SDC FIELD NOTE A WORKING PAPER | FN - LO - 197

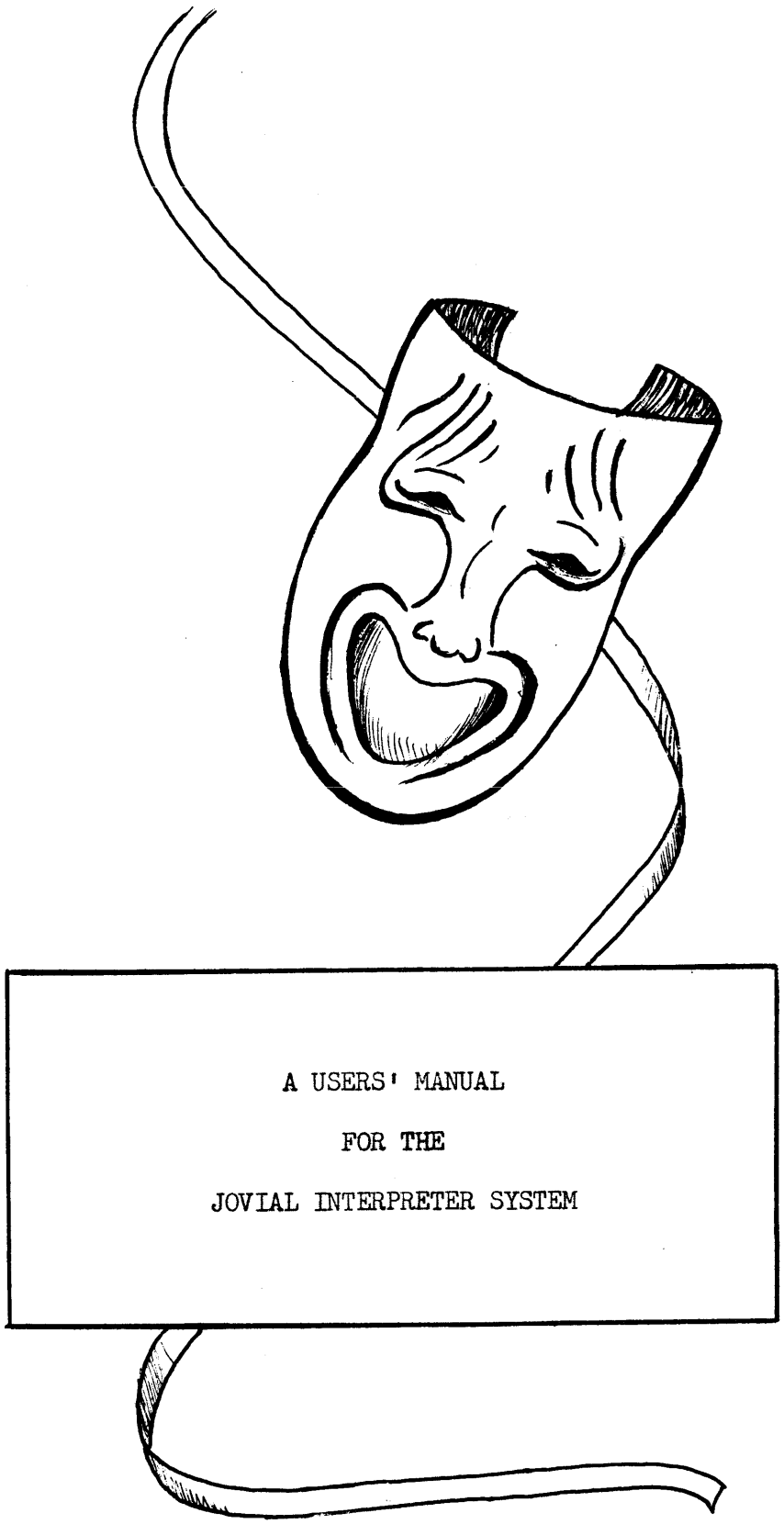
AUTHOR
K. E. Petersen

APPROVED
M. Blauer

(Produced under System Development Corporation subcontract No. 202 issued by International Electric Corporation in performance of contract AF-30(635)-11583)

1 February 1960

SYSTEM DEVELOPMENT CORPORATION • 2500 COLORADO AVENUE • SANTA MONICA, CALIFORNIA



A USERS' MANUAL

FOR THE

JOVIAL INTERPRETER SYSTEM

TABLE OF CONTENTS

PURPOSE OF THE USERS' MANUAL

This manual has been written as an attempt to collect in one document the various data required for optimum usage of the JOVIAL Interpreter System. It is felt that the programmers who will be using the system will find it more convenient to use one document rather than refer to the individual program specifications. This manual, of necessity, does not include the detail shown in the program specification, but does include all information necessary for use of the system.

The manual is divided into four major sections:

1. INTRODUCTION TO THE JOVIAL INTERPRETER SYSTEM
2. USE OF SYSTEM SUPPORT FUNCTIONS
   - THE ASSEMBLE MASTER COMPOOL FUNCTION
   - THE SYSTEM TAPE LOADING FUNCTION
3. USE OF THE JOVIAL PROGRAM INTERPRETATION FUNCTION
4. SAMPLE PROBLEMS

INTRODUCTION TO THE JOVIAL INTERPRETER SYSTEM

A PRE-TRANSLATOR CHECK OF PROGRAMS WRITTEN IN A HIGHER LEVEL LANGUAGE

The United States Air Force has contracted for an automated control system that will enable the Strategic Air Command to strike back more quickly and surely against enemy aggression anywhere in the world. This will be accomplished by a complex Data Processing System that will ease, but not relieve, the burden of command. A key part of this challenge has been assigned to the newly created SACCS Division of the System Development Corporation. The abbreviation SACCS stands for Strategic Air Command Control System.

Examination of available programming languages revealed that the use of a higher level language for the production of large-scale systems would be beneficial in the following respects: the burden of learning intricate machine code is lifted from the programmer. He is then free to concentrate on program logic and the development of his program. Translated code will, thereby, be more consistent and free of errors. The machine will be used more efficiently. Program code will be shorter and more easily read, thus permitting more efficient debugging and accurate modifications to the system sub-programs. The International Algebraic Language, "ALGOL," was used as a basis for the development of this higher level language. Certain additions and modifications were necessary to tailor this language to the needs of large system programming.*

*A large-scale system, in our frame of reference, means a Data Processing System consisting of fifty to one hundred sub-programs under some executive control. The total number of computer instructions may be in the 100,000 to 200,000 range, operating on perhaps a million or more words of stored data.

One important modification of the International Algebraic Language required for the programming of large computer systems arises from the need for a symbolic method of referring to data common to the many sub-programs in the system. To meet this problem, the essential characteristics of all common data are assembled into what is known as the Communication Pool. The ability to symbolically refer to this common data has been included in this language. The computer instructions necessary to locate and use the data are supplied by some type of translator or assembly program which has access to the Communication Pool. Other modifications or additions include internal data definitions, fixed point arithmetic capabilities, additional switching techniques, and modifications of the indexing method. This modified International Algebraic Language, that has been developed for our use in the programming of the Strategic Air Command Control System, has been designated as JOVIAL.*

The JOVIAL Language was made available much in advance of a program translator and of the Military Computer, for which machine language is ultimately needed. This created the following problem -- there would be a period of time during which programs would be coded in the JOVIAL Language and there would be no means of checking out these programs. Therefore, some method of program checking during this interim period was required.

*Jules I. Schwartz, "Programming Languages For Language Systems," paper submitted to the Western Joint Computer Conference, May 1960.

Two approaches were considered. The first involved direct simulation of the Military Computer. This method would necessitate the production of a translator that would produce programs in a binary machine language and a program that would directly simulate a machine operation of the Military Computer on some currently available computer. In addition to these two basic programs, additional support-type programs would be required to create a "system" that would coordinate and systematize the actions of the above programs in code checking programs written in the JOVIAL Language.

The second approach was a system that would operate the program in an interpretive manner. This interpretive function would be performed on the Intermediate Language statements resulting from the first pass of a translator program. The interpretive program executes these statements much in the same manner as a computer executes an instruction. However, there is no need to translate the JOVIAL program to direct machine language in the approach.

Both approaches require the writing of a program that would, as a first step, translate the higher level language coding into a series of Intermediate Language statements. A study revealed that the time required to complete the Translator-Simulator System would be considerably greater than that required to complete the Interpretive System. Therefore, it was decided that since system production time

was critical, the initial program checkout system would be developed using the interpretive approach. The translator-simulator approach was not discarded, but will be adopted as a second phase operation. Use of the Interpreter System allows program checking to begin approximately one year in advance of the receipt of the computer for which the programs are intended.

To implement the above decision, the JOVIAL Interpreter System was developed. This system, consisting of a series of interdependent programs coded to operate on the IBM 709, operates the JOVIAL Object Program interpretively in a simulated environment, producing as its output the data generated by the Object Program. The system is fully contained on a self-loading system master tape.

The operation of the JOVIAL Interpreter System is monitored at all times by an executive-type program known as the Test Control Program. The primary purpose of this program is to control the sequence of sub-program operation in accordance with the requirements of the system function being operated. The system has three operating functions:

1. Assembly of "Master" Communication Pool.
2. Assembly of the system program onto a system master tape.
3. Interpretation of a program written in the JOVIAL Language.

The major function of the system is the aforementioned "interpretation of programs written in the JOVIAL Language". Two support functions are also included. These functions were necessary in order that the resulting system be self-contained, i.e., no programs other than those included in the system are required for the accomplishment of any of its functions. (See Figure 1)

The first support function is that of creating the Master Communication Pool. Because of the importance of the Communication Pool concept, a description of its content and usage is pertinent at this time. A unique symbolic designator is assigned each piece of data used in the system. These symbolic designators together with the characteristics of the data they label are assembled into what is known as a Communication Tag Pool (COMPOOL). The data in the COMPOOL is arranged alphabetically according to the symbolic designators. Several levels of data arrangement are described by the COMPOOL. The lowest level consists of data referred to as items. Where several items are related, they are located together within a table. These related items form an entry. This entry may describe, for example, an aircraft. If more than one aircraft is to be described, then the table would contain additional entries (sets of related items) to describe additional aircraft.

Table descriptions include the following: symbol designators, location of this table in core storage, type of table, number of items in an entry, and the number of entries in the table. Item descriptions include: symbol designator, name of the table the item resides in, bit location of item within word and location of word within entry, and the type of item. Some common types are -- floating point, fixed integer, mixed fraction, and six-bit hollerith-coded items. It can be seen then that through the use of the COMPOOL the burden of defining and locating data in core storage is lifted from the programmer.

A description of each item of data is put on a punched card. These cards are prestored and the tape is read by the Assemble Master COMPOOL Program. Legality checks are performed to insure that the data characteristics are consistent within themselves. The characteristics are arranged alphabetically according to the symbolic designator assigned. If no errors are found, the assembled COMPOOL is stored on a binary tape for later inclusion on the system master tape. Discovery of an error will cause an appropriate comment to be stored on the delayed output tape. In addition to this a table of errors is kept. The partially assembled COMPOOL, along with this error correction table, is stored on a binary tape for reassemble purposes. The reassemble mode of the Assemble Master COMPOOL Program operates in the same manner as the assemble mode. However, the inputs in this case are error correction cards and the previously stored, partially completed COMPOOL. The reassemble procedure is repeated if all previous errors were not corrected. If all errors were

corrected properly, then the COMPOOL is stored on tape. An optional output of this program is an alphabetical list of the symbolic designators contained in the COMPOOL. This list is used for content control.

The second support function is provided by a system Tape Loading Program. Operation of this program is initiated upon the recognition of a control card by the Test Control Program, which then reads the Tape Loading Program into its specific core location and transfers control to it. The Tape Loading Program reads a series of tape control cards from the card reader. These cards are checked for content or logic errors and, if found to be correct, are used to establish a sequential table of tape control data to be used for the production of "new" system master tapes. The program uses the table of control data to selectively duplicate data from various input sources. The order of output files on the system master tape is controlled by the sequence of the tape control cards. The input sources vary and may include an "old" system master tape, new or revised system programs from tape or cards, or non-program tabular data such as a new "Master Communication Pool." Upon the successful completion of tape production, the program prints a table of contents listing for use in control and administration of the new tapes.

The major function of the JOVIAL Interpreter System is that of interpreting the operation of a program written in the JOVIAL Language. The primary input to this function is punched cards containing the JOVIAL-coded Object Program and simulation data. All input data cards are prestored off-line onto magnetic tape to speed internal operation.

This permits the stacking of multiple input decks on a single magnetic tape. The sequence of job operation is controlled by a series of test control cards in the card reader. A job is initiated by reading a test control card, this card defines the operation and contains coded option indicators that control system operation. After reading the test control card, the program positions the input tape to the data file associated with this test and transfers control to the first operating program. The Test Control Program monitors the test function throughout and is responsible for the establishment of the environment prior to the operation of each sub-program.

The first system sub-program to operate is section one of the Interpreter. The JOVIAL input cards are read from the prestored tape, and after being checked for logical inconsistencies, are subjected to an algorithm which produces for each JOVIAL statement a series of Intermediate Language statements that describe the operations necessary to perform the function(s) contained in the original JOVIAL statement. Another function is the development of a Variable Definition Table entry for each item of variable data referenced by the Object Program. This table of definitions is derived from two sources -- the Master COMPOOL associated with this test, or by definition within the JOVIAL Object Program. Once the total operating environment has been defined, it is possible to efficiently allocate core storage to the various tables and items. The Variable Definition Table is updated to reflect these storage addresses. The option exists on the test control card

of choosing one of the several Master COMPOOLS contained on the system master tape as the data definition source for this particular test, thus allowing testing of the JOVIAL-coded programs with the various versions of system COMPOOLS. Each entry in the Intermediate Language table describes an operation. Some typical operations are: data transfers, arithmetic computation, transfers to other Intermediate Language table entries, and logical decision-making operations. Data transfer statements refer to storage of final results. These locations may be temporary or defined by the COMPOOL. Arithmetic computations usually involve two variables, an arithmetic operator and a temporary location for storing the results. Logical decision entries involve comparing two variables and true and false exit addresses. The variable addresses contained in the Intermediate Language table entries refer to the entry in the variable description table which defines this piece of data. Contained in this description is the core location into which the final result will be stored. Production of the Intermediate Language is discontinued when erroenous or inconsistent statements are discovered. However, the balance of the JOVIAL input cards are audited in order to maximize error detection in a given computer run. These erroenous statements are written on the output tape and the test is discontinued upon the completion of this sub-program's operation.

An option available on the test control card allows the introduction of revisions to the JOVIAL program deck through the card reader. This allows prestored input tapes to be updated without having to prestore the entire input array.

The Assemble Baby COMPOOL Program operates next. Its purpose is the contruction of a Baby COMPOOL. This COMPOOL describes only that data referenced by the JOVIAL Object Program under test. The Baby COMPOOL is created in an effort to eliminate manipulation of data not required for the testing of this particular program, and to provide a control medium for use by the Data Simulation and Data Processing Programs. The primary inputs to this program are: the Variable Definition table, which obtains the data characteristics from either the Master COMPOOL or from program definition; the Intermediate Language table, which provides an indication as to whether the data is used and/or set by the program, this COMPOOL is arranged alphabetically according to symbolic designators in order to minimize access time. The COMPOOL is stored on a buffer tape for future use by the Data Processing Program and also retained in core storage for immediate use by the Data Simulation Program which operates next. In summary, it should be noted that this Baby COMPOOL privides the system sub-programs with the data characteristics, and an indication of their source, and an indication as to their usage by the Object Program.

The next program to operate in the test sequence is the Data Simulation Program. Its major function is to allow all system data to be preset to known values, as described in the Baby Compool; in short it provides the controlled environment necessary for program testing.

This is accomplished in the following manner. The programmer, after deciding which functions of the program are to be tested and the data utilized by these functions, determines the initial values desired. This information is entered into the system from a prestored input tape containing images of punched cards that contain the symbolic name of the data, its relative location within a table of data, and the value to be assigned. The Data Simulation Program reads the input data and after legality checking, by reference to the COMPOOL, converts them from a convenient form to properly scaled binary format, or, in the case of status-type data, enters the status symbology. An option exists that allows additions, deletions, or changes to be made to the input data on the prestored tape. This option is indicated on the test control card. The card reader is utilized as an overriding input medium allowing the revisions to be entered. The output is stored in appropriate relative core locations that will become the environment image when the JOVIAL Object Program is operated. This environment image is buffered onto tape.

The second function of the Data Simulation Program allows entry of expected results, i.e., the final value that the JOVIAL Object Program should generate for selected system data. This information is entered into the system in the same manner as the initial value except that the cards are distinguishable as expected results. The program processes this data and again creates an image that is buffered onto tape for comparison purposes at a later time.

The Data Simulation Program uses the Baby COMPOOL extensively during its operation. The symbolic name from an input card is extracted and searched for in the COMPOOL. A match provides all necessary information required for conversion and binary scaling of values, eliminating the need for programmer calculation and assignment of binary configurations. Another input to the program is tables of constants internal to the JOVIAL program. These are entered from a buffered tape as mentioned in the discussion of the first pass of the Interpreter Program. Discovery of an error during the operation of the Data Simulation Program will cause an appropriate comment to be logged on the delayed output tape. Discovery of errors at this time will not cause the test to be discontinued.

The second section of the Interpreter now operates. The primary function is to execute each Intermediate Language statement in a manner analogous to the execution of machine instructions by a three-address computer that has no accumulator. This implies that arithmetic computations may involve at most two variables and an address for storing the temporary result. Variable addresses refer to the appropriate entry in the Variable Definition table which contains the address and description of this data. These data descriptions are used to properly handle the computation. The temporary storage address is necessary to replace the function of the missing accumulator. A special situation exists for decision-type entries. In addition to the variable addresses, a true and a flase exit address is shown. Other types of entries include the following: data transfer entries, statements which transfer the Interpreter to another

series of entries; status-switch statements, which transfer the operation to the appropriate entry based on the current status of a status-type item; and numerical-switch entries, which transfer control to the appropriate entry based on the current value of the specified transcript, i.e., index register setting.

There are two secondary features to this program which aid in the debugging function of the system. The first one involves machine logic checks. All types of arithmetic computations which result in errors that would be discovered by a machine, such as overflow or underflow of conditions, and zero divide checks, are noted, and an appropriate comment is written on the delayed output tape.

The other feature is that of assisting the Data Processing Program in the performance of the interpretive trace function. A buffer tape containing dynamic snapshots of certain interpretive operations is produced for later use by the Data Processing Program.

The Data Processing Program is the last program to operate in the test sequence. Its purpose is to provide the programmer with useable output information relating to the interpretation of the JOVIAL Object Program. All system data is analyzed, and its characteristics are written out using the information from the Baby Communication Pool. Under the description of each piece of system data appear the numerical or symbolic values relating to this data. The final value that the JOVIAL Object Program generates is always shown. Initial and expected values for the

data are shown if they were previously introduced by the Data Simulation Program. An internal comparison is made of the final and expected values. When a difference between these values is discovered, this fact is noted in a table of differences for later use. The Program processes all data defined by the Baby Communication Pool and provides, as previously noted, a complete legible picture of the JOVIAL program output.

A significant output of the Data Processing Program is known as the Intermediate Language table printout. This printout compares roughly with a compiler output and is the programmer's only link with the internal computer operation of his program; remembering here that no direct translation to machine language is made. The Intermediate Language table printout is a sequential English-language translation of the individual Intermediate Language statements. They are arranged so that JOVIAL statements may be easily reconstructed.

One data processing option may be specified when initiating a test. This option concerns interpretive tracing. Input to this function is the buffer tape, containing dynamic "snapshots" of interpretive operations, created during the running of section two of the Interpreter. Two modes of operation are available; the first is called "automatic trace," and need not be designated on the test control card. In this mode, the table of comparative differences between the final and expected values is examined to determine the data location in error. A search is made of the dynamic snapshots to determine which statement(s) were responsible for incorrect program results.

The second mode of trace operation is a "full" trace. This mode specification in the test control card will cause the Data Processing Program to produce a complete sequential dynamic record of all transfer and data storage-type Intermediate Language statements. These statements are decoded from the buffer tape and translated into a readable format providing the programmer with a detailed picture of the JOVIAL program's operation.

The trace output is nothing more than a translation of an Intermediate Language statement of the data storage or transfer category. As such, they indicate any change in the output data area or internal transferring of control. These outputs are related to the Intermediate Language Table, and their effective use provides an extremely efficient debugging "tool."

All output data is stacked on a delayed output tape that is later processed as an off-line operation.

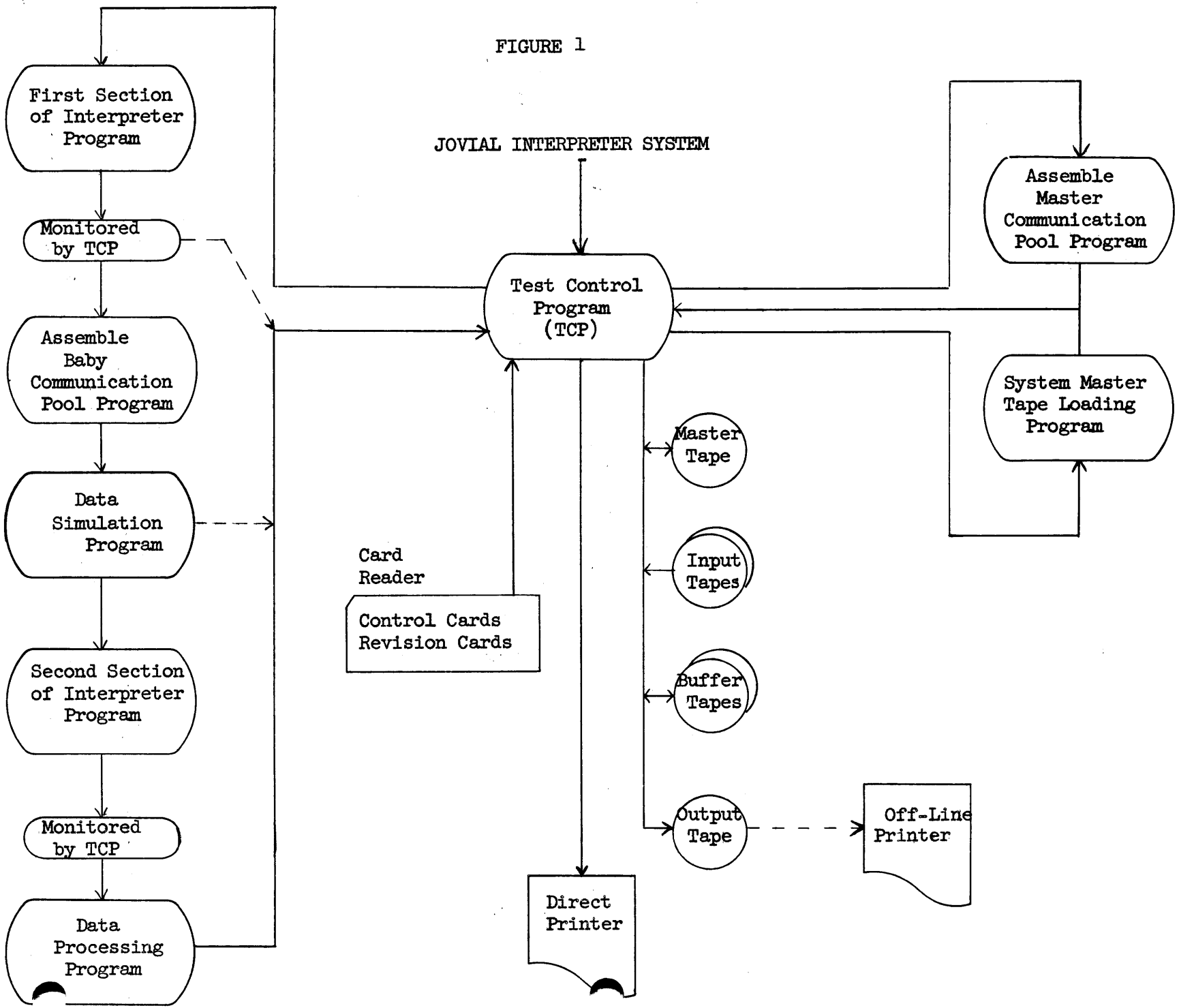
In conclusion it may be said that the JOVIAL Interpreter System offers a partial solution to the problems arising in the checkout of computer programs written in a higher level language. These problems are greatly increased due to the fact that the language translator and intended computer are not available.

The following significant benefits are derived through use of the system.

1. Programs written for an as yet undelivered computer may be checked for logical consistency.
2. The COMPOOL concept may be employed to organize data, thus relieving the programmer from that arduous task.
3. The clerical functions of program checkout are reduced considerably; the programmer is required to do little more than code his program and insert environment data.
4. The debugging aids are felt to be more than sufficient and are executed automatically.
5. All output is in a readable form, thus eliminating cumbersome translation techniques generally required to understand the output.

As noted above, the solution is only partial in that translation to computer language and code check on the intended computer will still be necessary. But it is believed that the time required for such checkout will have been greatly reduced. The use of this system will undoubtedly prove to be a useful tool for the preliminary checkout of computer programs, even after the language translator and the computer for which they were written is available.

FIGURE 1


JOVIAL INTERPRETER SYSTEM
First Section of Interpreter Program
Monitored by TCP
Assemble Baby Communication Pool Program
Data Simulation Program
Second Section of Interpreter Program
Monitored by TCP
Data Processing Program
Test Control Program (TCP)
Assemble Master Communication Pool Program
System Master Tape Loading Program
Master Tape
Input Tapes
Buffer Tapes
Output Tape
Off-Line Printer
Card Reader
Control Cards
Revision Cards
Direct Printer

USE OF THE SYSTEM SUPPORT FUNCTIONS

The two system support functions - creating a master compool and loading a system master tape - will be performed prior to initial system testing, and also whenever an addition to, or updating of, the system master tape requires it. The Assemble Master Compool (JAMCZ) and System Tape Loading (JMSTZ) Programs were written to perform these two functions.

THE ASSEMBLE MASTER COMPOOL FUNCTION

JAMCZ may operate in either of two modes, Assemble or Reassemble:

1. The Assemble Mode will produce one to three compools. If correctable errors are found during the assembly process, an error table will be included with the compool in error. All relevent information regarding the assembly will be printed out. The completed compools, with or without errors, will be written onto a buffer tape. If no errors are found, this fact is communicated to the Test Control Program. The System Tape Loading Program may then operate and would transfer the completed compools to the new system master tape. If errors were found, the Test Control Program will not [illegible]ate JMSTZ, and the buffer tape may be saved for input to the Reassemble Mode.

2. The Reassemble Mode will process the tape produced by a previous operation of JAMCZ, and which contains one to three compools. At least one of these compools contains errors to be corrected. Each compool from the input tape will be examined for a table of errors. If none is found, the compool is transferred to a new buffer tape; if an error table is in the compool file, corrections are obtained

from cards in the direct reader and placed in the proper entry of the compool. If no new errors are found and all errors in the error table are corrected, the compool is transferred to the buffer tape. If a new error is found, or all errors in the error table are not corrected, a new error table is formed, and the compool is transferred to the buffer tape. The Test Control Program is informed of the no-error or error results, and may, as in the Assemble Mode, operate JMSTZ for the no-error condition. Information printouts describe all important operations of the program.

INPUT

1. Assemble Mode

Input to the Assemble Mode consists of control card, ident card, and data cards. The control card is always placed in the direct reader. The ident and data cards are usually prestored on an input tape, but may follow the control card in the direct reader.

2. Reassemble Mode

Input to the Reassemble Mode consists of a control card, a binary tape containing one to three compools, at least one of which contains errors (these errors will be correctable for the mod of JAMCZ), and correction cards with the same format as the Assemble Mode input. The correction cards must maintain a one-to-one correspondence with an error table in the compool to be corrected. The control and data cards for the Reassemble Mode are always in the direct reader, never prestored.

3. Card Formats

Date Card.

The date card is the first input card required. It supplies the date used as a part of the output page headings of a test. While its presence is not required for successful operation of the system, it should be noted that the proper date will eliminate clerical confusion generally found when handling successive runs of the same test. Its format is as follows:

Columns 1-4	DATE
Columns 7-8	Current month coded numerically -- Ø1 = January, Ø2 = February, etc.
Columns 9-10	Day of the month coded numerically.
Columns 11-12	Year coded numerically.

Assemble Master Compool Control Card.

Columns 1-5	JAMCZ
Columns 7-12	Ident of first compool
Columns 13-18	Ident of second compool
Columns 19-24	Ident of third compool
Column 25	Blank for Assemble Mode 1-7 refers to which compools, if any, have revisions in the card reader for the Reassemble Mode 1 - card revisions to first compool 2 - card revisions to second compool 3 - card revisions to first and second compools 4 - card revisions to third compool 5 - card revisions to first and third compools 6 - card revisions to second and third compools 7 - card revisions to all three compools

Ident Card (See Figure 2) - This card will be used to assign an identification to the compool.

Column 3	C
Column 7	Subsystem - This is a letter designation assigned according to the method of assigning subsystem designation as documented in FN-LO-71-1.
Column 8	C (Compool)
Columns 9-10	Model and Version Number. These are assigned according to the method of assigning models and versions as documented in FN-LO-71-1.
Columns 11-12	Modification Number. This is updated following each assembly of the Compool.

Item Card (See Figure 3) - This card is used to put item entries into the Compool.

Columns 1-2	Collate Number. This will facilitate documentation. It will be used to help arrange the cards prior to their being printed out.
Column 3	I
Columns 7-12	Item Tag
Columns 14-18	Table Reference. This is the table in which the item is located.
Columns 20-21	Item Coding. The following designations for the codes are used:

FP Floating Point (All Floating Point coded items are assumed to be signed and of maximum machine register length.)
FI Fixed Integer
BH Hollerith
MX Mixed Number
ST Status
AD Address

Column 23	Signed/Unsigned. For all Fixed Integer and Mixed Number coded items, this Column will have an "S" if Signed or a "U" if Unsigned.
Columns 25-26	Number of Bits. This is the number of bits including the sign bit, if any. No figure will be found for

	Status or Floating Point coded items.
Columns 28-29	Number of Fractional Bits. This is for all Mixed Number coded items.
Columns 52-63	Units. Units are entered here for the convenience of documentation. They will not be used in assembling the Compool. Abbreviations can be used.
Columns 65-70	Card Number
Columns 75-80	See Ident Card

Parameter Item Card (See Figure 4) - This card is used to put parameter values into the Compool.

Columns 1-2	Collate Number
Column 3	V
Columns 7-12	Item Tag
Columns 20-21	Item Coding (See Item Card)
Column 23	Signed/Unsigned
Columns 25-26	Number of Bits (See Item Card)
Columns 28-29	Number of Fractional Bits (See Item Card)
Columns 31-50	Parameter Value. This is entered starting in Column 31 according to the form specified in FN-LO-75, Section 2.5.
Columns 52-63	Units (See Item Card)
Columns 65-70	Card Number
Columns 75-80	See Ident Card

Table Card (See Figure 5) - This card is used to put table entries into the Compool.

Columns 1-2	Collate Number
Column 3	T
Columns 7-11	Table Tag. This tag begins in Column 7 for four and five character tags.
Columns 13-16	Maximum number of entries in Table.
Column 18	Table Type. If a table has a variable number of entries, enter "V". If it has a fixed number of entries, enter "R".

Status Card (See Figure 6) - This card is used to reference statuses to the proper status items. Each status item card in the Assemble Compool Deck is followed by one or more status cards containing the alphanumeric designations for the item's statuses.

Columns 1-2	Collate Number
Column 3	S
Columns 7-12	Item Tag
Columns 14-63	Item Status Symbology. Starting in Column 14, each alphanumeric designation of the Item Status is listed in the order in which they will appear in the Compool. The status symbols are separated from each other by a blank column. There may be more than one Status Card for a status item depending upon the total number of statuses for this item.
Columns 65-72	Card Number. The first six numbers will be the numbers appearing on the Item Cards. The last two numbers will vary.
Columns 75-80	See Ident Card

End Card - This card is used to describe the end of the Assemble Compool Deck.

Columns 3 - 5	END

Assemble Mode Input Structure

A. If all input is in the direct reader.

1. Date Card
2. AMC Control Card
3. Compool Ident Card
4. Mixed deck of item, table, parameter and status cards. (The order is not fixed except that status cards must follow the item they refer to.)
5. End Card
6. For up to two additional assemblies 3, 4 and 5 may be repeated.

B. If input is prestored.

1. Date card in direct reader.
2. AMC Control Card in the direct reader.
3. A prestored tape containing "A" 3, 4, 5 (and 6 if there is more than one compool to be assembled this run).

Reassemble Mode Input Structure

A. Binary input tape on C-3 containing Compool(s) to be corrected.

B. Cards in the direct reader.

1. Date Card
2. AMC Control Card
3. Correction Cards
4. End Card

(3 and 4 will be repeated for each compool on the input tape needing corrections.)

FIG 2 IDENT CARD

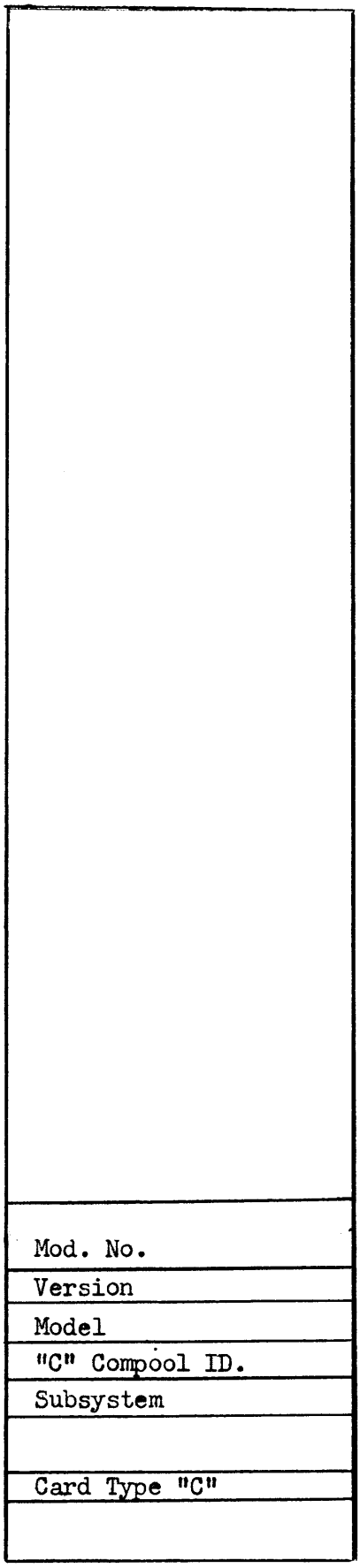


FIG 3 ITEM CARD

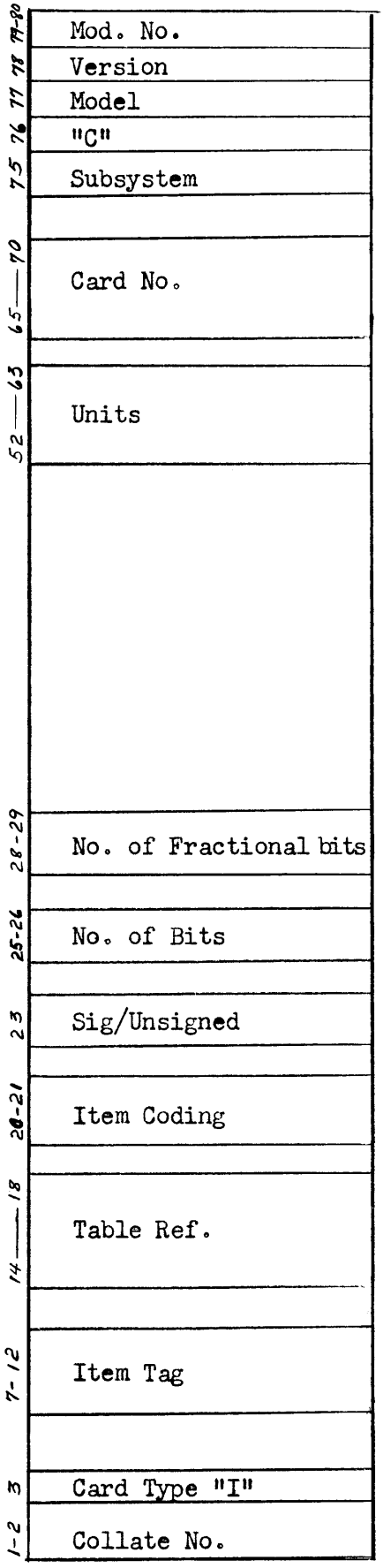


FIG 4 PARAMETER ITEM CARD

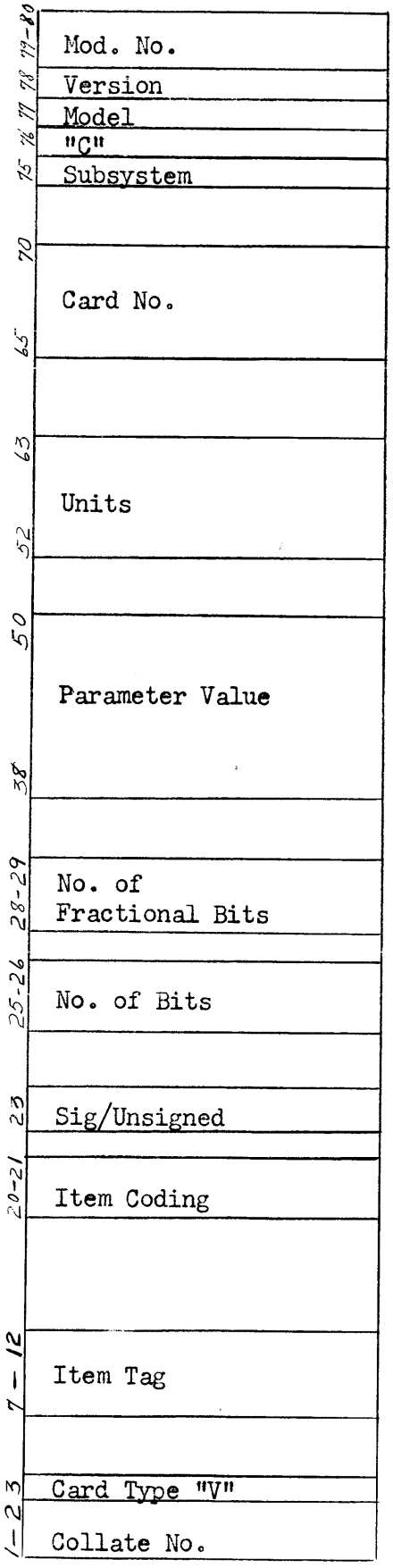
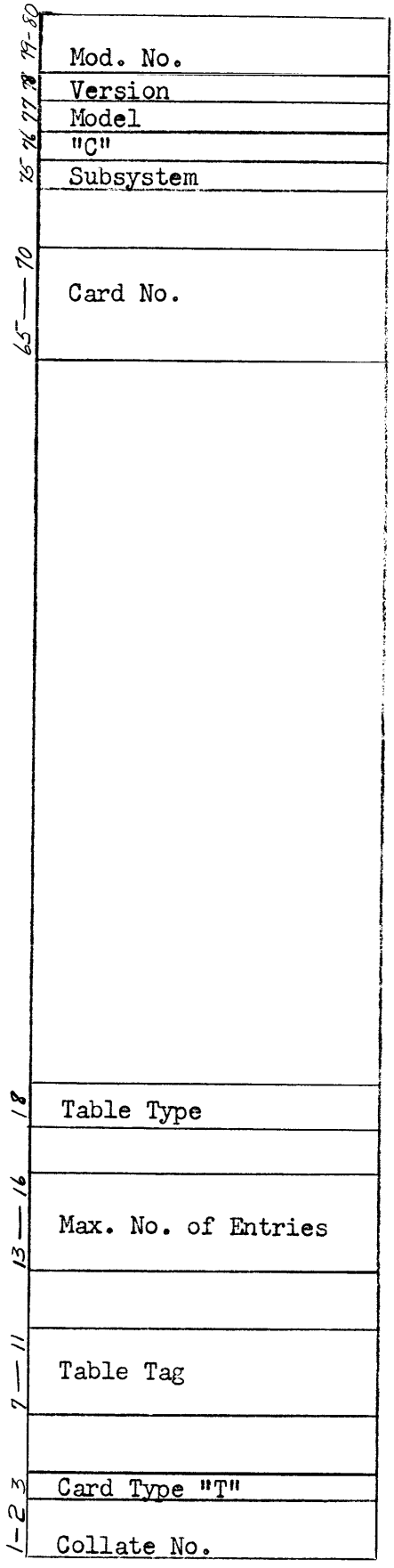


FIG 5 TABLE CARD



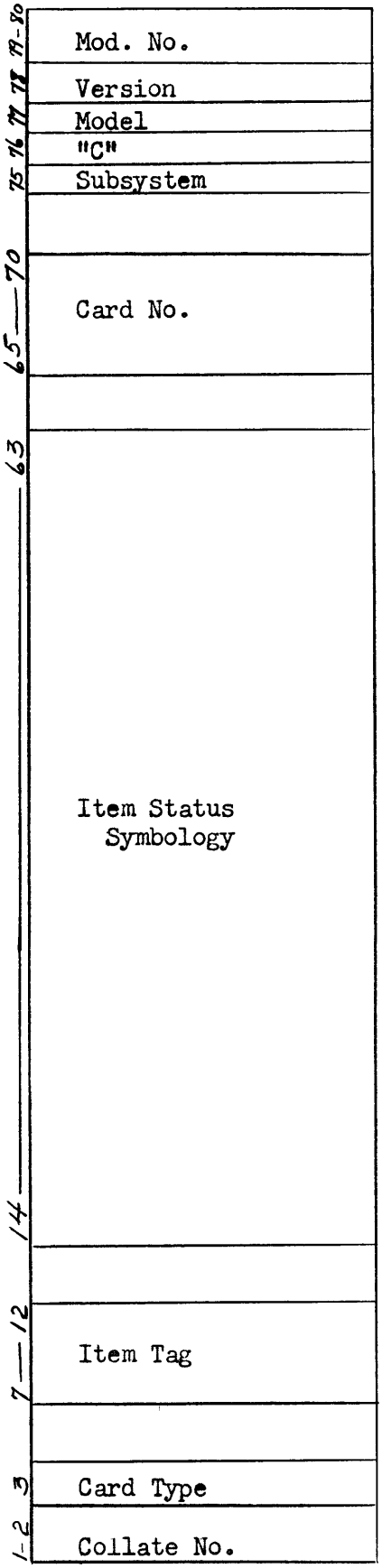


FIG 6 STATUS CARD

OUTPUT

Tape

Tape output of JAMCZ consists of one to three files on a buffer tape; one file for each Compool assembly. For an error free assembly, the file will contain three records:

RECORD 1 - IDENT

RECORD 2 - TABLE A

RECORD 3 - TABLE B

If correctable errors are discovered during an assembly, the file will contain an error record in addition to the above three.

RECORD 1 - IDENT

RECORD 2 - ERROR TABLE

RECORD 3 - TABLE A

RECORD 4 - TABLE B

MESSAGE PRINTOUTS

Heading

If the date card is missing from the test the Test Control Program will print

NO DATE

followed by

ASSEMBLE MASTER COMPOOL.

If the date card is present as the first card of the test deck,

ASSEMBLE MASTER COMPOOL DATE

will be printed.

Compool Summary

The program will produce an alphabetical list of all tables followed by items belonging to the table. No detailed description is given in this output, as this would duplicate the processing accomplished using the Baby Compool. Following this table and item summary is an alphabetical list of all value items.

Non-error Printing

A. Assemble Mode

If a Compool has been successfully assembled and transferred to the buffer tape, the program will print:

COMPOOL (IDENT) WAS ASSEMBLED SUCCESSFULLY AND IS NOW ON BUFFER TAPE A3

B. Reassemble Mode

1. If the program finds a Compool without errors in the binary input tape it will print:

   (IDENT) COPIED WITHOUT CHANGE AND IS NOW ON BUFFER TAPE A3

2. If a Compool has been corrected successfully, the program will print:

   (IDENT) CORRECTED SUCCESSFULLY AND IS NOW ON BUFFER TAPE A3

Error Printouts

A. Assemble Mode

1. If a read check is encountered while reading binary tape, the program will print:

   READ CHECK ON A2, PRESTORED INPUT

2. If a write check is encountered while writing on a binary tape, the program will print:

   WRITE CHECK ON A3, COMPOOL OUTPUT TAPE

3. If an item tag does not follow the specification in FN-LO-71, the program will print:

   (COMTAG) IS AN ILLEGAL ITEM TAG

   The card with the illegal tag will be printed below.

4. If a table COMTAG does not follow the specification in FN-LO-71, the program will print:

   TABLE TAG BELOW IS ILLEGAL

   The card with the illegal tag will be printed below.

5. If an error is found on any card, the card will be printed and the value of the column counter will be indicated.

6. If more than one hundred cards are in error, assembly will be terminated and the program will print:

   TOO MANY ERRORS TRY AGAIN LATER

7. If the first card does not have a "C" punched in column 3, the program will print:

   NO IDENT CARD

8. If the Ident on the Ident card does not match the Ident on the Control Card, the program will print:

   IDENTS PRINTED ON CARD BELOW DO NOT MATCH

   A printout of both cards will follow.

9. If an item contains a table tag punched in columns 14 to 18, but this table has no defining card, the program will print:

   NO DEFINING CARD FOR TABLE (COMTAG) IN ENTRY FOR ITEM (COMTAG)

10. If correctable errors were found during the assembly process and the Compool was transferred to the buffer tape, the program will print:

    COMPOOL (IDENT) CONTAINS CORRECTABLE ERRORS AND IS NOW ON BUFFER TAPE A3

11. If errors are found which are not correctable for this mod of JAMCZ Reassemble Program, the program will print:

    DECK FOR COMPOOL IDENT--MUST BE ASSEMBLED OVER FOR ERRORS LISTED ABOVE. NO TRANSFER TO TAPE

12. If a status item card contains no statuses and is not followed by a status card, the program will print:

    STATUS ITEM NOT FOLLOWED BY A STATUS ITEM CARD

13. If a status card has no statuses punched, the program will print:

    STATUS CARD HAS NO STATUSES

14. If a status card is not preceded by a status card or a status-coded item card, the program will print:

    STATUS CARD NOT PRECEDED BY A STATUS ITEM CARD

15. If the item tag on the status card does not match the item tag on the item card, the program will print:

    ITEM TAG ON STATUS CARD DOES NOT MATCH ITEM

B. Reassemble Mode

1. If a read check is encountered while reading the binary tape, the program will print:

    READ CHECK ON C3, COMPOOL INPUT TAPE

2. If a write check is encountered while writing on A3, the program will print:

    WRITE CHECK ON A3, COMPOOL OUTPUT TAPE

3. If the ident of the Compool on the input tape is different from the ident on the control card, the program will print:

    (IDENT)AND(IDENT) COMPOOL IDENTS DO NOT MATCH. CHECK YOUR CONTROL CARD

4. If a correction card does not match an entry in the Error Table, the program will print:

    (COMTAG) NOT IN THE ERROR TABLE

5. If an entry in the Error Table has not been corrected by a correction card, the program will print:

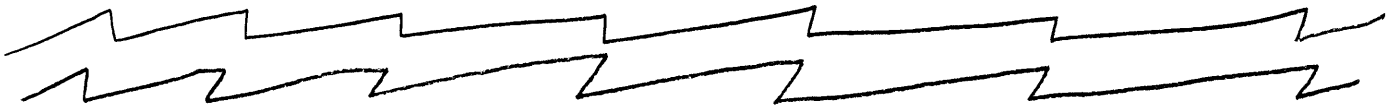
   (COMTAG) HAS NOT BEEN CORRECTED

6. If any error is found during a reassembly attempt, the program will print the error (2,3,4 or 5 above) and also print:

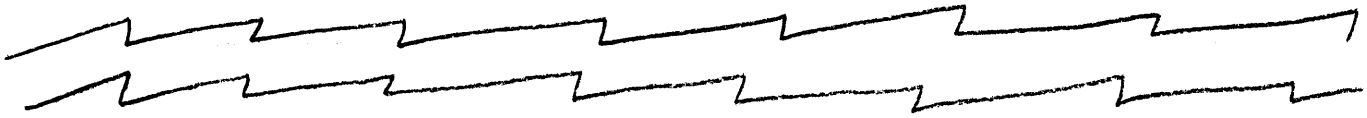
   (IDENT) HAS NOT BEEN CORRECTED. PLEASE SAVE OLD BUFFER TAPE AND AFTER CORRECTING MISTAKE INDICATED START OVER AGAIN

Sample Output

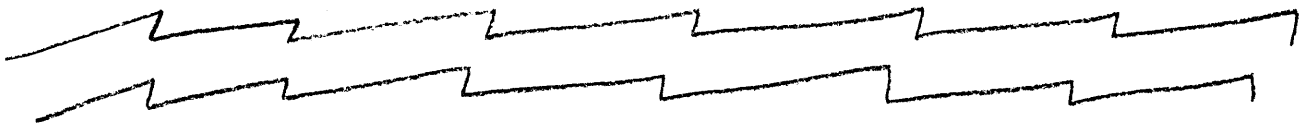
```
                                                              Page 1

   ASSEMBLE MASTER COMPOOL
01/14/60

SUMMARY INFORMATION ASSEMBLE MASTER COMPOOL.  COMPOOL IDENT      XXXXXX

     TABLE COM11                    ITEMS

          ITEMAB      ITEMAC      ITEMAD      .....................
```



```
   ASSEMBLE MASTER COMPOOL                                    Page 2
01/14/60

     TABLE COM12                    ITEMS

          ITEMBA      ITEMBB      ITEMBC      .....................
```



```
   ASSEMBLE MASTER COMPOOL                                   Page 10
01/14/60

     TABLE COM19                    ITEMS

          ITEMGA      ITEMGB      ITEMGC      .....................

COMPOOL XXXXXX WAS ASSEMBLED SUCCESSFULLY AND IS NOW ON BUFFER TAPE A3.
```



```
   ASSEMBLE MASTER                                           Page 11
01/14/60

SUMMARY INFORMATION ASSEMBLE MASTER COMPOOL.  COMPOOL IDENT      YYYYYY
```

The remaining format is the same as for compool XXXXXX.

Figure 7 Sample Output

THE SYSTEM TAPE LOADING FUNCTION

The System Tape Loading Function can produce from one to three identical binary output tapes by combining files from three possible sources of simultaneous input. Input data can be in the form of binary cards (7Ø9 or 7Ø4, row or column), loaded directly or on prestored tape, and binary tapes. Input and output hardware assignments are preset but can be changed. Output tapes are identified by a three-character alphanumeric code. Completed output tapes are accompanied by a table of contents.

INPUT

There are two categories of input to the System Tape Loading Function: Program Control Input which controls the output tape making operation and Data Input which is the material from which the output tapes are made.

A. Program Control Input

Program Control Input is always Hollerith-coded symbolic cards read from the on-line card reader in the following sequence:

1. Date Card

This card supplies the date which is used as a part of the output page headings of a test. While its presence is not required for successful operation of the system, it should be noted that the proper date will eliminate clerical confusion generally found when handling successive runs of the same test. Its format is as follows:

Columns 1-4	DATE
Columns 7-8	Current month coded numerically (Ø1 = January, Ø2 = February, etc.)

Columns 9-10 Day of month coded numerically.

Columns 11-12 Year coded numerically.

2. Master Control Card (See Figure 8)

Columns 1-3 MST

The remaining columns of the card are used only when changes to the following preset environment variables are desired:

Columns 4-6 Identification code of the output tapes = ØØØ.
(The code is changed by inserting the desired three alphanumeric characters in the columns.)

Columns 7-42 Input and output hardware assignments and their form are shown in the following chart.

Column 43 Three output tapes will be made.
(The preset number of outputs is changed by inserting the number desired in the column. Any number greater than three will be interpreted as three.)

Changes in the preset hardware assignments are made by entering the name of the particular hardware desired in the appropriate six-column field indicated in the last column of the chart. The first of the six columns indicates the form of the input or output:

B for binary tape

D for BCD tape (binary cards prestored on tape)

blank for binary cards

The second and third columns name a specific I/O unit in the following manner:

Tape unit	DSC and tape unit number, such as B2 or D5.
Card reader	The letters RD.

The last three columns of the field are unused and must remain blank. Any six-column field left completely blank indicates that the preset hardware should be used.

	PRESET HARDWARE ASSIGNMENTS			
Name of I/O Source	I/O Unit	Form	Intended Use	Card Column
Input 1	Tape Unit A1	Binary Tape	Old System Master Tape	7-12
Input 2	Tape Unit A4	Binary Cards (Prestored)	New Programs	13-18
Input 3	Tape Unit A3	Binary Tape	New Co[illegible]	19-24
Output 1	Tape Unit B3	Binary Tape	New System Master Tapes	25-30
Output 2	Tape Unit C3	Binary Tape		31-36
Output 3	Tape Unit D3	Binary Tape		37-42

Restrictions on hardware assignments:

1. The card reader can be used only on DSC A, and only on Input 2.
2. All inputs must be on the same DSC.
3. All outputs must be on different DSC's, and different also from the input channels.

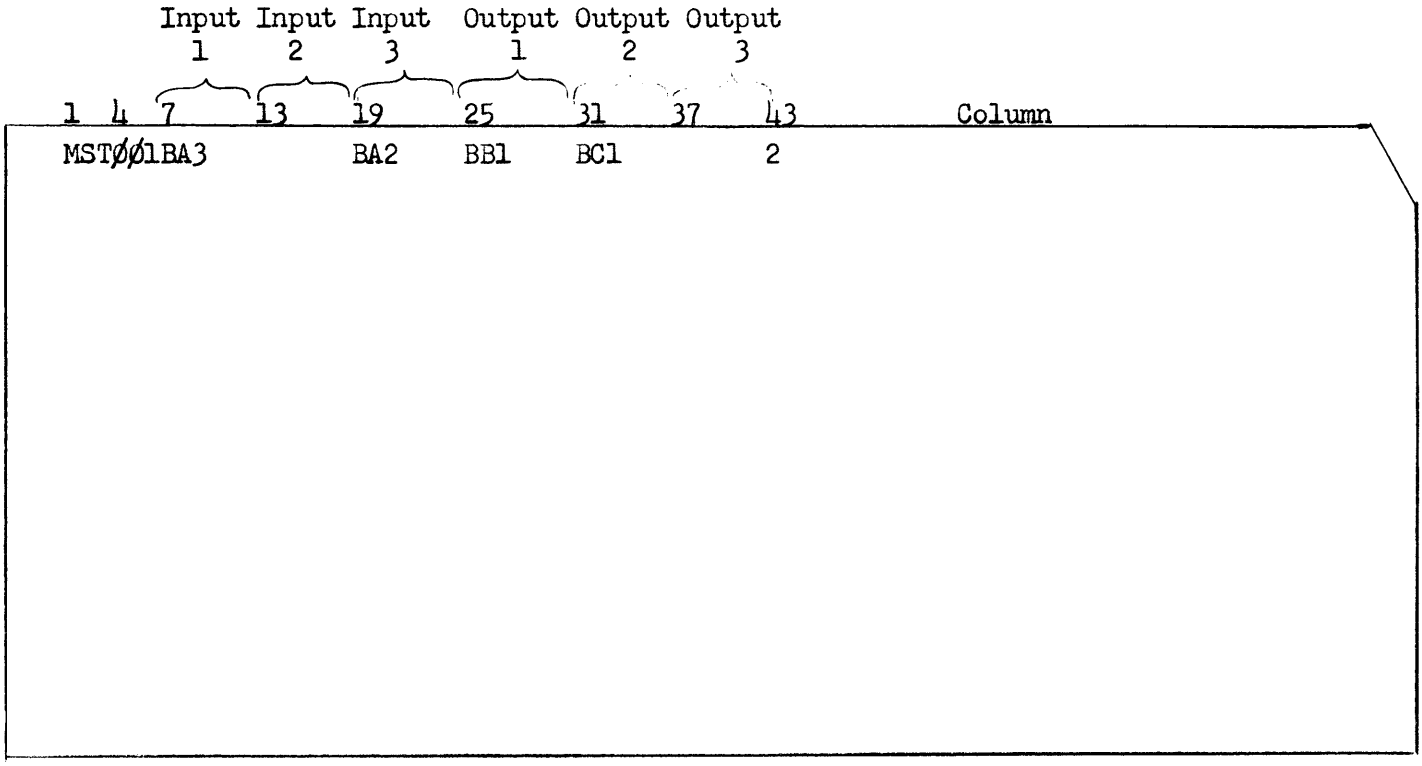


Figure 8

Sample Master Control Card

The following changes in preset environment variables are indicated:

1. Output identification code ØØ1.

2. Hardware assignments for Input 1, Input 3, Output 1, and Output 2 are changed to BA3 (binary tape on drive A3), BA2, BB1 and BC1 respectively. Input 2 and Output 3 remain as preset.

3. Two output tapes are to be made.

3. Data Control Cards (See Figure 9)

There are five possible command words which can appear in columns 1 through 6 of any Data Control Card:

DUPE	Reproduces from a binary tape.
MAKE	Reproduces from binary cards.
POS	Positions a binary tape to the beginning of the indicated file.
POS ID	Searches a binary tape for the file bearing the indicated identification tag, and positions the tape to the beginning of that file.
FINISH	Indicates the End of Data Control Cards.

Procedure: In order to reproduce any section (number of files) from an input source, the Data Control Cards must indicate for that section:

a) The input source (Input 1, 2 or 3) to be reproduced.

b) The location of the first file to be reproduced.

c) The number of files to be reproduced.

The input source is indicated by making an entry in the appropriate six-column I/O source field to the right of the command word. Columns 7-12, 13-18, 19-24, and 25-30 correspond to Input 1, Input 2, Input 3, and Outputs, respectively. There is only one field for the three possible output tapes since they are always identical. All command words and entries in I/O source fields must be left-justified.

The location of the first file and the number of files to be reproduced are most frequently indicated by one of the following Data

Control Card sequences:

a. A position card (POS or POS ID) followed by a DUPE card.

	Command Word	Appropriate I/O Source Field	Explanation
Example:	POS	5	Position input tape to the beginning of file #5.
	DUPE	3	Reproduce files #5, 6, and 7.
Example:	POS ID	JAMCZ	Position input tape to the beginning of the file with ID JAMCZ.
	DUPE	3	Reproduce files JAMCZ, JAMCZ + 1, and JAMCZ + 2.

b. A single production card (MAKE or DUPE) with an ordinal number in the I/O source field, indicating the number of the one file to be reproduced.

Example:	MAKE	8	Reproduce file #8.
Example:	DUPE	827	Reproduce file #827.

c. A multiple-entry production card which indicates the first and last files to be reproduced, spearated by a slash.

Example:	DUPE	8/12	Reproduce files #8, 9, 1Ø, 11, and 12.
Example:	MAKE	1/3	Reproduce files #1, 2, and 3.

Limitations:

a. The card sequence POS or POS ID followed by a MAKE card is illegal. Only a DUPE (a binary tape) can be positioned. Binary cards must be loaded in the card reader in the correct sequence.

b. The maximum number of files that can be input by any one of the preceding Data Control Card sequences is 1Ø23 decimal. An attempt to input a larger number will result in a "content" error. (See OUTPUT MESSAGES, below.)

Example: POS 8
DUPE 1Ø24

is an illegal sequence. One possible correct sequence would be:

POS 8
DUPE 1
POS 9
DUPE 1Ø23

Example: DUPE 1/1Ø25

is an illegal sequence. A correct one would be:

DUPE 1/1Ø23
POS 1Ø24
DUPE 2

Two additional Data Control Card sequences are available:

1. The output tapes can be positioned by using a POS or POS ID card with an entry in the Outputs columns, 25-3Ø. This may be helpful when the tape-making operation is interrupted, and it is desired to resume the operation using the partially made tapes rather than start all over again. The files that were successfully written on the output tapes can be identified from the Table of Contents. It should be noted that if a card positioning

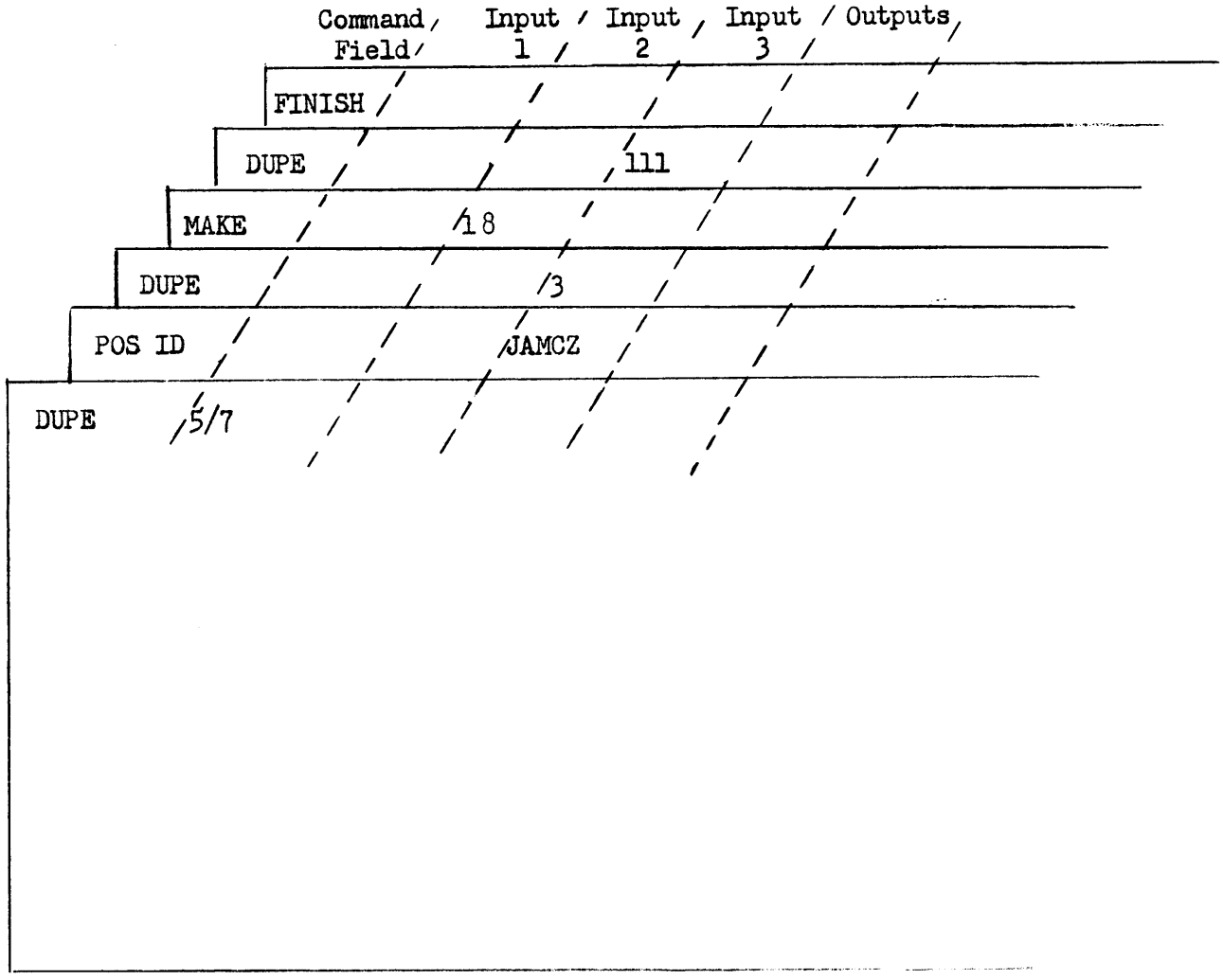


Figure 9

Sample Data Control Cards

The following operations are indicated:

1. Reproduce file #5, #6 & #7 from Input 1.

2. Reproduce files JAMCZ, JAMCZ+1 and JAMCZ+2 from Input 3.

3. Reproduce file #18 from Input 2.

4. Reproduce file #111 from Input 3.

the output tapes is used, it must be the first Data Control Card following the Master Control Card.

2. A MAKE or DUPE ALL Card. The letters ALL in a source field result in the reproduction of all files remaining on that source. In the case of a DUPE ALL, this would mean all of the files beyond the reader head, and for a MAKE ALL, all of the cards still in the card reader (or all of the prestored files beyond the reader head). It should be noted that the ALL card can be used to make duplicate tapes if it is the only Data Control Card.

		Input 1	
Example:	DUPE	ALL	Reproduce all files from Input 1 tape.

4. End of Data Control Cards

This is indicated by three cards: a FINISH card, followed by two completely blank cards. Any card following the second blank card will be interpreted as a Data Input Card.

B. Data Input

The System Tape Loading Function will load output tapes from one to three simultaneous input sources. The preset hardware assignments for Input 1, Input 2, and Input 3 should be used for an old System Master Tape, new programs, and new Compools, respectively. New programs are usually binary cards, while old System Master Tapes and new Compools are always binary tapes.

1. Binary Cards

Binary Data Input Cards can be in 7Ø4 or 7Ø9, row or column binary format. The usual sources of these formats are as follows:

SOS "punch absolute" on-line gives 7Ø9 row.

SOS "punch absolute" off-line gives 7Ø9 column.

SE9AP gives 7Ø4 row.

SCRAM RB mode gives 7Ø4 row.

SCRAM normal mode gives 7Ø4 column.

Note: Every SCRAM compilation results in a two-part binary deck, the first of which is a symbolic table, and the second the actual binary program. It is the second part of this deck which is used with JMSTZ.

Only binary cards in column format can be prestored on tape. Row binary cards must be input directly through the on-line card reader, in which case the Master Control Card should indicate that the card reader will be assigned to Input 2 with the letters RD in columns 26 and 27, column 25 remaining blank.

If there is any question about the format of a binary deck, it can usually be resolved by examining the individual cards, since formats can be recognized by their characteristic punches or lack of punches. A binary card in one of the following formats must have all of the punches or blanks described below for that format in order to be legitimate. It should be noted that the names of the formats do not mean that the cards are produced by, or used on, any particular

machine. They are only the names of the formats, and confusing ones at that.

7Ø9 Row	Columns 1 & 11 have no punches in the 9's row. Columns 1Ø & 12 have punches in the 9's row.
7Ø9 Column	Column 1 has no punches in the 12's and 8's rows. has some punches in the word count (11, Ø, 1, 2, & 3's row), and has punches in the 7's and 9's rows.
7Ø4 Row	Columns 1 through 13 have no punches in the 9's row.
7Ø4 Column	Column 1 has punches in the 7's and 9's rows, but has no punches in the remaining rows of the column. Column 2 has no punch in the 12's row.

If a deck of binary Data Input Cards is out of order, it will be sorted by the Tape Loading Function according to card address. If there are any cards with the same address in a given input file (two or more data words assigned the same location in core), the last of these conflicting cards to be read in will be the only one to appear on the output tapes.

The Data Input Deck must be composed of the following cards:

a. A symbolic identification card using standard Hollerith code and in the following format:

   Columns 1-5 IDENT

Column 6 — The number 4, if the data cards following the identification card are in 704 format, or a blank if they are in 709 format.

Columns 7-12 — The identification tag (six letters or less, left-justified, with no imbedded blanks) of the binary cards which follow the identification card.

b. One completely blank card.

c. The one-file deck of binary cards.

d. An "end-of-file" card which is completely blank except for:

Column 1 — a 7 and 9 punch for a column binary input deck.

Columns 10 & 12 — a 9 punch for a row binary input deck.

This group of cards (identification card, blank card, binary cards, and "end-of-file" card) represent a one-file input deck. Any additional input files that are to be used as Data Input should follow in the order in which they will be loaded on the output tapes.

2. Files on Binary Tape

There are two sources of binary tape inputs:

a. Old Master Tapes (previously made by the System Tape Loading Function). Their formats are described in OUTPUTS, below.

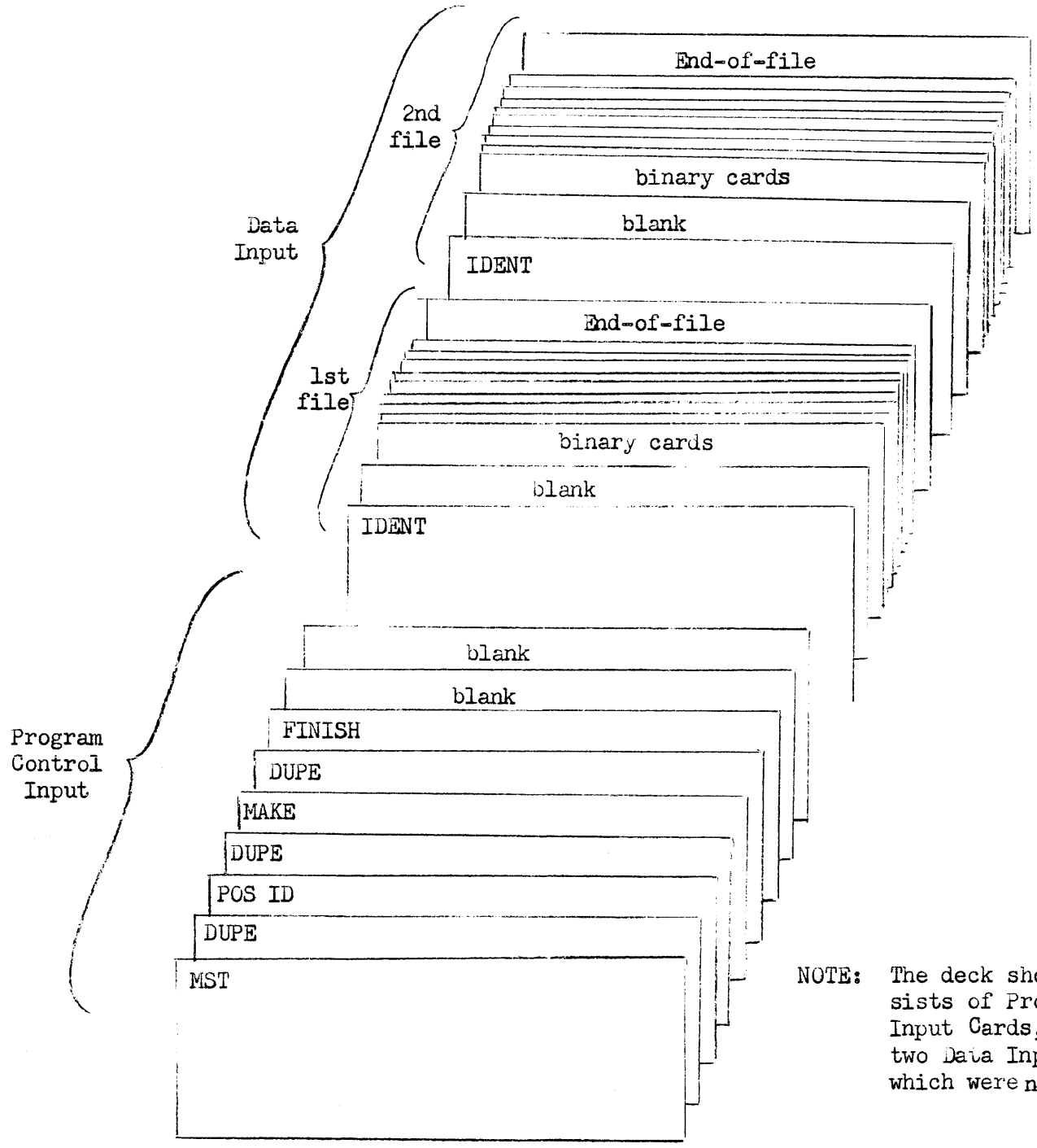
b. Output of the Assemble Master Compool Function.

C. Deck Structure (See Figure 10)

A complete input deck for the operation of the System Tape Loading Function consists of the following order of Program Control Cards:

1. Date Card.

2. Master Control Card.



NOTE: The deck shown consists of Program Control Input Cards, followed by two Data Input files which were not prestored.

Figure 10

Sample Input Deck Structure

3. Data Control Cards.

4. End of Data Control Cards.

If there are any Data Input Cards in column format, they should be prestored on tape prior to the operation of the System Tape Loading Function. If the Data Input Cards are in row format, or the prestoring equipment is unavailable or out of order, the Data Input Cards are read through the on-line card reader, in which case they should be placed directly after the Program Control Cards, and the Master Control Card should have the letters RD in columns 26 and 27, column 25 remaining blank, to indicate that the hardware used on Input 2 is changed.

Whether prestored on tape, or read through the on-line card reader, the Data Input Deck must have the following order of cards:

1. The identification card.
2. A blank card.
3. A deck of binary cards.
4. An "end-of-file" card.
5. Any number of additional files (in the order shown in 1, 2, 3, and 4, above) in the sequence in which they are to be used by the Tape Loading Function.

OUTPUT

A. Files on Binary Tape (See Figure 11)

The primary function of the System Tape Loading Function is the production of new System Master Tapes. The number of output tapes produced is determined by the number (1, 2, or 3) in column 43 of the Master Control Card.

If one tape is desired, output will appear on tape unit B3, if two tapes, on B3 and C3, and if three tapes, on B3, C3, and D3.

Output tapes will have the following format:

1. A "load-tape-into-core" record which enables a system tape to "load itself" into core. Use of the output tape containing this record begins with the pressing of the "Load-From-Tape" Switch on the 7Ø9 console which results in the reading into core of the Test Control Program and transfer of control to it.
2. An "end-of-record" gap.
3. A two-word header record whose first word contains the identification tag of the data records which follow it. The second word contains the DSC command IOCP L,Ø,C where C is the number of words in the data record which follows it, and L is the starting location of that data record.
4. An "end-of-record"gap.
5. The data record whose identification tag appears in the first word of the preceding header record.
6. An "end-of-record" gap (caused by a symbolic TCD card or an END card).
7. Any number of additional data records, within the physical limitations of the tape, followed by an "end-of-record" gap as just described in 5 and 6, above.
8. An "end-of-file" record which indicates the end of one program.
9. An "end-of-record" gap.
10. Any number of additional files, within the physical limitations of the tape, each of which consists of the header record, an "end-of-record" gap, data records (each with its appropriate "end-of-record" gap), an "end-of-file" record, and an "end-of-record" gap as just described in 3, 4, 5, 6, 7, 8, and 9, above.

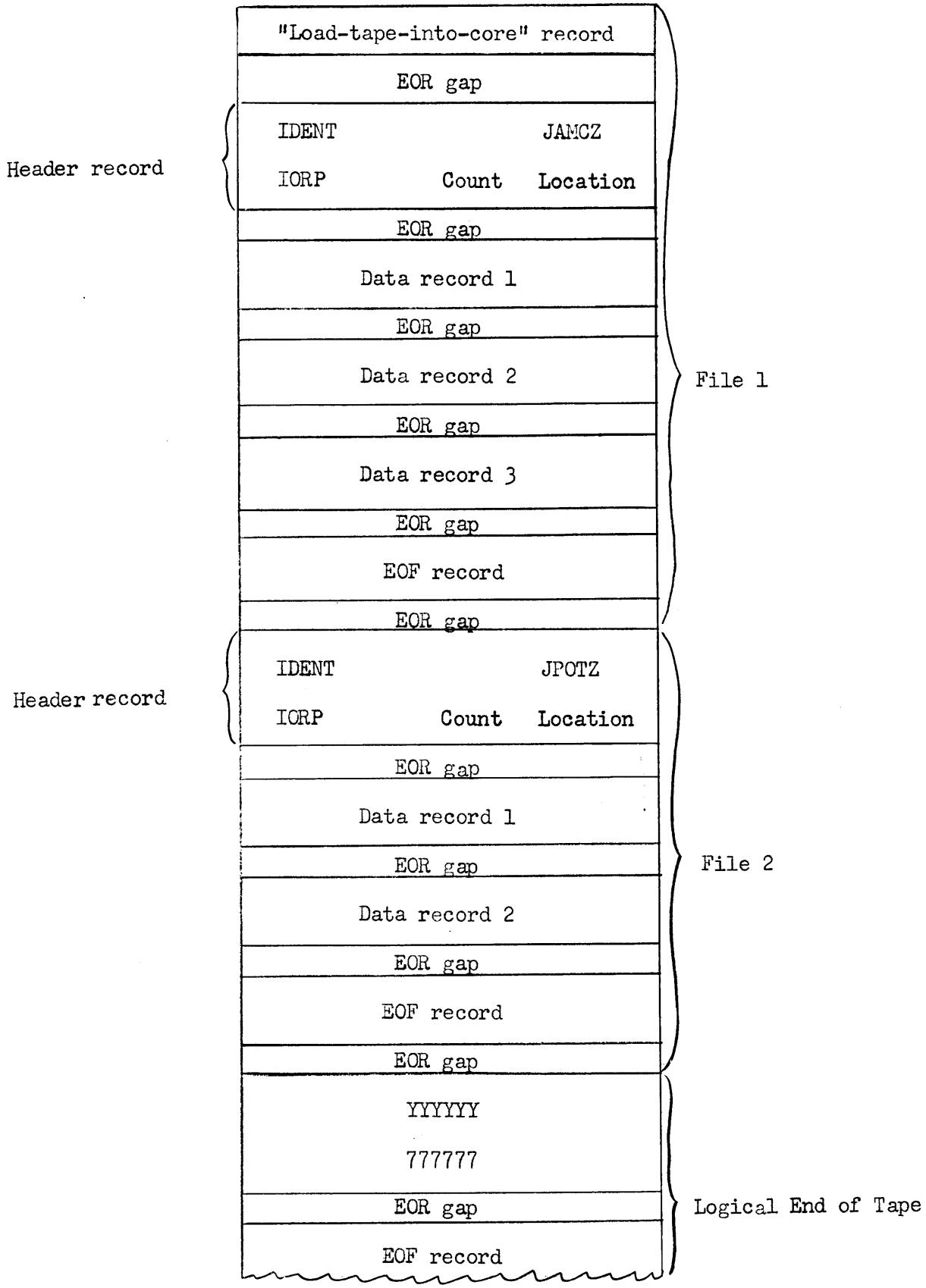


Figure 11-- Sample Format of an Output Tape - The tape shown contains file JAMCZ with the "load-tape-into-core" record and file JPOTZ.

11. A Logical End of Tape which consists of:

    a. A two-word record which contains the BCD registers of YYYYYY and 777777, respectively.

    b. An "end-of-record" gap.

    c. An "end-of-file" record.

MESSAGE PRINTOUTS

During the operation of the System Tape Loading Function, on- and off-line statements are printed which are needed to control the program as well as indicate the current status of the operation.

A. If the date card is omitted or contains an error, the following words will appear:

    NO DATE

B. The first line of output will be the words:

    SYSTEM TAPE LOADING PROGRAM

C. If the READY light on the card reader is out and the card reader is selected by the Tape Loading Program, the console operator is asked to "ready" the card reader and depress the START switch on the 7Ø9 console with the words:

    CD RDR NOT READY * * READY AND HIT STRT

    This most often occurs when:

    a. An incomplete deck of Program Control Cards is placed in the card reader.

    b. The STOP switch on the card reader is accidentally depressed.

D. The contents of all Program Control Cards (excluding blanks) will be listed in the order in which they are read.

Example: MSTØØ1

POSID JAMCZ

DUPE 2

MAKE 3

FINISH

E. If there are any errors in the control cards, one of the following statements will appear to the right of the incorrect card's print-out:

HAS CONTNT ERROR

HAS LOGIC ERROR

HAS SEQUTL ERROR

F. If any errors are found in the control cards after they have all been read in, or in a Compool prior to the operation of JMSTZ, one of the following statements will appear:

MSTER COMPOOL HAS ERRORS MOST WILL NOT MAKE TAPE

CNTROL CARDS HAVE ERRORS MOST WILL NOT MAKE TAPE

MSTER COMPOOL AND CONTROL CARDS HAVE ERRORS MOST WILL NOT MAKE TAPE

G. If any trouble is encountered during the execution of a read/write or input/output operation, an appropriate statement stating the operation in difficulty will appear.

$$\text{CAN'T} \begin{cases} \text{START} \\ \text{FINISH} \end{cases} \begin{cases} \text{READ} \\ \text{WRITE} \\ \text{POSTN} \end{cases} \text{OF} \begin{cases} \text{TAPE (a)} \\ \text{READER} \end{cases}$$

Where (a) is the name of the specific tape unit.

H. As each file is written on the output tapes, its corresponding entry in the Table of Contents of the new Master Tapes will be printed. Each entry will

have the following form:

FILE _(a)_ _ _(b)_ _ _(c)_ ON OUTPUT TAPES.

Where

(a) is the ordinal number of the file on the output tapes.

(b) is the identification tag of the file.

(c) is the modification or version number of the file (program).

I. Completion of a successful tape-making operation will be signaled by the words:

LOGICAL END OF TAPE ON OUTPUT TAPES

SAVE TAPES (a) (b) (c) AND LABEL AS FOLLOWS

IM VERSION _(d)_

Where

(a), (b), and (c) are the names of the output units.

IM stands for JOVIAL Interpreter System Master Tapes.

(d) contains the output tape identification code.

System Tape Loading Sample Problem

A. Problem

Assume that the current System Master Tape with the identification code ØØ1 is to be updated. It contains the following order of files:

1. Test Control Program, with the "load-tape-into-core" record as the first record. The program has the identification tag JTCPZ and modification number ØØ3.
2. Interpreter First Pass Program, JALLZØØ1.
3 & 4. Two Compools, JCXØØ3 and JCYØØ4, in a forgotten order. (Assume that the Table of Contents of Master Tape ØØ1 has been lost).

5. Assemble Baby Compool Program, JABCZØØ5.
6. Data Simulation Program, JSTRZØØ1.
7. Interpreter Second Pass Program, JOLLZØ1Ø.
8. Data Processing Program, JDSYZØØ2.
9. Assemble Master Compool Program, JAMCZØØ1.
10. System Tape Loading Program, JMSTZØØ1.
11. Logical End of Tape.

Two new System Master Tapes are to be made from the tape described above, with the following changes:

1. Compool JCXØØ3 is to be removed.
2. Compool JCXØØ5, which is the only file on a binary buffer tape, is to be added directly after Compool JCYØØ4.
3. The Interpreter Second Pass Program, JOLLZØ1Ø, is to be replaced with a new version, JOLLZØ11, which is a deck of 7Ø4 row binary cards produced by the SE9AP System.

The new master tapes will have the identification code ØØ2, and DSC B is not operating.

B. Procedure

1. A Program Control Input Deck is prepared consisting of the following cards:
   a. A Date Card containing the date February 1, 196Ø.
   b. A Master Control Card.

Column	Contents	Explanation
1-3	MST	
4-6	ØØ2	The new master tape identification number.

Column	Contents	Explanation
25-3Ø	BD3	Output 1 hardware is changed to D3 binary tape since DSC B is not operating.
43	2	Two output tapes will be made.

The remaining columns of the card remain blank. Note that the hardware now assigned to both Output 1 and Output 3 is tape unit D3. This, it would appear, should result in a "content" error statement to the right of the Master Control Card printout during the operation of the Tape Loading Function, since it is impossible to place two tapes on one tape unit. However, since only two output tapes will be made (Column 43, above), Output 3 will not be used, and so no error will be indicated. If three output tapes were to be made using this hardware set-up, the "content" error statement would appear.

c. Data Control Cards in the following order:

```
          Input Input Input
            1     2     3
DUPE  1/2
POS IDJCYØØ3
DUPE  1
DUPE              1
POS   5
DUPE  2
MAKE        1
DUPE  8/1Ø
```

The sequence of cards shown above indicates the following operations:

Position the current master tape to the beginning of file #1, and reproduce two files, JTCPZØØ3 and JALLZØØ1.

Position the current master tape to the beginning of the file JCYØØ3, and then reproduce one file.

Reproduce one file, JCXØØ5, from the new Compool buffer tape.

Position the current master tape to the beginning of file #5, and then reproduce two files, JABCZØØ5 and JSTRZØØ1.

Reproduce one file, JOLLZØ11, from the on-line card reader.

Reproduce files #8, 9, and 1Ø from the current master tape, JDSYZØØ2, JAMCZØØ1, and JMSTZØØ1, respectively.

d. End of Data Control Cards consisting of a FINISH card, followed by two completely blank cards. This will result in the writing of a Logical End of Tape on the output tapes.

2. A Data Input Deck is prepared consisting of the following cards:

a. An identification card.

Column	Contents	Explanation
1-5	IDENT	
6	4	The Data Input cards are in 7Ø4 format.
7-12	JOLLZ	Identification tag of the following binary cards.

b. A completely blank card.

c. The deck of 7Ø4 row binary cards which represent the new version of the Interpreter Second Pass Program, JOLLZØ11.

d. A row binary "end-of-file" card, containing a 9 punch in columns 10 and 12.

3. The decks prepared in 1 and 2 above, are placed together such that the Program Control Input Deck is read first by the card reader, followed by the Data Input Deck. (See Figure 3)

4. The current System Master Tape, 001, is placed on tape unit A1 and the new Compool buffer tape on tape unit A3. The combined deck of Control and Data Cards is placed in the card reader. Blank tapes are placed on tape units C3 and D3.

5. The "Load From Tape" Switch is depressed.

C. Output

The operation of the System Tape Loading Function will result in the printout shown in Figure 12, and two output tapes with the format shown in Figure 13.

```
DATE  Ø2Ø16Ø

MSTØØ2                    BD3              2

DUPE  1/2

POS IDJCYØØ4

DUPE  1

DUPE              1

POS   5

DUPE  2

MAKE        1

DUPE  8/1Ø

FINISH

FILE  ØØ1   JTCPZ    ØØ3   ON OUTPUT TAPES

FILE  ØØ2   JALLZ    ØØ1   ON OUTPUT TAPES

FILE  ØØ3   JCYØØ4         ON OUTPUT TAPES

FILE  ØØ4   JCXØØ5         ON OUTPUT TAPES

FILE  ØØ5   JABCZ    ØØ5   ON OUTPUT TAPES

FILE  ØØ6   JSTRZ    ØØ1   ON OUTPUT TAPES

FILE  ØØ7   JOLLZ    Ø11   ON OUTPUT TAPES

FILE  ØØ8   JDSYZ    ØØ2   ON OUTPUT TAPES

FILE  ØØ9   JAMCZ    ØØ1   ON OUTPUT TAPES

FILE  Ø1Ø   JMSTZ    ØØ1   ON OUTPUT TAPES

      LOGICAL END OF TAPE   ON OUTPUT TAPES

SAVE TAPES  D3  C3      AND LABEL AS FOLLOWS  IM VERSION ØØ2
```

Figure 12

Printout Produced in Tape Loading Example

The printout shown includes Program Control Cards followed by the New Master Tape Table of Contents.

File No.	Contents	
1	Test Control Program, JTCPZØØ3	"Load-tape-into-core" record JTCPZ IORP 6Ø67 764Ø data record EOF record
2	Interpreter First Pass Program, JALLZØØ1	JALLZ IORP 25237 764Ø data record EOF record
3	Compool, JCYØØ4	JCYØØ4 243Ø 4ØØØ 243Ø word data record 4ØØØ word data record EOF record
4	Compool, JCXØØ5	JCXØØ5 23ØØ 1Ø25 23ØØ word data record 1Ø25 word data record EOF record
5	Assemble Baby Compool, JABCZØØ5	JABCZ IORP 755 764Ø data record EOF record
6	Data Simulation Program, JSTRZØØ1	JSTRZ IORP 257Ø 764Ø data record EOF record
7	Interpreter Second Pass Program, JOLLZØ11	JOLLZ IORP 4753 764Ø data record EOF record
8	Data Processing Program, JDSYZØØ2	JDSYZ IORP 314? 764Ø data record EOF record
9	Assemble Master Compool Program, JAMCZØØ1	JAMCZ IORP 6067 764Ø data record EOF record
1Ø	System Tape Loading Program, JMSTZØØ1	JMSTZ IORP 12ØØØ 764Ø data record EOF record
11	Logical End of Tape file	777777 YYYYYY EOF record

Figure 13 - Format of Output Tapes Produced in Tape Loading Example.

USE OF THE JOVIAL INTERPRETATION FUNCTION

The major function of the JOVIAL Interpreter System is that of interpreting (i.e., translating, executing, and testing) programs written in the JOVIAL language. Certain basic concepts that are considered to be requisites for a testing activity are inherent in this function:

1. System control to be maintained throughout.
2. A method to be provided for the introduction of a controlled environment.
3. All output to be in a form that allows maximum analysis and evaluation of the operation with a minumum of clerical effort.

Interpretation of a program is perhaps one of the last steps in a cycle of events. The system to be produced must be defined, requirements set forth and programs designed; all this being done prior to the actual programming and coding of the system programs. The programming language employed in the production of SACCS is known as JOVIAL. No attempt will be made here to fully describe the language; it is suggested that readers be familar with FN-LO-34-2, by Jules Schwartz. Assuming that the cycle of events has progressed to the point where the JOVIAL program has been coded, the user or test designer is now ready to use the Interpreter System. Certain major decisions must be made prior to test preparation. First is the decision as to which COMPOOL should be associated with the testing of this program. Several Master COMPOOLS, reflecting different model or version data definitions, may be available at any one time and will physically be included on the JOVIAL Interpreter System Master Tape.

The process of choosing the applicable COMPOOL is a decision that must be made by the programmer. The second major decision available is the tracing mode

to be operated. Two modes are available. The first is an "automatic" mode which need not be designated on the Test Control Card. The second is a "full" mode and must be designated on the card prior to testing. Choice of mode will be dictated by the problems discovered in previous test runs. The assumption here is that the full trace mode will be chosen only when insufficient information is gained from the automatic mode.

INPUT

Input to a test is in the form of punched cards. These may later be prestored onto tape for input into the computer, but must originate as cards from the test designer. Formats of all required cards are explained below in the order of their use.

A. Date Card

The date card is the first input card required. It supplies the date used as a part of the output page headings of a test. While its presence is not required for successful operation of the system, it should be noted that the proper date will eliminate clerical confusion generally found when handling successive runs of the same test. Its format is as follows:

Columns 1-4	DATE
Columns 7-8	Current month coded numerically -- Ø1 = January, Ø2 = February, etc.
Columns 9-10	Day of the month coded numerically.
Columns 11-12	Year coded numerically.

B. Test Control Card

This card is used to initiate a test and to define the option indicators that control system operation.

Columns 1-6	Program Identification - The unique identification assigned to the JOVIAL-coded program.
Columns 7-10	TEST - Card identification indicating that this is a Test Control Card.
Columns 13-18	Test Number - A sequential number assigned to this test used as an aid in identification of the output.
Columns 19-24	COMPOOL Identification - The identification of the Master COMPOOL to be used for this test will be encoded here.
Columns 25-28	FULL or blank - Type of tracing mode requested. Automatic trace if left blank.
Columns 31-36	STAREV or blank - Indicates that card revisions to the data simulation input data are a part of the input deck.
Columns 37-42	POLREV or blank - Indicates that card revisions to the POL input data are a part of the input deck.

C. Test ID Card

This card is used to identify the input deck for a particular test and will be prestored if the test input is to be from tape. This card is

the first card of the test input deck.

Columns 1-6	Program Identification - This field should be the same as columns 1-6 of the Test Control Card. If these identification fields do not match, the test will be terminated.
Columns 7-12	TESTID - Card identification indicating that this is a TESTID Card.
Columns 13-60	Comments by the programmer - Any comments which the programmer desires as a part of the output heading, with the restriction that the programmer's name appear first. The comments will be printed exactly as written, thus necessitating the inclusion of all spacing.

D. JOVIAL Statement Card and Deck Format

Experience on the part of the individual JOVIAL programmer will probably be the best teacher in the matter of statement, card, and deck construction. At best, a list of general rules can be given here. The remarks pertaining to the deck construction are reasonable accurate, but remarks pertaining to statement and card format are probably incomplete.

1. Statement Format

Single terms, such as variables, constants, operators, etc., which consist of consecutive strings of letters or numbers, must not have any spaces between the first and last character.

Where two consecutive entities of the type mentioned above appear, they must be separated by at least one space (e.g., SWITCH ALPHA ...).

It is not necessary (although it might sometimes be desirable) to separate special characters (non-alphanumeric) from entities mentioned above and (under some circumstances) from each other.

Example: AB = 3* (BC-CD) / (EF +GH) $ In this statement, no spaces need appear.

Wherever one space is required, or permitted, any number of spaces may exist.

No implied multiplication is allowed. The expression (AB+BC) (CD + EF) is illegal. It should be written (AB +BC) * (CD + EF).

For Hollerith-type Constants, the number of characters within the parentheses must agree precisely with the number stated outside the parentheses. A blank is a legal Hollerith character and should be included in the count.

2. Card Format

Coding can begin at any column on the card.

Coding can stop at any column on the card, but must not extend beyond column 66.

More than one statement can appear on one card.

Example: The following "cards"

```
For I=NUMB$ BEGIN XY ($I$) =6$ AB ($I$) =Ø$
    OPQ ($I$) =I $ END
```

are equivalent to:

```
FOR I = NUMB$
BEGIN
XY ($I$) = 6$
AB ($I$) = Ø$
OPQ ($I$) = I$
END
```

A statement may continue for as many cards as necessary. The end of any one card, however, cannot be the middle of a single entity such as a variable name, etc.

3. Deck Format

The first card in any deck must be a card beginning with the declarator, START. Following this declarator, there can be no other JOVIAL statement. Any remarks, preferably program name, date, etc., should be added, since this card is logged on the printer when recognized.

All variable declarations except those for procedures must follow the START card and precede the first dynamic JOVIAL instruction.

All procedure variable declarations must precede the first procedure dynamic instructions.

The last card in the deck must have the operator TERM followed by the symbol "$" or a statement label followed by the symbol "$".

If the TERM is not followed by statement label, the first operating instruction is considered to be the first JOVIAL dynamic instruction. If it has a label, then the statement having this label will be considered to be the first dynamic instruction.

Other JOVIAL statements may precede the operator TERM on the same card, but none can follow.

NOTE

Here again it should be noted that a complete description of the JOVIAL language and its usage is contained in FN-LO-34-2.

E. Card Input to the Data Simulation Program

The main inputs to JSTRZ come from punched cards inserted via prestored tape or the card reader. Normally, system design specifies all inputs to be prestored, but in an unforeseen occurrence, such as failure of peripheral prestoring equipment or the like, the card reader is used to initiate the input.

There is the option, however, of having cards read in via the card reader when there are to be additions, deletions or changes to the data already prestored on tape. JSTRZ will be informed when to process card reader revisions by testing an indicator set by JTCPZ. Should a great number of additions, deletions, or changes have to be made, a new prestored tape must be generated, or in the case when there are many tests sequentially stacked on the prestored tape, the individual test in question may be bypassed at the programmer's discretion.

Card Format (See Figure 14)

Each input card contains information necessary to generate one item. This information is punched in fixed fields as follows:

Column 2 = "S" (JSTRZ Indicator) - An optional character used solely for identification purposes.

Columns 8-11 = Entry Value - A decimal number indicating the relative position in a table in which an item is to be stored. The entry value is right-justified, i.e., the least significant number must be in column 11. Leading zeros are not mandatory, but if used will not hinder the output. Where the entry value is zero there need not be any punches in the appropriate columns.

Columns 13-18 = Tag - Ranges up to six characters depending on the type.

ITEM - From four to six characters composed of, and identified by all letters.

TAGLESS ITEM - (Table Tag) - Four or five characters composed and identified by the following: three letters and one digit; three letters and two digits. (Tagless item inputs are presently not being considered for implementation into the Interpreter System Checkout. However, the capacity to process such a type has been installed in JSTRZ. Further clarification on its use and makeup is explained directly below under "Item Value.")

Columns 25-40 = Item Value - May be one of several forms, depending on the type of coding specified in the Baby COMPOOL. The form of the item value must be compatible with the type of coding for the particular item tag.

1. ST (Status) - Expressed as alphanumeric characters ranging from one to six characters. First character must be punched in column 25. If blanks are desired, an absence of a punch in the required columns is necessary. Machine format will show characters to be left-oriented with blanks in unused positions.
2. FP (Floating Point) - Expressed either as signed or unsigned decimal numbers. A decimal point is mandatory on the data card to distinguish between floating point and non-floating point numbers. It is possible to express an FP coded item value as a multiple of ten using the "$$" character as the base ten. For example:

   .272$$2 = .Ø272$$3 = 272Ø.$$-2 = 27.2

3. FI (Fixed Integer) - Expressed as a signed or unsigned decimal integer. No decimal point is mandatory; if one is used, the value will be converted erroneously. The "$$" character may be implemented with this type and if used must be followed by "AØ". If the latter notation is not used, the value will be converted erroneously. Example of proper format:

   272ØØ = 272$$2AØ = 272ØØØØ$$-2AØ

4. BH (B-coded Hollerith) - Expressed as alphanumeric characters ranging from one to six characters. Machine format will show

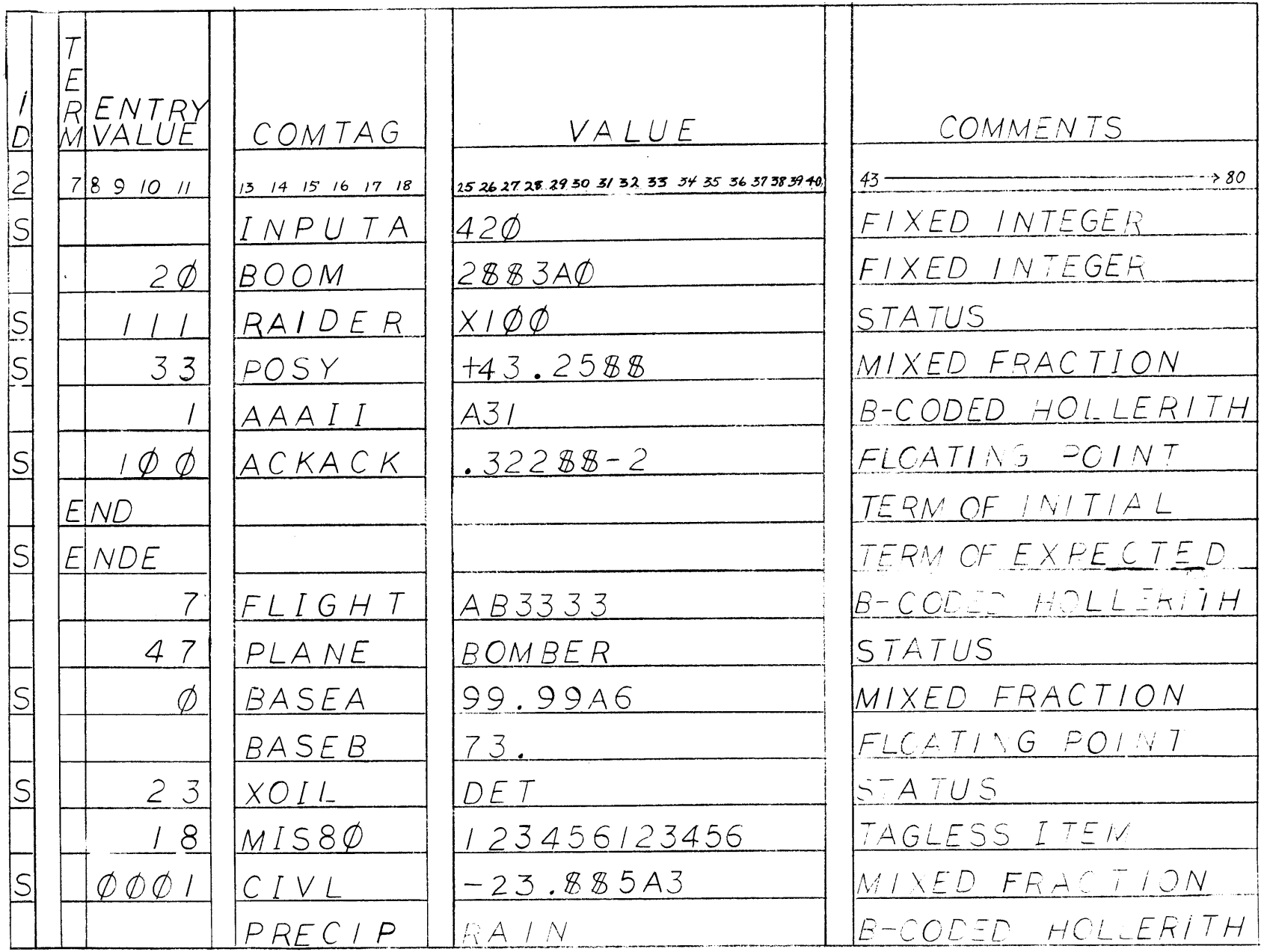
characters to be right-oriented with preceding zeros. First character must be punched in column 25.

5. MX (Mixed Fractions) - Expressed as either signed or unsigned decimal numbers. They may be expressed as powers of ten using the "$$" character. Here again, as in FP, the decimal point is mandatory. A distinguishing factor from FP is the positioning of the binary point. This is accomplished by indicating the number of bits to the left of bit position 35 where it is desired to place the binary point. This number must be the same as the value contained in point position in the Baby COMPOOL for this particular item. In the case where the numbers differ, the value in the COMPOOL is used. The significant character used to indicate the positioning of the binary point of an item is "A". If the "A" is omitted in a mixed fraction value, the converted result will be erroneous. In the case where the "AØ" is indicated, the binary point will be positioned at bit 35 and all fractional parts will be lost. For example:

272Ø.A3 = 27.2$$2A3 = 272ØØ.$$-2A3

6. AD (Indirect Addressing) - This model of JSTRZ is not equipped to process this type of item coding.

7. Tagless Item - In octal fraction form (twelve octal digits). To simulate such a type, the actual octal representation of what is to appear in the appropriate register of the table indicated by the entry value must be punched consecutively in columns 25 through 36 inclusive.



ID	TERM	ENTRY VALUE	COMTAG	VALUE	COMMENTS
2	7 8 9 10 11		13 14 15 16 17 18	25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40	43 → 80
S			INPUTA	42Ø	FIXED INTEGER
		2Ø	BOOM	2883AØ	FIXED INTEGER
S		111	RAIDER	X1ØØ	STATUS
S		33	POSY	+43.2588	MIXED FRACTION
		1	AAAII	A31	B-CODED HOLLERITH
S		1ØØ	ACKACK	.32288-2	FLOATING POINT
		END			TERM OF INITIAL
S		ENDE			TERM OF EXPECTED
		7	FLIGHT	AB3333	B-CODED HOLLERITH
		47	PLANE	BOMBER	STATUS
S		Ø	BASEA	99.99A6	MIXED FRACTION
			BASEB	73.	FLOATING POINT
S		23	XOIL	DET	STATUS
		18	MIS8Ø	123456123456	TAGLESS ITEM
S		ØØØ1	CIVL	-23.885A3	MIXED FRACTION
			PRECIP	RAIN	B-CODED HOLLERITH

Figure 14 - Sample Input Cards for JSTRZ

Order of Input

A pertinent procedural prerequisite to JSTRZ is the order of input. Due to the dual-processing of both the initial conditions and the expected results of a test, a differentiating element is essential in order not to confuse one meaningful factor from the other. Consequently the manner in which this input data is received by JSTRZ must be compatible with the predetermined format; otherwise, an erroneous test will result without the program being aware that it is in error.

FORMAT

Expected Results (optional)

"ENDE" Card (Columns 7-1Ø) (required with expected results)

Initial Conditions

"END" Card (Columns 7-9)

Each individual test does not necessarily have to include expected results. This is an optional feature used when it will benefit the final processing of the test results. When such an input is incorporated in a test, the data cards must be terminated by an "ENDE" card. Failure to include an "ENDE" card will result in the processing of both the expected results and the initial conditions as one meaningful input. Following the "ENDE" card are the initial conditions, terminated by an "END" card. Absence of an "END" card will cause a machine hang-up.

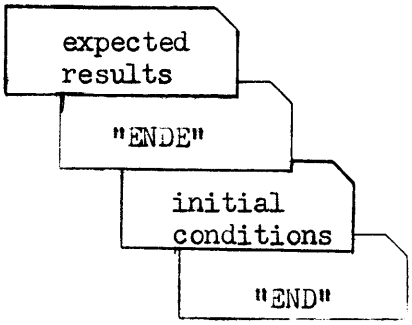
Card Reader Revisions

When there are additions, deletions, or changes to input data on the prestored tape, the card reader is utilized as a medium of receiving

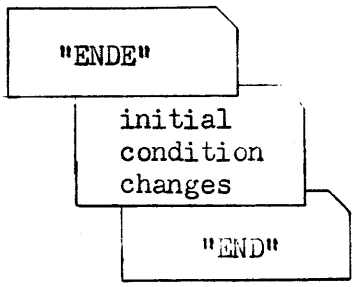
such input information. Here again, as on the prestored tape, the predetermined format is essential to be compatible with the test structure.

Possible cases when using the card reader:

1. INITIAL PRESTORED TAPE SETUP

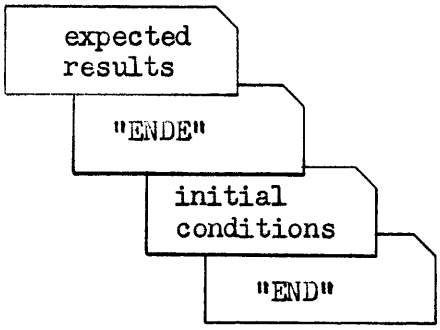


CARD READER SETUP

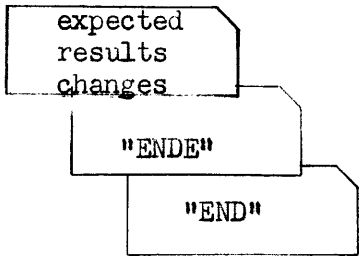


Here there are some expected results in the test and there are no changes to these, but there are some changes to the initial conditions. The first card in the card reader must be an "ENDE" card, followed by the changes to the initial conditions and terminated by an "END" card.

2. INITIAL PRESTORED TAPE SETUP

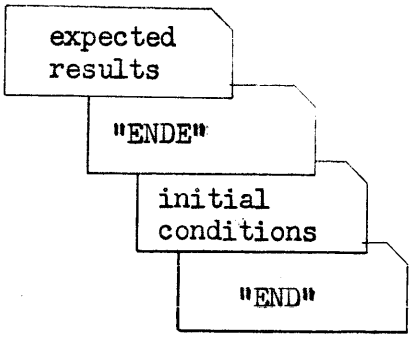


CARD READER SETUP



Here there are some expected results in the test and there are changes to these, but there are no changes to the initial conditions. The first set of cards in the card reader must be the changes to the expected results, terminated by an "ENDE" card and followed by a single "END" card.

3. INITIAL PRESTORED TAPE SETUP



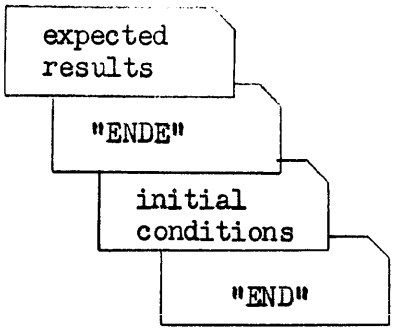
CARD READER SETUP

not

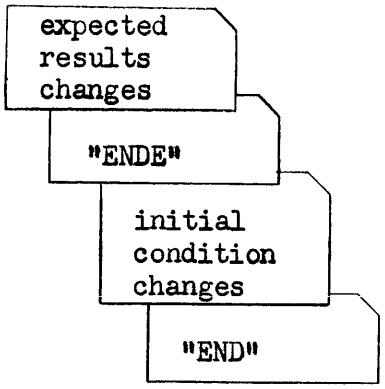
used

Here there are expected results in the test, but there are changes neither to these nor to the initial conditions. In this case there need not be any cards in the card reader.

4. INITIAL PRESTORED TAPE SETUP

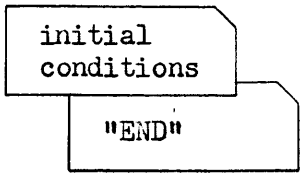


CARD READER SETUP



Here there are some expected results in the test and there are changes to these and also changes to the initial conditions. The first set of cards in the card reader must be the expected-results changes, terminated by an "ENDE" card, followed by the set of initial condition changes, terminated by an "END" card.

5. INITIAL PRESTORED TAPE SETUP



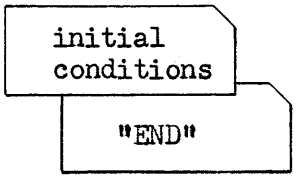
CARD READER SETUP

not

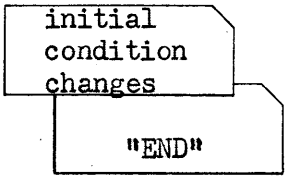
used

Here there are no expected results in the test and there are no changes to the initial conditions. In this case there need not be any cards in the card reader.

6. INITIAL PRESTORED TAPE SETUP



CARD READER SETUP



Here there are no expected results in the test, but there are changes to the initial conditions. The first set of cards in the card reader must be the changes to the initial conditions, terminated by an "END" card.

Note 1: Where no expected results were utilized in the initial composition of the test, it is impossible to insert such data via the card reader. The only way to adjust this addition to the test is to prestore all the input data over again

Note 2: To delete a simulated value, it is necessary to include on the data input card in the card reader the entry value, the COMTAG, and a zero in column 25.

Note 3: When there are to be revisions to prestored tape data, JTCPZ is instructed to set Sense Bit 8 from information inserted on the Test Control Card. This latter case is concerned

with columns 49-54 of the card. If there are revisions, STAREV is punched in these columns. If there are no revisions, these columns are left blank.

F. End of Test Card

The End of Test Card is the last card of the input deck and signifies to the Test Control Program that there is no further input associated with this test. This card is always placed in the card reader.

Columns 1-6	ENDTST

G. End of Logical Tape Card

The End of Logical Tape Card is required to be the last card in the input deck if the input deck is being prestored. This card is required to signify that this is the logical end of the prestored input tape and is always coded as follows:

Columns 1-6	YYYYYY
Columns 7-12	777777

Deck Structure

The sequence of operation of the subprograms that interpretively operate a JOVIAL Program require that the input data be arranged in a definite order. A test run will generally consist of several individual jobs. The control cards are always entered through the card reader, while the data inputs (JOVIAL Deck and Data Simulation Deck) may be entered from the card reader or prestored tape. Data input entry mode may not be mixed in a test run. Sense Switch One will be set to On (depressed), when the data input is from the card reader.

The order of input, if from the reader, will be as follows:

1. Date Card
2. Complete Data Input Decks for each job to be run, the cards being in the following sequence:
   a. Test Control Card
   b. TESTID Card
   c. JOVIAL Program Deck
   d. Data Simulation Deck
   e. ENDTST Card
3. Complete decks as (2) above for each job.

.

.

.

The order of input, if prestored, will be as follows:

1. In the card reader
   a. Date Card
   b. Sets of cards for each job to be run, the cards being in the following sequence.
      1) Test Control Card
      2) JOVIAL revisions (if applicable)
      3) Data Simulation revisions (if applicable)
      4) ENDTST Card
   c. Sets of cards as (b) above for each job.

   .
   .
   .

2. To be prestored
   a. Decks of Data Input Cards for each job to be run, the cards being in the following sequence:
      1) TESTID Card
      2) JOVIAL Program Deck
      3) Data Simulation Deck
      4) Blank Card
   b. Decks as (a) above for each job.

   .
   .
   .

   c. End of logical tape card.

Variations and rules for structuring the JOVIAL and Data Simulation Input Decks are discussed in more detail in the preceding section of the Manual.

OUTPUT

Message Printouts

Message printouts are used by the JOVIAL Interpreter System as a means of communication with the system user. The messages are divided into two categories. The first category contains those messages that are required for the computer operator as an indication of hardware-type errors and communication of the name of the system sub-program currently operating.

The second category contains all of the messages that are printed off-line as an integral part of the system's output. This category also includes the messages from the first category to facilitate understanding by the system user of any machine or hardware difficulties that may have been encountered during the running of a test.

A complete listing of the message printouts that may result from system operation follows. A description of the reason for the generation of each message is also included. The sequence of the following list approximates the sequence in which the messages might be output from the system.

A. The following messages may be generated during the operation of the Test Control Program prior to the initiation of a given test.

1. NO DATE

   This indicates the omission of a date card in the card reader, and is printed from communication registers DATE and DATE+1.

2. ERROR IN CARD, IMAGE BELOW

   If a JTCPZ input card was not of the proper format, this message is

printed out, followed by the actual card image.

3. TEST CONTROL CARD MISSING

Since a test cannot be initiated until a matching of program identifications occurs, this message denotes the source of error as opposed to an actual identification mismatch.

4. TEST DECK ID DOES NOT MATCH CONTROL CARD ID

When a mismatch of program identifications between these two cards occurs, this message is written.

5. NO TEST ON TAPE CORRESPONDING TO CONTROL CARD ID

If the test (JOVIAL-coded program) is on prestored tape, and the SEARCH subroutine of JTCPZ cannot locate this program, this message is written.

6. NO BACK MACRO
ILLEGAL BACK OPERATION

One of these two messages is generated within the SEARCH subroutine whenever it cannot position the tape to the program identification record after having obtained a matching of program identifications. All output messages denoted above will appear both on- and off-line, to facilitate tape handling by the computer operators, and to give a "permanent" message on the DLO tape as an aid in future system operation.

7. LOGICAL END-OF-TAPE
TAPE UNIT UNASSIGNED
NO READ MACRO
THREE UNSUCCESSFUL TRIES
ILLEGAL EOF
TAPE NOT ASSIGNED ON WREOF
CANNOT WRITE EOF ON DLO

The above messages will be generated by JTCPZ whenever the

input-output routines are rendered non-operative. The appropriate error message will appear under that program currently operating, thus localizing to some degree the source of error.

8. UNABLE TO READ IN ABOVE PROGRAM

This message is generated when the program in question cannot be located on the tape by the SEARCH subroutine.

B. The following messages may be generated during the operation of the First Pass of the Interpreter Program.

INTERPRETER FIRST PASS

Generated by the Test Control Program and printed for identification of program operation.

UNABLE TO LOCATE MASTER COMPOOL

Generated by the Test Control Program when the SEARCH routine cannot locate the Master Compool on the Master Tape.

Two types of output are printed by JALLZ. One is the START card, which is logged in its entirety as soon as it is read in. The other output available to the user consists of error messages. The format of the message is usually a function of the particular section of JALLZ from which is emanated. Except for those messages which are explicit and complete in their printed form, messages are given a unique reference number. The following discussion will include a list of these messages and their corresponding reference numbers (where applicable). Where it seems desirable, a fairly complete description of how the error is detected will be given.

In almost all cases, the existence of one or more of these messages will indicate that the program will not be tested any further. The only recourse

the user has is to repair his errors and try again.

Error messages will be described under the particular section of the program which produces them. A description of these sections is included in "Functions of JALLZ." (FN-LO-201)

Errors of Section A

The errors of this section will be printed out in the following format:

"MESSAGE STATEMENT LABEL +XXXXXX REFERENCE XXX"

The messages and their respective reference numbers and explanations are in the following list.

"ILLEGAL FIRST WORD"

21. First significant word (1) is non-alphanumeric first significant word (1) must be alphanumeric.
22. First significant word (1) is illegal JOVIAL terminology.
23. First significant word (1) has an illegal character following it.
24. First significant word (1) forms an illegal combination with word following it.

(1) First significant word is not necessarily the first actual word of the statement. For example, a statement label preceding an "IF" statement places the first word, "IF", in a second word position.

"EXCESSIVE WORD LENGTH"

25. Word is more than six consecutive alphanumeric characters.

"ILLEGALLY FORMED STATEMENT"

26. Illegal use of decimal point in this statement. Decimal point is used only in constants and following leading statement label.

27. Illegal character following label. Must have dollar sign, indicating end of statement, or bracket. (2)

28. Unrecognizable character for JOVIAL terminology found.

29. Should have set (=) at this point, but was not found.

30. Found comma, yet we are not in a "FOR" statement nor can comma be interpreted as parameter separator at this point.

31. Illegal to have equal sign at this point.

32. Statement has excess of words following legitimate portion of statement.

33. Illegal for status value type of operator to be first significant word of statement.

34. Illegal character in status value type of operator. Should be of form, V(STATUS).

"ILLEGAL CONSTANT"

36. Have accuracy factor twice in same constant.

(2) Check hardware language for bracket.

37. Illegal letter found in constant. Only letters A, accuracy, H, Ho Hollerity, and E, exponent, are allowed.

38. Letter H found, but is illegally poisitioned in constant.

39. Hollerith constant more than six characters long.

40. Right parenthesis on Hollerith constant is either missing or is illegally poisitioned.

41. Constant should be integer in this case, but is not an integer.

42. Alphanumeric word and constant form an illegal combination in this statement.

"ILLEGAL VARIABLE"

43. Variable is an illegal length or has an illegal character contained within it. All variables must be from two to six alphanumeric characters, beginning with a letter.

44. Variable is followed by an illegal character. All variables in this case must be followed by a right parenthesis or, if a subscript follows, by a bracket. (2)

45. Illegal character follows this variable. Variable starting statement must have set (=) following it.

46. Illegal subscripted variable poisition. Subscripted variable is used instead of variable. However, program will be compiled correctly.

"ILLEGAL SUBSCRIPT"

47. Subscript is illegal. Subscript must be either a single letter or an integer.

48. Subscript is followed by an illegal character. All subscripts in this case must be followed by a bracket (2) or, if increment or decrement follows, by a plus or minus sign.

49. Subscript is incremented or decremented by a non-integer. Only integer is legal in this case.

50. Subscript is illegal. Only can have single letter in this case.

"ILLEGAL ENTRY OPERATOR"

51. Entry is followed by an illegal character. Only set (=) can be used following entry.

52. Entry is set illegally. Entry may be set only to another entry or to zero.

"ILLEGAL BIT/BYTE OPERATOR"

53. Bit/Byte subscripted position is followed by an illegal character. This initial position must be followed by a bracket (2) or, if end position follows, by a comma.

54. Bit/Byte final bracket is followed by an illegal character. Left parenthesis and variable are expected, but left parenthesis is missing.

55. Illegal Bit/Byte subscripting. Subscript of Bit/Byte word must be single letter or integer.

"ILLEGAL STATEMENT LABEL"

56. Illogical to label this type of statement label will be deleted. However, program will be compiled correctly unless transfer is made to this label.

57. Label is illegal. Label must be two to six alphanumeric characters, beginning with a letter.

"ILLEGAL BEGIN OPERATOR"

58. Illogical to precede this statement with a "begin" operator. However, program will be compiled correctly as long as there is a compensating "END" operator following.

59. Illegal to precede this statement with a "BEGIN" operator. "BEGIN" will be deleted and compiling will continue.

"ILLEGAL SWITCH"

60. Label of switch is less than two alphanumeric characters. Label must be two to six alphanumeric characters beginning with a letter.

61. Illegal character following switch label. Must have set (=) for numeric switch and left parenthesis for status switch following switch label.

62. Illegal character following status item. Should have set (=) at this point.

63. Illegal character following set (=). Should have left parenthesis at this point.

64. Illegal for status value to be special character. Must have number or alphanumeric word as status. Can use Hollerith to indicate special character.

65. Separator between status value and switch branch label is illegal. Should have equal sign (=) in hardware language.

66. Illegal switch branch label. Label must be two to six alphanumeric characters beginning with a letter.

67. Illegal character following switch branch label. Must have comma or right parenthesis at this point.

"ILLEGALLY SIGNED EXPRESSION"

68. Illegal to precede left parenthesis with arithmetic sign. Sign will be ignored and program will continue to be compiled.

69. Illogical to have a series of arithmetic signs. Correct arithmetic sign will be computed and program will continue to be compiled.

"ILLEGALLY FORMED STATEMENT"

70. Illegal word or character following alphanumeric word.

71. Illegal word or character following arithmetic sign.

72. Illegal word or character following left parenthesis.

73. Illegal word or character following right parenthesis.

74. Illegal word or character following relational operator.

75. Illegal word or character following logical operators: AND, OR

76. Illegal word or character following dollar sign.

77. Illegal word or character following up arrow.

78. Illegal word or character following down arrow.

79. Illegal word or character following value type item.

80. Illegal word or character following logical operator, NOT.

81. Illegal word or character following constant.

"ILLEGAL FOR STATEMENT"

82. Illegal character following subscript. Should have set (=) following.

83. "ALL" type of "FOR" statement recognized. Should have item enclosed in parenthesis following the word "ALL", but was not found.

84. More than two commas found in "FOR" statement.

87. Comma not found following "B" factor of "FOR" statement.

"ILLEGAL PROCEDURE STATEMENT"

88. Procedure label is illegal. Label must be from two to six alphanumeric characters beginning with a letter.

"ILLEGAL VARIABLE DEFINITION"

89. Illegal position for item/table definition statement. All definitions must be at beginning of program or, if procedure variable definitions, at beginning of procedure.

"TABLE LENGTH EXCEEDED"

90. Constant table exceeded. Storage of succeeding constants will be stored at start of constant table once again.

91. Subscript table exceeded. Storage of succeeding subscripts will be assigned duplicate relative positions at start of subscript table once again.

92. Bit/Byte table exceeded. Program unaffected.

93. CAT table exceeded. This statement rejected.

Errors of Section A1

The standard format for an error printout of Section A1 is the following:

"VARIABLE DEFINITION ERROR N"

Where N is the reference number. Listed below are the reference numbers, together with the associated error types.

95. Undefined Variable -

A referenced variable has not been previously defined by the program, and is not listed in the COMPOOL.

96. Duplicate Definition -

The variable defined by the declaration has been defined previously by the program.

97. Variable Definition Limit Exceeded -

The total number of variables defined by the program and COMPOOL defined variable referenced by the program exceeds system capacity.

98. Status-item Value Limit Exceeded -

The total number of status values for all program defined status items and COMPOOL defined status items referenced by the program exceeds system capacity.

99. Identical Table Limit Exceeded -

The number of Identical Tables defined by the main program and any one procedure exceeds system capacity.

100. Item or Table Declaration does not, end with $

101. Table Name Format Error

102. Table Type Format Error

103. Error In Table No. of Entries

104. Identical Table name Format error (No. characters < 3)

105. Illegal Identical Table -

No table has been previously defined (or is listed in the COMPOOL) to which this table can be made identical.

106. No End to Table -

The end bracket which terminates definition of items with a table is missing.

107. Item Name Format Error -

108. Item Type Format Error -

109. Undefined COMPOOL Item -

A COMPOOL defined item referenced by this table is now undefined because its table has been redefined by the program as other than a table.

110. Status Item Has No Values -

111. Status Item Has Too Many Status Values (> 65) -

112. Error in No. of Bits or No. of Bits to Right of Point -

113. Signed/Unsigned Error -

114. Parameter Item Value Error -

115. Illegal Item Value in Table of Constants -

116. Too Many Constants in Table of Constants

117. No END to Constant List -

The END bracket which terminates a list of constants is missing.

118. Procedure Heading Format Error -

119. Procedure Dummy Parameter is Undefined -

120. Procedure Dummy Parameter Illegally Defined -

As parameter item, status item, table, or item within a table.

Errors of Section B

The error printouts logged by this section are considered to be descriptive enough so that further description of them in this document via numbers appears unnecessary. Following is a list of these error messages.

All error messages are composed of two printed lines. The first line is, "Statement Label XXXXX rejected by ILT analysis." The second line is any one of the following fourteen specific messages:

"THE CARD ANALYSIS TABLE IS EMPTY"

"THIS SUBSCRIPT IS CURRENTLY IN USE"

"THE PROGRAM HAS THE FOLLOWING NUMBER OF EXTRA BEGINS"

"CANNOT TEST AN INACTIVE SUBSCRIPT"

"SWITCH DECLARATION IS INCORRECT"

"PROCEDURE DECLARATION WITHIN A COMPOUND"

"THIS END HAS NO BEGIN FOR IT"

"STATEMENT HAS AN INCORRECT OPERATOR"

"SUBSCRIPT NOT SET BY A FOR STATEMENT"

"INTERMEDIATE LANGUAGE TABLE EXCEEDED"

"STATEMENT LABEL TABLE EXCEEDED"

"ACTIVE FOR STATEMENT TABLE EXCEEDED"

"SWITCH TABLE EXCEEDED"

"END TABLE EXCEEDED"

Errors of Sections B1 and B2

If any errors are detected by either Sections B1 or B2, a message is printed for the cause, another for the reject (indicating the routine) and indication is made to the control operation of the Interpreter. If an error occurs in Section B1 the routine rejects the entry and continues processing the statement in the Card Analysis Table (CAT). (The statement is rejected when the processing is complete.) In section B2, the statement is rejected immediately when an error is detected.

The reject message is:

STATEMENT ØØØØØØ + ØØØØ REJECTED BY XXXXXX ANALYSIS;

Where XXXXXX is "LEVEL" for section B1 and "LOG" for section B2.

The Hollerith Statement label follows STATEMENT. A decrement value follows + (since not all statements have programmer defined labels).

Section B1

The level anlysis routine expects the CAT Table to have a basic sequence of alternating terms with operator (A+B-C, etc.) with separators appropriately placed.

The terms are recognized by the Class Value ($6 < \text{CLASS} \leq 16$).

The operators and separators are determined first on Class Value ($1 \leq \text{CLASS} \leq 5$) and secondly on form value. The following list contains the legal forms:

form value	description
1 - 6	relational operators
7 - 10	arithmetic operators
11	OF operator
12	AND operator
13	OR operator
16	left parenthesis - separator
17	right parenthesis - separator
18	comma - separator
22	set (:=) operator
23	=: separator
25	up exponent - operator (separator)
26	down exponent - separator
58	comma - separator (special)

The separators are reflected in the counter which determines the level value assignment for the operators.

In most messages, the relative address in the CAT Table (RELC) is given for reference.

Message:

INCOMPLETE STATEMENT IN CAT RELC ØØØØ

only a partial statement in CAT (ends with an operator).

Message:

LEVEL COUNTER VALUE INCORRECT ØØØØ

Reason:

Illegal use of separators. Check to see for appropriate pairing of parenthesis, etc. Check is made at the end of the statement.

Message:

WRONG CLASS - FORM IN CAT RELC ØØØØ

Illegal form - class for entry. If terms and/or operators are out of sequence, the form or class is considered illegal.

Message:

ØØØØØØ TABLE LIMIT EXCEEDED RELCØØØØ

As an entry is added to each internal table, the length is checked first. If the limit is exceeded, the CAT entry is rejected and the table name is inserted in the message (NOT, LAT1, LAT2.) Obviously, all subsequent CAT entries calling for the same table will be reject.

Possible Error Messages - Section B2

This section is given an initial entrance parameter into the Statement Label Table (SLT). This is the label of the first entry in the ILT after an entry containing a relational operator. The routine will then process all such entries, inserting the correct branch points and allowing for the connectives (AND, OR) and negation (NOT.) A certain amount of legality checking is done.

Message:

ILT TABLE INCORRECT RELI ØØØØ ALE1

Reason:

Only a partial statement in ILT or the referencing from the SLT is incorrect (not after relational operator, etc.)

Message:

LAT2 TABLE INCORRECT REL2 ØØØØ ALE2

Reason:

Incorrect operator value (not AND or OR, i.e., 12 or 13)

Message:

SLT TABLE INCORRECT RELS ØØØØ ALE3

Reason:

Insufficient generated (or programmer-defined) labels in LST for relational operator entries in ILT. The next two conditions and/or statements should be labeled.

Message:

NEGATION ERROR - EXTENT = RELI ØØØØ ØØØØ ALE4

Reason:

Lack of correlation between NOT table and ILT. The extent of range of the NOT operator causing the difficulty is given. Within that range there should be two and only two extension (outside of range) branch points.

Errors of Section C

If any error occurs in this section, the operation of the Interpreter will be interrupted, causing the test program to be rejected. The cause of the interrupt will have been previously indicated. In order to catch as many errors as possible, this section operates in the following manner:

1. If errors occur in checking the Statement Label Table (SLT) for doubly defined labels, the SLT for doubly defined labels, the SLT processing (*WSRS) and the first pass (relative) in processing the Intermediate Language Table (ILT) will be completed before interrupt. Please note: The doubly defined labels will cause illegal relative addressing in ILT (pass 1).
2. If errors occur during the first pass at ILT (WILT 1), Section C will continue to the end of the pass before interrupt.

* Interpreter reference symbols

3. If illegal information is discovered in processing the Variable Table (VAT) for storage allocation (WVTA), VAT is completely processed before interrupt.

   If the amount of core required for the data tables (as specified in VAT) exceeds the available storage, the section will interrupt as in 5. below.

4. Calculation (end of WVTA) is made with respect to the input data. If the result is greater than the storage alloted in the Interpreter System for data reduction (DAISY), Section C is interrupted. The following formula is used for the calculation (expressed in JOVIAL):

```
IF 3*(NENT(VAT) + TABREG) + NENT (STAT)
   +2 + HPARAM LQ DSYLMT$ GOTO CONT$
   GOTO INTER$
```

   WHERE TABREG = length block of data tables

   HPARAM = length of parameter items block

   DSYLMT = limit defined by Interpreter System

   Note: 2 is added for control words needed for VAT and STAT.

   CONT indicates continue

   INTER indicates interrupt

5. If the allocation for the internal tables exceeds available storage (WSTOR), the Interpreter interrupts.

6. If errors occur during the second pass at ILT (absolute - WILT2), the Interpreter will continue to the end of the pass before interrupt.

POSSIBLE ERROR MESSAGES

The following are the error messages from Section C. The referencing on the right refers to the symbolic identification used.

Message:

PROGRAM INTERRUPTED IN JILL SECTION, CHECK
INTERPRETER, ERROR LIST

Reason:

a. The sense indicators are pre-set to interrupt after some routine (see explanation of SI control-WJIC.)

b. If errors are detected during a given processing routine, the sense indicators are dynamically set to interrupt (see preceding explanation of error recognition.)

Message:

ILLEGAL PROGRAM OPER. XR1 ØØØØØ XR2 ØØØØØ XR4 ØØØØØ

This message should not occur in normal program testing. It usually indicates that the Interpreter deck is out of order or instructions are missing due to system or machine failure. It could also indicate possible program error.

XR4 should give the complement of the absolute address of the location of the error test.

Message:

```
DOUBLY DEFINED STATEMENT LABELS

LABEL  PNSL RELI   LABEL  PNSL RELI

ØØØØØØ  ØØØ  ØØØØ  ØØØØØØ  ØØØ  ØØØØ
   |                              |
   ↓                              ↓
END DOUBLY DEFINED LABELS
```

Heading and end messages are printed out if there is at least one pair of labels equal. Equal labels are always printed in pairs. If three or more labels are equal, there will be a certain amount of repeats (i.e., if 1, 2, ... n equal labels, label numbers 2, 3, ... n - 1 will be repeated.) Conditions for equality (specified in JOVIAL):

```
IF LAB1 EQ LAB2 AND PNSL1 EQ PNSL2$
GOTO PRINT$
GOTO CONT$
```

Where LAB1 and LAB2 are the Hollerith labels, and PNSL1 and PNSL2 are the procedure numbers associated with the statement labels (integers.)

Messages for Processing ILT

Messages for indicating errors in ILT processing can be printed out in either pass. The numbers given at the right of the explanation indicate which pass (1,2). To cut down on the number of stored messages, the same message was used in similar cases with a specific reason number. All messages give reference relative information.

Message:

UNDEF. LABEL: ØØØØØØ PNSL ØØØØ RELI ØØØØ

If a given ILT entry contains a Hollerith label or reference to Hollerith labels to be converted to a relative address (RELI), this label is used to search the SLT table together with the current procedure number (WPNUI.)

Conditions for a label to be undefined (expressed in JOVIAL):

```
FOR I = ALL (SLT) $
BEGIN  IF  LAB1 EQ LAB2 [I] AND
WPNUI  EQ  PNSL [I] $ GOTO CONT$
END PRINT. ⟨ ,⟨ .....
```

There LAB1 is the label to be defined and WPNUI is the current procedure number.

If the above message is printed out for a procedure call label, the following message is printed, also:

Message:

SINCE PROC. CALL LABEL, I-O PARAMETERS
NOT PROCESSED.

The Input-Output parameter entries for that procedure call should then printout as illegal operators (see below - WE11.)

Message:

OPER VALUE ØØ INCORRECT IN ILT
RELI ØØØØ

Only certain operators are expected to appear in ILT at the time of the JILL processing. If an entry is rejected, subsequent errors can result, or, this reject could be caused by a previous error.

The following operator values are legal for a given pass:

Oper.	Pass 1	Pass 2
BAB	0	0
Rel. operators	1 - 6	1 - 6
Arith. operators	7 - 10	7 - 10
STOP	30	30
GOTO	31	31
RETURN	33	
TERM	34	34
END	20	20
SET	22	22
↑	25	25
Procedure Declaration	46	46
Item Swt. dec.	47	47
Subscript Switch declaration	48	48
Procedure call	50	50
Close declaration	57	57
Procedure declaration (1 output)	59	59

Message: WEI2

"ENTRY DATA WRONG IN ILT REAS. ØØØ"

RELI ØØØØ.

Reason Number	Explanation
1	Left term not a variable for status value (right term)
2	Branch points not set for relational operators
3	Form value not correct for procedure call operator
4	ILT entries out of sequence. No close declaration or procedure declaration before RETURN.
5	ILT entries out of sequence. No close declaration or procedure declaration before END.
6	No label in GOTO entry
8	Store class value incorrect.
9	Illegal term information for right or left term (absolute addressing.) Check table size on relative addresses.
10	Branch points (absolute addressing) incorrect. Check table size (ILT.)
11	No RELT for end for procedure declaration or close declaration
12	Illegal class for variable in END entry (assuming one output procedure end.)

Reason Number	Explanation
13	No input parameter after special one output procedure call
14	Interpreter limits exceeded. Too many nested procedure and close declarations for table controlling (CTRL) the associated ENDs.

Message:

ØØØØ TABLE INCOMPLETE

REAS.ØØØ RELI ØØØØ

Reason Number	Insert	Explanation
1	VAT	Variable of left term not correct for status item (information in variable table).
2	STAT	Status illegal (right term) for variable designated in left term.
3	SLT	SLT does not contain RELI of next two conditions and/or next two statements
4	VAT	Variables do not correspond to I-O parameters given in ILT for procedure call
5	ILT	No input parameter for special one output procedure call.

Message:

ILLEG. STATUS ØØØØØØ FOR SWT

RELI ØØØØ

If a status is used for an Item Switch declaration, which is not in the Status Table (STAT) for the variable given, this message is printed and program continues operating on the rest of the statuses for that switch.

POSSIBLE ERROR MESSAGES - VARIABLE

TABLE & STORAGE ALLOCATION

In processing the Variable Table (VAT) for allocating storage and inserting absolute addresses, the following messages may be printed. If an error is indicated, the entry is rejected. Note: The processing is done in two passes: 1st pass, tables and like tables; 2nd pass, items (parameter items are skipped on both passes - they are processed later.)

Reason Number	Explanation
1	VTYPE wrong - 1st pass
	The following are legal:
	0 - undefined - skipped
	1 - Item - skipped
	2 - Table - processed
	3 - Parameter item - skipped
	4 - Like tables - processed (and changed to 2)
2	VTYPE wrong - 2nd pass
	The following are legal:
	0 - undefined - skipped
	1 - Item - processed
	2 - Table - skipped
	3 - Parameter items - skipped

Checks are made to determine if the program input data will fit into available storage. The following message is printed and the Interpreter interrupted, if the data exceeds the capacity.

Message:

INTERPRETER STORAGE LIMIT

EXCEEDED ØØØØØØ

Insert	Explanation
DATA	After the 1st pass in processing the VAT table (allocating tables), the data block (see TABREG) is too large for high core.
INT.	After storage is allocated for the internal table, these tables plus the data block is too large for high core.
DAISY	After 1st pass at VAT, the data block information is used together with VAT and STAT to check the limit of the Interpreter System (DAISY - see formula described above under error recognition.)

Test Terminated by Error in POL

If during operation of JALLZ, an error is detected, control is returned to the Test Control Program. The above message is printed prior to termination of the test.

C. The following messages may be generated during the operation of the Assemble Baby Compool Program.

1. ASSEMBLE BABY COMPOOL

   Generated by the Test Control Program and printed for identification of program operation.

D. The following messages may be generated during the operation of the Data Simulation Program

1. DATA SIMULATION PROGRAM

   Generated by the Test Control Program and printed for identification of program operation.

2. THE FOLLOWING OFF-LINE LISTING INCLUDES THE ENTRY VALUE, THE COMTAG AND THE REASON FOR ITS REJECTION OR THE REASON FOR SOME CHANGE IN THE ITEM VALUE.

   Should JSTRZ legality checks detect an error in any data input card, the above comment is logged out.

3. (Card Image) ERRONEOUS PUNCH IN EITHER COLUMNS 3, 4, 5 or 6.

   (Card Image) ERRONEOUS PUNCH IN COLUMN 12.

   (Card Image) ERRONEOUS PUNCH IN ONE OR MORE COLUMNS 19-24.

   (Card Image) ERRONEOUS PUNCH IN COLUMN 41.

   An initial check on illegal punches in columns not used to indicate significant input produces the printouts listed above.

4. (Card Image) ERRONEOUS PUNCH IN COLUMN 7 (NOT E OR BLANK.)

Column 7 is used only when an "ENDE" or an "END" card is indicated. Any other punch causes the card in question to be deleted from the test and the printout listed above occurs.

5. (Card Image) ERRONEOUS PUNCH IN EITHER COLUMNS 8, 9, or 10.

(Card Image) ERRONEOUS PUNCH IN EITHER COLUMNS 8 or 9.

Where an "E" is found in column, a check is made to determine if the card is an "ENDE" or "END" card. If not, the above printout is shown.

6. (Card Image) COMTAG DOES NOT EXIST IN BABY COMPOOL.

An illegal communication tag, found by COMPOOL look-up, not to be included in the Baby COMPOOL.

7. (Card Image) BLANK IN COLUMN 25.

The significant input (item value) must begin in column 25. If not, the card is deleted and the above printout occurs.

8. (Card Image) ENTRY VALUE EXCEEDS MAXIMUM LIMITS.

A check is made of the number of entries contained in the Baby COMPOOL for this item, along with the entry value on the card. Where the entry valus is larger than limits provide, the above printout occurs.

9. (Card Image) ERRONEOUS CONVERSION OF ENTRY VALUE.

When the subroutine used to convert the decimal entry value to binary discovers an error internal to itself or in the entry value, the above printout occurs.

10. (Card Image) STATUS ITEM DOES NOT EXIST IN TABLE B OF BABY COMPOOL

When the status symbology, as indicated in the item value, is not

contained in the block of statuses in Table B of the Baby COMPOOL, the above printout occurs.

11. (Card Image) ITEM VALUE INCOMPATIBLE WITH FORMAT

Where the simulated value is different than the required format, the above printout occurs.

12. (Card Image) TAGLESS ITEM VALUE IS NOT OCTAL.

When the item value for a tagless item contains anything other than 12 octal digits, the above printout occurs.

13. (Card Image) JSTRZ CANNOT PROCESS INDIRECT ADDRESSING ITEM CODING.

Since JSTRZ is not equipped to process indirect addressing item coding, it will log out the above printout whenever such an item is contained in the data input deck.

14. (Card Image) ONLY ONE $ SIGN, EXPONENT SYMBOL IS $$.

A check is made to be sure that two dollar signs are used to indicate an exponent of the base 1Ø. When such is not the case, the above printout occurs.

15. (Card Image) POINT POSITION VALUE ON CARD IN ERROR, VALUE IN BABY COMPOOL USED TO POSITION MIXED FRACTION.

A check is made with point position as contained in the Baby COMPOOL. along with the point position as contained on the input card. When the two values differ, the one in the COMPOOL is used and the above printout occurs. In this case, the card is not deleted from the test.

16. (Card Image) ERRONEOUS CONVERSION OF ITEM VALUE.

Where the subroutine used to convert the data information discovers that the value is incompatible with storage requirements, the above printout occurs. In the case of a MX, FP, or FI, if the item value contains a character other than a digit, a comma, a decimal point, a dollar sign, an "A", a plus sign, or a minus sign, the card will be logged out as in error.

17. FAULTY USE OR ERRONEOUS TRANSFER OF INFORMATION ON THE ADDITIONAL INITIAL TABLE DATA TAPE, RETURN TO CONTROL.

FAULTY USE OR ERRONEOUS TRANSFER OF INFORMATION ON THE PRESTORED INPUT TAPE.

FAULTY USE OR ERRONEOUS TRANSFER OF INFORMATION ON THE EXPECTED RESULTS TAPE.

FAULTY USE OR ERRONEOUS TRANSFER OF INFORMATION ON THE INITIAL CONDITIONS OUTPUT TAPE.

Some printouts occur when there is faulty transmission of tape information. These will cause JSTRZ to relinquish control immediately to JTCPZ.

18. JSTRZ HAS FINISHED PROCESSING THE TABLE AND ITEM ENVIRONMENT FOR THIS TEST.

Upon completion of operation, JSTRZ will print the above comment.

E. The following messages may be generated during the operation of the Second Pass of the Interpreter Program.

1. INTERPRETER SECOND PASS

   Generated by the Test Control Program and printed for identification of program operation.

2. STOP _____ ______

   If the computer halts during the execution of a JOVIAL Program, the above will be printed. The first number is the ILT entry containing the HALT, the second is the octal location of the HALT instruction. After printing of the above, the next job is initiated.

3. INTERPRETER LOOPING EXIT TO TCP

   The above is printed after the operator depresses Sense Switch 5 upon noting that the Second Pass of the Interpreter is looping. Control is returned to JTCPZ which calls in JDSYZ to process the data available.

4. The printouts from JOLLZ consist of error messages only. The basic message form is:

   a. Identification EF ##.

   b. Relative location of the error in ILT.

   c. If the error is a loating point overflow or underflow, its location in the Interpreter plus one is given in octal, otherwise the area is blank.

   EXAMPLE: EF ## ##### #####

   IDENT ILT LOC. OV. UN. LOC.

5. Messages

   a. EF1: Floating add, subtract, multiply; either an overflow or underflow occurred in the MQ. Control is returned to the error location plus one.

b. EF2: Division by zero, MQ is destroyed and control is returned to the error location plus one.

c. EF3: Floating add, subtract, multiply; either an overflow or underflow occurred in the MQ or AC. Control is returned to the error location plus one.

d. EF4: Negative argument to logarithm subroutine. Control is returned to the next ILT entry.

e. EF5: Argument too large for the exponential subroutine. Control is returned to the next ILT entry.

f. EF6: Floating add, subtract, multiply; either an overflow or underflow occurred in the MQ or AC. Control is returned to the error location plus one.

g. EF7: Floating add, subtract, multiply; either an overflow or underflow occurred in the MQ or AC. Control is returned to the error location plus one.

h. EF8: The Object Program is setting status, hollerith, bit or byte information into a fixed mixed number. Control is returned to the next ILT entry.

i. EF9: Floating divide; either an overflow or underflow occurred in the MQ. Control is returned to the error location plus one.

j. EF1Ø: Floating divide; either an overflow or underflow occurred in the AC. Control is returned to the error location plus one.

k. EF11: Floating divide; either an overflow or underflow occurred in the AC or MQ. Control is returned to the error location plus one.

l. EF12: The Object Program is setting a non-status, hollerith, bit or byte information into a status or hollerith item. Control is returned to the next ILT entry.

m. EF13: Floating divide; either an overflow or underflow occurred in the AC or MQ. Control is returned to the error location plus one.

n. EF14: The Object Program is setting status, hollerith, bit or byte information into a floating number. Control is returned to the next ILT entry.

o. EF16: The number of entries of each table of the ENT instruction are not equal. Control is returned to the next ILT entry.

p. EF17: The left form of this ILT entry was ENT and the right form was not. Control is returned to the next ILT entry.

q. EF18: The operator of this ILT entry is not a set with the form of ENT. Control is returned to the next ILT entry.

F. The following messages may be generated during the operation of the Data Processing Program.

1. DATA PROCESSING PROGRAM

Generated by the Test Control Program and printed on the first line of the JDSYZ output for identification of program operation.

2. From the Trace and Store Subroutine:

a. UNASSIGNED UNIT FOR D-1. PRESS START TO CONTINUE.

Indicates to operator tape D-1 not available.

b. TAPE D-1 WRITE ERROR. CHANGE DRIVE AND PROCEED.

Indicates to operator a parity before any data was written on D-1.

c. TAPE D-1 WRITE ERROR. TRACE ABANDONED.

Informs programmer that no trace was performed because of a parity after data was written on tape.

3. From the Disassemble Compool and Data Comparison:

a. UNASSIGNED UNIT FOR C-1. PRESS START TO CONTINUE.

Indicates to operator, tape C-1 not available.

b. IDENT RECORD FOR BABY COMPOOL INCORRECT. NO DATA REDUCED. CONTROL RETURNED TO TCP.

Informs programmer that lack of Baby Compool forced abandonment of data reduction.

c. TAPE PARITY UNABLE TO READ BABY COMPOOL XXXXXX RECORD. NO DATA REDUCED. CONTROL RETURNED TO TCP.

Informs programmer that data reduction was abandoned due to a parity on read-in of Baby Compool Ident or Data Record.

d. IDENT RECORD FOR EXPECTED DATA INCORRECT. DATA BYPASSED.

Informs programmer that data reduction occurred without using any expected results.

e. IDENT RECORD FOR INITIAL DATA INCORRECT. DATA BYPASSED.

Informs programmer that data reduction occurred without using any initial conditions.

f. TAPE PARITY WHEN ATTEMPTING TO READ EXPECTED DATA RECORD. DATA NOT USED IN DATA REDUCTION.

Informs programmer that data reduction occurred without using any expected results.

g. TAPE PARITY WHEN ATTEMPTING TO READ INITIAL DATA RECORD. DATA NOT USED IN DATA REDUCTION.

Informs programmer that data reduction occurred without using initial conditions.

h. TILT TABLE FULL. AUTO TRACE STOPS HERE.

Informs programmer that auto trace performed only on those differences appearing prior to this comment.

i. THE NUMBER OF DIFFERENCES FOUND ARE XXXXXX.

Informs the programmer (in auto trace only) of the number of differing expected results.

j. COMREG XXXXXX CON XXXXXX SUB XXXXXX VAT XXXXXX SWT XXXXXX TLT XXXXXX SLT XXXXXX TABRG XXXXXX.

The variables contain the number of registers comprising the tables denoted at the left of each variable.

4. From Auto and Full Trace:

a. UNASSIGNED UNIT FOR D-1. PRESS START TO CONTINUE.

Informs operator tape D-1 not available.

b. SNAP TAPE TOO LARGE, INCOMPLETE TRACE.

Informs programmer that a full trace will be performed, which omits those snaps which are in excess of core capacity.

c. TAPE D-1 READ ERROR. INCOMPLETE TRACE.

Informs programmer that a full trace will be performed, omitting those snaps not read in when the parity occurred.

d. NO SNAP ON D-1 TAPE. NO TRACE PERFORMED.

Self-explanatory.

e. FULL TRACE FOLLOWS.

Informs programmer JDSYZ operating in full trace mode.

f. AUTO TRACE FOLLOWS.

Informs programmer JDSYZ operating in automatic trace mode.

5. From ILT Print:

a. EOF MISSING ON C1.

Indicates to programmer lack of EOF between data record and following ident record caused no ILT printout.

b. PARITY OR EOT ERROR WHEN READING C1.

Indicates to programmer that tape error caused ILT print to be abandoned.

c. XXXXXX IDENT ERROR (X's indicate table name). Indicates wrong ident caused abandonment of ILT print.

d. JDSYZ HAS DECODED ILT

Indicates successful ILT printout.

G. The following messages may be generated during the operation of the Test Control Program subsequent to the operation of a given test or series of tests.

1. ENDTST CARD MISSING

Although this type of error will not affect the system operation, it nevertheless appears to denote the omission of the ENDTST card in the deck of control cards.

2. TEST COMPLETE

Printed after the conclusion of operation of JDSYZ; this denotes the successful completion of system operation on one JOVIAL-coded program.

3. TEST CONTROL RUN COMPLETE

Printed upon the completion of an entire test run.

4. TAPE DISPOSTION

```
SAVE MASTER TAPE        DRIVE A1
PRINT OFF-LINE          DRIVE A2
PRINT OFF-LINE          DRIVE B1
DISPOSE OF C1, B1, D2
```

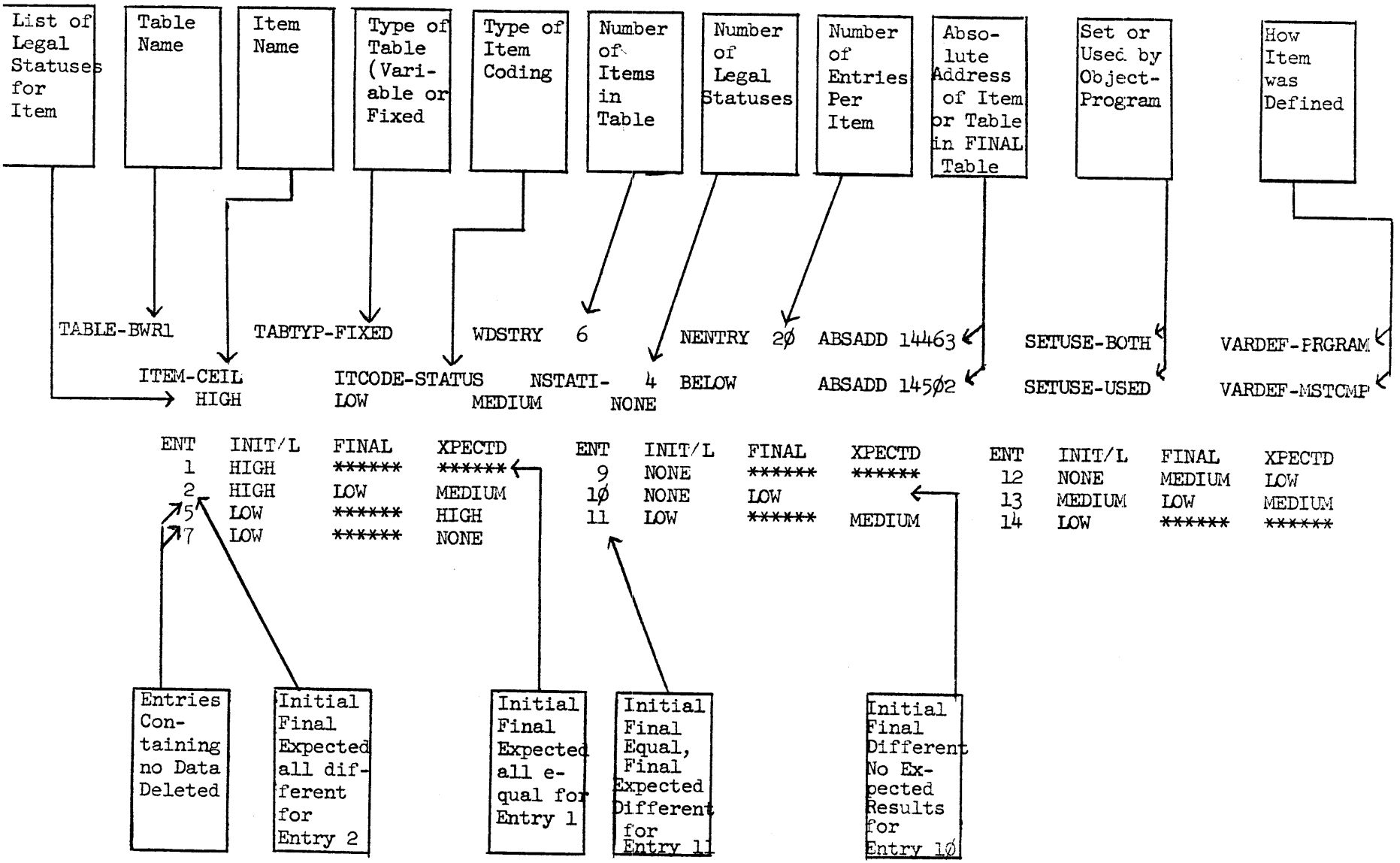
This message is printed upon completion of an entire test run, and indicates to the computer operators the procedure to be followed in handling the system tape.

5. ERROR DETECTED IN HELPFUL PACKAGE, TEST DISCONTINUED, CONTROL RETURNED TO TCP.

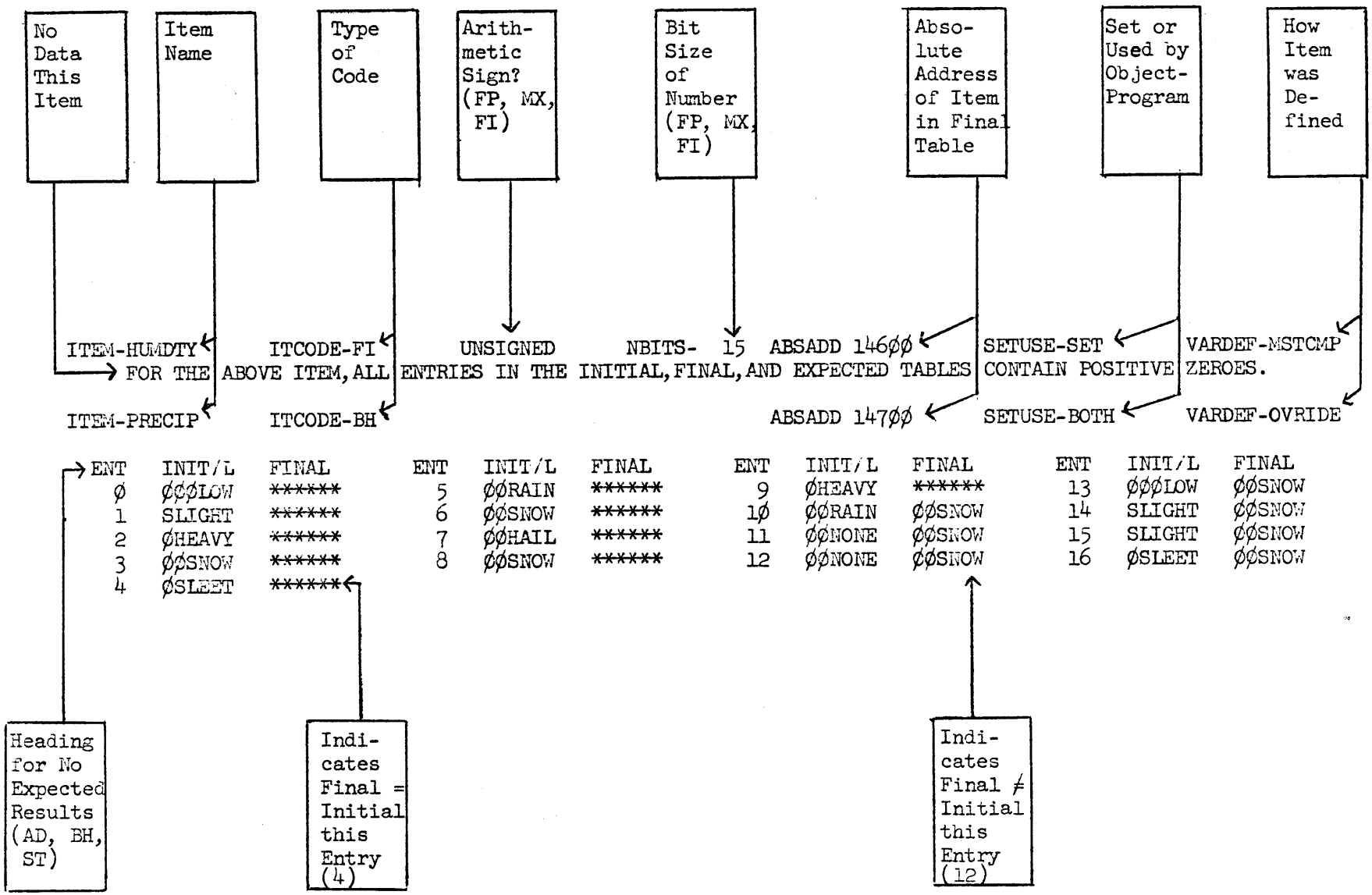
This message may occur at any time during a JOB run. The test is discontinued and the next JOB initiated.

Output From the Data Processing Program (JDSYZ)

The following five figures (15, 16, 17, 18 & 19) illustrate and describe the type of output the user will receive after his test has been run.

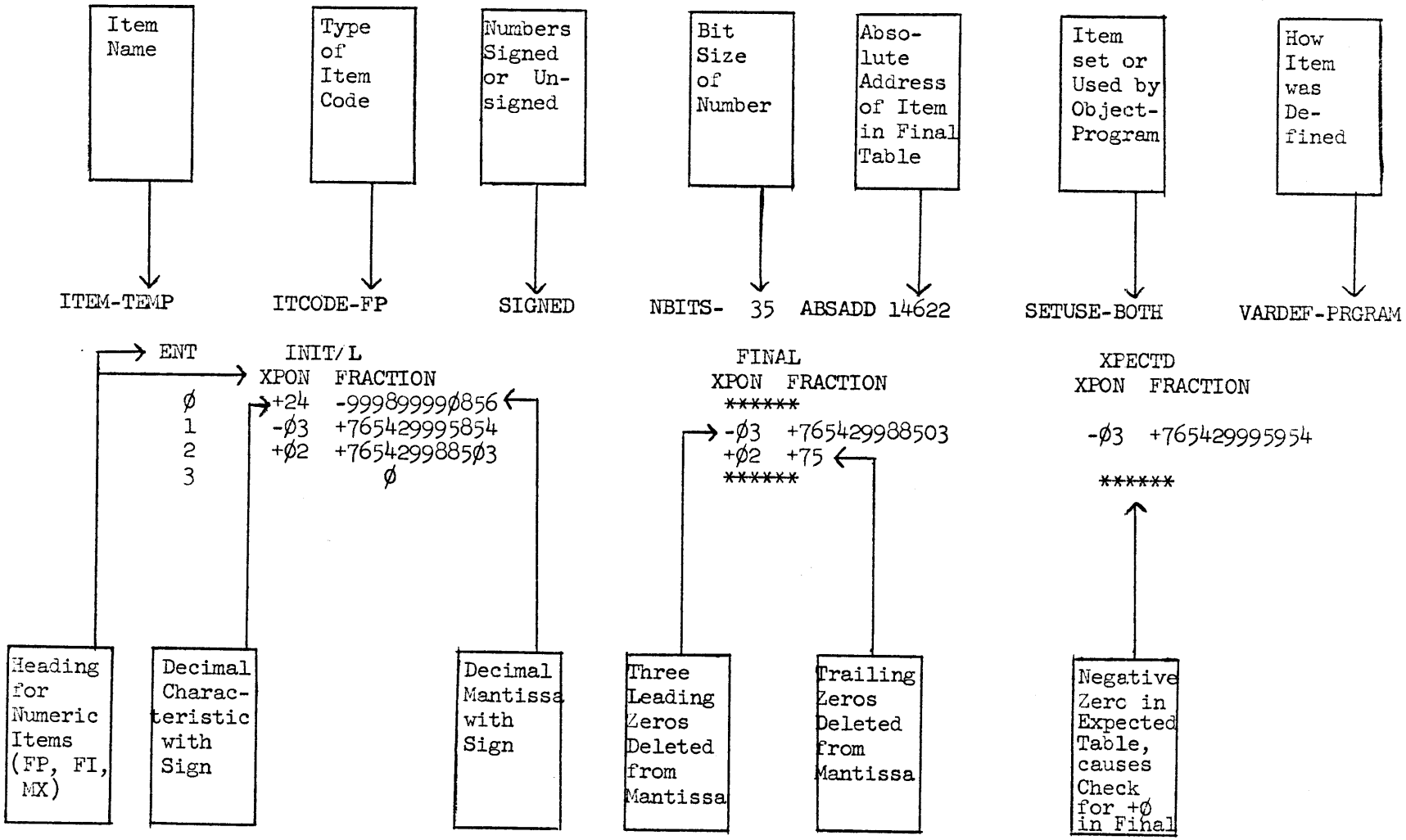


EXAMPLE OF TABLE ENTRY WITH STATUS ITEM AND EXPECTED RESULTS
FIGURE 15

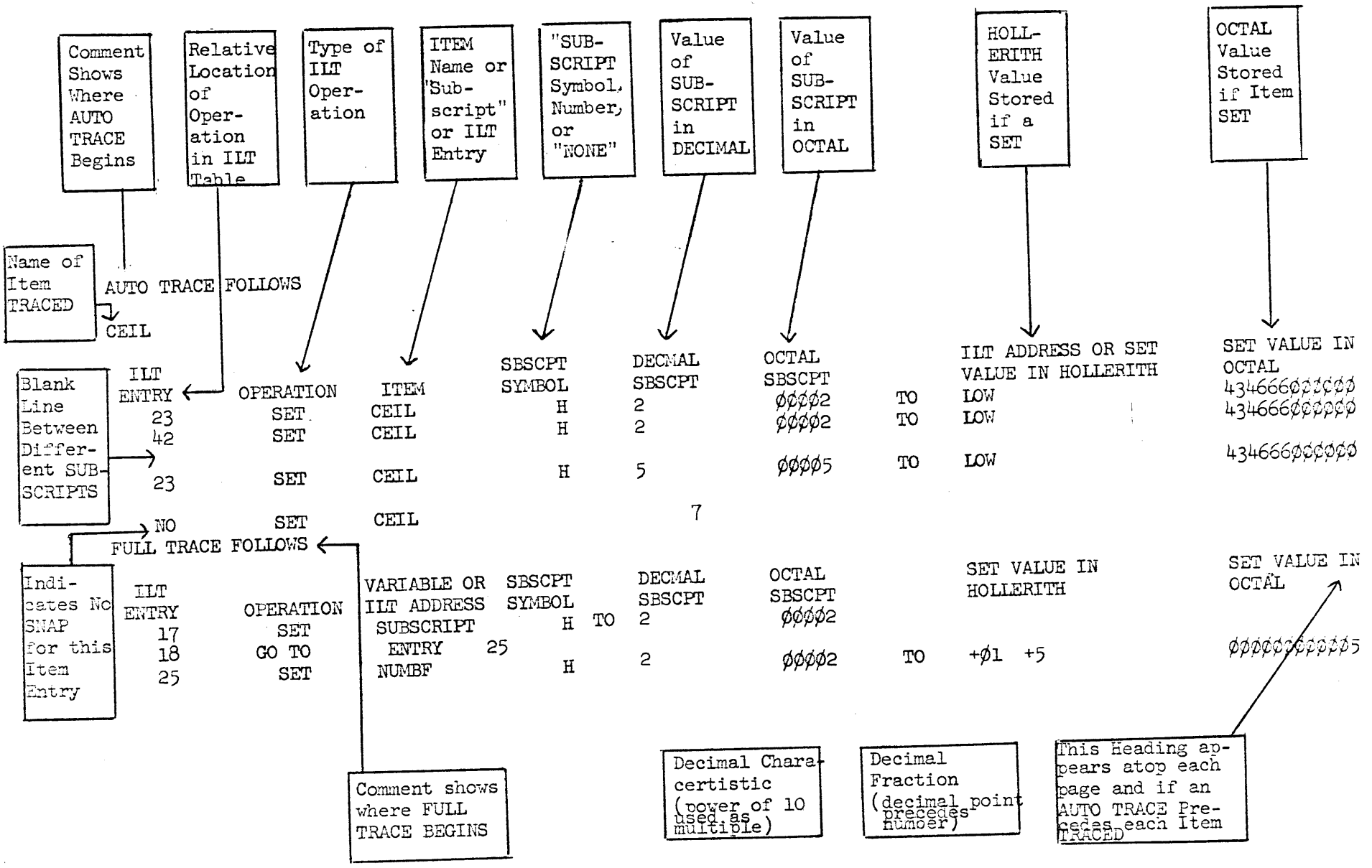


EXAMPLE OF FIXED INTEGER AND BINARY CODED HOLLERITH ITEMS WITH NO EXPECTED RESULTS

FIGURE 16



EXAMPLE OF ARITHMETICALLY-CODED ITEM DATA FORMAT
FIGURE 17



EXAMPLE OF AUTO AND FULL TRACE
FIGURE 8

NEW STATEMENT	ILT ENTRY	LEFT HALF TERM	OPERATION	RIGHT HALF TERM	ADDRESSE (S)
*	ø	SUBSCRIPT X	SET TO	CONST N	
	1	VAR SUB X ±N	PLUS	TEMP N	
*	2		GOTO		ILT NO N
	3	VAR	LS	VAR	TRUE N FALSE N
LABEL	4	VALUE VAR	DIV	VAR SUB X	TEMP N
	5		PROC DECLAR		ILT NO N
	6	N	STAT SW DECL	VAR	LABEL OF SWITCH
*	7	N	NUMR SW DECL		LABEL OF SWITCH
*	8		PROC CALL		LABEL OF PROCEDURE
	9		ONE OUTPT PR		ILT NO N
*	1ø	SUBSCRIPT X	INPUT PARAM	VAR	
	11	SUBSCRIPT X	OUTPT PARAM	VAR	
	12		END		
	13		TERM		ILT NO N

* = new statement
X = subscript letter
N = number of (ILT entry
(constant
(temporary storage
(entries in switch table
VAR = variable in Hollerith
LABEL - label of new statement

STRUCTURE OF ILT PRINTOUT
FIGURE 19

In an attempt to illustrate the many and possibly confusing concepts and rules that have been presented, two sample problems have been chosen that will illustrate system reaction to a program under test.

SAMPLE PROBLEM - FLITE

Four bombers are flying from a certain air base to various destinations. Initially there is enough fuel for the trip, but because of unexpected conditions, they may run short. At periodical intervals each bomber relays to the originating base data concerning remaining fuel supply and time in minutes to the destination. The programming example chosen checks the fuel consumption, amount of fuel remaining, and time to go, in order to determine whether a tanker should be sent to refuel the bomber.

Information is available to the program in the form of a table (FLIØ) of data relating to the aircraft. Each entry in the table describes the data for one aircraft. This data includes - "RATE," fuel consumption rate for this aircraft; "RFUEL," fuel remaining in aircraft; "TTDEST," time to destination; and "TANKER," an item of data that describes the current status of this aircraft from the refueling standpoint. "RFUEL" and "TTDEST" are assumed to be updated periodically from radio reports.

If enough fuel remains to reach destination, the "TANKER" status will be set to "NONE." If there is insufficient fuel to reach destination, a further test is made; if remaining fuel will last for less than 30 minutes, the "TANKER" status becomes "URGENT," otherwise it is set to "NEEDED."

A requisite for operation is a system master tape containing sub-programs and an appropriate Master Communication Pool. For our problem, we are creating a COMPOOL describing a finite set of data. A set of input cards is punched containing the data descriptions required. In actual practice, the COMPOOL would contain data descriptions for ALL system data.

The sample COMPOOL will contain the following data descriptions:

1. A table symbolically labeled FLTØ that contains four sets (entries) of items that relate to the four aircraft in the problem.
2. The item "RATE," a part of the FLTØ table, describes the rate of fuel consumption in gallons per minute. It is defined as an 8 bit unsigned mixed fixed point number with two fractional bits.
3. The item "RFUEL," a part of the FLTØ table, describes the number of gallons of fuel remaining. It is an unsigned 12 bit fixed point integer.
4. The item "TTDEST," a part of the FLTØ table, describes the number of minutes remaining to destination. It is an unsigned 10 bit fixed point integer.
5. The item "TANKER," a part of the FLTØ table, describes the "TANKER" status associated with this aircraft. Possible settings are:

   NONE - Tanker not needed.

   NEEDED - Tanker-bomber mating is required.

   URGENT - Tanker-bomber mating is required within 30 minutes.

The input deck required to create this COMPOOL is listed below. This operation will be performed through the direct card reader, therefore Sense Switch #1 will be down. The COMPOOL will be identified as JCXØØØ and will be addressed as such when later used. A direct copy of the printed output from the computer follows the listing of the input deck.

```
DATE  020160
JAMCZ JCX000
  C   JCX000                                                    0001  JCX00000
  T   FLTO     4 R                                             0002  JCX00000
  I   RATE   FLTO  MX U 8   2                                  0003  JCX00000
  I   RFUEL  FLTO  FI U 12                                     0004  JCX00000
  I   TANKER FLTO  ST NONE NEEDED URGENT                       0005  JCX00000
  I   TTDEST FLTO  FI U 10                                     0006  JCX00000
  END
```

ASSEMBLE MASTER COMPOOL

02/01/60 PAGE 1

SUMMARY INFORMATION. ASSEMBLE MASTER COMPOOL. COMPOOL IDENT JCXOOO

TABLE FLTO ITEMS

RATE RFUEL TANKER TTDEST

COMPOOL JCXOOO WAS ASSEMBLED SUCCESSFULLY AND IS ON BUFFER TAPE A3

TEST CONTROL RUN COMPLETE

TAPE DISPOSITION

SAVE MASTER TAPE DRIVE A1

PRINT OFF LINE DRIVE A2

PRINT OFF LINE DRIVE B1

DISPOSE OF C1, D1, D2

A COPY OF THE COMPUTER OUTPUT FROM THE ASSEMBLY OF THE COMPOOL JCXØØØ

FOR THE SAMPLE PROBLEM "FLITE"

Interpretation of the FLITE program may now proceed on the assumption that the compool JCXØØØ has been made a part of a System Master Tape through the use of the System Tape Loading Program.

The Input deck is prepared and a decision is made that the operation will be performed through the ON-LINE Card Reader. This will be test #1 of the FLITE program using compool JCXØØØ with a FULL Trace requested.

A listing of the deck to be placed in the card reader follows:

Note that the item FULTIM is described in the program since it is a temporary storage location.

```
DATE  020160
FLITE TEST  1     JCX000FULL
FLITE TESTID  K E PETERSEN  SAMPLE PROGRAM
START        FLITE  SAMPLE PROGRAM
       ITEM FULTIM A 8 U 2 $
      FOR I =  0,1, NENT(FLTO)-1 $
 BEGIN FULTIM =  RFUEL($I$)/RATE($I$) $
       IF FULTIM LS TTDEST($I$) $
  BEGIN IF FULTIM LQ 30 $
   BEGIN TANKER($I$) =  V(URGENT) $
         TEST $
   END
        TANKER($I$) =  V(NEEDED) $
        TEST $
  END
       TANKER($I$) =  V(NONE) $
 END
TERM $
          0 TANKER      URGENT
          1 TANKER      NONE
          2 TANKER      NEEDED
          3 TANKER      URGENT
```

```
      ENDE
          O RATE          5.75A2
          1 RATE          3.A2
          2 RATE          1.A2
          3 RATE          6.5A2
          O RFUEL         120
          1 RFUEL         200
          2 RFUEL         35
          3 RFUEL         200
          O TTDEST        50
          1 TTDEST        30
          2 TTDEST        60
          3 TTDEST        40
      END
ENDTST
```

A direct reproduction of the computer output follows. An examination of the output reveals that a difference was found between the expected and final values of entry #3 of the item "TANKER." This situation is caused by an error in the expected result for this item, but this fact can only be determined by the programmer after analysis of the program's operation.

NOTE: The table -- lTABLE is a pseudo table created by the system to facilitate the handling of ITEMS which are not a part of any other table. These ITEMS are generally temporary storage locations defined in the JOVIAL program.

```
FLITE 1        K E PETERSEN  SAMPLE PROGRAM                    JCX000      IMT001     02/01/60

INTERPRETER FIRST PASS

JCX000
START       FLITE  SAMPLE PROGRAM

ASSEMBLE BABY COMPOOL

DATA SIMULATION PROGRAM

JSTRZ HAS FINISHED PROCESSING THE TABLE AND ITEM ENVIRONMENT FOR THIS TEST

INTERPRETER SECOND PASS

DATA PROCESSING PROGRAM
```

TABLE-1TABLE TABTYP-FIXED WDSTRY 1 NENTRY 1 ABSADD 77776 SETUSE-NULL VARDEF-PRGRAM

ITEM-FULTIM ITCODE-MX UNSIGNED NBITS- 8 ABSADD 77777 SETUSE-BOTH VARDEF-PRGRAM

ENT	INIT/L		FINAL		XPECTD	
	XPON	FRACTION	XPON	FRACTION	XPON	FRACTION
0		0	+02	+3075		

TABLE-FLTO TABTYP-FIXED WDSTRY 4 NENTRY 4 ABSADD 77755 SETUSE-USED VARDEF-MSTCMP

ITEM-RATE ITCODE-MX UNSIGNED NBITS- 8 ABSADD 77762 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+01	+575	******			
1	+01	+3	******			
2	+01	+1	******			
3	+01	+65	******			

ITEM-RFUEL ITCODE-FI UNSIGNED NBITS- 12 ABSADD 77756 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+03	+12	******			
1	+03	+2	******			
2	+02	+35	******			
3	+03	+2	******			

ITEM-TANKER ITCODE-STATUS MSTATI- 3 BELOW. ABSADD 77772 SETUSE-SET VARDEF-MSTCMP

URGENT NEEDED NONE

ENT	INIT/L	FINAL	XPECTD
0	0	URGENT	******
1	0	NONE	******

ENT	INIT/L	FINAL	XPECTD
2	0	NEEDED	******

ENT	INIT/L	FINAL	XPECTD
3	0	NEEDED	URGENT

ITEM-TTDEST ITCODE-FI UNSIGNED NBITS- 10 ABSADD 77766 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+02	+5	******			
1	+02	+3	******			
2	+02	+6	******			
3	+02	+4	******			

COMREG 26 CON 7 SUB 28 VAT 29 SWT 1 ILT 81 SLT 13 TABRG 19

FULL TRACE FOLLOWS

ILT ENTRY	OPERATION	VARIABLE OR ILT ADDRESS	SBSCPT SYMBOL		DECMAL SBSCPT	OCTAL SBSCPT		SET VALUE IN HOLLERITH	SET VALUE IN OCTAL
0	SET	SUBSCRIPT	I	TO	0	00000			
2	SET	FULTIM	NONE		0	00000	TO	+02 +2075	000000000123
3	GO TO	ENTRY 4							
4	GO TO	ENTRY 5							
5	SET	TANKER	I		0	00000	TO	URGENT	645127254563
6	GO TO	ENTRY 10							
11	GO TO	ENTRY 12							
13	SET	SUBSCRIPT	I	TO	1	00001			
14	GO TO	ENTRY 1							
2	SET	FULTIM	NONE		0	00000	TO	+02 +665	000000000412
3	GO TO	ENTRY 9							
9	SET	TANKER	I		1	00001	TO	NONE	454645256060
11	GO TO	ENTRY 12							
13	SET	SUBSCRIPT	I	TO	2	00002			
14	GO TO	ENTRY 1							
2	SET	FULTIM	NONE		0	00000	TO	+02 +35	000000000214
3	GO TO	ENTRY 4							
4	GO TO	ENTRY 7							
7	SET	TANKER	I		2	00002	TO	NEEDED	452525242524
8	GO TO	ENTRY 10							
11	GO TO	ENTRY 12							
13	SET	SUBSCRIPT	I	TO	3	00003			
14	GO TO	ENTRY 1							
2	SET	FULTIM	NONE		0	00000	TO	+02 +3075	000000000173
3	GO TO	ENTRY 4							
4	GO TO	ENTRY 7							
7	SET	TANKER	I		3	00003	TO	NEEDED	452525242524
8	GO TO	ENTRY 10							
11	GO TO	ENTRY 15							

NEW STATEMENT	ILT ENTRY	LEFT HALF TERM	OPERATION	RIGHT HALF TERM	ADDRESSE (S)
*	0	SUBSCRIPT I	SET TO	CONST 0	
*	1	RFUEL SUB I	DIV	RATE SUB I	TEMP 1
	2	FULTIM	SET TO	TEMP 1	
*	3	FULTIM	LS	TTDEST SUB I	TRUE 4 FALSE 9
*	4	FULTIM	LQ	CONST +02 +3	TRUE 5 FALSE 7
*	5	TANKER SUB I	SET TO	VALUE URGENT	
*	6		GOTO		ILT NO 10
*	7	TANKER SUB I	SET TO	VALUE NEEDED	
*	8		GOTO		ILT NO 10
*	9	TANKER SUB I	SET TO	VALUE NONE	
*	10	(NENT)FLTO	MINUS	CONST +01 +1	TEMP 1
	11	SUBSCRIPT I	EQ	TEMP 1	TRUE 15 FALSE 12
	12	SUBSCRIPT I	PLUS	CONST +01 +1	TEMP 1
	13	SUBSCRIPT I	SET TO	TEMP 1	
	14		GOTO		ILT NO 1
*	15		TERM		ILT NO 0

JDSYZ HAS DECODED ILT.

TEST COMPLETE

TEST CONTROL RUN COMPLETE

TAPE DISPOSITION

SAVE MASTER TAPE DRIVE A1

PRINT OFF LINE DRIVE A2

PRINT OFF LINE DRIVE B1

DISPOSE OF C1, D1, D2

If an Automatic Trace had been requested, the output in place of the FULL trace would have been as follows:

Page 3 of the output (page 134 of this document) would have contained the notation THE NUMBER OF DIFFERENCES FOUND ARE 1

Page 4 of the output (page 135 of this document) would have been replaced by the page that follows.

AUTO TRACE FOLLOWS

TANKER

ILT ENTRY	OPERATION	VARIABLE OR ILT ADDRESS	SBSCPT SYMBOL	DECMAL SBSCPT	OCTAL SBSCPT		SET VALUE IN HOLLERITH	SET VALUE IN OCTAL
7	SET	TANKER	I	3	00003	TO	NEEDED	452525242524

SAMPLE PROBLEM - ADMIT

This sample program is a hypothetical college admissions procedure. The college in question is coeducational with an enrollment of 3Ø students per class. Admission is based on the following factors. The ratio of men to women in each class must be 3 to 2. A geographical distribution must be maintained. For this purpose the country is divided into 5 areas, North, East, South, West and Central. The distribution in each class must be 2Ø% North, 4Ø% East, 1Ø% South, 2Ø% West, and 1Ø% Central. College Board Entrance Exams are required, and considering each score of the 3 parts of the exam separately, no one score may be below 5Ø percentile, and the average score must be equal to or greater than 75 percentile. The applicants are considered on a first-come basis, that is, taking the applications in the order in which they are received, the first 3Ø qualified students are accepted.

Information is available to the program in the form of a table of the applicants and the necessary data concerning them. Each entry in the table contains the data for one applicant. This data includes: "NAME," name of the applicant; "SCOREA," "SCOREB," and "SCOREC" which are the applicant's scores for the 3 parts of the College Board Entrance Exam; "STASEX," which contains either ØM or ØF initially, the M or F designating the sex of the applicant; and "LOCA," the geographical area in which the applicant lives, North, East, etc.

The program successively tests each item of this data. As soon as the applicant fails to meet a specification, the program rejects the applicant and moves on to the next one. If an applicant is rejected, the first character of the "STASEX" item remains a "Ø." If the applicant is accepted, this character is changed to an "A."

In addition to the above mentioned items, several other items are defined for the program. They are: "AREA (Ø)". . . "AREA (4)," which contain the running totals of the number of applicants accepted from the 5 geographical areas; "NUMBM," which contains the running total of the number of men accepted; and "NUMBF," which contains the running total of the number of women accepted.

The program given here is not a sophisticated solution to the above problem but is rather designed to illustrate as many types of JOVIAL statements as possible. The program makes only one pass through the applicants, so that if, for example, 12 qualified women are not found, no provision is made to complete the class size of 3Ø by taking more than 18 men. This also applies to the geographical distribution quotas.

A requisite for operation is a system master tape containing subprograms and an appropriate Master Communication Pool (COMPOOL). For this problem we are creating a COMPOOL describing a finite set of data. A set of input cards is punched containing the data descriptions required. In actual practice the COMPOOL would contain data descriptions for the entire system; here we are considering only one part of a system.

The sample COMPOOL will contain the following data descriptions:

1. A table symbolically labeled "APP1" containing the data about the applicants. The number of entries in this table is equal to the number of the applicants, and each entry contains six items.
2. The item "NAME," a part of the "APP1" table, containing the name of the applicant. It is a binary-coded Hollerith item containing six Hollerith characters. Therefore, this item contains the first six letters of the applicant's last name.
3. The items "SCOREA," "SCOREB" and "SCOREC," parts of the "APP1" table, containing the applicants' scores for the three parts of the College Board Entrance Exam. Each item is a seven bit fixed point integer.
4. The item "STASEX," a part of the "APP1" table, containing two Hollerith characters, the first being set by the program to indicate the status of the applicant (i.e., A for accepted or Ø rejected), and the second to designate the sex of the applicant.
5. The item "LOCA," a part of the "APP1" table, containing the geographical area (as status values), in which the applicant lives. The possible values of this item are:

   NORTH

   EAST

   SOUTH

   WEST

   CENTRL

6. A table symbolically labeled "TOT1" containing the item "AREA." It is a fixed length table of five entries.

7. The item "AREA," the only item in the "TOT1" table, containing the running totals of the number of applicants accepted from each of the five geographical areas. "AREA(Ø)," the value of "AREA" in the first entry of the table, contains the total for the area North. "AREA(1)," "AREA(2)," "AREA(3)" and "AREA(4)" contain the totals for East, South, West and Central respectively. This item is a seven bit, fixed point, unsigned integer.

8. A table symbolically labeled "TOT2" containing the running totals of the number of men and women accepted. It is a fixed length table of one entry and this entry contains two items.

9. The item "NUMBM," a part of the "TOT2" table, containing the running total of the number of men accepted. It is a five bit, fixed point, unsigned integer.

10. The item "NUMBF," a part of the "TOT2" table, containing the running total of the number of women accepted. It is a five bit, fixed point, unsigned integer.

The input deck required to create this COMPOOL is listed below. The input deck will be entered via prestored tape. The COMPOOL will be identified as JCYØØØ and will be addressed as such when later used.

A direct copy of the printed output from the computer follows the listing of the input deck.

```
DATE  020160
JAMCZ JCY000
  C   JCY000
  T   APP1  70   R
  I   AREA   TOT1  FI U  7
  I   LOCA   APP1  ST
  S   LOCA   NORTH EAST SOUTH WEST CENTRL
  I   NAME   APP1  BH    36
  I   SCOREA APP1  FI U  7
  I   SCOREB APP1  FI U  7
  I   SCOREC APP1  FI U  7
  I   STASEX APP1  BH   12
  T   TOT1   5   R
  T   TOT2   1   R
  I   NUMBM  TOT2  FI U  5
  I   NUMBF  TOT2  FI U  5
  END
```

```
0001  JCY000
0002  JCY000
0003  JCY000
0004  JCY000
0005  JCY000
0006  JCY000
0007  JCY000
0008  JCY000
0009  JCY000
0010  JCY000
0011  JCY000
0012  JCY000
0013  JCY000
0014  JCY000
```

ASSEMBLE MASTER COMPOOL PAGE 1

02/01/60

SUMMARY INFORMATION. ASSEMBLE MASTER COMPOOL. COMPOOL IDENT JCYOOO

TABLE APP1 ITEMS

LOCA NAME SCOREA SCOREB SCOREC STASEX

ASSEMBLE MASTER COMPOOL PAGE 2

TABLE TOT1 ITEMS

AREA

ASSEMBLE MASTER COMPOOL PAGE 3

TABLE TOT2 ITEMS

NUMBF NUMBM

COMPOOL JCYOOO WAS ASSEMBLED SUCCESSFULLY AND IS ON BUFFER TAPE A3

TEST CONTROL RUN COMPLETE

TAPE DISPOSITION

SAVE MASTER TAPE DRIVE A1

PRINT OFF LINE DRIVE A2

PRINT OFF LINE DRIVE B1

DISPOSE OF C1, D1, D2

A COPY OF THE COMPUTER OUTPUT FROM THE ASSEMBLY OF THE COMPOOL JCYØØØ FOR THE SAMPLE PROBLEM "ADMIT"

FN-LO-197
1 February 1960
Page 1144

Interpretation of the ADMIT Program may now proceed on the assumption that the Compool JCYØØØ has been made a part of a system master tape through the use of the System Tape Loading Program.

The decision is made that the input deck will be prestored onto magnetic tape due to its size. Therefore, the control cards only will be inserted through the ON-LINE Card Reader. This will be test #1 of the ADMIT Program using Compool JCYØØØ with an automatic trace requested.

A listing of the control cards and the input deck follows.

```
THE FOLLOWING CARDS ARE ENTERED THROUGH THE CARD READER

DATE  020160
ADMIT TEST  1     JCY000
ENDTST


THE FOLLOWING CARDS ARE PRESTORED ONTO MAGNETIC TAPE

ADMIT TESTID  ELEANOR HURD  SAMPLE COLLEGE ADMISSION PROGRAM
START    ADMIT  E HURD  SAMPLE PROGRAM
      FOR J = 0 $
      FOR I = ALL(NAME) $
  BEGIN IF SCOREA($I$) LS 50 OR SCOREB ($I$) LS 50 OR
       SCOREC ($I$) LS 50 $
      TEST $
      IF (SCOREA($I$) + SCOREB($I$) + SCOREC($I$))/3 LS 75 $
      TEST $
      GOTO DISTRB($I$) $
      TEST $
      SWITCH DISTRB(LOCA) = (NORTH=A1, EAST=A2, SOUTH=A3,
                  WEST=A4, CENTRL=A5) $
 A1.  IF AREA($0$) EQ 6 $
      TEST $
      J = 0 $
      GOTO A6 $
 A2.  IF AREA($1$) EQ 12 $
      TEST $
      J = 1 $
      GOTO A6 $
 A3.  IF AREA($2$) EQ 3 $
      TEST $
      J = 2 $
      GOTO A6 $
 A4.  IF AREA($3$) EQ 6 $
      TEST $
      J = 3 $
      GOTO A6 $
 A5.  IF AREA($4$) EQ 3 $
      TEST $
      J = 4 $
 A6.  IF BYTE($1$)(STASEX($I$)) EQ 1H(M) AND NUMBM EQ 18 OR
```

```
          BYTE($1$)(STASEX($I$)) EQ 1H(F) AND NUMBF EQ 12 $
      TEST $
      IF BYTE($1$)(STASEX($I$)) EQ 1H(M) $
BEGIN NUMBM = NUMBM + 1 $
      GOTO A7 $
END
      NUMBF = NUMBF + 1 $
 A7.  AREA($J$) = AREA($J$) +1 $
      BYTE($0$)(STASEX($I$)) = 1H(A) $
      IF (NUMBF + NUMBM) NQ 30 $
END
      STOP$
TERM $
       0000 STASEX       0M
       0001 STASEX       AM
       0002 STASEX       AM
       0003 STASEX       AF
       0004 STASEX       0F
       0005 STASEX       0M
       0006 STASEX       0F
       0007 STASEX       0F
       0008 STASEX       AM
       0009 STASEX       AF
       0010 STASEX       AF
       0011 STASEX       0M
       0012 STASEX       AF
       0013 STASEX       AM
       0014 STASEX       0M
       0015 STASEX       0M
       0016 STASEX       AF
       0017 STASEX       AM
       0018 STASEX       AF
       0019 STASEX       AM
       0020 STASEX       0F
       0021 STASEX       AM
       0022 STASEX       AF
       0023 STASEX       0M
       0024 STASEX       AM
       0025 STASEX       0M
       0026 STASEX       AF
       0027 STASEX       AF
       0028 STASEX       0M
       0029 STASEX       AM
```

0030	STASEX	0M
0031	STASEX	AM
0032	STASEX	0F
0033	STASEX	AM
0034	STASEX	0F
0035	STASEX	AF
0036	STASEX	AM
0037	STASEX	0F
0038	STASEX	0M
0039	STASEX	0M
0040	STASEX	AF
0041	STASEX	AM
0042	STASEX	0F
0043	STASEX	AM
0044	STASEX	0M
0045	STASEX	AM
0046	STASEX	AF
0047	STASEX	0F
0048	STASEX	0F
0049	STASEX	0F
0050	STASEX	0F
0051	STASEX	AM
0052	STASEX	0M
0053	STASEX	0F
0054	STASEX	0M
0055	STASEX	0F
0056	STASEX	0M
0057	STASEX	0F
0058	STASEX	0M
0059	STASEX	0M
0060	STASEX	0F
0061	STASEX	0M
0062	STASEX	AM
0063	STASEX	0F
0064	STASEX	0M
0065	STASEX	AM
0066	STASEX	0M
0067	STASEX	0F
0068	STASEX	0M
0069	STASEX	0M
0	AREA	6
1	AREA	12
2	AREA	3

```
     3 AREA          6
     4 AREA          3
     0 NUMBM         18
     0 NUMBF         12
ENDE
 0000 NAME           AARON
 0001 NAME           ADRIAN
 0002 NAME           AHERN
 0003 NAME           ALDEN
 0004 NAME           ALLEN
 0005 NAME           ANDERS
 0006 NAME           ARMOUR
 0007 NAME           BARKER
 0008 NAME           BARRON
 0009 NAME           BARTON
 0010 NAME           BELL
 0011 NAME           BROWN
 0012 NAME           CARSON
 0013 NAME           CHING
 0014 NAME           CHOMA
 0015 NAME           COLLIN
 0016 NAME           DEITCH
 0017 NAME           DEMONE
 0018 NAME           DODGE
 0019 NAME           DREW
 0020 NAME           ELLIOT
 0021 NAME           EMMONS
 0022 NAME           EUSNER
 0023 NAME           EVANS
 0024 NAME           EWEN
 0025 NAME           EZZELL
 0026 NAME           FABIAN
 0027 NAME           FADDEN
 0028 NAME           FREED
 0029 NAME           FREUD
 0030 NAME           GATES
 0031 NAME           GERBER
 0032 NAME           GIDEON
 0033 NAME           GORDON
 0034 NAME           GOULD
 0035 NAME           HAMMER
 0036 NAME           HANDEL
 0037 NAME           HANSON
```

```
0038 NAME        HILL
0039 NAME        HOGEN
0040 NAME        HUGHES
0041 NAME        HUGGIN
0042 NAME        IBISCH
0043 NAME        IBSEN
0044 NAME        IDIOT
0045 NAME        JARVIS
0046 NAME        JAY
0047 NAME        JENSEN
0048 NAME        JONES
0049 NAME        JORDAN
0050 NAME        KATZ
0051 NAME        KELLY
0052 NAME        KERR
0053 NAME        KONRAD
0054 NAME        LADD
0055 NAME        LATONA
0056 NAME        LITTLE
0057 NAME        MACRAE
0058 NAME        MANLEY
0059 NAME        MASON
0060 NAME        MCCUE
0061 NAME        NORRIS
0062 NAME        ONEILL
0063 NAME        PALMER
0064 NAME        PAPP
0065 NAME        PARKER
0066 NAME        PARMI
0067 NAME        PELZER
0068 NAME        RAMSEY
0069 NAME        REISCH
0000 SCOREA      60
0001 SCOREA      70
0002 SCOREA      90
0003 SCOREA      85
0004 SCOREA      60
0005 SCOREA      70
0006 SCOREA      70
0007 SCOREA      40
0008 SCOREA      80
0009 SCOREA      90
0010 SCOREA      80
```

```
0011 SCOREA      70
0012 SCOREA      75
0013 SCOREA      60
0014 SCOREA      60
0015 SCOREA      80
0016 SCOREA      85
0017 SCOREA      75
0018 SCOREA      90
0019 SCOREA      80
0020 SCOREA      70
0021 SCOREA      80
0022 SCOREA      85
0023 SCOREA      90
0024 SCOREA      85
0025 SCOREA      70
0026 SCOREA      75
0027 SCOREA      80
0028 SCOREA      85
0029 SCOREA      95
0030 SCOREA      85
0031 SCOREA      80
0032 SCOREA      70
0033 SCOREA      80
0034 SCOREA      60
0035 SCOREA      80
0036 SCOREA      85
0037 SCOREA      80
0038 SCOREA      85
0039 SCOREA      70
0040 SCOREA      80
0041 SCOREA      85
0042 SCOREA      75
0043 SCOREA      95
0044 SCOREA      40
0045 SCOREA      85
0046 SCOREA      90
0047 SCOREA      85
0048 SCOREA      70
0049 SCOREA      80
0050 SCOREA      75
0051 SCOREA      85
0052 SCOREA      70
0053 SCOREA      85
```

```
0054 SCOREA      85
0055 SCOREA      90
0056 SCOREA      70
0057 SCOREA      85
0058 SCOREA      80
0059 SCOREA      60
0060 SCOREA      50
0061 SCOREA      80
0062 SCOREA      85
0063 SCOREA      80
0064 SCOREA      70
0065 SCOREA      80
0066 SCOREA      85
0067 SCOREA      90
0068 SCOREA      80
0069 SCOREA      85
0000 SCOREB      70
0001 SCOREB      80
0002 SCOREB      90
0003 SCOREB      80
0004 SCOREB      55
0005 SCOREB      70
0006 SCOREB      70
0007 SCOREB      80
0008 SCOREB      85
0009 SCOREB      95
0010 SCOREB      80
0011 SCOREB      70
0012 SCOREB      75
0013 SCOREB      80
0014 SCOREB      45
0015 SCOREB      80
0016 SCOREB      95
0017 SCOREB      80
0018 SCOREB      90
0019 SCOREB      85
0020 SCOREB      75
0021 SCOREB      85
0022 SCOREB      95
0023 SCOREB      90
0024 SCOREB      85
0025 SCOREB      70
0026 SCOREB      90
```

```
0027 SCOREB      80
0028 SCOREB      75
0029 SCOREB      95
0030 SCOREB      80
0031 SCOREB      85
0032 SCOREB      70
0033 SCOREB      80
0034 SCOREB      60
0035 SCOREB      85
0036 SCOREB      90
0037 SCOREB      80
0038 SCOREB      85
0039 SCOREB      70
0040 SCOREB      80
0041 SCOREB      90
0042 SCOREB      70
0043 SCOREB      95
0044 SCOREB      30
0045 SCOREB      85
0046 SCOREB      90
0047 SCOREB      80
0048 SCOREB      70
0049 SCOREB      80
0050 SCOREB      90
0051 SCOREB      85
0052 SCOREB      70
0053 SCOREB      85
0054 SCOREB      80
0055 SCOREB      80
0056 SCOREB      70
0057 SCOREB      85
0058 SCOREB      80
0059 SCOREB      60
0060 SCOREB      60
0061 SCOREB      80
0062 SCOREB      90
0063 SCOREB      90
0064 SCOREB      70
0065 SCOREB      80
0066 SCOREB      70
0067 SCOREB      90
0068 SCOREB      80
0069 SCOREB      85
```

```
0000 SCOREC      80
0001 SCOREC      80
0002 SCOREC      90
0003 SCOREC      90
0004 SCOREC      70
0005 SCOREC      70
0006 SCOREC      65
0007 SCOREC      90
0008 SCOREC      85
0009 SCOREC      80
0010 SCOREC      85
0011 SCOREC      75
0012 SCOREC      75
0013 SCOREC      90
0014 SCOREC      70
0015 SCOREC      60
0016 SCOREC      85
0017 SCOREC      90
0018 SCOREC      95
0019 SCOREC      80
0020 SCOREC      75
0021 SCOREC      85
0022 SCOREC      90
0023 SCOREC      80
0024 SCOREC      85
0025 SCOREC      70
0026 SCOREC      90
0027 SCOREC      80
0028 SCOREC      45
0029 SCOREC      95
0030 SCOREC      80
0031 SCOREC      85
0032 SCOREC      70
0033 SCOREC      80
0034 SCOREC      60
0035 SCOREC      95
0036 SCOREC      85
0037 SCOREC      80
0038 SCOREC      85
0039 SCOREC      75
0040 SCOREC      90
0041 SCOREC      85
0042 SCOREC      70
```

```
0043 SCOREC      90
0044 SCOREC      30
0045 SCOREC      85
0046 SCOREC      90
0047 SCOREC      80
0048 SCOREC      75
0049 SCOREC      80
0050 SCOREC      85
0051 SCOREC      90
0052 SCOREC      85
0053 SCOREC      80
0054 SCOREC      80
0055 SCOREC      80
0056 SCOREC      70
0057 SCOREC      85
0058 SCOREC      90
0059 SCOREC      70
0060 SCOREC      80
0061 SCOREC      80
0062 SCOREC      90
0063 SCOREC      90
0064 SCOREC      70
0065 SCOREC      80
0066 SCOREC      75
0067 SCOREC      95
0068 SCOREC      85
0069 SCOREC      85
0000 STASEX      0M
0001 STASEX      0M
0002 STASEX      0M
0003 STASEX      0F
0004 STASEX      0F
0005 STASEX      0M
0006 STASEX      0F
0007 STASEX      0F
0008 STASEX      0M
0009 STASEX      0F
0010 STASEX      0F
0011 STASEX      0M
0012 STASEX      0F
0013 STASEX      0M
0014 STASEX      0M
0015 STASEX      0M
```

0016	STASEX	OF
0017	STASEX	OM
0018	STASEX	OF
0019	STASEX	OM
0020	STASEX	OF
0021	STASEX	OM
0022	STASEX	OF
0023	STASEX	OM
0024	STASEX	OM
0025	STASEX	OM
0026	STASEX	OF
0027	STASEX	OF
0028	STASEX	OM
0029	STASEX	OM
0030	STASEX	OM
0031	STASEX	OM
0032	STASEX	OF
0033	STASEX	OM
0034	STASEX	OF
0035	STASEX	OF
0036	STASEX	OM
0037	STASEX	OF
0038	STASEX	OM
0039	STASEX	OM
0040	STASEX	OF
0041	STASEX	OM
0042	STASEX	OF
0043	STASEX	OM
0044	STASEX	OM
0045	STASEX	OM
0046	STASEX	OF
0047	STASEX	OF
0048	STASEX	OF
0049	STASEX	OF
0050	STASEX	OF
0051	STASEX	OM
0052	STASEX	OM
0053	STASEX	OF
0054	STASEX	OM
0055	STASEX	OF
0056	STASEX	OM
0057	STASEX	OF
0058	STASEX	OM

0059	STASEX	OM
0060	STASEX	OF
0061	STASEX	OM
0062	STASEX	OM
0063	STASEX	OF
0064	STASEX	OM
0065	STASEX	OM
0066	STASEX	OM
0067	STASEX	OF
0068	STASEX	OM
0069	STASEX	OM
0000	LOCA	SOUTH
0001	LOCA	WEST
0002	LOCA	EAST
0003	LOCA	EAST
0004	LOCA	NORTH
0005	LOCA	CENTRL
0006	LOCA	NORTH
0007	LOCA	EAST
0008	LOCA	SOUTH
0009	LOCA	EAST
0010	LOCA	CENTRL
0011	LOCA	NORTH
0012	LOCA	WEST
0013	LOCA	EAST
0014	LOCA	EAST
0015	LOCA	SOUTH
0016	LOCA	SOUTH
0017	LOCA	WEST
0018	LOCA	NORTH
0019	LOCA	EAST
0020	LOCA	CENTRL
0021	LOCA	NORTH
0022	LOCA	SOUTH
0023	LOCA	SOUTH
0024	LOCA	NORTH
0025	LOCA	NORTH
0026	LOCA	NORTH
0027	LOCA	WEST
0028	LOCA	WEST
0029	LOCA	CENTRL
0030	LOCA	SOUTH
0031	LOCA	WEST

```
0032 LOCA          NORTH
0033 LOCA          CENTRL
0034 LOCA          SOUTH
0035 LOCA          WEST
0036 LOCA          EAST
0037 LOCA          CENTRL
0038 LOCA          SOUTH
0039 LOCA          WEST
0040 LOCA          NORTH
0041 LOCA          EAST
0042 LOCA          EAST
0043 LOCA          EAST
0044 LOCA          CENTRL
0045 LOCA          EAST
0046 LOCA          EAST
0047 LOCA          WEST
0048 LOCA          SOUTH
0049 LOCA          EAST
0050 LOCA          WEST
0051 LOCA          EAST
0052 LOCA          SOUTH
0053 LOCA          NORTH
0054 LOCA          WEST
0055 LOCA          EAST
0056 LOCA          EAST
0057 LOCA          WEST
0058 LOCA          WEST
0059 LOCA          NORTH
0060 LOCA          SOUTH
0061 LOCA          WEST
0062 LOCA          EAST
0063 LOCA          EAST
0064 LOCA          EAST
0065 LOCA          NORTH
0066 LOCA          CENTRL
0067 LOCA          EAST
0068 LOCA          WEST
0069 LOCA          NORTH
END

YYYYYY777777
```

A direct reproduction of the computer output follows. An examination of the output reveals that the number of differences found between the expected and final results is none and that the program operated as was expected.

```
ADMIT 1      ELEANOR HURD  SAMPLE COLLEGE ADMISSION PROGRAM          JCY000    IMT001    02/01/60

INTERPRETER FIRST PASS

JCY000
START    ADMIT  E HURD  SAMPLE PROGRAM

ASSEMBLE BABY COMPOOL

DATA SIMULATION PROGRAM

JSTRZ HAS FINISHED PROCESSING THE TABLE AND ITEM ENVIRONMENT FOR THIS TEST

INTERPRETER SECOND PASS

DATA PROCESSING PROGRAM
```

ADMIT 1

TABLE-1TABLE TABTYP-FIXED WDSTRY NENTRY 1 ABSADD 77777 SETUSE-NULL VARDEF-PRGRAM

TABLE-APP1 TABTYP-FIXED WDSTRY 6 NENTRY 70 ABSADD 77132 SETUSE-NULL VARDEF-MSTCMP

ITEM-LOCA ITCODE-STATUS NSTATI- 5 BELOW. ABSADD 77563 SETUSE-NULL VARDEF-MSTCMP

	NORTH			EAST			SOUTH			WEST			CENTRL	
ENT	INIT/L	FINAL	ENT	INIT/L	FINAL	ENT	INIT/L	FINAL	ENT	INIT/L	FINAL	ENT	INIT/L	FINAL
0	SOUTH	******	18	NORTH	******	36	EAST	******	53	NORTH	******			
1	WEST	******	19	EAST	******	37	CENTRL	******	54	WEST	******			
2	EAST	******	20	CENTRL	******	38	SOUTH	******	55	EAST	******			
3	EAST	******	21	NORTH	******	39	WEST	******	56	EAST	******			
4	NORTH	******	22	SOUTH	******	40	NORTH	******	57	WEST	******			
5	CENTRL	******	23	SOUTH	******	41	EAST	******	58	WEST	******			
6	NORTH	******	24	NORTH	******	42	EAST	******	59	NORTH	******			
7	EAST	******	25	NORTH	******	43	EAST	******	60	SOUTH	******			
8	SOUTH	******	26	NORTH	******	44	CENTRL	******	61	WEST	******			
9	EAST	******	27	WEST	******	45	EAST	******	62	EAST	******			
10	CENTRL	******	28	WEST	******	46	EAST	******	63	EAST	******			
11	NORTH	******	29	CENTRL	******	47	WEST	******	64	EAST	******			
12	WEST	******	30	SOUTH	******	48	SOUTH	******	65	NORTH	******			
13	EAST	******	31	WEST	******	49	EAST	******	66	CENTRL	******			
14	EAST	******	32	NORTH	******	50	WEST	******	67	EAST	******			
15	SOUTH	******	33	CENTRL	******	51	EAST	******	68	WEST	******			
16	SOUTH	******	34	SOUTH	******	52	SOUTH	******	69	NORTH	******			
17	WEST	******	35	WEST	******									

ITEM-NAME ITCODE-BH NBITS 36 ABSADD 77133 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L	FINAL	ENT	INIT/L	FINAL	ENT	INIT/L	FINAL	ENT	INIT/L	FINAL
0	OAARON	******	18	ODODGE	******	36	HANDEL	******	53	KONRAD	******
1	ADRIAN	******	19	OODREW	******	37	HANSON	******	54	OOLADD	******
2	OAHERN	******	20	ELLIOT	******	38	OOHILL	******	55	LATONA	******
3	OALDEN	******	21	EMMONS	******	39	OHOGEN	******	56	LITTLE	******
4	OALLEN	******	22	EUSNER	******	40	HUGHES	******	57	MACRAE	******
5	ANDERS	******	23	OEVANS	******	41	HUGGIN	******	58	MANLEY	******
6	ARMOUR	******	24	OOEWEN	******	42	IBISCH	******	59	OMASON	******
7	BARKER	******	25	EZZELL	******	43	OIBSEN	******	60	OMCCUE	******
8	BARRON	******	26	FABIAN	******	44	OIDIOT	******	61	NORRIS	******
9	BARTON	******	27	FADDEN	******	45	JARVIS	******	62	ONEILL	******
10	OOBELL	******	28	OFREED	******	46	OOOJAY	******	63	PALMER	******
11	OBROWN	******	29	OFREUD	******	47	JENSEN	******	64	OOPAPP	******
12	CARSON	******	30	OGATES	******	48	OJONES	******	65	PARKER	******
13	OCHING	******	31	GERBER	******	49	JORDAN	******	66	OPARMI	******
14	OCHOMA	******	32	GIDEON	******	50	OOKATZ	******	67	PELZER	******
15	COLLIN	******	33	GORDON	******	51	OKELLY	******	68	RAMSEY	******
16	DEITCH	******	34	OGOULD	******	52	OOKERR	******	69	REISCH	******
17	DEMONE	******	35	HAMMER	******						

ITEM-SCOREA ITCODE-FI UNSIGNED NBITS- 7 ABSADD 77241 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+02	+6	******			
1	+02	+7	******			
2	+02	+9	******			
3	+02	+85	******			
4	+02	+6	******			

5	+02	+7	******
6	+02	+7	******
7	+02	+4	******
8	+02	+8	******
9	+02	+9	******
10	+02	+8	******
11	+02	+7	******
12	+02	+75	******
13	+02	+6	******
14	+02	+6	******
15	+02	+8	******
16	+02	+85	******
17	+02	+75	******
18	+02	+9	******
19	+02	+8	******
20	+02	+7	******
21	+02	+8	******
22	+02	+85	******
23	+02	+9	******
24	+02	+85	******
25	+02	+7	******
26	+02	+75	******
27	+02	+8	******
28	+02	+85	******
29	+02	+95	******
30	+02	+85	******
31	+02	+8	******
32	+02	+7	******
33	+02	+8	******
34	+02	+6	******
35	+02	+8	******
36	+02	+85	******
37	+02	+8	******
38	+02	+85	******
39	+02	+7	******
40	+02	+8	******
41	+02	+85	******
42	+02	+75	******
43	+02	+95	******
44	+02	+4	******
45	+02	+85	******
46	+02	+9	******
47	+02	+85	******
48	+02	+7	******
49	+02	+8	******
50	+02	+75	******
51	+02	+85	******
52	+02	+7	******
53	+02	+85	******
54	+02	+85	******
55	+02	+9	******
56	+02	+7	******
57	+02	+85	******
58	+02	+8	******
59	+02	+6	******
60	+02	+5	******
61	+02	+8	******
62	+02	+85	******
63	+02	+8	******

FN-LO-197
1 February 1960
Page 163

64	+02	+7	******
65	+02	+8	******
66	+02	+85	******
67	+02	+9	******
68	+02	+8	******
69	+02	+85	******

ITEM-SCOREB ITCODE-FI UNSIGNED NBITS- 7 ABSADD 77347 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+02	+7	******			
1	+02	+8	******			
2	+02	+9	******			
3	+02	+8	******			
4	+02	+55	******			
5	+02	+7	******			
6	+02	+7	******			
7	+02	+8	******			
8	+02	+85	******			
9	+02	+95	******			
10	+02	+8	******			
11	+02	+7	******			
12	+02	+75	******			
13	+02	+8	******			
14	+02	+45	******			
15	+02	+8	******			
16	+02	+95	******			
17	+02	+8	******			
18	+02	+9	******			
19	+02	+85	******			
20	+02	+75	******			
21	+02	+85	******			
22	+02	+95	******			
23	+02	+9	******			
24	+02	+85	******			
25	+02	+7	******			
26	+02	+9	******			
27	+02	+8	******			
28	+02	+75	******			
29	+02	+95	******			
30	+02	+8	******			
31	+02	+85	******			
32	+02	+7	******			
33	+02	+8	******			
34	+02	+6	******			
35	+02	+85	******			
36	+02	+9	******			
37	+02	+8	******			
38	+02	+85	******			
39	+02	+7	******			
40	+02	+8	******			
41	+02	+9	******			
42	+02	+7	******			
43	+02	+95	******			
44	+02	+3	******			
45	+02	+85	******			
46	+02	+9	******			

FN-10-197
1 February 1960
Page 164

47	+02	+8	******
48	+02	+7	******
49	+02	+8	******
50	+02	+9	******
51	+02	+85	******
52	+02	+7	******
53	+02	+85	******
54	+02	+8	******
55	+02	+8	******
56	+02	+7	******
57	+02	+85	******
58	+02	+8	******
59	+02	+6	******
60	+02	+6	******
61	+02	+8	******
62	+02	+9	******
63	+02	+9	******
64	+02	+7	******
65	+02	+8	******
66	+02	+7	******
67	+02	+9	******
68	+02	+8	******
69	+02	+85	******

ITEM-SCOREC ITCODE-FI UNSIGNED MBITS- 7 ABSADD 77455 SETUSE-USED VARDEF-MSTCMP

ENT	INIT/L XPON	INIT/L FRACTION	FINAL XPON	FINAL FRACTION	XPECTD XPON	XPECTD FRACTION
0	+02	+8	******			
1	+02	+8	******			
2	+02	+9	******			
3	+02	+9	******			
4	+02	+7	******			
5	+02	+7	******			
6	+02	+65	******			
7	+02	+9	******			
8	+02	+85	******			
9	+02	+8	******			
10	+02	+85	******			
11	+02	+75	******			
12	+02	+75	******			
13	+02	+9	******			
14	+02	+7	******			
15	+02	+6	******			
16	+02	+85	******			
17	+02	+9	******			
18	+02	+95	******			
19	+02	+8	******			
20	+02	+75	******			
21	+02	+85	******			
22	+02	+9	******			
23	+02	+8	******			
24	+02	+85	******			
25	+02	+7	******			
26	+02	+9	******			
27	+02	+8	******			
28	+02	+45	******			
29	+02	+95	******			

30	+02	+8	******
31	+02	+85	******
32	+02	+7	******
33	+02	+8	******
34	+02	+6	******
35	+02	+95	******
36	+02	+85	******
37	+02	+8	******
38	+02	+85	******
39	+02	+75	******
40	+02	+9	******
41	+02	+85	******
42	+02	+7	******
43	+02	+9	******
44	+02	+3	******
45	+02	+85	******
46	+02	+9	******
47	+02	+8	******
48	+02	+75	******
49	+02	+8	******
50	+02	+85	******
51	+02	+9	******
52	+02	+85	******
53	+02	+8	******
54	+02	+8	******
55	+02	+8	******
56	+02	+7	******
57	+02	+85	******
58	+02	+9	******
59	+02	+7	******
60	+02	+8	******
61	+02	+8	******
62	+02	+9	******
63	+02	+9	******
64	+02	+7	******
65	+02	+8	******
66	+02	+75	******
67	+02	+95	******
68	+02	+85	******
69	+02	+85	******

ITEM-STASEX ITCODE-BH NBITS 12 ABSADD 77671 SETUSE-BOTH VARDEF-MSTCMP

ENT	INIT/L	FINAL	XPECTD	ENT	INIT/L	FINAL	XPECTD	ENT	INIT/L	FINAL	XPECTD
0	00000M	******	******	24	00000M	0000AM	******	47	00000F	******	******
1	00000M	0000AM	******	25	00000M	******	******	48	00000F	******	******
2	00000M	0000AM	******	26	00000F	0000AF	******	49	00000F	******	******
3	00000F	0000AF	******	27	00000F	0000AF	******	50	00000F	******	******
4	00000F	******	******	28	00000M	******	******	51	00000M	0000AM	******
5	00000M	******	******	29	00000M	0000AM	******	52	00000M	******	******
6	00000F	******	******	30	00000M	******	******	53	00000F	******	******
7	00000F	******	******	31	00000M	0000AM	******	54	00000M	******	******
8	00000M	0000AM	******	32	00000F	******	******	55	00000F	******	******
9	00000F	0000AF	******	33	00000M	0000AM	******	56	00000M	******	******
10	00000F	0000AF	******	34	00000F	******	******	57	00000F	******	******
11	00000M	******	******	35	00000F	0000AF	******	58	00000M	******	******
12	00000F	0000AF	******	36	00000M	0000AM	******	59	00000M	******	******
13	00000M	0000AM	******	37	00000F	******	******	60	00000F	******	******

ADMIT 1

14	00000M	******	******	38	00000M	******	******	61	00000M	******	******
15	00000M	******	******	39	00000M	******	******	62	00000M	0000AM	******
16	00000F	0000AF	******	40	00000F	0000AF	******	63	00000F	******	******
17	00000M	0000AM	******	41	00000M	0000AM	******	64	00000M	******	******
18	00000F	0000AF	******	42	00000F	******	******	65	00000M	0000AM	******
19	00000M	0000AM	******	43	00000M	0000AM	******	66	00000M	******	******
20	00000F	******	******	44	00000M	******	******	67	00000F	******	******
21	00000M	0000AM	******	45	00000M	0000AM	******	68	00000M	******	******
22	00000F	0000AF	******	46	00000F	0000AF	******	69	00000M	******	******
23	00000M	******	******								

TABLE-TOT1 TABTYP-FIXED WDSTRY 1 NENTRY 5 ABSADD 77124 SETUSE-NULL VARDEF-MSTCMP

ITEM-AREA ITCODE-FI UNSIGNED NBITS- 7 ABSADD 77125 SETUSE-BOTH VARDEF-MSTCMP

ENT	INIT/L		FINAL		XPECID	
	XPON	FRACTION	XPON	FRACTION	XPON	FRACTION
0		0	+01	+6	******	
1		0	+02	+12	******	
2		0	+01	+3	******	
3		0	+01	+6	******	
4		0	+01	+3	******	

TABLE-TOT2 TABTYP-FIXED WDSTRY 2 NENTRY 1 ABSADD 77121 SETUSE-NULL VARDEF-MSTCMP

ITEM-NUMBF ITCODE-FI UNSIGNED NBITS- 5 ABSADD 77123 SETUSE-BOTH VARDEF-MSTCMP

ENT	INIT/L		FINAL		XPECTD	
	XPON	FRACTION	XPON	FRACTION	XPON	FRACTION
0		0	+02	+12	******	

ITEM-NUMBM ITCODE-FI UNSIGNED NBITS- 5 ABSADD 77122 SETUSE-BOTH VARDEF-MSTCMP

ENT	INIT/L		FINAL		XPECTD	
	XPON	FRACTION	XPON	FRACTION	XPON	FRACTION
0		0	+02	+18	******	

THE NUMBER OF DIFFERENCES FOUND ARE NONE

COMREG 46 CON 29 SUB 33 VAT 53 SWT 11 ILT 301 SLT 71 TABRG 431

NEW STATEMENT	ILT ENTRY	LEFT HALF TERM	OPERATION	RIGHT HALF TERM	ADDRESSE (S)
*	0	SUBSCRIPT J	SET TO	CONST 0	
*	1	SUBSCRIPT I	SET TO	CONST 0	
*	2	SCOREA SUB I	LS	CONST +02 +5	TRUE 5 FALSE 3
	3	SCOREB SUB I	LS	CONST +02 +5	TRUE 5 FALSE 4
	4	SCOREC SUB I	LS	CONST +02 +5	TRUE 5 FALSE 6
*	5		GOTO		ILT NO 53
*	6	SCOREA SUB I	PLUS	SCOREB SUB I	TEMP 1
	7	TEMP 1	PLUS	SCOREC SUB I	TEMP 1
	8	TEMP 1	DIV	CONST +01 +3	TEMP 1
	9	TEMP 1	LS	CONST +02 +75	TRUE 10 FALSE 11
*	10		GOTO		ILT NO 53
*	11		GOTO		ILT NO 13
*	12		GOTO		ILT NO 53
DISTRB	13	ENTRIES SWT 5 STAT SW DECL	LOCA		DISTRB
A1	14	AREA SUB +00000	EQ	CONST +01 +6	TRUE 15 FALSE 16
*	15		GOTO		ILT NO 53
*	16	SUBSCRIPT J	SET TO	CONST 0	
*	17		GOTO		ILT NO 33
A2	18	AREA SUB +00001	EQ	CONST +02 +12	TRUE 19 FALSE 20
*	19		GOTO		ILT NO 53
*	20	SUBSCRIPT J	SET TO	CONST +01 +1	
*	21		GOTO		ILT NO 33
A3	22	AREA SUB +00002	EQ	CONST +01 +3	TRUE 23 FALSE 24
*	23		GOTO		ILT NO 53
*	24	SUBSCRIPT J	SET TO	CONST +01 +2	
*	25		GOTO		ILT NO 33
A4	26	AREA SUB +00003	EQ	CONST +01 +6	TRUE 27 FALSE 28
*	27		GOTO		ILT NO 53

NEW STATEMENT	ILT ENTRY	LEFT HALF TERM	OPERATION	RIGHT HALF TERM	ADDRESSE (S)
*	28	SUBSCRIPT J	SET TO	CONST +01 +3	
*	29		GOTO		ILT NO 33
A5	30	AREA SUB +00004	EQ	CONST +01 +3	TRUE 31 FALSE 32
*	31		GOTO		ILT NO 53
*	32	SUBSCRIPT J	SET TO	CONST +01 +4	
A6	33	(BYTE)STASEX SUB I	EQ	CONST 00000M	TRUE 35 FALSE 36
	34	(BYTE) SUB +00001,SUB +00001			
	35	NUMBM	EQ	CONST +02 +18	TRUE 39 FALSE 36
	36	(BYTE)STASEX SUB I	EQ	CONST 00000F	TRUE 38 FALSE 40
	37	(BYTE) SUB +00001,SUB +00001			
	38	NUMBF	EQ	CONST +02 +12	TRUE 39 FALSE 40
*	39		GOTO		ILT NO 53
*	40	(BYTE)STASEX SUB I	EQ	CONST 00000M	TRUE 42 FALSE 45
	41	(BYTE) SUB +00001,SUB +00001			
*	42	NUMBM	PLUS	CONST +01 +1	TEMP 1
	43	NUMBM	SET TO	TEMP 1	
*	44		GOTO		ILT NO 47
*	45	NUMBF	PLUS	CONST +01 +1	TEMP 1
	46	NUMBF	SET TO	TEMP 1	
A7	47	AREA SUB J	PLUS	CONST +01 +1	TEMP 1
	48	AREA SUB J	SET TO	TEMP 1	
*	49	(BYTE)STASEX SUB I	SET TO	CONST 00000A	
	50	(BYTE) SUB +00000,SUB +00001			
*	51	NUMBF	PLUS	NUMBM	TEMP 1
	52	TEMP 1	NQ	CONST +02 +3	TRUE 53 FALSE 59
*	53	(MENT)NAME	MINUS	CONST +01 +1	TEMP 1
	54	SUBSCRIPT I	EQ	TEMP 1	TRUE 58 FALSE 55
	55	SUBSCRIPT I	PLUS	CONST +01 +1	TEMP 1

NEW STATEMENT	ILT ENTRY	LEFT HALF TERM	OPERATION	RIGHT HALF TERM	ADDRESSE (S)
	56	SUBSCRIPT I	SET TO	TEMP 1	
	57		GOTO	ILT NO	2
*	58		STOP	ILT NO	0
*	59		TERM	ILT NO	0

JOSYZ HAS DECODED ILT.

TEST COMPLETE

TEST CONTROL RUN COMPLETE

TAPE DISPOSITION

SAVE MASTER TAPE DRIVE A1

PRINT OFF LINE DRIVE A2

PRINT OFF LINE DRIVE B1

DISPOSE OF C1, D1, D2

Distribution:

SDC (Lodi)

SACCS Division Staff (1 ea.)
Programming Branch Staff (1 ea.)
Program Production Group (1 ea.)
Program Design Group - M. Mineart (20)
Program Requirements Group - F. Diaz (5)
CUSS Project - J. I. Schwartz (10)

SDC (Santa Monica)

J. D. Madden
R. Bosak
J. Matousek
B. Morriss
B. Dobbs (10)
E. Gordon
C. M. Lawson
D. E. Henley
G. Jacobs

IEC

Standard Distribution (35)