. The first phase of this effort which has been called Automatic Programming was the development of compilers. This allowed the programmer to use a high level language
which is oriented towards his task instead of assembly or machine languages to do his programming.

Feldman\(^{(1)}\) in his paper on automatic programming suggested that there are two types. The first is the development of aids to program problems in a specific domain of interest. He argues that these offer the only contemporary practical programming aids. The other type has to do with systems to automatically synthesize programs. The impetus for this type of aid has been the development of theorem proving programs. The principle is if one can generate a system to prove a theorem derived from a given input-output relationship, then the program to satisfy that relationship is extracted from the proof. Feldman points out that these systems are a long way from providing any practical uses. It should be noted that none of these systems are possible with present technology without some form of man-machine interaction.

The other area of CASD that should be discussed is the automatic verification of software programs. This is an even more complex task since it involves not only developing a technique of automation, but also requires a definition (which can be agreed upon) of the meaning of verification. Usually some compromise must be made and verification is defined in terms of a level of confidence that given the right inputs then the proper outputs will be produced. The automation technique may involve comparison of execution parameters or analysis of the flow chart.

As can be seen from the preceding discussion, there are a wide variety of computer aided design applications. All of these systems are based on the computer being used to assist the human operator in the design process. It is also evident that due to limitations in technology and sophistication, most all CAD systems require a man-machine interactive process.
SECTION 3. SURVEY OF CASD SYSTEMS

The purpose of this section is to summarize several examples of existing computer aided software design systems. This survey is not all inclusive, but does represent a cross-section of the present technology. The first of these is a total programming system which has several facilities to aid in the construction of programs. This system, called ECL, is an interactive on-line system which permits one to compose, execute, compile, and debug programs.

The goals of ECL are to allow problem oriented description of algorithms, data, and control over a wide range of application areas; to facilitate program construction and debugging; to assist in the development of highly efficient programs; and to facilitate smooth progression between initial program construction and the final realization of an efficient product. The most important property of ECL is that it allows the programmer to tailor the working environment to suit his needs.

There was an attempt to extend ECL to construct an automatic programming system by Cheatham and Wegbreit but they soon found that the man-machine combination was necessary. They did point out that the experience gained with this system shows the need for specialized theorem provers so that more automation can be realized.

The next system to be discussed is a program to design instruction languages for computers called ISDS (Instruction Set Design System). This system was developed during a research project to create

techniques for writing programs that solve design problems without intervention by human designers. The ISDS approach is as follows: select a design concept - a model of solutions to the design problem, select a data structure for instances of the design concept, create operators that perform analysis and specify single parts of the model, and finally, create programs that use cost, value and constraint information from the statement of the problem to apply the operators in some sequence that results in a solution to the problem.

It was found that ISDS fell short of solving the practical design problem because of lack of a sophisticated search strategy. It was suggested that ISDS might be useful in doing some of the more routine tasks in an interactive design system.

Another tool that has been developed to assist the operator in the design process is problem oriented languages. These are languages designed for the user which are command oriented and are easy to learn and use. Some examples of these languages are COGO which is used in solving geometric problems, STRESS which is used for structural analysis, DYANA for dynamic system analysis, PACER for chemical processing, ENPORT for energy systems, and TELSIM for simulation of continuous systems. Meta-systems have been developed to aid in the design of these problem-oriented languages. An example would be ICES (Integrated Civil Engineering Systems) which was used to develop COGO and STRESS. In the use of these systems it has been found that graphic, interactive systems increase man-machine effectiveness.

The next CASD system which I will review is EMILY. EMILY is a system designed to aid in the construction and modification of computer programs written in higher level languages. The user constructs his
program by selecting choices from the menu of acceptable sequences of characters to replace certain symbols in the basic program. An example of this would be as follows:

Starting with: [STMT]
             END

Then selecting: DO [ARITH] = [ARITHX] TO [ARITHX]
                 [STMT]
                 END

Which after several steps becomes:
       DO I = 1 TO 20
            [STMT]
            END

And finally:  DO I = 1 TO 20
              S = S + A (1)
              END

The syntax rules for selection of the new character sequence is defined in Backus-Naur Form (BNF) notation. At each step the alternatives which are available to the operator are displayed so that selection can be made simply by indicating the desired choice. The man-machine interface for this interactive system is a graphic display terminal with a light pen and functional keyboard. The EMILY system is language independent which means that it can be used with any programming language which can be described in BNF notation.

Several systems have been developed using the theorem proving approach to automatic program synthesis. These include DEDUCOM, QA3, and PROW which construct a program satisfying a given input-output relation by proving a theorem induced by the relationship and
extracting the program directly from the proof. All three are very limited in capabilities and are unable to produce programs which use loops. It has been suggested by Manna and Waldinger (3) that the power of these programs could be improved by using the induction principle and by making them interactive systems.

Another important aspect of automated software tools is in the automatic verification of programs. Systems like FLOW and PACE do this by measuring the thoroughness of a program. The FLOW system identifies and counts all executable statements, instruments the program for test monitoring, adds a test monitoring software component, records the flow of control through the program, and displays trace and statement usage frequency data to summarize program operation. The evaluation criterion is the test effectiveness ratio (TER) which is defined as follows:

\[
\text{TER} = \frac{\text{No. of statements exercised at least once}}{\text{Total number of executable statements}}
\]

One of the problems with FLOW is that it is possible to exercise all statements without all branches being taken.

Another system called PACE (Product Assurance Confidence Evaluator) solves this problem by evaluating another TER which is the ratio of the number of branches exercised at least once to the total number of executable branches. PACE does this by identifying all modes and potential branches and then executes all logical paths while keeping track of the branches executed. Thus, it is able to more accurately measure the thoroughness of the program under test.

This section has not shown all parts of computer aided software

design, but it has provided a brief description of several different CASD systems. It is evident by the comments of the system originators that all of them consider their accomplishments only a start in developing the potential of the use of the computer to aid in the development of software programs.
SECTION 4. MAN-MACHINE INTERFACE

Although the purpose of this report is to discuss computer aided software design, I feel that it is necessary to at least briefly summarize the important aspects of the man-machine interface. No CASD system can be useful without a well developed communication link between the computer and the operator. This statement presupposes the principle that a CASD system must be interactive to be practical.

Many attempts at developing automatic design systems have been made in the past few years but none have proved useful except under very limited constraints. This is not to say that development should not continue towards automating more of the design task, but it does raise the issue as to whether man can be eliminated from the design system. For the present, the necessity of the interactive design principle is a philosophical question upon which many opinions exist. It is, however, quite evident for now, and probably the next few years, that all productive design systems will be interactive.

Thus, the importance of the man-machine interface has been established and the following significant properties exist. The first is the necessity to provide a rapid system response time. If the operator becomes bored or irritated at the time required to display information and process inputs, then the whole system will be inefficient. The display of information can be done either by hard copy printout or by CRT terminals. The first of these is usually cheaper and has the advantage of providing a permanent copy of the system's communication, but is inherently slow. A cost trade-off should be done for each system, but in general the system inefficiencies in slow communication
will off-set the higher initial cost of CRT displays. If permanent records are a necessity, then the CRT display can be augmented with a hard copy device.

In today's age of high speed computers it may seem unnecessary to discuss the effects of processing times, but this can be a problem. The most noticeable offender is the time share system. This is a system in which many input-output terminals share a single processor so that the cost per terminal can be reduced. There are two problems which should be carefully considered. They are the percentage of time the processor must spend in keeping track of its internal functions and the availability of communication lines which do not severely restrict the terminal input-output speed.

The second important property of the man-machine interface is the requirement that the communication device be easy to use and learn. It is highly desirable that the operator does not have to spend a significant amount of his talents in communicating with the system. This will again lead to system inefficiencies and could be detrimental to the design process.

The way to overcome this problem is to make all messages clear and precise, to prompt the operator as to his expected response, and to provide for functionally oriented controls. The system should be practically self-explanatory, and the operator should be continuously informed of the system status. In particular, all operator responses should be keyed so the operator is never in doubt as to his actions. The functional controls can be either a specially designed control panel or some mixture of standard keyboard and special function keys. The special control functions should be the commonly used items so
that a minimal amount of effort is required in making these responses.

This section has established the need for concern about the man-
machine interface, and has pointed out two important properties.
These are the requirement for rapid communication, and the necessity
of making the system easy to learn and use. Most important, the CASD
system originator must be very conscious of the human aspect of the
interactive system.
SECTION 5. A PERSONAL EXAMPLE: APA

One of the reasons I selected computer aided software design as the topic of my graduate project was my own personal involvement in one aspect of CASD. I am presently working for Raytheon, Missile Systems Division, and one of our tasks is to develop test equipment for the various missiles. Approximately three years ago, we were involved in the design and development of a paper tape controlled automatic test system. One of the tasks I had was to develop the programming language and to assist the applications programmers in writing the missile test programs.

It seemed to me unduly tedious for the programmers (who were actually systems engineers with a strong understanding of the missile) to be forced into learning the specialized programming language. As a result of this, I instigated an assignment to develop a programming aid. The nature of this aid was to interactively question the programmer from a time share computer system as to the requirements of a given test step. The questions were arranged such that the answers could be automatically converted into a working code.

The questions were in English as were the responses, and the length of the answers were kept short. Every attempt was made to make the questions as relevant as possible, and the answers easy for the programmer to enter. An example of the type of questions is as follows: "Would you like to program a measurement instrument?" If the response was "yes", the program would continue to question the operator as to which instrument was to be programmed and what functions were to be controlled. If "no", the operator would be questioned on
the next area of concern. Two typical examples are shown in Appendix 1 and included at the end of each is a listing of the automatically generated, executable machine code. If the operator did not know the answer to a question or answered incorrectly, the program would automatically list the acceptable answers and then repeat the question.

The advantages to this system were that the programmer did not have to learn a special programming language and the structure of the application program was forced to be consistent with test system hardware requirements. It allowed the programmers to focus their attention on the problems and requirements of test procedures, and not the method of implementation.

There were two major disadvantages to this programming aid which eventually led to a second generation programming aid. The first of these was that the time required to communicate with the program was too long. This was due to the slow data transfer rates (ten characters per second) and the excessive program execution delays when the time share system was crowded (most of the working day).

The second problem was the special purpose nature of the programming aid. It would only work for writing applications programs for one particular automatic test system. It was difficult to make changes to the questions, acceptable answers, and the code transformation which was automatically made. This system was used to generate approximately 200 missile tests, but is no longer being used.

The second generation programming aid was developed by a software development company under my direction and specifications. It was written to operate on an RDS 704 minicomputer using three CRT terminals working in a time share mode as the operator interface. Due to
project fund limitations, the system has only been implemented with one CRT terminal at present. Included in Appendix 2 is a partial copy of the operating manual, but the following discussion will summarize the operating procedures.

This programming aid which is called the Conversational Test Program Preparation System was designed to allow the on-line preparation, completion and modification of application test programs. It works on the principle of displaying to the programmer an uncompleted test technique which can be filled in, edited, and then stored on magnetic tape for final processing. The test technique concept is one developed by Raytheon in an attempt to minimize the cost of generation and debug of application test programs.

We found that if one would analyze any given test program, that several groups of commonly used procedures could be identified. An example of one such group would be the requirement to make a measurement of a DC voltage on a given test point with a specified AC voltage stimulus and resistive load. Another example might be the requirement to make an AC voltage measurement on a specific output pin while two different DC voltages are being applied to different input pins.

It might seem that the number of groups, or as we call them, test techniques, required for testing a given system would be very close to the number of total tests required. This has not been the case for the system we have analyzed, and in fact, normally the number of test techniques required is less than ten percent of the total number of tests. Caution should be taken in generalizing this principle since it is only based on limited empirical evidence and has no theoretical proof. Thus, each system should be analyzed independently for
Guidelines which can be used are that the principle normally works better for systems which have several hundred total tests and is constructed of modules implemented with a common set of devices (i.e.; primarily transistors or digital integrated circuits of a given family, etc.).

The test technique philosophy has the additional benefit of being compatible with automation tools, and this was the basis for our second programming aid. Once a set of test techniques has been developed for a given system then the test program can be written by selecting the applicable test technique for a given test and then adding the information necessary to complete the test. This information is generally in the form of specific voltage levels, test point numbers, tolerance values, etc. The Conventional Test Program Preparation System works by displaying the selected technique and allowing the programmer to enter the test information via the CRT terminal keyboard. The system automatically compares the input data against a set of acceptable parameter data which is keyed to the limitations of the test language and test system hardware.

The type of syntax and range checking which can be performed on the input data is keyed to four kinds of parameter fields: Alphanumeric, Alpha, Integer and Decimal. The alphanumeric parameter fields can be filled in with any displayable character, or if properly defined any displayable character string which matches a preset acceptable parameter table. The alpha parameter field is identical to the alphanumeric except that any numeric characters will cause an error.

The integer and decimal fields both can only contain numeric characters but differ in that the decimal parameters must have a
decimal point while the integer cannot have one. Both maybe specified to be only certain values or to be acceptable only between some given maximum and minimum with or without a specified step size.

There are three modes of operation in the Conversational Test Program Preparation System and they are as follows: Procedure Preparation, Procedure Completion, and Procedure Modification. The procedure preparation mode allows the operator to specify the desired test technique, assign a test and step number to the completed test, fill in the test information, and then store the syntax checked test onto magnetic tape.

If all the test information is not available when the programmer initially generates a test, the procedure completion mode maybe used to recall the uncompleted test. The test is displayed to the operator with the incompleted parameters indicated. The operator can then complete the test and then store it on the magnetic tape. In the procedure completion mode, no syntax checking is performed.

In the third mode, procedure modification, the operator can recall any test that is in the test storage magnetic tape. The test is displayed to the operator and desired change can be made. Examples of the CRT display for each of the modes are shown in Appendix 3.

We are presently in the process of generating 800 tests which will be used in a computer based automatic test system. The Conversational Test Program Preparation System has been very useful in reducing the test generation time, but does have some limitations. One of the problems we have encountered is that the generation of the coded test technique is an error prone procedure. In this system the test technique must be coded off line in a special format and input to
the computer on IBM cards.

Another problem is the special format that the parameters within a test technique must have. The limitation in format is that each parameter must be followed by a field delimiter (i.e.; comma, right parentheses, or carriage return). This was done to simplify the system software and was consistent with the initial requirement that the application programs would be coded in the BASIC language.

The last major problem was the lack of syntax dependence between the parameters. It is a facet of almost every test language that the value of one parameter will affect subsequent parameters. This limitation has constrained the completeness of the syntax checking capabilities of the Conversational Test Program Preparation System.

We are presently in the process of constructing the third generation computer aided software design system which is called Application Programming Aid (APA). This system is being developed to eliminate the problems which we have encountered in the previous programming aid systems. The APA system uses a new concept in hardware which greatly simplifies the software required to implement this type of man-machine system.

The hardware is a PTS-100 Intelligent Terminal produced by Raytheon Data Systems (see Appendix 4 for descriptive brochure). The significant difference in this type system is that the CRT terminal and the processor are an integral pair. They both share the same memory such that to display a character on the CRT requires only an update to the memory allocated for the screen refresh storage. When a key is depressed on the keyboard it updates a certain location in the memory which can be interrupted by the processor in any desired sense.
This processor/display marriage allows for a more generalized programming aid. The APA system is being generated to allow applications programmers to fill in test techniques in constructing test programs but it could also be used in a variety of other applications. These might include inventory control, order taking, billing, or practically any information management system in which fixed formats are being filled in with variable data.

The following section describes the operating characteristics and user interaction of the APA system. There are three basic operating modes: Fill-in, Modify, and Generate. The fill-in mode is used to put data into existing test techniques for the purpose of generating a test. The modify mode is for making modifications to existing tests while the generate mode is used to construct new test techniques.

The following is a description of the steps required to use the APA system:

1. Load the APA program cassette into the cassette reader. This must have the Absolute/Relocating Loader in front of the APA program.

2. Push the IPL (Initial Program Load) button on the PTS-100 processor. This initializes the system.

3. The operator is now requested to select one of the operating modes by depressing the appropriate mode switch. The mode indicator will light up when the mode is selected.

4. If the Fill-in mode has been selected the following will occur:
   a. A message will be displayed requesting the operator to enter the test technique number which he desires to fill.
b. Depressing the index button will display a table of the test techniques available with a brief description of its characteristics. The system will expect the operator to input the desired test technique number.

c. The screen will now be filled with the 1st twenty lines of the selected test technique. The cursor will be placed at the first blank parameter ready for the operator to fill in the blank. The operator may do this in two ways: first, by using the alphanumeric keys to enter the parameter; and second, by depressing the acceptable parameter button.

d. If the operator chooses the first approach, he may enter any character available to construct the parameter (the cursor controls which would move the cursor out of the blank area will be disqualified). When this is completed, the cursor can be advanced to the next parameter by depressing the tab button. Depression of the tab will cause the processor to do a syntax check of the entered parameter. If an error has occurred, a message will be displayed to the operator, the error light will be lit, the cursor will be placed at the beginning of the blank parameter where the error occurred, and the operator will be given another chance to fill in the parameter.

e. If the operator chooses the second method, a message will be displayed to the operator explaining the acceptable inputs for this parameter. The manner in which the
operator may proceed depends on the parameter type. The following is a discussion of the parameter types and the means to enter them:

(1) Numeric: The parameter is any decimal number (with or without a decimal point) up to 10 characters long. It may be in exponential form (i.e., $-10.50 \times 10^{15}$); but the exponent is limited to two digits. The parameter must be within a specified minimum and maximum value with a given increment. The cursor is placed at the beginning of the blank and the operator enters the number. If an error occurs, the operator is notified and he may try again.

(2) Alphanumeric: The parameter can take two forms; one is that the parameter can be comprised of a string of any acceptable characters, the other is that it can be only those character strings that are part of a specified acceptable list. In the case of a specified list, all the acceptable strings will be displayed to the operator. The cursor will be placed by the first string in the list and the operator could select that item to be put into the parameter by depressing the enter parameter button. If the operator desires to use another member of the list, the cursor can be positioned next to the desired string by using the cursor up one line and cursor down one line. Once the cursor is properly positioned, the operator need only depress the enter
parameter key. The cursor cannot be moved out of the list. The operator may leave the parameter blank by depressing the tab key.

There are some alphanumeric list parameters which have variable numeric parts which, if all elements were included in the list, would cause the list to be excessively long. In this case, the element in the list will have a special character (#) in place of the numbers.

The number will be specified with some minimum and maximum values with a given increment. An example of an acceptable list would be as follows:

```
A1
VALUE
DATA (##) RANGE 00 to 99 in 01 STEPS
DATA (1)
##### RANGE 0.0 to 100.0 in 0.1 STEPS
500
1000.0
```

If the operator selected a string which had one of the special characters, then when the enter parameter key was depressed the element would be entered into the blank space with the special characters intact. The cursor would be placed under the 1st special character and the operator would fill it in as any other numeric parameter. The special character field can be used in only one place in each
element of the list (i.e., #J101-## is not a legal string).

f. In addition to doing syntax checking on each parameter, it is desirable to define certain acceptable relations between the parameters. This is accomplished by allowing the acceptable fillin for one parameter to depend on the contents of previous parameters. This relationship is such that for a given parameter, each acceptable parameter may be conditioned to the existence of a particular value(s) for a previous parameter(s). An example of this is as follows:

<table>
<thead>
<tr>
<th>PARAMETER X</th>
<th>CONDITIONAL EQUATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value 1</td>
<td>(PARA A, VALUE 1 - PARA B, VALUE 2) + PARA C, VALUE 1</td>
</tr>
<tr>
<td>Value 2</td>
<td>PARA B, VALUE 1 + PARA C, VALUE 1</td>
</tr>
</tbody>
</table>

This means that value 1 would be acceptable fillin for parameter X if and only if parameter A is value 1 and parameter B is value 2, or parameter C is value 1. Of course parameters A, B, and C must be prior to parameter X. There could exist a condition for which there would be no acceptable answer for a given parameter based on what the operator had put into the preceding parameters. This condition should be recognized as an error and be signified by a special error message. This will require the operator to go back and change some of
the previous parameters.
When the values of the previous parameters upon which a given parameter depends indicate that one and only one value of the given parameter is permissible, then that value will be automatically assigned to the given parameter. If the value is a numeric range, then the operator must enter the proper digits.

g. Another important interrelationship is the association between lines. This is defined as being able to associate a given line with another line such that it is an error to erase one and not the other. This line checking relationship should have chaining properties such that one line may be associated with another, and so on. All lines must be numbered but it is optional as to whether the number is a part of the source code or not.

h. It is possible to repeat any line that is displayed on the screen by positioning the cursor to the 1st position in the line and depressing the repeat line key. This will cause that line to be repeated and all the remaining lines on the screen to shift in order to make room. The parameters in the line repeated will have the same syntactical rules as in the original line.

i. After all parameters are filled in, the operator, by means of cursor and screen controls, can position
the cursor to any parameter and change its contents. All the rules and procedures in steps 4d thru 4h are applicable.

j. When the operator is completely satisfied with the test, he can store it in the tape cassette by depressing the store new key. When this is done, a message will be displayed requesting that the operator enter the test number label which is desired for this test. The operator will then enter the required digits.

5. If the modify mode has been selected, the following will occur:

a. A message will be displayed requesting that the desired test number be entered.

b. The screen will be filled with the 1st twenty lines of the selected test and the cursor will be placed at the 1st character of the 1st parameter.

c. The operator can change the parameter as he desires under the constraints imposed by steps 4d thru 4h.

d. If the operator desires to change the data in the protected areas, he may do so by pressing the protect disable key. This will cause the protect disable indicator to come on and will allow the operator to position the cursor anywhere on the screen to make changes. The
syntax checking in steps 4d thru 4h will still be in effect.

e. Once all changes are made, the modified test may be stored on the magnetic tape under the original number by depressing the store same key or put under a new test number by depressing the store new key.

f. Depressing the store new key will cause a message to be displayed requesting that the operator enter the test number label which is desired for the new test. The operator will then enter the required digits.

6. If the Generate mode has been selected, the following will occur:

a. The following general instructions will be displayed to the operator:

You may type any textual data desired on the screen in any format. In place of data which you want to be filled in later, the following special characters should be used:

- Alphanumeric (Any character) A
- Numeric N
- Alphanumeric (special list) L

Each of these fill-in areas (or parameters) must be preceded with the depression of the nonprint define parameter key and must conclude with the depression of the tab key. The operator will be queried by the system after the tab key is depressed, as to the required syntax rules to meet the requirements.
specified in steps 4e thru 4g.

b. After all information has been input, the test technique is entered into the library by pushing the update library key. The operator will be queried as to what the number of the new test technique is and what brief description of its characteristics should be put in the index table.

7. The mode switches will be controlled such that the following rules apply:

a. A system ready condition shall be defined as the condition of the system after the initial load or after depressing the Abort Key.

b. Transfer to the Generate Mode can only be done from the system ready condition.

c. Transfer to the Fill-in Mode can be done from the system ready condition or Modify Mode.

d. Transfer to the Modify Mode can be done from the system ready condition or Fill-in Mode.

It is desirable for the APA system to be simple to use and easy to learn to use. This goal will allow the system to be used by untrained personnel and yet provide an efficient means of generating syntax-error-free test program code. In order to accomplish this goal, a set of special keys have been defined to augment the standard keys available with the hardware.

The special keys required for the PTS-100 system when used in the APA configuration are defined as follows:

ADD LINE: Moves all the screen below the cursor down one line
with the last line on the screen being erased. The line the cursor is in is moved down one line, thus leaving the cursor in a blank line. Any textual data may be put into this line (i.e., the entire line becomes one parameter with any characters acceptable in the Fill-in and Modify Modes and just a blank line in the Generate Mode). Note: any parameters put in the blank line in the Generate Mode will cause all other previously assigned and subsequent parameter numbers to be updated.

**STORE NEW:** Causes the test which is presently in working memory to be stored on the magnetic tape cassette under a new test number. The operator will be queried as to the desired number. This key is nonoperative in the Generate Mode.

**STORE SAME:** Causes the test which is presently in working memory to be stored on the magnetic tape cassette under its existing test number. The old version should be eliminated from the file index. This key is nonoperative in the Generate Mode.

**ROLL FWD:** Moves all lines on the screen up one line. The top line is lost and the bottom line becomes the next line in the working storage.

**ROLL REV:** Moves all lines on the screen down one line. The bottom line is lost and the top line becomes the next line in the working storage.

**ACCEPT PARA:** This is used in the Fill-in and Modify Modes to
request the system to display to the operator a list of acceptable answers for a given parameter.

This key is nonoperative in the Generate Mode.

ENTER PARA: If the operator has requested a list of the acceptable parameters, then he may choose to enter one of the elements in the list by positioning the cursor to the desired element and then using this key to enter the parameter. This key is nonoperative in the Generate Mode.

UPDATE: This is used only in the Generate Mode to enter a new test technique into the test technique library which is on the magnetic tape cassette. It will cause the operator to be queried as to the identification of the new test technique.

INDEX: This is only used in the Fill-in Mode and is operable only when the operator is asked to enter the test technique number. At this point, depressing this key will cause a list of the available test techniques in the test technique library with a brief description of each to be displayed to the operator.

ABORT: This key may be used at any time to cause the system to return to the system ready state. The contents of working memory will be lost.

DEFN PARA: This is used only in the Generate Mode to indicate when a parameter field is desired. It must be depressed before the special characters are entered.
| **PROTECT:** | This is used only in the Modify Mode to enable and disable the protect areas in the test technique. Thus, this allows the operator to alter the data in the non-parameter fields in a test technique. A light on the display front panel will indicate the status of the protect function. |
| **REPEAT LINE:** | This is used to repeat the line in which the cursor is located. The syntactical rules in the new line will be the same as those of the original line. |
| **FILL-IN:** | This puts the system in the Fill-in Mode. It is nonoperatable in the Generate Mode. |
| **GENERATE:** | This puts the system in the Generate Mode. It is nonoperatable in the Fill-in and Modify Modes. |
| **MODIFY:** | This puts the system in the Modify Mode. It is nonoperatable in the Generate Mode. |

There are some other function keys which will be given special meaning. These keys are on the standard keyboard and will be defined as follows:

| **TAB:** | This key will be used by the operator to indicate to the system that he is finished with the parameter area that the cursor is presently in and he is ready to move on to the next area. |
| **PRINT:** | This key will be used by the system when the PTS-100 is interfaced to the RDS-704. It will initiate a routine to select a test or test technique which is on the cassette and transfer that to the 704, which will output the data on the line printer. The |
operator will be required to input the desired test or test technique number.

LOCAL/REMOTE: In the local condition, the PTS-100 will ignore the actions of the 704. While in the remote condition, the PTS-100 will assume that it is on-line with the RDS-704. The status of the Local/Remote condition will be displayed to the operator via lights on the display panel.

START: Nonoperative in every mode.

The APA system offers several advantages over the previous programming aids. One of these is the fact that the system can operate independent of any other support systems. The test techniques can be generated on the system and then stored into a magnetic tape cassette. The operator can then recall a test technique, complete the test, and store the results on another tape cassette. This cassette can be used to transmit the test program source code to the test station computer.

Another advantage of the APA system is the generality of the syntax checking rules. Almost any test language can be used as the source code of the test techniques and, thus the APA system can be used to generate test programs for practically any test system. Perhaps the most important property of the APA system is the simplicity of the man-machine interface. There are clearly labeled special control keys and lucid operator instructions such that an untrained operator can operate the system with ease. This is a necessity since the most capable person for defining the applications test program is either the design engineer or test engineer with a strong understanding of the unit under test, but without sufficient knowledge of the
test language or test system.

The development of the APA system should be completed by the end of the year, and by next year will provide a useful computer aided software design tool for generating application test programs.
SECTION 6. CONCLUSION

This report has shown how computers can be used to assist a human operator in the design of software programs. It has briefly summarized several examples of computer aided software design systems which represent a variety of different aspects of the current work in CASD. The interactive system properties have been analyzed and it has found that with present technology this is the only practical approach to CASD.

A detailed description of my experience with CASD systems has been presented. These three systems have exposed some of the many problems with CASD systems, and have shown how evolution has lead to an improved approach. The third generation system makes use of the properties of the interactive principle, rapid input-output response times, and functionally oriented control interface. Thus, it provides a generalized system which can solve a large selection of software design problems.

Finally, it is desirable to reflect on the challenges that our past and present accomplishments have produced. We have only but begun to uncover the significance of computer aided design. It may be found that as the machine performs more of the mundane design tasks, that the human designer will be free to be more creative. In fact the role of man in the design process may be completely redefined, but it is incumbent on the CASD system originator to insure that this evolution is orderly and consistent with the practical design needs. The cost of software design in relation to that of hardware design is increasing, and in order to keep an economic balance, the software
designer must be provided with the necessary design aid tools.
BIBLIOGRAPHY


<table>
<thead>
<tr>
<th></th>
<th>Author(s)</th>
<th>Title and Source</th>
</tr>
</thead>
</table>
APPENDIX 1. TYPICAL EXAMPLES OF FIRST PROGRAMMING AID
>LINK RATELO1
WOULD YOU LIKE TO START A NEW TEST TAPE? YES
WHAT IS THE NEW TEST NUMBER? TSG 17E
WHAT IS THE NEXT TEST NUMBER? XXX? IF NO MORE TESTS - TYPE NONE.001
DO YOU WANT A GENERAL RESET? YES
WOULD YOU LIKE TO PROGRAM THE OPTICAL READER? NO
WOULD YOU LIKE TO PROGRAM A MEASUREMENT INSTRUMENT? NO
WOULD YOU LIKE TO PROGRAM A STIMULUS INSTRUMENT? NO
WOULD YOU LIKE TO PROGRAM A FUNCTION IN THE UUT ADAPTER? NO
WOULD YOU LIKE TO PROGRAM ANY MISC SWITCHING? NO
WOULD YOU LIKE TO ROUTE A SIGNAL? YES
WHAT IS THE STATION DANGLE SET-UP? 1
**** NOTE - END ALL MATRIX ANSWERS WITH A PERIOD. ****
IS THE PATH MISSILE-TO-INSTRUMENT? YES
WOULD YOU LIKE TO SET OR RESET? SET
MISSILE TEST POINT - 2E118.
INSTRUMENT - DVML -
YOU FORGOT TO END WITH A PERIOD - TRY AGAIN.
INSTRUMENT - DVML -.
IS THE PATH MISSILE-TO-INSTRUMENT? NO
IS THE PATH INSTRUMENT-TO-INSTRUMENT? YES
WOULD YOU LIKE TO SET OR RESET? SET
FIRST INSTRUMENT - ACM LO.
SECOND INSTRUMENT - ACM HI.
INSTRUMENTS SPECIFIED CREATED IMPOSSIBLE PATH OR ACCEPTABLE BUSS LINES ARE BUSY.
IS THE PATH INSTRUMENT-TO-INSTRUMENT? YES
WOULD YOU LIKE TO SET OR RESET? SET
FIRST INSTRUMENT - ACM LO.
SECOND INSTRUMENT - COUNTER LO.
IS THE PATH INSTRUMENT-TO-INSTRUMENT? NO
WOULD YOU LIKE A TIME DELAY OR HALT PRIOR TO SET-UP COMPLETE? YES
TIME DELAY OR HALT? HALT
WOULD YOU LIKE A TIME DELAY OR HALT PRIOR TO DATA MEASUREMENT? NO
WOULD YOU LIKE TO TRIGGER A MEASUREMENT INSTRUMENT? YES
WHAT INSTRUMENT WOULD YOU LIKE TO TRIGGER? ACVM1
WOULD YOU LIKE TO TRIGGER A MEASUREMENT INSTRUMENT? NO
WOULD YOU LIKE A TIME DELAY OR HALT? YES
TIME DELAY OR HALT? TIME
HOW LONG? XXX.XX SECONDS (DON'T TYPE DECIMAL POINT) 00100
WOULD YOU LIKE TO PRINT A DATA VALUE? YES
WHAT INSTRUMENT WOULD YOU LIKE TO PRINT? VOLT1
WOULD YOU LIKE TO PRINT A DATA VALUE? NO
WOULD YOU LIKE TO COMPARE A DATA VALUE? NO
WOULD YOU LIKE TO MAKE ANOTHER MEASUREMENT? NO
WOULD YOU LIKE TO CHANGE ANY OF THE TEST SET-UP DURING EXECUTE PHASE? NO
WOULD YOU LIKE AN NEW TEST STEP? NO
WHAT IS THE NEXT TEST NUMBER? XXX? IF NO MORE TESTS - TYPE NONE. NONE
WOULD YOU LIKE AN UNFORMATTED LISTING AND PAPER TAPE OF THE TEST TAPE? NO
WOULD YOU LIKE A FORMATTED LISTING? YES
Would you like to restart this program? No

Would you like to develop a new test tape? Yes

What is the new tape number? TSG17E

What is the next test number - XXX? If no more tests - type none. 001

Do you want a general reset? Yes

Would you like to program a stimulas instrument? No

Would you like to program a measurement instrument? No

Would you like to change a signal path? No

Would you like to program a special test criteria? No

Would you like to program the optical reader? No

Would you like a time delay or halt prior to set-up-complete? Yes

Time delay or halt? Time

How long? XXXVXX seconds, where V indicates assumed decimal point. 00100

Would you like a time delay or halt prior to data measurement? Yes
TIME DELAY OR HALT?
?HALT
WOULD YOU LIKE TO TRIGGER A MEASUREMENT INSTRUMENT?
?YES
WHAT INSTRUMENT WOULD YOU LIKE TO TRIGGER?
?VMA
WOULD YOU LIKE TO TRIGGER A MEASUREMENT INSTRUMENT?
?NO
WOULD YOU LIKE A TIME DELAY OR HALT?
?YES
TIME DELAY OR HALT?
?TIME
HOW LONG? XXXVXX SECONDS, WHERE V INDICATES ASSUMED DECIMAL POINT.
?45000
WOULD YOU LIKE TO PRINT A DATA VALUE?
?YES
WHAT INSTRUMENT WOULD YOU LIKE TO PRINT?
?CTA
WOULD YOU LIKE TO PRINT A DATA VALUE?
?NO
WOULD YOU LIKE TO COMPARE A DATA VALUE?
?YES
WHAT INSTRUMENT WOULD YOU LIKE TO COMPARE?
?CTB
DO YOU WANT TO COMPARE THE FOUR MOST/LEAST SIGNIFICANT DIGITS?
?MOST
IF NOGO, WOULD YOU LIKE TO JUMP TO ANOTHER TEST?
?YES
WHAT IS THE TEST NUMBER?
?189
WOULD YOU LIKE TO COMPARE A DATA VALUE?
?YES
IS THERE NEW LIMITS?
?YES
WHAT IS THE NEW HIGH LIMIT - XXXX?
?2909
WHAT IS THE NEW LOW LIMIT - XXXX?
?0000
WHAT INSTRUMENT WOULD YOU LIKE TO COMPARE?
?SDA
DO YOU WANT TO COMPARE THE FOUR MOST/LEAST SIGNIFICANT DIGITS?
?LEAST
ANSWERS - MOST OR LEAST DO YOU WANT TO COMPARE THE FOUR MOST/LEAST SIG
?LEAST
IF NOGO, WOULD YOU LIKE TO JUMP TO ANOTHER TEST?
?NO
WOULD YOU LIKE TO COMPARE A DATA VALUE?
?YES
IS THERE NEW LIMITS?
?YES
WHAT IS THE NEW HIGH LIMIT - XXXX?
?1000
WHAT IS THE NEW LOW LIMIT - XXX?
?0023
WHAT INSTRUMENT WOULD YOU LIKE TO COMPARE?
?DDA
DO YOU WANT TO COMPARE THE FOUR MOST/LEAST SIGNIFICANT DIGITS?
?OST
ANSWERS - MOST OR LEAST DO YOU WANT TO COMPARE THE FOUR MOST/LEAST SIGN
?MOST
IF NOGO, WOULD YOU LIKE TO JUMP TO ANOTHER TEST?
?YES
WHAT IS THE TEST NUMBER?
?456
WOULD YOU LIKE TO COMPARE A DATA VALUE?
?NO
WOULD YOU LIKE TO MAKE ANOTHER MEASUREMENT?
?YES
ANSWERS - YES OR NO
WOULD YOU LIKE TO MAKE ANOTHER MEASUREMENT?
?YES
ANSWERS - YES OR NO
WOULD YOU LIKE TO MAKE ANOTHER MEASUREMENT?
?NO
WOULD YOU LIKE TO CHANGE ANY OF THE TEST SET-UP?
?NO
WOULD YOU LIKE TO ADD ANY COMMENTS?
?YES
WHAT IS THE COMMENT? - IF NO MORE, TYPE NONE.
?GOOD TEST
WHAT IS THE COMMENT - IF NO MORE, TYPE NONE.
?NONE
WHAT IS THE NEXT TEST NUMBER - XXX? IF NO MORE TESTS - TYPE NONE.
?NONE
WOULD YOU LIKE A PRINT OUT OF THE TEST TAPE?
?YES
IF YOU WANT A PAPER TAPE ALSO, TURN ON PAPER TAPE PUNCH, DEPRESS
REPEAT KEY (REPT) AND RUBOUT KEY AT THE SAME TIME UNTIL ENOUGH LEADER
TAPE IS PRODUCED, THEN HIT THE CARRIAGE RETURN KEY.
?
*102
#001
GOOD
ROO
PTDA11,20,32
$
H
IVMA
IVMB
PTDA14,25,34
PMPA25
IPRA
PMPA25,21
ICMA
S
JMP189
RCMA00
PCMA02,19,30
PMFA25,22
ICMA
RCMA00
PCMA01
PC22,33
PMFA25,21
ICMA
S
JMP45
APPENDIX 2. CONVERSATIONAL TEST PROGRAM PREPARATION MANUAL
Section I

PURPOSE

This system is designed to allow the on-line preparation, completion, and modification of computer controlled test procedures. Master test techniques are input from cards and are stored on magnetic tape in an incomplete state. Each technique is written as a separate record on tape to allow an orderly search and retrieval of individual tests as requested by an operator. Test fill-in parameter tolerances and increments are also read from cards and are stored with identifying information in core memory to be used in syntax and range checking. Upon completion of initialization and set-up, the system requests via a CRT terminal, operator input designating one of three operating modes:

1. Procedure preparation (section 2.3)
2. Procedure completion (section 2.4)
3. Procedure modification (section 2.5)

and processing begins. The system utilizes a time-share technique, allowing procedure composition from three terminals simultaneously. On completion of the procedure manipulation, an entry is made in the directory as to test number and step number. The new procedure is then written as a single record to the end of the procedure file, followed by an end-of-file mark and the current directory. The output tape is then rewound. During composition of test procedures a limited syntax and parameter range check is performed to limit the clerical errors made by the operator.
Section II

METHOD

As previously stated, this system operates in a time-shared mode, allowing procedure manipulation from up to three terminals simultaneously. In order to get the system started an initialization procedure is required.

2.1 INITIALIZATION

The 704 console teletype is used to initialize the system since the CRT displays are initially inactive. The initialization procedure consists of building a master test technique tape, inputting the parameter data into memory, and indicating which CRT terminals are to be activated. If a master test technique tape has already been built or the parameters are already core resident, then these steps of the initialization procedure may be bypassed. When using a previously generated master test technique tape, the procedures in the second file of the tape may be also used. This is determined from the request for directory question.

The initialization process proceeds as follows from the system teletype.

A. ARE PARAMETERS TO BE INPUT?
   Y - Build parameter table from cards
   N - Use existing parameter table in memory

B. ARE MASTER TECHNIQUES TO BE INPUT?
   Y - Build library tape from cards
      Initialize procedure file empty and proceeds to step D
   N - Use previously generated tape
C. IS PROCEDURE DIRECTORY TO BE INPUT?

Y - Search to second EOF and read procedure directory
N - Procedure directory and file will have no entries initially

D. WHAT CRT'S ARE TO BE MADE ACTIVE?

1, 2, or 3 - User inputs any combination to indicate which CRT's will be activated

After the last step, memory buffers are allocated according to the number of terminals requested and the requested terminals are activated. Thus where one terminal is activated up to 12,000 bytes are available for that terminal. However, when all three terminals have been activated only 4,000 bytes per terminal is available.

Note that the master test techniques and parameters are maintained on card decks. However, it is not necessary to input the card decks every time the system is initialized. For example, a previously produced magnetic tape may be used for the master test technique. If the program and parameters are still in memory, then the parameter deck need not be reread. Also it is possible to use a master test technique tape that contains previously prepared test procedures. If it is desired to retain this information on the tape, then the user should request that the directory be input from the tape. Newly prepared procedures will then be written at the end of the tape, thus preserving those already written on the tape. Otherwise, the directory will be developed from scratch and new procedures will be written immediately after the first EOF on the tape.

2.2 SET UP FROM TERMINALS

Once the initialization procedure has been completed, the user may begin normal operations from one of the activated CRT terminals.
Set up at the terminal consists of answering several questions which the system displays on the CRT screen. These are described below:

A. **MODE? @@**
   
The operator then types in two letters to specify one of the three following modes:
   
   - **PP** - Procedure Preparation
   - **PC** - Procedure Completion
   - **PM** - Procedure Modification

B. **TEST TECHNIQUE RECORD NUMBER? ####**
   
   This message is displayed if the PP mode has been selected. The operator must then input the record number of the test technique that is to be used. The set up will then proceed to Item D below.

C. **TEST PROCEDURE INPUT RECORD IDENTIFICATION:**
   
   - **RECORD NO. ####**
   - **TEST NO. ####**
   - **STEP NO. ####**

   If the PC or PM mode has been selected, this message is displayed instead of the message in step B. The operator must input either a record number or a test and step number of the procedure that is to be completed or modified. The program will consult the directory to find the number in the second file that contains the specified test and step number. If a record does not correspond to the specified test and step number, step C is repeated. Otherwise the program proceeds to step D. If a record number has been input, the program searches to the specified record in the second file.

D. **TEST PROCEDURE OUTPUT IDENTIFICATION:**
   
   - **TEST NO. ####**
   - **STEP NO. ####**

   These data are used to identify the output record. If the mode is
PC or PM, the set-up phase is terminated with this step.

E. INITIAL LINE NUMBER? ####

The user inputs a four digit number which represents the first line number of the program.

F. LINE NUMBER INCREMENT? ####

The user inputs a four digit number that represents the automatic increment to be applied to each line number after the first.

Normally 10 is used.

Note that in step B a record number specifies a record in the first file, while in step C the record number specifies a record in the second file. The record number in step C may be omitted and a test and step number input instead. In this case the directory is consulted to find the correct record. If more than one record contains the same identification, the last complete record on the tape is used.

The setup procedure initiates a tape search and read. The requesting CRT is locked out until the search and read have been completed. When all of the data is core resident, the first line of the technique or procedure is displayed and operator manipulation begins.

The following three sections describe in detail the requirements, capabilities, and limitations of each of the operating modes.

2.3 PROCEDURE PREPARATION

In this mode of operation the operator specifies a master technique, by record number, to be used in preparing a test procedure. The indicated technique is read from magnetic tape and displayed line by line on the operator's CRT terminal. Each line is automatically assigned a line number according to the initial and incremental line numbers supplied during the setup phase. Portions of the remainder of the
line contain special characters, as indicated in Figure 1, that can be replaced via operator input. The cursor is initially positioned at the first replaceable field, allowing the operator to overlay any or all of the characters in that field. The memory protect feature of the terminal is used to prevent overlaying non-replaceable test. This feature does not, however, suppress the overlay of field delimiters such as commas and parentheses. These delimiters are used by the system as field separators as it performs syntax and range checking, therefore overlaying a delimiter will produce ambiguous results. If a field separator is overlayed, the operator should replace it before the transmission key is depressed by moving the cursor to the proper position and depressing the input key for the desired delimiter and then moving the cursor back to its old position. Having completed modification of a given field, the operator is responsible for moving the cursor to the next character to be replaced. The memory protect feature aids in positioning the cursor by automatically advancing it if the operator attempts to place it within protected limits; however, this is only true on non-contiguous fields. When all the desired replacements have been completed, the operator depresses the 'half duplex' and 'send' keys and statement processing begins.

If replace characters are entered, the program performs single syntax and range checking before copying the data to the output buffer. When an error is detected, the entire line is redisplayed along with an error message and the operator is required to repeat the transaction. Those fields, which were entered correctly, are retained as they were entered. Erroneous fields are reset to the special character mark and the cursor is positioned for re-entry of these fields.
### REPLACE CHARACTERS OF MASTER TEST TECHNIQUES

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Alphanumeric field indicator. Reserved words composed of alphanumeric characters and connectors are legal.</td>
</tr>
<tr>
<td>@</td>
<td>Alpha indicator.</td>
</tr>
<tr>
<td>#</td>
<td>Integer indicator. Integer numeric data only is permitted. No decimal point is allowed.</td>
</tr>
<tr>
<td>%</td>
<td>Numeric fields indicator. Decimal numbers between -999999 and +999999 are legal, including fractional data. A decimal point must be included.</td>
</tr>
</tbody>
</table>

**Figure 1**
When a line is correctly entered, it is placed in the output buffer and the next sequential statement is displayed automatically. The previous 15 lines of the program are displayed at the top of the screen for reference by the operator. Each time a new line is successfully entered, the 15 lines on the top of the screen are rolled up one line, and the new line is entered in the 15th line. Processing continues in the manner until the next sequential statement is an end-of-technique line. The program then enters the new test procedure ID and record number in the procedure directory, and writes the procedure to magnetic tape. A request for operation mode is again displayed when the new test has been successfully recorded. See Figure 2 for sample transaction. To delete a line, the control Q is depressed followed by the 'half duplex' and 'send' keys. To abort a procedure, the control R is depressed followed by the 'half duplex' and 'send' keys.

2.4 PROCEDURE COMPLETION

During the procedure preparation phase an operator may choose to skip a replaceable field, leaving the special characters as they existed in the original technique statement. The result is an incomplete test procedure residing on magnetic tape. This procedure can be recalled in the procedure completion operating mode and a series of similar tests produced by filling in only those areas skipped when the procedure was constructed. No syntax and range checking are performed. Delimiter preservation is necessary, as characters are input over the replaceable fields.

The operator must input, upon request, the test ID (test and step number) indicating the procedure to be completed. The procedure is read into core and displayed one line at a time on the appropriate CRT.
Figure 2. SAMPLE TRANSACTIONS

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. 5002</td>
<td>DVMMU [#,@,#] FAIL: GO TO [####] 1</td>
<td>A. This is the original statement as displayed by the system. The cursor is positioned at the first legal replace field. The input replaces the indicator at the present cursor position.</td>
</tr>
<tr>
<td>B. 5002</td>
<td>DVMMU [1,#,#] FAIL: GO TO [####] 1</td>
<td>B. Input move cursor right character to prevent overlaying delimiter. No character is recorded.</td>
</tr>
<tr>
<td>C. 5002</td>
<td>DVMMU [1,#,#] FAIL: GO TO [####] 1</td>
<td>C. Input replace character.</td>
</tr>
<tr>
<td>D. 5002</td>
<td>DVMMU [1,V,#] FAIL: GO TO [####] 9</td>
<td>D. Skip delimiter.</td>
</tr>
<tr>
<td>E. 5002</td>
<td>DVMMU [1,V,#] FAIL: GO TO [####] 9</td>
<td>E. Input replace character.</td>
</tr>
<tr>
<td>F. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>F. Input to skip right parentheses. Because this places the cursor over a protected area, the cursor is automatically advanced.</td>
</tr>
<tr>
<td>G. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>G. Input final overlay character to complete statement.</td>
</tr>
<tr>
<td>H. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>H. Input final overlay character to complete statement.</td>
</tr>
<tr>
<td>I. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>I. Input final overlay character to complete statement.</td>
</tr>
<tr>
<td>J. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>J. Input final overlay character to complete statement.</td>
</tr>
<tr>
<td>K. 5002</td>
<td>DVMMU [1,V,2] FAIL: GO TO [####] 9</td>
<td>K. Input EOM, half duplex, and send characters. Keyboard is locked out and statement is input.</td>
</tr>
</tbody>
</table>

Note: First set terminal to local mode.
The previous 15 lines are displayed on the top of the screen for operator reference. When a line is encountered, that does not contain a special overlay character, it is written to the output buffer immediately and the next sequential line is interrogated. Each line containing incomplete fields is displayed for operator modification. The processing continues until an end-of-procedure is encountered, at which time the procedure is recorded on magnetic tape and a new entry is placed in the procedure directory. A request for operation mode is again displayed when the new test has been successfully recorded. To delete a line the control Q is depressed followed by the 'half duplex' and 'send' keys. To abort a procedure the control R is depressed followed by the 'half duplex' and 'send' keys.

2.5 PROCEDURE MODIFICATION

Procedure modification is an error correction mode of operation. The operator may indicate by procedure I.D. any test program, recorded on magnetic tape, which he wishes to modify. The procedure is read into core line by line for error correction. Syntax and range checking are not performed and the operator may modify any character in the displayed statement. When the 'send' key is depressed, the modified line is sent to the output buffer and the next sequential line is displayed. To delete a line a control Q character is depressed followed by the 'half duplex' and 'send' keys. The next sequential line is then automatically displayed. The processing continues until an end-of-procedure is encountered. The modification procedure is then written to magnetic tape, a new entry is placed in the procedure directory and written to magnetic tape. A request for operation mode is again displayed when the new test has been successfully recorded.
To abort the procedure a control R character is depressed followed by the 'half duplex' and 'send' keys.

2.6 SYNTAX AND RANGE CHECKING

Limited syntax and range checking is performed to minimize operator input errors. A set of syntax and range parameters (see section 2.7) are input during initialization and setup; to be used in keyword syntax and range and increment value checking on the input data.

When the operator depresses the 'send' key in the procedure preparation, the statement is read from the CRT and analyzed before it is sent to the input buffer.

2.6.1 SYNTAX CHECKING

A method of checking test statement syntax, that is simple but very effective, will be used. This method involves checking parameters within a particular test statement, but not checking the relationship between different test statements. Syntax checks performed on each statement will be included the following:

1. Correct field size
2. Type of input: text, integer number, decimal numeric, or alpha character
3. Valid literal for text (via table lookup)
4. Valid range for numerics (either continuous range or discrete increments)

2.6.2 RANGE CHECKING

Two types of numeric range checking will be performed. In the simplest form the number will be checked against a minimum and maximum value. Any number within the range will be accepted, or the number may be checked against a range with discrete points. For example,
a voltage may range from 0.0 to 10.0 volts in steps of .1 volt. This
This test of discrete steps is possible only if the step size is con-
stant. These tests, though simple, will eliminate many clerical er-
rors. The number of digits to the right of the decimal point must be
the same for any given parameter (MIN, MAX, STEP) and the input value.

2.7 PARAMETERS

A list of parameters associated with each statement type is in-
put from cards during initialization. These parameters are used by
the system for syntax and range checking. Figure 3 illustrates the
card format to be used in constructing the parameter deck and Figure 4
depicts the resulting core resident parameter tables.

Four parameter types are permitted as noted in Figure 1. Any
of these special characters may be inserted in a statement to indicate
the corresponding data type which must be input. The number of special
characters dictates the maximum number of overlay characters allowable
for a field. However, it is not necessary for the operator to replace
all of the characters in a field. If a parameter type is indicated in
a statement, but no entry has been made in the parameter table for
that type, a syntax error message results. The four argument types
are: alpha, integer, decimal, and alphanumeric.

2.7.1 Alpha - Character Code @

The alpha character code, appearing in a technique statement,
implies that only alpha data is legal. Any numeric data causes a syn-
tax error to be generated. As indicated in Figure 5, further re-
strictions may be placed on the input by setting Flag 1 in the parame-
ter table. If this flag is false, any letter of the alphabet is a
valid overlay character. However if Flag 1 is true, the input is
compared against the supplied list and if a compare is not found a syntax error is generated.

2.7.2 Integer - Character Code #

Integer data implies a composite of from 1 to 4 integer digits. The maximum integer value cannot exceed 9999 and the smallest allowable overlay is -999. These minimum and maximum values can be further limited by placing minimum and maximum boundaries in the parameter table for the appropriate statement. Other restrictions can be placed on the legal input by setting either flag 2 or 3 (see Figure 5) and listing the required information. Flag 1 allows the user to specify any number of discrete points to be checked to insure valid input. Flag 3 also allows the user to specify discrete point checking. However, he need only supply a constant increment which repetitively added to the previous result, beginning with the minimum value, produces a set of numbers between the specified minimum and maximum which are the desired discrete points. Any valid result causes the appropriate syntax or range error message to be displayed.

2.7.3 Decimal - Character Code %

When the decimal character is placed in a modifiable statement, a range of numbers from -999999. to 9999999., including fractional data is permitted. The only restriction is that the number of characters, including signs and decimal points, does not exceed eight. Minimum and maximum limits are implied and may also be specified in the parameter table for each statement type. Discrete point checking, as described in section 2.5.2, is also available for decimal input (see Figure 5). The same number of digits to the right of the decimal point must be maintained for the minimum, maximum, and increment.
CARD FORMAT FOR PARAMETERS

Card column 1 is used to indicate the type of card being processed:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>statement name</td>
</tr>
<tr>
<td>1</td>
<td>alpha parameter card</td>
</tr>
<tr>
<td>2</td>
<td>integer parameter card</td>
</tr>
<tr>
<td>3</td>
<td>decimal parameter card</td>
</tr>
<tr>
<td>4</td>
<td>alpha numeric parameter card</td>
</tr>
<tr>
<td>*</td>
<td>comment card</td>
</tr>
</tbody>
</table>

Further, if column 1 is 1, 2, 3, or 4 column 2 is used to determine whether the card is a flag card or data card. If column 2 contains a 1, columns 3, 4, 5, and 6 contain the appropriate flag setting. 0 = false, 1 equals true. See Figure 5 for flag definitions. Each statement, which can contain replaceable characters, must have a parameter entry and each entry must contain one entry per replaceable field.

<table>
<thead>
<tr>
<th>Card Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>* (card ignored)</td>
</tr>
<tr>
<td>1</td>
<td>0 (name card)</td>
</tr>
<tr>
<td>3-10</td>
<td>STATEMENT NAME</td>
</tr>
<tr>
<td>1</td>
<td>1,2,3,4</td>
</tr>
<tr>
<td>2</td>
<td>1 (flag card)</td>
</tr>
<tr>
<td>3-6</td>
<td>FLAG SETTINGS 0 or 1 FOR FLAGS 0,1,2,3,4 respectively</td>
</tr>
<tr>
<td>1</td>
<td>1 (alpha parameter)</td>
</tr>
<tr>
<td>2</td>
<td>BLANK (data card)</td>
</tr>
<tr>
<td>3-4</td>
<td>NUMBER OF CHARACTERS (1 parameter per card)</td>
</tr>
<tr>
<td>5-72</td>
<td>PARAMETER DATA</td>
</tr>
<tr>
<td>1</td>
<td>2 (integer parameter)</td>
</tr>
<tr>
<td>2</td>
<td>BLANK</td>
</tr>
<tr>
<td>3-72</td>
<td>DATA (each parameter may be from 1-4 characters and at least 1 space must be separate entries)</td>
</tr>
<tr>
<td>1</td>
<td>3 (decimal parameter)</td>
</tr>
<tr>
<td>2</td>
<td>BLANK</td>
</tr>
<tr>
<td>3-72</td>
<td>DATA (each parameter may be from 1-7 characters and at least 1 space must separate entries)</td>
</tr>
<tr>
<td>1</td>
<td>4 (alphanumeric parameter)</td>
</tr>
<tr>
<td>2</td>
<td>BLANK</td>
</tr>
<tr>
<td>3-4</td>
<td>NUMBER OF CHARACTERS (1 parameter per card)</td>
</tr>
<tr>
<td>5-72</td>
<td>PARAMETER DATA</td>
</tr>
</tbody>
</table>

Figure 3
<table>
<thead>
<tr>
<th>LINK TO NEXT STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT NAME</td>
</tr>
<tr>
<td>LINK TO NEXT PARAMETER</td>
</tr>
<tr>
<td>FLAG TYPE OF PARAMETER</td>
</tr>
<tr>
<td>FIRST PARAMETER</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>LINK TO NEXT PARAMETER</td>
</tr>
<tr>
<td>FLAG TYPE OF PARAMETER</td>
</tr>
<tr>
<td>SECOND PARAMETER</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>LINK TO NEXT STATEMENT</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>END OF STATEMENT</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>
### Alpha Type @

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>TYPE</th>
<th>Flags (Meaning if True)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to Next</td>
<td>Parameter</td>
<td>0</td>
</tr>
<tr>
<td>Bytecount</td>
<td>C1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>2</td>
</tr>
<tr>
<td>C4</td>
<td>C5</td>
<td>3</td>
</tr>
<tr>
<td>C6</td>
<td>Bytecount</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>

### Integer Type #

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>TYPE</th>
<th>Flags (Meaning if True)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to Next</td>
<td>Parameter</td>
<td>0</td>
</tr>
<tr>
<td>CHAR 1</td>
<td>CHAR 2</td>
<td>1</td>
</tr>
<tr>
<td>CHAR 3</td>
<td>CHAR 4</td>
<td>2</td>
</tr>
<tr>
<td>CHAR 1</td>
<td>CHAR 2</td>
<td></td>
</tr>
<tr>
<td>CHAR 3</td>
<td>CHAR 4</td>
<td></td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

### Decimal Type %

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>TYPE</th>
<th>Flags (Meaning if True)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to Next</td>
<td>Parameter</td>
<td>0</td>
</tr>
<tr>
<td>CHAR 1</td>
<td>CHAR 2</td>
<td>1</td>
</tr>
<tr>
<td>CHAR 3</td>
<td>CHAR 4</td>
<td>2</td>
</tr>
<tr>
<td>CHAR 5</td>
<td>CHAR 6</td>
<td></td>
</tr>
<tr>
<td>CHAR 7</td>
<td>CHAR 8</td>
<td></td>
</tr>
<tr>
<td>CHAR 1</td>
<td>CHAR 2</td>
<td></td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

### Alphanumeric $$

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>TYPE</th>
<th>Flags (Meaning if True)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to Next</td>
<td>Parameter</td>
<td>0</td>
</tr>
<tr>
<td>Bytecount</td>
<td>C1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>Bytecount</td>
<td>2</td>
</tr>
<tr>
<td>C1</td>
<td>C2</td>
<td>3</td>
</tr>
<tr>
<td>C3</td>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5
2.7.4 Alphanumeric - Character Code

The alphanumeric code allows the most flexibility in input data and, therefore, has the least amount of syntax checking capabilities. Any number of special characters may be used to indicate this input data type and, if Flag 1 in the parameter table is not set, the only restriction is that the operator does not input more data than special characters. On the other hand, if Flag 1 is set only those items specified in the parameter table are valid and the input and parameter must match exactly.

2.8 STATEMENT FORMAT

Each statement is limited to 65 columns of one card and each card contains only one input statement. The standard 026 keypunch character set is used and translations are made to ASCII codes. Figure 6 illustrates the acceptable character set for both statement and parameter input. Statements are punched in their entirety, including all delimiters and punctuation. Replaceable fields are punched as their corresponding type code according to Figure 1. The number of special characters inserted indicates the maximum size of the field. Statements must begin in card column 1, however, blanks may be inserted indicates the maximum anywhere, except within replaceable fields to increase the legibility of a line.

2.9 CRT DISPLAY FORMAT

Several statement types, system inquires, and system error messages are displayed at the appropriate CRT. The following sections are examples of the CRT display outputs. An up arrow indicates the initial cursor position.
<table>
<thead>
<tr>
<th>Char.</th>
<th>ASCII (Hex)</th>
<th>Holl.</th>
<th>Char.</th>
<th>ASCII (Hex)</th>
<th>Holl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C1</td>
<td>12-1</td>
<td>[</td>
<td>DB</td>
<td>12-5-8</td>
</tr>
<tr>
<td>B</td>
<td>C2</td>
<td>12-2</td>
<td>\</td>
<td>DC</td>
<td>0-6-8</td>
</tr>
<tr>
<td>C</td>
<td>C3</td>
<td>12-3</td>
<td>]</td>
<td>DD</td>
<td>11-5-8</td>
</tr>
<tr>
<td>D</td>
<td>C4</td>
<td>12-4</td>
<td>†</td>
<td>DE</td>
<td>7-8</td>
</tr>
<tr>
<td>E</td>
<td>C5</td>
<td>12-5</td>
<td>‘’</td>
<td>DF</td>
<td>2-8</td>
</tr>
<tr>
<td>F</td>
<td>C6</td>
<td>12-6</td>
<td>BLANK</td>
<td>AO</td>
<td>BLANK</td>
</tr>
<tr>
<td>G</td>
<td>C7</td>
<td>12-7</td>
<td>‘’</td>
<td>A1</td>
<td>11-2-8</td>
</tr>
<tr>
<td>H</td>
<td>C8</td>
<td>12-8</td>
<td>‘’</td>
<td>A2</td>
<td>0-5-8</td>
</tr>
<tr>
<td>I</td>
<td>C9</td>
<td>12-9</td>
<td>‘’</td>
<td>A3</td>
<td>0-7-8</td>
</tr>
<tr>
<td>J</td>
<td>CA</td>
<td>11-1</td>
<td>$</td>
<td>A4</td>
<td>11-3-8</td>
</tr>
<tr>
<td>K</td>
<td>CB</td>
<td>11-2</td>
<td>%</td>
<td>A5</td>
<td>11-7-8</td>
</tr>
<tr>
<td>L</td>
<td>CC</td>
<td>11-3</td>
<td>&amp;</td>
<td>A6</td>
<td>12-7-8</td>
</tr>
<tr>
<td>M</td>
<td>CD</td>
<td>11-4</td>
<td>(</td>
<td>A8</td>
<td>0-4-8</td>
</tr>
<tr>
<td>N</td>
<td>CE</td>
<td>11-5</td>
<td>)</td>
<td>A9</td>
<td>12-4-8</td>
</tr>
<tr>
<td>O</td>
<td>CF</td>
<td>11-6</td>
<td>*</td>
<td>AA</td>
<td>11-4-8</td>
</tr>
<tr>
<td>P</td>
<td>D0</td>
<td>11-7</td>
<td>+</td>
<td>AB</td>
<td>12</td>
</tr>
<tr>
<td>Q</td>
<td>D1</td>
<td>11-8</td>
<td>,</td>
<td>AC</td>
<td>0-3-8</td>
</tr>
<tr>
<td>R</td>
<td>D2</td>
<td>11-9</td>
<td>‘’</td>
<td>AD</td>
<td>11</td>
</tr>
<tr>
<td>S</td>
<td>D3</td>
<td>0-2</td>
<td>.</td>
<td>AE</td>
<td>12-3-8</td>
</tr>
<tr>
<td>T</td>
<td>D4</td>
<td>0-3</td>
<td>/</td>
<td>AF</td>
<td>0-1</td>
</tr>
<tr>
<td>U</td>
<td>D5</td>
<td>0-4</td>
<td>:</td>
<td>BA</td>
<td>5-8</td>
</tr>
<tr>
<td>V</td>
<td>D6</td>
<td>0-5</td>
<td>‘’</td>
<td>BB</td>
<td>11-6-8</td>
</tr>
<tr>
<td>W</td>
<td>D7</td>
<td>0-6</td>
<td>‘’</td>
<td>BC</td>
<td>12-6-8</td>
</tr>
<tr>
<td>X</td>
<td>D8</td>
<td>0-7</td>
<td>‘’</td>
<td>BD</td>
<td>3-8</td>
</tr>
<tr>
<td>Y</td>
<td>D9</td>
<td>0-8</td>
<td>‘’</td>
<td>BE</td>
<td>6-8</td>
</tr>
<tr>
<td>Z</td>
<td>DA</td>
<td>0-9</td>
<td>‘’</td>
<td>BF</td>
<td>12-2-8</td>
</tr>
<tr>
<td>0</td>
<td>B0</td>
<td>0</td>
<td>‘’</td>
<td>A7</td>
<td>4-8</td>
</tr>
<tr>
<td>1</td>
<td>B1</td>
<td>1</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>2</td>
<td>B2</td>
<td>2</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>3</td>
<td>B3</td>
<td>3</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>4</td>
<td>B4</td>
<td>4</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>5</td>
<td>B5</td>
<td>5</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>6</td>
<td>B6</td>
<td>6</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>7</td>
<td>B7</td>
<td>7</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>8</td>
<td>B8</td>
<td>8</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
<tr>
<td>9</td>
<td>B9</td>
<td>9</td>
<td>‘’</td>
<td>80</td>
<td>0-2-8</td>
</tr>
</tbody>
</table>

Figure 6
2.9.1 Sample Statement

5000 CTRMF ([#,@,#]) FAIL: GO TO [####]

2.9.2 Sample Error Messages

INCORRECT INPUT TYPE
RANGE ERROR

2.9.3 Sample Screen

0100 TEST 2100
0110 STEP 0001
0120 REM CHECK OF 400 HZ SOURCE
0250 DISPLAY ("DVMA", V, "VOLTS")
0260 COMPARE ([@, ., ., .]) FAIL: [####]

(This line reserved for error messages)

2.10 MAGNETIC TAPE FORMAT

As each master test technique is read from cards, it is recorded on magnetic tape as a single record for test preparation recall. Each test procedure, when completed, is also recorded on magnetic tape and the directory of procedures is written to conclude the tape recording. Figure 7 is a block example of a completed magnetic tape.
**MAGNETIC TAPE FORMAT**

<table>
<thead>
<tr>
<th>Master Test Technique 1</th>
<th>I</th>
<th>R</th>
<th>G</th>
<th>Master Test Technique 2</th>
<th>I</th>
<th>R</th>
<th>G</th>
<th>Master Test Technique X</th>
<th>I</th>
<th>E</th>
<th>F</th>
<th>I</th>
<th>R</th>
<th>G</th>
<th>Test Procedure 1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Test Procedure 2</th>
<th>I</th>
<th>R</th>
<th>G</th>
<th>Test Procedure 3</th>
<th>I</th>
<th>R</th>
<th>G</th>
<th>Test Procedure Y</th>
<th>I</th>
<th>E</th>
<th>O</th>
<th>R</th>
<th>G</th>
<th>P</th>
<th>G</th>
<th>Test Procedure Directory</th>
</tr>
</thead>
</table>

1. **MASTER TEST TECHNIQUE** - Incomplete test procedure as input during initialization and setup from cards.

2. **TEST PROCEDURE** - Modified technique or procedure as produced by operator.

3. **TEST PROCEDURE DIRECTORY** - This directory contains an I.D. for each test procedure written to tape.

The table is ordered by record number of the corresponding procedure.
APPENDIX 3. TYPICAL EXAMPLES OF SECOND PROGRAMMING AID
TEST [# # # #]

STEP [# #]

RESET

DVMSU ([#, ####, #])

ACS ([% % %, % % %, # #])

SIGMAT (1, [@ @ @ @ @ @,"DVM-HI")

SIGMAT (1, [@ @ @ @ @ @,"DVM-LO")

EXECUTE

DVMMU ([#, @, #])

COMPAR ([ $ $ $ $, % % % %, % % %,]"DVM","VOLTS") FAIL: GO TO [# # # #]

COMPLETE

PROCEDURE PREPARATION MODE
TEST 1001

STEP 02

RESET

DVMSU (1, 10, 0)

ACS (10.0, 15.5, 20)

SIGMAT (1, [0 0 0 0 0 0] "DVM-HI")

SIGMAT (1, [0 0 0 0 0 0] "DVM-LO")

EXECUTE

DVMMU (1, V, 1)

COMPAR (V, [VXX.X, XXX.X] "DVM", "VOLTS") FAIL: GO TO 1010

COMPLETE

PROCEDURE COMPLETION MODE
[TEST 1001]

[STEP 02]

[RESET]

[DVMSU (1, 10, 0)]

[ACS (10.0, 15.5, 20)]

[SIGMAT (1, "6J1-10", "DVM-HI")]

[SIGMAT (1, "6J1-35", "DVM-LO")]

[EXECUTE]

[DVMMU (1, V, 1)]

[COMPAR (V, 9.5, 8.3, "DVM", "VOLTS") FAIL: GO TO 1010]

[COMPLETE]

PROCEDURE MODIFICATION MODE
APPENDIX 4. PTS-100 INTELLIGENT TERMINAL DESCRIPTIVE BROCHURE
PTS-100 General Description

The PTS-100 Programmable Terminal System provides the user with flexibility unmatched by other systems. Custom configurations can be designed to suit applicable needs without compromise. As user requirements grow, the PTS-100 System can be expanded by adding modules of memory, terminals and peripherals.

All models of the PTS-100 utilize the same software systems and are made up of a basic set of modular building blocks, such as the Processor, the Memory System, and the I/O Adapters. The models differ in expandability and peripheral handling capability.

Standard Software

The PTS-100 operating software is sufficiently general to accommodate a variety of applications in the telecommunications, teleprocessing, and real-time or batch processing environments on a broad range of hardware configurations. It is made up of a library of programs that can be tailored to the user's application and installation needs. The programs that make up the PTS-100 operating software are classified as monitor, language translator, and utility programs.

A monitor control system, resident assemblers for the PTS-100, and cross assemblers for the Raytheon 700 Series and IBM 360/370 computers are available. The monitor control system provides a simplified user interface with I/O devices. Through a system generator a specific monitor control system can be generated for the user's particular configuration.

A wide range of utilities and diagnostics are also provided.
APPLICATION PACKAGES

2260/2848 Emulator

The 2260/2848 software package makes the PTS-100 System fully compatible with the IBM System 360/370 computer with considerable economies and meaningful improvements in system performance. Many of the 2260 options, such as nondestructive cursor, tab erase EOS, erase EOL, are standard in the PTS-100's 2260/2848 Emulator package. A 3270 software package is in development.

The PTS-100 may be connected to the System 360/370 directly or remotely. Data transmission in the remote mode is at line speeds up to 4800 baud; in the direct or local mode speeds of 80,000 bytes per second can be achieved.

Airline Reservation System

Raytheon Data Systems has developed a software package that interfaces the PTS-100 System with 2260/2848 and PARS/IPARS airline reservation programs utilized in IBM System 360/370 computers. Other computers in the airline environment have been interfaced, including Univac. Among the many airlines using RDS terminals are KLM, TWA, Air Canada, Air France, Quantas, SAA, Iberia, Swissair, Finnair, and Alitalia.

STANDARD PTS 100 SYSTEM MODELS

The PTS-100 Programmable Terminal System is available in four models. Each model consists of the same basic building block modules which differ in their expandability and peripheral handling abilities.

Model 1010A

Designed primarily for single display environments requiring a limited number of peripherals, Model 1010A is also applicable
operating either off line or in a data communications environment. The basic system includes the processor with 8K byte memory expandable to 16K, a high speed I/O bus (for customer engineer (CE) console), a multiplexer channel with four subchannels, modem adapter, integral display adapter/monitor controller generating the standard 64 symbol set, and one display terminal with the standard 67 key keyboard.

A choice of display capacity and format is offered: Screen sizes of 480, 960, and 1920 character capacities are available with line lengths of 40, 64 or 80 characters. One 1920 character display or two 480/960 character displays can be accommodated by this model.

Options - memory expansion to 16K; a second display terminal added if screen capacity is limited to 480 or 960 characters; a 96 symbol set for display line length of either 40 or 80 characters; an 82 key keyboard; and a display indicator panel.

Model 1015A - Expanded Stand-Alone Display System

The basic system includes as standard the processor, feature board 8K of memory expandable to 16K, high speed I/O bus (for CE console and channel interface controller (CIC)), a multiplexer channel with eight subchannels, a modem adapter, display adapter/monitor controller generating the standard 67 key keyboard. Other options are features that are the same as in the model 1010A, with remote program load, watchdog timer, and channel interface controller also available.

Model 1015B - Medium Cluster Display System

The basic system includes as standard the processor, the feature board, 8K of memory expandable to 16K, high speed I/O bus (for CE console and CIC), a multiplexer channel with eight subchannels, a modem adapter, an eight channel display adapter, a monitor controller
generating the standard 64 symbol set, and one display terminal with the standard 67 key keyboard.

The monitor controller can be installed either in the processor or in an optional remote concentration. Screen sizes of the CRT may be chosen from among the 480, 960, or 1920 character capacity in line lengths of 40, 64, or 80 characters.

Options - watchdog timer; remote program load; memory expansion to 16K; up to four monitor controllers generating the standard 64 symbol set. Also optional: the 96 symbol set (upper and lower case); the 82 key keyboard; remote concentrators; and a channel interface controller.

Model 1020B - Large Cluster Display

The basic system includes as standard the processor, the feature board, 16K bytes of memory expandable to 64K, a high speed I/O bus (for CE console, CIC, and disc controllers), a multiplexer channel with eight subchannels, a modem adapter, an eight channel display adapter, a monitor controller generating the standard 64 symbol set, and one display terminal with the standard 67 key keyboard.

Options - watchdog timer; remote program load; memory expansion to 64K; up to 24 monitor controllers. In addition, a second multiplexer channel may be added to increase the number of subchannels to 15. Also optional: 96 symbol set (upper and lower case); 82 key keyboard; remote concentrators; CIC; disc controllers.

KEY ELEMENTS OF THE PTS 100 SYSTEM

Central Processing Unit and Memory System

The processor is a 16 bit general purpose computer capable of executing 29 single or double word instructions. Memory addressing is by 16 bit word or 8 bit byte, and includes six addressing schemes:
Absolute, Dynamic Page and Indexed - each with or without single-level indirect mode.

An eight level hardware priority interrupt system, interval timer and high speed I/O bus also are standard.

The memory system uses random access MOS semiconductor devices and is expandable from 8K to 64K bytes in modules of 4K or 8K. It is organized in 16 bit words and has a cycle time of 1 µs.

Each 8K memory module operates independently. Memories are dual ported to permit simultaneous but noninterfering access to data by the processor and by the display adapter for automatic display refresh at the rate of 60 Hz. When both request operation at the same time the refresh is given priority. This results in a maximum 20% increase in processor access time with a maximum single cycle delay of 1.28 µs.

Feature Board

The feature board contains various fixed and switch-selectable options and features, such as watchdog timer, interval timer, initial program load ROM, optional remote program load ROM, alternate program load devices address, switch, and eight miniatures position-switch chips for configuration control.

Display Adapter

Attached directly to the memory bus, the Display Adapter provides the timing and synchronization to refresh the CRT. It also accumulates keyboard data from the terminals which may be located as far as 2000 feet from the processor. A single Display Adapter can drive eight output channels. Each channel is capable of handling four CRT Display Terminals.
Monitor Controller

The Monitor Controller, which may be located within the processor in the remote concentrator located up to 2000 feet from its associated Display Adapter, accepts data from the Display Adapter, converts it through a dot matrix generator to video form and separates it for each of the data displays. The standard Monitor Controller can handle a 64 character upper case symbol set and one 1920, two 960, or four 480 character Displays and Keyboards. Available with the 40 and 80 character display line option is a 96 symbol set for upper and lower case character generation. Displays may be located up to 100 feet from the Monitor Controller.

Integral Display Adapter/Monitor Controller

This provides a Display Adapter and Monitor Controller function for systems requiring a maximum of one 1920 or two 480/960 character Displays.

Multiplexer Channel

The Multiplexer Channel is attached to the I/O bus, and though appropriate device adapters on its subchannels interface low speed peripherals operating at rates up to 9600 bits per second. Block transfer of data between memory and peripheral device (automatic code transaction and character searching are handled) without program intervention. Up to two Multiplexer Channels, each having eight subchannels can be connected to a fully expanded PTS 100 System. One multiplexer subchannel is dedicated for connection of all Display Keyboards - whether or not Display Terminals are used. The multiplexer of the basic model (Model 1010A) system is limited to four subchannels (one is dedicated to the keyboard). All other models have the full
complement of eight subchannels.

High Speed I/O Bus

Up to eight high speed controllers may be attached to the I/O Bus to control peripheral devices operating faster than 9600 bps, such as disc memories, magnetic tape transports, and host processor channel interfaces (IBM System 360/370 multiplexer selector channel interface). Data transfer are over a bidirectional 16 bit bus at a maximum rate of 1,000,000 bytes per second.

Remote Concentrator

The Remote Concentrator which houses monitor controllers provides the capability of operating Display Terminals and other low speed peripherals located up to 2000 feet from the processor. A maximum of 16 Display Terminals with 480 character capacity (16 with 960 character capacity and 8 with 1920 character capacity) can be operated through a single remote concentrator.

PERIPHERALS

The following devices are among the announced peripherals presently available for the PTS-100 system.

Model 3401 Serial Printer

The Model 3401 Serial Printer provides as standard 30 characters/second receive only print in a master, plus 5 copies, friction paper feed, 80 character print line and up to 94 printable USASCII characters. An optional pin feed platen feature provides capability of using sprocket fed paper forms. Model 3401 serial printers attach to the Processing System via multiplexer subchannels. Each model 3401 serial printer includes a simplex adapter for multiplexer subchannel interface.
Model 3411 Serial Printer

The Model 3411 serial printer provides as standard 120 characters second receive only print in a master, plus 5 copies, 120 character print line and up to 94 printable USASCII characters. Model 3411 serial printers attach to the Processing System via multiplexer subchannels. Each model 3411 serial printer includes a simplex adapter for multiplexer subchannel interface.

Model 3201, 3202 Card Reader

The Model 3201 card reader provides 300 cards per minute reading of standard 80 column punched cards with either Hollerith or binary coding. A 600 card stacker and hopper are provided. The Model 3202 provides a similar function but with 1000 card per minute reading rate. Attached to a processing system multiplexer subchannel, the card reader function as a program load device for the PTS-100. A simplex adapter between the reader and multiplexer is provided with the device.

Model 3601 Magnetic Tape Cassette

Model 3601 magnetic tape cassette unit provides for recording and reading of variable length records on 300 foot magnetic tape cassettes. Fifteen hundred records of 80 character length (120K characters) and 320 records of 960 character length (307K characters) are representative of the storage capacities of an individual cassette record at 800 bits per inch.

The Model 3601 is provided with a read after write head to check validity of the record just written. This is accomplished by verification of the cyclic redundancy check character (CRC). This CRC character is also used to verify records when the cassette is used in a read mode. Five operating modes exist for the cassette: read
forward, write forward, backspace (1 record), erase one record, and
rewind. Functional use of the cassette is for program storage,
locally initiated data storage, and retrieval of data or prestored
display screen formats.

The cassette adapter interfacing the device and a multiplexer
subchannel controls from one to four cassettes, with one cassette op-
erational at a time. The adapter is included as a feature of the
Model 3601 magnetic tape unit.

Teletype Models

Model 3402 (RO), 3403(KSR), and 3404(ASR) printer versions of the
Teletype 33 provides 72 characters per line format, and 63 printable
characters - Functional uses include print only, interactive console,
and program load/dump via paper tape. Teletype terminals are attached
to the processing system via multiplexer subchannels with an adapter
supplied with the device. The RO version operates simplex; the KSR
and ASR versions operate full duplex.

Disc Storage Units

Disc units of 2.5(3711) and 5(3712) megabytes are available for
use with the 1020B processor. Large capacity discs are available on
special order and will be standard product line items in the near
future.