B1000 SYSTEM SOFTWARE RELEASE MARK 9.0

DOCUMENT/SYSNOTE1

SYSTEM NOTES

1. General Information

1.1. Introduction.

The Mark 9.0 software release includes a number of new program products and major enhancements, described in detail in the following sections of this release document.

References to the B1000 series include the B1900, B1800, and B1700 systems.

Starting with the Mark 9.0 release of the B1000 System Software, the release level numbers are in decimal instead of Roman numeral. This increases readability and compatibility with other Burroughs program product releases.

The SYSTEM NOTES are divided into two documents, SYSNOTE1 and SYSNOTE2. SYSNOTE1 contains all of the new enhancements and any temporary documentation for the software release. SYSNOTE1 consists of a general section and a section for each program product. A table of contents is provided to determine the section number in which a program product is documented. Each section has an Introduction subsection and optionally an Enhancement subsection and a Temporary Documentation subsection numbered 1, 2, and 3 respectively. The subsection number follows the section number with a period between them. Paragraphs are numbered in ascending order within a subsection. The paragraph number follows the subsection number with a period between them. SYSNOTE2 contains all of the known errors and restrictions listed in alphabetic order by Style-ID and program name.

Disk file headers have changed for the Mark 9.0 release. Mark 9.0 MCP allows a Mark 7.0 user pack to be readied under Mark 9.0; the MCP converts all Mark 7.0 disk file headers to the Mark 9.0 format. When changing the system software from Mark 9.0 to Mark 7.0, all user packs must first be readied on the Mark 8.0 release of the MCP, as the Mark 7.0 MCP is not able to convert Mark 9.0 disk file headers. No code file executed under the Mark 9.0 system software can be exected under any prior release of the system software. No system utility program from prior releases of the system software can be executed sucessfully under the Mark 9.0 software.

1.2. Enhancements.

Three new program products are available with the Mark 9.0 release of the system software:

1. The ANSI 74 COBOL (COBOL74) compiler program product.

2. The ANSI 77 FORTRAN (FORTRAN77) compiler program product.

3. The Interactive Basic (IBASIC) compiler program product scheduled for release approximately one quarter after the initial release of the Mark 9.0 system software.

The Mark 9.0 release also provides many important new functional capabilities and enhancements. The following paragraphs list some of the new features in this release.

1. Major changes have been made to the SDL and RPG interpreters to increase their performance.

2. New file attributes have been added and are available through the "FILE" equation control command.

3. The mechanism for writing to a printer file has been changed in the MMCP to speed up the process by creating fewer physical writes to the printer and to accommodate ANSI 74 COBOL requirements.

4. The output of the "CU" command has been enhanced to more completely indicate the allocation of system memory and/or program memory.

5. The "LC" command is no longer used to execute SYSTEM/LOAD.CAS. The "LC" command now allows a comment to be entered in the system log. SYSTEM/LOAD.CAS must now be executed explicitly.

6. The "WS" command now allows syntax to query the status of either the active schedule or the WAITING schedule, or both.

7. The MCP now accepts more than two operator commands in a single input string.

8. The MCP now allows a new disk file accessing method named "RELATIVE".

9. The "SD" message may now be used to specify additional system drives which are on different channels than the original system disk, as long as no two of these system disks are on the same unit number.

10. Up to 32 spindles of disk can now be on line at the same time.

11. For the Mark 9.0 release of the RPG compiler, the "*ALL" entry has been implemented, making the coding of RPG-DMS programs easier.

12. A program can now open a remote file with "SIMPLE.HEADERS" which provides the capabilities of a "HEADERS" file without the responsibilities of a Message Control System (MCS).

13. In the Mark 9.0 release of NDL, an MCS may now request that the LSN for a station being detached from a remote file be detached from the File Information Block (FIB) of the remote file (DETACH with CLEAR).

14. Two new "NDL/LIBRARY" files "COBOL74DEC" and "COBOL74SEL", have been added to permit COBOL74 programs to issue a "SEND" with advancing feature (for CRT devices only).

    New "MCS" control messages have been added for the Mark 9.0 release of NDL to accomodate ANSI 74 COBOL requirements.

15. Program switch number 5 of CANDE now determines whether CANDE is to run with file security.

16. For users of CANDE, a family name and a pack-id can now be specified for an output file if the user is logged on to CANDE with a privileged usercode/password.

17. Two new file types have been added to the CANDE program: COBOL74 and FORTRAN77.

18. The SYCOM program now allows DMS files to be transfered.

19. The format of the "JOBS" file in the SMCS program was changed to allow a user to specify a program-id in the "SIGN ON", "PASS", or "START" command to differ from the program name zip executed by SMCS as a result of the "SIGN ON", "PASS", or "START" command. In addition, the flags for the various program attributes have been simplified.

20. The HASP program now allows a three-character remote ID field in the HASP/PARAMETERS file.

21. PACK/INIT and SYSTEM/DISK.INIT now allow initialization of a single cylinder of a pack, without disturbing other data on the pack, providing the entire cylinder is available.

22. The new program CREATE/TABLE provides the ability to create user-specified translate tables.

23. The ability to print "BANNERS" has been added to SYSTEM/BACKUP.

24. The "COMPARE" function of the SYSTEM/COPY program can now be used independently of the "COPY" function.

25. The print fields in the "TABS/SUM" program have been increased to four digits, allowing a figure greater than 999 hours.

26. LOGCONVERT now retains the parent job number when a job is spawned.

Many of the items listed within the Enhancement subsections are fixes to problems reported in SYSNOTE2 for the Mark 8.0 release. These items are indicated by the FTR number that reported the problem surrounded by parentheses.

1.3. Temporary Documentation.

1.3.1. B1000 Software Flashes.

System Flashes provide a method of rapid dissemination of pertinent information in regard to program products. The text of a System Flash is as complete and concise as possible. The main purpose of a Flash is to expedite information about software problems that require immediate attention.

The flash number includes the release level to which the flash applies and when a new release of software is issued, all previous flashes are, in effect, cancelled. If the problems described by flashes to prior releases continue to apply, documentation regarding these problems will contained to be either in the permanent documentation, the release documentation, or reissued as a flash with a new number.

1.3.2. Field Trouble Report (form 1912003).

The standard form for reporting B1000 system software problems is the Field Trouble Report (FTR). This is a single page, six-copy form which provides TIO with information to process the report and resolve the problem being reported.

The following information and guidelines have been included in the System Notes in order that every one involved with the FTR process has a better understanding of the procedure and requirements when filling out an FTR.

When a problem is encountered in a customer installation it should be reported to the local Burroughs representative. If the problem cannot be solved at a local level an FTR should be filled out by the local Burroughs representative and forwarded to TIO

through the District office, or through Subsidiary Head Offices in the International Group.

The local personnel who fill out FTR's must be certain that the FTR is accompanied by all of the necessary supporting documentation or media.

When TIO receives the FTR, TIO acknowledges receipt of an FTR by mail within one working day and then conducts a thorough investigation to determine the extent of the problem. If the problem does not require a software change, TIO answers the FTR and returns the answer to the originator through the District/Subsidiary Office. If the problem requires software changes, the FTR is sent to the plant for correction. When the correction is made, the plant sends the answer to TIO and TIO forwards the answer to the originator through the District/Subsidiary Office.

The following guidelines must be followed when submitting FTR's:

A separate FTR for each problem.

Use FTR form (#1912003).

Use the FTR for reporting system software problems; not for requesting new features, reporting documentation errors, or asking questions.

Make it legible on all copies, preferably typewritten.

Ensure that all attachments are labeled with the FTR number and are listed on the FTR form.

There are specific instructions that must be followed when completing the FTR form. These instructions are listed on the back of the last copy (originator's copy) of the FTR form.

Attachments to an FTR:

The FTR should include any information that may help solve the problem. Some attachments that should always be considered for inclusion are:

A full memory dump that has been properly analyzed by MCPII/ANALYZER. All questions on the face of the dump must be answered and registers requested filled in if not done by the analyzer.

A program dump, if applicable.

A program trace file listing, if applicable, such as a DC/AUDIT listing for data communications programs, an SMCS/TRACE file listing for the SMCS program, or a CANDE/ANALYZER listing for the CANDE program.

A copy of the SPO log entries for the pertinent time period.

A code listing of the erroneous compiler output.

A source listing of user programs involved in the problem.

Source and object files on machine readable media.

Observations by site personnel that are pertinent to the problem.

Any notes, information, and comments by the person analyzing the problem.

Each attachment to the FTR must be labeled with the FTR number.

Return of Included Media

Machine readable media is returned if requested. Each item to be returned must be labeled with the return address.

Data Management Problems

Additional information may be required for data management problems. The following items should be considered for DMSII FTRs:

A small machine readable test case.

A machine readable copy of the DASDL source file and any applicable AUDIT files.

It may be necessary to have the local Burroughs representative run special debug traces of the database and include them with the FTR.

Memory Dumps

Both the program dump and full system dumps are valuable. Program dumps are useful when the problem is restricted to the program environment. Otherwise send a full system dump. (When in doubt, send a full system dump along with the program dump).

1.3.3. Memory Requirements.

Burroughs has consistently continued to implement new software features for the B1000, most of which have been at the request of our users.

Although every effort has been made to implement such features without an accompanying increase in system resource requirements, notably memory, it is becoming increasingly apparent that this is

not possible without seriously impacting the implementation and efficiency of such new features.

The effects, on users, of such increases in resources varies from release to release and from user to user. In general, the MCP's minimum permanent resident space has continued to increase each release. Non-permanent space requirements (e.g. space required for temporary tasks such as opening/closing a file, BOJ, EOJ etc.) also continue to increase each release, by varying amounts, as more efficient structures are implemented.

A possible decrease in efficiency from release to release is not necessarily a function of the system's memory size, but rather the efficiency with which that space is utilized. An installation which is not thrashing, and not even close to thrashing, will most likely not experience ary decrease in efficienty when implementing a new release. On the contrary, it is most likely that system performance will improve, due to new features, more efficiency structures, etc.

On the other hand, an installation which is already thrashing, will probably notice a deterioration in throughput ranging from mild (a few percent) to such severity that run times could approach infinity. An installation which was not thrashing on a previous release (but was dangerously close) might require, with the new release, an increase in memory sufficient to cause thrashing for the first time. Once again, the severity of thrashing depends on how close the installation was to thrashing prior to installation of the new release.

If it is suspected that thrashing is occurring, or may occur, Burroughs strongly recommends analysis of possible impact on the system's performance, before installing the new release.

The following is a comparison of memory requirements for the last three releases of software for the B1000.

	Mark 7.0	Mark 8.0	Mark 9.0
B1860 MEMORY REQUIREMENTS			
STATIC SYSTEM SPACE	21850	23335	26254
MMCP	3512	3536	5250
TOTAL SYSTEM SPACE	25362	26871	31504
RPG program + INTERPRETER	18469	18702	20547
B1860/RPG program THRASHING POINT	43831	45573	52051
B1714 MEMORY REQUIREMENTS			
STATIC SYSTEM SPACE	21850	23335	26254
LESS B1860 CODE	- 1754	- 1754	- 1754
B1714 STATIC SYSTEM SPACE	20096	21581	24500
MMCP	3512	3536	5250
TOTAL SYSTEM SPACE	23608	25117	29750
RPG program + INTERPRETER	18469	18702	20547
B1714/RPG program THRASHING POINT	42077	43819	50297

1.3.4. Software Included with the Mark 9.x 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 9.0.

Program Identification	Version	Compile Date	Remarks
AUDIT/ANALYZER	9.0	9/18/79	
BAS.INTRN3/AGGREGATE	6.1	5/09/77	
BASIC	9.0	7/13/79	
BASIC/INTERP3M	9.0	6/11/79	
BASIC/INTERP3S	9.0	6/11/79	
B500/IEP	9.0	9/13/79	
B500/INTERP2U	9.0	9/13/79	
CANDE	9.0	1/14/80	
CANDE/ANALYZER	9.0	6/24/79	
CANDE/TEACH.FILE	9.0	10/26/79	
CART/INIT	7.0	9/12/77	NOTE 3
CASSETTE/LOADER	6.1	9/07/76	NOTE 3
CASSETTE/MAKER	9.0	7/30/79	
CHECK/LOAD.DUMP	9.0	7/19/79	
CLEAR/START	9.0	11/27/79	NOTE 3
COBOL	9.0	1/24/80	
COBOL/INTERP1M	9.0	6/12/79	NOTE 2
COBOL/INTERP1S	9.0	6/12/79	NOTE 1
COBOL/XREF	8.0	5/15/79	
*COBOL74	9.0	1/16/80	
*COBOL74/INTERP1M	9.0	12/13/79	NOTE 2
CODE/ANALYZER	9.0	2/07/80	
COLDSTART/DISK	9.0	2/28/80	NOTE 3
COLDSTART/TAPE	8.0	4/23/79	NOTE 3
*CREATE/TABLE	9.0	10/03/79	
DASDL	9.0	2/12/80	
DASDL/ANALYZER	8.0	1/17/79	
DB/MAP	9.0	2/14/80	
DC/AUDIT	8.0	1/04/79	
DISK/ALLOCATOR	9.0	8/24/79	
DISK/DUMP	7.0	8/19/77	NOTE 3
DISKETTE/COPY	9.0	11/05/79	
DISKMAP/UTILITY	9.0	8/13/79	
DISKPACK/INTERCHANG	9.0	9/30/79	
DMPALL	9.0	8/14/79	
DMS/BUILDINQ	9.0	2/08/80	
DMS/HELPINQ	9.0	10/05/79	
DMS/INQUIRY	9.0	2/16/80	
DMS/REORG.READ	9.0	8/31/79	
DMS/REORG.WRIT	9.0	2/26/80	
DUMP/ANALYZER	9.0	2/15/80	
FILE/LOADER	9.0	6/14/79	
FILE/PUNCHER	9.0	6/22/79	
FOR.INTRIN	8.0	6/15/78	

FORTRAN	9.0	7/16/79	
FORTRAN/INTERP1M	9.0	6/11/79	NOTE 2
FORTRAN/INTERP1S	9.0	6/11/79	NOTE 1
FORTRAN/INTMAKER	5.1	1/12/76	
*FORTRAN77	9.0	1/24/80	
*FORTRAN77/ANALYZER	9.0	12/26/79	
*FORTRAN77/INTERP3M	9.0	1/31/80	NOTE 2
*FORTRAN77/INTRINSICS	9.0	11/30/79	
GISMO	9.0	2/14/80	
GISMO/DEBUG	9.0	2/14/80	
GISMO/SA	9.0	2/14/80	
HASP	9.0	2/09/80	
HASP/MODIFIER	9.0	9/04/79	
HASP/SPOOL	9.0	5/16/79	
INITIALIZE/ANALYZER	8.0	3/28/79	
INPUT/PCS.TABLES	8.0	1/26/79	
LDSAD	8.0	2/02/79	
LOGCONVERT	9.0	2/06/80	
MCPII	9.0	2/27/80	
MCPII/ANALYZER	9.0	2/01/80	
MCPII/MICRO.MCP	9.0	1/28/80	
MICRO.MCP/DEBUG	9.0	1/28/80	
NDL	9.0	10/04/79	
NDL/ADDRESS	9.0	1/29/80	
NDL/DUMP	9.0	6/05/79	
NDL/LIBRARY	9.0	2/14/80	
NDL/MACRO	9.0	1/29/80	
PACK/INIT	9.0	9/30/79	NOTE 3
QWIKLOG	9.0	10/23/79	
RD	9.0	2/06/80	
RECOVER/DATA.BASE	9.0	9/10/79	
RJE	7.0	1/17/78	
RJE/AUTOBACKUP	9.0	10/08/79	
RJE/CONTROLLER	8.0	5/25/79	
RJE/DCH	7.0	1/17/78	
RJE/MESSAGES	8.0	5/29/79	
RJE3780	9.0	11/02/79	
RPG	9.0	1/09/80	
RPG/BTF	8.0	10/30/78	
RPG/INTERP1M	9.0	7/30/79	NOTE 2
RPG/INTERP1S	9.0	7/30/79	NOTE 1
RPG/REORD	8.0	9/06/78	
RPG/REORG	8.0	9/06/78	
RPG/XREF	8.0	1/02/79	
SDL.INTRIN/AGGREGATE	8.0	2/07/80	
SDL/ERRORS	9.0	4/23/79	
SDL/INTERP1M	9.0	8/24/79	NOTE 2
SDL/INTERP1S	9.0	8/24/79	NOTE 1
SDL/INTERP1U	9.0	8/24/79	NOTE 3
SDL/XMAP	9.0	7/31/79	
SDL/XREF	9.0	7/30/79	
SMCS	9.0	2/15/80	
SORT	9.0	10/31/79	
SORT/MERGE	9.0	8/09/79	

SORT/QSORT	7.0	4/07/78	
SORT/TAPESORT	9.0	2/06/80	
SORT/UTILITY	9.0	2/13/80	
SORT/VSORT	9.0	2/06/80	
SQUASH/USER.DISK	9.0	8/21/79	
SSLOAD/MAKCAS	9.0	8/20/79	
STANDALONE/DISK.DUMP	9.0	1/11/80	NOTE 3
STANDALONE/INTERCHANG	9.0	9/30/79	NOTE 3
SYCOM	9.0	11/19/79	
SYSTEM/BACKUP	9.0	11/08/79	
SYSTEM/BUILDTRAIN	8.0	10/24/78	
SYSTEM/COMPARE	9.0	10/23/79	
SYSTEM/COPY	9.0	11/29/79	
SYSTEM/DISK.DUMP	9.0	1/23/80	
SYSTEM/DISK.INIT	9.0	9/30/79	
SYSTEM/ELOGOUT	9.0	9/25/79	
*SYSTEM/FILE.INIT	9.0	9/18/79	
SYSTEM/ICMD.INIT	8.0	5/07/79	
SYSTEM/INIT	9.0	12/14/79	
*SYSTEM/IS.MAINT	9.0	1/18/80	
SYSTEM/LDCONTRL	7.0	11/03/77	
SYSTEM/LOAD.CAS	9.0	10/15/79	
SYSTEM/LOAD.DUMP	9.0	7/25/79	
SYSTEM/LOGOUT	9.0	8/23/79	
SYSTEM/MAKEUSER	8.0	10/12/78	
SYSTEM/MARK.SEGS	9.0	5/26/79	
SYSTEM/SPOLOGOUT	9.0	10/01/79	
SYSTEM/TRAINTABLE	8.0	1/26/79	
TABS/ACCTS	9.0	9/24/79	
TABS/BILLING	9.0	2/06/80	
TABS/DMS	9.0	10/01/79	
TABS/ERR	9.0	10/01/79	
TABS/EXEC	9.0	9/25/79	
TABS/HDWR	9.0	9/25/79	
TABS/LOGOUT	9.0	10/31/79	
TABS/MIX	9.0	9/25/79	
TABS/REPORTGEN	9.0	2/06/80	
TABS/SUM	9.0	10/01/79	
TABS/UPDATE	9.0	2/06/80	
TAPECOPY	9.0	10/29/79	
TEXT/EDITOR	9.0	8/31/79	
TRANSLATE/BACKUP	9.0	2/08/80	
UPL	9.0	7/30/79	
1400/CREATE1311	6.0	9/06/76	
1400/IEP	9.0	9/31/79	
1400/INTERP2U	9.0	12/09/79	

Note 1: Interpreter is usable only on 81710 and 81830/81825 processors.

Note 2: Interpreter is usable only on 81900, 81800, and 81720 processors (other than the 81830/81825).

Note 3: Code file used in making stand-alone program cassettes by the SSLOAD/MAKCAS program. Attempts to execute the code file under MCP control should not be made.

1.3.5. B 1000 Reference Manuals.

The following manuals have been released:

#1067535	B 1800/B 1700	BASIC Reference Manual (6-79)
#1090586	B 1700	CANDE User's Manual (8-78)
#1057197	B 1800/B 1700	COBOL Reference Manual (8-78) PCN #2 (7-79)
#1090651	B 1800/B 1700	Data Comm Audit Reference Manual (3-79)
#1089992	B 1700	Data Communications Information Manual (12
#1127222	B 1000	Data Management System II (DMSII) Reference Manual (11-79)
#1108875	B 1000	Data Management System II Inquiry Reference Manual (12-79)
#1081882	B 1800/B 1700	FORTRAN Reference Manual (8-78)
#1090578	B 1800/B 1700	HASP Reference Manual (11-78)
#1068010	B 1700	MCP Reference Manual (8-75)
#1077568	B 1700	MIL Reference Manual (6-77)
#1073715	B 1800/B 1700	NDL Reference Manual (9-78)
#1090602	B 1800/B 1700	RJE Reference Manual (7-78)
#1057189	B 1800/B 1700	RPG Reference Manual (8-79)
#1081346	B 1700	SDL Reference Manual (12-74)
#1090644	B 1870/B 1860	Systems Reference Manual (3-79)
#1108982	B 1800/B 1700	System Software Operatior Guide Volume 1 (6-79) PCN #1 (2-80)
#1108966	B 1800/B 1700	System Software Operation Guide Volume 2 (1-80)
#1090669	B 1800/B 1700	TABS Reference Manual (3-79)
#1090610	B 1800/B 1700	TEXT/EDITOR Reference Manual (4-79)
#1067170	B 1700	UPL Reference Manual (12-73)
#1090636	B 1800/B 1700	3780 RJE Reference Manual (4-79)

1.3.6 Mark 9.0 Release Documentation.

A PCN to the B1000 System Software Operation Guide Volume 1 dated (2-80) is now available for distribution. This revision of the manual contains detailed descriptions of all new MCP features available with the Mark 9.0 release, and should be reviewed prior to installation of this software level.

References are made throughout this release document to new features and programs to where specific documentation is contained in the revised B1000 System Software Operation Guide or in one of the supplements to this SYSNOTE1 document. Detailed information on such features and programs is not included in these SYSNOTE1.

All backup files have upper and lower case letters and must be converted to upper case if a 96 character printer is not

available. A utility program named TRANSLATE/BACKUP is provided on the system tape to translate the files from lower to upper case characters.

The following paragraphs contain the names and a brief description of the supplemental documentation released along with the program products in this release.

1.3.7
DOCUMENT/B1830 18 PAGES ALL B1830 USERS

DOCUMENT/B1830 describes the operating instructions for the B1830.

1.3.8.
DOCUMENT/CANDE 139 PAGES ALL USERS OF CANDE

DOCUMENT/CANDE contains the documentation for the CANDE program product relative to the Mark 9.1 release.

1.3.9.
DOCUMENT/COBOL74 574 PAGES ALL USERS OF COBOL74

DOCUMENT/COBOL74 is the preliminary documentation for COBOL74.

1.3.10.
DOCUMENT/DMS 61 PAGES ALL USERS OF DMSII

DOCUMENT/DMS contains the temporary documentation for DMS.

1.3.11.
DOCUMENT/DMSRELLET 72 PAGES ALL USERS OF DMSII

DOCUMENT/DMSRELLET contains the Mark 8.0 system notes for DMSII.

1.3.12.
DOCUMENT/DMSUTILS 50 PAGES ALL USERS OF DMSII

DOCUMENTATION/DMSUTILS contains the temporary documertation for the DMSII debugging programs.

1.3.13.
DOCUMENT/FORTRAN77 227 PAGES ALL USERS OF FORTRAN77

DOCUMENT/FORTRAN77 contains the preliminary documentation for FORTRAN77.

1.3.14.
DOCUMENT/NDL 26 PAGES ALL USERS OF NDL

DOCUMENT/NDL contains the temporary documentation for NDL.

1.3.15.
DOCUMENT/RD 117 PAGES ALL B1000 USERS

DOCUMENT/RD is the preliminary reference manual for the RD program product.

1.3.16.
DOCUMENT/RPG.DMS 30 PAGES ALL USERS OF RPG

DOCUMENT/RPG.DMS contains the documentation on the RPG interface to DMSII.

1.3.17.
DOCUMENT/SMCS 147 PAGES ALL USERS OF SMCS

DOCUMENT/SMCS is the preliminary reference marual for the SMCS program product.

1.3.18.
DOCUMENT/SORT 76 PAGES ALL B1000 USERS

DOCUMENT/SORT is the preliminary reference manual for the sort programs.

1.3.19.
DOCUMENT/SYCOM 215 PAGES ALL USERS OF SYCOM

DOCUMENT/SYCOM is the preliminary reference manual for the SYCOM program product.

1.3.20.
DOCUMENT/SYSNOTE1 139 PAGES ALL B1000 USERS

DOCUMENT/SYSNOTE1 is the name of this document.

1.3.21.
DOCUMENT/SYSNOTE2 30 PAGES ALL B1000 USERS

DOCUMENT/SYSNOTE2 contains the known errors and restrictions for the Mark 9.0 release.

1.3.22.
DOCUMENT/UTILITY 47 PAGES ALL B1000 USERS

DOCUMENT/UTILITY contains temporary documentation for the system utilities.

2. MCPII

2.1. Introduction.

The Mark 9.0 release of the Master Control Program (MCP) contains numerous enhancements, and corrections to facilitate system utilization. Implementation of the COBOL74 language has necessitated changes in the handling of printer files, and has added certain file attributes that are available to all languages. These changes and enhancements are documented in the following paragraphs.

The Mark 9.0 MCP permits a Mark 7.0 user pack to be readied under Mark 9.0. The MCP converts all Mark 7.0 disk file headers to the Mark 9.0 format. When changing the system configuration from Mark 9.0 to Mark 7.0 all user packs must first be readied on the Mark 8.0 release of the MCP, as Mark 7.0 is not equipped to handle 9.0 disk file header formats.

Object programs compiled or executed under the Mark 9.0 system software can not subsequently be executed under prior released system software. Also, due to the changes to the Disk File Headers, all system utilities from prior released system software should not be executed under Mark 9.0.

It is recommended that the Logic Improvement Notice (LIN) 8282-039 be installed on all B1855 dual processor systems along with the Mark 9.0 release. This LIN identifies which processor is the Master, and which is the Slave, and aids in system debugging, as this information is printed in the system memory dump. LIN 8282-039 will be required to execute the Mark 10.0 System Software on a dual processor.

2.2. Enhancements.

2.2.1.
File attributes have been added to support the implementation of logical page, and variable length records in COBOL74. Besides being included in the COBOL74 syntax, these attributes are available to all languages through the "FILE" equation control command. Refer to Temporary Documentation for a description of these attributes.

2.2.2.
The mechanism for writing to a printer file has been changed in the MMCP to speed up the process by creating fewer physical writes to the printer. The MMCP now tanks logical writes to a printer file until the MMCP can perform a physcial write that conforms to the hardware makeup of the printer (write, then space). A description of this mechanism, and the ramifications for the user are described under Temporary Documentation.

2.2.3.
The output of the "CU" command has been enhanced for the Mark 9.0 release. Refer to Temporary Documentation for an example of the new output.

2.2.4.
The "LC" MCP input message no longer means "Load Cassette", but now means "Log Comment". This command allows an operator to enter necessary comments into the system log. A description of the syntax of this command is given in Temporary Documentation. Documentation for executing the SYSTEM/LOAD.CASS program is available in the form of a printer backup file named DOCUMENT/UTILITY.

2.2.5.
The syntax of the "WS" MCP input message has been changed to query the state of only the ACTIVE schedule, or only the WAITING schedule, or both (as in previous releases). Refer to Temporary Documentation for the syntax of this command.

2.2.6.
The use of the "THEN" command for job execution has been expanded in the Mark 9.0 MCP. The "THEN" command may now be used after an "ADD", "CO", "COMPARE", "COPY", "DIR", "EX", "LD", "LG", "LN", "PB", "PM", or "RC" command.

2.2.7.
The MCP now checks the available table before attempting to return a disk area to the available table, thereby avoiding the creation of an overlapping area (01-560-000010).

2.2.8.
When a write occurs to a random disk file with an invalid key, the error message given by the MCP is now consistent in all cases.

2.2.9.
The MCP now gives the error message "LPx IS NOT A TRAIN PRINTER OR THE TRANSLATOR IS ALREADY LOADED" if the command "LT LPx" is entered and "LPx" is a B-9249 (ODEC) printer (01-705-W30479).

2.2.10.
The MCP is now more consistent in the error message printed when a read operation is attempted on an output-only file ("READ REQUESTED ON OUTPUT FILE").

2.2.11.
The MCP now passes charge numbers automatically in "ZIP" commands. A charge number appearing in the "ZIP" statement, however, overrides the charge number of the "ZIPping" program which is passed by default. The charge number supplied by a usercode in the ZIP statement also overrides the default charge number.

2.2.12.
End-of-file is no longer reported to a user program when a backup file is being created. The MCP automatically closes the file when it is full, and automatically opens a new backup file. The identifier assigned to the second file reverts to the standard naming convention for backup files. The family name becomes "BACKUP.PRT", or a usercode if file security is in effect. The file-id is a backup print number, maintained by the MCP. All file attributes associated with the first backup file are retained in the second backup file. The operator is notified of the creation of the "extension" file.

2.2.13.
The MCP can now accept more than two operator commands in a single input string. Previously the MCP's scanner would only accept two commands, and improperly interpret any successive commands in the same string (ex. PM; PS 999; EG).

2.2.14.
The MCP no longer flushes a pseudo reader when the job that created it goes to "EOJ".

2.2.15.
The MCP now permits portions of low memory (OCxxxx) to be XM'ed without causing a halt in the subsequert "CLEAR/START" (12-097-DM1494).

2.2.16.
An invalid field in a header label on disk nc lorger causes an MCP halt during open.

2.2.17.
The "INVALID MNEMONIC" MCP outout message now prints the token that was in error (13-338-ALS033).

2.2.18.
The MCP now permits the open of a disk file with zero areas specified. The MCP uses the value for areas in the disk file header for the file. This permits a user to open a file without knowing the number of areas in the file (13-338-ALS0B1).

2.2.19.
The MCP now properly maintains the run structure status when a job is stopped with the "ST" command. A subsecuent "WY" command specifies that the job is stopped, and the status of the job before it was stopped.

2.2.20.
Invalid (non-existent) queue names that are found in a pseudo reader are now ignored, causing the parent queue and the "LS" boolean to be set to zero for the job being executed.

2.2.21.
Closing a tape with "NO.REWIND" when not at the end of a file no longer causes the tape descriptor chain tc be corrupted (13-388-G80360).

2.2.22.
The MCP now checks the integrity of the pack label whenever the pack label is read.

2.2.23.
The MCP now increments the data overlay count for SOL/UPL paged arrays every time a data overlay occurs.

2.2.24.
The Mark 9.0 MCP now gives the full file name for all "PD's". This includes the name of the pack on which the file is located. If no pack name is printed, the file is located or the system pack. When a usercode/password precedes the "PD" command, the MCP looks first on the default pack of the user, and if the file is not found there the MCP looks on the system disk.

2.2.25.
If the "VOL1" field on a disk pack label contains the word "DUMP" (placed there by DISK/DUMP) the Mark 9.0 MCP displays the message "DPx IS AVAILABLE FOR DISK/DUMP OR COLDSTART/DISK ONLY" when the pack is made ready (13-388-JN0065).

2.2.26.
The Mark 9.0 MCP only permits the "AUTOPRINT" queue to be opened "INPUT". Attempting to open the "AUTOPRINT" queue "OUTPUT" or "INPUT/OUTPUT" is disallowed.

2.2.27.
The MCP no longer creates a file of type "data" when a printer backup file is OU'ed to a disk which initially has too little available space to hold the file (22-060-78C024).

2.2.28.
The MMCP now halts properly when a reader/sorter fails to respond to a "STOP FLOW", and an MMCP tank overflow has occurred.

2.2.29.
The MCP no longer allows an "OU" as a response to a "NO HARDWARE" condition for "AUTOPRINT".

2.2.30.
The MCP now attaches the default pack identification in the "CH" command when using the "USER" MCP input message.

2.2.31.
The MCP now handles the invalid syntax when the name of the file is left out of the "NAME" attribute in the "FILE" equation command.

2.2.32.
When there is not enough memory to terminate a job, the MCP now displays the message "NO MEMORY FOR TERMINATION" as the status of the job, instead of halting.

2.2.33.
The Mark 9.0 MMCP has been corrected to more carefully check for an invalid station in a request from an MCS.

2.2.34.
A usercode is no longer retained after an "END" card, in a stream of cards input to SYSTEM/LDCONTRL.

2.2.35.
The MCP no longer halts when an attempt is made to open an optional file with "EXTEND", when the file is not present (43-444-100065).

2.2.36.
In the Mark 9.0 MCP, a program is terminated if it attempts to open a blocked punch file.

2.2.37.
In the Mark 9.0 MCP, the "OF" command is not permitted to be used on a file that is to be opened output only. Use the DUMMY.FILE attribute instead.

2.2.38.
In the Mark 9.0 MCP, if a program requests the line printer, and the printer is in use by the MCP, due to the "KB LP" option being set, the system resets the option and assigns the printer to the requesting job (52-000-030179).

2.2.39.
The "QMX" (queue max messages) file attribute now has a lower limit of one message.

2.2.40.
"RELATIVE" have been added to the types of files that may be specified in the FILE.TYPE file attribute.

2.2.41.
The integer specified in the "Q.FAMILY.SIZE" file attribute must not exceed 999 (previously 1000).

2.2.42.
The "SO" message may now be used to specify additional system drives which are on different channels from the original system disk. However, no two of these system disks may be on the same unit number.

2.2.43.
When a job running under a non-privileged usercode opens a printer/punch backup disk file that has the "USER.BACKUP.NAME" attribute set, and both the multi-file-id (which is not the usercode) and the file-id of the file name are specified, the multi-file-id is moved to the file-id and the multi-file-id is changed to the usercode of the job, thus destroying the user-specified file-id.

When a pack is specified in a orinter/punch backup disk file that has the "USER.BACKUP.NAME" attribute set, the specified pack-id overrides the default pack of the usercode being used to create the file, so that the file is not written to the user's default pack.

2.2.44.
The Mark 9.0 MCP now permits as many as 32 spindles of disk ("DPx") to be on line at the same time. The mnemonics for these disks are "DPA" through "DPZ", then "DP1" through "DP6".

2.2.45.
The MCP now only prints the name of a newly opened printer backup file if the "LIB" option is set, or if the backup file is an extension file.

2.3. Temporary Documentation.

2.3.1. New File Attributes.

FOOTING
The attribute "FOOTING" (which may be abbreviated "FOOT") specifies the number of lines from the beginning of the page body, within "PAGE.SIZE", to the point where the MCP begins to report end-of-page to the user. The "ON EOF" branch is taken in the program, but the record is still written, unless the record would be written in the lower margin, in which case a page advance is made to the beginning of the next logical page. "FOOTING" may be set to any value from 1 to 255 inclusive.

LOWER.MARGIN
The attribute "LOWER.MARGIN" (abbreviated "L.M") specifies the number of lines to leave blank between the page body of the current page and the upper margin of the next page. The "LOWER.MARGIN" may be set to any value from 0 to 255 inclusive.

MAXRECSIZE
The "MAXRECSIZE" file attribute specifies the maximum record size for a variable length record file. This attribute is equivalent to "RECORD.SIZE".

MINRECSIZE
The "MINRECSIZE" file attribute specifies the minimum record size for a variaole length record file.

PAGE.SIZE
The "PAGE.SIZE" file attribute (abbreviated "P.S") specifies the number of lines between the upper margin and the lower margin. "PAGE.SIZE" may be set to any value from 1 to 255 inclusive.

UPPER.MARGIN
The "UPPER.MARGIN" file attribute (abbreviated "U.M") specifies the number of lines to skip at the beginning of the logical page before the page body. The "UPPER.MARGIN" may be any value from 0 to 255 inclusive.

Note

> The logical page size should be calculated such that the upper margin plus the page body plus the lower margin equal the size of the physical page. If this is not done, eventually a line will be printed on the crease between the physical pages.

2.3.2. Logical/Physical Printer Output.
The Mark 9.0 MMCP now buffers logical printer output in order to take advantage of the hardware architecture of the printer, and to create only one physical output for each logical output.

The following diagram displays the relationship between a logical write operation to the printer, and its effect on the physical output to the printer:

Current Logical Request \ Pending Operation	Null	Write, No Space	Write Before Single Space
Write, No Space	No-op Pending = Write, No Space	Write, No Space Pending = Write, No Space	Write, Space 1 Pending = Write, No Space
Write/B Space 1	No-op Pending = Write, Space 1	Write, No Space Pending = Write, Space 1	Write, Space 1 Pending = Write, Space 1
Write/B Space 2	Write, Space 2 Pending = Null	Write, No Space Write, Space 2 Pending = Null	Write, Space 1 Write, Space 2 Pending = Null
Write/A Space 1	Space 1 Pending = Write, No Space	Write, Space 1 Pending = Write, No Space	Write, Space 2 Pending = Write, No Space
Write/A Space 2	Space 2 Pending = Write, No Space	Write, Space 2 Pending = Write, No Space	Write, Space 2 Space 1 Pending = Write, No Space
Write/B Channel	Write, Channel Pending = Null	Write, No Space Write, Channel Pending = Null	Write, Space 1 Write, Channel Pending = Null
Write/A Channel	Space, Channel Pending = Write, No Space	Write, Channel Pending = Write, No Space	Write, Space 1 Space, Channel Pending = Write, No Space
Space N	Space x Space (N-x) Pending = Null	Write, Space x Space (N-x) Pending = Null	Write, Space 2 Space (N-1) Pending = Null

Figure 1 - Logical/Physical Output Relationship

In the preceding diagram, the operations within the table correspond to the actual physical output operations that are performed, depending upon the current logical request supplied by the user, and any operations that are still pending from the previous request. Write/B and Write/A may be read "WRITE BEFORE ADVANCING", and "WRITE AFTER ADVANCING". "Space x" in the diagram is a space of 1 (if N = 1) or 2 lines (if N is greater than 1) on the printer. It should be noted that a second physical write to the printer is only sent if N-x is greater than zero.

As can be seen in the diagram, the physical position of the printer is usually one line of output behind the object program. Only when the Pending operation in the diagram is set to a null value does the physical position of the printer correspond to the position the object program believes it is in.

This phenomenon should present no problems for any existing printer programs unless the computer operator is required to look at the last line of output and in order to do so, must manually advance the paper. Under these circumstances, the operator may not see the actual last line of output written by the program, and may upset the format of the listing being printed.

Of particular concern are those programs which use special forms and which contain a "Forms Alignment" loop. These loops typically write one page of dummy output on a special form, advance the printer to the next form, display a message asking the operator if the form is aligned correctly and perform an "ACCEPT" operation, waiting for the reply. If the program does not advance to the next form itself and if, in order to see the dummy output that was written on the form, the operator must manually advance the paper and if the last logical operation performed by the program left a Pending operation that was not a null value, changes are probably necessary in the program. If the operator does not manually advance the paper, no changes are necessary.

It can be seen in the diagram that some logical requests result in two physical operations being initiated. Under these conditions it may be beneficial to supply each printer file with at least two buffers, if the execution time of the program is the only concern. Total system throughput is not impacted significantly regardless of the number of printer buffers and the types of the operations being performed. If the MCP must wait for the completion of any printer physical output operation, the time that is spent waiting is masked by the processing of other programs.

Along these same lines, it should be remembered that any time a Write operation is left pending and control is returned to the user, the MCP must have an available buffer to store the data that is to be written. If no buffer is available, control may not be returned to the requesting user until a buffer becomes available. Again, this time is overlapped with the processing of other programs and system throughput should not be significantly impacted.

2.3.3. CU Command.
The output of the "CU" command has been enhanced to more completely indicate the allocation of system memory, or program memory. The following is an example of the output of the "CU" command when no job number is given:

```
CU
 CORE USAGE  : 22:07:01.5
 SAVE= 125686 BYTES
 LARGEST OVERLAYABLE= 716389 BYTES
 PAY =1263 SAVE= 29771 BYTES , OVERLAYABLE= 24950 BYTES
 LED =1264 SAVE= 31334 BYTES , OVERLAYABLE= 84256 BYTES
 PAR =1262 SAVE= 12567 BYTES , OVERLAYABLE= 6435 BYTES
 INV =1265 SAVE= 626 BYTES , OVERLAYABLE= 0 BYTES (ROLLED OUT)
 REC =1266 SAVE= 1133 BYTES , OVERLAYABLE= 0 BYTES (ROLLED OUT)
```

The following example shows the output of the "CU" command when a job number precedes the command.

```
1265CU
 INV =1265 SAVE= 626 BYTES , OVERLAYABLE= 0 BYTES (ROLLED OUT)
```

2.3.4. LC Command.
The new "LC" command allows the operator to enter comments into the system log. The syntax of the "LC" command is as follows:

```
>-- LC -- <string> ---|
```

<String> is any string of characters, and is the comment desired. <String> does not need to have quotes as delimiters. The maximum length of <string> is 165 characters.

2.3.5. WS Command.
The syntax of the "WS" command has been enhanced to permit the operator to request that only the ACTIVE schedule be printed, or that only the WAITING schedule be printed. The syntax of the "WS" command is as follows:

```
>-- WS ----- <job number> -----|
          |                    |
          |-- ACTIVE ---------|
          |                    |
          |-- WAITING --------|
          |                    |
          |-- =/= ------------|
```

2.3.6. Complex Wait.
A change has been made in the MCP's handling of a complex wait between Mark 9.0 and Mark 10.0. In all releases through Mark 9.0, the MCP considered a wait of zero tenths of a second as a wait of one tenth of a second. In the Mark 10.0 release a wait of zero tenths as one of the conditions causes the program to fall out of the complex wait as soon the wait statement is

encountered. Depending on the program logic, the difference between Mark 9.0 and 10.0 may cause the complex wait to be satisfied for different reasons. This change should be taken into consideration for all programming that involves the use of the complex wait function.

3. Interpreters and Firmware

3.1. Introduction.

The Mark 9.0 firmware contains major changes to the SDL and RPG interpreters, designed to increase performance. The Mark 9.0 firmware also contains three new interpreters, utilized by the new FORTRAN77, COBOL74, and IBASIC languages.

The SDL and RPG interpreters have had their number of segments reduced from fourteen to six for SDL and from four to one for RPG.

The MCP.LEVEL field checked by the MCP at BOJ for all programs has been raised from 3 to 4 for all interpreters, meaning that Mark 9.0 interpreters cannot be executed with Mark 8.0 systems.

3.3. Temporary Documentation.

3.3.1. Interpreter Verification Attributes.

SDL Interpreter

```
ARCHITECTURE.NAME: "SDL"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE:
  2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15
Segments: 6
```

COBOL Interpreter

```
ARCHITECTURE.NAME: "COBOL"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: 0
Segments: 4
```

COBOL74 Interpreter

```
ARCHITECTURE.NAME: "COBOL74"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

RPG Interpreter

```
ARCHITECTURE.NAME: "RPG"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE:
  0, 1, 2, 3, 4, 5
Segments: 1
```

BASIC Interpreter

```
ARCHITECTURE.NAME: "BASIC"
COMPILER.LEVEL: 3
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

IBASIC Interpreter

```
ARCHITECTURE.NAME: "IBASIC"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

FORTRAN Interpreter

```
ARCHITECTURE.NAME: "FORTRAN"
COMPILER.LEVEL: 1
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: 0, 1
Segments: 1
```

FORTRAN77 Interpreter

```
ARCHITECTURE.NAME: "FORTRAN77"
COMPILER.LEVEL: 3
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

B500 Interpreter

```
ARCHITECTURE.NAME: "B500"
COMPILER.LEVEL: 2
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

1400 Interpreter

```
ARCHITECTURE.NAME: "1400"
COMPILER.LEVEL: 2
MCP.LEVEL: 4
GISMO.LEVEL: 3
ARCHITECTURE.ATTRIBUTES bits that are TRUE: None
Segments: 1
```

GISMO

```
GISMO.LEVEL: 3
```

4. RPG

4.1. Introduction.

The Mark 9.0 release of the RPG compiler includes corrections to outstanding problems, the new DMS "*ALL" entry, implementation of the U-indicators for DMS files, and compatibility changes to the "DELET" operation code.

A printer backup file labeled "DOCUMENT/RPG.DMS" contains a number of RPG DMS programming examples as well as documentation of the "DMSDBUG" debugging aid.

4.2. Enhancements.

4.2.1.
An input file with stacker select was opened input/output; allowing for possible erroneous punch failures (42-146-000006).
An input file with stacker select is now opened input only.

4.2.2.
The "XFOOT" operation code using "HALF ADJUST" did not half adjust properly if the result was a minus quantity (54-028-DM1142).
"XFOOT" with "HALF ADJUST" on a negative quantity now generates the correct result.

4.2.3.
The RPG Compiler was not syntaxing DMS files which began with a blank character. Instead the MCP displayed a console message "ATTEMPTED TO OPEN DMS.LIBR WITH 1ST CHAR OF PACK ID OR MFID OR FID BLANK" (13-338-8.0-09).
A fatal syntax error is now given on an invalid RPG name for a DMS file.

4.2.4.
An RPG program would halt with "INVALID SUBSCRIPT DS OR DP" when a Calculation Specification contained the following: 1) "MOVE" in the "OPERATION FIELD" 2) A subscripted array in "FACTOR 2", and 3) the subscript of the array is also specified in the "RESULT FIELD" and the value of the subscripted array which is being moved is larger that the "ENTRIES PER ARRAY" specified on the Extension Specifications (13-338-6.1-05).
This problem has been corrected.

4.2.5.
A "MULT" operation code where the larger number was specified in "FACTOR 1" executed faster than if the larger number was specified in "FACTOR 2".
This problem has been corrected by passing the largest number to the interpreter as "FACTOR 1".

4.2.6.
A syntax error was not generated when column 52 of the Input Specification (the "DECIMAL POSITIONS FIELD") contained the letter "O" (31-175-CLI001).
A syntax error is now given when column 52 of the Input Specifications contains anything except the digits 0-9 or blank.

4.2.7.
A syntax error is now generated when a "READ" operation code is specified for a DMS file.

4.2.8.
On the Output-Format Specifications the combination of an unsubscripted table in column 32 and a DASDL-defined field name in column 17 is now allowed.

4.2.9.
When columns 60-65 of the File Description Specifications were non-blank, for a file where the core index was meaningless, no warning was given.
A warning is now given.

4.2.10.
For the Mark 9.0 release the use of the word "*ALL" in columns 53-56 of the Input Specifications and columns 40-43 of the Output-Format Specifications relieves the user from having to code the fields for these Specifications when DASDL defined names are used. Refer to Temporary Documentation 4.3.1 for a complete description of the new "*ALL" entry.

4.2.11.
If a DMS file name declared on the File Description Specifications was not a DASDL-defined data set name, the DMS file name was not syntaxed by the RPG compiler. The result was that the operator had to intervene with a "DS" or "DP".
A fatal syntax error is now given on an invalid RPG name for a DMS file.

4.2.12.
Reserved words "UDATE", "UDAY", "UMONTH", "UYEAR", and "UDATE" were not syntaxed if used as field names on the Input Specifications, in the "RESULT FIELD" on Calculation Specifications, or as field names on Output Format Specifications using blank after.
This problem has been corrected.

4.2.13.
The Record Identification Indicators were being turned off prior to checking the status of the Halt Indicators (H0-H9) (13-338-8.0-13).
Now the Halt Indicators are checked before the Record Identification Indicators are turned off.

4.2.14.
The RPG compiler did not give a syntax error when a literal subscript for a vector in column 53 of the Input Specifications or column 32 of the Output-Format Specifications was outside the range of the DASDL-defined vector in column 44 of the Input Specifications or column 17 of the Output-Format Specifications. The compiler now gives a syntax error for the above condition.

4.2.15.
The external indicators (U1-U8) are now allowed for DMS files. Refer to Temporary Documentation for a description of external indicators.

4.2.16.
The alphabetic entries "AD" or "R" are now allowable entries in the "SEQUENCE FIELD" (columns 15-16 of the Input Specification).

4.2.17.
A letter "J" inserted in column 21 of the Control Card Specification was not producing the desired print format (the leading zeros were being suppressed) (81-7000-DM1081). For example what should have been "0,00" was printed as ",00". The letter "J" now produces the correct format.

4.2.18.
The "DELET" Operation Code, used for DMS files, must now contain the letter "S" in column 53 of the Calculation Specification. Previously a blank was required. The purpose for this change is to maintain a standard between other Burroughs RPG compilers for the sequential deletion operation.

4.3. Temporary Documentation.

4.3.1. *ALL Entry.
Prior to the "*ALL" entry the user had to explicitly define the Input and Output-Format Specifications for data management files. The "*ALL" entry does away with this requirement.

The syntax for the "*ALL" entry and some examples are given below:

Input Statement

The "*ALL" entry appears in columns 53-56 of the Input Specification. This still allows the user to do record identification of DMS files. The following template is used in the "*ALL" RPG-DMS examples to represent the column numbers within the Input and Output-Format Specifications.

```
----*----1----*----2----*----3----*----4----*----5----*----6----*----
1.   IA                                                  *ALL
```

This is the same as the user coding:

```
     IA
     I                                          X         X
     I                                          Y         Y
```

X and Y are the only DMS variables in data set A.

```
2.   IA                                                  *ALL
     I                                          X         Q
```

This is the same as the user coding:

```
     IA
     I                                          X         X
     I                                          Y         Y
     I                                          X         Q
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

```
3.   IA                                                  *ALL
     I                                          X         X              L1
```

This example shows how the user can effect control levels, matching fields, field record relation, or field indicators. In the above example a control level indicator "L1" is assigned to the field "X". It is as though the user had coded the following:

```
     IA
     I                                          X         X              L1
     I                                          Y         Y

4.   IA             01    1 CA                  X
     I                                          Y         A
     I              02    1 CB                  X         *ALL
```

This is the same as the user coding the followirg:

```
     IA             01    1 CA                  X
     I                                          Y         A
     I              02    1 CB                  X
     I                                          X         X
     I                                          Y         Y
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

5. In this example, assume that "X" and "Y" were declared in the DASDL source as "NUMBER (4,0)" and "R" and "S" were were declared in the DASDL source as "ALPHA (3)".

```
IA              01
I                                       R         X
I                                       S         Y
I               02                                *ALL
```

This is the same as the user coding:

```
IA              01
I                                       R         X
I                                       S         Y
I               02
I                                       X         X
I                                       Y         Y
I                                       R         R
I                                       S         S
```

The field lines coded for the 01 indicator defire local variables "X" and "Y" as "ALPHA (3)". The field lines produced by the "*ALL" entry define local variables "X" and "Y" as "NUMBER (4,0)". This produces the following syntax error; "REDEFINES FIELD NAME".

6.
```
IA              01    1 CA                Y         *ALL
I           OR  02
```

This is to illustrate that the "*ALL" entry applies to "OR" lines which follow the first record line. The same applies to "AND" lines. This is the same as the user coding:

```
IA              01    1 CA                Y
I           OR  02
I                                       X         X
I                                       Y         Y
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Source Listing

The following example of a source listing contains the Input Specifications which the RPG compiler generates. The RPG compiler generated lines have a pound sign (#) in column 7 of the Input Specifications.

For example if the user coded:

```
IA                                                      *ALL
```

The resulting source listing is:

```
IA                                                      *ALL
I#                                                      X
I#                                                      X
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

If the user coded:

```
IA                                                      *ALL
I                                              X        X              L1
```

The resulting source listing is:

```
IA                                                      *ALL
I                                              X        X              L1
I#                                                      Y
```

Output-Format Specifications

For Output-Format Specifications, the same concepts apply. The "*ALL" entry is contained in columns 40-43 of the Output-Format Specifications line. Output-Format Specifications are generated only for those DASDL-defined items which have been previously defined.

```
1.   OA       S        01              *ALL
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

This is the same as the user coding:

```
     OA       S        01
     O          X               X
     O          Y               Y

2.   OA       E        01              *ALL
     O          X                         +2
```

This is the same as the user coding:

```
     OA       E        01
     O          X                         +2
     O          Y               Y
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Interaction
The following example shows the interaction between the Input and Output-Format Specifications.

```
     IA             01   1 CA                  X         *ALL
     I              02
     I                                         X         X
```

This is the same as the user coding:

```
     IA             01   1 CA                  X
     I                                         X         X
     I                                         Y         Y
     I              02
     I                                         X         X
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

At input time, if "X" has the character "A" in position 1, the record identifying indicator 01 is set on and "X" and "Y" are moved in. Otherwise, the record identifying indicator 02 is set on and "X" is moved in.

Assume the user codes the Output-Format Specifications as follows:

```
   OA       E                          *ALL
```

Both "X" and "Y" are written at exception time. If the input record was an 02 type, the value of "Y" is either the value at input time or the result of a calculation operation. This may be reasonable, or it may not.

Code to Avoid
Several instances involving the "*ALL" entry should be avoided. These cases are scheduled to be fixed in the next release and are described below:

In the described examples both "X" and "Y" are DASDL-defined fields.

1.
```
   OA                                      *ALL
                  Y                X       B
----*----1----*----2----*----3----*----4----*----5----*----6----*----
```

The following code is generated:

```
     (line 1)    Y                X       B
     (line 2)    X                X
```

The RPG compiler generates code to move blank characters to the DASDL-defined field "X" after the value of "X" is moved to the DASDL-defined field "Y". The RPG Compiler then generates code to move the value of "X" to the DASDL-defined field "X". Since the

value of "X" is blank, the resulting output of "X" is blank. Note: that this problem results when both an RPG and DASDL-defined field are called the same name and the blank after is used.

2. IA

```
                                                    *ALL
                                              Y     A,X
```

The following source code would be generated:

```
----*----1----*----2----*----3----*----4----*----5----*----6----*----
     (line 1)                                 Y     A,X
     (line 2)                                 X     X
```

The value for "X" is not known. Again the problem results because both DASDL-defined and RPG-defined fields have the same name.

4.3.2 Corrections to the RPG Manual.

Please make the following changes to the B1000 Systems Report Program Generator (RPG) Reference Manual, form # 1057189 dated August 1979.

1. Reference page 8-3.

Under columns "15-16 SEQUENCE" delete the paragraph reading "When coding this field, the programmer must not use the alphabetic entries ND or Rb (b equals blank), because the compiler may mistake these for the ND or R of an AND or OR line".

2. Reference page 4-29.

The "LABELS" field, column 53 of the File Description Specification is renamed to be the "FILE OPEN" field.

Valid entries for this field are as follows:

Entry	Definition
Blank	Standard System Labels are expected/provided for this file. If the file is a disk file, access to this file by other programs is not restricted while the file is opened by this program.
B	For printer files; indicates that the file is to be forced to Printer Backup. For card files; indicates that the file is to be forced to Punch Backup.
F	Applies to printing files and indicates that special forms are required.
L	Applies to input or update disk files and indicates that this file is not available to other programs while the disk file is opened by this program.
R	Applies to input or update disk files and indicates that this file is available to other programs for input only while this disk file is opened by this program.
U	This file is expected/provided as an unlabeled file.

3. Reference page 5-8.

Figure 5-2 line 09 columns 46 through 57 should be shown as optional. Add the letter "O" to these columns.

4. Reference page 13-20.

Under columns 28-32 "OPERATION FIELD" delete item d. "FORCE" is a valid operation.

5. Reference page 13-22.

Under columns 54-59 "RESULTING INDICATORS" refer to item 1 of paragraph b. The statement reads "The data set specified in FACTOR 1 for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELETE operation codes". The statement should read "The data set specified in FACTOR 2 for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELET operation codes".

6. Reference page 13-9

Change 67-74 to 67-76. Add the following:

71-72 FILE CONDITION
This field may be used on input, update, output and restart data management files and indicates whether or not the file is conditioned by an external indicator. If a DMS file is conditioned by an external indicator the file is used only when that indicator is on. When the indicator is off, the file is treated on input as if it is at the end of file. In Calculation Specifications, if any DMS operations are done to the conditioned file, the operations should be conditioned or the same U-indicator specified for the DMS file. If the DMS operations are not conditioned by the same U-indicator and one of the indicators is off, the operations are ignored and the 01 resulting indicator, if used, is not affected. Likewise any output done to the DMS file should be conditioned by the same U-indicator used on the DMS file. If the same U-incicator is not used, the output record is built (and any blank after operations done), but the record is not written to the file.

It is important to note that DMS files are accessed by opening the database and that even though all DMS files in the program may be conditioned by U-indicators which are off, the database is opened and any exceptions that might normally occur at open time can still happen.

U-indicators being off may still require a recompile of the DASDL program to prevent a VERSIONERROR (of dictionary) at Data Base open time.

Since only disjoint data sets can be specified on File Description Specifications, any embedded data sets associated with a disjoint data set conditioned by a U-indicator are treated as having been conditioned by the same U-indicator. All Calculation Specifications and Output-Format Specifications affecting the embedded data set should be conditioned by the same U-indicator as the parents data set and are ignored if the U-indicator is off.

ENTRY	DEFINITION
U1-U8	The specified external indicator is used to condition the DMS file.
Blank	The DMS file is not conditioned by an external indicator.

Refer to FILE CONDITION (columns 71-71) on non-DMS files (page 4-34) for more information on external indicators.

73-74 These columns must be left blank for DMS files.

7. Reference page 13-53.

Make the same changes to 13-53 as were made to 13-9 above.

8. Reference page 13-54.

Refer to the next to the last paragraph; the first sentence should read "The implicit TREND operation generates an end transaction with audit and no syncpoint, except when the LR indicator is ON in which case the implicit TREND operation is generated with no audit, syncpoint."

9. Reference page 13-65.

Delete paragraph c.

10. Reference page 9-19.

Under "MOVE OPERATIONS" insert the following paragraph:

"Decimal positions are ignored when moving data from one numeric field to another numeric field. For example, if X is a field with three decimal positions and Y is a field with 2 decimal positions, a move of 10.000 from X to Y will yield 100.00 in Y."

5. COBOL

5.1. Introduction.

The Mark 9.0 release of the ANSI 68 COBOL compiler contains no new language constructs. Corrections have been made for problems found in prior releases.

5.2. Enhancements.

5.2.1.

The following problems reported in the Mark 8.0 System Notes have been corrected (the number in parentheses refers to the number of the problem reported in the Mark 8.0 System Notes):

a. Attempts to create library files when "DECIMAL-POINT IS COMMA" is specified now work correctly (01-691-SML034).

b. "$ CODE" now works correctly (04-724-NS0006).

c. A syntax error will now be generated when the "SELECT OPTIONAL..." clause is used with a nonsequential file (11-000-CHAF-A).

d. The statement "DIVIDE A(S1,S2) INTO B(S1,S2) GIVING C(S1,S2) ROUNDED ON SIZE ERROR MOVE ZEROS TO C(S1,S2)." no longer causes an "INVALID SUBSTRING" (12-089-HM0008).

e. A "DUMP" on an edited field now generates correct code (13-338-DM0731).

f. The compiler will now syntax a special character in a Procedure name (13-338-DM1303).

g. "SUBTRACT ID FROM ZERO" no longer causes the compiler to fail with "FIXUP COMPILER ERROR" (13-338-DM1431).

h. The problem reported with a "GO TO DEPENDING ON" having to have a comma after the last label name to generate code, was caused by a "NOTE" statement and was not a compiler error (16-328-000014).

i. A level 66 item followed by a level 03 item is now being syntaxed (23-000-IPL285).

j. Inclusion of a second "FILE SECTION" is now syntaxed (23-000-TON003).

k. A data-name that is greater than 30 characters long no longer causes the compiler to abort with "INVALID SUBSTRING" (31-041-CMI003).

l. A DIVIDE with a non-signed result field now gives a correct remainder (32-073-000125).

m. A compiler error message is now generated if the limit of 255 terminal performs or unique exit points is executed (43-757-CN8001).

n. A program using ISAM Files abnormally terminating in some cases with a "NO PROVISION FOR EOF" on the Tag File after records have been added has been corrected (61-244-050279).

o. A syntax error is generated if an asterisk is placed in column 8 instead of column 7 (61-244-050779).

p. "DIVIDE A INTO B GIVING B REMAINDER C" now puts the correct remainder in C (11-088-990001).

5.3. Temporary Documentation.

5.3.1. 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 will also be printed and will be called "TCARD". This number can then be used to find the source statement that is in error. In a DUMP/ANALYZER listing of the compiler, "TCARD" will apoear as the first item with a bit length of 24 in the "NAME STACK". Passes other than the initial pass will 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 Field Trouble Report (form #1912003) by the local Burroughs support personnel.

EXAMPLE:

The following is a sample of output using the "$ LISTP" option.

```
00001  000010$ LISTP TIME
00002  000100  IDENTIFICATION DIVISION.
00003  000800  ENVIRONMENT DIVISION.
00004  000900  CONFIGURATION SECTION.
00005  001200  INPUT-OUTPUT SECTION.
00006  001500  DATA DIVISION.
00007  002100  WORKING-STORAGE SECTION.
00008  002110  77  J-NAME PC X.
00009  002200  PROCEDURE DIVISION.
00010  002210  A-PARAGRAPH.
00011  002220      MOVE A-LINE TO B-LINE.
00012  002250      IF J-NAME = "1"
00013  002260       IF J-NAME = "2"
00014  002270        IF J-NAME = "3"
00015  002280         IF J-NAME = "4"
00016  002290          IF J-NAME = "5"
00017  002300           IF J-NAME = "6"
00018  002310            IF J-NAME = "7"
00019  002320             IF J-NAME = "8"
00020  002330              IF J-NAME = "9"
00021  002340               IF J-NAME = "10"
00022  002350                IF J-NAME = "11"
00023  002360                 IF J-NAME = "12"
00024  002370                  IF J-NAME = "13"
00025  002380                   IF J-NAME = "14"
00026  002390                    IF J-NAME = "15"
00027  002400                     IF J-NAME = "16"
00028  999998      STOP RUN.
00029  999999 END-OF-JOB.
***** ERROR 061 (FROM DNFILE) ON TCARD 00011, COLUMN 16
***** ERROR 061 (FROM DNFILE) ON TCARD 00011, COLUMN 26
***** ERROR 218 ON TCARD 00011, COLUMN 23
```

In trying to compile the previous example, the error condition "EVALUATION/PROGRAM POINTER STACK OVERFLOW, OS or DP" occurred. In taking a program dump, the first data item in the Name Stack with a bit length of 24 shows the value of TCARD to be 27. In using this number to refer to the LISTP listing it is found that the STACK OVERFLOW occurred while attempting to process a nested IF statement which contained more than the allowed number of nested IF's. Correcting this condition as well as the other errors listed, allows a successful recompilation.

5.3.2. 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:

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

2. 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 10 phases, as follows:

1. "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.

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

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

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

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

6. "DATSYN" - DATA DIVISION syntax checking.

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

8. "PROSYN" - PROCEDURE DIVISION syntax checking.

9. "CODEGEN" - Code generation.

10. "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 made 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:

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

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

3. Data names. Each data-name declared recuires 38 bits.

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

5. 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.

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

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

8. 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:

1. 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.

2. 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.

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

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

5. 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.

6. 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.

7. 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.

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

9. 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.

10. ACFILE "COBOLW/ACFILE" DISK
Used by PARSE, DICT, ONQUAL, 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-gererated 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.

11. ALLONOUT "COBOLW/ALLONOUT" 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.

12. 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.

5.3.3. 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. 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.

5.3.4. 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 job termination.

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

1. 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 program immediately.

2. WRITE the data record. Request error reporting. In the event of an error, the program takes the INVALID KEY branch or invokes the appropriate 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.

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:

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

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

3. INVALID KEY and USE Routine - Report or 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.

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.

5.3.5. Efficient Use of ISAM Constructs.

COBOL checks for duplicate key values at the time a WRITE statement is executed. 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.

5.3.6. ISAM Subroutine Segmentation.

The COBOL compiler automatically segments the ISAM-generated file-handling subroutines. 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).

5.3.7. 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, civide by 7, giving 505, and so forth.

5.3.8. NOISE-WORDS and their implications.
The following is a list of words which are ignored by the compiler known as NOISE-WORDS:

ADVANCING	FOR	SENTENCE
ARE	KEY	STANDARD
AT	KEYS	THAN
AUTO	MANUAL	WITH
DEMAND	ON	
IS	RECORD	

The misspelling of a word such that the resulting word is a NOISE-WORD, results in that new word (NOISE-WORD) being omitted by the scanner for the compiler. For example; the misspelling of the word "OF" as "ON" in the following statement results in both data-names receiving the contents of "DOLLAR-AMOUNT":

```
MOVE DOLLAR-AMOUNT TO AMOUNT ON AMT-GROUP.
```

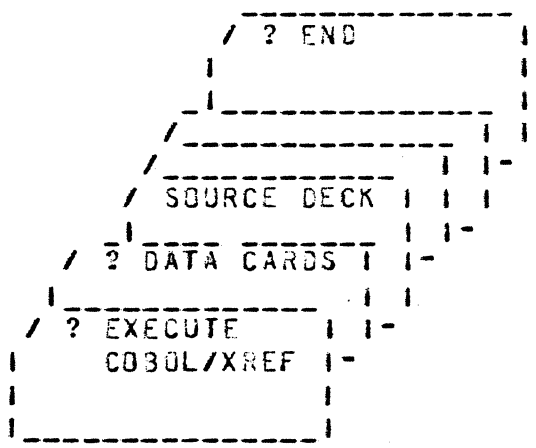
The following example shows that the "IF" condition is true only when KEY1 is less than 8 instead of being true wher KEY1 is less than KEY8:

```
IF KEY1 < KEY 8, GO TO WRAP-UP.
```

The use of contractions of key words as data-names is NOT a good practice, however, careful desk checking and screering of source code for misspelling of NOISE-WORDS can save hours of DUMP-READING.

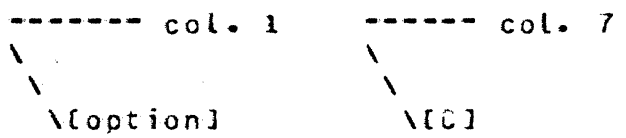
5.3.9. COBOL CROSS REFERENCE UTILITY PROGRAM.
The COBOL Cross Reference Utility Program (COBOL/XREF), accepts as input a syntactically correct COBOL source program, and depending on the option selected, outputs a cross reference listing, or a program source listing only, or both a cross reference and a program source listing.

The source program may reside on disk, magnetic tape, or punched cards. The following is an example of a COBOL/XREF execution card deck.



COBOL/XREF Execution Deck

The option entry specifies the location of the source medium and the output desired. Below is the format of the COBOL/XREF option card.



The option entry must begin in column 1.

The C denotes that the COBOL COPY verb is usec withir the source program. C, when used, must be in column 7. Requested library files must reside in the disk directory. The library sequence numbers within a source program are indicated by an L tc the left of the sequence number.

The options and their descriptions are as follows:

CARD — The input source program is punched cards. Produces a cross reference listing. CARD is the default input. If the option entry is omitted the input is assumed to be cards.

CDLIST — The input source program is punched cards. Produces both a source program listing and a cross reference.

DISK — The input source program having a file idertifier of COBOLW/SOURCE resides on disk. Produces a cross reference listing.

DKLIST — The input source program having a file identifier of COBOLW/SOURCE resides on disk. Produces bcth a program source listing and a cross reference.

LISTCD The input source program is punched cards. Produces a source program listing only.

LISTDK The input source program is on disk. Produces a source program listing only.

LISTTP* The input source program having a file identifier of SOLT is on magnetic tape. Produces a source program listing only.

TAPE The input source program having a file identifier of SOLT is on magnetic tape. Produces a cross reference listing.

TPLIST The input source program having a file identifier of SOLT is on magnetic tape. Produces both a source program and cross reference listing.

Internal File Names

Internal File Identifiers	External File Identifiers	Description
CARDS	CARDS	Input Source Cards
TAP-S	SOLT	Input Source Tape
DISKS	COBOLW/SOURCE*	Input Source Disk
CBXPRT	XREFER	Printer Output

*Refer to the COBOL Compiler Option NEW for the creation of this file.

Examples

Card:
```
? EXECUTE  COBOL/XREF
? DATA CARDS
CARD  C
   (source deck)
? END
```

Tape:
```
? EXECUTE COBOL/XREF
? DATA CARDS
TAPE
? END
```

Disk:
```
? EXECUTE COBOL/XREF
? DATA CARDS
DISK
? END
```

NOTE

If more than one cross reference file could exist, the FILE statement must be used to make each file identifier unique.

Certain options of the COBCL/XREF program are controlled by programmatic switches entered by the system operator. Refer to the SW control instruction attribute and the SW INPUT MESSAGE keyboard input message for details. The options and the programmatic switches and values required to control the options follow:

Switch	Value	Option
SW1	0	Do not cross reference literals.
SW1	1	Cross reference literals.
SW2	0	Vertical spacing for line printer is to be 6 line per inch, 56 lines per page.
SW2	1	Vertical spacing for line printer is to be 8 lines per inch, 76 lines per page.
SW3	0	No copy files required.
SW3	1	Copy files required.
SW5	0	CARD.
SW5	1	TAPE.
SW5	2	DISK.
SW5	3	TPLIST.
SW5	4	DKLIST.
SW5	5	LISTTP.
SW5	6	LISTDK.
SW6	0	Double space.
SW6	1	Single space.

6. BASIC

6.1. Introduction.

The Mark 9.0 release of BASIC contains no new language constructs.

7. FORTRAN

7.1. Introduction.

The Mark 9.0 release of FORTRAN contains no new language constructs.

7.3. Temporary Documentation.

7.3.1.

The following is a description of the capabilities for using remote files in Burroughs Standard FORTRAN (ANSI 66). The examples given assume the use of a TD830. Screen formatting on other terminals may require the use of other codes depending on the firmware of the terminal.

Burroughs FORTRAN permits the declaration of a remote terminal as a hardware input/output device. The following is an example of a FILE declaration statement to declare a terminal as remote.

```
FILE  8=RMT, UNIT=REMOTE, RECORD=1920
```

Only one remote file can be declared in a FORTRAN program. The entire screen (or any part thereof that is declared in the file declaration) is considered to be one record. The maximum size is 1920 characters. The record is written with 80 characters on a line (barring any carriage control characters). When the end of the line is reached, output proceeds on the next line.

Example:

```
      WRITE (8,100) (I(J),J=1,100)
  100 FORMAT (100A2)
```

The preceding example takes the first 2 bytes (after the 4 bits of type information) of each of the first 100 elements of array I and writes them to the remote terminal.

Executing a program that has a remote file declared and opened requires the following:

a. If the user is signed on to an SMCS, the following string is entered:

```
EX <program>/<name>
```

b. If the user is signed on to CANDE, execution proceeds as follows:

EX — Execute the workfile's object program.

or

EX <name> — Attaches the usercode to the unique name.

or

EX <family>/<name> — Complete name of program to be run.

Since a file is not opened until the execution of the first data transfer statement (READ, WRITE, etc.), a WRITE statement should be executed before any input from the terminal to insure that the file is open. If this is not done, the user may erroneously think that input is being supplied to the FORTRAN program, when it is actually being supplied to CANDE.

A convenient method for sending screen formatting characters to the terminal is through the use of hexadecimal data initialization. Hexadecimal constants, which may only be assigned in a DATA statement, are assigned to the variable without conversion or regard to type.

```
DATA HOME, BLINK /Z00C000000, Z0180000000/
DATA BRITE, RS, LF /Z03F0000000, Z01E0000000, Z0250000000/
```

The user must remember that the first four bits are type bits and are stripped off before transmission to the screen. This is why the first byte of each variable is 0 in the above example.

The following is an example of how to write using screen formatting characters (assume the program contains the previously given examples):

```
      WRITE (8,100) HOME, (LF, I=1,11), BLINK, BRITE, RS
  100 FORMAT (12A1, 38X, 2A1, 'HELP', A1)
```

The above example sends the cursor to the home position and clears the screen, then skips to the 12th line on the screen. It then skips over to column 39, prints HELP with the bright and blinking highlight characters turned on, and then terminates the highlighting with an RS.

More than one formatting character may be given within a variable. If in the example above we also have

```
DATA BLBR /Z0183F0000/
```

as well as the rest of the variables, the WRITE statement could look like the following:

```
      WRITE (8,100) HOME, (I(J), J=1,11), BLER, RS
  100 FORMAT (12A1, 38X, A2, 'HELP', A1)
```

Notice the only change in the FORMAT statement is A2 instead of 2A1.

To put the screen in forms mode, the following characters must be transmitted to the screen:

```
      DATA FMODE /Z027E60330/
      WRITE (8,100) FMODE
  100 FORMAT (A3)
```

The forms delimiting characters follow.

```
      DATA US, RS /Z01F000000, Z01E000000/
```

For more information on the use of forms characters refer to the TD830 manual, form number 1093788. The following is an example of a program that writes these forms characters and then reads from the field they delimit (remember that a READ in forms mode only receives the information contained within the delimited fields):

```
      WRITE (8,100) US, RS, FMODE
  100 FORMAT (37X, A1, 4X, A1, A3)
      READ (3,200) I
  200 FORMAT (I4)
      STOP
      END
```

The above example puts a 4 character forms field in the middle of the top line, then puts the screen in forms mode (nothing should follow the characters that put the screen in forms mode) returning the cursor to the beginning of the field. More than one forms field may appear on a screen. Transmitting from the beginning of the first forms field transmits all information in all the forms fields.

The following is an example of a program using most of the features of remote FORTRAN. It writes a message to the screen and receives input in forms mode. The input is retransmitted to the terminal in the middle of the screen. Two fields for input are given, both accepting information in A format (character).

```
      FILE   9=REMO,UNIT=REMOTE,RECORD=1920
      FILE   8=SPO,UNIT=SPO
             IMPLICIT INTEGER(A-Z)
             DATA HOME,REV,FMODE,US,RS,CR/Z00C000000,Z00E000000,
            +Z027E60300,Z01FC00000,Z01E000000,Z00D000000/
             DATA BLBR/Z0183F0000/
             WRITE (9,100)HOME,US,RS,US,RS,FMODE
         100 FORMAT(A1,20X,A1,4X,A1,100X,A1,4X,A1,A3)
             READ (9,200) I,J
         200 FORMAT(2A4)
             WRITE(9,300)HOME,(CR,K=1,11),BLBR,REV,I,RS,BLBR,J
         300 FORMAT(12A1,36X,A2,A1,A4,A1,A2,A4)
             STOP
             END
```

8. UPL

8.1. Introduction.

The Mark 9.0 version of the UPL compiler includes corrections to outstanding problems from the Mark 8.0 version of the UPL compiler. These corrections are documented in the following paragraphs.

8.2. Enhancements.

8.2.1.
The UPL compiler no longer generates incorrect code when a substructure is missing from a dummy declaration. The syntax error "SUBSTRUCTURE MISSING FROM DUMMY DECLARATION" is generated instead.

8.2.2.
The UPL compiler now displays the following warning message when the "&END" statement is missing from the source file and the UPL compiler reaches end of file in the source file (13-338-SWH109).

```
"MISSING &END"
```

This caused problems because conditional compilation blocks were not terminated.

8.2.3.
The UPL compiler now generates a syntax error when the statement of the following form is specified in the source file (C1-175-010019).

```
DECLARE (A,B,C)  FIXED;

A := -6 := C := 1;  % This statement is invalid.
```

The following syntax error is generated.

```
ADDRESS REQUIRED: ASSIGNMENT TO A VALUE IS NOT ALLOWED
```

8.2.4.
The default file type for a file declared in a UPL program is now "DISK". The default file type was "TAPE" (13-338-DB1000).

8.2.5.
"DO" blocks can be now nested 32 levels deep without requiring an identifier immediately following the word "DO".

8.2.6.
The UPL compiler no longer generates incorrect code when the "$ FORMAL CHECK" option is used. The UPL compiler used to generate incorrect code for formal parameters of type varying (01-700-60000Z).

8.2.7.
When the UPL compiler is compiling a library routine that is not a sequenced library file and a status request was entered to the UPL compiler, the UPL compiler used to respond with "SEQ <8 blanks>". The UPL compiler now responds with the current line number within the library file if the current source image does not contain a sequence number (13-338-ALSOC).

8.2.8.
The UPL compiler no longer responds with a reference to "SOL" in response to a status inquiry.

8.2.9.
The following warning message was grammatically incorrect when an underscore (_) character is used in the internal file name:

"UNDERSCORE REPLACEDBY PERIOD IN INTERNAL FILE NAME"

Now,

The warning message now reads:

"UNDERSCORE REPLACED BY PERIOD IN INTERNAL FILE NAME"

8.2.10.
The UPL compiler now writes the syntax error message to a file labelled "ERROR.LINE" printer when Program Switch 1 of the UPL compiler is equal to 1. This allows users of the "CANDE" program to easily retrieve the syntax error and warning messages on their terminal without having to list the complete source listing.

8.2.11.
The UPL compiler now generates the following syntax error message when the "$VOID" compiler option is used and the terminating sequence field number is not exactly 8 digits.

"SEQUENCE NUMBER MUST BE 8 DIGITS"

8.2.12.
The UPL compiler used to generate the following syntax error message when patching a formal declaration in a recompile.

"PARAM MISMATCH BETWEEN FORWARD AND PROCEDURE"

This syntax error message is misleading. The UPL compiler now generates the following syntax error message for the same error in addition to the previous message.

"CANNOT PATCH LEXIC LEVEL ZERO ON RECOMPILE"

8.2.13.
The UPL compiler now correctly checks the source file when a "CAT" operator is used. The UPL compiler used tc concatenate multiple source lines together. Concatnation between source lines is invalid.

Example:

```
00001000   "FIRST " CAT "SOURCE " CAT "LINE "
00002000   "SECOND " CAT "SOURCE " CAT "LINE"
```

The UPL compiler used to concatenate the two source line together to produce "FIRST SOURCE LINE SECOND SOURCE LINE". The UPL compiler now generates a syntax error message.

8.2.14.
The UPL compiler now terminates a compile with a syntax error if the "CREATE_MASTER" option is specified and not specified in the first source line. The UPL compiler used to terminate without any syntax errors and removed the existing source file.

8.2.15.
The UPL compiler no longer converts the period (.) character to the underscore (_) character when the period (.) character appears in a string delimited by the quote mark (") character. Alsc, the UPL compiler correctly converts the period (.) character to the underscore (_) character when a cuote mark (") character is missing from a string.

8.2.16.
A UPL program no longer terminates with "NO EOF PROVISION --DS OR DP" when monitoring procedures at run-time.

8.2.17.
The UPL compiler used to generated the following syntax error message when a semicolon (;) character did not follow a procedure end statement.

"IDENTIFIER MUST FOLLOW PROCEDURE"

The UPL compiler now generates the following syntax error message.

"MISSING SEMICOLON"

The UPL compiler terminates the compile once this error is detected.

8.2.18.
The UPL compiler now correctly recognizes the word "THEN" if it appears in the last four columns of a source line and the next source line contains a ampersand (&) character in column one. The word "THEN" was previously ignored.

8.2.19.
The UPL compiler no longer terminates with "INVALID SUBSCRIPT - DS CR DP" when a quote (") character is missing.

8.3. Temporary Documentation.

8.3.1. FIND_DUPLICATE_CHARACTERS.

Syntax of the Command:

```
FIND_DUPLICATE_CHARACTERS (<ref-string-1>,<count>, ----->

  >-- <character>,<ref-string-2>); ---------------------|
```

A duplicate character is defined to be three or more contiguous characters that are identical; thus "AAA" is a string containing duplicate characters, while "AA" contains no duplicate characters.

The "FIND_DUPLICATE_CHARACTERS" operator scans the text of the string referenced by ref-string-1, looking for duplicate characters. If a string of duplicate characters is found, scanning continues until the end of the string of duplicate characters is reached. The operator returns the following values.

ref-string-1	--modified to reference the substring that was not scanned; thus its first element is the character that immediately follows the duplicate character string.
count	--the number of duplicate characters found.
character	--the duplicated character.
ref-string-2	--references the substring that was scanned before the duplicate character string was found. This substring does not contain duplicate characters.

NOTE

Ref-string-1 and ref-string-2 must both be reference variables.

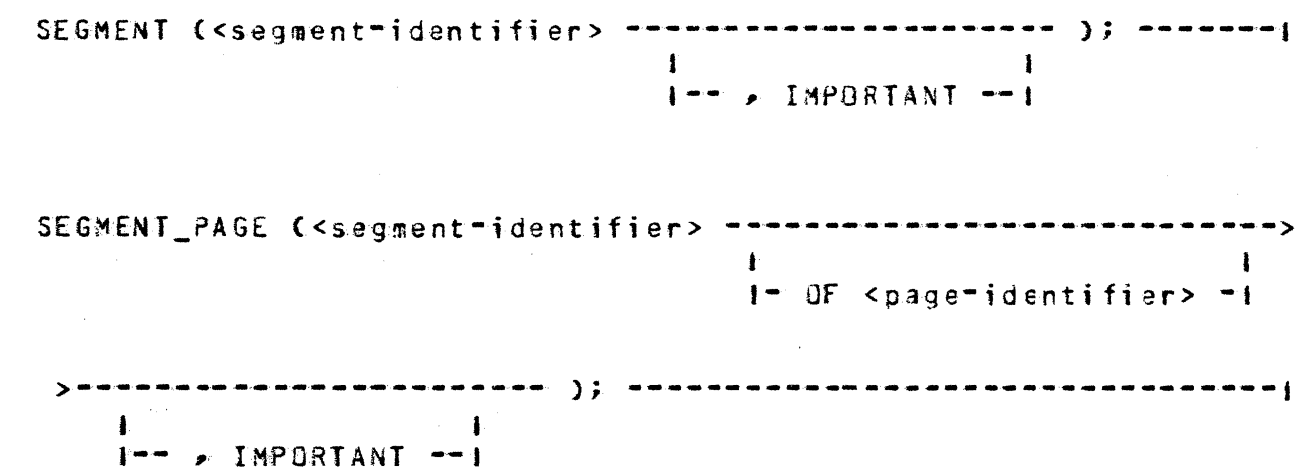
Example:

```
DECLARE STRING CHARACTER(21),
        REF_STRING_1 REFERENCE,
        REF_STRING_2 REFERENCE,
        CHAR   CHARACTER(1),
        COUNT  FIXED;
STRING:="THIS IS THE PLAAAAACE";
REFER REF_STRING_1 TO STRING;
FIND_DUPLICATE_CHARACTERS (REF_STRING_1, COUNT, CHAR,
    REF_STRING_2);
```

"FIND_DUPLICATE_CHARACTERS" would set the following values:

```
REF_STRING_1 = "CE"
COUNT = 5
CHAR = "A"
REF_STRING_2 = "THIS IS THE PL"
```

8.3.2. Important Segments.
Syntax of the Command:


SEGMENT (<segment-identifier>); ;
, IMPORTANT
SEGMENT_PAGE (<segment-identifier>
OF <page-identifier>
); ;
, IMPORTANT

The "IMPORTANT SEGMENT" construct is an extension of the UPL "SEGMENT" statement, which divides a program's object code into overlayable sections. In UPL, the programmer must explicitly segment the program if overlaying is to be allowed; if no "SEGMENT" statements appear, the entire program is compiled into a single segment. Run-time memory requirements for a program should decrease when that program is segmented, because not all segments must be resident in memory simultaneously.

When a program references a non-resident segment of code, that segment must be moved into main memory from disk. If no memory space is available, the newly called segment is written into the space occupied by a less important segment. This process is called "overlaying". The "IMPORTANT SEGMENT" construct gives a segment some protection from being overlaid.

Refer to volume 1 of the Software Operation Guide (form # 1108982) for a discussion of the Memory Management System.

The decay factor field (See Memory Management discussion) in the code segment dictionary is three bits long, but may have only one of two values--zero (2(1)000) for unimportant, or seven (2(1)111) for important. The MCP uses the direct complement of the decay factor; thus, in a memory dump, a decay-factor value of zero means the segment is important, and a value of seven means it is not important. The actual decay-factor assigned to a segment is stored in the memory link, and is labeled (in a dump) as the incoming, or current interval decay factor.

8.3.3. Processor Time.
The construct "PROCESSOR_TIME" yields the accumulated processor time since BOJ in tenths of a second; the returned value is a BIT(20) data item.

The following example illustrates the use of this construct.

```
DECLARE (PROC_TIME,HOURS,MINUTES,SECONDS,TENTHS) BIT(20);
          /*  E A R L Y    C O D E  */
    PROC_TIME := PROCESSOR_TIME;
         /*   C O D E   T O   B E   T I M E D  */
    PROC-TIME := PROCESSOR_TIME - PROC_TIME;
    HOURS     := PROC_TIME / 36000;
    MINUTES   := PROC_TIME MOD 36000 / 600;
    SECONDS   := PROC_TIME MOD 600 / 10;
    TENTHS    := PROC_TIME MOD 10;
         /*   L A T E    C O D E   */
FINI;
```

8.3.4. SECURITY and SECURITYUSE.
Two special words, "SECURITY" and "SECURITYUSE" have been added to UPL, and function identically to the keywords "PROTECTION" and "PROTECTION_IO". "PROTECTION" and "PROTECTION_IO" have been de-implemented in the Mark 9.0 Software Release. Both new security attributes can be specified in the "FILE" declaration statement of a UPL program; specification overrides the default protection values.

Valid settings for the "SECURITY" attribute are:

0 - Default
1 - Public
2 - Private
3 - Guard

Valid settings for the "SECURITYUSE" attribute are:

0 - I/O
1 - Input
2 - Output

For both attributes, zero (0) is the default value.

9. DMSII

9.1. Introduction.

The Mark 9.0 release of DMSII contains only minor changes, designed to increase performance and throughput. Databases are completely compatible between the Mark 8.0 and 9.0 release.

Databases being brought directly from the Mark 7.0 level to the Mark 9.0 level must be converted. The DMSII section of the Mark 8.0 System Software Notes (Section 9) is included in a supplemental document titled DOCUMENT/DMSRELLET for the users converting directly from Mark 7.0 to 9.0. Also included is the DOCUMENT/DMS, which was included with the Mark 8.0 release, which further documents the changes to DMSII in the Mark 8.0 release.

The Mark 9.0 release of the COBOL74 compiler contains the necessary constructs for database access. The COBOL74 compiler uses the same database library files as the COBOL compiler. For complete details, refer to the COBOL74 documentation.

Included with the DMSII Mark 9.0 release is a printer backup file labeled DOCUMENT/DMSUTILS which contains the operating instructions for the DMSII utility programs.

10. NDL

10.1. Introduction.

In the Mark 9.0 release of the Network Definition Language (NDL) a number of enhancements have been implemented. The major NDL enhancements are remote files with simple headers, DETACH with CLEAR, new MCS control messages for COBOL74, and the new library request for COBOL74.

All references in this section and the DOCUMENT/NDL refer to the B1800/B1700 Systems Network Definition Language (NDL) Reference Manual, form number 1073715 dated 10/78.

Documentation on the Mark 9.0 release of NDL is available with the program product in the form of a printer backup file labeled DOCUMENT/NDL.

10.2. Enhancements.

10.2.1.

The following problems reported in the Mark 8.0 System Notes have been corrected (the number in parentheses refers to the number of the problem reported in the Mark 8.0 System Notes):

a. The Network Controller will display the message "OVERFLOW @ FULL MESSAGES DISCARDED" and continue to execute in an overflow condition instead of aborting (12-618-270002).

b. The Network Controller no longer aborts with "NO PROVISION FOR EOF ON NDL/NIF FILE" (13-338-DM1498).

c. The Network Controller no longer aborts with "NO PROVISION FOR EOF ON STATION.SA" (51-017-000003).

d. Intergrity Integrity of line and station message counters is no longer violated by switched line disconnects and reconnects when stations are queued on the disconnecting lines in a line group (61-244-899249).

10.2.2.

In the Mark 9.0 release of NDL, a program can open a remote file with simple headers. With this type of remote file, the only valid message requests which may be issued are STATUS, REMOTE FILE INFO, and GOODRESULTS. The program opening a simple headers file will not receive any OPEN or CLOSE messages. Refer to DOCUMENT/NDL, paragraph 10.3.1.

10.2.3.
A new station attribute has been added, STATION(OPENED). STATION(OPENED) is set to true if the station is currently attached to a remote file. Refer to DOCUMENT/NDL, paragraph 10.3.2.

10.2.4.
A new option for "DETACH" has been added for the Mark 9.0 release of NDL. An MCS may now request that the LSN of a station being detached from a remote file be detached from the FIB of the remote file (DETACH WITH CLEAR). Refer to DOCUMENT/NDL, paragraph 10.3.3.

10.2.5.
The construct "QUEUE CRF" was changed for the Mark 9.0 release to queue the message to the remote file which originally used the station, STATION.FIRST.OPENER. Refer to DOCUMENT/NDL, paragraph 10.3.4.

10.2.6.
A new declaration, COBOL74DEC, and a new request, COBOL74SEL, have been added to the NDL/LIBRARY to permit COBOL74 programs to issue a "SEND" with advancing feature (for CRT devices only). Refer to DOCUMENT/NDL, paragraph 10.3.5.

10.2.7.
The advent of COBOL74 has brought with it the need for new control messages. The new control messages are described in DOCUMENT/NDL paragraphs 10.3.5. thru 10.3.12.

11. CANDE

11.1. Introduction.

B1000 CANDE provides generalized file preparation and updating capabilities in an interactive, terminal-oriented environment. The CANDE program is a Message Control System (MCS) that runs in conjunction with the B1000 NDL system. The NDL-generated Network Controller performs all the functions related to data communications, while CANDE performs the updating and text-editing functions.

Major CANDE program enhancements include the ability to specify a multi-file id for an output file, a new program switch to determine whether or not file security is enabled, a secure video field for entering the password during log-on at TD830 terminals, and the ability to queue up commands during the FIND, LIST, and REPLACE commands.

Documentation on the CANDE program is available with the program product in the form of a printer backup file labeled DOCUMENT/CANDE.

11.2. Enhancements.

11.2.1.

The following problems reported in the Mark 8.0 System Notes have been corrected (the numbers in parentheses refer to the number of the problem reported in the Mark 8.0 System Notes).

a. The COMPILE command now uses the file type specified in the command correctly (13-338-DM1320).

b. CANDE now zips a remove of the files, WORK, RECOVERY, and PAGE.FILE correctly even if they do not reside on system disk (13-338-DM1350).

c. If the terminal is in scroll mode, the LIST of an external file no longer has blank lines inserted in it (13-338-DM1351).

d. When a WRITE command is processed, CANDE only zips a remove of the backup file that would be created to prevent a duplicate file condition (13-338-DM1377).

e. The password can be changed with the PASSWORD command if there is a workfile present (13-338-DM1396).

f. Transmission of "NEXT " no longer affects PAGE LIT mode (13-338-DM1398).

11.2.2.
Program switch 5 was implemented to determine whether or not CANDE will run with file security. If program switch 5 is not zero, file security is enabled. Refer to DOCUMENT/CANDE.

11.2.3.
A multi-file id and a pack id can be specified for an output file if a user logs on to CANDE with a privileged usercode/password. In addition, output file names prefixed with an asterisk (*) will no longer be expanded so that the multi-file id contains the user's usercode. Refer to DOCUMENT/CANDE.

11.2.4.
Commands can be queued up during a FIND, LIST, or REPLACE command.

11.2.5.
A secure video field is displayed on TD830 terminals during the log-on for entering passwords.

11.2.6.
The format of the response to the WHAT command was changed to conserve datacomm usage.

11.2.7.
The :TEXT option of the FIND and REPLACE commands was changed to allow any abbreviation of TEXT.

11.2.8.
Two new file types have been added for this release: FORTRAN77 and COBOL74.

11.2.9.
The CANDE program has been compiled with certain code segments marked as "important" for optimal use with the pricrity memory management system:

```
(00,01)   % OUTSIDE
(00,02)   % CMD_DRIVER
(00,07)   % TERMINAL_OUT
(01,01)   % BIN_SEARCH
(01,0A)   % PAGE_DRIVE
(01,0B)   % PAGE_SHIFT
(01,0E)   % PAGE_IN_SETUP
(01,0F)   % PAGE_IN_SETUP_A
(01,10)   % PAGE_OUT
(01,11)   % PAGE_OUT_UTILS
(01,24)   % AUDIT_HANDLER
(02,00)   % SCAN_STUFF
```

12. TEXT/EDITOR

12.1. Introduction.

The Mark 9.0 release of the B1000 TEXT/EDITOR program includes enhancements and corrections to reported problems in the MARK 8.0 release of the program.

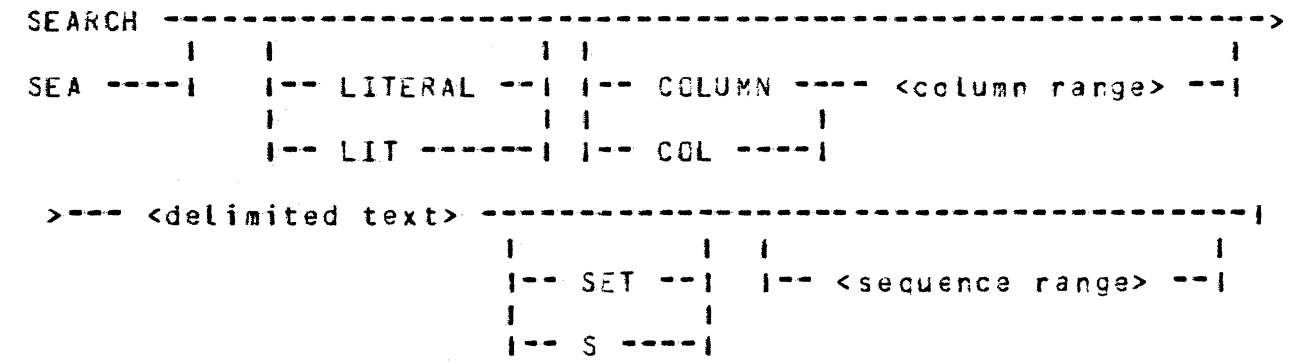
12.2. Enhancements.

12.2.1.
The "PB" command no longer causes an "INVALID SUBSTRING -- DS OR DP" condition when a printer backup file is not provided. The message "MISSING FILE NAME" is displayed instead.

12.2.2.
The "PD" command now also displays the creation date of the file along with the other information.

12.3. Temporary Documentation.

12.3.1.
The "SET" option has been added to the "SEARCH" command. If the "SET" option is specified, the cursor is positioned to the first character of the string in the source record which matched the <delimited text>.



12.3.2.
"Important" segments are now specified directly in the source code. The segments marked as "important" for use with Extended Segment Decay and Priority Memory Management are as follows:

```
(0,00)    % ZERO
(0,02)    % SYNTAX-SCAN
(0,06)    % TOKEN DECODER
(0,07)    % RECORD-UPDATE-INITIAL
(0,08)    % RECORD-UPDATE-FINAL
(0,09)    % UPDATE
```

12.3.3.
The following synonyms and abbreviations to command keywords are supported in the TEXT/EDITOR program:

Keyword	Synonyms/Abbreviation
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.

13. SMCS

13.1. Introduction.

The B1000 Supervisory Message Control System (SMCS) is provided for users requiring interactive data communications systems, offering the capability of a comprehensive Message Control System interface to the NDL Network Controller. The SMCS is intended to be the supervisor for a data communications software system which includes such Burroughs software as CANDE, TEXT/EDITOR, HASP, RJE, as well as other on-line packages of either Burroughs or customer origin.

Documentation on the SMCS program is available with the program product in the form of a printer backup file labeled DOCUMENT/SMCS.

13.2. Enhancements.

13.2.1.

The following problems reported in the Mark 8.0 System Notes have been corrected (the numbers in parentheses refer to the number of the problem reported in the Mark 8.0 System Notes:

a. SMCS no longer aborts with "NAME/VALUE STACK Overflow" when more than 240 characters are PASSed to SMCS (13-338-DM1380).

b. SMCS no longer hangs "waiting message space ir queue for write." (13-338-DM1452).

c. If two stations are logged on to SMCS with the same usercode/password, both of those stations can successfully execute a program that opens a remote file (13-338-DM1466).

d. SMCS approves the "OPEN" of a remote file for a program executed from a remote ODT station if the station is not logged on to the SMCS program with the "USER" command (13-338-DM1467).

e. SMCS will find the name of a program in the JOBS file correctly, due to new implementation of the JOBS file format, and will attach all users to the correct copy of the program (13-338-DM1501).

13.2.2.
All references to SPO have been changed to ODT (Operator Display Terminal). A remote SPO station is now referred to as a remote ODT station. The "SEND TO SPO" command is now "SEND TO ODT."

13.2.3.
The format for declaring a job in the SMCS "JOBS" file, SMCS/JOBS, was changed in order to allow a user to have the program identifier specified in the "SIGN ON", "PASS", or "START" command to be different than the actual name of the program to be spawned by the SMCS program as a result of the SIGN ON, PASS, or START command. In addition, the flags for the various program attributes have been simplified. If a JOBS file exists from a previous release of SMCS, the SMCS program will automatically convert it to the new format at beginning of job. This conversion is not possible, however, if the JOBS file has a usercode as the family name (multi-file-id). Refer to DOCUMENT/SMCS.

13.2.4.
The manner in which the SMCS program approves a remote file "OPEN" for a program executed from a remote ODT station with the "?EXECUTE" MCP control instruction was changed for the Mark 9.0 release of SMCS so that the LSN which executed the program is always approved in the OPEN request from the program. In previous releases of the SMCS program, a remote ODT station was forced to log on to SMCS prior to executing the program, or was forced to have the program open a remote file which was declared in the Network Controller and contained exactly one station (the LSN executing the job); otherwise the file OPEN request was denied. A second remote file open (I-O) is approved for the program if it is for stations not controlled by the SMCS program.

13.2.5.
All stations which are not declared as remote ODT stations are now forced to log on to the SMCS program with the USER command before making use of the SMCS program capabilities. Until the user logs on to SMCS, the only commands available from that station are HELP, ID, REPORT, SEND, and TRANSLATE. While the user is logged on, the ID command is not allowed.

13.2.6.
A user cannot SIGN ON or PASS to a program if that program is not declared in the "JOBS" file.

13.2.7.
The "SYSTEM" input command has changed. It is now used to specify the current system identifier and to control lower case input message translation for the system. However, the TRANSLATE option of the SYSTEM command is used only to set the default value, either ON or OFF and each terminal user has the option to change it for the individual terminal by using the "TRANSLATE" input command. Refer to DOCUMENT/SMCS.

13.2.8.
The header for the PROGRAM-PASS message now contains Time, Tallies and Toggles, and terminal type extracted from the originating message header.

13.2.9.
The SMCS will now forward any data string following the SIGN ON <program-name> as second input to the program identified by program-name if LOG-ON option is specified in the JOBS file for that program and the user is logged on to SMCS with the USER command. The SMCS program sends the user's usercode/password pair as first input and any string following the program name in the command syntax as second input.

13.2.10.
Program Switch 2 was implemented to inhibit the automatic execution at beginning of job of SMCS of any job that was in the JOBS file with the AUTO-START option specified. If program switch 2 = 1 the AUTO-START option in the JOBS file for all jobs will not be looked at.

13.2.11.
A new command feature, the "START" command, provides the capability to start a job from the host operator display terminal (ODT). The specified job must be in the JOBS file and is spawned by SMCS in the same way as those automatically spawned by SMCS as a result of the AUTO-START option specified in the JOBS file. Refer to DOCUMENT/SMCS.

13.2.12.
For both the "MAKE" and "CHANGE" commands, a station identifier is optional for those changes which affect only the initiating station.

13.2.13.
A new command feature, the "HARDWARE" command, provides the capability to specify to the SMCS program the physical print line size of the system's line printer. The SMCS program uses this information, which is stored in SMCS'S INFO file, when a listing of the trace file SMCS/TRACE is initiated with the TRACE list command. Refer to DOCUMENT/SMCS.

13.2.14.
When the "JOBS" input command is processed by the SMCS program or when a MX or WY MCP command is entered as a ZIP string, the SMCS will first determine if the program RD is executing and, if so, if it will support another PASSed station. If these conditions are true, the request is PASSed to RD for a response; otherwise, the SMCS requests the function of the MCP.

13.2.15.
Automatic log-off has been implemented for all staticns which go "not ready". If the station is signed on to a program and the EXCEPTION option is specified in the JOBS file for that program, SMCS sends an exception indicator to the program.

13.2.16.
The "SIGN OFF" command was changed to allow the CLEAR option. If CLEAR is specified in the command syntax, the staticn originating the "SIGN OFF" is Detached and Cleared from the program's remote file FIB. Refer to the NDL section in this document for information on DETACH-WITH-CLEAR. Refer to DOCUMENT/SMCS.

13.2.17.
SMCS now specifies the denial-reason in the remote file OPEN Reply message:

0 = No reason (approval value).
2 = Invalid COBOL74 password.
4 = Remote file table or program table is full.
6 = Invalid LSN list.

13.2.18.
The "PASSWORD" command has been implemented for users of COBOL74. One system password is set at the host Operator display terminal (ODT). For all COBOL74 ENABLE/DISABLE request messages forwarded to SMCS from the Network Controller which involve the password option, the SMCS program examines the password field which it maintains in the SMCS/INFO file. If this master password is blank no check is made; otherwise, password validation will be performed. If the password in the OPEN message frcm the Network Controller for the COBOL74 program is not the same as SMCS's system password, SMCS denies the OPEN with denial reason equal to 2. Refer to DOCUMENT/SMCS.

13.2.19.
The Program-PASS message format for a remote file with protocol type = 2 now contains the station's signal character. Refer to DOCUMENT/SMCS.

13.2.20.
The following changes were made to the SMCS program in order to allow a COBOL74 program to be executed through SMCS:

1. If the PASSWORD.USED field is set in the OPEN message (it will be set on an ENABLE before a RECEIVE), password verification is performed.

2. When the OPEN is approved, SMCS issues a CHANGE command to the Network Controller program to change to the diagnostic request for each LSN in the OPEN message. The diagnosic request is declared in the TERMINAL section of the Network Controller program as follows:

    DIAGNOSTIC = POLLTCTD: RECEIVE, COBCL74SEL: TRANSMIT.

3. When the CLOSE message is received, if the final close bit is set in the CLOSE message, the SMCS program issues a change request to the Network Controller to change each LSN back to the regular request.

4. When an ATTACH is processed, the SMCS changes the LSN to the diagnostic request. The LSN is changed back to the regular request when the DETACH is processed for the LSN.

5. SMCS will check password on ENABLE/DISABLE requests and deny with reason=2 if it does not match.

6. SMCS will handle C8 type messages fowarced by the Network Controller by changing the type to C9 and sending it to the station.

14. RJE

14.1. Introduction.

The Mark 9.0 release of the B1700 Remote Job Entry (RJE) system contains two separate RJE software packages:

a. An SDL Data Comm handler and INPUT/OUTPUT program (RJE).
b. An SDL Data Comm handler (RJE/DCH).

14.2. Enhancements.

14.2.1. RJE.

The RJE program has no major changes for the Mark 9.0 release.

14.2.2. RJE/DCH.

The RJE/DCH program has no major changes for the Mark 9.0 release.

14.3. Temporary Documention.

14.3.1. Run-Time Parameters.

IDLE.ON.LOG.OFF

When running on leased lines and the host system terminates the session, the RJE program attempts 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 communications 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 communications link whenever the data communicatcns link is broken if using a leased line.

NO.REMOTE

This parameter requests the RJE program not tc oper a remote file for communications with the SMCS. This parameter also overrides the program switch 9 setting.

Default: No remote file is opened.

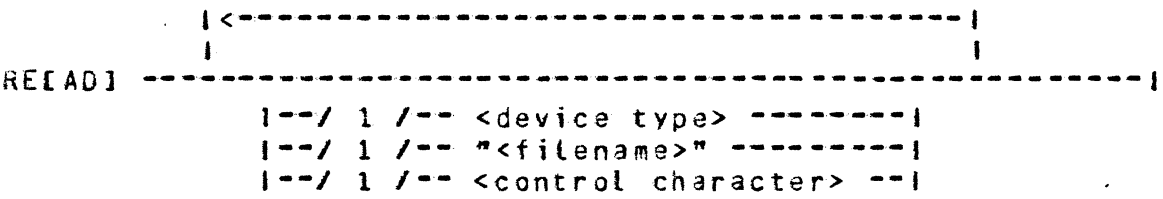
REMOTE

This parameter instructs tne RJE program to read from a remote file and process the messages as if they were entered from the Operator Display Terminal (ODT). This option car only be used

in conjunction with the SMCS program product.

Default: No remote file is opened.

14.3.2. READ Local Control Command.
The ".READ" command has the following syntax:



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 from which the RJE program is to read, 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 (?) character. 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" (@)@@) 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"
```

14.3.3. 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 output messages added in the Mark 7.0 release 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.

14.3.4. RJE Files.
The following files were added to the RJE program for the Mark 7.0 release:

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 9.0 release of the B1000 Host RJE program package contains several enhancements over the Mark 8.0 Host RJE program package.

15.2. Enhancements.

15.2.1.
The RJE/AUTOBACKUP program supports the Mark 9.0 printer backup format.

15.2.2.
The RJE/AUTOBACKUP program no longer halts with "INVALID SUBSTRING --DS OR DP" when the "SORT_SUMMARY_FILE" gets full.

15.2.3.
The RJE/AUTOBACKUP program no longer halts with "INVALID SUBSTRING -- DS OR DP" when an operand is missing after the "EQUAL" and "RANGE" options in the "*PB <file-id> KEY" command.

15.2.4.
The RJE/AUTOBACKUP program now correctly handles records with more than 100 blank characters.

15.2.5.
The RJE/AUTOBACKUP program no longer sends an extra end of file separator "<FS>2" character when sending an ending label record for a printer file.

15.2.6.
The RJE/AUTOBACKUP program now correctly handles the printing of space records when performing single or double spacing on the line printer.

15.3. Temporary Documentation.

15.3.1.
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.

15.3.2.
The RJE/CONTROLLER program cannot be executed without a usercode or with a non-privileged usercode.

15.3.3.
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.

15.3.4. 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
```

*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 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 B1800/B1700 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
```

15.3.5. 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.

15.3.6. 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. RJE3780

16.1. Introduction.

The B1000 RJE3780 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 B1000 remote computer. The central computer must be using the IBM 3780 data communications characteristics. The Mark 9.0 release of the B1000 RJE3780 program includes corrections to outstanding problems in the Mark 8.0 B1000 RJE3780 program.

16.2. Enhancements.

16.2.1.
The RJE3780 program now handles the <ESC>M characters correctly. When the <ESC>M characters are received by the program, the print line feed is suppressed causing the line printer to overprint the text which follows.

16.2.2.
The RJE3780 program now can send blank records in both compress and noncompress mode (13-338-DM1419).

16.2.3.
The RJE3780 program now waits to open a new backup printer file until a write to the print file is performed. The program used to open a new backup print file immediately after closing the current file regardless of whether there were any records for the file.

16.2.4.
The RJE3780 program now sends a "WACK" character when a message is received from the host system and the "@FILE" file is full. The program used to send a "NAK" character which could cause the host system to time-out. The "WACK" character is sent in response to enquiries from the host system until there is room to receive the next message. When there is enough room to receive the next message the appropriate "ACK" character is sent to the host system and data transmission continues (06-235-12403).

16.3. Temporary Documentation.

16.3.1. HOT READER Function.

The HOT READER function for the RJE3780 program is invoked by setting program switch 7 at BOJ. If this switch is equal to 1 (SW7=1), the RJE3780 program forces the input primary file device type to be a card reader. It then opens an input primary card file, which has the internal name of "INPUTFILE". If the file is present, the program bids for the line. If accepted, the RJE3780 program starts sending the file. If the host has control of the line, RJE3780 waits until the host gives up control of the line and attempts the line bid. If the file is not present, the program displays an error message on the Operator Display Terminal (ODT) and on the line printer and goes to end-of-job (EOJ).

When using the HOT READER function, the primary file must be present and must be a STREAM card deck; otherwise the program terminates. The RJE3780 program checks the value of switch 7 at BOJ. Since this value is checked only at beginning-of-job (BOJ), the HOT READER function cannot be turned off once it is turned on.

While running with a HOT READER, the stream deck which is the input primary file, remains open. Any card put in the reader is scanned by the RJE3780 program for a "<?>END", a "<?>DATA <filename>", or a ".Sx" card. An END or a DATA card causes End of File (EOF) indicator to be sent to the host for the current file. The next card encountered causes the start of a new file to be sent to the host. A ".Sx" card is processed as a request to start sending the secondary file to the host.

When End of File ("<?> TERMINATE") is encountered on the input primary file, the RJE3780 program sends an End of File indicator to the HOST for the current file being transmitted if necessary, closes all open files, and goes to end-of-job (EOJ).

If the HOT READER function is invoked, a ".Sx" command from the ODT is not valid.

The following is an example of the input primary file when using the HOT READER function:

```
<?>STREAM RJE3780/CARDS
<?>DATA CARDS
 /*SIGNON        REMOTE IC
<?>END
<?>DATA CARDSIN

   <card deck to be transmitted to the host>

<?>END
<?>DATA CARDS
 .SO PAYROLL
<?>END
<?>DATA CARDS
 .SO ACCOUNTING
<?>END
<?>TERMINATE RJE3780/CARDS
```

16.3.2. Handler Notes.

To determine the minimum buffersize to specify in the data communications handler, take the record size of QFILE from RJE3780 subtract 50 and add 4, if non-transparent, or 6 if transparent. If the record size of QFILE is 562 bytes, the minimum buffersize declared in the handler must be 516 bytes, if operating in transparent mode.

The IBM 3780 protocol allows for an ending block to be answered by a block of data. If this ability of the protocol is to be used, the minimum buffersize must be multiplied by 2, since the transmitted message and the received message are both stored in the one I/O buffer.

16.3.3.

If the DISPLAY option is set to a value "n", "SO DISPLAY <n>", the first 72 characters of "n" lines of every message from the host are displayed on the ODT. ".RO DISPLAY" resets the value to 0. "n" must be less than 16. The initial value is determined by program switch 1. Program switch 1 can be set to a value from 0 to 15 at execution time of the RJE3780 program.

16.3.4.

The following output message has been added and is displayed by the data communications handler:

WAITING FOR MCS QUEUE NOT FULL

The handler has a message to queue for the RJE3780 program but the queue is full. The handler NAKs any further messages from the host until the queue full condition is relieved.

16.3.5.
The following output messages were added to the RJE3780 program:

NOT ALLOWED IN HOT READER MODE

This message is displayed in response to a ".Sx" command from the local ODT. When using the HOT READER function, a ".Sx" is only valid in the stream deck.

STREAM DECK MISSING

This message appears on both the ODT and the line printer. It indicates that the HOT READER function was requested (SW7=1) but the STREAM deck was not present. The RJE3780 program goes to end-of-job (EOJ) and must be re-executed.

THE INPUT FILE IS A HOT READER AND

This message is displayec in response to a ".STATUS" ODT input command. It indicates that the HOT READER function has been invoked. The standard response follows on the next line.

<?END> CARD MISSING-EOF ASSUMED

This message is displayed on the ODT whenever a card deck is being sent to the HOST and a <?>DATA card is read before a <?>END card. The RJE3780 program assumes the user wants to start sending a new deck, and therefore end the deck currently being sent.

16.3.6.
If the host system requires a sign-on or log-on message to be sent, there is no facility in the RJE3780 program to automatically send a sign-on or log-on message to the host. The sign-on or log-on message must be entered by the operator.

17. HASP

17.1. Introduction.

The Mark 9.0 release of the B1000 HASP program contains several enhancements over the Mark 8.0 HASP program.

17.2. Enhancements.

17.2.1.
The HASP program now allows a three character remote ID field. Previously the field was restricted to two characters. Because of this change, the parameters file created by the Mark 8.0 (and earlier) version of the HASP/MODIFIER program is not valid for the Mark 9.0 release of the HASP program. The Mark 9.0 version of the HASP/MODIFIER program must be executed in order to create a valid parameters file.

17.2.2.
The HASP program now handles the date and time change at midnight and the "TR" MCP command correctly. The HASP program used to remain in "WAIT" status after the date and time changed at midnight and did not process any HASP commands (13-338-DM1331).

17.2.3.
The HASP/SPOOL program now correctly handles the special forms message $SRM<N>.PR<M>, where N = the remote system ID and M = the form number (13-338-DM1416).

17.2.4.
A problem with the HASP/SPOOL program has been corrected where the end of file was not correctly recognized for spool files. If the "NEW-SPOOL" option was used, permission was not granted to send or receive any new or waiting streams. This problem prevented the use of the "NEW-SPOOL" option (13-338-DM1413).

17.2.5.
The HASP program no longer "ACCEPTS" Operator Display Terminal (ODT) input until the "COMMUNICATION ESTABLISHED" message is displayed.

17.2.6.
The HASP program now terminates correctly when a ".CLOSE" command is entered. The program finishes transmission of all data streams and then logs off the host system even when there are line errors.

17.3. Temporary Documentation.

17.3.1. HASP MODIFIER.

The MEM = <integer> parameter is used to specify how much dynamic memory the HASP program is to use. The integer specified must be equal to or greater than the minimum dynamic memory required to run. The minimum dynamic memory for HASP in direct mode can be calculated using the following expression:

```
(32*(BUFFER+7)) + (1211*(MAX.PRT+MAX.PUN))
      + (IF PUNCH = 96 THEN 512 * 8)
```

When running HASP in SPOOL mode, the minimum dynamic memory for HASP can be calculated using the following expression:

```
(32*(BUFFER+7)) + (8*(MAX.PRT + MAX.PUN))
```

If the MEM parameter is not used, HASP/MODIFIER calculates the dynamic memory for HASP in direct mode using the following expression:

```
(32*(BUFFER + 14)) + (12061*(MAX.PRT + MAX.PUN))
      + (IF PUNCH = 96 THEN 512*8)
```

If the MEM parameter is not used HASP/MODIFIER calculates the dynamic memory for HASP in spool mode using the following expression:

```
(32*(BUFFER + 14)) + (8*(MAX.PRT + MAX.PUN))
```

18. SYSTEM UTILITIES

18.1. Introduction.

The B1000 system utility programs are designed for efficient and consistent operation of the B1000 computer system. Existing utility programs are continually modified and enhanced, thus maintaining the quality of the programs. As new areas of application are defined, new utility programs are developed to accomplish the tasks.

This section contains brief descriptions of new utility programs that are being released for the first time with the Mark 9.0 System Software. Also, enhancements and modifications made to existing utility programs are described here. The utility programs named below are discussed in this section.

CHECK/LOAD.DUMP	SYCOPY
*CREATE/TABLE	SYSTEM/BACKUP
DISK/COPY	SYSTEM/COPY
DISKETTE/COPY	SYSTEM/DISK.INIT
DISKMAP/UTILITY	*SYSTEM/FILE.INIT
DMPALL	*SYSTEM/IS.MAINT
LOGCONVERT	SYSTEM/LOAD.CAS
PACK/INIT	SYSTEM/LOAD.DUMP
SORT/COLLATE	SYSTEM/SPOLOGOUT
SQUASH/USER.DISK	

* = NEW UTILITY

Volume 2 of the System Software Operation Guide form #1108966 contains complete documentation on the system utility programs, describing the functions and operation of each.

Temporary documentation describing the operation and function of the three new utility programs, information on the new backup format for Mark 9.0, operating instructions for the enhanced SYSTEM/SPOLOGOUT program, operating instructions for the SYSTEM/LOAD.CAS program, and changes to the record format of LOGCONVERT are included in a printer backup file named DOCUMENT/UTILITY supplied as a supplement to these System Notes.

18.2. Enhancements.

18.2.1. CHECK/LOAD.DUMP.

CHECK/LOAD.DUMP is a companion utility of SYSTEM/LOAD.DUMP. Accordingly CHECK/LOAD.DUMP will be deleted when SYSTEM/LOAD.DUMP is deleted in the next software release. Refer to paragraph 18.2.18.

18.2.2. CREATE/TABLE.
CREATE/TABLE is a new program which replaces SORT/COLLATE. Refer to DOCUMENT/UTILITY.

18.2.3. DISK/COPY.
DISK/COPY is deleted with the Mark 9.0 release. The abilities of DISK/COPY are encompassed within SYSTEM/COPY.

18.2.4. DISKETTE/COPY.
A correction to the syntax for creating a pseudo deck is required. The stated syntax of "COPY FDXDSK <FILENAME> 96 1 PSEUDO" does not work. The output file ID is required. For example: "COPY FDXDSK <FILENAME> 96 1 DUMMY PSEUDO".

Setting SW2 to 1 will cause DISKETTE/COPY to halt on a data error and request the operator's OK via an accept message before continuing. Without SW2 set DISKETTE/COPY deletes the sector in error, displays an error message and continues.

18.2.5. DISKMAP/UTILITY.
The following problem has been corrected: When a disk pack was "SN"ed to become an additional system pack and the other system disk was a 5N head-per-track disk, DISKMAP/UTILITY would fail, showing 64 missing sectors on the disk pack.

A problem relating to head-per-track disk has been fixed where the "CHECK" option produced a listing showing no errors. The "CHECK" option will now give a listing showing "phase one errors" like the "SAVE" and "ALPHA" options currently do.

18.2.6. DMPALL.
INCL may now be used with ASCII on an output file.

A problem with DMPALL that produced an exact duplicate file when ASCII to EBCDIC translation was specified has been fixed.

18.2.7. LOGCONVERT.
LOGCONVERT now retains the parent job number when a job is spawned. This record is added to the SCHEDULE Record Format file. The record type field has been changed from a numeric field to an alpha field. Documentation for the LOGCONVERT progarm is available in a printer backup file labelled DOCUMENT/UTILITY.

18.2.8. PACK/INIT.
It is now possible to initial a single cylinder of a pack, without disturbing other data on the pack, providing the entire cylinder is available. This feature is incorporated into the cylinder initialize (CI) option and is enabled when the CYLINDER INITIALIZE option has been specified and the ODT is the input source (option is not allowed with card input). When the CYLINDER INITIALIZE option is specified the PACK/INIT program asks the question, "DO YOU WANT TO PURGE?". If a NO answer is given, only cylinder addresses which are in the available table

are accepted to initialize.

18.2.9. SORT/COLLATE
SORT/COLLATE is deleted with the Mark 9.0 release. The program CREATE/TABLE replaces the SORT/COLLATE ability to create translate tables. The SORT program has the ability to generate collate tables.

18.2.10. SQUASH/USER.DISK.
SQUASH/USER.DISK now displays appropriate messages at the time of the beginning and ending of the job and if the job can not be started due to some programs still running on the designated drive. The following messages were implemented:

1. At the time of beginning of job: "DO NOT DISTURB <unit-mnemonic> UNTIL FURTHER NOTICE".
2. At the end of job: "<unit-mnemonic> LABELLED <pack-id> IS NOW AVAILABLE FOR USE".
3. If pack to be used is busy: "ACTIVITY MUST CEASE ON <unit-mnemonic> FOR DISK SQUASH TO CONTINUE".

18.2.11. SYCOPY
The SYCOPY program is being deleted with the Mark 9.0 release. The SYSTEM/COPY program is now used for library tapes and the need to copy SYSTEM/LOAD.DUMP tapes no longer exists.

18.2.12. SYSTEM/BACKUP.
The ability to print BANNERS (a title page) has been added. BANNERS are printed if:

1. Switch 8 is set to any value from 1 to 15 inclusive.
2. The file is labeled.
3. The "forms" option for the file is not set.

BANNERS includes:

1. The usercode (if present) under which the program was executed.
2. The external file name in the creating program.
3. The actual file name on the disk or tape if different from 2.

Pack-IDs are omitted in cases 2 and 3. Each of the possible three items is separated by a line containing the creating program's name repeated across the page. Usercode multi-file-IDs are omitted from 2 and 3 if the usercode has already been printed.

"PB PRT/= LABELS" now gives the value of the AUTOPRINT (ATP) bit along with the header and record count.

The backup file format has changed with this release. Any user

program which relies on the previous format will need to be modified. Refer to DOCUMENT/UTILITY for details.

18.2.13. SYSTEM/COPY.

The Mark 9.3 release of SYSTEM/COPY includes the new separate COMPARE feature and the new VOLUMEINDEX attribute. These are documented below.

The COMPARE function of SYSTEM/COPY is now usable independently of the COPY function. COMPARE is used to compare previously created disk and/or library tape files. The syntax for the COMPARE command is similar to the COPY command syntax except for the restriction that only one input and one output unit can be specified. SYSTEM/COPY produces a listing of the files compared, flagging any errors. The syntax is listed below:

```
COMPARE-<file-id>-------------->FROM--<volume-id>--TO--<volume-id>;
                  !             !
                  +AS <file-id>+
```

Example:

```
COMPARE = FROM MYTAPE TO MYDISK (KIND=DISK);
```

In the example above, all the files on the tape MYTAPE are compared against the files with the same name on the disk MYDISK.

```
COMPARE FILE1 AS FILE2, XYZ/= FROM TAPE1 TO  TAPE2;
```

In the example above, FILE1 on TAPE1 is compared against FILE2 on TAPE2, and all files with a family name of XYZ on TAPE 1 are compared with those files with the same name on TAPE2.

It is now possible via the VOLUMEINDEX attribute to specify which reel of a multivolume input tape should be used. By default, SYSTEM/COPY starts with Reel #1, and scan all reels until it finds the desired files. The VOLUMEINDEX attribute is used to direct SYSTEM/COPY to start the search with a reel other then Reel #1.

Example:

```
COPY =/= FROM MYTAPE (VOLUMEINDEX=2)
```

The above example directs SYSTEM/COPY to ignore Reel #1 of the tape MYTAPE, and copy only the files on Reel #2 and all subsequent reels.

```
DIR TAPE "MYTAPE" (VOLUMEINDEX=2);
```

The above example shows that VOLUMEINDEX may also be used in a DIR command. In this example SYSTEM/COPY prints the directory contained on Reel #2 of MYTAPE. Each reel's directory lists the

files contained on that reel and all the following reels.

18.2.14. SYSTEM/DISK.INIT.
It is now possible to initialize a single cylinder of a pack, without disturbing other data on the pack, providing the entire cylinder is available. This feature is incorporated into the cylinder initialize (CI) option and is enabled when the CYLINDER INITIALIZE option has been specified and the ODT is the input source (option is not allowed with card input). When the CYLINDER INITIALIZE option is specified the SYSTEM/DISK.INIT program asks the question via the ACCEPT message "DO YOU WANT TO PURGE?". If a NO answer is given, only cylinder addresses which are in the available table are accepted to initialize.

18.2.15. SYSTEM/FILE.INIT.
SYSTEM/FILE.INIT is a new program used with COBOL74. Refer to DOCUMENT/UTILITY for details.

18.2.16. SYSTEM/IS.MAINT.
SYSTEM/IS.MAINT is a new program used with COBOL74. Refer to DOCUMENT/UTILITY for details.

18.2.17. SYSTEM/LOAD.CAS.
SYSTEM/LOAD.CAS is no longer executed via the "LC" input message. The "LC" now enters a comment into the System Log. SYSTEM/LOAD.CAS must be explicitly executed. Refer to DOCUMENT/UTILITY.

18.2.18. SYSTEM/LOAD.DUMP.
Beginning with the Mark 8.0 release SYSTEM/COPY became the vehicle for generating library tapes. SYSTEM/LOAD.DUMP was fully implemented in Mark 8.0 to provide the necessary time for change over of library tapes. In the Mark 9.0 release SYSTEM/LOAD.DUMP is modified to be input only by deleting the "DUMP" and "AD" input messages. In the next software release SYSTEM/LOAD.DUMP will be deleted.

18.2.19. SYSTEM/SPOLOGOUT.
SYSTEM/SPOLOGOUT has been changed to allow selective printing of the ODT output. Refer to DOCUMENT/UTILITY.

19. B500 AND 1400 INTERPRETERS

19.1. Introduction. There have been no changes made to the B500 and 1400 interpreters and to the B500/IEP and 1400/IEP.

20. TABS

20.1. Introduction. The Burroughs Time Analysis and Billing System (TABS) has one enhancement for the Mark 9.0 release. The Mark 8.0 files must be updated prior to running on Mark 9.0.

20.2. Enhancements.

20.2.1.
The print fields in TABS/SUM have been increased to 4 digits. This allows for a figure greater than 999 hours (21-156-700010).

20.3. Temporary Documentation.

20.3.1. "REPORT" and "ORDER" options.
The "REPORT" and "ORDER" options are used as input to the TABS/UPDATE program.

REPORT — This option allows the user to request reports by name. The "REPORT" option may be used in conjunction with switch settings (which also request reports); however, the TABS/EXEC reports cannot be specified with the switch 5 setting if the "REPORT" option requests either of the reports. The "ORDER" option must be used to specify the ordering to be used in the TABS/EXEC reports. The "REPORT" and "ORDER" attribute restrictions apply only to TABS/UPDATE input.

The "REPORT" option must appear on a physically separate record from any other option. Several reports may be requested on a single "REPORT" option record; the names of the reports must be separated by commas or by blanks.

ORDER — The "ORDER" option is used only in conjunction with the "REPORT" option, and only when the "REPORT" option has requested that one or both of the TABS/EXEC reports be printed. The "ORDER" option is used to specify the ordering to be followed in these reports. Valid order types are "TIME", "CHARGE", and "NAME". These specify that the ordering is to be based on elapsed time, charge number, or program name, respectively.

20.3.2. DMS reporting in TABS/EXEC reports.
The "Itemized Execution Report" generated by the TABS/EXEC program now includes information on database and DMS processing. The reported information consists of:

1. logical data base name
2. physical data base name
3. time of open
4. open type
5. unit name
6. device type
7. time of close
8. pack name
9. processor time devoted to this data base (incuded in job's processor time)
10. number of update operations
11. number of non-update operations
12. data base open time

20.3.3. TABS/DMS.
TABS/DMS is a report generation program in the TABS package. This program produces the "Data Base Management Statistics Report". TABS/DMS is executed only when this report is requested. Program switch 7 requests this report. Valid settings for switch 7 are zero (0), one (1), and three (3).

The "Data Base Management Statistics Report" provides statistics on all data base activity during a day. A section is printed for each DMS structure invocation; each section provides the following information.

1. structure name
2. time of invocation
3. physical data base name
4. job number
5. file name
6. hardware unit and device type
7. number of finds
8. number of inserts
9. number of updates
10. number of deletes
11. number of system changes
12. number of exceptions
13. number of reads
14. number of writes

The DMS statistics compiled by the TABS/DMS program are not reflected in the TABS billing reports. DMS information which is assigned to charge numbers and reflected in the billing reports is discussed in the TABS/EXEC "Itemized Execution Report".

20.3.4. Recovery and Restart Capability.

When any program in the TABS package terminates abnormally, TABS must not be re-executed until the integrity of the permanent TAB data files is ensured. Reprocessing even a single day's log information can result in inaccurate overbilling to system users because TABS maintains cumulative monthly statistics in its permanent data files. If the abnormal EOJ occurs during execution of TABS/UPDATE, re-execute the program; data is not corrupted. If the abnormal EOJ occurs during TABS/LOGOUT, then the recovery or restart capability of the TABS package must be used. Refer to the subsection "Discussion of Simple Recovery" for information concerning aborts of other programs in the TABS package.

If the system has been coldstarted since the last execution of the TABS package, the restart-recovery capability must be used to recover any pre-coldstart information either from data files or from unprocessed log files. When a system fails and a coldstart of the system disk pack is necessary, all MCP options, including the "LOG" option, are reset. Logs created after the coldstart are sequenced beginning with the number one. Any logs created before the coldstart but unprocessed by the TABS package are not processed unless a coldstart restart-recovery is requested. These pre-coldstart logs must have been backed up to be made available to the TABS package. Also, the TAB permanent data files must be available. If the coldstart occurred while TABS/LOGOUT was executing, the recovery option is invoked; otherwise, the package is restarted. Setting program switch 8 to 2 requests a coldstart restart or recovery. Duplicate reports may be produced; duplication (in this case) does not corrupt data.

The recovery option of the TABS package resumes processing at the day and time that was being processed when the abnormal EOJ occurred; the restart option re-runs the entire job, possibly reprocessing many days' information. The value of program switch 8 determines whether a simple recovery or a coldstart restart or recovery is performed. A simple (switch 8 = 0; no coldstart) restart must be done manually, and requires that both the log files and uncorrupted copies of the TABs data files be available. A simple recovery (switch 8 = 1; no coldstart) also requires log files and uncorrupted copies of the TABS data files to be available, and resumes processing at the day and time that was being processed when the abnormal EOJ occurred. When a coldstart restart or recovery is requested (switch 8 = 2), the TABS package determines if a recovery or restart is needed, based on whether the last TABS/LOGOUT execution terminated normally (restart) or abnormally (recover).

Switch settings for program switch 8

0 Default; normal run, without restart-recovery or simple restart (manual)
1 Recover after abnormal EOJ; no coldstart
2 Coldstart occurred; restart-recover.

No restart-recovery is required after a clear start unless the TABS package was executing at the time of the clear start, and was aborted.

Simple Restart

A restart re-executes the entire unsuccessful run, reproducing all requested reports. A restart is only valid when TABS/LOGOUT terminates abnormally.

A simple restart after an abnormal EOJ can be accomplished only in the following cases.

A. If the unsuccessful execution of TABS/LOGOUT is the first execution of the TABS package during a month, and the days being processed are the first days of the month, re-execute TABS/UPDATE with the create option (SW 8 = 1) set, and the permanent data files are re-initialized. Ther re-execute TABS/LOGOUT.

   Steps

   1. Remove all TAB files. ("RE TAB/=" or "RE MYPACK/TAB/=")

   2. Execute TABS/UPDATE with the create option set. ("EX TABS/UPDATE SW 8 = 1").

   3. If executing from a user pack, move the TAB files to the user pack.

   4. Execute TABS/LOGOUT ("EX TABS/LOGOUT"). Program switch 8 must be set to zero (0).

B. If the TABS package has been executed with the "SAVE" option on, then a copy of the TAB permanent data files is on the tape "TABSFILES", which was made at the end of the last successful execution of TABS/LOGOUT. After replacing the potentially corrupt TAB files with the backup copies on the tape "TABSFILES", re-execute TABS/LOGOUT.

Steps

1. Remove the TAB files ("RE TAB/=" or "RE MYPACK/TAB/=").

2. Use SYSTEM/COPY tc copy the backup files to disk ("COPY =/= FROM TABSFILES" or "COPY =/= FROM TABSFILES TO MYPACK (KIND=DISK)").

3. Execute TABS/LOGOUT ("EX TABS/LOGOUT"). Program switch 8 must be set to zero (0).

Simple Recovery (Switch 8 = 1)

The recovery option causes TABS/LOGOUT to process only those days not processed during the previous execution of the TABS/LOGOUT (the execution which ended with an abnormal EOJ). Recovery processing begins with the day and time in process at the time of the abnormal EOJ. Some operator intervention may be required.

Steps

1. Execute TABS/LOGOUT, requesting a simple recovery. ("EX TABS/LOGOUT; SW 8 = 1").

2. (Not always required.) Determine the programs to be executed during the recovery pass by answering the questions displayed by TABS/LOGOUT on the ODT.

   Questions are in the following format.

   % TABS/LOGOUT = <job #> DO YOU WANT TO EXECUTE TABS/<program>?
   % TABS/LOGOUT = <job #> ENTER YES OR NO.

   The operator must respond "YES" or "NO" in an accept message to the program. A "YES" response is entered when the reports generated by the program in question have not been produced; a "NO" response is entered if the reports have already been produced. In these situations, inadvertent duplication of reports is harmless, and does not corrupt data files.

Discussion of Simple Recovery

TABS/LOGOUT and TABS/REPORTGEN both process a large amount of data in permanent and temporary data files; therefore, if either fails to complete its processing tasks successfully, the day being processed at the time of the abnormal EOJ must be completely reprocessed. If the two above-named programs successfully complete their processing tasks, a recovery run continues processing the report-generating programs.

There are seven permanent TAB data files, but only two--TAB/CUSTOMERS and TAB/PERIPH-- contain month-to-date statistics. Only four of the nine daily programs-- TABS/LOGOUT, TABS/REPORTGEN, TABS/MIX, and TABS/HDWR-- alter the files and totals to the extent that duplicate execution produces erroneous information. TABS/LOGOUT remains in the mix for the entire execution sequence of the TABS package, but it finishes its daily data alterations before calling the other programs. The execution sequence is as follows.

1. TABS/LOGOUT (data altering)
2. TABS/REPORTGEN (data altering)
3. TABS/ACCTS
4. TABS/MIX (data altering)
5. TABS/SUM
6. TABS/HDWR (data altering)
7. TABS/EXEC
8. TABS/ERR
9. TABS/DMS

Only those programs whose reports are requested are executed; the execution sequence remains the same, with non-requested programs skipped.

When TABS/LOGOUT is executed with program switch 8 = 1, it checks the two permanent data files that contain month-to-date statistics (TAB/CUSTOMERS and TAB/PERIPH) to determine which of the data altering programs have been executed for the day being processed when the abnormal EOJ occurred. In no case should these programs be re-executed during recovery. TABS/LOGOUT only receives explicit information about the status of the data-altering programs; however, it can extrapolate information about the non-data-altering programs by noting the position in the execution sequence of the data-altering programs that have been executed and comparing the position in the sequence of the non-data-altering programs. For instance, if TABS/MIX has completed, then TABS/SUM has also completed; however, if TABS/MIX has not completed, then TABS/LOGOUT displays a message asking if TABS/SUM should be executed.

Recovery of a coldstart recovery-restart run which terminates abnormally is possible, if, in the coldstart recovery, TABS/LOGOUT has completed its log processing tasks. If TABS/REPORTGEN has gone to "Beginning of Task" (BOT), then TABS/LOGOUT has completed its log processing tasks. To recover from this situation, follow the steps outlined for a simple recovery run of TABS/LOGOUT.

Executing the TABS Package After A Coldstart

A coldstart restart-recovery run is required whenever a system has been coldstarted since the last execution of the TABS package if any pre-coldstart information is to be processed. If the last execution of the TABS package terminated normally, the restart option is invoked; if the last execution terminated abnormally, the recovery option is invoked. The TABS restart-recovery mechanism determines which option is to be invoked. Some operator intervention may be required to determine which programs are to be processed during the recovery pass. Refer to the preceding sections "Simple Recovery, Step 2", and "Discussion of Simple Recovery" for additional information.

A coldstart restart-recovery run can process only the information that is not destroyed during the coldstart. Maximum recovery of data occurs if all log files are backed up, and either the "SAVE" option is used on all TABS runs, or the TABS package (including all data files) is maintained on a user pack. If the TAB permanent data files have been destroyed, TABS/UPDATE must be executed with the "CREATE" option set to re-initialize these files.

Coldstart Recovery

To restart the TABS package and to recover all TABS information after a coldstart, pre-coldstart logs and the TAB permanent data files must be available. If the coldstart occurred while the TABS package was executing, causing it to terminate abnormally, and the TABS package is maintained on a user disk, the recovery mechanism resumes processing at the point at which the abnormal EOJ occurred.

In both cases described above (logs and TAB files are backed up, or TABS is maintained on a user disk), all monthly information remains intact. If the permanent data files available were not produced by the most recent run of the TABS package, however, duplicate reports may be produced.

Steps

1. Load all unprocessed pre-coldstart logs onto the system disk (or the user pack from which the TABS package is executed). TABS knows the number of the log with which it is to begin processing.

2. Load the saved TAB permanent data files onto the system disk. If the TABS package is maintained on a user disk, skip this step.

3. Execute TABS/LOGOUT, specifying the coldstart restart-recovery option.

```
EX TABS/LOGOUT; SW 8 = 2;
```

4. Verify that the logs that TABS/LOGOUT expects to process include all pre-coldstart logs. TABS/LOGOUT cisplays the following messages during verification.

```
%TABS/LOGOUT = <job#> CONSECUTIVE LOGS NUMBERED <#a> THRU
   <#b> ON DISK. PLEASE VERIFY THAT THESE ARE THE ONLY LOGS
   TO BE PROCESSED DURING COLDSTART RECOVERY.
%TABS/LOGOUT = <job#> ENTER YES TO VERIFY OR <#> TO OVERRIDE
  IF <#b> IS NOT TO BE THE LAST LOG PROCESSED DURING RECOVERY.
```

The operator should verify with an accept (AX) message. A "YES" response causes TABS/LOGOUT to begin processing the logs named. An override response (<job#>AX <#c>) changes the last log to be processed during recovery to the pre-coldstart log numbered #c. If #c is less than #b, the information in logs with numbers greater than #c is lost. In no case should #c be the number of a post-coldstart log, since only pre-coldstart logs are processed during the coldstart recovery pass.

5. Remove the log files processed during coldstart recovery after TABS displays a message specifying which logs have been processed. Even if the "REMOVE" option is set, TABS never removes pre-coldstart logfiles.

```
%TABS/LOGOUT = <job#> PROCESSING OF LOGS CREATED BEFORE
  COLDSTART COMPLETE. TABS WILL NOT REMOVE THEM FROM DISK.
```

After displaying the above message, TABS/LOGOUT begins processing post-coldstart logs and days.

Coldstart Restart

When either the log files or the TAB permanent data files are unavailable after a coldstart, information that was contained in those files is lost.

A. Permanent Data Files Available, Log Files Not Available

If the TAB permanent data files are available, but the logs were destroyed during coldstart, then the monthly totals accumulated in the permanent data files are valid, although the unprocessed information contained in the lost log files is not reflected. Processing begins with post-coldstart logs.

Steps

1. Execute TABS/BILLING if pre-coldstart billing reports and statistics are desired. The TAB permanent data files must be available.

2. Re-initialize the TAB permanent data files, by executing TABS/UPDATE with the CREATE option set. The TAB files must be re-initialized in order to set the log file pointers that TABS maintains.

```
EX TABS/UPDATE; SW 8 = 1;
```

3. Execute TABS/LOGOUT, without setting program switch 8. The restart-recovery capability of the TABS package is not invoked. TABS/LOGOUT begins processing the logs created after the coldstart.

B. Pre-coldstart Logs Available, TAB Files Not Available

If the pre-coldstart logs are available, but the TAB permanent data files are not, the information stored in the data files is lost. Only the unprocessed information stored in the log files can be recovered. Steps 4 and 5 below request user input to specify where the restart is to begin. Both responses are necessary for TABS to find its starting location, because the TABS pointers were stored in the lost data files.

Steps

1. Re-initialize the TAB permanent data files by running TABS/UPDATE with the "CREATE" (SW 8 = 1) option set.

```
EX TABS/UPDATE; SW 8 = 1;
```

2. Load the pre-coldstart logs to be processed during the coldstart recovery pass to the system disk. Include logs that were processed before the coldstart if recovery of the information contained in these logs is desired. Reprocessing the logs does not corrupt the data files, since the data files are being recreated.

3. Restart the TABS package by executing TABS/LOGOUT with the coldstart restart-recovery option set.

```
EX TABS/LOGOUT; SW 8 = 2;
```

4. Specify the pre-coldstart logs to be processed by specifying the smallest and largest log numbers to be processed during coldstart restart. Missing (unavailable) logs, after the first, do not affect the restart, but do require operator intervention during later processing to skip the logs.

TABS/LOGOUT prompts for the log number range with the following messages.

```
%TABS/LOGOUT = <job#> COLDSTART RESTART REQUESTED.
 PLEASE SPECIFY LOG NUMBERS CREATED BEFORE COLDSTART
 TO BE PROCESSED BY TABS.
%TABS/LOGOUT = <job#> ENTER <#> THRU <#> OR BLANKS
 IF COLDSTART RESTART IS NOT DESIRED.
```

The operator must respond with an accept message. Examples follow.

```
<job#> AX 5 THRU 10

<job#> AX                    %blank input; restart terminates
```

5. Specify the first date to be processed during the restart. Again, TABS prompts for this input. Information pertaining to days prior to the date specified is lost.

```
%TABS/LOGOUT = <job#> PLEASE SPECIFY STARTING DATE TO
                      PROCESS.
%TABS/LOGOUT = <job#> ENTER <MM/DD/YY> OR BLANKS IF ALL
 DAYS IN LOG ARE TO BE PROCESSED.
```

21. SYCOM

21.1. Introduction.

The Systems Communications Module (SYCOM) program provides the B1000 user with:

1. The ability to submit jobs to run on another system.

2. The ability to transfer files between two systems.

3. The ability to interactively obtain services and information from another system with the ability to communicate with interactive programs in a remote system.

Job-run output can be returned to the submitting system (upon request) or directed to a peripheral device on the processing system itself. Interactive output data is returned to the User System directly, via Network Controllers.

The SYCOM program along with its associated network controller is responsible for all data communications with other systems. This includes performing the necessary functions to establish line connections, receive/transmit data, and execute the correct line termination procedures.

The SYCOM program maintains a secondary level of control (MCP has primary control) over local peripheral devices such as the card reader, card punch, line printer, and magnetic tapes. It reads and transfers card, tape, and disk files, accepts input from and displays messages to the Operator Display Terminal (ODT) and receives data files/messages for printing, punching, display, or writing to either tape or disk. It provides the mechanism by which a terminal or application program in one system is able to communicate with specific application programs in another system. Program selection is based on the servicing system which dictates the processing requirements (programs) optioned to run in that system.

The SYCOM system is a program product which may be separately ordered. Documentation on the SYCOM program is available with the program product in the form of a printer backup file labeled DOCUMENT/SYCOM. The B1000 SYCOM program can be ordered through the local Burroughs office.

The Mark 9.0 release of the SYCOM program includes enhancements and corrections to reported problems. New features include the enhancements to the "READ", "ABORT", "COPY", "FETCH", "PUT", and "RESTART" commands. The new "BUFFER", "QS" and "SIGNAL" commands and the new "TRACEAREAS", "TRACEBA", "TRACERB", and "QUEUESIZE" parameters have been implemented. Also, the format of the SYCOM program's parameter file has been changed to make creation and updating of it much easier. These enhancements are documented in

the SYSTEMS COMMUNICATION MODULE (SYCOM) documentation. Corrections to outstanding problems are documented in the following paragraphs.

21.2. Enhancements.

21.2.1.
If a terminal is not connected, input messages to the SYCOM program no longer have to be prefixed with the period "." default pseudo-signal character.

21.2.2.
The "LOG" command generates the following information in addition to the hardware errors and software counters previously displayed:

1. The average number of characters per transmission.

2. The number of files that have been sent and received.

The "LOG CLEAR" command initializes to zero all counters which are displayed in response to a "LOG" command.

21.2.3.
The SYCOM program now handles all file separators correctly. Also, the following messages are now displayed on the Operator Display Terminal (ODT) when a print file is received from a host site:

PRINTER OPEN (When the host sends the beginning-of-file separator)
PRINTER CLOSED (When the host sends the end-of-file separator)

21.2.4.
The SYCOM program no longer sends a trailing record separator <RS> character in the transmission buffer, thereby reducing line activity for file transfers and card image transmissions. In prior releases of the SYCOM program, the record separator <RS> and end of text <ETX> characters were sent at the end of a card transmission buffer.

21.2.5.
All output messages which reference an internal file name in the SYCOM program have been changed to reflect their external file names.

21.2.6.
The SYCOM program now displays "PUT ABORTED" or "FETCH ABORTED" when a "PUT" or a "FETCH" command was aborted, respectively. In prior releases of the SYCOM program, only the message "PUT ABORTED" was displayed.

21.2.7.
The SYCOM program no longer halts with "FATAL ERROR 11490680" when a "CONNECT" is initiated and the program is "SIGNED OFF" the SMCS program before the connect is complete.

21.2.8.
When the number of consecutive timeouts received by the SYCOM program exceeds the value contained in the "TIMEOUTS" parameter, the SYCOM program displays "OFFLINE" in response to the "WHAT" command instead of "ONLINE, SENDING" or "ONLINE, RECEIVING".

An explicit "ONLINE" or "RESTART" command is required to re-establish communication, unless the remote SYCOM program has just gone to beginning-of-job or the remote operator entered either an "ONLINE" or "RESTART" command to the SYCOM program.

21.2.9.
A problem has been corrected in the SYCOM program which caused the message "CONNECT DENIED, PROBABLY NO SERVER PROGRAM" to be displayed in response to a "CONNECT" command when a server program did exist.

21.2.10.
The "LIST SCHEDULE" command displays all the files which are queued to be transferred.

21.2.11.
The SYCOM program no longer halts with "FATAL ERROR 34400000".

21.2.12.
The number of fetches is now displayed in response to the "WHAT" command.

21.2.13.
The SYCOM program now responds to a "CONNECT" command with the following connect reply:

CONNECT INITIATED [<virtual terminal number>] <station receive and transmit addresses>

CONNECT COMPLETED

21.2.14.
The SYCOM program now uses "PROTOCOL TYPE 2" wher running in conjunction with the SMCS program. If a user logs onto the SMCS program with a <usercode>/<password> and then enters a "PASS SYCOM <text>" to the SMCS program, the usercode/password pair is automatically appended to each file name specified in a "PUT" or "FETCH" command.

For example, if a user logs onto the SMCS program with the following command:

US USER/JOHN

then a "PASS SYCOM PUT A" or "PASS SYCOM FETCH A" is actually a put or fetch of file "(USER)/A", respectively. Also, if the default pack-id for "USER/JOHN" is "USERPACK", then a "PUT" or "FETCH" of file "A" is PUT or FETCH of file "USERPACK/(USER)/A".

21.2.15.
The SYCOM program now notifies all users "signed on" to the SYCOM program when the remote SYCOM program is terminating with the following message:

    REMOTE <sycom program name> TERMINATED

21.2.16.
Entering the "STOP" command now notifies all users signed onto the SYCOM program that the SYCOM program is stopping with the following message:

    "STOP" IN PROGRESS.

21.2.17.
All SYCOM program messages to a terminal now clear to the end of each message line. This makes reading the line easier.

21.2.18.
The use of special characters in passwords is now allowed, except for the comma "," and right parenthesis ")" characters.

21.2.19.
The "US" command now removes the current usercode/password pair without detaching the station. The station is disconnected if it was previously connected.

21.2.20.
For better synchronization of the SYCOM program, filenames are included in SYCOM's control messages.

21.2.21.
The SYCOM program now checks for the presence of a file when the "PUT" or "FETCH" command is entered, rather than waiting until the put or fetch request comes to the top of the put schedule. This helps to insure correct spelling of the filename.

21.2.22.
The SYCOM program now displays the reason for a denial of a "CONNECT". If the message "NDL ADDRESS ERROR" is displayed, there is no receiving address in the remote network controller to handle the connection. That is, no matching NDL address was found for the user-station's transmit address at the local system in the list of server-station receiving addresses at the remote system. If the message "STATION NOT ENABLED, PROBABLY NO SERVER PROGRAM" is displayed, there was no SMCS program in the remote system to handle the connect.

21.2.23.
The "VALUE STACK" size for the SYCOM program has beer reduced to 80,000 bits from 140,000 bits.

21.2.24.
If the SYCOM program has been executed or modified such that the multi-file-id for the PARAMETERS file is specified, the SYCOM program uses that multi-file-id for all of the workfiles of the SYCOM program. This allows users to run multiple copies of the SYCOM program with the same code file name. In doing so, SYCOM users can take advantage of the MCP re-entrant code capability which allows programs with the same code file name tc share the same code segments in memory.

The default multi-file-id for the "PARAMETERS" file is "<sycom>/PARAMETERS".

The following are some examples of changing the multi-file-id at execution time using the file equating MCP capability:

```
EXECUTE SYCOM FILE PARAMETERS NAME SYCOM.1/PARAMETERS;
EXECUTE SYCOM FILE PARAMETERS NAME (USER)/PARAMETERS;
EXECUTE SYCOM FILE PARAMETERS NAME SYCOM.2/PARAMETERS;
```

The following is an example of modifying the SYCOM program's "PARAMETERS" file:

```
MODIFY SYCOM FI PARAMETERS NAME (USER)/PARAMETERS;
```

21.2.25.
The quote (") characters are now included around the multi-file-id portion of a filename when a remove or change command is zipped to the MCP. This prevents a problem where the MCP commands "RE <multi-file-id>/OLDTRACE" and "CH <multi-file-id>/TRACE <multi-file-id>/OLDTRACE" would get the MCP error message "ZIPPED AN INVALID CONTROL CARD" immediately following beginning of job of the SYCOM program. This problem occurs if the name of the SYCOM program or the multi-file-id portion of the SYCOM parameters file is changed to contain special characters or lower case letters.

21.2.26.
Messages to a remote terminal are no longer truncated if the line overflows and the scroll option is set for the terminal.

21.2.27.
The message "MESSAGE SENT TO REMOTE SPO" is no longer displayed on the system ODT when the SYCOM program is running in RJE mode and logged onto a host system. This message is only valid when not in RJE mode.

21.2.28.
Printer and punch files are now actually opened when the "PRINTER OPEN - <printer filename>" or "PUNCH OPEN - <punch filename>" message is displayed on the system's ODT. Also, the number of lines printed for printer files and cards punched for punch files is displayed when the file is closed.

21.2.29.
The message "INVALID CARRIAGE CONTROL CHARACTER = @<character>@" is displayed on the system's ODT by the SYCOM program when an incorrect carriage control character is received. The SYCOM program replaces the incorrect character with a blank character and continues processing. A blank carriage control character is ignored by the SYCOM program.

21.2.30.
The "PROGRAM POINTER STACK SIZE" in the SYCOM program has been increased from 25 to 50 entries.

21.2.31.
The SYCOM program now updates the following fields in the disk file header when transferring a file.

```
VERSION_DATE
VERSION_TIME
PROTECTION
PROTECTION_IO
CREATE_TIME
```

Updating these fields makes transferring DMS files possible.

22. DC/AUDII

22.1. Introduction.

The Mark 9.0 release of DC/AUDIT contains no new changes.

22.3. Temporary Documentation.

22.3.1.

When logging a WR.APOL (write autopoll) operation, a slash (/) is included in the autopoll string. The slash indicates where the autopoll operation completed. The address is immediately to the left of slash.

If no slash is printed the operation completed on the last address printed.

22.3.2.

The address option is now set by default.

23. RD

23.1. Introduction.

The Remote Display (RD) program provides a method for an operator of a remote display terminal to obtain information about the system environment. This information includes file and program attributes and the current system status. Many of the inquiry commands in the RD program provide the same (and in some cases more) information as do their corresponding MCP commands. The Mark 9.0 release of the B1000 RD program includes enhancements and corrections to outstanding problems reported in the Mark 8.0 release of the RD program. These enhancements include 1) the implementing the "CP" (compute) command, 2) the ability to use a partial-file-id for the "PD", "KA", "DF", and "DF" commands, 3) implementing Program Switch 3 which assigns the maximum number of jobs that can be displayed in the program-table, 4) implementing the "SUMMARY" option to the "KA" command which causes the RD program to display statistics about a family of files, and 5) the RD program now uses "PROTOCOL TYPE 2" as defined by the SMCS. Complete documentation, including these enhancements, is available in the form of a printer backup file labeled DOCUMENT/RD. The corrections to outstanding problems are documented in the following paragraphs.

23.2. Enhancements.

23.2.1.
The "MIX" command can now be abbreviated as "MX".

The "R" command can no longer be used.

The displayed mix table summary has two new columns. They are "M" (media) and "OVLY". The "M" column displays a "1" if the job is in memory and a "0" if the job is "rolled out". The "OVLY" column displays the total number of code and data overlays for the job.

23.2.2.
The "TEACH" command can now be abbreviated as "TE" and "TEA". Entering "TEACH SWITCHES or TEACH SW displays RD's current switch settings and the meaning of each switch.

23.2.3.
The RD program's "RMT" remote file now uses "PROTOCOL TYPE 2" as defined by SMCS. This provides the RD program with the usercode, password, usercode index, and station signal character for a user at a station when the user passes input to the RD program or the user is logged on to the RD program with a usercode/password.

Any "PASS" input through the SMCS to the RD program causes the RD program to apply the usercode/password to file names unless the asterisk "*" character convention is used to override the usercode. This means that if a user is "logged on" to the SMCS, a "PASS RD" of any directory-search command, such as "PD", "KA", "BF", or "DF" for a single-name file that does not have the user's usercode as its multi-file-id must be preceded with an asterisk "*" character. For example, entering "PASS RD PD MCPII" causes the RD program to search the disk directory for a file labelled "(usercode)/MCPII" and entering "PASS RD PD *MCPII" causes the RD program to search the system disk directory for "MCPII". If the usercode has a default user pack-id, the RD program searches that disk for the file name.

Print files created by the "PRT" command have the usercode name applied to the print file name. This can be overridden by using the "NAME" option in the "PRT" command. The following examples illustrate the use of the "PRT" command.

```
PRT NAME A/B   (If "logged on" to the SMCS and
                the usercode is not privileged,
                then a security error results.)

PRT NAME *A    (Print file name is "A".)

PRT NAME A     (Print file name is "A" if not
                "logged on" to the SMCS. Print
                file name is "(usercode)/A" if
                "logged on" to the SMCS.)
```

The "SIGNAL" or "SIG" command and the use of program switches 4 and 5 for initialization of a network-wide signal character have been replaced by the use of the passing station's current signal character as provided by the SMCS. The SMCS supplied signal character is used by the RD program in the gereration of successive "PASS" input continuation messages.

23.2.4.

The "FN" command now lists the relative file number beginning at zero for all the internal file names in a program. Also, "FN" and "PQ" can be entered from the Operator Display Terminal (ODT) as ACCEPT input while the RD program is running in data communication mode (SWITCH 0 = 0).

Examples:

```
<RD program number>AXFN DMPALL
<RD program number>AXPQ
```

23.2.5.
A "PASS" of "BYE" or "BRK" stops the "OMX" (Online Mix) output. Previously, a "PASS RD OMX BYE" was required. The "OMX" part is now optional. The following example illustrates the new syntax:

```
PASS RD OMX PAGE 2              (Initiates the online mix)
  .
  . (later)
  .
PASS RD BRK or PASS RD BYE      (Terminates the online mix)
```

23.2.6.
All dates are now displayed in "MM/DD/YY" format, where "MM" is the month, "DD" is the day, and "YY" is the year.

23.2.7.
The RD program now includes the "UPDATE" column in the response to the "KA" command. The "UPDATE" column displays the last date that a file was updated as maintained by the MCP. Also, the user-count column has been abbreviated to "US" from "USR".

23.2.8.
The "BF LABELS" command handles both the Mark 8.0 and 9.0 backup file formats. Also, for dumpfiles (BF *DMP/= LABELS), the program name that is associated with each dumpfile is listed.

23.2.9.
The RD program uses a "DETACH-WITH-CLEAR" on all station detaches except those caused by entry of "OMX" while "signed on" to the RD program.

23.2.10.
The "US" command to log on and log off the RD program now replies with the usercode and program name. The following example illustrates the "US" command and its response:

```
Command:   US USER/PASS
Response:  (USER) LOGGED ON TO RD
```

23.2.11.
The "P" column (PROTECTED) in the mix table is now updated correctly.

23.2.12.
The "CLS" command with the RD program in batch mode (SWITCH 0 = 1 or 2) causes the RD program to close its printer file labelled "LINE".

23.2.13.
The problem in the RD program which caused the erroreous message "MESSAGE GTR BUFFERSIZE" to be displayed has been corrected.

23.2.14.
The RD program no longer halts with "READ OUT OF BOUNDS" when using the "PQ" command.

23.2.15.
The RD program now provides the asterisk "*" character in the file name for the "FN" command.

23.2.16.
The RD program no longer writes type 15 messages (DETACH-REPLY) to the network controller.

23.2.17.
The <disk address> and <number of sectors> displayed in response to the "KC" and "KP" commands are now also displayed when a user is "signed on" to the RD program.

23.2.18.
File names can now be specified with an asterisk in the multi-file-id even when the multi-file-id already has ten characters.

23.2.19.
The "FN", "QF", "KC", and "KP" commands now respond with "<complete file name> NOT ON DISK" if the file is not found.

24. DMS/INQUIRY

24.1. Introduction.

The Mark 9.0 release of the DMS/INQUIRY program contains enhancements to the OPTIONS and PRINTER statements, along with changes designed to increase throughput and performance.

The DMS/INQUIRY control files (<database name>/INQCTL) are compatible between the Mark 8.0 release and Mark 9.0 release.

24.2. Enhancements.

24.2.1.

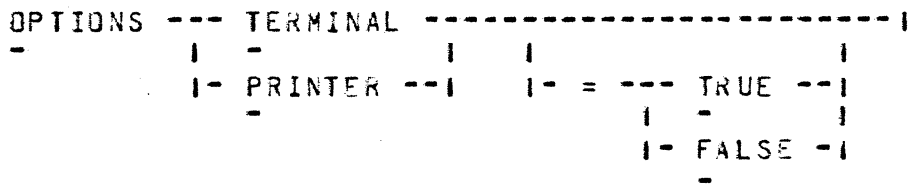
The DMS/INQUIRY program printer file can now be closed by the user. Specifying PRINTER RELEASE (abbreviated P R) causes the print file to be closed, if it is open. If the file is a backup print file, the backup print file is released.

Examples:

```
PRINTER RELEASE
P R
```

24.2.2.

The DMS/INQUIRY output can now be directed to the terminal and printer simultaneously. Because of this feature, the OPTIONS TERMINAL and OPTIONS PRINTER statements have been modified, as follows:



Specifying TRUE causes output to be directed to the device, specifying FALSE suppresses the output to the device. If TRUE or FALSE is not specified, TRUE is assumed. Specifying FALSE on one device causes the DMS/INQUIRY program to set the other device to TRUE, if not already TRUE.

Examples:

```
OPTIONS TERMINAL = TRUE
OPTIONS PRINTER = FALSE
O P = F
```

When the output is to both devices simultaneously, the terminal attributes (page width, format) override the printer attributes for the printer.

25. SORT

25.1. Introduction.

The Mark 9.0 release of the SORT programs consist of SORT, the SORT intrinsic programs (SORT/VSORT, SORT/GSORT, SORT/TAPESORT, and SORT/MERGE), and the SORT/UTILITY program.

With the SORT/UTILITY program, records in the input file can be selectively processed, including or excluding records from sort processing depending on specified conditions.

The SORT/UTILITY program can produce output records whose format and content differs from that of the input records. Fields in a record can be selectively rearranged or excluded. Literals can be inserted into chosen positions in the output records. Selected fields can be summed to provide user processing during the sort run.

Multiple record formats can be specified within a single output file. User specified criteria determine the arrargement and selection of fields in a record, and there can be several sets of criteria for each output file.

The SORT/UTILITY program is functionally compatible with the IBM $DSORT program. Specifications may be given in free form format, as in the SORT program, or in the fixed field format of $DSORT.

Temporary documentation describing the operation and functions of SORT/UTILITY and SORT is included in a printer backup file named DOCUMENT/SORT supplied as a supplement to these System Notes.

25.2. Enhancements.

25.2.1.
When an input record is declared as DEFAULT, SORT will not check for out of bounds keys. The MCP SORT INTERFACE will handle this task.

26. FORTRAN 77

26.1. Introduction.

Included in the Mark 9.0 release of the B1000 System Software is the initial release of the B1000 FORTRAN 77 compiler and interpreter. The new FORTRAN system will run on all B1000 computers except the B1825, B1830 and B1710 series of computers since these machines do not have cache or control memory.

The FORTRAN 77 software compiles and executes programs written in American National Standard Programming Language FORTRAN, ANSI X3.9-1978. This new definition of FORTRAN, known as FORTRAN 77, is not a rewrite of the previous FORTRAN, FORTRAN 66, but is an entirely new FORTRAN system. Although this is the case, the syntax of FORTRAN 77 is still somewhat similar to FORTRAN 66. A brief description of the major new features of FORTRAN 77 follows.

1. A data type of type CHARACTER is implemented as well as several string manipulation capabilities, such as concatenation, substring extraction and assignment, and string scanning.

2. More flexible control structures are available. Included are IF statements that give the option of having more than one executable statement after the relational expression, structured DO-loops, and multiple entry points in a subroutine.

3. Do-loops now allow increments of either positive or negative values. DO-loops also detect jumps into the range of the loop and flag them as an error.

4. Additional file attributes can be specified.

5. Conversion of data from one type to another can now be easily accomplished through the use of internal files.

The FORTRAN 77 software is a program product which may be ordered through the local Burroughs office. Documentation for FORTRAN 77 is available with the FORTRAN 77 software in the form of a printer backup file labeled DOCUMENT/FORTRAN77.

26.3. Temporary Documentation.

The following statements are implemented but the printer backup file labeled DOCUMENT/FORTRAN77 does not reflect this fact.

26.3.1.

DATE is an intrinsic function which returns the current date in any one of four specifiable formats. The form of the DATE intrinsic follows.

```
DATE (format)
```

Format may be any of the following:

```
"MMDDYY"
"YYMMDD"
"YYDDD"
"HHMMSS"        (time of day)
```

26.3.2.

TIME is an intrinsic function which returns the time function specified by two parameters. The form of the TIME intrinsic follows.

```
TIME (function,units)
```

Function may be any of the following:

```
"DAY"           (time of day)
"ELAPSED"       (elapsed time of the program)
"PROCESSOR"     (total processor time for the program)
"I/O"           (total I/O time for the program)
```

Units is a unit-of-time specification which specifies the units in which the time is to be printed. For example, .1 specifies that the time is to be printed in tenths of seconds. 1. specifies that the time is to be printed in seconds.

26.3.3.

The CALL ZIP statement zips a string of characters to the MCP. The syntax of the CALL ZIP statement is as follows:

```
CALL ZIP(string)
```

String is any control string valid to the MCP.
Example:

```
CALL ZIP("CO PROG FORTRAN77;FI CARDS NAM PROG1 DSK DEF;")
```

26.3.4.
The CALL DUMP statement causes a dumpfile to be produced. After the dumpfile has been produced, execution continues with the statement following the CALL DUMP statement. Proper form:

```
CALL DUMP
```

26.3.5.
The CALL EXIT statement causes termination of the program as if a STOP statement were executed. Proper form:

```
CALL EXIT
```

26.3.6.
The CALL OVERFL statement enables programmatic recovery when an exponent overflow or underflow has occurred. The form of the CALL OVERFL statement follows.

```
CALL OVERFL(I)
```

Execution of the CALL OVERFL statement sets I to a 1 if an exponent overflow has occurred in any variable since the last CALL OVERFL statement. I is set to a 2 if neither an exponent overflow nor underflow has occurred since the last CALL OVERFL statement. I is set to 3 if an exponent underflow has occurred in any variable since the last CALL OVERFL statement.

26.3.7.
FORTRAN77/ANALYZER is a program which analyzes either a FORTRAN 77 code file, a FORTRAN 77 intermediate code file, or both.

A code file is generated from the compile step and is used for the execution of the program. It contains information concerning the program parameter block, file parameter block, code segment dictionary, data page dictionary, data pages, laycut table, and the code to be executed.

An intermediate code file is generated by the compiler control card SAVEICM. This file consists of a directory and one or more intermediate code modules (ICM). The directory records the number of ICMs, their names, and their locations in the file.

An ICM consists of a header describing information pertinent to the module, as well as the address and size information of the following parts of the module:

- Data Page Dictionary (if any)
- File Parameter Block (if any)
- Code (subroutine, function, or main program)
- Reference Table (locations within the code where subroutines, functions, and common blocks are referenced)
- Dummy Argument Information Table (if any)
- Initialized Data Pages

The following entities generate an intermediate code module (ICM): the main program, a subroutine, a function, an initialized common block, and one or more FILE declarations. To execute the program and have the output file displayed on a remote terminal, use the following statement:

```
EX FORTRAN77/ANALYZER;FILE LINE REMOTE
```

To execute the program so that the output file is printed, use the following statement:

```
EX FORTRAN77/ANALYZER;SW 1=1;
```

The user provides input options via the DISPLAY/ACCEPT facility. In response to the command prompt: ENTER FILE NAME, the user enters the following statement:

```
<job#>AX <code file name or ICM file name>
```

The program verifies the file name and either displays an error message, or responds with the command prompt ENTER OPTIONS, NULL TO STOP, OR HELP. If the user enters HELP, the program requests the type of file (code or ICM) and displays a list of appropriate options according to the file type the user has indicated. These options determine the part of the code file or ICM to be analyzed. The list of options displayed for a code file follows.

PPB	Program Parameter Block
FPB	File Parameter Block
SD	Code Segment Dictionary
DD	Data Page Dictionary
DP [page number]	Data Page (one or all, default is all)
LT	Layout table
CODE [number, name or ALL]	
ALL	All of the above
GET file name	Get another file

For an ICM file, the list of options displayed follows.

ALL	Directory and all ICMs
DIRECTORY	Directory listing
an ICM name	ICM name to be analyzed
HEADER	ICM Header
DD	ICM Data Page Dictionary
FPB	ICM File Parameter Block
CODE	ICM Code
REF	ICM Reference Table
DAI	ICM Dummy Argument Information
DP	ICM Data Pages
GET file name	Get another file

Options are entered individually or serially. If entered

serially, they are separated by commas and/or blanks.

If an ICM file is specified with no other qualifying options, the default is an analysis of the entire ICM.

To terminate the program, a null entry is transmitted.

Program switch 1 determines whether or not page skipping is used in the printer listing. When set to zero, page skipping is employed; when set to one, it is suppressed.

The following error messages may occur.

"<segment name> : INVALID SEGMENT NAME. TRY AGAIN."

"<option name> : INVALID OPTION. TRY AGAIN."

"<page number> : PAGE NUMBER OUT OF BOUND. TRY AGAIN."

"<segment number> : SEGMENT OUT OF BOUND. TRY AGAIN."

"<token entered> : EITHER INVALID OPTION
OR INVALID ICM NAME
OR MORE THAN ONE ICM SPECIFIED."

"FILE MISSING : <file name entered>"

"INVALID FILE NAME"

"FORTRAN77 ICM OR CODE FILE REQUIRED"

26.3.8.

Many users will need to make a determination concerning conversion from 1966 FORTRAN to FORTRAN 77. To help in this determination, this section describes some of the problems that may be encountered in such a conversion.

1. String data can only be assigned to variables of type CHARACTER, not to numeric types. If a program uses the 1966 FORTRAN character manipulation facilities extensively, a total rewrite will probably be in order. All numeric items, which contain character data, will need to be changed to type CHARACTER. GETCH and PUTCH no longer work so these routines will have to be replaced.

2. CHARACTER and numeric data cannot reside in the same COMMON block nor can they be equivalenced.

3. Length specifier *<integer> works only for type CHARACTER (e.g. REAL *8 no longer works).

4. Maximum record length is reduced from 2047 to 1023.

5. Word format has changed so any variables initialized in a DATA statement with Z need to be changed. In many cases this change will only be the removal of the first hex character (4 bits) but other changes may recuire shifting of the data. The documentation to be released with FORTRAN 77 tells of the new word formats.

6. The DATA statement must come after all specification statements. 1966 FORTRAN allowed flexible placement of the DATA statement.

7. In 1966 FORTRAN a DO loop is always executed once. To produce the same result in FORTRAN 77, DO loops should be changed in the following manner:

   DO label v=e1, e2, e3 ====> DO label v=e1, MAX(e2,e1), e3

8. A jump into the body of a DO loop is a FORTRAN 77 compile-time error.

9. The DO iteration count cannot be changed within the loop. An attempt to change the DO variable is syntaxed at compile-time. Attempts to change other variables that the DO loop may depend on are allowed.

10. For random I/O operations record number syrtax changes from <unit-no>=<rec-no> to <unit-no>'<rec-no>, i.e., the equals sign must now be an apostrophe. (1966 FORTRAN allowed either the "=" or the "'").

11. Free-format I/O, as implemented in the Mark 7.0 release of 1966 FORTRAN cannot be used in FORTRAN 77.

12. FORTRAN 77 requires a comma between specifiers in a FORMAT statement. 1966 FORTRAN allowed comma omission.

13. H strings are only valid in FORMAT statements. To convert to FORTRAN 77, delete the H ard the integer preceding it, if any, and surround the string with quotation marks or apostrophes.

14. The capabilities available through use of the DATA and ERR action specifiers have been combined into one specifier in FORTRAN 77: the ERR specifier. Use of DATA will cause a compile-time error.

15. The NAMELIST statement will not be in the Mark 9.0 release. Although NAMELIST will compile without a syntax error, it will not execute.

16. The FILE statement has been changed extensively. Refer to the FORTRAN 77 documentation for the exact syntax.

17. Edit descriptor usage is more rigid in FORTRAN 77 than in 1966 FORTRAN. For example, INTEGER type data can no longer be written with an F edit descriptor. Refer to the FORTRAN 77 documentation for the exact specifications.

18. Carriage control character "-" (triple space) is not in FORTRAN 77.

19. The CHANGE statement is not in FORTRAN 77.

20. The LOCK statement has not been implemented but the FORTRAN 77 CLOSE statement has a KEEP attribute available to the user.

21. The PURGE statement is not in FORTRAN 77.

22. CALL SSWTCH is not in FORTRAN 77.

23. All program units that are compiled with the $SAVEICM compiler control option are saved under one name: .SBRTN. In order to save ICMs under a different name, a file name should be used with $SAVEICM (e.g. $SAVEICM "MYICM"). If a file name is not used, each subsequent compile will remove the file .SBRTN created from a previous compile.

24. The following compiler control images were not implemented:

    BCD
    CODE
    ERRORTRACE
    INITIAL
    LIBRARY
    LIBRARYPACK
    NEWINTRINSIC
    PROFILES
    SINGLE
    TRACEF
    XREF

25. The following compiler control images were implemented with changes:

CONTROL	(now LIST$)
INTERPPACK	(now INTERPRETER; entire name can be specified)
INTRINPACK	(now INTRINSICS; entire name can be specified)
RANGE	("RANGE" deleted)
SEQERR	(now SEQCHECK)

More complete information concerning compiler control images as implemented in FORTRAN 77 is available in the FORTRAN 77 documentation.

26. The following intrinsic functions were not implemented.

ALGAMA
COTAN
DCOTAN
DERF
DERFC
DEFLOAT
DGAMMA
DLGAMA
ERF
ERFC
GAMMA
HFIX
LGAMMA

27. The following intrinsic functions were implemented under a different name.

Function	New Name
ARCOS	ACOS
ARSIN	ASIN
DARCOS	DACOS
DARSIN	DASIN
LOG	ALOG
LOG10	ALOG10
MAX	MAXO
MIN	MINO

28. SPO is now ODT.

29. Asterisk (*) in an alternate return is now an ampersand (&).

30. Columns 1-5 of a continuation card must contain blanks.

31. Arrays used in EQUIVALENCE statements must have the same number of dimensions as were declared.

32. A simple I/O list is prohibited from being enclosed in parentheses.

33. Writing to a tape file after an endfile record has been written is prohibited.

34. The TIME intrinsic has changed. Refer to Temporary Documentation 26.3.2. for the correct syntax.

A suggested method of conversion is to compile a small 1966

FORTRAN program under FORTRAN 77 to become familiar with the compile-time errors and to then convert larger programs as the user sees fit.

27. COBOL74

27.1. Introduction.

Included in the Mark 9.0 release of the B1000 System Software is the initial release of the COBOL74 compiler and interpreter. COBOL74 will run on all B1000 computers except the B 1825, B1830 and B1710 series of computers since these machines do not have cache or control memory.

The COBOL74 software compiles and executes programs written in COBOL in accordance with the American National Standard Institute (ANSI), X3.23-1974.

New features included in COBOL74 are as follows.

1. New file organizations. There are now 3 file organizations: Sequential, Relative, and Indexed.

2. File Attributes. File Attributes provide the capability for defining, monitoring, or changing properties or attributes of a file.

3. Indexed files are now managed by the MCP, provide greatly improved performance over COBOL68 indexec files, and function in a manner similar to the DMSII Index Sequential Structures. Records may now be ADDed or DELETEd without invoking a sort intrinsic. Duplicates may optionally be allowed for alternate keys.

4. Linage clause for Sequential files. The LINAGE clause provides a means for specifyirg the pagesize, margins, and footing of a logical page.

5. Interprogram communication is now possible through the use of the "CALL" verb. A program may "CALL" another, pass parameters and share data.

6. The "INSPECT" verb provides the capability of scanning and replacing text, counting occurrences of a character or group of characters, and specifying the portion of the text on which to perform the function.

7. The "STRING" verb. Partial or complete contents of 2 or more data items can be concatenated into 1 by the use of the "STRING" verb.

8. The "UNSTRING" verb. Taking contiguous data in 1 data item and moving it to multiple receiving fields is also possible by the use of the "UNSTRING" verb.

9. Standard Debugging facilities. Users can now specify the conditions under which data items or procedures are to be

monitored during the execution of the program.

10. Standard Data Communications. Through the use of the "SEND" and "RECEIVE" verbs information can be obtained as to station name, length of message, how many stations to be addressed, and error information.

11. DMSII is available in COBOL74.

12. "$XREF" is now part of the compiler and will generate a program cross reference at compile time without requiring additional operator intervention.

13. Use of the option "$OMIT" provides the capability of conditional inclusion or exclusion of source images.

14. Dump Analyzer for COBOL74 prints the variable name in addition to the value of the data item. There is also improved error detection.

15. Table Handling has improved with individual bounds checking on subscripts and the capability of variable length tables.

Two new utilities have been introduced in this Mark 9.0 release for COBOL74. They are SYSTEM/IS.MAINT for index files and SYSTEM/FILE.INIT for relative files. Documentation for SYSTEM/IS.MAINT and SYSTEM/FILE.INIT is available in the form of a printer backup file labeled DOCUMENT/UTILITY.

The COBOL74 compiler is a program product which may be ordered through the local Burroughs office. Documentation for the COBOL74 compiler is available with the COBOL74 software in the form of a printer backup file labeled DOCUMENT/COBOL74.

27.3. Temporary Documentation.

27.3.1. Indexed File Naming Convention.

For an Index File, if a file is defined as "FD <file name> ..." and no "VALUE OF TITLE ..." clause is specified, then the external file names of the associated files will be as follows:

```
Cluster file   =  "<file name>"
Data file      =  "00<file name>"
All key files  =  "01<file name>" through "99<file name>"
```

If there is an external file name defined thrcugh the "VALUE OF TITLE" is "<family-id>/<file-id> on <disk pack-id>" clause and <file-id> is not blank then the external file ids are:

```
Cluster file   =  "<dp-id>/<family-id>/<file-id>"
Data file      =  "<dp-id>/<family-id>/00<file-id>"
All key files  =  "<dp-id>/<family-id>/01<file-id>" through
```

"<dp-id>/<family-id>/99<file-id>"

If <file-id> is blank then the naming convention applies to the <family-id>.

Cluster file	=	"<dp-id>/<family-id>/"
Data file	=	"<dp-id>/00<family-id>/"
All key files	=	"<dp-id>/01<family-id>/" through "<dp-id>/99<family-id>/"

The MCP disallows file equation of Indexed Files. To change the name of an Indexed File refer to SYSTEM/IS.MAINT in the Utilities Section of the System Notes.

28. IBASIC

28.1. Introduction.

Interactive BASIC (IBASIC) will be available in the Mark 9.0 release, although not in the initial release. IBASIC will run on all B1000 computers except the B1825, B1830 and B1710 series of computers since these machines are norcache, noncontrol memory machines.

BASIC, an acronym for Beginners All-purpose Symbolic Irstruction Code, was initially developed at Dartmouth College. The American National Standards Institute (ANSI) has developed a standard for BASIC using the original Dartmouth BASIC plus additional features as dictated by the industry. Burroughs has implemented the BASIC language according to the American National Standard for BASIC developed by ANSI.

BASIC is a problem-oriented language designed for a wide range of applications and may be easily applied to both educational and engineering/scientific processing tasks. The BASIC language is designed for use both by individuals who have little previous knowledge of computers as well as individuals with considerable programming experience. A distinct advantage of BASIC is that its rules of form and grammar are easily learned. Burroughs IBASIC is especially suited for the learning process since response to errors is nearly immediate for many of the errors common to novice programmers.

Burroughs IBASIC is implemented in a conversational mode: the user enters BASIC statements and interactive commands through a terminal to the IBASIC system whereupon IBASIC processes the input and responds with either the output for the command, a message informing the user of any syntax errors, or a request for more input.

The IBASIC program product is scheduled for release approximately one quarter after the initial release of the Mark 9.0 system software. The documentation will be available with the release of the IBASIC program product.

29. Mark 9.0 Software Patches

MCPII Patch #1

Insures that the file name of a ANSI 74 COBOL ISAM file can not be changed prior to execution.

MCPII Patch #2

Corrects a problem which caused the MCP tc halt or write corrupted data when writing Delayed Random File buffers.

MCPII #3

The ANSI 74 COBOL file attribute "FLEXIBLE" has been disabled.

DMS/REORG.WRIT #1

Corrects the problem where the reorganization statement "GENERATE <data-set> ORDERED BY <set>" caused the DMS/REORG.WRIT program to terminate with "ATTEMPT TO READ OUTPUT ONLY FILE".

TABLE OF CONTENTS