PROGRAMMING BY QUESTIONNAIRE: AUXILIARY PROGRAMS
Patricia L. Love and Paula M. Oldfather

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND
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. F41620-67-C-0015—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.
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.

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II.</td>
<td>GENERATOR DECK PROGRAMS</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Contents and Organization of the Generator Deck</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Sequencing</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Statement List and Decision Tables</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Blocking</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Editor Control Cards</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>The CHKOUT Program</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Block Sizes</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Deck Order Errors</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Identification Numbers</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Block-by-Block Information</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Replaceable Common Text Blocks</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Entities Blocks</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Transient Blocks</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Summary Information</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>The LOCATE Program</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>The KORCT Program</td>
<td>22</td>
</tr>
<tr>
<td>III.</td>
<td>JSSPG ANALYSIS PROGRAMS</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Debugging the JSSPG</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>The JSSPG Analysis Tape</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>The JHIST Program</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>The RHIST Program</td>
<td>30</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A.</td>
<td>THE CHKOUT PROGRAM</td>
<td>33</td>
</tr>
<tr>
<td>B.</td>
<td>THE LOCATE PROGRAM</td>
<td>45</td>
</tr>
<tr>
<td>C.</td>
<td>THE KORCT PROGRAM</td>
<td>51</td>
</tr>
<tr>
<td>D.</td>
<td>THE JHIST PROGRAM</td>
<td>55</td>
</tr>
<tr>
<td>E.</td>
<td>THE RHIST PROGRAM</td>
<td>62</td>
</tr>
</tbody>
</table>
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.

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

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

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 AAD0

THERE ARE 106 CARDS IN BLOCK  1 STARTING AT SEQUENCE ABD0
THERE ARE 126 CARDS IN BLOCK  2 STARTING AT SEQUENCE ALOO
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 BP00
THERE ARE 177 CARDS IN BLOCK  9 STARTING AT SEQUENCE BV50
THERE ARE 226 CARDS IN BLOCK 10 STARTING AT SEQUENCE CFU0
THERE ARE 184 CARDS IN BLOCK 11 STARTING AT SEQUENCE CON0
THERE ARE 21 CARDS IN BLOCK 12 STARTING AT SEQUENCE CWO0
THERE ARE 250 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 EFO0
THERE ARE 194 CARDS IN BLOCK 15 STARTING AT SEQUENCE FLGO
THERE ARE 247 CARDS IN BLOCK 16 STARTING AT SEQUENCE GDQ0
THERE ARE 171 CARDS IN BLOCK 17 STARTING AT SEQUENCE HPS0
THERE ARE 188 CARDS IN BLOCK 18 STARTING AT SEQUENCE HPX0
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 JNIB

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 check-
ing 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

147 150 165 238 259 260 261 262 264 266 267 268 291 292 293 352 355 356 357 358
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
536 550 560 561 573 574 575 580 582 587 588 595 597 599 653 657 659 662 663 664
645 666 674 675 676 677 678 679 686 687 696 697 699 757 758 798 813 814 815 816
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
428 429 430 431 432 433 434 435 436 437 453 552 555 557 562 563 565 571 572
428 701 702 703 704 705 706 707 711 712 713 714 715 721 722 723 725 732 733 741
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
22AAUO
FREE NUMBER OF UNITS OF THIS PRIMARY RESOURCE AVAILABLE.
22ABEO

REPLACEABLE COMMON TEXT CARD DUPLICATES PERMANENT COMMON TEXT OR ENTITIES
THE NEXT CARD OF SUB-DECK 9A1 CONTAINS...
581ABHO

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

<table>
<thead>
<tr>
<th>ID COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
<th>1D COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3</td>
<td>12</td>
<td>2</td>
<td>21</td>
<td>2</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>524</td>
<td>2</td>
<td>556</td>
<td>2</td>
<td>578</td>
<td>3</td>
<td>584</td>
<td>3</td>
</tr>
<tr>
<td>999</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

DUPLICATE NUMBER WITHIN ENTITIES BLOCK
+T PRIOD
+T QUTCM

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

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

Fig. 5 -- CHKOUT Output for an Entities Block
TRANSIENT BLOCK 13 WITH STARTING SEQUENCE CZAQ

NUMBER DUPLICATES COMMON TEXT OR ENTITIES
DO TO 10, FOR I = (11)(NO)

DUPLICATE NUMBER WITHIN BLOCK
GO TO 5
READ FREE(1), FOR EACH PRESR 1

THE FOLLOWING NUMBERS ARE AVAILABLE IN BLOCK 13

<table>
<thead>
<tr>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>52</th>
<th>53</th>
<th>54</th>
<th>55</th>
<th>56</th>
<th>57</th>
<th>58</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
<td>110</td>
<td>111</td>
<td>112</td>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
<td>117</td>
<td>118</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>122</td>
<td>123</td>
<td>124</td>
<td>125</td>
<td>126</td>
<td>127</td>
<td>128</td>
<td>129</td>
<td>130</td>
<td>131</td>
<td>132</td>
<td>133</td>
<td>134</td>
<td>135</td>
<td>136</td>
<td>137</td>
<td>138</td>
<td>139</td>
<td>140</td>
<td>141</td>
</tr>
<tr>
<td>143</td>
<td>144</td>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>149</td>
<td>150</td>
<td>151</td>
<td>152</td>
<td>153</td>
<td>154</td>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>159</td>
<td>160</td>
<td>161</td>
<td>162</td>
</tr>
</tbody>
</table>

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 2 QUESTIONS WERE SET IN BLOCK 13
15  20  21  25  26

THE FOLLOWING 2 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 | 32 | 33 | 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 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 |
| 196 | 197 | 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 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 |

### 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
<table>
<thead>
<tr>
<th>Id Count</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>10</td>
<td>28</td>
<td>10</td>
<td>28</td>
<td>10</td>
<td>13</td>
<td>31</td>
<td>10</td>
<td>31</td>
<td>10</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>10</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>70</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>28</td>
<td>10</td>
<td>28</td>
<td>10</td>
<td>13</td>
<td>31</td>
<td>10</td>
<td>13</td>
<td>31</td>
<td>10</td>
<td>13</td>
</tr>
</tbody>
</table>

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 CDBO.

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

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>Action code</th>
<th>Question and condition</th>
<th>Conditions</th>
<th>Actions</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
</tr>
<tr>
<td>119</td>
<td>3</td>
<td>ACTION</td>
<td>1234</td>
<td>234</td>
<td>345</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>QUESTION</td>
<td>2345</td>
<td>543</td>
<td>321</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>STATEMENT</td>
<td>2345</td>
<td>543</td>
<td>321</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>ACTION</td>
<td>1234</td>
<td>234</td>
<td>345</td>
</tr>
</tbody>
</table>

**LAST CARD**
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
DELETED - 193 X X X X
REPLACEMENT A 5 N N
REPLACED A 4 N N

WARNING...POSSIBLE ERROR...W 9 CARD REPLACED BY...
T ERROR CHECK
ERROR...THE FOLLOWING DATA CARD OUT OF SEQUENCE
B 10 N N
LAST CORRECT CARD WAS KHJO

END OF JOB

ATTEMPTING TO REMOVE NONEXISTENT CARD DHAA -NO ACTION TAKEN
INSERTION - 162 X X X X
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 dis-
tinguished 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.

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

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 NUMBER 9 COST = 497.33 TYPE = 2, ARRIVED 2.260, 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 = 0.00032, PROCESSION TIME = 0.66227
WORKED ON BY PRIMARY 2, SECONDARY 0, BEGAN WORK 0.264, FINISHED 1.81472
AT RESOURCE 4, INTO QUEUE AT 0.866, OUT OF QUEUE AT 0.879, Q-TIME = 0.45365, PROCESSION TIME = 0.66227
WORKED ON BY PRIMARY 4, SECONDARY 0, BEGAN WORK 0.879, FINISHED 2.969, PROCESSION TIME 1.669744
AT RESOURCE 6, INTO QUEUE AT 2.969, OUT OF QUEUE AT 7.333, Q-TIME = 4.36430, PROCESSION TIME = 0.38549
WORKED ON BY PRIMARY 6, SECONDARY 0, BEGAN WORK 7.333, FINISHED 7.667, PROCESSION TIME 0.333333
AT RESOURCE 8, INTERRUPTED AT 7.667, OUT OF QUEUE AT 8.688, Q-TIME = 0.02139, PROCESSION TIME = 0.01712
WORKED ON BY PRIMARY 8, SECONDARY 0, BEGAN WORK 8.688, FINISHED 7.705, PROCESSION TIME 0.17116
WORKED ON BY PRIMARY 8, SECONDARY 0, BEGAN WORK 7.705, FINISHED 7.943, PROCESSION TIME 0.237655
WORKED ON BY PRIMARY 8, SECONDARY 0, BEGAN WORK 7.943, FINISHED 8.667, PROCESSION TIME 0.42387
AT RESOURCE 10, INTO QUEUE AT 8.667, OUT OF QUEUE AT 8.841, Q-TIME = 0.71355, PROCESSION TIME = 0.21988
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 8.841, FINISHED 9.000, PROCESSION TIME 0.159377
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 9.000, OUT OF QUEUE AT 9.300, Q-TIME = 0.29965, PROCESSION TIME = 0.215881
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 9.300, FINISHED 9.516, PROCESSION TIME 0.215881

JOB NUMBER 50 COST = 45.237 TYPE = 2, ARRIVED 1.255, DUE 2.926, FINISHED 9.531, IN SHOP 8.278500
AT RESOURCE 2, INTO QUEUE AT 1.255, OUT OF QUEUE AT 1.316, Q-TIME = 0.06336, PROCESSION TIME = 0.35257
WORKED ON BY PRIMARY 2, SECONDARY 0, BEGAN WORK 1.316, FINISHED 1.637, PROCESSION TIME 0.320520
WORKED ON BY PRIMARY 2, SECONDARY 0, BEGAN WORK 1.637, OUT OF QUEUE AT 2.291, Q-TIME = 0.65425, PROCESSION TIME = 1.95109
WORKED ON BY PRIMARY 4, SECONDARY 0, BEGAN WORK 2.291, FINISHED 4.065, PROCESSION TIME 1.773113
WORKED ON BY PRIMARY 6, INTERRUPTED AT 4.065, OUT OF QUEUE AT 4.465, Q-TIME = 0.39975, PROCESSION TIME = 0.69098
WORKED ON BY PRIMARY 6, INTERRUPTED AT 4.465, OUT OF QUEUE AT 5.333, Q-TIME = 5.000, PROCESSION TIME 0.333333
WORKED ON BY PRIMARY 8, INTERRUPTED AT 5.000, OUT OF QUEUE AT 5.503, Q-TIME = 0.06912, PROCESSION TIME 0.49991
WORKED ON BY PRIMARY 8, INTERRUPTED AT 5.503, OUT OF QUEUE AT 6.667, Q-TIME = 0.067, PROCESSION TIME 0.164091
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 6.667, FINISHED 7.000, PROCESSION TIME 0.235938
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 7.000, OUT OF QUEUE AT 7.503, Q-TIME = 6.902, PROCESSION TIME 0.235938
WORKED ON BY PRIMARY 10, SECONDARY 0, BEGAN WORK 7.503, FINISHED 9.531, PROCESSION TIME 0.198166

JOB NUMBER 61 COST = 37.542 TYPE = 2, ARRIVED 1.578, DUE 2.915, FINISHED 9.553, IN SHOP 7.877056
WORKED ON BY PRIMARY 2, SECONDARY 0, BEGAN WORK 1.578, FINISHED 1.978, PROCESSION TIME 0.301513
WORKED ON BY PRIMARY 4, INTO QUEUE AT 1.578, OUT OF QUEUE AT 4.335, Q-TIME = 2.35724, PROCESSION TIME 0.31918
WORKED ON BY PRIMARY 4, INTO QUEUE AT 4.335, OUT OF QUEUE AT 4.482, Q-TIME = 4.682, PROCESSION TIME 0.347259
WORKED ON BY PRIMARY 6, INTERRUPTED AT 4.482, OUT OF QUEUE AT 5.333, Q-TIME = 3.65129, PROCESSION TIME 0.27877
WORKED ON BY PRIMARY 6, INTERRUPTED AT 5.333, OUT OF QUEUE AT 6.787, Q-TIME = 8.587, PROCESSION TIME 0.25425
WORKED ON BY PRIMARY 8, INTERRUPTED AT 6.787, OUT OF QUEUE AT 8.333, Q-TIME = 8.832, PROCESSION TIME 0.245415
WORKED ON BY PRIMARY 10, INTERRUPTED AT 8.333, OUT OF QUEUE AT 9.333, Q-TIME = 9.553, PROCESSION TIME 0.219751

JOB NUMBER 33 COST = 158.329 TYPE = 5, ARRIVED 0.845, DUE 1.830, FINISHED 9.584, IN SHOP 6.738675
WORKED ON BY PRIMARY 1, SECONDARY 0, BEGAN WORK 0.845, FINISHED 1.506, PROCESSION TIME 0.648223
WORKED ON BY PRIMARY 4, INTO QUEUE AT 1.506, OUT OF QUEUE AT 1.594, Q-TIME = 0.08814, PROCESSION TIME 1.33650
WORKED ON BY PRIMARY 4, INTO QUEUE AT 1.594, OUT OF QUEUE AT 2.072, Q-TIME = 2.000, PROCESSION TIME 0.406192
WORKED ON BY PRIMARY 4, INTO QUEUE AT 2.072, OUT OF QUEUE AT 2.072, Q-TIME = 0.07233, PROCESSION TIME 0.930311
WORKED ON BY PRIMARY 4, INTO QUEUE AT 2.072, OUT OF QUEUE AT 2.072, Q-TIME = 3.003, PROCESSION TIME 0.484794
WORKED ON BY PRIMARY 6, INTO QUEUE AT 3.487, OUT OF QUEUE AT 9.333, Q-TIME = 5.84500, PROCESSION TIME 0.057531
WORKED ON BY PRIMARY 6, INTO QUEUE AT 9.333, FINISHED 9.391, PROCESSION TIME 0.057531
WORKED ON BY PRIMARY 9, BEGAN WORK 9.391, FINISHED 9.584, PROCESSION 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,
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 avail-
ability 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
1 IDS E
22EES E
3 THERE 11/2
3 COUNT 12/2
4 START 1
5 SCAN 11/2
5 SK 12/2
6 OTHER 11/2
62ASK 12/2
7 MW *
8 HU *
9 HE *
10 HT *
11 HZ *
12 MAX *
13 CNTRL *
14 CMAX *
15 AMAX *
16 CBEGN *
17 ABEGN *
18 LNCTN *

1 18
2 R
3 1 I 1000 1
4 1 I 72 2
5 11 R
1000 1000
18 IDS 18 IDS
22EES 22EES
12000 12000
72 72
ZBBS ZBBS
1000 1000
12000 12000
5(A1)
5(A1)

WUETZ
12 R
13 R
14 18 Z

END INITIAL

350.3 350.3
50 50
MAX MAX
CNTR2 CNTR2
*1BFTC BLOCK
SUBROUTINE BLOCK(1,J)
  X
  DATA K /0/
  DIMENSION NQR(20)
  IF (J) EQ (9), GO TO 10
  LET K = K + 1
  LET NQR(K) = 1
  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
  X
  10 IF (K) LT (20,20,30)
END

*1BFTC COUNT
SUBROUTINE COUNT(NUMBR,M,NOSEQ)
  X
  LET M = 0
  X
  10 READ (9,5) IS,NSQ
  X
  5 FORMAT (1X,5HERROR,110,1X,5X,5HNOSEQ)
  IF (M) LT (1), RETURN
  LET N = N + 1
  IF (IS) GE (9999), RETURN
  IF (IS) EQ (0), GO TO 10
  IF (NSQ) GT (8), GO TO 40
  IF (NUMBR) LS (998), GO TO 40
  CALL TEST
  X
  WRITE (6,40) IS,NSQ
  X
  40 FORMAT (1X,5HERROR,110,1X,5HNOSEQ)
  IF (M) LT (1), RETURN
  LET LNCNT = LNCNT + 1
  GO TO 10
  X
  20 LET J = THERE(IS)
  X
  30 IF (J) LS (998), GO TO 20
  IF (NUMBR) LS (998), GO TO 10
  IF (NUMBR) EQ (9999), GO TO 10
  IF (J) EQ (9999), GO TO 40
  LET THERE(IS) = 9997
  GO TO 10
  X
  30 WRITE (6,41) IS,NSQ
  X
  41 FORMAT (1X,5HERROR,110,1X,5HNOSEQ)
  IF (M) LT (1), RETURN
  LET LNCNT = LNCNT + 1
  GO TO 10
  X
  30 LET THERE(IS) = NUMBR
  GO TO 10
END

*1BFTC MAIN
MAIN ROUTINE
DIMENSION TEXT(500,13)
X
  INTEGER GROUP, PERN, PCT
  REWIND TAPE 9
  LET LNCNT = 56
  CALL TEST
C
  BEGIN PASS 1
  CALL COUNT(9999,1,NOSEQ)
  LET PCT = 1
  LET PERN = 1
  LET GROUP = 0
  CALL TEST
  X
  WRITE (4,12)
  X
  12 FORMAT (1X,5HTEXT ARE,13,1X,30HCARDS IN PERMANENT COMMON TEXT/)
  LET LNCNT = LNCNT + 2
  X
  READ (9,11) IWMAT,ICODE,ISTART
  X
  11 FORMAT (1X,5HTEXT ARE,13,1X,30HCARDS IN PERMANENT COMMON TEXT/)
  IF (IWMAT) EQ (NW), GO TO 30
  X
  WRITE (4,13) ISTART
  X
  13 FORMAT (1X,5HHERE ARE,13,1X,30HCARDS IN PERMANENT COMMON TEXT/)
  X
  CALL EXIT
  X
  30 IF (ICODE) LE (4), GO TO 25
  X
  WRITE (4,14) ISTART
  X
  14 FORMAT (1X,5HHERE ARE,13,1X,30HCARDS IN PERMANENT COMMON TEXT/)
  X
  CALL EXIT

CALL EXIT
25 GO TO (40,50,40,50,60,50,40,50,80), ICODE
C REPLACABLE COMMON TEXT BLOCK
40 CALL COUNT(999,1,NOSSEQ)
   LET PERM = PCT + 1
   CALL TEST
   X WRITE (6,15) 1,NOSSEQ
   X 15 FORMAT (5X,9THERE ARE,14,1X,N14HCARDS IN REPLACEABLE COMMON TEXT B
   X BEGINNING AT SEQUENCE,1X,A4/)
   LET LNCT = LNCT + 2
   C TRANSIENT BLOCK
   50 LET GROUP = GROUP + 1
   CALL COUNT(1,1,NOSSEQ)
   CALL TEST
   X WRITE (6,16) 1,GROUP,NOSSEQ
   X 16 FORMAT (5X,9THERE ARE,14,1X,14HCARDS IN BLOCK,13,1X,20HSTARTING A
   X SEQUENCE,1X,A4/)
   LET LNCT = LNCT + 2
   LET I = MAX - PERM - 1
   IF (I) GR (CNTRL), GO TO 31
   CALL TEST
   X WRITE (6,17) 1,GROUP
   X 17 FORMAT (5X,14HWARNING...ONLY,13,1X,19HMORE CARDS IN BLOCK,13,1X)
   LET LNCT = LNCT + 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 MAX (GE (1), GO TO 23
   LET CMAX = 1
   LET CBEGIN = 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 (1), GO TO 27
   LET AMAX = 1
   LET ABEGIN = 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,5THINVALID SIGNAL CARD FOLLOWING DECISION TABLES AT SEQ
   X XUENCE,1X,A4/)
   CALL EXIT
C ENTITIES BLOCK
60 CALL COUNT(998,1,NOSSEQ)
   CALL TEST
   X WRITE (6,62) 1,NOSSEQ
   X 62 FORMAT (5X,9THERE ARE,14,1X,45HCARDS IN ENTITIES BLOCK BEGINNING
   X XAT SEQUENCE,1X,A4/)
   LET LNCT = LNCT + 2
   LET I = MAX - PERM - 1
   IF (I) GR (CNTRL), GO TO 65
   CALL TEST
   X WRITE (6,64) 1
   X 64 FORMAT (4X,14HWARNING...ONLY,13,1X,28HMORE CARDS IN ENTITIES BLOCK
   X X/)
   X 65 READ (9,11) IWHAT,ICODE,ISTART
   IF (IWHAT) EQ (HW), GO TO 25
   X WRITE (6,33) ISTART
   X 33 FORMAT (1X,45WH W CARD FOLLOWING ENTITIES BLOCK - SEQUENCE,1X,A4/)
   CALL EXIT
C END OF PASS 1. IN 'THERE(1D)', = 9999 MEANS PCT, = 999 MEANS RCT,
C = 998 MEANS ENTITIES, = 0 MEANS NOT USED, = 9997 MEANS ERROR,
C = 0 AND LS 100 MEANS TRANSIENT
80 REWIND TAPE 9
   LET I = CMAX
   LET J = CBEGIN
   CALL TEST
   X WRITE (6,81) 1
   X 81 FORMAT (1X,20HMAXIMUM CONDITIONS =,14,1X,17H. TABLE BEGINS AT,
   X X 1X,A4/)
   LET LNCT = LNCT + 3
   LET I = AMAX
   LET J = ABEGIN
   CALL TEST
   X WRITE (6,82) 1
   X 82 FORMAT (5X,17HMAXIMUM ACTIONS =,14,1X,17H. TABLE BEGINS AT,1X,A4)
LET LNCT = 56
CALL TEST
X WRITE (6,85)
X 83 FORMAT (5X,87H THE FOLLOWING NUMBERS HAVE NOT BEEN USED AND ARE AV
X XAILABLE FOR COMMON TEXT OR ENTITIES/)
X LET LNCT = LNCT + 3
X DO TO 85, FOR I = (9)(NIDS), WITH (THERE(1)) EQ (0)
X CALL BLOCK(1,0)
X LET THERE(1) = 9998
85 LOOP
CALL BLOCK(0,9)
IF (LNCT) GE (30), LET LNCT = 56
CALL TEST
X WRITE (6,86)
X 86 FORMAT (///5X,56H THE FOLLOWING NUMBERS WERE USED IN PERMANENT COMM
X XON TEXT///)
X LET LNCT = LNCT + 6
CALL BLOCK(1,0), FOR EACH IDS 1, WITH (THERE(1)) EQ (9999)
CALL BLOCK(0,9)
LET THERE(1) = 0, FOR EACH IDS 1, WITH (THERE(1)) LS (998)
IF (LNCT) NE (0), LET LNCT = 56
CALL TEST
X WRITE (6,87)
X 87 FORMAT (///5X,56H THE FOLLOWING NUMBERS WERE USED IN REPLACEABLE COMM
X XON TEXT BLOCKS/)
X LET LNCT = LNCT + 6
CALL BLOCK(1,0), FOR EACH IDS 1, WITH (THERE(1)) EQ (9999)
CALL BLOCK(0,9)
IF (LNCT) GE (30), LET LNCT = 56
CALL TEST
X WRITE (6,88)
X 88 FORMAT (///5X,50H THE FOLLOWING NUMBERS WERE USED IN ENTITIES BLOCK
X X5/)
X LET LNCT = LNCT + 6
CALL BLOCK(1,0), FOR EACH IDS 1, WITH (THERE(1)) EQ (998)
CALL BLOCK(0,9)
GO TO 52, FOR EACH IDS 1, WITH (THERE(1)) EQ (9997)
CALL TEST
X WRITE (6,53)
X 53 FORMAT (//9X,6Hkok,110,53H IS DUPLICATED ACROSS COMMON TEXT AND EN
X XITIES BLOCKS/)
X LET LNCT = LNCT + 3
52 LOOP
C BEGIN PASS 2
LET LENT = 1
LET GROUP = 0
LET ICOMM = 0
C SKIP PCT
X 100 READ (9,101) IS
X 101 FORMAT (12X,14)
X IF (IS) NE (9999), GO TO 100
X 102 READ (9,11) IWAT,IC ode,ISTART
X 110 GO TO (120,120,120,120,120,120,120,120)
X CODE
C REPLACEABLE COMMON TEXT OR DECK END
X 120 IF (IC ode) EQ (0), GO TO 129
X IF (LNCT) NE (0), LET LNCT = 56
CALL TEST
X WRITE (6,121) ICLUD,IC ode
X 121 FORMAT (5X,79H THE FOLLOWING NUMBERS ARE AVAILABLE FOR REPLACEABLE
X XCOMMON TEXT PRECEDING BLOCK,14,1X,22H WITH STARTING SEQUENCE,1X,A4
X X/)
X LET LNCT = LNCT + 3
LET ICLUD = ICLUD + 1000
CALL BLOCK(1,0), FOR EACH IDS 1, WITH (THERE(1)) EQ (999)
CALL BLOCK(0,9)
IF (LNCT) GE (30), LET LNCT = 56
CALL TEST
X WRITE (6,122)
X 122 FORMAT (///5X,49H THE FOLLOWING NUMBERS WERE USED IN THIS BLOCK///)
X LET LNCT = LNCT + 5
CALL BLOCK(1,0), FOR EACH IDS 1, WITH (THERE(1)) EQ (CLUD)
CALL BLOCK(0,9)
IF (LNCT) GE (30), LET LNCT = 56
CALL TEST
X WRITE (6,124)
X 124 FORMAT (///5X,84H COUNT OF NUMBER OF BLOCKS REQUIRING EACH REPLACEAB
X XLE COMMON TEXT CARD IN ABOVE BLOCK///1X,10l(4X,2HID,1X,5HCOUNT///)
X LET LNCT = LNCT + 7
DO TO 125, FOR EACH IDS 1, WITH (THERE(1)) EQ (CLUD)
CALL BLOCK(1,0)
CALL BLOCK(CASK(11),0)
125 LOOP
CALL BLOCK(0,9)
DO TO 301, FOR EACH IDS 1, WITH (THERE(1)) EQ (CLUD)
IF (CASK(1) - 1) = 302, 303, 304
302 CALL TEST
X WRITE (6, 305) 1
X 305 FORMAT (1X, SERROR, 110, 1X, 46MHAS INCLUDED IN ABOVE BLOCK AND NEVER
X X REQUIRED//)
GO TO 306
303 CALL TEST
X WRITE (6, 307) 1
X 307 FORMAT (1X, 7MWARNING, 110, 1X, 77MHAS REQUIRED BY ONLY ONE BLOCK AND
X X DOES NOT BELONG IN REPLACEABLE COMMON TEXT//)
306 LET LNCNT = LNCNT + 3
304 LET CASK(1) = 0
LET THERE(1) = 999
301 LOOP
129 IF (ICODE) EQ (9), GO TO 150
LET ICPLC = GROUP + 1
C PROCESS RCT
CALL STXT1(TEXT, ICPLC+1000, 999, ICOMM)
C TRANSIENT BLOCK
130 LET GROUP = GROUP + 1
CALL TXT1(TEXT, GROUP, ICODE)
GO TO 310
C ENTITIES BLOCK
140 IF (LNCNT) NE (0), LET LNCNT = 56
CALL TEST
CALL STXT1(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, 70M THE FOLLOWING NUMBERS ARE AVAILABLE FOR ENTITIES BL
X X GOD FOLLOWING BLOCK, 14, 1X, 22M WITH STARTING SEQUENCE, 1X, A44////)
LET LNCNT = LNCNT + 3
CALL BLOCK(1, 0), FOR EACH IDS I, WITH THERE(1) EQ (998)
CALL BLOCK(0, 9)
IF (LNCNT) GE (130), LET LNCNT = 56
CALL TEST
X WRITE (6, 147)
X 142 FORMAT (///5X, 54M THE FOLLOWING NUMBERS WERE USED IN THIS ENTITIES B
X X BLOCK/)
LET LNCNT = LNCNT + 5
CALL BLOCK(1, 0), FOR EACH IDS I, WITH THERE(1) EQ (200)
CALL BLOCK(0, 9)
DO TO 143, FOR EACH IDS I
LET J = THERE(1)
LET K = ASK(1)
IF (J) EQ (2000), GO TO 144
IF (J) NE (998), GO TO 143
IF (K) LS (IENT), GO TO 143
CALL TEST
X WRITE (6, 145) 1, K
X 145 FORMAT (///5X, 11M ENTITY CARD, 14, 1X, 21M HAS REQUIRED BY BLOCK, 14, 1X,
X X 2X-ENTITY IS NOT INCLUDED HERE/)
LET LNCNT = LNCNT + 3
144 LET K) GE (IENT), GO TO 149
GO TO 143
CALL TEST
LET J = START(1)
X WRITE (6, 146) I, TEXT(J, I), K = 1, 122, 1, TEXT(J, I)
X 146 FORMAT (///5X, 11M ENTITY CARD, 14, 1X, 39M HIS INCLUDED HERE AND WAS NEVER
X X REQUIRED/1, 12A4, 14, A44)
LET LNCNT = LNCNT + 3
149 LET THERE(1) = 998
143 LOOP
LET IENT = GROUP + 1
GO TO 102
C INITIALIZATION - SEE IF ENTITIES WERE REQUIRED
150 IF (IENT) NE (0), LET LNCNT = 56
CALL TEST
DO TO 151, FOR EACH IDS I, WITH THERE(1) EQ (998), WITH ASK(1)
LET LNCNT = LNCNT + 3
151 LOOP
LET IENT = GROUP + 1
IF (ICODE) NE (9), GO TO 130
C END OF PASS 2
RE形式 TAPE 9
IF(LMCN) NE (0), LET LMCN = 56
CALL TEST
X WRITE (6,211)
X 211 FORMAT (5X,10HSUMMARY INFORMATION//)
LET LMCN = LMCN + 3
X WRITE (6,212)
X 212 FORMAT (5X,48HTHE FOLLOWING NUMBERS MAY BE USED IN A NEW BLOCK/)
LET LMCN = LMCN + 3
CALL BLOCK(1,0), FOR EACH IDS I, WITH (THER(1)) LS (998)
CALL BLOCK(0,9)
IF (LMCN) GE (40), LET LMCN = 56
CALL TEST
X WRITE (6,203)
X 203 FORMAT (///5X,34HTHE FOLLOWING Z QUESTIONS WERE SET//)
LET LMCN = LMCN + 4
CALL BLOCK(1,0), FOR EACH ZEE I, WITH (THER(1)) NE (0)
CALL BLOCK(0,9)
IF (LMCN) GE (45), LET LMCN = 56
CALL TEST
X WRITE (6,204)
X 204 FORMAT (///5X,38HTHE FOLLOWING Z QUESTIONS WERE QUERIED/)
LET LMCN = LMCN + 4
CALL BLOCK(1,0), FOR EACH ZEE IS, WITH (ZASK(1)) NE (0)
CALL BLOCK(0,9)
DO TO 225, FOR EACH ZEE I, WITH (THER(1)) NE (0), WITH (ZASK(1))
X EQ (0)
CALL TEST
X WRITE (6,226) 1
X 226 FORMAT (///1X,20HWARNING...Z QUESTION,13,1X,25MAYS SET BUT NEVER QUE
X XRIED)
LET LMCN = LMCN + 2
225 LOOP
IF (LMCN) NE (0), LET LMCN = 56
CALL TEST
X WRITE (6,205)
X 205 FORMAT (5X,83HCOUNT OF NUMBER OF BLOCKS USING EACH ID NUMBER - EXC
X XCLUDING COMMON TEXT AND ENTITIES///1X,10(4X,2HID,1X,5HCOUNT//)
LET LMCN = LMCN + 5
DO TO 206, FOR EACH IDS I, WITH (THER(1)) LS (998)
CALL BLOCK(1,0)
CALL BLOCK(COUNT(1),0)
206 LOOP
CALL BLOCK(0,9)
CALL EXIT
END

*IBFTC STXTN
SUBROUTINE STXTN(TEXT,1N,MO,ASEQ)
C PROCESSES REPLACEABLE COMMON TEXT AND ENTITIES BLOCKS
DIMENSION TEXT(500,13)
LET I = 0
LET IERR = 0
LET 1T = 0
IF (1M) GE (2000), LET 1T = 1
LET I = I + 1
X READ (9,10) (TEXT(I,J), J = 1,12), 1S, TEXT(1,13)
X 10 FORMAT (12A6,14,A4)
IF (J) EQ (1), LET ASEQ = TEXT(1,13)
IF (1S) GE (9999), RETURN
IF (1S) EQ (0), GO TO 5
IF (1S) LE (8), GO TO 5
LET J = THERE(1S)
IF (J) EQ (MS), GO TO 30
IF (1T) NE (0), GO TO 20
C REPLACEABLE COMMON TEXT
IF (J) EQ (1M), GO TO 33
X ASSIGN 31 TO 1GO
50 IF (IERR) NE (0), GO TO 51
LET IERR = 1
IF (LMCN) NE (0), LET LMCN = 56
51 CALL TEST
X GO TO 1GO, (31,34,22,24)
X 31 WRITE (6,32) (TEXT(I,J), J = 1,12), 1S, TEXT(1,13)
X 32 FORMAT (///74HREPLACEABLE COMMON TEXT CARD DUPLICATES PERMANENT COM
X XMON TEXT OR ENTITIES///1X,12A6,14,A4)
60 LET LMCN = LMCN + 4
GO TO 30
X 33 ASSIGN 34 TO 1GO
GO TO 50
34 LET K = START(1S)
WRITE (6,35) (TEXT(K,J), J = 1,12), IS, TEXT(K,13), (TEXT(1,J),
X J = 1,12), IS, TEXT(1,13)
X 35 FORMAT (/5X,H DUPLICATE NUMBER WITHIN REPLACEABLE COMMON TEXT BLOCK
X X/2X,12A6,14,AA/1)
X 61 LET LNCNT = LNCNT + 6
X GO TO 20
X C ENTRIES
X 20 IF (J) EQ (1M), GO TO 21
X ASSIGN 22 TO 1GO
X GO TO 50
X X 22 WRITE (6,23) (TEXT(K,J), J = 1,12), IS, TEXT(1,13)
X X 23 FORMAT (/37H ENTRIES CARD DUPLICATES COMMON TEXT/1X,12A6,14,AA)
X GO TO 60
X X 21 ASSIGN 24 TO 1GO
X GO TO 50
X X 24 LET K = START(IS)
X X WRITE (6,29) (TEXT(K,J), J = 1,12), IS, TEXT(K,13), (TEXT(1,J),
X X J = 1,12), IS, TEXT(1,13)
X X 25 FORMAT(/39H DUPLICATE NUMBER WITHIN ENTRIES BLOCK/2X,12A6,14,A
X X 34/1)
X GO TO 61
X 30 LET THERE(IS) = IN
X LET START(IS) = 1
X GO TO 5
X END

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

*IBFTC TXTIN
X SUBROUTINE TXTIN(TEXT,IGROUP,ICODE)
X DIMENSION TEXT(500,13), ITEXT(16)
X C PROCESSES TRANSIENT BLOCKS AND THEIR DECISION TABLES
X LET I = 0
X 5 LET I = I + 1
X X READ (9,10) (TEXT(1,J), J = 1,12), IS, TEXT(1,13)
X X 10 FORMAT (12A6,14,AA)
X IF (I) NE (1), GO TO 11
X IF (LNCNT) NE (0), LET LNCNT = 56
X CALL TEST
X WRITE (6,6) IGROUP, TEXT(1,13)
X X 6 FORMAT (5X,15HTRANSIENT BLOCK,13,1X,22H WITH STARTING SEQUENCE,1X,
X X AA/1)
X LET LNCNT = LNCNT + 2
X 11 IF (IS) GE (9999), GO TO 57
X IF (IS) EQ (0), GO TO 5
X LET J = THERE(IS)
X IF (J) GE (998), GO TO 30
X IF (J) EQ (IGROUP), GO TO 35
X LET ISAVE = IS
X LET THERE(IS) = IGROUP
X LET START(IS) = 1
X GO TO 5
X 30 CALL TEST
X WRITE (6,32) (TEXT(1,K), K=1,12), IS, TEXT(1,13)
X X 32 FORMAT (/5X,41H NUMBER DUPLICATES COMMON TEXT OR ENTRIES/1X,12A6,
X X 14,AA/1)
X LET LNCNT = LNCNT + 4
X GO TO 5
X 35 CALL TEST
X LET L = START(IS)
X X WRITE (6,37) (TEXT(L,K), K=1,12), IS, TEXT(1,13), (TEXT(1,K), K=1,12,
X X IS,TEXT(1,13)
X X 37 FORMAT (/5X,20H DUPLICATE NUMBER WITHIN BLOCK/2X,12A6,14,AA/1)
X LET LNCNT = LNCNT + 5
X GO TO 5
X 57 IF (LNCNT) GE (25), LET LNCNT = 56
X CALL TEST
X WRITE (6,52) IGROUP
X X 52 FORMAT (/5X,44H THE FOLLOWING NUMBERS ARE AVAILABLE IN BLOCK,14/1)
X LET LNCNT = LNCNT + 4
X CALL BLOCK(1,0), FOR EACH IDS I, WITH ( THERE(IS) ) LS (998), WITH
X ( THERE(IS) ) NE (IGROUP)
X CALL BLOCK(0,9)
LET COUNT(1) = COUNT(1) + 1, FOR EACH IDS I, WITH (THERE(I)) EQ X(IGROUP)
C PROCESS DECISION TABLES
X 100 READ (9,101) (I_TEXT(I), I = 1,16)
X 101 FORMAT (2A1,14,11A6,2A4)
IF (I_TEXT(1)) EQ (HM), GO TO 200
IF (I_TEXT(2)) NE (H2), GO TO 105
LET J = I_TEXT(3)
LET ZASK(J) = IGROUP
IF (ZHER(J)) NE (01), GO TO 100
CALL TEST
X WRITE (6,102) J, (I_TEXT(I), I = 1,16)
X 102 FORMAT (/X,20HWARING... Z QUESTION,13,1X,40H WAS QUERIED BEFORE IT
X WAS SET - SEQUENCE,1X,A4/)
LET LNCNT = LNCNT + 3
GO TO 100
105 IF (I_TEXT(I)) NE (MU), GO TO 100
C ACTIONS
X 110 READ (9,101) (I_TEXT(I), I = 1,16)
IF (I_TEXT(1)) EQ (HE), GO TO 100
LET J = I_TEXT(3)
IF (I_TEXT(2)) NE (H2), GO TO 120
LET ZHER(J) = IGROUP
GO TO 110
120 LET ZASK(J) = IGROUP
LET X = THERE(J)
IF (X) LS (999), GO TO 121
IF (X) EQ (999), GO TO 122
IF (X) NE (999), GO TO 110
CALL TEST
X WRITE (6,123) J, (I_TEXT(I), I = 1,16)
X 123 FORMAT (/X,5HERROR,110,1X,35H WAS REQUIRED AND IS NOT IN THE DECK/
X 1X,2A1,14,11A6,2A4/)
LET LNCNT = LNCNT + 4
GO TO 110
122 CALL TEST
X WRITE (6,124) J, (I_TEXT(I), I = 1,16)
X 124 FORMAT (/X,5HERROR,110,1X,35H IS A REPLACEABLE COMMON TEXT CARD TH
X X IS NOT AVAILABLE TO THIS BLOCK/1X,2A1,14,11A6,2A4/)
LET LNCNT = LNCNT + 4
GO TO 110
121 IF (I) EQ (IGROUP), GO TO 110
CALL TEST
X WRITE (6,125) J, (I_TEXT(I), I = 1,16)
X 125 FORMAT (/X,5HERROR,110,1X,30H REQUIRED AND IS NOT PRESENT/1X,
X 1X,2A1,14,11A6,2A4/)
LET LNCNT = LNCNT + 4
GO TO 110
200 LET ICODE = I_TEXT(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, (I_TEXT(J,K), K = 1,12), I, I_TEXT(J,13)
X 202 FORMAT (/X,6HNUMBER,110,1X,35H INCLUDED HERE AND WAS NEVER REQU
X XIRED/1X,12A,14,A4/)
LET LNCNT = LNCNT + 4
201 LOOP
IF (LNCNT) GE (48), LET LNCNT = 56
CALL TEST
X WRITE (6,210) IGROUP
X 210 FORMAT (/X,43THE FOLLOWING Z QUESTIONS WERE SET IN BLOCK,14/)
LET LNCNT = LNCNT + 3
CALL BLOCK(1,0), FOR EACH ZEES I, WITH (ZHER(I)) EQ (IGROUP)
CALL BLOCK(0,9)
IF (LNCNT) GE (48), LET LNCNT = 56
CALL TEST
X WRITE (6,211)
X 211 FORMAT (/X,30THE FOLLOWING Z QUESTIONS WERE QUERIED/
X XIRED,1X,35THE 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   UTVP,4   SAVE RETURN INDEX
  N FILES STOP IF LOGICAL TAPE NUMBER EXCEEDS
  TRA   U STOP NUMBER OF FILES IN TABLE.
  NOP
  PAC   .4
  CLA   IOU,4
  PAX   .4
  TXL   U STOP-2.4,0
        STOP IF UNIT IS UNDEFINED
  UTVP  AXT  **,4
        RESTORE RETURN INDEX
  STD   2.4
        SET LOCATION OF FCB
  TRA   1.4
        RETURN TO MAIN PROGRAM

  * LXA   UTVP,4
  CLA*  -1,4
        RESTORE UNIT DESIGNATION
  U STOP 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 N FILES

  IOU  PZE 0
  PZE 0
  PZE 0
  PZE 0
  PZE FILO5.
  PZE FILO6.
  PZE FILO9.
  PZE 0
  PZE 0
  PZE 0
  PZE 0

  N FILES PZE == IOU-1

  * EXTERN FILO9.
  EXTERN FILO5., FILO6.
  EXTERN FEXEM., EXIT

  END
<table>
<thead>
<tr>
<th>Array No.</th>
<th>Array Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDS</td>
<td>Number of Identification Numbers</td>
</tr>
<tr>
<td>2</td>
<td>ZEES</td>
<td>Number of Z questions</td>
</tr>
<tr>
<td>3</td>
<td>THERE</td>
<td>For each Identification Number, indicates which Statement List block it was last used in</td>
</tr>
<tr>
<td>3</td>
<td>COUNT</td>
<td>For Transient Identification Numbers, indicates how many blocks it was used in</td>
</tr>
<tr>
<td>4</td>
<td>START</td>
<td>For each Identification Number, indicates position in storage relative to Statement List beginning</td>
</tr>
<tr>
<td>4</td>
<td>LNGTH</td>
<td>For each Identification Number, indicates number of Statement List cards that follow and have no numbers of their own</td>
</tr>
<tr>
<td>5</td>
<td>CASK</td>
<td>For each Common Text and Entities Identification Number, indicates number of Decision Table blocks referencing the number</td>
</tr>
<tr>
<td>5</td>
<td>ASK</td>
<td>For each Identification Number, indicates which Decision Table block last referenced the number</td>
</tr>
<tr>
<td>6</td>
<td>ZTHER</td>
<td>For each Z question, indicates last Decision Table block where question was set</td>
</tr>
<tr>
<td>6</td>
<td>ZASK</td>
<td>For each Z question, indicates last Decision Table block where question was queried</td>
</tr>
<tr>
<td>7</td>
<td>HW</td>
<td>Alphabetic W</td>
</tr>
<tr>
<td>8</td>
<td>HU</td>
<td>Alphabetic U</td>
</tr>
<tr>
<td>9</td>
<td>HE</td>
<td>Alphabetic E</td>
</tr>
<tr>
<td>10</td>
<td>HT</td>
<td>Alphabetic T</td>
</tr>
<tr>
<td>11</td>
<td>HZ</td>
<td>Alphabetic Z</td>
</tr>
<tr>
<td>12</td>
<td>MAX</td>
<td>Size of Statement List storage in Editor program</td>
</tr>
<tr>
<td>13</td>
<td>CNTRL</td>
<td>Desired warning point for imminent Statement List storage overflow</td>
</tr>
<tr>
<td>Array No.</td>
<td>Array Name</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>14</td>
<td>CMAX</td>
<td>Size of largest Conditions portion of a Decision Table</td>
</tr>
<tr>
<td>15</td>
<td>AMAX</td>
<td>Size of largest Actions portion of a Decision Table</td>
</tr>
<tr>
<td>16</td>
<td>CBEGN</td>
<td>Sequence number of CMAX table</td>
</tr>
<tr>
<td>17</td>
<td>ABEGN</td>
<td>Sequence number of AMAX table</td>
</tr>
<tr>
<td>18</td>
<td>LNCNT</td>
<td>Line count for carriage control</td>
</tr>
</tbody>
</table>

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

THE LOCATE PROGRAM
11DNOS E   CONV IX
2QUEST E   SHIFT IX
3LOOK 2   X
4START 2   X
5STOP 2    X
6ALPHA 1  
7BLANK   X
8HZZZZ   X
9HQUES   X
10HACTI  
11HSTAT  X
12MT    X
13HU    X
14HE    X
15MV    X
16TAPNO  X

1       16
1       R
2       R
3       R 82

1000  8  IDNOS  11  IDNOS
1       QUEST
C1-1
C1-2
C1-3
C1-4
C1-5
C1-6
C1-7
C1-8
C1-9
C1-10
C1-11
C1-12
C1-13
C1-14
20  1-15
C2-1
C2-2
C2-3
C2-4
C2-5
C2-6
C2-7
C2-8
C2-9
C2-10
C2-11
C2-12
C2-13
C2-14
C2-15
*IBFTC CHECK

SUBROUTINE CHECK(ITEXT,NO,ISEQ)
DIMENSION ITEXT(12)
C CHECKS STATEMENT LIST ONLY
IF (LOOK(1,NO)) EQ (10), RETURN
IF (LOOK(1,NO)) EQ (HACT1), RETURN
IF (START(1,NO)) GR (SHIFT1(1SEQ,4)), RETURN
IF (STOP(1,NO)) LS (SHIFT1(1SEQ,4)), RETURN
X WRITE (6,101) (ITEXT(I), I = 1,12), NO, ISEQ
X 10 FORMAT (/5X,12A6,14,5X,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,40/ATTEMPTING TO LOCATE THE SAME CARD TWICE///)
RETURN
X 2 WRITE (6,102)
X 102 FORMAT (/5X,72/NO W CARD AFTER PERMANENT COMMON TEXT OR ENTITIES
X  x - EXECUTION TERMINATED)
RETURN
X 3 WRITE (6,103)
X 103 FORMAT (/5X,64/NEITHER 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) IO,1D,INWHICH,1SEQ,JSEQ
X 10 FORMAT (1X,A1,14,1X,A6,59X,2A4)
IF (1D) EQ (10), GO TO 30
X WRITE (6,100) IO,1D,INWHICH,1SEQ,JSEQ
LET I = CONV(1D)
IF (LOOK(1,1D)) NE (40), GO TO 21
LET LOOK(1,1D) = INWHICH
LET J = SHIFT1(1SEQ,4)
LET START(1,1D) = J
IF (1SEQ) EQ (BLANK), LET START(1,1D) = 0
LET J = SHIFT1(1SEQ,4)
LET STOP(1,1D) = J
IF (1SEQ) EQ (BLANK), LET STOP(1,1D) = HZTZZ
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 (12A6,14,A4)
IF (NO) GE (9999), GO TO 40
IF (NO) EQ (0), GO TO 30
CALL CHECK1,ITEXT,NO,ISEQ
GO TO 30
C W CARD NEXT
X 40 READ (ITAPE,41) IW,ICODE
X 41 FORMAT (A1,15)
IF (IW) EQ (MN), 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) (I TEXT(I), I = 1, 12), NO, ISEQ
IF (NO) GE (9999), GO TO 70
IF (NO) EQ (0), GO TO 60
CALL CHECK(I TEXT, NO, ISEQ)
GO TO 60
C TRANSIENT STATEMENT LIST
X 70 READ (ITAPE, 31) (I TEXT(I), I = 1, 12), NO, ISEQ
IF (NO) GE (9999), GO TO 70
IF (NO) EQ (0), GO TO 70
CALL CHECK(I TEXT, NO, ISEQ)
GO TO 70
C DECISION TABLE
X 80 READ (ITAPE, 41) IW, I CODE
IF (IW) EQ (HM), GO TO 50
IF (IW) EQ (HT), GO TO 81
CALL ERR(2)
CALL EXIT
C CONDITIONS
X 81 READ (ITAPE, 82) IW, IQ, NO, I TEXT(I), I = 1, 12, ISEQ
X 82 FORMAT (2AI, 14, 1A6, 2A4)
IF (IW) EQ (HJ), GO TO 90
LET ID = CONV(IQ)
CALL TABLE(ID, NO, I TEXT, ISEQ, HACT)
GO TO 81
C ACTIONS
X 90 READ (ITAPE, 82) IW, IQ, NO, I TEXT(I), I = 1, 12, ISEQ
IF (IW) EQ (HE), GO TO 80
LET J = HSTAT
LET LSTAT = CONV(IQ)
IF (LJ) GR (0), LET J = HQUES
CALL TABLE(J, NO, I TEXT, ISEQ, I)
GO TO 90
100 CALL EXIT
END

*IBMAP
ENTRY GSHIFT
AXT **.4
GSHIFT TRA **
SX A **.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 PABLE (1, NO, I TEXT, ISEQ, IDONT)
DIMENSION I TEXT(12)
C CHECKS DECISION TABLE CARD
IF (LOOK(1, NO)) EQ (0), RETURN
IF (LOOK(1, NO)) EQ (IDONT), RETURN
IF (START(1, NO)) GR (SHIFT(ISEQ, 4)), RETURN
IF (STOP(1, NO)) LS (SHIFT(ISEQ, 4)), RETURN
LET IQ = ALPH(A)
X WRITE (6, 10) IQ, NO, (I TEXT(K), K = 1, 12), ISEQ
X 10 FORMAT (/6X, A1, 14, 1A6, A4, 5X, A4)
RETURN
END
Table 3
LOCATE SYSTEM VARIABLES

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

THE KORCT PROGRAM
$IBFTC KORCT

PROGRAM TO MAKE CHANGES TO THE GENERATOR TAPE

MOUNT GENERATOR TAPE ON FORTRAN UNIT 6. MOUNT BLANK TAPE ON UNIT 9

KORCT MAKES CHANGES BY WRITING OUT THE GENERATOR TAPE WITH THE

DESIRED CORRECTIONS, ADDITIONS, AND DELETIONS ON THE BLANK TAPE

DATA DECK CONSISTS OF CORRECTIONS TO BE MADE TO THE TAPE FOLLOWED

BY A BLANK CARD. DECK MUST BE ORDERED BY SEQUENCE NUMBER (COLUMNS

77-80)

A CORRECTION CARD WITH ONLY AN ALPHABETIC X IN COLUMN 5 AND A

SEQUENCE NUMBER WILL DELETE THE CORRESPONDING SEQUENCE NUMBER

FROM THE NEW TAPE

A CORRECTION CARD WITH A SEQUENCE NUMBER CORRESPONDING TO A

SEQUENCE NUMBER ON THE TAPE WILL REPLACE THAT CARD ON THE NEW

TAPE

A CORRECTION CARD WITH A SEQUENCE NUMBER UNLIKE ANY ON THE

GENERATOR TAPE WILL BE ADDED TO THE NEW TAPE AT THE PROPER PLACE

IN SEQUENCE

PROGRAM WILL PRINT OUT ALL CHANGES MADE TO THE GENERATOR TAPE

ERROR MESSAGES WILL PRINT IF EITHER THE DATA DECK OR THE GENERATOR

TAPE IS OUT OF SEQUENCE. THE PROGRAM WILL STOP WRITING A NEW TAPE

BUT WILL CONTINUE TO CHECK FOR ADDITIONAL SEQUENCING ERRORS

IF NO CORRECTION CARDS PRECEDE THE BLANK DATA CARD, THE PROGRAM

WILL SEQUENCE CHECK THE GENERATOR TAPE WITHOUT WRITING ON THE

BLANK TAPE

DIMENSION TEXT(151),NEW(151)
INTEGER TEXT,BLANK,HW,H9,HX
DATA HW,H9,HX,BLANK/2MW ,AH X ,IH /
LSEQ = 0
KSEQ = 0
KCHECK = 0
REPRINT 8
REWIND 9
20 READ (5,11) (NEW(J), J = 1,15)
11 FORMAT (A2, ,A6, ,A6, ,A4)
KCHECK = KCHECK + 1
IF (NEW(15).EQ.BLANK) GO TO 100
M = NEW(15)
L = 0
CALL SHIFT(M,L,A)
IF (L.LE.LSEQ) GO TO 500
LSEQ = L
CALL LSEQ = 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,A)
IF (K.LE.KSEQ) GO TO 600
KSEQ = K
KSEQ = TEXT(15)
61 IF (K.LT.L) 50,80,91
C INSERT A NEW CARD
91 IF (NEW(2).NE.HX) GO TO 90
WRITE (*,92) NEW(15)
92 FORMAT (//I6,1X,37HATTEMPTING TO REMOVE NONEXISTENT CARD,1X,A4,1X,16H
X-N0 ACTION TAKEN)
GO TO 95
90 WRITE (9,11) (NEW(J), J = 1,15)
WRITE (6,200) (NEW(J), J = 1,15)
200 FORMAT (/I6,1X,37HINSERTION ,A2, ,A6, ,A6, ,A4)
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,A)
IF (L.LE.LSEQ) GO TO 500
LSEQ = L
LLSEQ = NEW(15)
GO TO 61
C
50 WRITE (9,113) (TEXT(J), J = 1,15)
GO TO 30
C
REPLACE OLD CARD
80 IF (NEW(2).NE.64) GO TO 81
WRITE (6,203) (NEW(J), J = 1,15)
203 FORMAT (/X,$HDELETION ,A2,A4,11A6,2A4)
WRITE (6,204) (TEXT(J), J = 1,15)
204 FORMAT (/X,$HDELETED ,A2,A4,11A6,2A4)
GO TO 20
C
REPLACE OLD CARD
81 WRITE (9,113) (NEW(J), J = 1,15)
WRITE (6,205) (NEW(J), J = 1,15)
201 FORMAT (/X,$HREPLACEMENT ,A2,A4,11A6,2A4)
WRITE (6,202) (TEXT(J), J = 1,15)
202 FORMAT (/X,$HREPLACED ,A2,A4,11A6,2A4)
GO TO 20
70 IF (TEXT(1).NE.96) GO TO 60
IF (NEW(15).NE.TEXT(15)) GO TO 71
WRITE (6,301)
301 FORMAT (/S13H WARNING...POSSIBLE ERROR...W 9 CARD REPLACED BY...
305 WRITE (6,520) (NEW(J), J = 1,15)
WRITE (9,113) (NEW(J), J = 1,15)
IF (NEW(1).NE.MN) GO TO 310
IF (NEW(21).EQ.96) GO TO 110
308 IF (NEW(15).EQ.BLANK) GO TO 110
310 READ (5,111) (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 (/X,$HERROR...THE FOLLOWING DATA CARD OUT OF SEQUENCE)
WRITE (6,520) (NEW(J), J = 1,15)
520 FORMAT (/X,$HREPLACEMENT ,A4,11A6,2A4)
WRITE (6,530) LLSEQ
530 FORMAT (/X,$HREPLACED ,A4,11A6,2A4)
LSEQ = L
LLSEQ = NEW(15)
GO TO 308
71 WRITE (6,12)
12 FORMAT (/X,$HERROR...END OF TAPE REACHED BEFORE LAST DATA CARD W
XAS READ)
GO TO 105
100 READ (8,111) (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 (/X,$HERROR...THE FOLLOWING TAPE RECORD OUT OF SEQUENCE)
WRITE (6,520) (TEXT(J), J = 1,15)
WRITE (6,530) KSEQ
102 KSEQ = K
KSEQ = TEXT(15)
IF (KCHECK.EQ.1) GO TO 103
105 WRITE (9,113) (TEXT(J), J = 1,15)
103 IF (TEXT(1).NE.MN) GO TO 100
IF (TEXT(2).NE.MN) GO TO 100
110 END FILE 9
WRITE (6,999)
999 FORMAT (/S1X,$HEND OF JOB)
REWIND 8
REWIND 9
CALL EXIT
END
S1BMAP
ENTRY SHIFT
AHT  **.4
SHIFT TAA  **
  SXA  +=2.4
  LAC  SHIFT.4
  LOQ=  4.4
  MPY  =6
  STQ  IN
  CLA  IN
  STA  D
  CMS
  ADD= 36
  STA  E
  STA  E
  CAL= 3.4
  SLW= B
  LOQ= 2.4
  D
  LGL= **
  SLW= 3.4
  E
  LGL= **
  LOQ= B
  F
  LGR= **
  STQ= 2.4
  B
  BSS= 1
  IN
  BSS= 1
  END
Appendix D

THE JHIST PROGRAM
+T JOB 8
  T TYPE 11/2 I 1FJOBS I 1JOBS I NUMBR L X
  +
  +
  T NUMBR 12/2 I 2LJOBS X X
  +
  +
  T ADATE 2 F 2LJOBS X X
  +
  +
  T ADAT 3 F X
  +
  +
  T FINIS 4 F X
  +
  +
  T SHOPT 5 F X
  +
  +
  T SJJOBS 61/2 I X
  +
  +
  T PJOBS 62/2 I X
  +
  +
  T PSTAG 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 1 *
  +
  +
  T SRES 44/4 I X

1 2 5
1 2 2
3 R
4 R
5 2 R 2 3 4 R F 4(A6)
INTO QUEUE INTERRUPTED
BLANK
*IBMAP CHKKIT
ENTRY CHKKIT
AXT **,4
CHKKIT TRA **
SXA CHKKIT-1,4
CLA 32546
TZE QUIT
PAC 0,4
CLA 32547
TNZ CHKKIT-1
LOOK2 CLA 32548
TNZ CHKKIT-1
CLA -1,4
TNZ CHKKIT-1
QUIT LAC CHKKIT,4
CLA =0
STO* 2,4
TRA CHKKIT-1
END

*IBFCTC ERROR
SUBROUTINE ERROR(N)
X WRITE (6,10) N
X 10 FORMAT (///6ERROR,N)
RETURN
END

*IBFCTC 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

*IBFCTC JOBST
REPORT JOBST(JOB)
X JOB NUMBER * COST = * TYPE = * ARRIVED * * * *
X NUMBR(JOB) COST(JOB) TYPE(JOB) ADATE(JOB) DDATE(JOB)
END

EPORT
DUE * * * * FINISHED * * * * IN SHOP * * * * * * FINISH(JOB) SHOPT(JOB)
END

*IBFCTC MAIN
MAIN ROUTINE
X DIMENSION X(1002), J(1)
X EQUIVALENCE (X,J)
X REWRITE TAPE 9
X LET JMK = 0
X LET NUM = 0
X 100 READ (9) X
X LET N = 3
101 IF (J(M)) NE (9999), GO TO 10
C WRITE OUT ALL JOBS STILL IN MEMORY, THEN EXIT
111 LET T = X(H+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
X REMOVE FIRST JOB FROM JOBS
X CALL JOBST(JOB)
X CALL WORKS(JOB)
X DESTROY JOB
X GO TO 103
X 104 CALL EXIT
10 IF (J(M)) GR (7), GO TO 75
GO TO (1,2,3,3,5,6,7), J(M)
C
ARRIVAL
1 LET NUM = NUM + 1
CALL CHECK1(IJK)
IF (IJK) NE 101, GO TO 111
CREATE JOB
LET NUMBER(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 FIND1(X(N+3),JOB)
LET DDATE(JOB) = X(N+2)
LET FINISH(JOB) = X(N+5)
LET SHOPT(JOB) = FINISH(JOB) - ADATE(JOB)
C
WRITE OUT JOB HISTORY
REMOVE JOB FROM JOBS
CALL JOBST1(JOB)
CALL WORKS1(JOB)
DESTROY JOB
GO TO 75
C
INTO QUEUE
3 CALL CHECK1(IJK)
IF (IJK) NE 101, GO TO 111
CREATE JOB CALLED QUE
LET PRESQUE = J(N+1)
LET STOPQUE = X(N+5)
LET DDATEQUE = X(N+2)
CALL FIND1(X(N+3),JOB)
LET IS = LSTAG1(JOB)
FILE QUE IN STAG1(JOB)
LET SRESQUE = 1
IF (J(N)) EQ 31, GO TO 75
LET SRESQUE = 2
LET STOPIS = X(N+5)
LET PTIMEIS = STOPIS - STARTIS
GO TO 75
C
OUT OF QUEUE
5 CALL FIND1(X(N+4),JOB)
LET IQ = LSTAG1(JOB)
LET PTIMEIQ = X(N+5)
LET SHOPTIQ = PTIMEIQ - STOPIQ
GO TO 75
C
START WORK
6 CALL CHECK1(IJK)
IF (IJK) NE 101, GO TO 111
CALL FIND1(X(N+3),JOB)
LET IS = LSTAG1(JOB)
IF (STOPIS) EQ 101, GO TO 75
CREATE WORK
LET PRESSWORK = J(N+1)
LET SRESWORK = J(N+2)
LET STARTWORK = X(N+5)
FILE WORK IN STAG1(JOB)
GO TO 75
C
END OF PROCESS
7 CALL FIND1(X(N+3),JOB)
LET IS = LSTAG1(JOB)
LET STOPIS = X(N+5)
LET PTIMEIS = STOPIS - STARTIS
75 LET N = M + 6
IF (N>1002) 101,100,100
END

*IBFTC QUEST
REPORT QUEST(I)
X
AT RESOURCE = *****, EST-PROCESS-TIME = *****, OUT OF QUEUE AT
X
PRES(11) WORDSRES (11),1 WORDSRES (11),2 STOP(1)
END

****, Q-TIME = *****, EST-PROCESS-TIME = *****, X
PTIME(1)
SHOPT(1)
DDATE(1)
END
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:

REPORT WRKST(1)
WORKED ON BY PRIMARY *, SECONDARY *, BEGAN WORK *.
  X
  X
PRES(I) SRES(I) START(I)
END

***, FINISHED *****, PROCESS TIME *****
STOP(I) PTIME(I)
END
Table 4

JHIST SYSTEM VARIABLES

PERMANENT SYSTEM VARIABLES

<table>
<thead>
<tr>
<th>Array No.</th>
<th>Array Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FJOBS</td>
<td>First job in the JOBS set of jobs</td>
</tr>
<tr>
<td>2</td>
<td>LJOBS</td>
<td>Last job in the JOBS set</td>
</tr>
<tr>
<td>3</td>
<td>----</td>
<td>Dummy variable used for WORDS</td>
</tr>
<tr>
<td>4</td>
<td>----</td>
<td>Dummy variable used for WORDS</td>
</tr>
<tr>
<td>5</td>
<td>WORDS</td>
<td>Words used in constructing the queue reports</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB</td>
<td>Entity representing a job in the shop</td>
</tr>
<tr>
<td>TYPE</td>
<td>The job's type</td>
</tr>
<tr>
<td>NUMBR</td>
<td>The number assigned to the job when it entered the shop</td>
</tr>
<tr>
<td>ADATE</td>
<td>The time at which the job entered the shop</td>
</tr>
<tr>
<td>COST</td>
<td>The job's dollar value</td>
</tr>
<tr>
<td>FINIS</td>
<td>The time at which the job was completed, i.e., left the shop</td>
</tr>
<tr>
<td>SHOFT</td>
<td>The length of time that the job was in the shop (FINIS - ADATE)</td>
</tr>
<tr>
<td>SJOBS</td>
<td>Successor job in the JOBS set</td>
</tr>
<tr>
<td>PJ OBS</td>
<td>Predecessor job in the JOBS set</td>
</tr>
<tr>
<td>FSTAG</td>
<td>First report in the STAG set of process and queue reports</td>
</tr>
<tr>
<td>LSTAG</td>
<td>Last report in the STAG set</td>
</tr>
<tr>
<td>DDATE</td>
<td>The time at which the job was scheduled to leave the shop</td>
</tr>
</tbody>
</table>
Table 4 -- continued

<table>
<thead>
<tr>
<th>WORK</th>
<th>Entity representing a process report</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>The time at which processing started on the job</td>
</tr>
<tr>
<td>STOP</td>
<td>The time at which processing was completed or interrupted</td>
</tr>
<tr>
<td>PTIME</td>
<td>The amount of time spent processing the job (STOP - START)</td>
</tr>
<tr>
<td>SSTAG</td>
<td>Successor report in the STAG set of process and queue reports</td>
</tr>
<tr>
<td>PRES</td>
<td>The primary resource that processed the job</td>
</tr>
<tr>
<td>SRES</td>
<td>The secondary resource that processed the job</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JOB</th>
<th>Entity representing a queue report</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>Zero for a queue report, which distinguishes it from a process report</td>
</tr>
<tr>
<td>STOP</td>
<td>The time at which the job went into the queue or was interrupted</td>
</tr>
<tr>
<td>PTIME</td>
<td>The time at which the job was removed from the queue</td>
</tr>
<tr>
<td>SSTAG</td>
<td>Successor report in the STAG set of process and queue reports</td>
</tr>
<tr>
<td>PRES</td>
<td>The primary resource at which the job queued</td>
</tr>
<tr>
<td>SRES</td>
<td>One means that the job went into queue, two that it was interrupted</td>
</tr>
<tr>
<td>SHOPT</td>
<td>The length of time that the job was in queue (PTIME - STOP)</td>
</tr>
<tr>
<td>DDATE</td>
<td>The estimated process time of the job</td>
</tr>
</tbody>
</table>
Appendix E

THE RHIST PROGRAM
+ T JOB 4
+ T NUMBR 14/4 X
+ T PINQ 31/2 X
+ T SING 32/2 X
+ T PWRKG 41/2 X
+ T SWRKG 42/2 X
+ T SRECR 11/2 X
+ T TYPE 13/4 X
+ T WHEN 2 X
+ T NOINQ 3 X
+ T SICLE 34/4 X
+ T PWORK 41/4 X
+ T PIDLE 42/4 X
+ T SAVLR 42/2 X

1 11
2 R
3 T 1 Z 5 1
8 R
9 11 I 5 2

5 PRESR
5 SRESR
9999. QUIT TIME
BLANK
*IBFTC MAIN

MAIN ROUTINE
DIMENSION X(1002), J(1)
X EQUIVALENCE (X, J)
RECORD TAPE 9
READ TAPE 5, DONTP(1), FOR EACH PREROL
FORMAT 72(II)
READ FROM TAPE 5, DONT(11), FOR EACH SRESR
FORMAT 72(II)
LET IQ = 0
LET LOOSE = 0
LET NUM = 0
X 100 READ (9) X
LET M = 1
101 IF (J(M)) GE (999), GO TO 200
IF (X(N+5)) GE (QUIT), GO TO 200
IF (J(M)) GE (11), GO TO 150
GO TO (1,2,3,4,5,6,7,150,9,10), J(M)
C JOB ARRIVAL
1 CREATE JOB
LET NUM = NUM + 1
IF (NUM) LE (512), LET NUM = 1
LET NMBR(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 = NMBR(LOOSE)
LET W = WRITE(6,30)(1,X(N+1),X(N+2),X(N+3),J(N+4),X(N+5)
X 30 FORMAT (3X,3JOB,16,4H ARRIVED,F12.5H DUE,F12.5H COST,F12.5
X 35 FILE TYPE,F14,9H FINISHED,F12.5)
DESTROY JOB CALLED LOOSE
LET LOOSE = 0
GO TO 150
C JOB INTO QUEUE
3 LET I = J(M+1)
LET W = WRITE(6,30)(I,X(N+1),X(N+2),X(N+3),J(N+4),X(N+5)
LET W = WRITE(6,30)(I,X(N+1),X(N+2),X(N+3),J(N+4),X(N+5)
LET J(M+1) = X(M+5)
LET NMBR(ITEM) = NMBR(JOB)
LET TYPE(ITEM) = L
LET WHEN(ITEM) = X(N+5)
LET Q(N) = NMBR(ITEM)
LET ITEM IN RECR(1)
GO TO 150
C JOB INTERRUPTED
4 LET I = J(M+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(M+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 NQ(I) = NQ(I) - 1
LET L = 3
GO TO 45
31 LET I = J(M+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(M+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 PRIORITY(COMBO) = 1
48 LET AVAIL(1) = AVAIL(1) - M
LET WNGK(1) = WNGK(1) + M
LET PWORK(COMBO) = WNGK(1)
LET PILE(COMBO) = AVAIL(1)
FILE COMBO IN REC(1)
51 LET WHEN(COMBO) = X(N+5)
LET TYPE(COMBO) = L
IF (K) EQ (0), GO TO 49
LET SECDY(COMBO) = K
LET SECAY(K) = SECAY(K) - M
LET SCWRK(K) = SCWRK(K) + M
LET SWORK(COMBO) = SCWRK(K)
LET SILE(COMBO) = SECAY(K)
FILE COMBO IN AVLR(K)
49 IF (J(N)) LS (9), GO TO 150
CALL CHK(T1(Q)
IF (Q) NE (0), GO TO 200
GO TO (17,72,150), 1P
C END OF PROCESS
7 LET I = J(N+1)
FIND FIRST, FOR EACH JOB OF WNGK(1), 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 WNGK(1)
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 (DONT(1)) EQ (0), GO TO 150
61 CREATE JOB CALLED COMBO
GO TO 48
71 LET L = 7
LET AVAIL(1) = J(N+2)
LET WNGK(1) = 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 SECAY(K) = J(N+2)
LET SCWRK(K) = 0
LET IP = 3
GO TO 62
C RECORD NOW PROCESSED
150 CALL CHK(T1(Q)
IF (Q) NE (0), GO TO 200
LET N = N + 6
IF (N = 1002) 101,100,100
200 DO TO 205, FOR EACH PREM I
IF (DONTP(1)) EQ (0), GO TO 225
CALL PRT(1)
225 IF REC(1) IS EMPTY, GO TO 205
REMOVE FIRST JOB FROM REC(1)
IF (DONTP(1)) 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,1)
GO TO 201
212 CALL J2(JOB,1)
GO TO 201
213 CALL J3(JOB,1)
GO TO 201
214 CALL P1(JOB,1)
CALL P2(JOB,1)
206 IF (SECDY(JOB)) NE (0), GO TO 225
GO TO 201
215 CALL P2(JOB,1)  
   CALL P5(JOB,1)  
   IF (SECQ(JOB)) NE (0), GO TO 225  
   GO TO 201  
216 CALL P3(JOB,1)  
   GO TO 201  
218 CALL JR5(JOB,1)  
   CALL P5(JOB,1)  
   IF (SECQ(JOB)) NE (0), GO TO 225  
   GO TO 201  
217 CALL P4(JOB,1)  
201 DESTROY JOB  
   GO TO 225  
205 LOOP  
   DO TO 230, FOR EACH SRESR I  
   IF (DONT5(I)) EQ (0), GO TO 260  
   CALL SRP7(I)  
260 IF AVL(I) IS EMPTY, GO TO 230  
   REMOVE FIRST JOB FROM AVL(I)  
   IF (DONT5(I)) EQ (0), GO TO 231  
   GO TO (232,232,232,244,245,232,232,248,249,238), TYPE(JOB)  
232 CALL ERROR(I)  
238 CALL JR5(JOB,1)  
   CALL P5(JOB,1)  
   GO TO 231  
244 CALL S1(JOB,1)  
   CALL P5(JOB,1)  
   GO TO 231  
249 CALL S4(JOB,1)  
   GO TO 231  
248 CALL S3(JOB,1)  
   GO TO 231  
245 CALL S2(JOB,1)  
   CALL P5(JOB,1)  
231 DESTROY JOB  
   GO TO 260  
230 LOOP  
   IF (IQ) EQ (0), GO TO 270  
   LET IQ = 0  
   IF (J(M)) LS (9), GO TO 150  
   GO TO 101  
X 270 WRITE (6,250) X(M+5)  
X 250 FORMAT (/5X,20MEND HISTORY - TIME =,F10.5)  
   CALL EXIT  
END

*IBMAP CHKIT  
ENTRY CHKIT  
CHKIT TRA **  
CLA 32547  
TNZ CHKIT  
CLA 32546  
TNZ CHKIT  
SXN SAVE,4  
LAC CHKIT,4  
CLA **  
STO 2.4  
SAVE AXT **,4  
TRAP CHKIT  
END

*IBFTC ERROR  
SUBROUTINE ERROR(M)  
X WRITE (6,10) M  
X 10 FORMAT (/5X,6MERROR,1I0)  
CALL KRDMPP  
END
*IBFTC J1
  REPORT J1(JOB,1)
  X
  X  JOB NUMBER  INTO QUEUE AT ****** (DAY  HOUR  *), PRE
  X  NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN
  X  END
  X
  RCE  
  22 X
  SENTRY NUMBER IN QUEUE  =  *
  (JOB) HPART(WHEN(JOB)) NOINQ(JOB)
  END

*IBFTC J2
  REPORT J2(JOB,1)
  X
  X  JOB NUMBER  INTERRUPTED AT ****** (DAY  HOUR  *), PR
  X  NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB)) MPA
  END
  X
  RCE  
  22 X
  ESENTRY NUMBER IN QUEUE  =  *
  (WHEN(JOB)) NOINQ(JOB)
  END

*IBFTC J3
  REPORT J3(JOB,1)
  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
  RESENTRY NUMBER IN QUEUE  =  *
  RT(WHEN(JOB)) NOINQ(JOB)
  END

*IBFTC JRS
  REPORT JRS(JOB,1)
  X
  X  JOB NUMBER  RESTARTED AFTER SHIFT CHANGE WITH PRIMARY *, SE
  X  NUMBR(JOB) PRIMY(JOB)
  END
  X
  RCE  
  22 X
  CONDARY  *
  SECODY(JOB)
  END

*IBMAP KRDM
  ENTRY KRDM
  KRDM  PZE  0
  PZE  0
  END

*IBFTC P1
  REPORT P1(COMBO,1)
  X
  X  START WORK ON JOB NUMBER  AT ****** (DAY  HOUR  *), W
  X  NUMBR(COMBO) WHEN(COMBO) DPART(WHEN(COMBO)) HPART(W
  X  END
  X
  RCE  
  22 X
  1TH SECONDARY RESOURCE  *
  HEN(COMBO)) MPART(WHEN(COMBO)) SECDY(COMBO)
  END
*IBFTC P2
  REPORT P2(JOB,1)
  X
  X END PROCESS ON JOB NUMBER AT **** (DAY * HOUR * *) W
  X NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB))
  END

RCE  *  22 X

1TH SECONDARY RESOURCE *
MPART(WHEN(JOB)) SECDY(JOB)
END

*IBFTC P3
  REPORT P3(JOB,1)
  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

RCE  *  22 X

1 HIS TYPE WORKING, * IDLE
B) PIDLE(JOB)
END

*IBFTC P4
  REPORT P4(JOB,1)
  X
  X AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS PRIMARY RESOURCE
  X
  END

RCE  *  22 X

1S  *  1

PIDLE(JOB)
END

*IBFTC PRPT
  REPORT PRPT(1)
  X
  X
  END

RCE  *  2

X

END

*IBFTC PS
  REPORT PS(JOB,1)
  X
  X CURRENTLY OF THESE RESOURCE TYPES, THERE ARE * PRIMARIES WORK
  X PWORK(JOB)
  END

RCE  *  22 X

1ING, * IDLE, * SECONDARIES WORKING, * IDLE
PIDLE(JOB) SWORK(JOB) SIDLE(JOB)
END
*1BFTC S1
REPORT S1(JOB=1) SECONDARY RESO

X
X START WORK ON JOB NUMBER AT . . . . . (DAY HOUR . . ), W
    NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB))
    END

URCE 22 X

17TH PRIMARY RESOURCE
MPART(WHEN(JOB)) PRIMY(JOB)

END

*1BFTC S2
REPORT S2(JOB,1) SECONDARY RESO

X END PROCESS ON JOB NUMBER AT . . . . . (DAY HOUR . . ),
    NUMBR(JOB) WHEN(JOB) DPART(WHEN(JOB)) HPART(WHEN(JOB)) M
    END

URCE 22 X

WITH PRIMARY RESOURCE
PART(WHEN(JOB)) PRIMY(JOB)

END

*1BFTC S3
REPORT S3(JOB,1) SECONDARY RESO

X AT SHIFT CHANGE (DAY HOUR . . ) THERE ARE . . SECONDARIES OF
    DPART(WHEN(JOB)) HPART(WHEN(JOB)) MPART(WHEN(JOB)) SWORK
    END

URCE 22 X

THIS TYPE WORKING, IDLE
(JOB) SIDE(JOB)

END

*1BFTC S4
REPORT S4(JOB,1) SECONDARY RESO

X AFTER SHIFT CHANGE, THE NEW AVAILABILITY OF THIS SECONDARY RESOURC

END

URCE 22 X

E IS SIDE(JOB)

END

*1BFTC SRPT
REPORT SRPT(1) SECONDARY RESO

X END

URCE 22 X

END
Table 5

RHIST SYSTEM VARIABLES

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

*See listing on p. 64.*
Table 5 -- continued

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>DONT</td>
<td>Flag indicating whether or not reports are desired for the secondary resource</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>PWORK</th>
<th>The number of busy units at the primary resource after the occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIDLE</td>
<td>The number of available units at the primary resource after the occurrence</td>
</tr>
<tr>
<td>SAVLR</td>
<td>Successor in the AVLR set of records relating to a secondary resource</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>SRECR</th>
<th>Successor in the RECR set of records relating to a primary resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The type of change that the record represents</td>
</tr>
<tr>
<td>NUMBR</td>
<td>The number of the job causing the change</td>
</tr>
<tr>
<td>WHEN</td>
<td>The time of the change</td>
</tr>
<tr>
<td>NOINQ</td>
<td>The number of jobs in the queue after the change</td>
</tr>
</tbody>
</table>