MEMORANDUM
RM-5689-PR
AUGUST 1968

PROGRAMMING BY QUESTIONNAIRE: AUXILIARY PROGRAMS

Patricia L. Love and Paula M. Oldfather

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5689-PR
AUGUST 1968

PROGRAMMING BY QUESTIONNAIRE: AUXILIARY PROGRAMS

Patricia L. Love and Paula M. Oldfather

This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0045—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff. Research and Development, Hq USAF. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of the United States Air Force.

DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

Copyright © 1968
THE RAND CORPORATION

This Rand study is presented as a competent treatment of the subject, worthy of publication. The Rand Corporation vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the authors.

Published by The RAND Corporation

PREFACE

The RAND Corporation has played an active role in the use of digital computers for simulation models. As an outgrowth of this work, a technique was developed in the Logistics Department for greatly reducing the time and effort required to prepare large computer programs. The technique, called Programming by Questionnaire, was first described in A. S. Ginsberg, H. M. Markowitz, and P. M. Oldfather, Programming by Questionnaire, The RAND Corporation, RM-4460-PR, April 1965. Two subsequent Memoranda* have reported on the implementation of the technique and on the first application, the Job Shop Simulation Program Generator (JSSPG).** A familiarity with these preceding Memoranda is essential for an understanding of the present paper.

This Memorandum describes five auxiliary computer programs that were developed in conjunction with Programming by Questionnaire and the JSSPG that greatly facilitate the use of the technique and the analysis of the results from a JSSPG program. Three of the programs are general purpose and may be used with any application of Programming by Questionnaire; two programs are specific to the JSSPG.

P. M. Oldfather is a consultant to The RAND Corporation.

---

*P. M. Oldfather, A. S. Ginsberg, and H. M. Markowitz, Programming by Questionnaire: How to Construct a Program Generator, The RAND Corporation, RM-5129-PR, November 1966.

**P. M. Oldfather, A. S. Ginsberg, P. L. Love, and H. M. Markowitz, Programming by Questionnaire: The Job Shop Simulation Program Generator, The RAND Corporation, RM-5162-PR, July 1967.

SUMMARY

Programming by Questionnaire is a method by which many computer programs can be produced with a considerable saving of time, effort, and cost. The general concepts of the technique were discussed previously by A. S. Ginsberg, H. M. Markowitz, and P. M. Oldfather, Programming by Questionnaire, The RAND Corporation, RM-4460-PR, April 1965. Since that time, two other Memoranda have been written on the technique. The first presented the details of the technique; the second discussed its first application, the Job Shop Simulation Program Generator (JSSPG). The present Memorandum complements these two by presenting five small computer programs that facilitate the application of Programming by Questionnaire in general, and provide additional analysis of the JSSPG programs.

Three programs aid in the development of a program generator. The CHKOUT program checks the Generator Deck for logical consistency and provides other information essential for correcting and maintaining the deck. The LOCATE program finds specified cards within the deck, which is very large. Finally, the KORCT program makes corrections to the deck and checks to see that it is in order.

Two programs analyze the results of a JSSPG simulator. The JHIST program provides a complete history of the jobs in the simulated job shop, and the RHIST program provides a history of the resources.

CONTENTS

PREFACE iii

SUMMARY v

Section
I. INTRODUCTION 1

II. GENERATOR DECK PROGRAMS 3
Contents and Organization of the Generator Deck 3
Sequencing 3
Statement List and Decision Tables 4
Blocking 4
Editor Control Cards 5
The CHKOUT Program 6
Block Sizes 7
Deck Order Errors 7
Identification Numbers 9
Block-by-Block Information 11
Replaceable Common Text Blocks 11
Entities Blocks 13
Transient Blocks 13
Summary Information 17
The LOCATE Program 20
The KORCT Program 22

III. JSSPG ANALYSIS PROGRAMS 25
Debugging the JSSPG 25
The JSSPG Analysis Tape 26
The JHIST Program 26
The RHIST Program 30

Appendix
A. THE CHKOUT PROGRAM 33
B. THE LOCATE PROGRAM 45
C. THE KORCT PROGRAM 51
D. THE JHIST PROGRAM 55
E. THE RHIST PROGRAM 62

FIGURES

1. CHKOUT Output: Statement List Block Sizes 8

2. CHKOUT Output: Identification Number Classes 10

3. CHKOUT Error Messages for a Replaceable Common Text Block ... 12

4. CHKOUT Output for a Replaceable Common Text Block 12

5. CHKOUT Output for an Entities Block 14

6. CHKOUT Output for a Transient Block 15

7. CHKOUT Summary Information: Transient Identification Numbers and Z Questions 18

8. CHKOUT Summary Information: Identification Number Usage 19

9. Sample Deck for the LOCATE Program Showing the Use of the Decision Table Coding Form 21

10. Printouts from Two KORCT Runs Showing All the Messages..... 23

11. Contents of the JSSPG Analysis Tape 27

12. Sample JHIST Output 29

13. Sample RHIST Output 31

TABLES

1. W. Card Codes 6

2. CHKOUT System Variables 43

3. LOCATE System Variables 50

4. JHIST System Variables 60

5. RHIST System Variables 70

I. INTRODUCTION

The time and effort required to produce a computer program have often obstructed the practical application of computers. The development of advanced programming languages, however, has considerably reduced programming time and effort. Programming by Questionnaire further reduces the effort required to produce large computer programs within specified areas.

A program generator consists of four components:

1. A Questionnaire, written in English, defining the scope and logic of all the programs to be generated.
2. A Statement List, containing all the computer commands needed to construct any of the many programs.
3. A set of Decision Tables, specifying the commands required from the Statement List as a function of the Questionnaire choices.
4. The Editor program, for processing the Questionnaire, Statement List, and Decision Tables, thus building the desired program.

Of these four components, only the Editor is general purpose; the other three components are specific to a given area of application. Application areas and the basic concepts of Programming by Questionnaire were discussed in the first Memorandum* of this series and will not be presented here. Likewise, the details of the technique, the operation of the Editor program, and the construction of the other three components were specified in the second Memorandum.** The third Memorandum*** presented the Job Shop Simulation Program Generator (JSSPG), the first application of Programming by Questionnaire.

---

*A. S. Ginsberg, H. M. Markowitz, and P. M. Oldfather, Programming by Questionnaire, The RAND Corporation, RM-4460-PR, April 1965.

**P. M. Oldfather, A. S. Ginsberg, and H. M. Markowitz, Programming by Questionnaire: How to Construct a Program Generator, The RAND Corporation, RM-5129-PR, November 1966.

***P. M. Oldfather, A. S. Ginsberg, P. L. Love, and H. M. Markowitz, Programming by Questionnaire: The Job Shop Simulation Program Generator, The RAND Corporation, RM-5162-PR, July 1967.

This, the fourth and final Memorandum of the series, complements the last two by presenting five small computer programs that facilitate the application of Programming by Questionnaire and the analysis of the JSSPG programs.

Section II describes the three programs that aid in developing a program generator, regardless of its application area. These programs, like the Editor program, are general purpose. Section III presents two programs that are specific to the JSSPG application, providing further analysis of the output of the simulator.

Complete listings of the programs are contained in the five Appendixes.

II. GENERATOR DECK PROGRAMS

The Generator Deck of a program generator consists of the Statement List, Decision Tables, and the control cards for the Editor program. In any application of Programming by Questionnaire, this deck is very large; in the Job Shop Simulation Program Generator (JSSPG), it contained about 10,000 cards. Making corrections, locating a given card, and keeping such an enormous deck in order are major problems. The three programs presented in this section were designed to facilitate these operations. The CHKOUT program checks the Generator Deck for logical consistency and provides other information essential for correcting, adding to, and generating programs with the deck, such as block size and available Identification Numbers. The LOCATE program finds all occurrences of a specified card. The KORCT program makes additions, deletions, and replacements in the deck while checking its sequence order. Both CHKOUT and KORCT require that the Generator Deck be on a magnetic tape.

CONTENT AND ORGANIZATION OF THE GENERATOR DECK

The content and organization of the Generator Deck were described in detail in RM-5129-PR.* Only a summary of the characteristics will be presented here.

Sequencing

The Generator Deck consists of the Statement List, Decision Tables, and Editor control cards. The first 76 columns of most of these cards are processed by the Editor, leaving only four columns, 77-80, for deck sequencing. Consequently, alphabetic sequencing is used instead of numeric, allowing over 500,000 sequence numbers instead of less than 10,000. To allow space for insertions in the JSSPG Generator Deck, initial sequence numbers were assigned that consisted of three alphabetic characters and a zero. The sequence checking program, KORCT, places the zero at the top of the collating sequence. However, a zero is allowed only in the last column; all other characters must be alphabetic.

---

*Oldfather, Ginsberg, and Markowitz, op. cit., pp. 11-27.

Statement List and Decision Tables

Cards in the Statement List contain programming statements in the first 72 columns and Identification Numbers in columns 73-76. The Identification Numbers are used in the Decision Tables to select statements or groups of statements from the Statement List. Identification Numbers are numeric and range from 1 to 999, although 9999 is used on one of the Editor control cards. Identification Numbers must always be unique within a Statement List block.

Decision Tables are divided into Conditions and Actions. The Conditions reference questions on the Questionnaire; the Actions reference statements in the Statement List by their Identification Numbers. Questions are referenced by a letter, denoting the Questionnaire section (ranging from A to F in the JSSPG), and a number from 1 to 72. Questions beginning with the letter Z are reserved for internal use; i.e., the Questionnaire may not contain a section Z. Z questions are used to reflect previous actions in Decision Tables and may appear as both Conditions and Actions. However, a Z question should not be "queried" (appear as a Condition) before it is "answered" (appears in a preceding Decision Table as an Action).

Blocking

The Statement List is too large to be contained in available core storage as a whole. Hence, it is broken into blocks, of which there are four types. "Permanent Common Text" (PCT) contains those cards that may be referenced by any Decision Table in the Generator Deck. It is never overlaid by a subsequent block, and the Identification Numbers assigned to cards in the block may not be used in any other blocks.

"Replaceable Common Text" (RCT) contains cards that are referenced by more than one subsequent block of Decision Tables. These cards remain in memory until replaced by another block of RCT. Identification Numbers used in an RCT block may be reused only in subsequent RCT blocks.

The remainder of Statement List storage is used for blocks that remain in memory only for a short time. These blocks are of two kinds. Each "Transient" block has a block of Decision Tables associated with it, and may be referenced only by these Decision Tables. Identification Numbers in these blocks may be reused only in other Transient blocks. The "Entities" blocks are logically part of RCT in that they are referenced by a number of Decision Table blocks, but they occupy the transient area of storage. Decision Tables that reference an Entities block precede the block; an Entities block has no associated Decision Table block. Identification Number restrictions are the same as for RCT.

As mentioned above, Decision Tables are broken into blocks that are associated with Transient Statement List blocks. A Decision Table may reference any statement in PCT, the RCT block that precedes it, the Entities block that follows it, or its associated Transient block.

Editor Control Cards

Control cards are intermingled with Statement List and Decision Table blocks to inform the Editor program when the end of a block has been reached and what type of block it will encounter next. All Statement List blocks, regardless of their type, are terminated by a card with an Identification Number of 9999. The remainder of the card, columns 1-72, may be used for comments, since it is not processed.

The Editor also recognizes four types of "Signal" cards. Signal cards have an alphabetic code of T, U, E, or W in column 1, and the W card has a numeric code in column 6. The remaining columns of the Signal cards may be used for comments. The T, U, and E cards occur once per Decision Table. The T card indicates the beginning of a table, the U card separates the Conditions and the Actions, and the E card marks the end of the table. A W card marks the end of a Decision Table block, and the numeric code indicates the type of Statement List block that will appear next. A W card also appears immediately

after the PCT block, informing the Editor what kind of program (simulation or nonsimulation) will be generated and what type of Statement List block will follow. The last card in the Generator Deck is also a W card. The W card codes are given in Table 1.

Table 1

W CARD CODES

Code	Explanation
1	Simulation program to be generated; the next block is RCT.
2	Simulation program to be generated; the next block is Transient.
3	Nonsimulation program to be generated; the next block is RCT.
4	Nonsimulation program to be generated; the next block is Transient.
5	The next block is Entities.
6	The next block is Transient and contains the program's initialization; all routines for this program have been generated.
7	The next block is RCT.
8	The next block is Transient.
9	End of the Generator Deck.

THE CHKOUT PROGRAM

The CHKOUT program facilitates the development of a program generator. It partially debugs the Generator Deck and provides information required for making changes to the deck. Debugging includes catching deck-order errors, finding duplicated, missing, or unreferenced Identification Numbers, and locating Z questions that were incorrectly used. Additionally, CHKOUT produces lists of Identification Numbers used and available in each block, and a list of the block sizes with a warning as a block approaches overflow. CHKOUT passes the Generator Deck twice; consequently, the deck must be on a magnetic tape, and is mounted on FORTRAN logical unit 9.

A listing of the CHKOUT program appears in Appendix A. CHKOUT is written in SIMSCRIPT with a few FORTRAN statements inserted. The use and initialization of each of the SIMSCRIPT arrays are included

in the appendix following the program listing. Some of the arrays are also discussed below.

Block Sizes

On its first pass through the Generator Deck, CHKOUT produces block size information as shown in Fig. 1,* identifying each block of the Statement List by its type and starting sequence number, and assigning numbers to the Transient blocks. Arrays 12 and 13 in CHKOUT relate to block size. Array 12, MAX, is initialized to the total Statement List storage available in the Editor program. Array 13, CNTRL, is used to produce warning messages, as shown following block 13 in Fig. 1. When the available storage space will contain fewer additional cards than the value of CNTRL, the warning message is produced. For the JSSPG, CNTRL was set at 50 cards.

The Editor stores a complete Statement List block at a time, but only one Decision Table is in memory at a time. Consequently, the length of a Decision Table block is not important, but the sizes of the largest Conditions portion and Actions portion are. CHKOUT determines these sizes and outputs them with the sequence numbers of the Decision Tables in which the maxima were found (see Fig. 1). These sizes should be checked against the Editor initialization to assure that a Decision Table will not overflow its storage.

Deck-Order Errors

Three possible error conditions will cause CHKOUT to fail to complete the first pass of the deck. These errors all relate to the order of the Generator Deck.

Immediately following the 9999 card signalling the end of PCT, there should be a W card with a code of 1, 2, 3, or 4. If the W card is missing or if the code is greater than 4, CHKOUT will produce an error message and terminate execution. The other type of error occurs in a Decision Table block. Following an E card (which signals

*All examples are drawn from the JSSPG.

```
ERROR       272 IS A DUPLICATE WITHIN PERMANENT COMMON TEXT - SEQUENCE AAHM

THERE ARE 15 CARDS IN PERMANENT COMMON TEXT

ERROR         6 MAY NOT BE USED IN COMMON TEXT OR ENTITIES - SEQUENCE ABKR

    THERE ARE  35 CARDS IN REPLACEABLE COMMON TEXT BEGINNING AT SEQUENCE AAOO
       THERE ARE 106 CARDS IN BLOCK  1 STARTING AT SEQUENCE ABOO
       THERE ARE 116 CARDS IN BLOCK  2 STARTING AT SEQUENCE ALVO
       THERE ARE 185 CARDS IN BLOCK  3 STARTING AT SEQUENCE AXLO
       THERE ARE 215 CARDS IN BLOCK  4 STARTING AT SEQUENCE BBWO
       THERE ARE 200 CARDS IN BLOCK  5 STARTING AT SEQUENCE BIXA
       THERE ARE 151 CARDS IN BLOCK  6 STARTING AT SEQUENCE BLSA
       THERE ARE  22 CARDS IN BLOCK  7 STARTING AT SEQUENCE BOXF
       THERE ARE 119 CARDS IN BLOCK  8 STARTING AT SEQUENCE BPUO
       THERE ARE 177 CARDS IN BLOCK  9 STARTING AT SEQUENCE BVSO
       THERE ARE 226 CARDS IN BLOCK 10 STARTING AT SEQUENCE CFUO
       THERE ARE 184 CARDS IN BLOCK 11 STARTING AT SEQUENCE CONO
       THERE ARE  21 CARDS IN BLOCK 12 STARTING AT SEQUENCE CWGO
       THERE ARE 258 CARDS IN BLOCK 13 STARTING AT SEQUENCE CZAO
    WARNING...ONLY 42 MORE CARDS IN BLOCK 13

       THERE ARE 237 CARDS IN BLOCK 14 STARTING AT SEQUENCE EFOM
       THERE ARE 194 CARDS IN BLOCK 15 STARTING AT SEQUENCE FLGO
       THERE ARE 247 CARDS IN BLOCK 16 STARTING AT SEQUENCE GDQO
       THERE ARE 171 CARDS IN BLOCK 17 STARTING AT SEQUENCE HFSO
       THERE ARE 188 CARDS IN BLOCK 18 STARTING AT SEQUENCE HXLO
       THERE ARE 172 CARDS IN BLOCK 19 STARTING AT SEQUENCE IYIO
    THERE ARE  91 CARDS IN ENTITIES BLOCK BEGINNING AT SEQUENCE JKKO
       THERE ARE  73 CARDS IN BLOCK 20 STARTING AT SEQUENCE JNVB
    MAXIMUM CONDITIONS =  31 . TABLE BEGINS AT DQBO
    MAXIMUM ACTIONS =  49 . TABLE BEGINS AT FEZO
```

Fig. 1 -- CHKOUT Output: Statement List Block Sizes

the end of a Decision Table), the next card must be either a T card (indicating the start of another Decision Table) or a W card (indicating the end of the Decision Table block and the start of Statement List). If the card is neither a T card nor a W card, CHKOUT produces an error message and terminates execution.

None of these error messages are shown in Fig. 1.

Identification Numbers

Since Identification Numbers are used in Decision Tables to select statements from the Statement List, the same number may not be used on two statements that can be in memory at the same time. Consequently, it is difficult to find an Identification Number that can be used when adding to the Statement List. CHKOUT produces lists of the Identification Numbers that may be used in different blocks of Statement List and produces error messages when numbers are incorrectly used.

As CHKOUT passes the Generator Deck for the first time, it determines the "class" of each Identification Number: PCT, RCT, Entities, Transient, or unused. Identification Numbers 1 through 8 are reserved for internal use by the Editor program and may be used only in Transient blocks. If one of these numbers is used elsewhere, CHKOUT produces an error message like the one shown for card 6 in Fig. 1. Also on the first pass, CHKOUT looks for duplicate numbers within the PCT block. Duplications are reported in error messages like the one for card 272 in Fig. 1. No further processing or checking of PCT is performed on the second pass.

After completing the first pass through the Generator Deck, CHKOUT produces lists of Identification Numbers in all classes except Transient. (Transient numbers are given with the summary information, since adding new Transient blocks will usually require the use of other summary information.) Such lists are shown in Fig. 2. Numbers in the first list, those which have never been used, are intended primarily for use in PCT; however, they may also be used in RCT and Entities blocks if no other numbers are available for these blocks.

```
 THE FOLLOWING NUMBERS HAVE NOT BEEN USED AND ARE AVAILABLE FOR COMMON TEXT OR ENTITIES

 147  150  165  258  259  260  261  262  264  266  267  268  291  292  293  352  355  356  357  358
 359  360  361  363  365  367  368  369  373  374  375  376  377  379  380  400  418  438  454  455
 468  469  470  471  472  473  482  483  484  485  486  487  488  489  490  500  508  511  527  534
 536  550  560  561  573  574  575  580  582  587  588  595  597  599  655  657  659  662  663  664
 665  666  674  675  676  677  678  679  686  687  696  697  699  757  758  798  813  814  815  816
 817  820  822  826  830  832  836  842  843  844  845  846  847  848  849  851  852  853  854  855
 856  857  858  859  860  863  864  865  866  867  868  869  870  871  872  873  874  875  876  877
 878  879  880  881  882  883  884  885  886  887  888  889  890  892  893  894  895  896  897  898
 899  902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920
 921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940
 941  942  943  944  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
 961  962  963  964  965  966  967  968  969  970  971  972  973  974  975  976  977  978  979  980
 981  982  983  984  985  986  988  989  990  991  992  993  994  995  996  997  998 1000

 THE FOLLOWING NUMBERS WERE USED IN PERMANENT COMMON TEXT

  16   32   33  272  329  334  344  346  348  349  350  799  862

 THE FOLLOWING NUMBERS WERE USED IN REPLACEABLE COMMON TEXT BLOCKS

   9   12   21   22   27   30  131  263  323  366  524  556  578  584  591  598  700  800  891  987
 999

 THE FOLLOWING NUMBERS WERE USED IN ENTITIES BLOCKS

 142  143  156  191  192  193  194  195  196  197  316  321  351  408  410  413  424  425  426  427
 428  429  430  431  432  433  434  435  436  437  453  552  554  555  557  562  563  565  571  572
 698  701  702  703  704  705  706  707  711  712  713  714  715  721  722  723  725  732  733  741
 742  751  752  753  762  771  772  773  781  782  791  792  793  801  802  811  812  821

ERROR       581 IS DUPLICATED ACROSS COMMON TEXT AND ENTITIES BLOCKS
```

Fig. 2 -- CHKOUT Output: Identification Number Classes

Block-by-Block Information

The second time CHKOUT processes the Generator Deck, it produces more detailed information on each block and performs more error checking.

Replaceable Common Text Blocks. The Generator Deck may contain more than one block of RCT. Since RCT cards are referenced by Decision Tables that follow the RCT block, most of the information about the RCT block is not output until the next RCT block or the end of the Generator Deck is encountered. However, duplication of Identification Numbers is reported when the block is processed. Figure 3 shows the two types of error messages that can result from incorrect Identification Numbers in an RCT block.

When the next RCT block or the end of the deck is reached, information like that shown in Fig. 4 is produced for the previous RCT block. Note that the block is identified both by the starting sequence number and by the number of the Transient block it precedes.

Identification Numbers appearing in the list of available numbers are those that are in the RCT class but were not used in the RCT block under consideration. Hence, if only one RCT block exists, as is the case with the JSSPG, no numbers will be shown as available for the block, and needed numbers will have to be chosen from those that were never used.

The count of the number of blocks requiring each RCT card is included in case the unused numbers are exhausted. Any RCT card that is required by only two Transient blocks can economically be duplicated in those blocks under a different Identification Number, and the original number thus made available. (Another way of making numbers available will be discussed under Summary Information.) Cards that were never required should be removed from the deck; these cards are indicated by both a count of zero and an error message like the one for card 263 in Fig. 4. Cards that were referenced by only one block do not belong in RCT and should be moved to the Transient block where they are required. These cards are indicated in warning messages like the one for 598 in Fig. 4.

```
DUPLICATE NUMBER WITHIN REPLACEABLE COMMON TEXT BLOCK
                    JOB ROUTING                                    22AAU0
FREE        NUMBER OF UNITS OF THIS PRIMARY RESOURCE AVAILABLE.    22ABE0


REPLACEABLE COMMON TEXT CARD DUPLICATES PERMANENT COMMON TEXT OR ENTITIES
  THE NEXT CARD OF SUB-DECK (A) CONTAINS...                        581ABH0
```

Fig. 3 -- CHKOUT Error Messages for a Replaceable Common Text Block

```
THE FOLLOWING NUMBERS ARE AVAILABLE FOR REPLACEABLE COMMON TEXT PRECEDING BLOCK   1 WITH STARTING SEQUENCE AA00



THE FOLLOWING NUMBERS WERE USED IN THIS BLOCK

  9    12    21    22    27    30   131   263   323   366   524   556   578   584   591   598   700   800   891   987
999

COUNT OF NUMBER OF BLOCKS REQUIRING EACH REPLACEABLE COMMON TEXT CARD IN ABOVE BLOCK
 ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT    ID COUNT

  9    3   12    2   21    2   22    2   27   14   30    3  131    2  263    0  323    2  366    3
524    2  556    2  578    3  584    3  591    2  598    1  700    4  800    4  891    2  987    3
999    8

ERROR       263 WAS INCLUDED IN ABOVE BLOCK AND NEVER REQUIRED

WARNING      598 WAS REQUIRED BY ONLY ONE BLOCK AND DOES NOT BELONG IN REPLACEABLE COMMON TEXT
```

Fig. 4 -- CHKOUT Output for a Replaceable Common Text Block

Entities Blocks. Figure 5 shows the output that CHKOUT produces whenever an Entities block is encountered. The Identification Numbers listed as available are those in the Entities class that were not used in the current block. Hence, if only one Entities block exists, there will be no available numbers and required numbers will have to be selected from the list of unused numbers.

Several different error messages may accompany the Identification Number lists. Three types are shown in Fig. 5. Additionally, in a Generator Deck containing more than one Entities block, a Decision Table may erroneously require an Entities card that is not in the block the Decision Table is allowed to reference. Such errors are reported with the Identification Number lists for the Entities block, but no example is shown here since the JSSPG has only one Entities block.

The final type of error is not reported with the Entities block, although the error relates to the Entities block. The message appears with the Transient block containing the initialization cards for the generated program. Since this block must be the last block for the program, any requests for Entities cards must have been satisfied before the initialization block is reached. Any unsatisfied requests are reported in error messages.

Transient Blocks. Transient blocks make up most of the Statement List. Since they are so numerous, CHKOUT assigns each block a number representing its position in the deck to aid in identifying the block.

The output for a Transient block is shown in Fig. 6. As CHKOUT processes the Statement List, it checks for Identification Number duplication. Two types of duplication are possible; error messages for both types are shown in Fig. 6 preceding the list of available Identification Numbers.

Since a Transient Statement List block has a block of Decision Tables associated with it, error checking is also performed on the Decision Tables. Five types of errors are possible, of which four are shown in Fig. 6. A Z question can be queried before it is set (see Z 17), which is a logical error since the answer will always be

```
ENTITIES CARD DUPLICATES COMMON TEXT
+T STAGE                                                    581JKXA

DUPLICATE NUMBER WITHIN ENTITIES BLOCK
+T PRIOD                                                    434JKMM
+T OUTCM                                                    434JMZO

     THE FOLLOWING NUMBERS ARE AVAILABLE FOR ENTITIES BLOCK FOLLOWING BLOCK  19 WITH STARTING SEQUENCE JKKO


     THE FOLLOWING NUMBERS WERE USED IN THIS ENTITIES BLOCK

 142  143  156  191  192  193  194  195  196  197  316  321  351  408  410  413  424  425  426  427
 428  429  430  431  432  433  434  435  436  437  453  552  554  555  557  562  563  565  571  572
 581  698  701  702  703  704  705  706  707  711  712  713  714  715  721  722  723  725  732  733
 741  742  751  752  753  762  771  772  773  781  782  791  792  793  801  802  811  812  821

     ENTITY CARD 698 IS INCLUDED HERE AND WAS NEVER REQUIRED
+                N ENDTM      F                                 698JLLR
```

Fig. 5 -- CHKOUT Output for an Entities Block

```
TRANSIENT BLOCK 13 WITH STARTING SEQUENCE CZAO

NUMBER DUPLICATES COMMON TEXT OR ENTITIES
  DO TO 10, FOR I = (1)(NO)                                    430CZVO

DUPLICATE NUMBER WITHIN BLOCK
  GO TO 5                                                      20CZSA
  READ FREE(I), FOR EACH PRESR I                               20DGWO

THE FOLLOWING NUMBERS ARE AVAILABLE IN BLOCK  13

 39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58
 59   60   61   62   63   64   65   66   67   68   70   71   72   73   74   75   76   77   78   79
 80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98   99
100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119
120  121  122  123  124  125  126  127  128  129  130  132  133  134  135  136  137  138  139  140
141  144  145  146  148  149  151  152  153  154  162  164  166  167  168  169  170  174  175  176
177  178  182  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215
216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235
236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  251  253  254  255
256  257  265  269  270  271  273  274  275  276  277  278  279  280  281  282  283  284  285  286
287  288  289  290  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309
310  311  312  313  314  315  317  318  319  320  322  324  325  326  327  328  330  331  332  333
335  336  337  338  339  340  341  342  343  345  347  353  354  362  364  370  371  372  378  381
382  383  384  385  386  391  392  393  394  395  396  397  398  399  401  402  403  404  405  406
407  422  439  440  441  442  443  444  445  446  447  448  449  450  451  452  456  457  458  459
460  461  462  463  464  465  466  467  474  475  476  477  478  479  480  481  491  492  493  494
495  496  497  498  499  501  502  518  519  520  521  522  523  525  526  528  529  530  531  532
533  535  537  538  539  540  541  542  543  544  545  546  547  548  549  553  558  559  564  566
567  568  569  570  579  585  586  589  593  594  596  600  601  604  605  607  608  609  610  611
612  616  617  621  622  631  632  647  648  652  653  656  658  667  668  669  670  671  672  673
680  681  682  683  684  685  688  689  690  691  692  693  694  695  730  731  763  831  833  836
835  837  838  839  840  841  850  861  900  901

  WARNING...Z QUESTION 17 WAS QUERIED BEFORE IT WAS SET - SEQUENCE DHAM

ERROR       165 WAS REQUIRED AND IS NOT IN THE DECK
  165   X X     X                                                DMNA

ERROR       111 IS REQUIRED AND IS NOT PRESENT
  111 X X     X                                                  ECER

  NUMBER       163 IS INCLUDED HERE AND WAS NEVER REQUIRED
    LET N = N + 1                                              163DGGI
```

Fig. 6 -- CHKOUT Output for a Transient Block

```
THE FOLLOWING Z QUESTIONS WERE SET IN BLOCK  13

15    20    21    25    26


THE FOLLOWING Z QUESTIONS WERE QUERIED

17    20    21    25


THE FOLLOWING COMMON TEXT AND ENTITY CARDS WERE REQUIRED

 9    12    21    22    30   142   143   156   165   191   192   193   194   195   196   272   316   329   334   344
348   349   350   413   424   425   426   427   428   429   430   432   433   435   436   437   524   556   562   563
584   591   700   701   702   703   704   705   706   707   711   712   713   715   722   723   725   733   751   800
821   862   987   999
```


Fig. 6 -- continued

"No." A reference can be made to an Identification Number that was never used on any Statement List card, as is the case with 165. Or the number could have been used in some Transient block, but not in the one that the Decision Table references (see 111). The fourth error shown (163) occurs when the Statement List card is present, but no Decision Table references it. Finally, a Decision Table may reference an RCT card that is not in the last RCT block preceding the Decision Table. Since this error can only occur when the Generator Deck contains more than one RCT block and the JSSPG deck has only one, the error cannot be shown.

In addition to error messages and available Identification Numbers, CHKOUT produces a summary of the Z questions queried and set by the Decision Tables in the block. Also shown is a list of the Common Text (both Replaceable and Permanent) and Entities cards that were referenced. Both summaries are shown in Fig. 6.

Summary Information

After the entire Generator Deck has been processed, CHKOUT produces a deck summary (Fig. 7). The summary includes a list of the Identification Numbers in the Transient class, i.e., those that may be used in creating a new Transient block. Also included are lists of Z questions that were set and of those that were queried. If a Z question was queried but never set, an error message would have appeared earlier and would be reflected in the lists here (see Z 17). Questions that were set but never queried appear in the lists, and are also reported in a warning message (see Z 15).

The final output of CHKOUT is a list of Identification Numbers in the Transient class, with a count of the number of blocks in which they were used. A portion of this list is shown in Fig. 8. If the list of unused numbers is exhausted and such a number is required, the count shows which Transient numbers can easily be made unused, i.e., those that have been used in only one block and thus require a minimum of change to the deck.

SUMMARY INFORMATION

THE FOLLOWING NUMBERS MAY BE USED IN A NEW BLOCK

1	2	3	4	5	6	7	8	10	11	13	14	15	17	18	19	20	23	24	25
26	28	29	31	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129
130	132	133	134	135	136	137	138	139	140	141	144	145	146	148	149	151	152	153	154
155	157	158	159	160	161	162	163	164	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	198	199	200	201	202	203
204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252	253	254	255	256	257	265	269	270	271	273	274
275	276	277	278	279	280	281	281	283	284	285	286	287	288	289	290	294	295	296	297
298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	317	318
319	320	327	324	325	326	327	328	330	331	332	333	335	336	337	338	339	340	341	342
343	345	347	353	354	362	364	370	371	372	378	381	382	383	384	385	386	387	388	389
390	391	392	393	394	395	396	397	398	399	401	402	403	404	405	406	407	409	411	412
414	415	416	417	419	420	421	422	423	439	440	441	442	443	444	445	446	447	448	449
450	451	452	456	457	458	459	460	461	462	463	464	465	466	467	474	475	476	477	478
479	480	481	491	492	493	494	495	496	497	498	499	501	502	503	504	505	506	507	509
510	512	513	514	515	516	517	518	519	520	521	522	523	525	526	528	529	530	531	532
533	535	537	538	539	540	541	542	543	544	545	546	547	548	549	551	553	558	559	564
566	567	568	569	570	576	577	579	583	585	586	589	590	592	593	594	596	600	601	602
603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622
623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642
643	644	645	646	647	648	649	650	651	652	653	654	656	658	660	661	667	668	669	670
671	672	673	680	681	682	683	684	685	688	689	690	691	692	693	694	695	708	709	710
716	717	718	719	720	724	726	727	728	729	730	731	734	735	736	737	738	739	740	743
744	745	746	747	748	749	750	754	755	756	759	760	761	763	764	765	766	767	768	769
770	774	775	776	777	778	779	780	783	784	785	786	787	788	789	790	794	795	796	797
803	804	805	806	807	808	809	810	818	819	823	824	825	827	828	829	831	833	834	835
837	838	839	840	841	850	861	900	901											

THE FOLLOWING Z QUESTIONS WERE SET

1	2	3	4	5	6	7	8	9	10	15	20	21	25	26	30	31	32

THE FOLLOWING Z QUESTIONS WERE QUERIED

1	2	3	4	5	6	7	8	9	10	17	20	21	25	26	30	31	32

WARNING...Z QUESTION 15 WAS SET BUT NEVER QUERIED

Fig. 7 -- CHKOUT Summary Information: Transient Identification Numbers and Z Questions

COUNT OF NUMBER OF BLOCKS USING EACH ID NUMBER - EXCLUDING COMMON TEXT AND ENTITIES

ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT	ID	COUNT
1	13	2	13	3	12	4	12	5	11	6	10	7	9	8	8	10	7	11	9
13	9	14	9	15	8	17	9	18	8	19	7	20	9	23	10	24	9	25	10
26	10	28	10	29	10	31	10	34	9	35	10	36	10	37	9	38	11	39	10
40	10	41	8	42	8	43	7	44	7	45	7	46	6	47	6	48	6	49	5
50	5	51	5	52	6	53	6	54	5	55	5	56	5	57	7	58	7	59	5
60	6	61	7	62	6	63	7	64	6	65	6	66	6	67	6	68	6	69	6
70	6	71	6	72	5	73	6	74	5	75	5	76	4	77	6	78	5	79	4
80	3	81	4	82	3	83	3	84	4	85	4	86	4	87	4	88	4	89	4
90	4	91	4	92	4	93	5	94	5	95	5	96	5	97	4	98	3	99	4
100	4	101	3	102	4	103	4	104	4	105	4	106	5	107	4	108	3	109	2
110	3	111	5	112	3	113	3	114	3	115	4	116	3	117	3	118	3	119	2
120	2	121	3	122	1	123	3	124	1	125	2	126	3	127	2	128	3	129	2
130	2	132	4	133	2	134	2	135	2	136	2	137	2	138	2	139	2	140	2
141	2	144	1	145	1	146	1	148	1	149	1	151	1	152	1	153	2	154	1
155	1	157	1	158	1	159	1	160	3	161	3	162	2	163	3	164	1	166	1
167	1	168	1	169	1	170	1	171	1	172	1	173	1	174	1	175	1	176	1
177	1	178	1	179	1	180	1	181	1	182	1	183	1	184	1	185	1	186	1
187	1	188	1	189	1	190	1	198	1	199	1	200	4	201	4	202	4	203	4
204	4	205	4	206	4	207	4	208	3	209	3	210	3	211	3	212	3	213	3
214	3	215	3	216	3	217	3	218	3	219	3	220	2	221	2	222	2	223	2
224	3	225	2	226	2	227	2	228	2	229	2	230	2	231	2	232	2	233	2
234	2	235	2	236	2	237	2	238	2	239	2	240	2	241	1	242	1	243	1
244	2	245	1	246	2	247	2	248	1	249	1	250	1	251	2	252	1	253	1
254	1	255	1	256	1	257	1	265	1	269	1	270	1	271	1	273	1	274	1
275	1	276	1	277	1	278	1	279	1	280	1	281	1	282	1	283	1	284	1
285	1	286	1	287	1	288	1	289	1	290	1	294	1	295	1	296	1	297	1
298	1	299	1	300	1	301	2	302	2	303	1	304	2	305	2	306	1	307	1
308	1	309	1	310	2	311	2	312	1	313	1	314	1	315	1	317	1	318	1
319	1	320	1	322	1	324	1	325	1	326	1	327	1	328	1	330	1	331	1
332	1	333	1	335	1	336	1	337	1	338	1	339	1	340	2	341	1	342	1
343	1	345	1	347	1	353	1	354	1	362	1	364	1	370	1	371	1	372	1
378	1	381	1	382	1	383	1	384	1	385	1	386	1	387	1	388	1	389	1
390	1	391	1	392	1	393	1	394	1	395	1	396	1	397	1	398	1	399	1
401	1	402	1	403	1	404	1	405	1	406	1	407	1	409	1	411	1	412	1
414	1	415	1	416	1	417	1	419	1	420	1	421	1	422	1	423	1	439	1
440	1	441	1	442	1	443	1	444	1	445	1	446	1	447	1	448	1	449	1
450	1	451	1	452	1	456	1	457	1	458	1	459	1	460	1	461	1	462	1
463	1	464	1	465	1	466	1	467	1	474	1	475	1	476	1	477	1	478	1
479	1	480	1	481	1	491	1	492	1	493	1	494	1	495	1	496	1	497	1
498	1	499	1	501	2	502	3	503	2	504	1	505	2	506	2	507	2	509	2
510	2	512	2	513	2	514	1	515	1	516	1	517	1	518	1	519	1	520	1
521	1	522	1	523	1	525	1	526	1	528	1	529	1	530	1	531	1	532	1
533	1	535	1	537	1	538	1	539	1	540	1	541	1	542	1	543	1	544	1
545	1	546	1	547	1	548	1	549	1	551	1	553	1	558	1	559	1	564	1
566	1	567	1	568	1	569	1	570	1	576	1	577	1	579	1	583	1	585	1
586	1	589	1	590	1	592	1	593	1	594	1	596	1	600	1	601	1	602	1
603	1	604	1	605	1	606	1	607	1	608	1	609	1	610	1	611	1	612	2
613	2	614	1	615	1	616	1	617	1	618	2	619	1	620	1	621	1	622	1
623	1	624	1	625	1	626	1	627	1	628	1	629	1	630	1	631	1	632	1
633	1	634	1	635	1	636	1	637	1	638	1	639	1	640	1	641	1	642	1
643	1	644	1	645	1	646	1	647	1	648	1	649	1	650	1	651	1	652	1

Fig. 8 -- CHKOUT Summary Information: Identification Number Usage

THE LOCATE PROGRAM

In a deck the size of a Generator Deck, it is difficult to find all references to a particular Identification Number or Questionnaire question by scanning a listing of the deck. Since such searches are not unusual, the LOCATE program was written to perform them.

LOCATE requires an input deck containing the question numbers and Identification Numbers that are to be found, with additional information regarding the scope of the search. Figure 9 shows a sample LOCATE input deck on a Decision Table coding form, since this form is almost perfectly suited to the purpose. The Question Card/Action Code column (column 2) indicates the Questionnaire section if a question is to be located, and is blank otherwise. The Question Number/Identification Number field is used as indicated by its name. The Conditions/Actions field is used, beginning in column 8, to indicate the type of reference sought: QUESTION, ACTION, or STATEMENT. If this field is blank, all references are found. The Tag and Sequence I.D. fields are used to delimit the search. The Tag field contains the sequence number where the search is to begin; a blank indicates the beginning of the deck. The Sequence I.D. field contains the sequence number where the search is to end; a blank here indicates the end of the deck.

The cards in the deck in Fig. 9 would produce output as follows:

Card 1: All Statement List cards with Identification Number 243, and all Decision Table Action cards referencing 243 in the entire deck.

Card 2: All Decision Table Action cards referencing Identification Number 119 that have sequence numbers between ARKO and BJMA.

Card 3: All appearances of question Z 3 as a Condition before sequence number CDRO.

Card 4: All appearances of question B 2 after sequence number EMMO. Questionnaire questions appear only as Conditions, so QUESTION need not be specified in the Conditions/Actions field.

Card 5: All Statement List cards with Identification Number 722 in the entire deck.

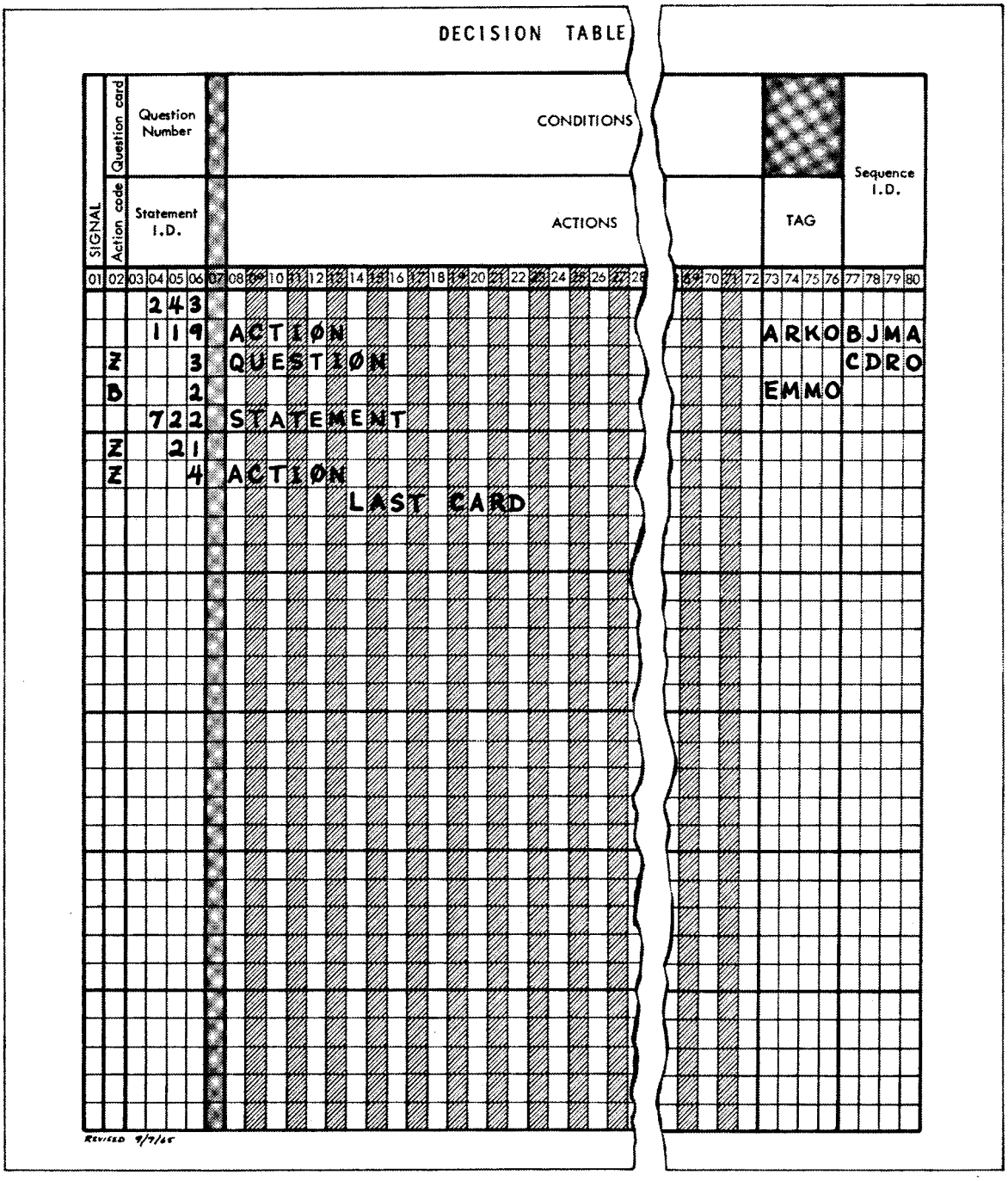

Fig. 9 -- Sample deck for the LOCATE program showing the use of the Decision Table coding form

Card 6: All references to question Z 21, either as a Condition or as an Action, in the entire deck.

Card 7: All appearances of question Z 4 as an Action in the entire deck.

The deck must always be terminated with a card that is blank in columns 1-6. Each question number or Identification Number may appear in the deck only once; LOCATE will produce an error message if the same number is requested more than once and will ignore all but the first request.

The output from LOCATE appears in sequence number order, i.e., the output lines are not sorted by Identification Number or question number.

The LOCATE program is written in SIMSCRIPT; it is shown in Appendix B. The usage of the arrays and their initialization are included in the appendix following the program listing.

THE KORCT PROGRAM

Since the Generator Deck is so large, it should be kept on magnetic tape instead of being used in card form. KORCT is a FORTRAN IV program that makes corrections to the tape and checks its sequence order.

KORCT uses the sequence numbers in the Generator Deck to make the corrections. The existing Generator Tape is mounted on FORTRAN logical unit 8, and a blank output tape is mounted on unit 9. A correction deck containing the new cards is prepared and ordered by sequence number. This deck is terminated with a blank card. KORCT applies the corrections to the existing tape, producing the new Generator Tape while checking the sequencing of the original tape and of the correction deck. If either is out of order, KORCT issues an error message (see Fig. 10) and stops writing the new tape, although it continues to sequence-check the original.

The correction deck can contain insertion, replacement, and deletion cards. An insertion card has a sequence number that does not correspond to that of any card on the tape, and is inserted in the sequence. Replacement and deletion cards have sequence numbers

```
DELETION        X                                                    DHHO
DELETED       - 193 X X X X                                          3DHHO

REPLACEMENT   A   5 N N                                              DHTO
REPLACED      A   4 N   N                                           DHTO

WARNING...POSSIBLE ERROR...W 9 CARD REPLACED BY...
              T     ERROR CHECK                                      KHJO

ERROR...THE FOLLOWING DATA CARD OUT OF SEQUENCE
              B  10 N N                                            DHXX
        LAST CORRECT CARD WAS KHJO


END OF JOB
```

```
ATTEMPTING TO REMOVE NONEXISTENT CARD DHAA -NO ACTION TAKEN

INSERTION     = 162 X X X X                                          DHCC

ERROR...END OF TAPE REACHED BEFORE LAST DATA CARD WAS READ


END OF JOB
```

Fig. 10 -- Printouts from two KORCT runs showing all the messages

matching those of cards on the tape. A deletion card causes the removal of its corresponding card from the tape, while a replacement card takes the place of the original card. A deletion card is distinguished from a replacement or insertion card by its content; it is totally blank except for its sequence number and an alphabetic X punch in column 5. If a deletion card has no corresponding card on the tape, KORCT produces a warning message (see sequence DHAA, Fig. 10) and ignores the card.

As KORCT applies the correction deck to the tape, it produces a listing of all the changes made to the tape (see Fig. 10). Both the correction and original cards are shown, providing a permanent record of the changes.

KORCT recognizes the end-of-deck card (the W card with the 9 code). If this card is replaced by a card in the correction deck, as

is necessary when adding to the end of the deck, KORCT issues a warning message indicating a possible error and continues execution, expecting to find a new end-of-deck card in the correction deck. When the new W 9 card is encountered, KORCT terminates execution. If the W 9 card is found on the original tape while corrections remain to be applied, but no card replaces the W 9 card, KORCT issues an error message and terminates. Thus, the W 9 card must be replaced in order to be able to add new cards at the end.

KORCT can be executed without a correction deck, although the terminal blank card is still required. In this mode, KORCT will only sequence-check the original tape and will produce no copy, since it is able to determine almost immediately that no corrections are to be applied. However, the output tape must still be mounted, since KORCT will attempt to rewind the tape before it discovers there are no corrections.

The listing of the KORCT program is given in Appendix C.

III. JSSPG ANALYSIS PROGRAMS

DEBUGGING THE JSSPG

The problem of debugging any computer program is never minor, nor are the techniques to be used clearly defined. Debugging a Program Generator presents a special problem, since the tremendous number of programs that can be generated makes it impossible to debug all possible programs. The best course of action is to debug as many programs as possible, judiciously selecting them so that they exercise all Questionnaire options at least once, and include all combinations of interrelated options. However, even the best possible selections result in a very large number of programs to be individually checked. For the JSSPG, over fifty programs were required, although for many of these only a small portion of the logic had to be checked.

Debugging these fifty programs was simplified by two techniques. First, the options for each program were selected in such a way that, by careful specification of data, the programs would produce the same outputs, although the logic exercised varied considerably. For example, several options were concerned with whether or not there were different types of jobs, and if so, whether certain characteristics depended on the job type. One generated program had no job types, while another had types and everything depended on the job type. For direct comparison, the number of job types in the second model was set to one, and the results of the simulations were identical. The second model was then reinitialized for a full-scale run. Many of the available features in the JSSPG were checked in this fashion.

The second technique involved tracing the activities carried out in the simulated job shop. Such tracing is usually done by inserting commands in the program that report on the various activities in the simulated model. With the JSSPG this was undesirable since the programs were generated instead of coded, and changing the Generator Deck often introduces errors. Instead, the two programs presented in this section were written to provide the trace information. They subsequently proved to be useful in understanding the processes

occurring within the job shop, and may give future insight into why certain decision rules in a job shop work as they do.

THE JSSPG ANALYSIS TAPE

Whenever a change occurs in the simulated job shop, the simulator writes a message on the analysis tape for later processing by the analysis programs. Each message consists of a code number indicating the type of change that took place, the simulated time of the change, and four other items of data that depend on the type of change. These messages appear in simulated time sequence. A complete list of the messages is given in RM-5162-PR,* and Fig. 11 contains a sample printout of the tape. As can readily be seen, it is difficult to analyze the shop activities from this tape.

The JSSPG analysis programs process this tape when the simulation is complete. The two programs presented here, JHIST and RHIST, also process the tape. Both programs trace jobs and processes through the simulation and present the data from the analysis tape in an easily usable form. The application of the programs is subject to two restrictions, however. First, they may be used only with the analysis tape produced when the standard analysis available in the JSSPG is selected (Questionnaire option E 2), and the analysis must include all three types of statistics (job, resource, and queue). Second, both programs identify a job by its cost attribute. Hence the cost attribute must be of such a nature that it is never identical for two different jobs.

THE JHIST PROGRAM

JHIST is a nonsimulation program written in SIMSCRIPT, which presents a history of each job processed in the simulated job shop. JHIST sorts the messages on the analysis tape by the job; messages relating to a given job are grouped together and appear in the order of occurrence within the job grouping. The jobs themselves appear in the order in which they were completed and left the shop.

---

*Oldfather, Ginsberg, Love, and Markowitz, op. cit., pp. 57-59.

3	6	0.18712	111.66013	0	4.17238
5	3	0.39723	3.72431	1134.88171	4.17238
6	3	0	1134.88171	0	4.17238
1	1	181.60099	0	0	4.18158
3	1	0.06648	181.60099	0	4.18158
7	3	0	163.25816	0	4.20709
3	6	0.38618	163.25816	0	4.20709
5	3	0.45477	3.07817	1132.94337	4.20709
6	3	0	1132.94337	0	4.20709
1	5	144.64083	0	0	4.21146
3	1	0.01865	144.64083	0	4.21146
7	10	0	968.66334	0	4.22732
3	3	0.33381	968.66334	0	4.22732
5	10	0.65299	4.00000	1021.16349	4.22732
6	10	0	1021.16349	0	4.22732
1	4	253.33208	0	0	4.24137
13	2	0	0	0	4.24137
6	2	0	253.33208	0	4.24137
7	10	0	758.67401	0	4.27666
3	3	0.67769	758.67401	0	4.27666
5	10	0.33259	4.00000	191.17618	4.27666
6	10	0	191.17618	0	4.27666
7	2	0	172.86094	0	4.29023
3	4	0.97457	172.86094	0	4.29023
7	3	0	710.18796	0	4.32223
3	6	0.43728	710.18796	0	4.32223
5	3	0.97978	3.90738	1091.31540	4.32223
6	3	0	1091.31540	0	4.32223
1	2	37.08110	0	0	4.32449
13	2	0	0	0	4.32449
6	2	0	37.08110	0	4.32449
8	0	0	0	0	4.33333
9	1	5	0	0	4.33333

Fig. 11 -- Contents of the JSSPG Analysis Tape

Figure 12 shows a sample of the output from JHIST. The first line of a job report contains summary information about the job itself: its cost, job type, arrival date, due date, finished date, and total time in the shop. Note that each job also has a number. The numbers are assigned in the order in which the jobs arrive in the shop and may be used for referring to the output of the RHIST program, which assigns job numbers in the same order.

Subsequent lines of output are of two types: a process report and a queue report. The process report contains the numbers of the primary and secondary resources that processed the job (if the model has no secondary resources, the number for the secondary is zero), the time that work was begun on the job, the time that work stopped on the job, and the total time spent processing the job. Work is stopped on a job either when the job is completed, or when the job must be interrupted because of a shift change at which the available resources decrease. If the job was interrupted, the process report is immediately followed by a queue report stating that the job was interrupted. (For example, see job 9 at resource 6.) Queue reports are also produced when a job goes into queue before work begins on it at a particular primary resource (see job 9 at resource 2). A queue report contains the primary resource number where the job queued, whether the job was interrupted or queued, the time of interrupt or arrival at the queue, the time at which the job was removed from the queue, the total time the job remained in the queue, and the estimated processing time on the job (the last value is included because some of the job selection rules use the estimated processing time).

JHIST terminates execution when the end of simulation message is reached on the analysis tape or when all available core storage has been used for the messages. Core storage can readily be exhausted, since all messages relating to a given job are retained in core until the job leaves the shop. Upon either type of termination, JHIST writes a message stating that the jobs detailed beyond that point are not completed and giving the time of the exit. All jobs still in the shop are then printed out in job-number order.

```
                                    JOB REPORT
JOB NUMBER     9   COST =   497.136   TYPE =   2, ARRIVED    0.260, DUE    3.054,   FINISHED    9.516,  IN SHOP    9.255283
      AT RESOURCE   2,  INTO QUEUE AT    0.260, OUT OF QUEUE AT    0.264, Q-TIME =    0.00332, EST-PROCESS-TIME =    0.66227
   WORKED ON BY PRIMARY    2, SECONDARY     0,  BEGAN WORK    0.264,  FINISHED    0.866,  PROCESS TIME    0.602067
      AT RESOURCE   4,  INTO QUEUE AT    0.866, OUT OF QUEUE AT    1.319, Q-TIME =    0.45365, EST-PROCESS-TIME =    1.81472
   WORKED ON BY PRIMARY    4, SECONDARY     0,  BEGAN WORK    1.319,  FINISHED    2.969,  PROCESS TIME    1.649744
      AT RESOURCE   6,  INTO QUEUE AT    2.969, OUT OF QUEUE AT    7.333, Q-TIME =    4.36430, EST-PROCESS-TIME =    0.38549
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    7.333,  FINISHED    7.667,  PROCESS TIME    0.333333
      AT RESOURCE   6, INTERRUPTED AT    7.667, OUT OF QUEUE AT    7.688, Q-TIME =    0.02139, EST-PROCESS-TIME =    0.01712
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    7.688,  FINISHED    7.705,  PROCESS TIME    0.017116
   WORKED ON BY PRIMARY    8, SECONDARY     0,  BEGAN WORK    7.705,  FINISHED    7.943,  PROCESS TIME    0.237865
      AT RESOURCE  10,  INTO QUEUE AT    7.943, OUT OF QUEUE AT    8.657, Q-TIME =    0.71355, EST-PROCESS-TIME =    0.42387
   WORKED ON BY PRIMARY   10, SECONDARY     0,  BEGAN WORK    8.657,  FINISHED    8.667,  PROCESS TIME    0.010083
      AT RESOURCE  10, INTERRUPTED AT    8.667, OUT OF QUEUE AT    8.841, Q-TIME =    0.17396, EST-PROCESS-TIME =    0.37526
   WORKED ON BY PRIMARY   10, SECONDARY     0,  BEGAN WORK    8.841,  FINISHED    9.000,  PROCESS TIME    0.159377
      AT RESOURCE  10, INTERRUPTED AT    9.000, OUT OF QUEUE AT    9.300, Q-TIME =    0.29965, EST-PROCESS-TIME =    0.21588
   WORKED ON BY PRIMARY   10, SECONDARY     0,  BEGAN WORK    9.300,  FINISHED    9.516,  PROCESS TIME    0.215881
                                    JOB REPORT
JOB NUMBER    50   COST =    45.237   TYPE =   2, ARRIVED    1.253, DUE    2.926,   FINISHED    9.531,  IN SHOP    8.278500
      AT RESOURCE   2,  INTO QUEUE AT    1.253, OUT OF QUEUE AT    1.316, Q-TIME =    0.06336, EST-PROCESS-TIME =    0.35257
   WORKED ON BY PRIMARY    2, SECONDARY     0,  BEGAN WORK    1.316,  FINISHED    1.637,  PROCESS TIME    0.320520
      AT RESOURCE   4,  INTO QUEUE AT    1.637, OUT OF QUEUE AT    2.291, Q-TIME =    0.65425, EST-PROCESS-TIME =    1.95108
   WORKED ON BY PRIMARY    4, SECONDARY     0,  BEGAN WORK    2.291,  FINISHED    4.065,  PROCESS TIME    1.773713
      AT RESOURCE   6,  INTO QUEUE AT    4.065, OUT OF QUEUE AT    4.465, Q-TIME =    0.39975, EST-PROCESS-TIME =    0.69908
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    4.465,  FINISHED    5.000,  PROCESS TIME    0.535405
      AT RESOURCE   6, INTERRUPTED AT    5.000, OUT OF QUEUE AT    5.333, Q-TIME =    0.33333, EST-PROCESS-TIME =    0.10012
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    5.333,  FINISHED    5.433,  PROCESS TIME    0.100118
      AT RESOURCE   8,  INTO QUEUE AT    5.433, OUT OF QUEUE AT    5.503, Q-TIME =    0.06912, EST-PROCESS-TIME =    0.43991
   WORKED ON BY PRIMARY    8, SECONDARY     0,  BEGAN WORK    5.503,  FINISHED    5.667,  PROCESS TIME    0.164091
      AT RESOURCE   8, INTERRUPTED AT    5.667, OUT OF QUEUE AT    6.667, Q-TIME =    1.00000, EST-PROCESS-TIME =    0.23583
   WORKED ON BY PRIMARY    8, SECONDARY     0,  BEGAN WORK    6.667,  FINISHED    6.902,  PROCESS TIME    0.235831
      AT RESOURCE  10,  INTO QUEUE AT    6.902, OUT OF QUEUE AT    9.333, Q-TIME =    2.43084, EST-PROCESS-TIME =    0.21798
   WORKED ON BY PRIMARY   10, SECONDARY     0,  BEGAN WORK    9.333,  FINISHED    9.531,  PROCESS TIME    0.198166
                                    JOB REPORT
JOB NUMBER    61   COST =    37.542   TYPE =   2, ARRIVED    1.676, DUE    2.915,   FINISHED    9.553,  IN SHOP    7.877056
   WORKED ON BY PRIMARY    2, SECONDARY     0,  BEGAN WORK    1.676,  FINISHED    1.978,  PROCESS TIME    0.301513
      AT RESOURCE   4.  INTO QUEUE AT    1.978, OUT OF QUEUE AT    4.335, Q-TIME =    2.35724, EST-PROCESS-TIME =    0.38199
   WORKED ON BY PRIMARY    4, SECONDARY     0,  BEGAN WORK    4.335,  FINISHED    4.682,  PROCESS TIME    0.347259
      AT RESOURCE   6,  INTO QUEUE AT    4.682, OUT OF QUEUE AT    8.333, Q-TIME =    3.65129, EST-PROCESS-TIME =    0.27877
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    8.333,  FINISHED    8.587,  PROCESS TIME    0.253425
   WORKED ON BY PRIMARY    8, SECONDARY     0,  BEGAN WORK    8.587,  FINISHED    8.832,  PROCESS TIME    0.245415
      AT RESOURCE  10,  INTO QUEUE AT    8.832, OUT OF QUEUE AT    9.333, Q-TIME =    0.50116, EST-PROCESS-TIME =    0.24173
   WORKED ON BY PRIMARY   10, SECONDARY     0,  BEGAN WORK    9.333,  FINISHED    9.553,  PROCESS TIME    0.219751
                                    JOB REPORT
JOB NUMBER    33   COST =   158.329   TYPE =   5, ARRIVED    0.845, DUE    1.830,   FINISHED    9.584,  IN SHOP    8.738675
      AT RESOURCE   1,  INTO QUEUE AT    0.845, OUT OF QUEUE AT    0.857, Q-TIME =    0.01229, EST-PROCESS-TIME =    0.64823
   WORKED ON BY PRIMARY    1, SECONDARY     0,  BEGAN WORK    0.857,  FINISHED    1.506,  PROCESS TIME    0.648229
      AT RESOURCE   4,  INTO QUEUE AT    1.506, OUT OF QUEUE AT    1.594, Q-TIME =    0.08814, EST-PROCESS-TIME =    1.33650
   WORKED ON BY PRIMARY    4, SECONDARY     0,  BEGAN WORK    1.594,  FINISHED    2.000,  PROCESS TIME    0.406192
      AT RESOURCE   4, INTERRUPTED AT    2.000, OUT OF QUEUE AT    2.072, Q-TIME =    0.07233, EST-PROCESS-TIME =    0.93031
   WORKED ON BY PRIMARY    4, SECONDARY     0,  BEGAN WORK    2.072,  FINISHED    3.003,  PROCESS TIME    0.930311
   WORKED ON BY PRIMARY    7, SECONDARY     0,  BEGAN WORK    3.003,  FINISHED    3.487,  PROCESS TIME    0.484794
      AT RESOURCE   6,  INTO QUEUE AT    3.487, OUT OF QUEUE AT    9.333, Q-TIME =    5.84590, EST-PROCESS-TIME =    0.05753
   WORKED ON BY PRIMARY    6, SECONDARY     0,  BEGAN WORK    9.333,  FINISHED    9.391,  PROCESS TIME    0.057531
   WORKED ON BY PRIMARY    9, SECONDARY     0,  BEGAN WORK    9.391,  FINISHED    9.584,  PROCESS TIME    0.192967
```

Fig. 12 -- Sample JHIST output

A listing of the JHIST program appears in Appendix D. The usage of the various arrays and attributes and their initialization are included following the listing.

THE RHIST PROGRAM

RHIST is a nonsimulation SIMSCRIPT program that produces a history of the work performed by each resource in the job shop. Like JHIST, it uses the analysis tape from the simulator and sorts the simulator messages, but the sort is based on resource, not job. The activities of a particular primary or secondary resource are reported in time sequence. Figure 13 shows an example of the output from RHIST containing all possible messages.

The user of the RHIST program may be selective about the resources on which reports are given, and need not get reports for the entire simulation. RHIST requires two data cards for resource selection, the first for the primary resources, and the second for the secondaries. The resources desired are indicated by numeric punches (other than zero or a sign punch) in the card columns corresponding to the resource numbers. For example, if the first card has punches in only columns 3 and 4 and the second card is blank, only messages applying to the third and fourth primaries will be output.

Additionally, a permanent system variable, called QUIT, is initialized to the length of simulated time for which reports are desired. If QUIT was initialized to 10.0, for example, RHIST would summarize information only for the first ten days of the simulation.

RHIST, like JHIST, constructs messages and stores them until all the messages for a resource are collected. Unlike JHIST, however, RHIST is able to continue processing after running out of storage space. The history of activities for each resource is printed out and the messages destroyed, freeing the storage so that RHIST can continue. Consequently, although the listing may be in several parts, the complete resource history can be produced.

When the status of a resource changes, RHIST produces a message showing the change. Whenever work is begun or completed on a job,

```
                                PRIMARY RESOURCE      8

START WORK ON JOB NUMBER  38 AT     4.65221 (DAY     4 HOUR 1540), WITH SECONDARY RESOURCE     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    6 PRIMARIES WORKING,    0 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    49 INTO QUEUE AT     4.66373 (DAY     4 HOUR 1556). PRESENT NUMBER IN QUEUE =     1

AT SHIFT CHANGE (DAY     4 HOUR 16 0) THERE ARE     6 PRIMARIES OF THIS TYPE WORKING,     0 IDLE
AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS PRIMARY RESOURCE IS      5

JOB NUMBER    43 RESTARTED AFTER SHIFT CHANGE WITH PRIMARY     8, SECONDARY     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    1 PRIMARIES WORKING,    4 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    36 RESTARTED AFTER SHIFT CHANGE WITH PRIMARY     8, SECONDARY     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    2 PRIMARIES WORKING,    3 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    79 RESTARTED AFTER SHIFT CHANGE WITH PRIMARY     8, SECONDARY     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    3 PRIMARIES WORKING,    2 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER     8 RESTARTED AFTER SHIFT CHANGE WITH PRIMARY     8, SECONDARY     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    4 PRIMARIES WORKING,    1 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    73 RESTARTED AFTER SHIFT CHANGE WITH PRIMARY     8, SECONDARY     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    5 PRIMARIES WORKING,    0 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    38 INTERRUPTED AT     4.66667 (DAY     4 HOUR 16 0). PRESENT NUMBER IN QUEUE =     2

END PROCESS ON JOB NUMBER  36 AT     4.82774 (DAY     4 HOUR 1952), WITH SECONDARY RESOURCE     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    4 PRIMARIES WORKING,    1 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    38 OUT OF QUEUE AT     4.82774 (DAY     4 HOUR 1952). PRESENT NUMBER IN QUEUE =     1

START WORK ON JOB NUMBER  38 AT     4.82774 (DAY     4 HOUR 1952), WITH SECONDARY RESOURCE     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    5 PRIMARIES WORKING,    0 IDLE,     0 SECONDARIES WORKING,    0 IDLE

END PROCESS ON JOB NUMBER   8 AT     4.95556 (DAY     4 HOUR 2257), WITH SECONDARY RESOURCE     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    4 PRIMARIES WORKING,    1 IDLE,     0 SECONDARIES WORKING,    0 IDLE

JOB NUMBER    49 OUT OF QUEUE AT     4.95556 (DAY     4 HOUR 2257). PRESENT NUMBER IN QUEUE =     0

START WORK ON JOB NUMBER  49 AT     4.95556 (DAY     4 HOUR 2257), WITH SECONDARY RESOURCE     0
   CURRENTLY OF THESE RESOURCE TYPES, THERE ARE    5 PRIMARIES WORKING,    0 IDLE,     0 SECONDARIES WORKING,    0 IDLE

AT SHIFT CHANGE (DAY     5 HOUR  0 0) THERE ARE     5 PRIMARIES OF THIS TYPE WORKING,     0 IDLE
AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS PRIMARY RESOURCE IS      0

JOB NUMBER    43 INTERRUPTED AT     5.00000 (DAY     5 HOUR  0 0). PRESENT NUMBER IN QUEUE =     1

JOB NUMBER    79 INTERRUPTED AT     5.00000 (DAY     5 HOUR  0 0). PRESENT NUMBER IN QUEUE =     2

JOB NUMBER    73 INTERRUPTED AT     5.00000 (DAY     5 HOUR  0 0). PRESENT NUMBER IN QUEUE =     3

JOB NUMBER    38 INTERRUPTED AT     5.00000 (DAY     5 HOUR  0 0). PRESENT NUMBER IN QUEUE =     4
```

Fig. 13 -- Sample RHIST output

the fact is reported, giving the simulated time, the job number, the number of the other resource working on the job if the model has both primaries and secondaries, and the current status of both the primary and secondary resources, i.e., the number of units working and idle. Whenever a shift-change occurs, the time of the change is given with the status of the resource before the change and the new availability after the shift-change. Immediately after a shift-change, if jobs were in process before the change, all such jobs are accounted for as either restarted or interrupted. Finally, whenever a job goes into or out of queue or is interrupted, the time, job number, and current queue status are reported. All of these messages are shown in Fig. 13.

A listing of the RHIST program and the use and initialization of the system variables are given in Appendix E.

Appendix A

THE CHKOUT PROGRAM

```
+                          1IDS    E
+                          2ZEES   E
+                          3THERE  11/2  I
+                          3COUNT  12/2  I
+                          4START  1     I
+                          5CASK   11/2  I
+                          5ASK    12/2  I
+                          6ZTHER  11/2  I
+                          6ZASK   12/2  I
+                          7HW           I*
+                          8HU           I*
+                          9HE           I*
+                         10HT           I*
+                         11HZ           I*
+                         12MAX          I*
+                         13CNTRL        I*
+                         14CMAX         I
+                         15AMAX         I
+                         16CBEGN        I
+                         17ABEGN        I
+                         18LNCNT        I
```

```
1          18
   1        R                          1000            IDS
   2        R                          72              ZBBS
   3    5 1  Z 1000   1
   6      1  Z   72   2
   7   11    R                           5(A1)
WUETZ
  12        R                          350             MAX
  13        R                          50              CNTR2
  14   18     Z
                END INITIAL
```

43

```
*IBFTC BLOCK
      SUBROUTINE BLOCK(I,J)
X     DATA K /0/
      DIMENSION NQR(20)
      IF (J) EQ (9), GO TO 10
      LET K = K + 1
      LET NQR(K) = I
      IF (K) LS (20), RETURN
      CALL TEST
X  30 WRITE (6,11) (NQR(L), L = 1,K)
X  11 FORMAT (1X,20I6)
      LET LNCNT = LNCNT + 1
      LET K = 0
      IF (J) NE (9), RETURN
X  20 WRITE (6,12)
X  12 FORMAT (/)
      LET LNCNT = LNCNT + 2
      RETURN
   10 IF (K) 20,20,30
      END


*IBFTC COUNT
      SUBROUTINE COUNT(NUMBR,N,NOSEQ)
      LET N = 0
X  10 READ (9,5) IS,NSQ
X   5 FORMAT (72X,I4,A4)
      IF (N) EQ (0), LET NOSEQ = NSQ
      LET N = N + 1
      IF (IS) GE (9999), RETURN
      IF (IS) EQ (0), GO TO 10
      IF (IS) GR (8), GO TO 40
      IF (NUMBR) LS (998), GO TO 40
      CALL TEST
X     WRITE (6,50) IS,NSQ
X  50 FORMAT (/1X,5HERROR,I10,1X,53HMAY NOT BE USED IN COMMON TEXT OR EN
X    XTITIES - SEQUENCE,1X,A4//)
      LET LNCNT = LNCNT + 5
      GO TO 10
   40 LET J = THERE(IS)
      IF (J) LS (998), GO TO 20
      IF (NUMBR) LS (998), GO TO 10
      IF (NUMBR) EQ (9999), GO TO 60
      IF (J) EQ (NUMBR), GO TO 10
      LET THERE(IS) = 9997
      GO TO 10
X  60 WRITE (6,61) IS, NSQ
X  61 FORMAT (1X,5HERROR,I10,1X,54HIS A DUPLICATE WITHIN PERMANENT COMMO
X    XN TEXT - SEQUENCE,1X,A4//)
      LET LNCNT = LNCNT + 3
      GO TO 10
   20 LET THERE(IS) = NUMBR
      GO TO 10
      END


*IBFTC MAIN
      MAIN ROUTINE
      DIMENSION TEXT(500,13)
X     INTEGER GROUP, PERM, PCT
      REWIND TAPE 9
      LET LNCNT = 56
      CALL TEST
C     BEGIN PASS 1
      CALL COUNT(9999,I,NOSEQ)
      LET PCT = I
      LET PERM = I
      LET GROUP = 0
      CALL TEST
X     WRITE (6,12) I
X  12 FORMAT (1X,9HTHERE ARE,I3,1X,30HCARDS IN PERMANENT COMMON TEXT/)
      LET LNCNT = LNCNT + 2
X     READ (9,11) IWHAT,ICODE,ISTART
X  11 FORMAT (A2,I4,70X,A4)
      IF (IWHAT) EQ (HW), GO TO 30
X     WRITE (6,13) ISTART
X  13 FORMAT (1X,52HNO W CARD FOLLOWING PERMANENT COMMON TEXT - SEQUENCE
X    X,1X,A4)
      CALL EXIT
   30 IF (ICODE) LE (4), GO TO 25
X     WRITE (6,14) ISTART
X  14 FORMAT (1X,62HILLEGAL W CARD CODE FOLLOWING PERMANENT COMMON TEXT
X    X- SEQUENCE,1X,A4)
```

```
      CALL EXIT
   25 GO TO (40,50,40,50,60,50,40,50,80), ICODE
C     REPLACEABLE COMMON TEXT BLOCK
   40 CALL COUNT(999,I,NOSEQ)
      LET PERM = PCT + I
      CALL TEST
X     WRITE (6,15) I,NOSEQ
X  15 FORMAT (5X,9HTHERE ARE,I4,1X,54HCARDS IN REPLACEABLE COMMON TEXT B
X    XEGINNING AT SEQUENCE,1X,A4/)
      LET LNCNT = LNCNT + 2
C     TRANSIENT BLOCK
   50 LET GROUP = GROUP + 1
      CALL COUNT(GROUP,I,NOSEQ)
      CALL TEST
X     WRITE (6,16) I,GROUP,NOSEQ
X  16 FORMAT (8X,9HTHERE ARE,I4,1X,14HCARDS IN BLOCK,I3,1X,20HSTARTING A
X    XT SEQUENCE,1X,A4/)
      LET LNCNT = LNCNT + 2
      LET I = MAX - PERM - I
      IF (I) GR (CNTRL), GO TO 31
      CALL TEST
X     WRITE (6,17) I,GROUP
X  17 FORMAT (5X,14HWARNING...ONLY,I3,1X,19HMORE CARDS IN BLOCK,I3//)
      LET LNCNT = LNCNT + 3
   31 LET IC = 1
   20 LET I = 0
X  21 READ (9,11) IWHAT,ICODE,ISTART
      IF (I) EQ (0), LET ISAVE = ISTART
      LET I = I + 1 - IC
      LET IC = 0
      IF (IWHAT) NE (HU), GO TO 21
      IF (CMAX) GE (I), GO TO 23
      LET CMAX = I
      LET CBEGN = ISAVE
   23 LET I = 0
X  22 READ (9,11) IWHAT,ICODE,ISTART
      IF (I) EQ (0), LET ISAVE = ISTART
      LET I = I + 1
      IF (IWHAT) NE (HE), GO TO 22
      IF (AMAX) GE (I), GO TO 27
      LET AMAX = I
      LET ABEGN = ISAVE
X  27 READ (9,11) IWHAT,ICODE,ISTART
      IF (IWHAT) EQ (HT), GO TO 20
      IF (IWHAT) EQ (HW), GO TO 25
X     WRITE (6,18) ISTART
X  18 FORMAT (1X,57HINVALID SIGNAL CARD FOLLOWING DECISION TABLES AT SEQ
X    XUENCE,1X,A4)
      CALL EXIT
C     ENTITIES BLOCK
   60 CALL COUNT(998,I,NOSEQ)
      CALL TEST
X     WRITE (6,63) I,NOSEQ
X  63 FORMAT (5X,9HTHERE ARE,I4,1X,45HCARDS IN ENTITIES BLOCK BEGINNING
X    XAT SEQUENCE,1X,A4/)
      LET LNCNT = LNCNT + 2
      LET I = MAX - PERM - I
      IF (I) GR (CNTRL), GO TO 65
      CALL TEST
X     WRITE (6,64) I
X  64 FORMAT (4X,14HWARNING...ONLY,I3,1X,28HMORE CARDS IN ENTITIES BLOCK
X    X//)
      LET LNCNT = LNCNT + 3
X  65 READ (9,11) IWHAT,ICODE,ISTART
      IF (IWHAT) EQ (HW), GO TO 25
X     WRITE (6,33) ISTART
X  33 FORMAT (1X,45HNO W CARD FOLLOWING ENTITIES BLOCK - SEQUENCE,1X,A4)
      CALL EXIT
C     END OF PASS 1. IN 'THERE(ID)', = 9999 MEANS PCT, = 999 MEANS RCT,
C       = 998 MEANS ENTITIES, = 0 MEANS NOT USED, = 9997 MEANS ERROR,
C       GR 0 AND LS 100 MEANS TRANSIENT
   80 REWIND TAPE 9
      LET I = CMAX
      LET J = CBEGN
      CALL TEST
X     WRITE (6,81) I,J
X  81 FORMAT  (/5X,20HMAXIMUM CONDITIONS =,I4,1X,17H. TABLE BEGINS AT,
X    X 1X,A4/)
      LET LNCNT = LNCNT + 3
      LET I = AMAX
      LET J = ABEGN
      CALL TEST
X     WRITE (6,82) I,J
X  82 FORMAT (5X,17HMAXIMUM ACTIONS =,I4,1X,17H. TABLE BEGINS AT,1X,A4)
```

```
      LET LNCNT = 56
      CALL TEST
X     WRITE (6,83)
X  83 FORMAT (5X, 86HTHE FOLLOWING NUMBERS HAVE NOT BEEN USED AND ARE AV
X    XAILABLE FOR COMMON TEXT OR ENTITIES//)
      LET LNCNT = LNCNT + 3
      DO TO 85, FOR I = (9)(NIDS), WITH (THERE(I)) EQ (0)
      CALL BLOCK(I,0)
      LET THERE(I) = 9998
   85 LOOP
      CALL BLOCK(0,9)
      IF (LNCNT) GR (30), LET LNCNT = 56
      CALL TEST
X     WRITE (6,86)
X  86 FORMAT (///5X,56HTHE FOLLOWING NUMBERS WERE USED IN PERMANENT COMM
X    XON TEXT//)
      LET LNCNT = LNCNT + 6
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (9999)
      CALL BLOCK(0,9)
      LET THERE(I) = 0, FOR EACH IDS I, WITH (THERE(I)) LS (998)
      IF (LNCNT) NE (0), LET LNCNT = 56
      CALL TEST
X     WRITE (6,87)
X  87 FORMAT (5X,65HTHE FOLLOWING NUMBERS WERE USED IN REPLACEABLE COMMO
X    XN TEXT BLOCKS//)
      LET LNCNT = LNCNT + 3
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (999)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (30), LET LNCNT = 56
      CALL TEST
X     WRITE (6,88)
X  88 FORMAT (///5X,50HTHE FOLLOWING NUMBERS WERE USED IN ENTITIES BLOCK
X    XS//)
      LET LNCNT = LNCNT + 6
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (998)
      CALL BLOCK(0,9)
      GO TO 52, FOR EACH IDS I, WITH (THERE(I)) EQ (9997)
      CALL TEST
X     WRITE (6,53) I
X  53 FORMAT (1X,5HERROR,I10,53H IS DUPLICATED ACROSS COMMON TEXT AND EN
X    XTITIES BLOCKS//)
      LET LNCNT = LNCNT + 3
   52 LOOP
C     BEGIN PASS 2
      LET ILENT = 1
      LET GROUP = 0
      LET ICOMN = 0
C     SKIP PCT
X 100 READ (9,101) IS
X 101 FORMAT (72X,I4)
      IF (IS) NE (9999), GO TO 100
X 102 READ (9,11) IWHAT,ICODE,ISTART
  110 GO TO (120,130,120,130,140,150,120,130,120), ICODE
C     REPLACEABLE COMMON TEXT OR DECK END
  120 IF (ICOMN) EQ (0), GO TO 129
      IF (LNCNT) NE (0), LET LNCNT = 56
      CALL TEST
X     WRITE (6,121) ICPLC,ICOMN
X 121 FORMAT (5X,79HTHE FOLLOWING NUMBERS ARE AVAILABLE FOR REPLACEABLE
X    XCOMMON TEXT PRECEDING BLOCK,I4,1X,22HWITH STARTING SEQUENCE,1X,A4
X    X//)
      LET LNCNT = LNCNT + 3
      LET ICPLC = ICPLC + 1000
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (999)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (30), LET LNCNT = 56
      CALL TEST
X     WRITE (6,122)
X 122 FORMAT (//5X,45HTHE FOLLOWING NUMBERS WERE USED IN THIS BLOCK//)
      LET LNCNT = LNCNT + 5
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (ICPLC)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (30), LET LNCNT = 56
      CALL TEST
X     WRITE (6,124)
X 124 FORMAT (//5X,84HCOUNT OF NUMBER OF BLOCKS REQUIRING EACH REPLACEAB
X    XLE COMMON TEXT CARD IN ABOVE BLOCK//1X,10(4X,2HID,1X,5HCOUNT)//)
      LET LNCNT = LNCNT + 7
      DO TO 125, FOR EACH IDS I, WITH (THERE(I)) EQ (ICPLC)
      CALL BLOCK(I,0)
      CALL BLOCK(CASK(I),0)
  125 LOOP
      CALL BLOCK(0,9)
      DO TO 301, FOR EACH IDS I, WITH (THERE(I)) EQ (ICPLC)
```

46

```
      IF (CASK(I) - 1) 302,303,304
  302 CALL TEST
X     WRITE (6,305) I
X 305 FORMAT (1X,5HERROR,I10,1X,46HWAS INCLUDED IN ABOVE BLOCK AND NEVER
X    X REQUIRED//)
      GO TO 306
  303 CALL TEST
X     WRITE (6,307) I
X 307 FORMAT (1X, 7HWARNING,I10,1X,77HWAS REQUIRED BY ONLY ONE BLOCK AND
X    X DOES NOT BELONG IN REPLACEABLE COMMON TEXT//)
  306 LET LNCNT = LNCNT + 3
  304 LET CASK(I) = 0
      LET THERE(I) = 999
  301 LOOP
  129 IF (ICODE) EQ (9), GO TO 150
      LET ICPLC = GROUP + 1
C     PROCESS RCT
      CALL STXTIN(TEXT,ICPLC+1000,999,ICOMN)
C     TRANSIENT BLOCK
  130 LET GROUP = GROUP + 1
      CALL TXTIN(TEXT,GROUP,ICODE)
      GO TO 110
C     ENTITIES BLOCK
  140 IF (LNCNT) NE (0), LET LNCNT = 56
      CALL TEST
      CALL STXTIN(TEXT,2000,998,IENT)
      IF (LNCNT) EQ (0), GO TO 310
X     WRITE (6,311)
X 311 FORMAT (///)
      LET LNCNT = LNCNT + 4
      IF (LNCNT) GE (30), LET LNCNT = 56
  310 CALL TEST
X     WRITE (6,141) GROUP,IENT
X 141 FORMAT (5X, 70HTHE FOLLOWING NUMBERS ARE AVAILABLE FOR ENTITIES BL
X    XOCK FOLLOWING BLOCK,I4,1X,22HWITH STARTING SEQUENCE,1X,A4//)
      LET LNCNT = LNCNT + 3
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (998)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (30), LET LNCNT = 56
      CALL TEST
X     WRITE (6,147)
X 142 FORMAT (//5X,54HTHE FOLLOWING NUMBERS WERE USED IN THIS ENTITIES B
X    XLOCK//)
      LET LNCNT = LNCNT + 5
      CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) EQ (2000)
      CALL BLOCK(0,9)
      DO TO 143, FOR EACH IDS I
      LET J = THERE(I)
      LET K = ASK(I)
      IF (J) EQ (2000), GO TO 144
      IF (J) NE (998), GO TO 143
      IF (K) LS (ILENT), GO TO 143
      CALL TEST
X     WRITE (6,145) I,K
X 145 FORMAT (/5X,11HENTITY CARD,I4,1X,21HWAS REQUIRED BY BLOCK,I4,1X,
X    X24HAND IS NOT INCLUDED HERE/)
      LET LNCNT = LNCNT + 3
      GO TO 143
  144 IF (K) GE (ILENT), GO TO 149
      CALL TEST
      LET J = START(I)
X     WRITE (6,146) I,(TEXT(J,K), K = 1,12), I, TEXT(J,13)
X 146 FORMAT (/5X,11HENTITY CARD,I4,1X,39HIS INCLUDED HERE AND WAS NEVER
X    X REQUIRED/1X,12A6,I4,A4)
      LET LNCNT = LNCNT + 3
  149 LET THERE(I) = 998
  143 LOOP
      LET ILENT = GROUP + 1
      GO TO 102
C     INITIALIZATION - SEE IF ENTITIES WERE REQUIRED
  150 IF (ILENT) NE (0), LET LNCNT = 56
      CALL TEST
      DO TO 151, FOR EACH IDS I, WITH (THERE(I)) EQ (998), WITH (ASK(I))
     X GE (ILENT)
      LET J = ASK(I)
      CALL TEST
X     WRITE (6,152) I,J
X 152 FORMAT (15X,11HENTITY CARD,I4,1X,21HWAS REQUIRED BY BLOCK,I4,1X,
X    X38HWHICH HAS NO ASSOCIATED ENTITIES BLOCK/)
      LET LNCNT = LNCNT + 3
  151 LOOP
      LET ILENT = GROUP + 1
      IF (ICODE) NE (9), GO TO 130
C     END OF PASS 2
```

47

```
      REWIND TAPE 9
      IF(LNCNT) NE (0), LET LNCNT = 56
      CALL TEST
X     WRITE (6,211)
X 211 FORMAT (5X,19HSUMMARY INFORMATION//)
      LET LNCNT = LNCNT + 3
X     WRITE (6,212)
X 212 FORMAT (5X,48HTHE FOLLOWING NUMBERS MAY BE USED IN A NEW BLOCK//)
      LET LNCNT = LNCNT + 3
      CALL BLOCK(1,0), FOR EACH IDS I, WITH (THERE(I)) LS (998)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (40), LET LNCNT = 56
      CALL TEST
X     WRITE (6,203)
X 203 FORMAT (//5X,34HTHE FOLLOWING Z QUESTIONS WERE SET/)
      LET LNCNT = LNCNT + 4
      CALL BLOCK(1,0), FOR EACH ZEES I, WITH (ZTHER(I)) NE (0)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (45), LET LNCNT = 56
      CALL TEST
X     WRITE (6,204)
X 204 FORMAT (//5X,38HTHE FOLLOWING Z QUESTIONS WERE QUERIED/)
      LET LNCNT = LNCNT + 4
      CALL BLOCK(1,0), FOR EACH ZEES I, WITH (ZASK(I)) NE (0)
      CALL BLOCK(0,9)
      DO TO 225, FOR EACH ZEES I, WITH (ZTHER(I)) NE (0), WITH (ZASK(I))
     X EQ (0)
      CALL TEST
X     WRITE (6,226) I
X 226 FORMAT (/1X,20HWARNING...Z QUESTION,I3,1X,25HWAS SET BUT NEVER QUE
X    XRIED)
      LET LNCNT = LNCNT + 2
  225 LOOP
      IF (LNCNT) NE (0), LET LNCNT = 56
      CALL TEST
X     WRITE (6,205)
X 205 FORMAT (5X,83HCOUNT OF NUMBER OF BLOCKS USING EACH ID NUMBER - EXC
X    XLUDING COMMON TEXT AND ENTITIES//1X,10(4X,2HID,1X,5HCOUNT)//)
      LET LNCNT = LNCNT + 5
      DO TO 206, FOR EACH IDS I, WITH (THERE(I)) LS (998)
      CALL BLOCK(I,0)
      CALL BLOCK(COUNT(I),0)
  206 LOOP
      CALL BLOCK(0,9)
      CALL EXIT
      END


*IBFTC STXTIN
      SUBROUTINE STXTIN(TEXT,IN,NO,AOSEQ)
C     PROCESSES REPLACEABLE COMMON TEXT AND ENTITIES BLOCKS
      DIMENSION TEXT(500,13)
      LET I = 0
      LET IERR = 0
      LET IT = 0
      IF (IN) GE (2000), LET IT = 1
    5 LET I = I + 1
X     READ (9,10) (TEXT(I,J), J = 1,12), IS, TEXT(I,13)
X  10 FORMAT (12A6,I4,A4)
      IF (I) EQ (1), LET AOSEQ = TEXT(1,13)
      IF (IS) GE (9999), RETURN
      IF (IS) EQ (0), GO TO 5
      IF (IS) LE (8), GO TO 5
      LET J = THERE(IS)
      IF (J) EQ (NO), GO TO 30
      IF (IT) NE (0), GO TO 20
C     REPLACEABLE COMMON TEXT
      IF (J) EQ (IN), GO TO 33
X     ASSIGN 31 TO IGO
   50 IF (IERR) NE (0), GO TO 51
      LET IERR = 1
      IF (LNCNT) NE (0), LET LNCNT = 56
   51 CALL TEST
X     GO TO IGO, (31,34,22,24)
X  31 WRITE (6,32) (TEXT(I,J), J=1,12), IS, TEXT(I,13)
X  32 FORMAT(//74H REPLACEABLE COMMON TEXT CARD DUPLICATES PERMANENT COM
X    XMON TEXT OR ENTITIES/1X,12A6,I4.44)
   60 LET LNCNT = LNCNT + 4
      GO TO 30
X  33 ASSIGN 34 TO IGO
      GO TO 50
   34 LET K = START(IS)
```

```
X       WRITE (6,35) (TEXT(K,J), J = 1,12), IS, TEXT(K,13), (TEXT(I,J),
X      X J = 1,12), IS, TEXT(I,13)
X    35 FORMAT(//54H DUPLICATE NUMBER WITHIN REPLACEABLE COMMON TEXT BLOCK
X      X/2(1X,12A6,I4,A4/))
     61 LET LNCNT = LNCNT + 6
        GO TO 5
C       ENTITIES
     20 IF (J) EQ (IN), GO TO 21
X       ASSIGN 22 TO IGO
        GO TO 50
X    22 WRITE (6,23) (TEXT(I,J), J = 1,12), IS, TEXT(I,13)
X    23 FORMAT(//37H ENTITIES CARD DUPLICATES COMMON TEXT/1X,12A6,I4,A4)
        GO TO 60
X    21 ASSIGN 24 TO IGO
        GO TO 50
     24 LET K = START(IS)
X       WRITE (6,25) (TEXT(K,J), J = 1,12), IS, TEXT(K,13), (TEXT(I,J),
X      X J = 1,12), IS, TEXT(I,13)
X    25 FORMAT(//39H DUPLICATE NUMBER WITHIN ENTITIES BLOCK/2(1X,12A6,I4,A
X      X4/))
        GO TO 61
     30 LET THERE(IS) = IN
        LET START(IS) = I
        GO TO 5
        END


*IBFTC TEST
        SUBROUTINE TEST
        IF (LNCNT) LE (55), RETURN
X       WRITE (6,10)
X    10 FORMAT (1H1)
        LET LNCNT = 0
        RETURN
        END


*IBFTC TXTIN
        SUBROUTINE TXTIN(TEXT,IGROUP,ICODE)
        DIMENSION TEXT(500,13), ITEXT(16)
C       PROCESSES TRANSIENT BLOCKS AND THEIR DECISION TABLES
        LET I = 0
      5 LET I = I + 1
X       READ (9,10) (TEXT(I,J), J = 1,12), IS, TEXT(I,13)
X    10 FORMAT (12A6,I4,A4)
        IF (I) NE (1), GO TO 11
        IF (LNCNT) NE (0), LET LNCNT = 56
        CALL TEST
X       WRITE (6,6) IGROUP,  TEXT(1,13)
X     6 FORMAT (5X,15HTRANSIENT BLOCK,I3,1X,22HWITH STARTING SEQUENCE,1X,
X      X A4/)
        LET LNCNT = LNCNT + 2
     11 IF (IS) GE (9999), GO TO 57
        IF (IS) EQ (0), GO TO 5
        LET J = THERE(IS)
        IF (J) GE (998), GO TO 30
        IF (J) EQ (IGROUP), GO TO 35
        LET ISAVE = IS
        LET THERE(IS) = IGROUP
        LET START(IS) = I
        GO TO 5
     30 CALL TEST
X       WRITE (6,32) (TEXT(I,K), K=1,12), IS, TEXT(I,13)
X    32 FORMAT (/5X,41HNUMBER DUPLICATES COMMON TEXT OR ENTITIES/1X,12A6,
X      X I4,A4/)
        LET LNCNT = LNCNT + 4
        GO TO 5
     35 CALL TEST
        LET L = START(IS)
X       WRITE (6,37) (TEXT(L,K), K=1,12),IS,TEXT(L,13),(TEXT(I,K),K=1,12),
X      X IS,TEXT(I,13)
X    37 FORMAT (/5X,29HDUPLICATE NUMBER WITHIN BLOCK/2(1X,12A6,I4,A4/))
        LET LNCNT = LNCNT + 5
        GO TO 5
     57 IF (LNCNT) GE (25),LET LNCNT = 56
        CALL TEST
X       WRITE (6,52) IGROUP
X    52 FORMAT  (/5X,44HTHE FOLLOWING NUMBERS ARE AVAILABLE IN BLOCK,I4//)
        LET LNCNT = LNCNT + 4
        CALL BLOCK(I,0), FOR EACH IDS I, WITH (THERE(I)) LS (998), WITH
      X (THERE(I)) NE (IGROUP)
        CALL BLOCK(0,9)
```

```
      LET COUNT(I) = COUNT(I) + 1, FOR EACH IDS I, WITH (THERE(I)) EQ
     X(IGROUP)
C     PROCESS DECISION TABLES
X 100 READ (9,101) (ITEXT(I), I = 1,16)
X 101 FORMAT (2A1,I4,11A6,2A4)
      IF (ITEXT(1)) EQ (HW), GO TO 200
      IF (ITEXT(2)) NE (HZ), GO TO 105
      LET J = ITEXT(3)
      LET ZASK(J) = IGROUP
      IF (ZTHER(J)) NE (0), GO TO 100
      CALL TEST
X     WRITE (6,102) J,ITEXT(16)
X 102 FORMAT (/5X,20HWARNING...Z QUESTION,I3,1X,40HWAS QUERIED BEFORE IT
X    X WAS SET - SEQUENCE,1X,A4/)
      LET LNCNT = LNCNT + 3
      GO TO 100
  105 IF (ITEXT(1)) NE (HU), GO TO 100
C     ACTIONS
X 110 READ (9,101) (ITEXT(I), I = 1,16)
      IF (ITEXT(1)) EQ (HE), GO TO 100
      LET J = ITEXT(3)
      IF (ITEXT(2)) NE (HZ), GO TO 120
      LET ZTHER(J) = IGROUP
      GO TO 110
  120 LET ASK(J) = IGROUP
      LET I = THERE(J)
      IF (I) LS (998), GO TO 121
      IF (I) EQ (999), GO TO 122
      IF (I) NE (9998), GO TO 110
      CALL TEST
X     WRITE (6,123) J, (ITEXT(I), I = 1,16)
X 123 FORMAT (/1X,5HERROR,I10,1X,35HWAS REQUIRED AND IS NOT IN THE DECK/
X    X 1X,2A1,I4,11A6,2A4/)
      LET LNCNT = LNCNT + 4
      GO TO 110
  122 CALL TEST
X     WRITE (6,124) J, (ITEXT(I), I = 1,16)
X 124 FORMAT (/1X,5HERROR,I10,1X,69HIS A REPLACEABLE COMMON TEXT CARD TH
X    XAT IS NOT AVAILABLE TO THIS BLOCK/1X,2A1,I4,11A6,2A4/)
      LET LNCNT = LNCNT + 4
      GO TO 110
  121 IF (I) EQ (IGROUP), GO TO 110
      CALL TEST
X     WRITE (6,125) J, (ITEXT(I), I = 1,16)
X 125 FORMAT (/1X,5HERROR,I10,1X,30HIS REQUIRED AND IS NOT PRESENT/1X,
X    X 2A1,I4,11A6,2A4/)
      LET LNCNT = LNCNT + 4
      GO TO 110
  200 LET ICODE = ITEXT(3)
      DO TO 201, FOR EACH IDS I, WITH (THERE(I)) EQ (IGROUP), WITH (ASK
     X (I)) NE (IGROUP)
      CALL TEST
      LET J = START(I)
X     WRITE (6,202) I, (TEXT(J,K), K = 1,12), I, TEXT(J,13)
X 202 FORMAT (/5X,6HNUMBER,I10,1X,39HIS INCLUDED HERE AND WAS NEVER REQU
X    XIRED/1X,12A6,I4,A4/)
      LET LNCNT = LNCNT + 4
  201 LOOP
      IF (LNCNT) GE (48), LET LNCNT = 56
      CALL TEST
X     WRITE (6,210) IGROUP
X 210 FORMAT (/5X,43HTHE FOLLOWING Z QUESTIONS WERE SET IN BLOCK,I4/)
      LET LNCNT = LNCNT + 3
      CALL BLOCK(1,0), FOR EACH ZEES I, WITH (ZTHER(I)) EQ (IGROUP)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (48), LET LNCNT = 56
      CALL TEST
X     WRITE (6,211)
X 211 FORMAT (/5X,38HTHE FOLLOWING Z QUESTIONS WERE QUERIED/)
      LET LNCNT = LNCNT + 3
      CALL BLOCK(1,0), FOR EACH ZEES I, WITH (ZASK(I)) EQ (IGROUP)
      CALL BLOCK(0,9)
      IF (LNCNT) GE (45), LET LNCNT = 56
      CALL TEST
X     WRITE (6,212)
X 212 FORMAT (/5X,56HTHE FOLLOWING COMMON TEXT AND ENTITY CARDS WERE REQ
X    XUIRED/)
      LET LNCNT = LNCNT + 3
      DO TO 215, FOR EACH IDS I, WITH (THERE(I)) GE (998), WITH (ASK(I))
     X EQ (IGROUP)
      CALL BLOCK(1,0)
      LET CASK(I) = CASK(I) + 1
  215 LOOP
      CALL BLOCK(0,9)
      RETURN
      END
```

```
*IBMAP UTVAR
       ENTRY   UTVAR.
*
*ENTRY FROM MAIN PROGRAM TO DEFINE A VARIABLE UNIT
*
UTVAR. SXA     UTVX,4              SAVE RETURN INDEX
       LAS     NFILES              STOP IF LOGICAL TAPE NUMBER EXCEEDS
       TRA     USTOP                  NUMBER OF FILES IN TABLE.
       NOP
       PAC     ,4
       CLA     IOU,4               PICKUP ADDRESS OF FCB POINTER
       PAX     ,4
       TXL     USTOP-2,4,0         STOP IF UNIT IS UNDEFINED
UTVX   AXT     **,4                RESTORE RETURN INDEX
       STO     2,4                 SET LOCATION OF FCB
       TRA     1,4                 RETURN TO MAIN PROGRAM
*
       LXA     UTVX,4
       CLA*    -1,4                RESTORE UNIT DESIGNATION
USTOP  TSL     FEXEM.              ERROR, ILLEGAL UNIT REQUESTED.
       PZE     EXIT,,32            NO OPTIONAL RETURN
*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
*INPUT-OUTPUT LOGICAL UNIT TABLE
*
*ADDITIONS OR DELETIONS SHOULD BE MADE BETWEEN IOU AND NFILES
*
IOU    PZE     0
       PZE     0
       PZE     0
       PZE     0
       PZE     0
       PZE     FIL05.              FN LOGICAL UNIT 5
       PZE     FIL06.              FN LOGICAL UNIT 6
       PZE     0
       PZE     0
       PZE     FIL09.
       PZE     0
NFILES PZE     *-IOU-1
*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
       EXTERN  FIL09.
       EXTERN  FIL05.,FIL06.
       EXTERN  FEXEM.,EXIT
       END
```

Table 2

CHKOUT SYSTEM VARIABLES

Array No.	Array Name	Explanation
1	IDS	Number of Identification Numbers
2	ZEES	Number of Z questions
3	THERE	For each Identification Number, indicates which Statement List block it was last used in
3	COUNT	For Transient Identification Numbers, indicates how many blocks it was used in
4	START	For each Identification Number, indicates position in storage relative to Statement List beginning
4	LNGTH	For each Identification Number, indicates number of Statement List cards that follow and have no numbers of their own
5	CASK	For each Common Text and Entities Identification Number, indicates number of Decision Table blocks referencing the number
5	ASK	For each Identification Number, indicates which Decision Table block last referenced the number
6	ZTHER	For each Z question, indicates last Decision Table block where question was set
6	ZASK	For each Z question, indicates last Decision Table block where question was queried
7	HW	Alphabetic W
8	HU	Alphabetic U
9	HE	Alphabetic E
10	HT	Alphabetic T
11	HZ	Alphabetic Z
12	MAX	Size of Statement List storage in Editor program
13	CNTRL	Desired warning point for imminent Statement List storage overflow

Table 2 -- continued

Array No.	Array Name	Explanation
14	CMAX	Size of largest Conditions portion of a Decision Table
15	AMAX	Size of largest Actions portion of a Decision Table
16	CBEGN	Sequence number of CMAX table
17	ABEGN	Sequence number of AMAX table
18	LNCNT	Line count for carriage control

NOTE: Generator Tape is mounted on FORTRAN logical unit 9.

Appendix B

THE LOCATE PROGRAM

```
+                        1IDNOS E                    CONV IX
+                        2QUEST E                    SHIFTIX
+                        3LOOK  2    I                     X
+                        4START 2    I                     X
+                        5STOP  2    I                     X
+                        6ALPHA 1    I*                    X
+                        7BLANK      I*                    X
+                        8HZZZZ      I*                    X
+                        9HQUES      I*                    X
+                       10HACTI      I*                    X
+                       11HSTAT      I*                    X
+                       12HT         I*                    X
+                       13HU         I*                    X
+                       14HE         I*                    X
+                       15HW         I*                    X
+                       16TAPNO      I*                    X
```

```
1          16
   1        8                                  1000          IDNOS
   2        R                                  8             QUEST
   3     2 R     8   2                 R          70(11)
                                                          C 1-1
                                                          C 1-2
                                                          C 1-3
                                                          C 1-4
                                                          C 1-5
                                                          C 1-6
                                                          C 1-7
                                                          C 1-8
                                                          C 1-9
                                                          C 1-10
                                                          C 1-11
                                                          C 1-12
                                                          C 1-13
                                                          C 1-14
                                                         20 1-15
                                                          C 2-1
                                                          2 2-2
                                                          C 3-1
                                                          2 3-2
                                                          C 4-1
                                                          2 4-2
                                                          C 5-1
                                                          2 5-2
                                                          C 6-1
                                                          2 6-2
                                                          C 7-1
                                                          2 7-2
                                                          C 8-1
                                                          2 8-2
```

```
4     2 R     8   2                R          70(I1)
                                                    C 1-1
                                                    C 1-2
                                                    C 1-3
                                                    C 1-4
                                                    C 1-5
                                                    C 1-6
                                                    C 1-7
                                                    C 1-8
                                                    C 1-9
                                                    C 1-10
                                                    C 1-11
                                                    C 1-12
                                                    C 1-13
                                                    C 1-14
                                                   20 1-15
                                                    C 2-1
                                                    2 2-2
                                                    C 3-1
                                                    2 3-2
                                                    C 4-1
                                                    2 4-2
                                                    C 5-1
                                                    2 5-2
                                                    C 6-1
                                                    2 6-2
                                                    C 7-1
                                                    2 7-2
                                                    C 8-1
                                                    2 8-2
5     2 R     8   2                R          70(I1)
                                                    C 1-1
                                                    C 1-2
                                                    C 1-3
                                                    C 1-4
                                                    C 1-5
                                                    C 1-6
                                                    C 1-7
                                                    C 1-8
                                                    C 1-9
                                                    C 1-10
                                                    C 1-11
                                                    C 1-12
                                                    C 1-13
                                                    C 1-14
                                                   20 1-15
                                                    C 2-1
                                                    2 2-2
                                                    C 3-1
                                                    2 3-2
                                                    C 4-1
                                                    2 4-2
                                                    C 5-1
                                                    2 5-2
                                                    C 6-1
                                                    2 6-2
                                                    C 7-1
                                                    2 7-2
                                                    C 8-1
                                                    2 8-2
 6     1 R     8   2                            8(A1)
ABCDEFZ
 7   11   R                                     5(A6)
    OOZZZZQUESTIACTIONSTATEM
12   15   R                                     4(A1)
TUEW
16        R                                     9          TAPNO
```

```
*IBFTC CHECK
      SUBROUTINE CHECK(ITEXT,NO,ISEQ)
      DIMENSION ITEXT(12)
C     CHECKS STATEMENT LIST ONLY
      IF (LOOK(1,NO)) EQ (0), RETURN
      IF (LOOK(1,NO)) EQ (HACTI), RETURN
      IF (START(1,NO)) GR (SHIFT(ISEQ,4)), RETURN
      IF (STOP(1,NO)) LS (SHIFT(ISEQ,4)), RETURN
X     WRITE (6,10) (ITEXT(I), I = 1,12), NO, ISEQ
X  10 FORMAT (/5X,12A6,I4,5X,A4)
      RETURN
      END



*IBFTC CONV
      FUNCTION CONV(I)
      FIND FIRST, FOR EACH QUEST J, WITH (ALPHA(J)) EQ (I), IF NONE,
     X LET J = 1
      LET CONV = J
      RETURN
      END



*IBFTC ERR
      SUBROUTINE ERR(N)
      GO TO (1,2,3), N
X   1 WRITE (6,101)
X 101 FORMAT (///5X,40HATTEMPTING TO LOCATE THE SAME CARD TWICE///)
      RETURN
X   2 WRITE (6,102)
X 102 FORMAT (///5X,72HNO W CARD AFTER PERMANENT COMMON TEXT OR ENTITIES
X    X - EXECUTION TERMINATED)
      RETURN
X   3 WRITE (6,103)
X 103 FORMAT (///5X,64HNEITHER W NOR T CARD AFTER DECISION TABLE - EXECU
X    XTION TERMINATED)
      RETURN
      END



*IBFTC MAIN
      MAIN ROUTINE
      DIMENSION ITEXT(12)
      LET ITAPE = TAPNO
C     READ IN CARDS TO BE LOCATED
X  20 READ (5,10) IQ,ID,IWHICH,ISEQ,JSEQ
X  10 FORMAT (1X,A1,I4,1X,A6,59X,2A4)
      IF (ID) EQ (0), GO TO 30
X     WRITE (6,10) IQ,ID,IWHICH,ISEQ,JSEQ
      LET I = CONV(IQ)
      IF (LOOK(I,ID)) NE (0), GO TO 21
      LET LOOK(I,ID) = IWHICH
      LET J = SHIFT(ISEQ,4)
      LET START(I,ID) = J
      IF (ISEQ) EQ (BLANK), LET START(I,ID) = 0
      LET J = SHIFT(JSEQ,4)
      LET STOP(I,ID) = J
      IF (JSEQ) EQ (BLANK), LET STOP(I,ID) = HZZZZ
      GO TO 20
   21 CALL ERR(1)
      GO TO 20
C     PERMANENT COMMON TEXT OR ENTITIES
X  30 READ (ITAPE,31) (ITEXT(I), I = 1,12), NO,ISEQ
X  31 FORMAT (12A6,I4,A4)
      IF (NO) GE (9999), GO TO 40
      IF (NO) EQ (0), GO TO 30
      CALL CHECK(ITEXT,NO,ISEQ)
      GO TO 30
C     W CARD NEXT
X  40 READ (ITAPE,41) IW,ICODE
X  41 FORMAT (A1,I5)
      IF (IW) EQ (HW), GO TO 50
      CALL ERR(2)
      CALL EXIT
```

```
   50 GO TO (60,70,60,70,30,70,60,70,100), ICODE
C     REPLACEABLE COMMON TEXT
X  60 READ (ITAPE,31) (ITEXT(I), I = 1,12), NO, ISEQ
      IF (NO) GE (9999), GO TO 70
      IF (NO) EQ (0), GO TO 60
      CALL CHECK(ITEXT,NO,ISEQ)
      GO TO 60
C     TRANSIENT STATEMENT LIST
X  70 READ (ITAPE,31) (ITEXT(I), I = 1,12), NO, ISEQ
      IF (NO) GE (9999), GO TO 80
      IF (NO) EQ (0), GO TO 70
      CALL CHECK(ITEXT,NO,ISEQ)
      GO TO 70
C     DECISION TABLE
X  80 READ (ITAPE,41) IW,ICODE
      IF (IW) EQ (HW), GO TO 50
      IF (IW) EQ (HT), GO TO 81
      CALL ERR(3)
      CALL EXIT
C     CONDITIONS
X  81 READ (ITAPE,82) IW,IQ,NO,(ITEXT(I), I = 1,12),ISEQ
X  82 FORMAT (2A1,I4,11A6,2A4)
      IF (IW) EQ (HU), GO TO 90
      LET ID = CONV(IQ)
      CALL TABLE(ID,NO,ITEXT,ISEQ,HACTI)
      GO TO 81
C     ACTIONS
X  90 READ (ITAPE,82) IW,IQ,NO,(ITEXT(I),I = 1,12),ISEQ
      IF (IW) EQ (HE), GO TO 80
      LET I = HSTAT
      LET J = CONV(IQ)
      IF (J) GR (1), LET I = HQUES
      CALL TABLE(J,NO,ITEXT,ISEQ,I)
      GO TO 90
  100 CALL EXIT
      END


*IBMAP SHIFT
       ENTRY   GSHIFT
       AXT     **,4
GSHIFT TRA     **
       SXA     *-2
       LAC     GSHIFT,4
       LDQ*    3,4
       MPY     =6
       STQ     IN
       CLA     IN
       STA     D
       CAL     B
       LDQ*    2,4
D      LGL     **
       SLW*    4,4
       TRA     GSHIFT-1
B      PZE     0
IN     PZE     0
       END


*IBFTC TABLE
      SUBROUTINE TABLE (I,NO,ITEXT,ISEQ,IDONT)
      DIMENSION ITEXT(12)
C     CHECKS DECISION TABLE CARD
      IF (LOOK(I,NO)) EQ (0), RETURN
      IF (LOOK(I,NO)) EQ (IDONT), RETURN
      IF (START(I,NO)) GR (SHIFT(ISEQ,4)), RETURN
      IF (STOP(I,NO)) LS (SHIFT(ISEQ,4)), RETURN
      LET IQ = ALPHA(I)
X     WRITE (6,10) IQ,NO,(ITEXT(K), K = 1,12), ISEQ
X  10 FORMAT (/6X,A1,I4,11A6,A4,5X,A4)
      RETURN
      END
```

Table 3

LOCATE SYSTEM VARIABLES

Array No.	Array Name	Explanation
1	IDNOS	Number of Identification Numbers
2	QUEST	Number of Questionnaire sections + 2 (one for statements and one for Z questions)
3	LOOK	Ragged table with QUEST rows, which contains type of search (QUESTION, ACTION, STATEMENT, blank)
4	START	Ragged table with QUEST rows, which contains sequence number for beginning the search
5	STOP	Ragged table with QUEST rows, which contains sequence number for ending the search
6	ALPHA	For each QUEST, the code appearing in column 2 of the request card (blank for an Identification Number, the Questionnaire section for questions)
7	BLANK	Alphanumeric blank
8	HZZZZ	Sequence number ZZZZ
9	HQUES	Alphabetic QUESTI, indication a QUESTION request
10	HACTI	Alphabetic ACTION, indicating an ACTION request
11	HSTAT	Alphabetic STATEM, indicating a STATEMENT request
12	HT	Alphabetic T
13	HU	Alphabetic U
14	HE	Alphabetic E
15	HW	Alphabetic W
16	TAPNO	FORTRAN logical unit number of the Generator Deck

Appendix C

THE KORCT PROGRAM

```
$IBFTC KORCT
C      PROGRAM TO MAKE CHANGES TO THE GENERATOR TAPE
C      MOUNT GENERATOR TAPE ON FORTRAN UNIT 8. MOUNT BLANK TAPE ON UNIT 9
C
C      KORCT MAKES CHANGES BY WRITING OUT THE GENERATOR TAPE WITH THE
C       DESIRED CORRECTIONS, ADDITIONS, AND DELETIONS ON THE BLANK TAPE
C
C      DATA DECK CONSISTS OF CORRECTIONS TO BE MADE TO THE TAPE FOLLOWED
C       BY A BLANK CARD. DECK MUST BE ORDERED BY SEQUENCE NUMBER (COLUMNS
C       77-80)
C
C      A CORRECTION CARD WITH ONLY AN ALPHABETIC X IN COLUMN 5 AND A
C       SEQUENCE NUMBER WILL DELETE THE CORRESPONDING SEQUENCE NUMBER
C       FROM THE NEW TAPE
C
C      A CORRECTION CARD WITH A SEQUENCE NUMBER CORRESPONDING TO A
C       SEQUENCE NUMBER ON THE TAPE WILL REPLACE THAT CARD ON THE NEW
C       TAPE
C
C      A CORRECTION CARD WITH A SEQUENCE NUMBER UNLIKE ANY ON THE
C       GENERATOR TAPE WILL BE ADDED TO THE NEW TAPE AT THE PROPER PLACE
C       IN SEQUENCE
C
C      PROGRAM WILL PRINT OUT ALL CHANGES MADE TO THE GENERATOR TAPE
C
C      ERROR MESSAGES WILL PRINT IF EITHER THE DATA DECK OR THE GENERATOR
C       TAPE IS OUT OF SEQUENCE. THE PROGRAM WILL STOP WRITING A NEW TAPE
C       BUT WILL CONTINUE TO CHECK FOR ADDITIONAL SEQUENCING ERRORS
C
C      IF NO CORRECTION CARDS PRECEDE THE BLANK DATA CARD, THE PROGRAM
C       WILL SEQUENCE CHECK THE GENERATOR TAPE WITHOUT WRITING ON THE
C       BLANK TAPE
C
      DIMENSION TEXT(15),NEW(15)
      INTEGER TEXT,BLANK,HW,H9,HX
      DATA HW,H9,HX,BLANK/2HW ,4H   9,4H  X ,1H /
      LSEQ = 0
      KSEQ = 0
      KCHECK = 0
      REWIND 8
      REWIND 9
   20 READ (5,11) (NEW(J), J = 1,15)
   11 FORMAT (A2,A4,11A6,2A4)
      KCHECK = KCHECK + 1
      IF (NEW(15).EQ.BLANK) GO TO 100
      M = NEW(15)
      L = 0
      CALL SHIFT(M,L,4)
      IF (L.LE.LSEQ) GO TO 500
      LSEQ = L
      LLSEQ = NEW(15)
C     READ AN OLD CARD
   30 READ (8,11) (TEXT(J), J = 1,15)
      IF (TEXT(1).EQ.HW) GO TO 70
   60 M = TEXT(15)
      K = 0
      CALL SHIFT(M,K,4)
      IF (K.LE.KSEQ) GO TO 600
      KSEQ = K
      KKSEQ = TEXT(15)
   61 IF (K-L) 50,80,91
C     INSERT A NEW CARD
   91 IF (NEW(2).NE.HX) GO TO 90
      WRITE (6,92) NEW(15)
   92 FORMAT (//1X,37HATTEMPTING TO REMOVE NONEXISTENT CARD,1X,A4,1X,16H
     X-NO ACTION TAKEN)
      GO TO 95
   90 WRITE (9,11) (NEW(J), J = 1,15)
      WRITE (6,200) (NEW(J), J = 1,15)
  200 FORMAT (/1X,13HINSERTION    ,A2,A4,11A6,2A4)
   95 READ (5,11) (NEW(J), J = 1,15)
      KCHECK = KCHECK + 1
      IF (NEW(15).EQ.BLANK) GO TO 105
      M = NEW(15)
      L = 0
      CALL SHIFT(M,L,4)
      IF (L.LE.LSEQ) GO TO 500
```

```
      LSEQ = L
      LLSEQ = NEW(15)
      GO TO 61
C     WRITE OUT OLD CARD
   50 WRITE (9,11) (TEXT(J), J = 1,15)
      GO TO 30
C     REMOVE OLD CARD
   80 IF (NEW(2).NE.HX) GO TO 81
      WRITE (6,203) (NEW(J), J = 1,15)
  203 FORMAT (/1X,13HDELETION     ,A2,A4,11A6,2A4)
      WRITE (6,204) (TEXT(J), J = 1,15)
  204 FORMAT (1X,13HDELETED      ,A2,A4,11A6,2A4)
      GO TO 20
C     REPLACE OLD CARD
   81 WRITE (9,11) (NEW(J), J = 1,15)
      WRITE (6,201) (NEW(J), J = 1,15)
  201 FORMAT (/1X,13HREPLACEMENT  ,A2,A4,11A6,2A4)
      WRITE (6,202) (TEXT(J), J = 1,15)
  202 FORMAT (1X,13HREPLACED     ,A2,A4,11A6,2A4)
      GO TO 20
   70 IF (TEXT(2).NE.H9) GO TO 60
      IF (NEW(15).NE.TEXT(15)) GO TO 71
      WRITE (6,301)
  301 FORMAT (//51H WARNING...POSSIBLE ERROR...W 9 CARD REPLACED BY...)
  305 WRITE (6,520) (NEW(J), J = 1,15)
      WRITE (9,11) (NEW(J), J = 1,15)
      IF (NEW(1).NE.HW) GO TO 310
      IF (NEW(2).EQ.H9) GO TO 110
  308 IF (NEW(15).EQ.BLANK) GO TO 110
  310 READ (5,11) (NEW(J), J = 1,15)
      M = NEW(15)
      L = 0
      CALL SHIFT(M,L,4)
      IF (L.LE.LSEQ) GO TO 500
      LSEQ = L
      LLSEQ = NEW(15)
      IF (KCHECK.EQ.1) GO TO 308
      GO TO 305
C     DATA CARD OUT OF SEQUENCE
  500 KCHECK = 1
      WRITE (6,510)
  510 FORMAT (/1X,47HERROR...THE FOLLOWING DATA CARD OUT OF SEQUENCE)
      WRITE (6,520) (NEW(J), J = 1,15)
  520 FORMAT (14X,A2,A4,11A6,2A4)
      WRITE (6,530) LLSEQ
  530 FORMAT (9X,21HLAST CORRECT CARD WAS,1X,A4)
      LSEQ = L
      LLSEQ = NEW(15)
      GO TO 308
   71 WRITE (6,12)
   12 FORMAT (/1X,58HERROR...END OF TAPE REACHED BEFORE LAST DATA CARD W
     XAS READ)
      GO TO 105
  100 READ (8,11) (TEXT(J), J = 1,15)
      M = TEXT(15)
      K = 0
      CALL SHIFT(M,K,4)
      IF (K.GT.KSEQ) GO TO 102
C     TAPE OUT OF SEQUENCE
  600 KCHECK = 1
      WRITE (6,610)
  610 FORMAT (/1X,49HERROR...THE FOLLOWING TAPE RECORD OUT OF SEQUENCE)
      WRITE (6,520) (TEXT(J), J = 1,15)
      WRITE (6,530) KKSEQ
  102 KSEQ = K
      KKSEQ = TEXT(15)
      IF (KCHECK.EQ.1) GO TO 103
  105 WRITE (9,11) (TEXT(J), J = 1,15)
  103 IF (TEXT(1).NE.HW) GO TO 100
      IF (TEXT(2).NE.H9) GO TO 100
  110 END FILE 9
      WRITE (6,999)
  999 FORMAT (///1X,10HEND OF JOB)
      REWIND 8
      REWIND 9
      CALL EXIT
      END
```

```
$IBMAP SHIFT
       ENTRY   SHIFT
       AXT     **,4
SHIFT  TRA     **
       SXA     *-2,4
       LAC     SHIFT,4
       LDQ*    4,4
       MPY     =6
       STQ     IN
       CLA     IN
       STA     D
       CHS
       ADD     =36
       STA     E
       STA     E
       CAL*    3,4
       SLW     B
       LDQ*    2,4
D      LGL     **
       SLW*    3,4
E      LGL     **
       LDQ     B
F      LGR     **
       STQ*    2,4
       TRA     SHIFT-1
B      BSS     1
IN     BSS     1
       END
```

Appendix D

THE JHIST PROGRAM

```
+T JOB  8        T TYPE  11/2 I  IFJOBS       I     JOBS   *NUMBR L       X
+                T NUMBR 12/2 I                                            X
+                T ADATE 2     F  2LJOBS       I                           X
+                T COST  3     F                                           X
+                T FINIS 4     F                                           X
+                T SHOPT 5     F                                           X
+                T SJOBS 61/2 I                                            X
+                T PJOBS 62/2 I                                            X
+                T FSTAG 71/2 I                                            X
+                T LSTAG 72/2 I                                            X
+                T DDATE 8     F                                           X
+T WORK 4         T START 1     F                     STAG1 *             X
+                T STOP  2     F                                           X
+                T PTIME 3     F                                           X
+                T SSTAG 41/2 I                                            X
+                T PRES  43/4 I  5WORD  2     I*                           X
+                T SRES  44/4 I                                           X
```

```
1          5
   1   2    Z
   3        R                                        2
   4        R                                        2
   5     2  R     2   3   2   4     R  F            4(A6)
 INTO QUEUE INTERRUPTED
                  BLANK
```

```
*IBMAP CHCKIT
       ENTRY CHCKIT
       AXT     **,4
CHCKIT TRA     **
       SXA     CHCKIT-1,4
       CLA     32546
       TZE     QUIT
       PAC     0,4
       CLA     32547
       TNZ     CHCKIT-1
LOOK2  CLA     32548
       TNZ     CHCKIT-1
       CLA     -1,4
       TNZ     CHCKIT-1
QUIT   LAC     CHCKIT,4
       CLA     =9
       STO*    2,4
       TRA     CHCKIT-1
       END



*IBFTC ERROR
      SUBROUTINE ERROR(N)
X     WRITE (6,10) N
X  10 FORMAT (///6HERROR,I10)
      RETURN
      END



*IBFTC FINDIT
      SUBROUTINE FINDIT(CST,JOB)
      FIND FIRST, FOR EACH JOB OF JOBS, WITH (COST(JOB)) EQ (CST), IF
     X NONE, GO TO 10
      RETURN
   10 CALL ERROR(1)
      CALL EXIT
      END



*IBFTC JOBST
      REPORT JOBST(JOB)
X                                                                  JOB R
X     JOB NUMBER     *   COST =     *.***   TYPE =   *, ARRIVED    *.***
 X          NUMBR(JOB)   COST(JOB)   TYPE(JOB)   ADATE(JOB)   DDATE(JOB)
                            END
                                                                       X
EPORT                                                                 X
, DUE    *.***,   FINISHED    *,***,  IN SHOP    *,******
   FINIS(JOB)   SHOPT(JOB)
               END



*IBFTC MAIN
      MAIN ROUTINE
      DIMENSION X(1002), J(1)
X     EQUIVALENCE (X,J)
      REWIND TAPE 9
      LET IJK = 0
      LET NUM = 0
X 100 READ (9) X
      LET N = 1
  101 IF (J(N)) NE (9999), GO TO 10
C     WRITE OUT ALL JOBS STILL IN MEMORY, THEN EXIT
  111 LET T = X(N+5)
X     WRITE (6,102) T
X 102 FORMAT (//5X,43HTHE FOLLOWING JOBS ARE STILL IN SHOP - TIME,F10.5
X    X //)
  103 IF JOBS IS EMPTY, GO TO 104
      REMOVE FIRST JOB FROM JOBS
      CALL JOBST(JOB)
      CALL WORKS(JOB)
      DESTROY JOB
      GO TO 103
  104 CALL EXIT
   10 IF (J(N)) GR (7), GO TO 75
      GO TO (1,2,3,3,5,6,7), J(N)
```

```
C     ARRIVAL
    1 LET NUM = NUM + 1
      CALL CHCKIT(IJK)
      IF (IJK) NE (0), GO TO 111
      CREATE JOB
      LET NUMBR(JOB) = NUM
      LET TYPE(JOB) = J(N+1)
      LET ADATE(JOB)= X(N+5)
      LET COST(JOB) = X(N+2)
      FILE JOB IN JOBS
      GO TO 75
C     FINISH
    2 CALL FINDIT(X(N+3),JOB)
      LET DDATE(JOB) = X(N+2)
      LET FINIS(JOB) = X(N+5)
      LET SHOPT(JOB) = FINIS(JOB) - ADATE(JOB)
C     WRITE OUT JOB HISTORY
      REMOVE JOB FROM JOBS
      CALL JOBST(JOB)
      CALL WORKS(JOB)
      DESTROY JOB
      GO TO 75
C     INTO QUEUE
    3 CALL CHCKIT(IJK)
      IF (IJK) NE (0), GO TO 111
      CREATE JOB CALLED QUE
      LET PRES(QUE) = J(N+1)
      LET STOP(QUE) = X(N+5)
      LET DDATE(QUE) = X(N+2)
      CALL FINDIT(X(N+3),JOB)
      LET IS = LSTAG(JOB)
      FILE QUE IN STAG(JOB)
      LET SRES(QUE) = 1
      IF (J(N)) EQ (3), GO TO 75
      LET SRES(QUE) = 2
      LET STOP(IS) = X(N+5)
      LET PTIME(IS) = STOP(IS) - START(IS)
      GO TO 75
C     OUT OF QUEUE
    5 CALL FINDIT(X(N+4),JOB)
      LET IQ = LSTAG(JOB)
      LET PTIME(IQ) = X(N+5)
      LET SHOPT(IQ) = PTIME(IQ) - STOP(IQ)
      GO TO 75
C     START WORK
    6 CALL CHCKIT(IJK)
      IF (IJK) NE (0), GO TO 111
      CALL FINDIT(X(N+3),JOB)
      LET IS = LSTAG(JOB)
      IF (STOP(IS)) EQ (0), GO TO 75
      CREATE WORK
      LET PRES(WORK) = J(N+1)
      LET SRES(WORK) = J(N+2)
      LET START(WORK) = X(N+5)
      FILE WORK IN STAG(JOB)
      GO TO 75
C     END OF PROCESS
    7 CALL FINDIT(X(N+3),JOB)
      LET IS = LSTAG(JOB)
      LET STOP(IS) = X(N+5)
      LET PTIME(IS) = STOP(IS) - START(IS)
   75 LET N = N + 6
      IF (N-1002) 101,100,100
      END



*IBFTC QUEST
      REPORT QUEST(I)
X               AT RESOURCE   *, *A*****A*** AT ***.***, OUT OF QUEUE AT
 X              PRES(I) WORD(SRES (I),1) WORD(SRES (I),2)   STOP(I)
                      END
                                                                        X
***.***, Q-TIME = ***.*****, EST-PROCESS-TIME = ***.*****               X
PTIME(I)             SHOPT(I)                        DDATE(I)
                     END
```

```
*IBFTC WORKS
      SUBROUTINE WORKS(JOB)
   10 IF STAG(JOB) IS EMPTY, RETURN
      REMOVE FIRST WORK FROM STAG(JOB)
      IF (START(WORK)) EQ (0), GO TO 20
      CALL WRKST(WORK)
      DESTROY WORK
      GO TO 10
   20 CALL QUEST(WORK)
      DESTROY JOB CALLED WORK
      GO TO 10
      END
```

```
*IBFTC WRKST
      REPORT WRKST(I)
X           WORKED ON BY PRIMARY    *, SECONDARY    *,  BEGAN WORK    *.
 X                          PRES(I)          SRES(I)          START(I)
                           END                                        X
***,  FINISHED    *.***,  PROCESS TIME    *.******                    X
          STOP(I)                        PTIME(I)
              END
```

Table 4

JHIST SYSTEM VARIABLES

PERMANENT SYSTEM VARIABLES

Array No.	Array Name	Explanation
1	FJOBS	First job in the JOBS set of jobs
2	LJOBS	Last job in the JOBS set
3	-----	Dummy variable used for WORDS
4	-----	Dummy variable used for WORDS
5	WORDS	Words used in constructing the queue reports

TEMPORARY ENTITIES AND THEIR ATTRIBUTES

NOTE: To conserve space, two entity definitions, JOB and WORK, are used to represent three entities: jobs, process reports, and queue reports.

Name	Explanation
JOB	Entity representing a job in the shop
TYPE	The job's type
NUMBR	The number assigned to the job when it entered the shop
ADATE	The time at which the job entered the shop
COST	The job's dollar value
FINIS	The time at which the job was completed, i.e., left the shop
SHOPT	The length of time that the job was in the shop (FINIS - ADATE)
SJOBS	Successor job in the JOBS set
PJOBS	Predecessor job in the JOBS set
FSTAG	First report in the STAG set of process and queue reports
LSTAG	Last report in the STAG set
DDATE	The time at which the job was scheduled to leave the shop

Table 4 -- continued

WORK	Entity representing a process report
START	The time at which processing started on the job
STOP	The time at which processing was completed or interrupted
PTIME	The amount of time spent processing the job (STOP - START)
SSTAG	Successor report in the STAG set of process and queue reports
PRES	The primary resource that processed the job
SRES	The secondary resource that processed the job
JOB	Entity representing a queue report
START	Zero for a queue report, which distinguishes it from a process report
STOP	The time at which the job went into the queue or was interrupted
PTIME	The time at which the job was removed from the queue
SSTAG	Successor report in the STAG set of process and queue reports
PRES	The primary resource at which the job queued
SRES	One means that the job went into queue, two that it was interrupted
SHOPT	The length of time that the job was in queue (PTIME - STOP)
DDATE	The estimated process time of the job

Appendix E

THE RHIST PROGRAM

```
+                T NUMBR 14/4 I  1PRESR E                                  X
+                                2SRESR E                                  X
+T JOB  4
+                T PINQ  31/2 I  3FRECR 11/2 I    RECR1 *                  X
+                T SINQ  32/2 I  3LRECR 12/2 I                             X
+                T PWRKG 41/2 I  4FINQ  11/2 I    INQ 1  *NUMBR L          X
+                T SWRKG 42/2 I  4LINQ  12/2 I                             X
+                                5FWRKG 11/2 I    WRKG1  *NUMBR L          X
+                T SRECR 11/2 I  5LWRKG 12/2 I                             X
+                T TYPE  13/4 I  6AVAIL 11/2 I                             X
+                T WHEN  2    F  6WRKNG 12/2 I                             X
+                                7NINQ  11/2 I                             X
+                                7DONTP 12/2 I                             X
+                T NOINQ 3    I  8QUIT       F*                            X
+                                9FAVLR 11/2 I    AVLR1 *                  X
+                T SECDY 31/4 I  9LAVLR 12/2 I                             X
+                T PRIMY 32/4 I 10SECAV 11/2 I                             X
+                T SWORK 33/4 I 10SCWRK 12/2 I                             X
+                                11DONTS 1    I                            X
+                T SIDLE 34/4 I                                            X
+                T PWORK 41/4 I                                            X
+                T PIDLE 42/4 I                                            X
+                T SAVLR 42/2 I                                            X
```

```
1          11
   1        R                          5              PRESR
   2        R                          5              SRESR
   3   7 1  Z    5   1
   8        R                          9999.          QUIT TIME
   9  11 1  Z    5   2                                BLANK
```

```
*IBFTC MAIN
      MAIN ROUTINE
      DIMENSION X(1002), J(1)
X     EQUIVALENCE (X,J)
      REWIND TAPE 9
      READ FROM TAPE 5, DONTP(I), FOR EACH PRESR I
      FORMAT 72(11)
      READ FROM TAPE 5, DONTS(I), FOR EACH SRESR I
      FORMAT 72(I1)
      LET IQ = 0
      LET LOOSE = 0
      LET NUM = 0
X 100 READ (9) X
      LET N = 1
  101 IF (J(N))    GE (999), GO TO 200
      IF (X(N+5))  GE (QUIT), GO TO 200
      IF (J(N)) GE (11), GO TO 150
      GO TO (1,2,3,4,5,6,7,150,9,10), J(N)
C     JOB ARRIVAL
    1 CREATE JOB
      LET NUM = NUM +1
      IF (NUM) GE (512), LET NUM = 1
      LET NUMBR(JOB) = NUM
      LET WHEN(JOB) = X(N+2)
      IF (LOOSE) NE (0), CALL ERROR(1)
      LET LOOSE = JOB
      GO TO 150
C     JOB FINISHED
    2 IF (LOOSE) EQ (0), CALL ERROR(2)
      LET I = NUMBR(LOOSE)
X     WRITE (6,30) I,X(N+1),X(N+2),X(N+3),J(N+4), X(N+5)
X  30 FORMAT (1X,3HJOB,I4,8H ARRIVED,F12.5,4H DUE,F12.5,5H COST,F12.5
X    X5H TYPE,I4,9H FINISHED,F12.5)
      DESTROY JOB CALLED LOOSE
      LET LOOSE = 0
      GO TO 150
C     JOB INTO QUEUE
    3 LET I = J(N+1)
      IF (LOOSE) EQ (0), CALL ERROR(3)
      LET JOB = LOOSE
      LET LOOSE = 0
      LET L = 1
   40 LET NINQ(I) = NINQ(I) + 1
      FILE JOB IN INQ(I)
   45 CREATE JOB CALLED ITEM
      LET NUMBR(ITEM) = NUMBR(JOB)
      LET TYPE(ITEM) = L
      LET WHEN(ITEM) = X(N+5)
      LET NOINQ(ITEM) = NINQ(I)
      FILE ITEM IN RECR(I)
      GO TO 150
C     JOB INTERRUPTED
    4 LET I = J(N+1)
      FIND FIRST, FOR EACH JOB OF WRKG(I), WITH (WHEN(JOB)) EQ (X(N+3)),
     X IF NONE, CALL ERROR(4)
      LET L = 2
      REMOVE JOB FROM WRKG(I)
      GO TO 40
C     JOB OUT OF QUEUE
    5 LET I = J(N+1)
      FIND FIRST, FOR EACH JOB OF INQ(I), WITH (WHEN(JOB)) EQ (X(N+4)),
     X IF NONE, CALL ERROR(5)
      IF (LOOSE) NE (0), CALL ERROR(6)
      LET LOOSE = JOB
      REMOVE JOB FROM INQ(I)
      LET NINQ(I) = NINQ(I) - 1
      LET L = 3
      GO TO 45
   31 LET I = J(N+1)
      LET L = 10
      LET M = 1
      FIND FIRST, FOR EACH JOB OF WRKG(I), WITH (WHEN(JOB)) EQ (X(N+3)),
     X IF NONE, CALL ERROR(7)
      GO TO 47
C     START WORK
    6 IF (LOOSE) EQ (0), GO TO 31
      LET JOB = LOOSE
      LET LOOSE = 0
      LET I = J(N+1)
      FILE JOB IN WRKG(I)
      LET M = 1
      LET L = 4
```

```
   47 CREATE JOB CALLED COMBO
      LET NUMBR(COMBO) = NUMBR(JOB)
      LET K = J(N+2)
      LET PRIMY(COMBO) = I
   48 LET AVAIL(I) = AVAIL(I) - M
      LET WRKNG(I) = WRKNG(I) + M
      LET PWORK(COMBO) = WRKNG(I)
      LET PIDLE(COMBO) = AVAIL(I)
      FILE COMBO IN RECR(I)
   51 LET WHEN(COMBO) = X(N+5)
      LET TYPE(COMBO) = L
      IF (K) EQ (0), GO TO 49
      LET SECDY(COMBO) = K
      LET SECAV(K) = SECAV(K) - M
      LET SCWRK(K) = SCWRK(K) + M
      LET SWORK(COMBO) = SCWRK(K)
      LET SIDLE(COMBO) = SECAV(K)
      FILE COMBO IN AVLR(K)
   49 IF (J(N)) LS (9), GO TO 150
      CALL CHCKIT(IQ)
      IF (IQ) NE (0), GO TO 200
      GO TO (71,72,150), IP
C     END OF PROCESS
    7 LET I = J(N+1)
      FIND FIRST, FOR EACH JOB OF WRKG(I), WITH (WHEN(JOB)) EQ (X(N+3)),
     X IF NONE, CALL ERROR(8)
      IF (LOOSE) NE (0), CALL ERROR(9)
      LET LOOSE = JOB
      REMOVE JOB FROM WRKG(I)
      LET M = -1
      LET L = 5
      GO TO 47
C     SHIFT CHANGE - PRIMARY
    9 LET K = 0
      LET L = 6
      LET IP = 1
      LET M = 0
      LET I = J(N+1)
      IF (DONTP(I)) EQ (0), GO TO 150
   61 CREATE JOB CALLED COMBO
      GO TO 48
   71 LET L = 7
      LET AVAIL(I) = J(N+2)
      LET WRKNG(I) = 0
      LET IP = 3
      GO TO 61
C     SHIFT CHANGE - SECONDARY
   10 LET K = J(N+1)
      IF (DONTS(K)) EQ (0), GO TO 150
      LET L = 8
      LET IP = 2
      LET M = 0
   62 CREATE JOB CALLED COMBO
      GO TO 51
   72 LET L = 9
      LET SECAV(K) = J(N+2)
      LET SCWRK(K) = 0
      LET IP =3
      GO TO 62
C     RECORD NOW PROCESSED
  150 CALL CHCKIT(IQ)
      IF (IQ) NE (0), GO TO 200
      LET N = N + 6
      IF (N - 1002) 101,100,100
  200 DO TO 205, FOR EACH PRESR I
      IF (DONTP(I)) EQ (0), GO TO 225
      CALL PRPT(I)
  225 IF RECR(I) IS EMPTY, GO TO 205
      REMOVE FIRST JOB FROM RECR(I)
      IF (DONTP(I)) EQ (0), GO TO 206
      GO TO (211,212,213,214,215,216,217,220,220,218), TYPE(JOB)
  220 CALL ERROR(10)
  211 CALL J1(JOB,I)
      GO TO 201
  212 CALL J2(JOB,I)
      GO TO 201
  213 CALL J3(JOB,I)
      GO TO 201
  214 CALL P1(JOB,I)
      CALL PS(JOB,I)
  206 IF (SECDY(JOB)) NE (0), GO TO 225
      GO TO 201
```

```
  215 CALL P2(JOB,I)
      CALL PS(JOB,I)
      IF (SECDY(JOB)) NE (0), GO TO 225
      GO TO 201
  216 CALL P3(JOB,I)
      GO TO 201
  218 CALL JRS(JOB,I)
      CALL PS(JOB,I)
      IF (SECDY(JOB)) NE (0), GO TO 225
      GO TO 201
  217 CALL P4(JOB,I)
  201 DESTROY JOB
      GO TO 225
  205 LOOP
      DO TO 230, FOR EACH SRESR I
      IF (DONTS(I)) EQ (0), GO TO 260
      CALL SRPT(I)
  260 IF AVLR(I) IS EMPTY, GO TO 230
      REMOVE FIRST JOB FROM AVLR(I)
      IF (DONTS(I)) EQ (0), GO TO 231
      GO TO (232,232,232,244,245,232,232,248,249,238), TYPE(JOB)
  232 CALL ERROR(11)
  238 CALL JRS(JOB,I)
      CALL PS(JOB,I)
      GO TO 231
  244 CALL S1(JOB,I)
      CALL PS(JOB,I)
      GO TO 231
  249 CALL S4(JOB,I)
      GO TO 231
  248 CALL S3(JOB,I)
      GO TO 231
  245 CALL S2(JOB,I)
      CALL PS(JOB,I)
  231 DESTROY JOB
      GO TO 260
  230 LOOP
      IF (IQ) EQ (0), GO TO 270
      LET IQ = 0
      IF (J(N)) LS (9), GO TO 150
      GO TO 101
X 270 WRITE (6,250) X(N+5)
X 250 FORMAT (//5X,20HEND HISTORY - TIME =,F10.5)
      CALL EXIT
      END
```

```
*IBMAP CHCKIT
       ENTRY   CHCKIT
CHCKIT TRA     **
       CLA     32547
       TNZ     CHCKIT
       CLA     32546
       TNZ     CHCKIT
       SXA     SAVE,4
       LAC     CHCKIT,4
       CLA     =9
       STO*    2,4
SAVE   AXT     **,4
       TRA*    CHCKIT
       END
```

```
*IBFTC ERROR
       SUBROUTINE ERROR(N)
X      WRITE (6,10) N
X  10 FORMAT (//5X,6HERROR,I10)
       CALL KRDMP
       END
```

```
*IBFTC J1
      REPORT J1(JOB,I)
X                                                                PRIMARY RESOU
 X
X     JOB NUMBER    * INTO QUEUE AT    *.***** (DAY    * HOUR  * *). PRE
 X              NUMBR(JOB)         WHEN(JOB) DPART(WHEN(JOB))  HPART(WHEN
                        END
                                                                           X
RCE      *                                                              22 X
         I
SENT NUMBER IN QUEUE =     *                                             1
(JOB)) MPART(WHEN(JOB))    NOINQ(JOB)
             END


*IBFTC J2
      REPORT J2(JOB,I)
X                                                                PRIMARY RESOU
 X
X     JOB NUMBER    * INTERRUPTED AT    *.***** (DAY    * HOUR  * *). PR
 X        NUMBR(JOB)  WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB))  MPART
                  END
                                                                           X
RCE      *                                                              22 X
         I
ESENT NUMBER IN QUEUE =    *                                             1
(WHEN(JOB))      NOINQ(JOB)
                  END


*IBFTC J3
      REPORT J3(JOB,I)
X                                                                PRIMARY RESOU
 X
X     JOB NUMBER    * OUT OF QUEUE AT    *.***** (DAY    * HOUR  * *). P
 X          NUMBR(JOB)   WHEN(JOB) DPART(WHEN(JOB))  HPART(WHEN(JOB)) MPA
                  END
                                                                           X
RCE      *                                                              22 X
         I
RESENT NUMBER IN QUEUE =    *                                            1
RT(WHEN(JOB))      NOINQ(JOB)
               END


*IBFTC JRS
      REPORT JRS(JOB,I)
X                                                                         RESOU
 X
X     JOB NUMBER    * RESTARTED SFTER SHIFT CHANGE WITH PRIMARY    *, SE
 X        NUMBR(JOB)             PRIMY(JOB)
         END
                                                                           X
RCE      *                                                              22 X
         I
CONDARY    *
         SECDY(JOB)
           END


*IBMAP KRDMP
       ENTRY   KRDMP
KRDMP  PZE     0
       PZE     0
       END


*IBFTC P1
      REPORT P1(COMBO,I)
X                                                                PRIMARY RESOU
 X
X     START WORK ON JOB NUMBER   * AT    *.***** (DAY    * HOUR  * *), W
 X                 NUMBR(COMBO)  WHEN(COMBO) DPART(WHEN(COMBO)) HPART(W
                        END
                                                                           X
RCE      *                                                              22 X
         I
ITH SECONDARY RESOURCE    *
HEN(COMBO)) MPART(WHEN(COMBO))  SECDY(COMBO)
                    END
```

```
*IBFTC P2
      REPORT P2(JOB,I)
X                                                                  PRIMARY RESOU
 X
X     END PROCESS ON JOB NUMBER   * AT    *.***** (DAY   * HOUR  * *), W
 X             NUMBR(JOB)   WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB))
                            END
                                                                             X
RCE     *                                                             22 X
        I
ITH SECONDARY RESOURCE    *
 MPART(WHEN(JOB))   SECDY(JOB)
             END


*IBFTC P3
      REPORT P3(JOB,I)
X                                                                  PRIMARY RESOU
 X
X     AT SHIFT CHANGE (DAY    * HOUR  * *) THERE ARE    * PRIMARIES OF T
 X          DPART(WHEN(JOB)) HPART(WHEN(JOB)) MPART(WHEN(JOB))  PWORK(JO
                    END
                                                                             X
RCE     *                                                             22 X
        I
HIS TYPE WORKING,    * IDLE
8)  PIDLE(JOB)
             END


*IBFTC P4
      REPORT P4(JOB,I)
X                                                                  PRIMARY RESOU
 X
X     AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS PRIMARY RESOURCE
 X
                          END
                                                                             X
RCE     *                                                             22 X
        I
IS      *                                                              1
        PIDLE(JOB)
          END


*IBFTC PRPT
      REPORT PRPT(I)
X                                                                  PRIMARY RESOU
 X
                            END
                                                                             X
RCE     *                                                              2
        I
               END


*IBFTC PS
      REPORT PS(JOB,I)
X                                                                        RESOU
 X
X        CURRENTLY OF THESE RESOURCE TYPES, THERE ARE   * PRIMARIES WORK
 X                                                    PWORK(JOB)
                                 END
                                                                             X
RCE     *                                                             22 X
        I
ING,   * IDLE,    * SECONDARIES WORKING,   * IDLE                      1
  PIDLE(JOB)    SWORK(JOB)             SIDLE(JOB)
                  END
```

```
*IBFTC S1
      REPORT S1(JOB=I)
X                                                                SECONDARY RESO
 X
X     START WORK ON JOB NUMBER   * AT    *.***** (DAY    * HOUR  * *), W
 X            NUMBR(JOB)  WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB))
                           END
                                                                             X
URCE     *                                                             22 X
         I
ITH PRIMARY RESOURCE    *
MPART(WHEN(JOB))   PRIMY(JOB)
                END



*IBFTC S2
      REPORT S2(JOB,I)
X                                                                SECONDARY RESO
 X
X     END PROCESS ON JOB NUMBER   * AT    *.***** (DAY    * HOUR  * *),
 X               NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB)) M
                    END
                                                                             X
URCE     *                                                             22 X
         I
WITH PRIMARY RESOURCE    *
PART(WHEN(JOB))      PRIMY(JOB)
          END



*IBFTC S3
      REPORT S3(JOB,I)
X                                                                SECONDARY RESO
 X
X     AT SHIFT CHANGE (DAY    * HOUR  * *) THERE ARE    * SECONDARIES OF
 X            DPART(WHEN(JOB)) HPART(WHEN(JOB))  MPART(WHEN(JOB))  SWORK
                               END
                                                                             X
URCE     *                                                             22 X
         I
 THIS TYPE WORKING,    * IDLE
(JOB)              SIDLE(JOB)
           END



*IBFTC S4
      REPORT S4(JOB,I)
X                                                                SECONDARY RESO
 X
X     AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS SECONDARY RESOURC
 X
                        END
                                                                             X
URCE     *                                                             22 X
         I
E IS     *                                                              1
      SIDLE(JOB)
        END



*IBFTC SRPT
      REPORT SRPT(I)
X                                                                SECONDARY RESO
 X
                    END
                                                                             X
URCE     *                                                              2
         I
              END
```

Table 5

RHIST SYSTEM VARIABLES[a]

PERMANENT SYSTEM VARIABLES

Array No.	Array Name	Explanation
1	PRESR	Permanent entity initialized to the number of primary resources in the job shop
2	SRESR	Permanent entity initialized to the number of secondary resources
3	FRECR	First report in the RECR set of records relating to a primary resource
3	LRECR	Last report in the RECR set
4	FINQ	First job in the INQ set of jobs currently in queue at a primary resource
4	LINQ	Last job in the INQ set
5	FWRKG	First job in the WRKG set of jobs being processed by a primary resource
5	LWRKG	Last job in the WRKG set
6	AVAIL	Number of available units of a primary resource
6	WRKNG	Number of busy units of a primary resource
7	NINQ	Number of jobs currently in the queue of a primary resource
7	DONTP	Flag indicating whether or not reports are desired for primary resource
8	QUIT	Variable initialized to the simulated time at which reports are to terminate
9	FAVLR	First report in the AVLR set of records relating to a secondary resource
9	LAVLR	Last report in the AVLR set
10	SECAV	Number of available units of a secondary resource
10	SCWRK	Number of busy units of a secondary resource

[a]See listing on p. 64.

Table 5 -- continued

11	DONTS	Flag indicating whether or not reports are desired for the secondary resource

TEMPORARY ENTITIES AND THEIR ATTRIBUTES

NOTE: One entity definition, JOB, is used for all record types

Name	Explanation
Job Record:	Represents a job in the shop
NUMBR	Number assigned to the job when it entered the shop
WHEN	The job's dollar value
PINQ	Predecessor job in the INQ set of jobs in queue at a primary resource
SINQ	Successor job in the INQ set
PWRKG	Predecessor job in the WRKG set of jobs being processed by a primary resource
SWRKG	Successor job in the WRKG set
Shift or process record:	Represents a change in the status of a resource
SRECR	Successor in the RECR set of records relating to a primary resource
TYPE	The type of occurrence that the record represents
NUMBR	For a process record, the number of the job -- not used for a shift record
WHEN	The time of the occurrence
SECDY	The secondary resource affected by the occurrence
PRIMY	The primary resource affected by the occurrence
SWORK	The number of busy units at the secondary resource after the occurrence
SIDLE	The number of available units at the secondary resource after the occurrence.

Table 5 -- continued

PWORK	The number of busy units at the primary resource after the occurrence
PIDLE	The number of available units at the primary resource after the occurrence
SAVLR	Successor in the AVLR set of records relating to a secondary resource

Queue or interrupt record: Represents a change in the queue status at a primary resource

SRECR	Successor in the RECR set of records relating to a primary resource
TYPE	The type of change that the record represents
NUMBR	The number of the job causing the change
WHEN	The time of the change
NOINQ	The number of jobs in the queue after the change