TITLE: B1800/B1700 SYSTEM SOFTWARE RELEASE MARK VII.0

FILE ID: DOCUMENT/RELEASE TAPE ID: SYSTEM

TABLE OF CONTENTS

B1800/B1700 System Software Release Mark VII.0

1. General Information

1.1. Introduction.

The Mark VII.0 software release includes a number of major enhancements, described in detail in the following sections of this release document. Among the new enhancements and features are the following (the numbers in parentheses refer to the paragraphs in this document that describe the features in detail):

1. New memory management algorithms have been implemented in the MCP, using three optional levels of sophistication and complexity (2.2.2).

2. Capabilities for dynamically monitoring and displaying system performance have been implemented in the MCP and GISMO/DEBUG (2.2.3).

3. Mix-numbers have been replaced by job-numbers, allowing use of a single, consistent number for all job identification (2.2.4).

4. The XM function has been implemented in the MCP and System Initializer (SYSTEM/INIT), allowing specified areas of memory to be removed from use during the CLEAR/START operation (2.2.10).

5. The meaning of the STATE light has been changed to allow providing a more accurate and meaningful indication of how "busy" the processor is (2.2.13).

6. The B 1800/B 1700 Time Analysis and Billing System (TABS) has been implemented, including new event-oriented records placed in the SYSTEM/LOG by the MCP (2.2.39 and 20.1).

7. The capability to create Installation Allocated Disk files has been implemented in the MCP and a new utility program, DISK/ALLOCATOR (2.2.40 and 18.2.5).

8. The AUTOLOAD facility has been implemented in the MCP and DISKETTE/COPY, allowing pseudo-reader files on diskette to be copied to system disk without operator action (2.2.44 and 18.2.6).

9. The two methods for I/O SEQUENTIAL disk file access, originally implemented in patch #40 to Mark VI.1, have been included in the Mark VII.0 MCP and MICRO.MCP (2.2.64).

26. The Supervisory Message Control System (SMCS) program product, designed to be the "supervisor" for a highly-interactive data communications system, is now available (13.1).

27. The HOST RJE system now includes a CALLBACK facility, allowing a remote terminal user to log off and then be automatically called back when his jobs have been completed (15.2.1).

28. The RJE3780 program product, which allows the B 1800/B 1700 to perform as an RJE station to a host system using the IBM 3780 discipline, is now available (16.1).

29. "Hot Readers", as well as new local commands, have been implemented in HASP (17.2.3 and 17.2.4).

30. DMPALL is now able to process ISAM input files sequentially, using the associated TAG file as the source of the relative record numbers (18.2.8).

31. A disk reconfiguration option has been implemented in PACK/INIT and SYSTEM/DISK.INIT, allowing the pack type to be changed without reinitialization (18.2.12).

32. The capability for declaration of in-line COLLATE tables has been implemented in SORT (18.2.14).

33. SORT now has the ability to produce COBOL- and RPG-compatible TAG files as output (18.2.14).

34. SORT now provides a capability for maintaining the original order from the input file of all records that have equal key fields (18.2.14).

35. A new free-standing system utility program, STANDALONE/DISK.DUMP, is now available, which allows copying of files between two disks in a file-by-file mode rather than sector-by-sector mode (18.2.15).

36. SYSTEM/COMPARE, a generalized system utility program for comparing files, is now available (18.2.17).

37. SYSTEM/ELOGOUT has been redesigned, providing more readable and useful reports (18.2.18).

38. SYSTEM/ICMD.INIT, a normal-state program used for initializing the diskette, is now available (18.2.19).

39. SYSTEM/MARK.SEGS, a new system utility program used for marking "important" code segments (for use with Priority Memory Management and Extended Segment Decay), is now available (18.2.21).

40. The B500 Magnetic Tape Memory Write (MWR) instruction is now supported in the B500 Interpreter (19.2.1).

41. The B500 Interpreter now supports the B500 "double-density" disk (19.2.2).

42. The ability to separate object program files and data files has been provided in the 1400 Interpreter (19.3.1).

Specific details on these and other enhancements is contained in the various sections of this document that follow. Where applicable, temporary documentation on syntax and use of new functions is also provided.

1.2. Software Included with the Mark VII.0 Release.

The following list details the software that is available as part of this release. Programs preceded by an asterisk are being released for the first time on Mark VII.0.

Program Identification	Version	Compile Date	Remarks
MCPII	VII.0.1	7/11/78	
MCPII/MICRO.MCP	VII.0	5/25/78	
MICRO.MCP/DEBUG	VII.0	5/25/78	
GISMO	VII.0.1	7/ 1/78	
GISMO/DEBUG	VII.0.1	7/ 1/78	
BASIC	VI.1	8/22/77	
BASIC INTRINSICS	VI.1	5/ 9/77	
BASIC/INTERP3S	VII.0	4/18/78	Note 1
BASIC/INTERP3M	VII.0	4/18/78	Note 2
COBOL	VII.0	6/ 8/78	
COBOL/XREF	VII.0	7/11/78	
COBOL/INTERP1S	VII.0	6/20/77	Note 1
COBOL/INTERP1M	VII.0	6/20/77	Note 2
DASDL	VII.0	6/10/78	
RECOVER/DATA.BASE	VII.0	6/ 1/78	
DMS/REORG.READ	VII.0	4/ 4/78	
DMS/REORG.WRIT	VII.0	5/ 3/78	
RPG	VII.0	3/14/78	
RPG/BTF	VII.0	8/ 7/77	
RPG/REORD	VII.0	8/ 7/77	
RPG/REORG	VII.0	8/ 7/77	
RPG/XREF	VI.0	8/29/75	
RPG/INTERP1S	VII.0	1/26/78	Note 1
RPG/INTERP1M	VII.0	1/26/78	Note 2
FORTRAN	VII.0	6/15/78	
FORTRAN INTRINSICS	VII.0	6/15/78	
FORTRAN/INTMAKER	V.1	1/12/76	
FORTRAN/INTERP1S	VII.0	6/ 5/78	Note 1
FORTRAN/INTERP1M	VII.0	6/ 5/78	Note 2
UPL	VII.0	2/20/78	
SDL/ERRORS	VII.0	10/19/77	
SDL/XMAP	VII.0	3/23/77	
SDL/XREF	VII.0	10/27/77	
SDL INTRINSICS	VII.0	8/19/77	
SDL/INTERP1S	VII.0	10/10/77	Note 1
SDL/INTERP1M	VII.0	10/10/77	Note 2
NDL	VII.0	4/27/78	
NDL/MACRO	VII.0	6/12/78	
NDL/ADDRESS	VII.0	6/12/78	
NDL/LIBRARY	VII.0	6/26/78	
NDL/DUMP	VII.0	4/28/78	
DC/AUDIT	VII.0	1/17/78	
SOURCE/MCSI	V.1	12/ 1/75	
SOURCE/MCSII	VI.1	1/24/77	

Program Identification	Version	Compile Date	Remarks
*SMCS	VII.0	4/28/78	
CANDE	VII.0	5/19/78	
CANDE/ANALYZER	VII.0	1/24/78	
CANDE/TEACH.FILE	VII.0	11/ 9/77	
TEXT/EDITOR	VII.0	6/30/78	
RJE	VII.0	1/17/78	
RJE/DCH	VII.0	1/17/78	
RJE/MCS	VII.0	1/17/78	
RJE/AUTOBACKUP	VII.0	10/10/77	
RJE/CONTROLLER	VII.0	1/30/78	
RJE/MESSAGES	VII.0	1/18/78	
*RJE3780	VII.0	6/ 6/78	
HASP	VII.0	4/14/78	
HASP/SPOOL	VII.0	5/24/78	
HASP/MODIFIER	VII.0	2/ 6/78	
B500/IEP	VII.0	9/30/77	
B500/INTERP2U	VII.0	7/12/78	
1400/IEP	VII.0	12/ 1/77	
1400/INTERP2U	VII.0	4/26/78	
1400/CREATE1311	VI.0	9/ 6/76	
SORT	VII.0	3/22/78	
SORT/COLLATE	VI.1	8/19/77	
SORT/MERGE	VII.0	9/ 9/77	
SORT/QSORT	VII.0	4/ 7/78	
SORT/TAPESORT	VII.0	11/ 7/77	
SORT/VSORT	VII.0	3/23/78	
MCPII/ANALYZER	VII.0	6/30/78	
DUMP/ANALYZER	VII.0	6/22/78	
CODE/ANALYZER	VII.0	12/ 7/77	
DISK/COPY	VII.0	3/29/78	
DISKETTE/COPY	VII.0	4/26/78	
*SYSTEM/ICMD.INIT	VII.0	12/21/77	
SYSTEM/BACKUP	VII.0	6/12/78	
*SYSTEM/COMPARE	VII.0	6/28/78	
SYSTEM/LOGOUT	VII.0	1/24/78	
SYSTEM/ELOGOUT	VII.0	6/ 3/78	
SYSTEM/SPOLOGOUT	VI.1	8/22/77	
SYSTEM/LDCONTRL	VII.0	11/ 3/77	
SYSTEM/LOAD.CAS	VII.0	9/30/77	
SYSTEM/LOAD.DUMP	VII.0	5/23/78	
SYSTEM/DISK.INIT	VII.0	6/ 5/78	
SYSTEM/DISK.DUMP	VII.0	7/28/77	
SYSTEM/BUILDTRAIN	VI.1	8/19/77	
SYSTEM/TRAINTABLE	VII.0	10/21/77	
INPUT/PCS.TABLES	VII.0	10/21/77	
SYSTEM/MAKEUSER	VII.0	1/27/78	

1.3. B 1800/B 1700 Reference Manuals.

The following manuals have been released:

#1057155	B 1700 Systems Reference Manual (4-72)
#1057155-001	PCN #1 (9-8-72)
#1057155-002	PCN #2 (9-28-73)
#1057189	B 1700 RPG Reference Manual (8-77)
#1057197	B 1800/B 1700 COBOL Reference Manual (9-77)
#1067170	B 1700 UPL Reference Manual (12-73)
#1067535	B 1700 BASIC Reference Manual (6-73)
#1068731	B 1800/B 1700 System Software Operational Guide (8-78)
#1072568	B 1700 MIL Reference Manual (5-77)
#1073715	B 1700 NDL Reference Manual (6-77)
#1081346	B 1700 SDL Reference Manual (12-74)
#1081882	B 1800/B 1700 FORTRAN Reference Manual (4-78)
#1088010	B 1700 MCP Reference Manual (8-75)
#1089794	B 1700 DMSII Reference Manual (1-76)
#1089992	B 1700 Data Communications Information Manual (12-75)
#1090578	B 1700 HASP Reference Manual (8-76)
#1090586	B 1700 CANDE User's Manual (8-76)
#1090602	B 1800/B 1700 RJE Reference Manual (7-78)
#1090610	B 1800/B 1700 TEXT/EDITOR Reference Manual (7-78)

The revised edition of the B 1800/B1700 System Software Operational Guide (dated 8-78) is now available for distribution. This revision of the manual contains detailed descriptions of all new features and utility programs available with the Mark VII.0 release, and should be reviewed thoroughly prior to attempting installation of this software level.

Reference will be made throughout this release document to new features and programs, where specific documentation is contained in the revised B 1800/B 1700 System Software Operational Guide. Detailed information on such features and programs is not included in this release document.

1.4. B 1800/B 1700 Software Flashes.

The following list reflects the current status of all B 1800/B 1700 Software Flashes issued since January 1, 1977:

Flash	Release Fixed	Remarks
175	VI.1	
176	VI.1	
177	----	Information only
178	VI.1	
179	VI.1	
180	VI.1	
181	VI.1	
182	VI.1	
183	VII.0	Also RPG VI.1.1-VI.1.5
184	----	Information only
185	VII.0	Also SYSTEM/LOAD.CAS VI.1.1
186	VII.0	
187	VII.0	
188	VII.0	Also FOR.INTRIN VI.1.1
189	VII.0	
190	VII.0	
191	VII.0	
192	VII.0	
193	VII.0	Also RPG VI.1.6 & MCPII VI.1.40
194	VII.0	Also COBOL VI.1.4
195	VII.0	
196	VI.1	
197	VII.0	Also RPG VI.1.8
198	VII.0	
199	VII.0	
200	VII.0	Also DASDL VI.1.4
201	VII.0	
202	VII.0	
203	VII.0	Also MCPII VI.1.69

2. MCPII (SMCP)

2.1. Introduction.

In addition to the implementation of new MCP Memory Management algorithms, a number of enhancements have been included in the Mark VII.0 MCP. Among these are the replacement of program mix numbers by job numbers, the ability to explicitly remove portions of memory from use (XM), and the ability to explicitly allocate disk files (Installation Allocated Disk).

Each of these changes/enhancements is documented either in the paragraphs that follow or in the revised B 1800/B 1700 System Software Operational Guide.

2.2. Enhancements.

2.2.1.

The following problems reported in the Mark VI.1 release letter have been fixed (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. Information beyond column 72 is now ignored in all control cards (2.3.7).

b. SYSTEM/LOAD.DUMP will no longer start executing due to a feed check, read check, or empty hopper while reading input cards for an ADD, DUMP, LOAD, or UNLOAD operation. An ending semicolon is required before SYSTEM/LOAD.DUMP will begin (2.3.8).

c. Programs waiting on a COMPLEX.WAIT can now be marked as candidates for rollout (2.3.10). Unless "frozen", any program waiting on a COMPLEX.WAIT will be rolled out if none of the events upon which it is waiting has occurred after two (2) MCP N.SECOND intervals.

d. Entries in the ELOG for disk errors on multiple system drives now contain the correct unit-mnemonic (2.3.11).

e. USER.BACKUP.NAME files on the PBD.DESIGNATION disk can now be printed/punched without the necessity of including the pack-id in the PB message (2.3.12).

f. Closing an output printer or punch file with PURGE no longer fails to enter the file in the directory if it is directed to backup disk (2.3.13).

g. If a program which has a charge number calls a SORT intrinsic, the charge number is now passed along to the intrinsic regardless of the setting of the CHRG option (2.3.17).

h. It is now possible to print/punch more than one secured backup file with a single PB command by the use of the following message format:

    USER=<usercode>/<password> PB <specifier>

    where <specifier> can be =/=, PRT/=, PRN/=, or PCH/= (2.3.18).

i. Printer/punch files are no longer automatically directed to backup if the REPETITIONS value is equal to one (2.3.19).

j. The <integer> used with the AB input message cannot be greater than four (2.3.20).

k. System failures no longer occur when a tape file is closed (NO.REWIND) when not at EOF and an attempt is made to reopen the file (NO.REWIND) (2.3.21).

l. A CL message referencing a MICR Reader-Sorter no longer hangs the MCP (2.3.23).

m. A PG or RP input message that references a tape drive with no tape mounted on it no longer results in a spurious NO WRITE RING message (2.3.25).

n. If a tape with a single-track error is mounted, the MCP no longer loops indefinitely trying to ready the unit (2.3.26).

o. If AUTOPRINT is active, any printer backup disk file that is closed will now be entered in the AUTOPRINT QUEUE, regardless of whether or not the backup file is directed to the PBD.DESIGNATION disk or has a family-name of BACKUP.PRT (2.3.27). A new attribute of the FILE statement, AUTOPRINT, may be used to prevent the file from being entered in the AUTOPRINT QUEUE (refer to paragraph 2.2.9 of this document and to the revised Software Operational Guide for details).

p. If SYSTEM/BACKUP is executed by AUTOPRINT and no printer has been reserved with the AB input message, the message requesting a printer now indicates that an AB message is required (2.3.30).

q. MICR Reader-Sorter programs can now be DS-ed without causing MCP problems (2.3.31).

2.2.2.
New MCP Memory Management algorithms have been implemented, on three (optional) levels of sophistication and complexity:

a. First-In, First-Out
b. First-In, First-Out with Thrashing Detection
c. Memory Priority with Thrashing Detection

Complete documentation on the implementation, operational details, advantages, and disadvantages of each of these levels is contained in appendix A of the revised Software Operational Guide. A thorough review of this documentation is recommended, because a number of new operational features are connected with the new Memory Management algorithms, including the concept of separate Memory and Processor Priorities, Extended Segment Decay, and specification of thrashing detection parameters.

2.2.3.
Capabilities have been implemented in the MCP (in conjunction with GISMO/DEBUG) for dynamically monitoring and displaying system performance on the system console. Complete documentation on these powerful performance-monitoring features is included in appendix B of the revised Software Operational Guide.

2.2.4.
Any program that is executed is now identified by a single number, the job-number, from the time that the program is scheduled until it reaches end-of-job. Mix-numbers, as such, no longer exist. Job-numbers are incremented from 1 through 9999, at which time they are reset to 1 by the MCP.

All references to a program (whether in the mix or in the schedule) are now made using the job-number. Prior to Mark VII.0, some messages used the mix-number as a parameter and others used the job-number.

A second number, the Job Accounting Number, is maintained as a completely unique job identifier for the purposes of the system log. This number is not reset at 9999 (the maximum value is 16777215), and is only accessible to programs which process the log file.

2.2.5.
Because each program now has a unique job-number, programs such as SORT intrinsics (for example, SORT/VSORT) are no longer "hidden". That is, such jobs (referred to as "tasks") now display their beginning and ending messages on the SPO. These messages identify the "beginning-of-task" (BOT) and "end-of-task" (EOT), and specify the program which "called" the task (as the job-number contained in parentheses).

2.2.6.
For compatibility with B 7000/B 6000 series systems, the PROTECTION and PROTECTION.IO attributes have been changed to SECURITYTYPE and SECURITYUSE, respectively, for use with the FILE statement and the MH input message. The abbreviations are as follows:

SECURITYTYPE	SEC
SECURITYUSE	SUS

The PROTECTION and PROTECTION.IO keywords (as well as their abbreviations) are still allowed for Mark VII.0; however, they will be deleted in a future release.

2.2.7.
Session numbers (specified in Job Spawning Control Instructions with the SZ control attribute) have been divided into three distinct ranges, each with different characteristics and restrictions. Complete documentation on session numbers is contained in the revised Software Operational Guide.

2.2.8.
The maximum CHARGE number that can be specified has been increased to seven digits (9999999).

2.2.9.
The AUTOPRINT attribute (FILE statement) has been implemented. This attribute, which is SET by default, specifies that a printer backup file is to be entered into the AUTOPRINT QUEUE when the file is closed. Specifying "NO AUTOPRINT" prevents a printer backup file from being printed by the MCP AUTOPRINT mechanism. The AUTOPRINT attribute may be abbreviated as ATP.

In Mark VI.1, any printer backup file having the default family-name ("BACKUP.PRT") and which resided on the PBD.DESIGNATION disk was entered into the AUTOPRINT QUEUE when it was released. In Mark VII.0, all backup files are entered into the AUTOPRINT QUEUE when they are released, unless "NO AUTOPRINT" is specified. Backup files already existing on disk when AUTOPRINT is invoked (for example, when the AB message is entered), are only placed in the AUTOPRINT QUEUE if they have the default family-name and reside on the PBD.DESIGNATION disk (files with "NO AUTOPRINT" specified are still ignored by SYSTEM/BACKUP, even if they are found in the AUTOPRINT QUEUE).

2.2.10.
The XM input message has been implemented, allowing explicit removal of specified areas of memory from use by the system. The specifications made in the XM message are entered by the MCP in a table (the XM TABLE) maintained on disk; actual deletion of the designated areas is performed by the System Initializer during the next CLEAR/START operation. Complete documentation on the syntax and usage of the XM input message is contained in the revised Software Operational Guide.

2.2.11.
The PROTOCOL attribute (FILE statement) has been implemented, allowing a value to be passed from a program to an MCS in the remote file OPEN and OPEN-REPLY messages. Refer to paragraph 10.4.11 of this document and to the revised Software Operational Guide for additional information.

2.2.12.
The "file-identifier RELEASED" message is now only displayed when a backup file is closed if the BREL (Backup File Released) option is set (refer to the revised Software Operational Guide for details).

2.2.13.
The information provided by the STATE light, when running without the performance-monitoring feature, has been changed in order to provide a better indication of how "busy" the processor is.

The implementation prior to Mark VII.0 attempted to distinguish between "normal state" and "control state". The STATE light was:

ON When either the SMCP or the MICRO.SCHEDULER is running.

OFF When the MMCP or any user program is running.

The problem with that implementation was that it provided little information about how busy the system really is, since both meanings of the STATE light still indicate that the processor is being used. In addition, the light could be somewhat misleading in some cases. Logical I/O operations performed by the MMCP were considered as part of "normal state" (user program) time; however, there are some exceptional cases of logical I/O that must be handled by the SMCP. These were inconsistently displayed as part of the "control state" time.

In order to provide a more useful indication of processor use, the Mark VII.0 release uses the STATE light to distinguish between system idle time and non-idle time. The STATE light is:

ON When every process in the system is waiting (idle).

OFF When the processor is executing some process.

The advantages of this implementation are as follows:

a. The MICRO.SCHEDULER can easily and unambiguously provide an indication of how busy the system is.

b. With control of the STATE light limited to the MICRO.SCHEDULER (unlike previous releases, where each interpreter set or reset the light as it gained or gave up control), other performance-monitoring functions can be provided through the STATE light.

The main disadvantage is that it is different. To someone who is used to previous releases, the user job appears to be very busy (unless the system is idle), or it may even appear to be hung in a loop.

2.2.14.
The MCP no longer locks an input multi-file tape when end-of-reel is encountered during a multi-file search. Note: this change results in the same tape being searched again if another tape with the same family-name is not mounted and ready prior to the first tape rewinding to beginning-of-tape.

2.2.15.
The following changes to the MCP printer/punch backup mechanism have been made in this release:

a. It is now possible to print/punch all backup files on disk having a specified family-name, using a "PB <family>/=" input message. No options other than SAVE, LABEL, or LABELS may be specified when using this form of the PB message. For example,

```
PB TWO.PLY/=
PB DPC BILLS/= LABELS
```

b. Printer/punch backup files created with the USER.BACKUP.NAME attribute specified are not removed from disk by SYSTEM/BACKUP after they have been printed/punched, unless switch 2 in SYSTEM/BACKUP is set to a non-zero value. This feature does not apply to any backup file whose name is simply changed to a non-default backup-file-identifier; USER.BACKUP.NAME must have been specified when the backup file was created.

c. All output printer/punch backup files are now directed to the PBD.DESIGNATION disk (even if a specific pack-id has been designated for the file), unless the USER.BACKUP.NAME attribute is set. This does not apply, however, to secured backup files, which are directed to the default pack specified in the (SYSTEM)/USERCODE file. If no default pack is assigned, secured backup files are directed to the PBD.DESIGNATION disk.

Additional documentation on all features of SYSTEM/BACKUP and the PB input message is included in the revised Software Operational Guide.

2.2.16.
The Network Controller identified in the NAME TABLE will no longer be executed more than once when multiple Data Comm jobs are executed at the same time.

2.2.17.
The MCP now prevents a "DYNAMIC ... FILE ..." control instruction from referencing any file that is not completely closed. A file is not closed if a File Information Block (FIB) is still assigned; the FIB is only returned when the file is closed with RELEASE, LOCK, PURGE, or REMOVE (the MCP does, however, allow a "DYNAMIC" control instruction to reference a tape file that has been closed with NO.REWIND). A "DYNAMIC" control instruction referencing any of the following attributes is allowed by the MCP, however, whether or not the file is completely closed:

EOP
AUTOPRINT
INVALID.CHARACTERS
LOCK

In a similar manner, the MCP prevents a WRITE.FPB communicate by a program from referencing a file that is not completely closed. For Mark VII.0, a warning message is displayed and the WRITE.FPB is allowed to proceed; in Mark VIII.0, the program will be DS-ed immediately. For COBOL programs which use the "VALUE OF ID IS data-name" construct, WRITE.FPB communicates are generated by the compiler (transparent to the programmer). Since it is impossible to have control in the source program over when these communicates are issued, WRITE.FPB communicates by a COBOL program which reference a file that is not completely closed are simply ignored by the MCP (no warning message is displayed).

2.2.18.
If the default pack specified for a usercode is not present when an output disk file is opened by a program running under that usercode, the MCP sets the "override" bit in the (SYSTEM)/USERCODE file, directs the file to the system disk, and displays a warning message on the SPO. Once the override bit is set, all output files created by any program running under that usercode are directed to system disk (without any warning messages), regardless of whether or not the required pack is on line. The override bit can be explicitly reset using the RV input message. Refer to paragraph 2.2.52 of this document and also to the revised Software Operational Guide for details.

2.2.19.
The RM response to a DUPLICATE FILE situation with SYSTEM/LOAD.DUMP no longer causes the output file to have incorrect attributes (Flash #190).

2.2.20.
A disk SQUASH operation referencing Electronics Unit 1 of a head-per-track system disk no longer halts the system with an irrecoverable "NOT READY" disk exception condition (Flash #199).

2.2.21.
The MCP now generates the correct I/O opcode for Extended Result Descriptor (ERD) reads on B 9484 Disk Packs connected to Disk Pack Control-1 (Flash #203).

2.2.22.
The MCP will no longer reinstate a program after displaying the following terminal error message:

TAPE NOT POSITIONED AT EOF1 LABEL

2.2.23.
It is no longer possible for a program running under a non-privileged usercode/password (or a program not running under File Security) to open a secured output disk file having two names where the family-name is not the same as the usercode. In addition, a program which is not running under a privileged usercode cannot CLOSE an existing secured disk file with PURGE if the family-name is not the same as the usercode.

One of the areas most often affected by this security improvement is where it is desired to PB a secured, public backup file, using the following input message:

PB (<usercode>)/#<integer>

Since SYSTEM/BACKUP is executed without a usercode in this case, the backup file cannot be closed with PURGE, and remains on disk after printing. The proper way to do this would be either to prefix the PB message above with a privileged usercode/password, or to enter the message as follows:

USER=<usercode>/<password> PB <integer>

2.2.24.
A NOT READY encountered on a tape unit during a tape file OPEN no longer hangs the MCP.

2.2.25.
The MCP now handles blanks within file-identifiers on tape labels as follows:

a. Leading blanks are disallowed,
b. Trailing blanks are truncated, and
c. Embedded blanks are allowed.

2.2.26.
The MCP will now initiate a physical WRITE of the last partial block on a tape output file following a CLOSE request if the preceding I/O encountered end-of-reel, preventing a program hang.

2.2.27.
The MCP will no longer halt when attempting to sort a variable-length record disk file having an EOF.POINTER of zero.

2.2.28.
The MCP will no longer halt with an INVALID CASE when processing a WY input message referencing the NDL Network Controller.

2.2.29.
The MCP will no longer halt if an attempt is made to assign an input card file which has stacker-select specified to an 80-column reader.

2.2.30.
A CL message referencing the MICR Reader-Sorter or a DS of the program is now allowed after receiving the following error message:

POCKET LIGHT COMMUNICATE REQUESTED WHILE SORTER FLOWING

2.2.31.
A TERMINATE card no longer halts the MCP if the file-identifier is omitted. The TERMINATE card is now ignored by the MCP if the file-identifier is missing or does not match the file-identifier on the corresponding STREAM card.

2.2.32.
The MCP now allows an SD to be requested on a disk initialized as SYSTEM (S), without having to first reinitialize the disk as UNRESTRICTED (U).

2.2.33.
A MX, WY, or WS= message entered on the host SPO now displays all jobs running or in the schedule, without the necessity of first setting the RMSG option.

2.2.34.
All known problems causing MCP NAME/VALUE STACK OVERFLOW halts have been fixed.

2.2.35.
An ST message now forces the N.SECOND routine to be called immediately, causing immediate rollout of the stopped job. In addition, a new option of the ST message allows a program to be stopped, then automatically restarted when another program in the mix terminates. Refer to the revised Software Operational Guide for details.

2.2.36.
The SL input message no longer allows non-numeric parameters to be specified.

2.2.37.
A space is now required to delimit bit-type operands (those enclosed by "at" signs) in the CP input message.

2.2.38.
The NC option of the PM input message is no longer allowed. DUMP/ANALYZER automatically zip-executes NDL/DUMP after printing the analysis of an NDL Network Controller dump.

2.2.39.
The TABS option has been implemented. When this option is set, the MCP includes certain additional records in the LOG for use by TABS (Time Analysis and Billing System). Refer to the TABS document and to the revised Software Operational Guide for further information.

2.2.40.
The MCP now provides for creation of disk files with user-specified attributes and absolute disk addresses (Installation Allocated Disk). Refer to the DISK/ALLOCATOR section in the revised Software Operational Guide for details on Installation Allocated Disk.

2.2.41.
The IC (Interpreter Count) input message has been implemented, allowing specification of the maximum number of interpreters that can be run concurrently. Refer to the revised Software Operational Guide for further information.

2.2.42.
The MCP now attempts to do a "deep read" (for error correction) on cassette tape. This capability requires installation of a hardware change to the cassette control (LIN #L6958-002 for hybrid controls and LIN #L6958-003 for wirewrapped controls). Attempting to use cassette tape on Mark VII.0 without the proper hardware change installed will hang the MCP. This is applicable only to those B 1800/B 1700 systems that have magnetic tape cassette peripheral units (which provide the ability to write cassette files), not to systems which only have the console cassette drive.

2.2.43.
The REWIND CASSETTE message displayed by the System Initializer during a CLEAR/START operation has been deleted. A SYSTEM MEMORY DUMP COMPLETE message is displayed by the MCP, however, if a memory dump was requested during the CLEAR/START.

2.2.44.
Automatic loading of pseudo decks from diskette has been implemented in the MCP. A pseudo deck on diskette is identified by the presence of "PSR" in the first three bytes of a header label. Refer to the DISKETTE/COPY section in the revised Software Operational Guide for further details.

2.2.45.
Commas are again required to separate file-names in the list of files specified in a REMOVE control instruction. This requirement (primarily a safeguard) was inadvertently deleted in Mark VI.1.

2.2.46.
A tape labeled "SCRATCH" is now handled properly as an input file by the MCP.

2.2.47.
The MIX LIMIT (ML input message) now applies only to jobs having a PROCESSOR.PRIORITY less than 9. Programs with a PROCESSOR.PRIORITY of 9 or greater can enter the mix (given sufficient system resources) regardless of the setting of the MIX LIMIT. In addition, any job that is executed with a PROCESSOR.PRIORITY of 9 or greater is automatically given a SCHEDULE.PRIORITY the same as the PROCESSOR.PRIORITY (up to a maximum of 14), unless an explicit SCHEDULE.PRIORITY attribute is also specified. Note that the PRIORITY assumes the functions of PROCESSOR.PRIORITY if the Priority Memory Management algorithm is not being used by the MCP. Refer to the revised Software Operational Guide for complete information.

2.2.48.
The Julian date has been added to all output messages which display the current date.

2.2.49.
The RX input message now requires the serial number of the disk as a verification (refer to the revised Software Operational Guide for complete syntax).

2.2.50.
The response to the OL input message on a user pack/cartridge now includes the number of users currently using the disk.

2.2.51.
A new run-time attribute, NODIF, has been implemented. Specification of this attribute with a COMPILE, EXECUTE, or DYNAMIC control instruction (not allowed with MODIFY) prevents the premature termination of spawned jobs due to "DEATH IN FAMILY". Refer to the revised Software Operational Guide for details.

2.2.52.
The RV input message has been implemented to allow the "override" bit in the (SYSTEM)/USERCODE file to be reset for a particular usercode. Refer to the revised Software Operational Guide for further information.

2.2.53.
The equal sign (=) is now allowed between USER (or US) and the <usercode>/<password>.

2.2.54.
The MCP no longer halts when an ST message followed by a GO message is entered for a program awaiting a Data Comm Open Permit.

2.2.55.
The MCP no longer halts when an attempt is made to remove a backup file directed to a specific user pack which exceeded its maximum allowable size.

2.2.56.
The "CORE.COPY" of an in-use disk file header is now only printed by the KA routine if the DBUG option is set.

2.2.57.
The MCP no longer halts on an irrecoverable disk I/O error or an invalid key when attempting to enter a message in a queue.

2.2.58.
The MCP no longer halts with a READ OUT-OF-BOUNDS while processing a QC input message.

2.2.59.
A WY on a program attempting to perform a DMS open where the database is locked now contains the correct file-identifier in the response.

2.2.60.
Attempting to print part of the SPO queue (for example, "KB 100") when the SPO queue contains some uninitialized records no longer halts the MCP.

2.2.61.
Due to implementation of the MEMORY.PRIORITY attribute, the MP input message, previously used to list the Multi-Pack File Tables, is now used to interrogate or change the memory priority of a program in the mix. The MU input message is now used to list the Multi-Pack File Tables. Complete documentation on both of these input messages is contained in the revised Software Operational Guide.

2.2.62.
The MCP now properly handles the Working Available Table if an SD input message is used to assign an additional system drive, and the Working Available Table on the existing system disk(s) has extension segments.

2.2.63.
It is now possible to interrogate (with the LT input message) which print translator was selected while the printer is in use.

2.2.64.
The two methods for accessing I/O SEQUENTIAL disk files, first implemented in patch #40 to the Mark VI.1 MCP, are also included in Mark VII.0. Appendix C to the revised Software Operational Guide contains detailed documentation on all methods available for accessing disk files on B 1800/B 1700 systems.

2.3. Known Errors and Restrictions.

2.3.1.
The "INVALID.CHARACTERS=1" option of the FILE statement is not functional.

2.3.2.
A "CLOSE PURGE" on an unlabeled tape does not work, because the MCP cannot find a tape volume-identifier. This will be considered a permanent restriction.

2.3.3.
The MCP will terminate any program that attempts to OPEN (output) a file with odd-length logical records on Phase-Encoded (PE) tape. Input files with odd-length records are allowed; however, caution should be used to insure that "garbage" records are not read by the program. This will be considered a permanent restriction.

2.3.4.
The "retry count" in the ELOG entry for a non-user I/O (i.e., an I/O descriptor not associated with a File Information Block) will always be recorded as zero.

2.3.5.
A program attempting to open a blocked file on an 80- or 96-column data recorder will be DS-ed by the MCP. This will be considered a permanent restriction.

2.3.6.
If any system utility program using "Direct I/O" (such as SYSTEM/DISK.INIT) is DS-ed, the drive it was using must be readied manually ("RYDCx" or "RYDPx") to return it to the system. This will be considered a permanent restriction.

2.3.7.
Record count, block count, and cumulative time open is incorrect in the LOG for REMOTE and QUEUE files only.

2.3.8.
Jobs still in the schedule when a CLEAR/START occurs are marked in the log as "RS-ED" rather than "ABORTED".

2.3.9.
A transferred LOG, ELOG, or SPOLOG file that has only one disk area assigned is not reduced in size to eliminate wasted sectors at the end of the area.

2.3.10.
The MCP fails with unpredictable results if an attempt is made to enter an OF message referencing a MICR Reader-Sorter file.

2.3.11.
A "task" (for example, a SORT intrinsic) in the mix is counted by the MCP as a separate job when determining whether or not the MIX LIMIT has been reached.

2.3.12.
It is not possible to KP a PRIVATE file by file-identifier, even if the KP message is prefaced with the appropriate usercode and password.

2.3.13.
Column one of the beginning label card on an 80-column punch output file is not being punched with an invalid character.

2.3.14.
If an ADD, DUMP, LOAD, or UNLOAD message is preceded by a USER=<usercode>/<password>, no file-identifier having a family-name may be specified in the file-name list if the family-name is not the same as the specified usercode. For example, the following statements are valid:

```
US SITE/A DUMP TO T =/=;
USER=X/Y LOAD FROM TAPE A, B, (X)/C;
```

The following statement, however, is illegal:

```
USER A/B ADD FROM XXX C/D;
```

2.3.15.
If the message "PGMTx;POMTx" is entered for a Phase-Encoded (PE) tape, the tape is unloaded before the purge is done. The next tape mounted on the drive will be purged if it contains a write ring.

2.3.16.
The RX input message does not remove any files having a usercode as the family-name, even if they are not specified in the file-name list.

2.3.17.
Entering a CL input message referencing a tape unit doing a "SPACE" communicate on a multi-file tape hangs the MCP.

2.3.18.
The following SPO input sequence generates "garbage" SPO messages and does not assign the printer properly.

```
<job-number>FMLPx;EX <program-id>
```

2.3.19.
By default, the line width for the CRT SPO is erroneously set to 72 instead of the correct value of 80. A "KB W 80" is required after CLEAR/START to set the proper maximum line width.

2.3.20.
If an SDL/UPL program has frozen itself (FREEZE_PROGRAM), the MCP erroneously resets the "freeze" condition if the program does a REMOTE file open. This does not occur during other types of file opens, nor does it occur if the program was executed with the FREEZE option specified.

NOTE

In Mark VIII.0, the FREEZE_PROGRAM/THAW_PROGRAM communicate mechanism will be changed. For each program, the MCP will maintain an 8-bit "counter" of FREEZE communicates, which will be bumped for each FREEZE_PROGRAM and decremented for each THAW_PROGRAM. A program will be considered "frozen" as long as the counter is non-zero. Overflow/underflow is ignored; thus, FREEZE_PROGRAM and THAW_PROGRAM communicates must be properly matched in order to work as intended.

2.3.21.
The LOCKED message displayed after an SNL input message displays the old tape serial number rather than the new tape serial number.

2.3.22.
Attempting to modify a code file via an OBJ statement using quotes (") and semicolons (;) to delimit parameters can result in various syntax errors. For example:

```
COMPILE ... OBJ FILE <file-id> PACK.ID=" ";CG 9999999
```

2.3.23.
Relabeling (RL message) the PBD.DESIGNATION disk (BD message) is no longer allowed.

2.3.24.
Output printer backup files that have USER.BACKUP.NAME specified are not entered in the AUTOPRINT QUEUE if they are directed to a disk other than the PBD.DESIGNATION disk.

2.3.25.
The RB DMP/= and BF DMP/= input messages do not work correctly when the PBD.DESIGNATION specifies a user disk.

2.3.26.
When a disk file is opened INPUT by one or more programs, if another program opens the same file I/O LOCK ACCESS, a lock bit gets set in the disk file header (DFH). When the program which opened the file I/O LOCK ACCESS goes to end-of-job or closes the file, the lock bit is not reset. This prevents other programs from opening the file until it is closed by all programs.

2.3.27.
If a file is modified to a non-existent unit (UNIT.NAME attribute of the FILE statement), the program will be terminated if it attempts to open the file.

2.3.28.
The RX input message requires that a slash (/) separate the unit-mnemonic (or pack-id) and serial-number specifications. For example:

```
RX DPA/123456 FILEA, FILEB/FILEC, ... ;
```

2.3.29.
The MCP will erroneously reinstate a program if an ST message followed by a GO message is entered for that program while it is awaiting DS or DP. The results of reinstating the program in such a case are unpredictable, and can cause system failures, data corruption, and so forth.

2.3.30.
A program in the schedule which has the NODIF attribute set is RS-ed if the parent program terminates abnormally.

2.3.31.
Attempts to relabel (RL input message) a system pack causes an MCP READ OUT-OF-BOUNDS halt.

2.3.32.
A COMPILE and GO with an error in any OBJ attribute causes the "GO" portion to be RS-ed and the temporary object code file to be removed from disk.

2.3.33.
Any job attempting to open an input file with a RECORD.SIZE equal to zero and the DEFAULT attribute reset will be DS-ed.

2.3.34.
If two programs are accessing the same disk file and one of them closes the file with PURGE, the file will be removed from disk when it is closed by the second program (regardless of which CLOSE option is specified).

2.3.35.
When attempting to call SORT/QSORT with large records, certain combinations of sort memory size, number of input records, and number of sort keys can cause an MCP "DIVIDE BY ZERO" halt (L=200000A2).

Temporary fix: Increase the amount of memory assigned for the SORT.

3. Interpreters and Firmware

3.1. Introduction.

The Mark VII.0 firmware contains a number of significant changes, most notably pertaining to the new Memory Management algorithms.

CLEAR/START and SYSTEM/INIT have been redesigned, resulting in a different memory layout following the CLEAR/START operation, implementing the XM function, and providing more information in memory dumps.

The MCP.LEVEL field checked by the MCP at BOJ for all programs has been raised from 1 to 2 for all interpreters, meaning that Mark VII.0 interpreters cannot be retrofitted to Mark VI.1 systems. In addition, Mark VII.0 CLEAR/START cassettes are completely incompatible with all other release versions.

3.2. Enhancements.

3.2.1. GISMO

a. The task of finding a suitable area of memory when requested is now a GISMO function. This "FINDWINDOW" procedure uses appropriate criteria such as availability and memory priority to select the precise area to be deallocated. The SMCP, using the location and size passed to it by GISMO, is still responsible for rolling out any data segments in the area to disk.

b. New Performance Monitoring capabilities, as described in appendix B of the revised Software Operational Guide, have been implemented in the Mark VII.0 GISMO. Note that all of the performance-monitoring functions require the use of GISMO/DEBUG.

c. With the exception of console interrupts when the SDL Interpreter is in control, all interpreter halts are the result of information passed to GISMO, which performs the actual halt.

d. A problem has been corrected in which GISMO, in certain instances, was incorrectly handling disk pack retry operations with skew and offset, treating them instead as verification operations.

3.2.2. MICRO.MCP (MMCP)

a. The Mark VII.0 MMCP release has few changes of user significance. Certain changes in the MMCP-SMCP interface have been made, none of which are visible to the system user.

b. A problem involving the incorrect handling of end-of-tape (EOT) on Paper Tape Reader files has been corrected.

3.2.3. CLEAR/START

A number of functions have been moved from SYSTEM/INIT to CLEAR/START, where they are treated as subprograms by SYSTEM/INIT. This causes the physical quantity of tape used by CLEAR/START to be noticably longer than that used in the Mark VI.1 cassette. Thus, users should not suspect that the computer is "running away" when the tape continues to turn beyond the customary point.

NOTE

Depressing the CLEAR button when beginning a CLEAR/START destroys the TAS pointer in the processor. It is therefore essential, when taking a memory dump after an uninterruptible system loop (one for which the HALT button must be depressed to stop the processor) that the contents of the TAS register be recorded before doing the CLEAR/START. Likewise, for uninterruptible hangs, the user is urged to record FA and FB as well as the registers requested on the dump (L, T, X, Y, and A, plus MBR and TOPM on 1720 series processors). Interruptible loops (those which are halted by the INTERRUPT switch) become defined halts for which no additional information must be obtained by the system operator when taking a dump.

3.2.4. System Initializer (SYSTEM/INIT)

a. SYSTEM/INIT is now loaded over GISMO in memory, if one existed; otherwise, it is loaded in mid-memory. This allows the interrupt queue to be printed in memory dumps, whereas in past releases this information at the top of memory was wiped out by the loading of SYSTEM/INIT. In many cases, this permits diagnosis of system hangs caused by faulty I/O controls generating bad interrupt entries.

b. The A Stack and processor scratchpads are now extracted by SYSTEM/INIT and included in SYSTEM/DUMPFILE for printing by MCPII/ANALYZER.

c. SYSTEM/INIT now loads GISMO beyond the SMCP LIMIT.REGISTER rather than in low memory, providing greater protection from accidental corruption.

d. The XM function, which links areas out of available memory, has been coded in SYSTEM/INIT.

e. The memory dump functions previously performed by SYSTEM/MEMDUMP are now done by SYSTEM/INIT. SYSTEM/MEMDUMP is, therefore, no longer released.

3.2.5. SDL Interpreter

No major changes have been made to the Mark VII.0 SDL Interpreter.

3.2.6. COBOL Interpreter

No major changes have been made to the Mark VII.0 COBOL Interpreter.

3.2.7. BASIC Interpreter

No major changes have been made to the Mark VII.0 BASIC Interpreter.

3.2.8. RPG Interpreter

The divide algorithm in the RPG Interpreter has been optimized, improving the speed of divide and square root operations.

3.2.9. FORTRAN Interpreter

The addition of new code in the Mark VI.1 FORTRAN Interpreter pushed the code used for double-precision handling beyond the M.MEMORY.MINIMUM point. This had the unintentional effect of degrading performance of programs making heavy use of double-precision functions on a B1720 processor having 4KB of Control Memory. The Mark VII.0 FORTRAN Interpreter has been somewhat rearranged to return the double-precision code to Control Memory, thus restoring the original speed of such programs.

3.3. Known Errors and Restrictions.

3.3.1. SDL Interpreter

a. The SDL Interpreter is incorrectly checking for field overflow beyond the 65,535-bit maximum length capability of SDL/UPL when performing X_ADD and X_MUL operations. This can result in erroneous "NAME/VALUE STACK OVERFLOW" errors instead of the desired "EXPRESSION OUT OF RANGE" error. Note that this limit applies to the temporary area in the VALUE STACK in which the result is formed; the length of the field in which the result is ultimately stored is irrelevant.

b. A CLEAR of an array of type RECORD produces incorrect results.

c. The LENGTH function, when applied to an item of type RECORD, returns the length in characters rather than bits.

3.3.2. FORTRAN Interpreter

a. The CSIN (Complex Sine) intrinsic produces results with incorrect sign of the real part in certain cases.

b. If a portion of an array is passed to a subprogram, which in turn passes a still smaller portion to a second subprogram, the FORTRAN Interpreter in certain cases attempts to access a non-existent data segment, resulting in program termination.

3.4. Interpreter Verification Attributes.

SDL Interpreter (SDL/INTERP1M and SDL/INTERP1S)

```
ARCHITECTURE.NAME: "SDL"
COMPILER.LEVEL: 1
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE:
  2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13
Segments: 13
```

COBOL Interpreter (COBOL/INTERP1M and COBOL/INTERP1S)

```
ARCHITECTURE.NAME: "COBOL"
COMPILER.LEVEL: 1
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: 0
Segments: 4
```

RPG Interpreter (RPG/INTERP1M and RPG/INTERP1S)

```
ARCHITECTURE.NAME: "RPG"
COMPILER.LEVEL: 1
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: 0, 1, 2, 3, 4
Segments: 7
```

BASIC Interpreter (BASIC/INTERP3M and BASIC/INTERP3S)

```
ARCHITECTURE.NAME: "BASIC"
COMPILER.LEVEL: 3
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: None
Segments: 1
```

FORTRAN Interpreter (FORTRAN/INTERP1M and FORTRAN/INTERP1S)

```
ARCHITECTURE.NAME: "FORTRAN"
COMPILER.LEVEL: 1
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: 0, 1
Segments: 1
```

B500 Interpreter (B500/INTERP2U)

```
ARCHITECTURE.NAME: "B500"
COMPILER.LEVEL: 2
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: None
Segments: 1
```

IBM 1400 Interpreter (1400/INTERP2U)

```
ARCHITECTURE.NAME: "1400"
COMPILER.LEVEL: 2
MCP.LEVEL: 2
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE: None
Segments: 1
```

GISMO (GISMO and GISMO/DEBUG)

```
GISMO.LEVEL: 2
ARCHITECTURE.ATTRIBUTE bits that are TRUE:
  1, 3, 21, 22, 23, 41, 42, 71, 73, 74, 75, 76
Segments: 1
```

MICRO.MCP (MCPII/MICRO.MCP and MICRO.MCP/DEBUG)

```
ARCHITECTURE.ATTRIBUTE bits that are TRUE:
  1, 3, 21, 22, 23, 20, 43, 44, 45, 52, 53
  70, 71, 73, 75, 77, 79
Segments: 16
```

4. RPG

4.1. Introduction.

The Mark VII.0 release of the RPG compiler includes corrections to outstanding problems, the capability to mairtain a library of source files, and increases the maximum number of files allowed in an RPG program. Each of these changes is documented in the paragraphs that follow.

4.2. Enhancements.

4.2.1.
The following problems reported in the Mark VI.1 release letter have been corrected (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. Right-signed numeric fields written to a printer file no longer have their sign fields replaced with an "F" (4.3.2).

b. The RPG compiler no longer terminates with an INVALID SUBSCRIPT error when the File Specification card has entries shifted to the left (4.3.6).

c. Sequence checking is now performed for compile-time tables and arrays when ascending or descending sequence is specified (4.3.12).

4.2.2.
The "X" edit code has been corrected to remove only the positive sign.

4.2.3.
An RPG program with more than 64 arrays declared no longer terminates with a WRITE OUT-OF-BOUNDS error.

4.2.4.
When using the "MOVEL" operation to move a smaller numeric field to a larger alpha field, the sign is placed in the zone position of the rightmost character moved. For example, "MOVEL" of '-6789' to a field containing 'ABCDEFGH' will give the result '678REFGH' (R represents -9).

4.2.5.
The RPG compiler now supports source file maintenance. Refer to paragraph 4.4.2 for temporary documentation on the syntax.

4.2.6.
The RPG compiler no longer terminates with an INVALID SUBSCRIPT error when a constant in Factor 1 or Factor 2 of the Calculation Specifications contains a 9-digit field followed by a decimal point.

4.2.7.
The RPG compiler will now syntax null constants (two single quotes on a line in the Output Specifications).

4.2.8.
The RPG compiler will no longer terminate with an INVALID CASE error under the following conditions:

a. A "Z-ADD" operation is to be performed,

b. Factor 2 is a literal with decimal positions declared, and

c. The result field has zero decimal positions declared.

4.2.9.
The divide algorithm in the RPG Interpreter has been optimized. The new interpreter improves the speed for division when:

a. Each operand is less than six bytes in length, or

b. There are leading zeros to the left of the decimal point.

Square root operations have also been significantly improved.

4.2.10.
When compiling with switch 1 set to 1, a printer file called "ERROR.LINE" will be created, containing any syntax errors or warnings detected in the compiled RPG program. The MCP automatically appends a "SW1=1" attribute to any COMPILE control instruction zipped with a session number between 1 and 1023, inclusive (for example, a COMPILE performed through CANDE).

4.2.11.
A warning will be generated by the RPG compiler when the value for "$ RPERA" is not an even multiple of the records per block. The RPG compiler will automatically adjust the sizes upward as necessary.

4.2.12.
The RPG compiler has optimized the addition of records to an indexed file, especially when the records are added in order.

4.2.13.
Input Specifications no longer have to repeat the decimal positions for numeric arrays.

4.2.14.
Multiple Control Card Specifications will now be given a syntax error.

4.2.15.
Matching record and control level indicators are no longer required to be in the first input field declared by a group of consecutive field record relation indicators.

4.2.16.
"$ NAMES" no longer generates two copies of the "NAMES" listing when there are syntax errors.

4.2.17.
A warning is now generated by the RPG compiler when the "1P" indicator occurs on an exception or total line with "AND" or "OR" specified.

4.2.18.
A syntax error is now generated by the RPG compiler when the "FROM FILE" field is specified on an Extension Specifications line and the "ENTRIES PER RECORD" field is left blank.

4.2.19.
A syntax error is now generated by the RPG compiler when the floating dollar or check protect signs are encountered with the edit codes "X", "Y", or "Z".

4.2.20.
All known line skipping problems with RPG I have been corrected.

4.2.21.
The RPG compiler now correctly handles output chain files when a "B" is entered in column 16 of the Control Card Specification.

4.2.22.
The RPG compiler now correctly handles arrays and tables when the elements are larger than 256 bytes and when the array or table is declared as being ordered.

4.2.23.
The RPG/BTF program has been enhanced to be more compatible with the RPG compiler. The enhancements are as follows:

a. The internal file name for the input card file is now "CARDS" instead of "SOURCE".

b. The DEFAULT file-attribute bit is now set.

4.2.24.
Line skipping is now performed correctly for programs compiled with RPG I. For example, if the line printer is on line 5 and a skip to line 5 is requested, then a skip is not performed.

4.2.25.
A channel 12 punch in the carriage control tape for line printers is now allowed when using SYSTEM/BACKUP.

4.3. Known Errors and Restrictions.

4.3.1.
A modify of the PACK.ID for indexed-sequential update files with additions is not forwarded to the RPG/REORD program, since the file information used by RPG/REORD is the result of a compile-time interface function. This restricts RPG/REORD to using the file information specified at compile time. For improved performance and flexibility, it is suggested that the standard SORT be used instead of RPG/REORD by removing all "$ REORG" cards.

4.3.2.
The "Y" edit code now only operates correctly on a field with four or six characters. A packed field implies a sign.

4.3.3.
The RPG compiler does not syntax a "SETOF" operation code that turns off the last record (LR) indicator. The results can be unpredictable if used.

4.3.4.
Due to the tag file naming convention, RPG indexed-sequential files cannot be used when executing under the file security system (unless executed under a privileged usercode/password). "B"-indexed files are not affected by the file security system even when executed under a non-privileged usercode/password.

4.3.5.
Source file maintenance of compile-time vector data is not possible.

4.3.6.
Tape files are not closed with RELEASE when the TAPE REWIND field, column 70 of the File Description Specifications, is left blank.

4.3.7.
The "TESTZ" operation does not test the zone position of numeric fields.

4.3.8.
The "TESTN" operation is not a valid command for the Mark VII.0 release of the RPG compiler.

4.3.9.
A "J" or "I" in column 21 of the Control Card Specifications and a "1" edit code in column 38 of the Output Specification yields ",00" for a zero value instead of "0,00".

4.3.10.
The RPG compiler generates two syntax errors when a null constant (two single quotes) is encountered in the Output Specifications. The syntax error message "INVALID CONSTANT SIZE" is correct; however, the syntax error message "INVALID ENDING POSITION" is incorrect.

4.3.11.
A variable with a value of zero and decimal positions declared fills asterisks only to the left of the decimal point when the "K" edit code and the "*" check protect symbol are specified. The result should be asterisk fill in all positions.

4.3.12.
The RPG compiler terminates with an INVALID SUBSCRIPT error when the tag name referenced by a "GOTO" operation begins in column 34 instead of column 33.

4.3.13.
The RPG compiler generates the following syntax error when the result field length of a "MVR" operation is greater than the length of the field specified in Factor 2 of the previous "DIV":

RESULT FIELD LENGTH FOR MVR MUST EQUAL FACTOR 2 FOR PREVIOUS DIV

4.3.14.
An RPG program will terminate with an INVALID SUBSCRIPT error when a Calculation Specification contains the following:

a. 'MOVE' in the Operation field,

b. A subscripted array (numeric) in Factor 2,

c. The subscript of the subscripted array (specified in Factor 2) is entered in the Result Field, and

d. The numeric value of the subscripted array which is being moved is larger than the Entries Per Array specified on the Extension Specifications.

For example, if the array 'ARR' has 10 elements and 'ARR,X' equals 11, then 'MOVE ARR,X (to) X' results in an INVALID SUBSCRIPT error.

4.3.15.
The RPG compiler generates incorrect code under the following conditions:

a. The elements of a dynamic vector are described (in the Extension Specifications) as unpacked numeric with a length that is an even number of bytes, and

b. The vector is described as packed in the Input Specifications.

Vectors having a length that is an odd number of bytes generate correct code.

4.4. Temporary Documentation.

4.4.1. Number of Files

The maximum number of files that may be declared in an RPG program has been increased to 31. Each indexed file and its associated tag file counts as one file.

4.4.2. Source File Maintenance

The following dollar card options may appear anywhere within the B 1800/B 1700 source file:

$ CHECK — Causes all patching records to be sequence checked. Syntax errors are generated for any patch record that is out of sequence. If the "$ CHECK" option is not specified, then sequence warnings are issued. The warning is denoted by an "S" appearing to the left of the sequence number on the output listing.

$ MERGE — Causes the following records in the patch file to be merged with an existing source file. The source file is expected to reside on tape or disk (disk by default), and has internal and external file names of "SOURCE". The DEFAULT file attribute is set.

If it is desirable to change the external file name or input device from disk to tape, then a FILE control card must be used. The "$ NEW" option can be used with the "$ MERGE" option to create a new output source file with patches.

The sequence fields are used to control the merge sequence. If the sequence fields of a patch record and a source record match, then the patch record replaces the source record; otherwise, patch records are merged with the tape or disk file. The existing source file remains unchanged.

A "P" is appended to the left of the sequence field on the source listing to indicate that the image is a patch record.

$ NEW — Creates a new output source file. If the "$ MERGE" option is used, then the patches are included in the new source file. The new source file is created on disk or tape (disk by default), having internal and external file names of "NEWSOURCE". If it is desirable to change the external file name or output device from disk to tape, then FILE control cards must be used.

$ SEQ — Starts sequencing the subsequent source lines on the output listing. If the "$ NEW" option is used, then the subsequent new source file is sequenced. The starting sequence number begins with the number contained in the sequence field, columns 1-5, of the "$ SEQ" option. If the sequence field is blank, then the starting sequence number is zero. The subsequent records have sequence numbers which are incremented by the value contained in columns 22-24 of the "$ SEQ" option. If columns 22-24 are blank, then 010 is assigned.

The following options may appear anywhere within the B 1800/B 1700 RPG source program. These options direct the compiler to perform specific source file maintenance functions. All of the following options may be reset.

$ LIBR — Copies source records from a library file located on tape or disk to a new or merging file. The $ LIBR option can reference different library files. Columns 15-24 contain the pack-id, columns 25-34 the family-name, columns 35-44 the file-id, columns 45-49 the starting sequence number in the library file (optional), and columns 50-54 the ending sequence number in the library file (optional).

An "L" is placed to the left of the sequence field in the output listing to identify that the record is from a library source file.

$ NEWID — Causes the 6-character identifier specified in columns 15-20 of the "$ NEWID" option card to be placed in columns 75-80 of the subsequent lines on the output listing. If the "$ NEW" option is used, then the specified identifier is also placed in columns 75-80 of the subsequent source records in the new source file.

$ VOID Deletes records of the source file on the output listing. If the "$ NEW" option is used, then the deleted records are not included in the new source file. Source lines or records are deleted up to and including the 5-digit void limit. If the void limit field, columns 20-24, is blank, then only the source record with a sequence number matching the sequence field of the "$ VOID" option card is deleted.

5. COBOL

5.1. Introduction.

The Mark VII.0 release of the COBOL compiler includes the syntax for "OPEN EXTEND" for sequential disk files, corrections for several problems found in prior releases, and Burroughs extensions to the "AT END" and "INVALID KEY" clauses for the Indexed I-O Module.

5.2. Enhancements.

5.2.1.

The following problems reported in the Mark VI.1 release letter have been corrected (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. The compiler now checks the amount of dynamic memory during BOJ, and aborts immediately if less than 18000 bits is assigned (5.3.7).

b. "WORKING-STORAGE SECTION" is now optional with or without "DATA-BASE SECTION" (5.3.14).

c. Misspelling or omitting "INVOKE" no longer causes "NO PROVISION FOR EOF ON DNINFO" during compilation (5.3.18).

d. "DISPLAY <edited-field>" now generates correct code (5.3.25).

e. The message concerning "INSUFFICIENT DYNAMIC MEMORY" is now printed at the end of the listing, as well as being displayed on the SPO (5.3.33).

f. The "TAG-SEARCH" option of the "SORT" statement, when used with J-signed keys, now generates correct code (5.3.36).

g. A file assigned to "CARD96" now generates the correct hardware type (5.3.38).

h. A "PERFORM ... VARYING ... AFTER ..." statement now sets the "FROM" data-name to its "current" value rather than its "initial" value (5.3.39).

5.2.2.

"COMPUTE C = <expression>", where C is too small to hold the result and no "ON SIZE ERROR" clause is specified, now results in left digit truncation instead of program termination.

5.2.3.
The statement "COMPUTE A = 1 + B/(C * 4)" no longer causes the divisor and dividend to be reversed.

5.2.4.
Multiple "USE" procedures for a file without an "INVALID KEY" clause on a READ or WRITE no longer causes compile-time errors.

5.2.5.
An embedded data set immediately preceded by a group or elementary level with an "OCCURS" clause no longer causes the compiler to abort.

5.2.6.
The file "ERROR.LINE" is now unblocked.

5.2.7.
Correct code is now generated when both data segmentation and ISAM are used in the same program.

5.2.8.
COBOL ISAM programs without a WRITE statement no longer cause an "INVALID SUBSCRIPT" error during execution.

5.2.9.
For Indexed files, the tag file is now able to contain as many records as its related data file. Refer to paragraph 5.4.8 for further information.

5.2.10.
Erroneous "DUPLICATE NAMES" syntax errors no longer occur when the dynamic memory allocated to the compiler is less than:

(the number of explicit data names + 2) * 38

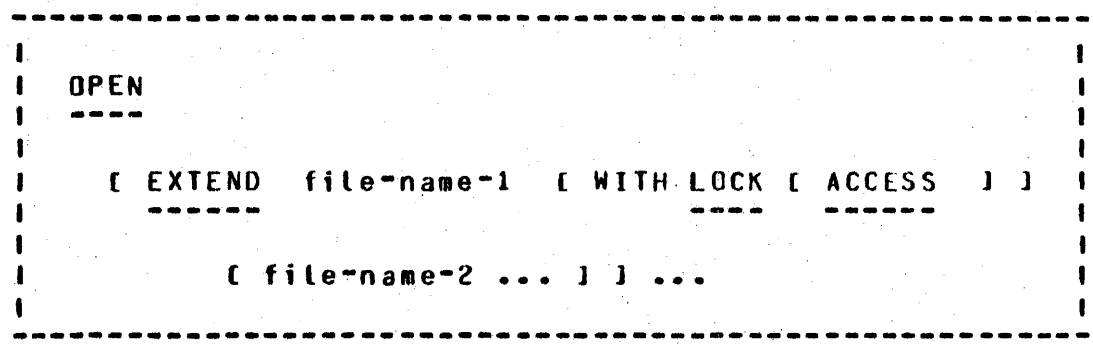
5.2.11.
The statement "COMPUTE A = B ON SIZE ERROR", where B is smaller in size than A now generates correct code.

5.2.12.
Erroneous "PICTURE TABLE FULL" syntax errors no longer occur when the number of unique pictures is greater than 255.

5.2.13.
A "WRITE OUT-OF-BOUNDS" or "INVALID SUBSCRIPT" error no longer occurs when a file named in a "MERGE" statement has a "VALUE OF ID IS data-name" clause.

5.2.14.
An I-O verb within an "INVALID KEY" clause no longer causes an "INVALID SUBSTRING" during compilation.

5.2.15.
The "EXTEND" phrase has been added to the COBOL syntax for the OPEN statement, as follows:



Refer to paragraph 5.4.1 for further information and examples.

5.2.16.
"SEARCH ... WHEN ..." may be terminated by a conditional or compiler-directing statement, as well as an imperative statement.

5.2.17.
"OPEN file-name ACCESS" is now syntaxed correctly.

5.2.18.
"DISPLAY Z9.9" now includes the last character in the display.

5.2.19.
The compiler ignores all except the last "BLOCK CONTAINS" clause when multiple "BLOCK CONTAINS" clauses are used in a File Description (FD).

5.2.20.
"PERFORM ... VARYING ..." nested deep in an "IF ... ELSE ..." statement is now syntaxed correctly.

5.2.21.
The compiler can now handle up to 9999 errors and warnings to be printed at the end of the listing.

5.2.22.
ISAM subroutines are now segmented. Refer to paragraph 5.4.11 for further information.

5.2.23.
COBOL/XREF now recognizes and picks up library COPY statements with embedded spaces.

5.3. Known Errors and Restrictions.

5.3.1.
A subscripted reference where the subscripts are numeric literals causes a syntax error, "LITERAL SUBSCRIPT CAUSES OUT OF BOUNDS ERROR", if the compiler calculates an address that exceeds the boundary. Therefore, it is possible to exceed the "OCCURS" count and not receive a syntax error. For example:

```
01 A.
   03 B OCCURS 5 TIMES.
      05 C OCCURS 10 TIMES PC X.

C(5,11) will receive a syntax error.
C(2,11) will not receive a syntax error.
```

5.3.2.
An "OPEN INPUT" on a printer or punch file is not syntaxed.

5.3.3.
A COBOL program is unable to inquire as to the presence or status of a disk file.

5.3.4.
A file declared "OPTIONAL" is not being syntaxed if opened OUTPUT.

5.3.5.
During compilation, it is not possible to call the COBOL/XREF program by a dollar option.

5.3.6.
In order to compile with COBOL on a system having a memory size of 48KB or less, the following modifications to the compiler should be made:

```
? MODIFY COBOL
? FILE NEWSOURCE AREAS   = 10 BUFFERS = 1;
? FILE LIBRARY   AREAS   = 10 BUFFERS = 1;
? FILE REPORT    AREAS   = 10 BUFFERS = 1;
? FILE ALLDNOUT  AREAS   = 10 BUFFERS = 1 RECORDS.BLOCK 1;
? FILE DNFILEX   AREAS   = 10 BUFFERS = 1 RECORDS.BLOCK 1;
? FILE ADNFILE   AREAS   = 10 BUFFERS = 1 RECORDS.BLOCK 1;
? FILE ADFILE    AREAS   = 10 BUFFERS = 1 RECORDS.BLOCK 1;
? FILE SEGFILE   AREAS   = 10;
? FILE CARDS     BUFFERS =  1;
? FILE SOURCE    BUFFERS =  1;
? FILE LINE      BUFFERS =  1;
```

In addition, a MEMORY card should be used at compile time:

```
? COMPILE <program-id> COBOL LIBRARY MEMORY = 18000
? DATA CARDS
  <source cards>
? END
```

Dynamic memory must not be modified permanently in the compiler.

5.3.7.
The compiler does not check for correct spelling of the word "IS" in a "VALUE OF ID IS" or a "DATA RECORD IS" clause.

5.3.8.
Reserved words used as data-names may cause compiler loops.

5.3.9.
It is not possible to specify the number of lines allowed per page on the compile listing.

5.3.10.
The "TIME" option does not return the time in tenths of seconds.

5.3.11.
"READ" statements are unable to perform stacker selection.

5.3.12.
The compilation date/time does not print on the same line as the "DATE-COMPILED" statement.

5.3.13.
"MONITOR" overprints the last print file line when printing the first monitor line of a page.

5.3.14.
"WRITE data-name-1 FROM data-name-2" does not result in a "RECEIVING FIELD TRUNCATION" warning if the size of data-name-2 exceeds that of data-name-1.

5.3.15.
The compiler will fail with a "READ OUT-OF-BOUNDS" when compiling with many condition-names and insufficient dynamic memory.

5.3.16.
"REMARKS ..." coded in column 12 following a "DATE-COMPILED" statement causes an incorrect "MISSING PERIOD" syntax error.

5.3.17.
In some cases, dollar options may be put into effect before they appear in the source.

5.3.18.
The word "KEY" in a COMPUTE statement is treated as a noise word instead of being syntaxed.

5.3.19.
Compares of computational items to non-numeric literals are not being syntaxed.

5.3.20.
The editing symbols "+" and "-" do not function correctly if the field to be edited contains minus zero. Note that minus zero can only result from input or truncation, not from arithmetic operations.

5.3.21.
There are several problems concerning the placement of a syntax error message together with its related source statement.

5.3.22.
A missing period in a "MODIFY ... VIA ..." statement may cause all succeeding paragraph-names to be flagged as duplicates.

5.3.23.
"DUMP <edited-field>" generates incorrect code.

5.3.24.
The "FILE CONTAINS" clause is not syntaxed for number of areas greater than 105.

5.3.25.
The compiler fails with "INVALID SUBSCRIPT" on the following condition:

```
IF (A * (10 ** (2-B))) > 9999999 OR < -9999999 GO TO C.
```

5.3.26.
The appearance of an unmatched and unexpected "DATA" in the "DATA DIVISION" may cause a "NO PROVISION FOR EOF ON DNINFO" during compilation.

5.3.27.
When "VALUE IS HIGH-VALUES" is specified, the sign of a "signed computational" item is incorrectly initialized. For example:

```
ABC PIC S99 CMP VA HIGH-VALUES.   (Initialized as 99Ø3)
```

5.3.28.
Multiple receiving fields on a "READ ... INTO ..." are not syntaxed.

5.3.29.
The appearance of the character @78@ in a literal may cause erroneous syntax errors.

5.3.30.
Attempts to create library files may be syntaxed when "DECIMAL-POINT IS COMMA" is specified and a period follows the library-name. Note that the period is not required.

5.3.31.
Use of a "MONITOR" statement with names of 30 characters in length may cause compiler errors.

5.3.32.
The statement "ADD -literal data-name-1 GIVING data-name-2" loses the minus sign for the literal.

5.3.33.
Referencing multiple files by a single "CLOSE" statement in some cases is syntaxed erroneously. Note that:

```
CLOSE A, B, C
```

generates the same code as and is no more efficient than:

```
CLOSE A, CLOSE B, CLOSE C
```

5.3.34.
The statement "WRITE ... BEFORE <n> LINES" only allows values for <n> between 1 and 63, inclusive.

5.3.35.
"$ CODE" before the final paragraph in a program causes code to be generated for the entire program.

5.3.36.
"DIVIDE A INTO B GIVING B REMAINDER C" gives incorrect results for C.

5.3.37.
The error message regarding data length limit does not state the correct problem, which is that the limit only pertains to FILE SECTION entries and referenced WORKING-STORAGE SECTION entries.

5.3.38.
"<data-name> PIC SV999 VA 0." causes a "RECEIVING FIELD TRUNCATION" warning message to be printed.

5.3.39.
The compiler syntaxes 88 levels in DMS descriptions.

5.3.40.
"GO TO #INV KEY RTN#." causes the compiler to loop in PARSE.

5.3.41.
Misleading syntax errors are produced when the continuation of a DMS statement begins prior to column 12.

5.3.42.
"READ ... INTO ..." works incorrectly if the first defined record for the file being read is smaller than the actual record size.

5.3.43.
Inclusion of a second "FILE SECTION" is not syntaxed and causes multiple uses of the same File Parameter Block (FPB).

5.3.44.
At execution time an "INVALID TYPE IN COMMUNICATE MESSAGE POINTER" error occurs when using "$ NOCOP" and ISAM files.

5.3.45.
An 88 level with both numeric and non-numeric literals is not syntaxed.

5.3.46.
The statement "IF (- data-name-1) GREATER data-name-2 ..." sometimes causes an "INVALID SUBSCRIPT" during compilation.

5.3.47.
"SEGMENT-LIMIT IS <integer>" is not syntaxed for the hyphen and <integer> is not used.

5.3.48.
Syntax errors #161 and #337 occur when the data-name in a "VALUE OF ID IS data-name" clause is a RENAMES operand.

5.3.49.
A "WAIT" statement immediately followed by a monitored label generates incorrect code (a zero length literal).

5.3.50.
The statement "COMPUTE A = B ** C(I J)" causes a "FIXUP COMPILER ERROR" message, if it is the first exponentiation encountered in the source file and "C" is subscripted.

5.3.51.
A missing quote in a COPY statement causes the compiler to abort with a "NO PROVISION FOR EOF ON ALLDNOUT" message.

5.3.52.
The compiler currently allows more than one data base to be invoked in a single program, causing unpredictable results during execution.

5.3.53.
The nesting of DMS verbs may cause the compiler to abort with a NAME/VALUE STACK OVERFLOW.

5.3.54.
An ISAM file with "VALUE OF ID IS data-name" is not syntaxed and does not function properly at run time.

5.3.55.
The following statement generates code that reverses the dividend and divisor:

```
COMPUTE A ROUNDED = B - C - (D * (B - C)) / 100.
```

The following statement, however, functions correctly when the extra parentheses are used:

```
COMPUTE A ROUNDED = B - C - ((D * (B - C)) / 100).
```

5.3.56.
Moving a data-name to itself, where the receiving field contains "BLANK WHEN ZERO", results in the entire field containing zeros.

5.3.57.
The use of paragraph names instead of section names with the "SORT" verb is not syntaxed and at run time does not function properly.

5.3.58.
If the label names in a "GO TO ... DEPENDING ON" statement are separated by commas, then the last label name must be followed by a comma, or no code generation takes place.

5.3.59.
In the statement "SUBTRACT 1 FROM literal", the literal is not syntaxed and causes the compiler to abort with "WRITE OUT-OF-BOUNDS".

Temporary fix: Replace the literal with a data-name.

5.3.60.
When the number of "01" levels exceeds 127, wraparound addressing and run-time failures occur.

5.3.61.
The statement "EXAMINE data-name REPLACING LEADING SPACE." is incomplete, but is not syntaxed.

5.3.62.
A MOVE of "LOW-VALUES" to a subscripted data-name whose picture is CMP-3 causes a "FIXUP COMPILER ERROR".

5.4. Temporary Documentation.

5.4.1. OPEN EXTEND

The "EXTEND" phrase can only be used with sequential disk files. When the "EXTEND" phrase is specified, the "OPEN" statement positions the file immediately following the last logical record of that file. Subsequent "WRITE" statements referencing the file add records to the file as though the file had been opened with the "OUTPUT" phrase.

Example:

In the following COBOL program segment, assume that the files FILE-A and FILE-B are disk files:

```
    01  IX  PIC 99 VA 0.
        :
        :
    WRITE-FILES.
        ADD 1 TO IX.
        WRITE RECORD-A FROM IX.
        WRITE RECORD-B FROM IX.
    MAIN-CODE.
        OPEN OUTPUT FILE-A, FILE-B.
   * BOTH FILES ARE OPENED AS "NEW" OUTPUT FILES
        PERFORM WRITE-FILES 6 TIMES.
        CLOSE FILE-A LOCK, FILE-B.
   * FILE-A IS CLOSED AND ENTERED IN THE DISK DIRECTORY
   * FILE-B IS CLOSED ("TEMPORARY"); NOT ENTERED IN THE DISK DIRECTORY
   * THE CURRENT RECORD POINTER IN FILE-B IS RESET TO RECORD 1
   * AT THIS POINT, THE FILES CONTAIN THE FOLLOWING RECORDS:
   *   FILE-A:  (EOF=6)  01  02  03  04  05  06
   *   FILE-B:  (EOF=6)  01  02  03  04  05  06
        OPEN OUTPUT FILE-A, FILE-B.
   * FILE-A IS OPENED AS A "NEW" OUTPUT FILE
   * FILE-B IS RE-OPENED, ALLOWING EXISTING RECORDS TO BE OVERWRITTEN
        PERFORM WRITE-FILES 4 TIMES.
        CLOSE FILE-A REMOVE, FILE-B LOCK.
   * BOTH FILES ARE CLOSED AND ENTERED IN THE DISK DIRECTORY
   * AT THIS POINT, THE FILES CONTAIN THE FOLLOWING RECORDS:
   *   FILE-A:  (EOF=4)  07  08  09  10
   *   FILE-B:  (EOF=6)  07  08  09  10  05  06
        OPEN EXTEND FILE-A, FILE-B.
   * THE EXISTING FILES ARE RE-OPENED OUTPUT, AND THE CURRENT RECORD
   * POINTERS ARE POSITIONED TO THE END OF THE FILES
        PERFORM WRITE-FILES 4 TIMES.
        CLOSE FILE-A REMOVE, FILE-B REMOVE.
   * BOTH FILES ARE CLOSED AND ENTERED IN THE DISK DIRECTORY
   * AT THIS POINT, THE FILES CONTAIN THE FOLLOWING RECORDS:
   *   FILE-A:  (EOF=8)  01  02  03  04  11  12  13  14
   *   FILE-B:  (EOF=10) 01  02  03  04  05  06  11  12  13  14
```

5.4.2. DIVIDE

The use of "MOD" and "REMAINDER" in the same DIVIDE verb is prohibited and a syntax error is generated.

5.4.3. HIGH-VALUES, LOW-VALUES, and MOVE ALL

The table below summarizes the actions currently taken by the compiler, and points out several inconsistencies and known problems. The first two columns represent the initialized values assigned by the "VALUE" clause in the DATA DIVISION and the remaining columns show the values after the "MOVE" statement.

PICTURE	XX	99	S99	99 CMP	S99 CMP
VA HIGH-VALUE	F9F9	F9F9	F9F9**	99	990*
VA LOW-VALUE	4040	4040	4040**	00	000**
MOVE HIGH-VALUE	F9F9	F9F9	C9F9	99	C99
MOVE LOW-VALUE	4040	4040	4040**	00	C00
MOVE ALL "9"	F9F9	F9F9	C9F9	99	C99
MOVE ALL "0"	F0F0	F0F0	C0F0	00	C00
MOVE ALL ∂FF∂	FFFF	FFFF	CFFF	FF	CFF
MOVE ALL ∂00∂	F0F0	F0F0	C0F0	00	C00
MOVE ZEROS	F0F0	F0F0	C0F0	00	C00
MOVE SPACES	4040	4040***	4040***	00***	C00***
MOVE ALL " "	4040	4040	4040**	00	C00

* Known Error (refer to paragraph 5.3.27)
** Inconsistent
*** Will be syntaxed in a future release

5.4.4. When the Compiler Produces No Listing

If the COBOL compiler comes to an abnormal end-of-job and no source listing is produced, there are certain steps that can be taken to find a temporary solution to the problem.

The first step is to determine the statement or statements in the COBOL source which cause the problem. The "LISTP" dollar option causes the compiler to list the source images in the initial pass. Errors detected in that pass are listed as they occur. As the source images are listed, a sequential number is printed on the left side of the listing. This number is used by the compiler to reference source images. In some errors that are printed, this number, called "TCARD" is also printed. This number can then be used to find the source statement that is in error. In a DUMP/ANALYZER listing of the compiler, "TCARD" appears as the first 24-bit item in the "NAME STACK". Passes other than the initial pass list other errors as they occur. The errors found using "LISTP" should be corrected, then the program should be recompiled. The "$ TIME" option displays the current phase of the compiler being executed and is most beneficial for debugging.

Whether or not a temporary solution is found, the problem (along with all materials necessary to reproduce it) should be submitted together with a B1700 Field Trouble Report (form #1909603) by the local Burroughs support personnel.

5.4.5. Compiler General Information and Optimization

In order to achieve better utilization of memory on small memory size systems, the COBOL compiler has been implemented in a multi-phase manner. In multi-phase compilation the original text is transformed to a more convenient form by passing it against a part of the compiler. The transformed text is then passed to the next phase via an intermediate work file. In this way the data is managed in a somewhat sequential manner, minimizing random overlaying. Code overlays are minimized in the same way because only a part of the code is invoked to transform all the data for the phase.

Two disadvantages of multi-phase compilation are:

a. If a large amount of memory is available at compile time, there is no way to combine phases.

b. A certain fixed overhead is required for opening and closing of the intermediate files, even for an extremely small source file.

The COBOL compiler is divided into ten phases, as follows:

a. "PARSE" - Initial Parsing. Merges source language inputs, creates or copies library files, creates report file, scans input and writes tokens to DNFILE and ALLDNOUT for further processing.

b. "DICT" - Dictionary Processing. Builds a dictionary of all declared data-names and procedure-names.

c. "DNQUAL" - Data-name Qualification Resolution.

d. "LQUAL" - Label (procedure-name) Qualification Resolution.

e. "MERGE" - Combine constant information about each data-name and procedure-name with a unique occurrence number.

f. "DATSYN" - DATA DIVISION syntax checking.

g. "EXPLODE" - Expand tokens in the PROCEDURE DIVISION to include all known attributes for that token.

h. "PROSYN" - PROCEDURE DIVISION syntax checking.

i. "CODEGEN" - Code generation.

j. "FIXUP" - Produces a codefile (the executable object code) and a listing.

When designing a compiler for many memory size configurations, it is important to effectively utilize additional memory if it is available. The COBOL compiler accomplishes this by managing the dynamic memory allocated at compile time.

A certain minimum dynamic memory is required for building lists and tables during the different phases. Associated with this minimum memory are various limits (for example, the number of data-names, procedure-names, and so forth). If a particular source program exceeds any of these limits, more space must be dedicated by increasing the dynamic memory and recompiling. An attempt has been made to make these restrictions "reasonable".

If additional dynamic memory is available, the compiler is designed to use that space for a significant speed gain. Some details regarding the use of dynamic memory may be found in the following paragraphs.

The dynamic memory allocated to the COBOL compiler is used for many things. For the purposes of this discussion, "text space" is defined as the space required to store a data-name, literal, or procedure-name. For example, the text space for the data-name "MASTER-FILE" is 11 bytes. Some of the uses of dynamic memory and the amounts required are as follows:

a. COPY .... REPLACING A BY B" pairs. Each data-name or literal requires its own text space, plus 9 bytes per pair.

b. Mnemonic names. Each name requires its own text space plus 8 bytes.

c. Data names. Each data-name declared requires 38 bits.

d. Procedure names. Each procedure-name declared requires 41 bits.

e. Condition names. Each "88" value list requires its own text space plus 7 bytes, plus an additional 5 bytes for each separate literal in the "VALUE" list.

f. MONITOR. Each data-name monitored requires 42 bits. Each procedure-name monitored requires its own text space plus 3 bytes.

g. Library files. Creating or copying library files requires 486 bytes.

h. Dollar options. "MERGE" and "NEW" each require 486 bytes. "LISTP" requires 326 bytes.

In order to minimize the intermediate file size, information about the original text is distributed to several files which are ordered to each other. The individual files are then processed without the need to copy extraneous information. When all of the transformations are complete, the files are merged.

The following list provides some general information about the files used and created by the compiler, such as internal name, external name, device type, phases in which used, access type, open type, which system drive directed to, and purpose:

a. NEWSOURCE "COBOLW/SOURCE" OPTIONAL DISK
Created by PARSE. Contains source card images to which patch cards have been applied. Sequential access, OUTPUT. Directed to system drive 1, if possible.

b. LIBRARY "COBOLW/LIBRARY" OPTIONAL DISK
Used by PARSE to create or read library files. Sequential access, INPUT and OUTPUT. Directed to system drive 1, if possible.

c. CARDS "CARDS" READER
Used by PARSE. Contains source card images or patch cards. Sequential access, INPUT.

d. SOURCE "COBOLW/SOURCE" OPTIONAL DISK
Used by PARSE. Contains source card images to which patches may be applied. Sequential access, INPUT.

e. REPORT "COBOLW/REPORT" DISK
Used by PARSE (OUTPUT) and FIXUP (INPUT). Contains all card images processed, including dollar cards, library cards, and patch cards. Used to print the listing. Sequential access. Directed to system drive 1, if possible.

f. DNFILEX "COBOLW/DNFILE" DISK
Used by DICT, DNQUAL, and LQUAL. Contains all picture strings and variables reduced to an occurrence number. Also used by MERGE and DATSYN to contain an entry for each unique picture string of the DATA DIVISION. Sequential access. Directed to system drive 0.

g. SEGFILE "COBOLW/SEGFILE" DISK
Used by MERGE, DATSYN, CODEGEN, and FIXUP. Contains edit masks, values to which the WORKING-STORAGE variables should be initialized, error or warning messages and various other information which is used to build the codefile and supply additional data for the listing. Random access. Directed to system drive 0.

h. LINE "LINE" PRINTER
Used by FIXUP for listing, or by any phase for compiler debugging output.

i. ADNFILE "COBOLW/ADNFILE" DISK
Used by MERGE, DATSYN, EXPLODE, and PROSYN. Contains merged tokens of ALLDNOUT and ADFILE. Sequential access. Directed to system drive 1, if possible.

j. ADFILE "COBOLW/ADFILE" DISK
Used by PARSE, DICT, DNQUAL, LQUAL, and MERGE to contain picture strings and all variable names isolated (for example, section-names, paragraph-names, and data-names). Also used by CODEGEN and FIXUP to contain explicit (procedure-names) and implicit (compiler-generated branch points) label attributes used for generating the correct branch addresses. Also used by EXPLODE and PROSYN to contain tokens which have been expanded to include all the known attributes for that token. Random access. Directed to system drive 1, if possible (note: this is a change from the Mark VI.1 release letter documentation).

k. ALLDNOUT "COBOLW/ALLDNOUT" DISK
Used by PARSE and MERGE to contain all constant information about each token processed. Also used by DATSYN and EXPLODE to contain an entry for each explicit data-name (excluding FILLER entries). Each entry contains attributes such as usage, address, length, number of subscripts required, and so forth. Also used by PROSYN and CODEGEN to contain tokens that have been syntax checked, rearranged, and simplified for code generation. Random access. Directed to system drive 1, if possible.

l. CODEFILE "COBOLW/CODEFILE" DISK
Created by FIXUP. Contains the object program according to MCP specifications. Not created if syntax errors are detected. Random access. Directed to system drive 0.

If the system being used for compilation does not have multiple system drives, those files which are directed to system drive 1 can be modified to reside on a user pack, if one is available, to achieve better disk utilization.

As a general rule, the most effective way to optimize COBOL compiles is to utilize dynamic memory to minimize disk accesses. For several phases where sufficient dynamic memory to maintain information in memory rather than on disk is allocated, files are loaded into tables and accessed, rather than randomly accessed on disk. As a result, compile times can be significantly reduced.

In order to determine whether the labels are being processed in memory or on disk, the dollar option card "TIME" can be used. This option causes compile statistics to be printed following the normal compilation output. The fields called EXPLICIT.LABELS and EXPLICIT.DATA.NAMES can be multiplied by the amount of memory required for each to determine the minimum amount of dynamic memory.

5.4.6. New and Changed Compiler Error Messages

(151) INVALID DESCRIPTION FOR ACTUAL KEY
(366) MERGE COMPILER ERROR, BAD PC.OCCUR, SUBMIT SOURCE AND TROUBLE REPORT

5.4.7. Introduction to the Indexed I-O Module

Function

The Indexed I-O Module provides the capability to access records of a mass storage file in either a RANDOM or SEQUENTIAL manner. Each record in an Indexed file is uniquely identified by the value of a key within that record.

Level Characteristics

The B 1800/B 1700 COBOL compiler utilizes the ANSI-74 syntax for the Indexed I-O Module, with the exception of the "DELETE" verb and the ALTERNATE KEYS option, which are not implemented, and the Burroughs extensions to the "AT END" and "INVALID KEY" clauses.

Language Concepts

Organization

A file whose "ORGANIZATION IS INDEXED" is a mass storage file in which data records are accessed by the value of a key. A record description includes a key data item which is associated with an index. The index provides a logical path to the data records according to the contents of a data item within each record, which is the record key for that index.

The data item named in the RECORD KEY clause of the FILE-CONTROL entry for a file is the "prime" record key for that file. For purposes of inserting and updating records in a file, each record is identified solely by the value of its prime record key. This value must, therefore, be unique and must not be changed when updating the record.

Access Modes

In the "SEQUENTIAL" access mode, the sequence in which records are accessed is the ascending order of the record key values.

In the "RANDOM" access mode, the sequence in which records are accessed is controlled by the programmer. The desired record is accessed by placing the value of its record key in a record key data item.

In the "DYNAMIC" access mode, the programmer may change at will from SEQUENTIAL to RANDOM access using appropriate forms of input-output statements.

Current Record Pointer

The current record pointer is a conceptual entity used in this document to facilitate specification of the next record to be accessed within a given file. The concept of the current record pointer has no meaning for a file opened in the OUTPUT mode. The setting of the current record pointer is affected only by the "OPEN", "START", and "READ" statements.

I-O Status

If the "FILE STATUS" clause is specified in a FILE-CONTROL entry, a value is placed into the specified two-character data item during the execution of an "OPEN", "CLOSE", "READ", "WRITE", "REWRITE", or "START" statement and before any applicable "USE" procedure is executed, to indicate to the COBOL program the status of that input-output operation.

Status Key 1

The leftmost character position of the FILE STATUS data item is known as Status Key 1 and is set to indicate one of the following conditions upon completion of the input-output operation:

0 Successful Completion
1 AT END
2 INVALID KEY
3 Permanent Error
9 Implementor Defined

The meanings of the above indications are as follows:

0 Successful Completion. The input-output statement was successfully executed.

1 AT END. The Format 1 "READ" statement was unsuccessfully executed as a result of an attempt to read a record when no next logical record exists in the file.

2 INVALID KEY. The input-output statement was unsuccessfully executed as a result of one of the following:

Sequence Error
Duplicate Key
No Record Found

3 Permanent Error. The input-output statement was unsuccessful as the result of an input-output error, such as Data Check, Parity Error, or a Transmission Error.

9 Implementor Defined. The input-output statement was unsuccessfully executed as a result of a condition that is specified by the implementor. This value is used only to indicate a condition not indicated by other defined values of Status Key 1, or by specified combinations of Status Key 1 and Status Key 2.

Status Key 2

The rightmost character position of the FILE STATUS data item is known as Status Key 2 and is used to further describe the results of the input-output operation. This character contains a value as follows:

a. If no further information is available concerning the input-output operation, then Status Key 2 contains a value of "0".

b. When Status Key 1 contains a value of "2" (indicating an "INVALID KEY" condition), Status Key 2 is used to designate the cause of that condition, as follows:

   1. A value of "1" in Status Key 2 indicates a sequence error for a sequentially-accessed Indexed file, meaning that the ascending sequence requirements of successive record key values have been violated (refer to the "WRITE" statement), or the prime record key value has been changed by the COBOL program between the successful execution of a "READ" statement and the execution of the next "REWRITE" statement for that file.

   2. A value of "2" in Status Key 2 indicates a duplicate key value, meaning that an attempt has been made to "WRITE" or "REWRITE" a record that would create a duplicate key in an indexed file.

   3. A value of "3" in Status Key 2 indicates no record found, meaning that an attempt has been made to access a record identified by a key, and that record does not exist in the file.

c. When Status Key 1 contains a value of "9" (indicating an implementor-defined condition), a value of "3" in Status Key 2 indicates that the record key value contained in the data file record is not equal to the record key value contained in the associated tag file record.

Valid Combinations of Status Keys 1 and 2

Allowable combinations for the value of Status Key 1 and Status Key 2 are shown in the following table. An "X" at an intersection indicates a valid combination.

Status Key 1	Status Key 2				
	No Further Information (0)	Sequence Error (1)	Duplicate Key (2)	No Record Found (3)	
Successful Completion (0)	X				
AT END (1)	X				
INVALID KEY (2)		X	X	X	
Permanent Error (3)	X				
Implementor Defined (9)				X	

The "INVALID KEY" Condition

An "INVALID KEY" condition can occur as a result of the execution of a "START", "READ", "WRITE", or "REWRITE" statement. For details on the causes of this condition, refer to the appropriate statement.

When the "INVALID KEY" condition is recognized, actions are taken in the following order:

a. A value is placed into the FILE STATUS data item, if specified for the file, to indicate an "INVALID KEY" condition (refer to I-O Status).

b. If the "INVALID KEY" phrase is specified in the statement causing the condition, control is transferred to the "INVALID KEY" statement. Any "USE" procedure specified for the file is not executed.

c. If the "INVALID KEY" phrase is not specified but a "USE" procedure is specified, either explicitly or implicitly, for the file, that procedure is executed.

When the "INVALID KEY" condition occurs, execution of the input-output statement which recognized the condition is unsuccessful and the file is not affected.

The "AT END" Condition

An "AT END" condition can occur as a result of execution of a "READ" statement. For details of the causes of this condition, refer to the "READ" statement.

General Implementation Information

Physical Files

As implemented, Indexed files consist of two physical files:

a. The data file contains records written to the file by the program and is maintained in the order in which the records are first written.

b. The "tag" file contains the record key value and the relative record number of its associated record in the data file. This file is ordered on record key value, and is read, written, and maintained transparent to the COBOL user by compiler-generated subroutines. These subroutines, or "Indexed Control System" (ICS), also provide for opening and closing of the tag file.

Tag Files

A tag file record contains a record key and a pointer to the relative location in the data file of the data record whose record key is the same value as that in the tag file record. The record key is an exact copy of the record key field in the data file record. Duplicate keys are not allowed. The tag file is blocked so that as many records as possible are placed into a 180-character block. If the record key is greater than 176 characters in length, the tag file is unblocked.

The internal and external names of the tag file are generated by the ICS by prefixing the word "TAG" to the internal and external names of the data file, as shown in the following examples:

Internal File Names		External File Names	
Data File	Tag File	Data File	Tag File
FILE1	TAGFILE1	A/B	TAGA/B
FILE2	TAGFILE2	FILE2	TAGFILE2
FILE3	TAGFILE3	CCC/A/B	CCC/TAGA/B
FILE4	TAGFILE4	A/ABCDEFGHI	TAGA/ABCDEFGHI
FILE5	TAGFILE5	A/B/	A/TAGB/

RPG and COBOL File Compatibility

In order to accommodate those users wishing to process Indexed files by both RPG and COBOL programs, the "RPGTAG" dollar option is available in the COBOL compiler. This option, when set, causes the compiler to generate code to process Indexed files using tag files with the same format and external names as if those tag files had been created by an RPG program. These files cannot, however, be used if the program is executed under File Security (unless run with a privileged usercode/password).

The difference in tag file format is due to the use by COBOL of an eight-digit pointer in the tag file records, while RPG uses a six-digit pointer. The larger pointer enables COBOL users to process larger files than RPG limitations allow.

The difference in the external names of the tag files is due to the prefixing by COBOL of the word "TAG" to the family-name of the data file to produce the tag file name, where RPG prefixes "TAG" to the sub-directory name. Prefixing the family-name rather than the sub-directory name enables COBOL to generate, in some cases, a single-file name for the tag file. This allows Indexed files to be processed under the File Security system if desired. If "RPGTAGS" had been specified in the COBOL program which created the tag files in the preceding example, the external tag file names would have been created as follows:

```
A/TAGB
FILE2/TAG
CCC/A/TAGB
A/TAGABCDEFG
A/B/TAG
```

Another point to consider if it is necessary to access the same Indexed files with both COBOL and RPG programs is that COBOL only permits alphanumeric keys.

Rough Tables

In order to minimize disk accesses when searching the tag file for record keys, the ICS, as a function of opening the tag file, creates a "rough table" in memory which logically breaks the tag file into partitions. This table consists of as many entries as will fit into the space provided (refer to the "VALUE OF" clause) or, by default, ten (10) entries. Each entry consists of the record key which occupies the last record in the partition. The number of records contained in a partition is determined by the number of records in the file divided by the number of table entries allowed.

A rough table is built for all opens of files with "ORGANIZATION INDEXED", except for those opened OUTPUT with SEQUENTIAL access, and those which are opened INPUT with SEQUENTIAL access for which there is no "START" statement contained in the program. As the

rough table is built, the entries are sequence-checked to insure that the binary search algorithm can function correctly. If the entries are not in sequence, the program is immediately terminated.

Searching the Tag File

A binary search method is used to locate the desired key between successive entries in the rough table. The partition boundaries are then calculated by multiplying these entry numbers by the number of tag records contained in the partitions. A binary search of the tag file between those boundaries is then done to find the required record. When the tag record is found, the relative record number is used to make the associated record from the data file available to the COBOL program. If the key is not found in the above manner and records have been added to the file since the last reordering of the tag file, the records which have been added to the tag file are serially searched for the required key.

In an effort to minimize the number of disk reads required to locate a particular record, the following optimizations have been used:

a. The required key is checked for equal to the key found in the rough table. If they are equal, the tag file record is read without requiring the binary search of that partition.

b. If the partition found in the rough table contains the end-of-file record, the required key is compared to the key of the end-of-file record. If the requested key is greater, the added records (if any) are searched, and the binary search of that partition is avoided.

b. When records are added, the highest and lowest keys added are saved. The required key is range-checked with those values. If the required key is outside that range, the serial search of the added records is unnecessary.

Adding Records to the Files

When the tag and data files are opened, their end-of-file pointers (which must be equal) are obtained. This end-of-file pointer (EOF-POINTER) is placed in NEW-EOF-POINTER. New records are appended to the end of each file by adding one to NEW-EOF-POINTER and using the resulting value as the actual key of both the tag and data files.

When the file is closed, or when a "START", or "READ ... NEXT" statement is executed and records have been added to the file, the tag file is sorted (does not apply to SEQUENTIAL access).

ENVIRONMENT DIVISION In The Indexed I-O Module

The FILE-CONTROL Paragraph

Function

The FILE-CONTROL entry names a file and may specify other file-related information. This description of the FILE-CONTROL entry only describes those clauses which are limited to the Indexed I-O Module.

General Format

```
SELECT file-name
------

          (   DISK    )
          (   ----    )
ASSIGN TO (           )
------    ( DISKPACK  )
          ( --------  )

ORGANIZATION IS INDEXED
------------    -------

[                  (SEQUENTIAL) ]
[                  (-----------) ]
[ ACCESS MODE IS   (RANDOM     ) ]
[ ------           (------     ) ]
[                  (DYNAMIC    ) ]
[                  (-------    ) ]

RECORD KEY IS data-name-1
------

[ FILE STATUS IS data-name-2]
       ------
```

Syntax Rules

a. The "SELECT" clause must be specified first in the FILE-CONTROL entry. The clauses which follow the "SELECT" clause may appear in any order.

b. Each file described in the DATA DIVISION must be named once and only once as file-name in the FILE-CONTROL paragraph. Each file specified in the FILE-CONTROL entry must have a File Description entry in the DATA DIVISION.

c. If the "ACCESS" clause is not specified, "ACCESS MODE IS SEQUENTIAL" is implied.

d. Data-name-2 must be defined in the DATA DIVISION as a two-character data item of the category alphanumeric and must not be defined in the FILE SECTION.

e. Data-name-1 and data-name-2 may be qualified.

f. The data item referenced by data-name-1 must be defined as a data item of the category alphanumeric within a Record Description entry associated with that file-name.

g. Data-name-1 cannot describe an item whose size is variable (refer to the "OCCURS" clause).

General Rules

a. The "ASSIGN" clause specifies the association of the file referenced by file-name to a mass storage medium.

b. The "ORGANIZATION" clause specifies the logical structure of a file. The file organization is established at the time a file is created and cannot subsequently be changed.

c. When "ACCESS MODE IS SEQUENTIAL", records in the file are accessed in the sequence dictated by the file organization. For Indexed files, this sequence is the order of ascending record key values within a given key of reference.

d. If "ACCESS MODE IS RANDOM", the value of the record key data item indicates the record to be accessed.

e. When "ACCESS MODE IS DYNAMIC", records in the file may be accessed sequentially and/or randomly (refer to General Rules c and d).

f. When the "FILE STATUS" clause is specified, a value is moved by the ICS into the data item specified by data-name-2 after the execution of any statement that references that file either explicitly or implicitly. This value indicates the status of execution of the statement.

g. The "RECORD KEY" clause specifies the record key for the file. The values of the record key must be unique among records of the file. The record key provides an access path to records in an Indexed file.

h. The Data Description of data-name-1, as well as its relative location within a record, must be the same as that used when the file was created.

DATA DIVISION in the Indexed I-O Module

The File Description - Entry Skeleton

Function

The File Description furnishes information concerning the physical structure, identification, and record names pertaining to a given file. This discussion of the File Description entry (FD), provides details only of those clauses which have special meaning in the Indexed I-O Module.

General Format

```
FD  file-name
--

[(VALUE)    (IDENTIFICATION)          (data-name-1)]
[(-----) OF (--------------)  IS      (             )]
[(VA    )   (ID              )       (literal-1    )]
[(--    )   (--              )                     ]

   [ VALUE OF CORE-INDEX IS integer-1 CHARACTERS ]
   [ -----    ----------                         ]
```

Syntax Rules

a. The level indicator FD identifies the beginning of a File Description and must precede the file-name.

b. The clauses which follow the name of the file are optional in many cases, and their order of appearance is immaterial.

c. One or more Record Description entries must follow the File Description entry.

The "VALUE OF" Clause

Function

The "VALUE OF" clause provides for the identification of a data file and its related tag file, as well as allowing specification of the amount of memory to be allocated to the rough table.

General Format

```
------------------------------------------------------------
|                                                          |
|  (VALUE)    (IDENTIFICATION)        (data-name-1)        |
|  (-----) OF (--------------)   IS   (           )        |
|  (VA    )   (ID            )        (literal-1  )        |
|  (--    )   (--            )                             |
|                                                          |
|    [ VALUE OF CORE-INDEX IS integer-1 CHARACTERS ]       |
|    [ -----    ----------                         ]       |
|                                                          |
------------------------------------------------------------
```

Syntax Rules

Data-name-1 should be qualified when necessary, but cannot be subscripted or indexed, nor can it be an item described with the "USAGE IS INDEX" clause.

General Rules

a. For the IDENTIFICATION option:

1. For an input file, the MCP checks to see if a data file whose name is equal to the value of literal-1 or data-name-1, whichever has been specified, as well as its associated tag file are present in the disk directory.

2. For an output file, at the appropriate time a data file whose name is made equal to the value of literal-1 or data-name-1, whichever has been specified, as well as its associated tag file are inserted into the disk directory (refer to Tag Files).

b. For the "CORE-INDEX" option, the number of characters specified by integer-1 is allocated for storage of the rough table. If this option is omitted, the number of characters required to contain ten (10) entries is allocated (refer to Rough Tables).

PROCEDURE DIVISION in the Indexed I-O Module

The CLOSE Statement

Function

The "CLOSE" statement terminates the processing of a file and specifies the disposition of the file.

General Format

```
                                                    [      (LOCK      ) ]
                                                    [      (----      ) ]
CLOSE file-name-1                                   [WITH  (RELEASE   ) ]
-----                                               [      (--------  ) ]
                                                    [      (REMOVE    ) ]
                                                    [      (------    ) ]
                                                    [      (PURGE     ) ]
                                                    [      (-----     ) ]

    [                                               [      (LOCK      ) ]       ]
    [                                               [      (----      ) ]       ]
    [ file-name-2                                   [WITH  (RELEASE   ) ]  ...  ]
    [                                               [      (--------  ) ]       ]
    [                                               [      (REMOVE    ) ]       ]
    [                                               [      (------    ) ]       ]
    [                                               [      (PURGE     ) ]       ]
    [                                               [      (-----     ) ]       ]
```

Syntax Rules

The files referenced in the "CLOSE" statement need not all have the same ORGANIZATION or ACCESS.

General Rules

a. A "CLOSE" statement may only be executed for a file in the OPEN mode.

b. Indexed files are labeled mass storage files. The results of executing each type of "CLOSE" are summarized in the following table:

CLOSE Statement Format	File Disposition
CLOSE	1, 3
CLOSE WITH LOCK	1, 2, 4
CLOSE WITH RELEASE	1, 2, 4
CLOSE WITH REMOVE	1, 4, 5
CLOSE WITH PURGE	1, 6

The definitions of the symbols in the table are as follows:

1. File CLOSE. The "CLOSE" operation marks the logical file as closed.

2. File Lock. The physical file is made a permanent file and is entered into the MCP disk directory.

3. File Retention. The association between the logical file and the physical file is retained. A new file is not entered in the disk directory, and is lost at end-of-job (or overwritten following a subsequent "OPEN OUTPUT"), if not closed with "LOCK", "RELEASE", or "REMOVE".

4. File Release. The association between the logical file and the physical file is severed. The areas of memory allocated for buffers are released to the system.

5. File Remove. This option causes the MCP to remove from the disk directory a file which has the same external name as the file being closed. This action takes place prior to entering the external name in the disk directory.

6. File Purge. If the file is a permanent file, the file is removed from the disk directory, and the storage area occupied by the file is released as available to the MCP.

c. If a file is in the OPEN mode when a "STOP RUN" statement is executed or when an abnormal termination occurs, the action taken is to close the file as though a simple "CLOSE" statement had been executed.

d. If a "CLOSE" statement has been executed for a file, no other statement can be executed that references that file, either explicitly or implicitly, unless an intervening "OPEN" statement for that file is executed.

The execution of a "CLOSE" statement has no effect upon the contents or the availability of the file's record area.

The OPEN Statement

Function

The "OPEN" statement initiates the processing of files. It also performs checking and/or writing of labels and other input-output operations.

General Format

```
                                                                    
        (INPUT  file-name-1 [WITH LOCK [ACCESS]]                 )  
        (-----                ----  ------                       )  
        (                                                        )  
        (        [ file-name-2 [WITH LOCK [ACCESS]] ] ...        )  
        (                              ----  ------              )  
        (                                                        )  
  OPEN  (OUTPUT file-name-3 [ file-name-4] ...                   )  
  ----  (------                                                 )  
        (                                                        )  
        (I-O file-name-5 [WITH LOCK [ACCESS]]                    )  
        (---               ----  ------                         )  
        (                                                        )  
        (       [ file-name-6 [WITH LOCK [ACCESS]]  ] ...        )  
        (                     ----  ------                       )  
                                                                    
```

Syntax Rules

The files referenced in the "OPEN" statement need not all have the same ORGANIZATION or ACCESS.

General Rules

a. The successful execution of an "OPEN" statement determines the availability of the file and results in the file being in an OPEN mode.

b. The execution of an "OPEN" statement does not affect either the contents or availability of the file's record area.

c. When a given file is not in an OPEN mode, no statement (except for a "SORT" or "MERGE" statement with the "USING" or "GIVING" phrases) can be executed that references the file, either explicitly or implicitly.

d. A file may be opened with the "INPUT", "OUTPUT", and "I-O" phrases in the same program. Following the initial execution of an "OPEN" statement for a file, each subsequent "OPEN" statement executed for the same file must be preceded by the execution of a "CLOSE" statement for the file.

e. An "OPEN" statement must be successfully executed prior to the execution of any of the permissible input-output statements. In the following table, an "X" indicates that the specified statement, used in the file access mode given for that row, may be used with the OPEN mode given at the top of the column.

File Access Mode	Statement	OPEN Mode: INPUT	OUTPUT	INPUT-OUTPUT
SEQUENTIAL	READ	X		X
	WRITE		X	
	REWRITE			X
	START	X		X
RANDOM	READ	X		X
	WRITE		X	X
	REWRITE			X
	START			
DYNAMIC	READ	X		X
	WRITE		X	X
	REWRITE			X
	START	X		X

f. Execution of the "OPEN" statement does not obtain or release the first data record.

g. The File Description entry for file-name-1, file-name-2, file-name-5, or file-name-6 must be equivalent to that used when the file was created.

h. For files being opened with the "INPUT" or "I-O" phrase, the "OPEN" statement sets the current record pointer to the first record currently existing within the file. For Indexed files, the "prime" record key is established as the key of reference and is used to determine the first record to be accessed. If no records exist in the file, the current record pointer is set such that the next executed Format 1 "READ" statement for the file results in an "AT END" condition.

i. The "I-O" phrase permits the opening of a file for both INPUT and OUTPUT operations. Since this phrase implies the existence of the file, it cannot be used if the file is being initially created.

j. Upon successful execution of an "OPEN" statement with the "OUTPUT" phrase specified, a file is created. At that time the associated file contains no data records.

k. When an "OPEN WITH LOCK" (without "ACCESS") is executed, the following occurs:

   1. If the specified file is already in an OPEN mode by another program or by a different file-name within this program, the program is suspended awaiting exclusive availability of the file.

   2. If the file is not currently in an OPEN mode by any program, the file is opened.

l. When an "OPEN WITH LOCK ACCESS" is executed, the program is suspended if any of the following conditions exist:

   1. The file is currently OPEN WITH LOCK.

   2. The file is currently OPEN INPUT-OUTPUT.

   3. This OPEN is I-O and the file is currently OPEN INPUT LOCK ACCESS.

   The program remains suspended until none of these conditions remain, or until the program is terminated. In any other case, the "OPEN WITH LOCK ACCESS" causes the file to be opened.

The READ Statement

Function

For SEQUENTIAL access, the "READ" statement makes available the next logical record from a file. For RANDOM access, the "READ" statement makes available a specified record from a mass storage file.

General Format

Format 1:

```
READ file-name [NEXT] RECORD [INTO identifier]
     [ AT END statement-1 [ELSE statement-2]]
```

Format 2:

```
READ file-name RECORD [INTO identifier]
     [ KEY IS data-name]
     [ INVALID KEY statement-1 [ELSE statement-2]]
```

Syntax Rules

a. The storage area associated with identifier and the storage area which is the record area associated with file-name must not be the same.

b. Data-name must be the name of a data item specified as a record key associated with file-name.

c. Data-name may be qualified.

d. Format 1 must be used for all files in SEQUENTIAL access mode.

e. The "NEXT" phrase must be specified for files in DYNAMIC access mode, when records are to be retrieved sequentially.

f. Format 2 is used for files in RANDOM access mode or for files in DYNAMIC access mode when records are to be retrieved randomly.

g. The "INVALID KEY" phrase or the "AT END" phrase must be specified if no applicable "USE" procedure is specified for file-name.

General Rules

a. The associated file must be OPEN in the INPUT or I-O mode at the time this statement is executed (refer to the OPEN Statement).

b. The record to be made available by a Format 1 "READ" statement is determined as follows:

   1. The record pointed to by the current record pointer is made available, provided that the current record pointer was positioned by a "START" or "OPEN" statement.

   2. If the current record pointer was positioned by the execution of a previous "READ" statement, the current record pointer is updated to point to the next existing record in the file and then that record is made available.

c. The execution of the "READ" statement causes the value of the File Status data item, if any, associated with file-name to be updated (refer to I-O Status).

d. Regardless of the method used to overlap access time with processing time, the concept of the "READ" statement is unchanged in that a record is available to the object program prior to the execution of any statement following the "READ" statement.

e. When the logical records of a file are described with more than one Record Description, these records automatically share the same storage area; this is equivalent to an implicit redefinition of the area.

f. If the "INTO" phrase is specified, the record being read is moved from the record area to the area specified by identifier according to the rules specified for the "MOVE" statement (without the "CORRESPONDING" phrase) with the sending area considered to be a group item whose size is equal to the maximum record size for the file. The implied move does not occur if the execution of the "READ"

statement was unsuccessful. Any subscripting or indexing associated with identifier is evaluated after the record has been read and immediately before it is moved to the data item.

g. When the "INTO" phrase is used, the record being read is available in both the input record area and the data area associated with identifier.

h. If, at the time of execution of a Format 1 "READ" statement, the position of the current record pointer for the file is undefined, the execution of that "READ" statement is unsuccessful.

i. If, at the time of the execution of a Format 1 "READ" statement, no next logical record exists in the file, the "AT END" condition occurs, and the execution of the "READ" statement is unsuccessful (refer to I-O Status).

j. When the "AT END" condition is recognized the following actions are taken in the specified order:

   1. A value is placed into the File Status data item, if specified for this file, to indicate an "AT END" condition (refer to I-O Status).

   2. If the "AT END" phrase is specified in the statement causing the condition, control is transferred to the "AT END" statement-1. Any "USE" procedure specified for the file is not executed.

   3. If the "AT END" phrase is not specified, then a "USE" procedure must be specified, either explicitly or implicitly, for the file, and that procedure is executed.

   When the "AT END" condition occurs, execution of the input-output statement which caused the condition is unsuccessful.

k. Following the unsuccessful execution of any "READ" statement, the contents of the associated record area and the position of the current record pointer are undefined. For Indexed files, the key of reference is also undefined.

l. When the "AT END" condition has been recognized, a Format 1 "READ" statement for the file must not be executed without first executing one of the following:

   1. A successful "CLOSE" statement followed by the execution of a successful "OPEN" statement for the file.

   2. A successful "START" statement for the file.

3. A successful Format 2 "READ" statement for the file.

m. For a file for which DYNAMIC access mode is specified, a Format 1 "READ" statement with the "NEXT" phrase specified causes the next logical record to be retrieved from that file as described in General Rule b.

n. For an Indexed file, if the "KEY" phrase is specified in a Format 2 "READ" statement, data-name is established as the key of reference for this retrieval. If the DYNAMIC access mode is specified, this key of reference is also used for retrievals by any subsequent executions of Format 1 "READ" statements for the file until a different key of reference is established.

o. If the "KEY" phrase is not specified in a Format 2 "READ" statement, the "prime" record key is established as the key of reference for this retrieval. If the DYNAMIC access mode is specified, this key of reference is also used for retrievals by any subsequent executions of Format 1 "READ" statements for the file until a different key of reference is established.

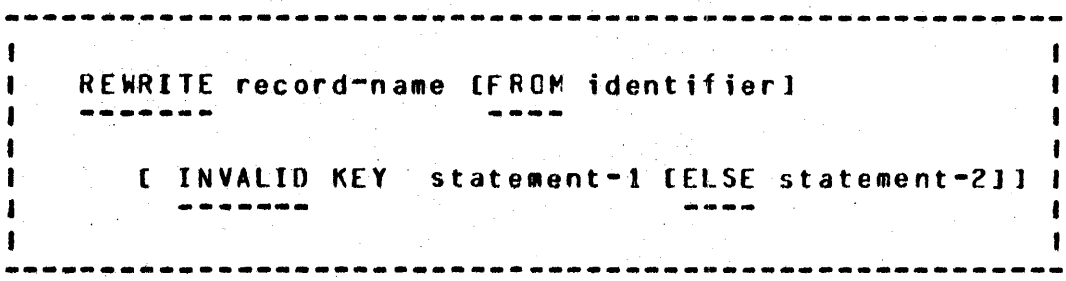
p. Execution of a Format 2 "READ" statement causes the value of the key of reference to be compared with the value contained in the corresponding data item of the stored records in the file, until a record having an equal value is found. The current pointer is positioned to this record, which is then made available. If no record can be so identified, the "INVALID KEY" condition exists and execution of the "READ" statement is unsuccessful (refer to the "INVALID KEY" Condition).

The REWRITE Statement

Function

The "REWRITE" statement logically replaces a record existing in a mass storage file.

General Format



Syntax Rules

a. Record-name and identifier must not refer to the same storage area.

b. Record-name is the name of a logical record in the FILE SECTION of the DATA DIVISION and may be qualified.

c. The "INVALID KEY" phrase must be specified in the "REWRITE" statement for files for which an appropriate "USE" procedure is not specified.

General Rules

a. The file associated with record-name must be OPEN in the I-O mode at the time of execution of this statement (refer to the OPEN Statement).

b. For files in the SEQUENTIAL access mode, the last input-output statement executed for the associated file prior to the execution of the "REWRITE" statement must have been a successfully-executed "READ" statement. The record that was accessed by the "READ" statement is logically replaced.

c. The number of character positions in the record referenced by record-name must be equal to the number of character positions in the record being replaced.

d. The logical record released by successful execution of the "REWRITE" statement is no longer available in the record area unless the associated file is named in a "SAME RECORD AREA" phrase, in which case the logical record is available to the program as a record of other files.

e. The execution of a "REWRITE" statement with the "FROM" phrase is equivalent to the execution of:

   MOVE identifier TO record-name

   followed by the execution of the same "REWRITE" statement without the "FROM" phrase. The contents of the record area prior to the execution of the implicit MOVE statement have no effect on the execution of the "REWRITE" statement.

f. The current record pointer is not affected by the execution of a "REWRITE" statement.

g. The execution of the "REWRITE" statement causes the value of the File Status data item, if any, associated with the file to be updated (refer to I-O Status).

h. For a file in the SEQUENTIAL mode, the record to be replaced is specified by the value contained in the prime record key. When the "REWRITE" statement is executed, the value contained in the prime record key data item of the record to be replaced must be equal to the value of the prime record key of the last record read from this file.

i. For a file in the RANDOM or DYNAMIC access mode, the record to be replaced is specified by the prime record key data item.

j. The "INVALID KEY" condition exists when:

   1. The access mode is SEQUENTIAL and the value contained in the prime record key data item of the record to be replaced is not equal to the value of the prime record key of the last record read from this file, or

   2. The value contained in the prime record key data item does not equal that of any record stored in the file.

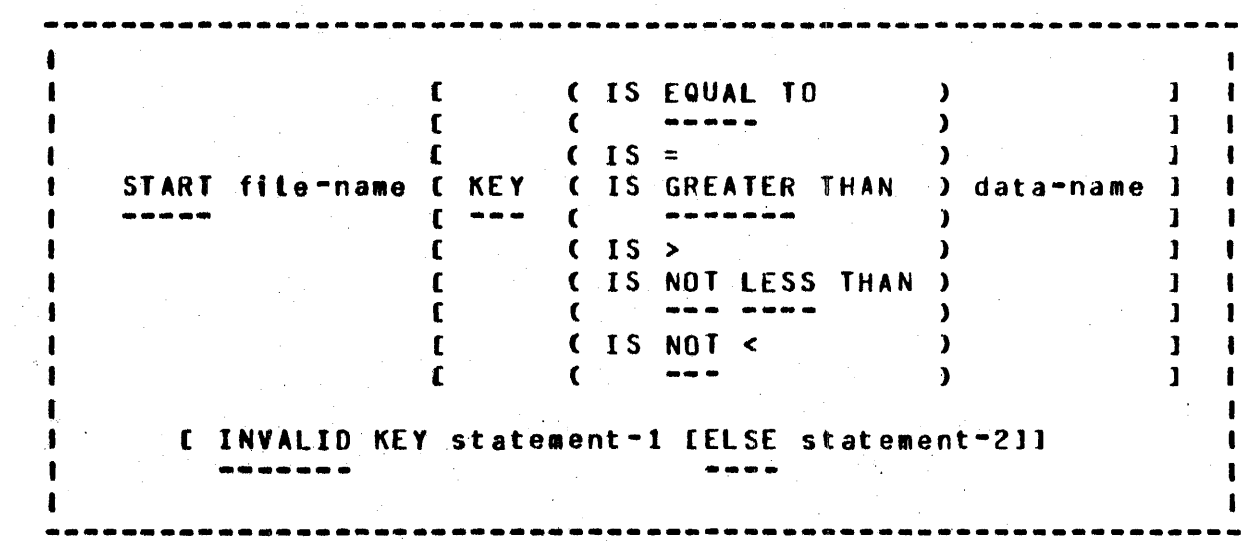
   The updating operation does not take place and the data in the record area is unaffected (refer to the "INVALID KEY" Condition).

The START Statement

Function

The "START" statement provides a basis for logical positioning within an indexed file, for subsequent sequential retrieval of records.

General Format



NOTE

The required relational characters ">", "<", and "=" are not underlined in the syntax diagram above to avoid confusion with other symbols such as "greater than or equal to".

Syntax Rules

a. File-name must be the name of an Indexed file.

b. File-name must be the name of a file with SEQUENTIAL or DYNAMIC access.

c. Data-name may be qualified.

d. The "INVALID KEY" phrase must be specified if no applicable "USE" procedure is specified for file-name.

e. If the "KEY" phrase is specified, data-name must reference the data item specified as the record key associated with file-name.

General Rules

a. File-name must be OPEN in the INPUT or I-O mode at the time that the "START" statement is executed (refer to the OPEN Statement).

b. If the "KEY" phrase is not specified, the relational operator "IS EQUAL TO" is implied.

c. The type of comparison specified by the relational operator in the "KEY" phrase occurs between the key associated with a record in the file referenced by file-name and a data item as specified in Syntax Rule e. The results of that comparison are as follows:

   1. The current record pointer is positioned to the first logical record currently existing in the file whose key satisfies the comparison.

   2. If the comparison is not satisfied by any record in the file, an "INVALID KEY" condition exists, the execution of the "START" statement is unsuccessful, and the position of the current record pointer is undefined (refer to the "INVALID KEY" Condition).

d. The execution of the "START" statement causes the value of the File Status data item, if any, associated with file-name to be updated (refer to I-C Status).

e. If the "KEY" phrase is not specified, the comparison described in General Rule c uses the data item referenced in the "RECORD KEY" phrase associated with file-name.

f. Upon completion of the successful execution of the "START" statement, a key of reference is established and is used in subsequent Format 1 "READ" statements (refer to the READ statement).

g. If the execution of the "START" statement is not successful, the key of reference is undefined.

The USE Statement

Function

The "USE" statement specifies procedures for input-output error or exception handling that are in addition to the standard procedures provided by the MCP.

General Format

```
                                  (EXCEPTION)
USE AFTER STANDARD                (---------)  PROCEDURE ON
--- -----                         (ERROR    )  ---------
                                  (-----    )

        ( file-name-1 [ file-name-2 ] ... )
        (                                 )
        ( INPUT                           )
        ( -----                           )
        ( OUTPUT                          )
        ( ------                          )
        ( I-O                             )
        ( ---                             )
```

Syntax Rules

a. A "USE" statement, when present, must immediately follow a section header in the DECLARATIVES SECTION and must be followed by a period followed by a space. The remainder of the section must consist of one or more procedural paragraphs that define the procedures to be used.

b. The "USE" statement itself is never executed; it merely defines the conditions calling for the execution of the "USE" procedures.

c. The same file-name can appear in a different specific arrangement of the format. Appearance of a file-name in a "USE" statement must not cause the simultaneous request for execution of more than one "USE" procedure.

d. The words "ERROR" and "EXCEPTION" are synonymous and may be used interchangeably.

e. The files implicitly or explicitly referenced in a "USE" statement need not all have the same ORGANIZATION or ACCESS.

General Rules

a. The designated procedures are executed by the ICS after completing the standard input-output error routine, or upon recognition of the "INVALID KEY" or "AT END" conditions, when the "INVALID KEY" phrase or "AT END" phrase, respectively, has not been specified in the input-output statement.

b. After execution of a "USE" procedure, control is returned to the invoking routine.

c. Within a "USE" procedure, there must not be any reference to any non-declarative procedures. Conversely, in the non-declarative portion there must be no reference to procedure-names that appear in the declarative portion, except that "PERFORM" statements may refer to a "USE" procedure or to the paragraphs contained within the "USE" procedure.

d. Within a "USE" procedure, there must not be the execution of any statement that would cause the execution of a "USE" procedure that had previously been invoked and had not yet returned control to the invoking routine.

The WRITE Statement

Function

The "WRITE" statement releases a logical record for an output or input-output file.

General Format

```
WRITE record-name [ FROM identifier ]

    [ INVALID KEY  statement-1 [ELSE statement-2]]
```

Syntax Rules

a. Record-name and identifier must not reference the same storage area.

b. The record-name is the name of a logical record in the FILE SECTION of the DATA DIVISION and may be qualified.

c. The "INVALID KEY" phrase must be specified if an applicable "USE" procedure is not specified for the associated file.

General Rules

a. The associated file must be OPEN in the OUTPUT or I-O mode at the time of the execution of this statement (refer to the OPEN Statement).

b. The logical record released by successful execution of the "WRITE" statement is no longer available in the record area unless the associated file is named in a "SAME RECORD AREA" clause, in which case the logical record is also available to the program as a record of other files.

c. The execution of the "WRITE" statement with the "FROM" phrase is equivalent to the execution of:

   MOVE identifier TO record-name

   followed by the execution of the same "WRITE" statement without the "FROM" phrase. The contents of the record area prior to the execution of the implicit MOVE statement have no effect on the execution of the "WRITE" statement.

After execution of the "WRITE" statement is complete, the information in the area referenced by identifier is available, even though the information in the area referenced by record-name may not be (refer to General Rule b).

d. The current record pointer is unaffected by the execution of a "WRITE" statement.

e. The execution of the "WRITE" statement causes the value of the File Status data item, if any, associated with the file to be updated (refer to I-O Status).

f. The maximum record size for a file is established at the time the file is created and must not subsequently be changed.

g. The number of character positions on a mass storage device required to store a logical record in a file may or may not be equal to the number of character positions defined by the logical description of that record in the program.

h. The execution of the "WRITE" statement releases a logical record to the MCP.

i. Execution of the "WRITE" statement causes the contents of the record area to be released. The ICS utilizes the contents of the record key in such a way that subsequent access to that record may be made based upon the value of that specified record key.

j. The value of the prime record key must be unique within the records in the file.

k. The data item specified as the prime record key must be set by the program to the desired value prior to the execution of the "WRITE" statement (refer to General Rule c).

l. If SEQUENTIAL access mode is specified, records must be released to the ICS in ascending order of prime record key values.

m. If RANDOM or DYNAMIC access mode is specified, records may be released to the ICS in any program-specified order.

n. The "INVALID KEY" condition exists under the following circumstances:

   1. When SEQUENTIAL access mode is specified for a file opened in the OUTPUT mode, and the value of the prime record key is not greater than the value of the prime record key of the previous record, or

2. When the file is opened in the OUTPUT or I-O mode, and the value of the prime record key is equal to the value of a prime record key of a record already existing in the file.

o. When the "INVALID KEY" condition is recognized, the execution of the "WRITE" statement is unsuccessful, the contents of the record area are unaffected and the File Status data item, if any, associated with file-name of the associated file is set to a value indicating the cause of the condition. Execution of the program proceeds according to the rules stated in the paragraph on the "INVALID KEY" condition (refer also to I-O Status).

5.4.8. Use of the ISAM Attributes

When a disk file is first created (OPEN OUTPUT ... CLOSE LOCK), four attributes are fixed and cannot be changed until the file is recreated. These are:

RECORD.SIZE (RSZ)
RECORDS.BLOCK (R.B)
BLOCKS.AREA (B.A)
AREAS (ARE)

The maximum number of records that can be contained in the file can be calculated as:

Maximum Records = R.B * B.A * ARE

All of these attributes can be specified by the programmer via the "RECORD CONTAINS", "BLOCK CONTAINS", and "FILE CONTAINS" clauses. For ISAM files, the compiler generates two files for each ISAM "SELECT" clause (the data file and the tag file). The programmer does not have any direct control over the attributes of the tag file. The compiler sets the tag file RECORD.SIZE to the size of the record key plus either 3 bytes (if $ RPGTAGS is specified) or 4 bytes. The RECORDS.BLOCK is calculated to get a block size of 180 bytes or less. The AREAS is set the same as that of the data file. Effective with the Mark VII.0 compiler, the BLOCKS.AREA is set to insure that the tag file can hold as many records as the data file. For example:

T-B.A=(D-R.B*D-B.A)/T-R.B+(if any remainder then 1 else 0)

where "T-" means "tag file" and "D-" means "data file". Extreme caution should be used when changing any of these attributes via the "MODIFY" or "FILE" statements. If the data file is changed, the tag file must be changed correctly.

5.4.9. ISAM File Recovery

When a record is added to an ISAM file, the compiler must generate code to do two WRITEs, one to the tag file and one to the data file. This immediately brings up some potential problems in the event of abnormal end-of-job.

Currently the code generated by a RANDOM WRITE is as follows:

a. WRITE the tag record. Do not request any error reporting. This means that any error on the WRITE (Parity, Incomplete I/O, Full File) causes the MCP to terminate the progam immediately.

b. WRITE the data record. Request error reporting. In the event of an error, the program takes the INVALID KEY branch or the USE routine.

The above code sequence could result in the tag file having a record that points to a non-existent record in the data file. This problem will be fixed in a future release.

In the WRITE of the data file, the error reporting requested and action taken depends on the presence of an INVALID KEY statement, a USE routine, or both:

a. INVALID KEY - Report on EOF (Full File). Return a status of 10 (should be 24) and take the INVALID KEY branch.

b. USE Routine - Report on Parity. Return a status key of 10 (should be 30) and enter the USE routine.

c. INVALID KEY and USE Routine - Report on EOF or Parity. If Parity, return a status key of 30 and enter the use routine. If EOF, return a status key of 10 (should be 24) and take the INVALID KEY branch.

The erroneous status key values will be corrected in a future release.

If an error occurs and reporting of that error has not been requested, the MCP terminates the program with the following message:

NO PROVISION FOR <error> ON FILE <file-name>

If the failure occurred between the WRITE of the tag record and the WRITE of the data record, the EOF.POINTERs on the two files are different. Two choices are available at this point: either reload both files or copy the larger file (disk-to-disk) leaving out the extra record (for example, use DMPALL with the INCLUDE <EOF.POINTER - 1> option. If the EOF.POINTERs are the same, the only problem is that the tag file has not been sorted. It is therefore possible to sort the tag file (using the SORT utility program) and continue processing.

It should be noted that if corruption occurs and no recovery is effected, the next program to use the file may not detect the error. The only check made is during the building of the rough table (ACCESS MODE IS RANDOM or START is used). The records read in building the rough table are sequenced-checked and a fatal error occurs in the event of a sequence error. However, if the few records read by the "build rough table" routine are in order, or no rough table is built, then the program proceeds. In this case, records that are present may not be found and duplicates may be introduced.

5.4.10. Efficient Use of ISAM Constructs.

COBOL is required, by ANSI-74 syntax, to check for duplicates at the time of the WRITE. The compiler generates code so that before each RANDOM WRITE the file is searched for duplicates. If no duplicate is found the record is written, but if the record already exists in the file, the INVALID KEY branch or USE routine is taken with the status key set to 22.

As an example, assume a simple file maintenance program that accepts "add", "change", and "delete" records. In COBOL the efficient way to handle this case would be to do a "keyed" read only if the record type is a "change" or "delete". If the record type is an "add", then do not do a READ; do a WRITE with an INVALID KEY statement and a STATUS KEY clause. If the record is a duplicate, the INVALID KEY branch is taken and the status key is set to a 22. This allows only one search of the file (instead of two) to be performed for each "add"-type record.

COBOL has two methods for creating an ISAM file; either Output ordered (sequential) or Random. It is not possible to sequentially write an ISAM file with the records unordered. Unless care is taken, this may result in programs that run extremely slowly, as described in the following examples.

The first example deals with a "load" of an ISAM file from an unordered source. If the file is opened OUTPUT with an access mode of RANDOM, the program runs very slowly. As each record is written the ISAM routines must check for a duplicate record in the file (note: an optimization has been incorporated in the ISAM routines where the check for duplicates is not done if the record to be added is greater than or less than any record previously added). If the file is being written in a random manner, before the nth record is written, the n-1 previous records must be read. In the worst case, the addition of n records can result in 2n writes (tag and data) and the summation of 2n-m where m varies from 2 to n reads. For example, the addition of 6 records could take 12 writes and 4+3+2+1 reads. This can obviously be a very slow process for large files. For large files, it is much better to first sort the input file and then do sequential writes.

The second example involves removing certain records, flagged as "to be deleted", from an existing ISAM file. The best method of handling this is to open the existing ISAM file INPUT SEQUENTIAL and open a new ISAM file OUTPUT SEQUENTIAL, then READ and WRITE sequentially. The READs may cause some disk arm movement if the data records are random, but the WRITEs will be very fast. This method gives the added advantage of ordering the data records in the new file, providing faster access in all future sequential access cases.

5.4.11. ISAM Subroutine Segmentation

Effective with Mark VII.0, the COBOL compiler automatically segments the ISAM-generated file-handling subroutines. It is necessary to recompile existing ISAM programs in order to have segmented subroutines, but old programs need not be recompiled if segmentation is not important.

The segmented subroutines start in the first segment higher than the highest user segment. For example, if a program has user segments "0, 1, 2, and 3" and two ISAM files, then the ISAM subroutines are in segments 4 and 5. If the program also has exponentiation, then the "exponentiation" subroutines are located in segments 6 and 7 (the two highest segments generated).

5.4.12. Specifying the "VALUE OF CORE-INDEX"

The specification of "integer-1" in the "VALUE OF CORE-INDEX" option is in "bytes". If the value specified is too small, then disk activity increases and throughput decreases. If the value specified is too large, then memory is wasted. There are several calculations necessary before an arbitrary value should be assigned. These calculations are in the format shown in the following formulas:

```
Rough table entries = CORE-INDEX / RECORD.KEY.SIZE

TAG.RECORD.SIZE = RECORD.KEY.SIZE + (if $ RPGTAGS then 3 else 4)

TAG.RECORDS.BLOCK = 180 / TAG.RECORD.SIZE
```

When the number of rough table entries equals the number of tag file blocks, minimum disk activity is achieved to secure the tag record that holds the key requested. The rough table is binarily searched to find what part of the tag file holds the key. That partition of the tag file is then binarily searched (by randomly reading the tag records) until the record key is either found or it can be determined that the requested key does not exist. Therefore the optimum CORE-INDEX (lowest memory for the lowest physical I/O activity) may be found as:

```
Optimum CORE-INDEX = (TAG.EOF/TAG.RECORDS.BLOCK)*RECORD.KEY.SIZE
```

Increasing the CORE-INDEX beyond this value (up to TAG.EOF * RECORD.KEY.SIZE) will reduce the number of logical I/O operations, but will not decrease the number of physical I/O operations. Increasing the CORE-INDEX further wastes memory.

Example:

```
FILE CONTAINS 36 BY 540 RECORDS
BLOCK CONTAINS 6 RECORDS
RECORD.KEY.SIZE IS 4 BYTES
```

Assuming that the file is full (TAG.EOF=19440), applying the previous formulas results in the following:

```
      TAG.RECORD.SIZE = 8 bytes      (no $ RPGTAGS)
    TAG.RECORDS.BLOCK = 22
  Optimum CORE-INDEX = 3,532 bytes
```

If allocation of 3532 bytes for the rough table is considered excessive, the following formula may be applied to decrease the size of the rough table and still achieve a high degree of efficiency:

```
  New CORE-INDEX = Optimum CORE-INDEX/((2**n)-1)  (rounded up)
```

where "n" varies from 2 by 1 until the new CORE-INDEX is an acceptable value. For example, if 3532 is too much, then divide by 3, giving 1178. If this is still too high, divide by 7, giving 505, and so forth.

6. BASIC

6.1. Introduction.

The Mark VII.0 release of BASIC contains no updates or enhancements to the current implementation.

A supplement to this release document, containing complete documentation on BASIC, is available in the form of a printer backup file labeled DOCUMENT/BASIC.

6.2. Known Errors and Restrictions.

6.2.1.
The BASIC compiler does not generate a syntax error when arrays of different dimensions are assigned.

Example:

```
10 DIM A(10), B(5,2)
20 MAT A=B
30 MAT B=A
```

The assignment statements should produce two syntax errors; however, they compile error-free, generating a run-time INVALID SUBSCRIPT error.

7. FORTRAN

7.1. Introduction.

The Mark VII.0 release of FORTRAN features the implementation of free-format I/O, or slash editing. In addition, a number of syntactically incorrect constructs which previously failed to cause syntax errors are now diagnosed correctly.

7.2. Enhancements.

7.2.1.
Truncation of dynamic memory to 2560 words in the object program can now be overridden by using the $ DYNAMIC compiler option. For example, the following control statement will extend dynamic memory to 3000 words:

```
$ DYNAMIC 3000
```

7.2.2.
Free-format I/O, or slash editing, has been implemented. Temporary documentation is contained in paragraph 7.4.1.

7.2.3.
A new compiler option, $ TRUNCATE, allows variable names to be extended to more than six characters without causing a syntax error. Such names are then truncated to six characters by the compiler. Care must be taken to insure that these truncated names are still unique.

7.2.4.
A new compiler option, $ [NO] SUPPRESS, has been implemented to allow suppression of warning messages on the output listing. The default is NO SUPPRESS.

7.2.5.
Named BLOCK DATA subroutines are now permitted. To be bound in, the BLOCK DATA subroutine must be specified in an EXTERNAL statement. For example:

```
BLOCK DATA DAT1
```

7.3. Known Errors and Restrictions.

7.3.1.
The compiler fails to syntax duplicate identifier names in a COMMON statement, when the last two elements of the COMMON are the duplicates.

7.3.2.
A READ statement having the erroneous syntax:

```
READ (<unit>) <namelist id>
```

rather than the correct syntax:

```
READ (<unit>, <namelist id>)
```

generates a call to a non-existent intrinsic ".RDSU" (assuming an ordinary unformatted READ). The error is detected at bind time.

7.3.3.
The NAMELIST name "DATA" causes a number of irrelevant syntax errors when used in an I/O statement.

7.3.4.
The compiler is erroneously syntaxing a correct ASSIGN statement. For example, the statement:

```
ASSIGN 46 TO NASSGN
```

results in the following error message:

```
ERROR MAX LENGTH OF IDENTIFIER IS 6 CHARACTERS
```

7.3.5.
The assignment statements "F2 = 99999999999999999." and "F2 = .999999999E+10" fail to receive a truncation error message.

7.3.6.
The format "F1.8" compiles without syntax errors, but upon execution produces only one position of output.

7.3.7.
Listing a FORTRAN backup file under control of CANDE correctly lists the file, but results in the following warning message:

```
WARNING: RECORD SIZE IS GREATER THAN 133; TRUNCATION WILL OCCUR
```

7.3.8.
The FORTRAN compiler is erroneously syntaxing calls to subroutines whose names were passed as arguments. For example:

```
SUBROUTINE BUG(ERROR)
CALL ERROR (NAVY,TEST)
```

Temporary fix: Use the EXTERNAL statement.

7.4. Temporary Documention.

7.4.1. Free-format I/O (Slash Editing)

Input Files

Free-format input reads from the input file until the I/O variable list is exhausted.

General Format

READ /,m
READ (u,/)m
READ (u,/,l)m
where u represents a unit number; l is an action specifier list; and m is an I/O variable list.

Syntax Rules

a. Values in the input file must be separated by commas.

b. Blanks (even embedded blanks) are ignored.

General Rules

a. The end of a physical record terminates the current value, and the next value is read from the succeeding physical record.

b. Only certain types of input are permitted for a specific type of variable:

```
Integer:  I format
Real:     I, E, F, or D format
Double:   I, E, F, or D format
Complex:  I, E, F, or D format
Logical:  L format
```

c. Values of 0 or FALSE are the default if a blank or null field is read.

Example

```
LOGICAL O
READ (5,/,END=50) T,I,O
```

The data record might contain:

```
1234.56,227,1
```

Output Files

Free-format output writes values to the output file until the I/O variable list is exhausted.

General Format

WRITE (u,/)m
WRITE (u,//)m
WRITE (u,*/)m
WRITE (u,*//)m
PRINT /,m
PRINT //,m
PRINT */,m
PRINT *//,m
PUNCH /,m
PUNCH //,m
PUNCH */,m
PUNCH *//,m
where u represents the unit number; and m is the I/O variable list.

General Rules

a. Values are output in the most appropriate format with trailing zeros and blanks removed.

b. Values are separated by a comma and a blank if the second slash is not included.

c. Values are separated by two blanks if the second slash is present (note that this output cannot be read in by free-format input due to the lack of a comma).

d. Values are preceded by "<variable name>=" if the optional asterisk is present. If the value is an expression or array element, "<EXP>" will be substituted for the name.

e. If a value, the "<variable name>=" (if required), the preceding blank, and the trailing comma or blank cannot all fit into the end of a particular record, they are all placed in the succeeding record.

f. Literal strings up to 255 characters in length may be output using free-format I/O; but each string must be able to fit within a single record.

Example

```
LOGICAL D
A= 27.2
I= 19
D= .FALSE.
PRINT *//,A,I,D
```

Execution of this program segment produces the following output:

```
A=27.2  I=19  D=F
```

Intrinsics

Twenty-four intrinsics have been added to provide this feature:

.ISSR	Initiate a Serial Slash-edited (free-format) Read
.ISSW	Initiate a Serial Slash-edited (free-format) Write
.RCAS	Read a Complex Array Slash-edited (free-format)
.RCSS	Read a Complex Scalar Slash-edited (free-format)
.RDAS	Read a Double Array Slash-edited (free-format)
.RDSS	Read a Double Scalar Slash-edited (free-format)
.RIAS	Read an Integer Array Slash-edited (free-format)
.RISS	Read an Integer Scalar Slash-edited (free-format)
.RLAS	Read a Logical Array Slash-edited (free-format)
.RLSS	Read a Logical Scalar Slash-edited (free-format)
.RRAS	Read a Real Array Slash-edited (free-format)
.RRSS	Read a Real Scalar Slash-edited (free-format)
.WCAS	Write a Complex Array Slash-edited (free-fomat)
.WCSS	Write a Complex Scalar Slash-edited (free-format)
.WDAS	Write a Double Array Slash-edited (free-format)
.WDSS	Write a Double Scalar Slash-edited (free-format)
.WHAS	Write a Hollerith Array Slash-edited (free-format)
.WHSS	Write a Hollerith Scalar Slash-edited (free-format)
.WIAS	Write an Integer Array Slash-edited (free-format)
.WISS	Write an Integer Scalar Slash-edited (free-format)
.WLAS	Write a Logical Array Slash-edited (free-format)
.WLSS	Write a Logical Scalar Slash-edited (free-format)
.WRAS	Write a Real Array Slash-edited (free-format)
.WRSS	Write a Real Scalar Slash-edited (free-format)

8. UPL

8.1. Introduction.

The Mark VII.0 version of UPL adds no new constructs to the UPL language. Certain portions of the compiler have been recoded to increase efficiency, and all known problems in the compiler have been corrected.

8.2. Enhancements.

8.2.1.
A problem has been fixed in which variables appearing within DEFINE expansions were not always appearing in XREF listings.

8.2.2.
The FPB.PSEUDO boolean is now set by the compiler when a file is declared with "FILE_TYPE = PSR_DECK", eliminating the need to modify that file after compilation as discussed in this section of the Mark V.1 release letter.

8.2.3.
The compiler now gives a syntax error if a "$ LIBRARY" card appears within a library file.

8.2.4.
A problem has been fixed in which the compiler would fail if a token ended in column 72 and the next source image was a "$" statement.

8.2.5.
Several unusual and erroneous syntactic constructs which caused the Mark VI.1 UPL compiler to terminate with an INVALID CASE or INVALID SUBSCRIPT error are now detected and diagnosed with error messages.

8.2.6.
If a "DEVICE=<hardware type>" phrase is not included in a file declaration statement, the hardware will default to disk rather than to tape as in previous releases of UPL.

8.2.7.
"ON" clauses (ON FILE_LOCKED, ON FILE_MISSING) are now permitted in "OPEN" statements for files of hardware type "REMOTE". In the Mark VI.1 release of UPL, such clauses would cause syntax errors.

8.2.8.
The algorithm for computing dynamic memory has been corrected, avoiding the large values the compiler would erroneously assign in certain cases.

8.2.9.
The compiler will no longer terminate with an INVALID SUBSCRIPT error if an "&" statement has no syntactic tokens following the ampersand.

8.2.10.
SDL Intrinsic #4, which handles the "CONVERT" function, now uses the "TRANSLATE" statement to perform bit-to-character conversions, substantially improving the execution speed of this function.

8.2.11.
A problem has been fixed in which a percent sign appearing in column 72 of a source image which was part of a DEFINE definition caused the next source image to be skipped when that DEFINE was expanded.

8.2.12.
The compiler now emits a warning if the number of characters in a DEFINE is more than 4000, but fewer than the limit of 4095. This warning is intended to alert the programmer to the possibility of future development problems if the definition is increased in size.

8.2.13.
An error message will be emitted if the total stack size within a compiled program exceeds 128,000 bytes.

8.2.14.
The compiler will no longer abort if the syntactic token "FINI" appears in columns 69-72 of a source image.

8.2.15.
The occurrence of any unquoted special character other than "/" on a "$ LIBRARY" record is now syntaxed. For example, the common error of accidentally ending such a record with a semicolon (causing the compiler to incorporate the semicolon as part of the library file name and then to attempt to open the wrong file) can no longer occur.

8.2.16.
The compiler now properly handles a dynamic declaration of type "RECORD".

8.3. Known Errors and Restrictions.

8.3.1.
A "STATUS" inquiry from a remote terminal can sometimes "catch" the compiler between passes, causing meaningless information to be returned in the sequence number portion of the response. This can sometimes cause unpredictable transmissions or hangs if the information generated includes special data comm characters, requiring a "?CQ" to clear the transmission.

8.3.2.
The user of extended arithmetic operations should be aware of the incorrect detection of field overflows beyond the 65,535-bit maximum capacity of UPL. Refer to the SDL interpreter section of this release document for details.

8.3.3.
The "COMPILE_CARD_INFO" statement documented in the MARK VI.1 release letter is in fact not a part of the UPL language and the documentation should be ignored. The UPL language is designed to be isolated from system-dependent functions, and this includes COMPILE_CARD_INFO, whose information content and format is highly changeable between software releases. A statement similar in intent, but returning consistent information in a non-changing format, is planned for a future release.

8.4. Temporary Documentation.

8.4.1.
A subscript is not permitted with the queue family name when using the "READ_OK" variant of the WAIT statement upon a queue family. This form of the WAIT statement examines all members of the queue family, waiting for a message to appear in at least one queue. Thus it is invalid to attempt to specify one particular family member in the "WAIT" statement.

8.4.2.
A "MESSAGE_COUNT" statement is mandatory to ensure correct results on a following "WAIT" statement waiting for "Q_WRITE_OCCURRED".

8.4.3.
The "ON FILE_LOCKED" clause documented in the UPL manual as part of the "SEARCH_DIRECTORY" statement is not, and never has been functional. As the number of users is one subfield returned with this statement, the implementation of the "ON FILE_LOCKED" clause would be redundant, and is hence not planned.

8.4.4.
The following is a complete list of allowable file attributes:

```
ALL_AREAS_AT_OPEN
AREA_BY_CYLINDER (not implemented)
AREAS
BUFFERS
DEVICE
END_OF_PAGE_ACTION
EU_INCREMENTED
EU_SPECIAL
FILE_TYPE
INVALID_CHARACTERS
LABEL
LABEL_TYPE
LOCK
MODE
MONITOR_INPUT_FILE
MONITOR_OUTPUT_FILE
MULTI_PACK
NUMBER_OF_STATIONS
OPEN_OPTION
OPTIONAL
PACK_ID
PROTECTION
PROTECTION_IO
RECORDS
REEL
REMOTE_KEY
SAVE
SERIAL
TRANSLATE
USE_INPUT_BLOCKING
USER_NAMED_BACKUP
VARIABLE
WORK_FILE
```

8.4.5.
In the absence of a "RECORDS" phrase in a file declaration, the following default values are assigned:

Device	Record Size
DISK	180 bytes
TAPE	80 bytes
any 96-column CARD device	96 bytes
any other CARD device	80 bytes
PRINTER	132 bytes
all other devices	72 bytes

As always, the USE_INPUT_BLOCKING clause will cause the FPB.DEFAULT boolean to be set for the file, causing the actual record size of an already-existent file opened INPUT to be used instead of the above values.

9. DMSII

9.1. Introduction.

The major new feature in the Mark VII.0 release of DMSII is the implementation of record-level lockout on DMSII data set records. In addition, a restart capability has been added to the reorganization programs, several changes have been made to DASDL syntax, and many problems from previous releases have been corrected.

9.2. Enhancements.

9.2.1.

The following problems reported in the Mark VI.1 release letter have been fixed (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. "WHERE", "VERIFY", or "REQUIRED" statements which reference data items larger than 1000 bytes in length will no longer cause an MCP NAME/VALUE STACK OVERFLOW halt (17.3.1).

b. The population specified for an automatic subset is now used by the DASDL compiler to calculate physical parameters for that subset; earlier compilers used the population of the data set for these calculations (17.3.2). Note that any automatic subset generated under Mark VI.1 or earlier versions of DASDL, and which had an explicit POPULATION statement which differed from the population specified for the data set, will require reorganization if an UPDATE or REORGANIZE compile is performed on the data base. This can be prevented by setting the population for the subset equal to that of the data set, or by removing the POPULATION statement from the subset declaration.

c. A second "$ MERGE" statement is now ignored by the DASDL compiler (17.3.3).

d. The DASDL compiler no longer generates COBOL library files in which a period precedes the first "02" item (17.3.4).

e. Although DMSII normally does not perform a complete close on a data base until all users have closed it, if all update users have closed the data base, but there are still inquiry users active, a "pseudo close" is performed on the data base dictionary to reflect the fact that no more updates are being done. This prevents the data base from requiring recovery if a system failure should occur before the data base becomes inactive (17.3.7).

f. Prime data sets for which segments per area is not a multiple of segments per block can now be accessed in data set order. The problem within the DASDL compiler which created this situation has also been corrected (17.3.8).

g. The correct set of COBOL library files, labeled "#<data-base-name>/<data-set-name>", will now be present on disk after a reorganization in which a manual subset is added to, or deleted from, a data set which is also being PURGED (17.3.9).

h. DEADLOCK is now detected if a program has multiple invokes of a data set and attempts to MODIFY a record by the second invoke, when that record is already locked by the first invoke (17.3.10).

9.2.2.
The following problems reported by System Flashes since the Mark VI.1 Release have been corrected:

a. The Disk Search Operator is now retried automatically by GISMO if an I/O exception condition occurs (Flash #186). This has required that the Search Op be rewritten to bring the entire index table into memory before proceeding with the search. In previous releases, the search was performed on each individual index entry as it was read into memory; the memory required for this was just the length of a single entry.

b. The INITIALIZE statement in DASDL is now optional in all compiles. The DASDL compiler assumes INITIALIZE unless a compile for syntax is performed (Flash #191).

c. The DASDL compiler will now syntax an attempt to change the attributes, during a reorganize compile, of any item which is used as a KEY for an ordered manual subset (Flash #192).

d. Audit and recovery may now be added to, or deleted from, a data base during a reorganize DASDL compile (Flash #195).

Refer also to paragraph 9.2.6 below for further information relative to items (b) and (d) above.

9.2.3.
Data set records are now locked at the record level. Prior to Mark VII.0 DMSII, if a physical block contained more than one data set record, and one of those records was locked by a user, no other user could access any of the records in that physical block until the first user had released the block. Under Mark VII.0 DMSII, only the specific record requested by the user program is locked. In addition, the lock is non-exclusive; that is, while a record is locked, other users may perform a FIND on that record without receiving a DEADLOCK exception.

The implementation of record-level lockout has required several changes to the manner in which DMSII allocates and manages buffers while processing a data base. The Lock Descriptor Table and the Lock Descriptors themselves have been changed. The function served by Lock Descriptors prior to Mark VII.0 is now performed by Buffer Descriptors and a new type of Lock Descriptor. The first 144 bits of each DMSII buffer contains the Buffer Descriptor for that buffer; this descriptor identifies the disk address of the data contained in the buffer, the number of users for the buffer, and the structure number for the buffer. In addition, there is a Lock Descriptor present for each record which is currently locked; this descriptor identifies the buffer which contains the locked record, and further describes which of the records within that buffer is locked, to which user (if any) the record is assigned, and to which of the three classes of lock the record belongs:

a. User Lock. A user has explicitly locked the record by a MODIFY operation. Only data set records, both disjoint and embedded, can be locked in this fashion. The records described by this type of Lock Descriptor are available to be read by other users (non-exclusive lock).

b. MCP Read. This is a temporary condition, used by DMSII when reading information from disk. This type of Lock Descriptor is used to prevent the MCP Memory Management routines from overlaying the buffer until DMSII has completed the current operation. The record described is not locked in the sense of being unavailable to other DMSII users.

c. MCP Lock. This is also a temporary condition, but the record described by this Lock Descriptor is locked exclusively. This is used by DMSII when a record has been updated by a partially completed operation. Only the MCP can access the information in this record until the current operation completes, at which time the lock status of the record will return to the state it was in prior to the operation.

There is a field within the Buffer Descriptor which allows DMSII to maintain the buffers as a linked list within memory. Only the MCP Memory Management routines can deallocate DMSII buffers, at which time the descriptor for that buffer is delinked from the Buffer Descriptor chain.

In previous DMSII releases, the BUFFERS parameter in DASDL was used to specify the length of the Lock Descriptor Table. With the Mark VII.0 release, however, the length of this table is fixed at 20 Lock Descriptors, each descriptor being 90 bits in length. If 20 is not enough, additional tables of 20 are allocated as required. Any overflow Lock Descriptor Tables remain in memory until the data base is closed.

9.2.3.1.
Because of the changes in the allocation and maintenance of the Lock Descriptor table, as described above, the BUFFERS clause has been removed from the DASDL syntax. The compiler will ignore any attempt to set this parameter.

9.2.3.2.
The BUFFERS clause has been removed from the SM input message. Refer to paragraph 9.2.7 of this document for the new syntax of the SM message.

9.2.4.
The reorganization programs, DMS/REORG.READ and DMS/REORG.WRIT, are now restartable. If the reorganization process was interrupted by a CLEAR/START or by DS-ing either of the reorganization programs, the reorganization can be restarted by re-executing DMS/REORG.READ. This can be done unless the reorganization was interrupted for one of the following reasons:

a. Either of the reorganization programs encountered an I/O exception condition in the temporary data base dictionary or in the reorganization control file. The temporary dictionary is labeled "#<data-base-name>/DICTIONARY"; the DASDL listing includes the name of the reorganization control file.

b. Any data base exception condition other than an IOERROR.

c. Reorganization program errors.

d. The number of areas for a file is being changed, and the new number of areas requested exceeds the number already allocated.

e. Index sequential table overflow during a garbage collection.

The listings generated by the reorganization programs can be used to determine if the reason for the interruption was one of the above. If any of these conditions occurs, the pre-reorganization versions of any data base files which were removed by DMS/REORG.WRIT must be restored before the data base can be used or another attempt at reorganization can be made; this must also be done if the user decides not to restart a reorganization which is restartable.

Both of the reorganization programs must be terminated before restarting DMS/REORG.READ.

In general, the user need not be aware of the particular phase of the reorganization being performed at the time of the interruption. However, there are a few situations in which the user must take some action before restarting the reorganization. The existence of these situations can be determined from the printed listing generated by DMS/REORG.WRIT. These situations are the following:

a. If DMS/REORG.WRIT was in the process of changing the number of areas for existing files (the message "BEGIN FILE AREAS CHANGE" appears in the listing, but the message "END FILE AREAS CHANGE" does not), all of the data base files which were to have their areas changed must be restored to their pre-reorganization state.

b. If DMS/REORG.WRIT was in the process of copying files from a work pack to the pack on which the files are permanently assigned, and an I/O exception condition occurred while reading the work file, then the reorganization programs must recreate that work file. If any of the pre-reorganization files needed to recreate the corrupted work file does not exist in the new data base, but new files with the same names do exist in the new data base, DMS/REORG.READ will display a message that the newly created files with the same name must be saved, and the pre-reorganization files must be reloaded. After the corrupted file has been recreated, the backup copies of the new files can be restored.

c. If the interruption occurs after DMS/REORG.WRIT has removed the old data base dictionary, but before it has changed the name of the temporary dictionary, the user can change the name, and it is not necessary to restart the reorganization programs.

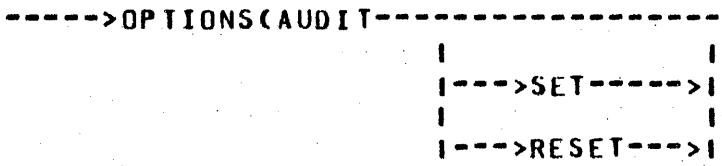
2.2.5.

As described in paragraph 17.4.8 of the Mark VI.1 release letter, DMSII no longer maintains the control items RESTART TYPE and TRANSACTION COUNT within the restart data set record. Programmatic restart procedures which rely on the settings of these items must be rewritten before attempting to implement the Mark VII.0 release, as the results of such procedures are unpredictable.

The DASDL compiler will treat these data items as having type NUMBER(1) and NUMBER(6), respectively. Additionally, users can change their data base descriptions such that the item types for these fields have the above values, without being forced to perform a reorganization of the data base. The Mark VIII.0 DASDL compiler will treat these data types as syntax errors.

9.2.6.
In order to increase the compatibility of DASDL syntax among all of the Burroughs DASDL compilers, a major effort to standardize the DASDL language is currently in process. This standardization has required that several changes be made to the DASDL syntax for the B 1800/B 1700. Most of these changes are additions of synonyms for existing terms, and will not directly affect existing data base descriptions when an update or reorganization compile is performed. However, it is possible through use of a new dollar-card option, "STANDARD", for the DASDL compiler to generate warnings whenever any syntax is encountered which does not conform to the standard syntax for the DASDL language.

9.2.6.1.
The syntax of the OPTIONS statement has been expanded to the following:



When the AUDIT option is used, the data base must contain exactly one restart data set. If the option is SET, DMSII will maintain an audit trail of all updates to the data base. If the option is RESET, no audit trail will be maintained, but the restart data set must still be present. This allows a user to selectively disable the auditing of certain processes, such as the initial load of the data base, when the amount of overhead required to audit the process and, if a failure should occur, recover the data base and restart the process, make it more practical to by-pass the audit and reload the data base and start the process again.

When AUDIT is reset, DMSII ignores any BEGIN or END-TRANSACTION requests issued by user programs. It is therefore possible, if a system is being debugged with AUDIT reset, to attempt DMSII operations which would have resulted in exceptions if AUDIT had been set. For example, an AUDITERROR would be generated if the program were to attempt to perform a BEGIN-TRANSACTION or data base CLOSE while in transaction state, or an END-TRANSACTION while not in transaction state; if AUDIT were reset when any of these operations was attempted, no exception would be detected.

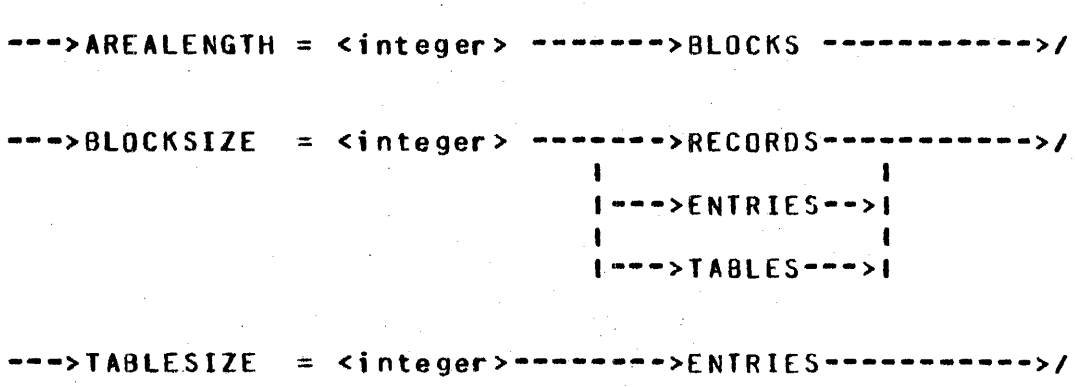
AUDIT can also be SET or RESET through the SM input message. Refer to paragraph 9.2.7 of this document, as well as to the revised Software Operational Guide, for the updated syntax of this message.

9.2.6.2.
The following table lists keywords which have been implemented for Mark VII.0 in order to conform to the standard syntax for the DASDL language, and which are synonymous with keywords used by the pre-VII.0 versions of the compiler. Opposite each entry in the table is the pre-VII.0 synonym for the keyword; the DASDL compiler will include a warning in the listing upon encountering these pre-VII.0 keywords if $ STANDARD has been specified.

Keyword	Synonym
AREALENGTH	AREASIZE
FAMILYNAME	PACK
TAPEPE	PETAPE
MAXENTRIES	POPULATION
MAXRECORDS	POPULATION

MAXENTRIES is used to specify the maximum number of entries for a set or subset, and MAXRECORDS is used to specify the maximum number of records for either disjoint or embedded data sets.

9.2.6.3.
The syntax for the physical parameters AREALENGTH, BLOCKSIZE, and TABLESIZE has been changed to the following:

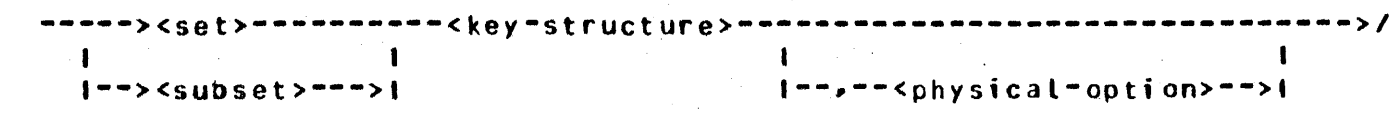


If $ STANDARD is set, the absence of the unit specifiers (BLOCKS, RECORDS, ENTRIES, or TABLES) in any of the above will result in a warning. In the case of BLOCKSIZE, if the unit specified does not apply to the structure being described, a warning will also be generated.

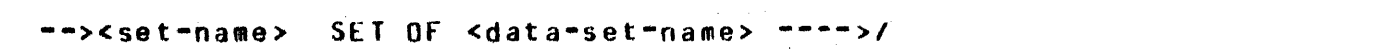
9.2.6.4.
INITIALIZE is assumed whenever the DASDL compile is not for syntax only. If $ STANDARD is set, DASDL will generate a warning when INITIALIZE is encountered.

9.2.6.5.
The syntax for set or subset declarations has been changed to the following:

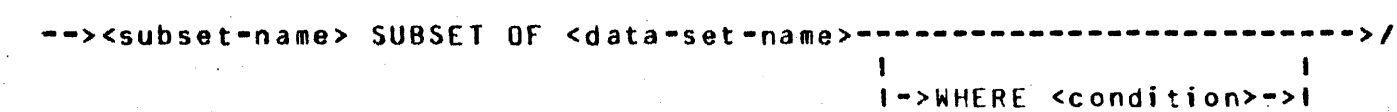
<set or subset>:



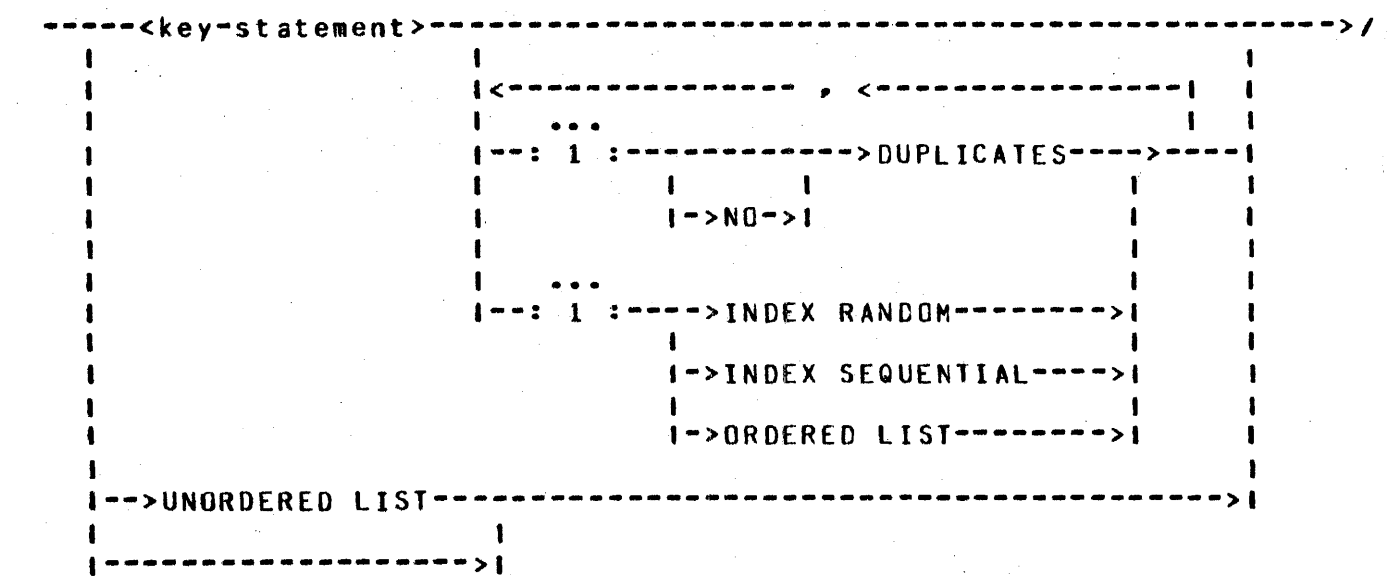
<set>:



<subset>:



<key-structure>:



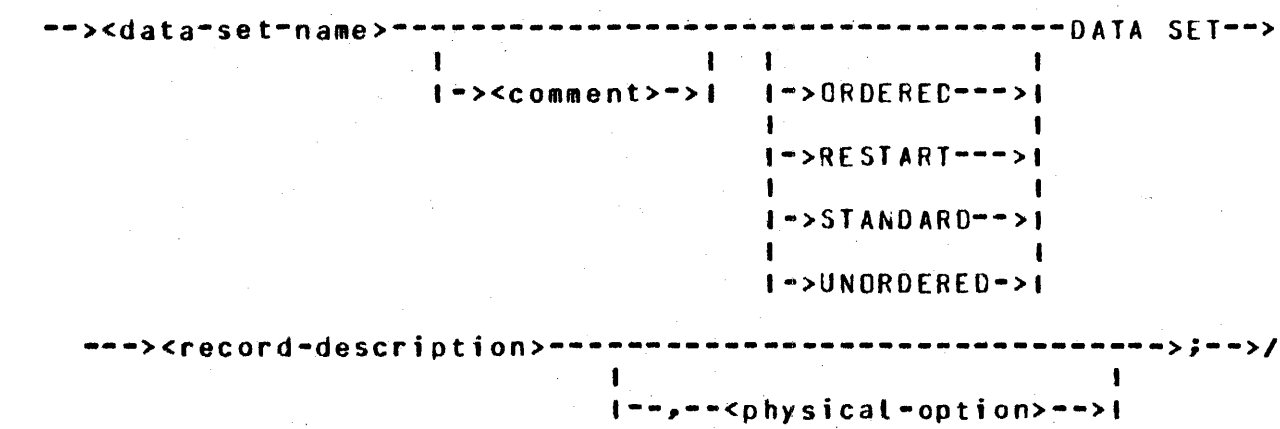
The syntax for the <key-statement> and <condition> is unchanged. <physical-option> refers to any physical parameters which may be applicable to the structure being described, such as LOADFACTOR, PRIME, MODULUS, and so forth. Note that these physical parameters can appear either within the description of the structure, as shown above, or at the end of the data base description, as in previous releases of DASDL.

If $ STANDARD is set, the keywords ORDERED and RETRIEVAL preceding the keyword SET, and the keywords ORDERED and UNORDERED preceding the keyword SUBSET, will generate warnings. INDEX SEQUENTIAL is the default type for all sets and automatic subsets.

Because of these changes to the set and subset syntax, the TYPE=INDEX SEQUENTIAL and TYPE=INDEX RANDOM statements have been removed from the DASDL syntax.

9.2.6.6.
The syntax for a data set has been changed to the following:

<data-set>:



The data set types of RESTART, ORDERED, and UNORDERED are unchanged from previous releases. The data set type of STANDARD is the default, and describes a simple disjoint data set.

As in the case of sets and subsets, the <physical-options> are used to set any physical parameters for the data set, and may appear either with the data set description or at the end of the DASDL source deck.

9.2.6.7.
Quoted comment strings may only appear between an item name and the type declaration for that item (for example, between the name of a data set and the keywords DATA SET, or between a data item name and the keyword NUMBER). The DASDL compiler will generate a warning if any quoted comments are encountered anywhere in the data base description other than as specified.

9.2.6.8.
The user can designate permanent DASDL options by coding two consecutive dollar signs ($$) in columns 1 and 2 of an option record; these options are included in any new DASDL source file created by the inclusion of a NEW dollar option within the source file. A single dollar sign in column 1 of the option record indicates a temporary option; the options on such a record are not included in any new source file which is created.

9.2.6.9.
The keywords SET and RESET may precede any DASDL option. The keywords NO and RESET are synonymous, and have the effect of resetting the following option. SET enables an option, and is assumed if an option appears without an explicit SET, RESET, or NO.

9.2.6.10.
The following is a list of dollar options which have been added to the DASDL compiler for the Mark VII.0 Release:

DEBUG — This option causes the DASDL compiler to include its internal reorganization control information in the listing generated during a reorganization compile. This option should only be set when problems with reorganization are encountered, and only for the purpose of assisting the Burroughs software development and support personnel with the isolation of such problems. Due to the nature of the information generated and the likelihood that such information (and the format in which it is printed) may change with each release, Burroughs does not intend to provide and maintain documentation concerning decoding of this information.

INCLUDE <file-id> — This option allows specification of a library file to be included in the data base description. The file specified by <file-id> must be present on disk, and must consist of one or more DASDL source statements. The DASDL source statements within the library file can include any dollar option statements other than another INCLUDE, or a VOID statement.

INCLNEW — When enabled, this option instructs the DASDL compiler to include any source statements processed as a result of an INCLUDE option within any new source file generated by means of the NEW option.

LIST$ — This option causes all temporary DASDL option statements to be included in the compiler listing. This option has no effect if the LIST option has been disabled.

LISTINCL — This option causes all source statements processed as a result of an INCLUDE option to be included in the compiler listing. This option has no effect if the LIST option has been disabled.

SEQUENCE <b> <+i> This is the same as the SEQ option. If specified, this option causes the DASDL compiler to resequence the source statements. If NEW is also enabled, the new sequence numbers are permanently assigned to the new source file. If specified, <b> is used as the base for the new sequence numbers; this base defaults to "00001000". If specified, <+i> is used as the increment for the new sequence numbers; the default increment is +1000.

STANDARD This option causes the DASDL compiler to generate warning messages when it encounters syntax which does not conform to the standard syntax for the DASDL language.

WARNSUPR This is the same as the SUPPRESS option. Warning messages are suppressed in the compile listing.

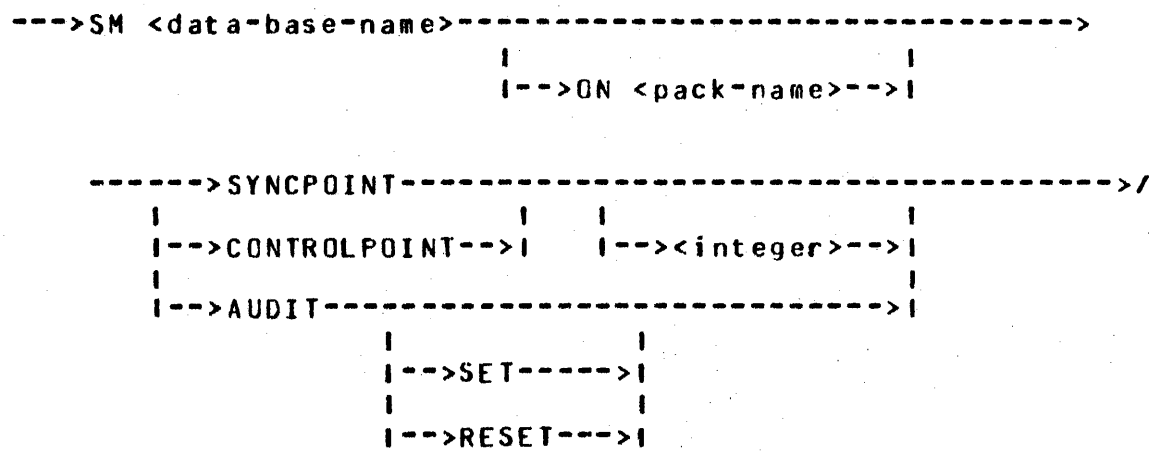
UPDATE <db-id> The UPDATE option is the same as in previous releases. However, the <db-id> can be used to change the name of an existing data base from <db-id> to the name on the DASDL COMPILE control statement.

PAGE This option immediately causes a page advance in the DASDL listing.

DELETE When enabled, this option causes source input records in the secondary input file ("SOURCE") to be discarded by the DASDL compiler. This option can only appear on a source image in the primary input file ("CARDS"), and has no effect unless MERGE was enabled. The merging process is not altered by this option, but the source statements in the secondary file are ignored until the DELETE option is disabled. If a new source file is being created, the deleted records will not appear in that file.

2.2.6.11.
If $ STANDARD is specified and an attempt is made to set the BUFFERS parameter, a warning is generated.

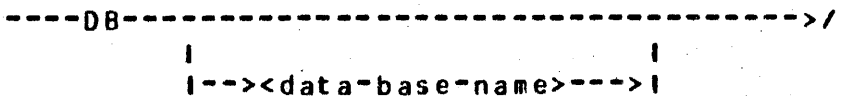
2.2.7.
The syntax of the SM input message has been changed to the following:



If the data base does not have AUDIT specified in the DASDL source, the MCP will not perform the SM. The data base must be inactive in order to change any of the settings. If no <integer> is supplied for SYNCPOINT or CONTROLPOINT, or if neither SET nor RESET is specified for AUDIT, the current setting of the parameter is displayed. The data base does not need to be inactive to query the settings of these parameters.

2.2.8.
The presence of an active multi-line control will no longer cause unpredictable results when using the SM input message.

2.2.9.
The syntax of the DB input message has been changed to the following:



If <data-base-name> is supplied, and that data base is active, the MCP will display the current values of the data base statistics normally displayed when the data base is physically closed.

2.2.10.
The "DATE.COMPILED" field within the Program Parameter Block (PPB) of the reorganization programs will reflect the date of the reorganization DASDL compile.

2.2.11.
A "FIND AT KEY" on an ordered set or automatic subset which specifies a key value higher than any key in the index will no longer result in wrap-around on the index if the next operation on the index is a "FIND NEXT".

2.2.12.
Quoted comments which end in column 72 of a DASDL source statement will no longer cause DASDL to terminate with an INVALID SUBSTRING error.

2.2.13.
When processing indexed structures, DMSII now releases any intermediate index tables as soon as those tables have been processed. For index random structures, only the base table and current overflow table will be locked in memory. For index sequential structures, only the root coarse table and current fine table will be locked for a FIND or MODIFY operation; for an update operation, any intermediate coarse tables also affected by the operation will be locked. All index tables are released at the end of an operation.

2.2.14.
The DASDL compiler no longer generates incorrect sequence numbers if a sequence number appears in columns 73-80 of the dollar statement containing the SEQ option.

2.2.15.
The DASDL compiler no longer terminates with an INVALID SUBSCRIPT if a special character is encountered while scanning for the beginning of a token.

2.2.16.
The comma between key specifications and the population statement for manual subsets is now optional in the DASDL source deck.

2.2.17.
The DASDL compiler no longer generates invalid Next Available - Highest Open (NAHO) information for non-prime data sets having more than 20 areas.

2.2.18.
Unpaired parentheses in a key clause are now detected by the DASDL compiler.

2.2.19.
DMSII now resets the SEARCH.N.LINK.COUNT to zero when a data base is initially opened.

9.2.20.
Due to the rewriting of the MCP Memory Management routines, all DMSII buffers are now overlayed only by these routines. Prior to the Mark VII.0 release, DMSII buffers could be overlayed by either DMSII or the Memory Management routines. As a result of this, all known problems with I/O exception handling have been corrected. This includes the following:

a. A NOTFOUND exception, rather than an IOERROR, was returned to a program.

b. DMSII was displaying the following message:

IO ERROR - DMS WRITE ON STRUCTURE NUMBER <nn>

However, the programs currently accessing the data base were not forced to terminate, nor was Dump Recovery required.

9.2.21.
The DASDL compiler now uses paged arrays exclusively for all of its tables. While this can cause compile speed to be much more sensitive to the amount of dynamic memory assigned, this has been done to eliminate all of the problems which arose in previous versions of the compiler due to errors in the compiler's internal memory management. The problems corrected by this change include the following:

a. The DASDL compiler would generate different results, depending upon the order of the source statements.

b. The DASDL compiler would abort, after displaying the message:

*** COMPILER ERROR AT <seq-no>

c. The DASDL compiler would hang in a loop.

9.2.22.
DMS/REORG.WRIT no longer terminates with a NAME/VALUE STACK OVERFLOW when purging an index which is stored in a file with other indexes.

9.2.23.
When performing an update or reorganization compile, the DASDL compiler now compares WHERE and VERIFY conditions for strict equality first, and if that comparison fails, the compiler then checks for equivalence. In previous releases of the compiler, the test was only for equivalence. Because of the test for equality, update and reorganization compiles using the Mark VII.0 DASDL compiler require less time and dynamic memory than similar compiles using earlier versions of the compiler.

9.2.24.
The DASDL compiler now allows duplicate data names within a data base description, with the following restrictions:

a. Structure names must be unique within the data base.

b. Data names for items used as key fields must be unique within the data base.

c. Data names for items not used as key fields must be unique within each data set.

9.3. Known Errors and Restrictions.

9.3.1.
An MCP INVALID SUBSTRING halt (L=@000006@) can occur if

(SPLITFACTOR) x (Entries per Coarse Table)

for an index sequential set exceeds 511, or if either LOADFACTOR or SPLITFACTOR for a set is 100%.

9.3.2.
When processing sequentially through an Ordered Set or Automatic Subset, if a NOTFOUND exception is returned on a FIND NEXT operation, the currents for the structure will be in the same state as if the data base had just been opened. That is, another FIND NEXT at this point will return the first record in the set (wrap-around). A FIND PRIOR at this point is invalid, returning a NOTFOUND exception.

9.3.3.
The DASDL compiler may terminate with an INVALID SUBSCRIPT when generating the code to perform "REQUIRED", "WHERE" and "VERIFY" checking, if the amount of code necessary to perform all of these operations exceeds 65,535 bits. The code generated by DASDL is bound into the MCP at data base open time, and the limit of 65,535 is a function of the SDL S-machine.

9.3.4.
An MCP CONTROL STACK OVERFLOW halt (L=@000002@) can occur if disk I/O errors occur during data base CLOSE.

9.3.5.
When "$ COBOL" is specified in the DASDL source file, the DASDL compiler generates a syntax error if any of the reserved words listed in appendix A of the COBOL manual are encountered. This includes all of the reserved words which are identified as "For future use or standards requirements", even though the COBOL compiler itself does not generate syntax errors upon encountering these words.

9.4. Temporary Documentation.

9.4.1.
Because of changes which have been made to the data base dictionary format, a $ CONVERT DASDL compile must be performed before a Mark VI.1 data base can be accessed with Mark VII.0 DMSII. As was done in the Mark VI.1 release, the $ CONVERT statement should be the only record in the DASDL source file. Existing COBOL programs can access a data base under Mark VII.0 without being recompiled; however, if the data base uses Audit and Recovery, any update programs must conform to the restart procedures outlined by paragraph 9.2.5 of this document and paragraph 17.4.8 of the Mark VI.1 release letter.

9.4.2.
Because of the changes to the data base dictionary mentioned in paragraph 9.4.1, it is not possible to process a data base under the Mark VI.1 release of DMSII after it has been updated by the Mark VII.0 release. Also, due to changes in the format of DMSII audit file records, it is not possible to perform a Dump Recovery under the Mark VI.1 version of RECOVER/DATA.BASE using audit files created by Mark VII.0 DMSII. Therefore, the only way in which a user can return to the Mark VI.1 release after having processed a data base on Mark VII.0 is to reprccess all updates which had been performed under Mark VII.0 against the Mark VI.1 data base.

9.4.3.
If a data base dictionary resides on a user pack, DMSII does not consider the pack-id to be part of the data base name. Therefore, a user cannot have two data bases open with the same data base name, even though the data base dictionaries reside on separate disk packs. When a program attempts to open the second data base, DMSII will assign the previously open data base to that user, causing Version Errors. If $ NO VERSIONCHECK was set in the DASDL compile, however, the results are unpredictable. This is considered a permanent restriction.

9.4.4.
For sets and subsets which allow duplicates, DMSII does not retain the duplicate occurrences of a key in the order in which the entries were added to the set or subset. If this order is required, a key field must be added to the key specification for the set in order to make such duplicates unique. DMSII also does not guarantee that disjoint data sets and unordered lists (embedded data sets and manual subsets) will be returned in the order in which they were added to the data base.

9.4.5.
Any DASDL option which requires a parameter must be the last option on a dollar record. This includes the TABLESIZE, UPDATE, and INCLUDE options.

2.4.6.
The following options can only be set or reset prior to the first source statement in the DASDL source file (i.e., the first non-dollar statement): CONVERT, FILE, MERGE, NEW, REORGANIZE, SOURCE, SOURCEONLY, STRUCTURE, TABLESIZE, TAPE, UPDATE, and VERSIONCHECK.

2.4.7.
Initial evaluations of the Priority Memory Management algorithms in Mark VII.0 indicate that, wherever possible, DMSII users should have the MPRI option set. With this option set, the length of time which DMSII buffers remain in memory is directly dependent upon the frequency with which those buffers are used; if items such as index tables are repeatedly accessed, they tend to remain in memory. This results in significantly fewer I/O's, both reads and writes, being performed.

2.4.8.
DMSII maintains no upper bound for the number of buffers which can be present at any one time. Whenever a program issues a DMSII request, as many buffers as are necessary to complete that request will be used; any of these buffers which are not locked by the operation are available to be overlayed immediately. Buffers which contain locked records cannot be overlayed until the records are unlocked; if any records within a buffer have been updated, that buffer cannot be overlayed until it has been written to disk. Also, if a data base uses Audit and Recovery, an updated DMSII buffer cannot be written to disk until the audit of the last update within that buffer has been written to the audit trail; this is done to insure that no part of the data base on disk is ever more recent than any of the updates contained within the physical audit trail. Because of this, the amount of memory used by DMSII to perform an update function will be greater if Audit and Recovery is used.

Although DMSII buffers may be candidates to be overlayed (as described above), they will be overlayed by the new Memory Management routines only if there is insufficient available memory to satisfy a request. Because of this, system memory dumps will tend to indicate that memory is saturated with DMSII buffers, regardless of the amount of memory on a particular system or the nature of the jobs running in the mix. This memory saturation is misleading, since the amount of memory dedicated to DMSII buffers will increase or decrease as the memory requirements for non-DMSII functions decrease or increase.

2.4.9.
Prior to Mark VII.0, the memory links describing DMSII buffers did not reference the user to whom the buffers had been assigned. Because of the new Priority Memory Management, DMSII always passes the Limit Register of user programs to the Memory Management routines whenever a buffer is assigned to a user, and that Limit Register is used by Memory Management to place the user's job number in the memory link for the buffer. Because of

this, the response to the CU input message may show that the amount of memory assigned to a user program is higher under Mark VII.0 than it was under VI.1, even though no changes have been made to the program or to the data base. This apparent increase in memory requirements simply reflects a change in the manner in which the memory links are set up, not an actual increase in DMSII memory requirements.

This apparent increase will occur whether or not the Priority Memory Management algorithm is used.

2.4.10.
When using the new Priority Memory Management routines, the MEMORY.PRIORITY assigned to DMSII buffers will be the MEMORY.PRIORITY of the user to whom the buffer is assigned.

2.4.11.
If the user finds it necessary to temporarily change the pack assignment for a data base file, this can be done without an update DASDL compilation. In addition to copying the file to the new pack, the user must "IL" the file each time either DMSII or RECOVER/DATA.BASE attempts to open the file. If the pack reassignment becomes permanent, the user should perform an update DASDL compile, in order to change the pack assignment within the data base dictionary.

Although it was possible to "IL" data base files in previous releases, RECOVER/DATA.BASE would not detect the pack reassignment, and would abort with the message:

```
READ.FILE.HEADER FAILURE
```

10. NDL

10.1. Introduction.

The Mark VII.0 release of NDL includes corrections to outstanding problems, new NDL compiler statements, and enhanced performance when using the MCP Priority Memory Management algorithm. Each of the changes made is documented separately in the paragraphs that follow.

The MCS source files, SOURCE/MCSI and SOURCE/MCSII, are illustrative only. These files are not to be used to compile a running MCS in a normal Data Comm environment. Support is neither intended nor implied.

Temporary documentation describing the NDL/LIBRARY Requests and Controls is supplied as a supplement to this release letter in a printer backup file labeled DOCUMENT/NDLIBRARY.

All references in this section to the NDL Reference Manual refer to the B 1700 Systems Network Definition Language (NDL) Reference Manual (form #1073715, dated 6/77).

10.2. Enhancements.

10.2.1.

The following problems reported in the Mark VI.1 release letter have been fixed (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. The hexadecimal codes for "SI" and "SO" are now correct (9.3.2).

b. Use of the input message "MAKE STATION nn [NOT] READY" no longer causes a READ OUT-OF-BOUNDS when using the DYNAMIC request set (9.3.3.).

10.2.2.

The DYNAMIC request now uses STATION TALLY[0] to count consecutive timeouts for a station, thus allowing more than eight Data Comm lines.

10.2.3.

The NDL compiler now cross checks POLL statements in the control with the AUTOPOLL statement in the LINE section. If the control contains a POLL statement but the LINE section does not contain the AUTOPOLL statement, an error message is issued.

If the control does not contain a POLL statement but the LINE section contains an AUTOPOLL statement, an advisory message is issued. This causes unnecessary space to be allocated for the autopoll buffer.

10.2.4.
The blocking factor of the NDL/LIBRARY file has been changed from 5 to 9 records per block to save disk space.

10.2.5.
The number of disk segments required for the object code file has been reduced from 1000 to 750.

10.2.6.
A QC input message on a Network Controller using RJECTL now properly terminates the Network controller.

10.2.7.
The NDL compiler marks the request and control code segments as "important" (for use with Extended Segment Decay and Priority Memory Management) when a Network Controller is compiled.

10.2.8.
Ampersand control records may now be embedded in the NDL/LIBRARY source file.

10.3. Known Errors and Restrictions.

10.3.1.
If IOLOG is started before a remote file open occurs, the Network Controller aborts with a NAME/VALUE STACK OVERFLOW. To use IOLOG before the first remote file has been opened, increase the VALUE STACK in the Network Controller by 5000 bits.

10.3.2.
It is not possible to run the HASP and RJE3780 programs interchangeably on the same BISYNC line without first bringing the Network Controller to EOJ.

10.3.3.
When an MCS changes a phone number, dashes do not get changed to time delay characters of @FF@.

10.3.4.
Attempting to attach more than one station on a line being autopolled and having a phone number terminates the Network Controller with a READ OUT-OF-BOUNDS error.

10.3.5.
Using an MCS to change "DIAGNOSTIC ON" when no diagnostic request is compiled into the Network Controller terminates the Network Controller with RUN ERROR 180.

10.3.6.
When a program opens two or more remote files, the program is aborted with "MCS ERROR OPEN DENIED". The remote file is not released from the DS-ED job and the station's primary file is not returned to the MCS.

10.3.7.
After a STATUS request from an MCS, the NIF file is opened and stays open, causing memory to be tied up for the NIF file.

10.3.8.
The maximum number of stations that can be defined in a Network Controller is 255.

10.3.9.
The "WARNING: ADDRESS NOT DEFINED" message created in the AUTODYNCTL control when compiling is a result of NUL characters in the poll string. The NDL compiler is not expecting a poll string using NUL characters.

10.3.10.
When using switched Data Comm lines, the CANDEIOTTY request does not answer the second caller on the same Data Comm line.

10.3.11.
When using the SELECTCTD request and a NAK has been received in response to a block of text, the request retransmits the block of text the number of times specified by RETRY plus 1. During this time no other station is selected or polled. When the retry limit reaches zero, the text is discarded without any notification to the operator or the MCS (if present).

10.4. Temporary Documentation.

10.4.1.

The following terminals are now qualified for use with NDL:

Terminal	Direct	Async	Sync
B9350		X	
B9352		X	
B9353		X	
DC110/TU756		X	
DC141/TU500	X	X	
DC141/TU700	X	X	
DC141/TD830		X	X
Redactor I		X	
S1200		X	
TC500	X	X	
TC700	X	X	
TC3500	X	X	X
TC4001		X	
TC4201		X	
TC5110		X	
TC5201		X	
TC5210		X	
TD700	X	X	X
TD800	X	X	X
TD820	X	X	X
TD830	X	X	X
TT102/142	X	X	
TU910/TU500		X	
B80	X	X	X
B800	X	X	X
B700			X
B1800/B1700	X	X	X

INPUT ATTACHED

10.4.2.
The following is a list of terminals and their terminal types:

Type	Terminal
0	B9350 (Teletype)
1	B9353
2	B9352 (Wide)
3	B9352 (Narrow)
7	TT102
10	Redactor I
13	DC110
14	DC140
17	TU500
21	TC500
23	TC700
25	TC3500
26	TC4000
27	TC5100
32	TD700
32	TC4000 (When using alternate transmission numbers of 0 and 1.)
41	TD801
42	TD802
43	TD821
44	TD822
45	TD831
46	TD832
62	B1800/B1700
63	B6700

10.4.3.
The following is a description of the system status variable OUTPUTATTACHED.

To avoid writing a message back to the top of the station queue from which the message was just read (due to a NAK response to a SELECT, for example), the message is left in the station queue as long as possible. This implies that when a line control procedure performs an INITIATE OUTPUT or any other INITIATE which results in output, the message remains in the queue and the request is entered. At this point, OUTPUTATTACHED is false.

When the request executes any of the following statements the message is moved from the station queue to the Network Controller's line buffer (at this point, OUTPUTATTACHED is true):

```
INITIALIZE TALLY (n) Where 0<n<2
INITIALIZE TOG[n]    Where 0<n<7
TRANSMIT TEXT
ATTACH OUTPUT
```

When the request executes any of the following statements, OUTPUTATTACHED is reset:

TERMINATE OUTPUT
TERMINATE OUTPUT (RETURN)
TERMINATE RELEASE STATION
TERMINATE DISCONNECT
TERMINATE ECHO
TERMINATE ERROR
TERMINATE INPUT
~~DETACH OUTPUT~~ — gen
QUEUE OUTPUT
Term NOINPUT

10.4.4.
The following is a description of the LINE CONTROL KEY when the value is 5.

The control is entered with an output message queued and:

a. The line is not currently connected, and
b. An ACU adapter is available to connect the line as indicated by a strap in the regular adapter, and
c. A telephone number is provided.

10.4.5.
The Declaration Section has been enhanced to allow for global defines. The following is a description of the DEFINE syntax:

Syntax:

<DEFINE STATEMENT> ::= DEFINE <DEFINE DECLARATION LIST>.

<DEFINE DECLARATION> ::= <DEFINE DECLARATION> |
<DEFINE DECLARATION>,
<DEFINE DECLARATION LIST>.

<DEFINE DECLARATION> ::= <DEFINE ID> = #<DEFINITION>#

<DEFINE ID> ::= <IDENTIFIER>

<DEFINITION> ::= <TEXT>

<TEXT> ::= Any sequence of characters except a # which is not contained within a character string.

Semantics:

Beginning with the first REQUEST section, any <DEFINE ID> found by the compiler is replaced by its <DEFINITION>. Nested DEFINE statements are permitted with no restriction on the number of levels.

Restrictions:

a. <DEFINE ID> may not be an NDL reserved word, or an <IDENTIFIER> which is already a constant. <DEFINE ID> cannot be redefined.
b. Control ($) records must not contain defined symbols.
c. Any control record detected while processing a DEFINE statement is ignored.
d. Comments (%) are allowed within the <DEFINITION>, but are discarded during processing.
e. A <DEFINE> must not define itself.
f. A <DEFINITION> must not exceed 327 characters.
g. Parametric defines are not allowed.
h. Nested defines are expanded without regard to the number of levels until the total size of the expansion and remaining source image exceeds approximately 400 characters.
i. All DEFINE statements are global.

Examples:

```
DEFINE CHECKRETRY = #IF RETRY GT 0 THEN
                        DO.
                           RETRY := RETRY - 1.
                           TERMINATE NOINPUT.
                        END.#

DEFINE REQUESTHRU = # LINE (CONTRL KEY) = 0. #

DEFINE JINX       = # RETRY = 0. #
```

10.4.6.
The INPUT MESSAGE column of table 3-3 (Terminate Options) on pages 3-44 and 3-45 of the NDL Reference Manual should be changed to "Lost" for the following TERMINATE options:

```
ERROR
NOINPUT
OUTPUT
OUTPUT(RETURN)
```

The OUTPUT MESSAGE column of table 3-3 should be changed to "Lost" for the following TERMINATE options:

```
INPUT
OUTPUT
OUTPUT(RETURN)
```

10.4.7.
A new attribute has been added to the Station Section.

STATION READY Statement

Syntax:

<STATION READY STATEMENT> ::= READY = <LOGICAL VALUE>

Semantics:

If <LOGICAL VALUE> is FALSE, it allows the Network Controller to complete the BOJ functions but does not allow it to poll this station. READY defaults to TRUE if this statement is omitted from the <STATION DEFINITION>.

10.4.8.
The VARIANT in the DATA MESSAGE on page 4-6 of the NDL Reference Manual should read as follows:

5 Retain the text on a GOOD-RESULTS-REPLY message.

10.4.9.
The JOB-NO in the DATA MESSAGE on page 4-7 of the NDL Reference Manual is replaced by TIME, and is the time at which the Network Controller started processing the message.

10.4.10.
Two error values have been added to the DATA MESSAGE ERROR field on page 4-7 of the NDL Reference Manual, as follows:

Value	Meaning
12	Exception Occurred
13	ACU Adapter
14-15	Reserved

10.4.11.
Two additional fields, PROTOCOL-TYPE and SESSION, have been added to the OPEN and OPEN-REPLY messages on page 4-10 of the NDL Reference Manual. These two fields replace the FILLER statement between FILE-NAME and STATION-LIST. The following table illustrates their positions and field lengths:

	NC-MCS		MCS-NC		
Field Name	W	R	W	R	PIC
FILE-NAME	*	*			X(10)
PROTOCOL-TYPE	*	*			99
SESSION	*	*			9999
STATION-LIST	*	*	*	*	(999)#

The semantics for PROTOCOL-TYPE and SESSION are as follows:

PROTOCOL-TYPE indicates the type of remote file intercommunication desired by the user program opening the file. The defined values are:

00 - Input messages are TYPE 01.
01 - Input messages are TYPE 50 or greater.

SESSION is the remote session number associated with the user program performing the open. SESSION is equal to 0000 if there is no session association.

10.4.12.

The STATUS and STATUS-REPLY MCS messages have been expanded. The additional fields follow the REMOTE-FILE-HAS-HEADERS field, currently defined on page 4-25 of the NDL Reference Manual. The new formats of the STATUS and STATUS-REPLY messages are as follows:

	MCS-NC		NC-MCS		
Field Name	W	R	W	R	PIC
STATION-PHONE	-	-	*	*	X(20)
STATION-VALID	-	-	*	*	9
STATION-LINE-NO	-	-	*	*	99
STATION-SECONDARY-FILE-NO	-	-	*	*	999
FILLER					9(10)
LINE-COUNT	-	-	*	*	99
LINE-INFO	-	-	*	*	(999)#
LINE #99					
ACU #9					

The semantics for the STATUS and STATUS-REPLY messages are as follows:

STATION-NUMBER is the telephone number assignment for the station.

STATION-VALID is a field in the station table which is set to 1 if the station is valid; otherwise, it is set to 0.

STATION-LINE-NO is the current line assignment for the station.

STATION-SECONDARY-FILE-NO is the remote file number of the station's secondary file. If the remote file number is 000, there is no secondary file.

LINE-COUNT is the number of lines on which the station is defined.

LINE-INFO is a number of three-character fields describing the line number (LINE) and its associated DIALOUT status (ACU).

10.4.13.
A new CHANGE-TYPE has been added to the CHANGE and CHANGE-REPLY messages on page 4-28 of the NDL Reference Manual, as follows:

CHANGE-TYPE	Field To Be Changed	Format
12	STATION-PHONE	X(20)

The CHANGE-RESULT should read as follows:

CHANGE-RESULT returns a 1 if the CHANGE was correct. A 0 indicates that the LSN field was invalid. A 2 indicates that the CHANGE-TYPE was invalid.

10.4.14.
An MCS can now do a Remote File Info operation on any remote file. The Remote File Info has been expanded and an additional capability added.

The format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY messages is as follows:

	MCS-NC		NC-MCS		
Field Name	W	R	W	R	PIC
MESSAGE-TYPE	+	*	*	*	99
JOB-NO			*	*	9(7)
TIME			*	*	9(7)
REMOTE-FILE-NO	+(x)	*	*	*	999
OUTPUT-MESSAGES-QUEUED			*	*	9(4)
INPUT-MESSAGES-QUEUED			*	*	9(4)
CURRENT-STATION			*	*	999
OTHER-RF-REQUEST	*	*			9
OTHER-RF-ERROR			*	*	9
OPEN-APPROVER-RF-NO			*	*	99
FILLER					99
LSN-LIST			*	*	(999)#
OUTPUT-QUEUED-LIST			*	*	(999)#

(x) Mandatory only if OTHER-RF-REQUEST is a 1.

The semantics for the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY messages are as follows:

MESSAGE-TYPE is a field which identifies this as a REMOTE-FILE-INFO (28) or a REMOTE-FILE-INFO-REPLY (29).

JOB-NUMBER is the job number of the job making this request.

TIME is the time the reply was sent.

REMOTE-FILE-NO is the number of the remote file of this program if OTHER-RF-REQUEST is set to 0. It is the number of the target remote file if OTHER-RF-REQUEST is set to 1.

OUTPUT-MESSAGES-QUEUED represents the total number of messages queued for all stations attached to the remote file.

CURRENT-STATIONS represents the number of stations attached to the remote file.

OTHER-RF-REQUEST is 0 if the request is for the writing MCS's file. It is a 1 if REMOTE-FILE-NO contains the file number about which information is requested.

OTHER-RF-ERROR is 1 in the REMOTE-FILE-INFO-REPLY message if the requestor set OTHER-RF-REQUEST to 1 and the REMOTE-FILE-NO supplied was non-existent.

OPEN-APPROVER-RF-NO is the remote file number of the MCS which approved the requestor's open.

LSN-LIST has an entry by LSN for every current station.

OUTPUT-QUEUED-LIST has the number of output messages queued for each of the current stations. The total should agree with OUTPUT-MESSAGES-QUEUED.

10.4.15.
The following is a description of USER MESSAGES.

An executing MCS can interface with a user remote file without headers through the record format shown below:

	MCS-USER		USER-MCS		
Field Name	W	R	W	R	PIC
MESSAGE.TYPE	*	*	*	*	99
FILLER	*	-	-		9
LSN	*	*	*	*	999
TEXT.SIZE	*	*	*	*	9999
TEXT	*	*	*	*	X(TEXT.SIZE)

The semantics of the fields of the USER-DEFINED message record are as follows:

MESSAGE.TYPE is established by the user program and must be a number greater than 49 and less than 100.

LSN is the Logical Station Number to which the data belongs.

TEXT.SIZE is the number of characters in the text field.

TEXT is the character string which is being passed.

10.4.16.
A new $ control option, LIBINFO, has been added to the NDL compiler. When the LIBINFO control option is included in an NDL compile, a list of all requests and controls in the NDL/LIBRARY file is printed on a separate page at the end of the compile listing.

10.4.17.
The description of the $ LIBRARY control option should read as follows:

LIBRARY <identifier>	The NDL source code specified by the <identifier> is retrieved from the NDL/LIBRARY file and inserted in the user's program following the $ LIBRARY option.

10.4.18.
The maximum allowable Network Information File (NIF) table sizes have been increased from 16K bytes to 40K bytes.

10.4.19.
The word DEFINE is now a reserved word.

10.4.20.
The DYNAMIC requests and control have several advantages over the AUTOPOLL requests and control, as follows:

a. Stations that are not on line (powered off) are deleted from the autopoll string.
b. If output cannot be sent because the terminal is in TRANSMIT or LOCAL, all terminals are autopolled once before the terminal is selected again.
c. When an output message is sent, a FAST-SELECT is used the first time. If the FAST-SELECT fails, a normal SELECT is done to conserve line time.
d. The control is designed to handle groups and switched lines.
e. If a terminal has output queued for it and the terminal is powered off or taken off line, the control stops trying to send output until the control can poll the terminal again.

Reconfiguration:

The control does an automatic reconfiguration every three minutes. The 3 minute figure should be increased for large networks or the control may stay in reconfiguration mode continously. Reconfiguration is accomplished by putting previously off-line terminals on the end of the poll string so as not to interfere with active terminals.

To change the amount of time before an automatic reconfiguration occurs, two source statements must be changed in the AUTODYNCTL. The first statement is at sequence number 04505000 and is as follows:

```
IF LINE(QUEUED) OR LINE(TALLY[3]) = 6 THEN%
```

The constant 6 must be changed using the following formula:

```
NUMBER OF MINUTES BEFORE RECONFIGURATION * 2
```

The second statement is at sequence number 04535000 and is as follows:

```
IF TIME - TIME(TALLY) GT 1800 THEN%
```

The constant 1800 must be changed using the following formula:

```
NUMBER OF MINUTES BEFORE RECONFIGURATION * 600
```

Once a terminal is active, the terminal is not removed from the autopoll string unless the terminal times out 3 consecutive times.

The length of the autopoll buffer declared in the Line Section must be 5 times the number of stations on the line plus 5.

10.4.21.
When using the AUDIT statement in NDL to audit messages for recovery, the VALUE stack size must be increased. First, multiply the largest record to be audited by 2; then convert to bits by multiplying by 8. Add this figure to the VALUE stack size given by the NDL compiler for the Network Controller.

10.4.22.
To decrease the overhead of the IOLOG on the Network Controller, modify the Network Controller as follows:

```
MODIFY <network controller name>
FILE IOLOG BUFFERS=2 RECORDS.BLOCK=<integer>;
```

where <integer> is the largest message received or transmitted, divided by 131, plus 1. This modification has no effect on the memory requirements of the Network Controller unless data comm activity is being audited (that is, the IOLOG option is set).

10.4.23.
MAX TALLY must be set in the Declaration Section to the number of lines plus 1 when using the DYNAMIC requests and control.

10.4.24.
When retries becomes 0 and just before the station is temporarily deleted from the autopoll list in CANDEPOLTD, the station is single polled once to check that it is timing out consistently and not just intermittently. If EOT or SOH is received, the station is assumed active and is not deleted from the autopoll list; otherwise, the station is removed from the autopoll list.

10.4.25.
If program switch 9 is set to a value other than zero in the Network Controller, the file STATION.SAVE remains open. This file is used for a number of functions, especially when running with an MCS, and must be opened and closed each time it is referenced. In large networks, where ATTACH/DETACH, STATUS, and other functions are frequently requested, it may be advantageous to specify that the STATION.SAVE file remain open constantly in order to remove the MCP overhead of many OPEN and CLOSE requests. The memory required by the STATION.SAVE file when it is open is approximately 600 bytes.

10.4.26.
A participating MCS that approved a remote file open can detach a station from that remote file.

10.4.27.
With the Mark VIII.0 release NDL, an MCS will no longer be able to change STATION (ENABLED).

11. CANDE

11.1. Introduction.

Included as part of the Mark VII.0 software is the B 1800/B 1700 Command and Edit (CANDE) language. B 1800/B 1700 CANDE provides generalized file preparation and updating capabilities in an interactive, terminal-oriented environment.

B 1800/B 1700 CANDE is a Message Control System (MCS) that runs in conjunction with the B 1800/B 1700 NDL system. The NDL-generated Network Controller performs all the functions related to data communications, while CANDE performs file updating and text-editing functions.

The B 1700 CANDE User's Manual (form #1090586, dated 8-76) is available through normal distribution channels. New features not yet included in the manual are described in the paragraphs that follow.

11.2. Enhancements.

11.2.1.

The following problems reported in the MARK VI.1 release letter have been fixed (the numbers in parentheses refer to the paragraphs in the Mark VI.1 release letter which originally described the problem):

a. CANDE no longer reports a sequence error when trying to list an empty file (21.3.1).

b. CANDE does not reject a filename that begins with an integer or contains a hyphen if the filename is enclosed in quotes (21.3.2 and 21.3.3).

c. CANDE will display a message at log on time that a default pack is not on line. At this time the MCP will set the override bit "ON" for that usercode. When the pack is brought on line the user can reset the override bit with the MCP RV input message (21.3.4).

d. The ability to create a unique filename has been added to the WRITE command (21.3.6). For example:

```
WRITE <filename> AS <filename>
```

e. After saving a "DATA" file, CANDE now allows the user to get another file or list a backup file without resequencing or removing the work file (21.3.7).

f. A new option, "OVERRIDE", has been added to the "RESEQ" command to properly resequence a file with blank sequence numbers (21.3.8).

g. The "REPLACE LIT / / //" command no longer causes CANDE to loop (21.3.11).

h. CANDE displays the proper error message when the "TITLE" command is entered in the form of <name-1>/<name-2> (21.3.12).

i. CANDE uses the correct pack-id when the "RUN" command is used from a user pack (21.3.13).

11.2.2.
When listing printer backup files with CANDE, the second line is not displayed if it contains all blanks.

11.2.3.
Starting with the Mark VII.0 release of CANDE, a GET or MAKE on a file with a type of SDL assumes the underscore as the identifier break character. UPL files continue to use the period as the identifier break character.

11.2.4.
CANDE automatically logs a terminal off when the retry count in the Network Controller has been exceeded. The Network Controller, beginning with Mark VII.0, resets the timeout counter each time a valid reply is received on a line. This will help to increase the time before CANDE automatically logs a terminal off. The time can also be increased by setting the RETRY value in the Network Controller to a larger number.

11.2.5.
The maximum message size allowed with the "?SS" command has been increased to 240 characters.

11.2.6.
The maximum number of terminals that can sign on to CANDE at the same time has been increased to 64.

11.2.7.
CANDE now accepts 10-character filenames, with the following exceptions.

a. The family-name is still limited to seven characters when CANDE is executed with a usercode.

b. The COMPILE, EXECUTE, RUN, and WRITE verbs may only allow up to nine characters when it is necessary for them to concatenate the filename with an additional character to form a new identifier.

11.2.8.
Keywords may now be entered in lower case; however, CANDE no longer accepts a usercode/password in lower-case letters during the log-on operation.

11.2.9.
MCP responses to "?" commands have been increased to a maximum of 250 characters.

11.2.10.
A new command, AUDIT, has been added to CANDE to allow the auditing function to be dynamically started or stopped. Refer to paragraph 11.4.13 for complete documentation.

11.3. Known Errors and Restrictions.

11.3.1.
A margin number is ignored on a single "SEQ" entry.

11.3.2.
If CANDE is executed under a usercode and the system fails for some reason, CANDE must be re-executed under the same usercode in order to recover the files that were being used when the system failed.

11.3.3.
When 80 or more carriage return characters are entered from a TTY device, CANDE will terminate with an "INVALID SUBSTRING" error.

11.3.4.
CANDE will be DS-ed if disk errors are encountered while saving a file.

11.4. Temporary Documentation.

11.4.1. Usercodes and File Security

If executed under a privileged usercode/password, CANDE uses the File Security mechanism implemented in the MCP. As a result, users must be aware of the following items:

a. Usercodes are delimited by parentheses and, in the case of CANDE, are limited to seven characters in length. Log-on is accomplished by entering the usercode without parentheses.

b. Usercodes and passwords may have a default pack associated with them. If a default pack has been defined, CANDE reads and writes all files on this pack. The default pack may be overridden by explicitly specifying a pack-id with the "ON" attribute.

c. All files saved through CANDE are saved as "PUBLIC" or "PRIVATE", according to the default protection specified in the (SYSTEM)/USERCODE file. PRIVATE files can be accessed only by a program running under the same usercode as that of the file or by running with a "privileged" usercode/password. The security of a file can be changed by using the MCP MH input message.

d. CANDE has a facility for two-way communication with the MCP. Messages may be zipped to the MCP from the remote terminal. Responses to these messages are directed back to CANDE, which then routes them to the requesting terminal. For example:

?WY	Returns the status of only that user's job(s).
?TD	Returns the current time and date.
?MX	Returns the mix for only that user's job(s).
?MO	Modifies only files belonging to that user.
?RE (<usercode>)/<filename>	Removes the file only if the usercode in the filename is the same as the usercode under which the user is running.

e. CANDE can list, on the terminal, printer backup files which have been created by jobs executed through CANDE. Backup files are locked in the disk directory with a name of the form "(<usercode>)/#<integer>". The command "LIST #<integer>" displays the backup file on the screen for review. If CANDE is not executed under a usercode, the facilities of backup file naming, file security, and MCP communication are not provided.

CANDE Execution

To bring up the CANDE system under File Security, a PUBLIC, privileged usercode/password pair that does not have a default pack associated should be selected from the (SYSTEM)/USERCODE file. This enables CANDE's recovery, tank, and workfile to be placed on system disk for easier maintenance. To execute CANDE, enter:

```
USER=<usercode>/<password> EXECUTE CANDE
```

The MCP automatically executes the Network Controller, if one is present in the "C" entry of the NAME TABLE. The Network Controller is placed in the NAME TABLE by using the MCP CM input message.

To request CANDE to zip-execute a Network Controller named CANDE/HANDLER (if a Network Controller is not identified in the NAME TABLE), CANDE must be executed with switch 0 set to 1 (SW0=1), as in the following example:

```
USER=<usercode>/<password> EXECUTE CANDE SW0=1
```

If the Network Controller was executed by the MCP from the "C" entry in the NAME TABLE, and the last remote file "CLOSE" has been processed, the MCP will queue a "QC" for the Network Controller after three "N.SECOND" intervals (if no intervening remote file opens occur).

NOTE

CANDE may be executed without a usercode. If this is done, CANDE does not provide file security or MCP communication.

Disk Pack Defaults

Since the MCP forces files to the default pack associated with the usercode/password, syntax has been implemented to access files on system disk if a default pack was specified in the (SYSTEM)/USERCODE file. The following table applies to all commands allowing filenames, except for "SAVE".

Enter	Pack-id	Family-name	File-id
A	<default>	(UC)	A
A/B		A	B
A/B ON C	C	A	B
(UC)/A	<default>	(UC)	A
(UC)/A ON C	C	(UC)	A
*(UC)/A		(UC)	A
*A		A	

A file-identifier may not exceed 10 characters in length, including the "*", "(", and ")" characters. "*(ABCDEFGH)", for example, is invalid.

An asterisk entered preceding a filename denotes that the file is on system disk, and CANDE looks for a literal name following the asterisk. To access another user's file, that file must be changed from "PRIVATE" to "PUBLIC" by the owner or be accessed by a privileged usercode. When saving a file, it cannot be saved under a different usercode. Thus, the command "SAVE AS *(<usercode>)/A" is illegal; "SAVE AS *A" will save the file as "(<usercode>)/A" on system disk. A command such as "SAVE AS A" will save the file on disk as "<default pack>/(<usercode>)/A".

Workfile

CANDE can be directed to read any file to which the user is allowed access. However, changes may be made only to one file, the "workfile". The user gets a new workfile by doing a "MAKE", or may recall an existing workfile by doing a "GET". Additions and corrections may be performed on the workfile by single-line entries or by using the editing commands.

A CANDE workfile consists of two parts. The "tank" (which is invisible to the user) is where additions and edited lines are placed until an "UPDATE" takes place. The tank contains a maximum of 32 entries. If an explicit "UPDATE" has not been issued by the time the tank is full, an automatic update is done to merge the contents of the tank with the workfile. An update incorporates all additions and corrections to the workfile, creating a new copy of the workfile and an empty tank. An update does not make the workfile a permanent file on disk. A "SAVE" is required to make the workfile a permanent file and to discard the original file. A "SAVE" creates a new file on disk with an empty tank and workfile.

The second part of the workfile consists of n+1 blocks within CANDE's master workfile (named "CANDE/WORKFILE"). The n is the number of contiguous blocks (each 64 records long) needed to contain all currently changed records, including deletes. Note that CANDE never writes to the original source file. The file is only changed at "SAVE" time.

Record Modification

If a file is present (a "GET", "MAKE", or "RECOVER" has been processed), the user is assumed to be in a state of editing the file. Any input consisting only of records having valid sequence numbers with no sequence errors is treated as records to be entered into the file. After the input is processed, a "NEXT" is performed automatically, meaning that the next "PAGE" is displayed, beginning with the record that follows the last one entered.

"PAGE", "LIST" (except from an external file), "BACK", "SAME", "NEXT", "-", and "+" explicitly request a page to be displayed. For CRT devices, records may then be edited and the page (or partial page) transmitted back. For non-CRT devices, the records must be retyped.

For recovery purposes, CANDE does not clear the CRT screen at log-on time. The user may recover the file and transmit to the system any records left on the screen at the time of the failure.

The buffer used by the "PAGE" system, located in dynamic memory in Mark VI.1, is now part of static memory (in the VALUE STACK). This buffer is used to "roll in" the last output in order to

compare the current input against it. This comparison allows CANDE to add to the tank only those records which have changed, and is required to prevent writing unnecessary records to a file when a "SAVE: PATCH" command is entered. It also makes input processing more efficient.

The comparison occurs on a record-by-record basis as long as the sequence numbers match. If a sequence number is encountered in the input which does not exist in the last output copy, then comparison ceases and the record is added. Processing continues in this fashion until either the sequence number of an input record is greater than or equal to that of the last output record (at which time comparison ceases), or until all input records have been processed. This prevents unnecessary error reports or "ADVISORIES". Also, synchronization of record comparison is maintained and unchanged records are not added to the tank. If the page of input returned to CANDE is missing any records initially displayed (that is, the user has performed a "LINE DELETE" on the terminal), the missing records are not automatically deleted from the workfile.

Line Deletion

Single lines may be deleted through use of the "VOID" command. A "V" must be the first character of the line and the line must contain a valid sequence number. Any number of blanks, or none at all, may separate the command and the sequence number.

A delete record will be queued for later processing and input processing will continue. If the sequence number is in error, the record will be treated as a sequence error.

NOTE

> This form of deletion may be time-consuming, because each deletion request will cause an update. No optimization of successive, contiguous "VOIDS" is attempted.

File Positioning

CANDE allows the user to move backward and forward throughout the file, once "PAGE" or "LIST" is invoked. "PAGE" or "LIST" in any form causes the file "position" pointers (for "BACK", "SAME", "NEXT", "-", or "+") to be set to the requested position. For CRT devices, after displaying a page, "NEXT" is written at the home position leaving the cursor immediately after it. Thus, transmission without moving the cursor will effect "NEXT" automatically. Any other string transmitted in the first line is treated as a command input. If transmission includes more than the first line, it is assumed to be input records. The first line is ignored and the input is processed.

File positioning is controlled through the following commands:

BACK <n>	Display current -<n>th page.
SAME	Display current page again.
NEXT <n>	Display current +<n>th page.
-<n>	Display page at current -<n>th record.
+<n>	Display page at current +<n>th record.

After a "GET", "MAKE", or "RECOVER", the above commands are operative. If <n> is not specified, a value of one (1) is assumed.

"+<n>" and "-<n>" shift exactly <n> records in the direction requested. "BACK <n>" and "NEXT <n>" shift approximately <n> pages in the direction requested. In this way, "BACK" and "NEXT" may be used to position the file to approximately the desired area quickly without reading through each record on the way from the current position to the requested one. Once in the approximate area, the position may be adjusted accurately by using "+<n>" and "-<n>". The range of records shifted is:

$$1 <= r <= 131071$$

where r=n for "+" and "-", and r=n*pagesize for "BACK" and "NEXT".

Recovery

Recovery of a user's workfile is necessary when CANDE is executed after an abnormal termination or when the user's terminal incurs consecutive timeouts. The recovery procedure involves creation of the recovery file and communication of this fact to the user during log-on. The recovery file is named "<usercode>/recovery<ab>". <ab> is a unique 2-character sequence created as follows:

a = (<usercode index>/36)th character of atom (see below)
b = (<usercode index> mod 36)th character of atom

where atom = ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.

For example, the character "A" is the 0th character of atom, and the character "9" is the 35th character of atom.

This scheme produces a unique recovery file for each usercode/password pair. To avoid loss of the file upon recovery, users should not intentionally create files named in this manner.

After log-on each user who was on the system when it aborted has the option of recovering the workfile. The user must enter "RECOVER" to get the previous workfile. If the user does not wish to recover, then another session may be started. When a system failure occurs, it is possible to lose up to four records in the tank file.

Displays and Errors

"TERMINAL PAGE" sets the page size as requested. See paragraph 11.4.10 for the default page sizes depending on terminal type.

All displays to a device having a page size greater than one begin on the second line of the screen and end with the column indicator on the last line.

Any line whose first token is a valid CANDE command is taken as a new command and executed after the records to that point have been processed.

All numerics in sequence fields indicate to CANDE that after input is processed, the next page should be displayed.

Sequence errors are defined as either:

a. Numeric but out of sequence, or

b. Beginning with a non-numeric character which is not a valid CANDE verb or a "V" (refer to Line Deletion).

The action taken in such a case is to leave the screen alone and notify the user on the top line of the screen.

Inserts are defined as numbers that are in sequence but which were not put on the screen by CANDE. The action taken is to add the record(s) to the tank and to discontinue comparison against the copy until the next copy sequence number is encountered.

For devices whose page size is one, the following operating characteristics are valid:

a. The "BREAK" key (teletype only) clears the output queue and terminates the operation.

b. When "SEQ" displays an existing record, CANDE sends a CR (Carriage Return), LF (Line Feed), and re-types the sequence number, leaving the carriage immediately after the number to allow the line to be changed and re-entered. If only an ETX or CR is returned, the record is not changed and the command continues. If it was not an existing record (only a new sequence number), then returning only an ETX or CR terminates the command.

c. "PAGE" treats displays as noted above for "SEQ".

d. "LIST" is continuous, rather than paged.

e. "BACK", "SAME", "NEXT", "-", and "+" are operative.

f. "V" is not operative.

CRT-type Devices

Paged output to a CRT-type device is preceded by a "HOME and CLEAR" character and has the CR character appended after the rightmost non-blank character in each line in order to decrease transmission time. If the line is blank, it consists of a CR character only. Single-line output to a TD820/830 series device has the LC (CLEAR to END of LINE) character appended to it, in order to clear the remainder of the line.

Scrolling

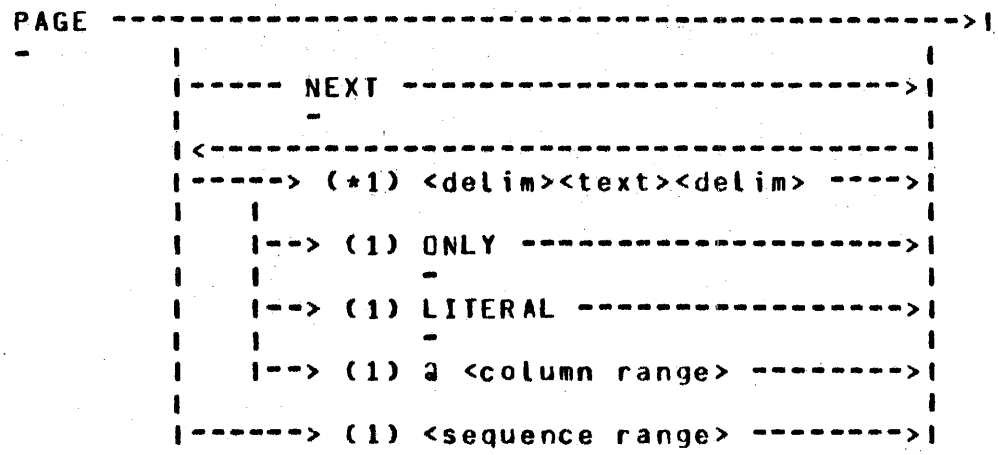
The NDL CANDE library requests (CANDEPOLTD, CANDEFSLTD, and CANDESELTD) contain a scrolling capability available to users running CANDE and to those running an application program under CANDE. Scrolling is disabled by default. Enter "?+" to enable scrolling and "?-" to disable scrolling.

All output (except full page output produced by "LIST", "PAGE", and "SEQ") is written one line at a time at the bottom of the screen after shifting the screen up one line (the top line is deleted). Any input except paged input is scrolled to the bottom of the screen.

11.4.2. PAGE

The "PAGE" command displays full screens of information at the terminal for editing, and accepts full screens of input from the terminal. The operation continues until the command parameters have been satisfied or any command is entered on the top line of the terminal.

Syntax:



<delim><text><delim>	Scan each record within the sequence range for the delimited text. Display page starting with record in which text is found.
<sequence range>	Begin at S1, terminate after S2. S1 alone implies S2 = end. Default = BOF-EOF.
ONLY	Display only records with target text.
LITERAL	As in REPLACE, FIND.
a	As in REPLACE, FIND.
NEXT	Resume action requested by previous "PAGE" command.

Example:

Assume that the page size=2 and the file contains the following records:

```
100   ABCDEF
200   GHIJKK
300   LMNOPQ
400   RSTUVW
500   XYZABCD
600   EFGHIJKK
700   LMNOPQ
```

Command	Response
PAGE ONLY /KK/ LITERAL	Display 200, 600, terminate.
PAGE 100	Display record 100, terminate.

For CRT devices, "NEXT" is written at the home position. For text searching, "NEXT" means begin searching with the next record after the one first displayed (if "ONLY" is not specified) or the last one displayed (if "ONLY" is specified).

NOTE

The meaning of a single sequence number in a sequence range specification now differs between "LIST" and "PAGE". "LIST <seq>" displays the single record at sequence number <seq>. "PAGE <seq>" displays a page of records, the first of which is the record at sequence number <seq>.

11.4.3. GET

The "GET" command has been changed to default to the "NO CHECK" option and has added the asterisk ("*") prior to the filename. When "*" is specified, only the literal name following the "*" is looked for in the disk directory. This allows a "GET" on a single-name file. If the "*" is used in front of a two-name file, it indicates the file is on the system disk.

11.4.4. LIST and SAVE

The "LIST" and "SAVE" commands have added the asterisk "*" to specify that the file is on system disk, and that the pack-id in the usercode file does not apply.

11.4.5. WHAT

The response to the "WHAT" command has added the following information:

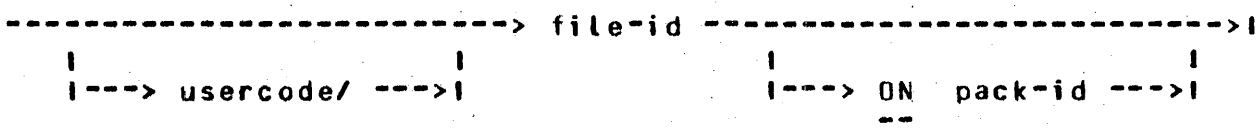
Source file size
Tank & workfile size
Source file pack-id
Patch file-id

11.4.6. Filename

A filename is used to identify a file in the B 1800/B 1700 system. When a file is created through CANDE, the user supplies the file-id (an alphanumeric identifier of nine or fewer characters with the first character being non-numeric) and CANDE adds the usercode to form the complete filename (for example, <usercode>/file-id).

The user may access any file under his usercode by simply specifying the <file-id>. The user may also access PUBLIC files belonging to another user in a read-only mode, by specifying the <usercode>/<file-id> of the file being requested. Users may also access files on user packs or cartridges by specifying "ON <pack-id>" after the <file-id> or <usercode>/<file-id>.

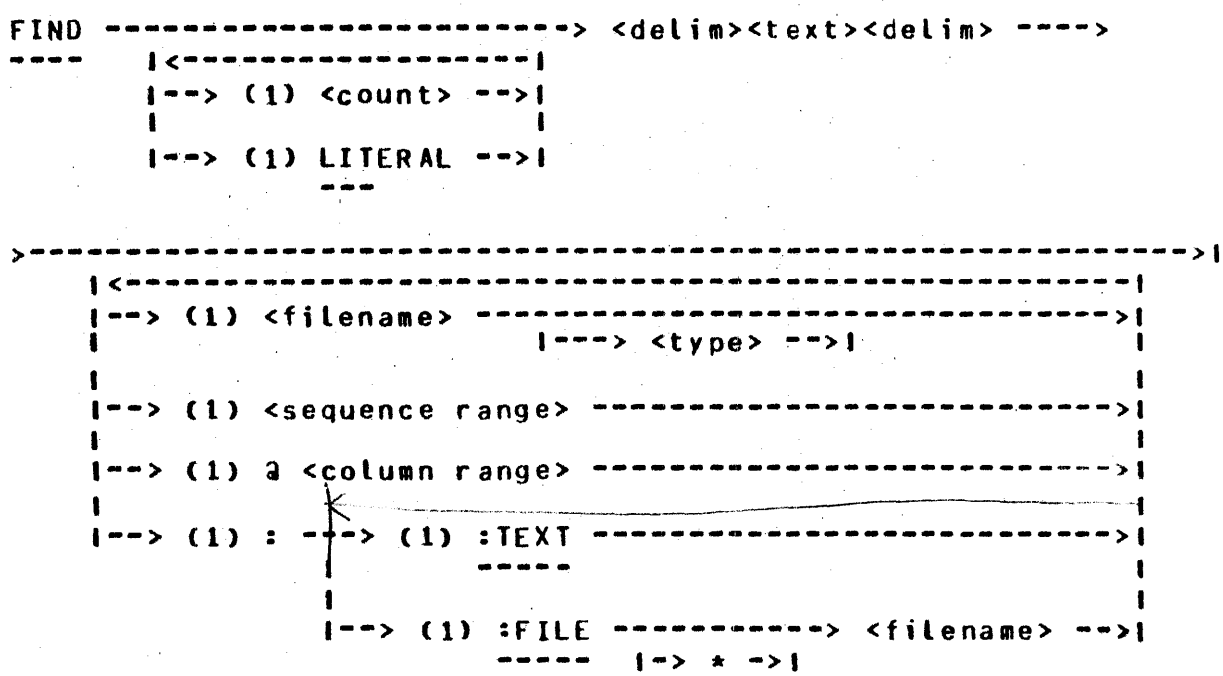
Syntax:



11.4.7. FIND

The "FIND" command searches a file for appearances of specific text. Output indicating the result of the search may be directed to the terminal. The output of a "FIND" is prompt-driven. This means that the command fills a screen to its page size; the user must then transmit a character to get the next page of output. A "?BRK" breaks the output as soon as possible.

Syntax:



Semantics:

Only one target may be sought with a "FIND". The delimiters surrounding the target <text> may be any delimiter. The <text> may contain any characters except the delimiter. The text field may not be empty. The text field is treated as a token string unless "LITERAL" is specified.

The search is successful whenever the string of characters in the <text> is found in a line of the file.

If the integer <count> appears, the search for the associated <text> is terminated after it has been found the specified number of times in the file.

If a <filename> is not specified, the workfile is searched by default. Specification of a <filename> causes searching of the designated file, which may be any file the user is allowed to read.

A <type> may be specified to give the location of sequence numbers in the records to be searched. If no <type> is specified and the user does not have a workfile, a type of "SEQ" is assumed. Otherwise, the file type will default to that of the workfile.

If a sequence range is specified, only records falling within that range are searched.

A column range, if provided, indicates that only a specified part of each record is to be examined. Note that a <column range> cannot fall within the sequence-number field (for example, columns 73-80 for SDL files).

The defaults for <sequence range> and <column range> are the entire file and the entire record, respectively.

The option ":TEXT", if specified, displays the entire line containing the <text> on the terminal. Otherwise, only the sequence numbers of those records containing the string are displayed.

The option ":FILE <filename>" directs output from the "FIND" to a disk file labeled "(<usercode>)/<filename>". The user may not specify a usercode with this option; his own usercode is used as the family-name. If the option ":FILE *<filename>" is used, the file is forced to the system disk and is not allowed to go to the default pack associated with the usercode.

Examples:

```
FIND / fileid / 100-900 :TEXT
```

Search the workfile for the occurrence of " fileid " within the records numbered 100-900 and output any lines found.

```
FIND 10 / STUFF /
```

Find the first ten occurrences of " STUFF " within the workfile and output the sequence number of the records found.

By entering RECOVER after logging on, the user can recover his workfile.
RECOVER & :PURGE allows the user to remove a recovery file which he does not want to recover.

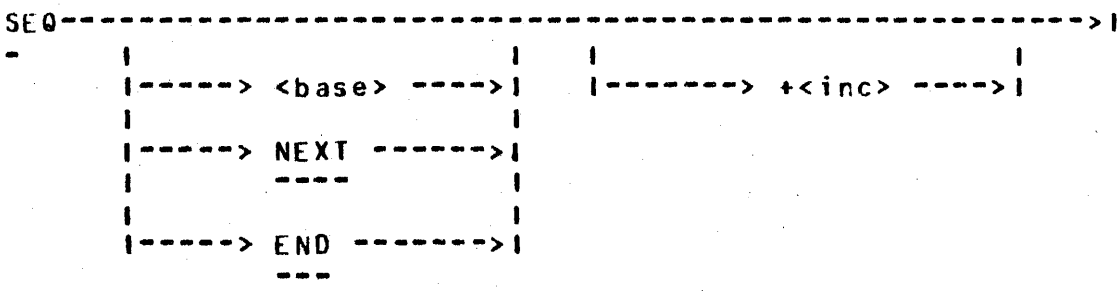
11.4.8. RECOVER

The "RECOVER" command has added the ":PURGE" option. This simplifies the removal of recovery files left on disk.

11.4.9. SEQ

The "SEQ" command invokes automatic sequence mode, causing CANDE to provide the sequence number for each new line. With the exception of "@" (margin) and "?" (control command), no commands are recognized in automatic sequence mode.

Syntax:



NEXT Begin sequence numbers with last output +<inc> from previous SEQ.

END Begin sequence numbers with last record sequence.

Semantics:

An initial value is assigned as the sequence number of the next line to be entered; the value is incremented for each subsequent line. Values are specified by an integer <base> and <inc> or by defaulting to values of 100 for <base> and <inc>. If "NEXT" is specified, sequence numbering begins where the last sequence number, for that user, was terminated.

Sequence mode may be terminated by transmitting back only the sequence number.

"SEQ" displays existing records as well as the newly-created sequence numbers. Display begins with the last record from the previous input page if the page size is greater than one. "BACK", "SAME", "NEXT", "-", and "+" terminate the "SEQ" mode and display the requested page. For CRT devices, returning only the sequence number of the first record terminates the "SEQ" command. Blank records are added to the file. Margin is not applied.

Example:

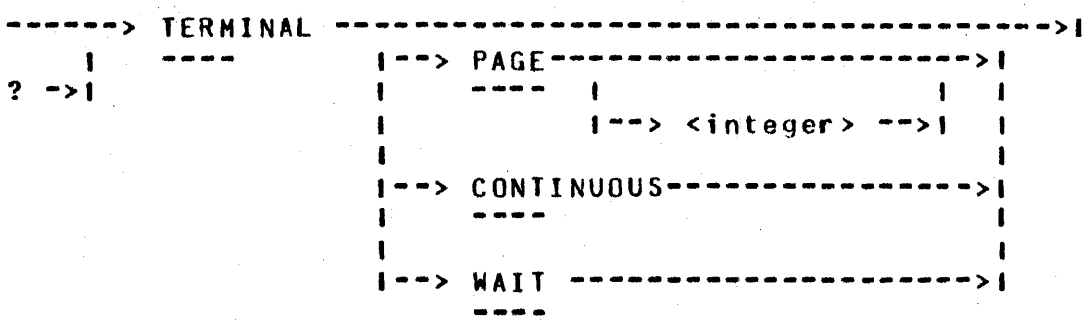
```
S 10+10
```

Enter automatic sequence mode using an initial and increment value of 10.

11.4.10. TERMINAL

The "TERMINAL" command allows the user to specify or query the number of lines per page or screen for the terminal, or to modify the output mode of the terminal.

Syntax:



Semantics:

"PAGE" specifies the number of lines per page or per screen, as the integer page length. Default page sizes are as follows:

Device	Terminal Type	Page Size
B9350 Teletype	0	1
TC Series	21,23,24,25,26,27	1
TD700	32	2
TD801	41	10
TD802	42	22
TD821	43	10
TD822	44	22
TD831	45	10
TD832	46	22

The default page sizes are also the maximum possible values.

If no parameters are specified, the current page size and current output mode are displayed. If a page length is not specified, the current page size is displayed.

"CONTINUOUS" causes the terminal to be left in "RECEIVE" unconditionally after each output, if "scroll" is enabled.

"WAIT" is the default output mode and causes the terminal to switch to "LOCAL" after most outputs.

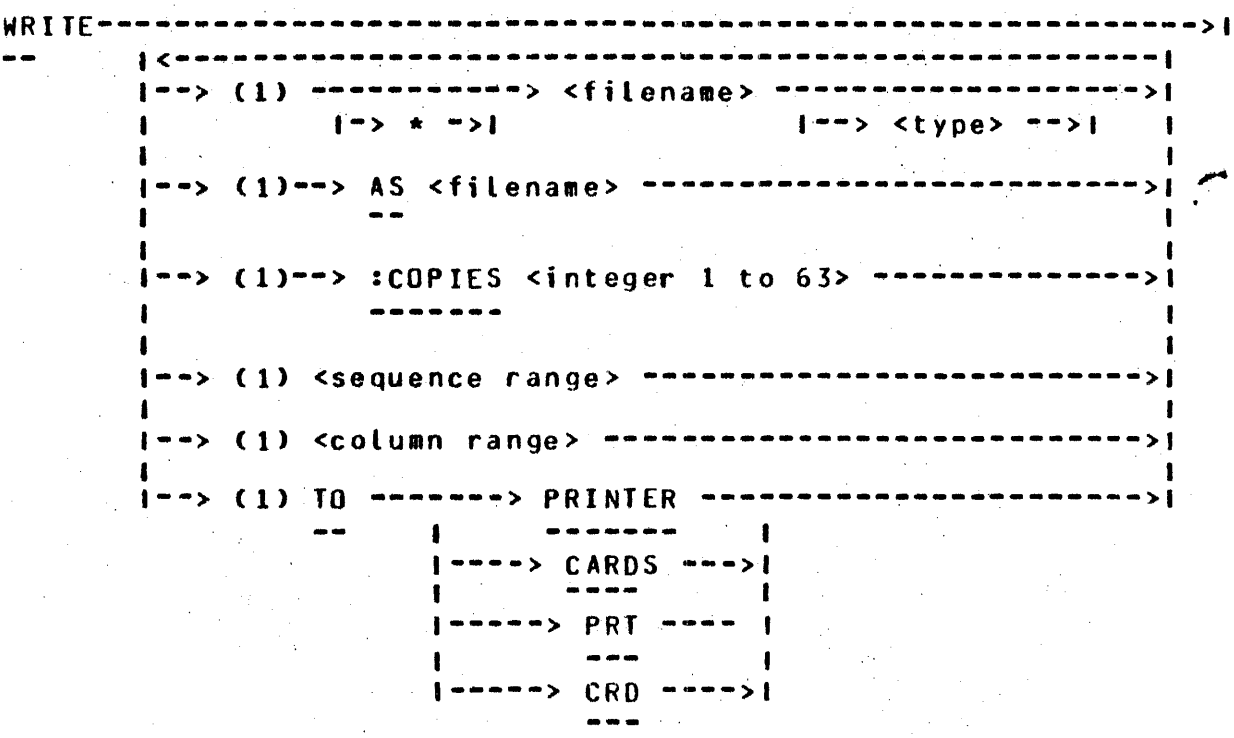
NOTE

TD830 series terminals are treated as TD820 terminals, except for FAST-SELECT and the "stop highlight" character. TD830 terminals on two-wire direct-connect lines should have the system adapter strapped for a 15-millisecond WRITE delay.

11.4.11. WRITE

The "WRITE" command prints or punches the contents of the workfile or some other file. Options are available for selecting specific lines or columns. The output of the "WRITE" command is forced to printer or punch backup.

Syntax:



Semantics:

By default, the contents of the workfile are printed. If a <filename> is specified and the user has access rights (or if the file is PUBLIC), it is used. A <sequence range> and a <column range> may be specified; otherwise, the whole file and the entire line are printed or punched.

The COPIES option is honored by SYSTEM/BACKUP unless overridden by the operator. If this option is not specified, the default number of copies produced is 1.

This command should be used in conjunction with the AUTOPRINT mechanism in the MCP. If AUTOPRINT is active, "WRITE" generates a backup file that can be printed or punched automatically.

Examples:

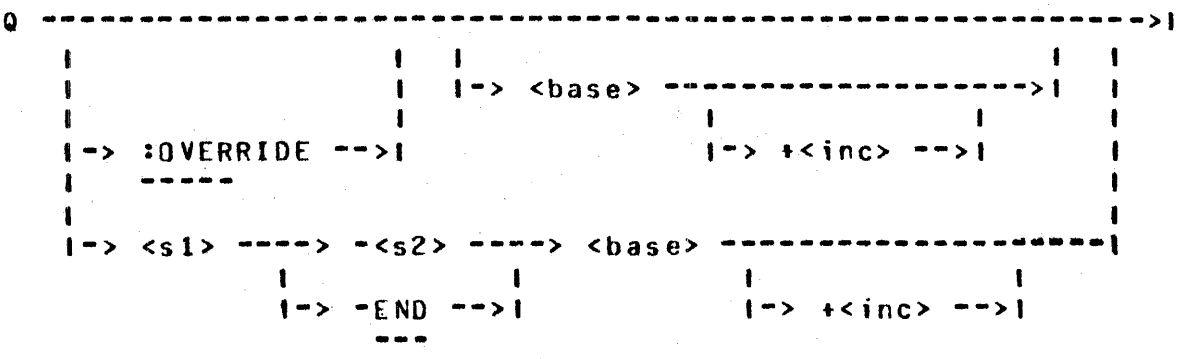
WRITE	Writes to a printer backup file all records in the user's current workfile.
WRITE A TO CARDS	Writes to a card punch backup file all records in file "<default pack>/(<usercode>)/A"
WRITE 100-1000 TO PRT	Writes to a printer backup file records 100 through 1000, inclusive.
WRITE *(<usercode>)/A	Writes to a printer backup file all records in file "(<usercode>)/A".

Note that the AUTOPRINT facility cannot be utilized if CANDE is not executed under a usercode.

11.4.12. RESEQ

The RESEQ command assigns new sequence numbers to lines in the workfile, without changing the order in which the lines appear. The default value for RESEQ is a base of 100 and an increment of 100 for all records in the workfile.

Syntax:



Semantics:

The base value is assigned as the sequence number of the first line to be renumbered; each subsequent line number is equal to the previous line number plus the increment value.

The "OVERRIDE" option orders CANDE to disregard any sequence errors (for example, a record with a blank sequence-number field) found in the workfile. If "OVERRIDE" is not specified and CANDE discovers a sequence error in the original workfile, it discontinues the resequencing operation at the point at which the error is found, then informs the user that it cannot continue processing the file. When CANDE fails during a resequencing operation, the "OVERRIDE" option should be used to recover; otherwise, this option should not be used.

For example, if CANDE discovers a sequence error while it is loading a file, RESEQ (with the appropriate base and increment values) is all that is required in order to resequence the file properly.

If a range of sequence numbers is provided (<s1>-<s2>), only the specified part of the workfile is renumbered (by default, the entire workfile is renumbered). When this range is provided, an initial value must be specified as the <base>. The increment (<inc>) for successive new sequence numbers may be specified; if none appears, the default value of 100 is assumed.

When resequencing BASIC files, sequence numbers in the text are modified to match the new sequence numbers on the appropriate records.

RESEQ does not change the apparent order of the lines. <s1> must be less than or equal to <base>, and the <base> plus the number of lines times the <inc> must not exceed <s2>+<inc>.

Examples:

```
RESEQ 200
```

Renumber the entire workfile, starting with an initial record number of 200 and proceeding in increments of 100.

```
RESEQ 100-700  100+10
```

Resequence lines 100 through 700, starting with an initial record number of 100 and incrementing by 10.

```
RESEQ +20
```

Resequence the entire workfile, with a base of 100 and incrementing by 20.

```
RESEQ 1000-1500 1100+10
```

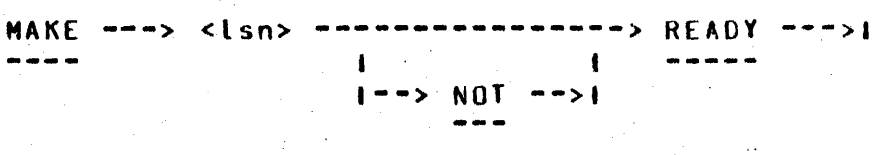
Resequence records 1000 through 1500, starting with an initial record number of 1100 and incrementing by 10.

11.4.13. Operator Commands

The system operator has certain commands that can be directed to CANDE. These commands are "WHO", "STOP", "ALL", "MAKE", and "AUDIT". The "WHO" command prints a list of usercodes and associated logical station numbers that are currently logged on to CANDE. The "STOP" command brings CANDE to an orderly conclusion (if there are no users logged on). The "ALL" command allows the operator to broadcast a message to all terminals. The "MAKE" command allows the operator to make a station "READY" or "NOT READY". The "AUDIT" command allows the auditing function to be dynamically started or stopped.

MAKE Command

Syntax:

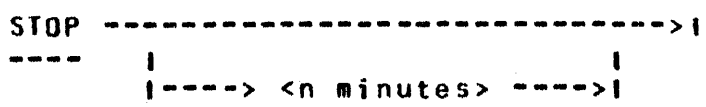


Semantics:

The "READY" flag in the Network Controller's tables for the specified <lsn> is changed as requested.

STOP Command

Syntax:



Semantics:

The "STOP" command provides for controlled shutdown of the CANDE system within <n> minutes (<n> is optional, with a default of 2). When entered at the SPO, the command causes the following:

a. If no users are logged on, CANDE goes to EOJ.

b. Each logged-on user is sent the following message:

```
CANDE WILL STOP IN <n> MINUTES. PLEASE SAVE OR
  REMOVE ANY ACTIVE FILE AND LOG-OFF.
```

c. At the same time each logged-on station is reported at the SPO. After all logged-on stations are reported, the following message is displayed at the SPO:

```
CANDE WILL ADVISE AUTOMATICALY AT <hh:mm:ss.t>
  IF USERS ARE STILL ACTIVE.
```

From this point, only the following commands are allowed:

BYE	UPDATE	Control Commands ("?")
RECOVER	REMOVE	WHAT
DCSTATUS	SAVE	RESEQ
FILES	TEACH	

One more page of input records can be processed from each station, so that a page in the process of being edited when the "STOP" occurs will not be lost.

Note that no further "single-line" input is allowed, except when that line comes from a device with a page size of 1, in which case it is treated as "one more page of input records", as above.

All other commands receive the following response:

```
CANDE TERMINATION IN PROCESS. PLEASE SAVE OR
  REMOVE ANY ACTIVE FILE AND LOG-OFF.
```

If <n> minutes elapse and there are still users logged on, the following message is displayed at the SPO:

```
SHUT DOWN TIME HAS ELAPSED WITH USERS STILL ACTIVE.
  OPERATOR ACTION REQUIRED.
```

None of the above conditions change, except that CANDE does not advise the operator again.

The automatic shutdown function is intended for the system operator's convenience, not terminal user inconvenience. To that end, CANDE will never shut down while users are active. It is the responsibility of the terminal user, however, to log off when no activity is anticipated for some time.

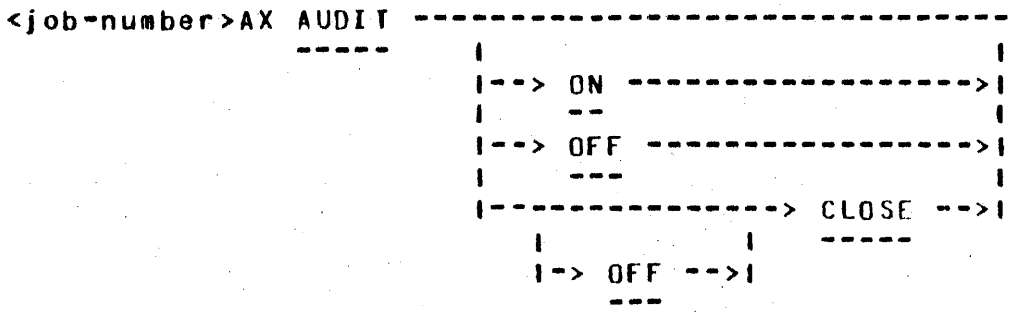
AUDII Command

CANDE audits each user-CANDE transaction whenever CANDE is executed with enough dynamic memory to handle its auditing functions. The AUDIT command allows the auditing capabilities to be set or reset without re-executing CANDE (when dynamic memory has been provided).

The audit file is automatically created if CANDE is executed with a dynamic memory size of at least 5760 bits. If CANDE is not executed in this fashion, auditing will be denied if requested (NO DYNAMIC MEMORY FOR AUDITING) and CANDE must be re-executed with sufficient dynamic memory (i.e., a minimum of 5760 bits).

Whenever an audit file is closed through the CLOSE option, the user should immediately print it with CANDE/ANALYZER, remove it from the disk directory, or change its name so that the next CANDE/AUDIT file opened does not cause a "duplicate-file" error condition. Since initialization of the audit file is a time-consuming process, the user should be aware that an AUDIT CLOSE followed by an AUDIT ON will cause a temporary suspension of data comm service by CANDE while the audit file is being re-initialized.

Syntax:



Semantics:

AUDIT	Displays the current status of AUDIT; whether it is ON or OFF and whether or not the AUDIT file is open or closed.
AUDIT ON	Sets the AUDIT option ON and opens and initializes the audit file (if it was closed).
AUDIT OFF	Resets the AUDIT option but leaves the AUDIT disk file pointer intact and leaves the AUDIT file open so that the next AUDIT ON command begins where auditing left off.
AUDIT [OFF] CLOSE	Resets the AUDIT option and closes the audit file on disk. The next AUDIT ON causes a new audit file of the same name (CANDE/AUDIT) to be created and initialized. The duplicate-file restrictions mentioned above apply to this command. Note that CLOSE is sufficient to close the audit file and reset the audit option to OFF, but that OFF CLOSE, although redundant, is not prohibited syntax.

CANDE writes an "audit-on" and "audit-off" record into the audit file whenever the AUDIT option is set or reset, so that those who examine the file will know the period of time during which auditing was not done.

11.4.14.
CANDE is released with the following segments marked as "important" for use with Extended Segment Decay and Priority Memory Management:

(0,01)
(0,02)
(0,07)
(1,01)
(1,10)
(1,11)
(1,14)
(1,15)
(1,16)
(1,37)
(1,57)
(2,00)

11.4.15. CANDE/ANALYZER

Introduction

CANDE/ANALYZER is a companion program to CANDE and is intended primarily to aid in debugging CANDE. In an interactive real-time program such as CANDE, errors can exist which become evident only upon entering some unusual sequence of commands or which involve a particular interaction between two or more users. When such errors occur, it is often difficult or impossible to know precisely what caused the failure. CANDE/ANALYZER should help resolve this problem.

Two principles guide the analysis process:

a. Since errors are the exception rather than the rule, the performance of CANDE should not be noticeably affected by the added function of maintaining an audit file.

b. When an analysis is required, information should be provided which is as complete and readable as possible. It is not possible to know in advance what information will be useful in isolating errors.

Analyzer-Auditfile Interface

If at least 5760 bits of dynamic memory is assigned at BOJ, CANDE creates a trace file (called "CANDE/AUDIT") and writes information about each user-CANDE transaction into that file. The CANDE/AUDIT file is organized in such a manner that the time required to create the trace is minimized, and is therefore not easily read. CANDE/ANALYZER assumes the burden of reading the audit file and breaking it down into a format easily read by the user.

CANDE/ANALYZER produces two general types of information:

a. An analysis of the information included in CANDE/TANK.FILE, CANDE/WORK.FILE, and CANDE/RECOVERY.

b. A trace of all messages sent and received by CANDE.

Structure of the Audit File

The audit file, CANDE/AUDIT, is created at BOJ if CANDE finds that there are at least 720 bytes of CANDE memory available for all of the audit buffer.

All messages which go through the file "MCSQUEUE" are written to the audit file, a circular buffer of 1000 360-byte records. The first record in the file is called the Audit Parameter Block (APB) and is initialized by CANDE at BOJ (or when the AUDIT option is turned ON) to contain information about CANDE itself (compile date and time, job number, etc.). The remaining 999 records constitute the circular buffer, in which CANDE packs as many as fifteen messages per record. The first four bytes of each record describe the contents of that record.

Optional Features

Optional features associated with the CANDE/ANALYZER program are set through the program switches. Default (zero) settings mean that the program analyzes the user-resident areas for logged-on stations only, prints the entire trace audit for the last 100 entries, and does not zip-execute CANDE. After analyzing CANDE's work, tank, and recovery files, the audit file is not removed.

ANALYZE the CANDE files CANDE/WORK.FILE and CANDE/RECOVERY even if CANDE/AUDIT is not present,

CANDE/ANALYZER will run even when the CANDE/AUDIT file is not present.

0 Analyze the CANDE files CANDE/WORK.FILE and CANDE/RECOVERY even though the CANDE/AUDIT file is not present

1-15 Do not attempt analysis of CANDE files if CANDE/AUDIT file is not present.

The functions of the program switches are defined as follows:

Switch	Value	Function
0	0	Print only those stations which have the "LOGGED.ON" bit in the USER.RESIDENT area true.
0	1-15	Print all stations.
2	0	Print trace audit.
2	1-15	Do not print trace audit.
3	0	Print analysis of the USER.RESIDENT areas for the various stations.
3	1-15	Do not print USER.RESIDENT areas.
4	0	When printing the USER.RESIDENT areas, also print the workfile contents for each user.
4	1-15	When printing the USER.RESIDENT areas, do not print the contents of the workfile.
5	0	Print ~~only~~ the entire trace ~~the final 100 entries of the trace~~.
5	1	Print only the final 500 entries of the trace.
5	2-15	Print ~~the entire trace~~ only the final 100 entries of the trace.
6	0	Do not Zip-execute CANDE after analyzing the work, ~~tank,~~ and recovery files.
6	1-15	~~Do not~~ zip-execute CANDE.
7	0	Do not Remove the audit file after analysis.
7	1-15	~~Do not~~ remove the audit file. after analysis if there are no other users of it.
8	0	Do not print information in the work, tank, or recovery files when they are in use or missing. 1-15
8	1-15	Print information in the work, tank, and recovery files even if they are already in use. This means that the information printed may be of questionable integrity, since CANDE may be changing such information even as it is being accessed by CANDE/ANALYZER.
9	0	Print the entire fine table for each user's workfile. The rough table for each user's workfile is always printed
	1-15	Do not print ~~the~~ each user's fine table.

12. TEXT/EDITOR

12.1. Introduction.

A number of minor changes and enhancements have been included in the Mark VII.0 version of TEXT/EDITOR. Each of these changes is documented separately in the paragraphs that follow.

The B 1800/B 1700 TEXT/EDITOR Reference Manual (form #1090610, dated 7-78) is now available, and reflects syntax and operation of TEXT/EDITOR as of the Mark VII.0 release. Temporary documentation on changes not included in the manual is contained in paragraph 12.3 of this release document.

12.2. Enhancements.

12.2.1.
TEXT/EDITOR is now compatible with File Security, and thus can be executed under a non-privileged usercode/password. Section 7 of the revised TEXT/EDITOR Reference Manual contains details on operational differences and restrictions when running under File Security.

12.2.2.
The RESTART command now handles recovery of the patch file, if one was associated with the workfile being recovered. A message is displayed on the terminal to indicate the successful recovery of the patch file in such a case, and the PATCH option is unconditionally SET.

12.2.3.
The PB command has been implemented, allowing printer backup disk files to be displayed on the terminal. The syntax of the PB command and a description of its function is presented on page 4-23 of the revised TEXT/EDITOR Reference Manual.

12.2.4.
A new command, BACK, has been implemented. Its function is identical to that of the NEXT command, except that movement through the workfile is in a reverse direction rather than a forward direction.

12.2.5.
The scanner used for the FIND, REPLACE, and SEARCH commands has been totally redesigned to allow for lower-case letters in SDL and UPL tokens. As a side benefit, the definition of a "token" for workfile types of SDL, UPL, MIL, DATA, and PSEUDO has been changed to accurately reflect the semantics of each language or usage. The new definitions for "token" are detailed on pages 4-9 through 4-11 of the revised TEXT/EDITOR Reference Manual.

12.2.6.
The performance of FIND, REPLACE, and SEARCH when LITERAL is specified has been substantially improved.

12.2.7.
For the convenience of users who must also use CANDE, the following changes to improve compatibility have been implemented:

a. CANDE-type abbreviations for a number of command keywords are now supported, as well as one-character abbreviations for all workfile types. These abbreviations are not yet documented in the TEXT/EDITOR Reference Manual; however, a complete list is presented in paragraph 12.3.1 of this release document.

b. The RECALL command keyword has been changed to SAME. RECALL is still allowed for Mark VII.0; however, it will be removed in Mark VIII.0.

c. The plus sign (+) is now allowed as an optional separator between the <base> and <increment> values in the <resequence-parameters>. The hyphen (-) is allowed as an optional separator between the <sequence-1> and <sequence-2> (or END) values in the <sequence-range>, and between the <column-1> and <column-2> values in the <column-range>.

12.2.8.
Because the MCP now provides for rollout of a program in "WAIT" status (that is, a program performing a COMPLEX.WAIT), the WAIT option implemented in Mark VI.1 TEXT/EDITOR is no longer required and has been deleted.

12.2.9.
The performance of the LIST command has been improved by removing the transmission of trailing blanks at the end of each source line sent to the terminal. A carriage-return control character is now appended to each line following the last non-blank character. This change (also implemented for the PB command) can greatly reduce message transmission time, and can result in lower memory requirements for messages passed to the Network Controller.

12.2.10.
The method by which output messages for the TD831 terminal (1200-byte data comm buffer) are handled by TEXT/EDITOR has changed slightly. Any output message of 1150 bytes or less is sent in one transmission; longer messages are sent in two transmissions with a 3-second delay between them. Together with the new handling for the LIST and PB commands, full-screen output to the TD831 can often be sent in one transmission rather than in two separate transmissions.

12.3. Temporary Documentation.

12.3.1.

The following abbreviations to command keywords are now supported in TEXT/EDITOR:

Keyword	Abbreviations
COPY	WRITE, WR
CREATE	MAKE, M
DELETE	DEL
HELLO	HE
INSERT	IN
LIST	L
LOAD	GET, G
MERGE	MER
MOVE	MO
PRINT	P
REMOVE	REM
RENAME	TITLE, TI
REPLACE	REP, FIX, F
RESEQ	RES
RESTART	RECOVER, REC
RMERGE	RM
SAVE	SA
SEARCH	SEA
SEQ	S
STATUS	WHAT
ZIP	?

In addition, any <file-type> specification can be abbreviated as the first character of the <file-type> keyword (for example, COBOL can be abbreviated as C).

12.3.2.

TEXT/EDITOR is released with six segments marked as "important" for use with Extended Segment Decay and Priority Memory Management. For information, the deck used to mark these segments is as follows:

```
? EXECUTE SYSTEM/MARK.SEGS
? FILE CODE NAME=TEXT/EDITOR;
? DATA CARDS
(0,00) 1 % ZERO
(0,02) 1 % SYNTAX_SCAN
(0,06) 1 % TOKEN_DECODER
(0,07) 1 % RECORD_UPDATE_INITIAL
(0,08) 1 % RECORD_UPDATE_FINAL
(0,09) 1 % UPDATE
? END
```

13. SMCS

13.1. Introduction.

Included as part of the MARK VII.0 software is the initial release of the B 1800/B 1700 Supervisory Message Control System (SMCS).

The Supervisory Message Control System (SMCS) is provided for users requiring interactive data communications systems, offering the capability of a comprehensive Message Control System (MCS) interface to the NDL Network Controller.

The SMCS can be used for a multitude of applications and by users with varying degrees of expertise. The SMCS user need not have any knowledge of programming or of NDL; however, a basic familiarity with the B 1800/B 1700 system, the MCP, and the application programs to be used is necessary.

The SMCS program is intended to be the "supervisor" for a data communications software system which includes such Burroughs programs as CANDE, TEXT/EDITOR, HASP, and RJE, as well as other on-line packages of either Burroughs or customer origin.

The Illustrative MCS files (MCSI and MCSII) are released as source file to guide those users who wish to write their own MCS. The Mark VII.0 software release includes MCSI and MCSII as source files and SMCS as a code file. MCSI and MCSII are still intended to be used only as a guide, and are not supported in any way.

The SMCS is a program product which may be separately ordered. Documentation on the SMCS is available with the program product in the form of a printer backup file labeled DOCUMENT/SMCS. The B 1800/B 1700 SMCS can be ordered through the local Burroughs branch office from Program Products Distribution.

14. RJE

14.1. Introduction.

The Mark VII.0 release of the B 1800/B 1700 Remote Job Entry (RJE) system contains three seperate RJE software packages:

a. An NDL Data Comm handler and an SDL INPUT/OUTPUT program (RJE/MCS).
b. An SDL Data Comm handler and INPUT/OUTPUT program (RJE).
c. An SDL Data Comm handler (RJE/DCH).

14.2. Enhancements.

14.2.1. RJE/MCS

The RJE/MCS program has no major changes for the Mark VII.0 release.

14.2.2. RJE

Three new run-time parameters have been implemented:

a. IDLE.ON.LOG.OFF
b. NO.REMOTE
c. REMOTE

The local control command .READ has been enhanced to allow reading a specified file from a card, disk, or tape device.

A Remote Terminal Interface has been implemented, which allows one or more terminals to communicate with a single copy of the RJE program that is on-line to a host system using the Supervisory Message Control System (SMCS). This remote terminal interface is only available to users of the SMCS program product. Documentation on the SMCS is supplied in the form of a printer backup file labeled DOCUMENT/SMCS to users of the SMCS program product.

14.2.3. RJEDCH

The RJE/DCH program has no major changes for the Mark VII.0 release.

14.3. Known Errors and Restrictions.

14.3.1.
BCL input files cannot be effectively used because of the inability to dynamically change the translate table once the card file has been opened.

14.3.2.
The RJE/MCS program at times closes the print or punch file before the last buffer is processed if the host system sends an EOF record at the end of the data stream. This occurs because the EOF record is sent to a queue which differs from that of the output stream, and may be processed before the last buffer of the output stream.

14.4. Temporary Documention.

14.4.1. Run-Time Parameters

IDLE.ON.LOG.OFF

When running on leased lines and the host system terminates the session, the RJE program will attempt to re-establish the Data Comm link. If this feature is not desired, the IDLE.ON.LOG.OFF parameter causes the RJE program to place itself in an IDLE state whenever the RJE program logs off the host system or whenever the host system terminates the Data Comm link. The RJE program resumes normal operation whenever it receives a message from the host system or when the RJE program has something to send to the host system.

Default: The RJE program establishes the Data Comm link whenever the Data Comm link is broken if using a leased line.

NO.REMOTE

This parameter requests the RJE program not to open a remote file for communications with the SMCS. This parameter also overrides the program switch 9 setting.

Default: No remote file is opened.

REMOTE

This parameter instructs the RJE program to read from a remote file and process the messages as if they were entered from the Remote Supervisory Console keyboard. This option can only be used in conjunction with the SMCS program product.

Default: No remote file is opened.

14.4.2. .READ or .RE (Read) Local Control Command

The .READ (or .RE) command has the following format:

```
.READ [<device type>] [<"filename">] [<control character>]
```

The .READ command causes the RJE program to open a card-image file and begin transmitting the data to the host system. When end-of-file has been detected the card-image file is closed. One .READ command is required to read each card-image file.

The .READ command is only valid if the RJE program has been properly logged-on to the host system. It is possible to override this feature by entering the .RY CR local command.

The <device type>, <"filename">, and <control character> parameters are optional and can be entered in any sequence following the .READ command.

The <device type> specifies the type of device the RJE program is to read from, and must be one of the following:

```
CARD
DISK
TAPE
```

The default device type is CARD.

The <"filename"> specifies the label of the file. The RJE program uses the filename specified to locate the file on the designated device type. The filename must be surrounded by quotes (") with no embedded blanks. The default file name is "RJE/CARDS".

The <control character>, if specified, must be a question mark (?). This option informs the RJE program that the file to be opened contains control cards. The RJE program changes question marks in column one of all cards to a NULL (@00@) code before transmitting the record. The host system can then recognize the record as a control card. The control character is only valid for files on device types of DISK or TAPE.

Examples:

```
<job-number>AX.READ
<job-number>AX.RE "USERPACK/SOURCE/FILE" DISK ?
<job-number>AX.RE DISK ?
<job-number>AX.READ ? TAPE "SOURCE"
```

14.4.3. Remote Supervisory Console Output Messages

The RJE program displays messages in response to local control messages, or when a condition is detected of which the operator should be notified. The new output messages are as follows:

BEGIN TRANSMISSION <device type> FILE <filename>

This message is displayed whenever the RJE program begins transmitting a file to the host system. The <device type> is either CARD, DISK, or TAPE and the <filename> is the label of the file.

CARD READER ACTIVE WILL TRANSMIT FILE <filename> WHEN READER BECOMES AVAILABLE

This message is displayed in response to a .READ or .RE local command. If a file was currently being transmitted to the host system when the .READ was entered, the RJE program saves the local command and transmits the file when the reader becomes available.

END TRANSMISSION <filename>

This message indicates that the RJE program has detected end-of-file (EOF) on the input file and has closed the file. A special end-of-file control message is transmitted to the host system.

REQUEST IGNORED: REQUEST QUEUE FULL

A .READ or .RE local command was entered while the reader was still active and the request queue is full (100 entries maximum). Re-enter the local command .READ after the current file has been transmitted to the host system.

14.4.4. RJE Files

The following files have been added to the RJE program.

INTERNAL READ REQUEST QUEUE

```
Internal Name: REQUEST.QUEUE
Label: RJE/REQUEST
Device: DISK RANDOM
```

MCS REMOTE FILE

```
Internal Name: REMOTE
Label: MCSFILE
Device: REMOTE (255) WITH HEADERS
Records: 173/1
Number Of Stations = 0
```

15. Host RJE

15.1. Introduction.

The Mark VII.0 release of the B 1800/B 1700 Host RJE system contains several enhancements over the Mark VI.1 version. Each of these changes is detailed in the paragraphs that follow.

15.2. Enhancements.

15.2.1.
In order to obtain a more efficient utilization of a remote terminal and its Data Comm line, the automatic CALLBACK feature has been implemented. A remote user can log off with the CALLBACK option set while jobs are still executing, making the Data Comm line available for another user. Upon completion of the original user's jobs, the RJE/CONTROLLER program finds an available dialout line and automatically dials the phone number supplied by the remote user. When communication is established, all printer or punch output is transmitted back to the remote user automatically if the RJE AUTOPRINT and AUTOPUNCH options are set. Refer to paragraph 15.4.2 for detailed information concerning the CALLBACK feature.

15.2.2.
There is one new Remote Supervisory Console (RSC) command:

*PH (Phone Number)

Refer to paragraph 15.4.1 for detailed information concerning this RSC command.

15.2.3.
The following RSC commands have been enhanced.

*RO (Reset RJE Option)
*SO (Set RJE Option)
*TO (Display RJE Options)
*US (Usercode/Password)

Refer to paragraph 15.4.2 for detailed information concerning these RSC commands.

15.2.4.
An end-of-file message is transmitted at the end of a print or punch stream. This function can be modified by setting the RJE/AUTOBACKUP program switch 1 to any of the following values:

0 = Transmit EOF message for both print and punch streams.

1 = Transmit EOF message for the punch stream but not the print stream.

2 = Transmit EOF message for the print stream but not the punch stream.

3 = Do not transmit EOF message for either print or punch streams.

15.2.5.
The RJE/CONTROLLER program can no longer be executed without a usercode or with a non-privileged usercode.

15.2.6.
The maximum length of an RSC message transmission is 80 characters. If an RSC message is longer than 80 characters, multiple records are transmitted in increments of 80 characters to the Remote Supervisory Console.

15.3. Known Errors and Restrictions.

15.3.1.
The Mark VII.0 RJE/CONTROLLER and RJE/AUTOBACKUP programs are not compatible with the Mark VI.1 MCP release.

15.4. Temporary Documentation.

15.4.1. New Remote Supervisory Console (RSC) Command

*PH (Phone Number)

The *PH command can be entered in either of the following formats:

Format 1: *PH
Format 2: *PH = <phone number>

The *PH input command allows the remote system operator to interrogate the current phone number, or to enter or change the phone number to be used when automatically re-establishing connection.

Format 1 informs the remote system operator of the current phone number.

Format 2 allows the remote system operator to enter or change the phone number. A phone number is required when the CALLBACK option is set and is used for automatically re-establishing connection to transmit the remote user's printer or punch files. The remote system operator is notified that the phone number has been changed by displaying the change on the RSC.

The phone number can be a maximum of 20 characters in length, including dashes as separators.

Examples:

```
*PH
*PH 1-222-333-4444-555
*PH 333-4444
*PH 1-333-4444
*PH 1-222-333-4444
```

15.4.2. Enhanced Remote Supervisory Console (RSC) Commands

*RO (Reset RJE Option)

Format: *RO <option-name>

The *RO input command allows the remote system operator to reset the options used by the RJE/CONTROLLER program. RJE/CONTROLLER replies with a verification that the option has been reset after each *RO command.

Refer to the documentation on the *SO input command below for a description of the allowable options.

Examples:

```
*RO AUTOPRINT
*RO CALLBACK
*RO AUTOPUNCH
```

*SO (Set RJE Option)

Format: *SO <option-name>

The *SO input command allows the remote system operator to set the options used by the RJE/CONTROLLER program. RJE/CONTROLLER replies with a verification that the option has been set after each *SO message.

There are three allowable run-time remote terminal options:

AUTOPRINT When this option is set, all print files are automatically transmitted to the remote terminal system as soon as the job has been completed, without intervention by the remote system operator.

AUTOPUNCH When this option is set, all punch files are automatically transmitted to the remote terminal system as soon as the job has been completed, without intervention by the remote system operator.

CALLBACK When this option is set, if the remote system operator logs off with a job active, connection is automatically re-established upon completion of that job using the phone number supplied by the *PH RSC input command. The printer and/or punch backup files are then transmitted to the remote terminal user. The Data Comm line is available for use by other users during the time between log off and re-establishment.

Examples:

```
*SO AUTOPRINT
*SO CALLBACK
*SO AUTOPUNCH
```

*TO (Display RJE Options)

The *TO command can be entered in either of the following formats:

```
Format 1:  *TO
Format 2:  *TO <option>
```

The *TO input command allows the remote system operator to interrogate the status of the remote options.

Examples:

```
*TO
*TO AUTOPUNCH
*TO AUTOPRINT
*TO CALLBACK
```

*US (Usercode/Password)

The *US command can be entered in any of the following formats:

```
Format 1:  *US
Format 2:  *US <usercode>[/<password>] PB <options>
Format 3:  *US <usercode>[/<password>] PD <options>
Format 4:  *US <usercode>[/<password>] RB <options>
```

Format 1 allows the remote system operator to obtain a list of every remote terminal system in the network by Remote Station Number (RSN).

Format 2 allows the remote system operator to initiate the transmission of a print or punch file to the remote terminal that was entered under the specified usercode.

Format 3 allows the remote system operator to interrogate the host system's output file directory for files entered under the specified usercode.

Format 4 allows the remote system operator to remove entries from both the directory and the system that were entered under the specified usercode.

Refer to the *PB, *PD, and *RB RSC input commands in the B 1800/B 1700 Remote Job Entry (RJE) Reference Manual (form #1090602, dated 7-78) for the syntax of the <options>.

Examples:

```
*US
*US PAYROLL/ACCT PB J 125
*US PAYROLL/ACCT PB = LABELS
*US LEDGER/FILE  PD J 495
*US PAYR PD =
*US PAYR RB S 1001
*US PAYROLL/ACCT RB 87
```

15.4.3. New Remote Supervisory Console (RSC) Output Messages

#BYE ABORTED, NEED PHONE NUMBER

Originator: RJE/CONTROLLER
The CALLBACK option was set but no phone number was available for the automatic CALLBACK feature.

#BYE ABORTED, NO DIALOUT LINE AVAILABLE

Originator: RJE/CONTROLLER
The DIALOUT hardware is not present.

#CALLBACK ABORTED, NO JOBS ACTIVE

Originator: RJE/CONTROLLER
This message indicates that the automatic CALLBACK option was set and the remote terminal has logged off without any jobs executing at the host system.

#TO BE CALLED BACK

Originator: RJE/CONTROLLER
This message indicates that the CALLBACK option has been set and a phone number entered. The remote terminal is automatically called back when the first job executing has completed. When re-establishment has occurred with the remote terminal, the print or punch backup files are transmitted to the remote terminal.

15.4.4. RJE Host System Output Messages

#INCONSISTENCY BETWEEN LSN'S OF RSN

Originator: RJE/CONTROLLER
This message indicates that a possible hardware malfunction has occurred. An automatic program DUMP is produced and the RJE/CONTROLLER goes to EOJ.

#ILLEGAL TERMINAL TYPE

Originator: RJE/CONTROLLER
This message indicates that a possible hardware malfunction has occurred. An automatic program DUMP is produced and the RJE/CONTROLLER goes to EOJ.

#INVALID LSN FROM STATION STATUS

Originator: RJE/CONTROLLER
This message indicates that a possible hardware malfunction has occurred. An automatic program DUMP is produced and the RJE/CONTROLLER goes to EOJ.

16. RJE 3780

16.1. Introduction.

Included as part of the Mark VII.0 software is the initial release of the B 1800/B 1700 3780 Remote Job Entry (RJE) system.

The B 1800/B 1700 RJE 3780 program allows the remote user to transmit queries, programs, and data files to a central computer for processing, and upon completion, to receive the final printer or punch output at the B 1800/B 1700 remote computer. The central system must be using the IBM 3780 Data Comm characteristics.

The RJE 3780 system is a program product which may be separately ordered. Documentation on the RJE 3780 system is available with the program product in the form of a printer backup file labeled DOCUMENT/RJE3780. The B 1800/B 1700 RJE 3780 system can be ordered through the local Burroughs branch office from Program Products Distribution.

17. HASP

17.1. Introduction.

The Mark VII.0 release of the B 1800/B 1700 HASP system contains several enhancements over the Mark VI.1 version, including new local commands, improved performance, hot readers, and a remote terminal interface.

17.2. Enhancements.

17.2.1. HASP

The following problems have been corrected:

a. The alternate file is reset after a secondary file is opened, causing the next .SR local command to read the correct file.

b. Punch files may now be routed directly to the punch device instead of going to punch backup first.

c. "Wait-a-bit" is now set for a stream as soon as a record is queued (that is, when the line printer gets behind).

d. The HASP program now counts consecutive timeouts and displays a message every 15 timeouts as an indication that the host is no longer accepting transmission.

e. The Functional Control Sequence (FCS) character is checked for "wait-a-bit" on duplicated transmission blocks.

17.2.2. HASP/MODIFIER

The following two run-time parameters have been added:

REMOTE

The REMOTE parameter is used to order the HASP program to open a remote file and to process messages read from that remote file. A Network Controller and the Supervisory Message Control System (SMCS) are required in order to use the remote terminal interface. The remote terminal interface cannot be used in the Spool Mode.

MEM=<integer>

The MEM parameter is used to inform HASP/MODIFIER how much dynamic memory the HASP program is to use. The <integer> must be equal to or greater than the minimum dynamic memory. The minimum dynamic memory can be calculated by using the following expression:

```
((BUFFER + 14) * 4 * 8 + ((MAX.PRT + MAX.PUN) * 1211) +
  (IF PUNCH = 96 THEN 512 * 8))
```

If the MEM parameter is not used, the HASP/MODIFIER program calculates dynamic memory using the following expression:

```
((BUFFER + 14) * 4 * 8 + (((MAX.PRT + MAX.PUN) * 1211) *
  (10 * 1085)) + (IF PUNCH = 96 THEN 512 * 8))
```

17.2.3. HASP - Direct Mode

Two new local commands have been implemented, as follows:

.HRn (Hot Reader)

.WHAT (Display Reader Status)

Changes have been made to the following three local commands:

.EOFn (End-of-File)

.GOn (Continue transmission)

.S (Start Reader)

The following local command has been deleted:

.JCLn (JCL Reader)

Refer to paragraph 17.4 for a complete description of the new local commands and the new functions of the existing local commands.

A Remote Terminal Interface has been implemented, which allows one or more terminals to communicate with a single copy of the HASP program that is on-line to a host system, using the Supervisory Message Control System (SMCS). This remote terminal interface is only available to users of the SMCS program product. Documentation on the SMCS is supplied in the form of a printer backup file labeled DOCUMENT/SMCS to users of the SMCS program product.

17.2.4. HASP - Direct Mode and Spool Mode

The maximum number of output devices allowed has been changed from 8 to 7. The sum of the values specified in MAX.PUN=n and MAX.PRT=n must not be greater than 7.

17.3. Known Errors and Restrictions.

17.3.1.
The HASP program enters an indefinite wait loop at midnight due to the system clock being set to zero by the MCP.

17.3.2.
The program name "HASP" cannot be changed to any other name when using the Supervisory Message Control System (SMCS) remote terminal interface.

17.4. Temporary Documentation.

17.4.1. HASP in Direct Mode

.EOFn

The .EOFn local command instructs the B 1800/B 1700 HASP program to stop reading from an input reader and to send the end-of-stream indicator to the central system. The letter "n" in the .EOF command specifies the reader number from which the HASP program is to stop reading. The .EOFn command also deactivates the specified reader. This command is only valid when a secondary file specifier was not found. The .SD, .SR, and .ST commands explain the use of secondary file specifiers.

.GOn

The .GOn local command instructs the B 1800/B 1700 HASP program to continue reading from an input reader. The letter "n" in the .GO command specifies the reader number from which the HASP program is to continue reading. This command is only valid when a secondary file specifier was not found. The .SD, .SR, and .ST commands explain the use of secondary file specifiers.

.HRn

The .HRn local command allows the user to permanently assign a card reader device to the HASP program. The optional letter "n" in the .HR command specifies the reader number from which the HASP program is to read and, if used, must be a value between 1 and 4, inclusive. The reader number cannot exceed the value of the MAX.RDR run-time parameter.

Each reader has a separate file, having internal file names of INPUT0 through INPUT3. The numeric portion of the name is the reader number minus one.

If the .HR local command is used without specifying a reader to be used, the HASP program uses the first non-active reader.

The .HR command allows the user to dedicate a card reader to the HASP program. The user may transmit a card deck to the central computer by entering the deck into the card reader, eliminating the need to enter the .S local command on the console keyboard. The following rules must be adhered to when using the .HR local command:

a. When the .HR command is initiated, the following card must be fed into the reader assigned by the .HR command:

```
?STREAM HASP/CARDS
```

The label HASP/CARDS cannot be changed.

b. Any card deck can be transmitted to the central computer by placing the card deck in the read hopper of the card reader and pressing the START button. The first card of the card deck must have the following format:

```
?DATA <filename>
```

Card columns 1 through 5 cannot contain any embedded spaces. The file name can be either one name or two names separated by a slash.

c. The last card in the card deck must have the following format:

```
?END
```

Card columns 1 through 4 cannot contain any embedded spaces.

d. To release a reader from the .HR command, enter the following card through the card reader:

```
?TERMINATE HASP/CARDS
```

Example:

```
?STREAM HASP/CARDS
?DATA CARDS
.SD MICR/JCL1
.SD MICR/FILE1
.SD MICR/JCLEND1
?END
?DATA CARDS
.SD MICR/JCL2
.SD MICR/FILE2
.SD MICR/JCLEND2
?END
?DATA CARDS
//TEST1 JOB (123456,123456,HDH),CLASS=A,TIME=8
```

```
//LIST EXEC COBUCLG,PARM.COB='LOAD'
//COB.SYSPUNCH DD SYSOUT=B
//COB.SYSIN DD *

        <source deck>

/*
//LKED.SYSDUMP DD SYSOUT=A
//GO.SYSPRINT  DD SYSOUT=A
//GO.SYSIN DD *

        <input data>

/*
//
?END
?TERMINATE HASP/CARDS
```

.San <filename>

The .S local command causes the HASP program to read a file, compress the data, and transmit the file to the central computer. The letter "a" in the .S command specifies the type of device from which the HASP program is to read. The following is a list of valid 1-character mnemonics:

D = Disk
R = Card Reader
T = Tape

The letter "n" in the .S command specifies the reader number from which the HASP program is to read, and must be a value between 1 and 4, inclusive. If the reader number is used, no queuing of the command is done if that reader is active.

The <filename> in the .S command is the label of the file. The HASP program uses the <filename> to locate the file on the specified device.

The maximum number of readers that can be active at one time is determined by the run-time parameter MAX.RDR. Each reader has a separate file, having internal names of INPUT0 through INPUT3. When the .S command is entered to the HASP program, a search is done to find a non-active reader.

If a non-active reader is found, the file is marked active and the file is opened using the file name and device type specified. If the file is not found, the HASP program displays a message and the operator must re-enter the command.

If all readers are active and the reader number was not specified in the .S command, the HASP program queues the command and notifies the operator that the command is queued.

When a reader becomes non-active, the queued ccmmand is read by the HASP program and the operator is notified that the command is in process.

If the reader specified in the .S command is active, the HASP program notifies the operator that the reader is active and the .S command is not queued. The operator must re-enter the command when the reader is not active.

Multiple job steps may also be built and transmitted to the central computer as a single job to be executed serially by the central computer. The data being sent to the central computer can be on disk, tape, or cards.

In order to create multiple job steps to be executed serially, the local commands .SD, .SR, and .ST have been made available and can be included as a part of the input file. The .SD, .SR, or .ST commands start in position one of the card image. Position four must be blank. The file name of the data file starts in position five. Any number of .SD, .SR, and .ST local commands may be included as a part of the input file.

All card images that do not have .SD, .SR, or .ST in the first three character positions followed by a blank are transmitted to the central computer.

When either .SD, .SR, or .ST is detected, the B 1800/B 1700 HASP program stops reading from the primary input file and modifies a secondary file to the device type and file name specified. This secondary file is then opened and transmitted to the central computer. When the secondary file has reached end-of-file, that file is closed and input is read again from the primary file. If the next card image contains a local command this same procedure is performed; otherwise, the card is transmitted to the central computer. If the secondary file contains .SD, .SR, or .ST local commands, they are treated as data and transmitted to the central computer.

The end-of-stream control message is sent to the central computer when the primary input file has reached end-of-file, or when explicitly requested by the B 1800/B 1700 HASP operator. Refer to the .EOFn local command for additional information.

An error will occur when using the .SD, .SR or .ST local commands and the secondary file is not present. If the file is not present, the B 1800/B 1700 HASP program waits for an operator response to continue reading (.GOn command) from the input file or to terminate (.EOFn command) the job stream and send the end-of-stream message to the central computer.

When the secondary file is not present, the operator has three options:

a. Correct the card and place it back into the card reader.

b. Bypass the record and continue.

c. Close the input file and send the end-of-stream message to the central computer.

The B 1800/B 1700 HASP operator's response for items a and b is the local command .GOn, and .EOFn for item c.

The correct format and sequence of the job stream transmitted in this manner is the responsibility of the user.

The following is an example of how the merging of secondary files can be performed. Assume that the operator enters the following command to the HASP program:

```
<job-number>AX.SR CARDS
```

The HASP program identified by <job-number> opens a card file labeled CARDS, which contains the following information:

```
?DATA CARDS
.SD MICR/STARTJCL
.SD MICR/FILE1
.SD MICR/FILE2
.ST MICR/FILE3
.SD MICR/ENDJCL
?END CARDS
```

The files labeled MICR/STARTJCL, MICR/FILE1, and MICR/FILE2 are transmitted to the central system from a disk device. The file labeled MICR/FILE3 is then transmitted from a tape device. The file labeled MICR/ENDJCL is then transmitted from a disk device. When the HASP program has detected EOF on the file labeled CARDS, the end-of-stream indicator is transmitted to the central system and the file labeled CARDS is closed.

.WHAT

The .WHAT local command allows the HASP user to inquire about the status of all readers. A typical response would be formatted as follows:

```
READER 1 HOT READER
READER 2 NOT ACTIVE
READER 3 CARDS ON RECORD 3425.
READER 4 NOT ACTIVE
```

18. System Utilities

18.1. Introduction.

This section contains descriptions of new system utility programs (released for the first time on Mark VII.0), as well as modifications and enhancements made to existing system utility programs. Documentation on the following programs is included in this section:

CHECK/LOAD.DUMP
CODE/ANALYZER
COLDSTART/DISK
COLDSTART/TAPE
DISK/ALLOCATOR
DISKETTE/COPY
DISKMAP/UTILITY
DMPALL
DUMP/ANALYZER
LOGCONVERT
MCPII/ANALYZER
Disk Pack Initializer (PACK/INIT)
QWIKLOG
SORT
STANDALONE/DISK.DUMP
SYSTEM/BACKUP
SYSTEM/COMPARE
SYSTEM/DISK.INIT
SYSTEM/ICMD.INIT
SYSTEM/ELOGOUT
SYSTEM/MAKEUSER
SYSTEM/MARK.SEGS

The following utilities have received either no changes or only minor modification in order to insure compatibility with the Mark VII.0 system. Separate documentation on them is not included.

Disk Cartridge Initializer (CART/INIT)
CASSETTE/MAKER
DISK/COPY
DISK/DUMP
DISKPACK/INTERCHANG
FILE/LOADER
FILE/PUNCHER
SORT/COLLATE
SORT/MERGE
SORT/QSORT
SORT/TAPESORT
SORT/VSORT
SSLOAD/MAKCAS
STANDALONE/INTERCHANG
SYCOPY
SYSTEM/BUILDTRAIN
SYSTEM/DISK.DUMP
SYSTEM/LDCONTRL
SYSTEM/LOAD.CAS
SYSTEM/LOAD.DUMP
SYSTEM/LOGOUT
SYSTEM/SPOLOGOUT
TAPECOPY

A complete description of the functions and related syntax of each system utility program is contained in section 3 of the revised System Software Operational Guide.

18.2. Enhancements.

18.2.1. CHECK/LOAD.DUMP

a. If CHECK/LOAD.DUMP is executed with switch 2 set to a non-zero value, the output listing will not be printed. The total number of errors detected on the library tape will be displayed on the SPO at EOJ, regardless of the setting of switch 2.

b. If CHECK/LOAD.DUMP is executed under a usercode/password and switch 0 is set to a non-zero value, MCP File Security naming conventions apply to file-identifiers entered through ACCEPT messages. That is, usercodes are implicitly added to single-name file-identifiers (unless the first character is an asterisk). If CHECK/LOAD.DUMP added the usercode to the file-identifier and the file could not be found in the tape directory, the usercode will be removed and a second scan of the directory will be made to attempt tc find the file.

18.2.2. CODE/ANALYZER

CODE/ANALYZER indicates on the output listing any code segments that are marked as "important", for use with Extended Segment Decay and Priority Memory Management.

18.2.3. COLDSTART/DISK

A problem that could cause "NO AVAILABLE DISK" conditions following COLDSTART of a head-per-track disk has been corrected.

18.2.4. COLDSTART/TAPE

COLDSTART from tape now accepts 7-track tapes having labels written in either even or odd parity.

18.2.5. DISK/ALLOCATOR

This is the initial release of DISK/ALLOCATOR, a program used to create Installation Allocated Disk files. Such files are created with user-specified attributes and absolute disk addresses. Refer to the revised Software Operational Guide for complete documentation.

18.2.6. DISKETTE/COPY

a. DISKETTE/COPY now has a provision for relabeling diskettes, using the RL input command specification.

b. DISKETTE/COPY now recognizes the diskette file BYPASS flag, allowing files with duplicate names to exist on the same diskette.

c. There is now a provision for automatic loading of pseudo-reader files from a diskette to systems disk. When a diskette is readied by the MCP, the diskette is scanned for any files having the first three characters of the file-name equal to "PSR". If the MCP finds one or more such files, DISKETTE/COPY is automatically executed, which then copies the pseudo-reader files from the diskette to systems disk.

18.2.7. DISKMAP/UTILITY

When analyzing a disk with neither MAP nor CHECK specified, it is no longer necessary for all activity on the disk to cease during Phase 1.

18.2.8. DMPALL

a. In order to simplify operation, the three functions provided by program switch 9 have now been broken down into separate switches. If switch 7 is set to 1, DMPALL terminates after performing the current specification. If switch 8 is set to 1, card output files are not interpreted. If switch 9 is set to 1, input file errors are ignored.

b. Setting switch 1 to 1 while DMPALL is performing a function now causes DMPALL to immediately terminate the current function and go on to the next input specification. DMPALL automatically resets switch 1 to 0 in such a case.

c. The KEY, SEARCH, SELECT, and EXCLUDE options have been expanded to allow comparison on bit or digit boundaries within a record.

d. DMPALL now can access COBOL and RPG index-sequential files in a sequential manner, using the associated TAG file.

Refer to the revised Software Operational Guide for syntax and detailed information on all of the above enhancements.

18.2.9. DUMP/ANALYZER

The NC option of the PM input message is no longer allowed by the MCP. A change has been made to DUMP/ANALYZER to recognize that the dumpfile being analyzed is that of an NDL Network Controller; if so, NDL/DUMP is zip-executed automatically following the standard memory dump analysis.

18.2.10. LOGCONVERT

a. Several new record types appear in the log when the TABS option is set. These records log the opening and closing of files, the transfer of the log file, and the assignment and release of peripheral units by special "Direct I/O" programs (such as SYSTEM/DISK.INIT). Refer to the revised Software Operational Guide for the format of these new records.

b. The CLEAR/START Record has a number of additional fields, including the GISMO and MICRO.MCP file-identifiers and the setting of the TABS option. The MCP VERSION field has also been increased to 10 bytes in length. Refer to the revised Software Operational Guide for the complete format of the CLEAR/START Record.

c. Within the PPB and FPB records, the Job Accounting Number and Job Number now appear in place of the Jcb and Mix numbers, respectively. Due to a change in the size of both these fields since Mark VI.1, and to other output record format changes, programs which process the LOGCONVERT output file must be modified. Refer to the revised Software Operational Guide for the new layouts of all record types.

d. LOGCONVERT now verifies that it is being run under the correct software release and terminates if there is a mismatch. This process ensures the validity of the NEWLOGFILE data.

18.2.11. MCPII/ANALYZER

Due to a change in the layout of memory by the System Initializer, MCPII/ANALYZER is now able to perform the Code Segment Comparison operation on segment zero of the MCP (which could not be done in Mark VI.1).

Code segments 9 and 10 of the MICRO.MCP are still the only comparison failures that should be expected to occur with Mark VII.0.

18.2.12. PACK/INIT and SYSTEM/DISK.INIT

a. The initialization specifications have been changed, allowing explicit error limits to be supplied to the initializer. Refer to the revised Software Operational Guide for complete syntax of all specifications.

b. A RECONFIGURATION option is now available which allows changing the pack type. For example, it is now possible to reconfigure a pack that was previously initialized as UNRESTRICTED (U) into a SYSTEM (S) pack. Reconfiguration is available only if the pack was initialized by the Mark VI.1.1 or Mark VII.0 PACK/INIT or SYSTEM/DISK.INIT, and the Master Available Table on the pack is good. If reconfiguration is requested, the pack is purged, verified, and the type is changed as specified.

Reconfiguration can be requested when using either SPO or card input. If entering specifications on the SPO, the following message is displayed if the initializer encounters a pack that can be reconfigured (that is, the pack was initialized by at least the Mark VI.1.1 initializer, and has a good Master Available Table):

IS RECONFIGURATION DESIRED? <YES OR NO>

If a YES response is entered, the initializer displays the following request:

ENTER PACK TYPE - <U, S, R, OR I>

Once the appropriate response has been entered, the reconfiguration process is initiated.

Reconfiguration can be requested when using card input specifications by including an R between the <drive> and <serial-number> parameters.

NOTE

LIN #8290-002 must be installed in the Disk Pack Control in order to allow relocation of marginal sectors on B 9484-25/B 9484-55 Disk Packs when using either of the new Disk Pack Initializers. Without this LIN, an attempt by the initializer to initiate a RELOCATE operation will result in a system halt or hang, requiring manual clearing of the Disk Pack Electronics Controller (DPEC).

18.2.15. QWIKLOG

a. The Mix Number has been replaced by the Job Accounting Number in both the Job Summary and Graph listings. The Job Accounting Number corresponds in principle to the old Job Number in that it continuously increments, thus providing a unique identification number for every job. It is, however, accessible only to programs which process log files. The Job Number is a value which increments from 1 to 9999, and then is reset back to 1. This number now identifies the job in all MCP input and output messages, some of which previously used the mix number and others of which used the job number.

b. Memory Priority is now included in the Job Summary listing, appearing on the same line as the Schedule Priority and Processor Priority.

c. All uses of special characters not on the basic 48-character print train have been replaced with legal characters.

d. Abnormally-terminated jobs are now spotlighted in the Graph listing by replacing the "C" or "E" character for the last minute of any DS-ed or Aborted job with an "X".

e. The setting of the TABS option has no effect upon the output of QWIKLOG. Records generated by the MCP when the TABS option is set are ignored.

f. Two formatting irregularities within the Graph listing have been fixed. A job name component whose only special character is a leading asterisk (for example, *AARDVARK) is no longer quoted. A job with an empty file-id component on a user pack (such as USERPACK/YAK-TALLY/) now shows the trailing slash character which was omitted in the Mark VI.1 QWIKLOG.

g. QWIKLOG now verifies that it is being run under the correct software release and terminates in case of a mismatch. This ensures the validity of the printed output, as the meaning and format of log records is subject to substantial change between software releases.

18.2.14. SORT

a. A COLLATE table can now be specified directly in the SORT specifications, without the necessity of creating a file separately using SORT/COLLATE.

b. SORT can now produce an output TAG file compatible with COBOL or RPG index-sequential files.

c. The SORT output file is now explicitly closed; therefore, the output file no longer remains on disk or causes an existing file to be removed if the SORT intrinsic is DS-ed.

d. The "STABLE" option has been added, which allows the original order of records with duplicate keys to be maintained in the output file.

e. The SORT/SKELETON file is no longer required (nor is it used) when compiling a SORT program to LIBRARY.

18.2.15. STANDALONE/DISK.DUMP

This is the initial release of STANDALONE/DISK.DUMP, a disk-to-disk copy program that operates in a manner similiar to DISK/DUMP. STANDALONE/DISK.DUMP, however, copies the files on the input disk individually, rather than copying the entire disk (sector-by-sector). This results in a "squashed" output disk, as well as allowing the input disk to have "XD-ed" areas. Refer to the revised Software Operational Guide for details on this system utility program.

18.2.16. SYSTEM/BACKUP

a. Problems which caused multiple copies of some listings to not begin on a new page have been corrected.

b. Problems involving requests to print/punch all files on a backup tape (that is, PB MTx =/=) have been corrected.

c. Backup files with SPECIAL FORMS specified are now explicitly ignored by the AUTOPRINT mechanism.

d. A backup file created with the USER.BACKUP.NAME attribute is no longer removed from disk by SYSTEM/BACKUP following printing/punching. If SYSTEM/BACKUP switch 2 is set to 1, however, such backup files will be removed unless the SAVE, KEY, or RECORD option is specified in the PB input message.

18.2.17. SYSTEM/COMPARE

This is the initial release of SYSTEM/COMPARE, a utility program which can compare any two files. SYSTEM/COMPARE prints any records that fail to compare in both EBCDIC and hexadecimal representations, flagging each hexadecimal digit that is in error. Refer to the revised Software Operational Guide for details on this system utility program.

18.2.18. SYSTEM/ELOGOUT

This program has been completely rewritten in order to provide more detailed and comprehensive output. Errors are now reported in chronological order, sorted by unit-mnemonic and hardware address. Appropriate totals are summarized at the end of the output listing. Result descriptors and Extended Result Descriptors for errors on peripheral units are summarized and decoded. Errors on disk devices are sorted into ascending sequence, summarized by address and data transfer length, and decoded into the actual hardware locations (cylinder, track, and sector).

18.2.19. SYSTEM/ICMD.INIT

This is the initial release of SYSTEM/ICMD.INIT, a program used to initialize diskettes. Refer to the revised Software Operational Guide for details on this system utility program.

18.2.20. SYSTEM/MAKEUSER

a. If there are other jobs in the mix, SYSTEM/MAKEUSER requests verification before performing a CREATE. If programs are running under File Security when a CREATE is performed, unpredictable results can occur.

b. SYSTEM/MAKEUSER now checks to ensure that identical usercodes have the same default pack-id and security type (PUBLIC or PRIVATE).

c. The external names of both the input and output card files have been changed to NEW/USER.CODES.

18.2.21. SYSTEM/MARK.SEGS

This is the first release of SYSTEM/MARK.SEGS, a program used to mark specific program code segments as important for use with Extended Segment Decay and Priority Memory Management. Refer to the revised Software Operational Guide for details on this system utility program.

18.3. Known Errors and Restrictions.

18.3.1. COLDSTART/DISK

If there are pseudo decks on the input system disk, they will be copied to the output disk if a complete copy is requested. Such decks will be unusable, however, and cannot be removed.

18.3.2. DISKMAP/UTILITY

In Mark VII.0, the MCP does not allow a user disk to be accessed by a normal-state program unless the program has exclusive use of the disk. Therefore, it will still be necessary to cease all activity on the user disk prior to the start of Phase 1 in DISKMAP/UTILITY. This will be corrected in the Mark VIII.0 MCP.

18.3.3. QWIKLOG

Jobs terminated due to DEATH IN FAMILY are listed as "ABORTED" in the Job Summary. This MCP problem will be corrected in the Mark VIII.0 software release.

18.3.4. SORT

If a SORT parameter ends in position 96 of an input specification record, an erroneous syntax error may be generated.

18.3.5. SYSTEM/BACKUP

If either an output unit-mnemonic or COPIES is specified in the PB input message, the backup file is not removed from disk after printing/punching.

19. B 500 and 1400 Interpreters

19.1. Introduction.

This section contains descriptions of the Mark VII.0 enhancements to the B 500 and 1400 Interpreters, and the associated changes to the B 500 and 1400 Interpreter Environment Programs (IEP).

19.2. B 500 Interpreter and IEP.

19.2.1.
The B 500 Interpreter now supports the Magnetic Tape Memory Write (MWR) B 500 operator.

19.2.2.
The B 500 Interpreter now supports B 500 double density disk. The literal "DOUBLE.DENSITY" must be included in the IEP specifications for the EU to enable the double density mechanism. For example:

```
EUO NAME = A/B, DOUBLE.DENSITY;
```

A B 500 EU B 1800/B 1700 disk file must have been created (the "NEW" attribute set) with DOUBLE.DENSITY in order for the B 1800/B 1700 file size to be large enough to contain the 400,000 B 500 disk segments.

19.2.3.
The EOJ 999 option has been changed and enhanced. The literal "EOJ.999" is no longer recognized in the B 500 IEP specifications. The new syntax is "EOJ = <3 characters>;" (the equal sign is optional). The first of the <3 characters> must be the digit "9", and must be followed by two digits or characters. When the B 500 Interpreter executes a B 500 halt instruction with the M and N variants equal to the two digits or characters, a B 1800/B 1700 end-of-job occurs. For example:

```
EOJ = 999;
EOJ = 989;
```

The B 500 Interpreter command "EOJ999" has also been changed accordingly. The new syntax is "EOJ<3 characters>", where the <3 characters> format is the same as in the IEP specifications. For example:

```
<job-number>AXEOJ989
```

When executed with program switch 9 set to 1 and EOJ not specified in the B 500 IEP specifications, then the EOJ defaults to EOJ=999.

19.3. 1400 Interpreter and IEP.

19.3.1.

A provision has been added to the INTOPT options to allow 1400 object program decks to be separate from the data decks. When the INTOPT card "INTOPT SYSIN" is encountered, the 1400 Interpreter temporarily stops reading the card file (the card file remains open), and starts reading from the "SYSIN" file. Upon reaching end-of-file on the SYSIN file, the 1400 Interpreter resumes reading the regular card file. "SYSIN" can be looked upon as "inserting" the SYSIN file into the card file. The following example illustrates the use of the SYSIN option:

Card File	SYSIN File
? DATA CARDS.1400	? DATA CARDS.1400
INTOPT SYSIN	<object program deck>
<data deck>	? END
? END	

This has the same effect as:

```
? DATA CARDS.1400
<object program deck>
<data deck>
? END
```

The SYSIN file is by default a card file, but can be file-equated to a B 1800/B 1700 disk or tape file.

The 1400/IEP syntax has been enhanced to allow specification of the SYSIN file, and the CARD.READER syntax has been enhanced to facilitate specification of B 1800/B 1700 disk or tape files, as follows:

```
                    ( NAME = <PACK-ID>/<FAMILY-NAME>/<FILE-NAME> )
                    (                                            )
( CARD.READER )     ( ( PACK.ID )                                )
(             )     ( (         ) = <PACK-ID>                    )
( SYSIN       )     ( ( PID     )                                ) ;
                    (                                            )
                    (           ( CARD.READER )                  )
                    ( DEVICE =  ( CRD         )                  )
                    (           ( DISK        )                  )
                    (           ( TAPE        )                  )
```

The following example specifies that the SYSIN file is a disk file labeled "1400/PROG" and that the card file is a card deck labeled "CARDS":

```
CARD.READER NAME = CARDS, DEVICE = CARD.READER;
SYSIN  NAME = 1400/PROG, DEVICE = DISK;
```

The default name for both the CARD.READER and SYSIN files is "CARDS.1400". "DEVICE = CARD.READER" is the default if a device is not specified. Also note that the disk file is not a pseudo reader, but is created using the "CRDDSK" function of DMPALL (or any similar card-to-disk program).

All INTOPT cards must precede any 1400 program or data cards. The SYSIN file may contain INTOPT cards, as long as the preceding rule is not violated. Upon encountering an INTOPT RCB.ALLOWED card, both the SYSIN file and the card file are closed, and the binary card file is then opened.

The B 1800/B 1700 internal file name for the SYSIN file is "SYSIN". If the environment was compiled with a 1400/IEP prior to the Mark VII.0 release, the internal name of the SYSIN file is "CARDS.2".

20. TABS

20.1. Introduction.

The Burroughs Electronic Data Processing Time Analysis and Billing System (TABS), as implemented for the B 1800/B 1700 systems, provides each user with the ability to obtain a comprehensive analysis of the system log files.

TABS is designed to analyze the log files, using the extracted statistics to generate reports on machine utilization and program performance. As the reports are produced, month-to-date and billing-period-to-date statistics are maintained in permanent data files. Together with the information on installation costs assigned by the user, these statistics can be used to distribute the cost of the entire system equitably among the individuals, departments, or applications utilizing the computer system.

The TABS package, consisting of ten separate programs, is a program product which may be ordered separately. Documentation on the use of the TABS programs is available with the program product in the form of a printer backup file labeled DOCUMENT/TABS. The B 1800/B 1700 TABS package can be ordered through the local Burroughs branch office from Program Products Distribution.

B1800/B1700 System Software Release Mark VII.0

1. General Information

1.1. Introduction.

The Mark VII.0 software release includes a number of major enhancements, described in detail in the following sections of this release document. Among the new enhancements and features are the following (the numbers in parentheses refer to the paragraphs in this document that describe the features in detail):

1. New memory management algorithms have been implemented in the MCP, using three optional levels of sophistication and complexity (2.2.2).

2. Capabilities for dynamically monitoring and displaying system performance have been implemented in the MCP and GISMO/DEBUG (2.2.3).

3. Mix-numbers have been replaced by job-numbers, allowing use of a single, consistent number for all job identification (2.2.4).

4. The XM function has been implemented in the MCP and System Initializer (SYSTEM/INIT), allowing specified areas of memory to be removed from use during the CLEAR/START operation (2.2.10).

5. The meaning of the STATE light has been changed to allow providing a more accurate and meaningful indication of how "busy" the processor is (2.2.13).

6. The B 1800/B 1700 Time Analysis and Billing System (TABS) has been implemented, including new event-oriented records placed in the SYSTEM/LOG by the MCP (2.2.39 and 20.1).

7. The capability to create Installation Allocated Disk files has been implemented in the MCP and a new utility program, DISK/ALLOCATOR (2.2.40 and 18.2.5).

8. The AUTOLOAD facility has been implemented in the MCP and DISKETTE/COPY, allowing pseudo-reader files on diskette to be copied to system disk without operator action (2.2.44 and 18.2.6).

9. The two methods for I/O SEQUENTIAL disk file access, originally implemented in patch #40 to Mark VI.1, have been included in the Mark VII.0 MCP and MICRO-MCP (2.2.64).

10. The performance of the DIV (divide) and SQRT (square root) operations in RPG has been significantly improved (4.2.9).

11. The maximum number of files that may be declared in an RPG source program has been increased to 31 (4.4.1).

12. Source file maintenance capability has been implemented in RPG, allowing disk or tape source-image files to be updated with patch records, as well as providing the source library copy facility (4.4.2).

13. The EXTEND option of the OPEN verb has been implemented in COBOL, allowing an existing disk file to be opened output with automatic positioning of the record pointer to the end of the file (5.2.15 and 5.4.1).

14. The COBOL compiler now segments the generated subroutines used for handling ISAM files (5.4.11).

15. The free-format I/O ("slash editing") capability has been implemented in FORTRAN (7.4.1).

16. DMSII now provides record-level (rather than physical block-level) lockout (9.2.3).

17. The DMSII reorganization programs are now restartable (9.2.4).

18. A number of standardization changes have been made to DASDL syntax, including implementation of new compiler options (9.2.6).

19. In conjunction with Priority Memory Management and Extended Segment Decay, data comm peformance has been significantly improved (10.2.7).

20. The capability to declare global defines has been provided in the NDL compiler (10.4.5).

21. It is now possible for an MCS to perform a REMOTE-FILE-INFO operation on any remote file (10.4.14).

22. CANDE can now support up to 64 terminals logged on at the same time (11.2.6).

23. The CANDE AUDIT capability can now be dynamically started or stopped, using operator commands (11.2.10 and 11.4.13).

24. TEXT/EDITOR now provides the ability to display a printer backup disk file at the terminal. (12.2.3).

25. TEXT/EDITOR includes a number of significant improvements in the FIND, REPLACE, and SEARCH commands (12.2.5 and 12.2.6).

Program Identification	Version	Compile Date	Remarks
*SMCS	VII.0	4/28/78	
CANDE	VII.0	5/19/78	
CANDE/ANALYZER	VII.0	1/24/78	
CANDE/TEACH.FILE	VII.0	11/ 9/77	
TEXT/EDITOR	VII.0	6/30/78	
RJE	VII.0	1/17/78	
RJE/DCH	VII.0	1/17/78	
RJE/MCS	VII.0	1/17/78	
RJE/AUTOBACKUP	VII.0	10/10/77	
RJE/CONTROLLER	VII.0	1/30/78	
RJE/MESSAGES	VII.0	1/18/78	
*RJE3780	VII.0	6/ 6/78	
HASP	VII.0	4/14/78	
HASP/SPOOL	VII.0	5/24/78	
HASP/MODIFIER	VII.0	2/ 6/78	
B500/IEP	VII.0	9/30/77	
B500/INTERP2U	VII.0	7/12/78	
1400/IEP	VII.0	12/ 1/77	
1400/INTERP2U	VII.0	4/26/78	
1400/CREATE1311	VI.0	9/ 6/76	
SORT	VII.0	3/22/78	
SORT/COLLATE	VI.1	8/19/77	
SORT/MERGE	VII.0	9/ 9/77	
SORT/QSORT	VII.0	4/ 7/78	
SORT/TAPESORT	VII.0	11/ 7/77	
SORT/VSORT	VII.0	3/23/78	
MCPII/ANALYZER	VII.0	6/30/78	
DUMP/ANALYZER	VII.0	6/22/78	
CODE/ANALYZER	VII.0	12/ 7/77	
DISK/COPY	VII.0	3/29/78	
DISKETTE/COPY	VII.0	4/26/78	
*SYSTEM/ICMD.INIT	VII.0	12/21/77	
SYSTEM/BACKUP	VII.0	6/12/78	
*SYSTEM/COMPARE	VII.0	6/28/78	
SYSTEM/LOGOUT	VII.0	1/24/78	
SYSTEM/ELOGOUT	VII.0	6/ 3/78	
SYSTEM/SPOLOGOUT	VI.1	8/22/77	
SYSTEM/LDCONTRL	VII.0	11/ 3/77	
SYSTEM/LOAD.CAS	VII.0	9/30/77	
SYSTEM/LOAD.DUMP	VII.0	5/23/78	
SYSTEM/DISK.INIT	VII.0	6/ 5/78	
SYSTEM/DISK.DUMP	VII.0	7/28/77	
SYSTEM/BUILDTRAIN	VI.1	8/19/77	
SYSTEM/TRAINTABLE	VII.0	10/21/77	
INPUT/PC5.TABLES	VII.0	10/21/77	
SYSTEM/MAKEUSER	VII.0	1/27/78	

Program Identification	Version	Compile Date	Remarks
*SYSTEM/MARK.SEGS	VII.0	3/23/78	
*DISK/ALLOCATOR	VII.0	8/22/77	
SYCOPY	VII.0	8/ 4/77	
TAPECOPY	VII.0	8/22/77	
CHECK/LOAD.DUMP	VII.0	6/ 3/78	
DISKMAP/UTILITY	VII.0	6/ 3/78	
QWIKLOG	VII.0	6/28/78	
LOGCONVERT	VII.0	6/28/78	
FILE/LOADER	V.1	10/29/75	
FILE/PUNCHER	V.0	1/31/75	
DMPALL	VII.0	6/27/78	
*TABS/LOGOUT	VII.0	4/15/78	
*TABS/ACCTS	VII.0	4/13/78	
*TABS/BILLING	VII.0	5/ 9/78	
*TABS/ERR	VII.0	5/11/78	
*TABS/EXEC	VII.0	5/ 3/78	
*TABS/HDWR	VII.0	4/17/78	
*TABS/MIX	VII.0	4/13/78	
*TABS/REPORTGEN	VII.0	4/17/78	
*TABS/SUM	VII.0	4/13/78	
*TABS/UPDATE	VII.0	4/13/78	
DISK/DUMP	VI.1	8/19/77	Note 3
*STANDALONE/DISK.DUMP	VII.0	6/12/78	Note 3
DISK PACK INITIALIZER	VII.0	6/27/78	Note 3
DISK CARTRIDGE INITIALIZER	VII.0	8/29/77	Note 3
DISKPACK/INTERCHANG	VI.0	12/ 2/76	
STANDALONE/INTERCHANG	VI.1	8/19/77	Note 3
SYSTEM/INIT	VII.0.1	7/ 1/78	
CLEAR/START	VII.0	2/17/78	Note 3
COLDSTART/TAPE	VII.0	9/30/77	Note 3
COLDSTART/DISK	VII.0	12/ 2/77	Note 3
SSLOAD/MAKCAS	VII.0	12/ 5/77	
CASSETTE/MAKER	VII.0	9/19/77	
CASSETTE/LOADER	VI.1	9/ 7/76	Note 3
SDL/INTERP1U	VII.0	10/10/77	Note 3
GISMO/SA	VII.0.1	7/ 1/78	Note 3

Note 1: Interpreter is usable only on B 1710 and B 1830/B 1825 processors.

Note 2: Interpreter is usable only on B 1720 and B 1800 processors (other than the B 1830/B 1825).

Note 3: Code file used in making stand-alone program cassettes, using the SSLOAD/MAKCAS program. Attempts to execute the code file under MCP control should not be made.