

CORPORATE UNIT Computer Systems Group		LOCATION Santa Barbara Plant	DEPT. Language Development
NAME Language Development Department Personnel			DATE 20 August 1974
FROM F. Alan Goodman, Manager		DEPT. & LOCATION Language Development	
SUBJECT:		C.C.	

PRODUCT SPECS/USER MANUALS

R. Gonzalez

I've attached a list of how I view our responsibility to product specifications. In essence, all compilers will have two product specs, being

- a) The appropriate S-language
- b) The compiler product spec

The compiler product spec should address itself to

- a) A global description of the compiler
- b) Resources taken (memory/disk, etc.)
- c) Approximate performance or performance formulae
- *d) Compiler directed controls (\$ cards, conditional compilation, etc.)
- **e) Reference to the source syntax and semantics to a standards manual or user manual as appropriate.

Notes: * This portion will be duplicated in the product spec and user manual but, hopefully, will be static in nature.

** By referencing the source syntax/semantics in this fashion, the product specification will tend to be a stable document.

It is most desirable that once a standards manual/user manual is prepared for a software product, that the newly formed TRANSACTION scheme will provide the vehicle to update the appropriate document by persons other than the developers.

This is just another topic that indicates that if we document the TRANSACTION/TROUBLE reports well at development time, we can get out of the business of updating documents.



F. Alan Goodman, Manager
Language Development

gp
Attachment

(Engineering Responsibility)

(TIO Responsibility)

PRODUCT SPECIFICATIONUSER MANUALS

	Due to Sys Ctl	Responsible Person	
SDL/UPL Compiler	12/31/74	McCrea	SDL User Manual
SDL S-Language	12/31/74	McCrea	UPL User Manual
MIL Compiler	12/31/74	Thomas	MIL User Manual
RPG Compiler	3/31/75	Berman	RPG User Manual
RPG S-Language	3/31/75	Berman	
COBOL Compiler	12/31/74	Berman	COBOL User Manual
COBOL S-Language	10/01/74	Berman	
BASIC Compiler	12/31/74	Hedges	BASIC User Manual
BASIC S-Language	11/01/74	Hedges	
FORTRAN Compiler	12/31/74	Jobe	FORTRAN User Manual
FORTRAN S-Language	12/31/74	Jobe	
MCPIII Compiler	6/30/75	McCrea	(MCPIII Manual) ?
MCPIII S-Language	6/30/75	McCrea	
B300 Emulator (Free-standing)	TBD	McLean	B300 Emulator User Manual
B300 Emulator (MCP)	TBD	McLean	
1400 Emulator (Free-standing)	TBD	Belgard	1400 Emulator User Manual
1400 Emulator (MCP)	TBD	Belgard	
Emulator Loader	TBD	McLean	

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara Plant	DEPT. Language Development
NAME Rex Kerley		DATE 5 September 1974
FROM F. Alan Goodman, Manager		DEPT. & LOCATION Language Development
SUBJECT:		C.C.

1400 FREESTANDING EMULATOR "LEFT" ITEMS

R. Belgard

These are the items that you're to finish up prior to transferring into the MIL compiler project:

1. Run extended disk diagnostics successfully.
2. Run extended arithmetic diagnostics successfully.
3. Add PE tape capability. This should be done such that the emulator can run with 2K M-memory.
4. Add 7-track capability.
5. Add SCAN DISK capability.
6. Determine the disposition of two pages of miscellaneous issues with Dave Krajcar with respect to frequency of use, etc.
7. Complete documentation input to the above including product spec PS#2204-8680.

Rich Belgard will assume general project support at the completion of the above and in addition will

1. Add SYSOUT capability.

The following represents items that will be considered for later:

1. A discussion of COLUMN BINARY by Dave Krajcar in case this is to be implemented in the future.
2. To run 1401 COBOL. This would be desirable to do but would, in no way, hold up the emulator product.
3. To run 1401 F.A.S.T. This again would be desirable but may not be possible due to the absence of COLUMN BINARY.

Rex Kerley
5 September 1974
Page 2

4. Remove any machine dependency in the emulator, i.e., make a single 1714/1726 emulator product.

A handwritten signature in cursive script that reads "F. Alan Goodman". The signature is written in dark ink and is positioned above the typed name.

F. Alan Goodman, Manager
Language Development

gp

21 AUG 1974

August 21, 1974

To: Al Goodman
From: Larry Thomas
Subject: MIL Compiler

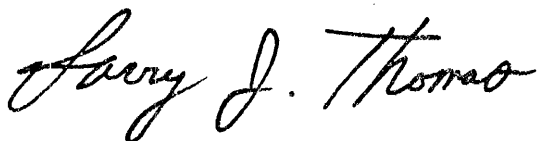
cc. Hank Havery
✓ Rex Kerley
Don McCrea

In anticipation of turning the MIL Compiler over to the Implementation Language Group, I have prepared a list of problems and suggestions for the compiler (attached). In general, compilation speed, at least on larger machines, seems to be adequate (up to 1500 CPM). Unless pressure exists to improve speeds on smaller machines, most effort in the near future should be concentrated on:

1. Making the compiler more reliable.
2. Making the output more responsive to users needs.
3. Making the language easier to use.

The compiler is generally in pretty good shape. There are problems however, and some of these problems are more important than others. I have organized the attached list into six arbitrary categories. These are:

1. Hi priority problems (only relatively speaking)
2. Medium priority problems
3. Error reporting problems
4. Hi priority suggestions
5. Medium priority suggestions.
6. Speed improvement suggestions.



Larry Thomas
Implementation and Virtual Machine Section
Language Development Department.

HI PRIORITY PROBLEMS

The MIL users manual is in bad shape. Progress on the BNF version is satisfactory but it should be completed as soon as possible. Scheduled completion is currently December, 1974.

The mechanism which generates a listing needs to be re-written. This mechanism essentially has to merge four components: code, code parameters, source, and source listing parameters. Only the first three components are well defined. The fourth component availability and the merging mechanism itself are ill conceived and should be redone. Some specifics associated with this problem are:

1. Printing the addresses of local declare elements is clumsy.
2. If the LIST \$ option is turned off at the beginning of a compilation and is later turned on and the pre-pass is listed then the final listing does not contain a heading.
3. Page numbers get messed up when \$ LIST is turned on and off.
4. An entire page of source only code at the beginning of a source file causes the headings to print at the wrong time on the listing.
5. The following sequence of cards results in the last one not being printed.

```
$LIST  
%
```

FINI

6. Whenever the compiler is running on a machine with a bad disk, and the bad disk causes the information contained in one of the four components to be wrong, then the entire listing is messed up starting from the point in error. This seems to happen quite frequently on some of our machines.

The code that generates the parameter blocks allows only a limited amount of space for conditional inclusion names. This currently causes problems when MIL analyzer is run against GISMO object files.

MEDIUM PRIORITY PROBLEMS

\$ FORCE does not work correctly because some errors seem to cause a lot of the branch addresses in the code file to be incorrect. The exact cause of this problem has not been identified.

IF-SKIP optimization needs to be reviewed. There seems to be some not uncommon times when the compiler does not generate optimal code when it should. (Hank Havery and Alan Grubb have some examples)

When adding a new reserved word to the compiler, the item in the symbol table is not truncated to the size that an incoming source item would be. This is an inconvenience for the person modifying the compiler.

\$NO COMPILE does not list \$ cards even when \$ DOLLAR is specified.

An external label reference before any non external label reference or declaration causes the compiler to loop when it encounters the first CODE, SEGMENT statement. Appears to be a problem only in test cases but should be fixed pronto.

The following code will not generate what the user wants and, depending on the block name, may not generate a syntax error. The only solution may be to spell out the problem in the users manual.

```
IF SUBSET THEN INCLUDE
BEGIN SUBSET
:
END ELSE SUBSET
BEGIN NOT, SUBSET
:
END NOT, SUBSET
```

The compiler will not accept all legal system file names on the & LIBRARY card

ERROR REPORTING PROBLEMS

An error occurring within a TABLE statement causes the compiler to get lost . Other strange errors then get reported thus confusing the issue. The compiler should probably try to scan to the END associated with the TABLE statement when it finds this kind of error.

Errors in strings should point at the character in error rather than at the string terminator.

Empty strings generate different errors depending on whether or not it is a MOVE or WRITE.STRING statement. Should be consistent.

A string with no closing bracket in a MOVE or WRITE.STRING statement generates an inappropriate error message.

Better checking of what can and cannot be changed on \$ cards after a compilation begins needs to be looked into. Perhaps should not allow change after an option has been referenced by the compiler or used by the user. Otherwise change would be OK.

HIGH PRIORITY SUGGESTIONS

Add patch control facility.

Allow arithmetic expressions in the remaps and length portions of the DECLARE statement.

Do something about problem of branch out of range conditions in long programs. (See TR N00102).

Supply a warning message when void range overlaps card patches.

Supply an error message when VOID begin seq no. is greater than VOID end seq. no.

Add "specials" so that words that are currently reserved could be defined by the user.

Add a VALUE statement to the DECLARE statement. This should eliminate TABLES in many cases.

Separate the ERROR file from the regular listing file.

MEDIUM PRIORITY SUGGESTIONS

Give the user the ability to specify (via \$ options) which subset of the following items he would like displayed on his listing on a line by line basis. (Segment names, Block names, Some other user specified name, BEGIN-END nesting level, Output card sequence number if asked for punched deck, Code address relative to the beginning of the code file, Code address relative to the current code block). The problem with this whole thing is that there is not enough room on the listing, so some trade offs will have to be made.

Optionally produce output similar to that produced by the MIL ANALYZER with each listing.

Provide syntax independent conditional inclusion mechanism. (perhaps similar to SDL's facility).

Provide a more general segmentation facility. (See Alan Grubb for his ideas).

Optimize segment handling to avoid emitting un-needed code.

Allow arbitrary boolean expressions to appear in IF ... INCLUDE statements.

Add new syntax to eliminate the error prone, unappealing, and unreadable constructs:

```
BIAS...TEST
INC...TEST
DEC...TEST
```

Add subtitles to listing that can be changed dynamically.

Consider eliminating the level emitting at the end of the program. Need to poll users to see if it is used.

Provide a more general way of emitting a hash checking micro.

Provide better syntax to make it easier for the user to specify the level of his program.

MEDIUM PRIORITY SUGGESTIONS (continued)

Add a continuation card facility. Perhaps -% would do for syntax.

Add TRACE pseudos for use with MIL emulator.

ADD a facility which would allow over-riding of \$ cards which are embedded in the source. Examples \$LIST - \$NO LIST, \$SEQ - \$NO SEQ.

SPEED IMPROVEMENT SUGGESTIONS

Provide library object files. Currently we have only library source files. This should not be too hard to do and it would help diagnostics and emulators which end up having to re-compile I/O drivers over and over again

Re-write scanner to make more modular and to use the new scan S-ops. The current scanner is the original one written three and a half years ago. When this is done, another pass should probably be added to perform all of the scanning before code emitting begins. This should improve compilation speed in small machines.

Investigate reorganization of reserved words and perhaps initial declarations in the symbol table so that items that hash to the same stack head reside in the same page or in contiguous pages. This should reduce page faults. The SEARCH, SERIAL, LIST op might then be used to good advantage also.

Handling of the macros should be investigated. Some space is wasted when the macro records are stored away. The new POSITION communicate would also be advantageous in speeding up macro processing.

5 Sept. 1974

To: Distribution
From: Larry Thomas
Subject: Overview of MIL COMPILER Organization

cc:

Al Goodman
Hank Havery

You are invited to attend a class which will briefly cover the overall organization of the Mil Compiler. The class will be held on Monday and Tuesday, Sept. 9 and 10, 1974, from 2:30 to 4:30 in the new conference room (by Mr. Bunker's office). Monday's class will cover in very general terms the compiler's organization. Tuesday's session will be reserved for detailed questions and answers.



Larry Thomas
Implementation and Virtual Machine Section
Language Development

Distribution:

Dennis Austin
Rex Kerley
Don McCrea
Dick Palmer
Chip Pardini

OPERATION OF THE
MIL COMPILER

I	INITIALIZE	04603000 — 05123000
II	COMPILE.IT.ALL -- the compile phase	09846000 — 10810000
III	WRAP.UP.COMPILE	10816000
IV	CLEAN.UP.LABELS -- the fixup phase	10949000
V	GENERATE.ALL.OUTPUT.FILES -- the listing phase	11345000

The MIL compiler is generally considered a one pass compiler since all symbol table lookup and 90% of the code generation is done in the compile phase.

Test/confidence/verification

Speeds

lists of use(s) EMULATORS
INTERPRETERS
DIAGNOSTICS

TR

SDC page #'s. space (asoftic)

MIL XREF labels/page #'s on PBD.

4660

I INITIALIZE

- A. Global variables
- B. Main symbol table
- C. Compile phase files

4860

II COMPILE.IT.ALL

- A. Makes an initial call on READACARD which
 - 1. Handles \$ cards until 1st non-\$ card
 - 2. Calls on INITIALIZE.COMPILE to
 - a) lock in some \$ options(e.g. MERGE)
 - b) get the time and COMPILE.CARD.INFO
 - c) start off the xref and release tape processes
- B. Handles all source records
 - 1. Handles label declarations
 - 2. Scans for the keyword(e.g. MOVE), looks it up in the symbol table and uses symtype to select a particular case in the big COMPILE.IT.ALL case statement
 - 3. Each keyword case
 - a) handles its own semantics by individually driving
 - 1) the scanner
 - 2) the code emitter
 - 3) the code parameter emitter
 - b) handles most of its own error checking
 - 4. Generates the preliminary listing file, LINESAVE, on disk

III WRAP.UP.COMPILE

10900

- A. Completes and closes compile phase files
- B. Checks for FINI record
- C. Closes out any open code segments
- D. Zips MIL/XREFER if not \$ RELEASE

IV CLEAN.UP.LABELS

11300

- A. Changes the code parameter file, PARAM.FILE, to a random file so that it can be read backwards to finish the optimization
- B. The first part uses the code parameter file and the COMLAB table to factor in the effects of SEGMENT and ADJUST statements. At this time, COMLAB.ADDR is changed to be the code address the label was associated with in the compile phase(copied from PARAM.FILE). At the same time, adjustments to this code address are accumulated in another field in the COMLAB table.
- C. The second part, FINISH.CLEANUP.PASS, massages the COMLAB table to reflect the actual final addresses to be associated with each label. This information is put into COMLAB.ADDR.
- D. The PARAM.FILE is changed back to a serial file and the symbol table is searched for labels which were referenced but never declared.

V GENERATE.ALL.OUTPUT.FILES

1 2960 (inner loop 12590)

- A. SET.UP.OUTPUT.PASS initializes listing phase variables and files
- B. GENERATE.TEMP.CODE.FILE.AND.LISTING merges the information from the compile phase contained in the code(CODE.FILE), listing(LINESAVE) and code parameter(PARAM.FILE) files. It uses these files and the COMLAB table to simultaneously generate
 1. The listing
 2. The list portion of the RELEASE tape
 3. The code portion(no PPB, IPB, etc.) of the final code file by emitting extra code for IF's that branch more than 15 words, special LOAD options, etc.
- C. BUILD.CODE.FILES outputs the final code file by adding the PPB, IPB, conditional include block, correspondence table and segment dictionaries to the TEMPCODEFILE produced in B.3 above, and punches an object deck if \$ DECK was specified
- D. Finishes up the RELEASE tape by adding the punch, code, source and xref files and thens zips MII/XREFER(if \$ RELEASE was specified)

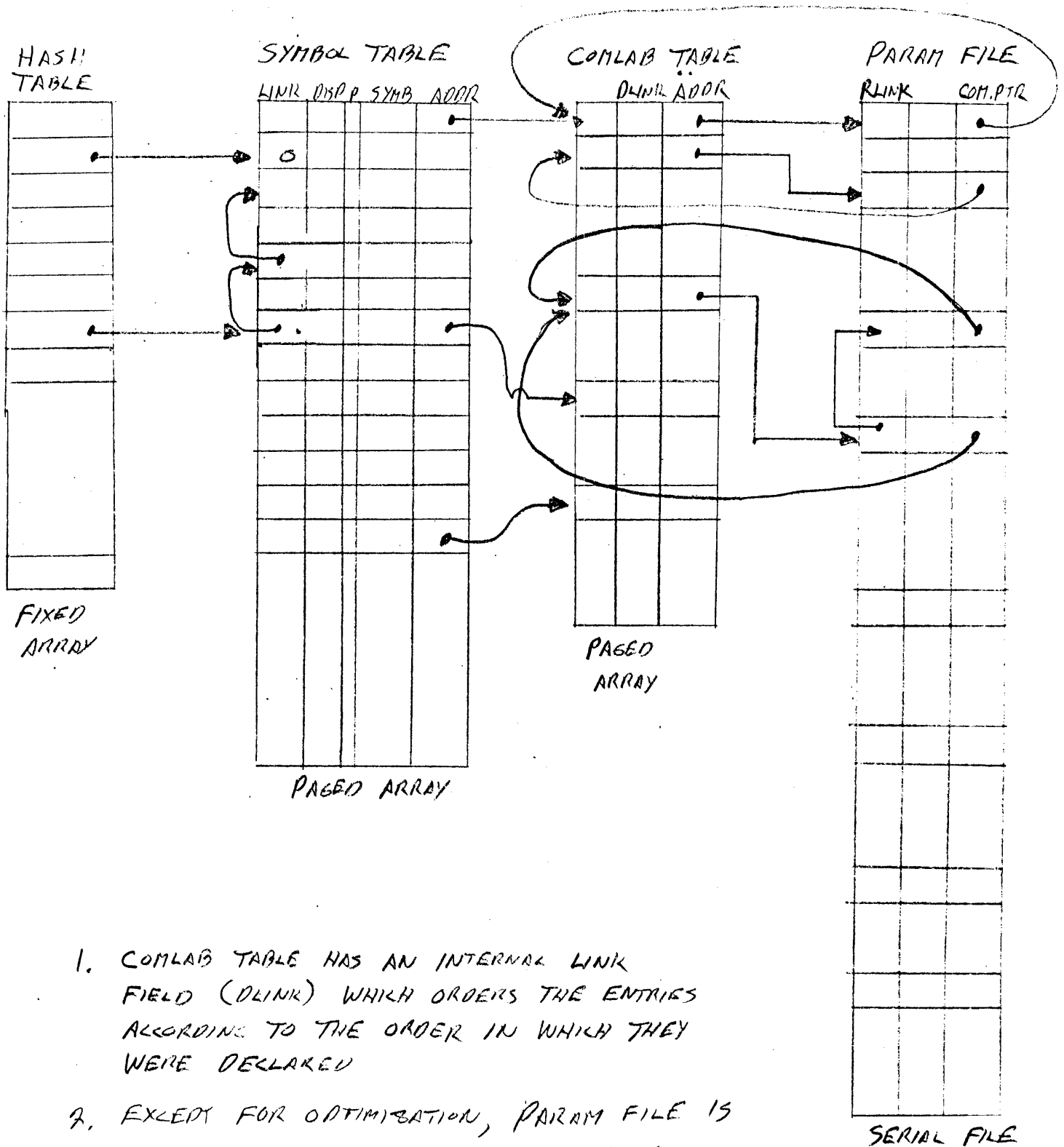
MIL SYMBOL TABLES

1. EACH ENTRY IS VARIABLE IN LENGTH.
2. FIRST 7 BYTES OF ENTRY CONTAIN SYMBOL ATTRIBUTES.
3. NEXT m BYTES ARE THE SYMBOL ITSELF.
4. NEXT 3 BYTES ARE OPTIONAL AND MAY CONTAIN ADDITIONAL ATTRIBUTES.
5. THE LAST THREE ATTRIBUTES IN THE FIRST 7 BYTES ARE ALWAYS:
 - a. DISP = BYTE DISPLACEMENT FROM THE BEGINNING OF THIS ENTRY TO THE BEGINNING OF THE NEXT ENTRY. THIS MAY SPAN PAGE BOUNDARIES.
 - b. LENGTH = THE LENGTH IN BYTES OF THE SYMBOL.
 - c. LINK = THE BYTE ADDRESS OF THE PREVIOUS ENTRY IN THE SYMBOL TABLE WHICH HAS THE SAME HASH CODE AS THIS ENTRY'S SYMBOL.
6. THE SYMBOL TABLES ARE ORGANIZED AS A PAGED ARRAY WITH 256 BYTES PER ELEMENT, 1 ELEMENT PER PAGE.
7. EACH ELEMENT IS ACCESSED USING SUBSTR & SUBBIT TO ISOLATE THE ENTRIES AND THE FIELDS WITHIN THE ENTRIES.
8. EACH ENTRY MUST BE CONTAINED ENTIRELY WITHIN ONE PAGE. THUS, SPACE MAY BE WASTED AT THE END OF A PAGE IF THAT SPACE COULD NOT CONTAIN THE NEXT ENTRY.
9. THERE ARE TWO SYMBOL TABLES
 - a. ONE FOR RESERVED WORDS, DEFINE IDENTIFIERS, DECLARE IDENTIFIERS, MACRO IDENTIFIERS
 - b. ONE FOR UNIQUE LABELS AND POINT LABELS
10. THE FIRST SYMBOL TABLE GROWS AND CONTRACTS AS COMPILATIONS PROCEED THRU PARAMETRIC MACROS AND BEGIN-END BLOCKS. CONTAINING LOCAL DEFINES OR DECLARES. THIS IS ACCOMPLISHED BY SAVING AND RESTORING THE HASH TABLE.

MIL LABEL HANDLING

1. LABEL SYMBOLS ARE ORGANIZED INTO A SYMBOL TABLE JUST LIKE OTHER SYMBOLS.
2. REGULAR (UNIQUE) LABELS ARE ENTERED THE FIRST TIME THEY ARE SEEN
3. POINT LABELS ARE GIVEN TWO ENTRIES IN THE SYMBOL TABLE, ONE ENTRY FOR THE FORWARD REFERENCES AND ONE ENTRY FOR BACK REFERENCES. THESE SAME TWO ENTRIES ARE USED REGARDLESS OF THE NUMBER OF TIMES THE SAME POINT LABEL TOKEN IS DECLARED.
4. EACH TIME A NEW LABEL IS DECLARED OR REFERENCED IT IS GIVEN A UNIQUE ENTRY IN THE COMMON LABEL PARAMETER TABLE. (THE FIRST FORWARD REFERENCE TO A PARTICULAR POINT LABEL OR ANOTHER DECLARATION OF A POINT LABEL CAUSES THIS TO HAPPEN)
5. THE SYMBOL TABLE ENTRY POINTS TO THE COMMON LABEL PARAMETER TABLE ENTRY VIA SYMLABADDR. (THIS FIELD IS NORMALLY NOT CHANGED EXCEPT WHEN THERE IS MORE THAN ONE DECLARATION OF THE SAME POINT LABEL TOKEN.)
6. THE COMMON LABEL PARAMETER TABLE (COMLAB TABLE) IS SEPARATED FROM THE SYMBOL TABLE INTO A SEPARATE, CONCISE PAGED ARRAY BECAUSE IN LATTER FIXUP PASSES THE TABLE IS ACCESSED VERY OFTEN AND VERY RANDOMLY.
7. ANOTHER TABLE (A SERIAL FILE PARAM. FILE) CONTAINS FURTHER INFORMATION ABOUT EACH LABEL DECLARATION AND REFERENCE.
8. COMLAB.ADDR POINTS TO THE LAST PARAM. FILE ENTRY MADE FOR FORWARD REFERENCES UNTIL THE LABEL IS DECLARED. ONCE A LABEL IS DECLARED, COMLAB.ADDR WILL ALWAYS POINT TO THE PARAM. FILE ENTRY FOR THE LABEL DECLARATION. THE SECOND PASS CHANGES COMLAB.ADDR TO BE THE ACTUAL LABEL ADDRESS.

FIRST PASS LABEL TABLE ORGANIZATION



1. COMLAB TABLE HAS AN INTERNAL LINK FIELD (DLINK) WHICH ORDERS THE ENTRIES ACCORDING TO THE ORDER IN WHICH THEY WERE DECLARED
2. EXCEPT FOR OPTIMISATION, PARAM FILE IS ACCESSED SERIALLY. OPTIMIZATION IS ACCOMPLISHED BY BLOCKING ADD BUFFERING THE FILE INTERNAL TO THE COMPILER TO PROVIDE A SUFFICIENT NUMBER OF RANDOMLY ACCESSABLE ENTRIES.

MIL
DEFINE HANDLING

1. When a DEFINE statement is encountered in the source, the define string is placed into a paged array reserved for define strings, and the defined token is inserted into the symbol table along with address and length info about the associated define string.
2. The scanner, in general, looks up every symbol in the symbol table. When it encounters a defined symbol, it will locate the define string, copy it into the next available location in a 100 byte define buffer, save current information about the scan pointer in a stack, change the scan pointer to point to the beginning of the define string in the define buffer, and finally recursively call the scanner to scan the first token in the define string(the procedures themselves are not recursive). Before the end of the define string is reached, this same process could happen again until either the maximum define nesting level is reached or the 100 character buffer is filled.
3. The end of each define string(and the end of a record) is identified by a %.
4. When the scanner scans a %, it will first check the define nesting level(DL). If DL is non zero, it will restore the scan pointer from the top of the DL stack and then resume scanning from there. Otherwise, it returns an end of record flag.

MIL
MACRO HANDLING

1. Non parametric MACROs: the macro records are scanned to eliminate unnecessary blanks and then placed into a paged array.
2. Parametric MACROs: the macro records are scanned, the formal parameter instances are replaced with an internal symbol identifying the macro number and parameter number, and the resulting record is put into a paged array.
3. Parametric MACRO invocation: the internal formal parameters are generated as defined tokens and the actual parameters are made define strings and associated with the internal formal parameter tokens.
4. At the end of parametric macro invocation, the symbol table and define string paged array pointers are cut back to the state they were in before the parametric macro invocation.

MIL COMPILER
IF, SKIP OPTIMIZATION

1. It is assumed that all instructions will generate the minimum amount of code in the first(compile) phase.
2. Any instruction which is potentially a branch(IF, SKIP, GO TO and CALL) always has a label associated with it(exceptions: INC TEST, DEC TEST and BIAS TEST). The label may be internally generated if it is not explicit in the source.
3. Each such instruction has a parameter associated with it in the serial PARAM.FILE. Each label also has a parameter in the PARAM.FILE.
4. The PARAM.FILE is blocked by the compiler itself to guarantee that, for IF and SKIP, the label parameter and the reference parameter will always be available to the compiler for inspection and change if the statements can be optimized.
5. If the compile phase knows for certain that an IF or a SKIP will generate exactly one or exactly two micro words, the parameters are marked accordingly and optimization is complete for these statements. Otherwise, the optimization decision is left ~~the~~ ^{to} the fixup phase.
6. Optimization attempts in the compile phase occur whenever a statement referencing a previously declared label is encountered or when a label which had forward references is declared.
7. In an effort to complement the compile phase attempt at optimization, the fixup phase scans the PARAM.FILE in reverse. This approach does not guarantee complete optimization, but it does seem to cover the majority of cases.

Burroughs Corporation

INTER-OFFICE CORRESPONDENCE

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara	DEPT. Software Activity
NAME Distribution		DATE September 30, 1975
FROM K. King	DEPT. & LOCATION Applications & Operations	

SUBJECT:

C.C.

SOFTWARE PRODUCT SPECIFICATIONS STATUS REPORT

MICOR

Distribution:

- | | | |
|------------|-------------|-------------------|
| D. Austin | R. Gonzalez | K. Meyers |
| A. Bates | A. Goodman | C. Petersen |
| R. Bauerle | G. Hammond | B. Rappaport |
| R. Berman | H. Havery | D. Roberts |
| S. Bryan | J. Jobe | L. Thomas |
| T. Cardona | R. Kerley ✓ | A. van der Linden |
| J. Casey | G. Lapman | R. Vail |
| D. Goebel | C. Logan | A. Yardi |
| | R. Maxwell | |

Enclosed is an updated status report for software product specifications that are in the process of being revised or written, based upon information of September 30, 1975. The report has been revised to give a more accurate indication of the progress of each specification. The dates in the columns are explained below.

- Column 1. Gives the date by which the programmer should submit the product spec. to the documentation group.
- Column 2. The date on which work actually began on formatting the document with DOCEDR.
- Column 3. The first rough draft was returned to the programmer for review and corrections.
- Column 4. The day the spec. was released by documentation for managerial review.
- Column 5. The date when the product spec. completed managerial review and was released to Records for distribution.

The column headed TYPE will have several possible entries. The first entry gives the type of document (i.e., spec., manual, documentation). The ESTM entry gives the estimated dates for submittal to and completion by documentation. CMPL shows the actual date on which each stage was completed.

Any questions regarding scheduling or completion of a document should be directed to Jack Casey, Ext. 389.

K. King
Documentation
Applications & Operations

jg

SOFTWARE PRODUCT SPECIFICATION
 STATUS REPORT
 10/03/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PROC.	ASSN. DATE OCCU.	FIRST ROUGH DRAFT	PROJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS

				DATA COMM							
	PROD	2212 5256	A	HASP (4 SECTIONS)						MAXWELL	RESCHEDULED
	ESTM				1/01/75		2/21/75				
	CMPL				9/04/74	9/23/75					
	ESTM						11/14/75				
	PROD	2212 5272	A	RJE/DCH						MAXWELL	RESCHEDULED
	ESTM				6/06/75		8/15/75				
	CMPL				6/05/75	7/01/75	8/04/75	8/04/75			
	PROD	2212 5421	A	DC/AUDIT						MAXWELL	
	ESTM				7/01/75		7/31/75				
	CMPL				6/16/75	6/16/75					
	ESTM						7/31/75				
	CMPL						7/03/75				
	PROD	2212 5439	A	TEXT/EDITOR						LOGAN	
	ESTM				11/01/75		11/30/75				
	PROD	2212 5447	A	ILLUSTRATED MCS						BATES	
	ESTM				12/01/75		1/30/76				
	PROD	2212 5454	A	DATACOMM SYS						BATES	
	ESTM				2/01/76		4/01/76				
	PROD	0000 0000	A	CANDE						LOGAN	DATES TBD

				OP. SYS							
	PROD	2212 5452	A	MCP II						BAUERLE	DATES TBD
	PROD	1912 5681	A	MCP I						BAUERLE	DATES TBD
	PROD	2212 5470	A	DMS						BAUERLE	DATES TBD
	PROD	0000 0000	A	I/O DRIVER						BAUERLE	DATES TBD
	PROD	0000 0000	A	MCP UTILITIES						BAUERLE	DATES TBD

				SFTWR. UTIL.							
	PROD	2212 5488	A	CART INITIALIZER	10/01/75		10/30/75			PETERSEN	

SOFTWARE PRODUCT SPECIFICATION
STATUS REPORT
10/03/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PROG.	ASSN. DATE DCU.	FIRST RCUGH DRAFT	PROJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS
	PROD	2212 5496	A	CART DUMP	10/01/75		10/30/75			PETERSEN	
	PROD CMPL	2212 5157	B	CASSETTE/MAKER	10/01/75 9/15/75		10/30/75			PETERSEN	
	PROD ESTM	0000 0000	A	IN-PLACE SORT	11/01/75		11/30/75			VAIL	
		----		PROG LANGS							
	PROD ESTM CMPL ESTM CMPL	2201 2389	G	SDL S-LANGUAGE	12/31/74 12/30/74	5/01/75	6/30/75 11/30/75 9/05/75			AUSTIN	
	PROD ESTM CMPL ESTM CMPL	2212 5405	A	SDL (BNF VERSION)	12/31/74 12/30/74	3/01/75	8/01/75 8/01/75 9/05/75			AUSTIN	
	PROD ESTM	2212 5306	A	RPG S-LANGUAGE	12/01/75		1/30/76			RAPPAPORT	
	PROD ESTM	2205 1155	B	RPG COMPILER	10/01/75		10/30/75			RAPPAPORT	
	PROD	2212 5298	A	MIL COMPILER						HAVERY	DATES TBC
	PROD ESTM CMPL ESTM	2201 6729	E	COBOL S-LANGUAGE	10/01/74 9/11/74	8/15/75	12/01/74 11/30/75			BERMAN	RESCHEDULED
	PROD	2212 5397	A	COBOL COMPILER DCC						BERMAN	DATES TBC
	PROD ESTM CMPL	2212 5348	A	9500 V M INTERP	11/01/75 7/22/75		12/01/75			ROBERTS	NEW AT 5.1
	PROD ESTM	2212 5355	A	1400 V M INTERP	10/01/75		10/30/75			LAPMAN	NEW: 1QTR76
	PROD	2212 5313	A	9500 INTP ENV PROG						ROBERTS	NEW AT 5.1

SOFTWARE PRODUCT SPECIFICATION
 STATUS REPORT
 10/03/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PRG.	ASSN. DATE OCCU.	FIRST ROUGH DRAFT	PROJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS
	ESTM					8/15/75		9/15/75			
	CMPL					9/22/75					
PROD		0000 0000	A	1400 EMUL ENV PRGR						KERLEY	NEW: 1GTR76
PROD		2200 9849	E	B500 EMUL FR-STNDG						KERLEY	DATES TBD
PROD		2204 8680	C	1400 EMUL FR-STNDG						KERLEY	DATES TBD
PROD		0000 0000	A	CREATE/B500 DISK						KERLEY	
ESTM						9/01/75		10/01/75			
PROD		0000 0000	A	CREATE/VRTL 1311						KERLEY	DATES TBD

----- DOCUMENTS/MANUALS

MANL		0000 0000		QCMCPI/II							
ESTM						9/05/75		10/05/75			
CMPL						9/05/75	9/06/75				
ESTM								10/03/75			
MANL		0000 0000	A	REMOTE EDIT MANL						CASEY	
ESTM						8/15/75		9/15/75			
CMPL						8/15/75	8/15/75				
ESTM								9/15/75			
CMPL								9/15/75			
DOC		0000 0000	A	NEWSLETTER (V.1)						CASEY	
ESTM						9/01/75		9/15/75			
DOC		0000 0000	A	B1700 SYS INDEX						CASEY	
ESTM						9/15/75		9/30/75			
CMPL								9/02/75			

----- RELEASE DOCUM

DOC		0000 0000	A	V.1 RELEASE						CASEY	
ESTM						10/01/75		10/30/75			

----- COMPLETED SPECS

SOFTWARE PRODUCT SPECIFICATION
 STATUS REPORT
 10/23/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PROG.	ASSN. DATE DCCU.	FIRST ROUGH DRAFT	PRCJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS
PRDD		2201 2389	F	SDL S-LANG					3/27/72	AUSTIN	SCHEDULED FOR REV.
PRDD		2201 6729	D	COBOL S-LANG					11/28/72	BERMAN	SCHEDULED FOR REV.
PRDD		2200 9849	D	B300 EMUL FR-STNDG					1/22/73	KERLEY	SCHEDULED FOR REV.
PRDD		2204 8680	B	1400 SERIES EMUL					1/22/73	KERLEY	SCHEDULED FOR REV.
PRDD		2212 5082	A	SPD CART INIT	9/11/74	9/24/74	10/15/74	10/25/74	10/28/74	PETERSEN	
PRDD		2212 5066	A	DISK COPY	9/11/74	10/04/74	10/17/74	10/25/74	10/28/74	PETERSEN	
PRDD		2212 5124	A	FILE PUNCHER	9/11/74	9/24/74	10/15/74	10/25/74	10/28/74	PETERSEN	
PRDD		2212 5074	A	DISK CART INIT	9/10/74	10/04/74	10/17/74	10/25/74	10/28/74	PETERSEN	
PRDD		2212 5090	A	DISK PACK INIT	9/11/74	9/24/74	10/17/74	10/25/74	10/28/74	PETERSEN	
PRDD		2212 5140	A	CASSETTE LOADER	9/11/74	10/08/74	11/06/74	11/14/74	11/15/74	PETERSEN	
PRDD		2212 5165	A	SSL0AD MAKCAS	9/10/74	10/28/74	11/06/74	11/14/74	11/15/74	PETERSEN	
PRDD		2210 0135	C	BASIC S-LANGUAGE	8/23/74	10/15/74	11/07/74	11/13/74	11/15/74	HEDGES	
PRDD		2212 5108	A	DISK DUMP	9/10/74	10/28/74	11/06/74	11/18/74	11/19/74	PETERSEN	
PRDD		2212 5157	A	CASSETTE MAKER	9/11/74	10/28/74	11/06/74	11/18/74	11/19/74	PETERSEN	
PRDD		2212 5132	A	FILE LOADER	9/10/74	10/28/74	11/20/74	12/02/74	12/06/74	PETERSEN	
PRDD		2212 5173	A	CARTRIDGE/DUMP	11/27/74	12/03/74	12/05/74	12/13/74	12/16/74	PETERSEN	
PRDD		2212 5280	A	BASIC COMPILER	12/01/74	12/12/74	1/13/75	1/23/75	1/25/75	HEDGES	
PRDD		2212 5116	A	DMPALL	9/11/74	11/08/74	12/16/74	12/29/74	1/30/75	PETERSEN	
PRDD		2212 5215	A	NOL/LIBRARY	10/30/74	11/27/74	1/07/75	1/31/75	2/03/75	COWLER	
PRDD		2212 5181	A	TAPE SORT INTRIN	9/10/74	12/12/74	1/14/75	2/03/75	2/11/75	CARCCNA	
PRDD		2201 6752	C	B1700 SORT LANG	10/03/74	11/22/74	2/10/75	2/27/75	2/28/75	VAIL	
PRDD		2212 5064	A	DISK SORT INTRIN	11/14/74	12/28/74	1/29/75	3/11/75	3/15/75	DEPENDAHL	
PRDD		2212 5231	A	RJE/TERMINAL (COL)	9/04/74	12/16/74	12/28/74	3/17/75	3/18/75	GOEBEL	
PRDD		2212 5240	A	RJE/TERMINAL (NOL)	9/04/74	12/16/74	12/28/74	3/17/75	3/18/75	GOEBEL	

SOFTWARE PRODUCT SPECIFICATION
 STATUS REPORT
 10/03/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PROG.	ASSN. DATE CCCU.	FIRST ROUGH DRAFT	PRCJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS
PROD		2212 5199	A	SYCOPY	2/14/75	3/20/75	3/24/75	4/03/75	4/04/75	PETERSEN	
PROD		2212 5207	A	MERGE INTRINSIC	12/27/74	1/29/75	3/13/75	3/27/75	4/04/75	VAIL	
PROD		2212 5389	A	SDL/UPL COMPILER	12/31/74	2/21/75	3/21/75	4/03/75	4/07/75	AUSTIN	
PROD		2212 5371	A	TRANS TABLE GEN	3/24/75	4/22/75	6/09/75	6/13/75	6/16/75	DEPENDAFL	
PROD		2212 5214	A	COBOL COMPILER	12/31/74	6/11/75	6/19/75	6/28/75	7/01/75	BERMAN	
PROD		2201 6737	A	FORTRAN S-LANG.	12/31/75	6/07/75	6/19/75	7/29/75	8/04/75	JOBE	
PROD		2212 5215	B	NDL/LIBRARY	4/24/75	6/06/75	6/20/75	8/05/75	8/08/75	DUPAR	
PROD		2212 5322	A	FORTRAN COMPILER	12/31/74	6/13/75	6/19/75	8/06/75	8/11/75	JOBE	
PROD		2201 6737	B	FORTRAN S-LANG.	8/04/75	8/18/75	8/27/75	8/27/75	8/28/75	JOBE	
PROD		2212 5330	A	EMUL LOADER	7/14/75	7/23/75	8/22/75	9/02/75	9/02/75	KERLEY	
PROD		2212 5223	A	NDL	6/16/75	8/07/75	8/11/75	9/08/75	9/09/75	BATES	
PROD		2212 5231	B	RJE/TERMINAL (SDL)	8/01/75	8/15/75	9/15/75	9/19/75	9/22/75	MAXWELL	
PROD		2212 5249	B	RJE/TERMINAL (NDL)	8/01/75	8/15/75	9/15/75	9/19/75	9/22/75	MAXWELL	

SOFTWARE PRODUCT SPECIFICATION
 STATUS REPORT
 10/03/75

STAT	TYPE	PRODUCT SPEC NUMBER	REV	PRODUCT SPEC TITLE	DEV. DRAFT PROG.	ASSN. DATE DCCU.	FIRST ROUGH DRAFT	PROJ REVIEW	MANAGERS REVIEW	RESPONSIBLE PERSON	COMMENTS

				DOCUMENTS ON HAND							WEEK: SEPT 28 - OCT 4
		----	----	GUT:MGR'S APPROVAL							
PROD		2212 5272	A	B1700 RJE/DCH			8/04/75	8/04/75		BRYAN	
PROD		2212 5421	A	DC/AUDIT			6/16/75	6/16/75		CASEY	
		----	----	IN: DOCU OFFICE							
PROD		2201 6729	E	COBOL S-LANG		9/08/75				KING	FOR REVISION
MANL		0000 0000		QCMCPI/II		9/05/75				BRYAN	
PROD		2212 5256	A	B1700 HASP		8/08/75				BRYAN	
		----	----	IN: PROG OFFICE							
PROD		2201 2389	G	SDL S-LANGUAGE			9/05/75			KING	
PROD		2212 5405	A	SDL (BNF VERSION)			9/05/75			CASEY	
		----	----	IN: TO BE SCHED							
PROD		2212 5397	A	COBOL COMPILER DOC						KING	
PROD		2212 5462	A	MCPII						CASEY	FOR TRANS TO UC/LC
PROD		2212 5470	A	DMS						BRYAN	FOR TRANS TO LC/LC
PROD		2212 5157	B	CASSETTE/MAKER							
PROD		2212 5348	A	B500 V M INTERP							
PROD		2212 5313	A	B500 INTP ENV PROG							

TO:

CORPORATE UNIT Computer Systems Group		LOCATION Santa Barbara Plant	DEPT. Programming Activity
NAME Programming Activity Personnel		DATE 8 September 1975	
FROM W. Michael Denny		DEPT. & LOCATION Systems Technology	
SUBJECT:		C.C.	

CPM PERFORMANCE MONITOR

As of September 8, 1975 five machines in the Software Development Machine Room are connected to the CPM Performance Monitor. These are:

- #384 (Closed Shop-B1726)
- #366 (MCP Development-B1726)
- #377 (Language Development-B1726)
- #308 (Data Comm-B1726)
- #1186 (128KB-B1714) S-Processor - 2 ???

The CPM cable connections to all these machines have been made to their respective backplanes and the cables are routed under the floor and out behind the CPM. Each group of cables is clearly labeled at the CPM end with the number of the respective machine and the proper socket into which each cable is plugged on the CPM.

With this arrangement, it is possible to measure software on any of the above machines with only about five minutes' work connecting up the cables to the CPM.

If anyone is interested in a class on how to wire the CPM plugboard, please contact Mike Denny.

Mike Denny

W. Michael Denny
Systems Technology

gp

SOURCE IMAGE	IDENTIFICATION
ENVIRONMENT.NAME "B300"/"EMULATOR"	: 00001000
MEMORY.SIZE 9.6K	: 00002000
CARD.READER "CARDS"	: 00003000
PRINTER1 BACKUP DISK/TAPE	: 00004000
120 POSITIONS	: 00005000
FORMS	: 00006000
HARDWARE	: 00007000
PRINTER2 BACKUP DISK	: 00008000
132 POSITIONS	: 00009000
PUNCH	: 00010000
SU0 "HAPPY"/"DAYS"	: 00011000
CARTRIDGE	: 00012000
NEW	: 00013000
SU1 "FILE"/"ONE"	: 00014000
OLD	: 00015000
PACK	: 00016000
SU2 "DISK"/"FILE"	: 00017000
FILE	: 00018000
NEW	: 00019000
SAVE.AT.COMPLETION	: 00020000
SU3 "PACK"/"FAMILY"/"NAME"	: 00021000
SU4 "AA"	: 00022000
SU5 "BB"	: 00023000
SU6 "CC"	: 00024000
SU7 "CC"	: 00025000
SU8 "DD"	: 00026000
SU9	: 00027000
TAPE1 MTA	: 00028000
PARITY ODD	: 00029000
TAPE	: 00030000
TAPE2 MTA	: 00031000
TAPE,7	: 00032000
PARITY EVEN	: 00033000
TAPE3 MTC	: 00034000
TAPE,9	: 00035000
TAPE4 MTD	: 00036000
TAPE5 MTE	: 00037000
TAPE6 MTF	: 00038000
TRACE OFF	: 00039000
ENABLE.TERMINATE.PSEUDO	: 00040000
END	: 00041000

END B300 EMULATOR ENVIRONMENT COERSION
 NO ERRORS IN ENVIRONMENT SPECIFICATION
 B300 /EMULATOR SUCESSFULLY COERCED

ENVIRONMENT CHARACTERISTICS:

CORE REQUIRED TO RUN: 0179995 BITS

DEVICES SPECIFIED:

- CARD.READER
- PRINTER1
- PRINTER2
- PUNCH
- SU0
- SU1
- SU2
- SU3
- SU4
- SU5
- SU6
- SU7
- SU8
- SU9
- TAPE1
- TAPE2
- TAPE3
- TAPE4
- TAPE5
- TAPE6

OTHER CHARACTERISTICS:

- TERMINATE.PSEUDO ENABLED
- TAPE.BUFFER.SIZE = 0115200 BITS(DEFAULT)

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara Plant	DEPT. Programming Activity
NAME Programming Activity Personnel		DATE 9 September 1975
FROM B. Dodson	DEPT. & LOCATION Language Development	
SUBJECT		C.C.

SYSTEM/PATCH

Today I am releasing a version of SYSTEM/PATCH with the following changes:

1. The maximum number of patch cards is increased from 4086 to 16374.
2. A new option \$#PAGEP is added. This causes each patch to begin on a new page in the LISTP listing.
3. A new option \$#END is added. This serves as an end of file for SYSTEM/PATCH so that extra cards stored behind it are ignored.
4. The sequence field is blanked out on \$? cards (pseudo control cards).
5. When generating a new symbolic, \$LIST no longer has an effect. To get a listing of the new symbolic, the option \$# LISTNEW (alias LIST.NEW) has been added.

I encourage use of the pseudo deck mode of operation for SYSTEM/PATCH, especially for closed shop jobs.

Bill

B. Dodson
Competitive Languages Section
Language Development

8P

μ code on console

2-3-68 RTD

*NO TERMINALS / dedicated systems
closed shop turnaround*

I. Weaknesses of Current System

- A. Not enough jobs.
- B. Too many jobs.
- C. No one responsible for efficient operation.
- D. Bypassing or ignoring queue.
- E. Sign up just in case, then not ready.
- F. Printer tied up too long. *LAX EOF LPA*
- G. Card Reader tied up too long. *CRA or "LD"*
- H. Over running scheduled time

II. Goals

- A. Keep system running efficiently at all times, using controlled multiprogramming.
- B. Be able to run small jobs without queue sign up or long wait.
- C. Be able to leave simple jobs to be run during available time.
- D. Minimize stand alone (non-multiprogramming) time but allow for it when necessary.
- E. Allow for using text editors.
- F. Improve closed shop turnaround.

III. Proposed System

- A. Divide day into equivalent 30 minute sign up periods. Multi-program during all these periods with minimal exceptions.
- B. Maintain a queue of work to be done. Ideally and after this will consist of persons waiting with short jobs but will overflow into written queue.
- C. Sign up for a period implies:
 - 1. Commitment to be at machine during entire period.
 - 2. First priority.
 - 3. Responsibility for efficient operation of machine. This includes calling next person in queue and resolving problems and conflicts.
 - 4. Ensuring integrity of queue.

III. Proposed System (Continued)

- D. If a queue develops the following apply:
 - 1. If next person is not at posted phone number, bypass but leave at top of queue.
 - 2. If next person is not ready, move to bottom of queue.
 - 3. Person calling scratches name of person contacted.
- E. Provide for simple jobs to be run in absentia with following:
 - 1. No action required except putting deck through reader.
 - 2. Deck set up to force output to backup.
 - 3. Backup files not removed except at 8:00 and 12:00.
 - 4. SPO printout not removed from machine.
- F. To avoid tying up peripherals, use the following:
 - 1. Modify SDL and MIL to send output to backup. (Keep standard file cards on machine to override.)
 - 2. Use SYSTEM/PATCH or load control to avoid tying up readers.
 - 3. Minimize listings.
- G. To avoid overloading machine, do all create masters, long compiles, full listings, xrefs, etc. in closed shop. Submit formal complaint any time held up by closed shop turnaround.
- H. To improve multiprogramming efficiency, observe following:
 - 1. Modify SDL to use 50,000 bits dynamic.
 - 2. Avoid running two processor bound jobs together. This applies especially to SDL and MIL compiles.
 - 3. Run long jobs at lower priority. This gets short jobs out of the way resulting in best efficiency.
- I. Indicate the nature of work for all queue entries using the following classifications:
 - 1. I/O (LOAD/DUMP, DMPALL, listing, punching cards, etc.)
 - 2. Small jobs
 - 3. Recomp
 - 4. MIL compile
 - 5. Text Editor (Gordon Browne's)

III. Proposed System (Continued)

I. (Continued)

6. CANDE

7. Stand alone

J. If machine is preempted for emergency, schedule slides. Preemption is responsible for reporting new times to all persons signed up.

K. Anyone switching to non-standard software is responsible for switching back before leaving.

L. Full compiles of compilers allowed if nothing else to run.

Clear up

Burroughs Corporation

INTER-OFFICE CORRESPONDENCE

CORPORATE UNIT Computer Systems Group	LOCATION Santa Barbara Plant	DEPT. Various
NAME Distribution		DATE 17 December 1975
FROM F. Alan Goodman, Manager	DEPT. & LOCATION Language Development	

SUBJECT:

FILE NAMING CONVENTIONS - LANGUAGE DEVELOPMENT DEPARTMENT

Distribution:

Language Development Dept.
Software Activity Managers
E. Munsch

Starting with the Mark 6.0 release, the file names from this department will conform to the following conventions with respect to SOURCE files and BACKUP files:

Valid family names

SOURCE	Source
BACKUP.PRT	Backup-listing
BACKUP.XRF	Backup-XREF
BACKUP.XMP	Backup-XMAP

Project names will be abbreviated as follows:

BAS	(BASIC)
B500	(B500)
COB	(COBOL)
FOR	(FORTRAN)
MIL	(MIL)
RPG	(RPG)
SDL	(SDL)
UPL	(UPL)
1400	(1400)

Source file names will be:

SOURCE/Project

}	Empty for compilers
	IT - for intrinsics; up to 4 characters may follow
	IN - interpreters; up to 4 characters may follow
	LB - Libraries; up to 4 characters may follow
	Escape - for programs not fitting into scheme above

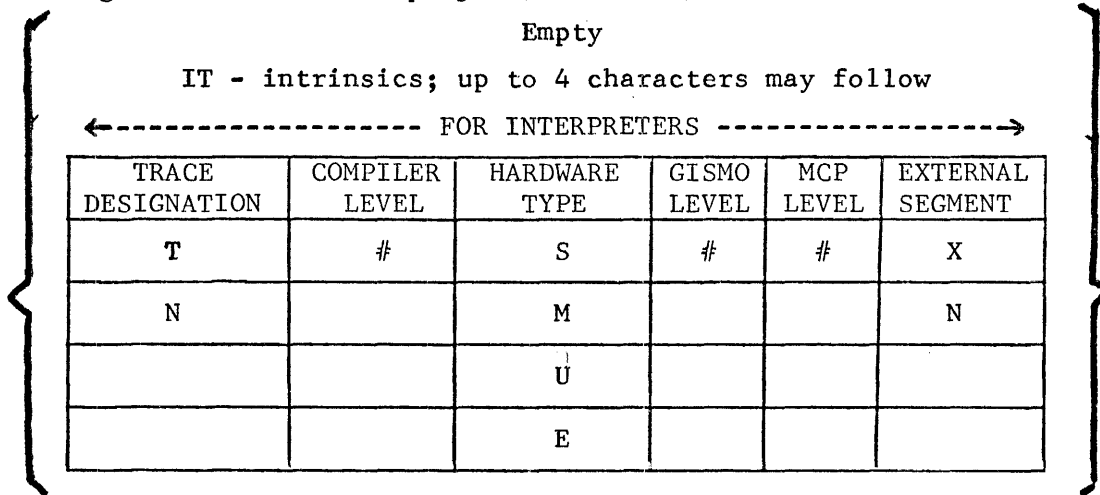
e.g. SOURCE/FOR

(FORTRAN compiler)

No Freestanding!

Backup files will be: BACKUP. PRT /Project (**)
XRF
XMP

(**The diagram below follows project)



Escape

Here "TRACE DESIGNATION" denotes whether the interpreter is a trace interpreter; "COMPILER LEVEL", "GISMO LEVEL", and "MCP LEVEL" give the level designations for the appropriate pieces of software; "HARDWARE TYPE" denotes:

- S - 1710 machine,
- M - 1720 machine,
- U - Universal to run on all 1700s,
- E - Entry level machine;

and "EXTERNAL SEGMENTS" describes whether the interpreter is segmented, (X), or not, (N).

The following files are currently contemplated as part of each MARK release.

BASIC

Source

Backup

SOURCE/BAS (BASIC Compiler)

BACKUP.XRF/BAS
BACKUP.PRT/BAS

SOURCE/BASIN (BASIC Interpreter)

(BASIC Interpreter)

BACKUP.PRT/BAST1U31N
BACKUP.PRT/BASN1S31X
BACKUP.PRT/BASN1M31X
BACKUP.XRF/BAST1U31N
BACKUP.XRF/BASN1S31X
BACKUP.XRF/BASN1M31X

SOURCE/BASITCAT
SOURCE/BASITCON
SOURCE/BASITDCT
SOURCE/BASITEOJ
SOURCE/BASITERR
SOURCE/BASITEVI
SOURCE/BASITEXP
SOURCE/BASITFAP
SOURCE/BASITFBS
SOURCE/BASITFDE
SOURCE/BASITFEM
SOURCE/BASITFMA
SOURCE/BASITFRE
SOURCE/BASITFSC
SOURCE/BASITFUN
SOURCE/BASITIDN
SOURCE/BASITINT
SOURCE/BASITINV
SOURCE/BASITIOB
SOURCE/BASITLOG
SOURCE/BASITFMT
SOURCE/BASITFOC
SOURCE/BASITMAD
SOURCE/BASITMAS
SOURCE/BASITMMY
SOURCE/BASITMOD
SOURCE/BASITMRI
SOURCE/BASITMSB
SOURCE/BASITMSM
SOURCE/BASITNUL
SOURCE/BASITPRM
SOURCE/BASITPWR

BACKUP.PRT/BASITCAT
BACKUP.PRT/BASITCON
BACKUP.PRT/BASITDCT
BACKUP.PRT/BASITEOJ
BACKUP.PRT/BASITERR
BACKUP.PRT/BASITEVI
BACKUP.PRT/BASITEXP
BACKUP.PRT/BASITFAP
BACKUP.PRT/BASITFBS
BACKUP.PRT/BASITFDE
BACKUP.PRT/BASITFEM
BACKUP.PRT/BASITFMA
BACKUP.PRT/BASITFRE
BACKUP.PRT/BASITFSC
BACKUP.PRT/BASITFUN
BACKUP.PRT/BASITIDN
BACKUP.PRT/BASITINT
BACKUP.PRT/BASITINV
BACKUP.PRT/BASITIOB
BACKUP.PRT/BASITLOG
BACKUP.PRT/BASITFMT
BACKUP.PRT/BASITFOC
BACKUP.PRT/BASITMAD
BACKUP.PRT/BASITMAS
BACKUP.PRT/BASITMMY
BACKUP.PRT/BASITMOD
BACKUP.PRT/BASITMRI
BACKUP.PRT/BASITMSB
BACKUP.PRT/BASITMSM
BACKUP.PRT/BASITNUL
BACKUP.PRT/BASITPRM
BACKUP.PRT/BASITPWR

BASIC - Continued

Source

SOURCE/BASITREP
SOURCE/BASITRIA
SOURCE/BASITRND
SOURCE/BASITRNI
SOURCE/BASITSCN
SOURCE/BASITSGN
SOURCE/BASITSQR
SOURCE/BASITSTR
SOURCE/BASITTRG
SOURCE/BASITTRM
SOURCE/BASITTRN
SOURCE/BASITVAL
SOURCE/BASITZER
SOURCE/BASITPRT

Backup

BACKUP.PRT/BASITREP
BACKUP.PRT/BASITRIA
BACKUP.PRT/BASITRND
BACKUP.PRT/BASITRNI
BACKUP.PRT/BASITSCN
BACKUP.PRT/BASITSGN
BACKUP.PRT/BASITSQR
BACKUP.PRT/BASITSTR
BACKUP.PRT/BASITTRG
BACKUP.PRT/BASITTRM
BACKUP.PRT/BASITTRN
BACKUP.PRT/BASITVAL
BACKUP.PRT/BASITZER
BACKUP.PRT/BASITPRT

B500

Source

SOURCE/B500IN
SOURCE/B500IEP (Interpreter environment program)

Backup

BACKUP.PRT/B500IEP
BACKUP.XRF/B500IEP
BACKUP.PRT/B500T1U31X
BACKUP.XRF/B500T1U31X
BACKUP.PRT/B500T1U31N
BACKUP.XRF/B500T1U31N

COBOL

Source

SOURCE/COB (COBOL Compiler)
SOURCE/COBLB (COBOL Library)

Backup

BACKUP.PRT/COB

SOURCE/COBIN

(COBOL Intrepreters)

BACKUP.PRT/COBN1S31X
BACKUP.PRT/COBN1M31X
BACKUP.PRT/COBT1U31N

(COBOL Micr Interpreters)

BACKUP.XRF/COBN1S31X
BACKUP.XRF/COBN1S31X
BACKUP.XRF/COBN1U31X
BACKUP.PRT/COBMICR1S
BACKUP.PRT/COBMICR1M

FORTRAN

Source

Backup

SOURCE/FOR (FORTRAN Compiler)
SOURCE/FORIT (FORTRAN Intrinsic source)
SOURCE/FORITMAK (Intrinsic Maker)

BACKUP.PRT/FOR (FORTRAN Compiler)
BACKUP.PRT/FORIT (Intrinsic Listing)
BACKUP.PRT/FORITMAK (Intrinsic Maker)

SOURCE/FORIN (Interpreter Source)

BACKUP.PRT/FORT1U31N (FORTRAN Interpreter)
BACKUP.PRT/FORN1S31X
BACKUP.PRT/FORN1M31X

BACKUP.XRF/FORT1U31N
BACKUP.XRF/FORN1S31X
BACKUP.XRF/FORN1M31X

MIL

Source

Backup

SOURCE/MIL

BACKUP.PRT/MIL

RPG

Source

SOURCE/RPG (RPG Compiler)

SOURCE/RPGIN (RPG Interpreters)

SOURCE/RPGBTF (Build tag file
for IS updates)
SOURCE/RPGREORD

Backup

BACKUP.PRT/RPG (RPG Compiler)

BACKUP.PRT/RPGT1U31N (RPG Interpreters)
BACKUP.PRT/RPGN1S31X
BACKUP.PRT/RPGN1M31X

BACKUP.XRF/RPGT1U31N
BACKUP.XRF/RPGN1S31X
BACKUP.XRF/RPGN1M31X

BACKUP.PRT/RPGBTF

BACKUP.PRT/RPGREORD

SDL/UPL

Source

SOURCE/SDLUPL (SDL/UPL Source)

SOURCE/SDLIN (SDL Interpreter)

Backup

BACKUP.PRT/SDL (SDL)
BACKUP.PRT/UPL (UPL)

BACKUP.PRT/IT00
BACKUP.PRT/IT01 (Intrinsics)
BACKUP.PRT/IT02
BACKUP.PRT/IT03
BACKUP.PRT/IT04
BACKUP.PRT/IT05
BACKUP.PRT/IT06
BACKUP.PRT/IT07
BACKUP.PRT/IT08
BACKUP.PRT/IT09
BACKUP.PRT/IT10
BACKUP.PRT/IT11
BACKUP.PRT/IT12
BACKUP.PRT/IT13
BACKUP.PRT/IT14

SDL/UPL - Continued

Source

Backup

BACKUP.PRT/SDLUPLXRF (XREF PGM)
BACKUP.PRT/SDLUPLXMP (XMAP PGM)
BACKUP.PRT/SDLT1U31N (Interpreters)
BACKUP.PRT/SDLN1U31X
BACKUP.PRT/SDLN1M31X
BACKUP.PRT/SDLN1U31N
BACKUP.PRT/SDLN1E31X
BACKUP.PRT/SDLIN.MCPI (MCPI Special Interp.)
BACKUP.PRT/SDLTC.MCPI

1400

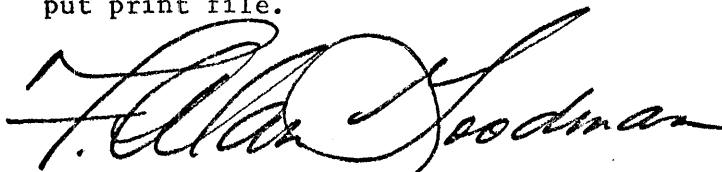
Source

Backup

SOURCE/1400IV
SOURCE/1400 IEP

BACKUP.PRT/1400IEP
BACKUP.XRF/1400IEP
BACKUP.PRT/1400T1U31X
BACKUP.PRT/1400T1U31N
BACKUP.XRF/1400T1U31X
BACKUP.XRF/1400T1U31N

It should be noted that separate BACKUP.XRF listings for each interpreter might disappear at such time that MIL includes XREF listings with the output print file.



F. Alan Goodman, Manager
Language Development

Burroughs
Program
Product
Conversion
Aid
Specifications

IBM 1401, 1440, 1460 Free Standing Emulator on the B 1720

I 14

Creates IBM 1401, 1440, 1460 Environment

STYLE NUMBER

CS 1720 I14

CAPABILITY

The IBM 1401, 1440, 1460 Emulator simplifies the transition between the IBM 1401, 1440, 1460 and Burroughs Small Systems. The Emulator creates an IBM 1401, 1440, 1460 environment within the B 1720 memory. This permits the direct execution of 1401, 1440, 1460 object programs.

Emulation of IBM 1401, 1440, 1460 programs provides an excellent interim conversion solution until these programs can be redesigned to take full advantage of the advanced architecture of Burroughs B 1720 Systems. This capability allows emulation without incurring a peak workload and without duplicate machine costs normally associated with conversion.

The IBM 1401, 1440, 1460 Emulator supports most 1401, 1440, 1460 functions and all standard peripheral input/output devices except MICR, data communications and paper tape devices. Additionally, programs specifically written for 4 K 1401, 1440, 1460 systems that are dependent on Address Register settings after execution, may not be emulated correctly.

GENERAL FEATURES

- Provides the same operating environment as defined for the IBM 1401, 1440, 1460 object program.

- Maintains the arithmetic, logical and data movement logic of the original system.
- Accommodates indexing.
- Accommodates sense switches.
- Accommodates process overlap feature.
- Accommodates disk emulation.
- Accommodates emulation of 4 K, 8 K, 12 K, or 16 K character core memory systems.

Emulated Instructions

General

The following instructions are executed by the IBM 1401, 1440, 1460 Emulator:

- Input/Output
- Arithmetic
- Logic
- Move and Load
- Magnetic Tape
- Disk
- Console Printer
- Miscellaneous

Input/Output

The following Input/Output instructions are emulated:

- Read Card (and Branch) — R
- Write Line (and Branch) — W
- Write Word Marks (and Branch) — WM
- Write and Read (and Branch) — WR
- Punch Card (and Branch) — P
- Read and Punch (and Branch) — RP
- Write and Punch (and Branch) — WP
- Write, Read and Punch (and Branch) — WRP
- Start Read Feed (same as NOP) — SRF
- Start Punch Feed (same as NOP) — SPF

- Write Disk Track Sectors with Addresses and Word Marks – WDTW
- Write Disk Sectors with Word Marks – WDW

Console Printer

A maximum of 60 characters is allowed during a Console Printer Read instruction. The IBM 1401, 1440, 1460 Emulator emulates the following Console Printer instructions:

- Read Console – MU-R
- Write Console – MU-W
- Read Console with Word Marks – LU-R
- Write Console with Word Marks – LU-W

Miscellaneous

The following Miscellaneous instructions are emulated:

- Compare – C
- Control Carriage on Printer – CC
- Clear Storage (and Branch) – CS
- Clear Word Mark – CW
- Halt (and Branch) – H
- Modify Address – MA
- No Operation – NOP
- Store A-Address Register – SAR
- Store B-Address Register – SBR
- Overlap On, Off, Reset (and Branch) (same as NOP) – SS
- Select Stacker Pocket (and Branch) – SS. Punch stacker number four goes to auxiliary; otherwise – NOP
- Set Word Mark – SW

PRODUCT IDENTIFICATION

Program ID	Description	Media	Support Category
EM1400	IBM 1401, 1440, 1460 Emulator	Object Code	A*

SYSTEM REQUIREMENTS

Central Processor:	B 1720 central processor with console printer
Memory:	4 KB M-memory and 48 KB of Main Memory
Peripherals:	Similar I/O devices to execute IBM 1401, 1440, 1460 program requirements
Disk:	At least one disk subsystem



* Support Category A indicates a "Supported Licensed Program" as defined in the Program Products License.

Arithmetic

The IBM 1401, 1440, 1460 Emulator emulates the following Arithmetic instructions:

- Add – A
- Subtract – S
- Zero and Add – ZA
- Zero and Subtract – ZS
- Multiply – M
- Divide – D

Logic

- Branch Unconditional – B
- Branch If Equal Compare – BE
- Branch If High Compare – BH
- Branch If Low Compare – BL
- Branch If Unequal Compare – BU
- Branch If Arithmetic Overflow – BAV
- Branch If Carriage Channel No. 9 (same as NOP) – BC9
- Branch If Carriage Channel No. 12 – BCV
- Branch If Character Unequal – BCE
- Branch Bit Equal – BBE
- Branch If End of Reel – BEF
- Branch If Tape Error – BER
- Branch on I/O Check Stops or Busy (same as NOP) – BIN
- Branch after Console Write – BIN-Q
- Branch If Last Card Switch On – BLC
- Branch If Sense Switch B (C-G) On – BSS
- Branch on Word Mark and/or Zone Condition – BWZ
- Branch If Access Inoperable (same as NOP) – BIN
- Branch If Wrong Length Record – BIN
- Branch If Unequal – Address Compare – BIN
- Branch If Any Disk Condition – BIN
- Branch If Disk Access Busy – BIN

Move and Load

The IBM 1401, 1440, 1460 Emulator emulates the following Move and Load instructions:

- Load Characters to a Word Mark – LCA
- Move Numeric – MN

- Move Zone – MZ
- Move Characters and Edit – MCE
- Move Characters to Record Mark or Group-Mark Word Mark – MCM
- Move Characters and Suppress Zeros – MCS
- Move Characters to A or B Word Mark – MCW

Magnetic Tape

The following Magnetic Tape instructions (for both 7-track and 9-track tape) are emulated:

- Back Space Tape Record – BSP
- Read/Write Tape in Overlap – MU
- Read Tape – RT
- Read Tape with Word Marks – RTW
- Rewind Tape – RWD
- Rewind Tape and Unload – RWU
- Skip and Blank Tape – SKP
- Write Tape – WT
- Write Tape with Word Marks – WTW
- Write Tape Mark – WTM
- Diagnostic Read – CU

Disk

The following Disk instructions are emulated:

- Read Disk Sectors – RD
- Read Disk with Sector Count Overlay – RDCO
- Read Disk with Sector Count Overlay with Word Marks – RDCOW
- Read Disk Track Sectors with Addresses – RDT
- Read Disk Track Sectors with Addresses and Word Marks – RDTW
- Read Disk Sector with Word Marks – RDW
- Seek Disk – SD
- Write Disk Sectors – WD
- Write Disk Check – WDC
- Write Disk with Sector Count Overlay – WDCO
- Write Disk with Sector Count Overlay with Word Marks – WDCOW
- Write Disk Check with Word Marks – WDCW
- Write Disk Track Sectors with Address – WDTA

Burroughs



Kerney

-7 MAY 1971

TO: RELGEN Users
FROM: Bill Firestone

Attached is updated information on RELGEN and a description of a related program, RELGEN/MASTER. The most important change is the choice of entering RELGEN parameters either from cards, as in previous versions, or from the SPO.

R E L G E N

RELEASE TAPE GENERATOR

VERSION 1.6

RELGEN IS A UTILITY PROGRAM USED TO READ AND REPRODUCE INFORMATION ON A "RELEASE" TAPE. THE INFORMATION IS ASSUMED TO BE ON A RELEASE TAPE GENERATED BY THE MIL COMPILER OR A MASTER RELEASE TAPE AS OUTPUT BY RELGEN/MASTER. TO REPRODUCE INFORMATION ON THE TAPE, PARAMETER CARDS ARE INPUT TO THE PROGRAM. VALID PARAMETERS AND THEIR MEANINGS ARE:

- LIST(LI) ----- GENERATE A LISTING TO THE PRINTER
- PUNCH(PU) ----- PUNCH THE OBJECT CODE
- COPY(CO) ----- COPY THE RELEASE TAPE TO A NEW TAPE
- SOURCE(SO) --- COPY SOURCE FILE TO DISK FILE "SOURCE"
- OBJECT(OB) --- COPY THE CODE FILE TO DISK FILE "CODE"
- MASTER(MA) --- THE INPUT TAPE IS A MASTER RELEASE TAPE (IF USED, THIS MUST BE ON THE FIRST CARD AND MUST BE THE FIRST PARAMETER ON THE CARD)
- MCOPY(MC) ----- COPY AN ENTIRE MASTER RELEASE TAPE TO A NEW TAPE
- FILE(FI) <FILE.NAME> --- FOR A MASTER RELEASE TAPE, ALL SUBSEQUENT REFERENCES ARE TO FILE <FILE.NAME>

PARAMETERS MAY BE ENTERED EITHER FROM CARDS OR FROM THE SPO. RELGEN ASSUMES SPO INPUT UNLESS THE USER INCLUDES A FILE STATEMENT:

?FILE SPEC NAME = <ANY.FILE.NAME>

AT EXECUTION TIME IN WHICH CASE INPUT IS FROM A CARD FILE CALLED <ANY.FILE.NAME>. IF INPUT IS FROM THE SPO, THE PROGRAM GOES TO EOJ WHEN A BLANK LINE IS ENTERED FROM THE SPO:

<MIX.NO>AX

FOR EXAMPLE, THE DECK

- ?EX RELGEN FILE SPEC NAME CARDS
- ?DATA CARDS
- COPY LIST PUNCH
- ?END

WOULD COPY THE RELEASE TAPE TO A NEW TAPE AND CAUSE BOTH A SOURCE LISTING AND OBJECT DECK TO BE PRODUCED. IF MULTIPLE COPIES OF INFORMATION ON A RELEASE TAPE ARE DESIRED, A NEW PARAMETER CARD MUST BE USED FOR EACH ONE. FOR EXAMPLE, THE DECK

*
*
* ?EX RELGEN FI SPEC NAME CARDS
* ?DATA CARDS
* LIST
* PUNCH LIST
* ?END

*
* WOULD PRODUCE TWO SEPARATE LISTINGS AND ONE PUNCHED OBJECT
* DICK. IF A TAPE CONTAINS A MASTER RELEASE TAPE (AS OUTPUT
* BY RELGEN/MASTER) THE FOLLOWING WOULD PRODUCE LISTINGS OF
* FILES "COBOL/INTERP" AND "SDL":

*
* ? EX RELGEN FI SPEC NAME SPEC
* ?DATA SPEC
* MASTER FILE COBOL/INTERP LIST
* FILE SDL LIST
* ?END

*
* THE FOLLOWING FILES ARE USED BY RELGEN:
* SPEC ----- PARAMETER CARD INPUT ONLY IF GIVEN IN AN
* MCP FILE STATEMENT; ELSE INPUT IS FROM
* THE SPO
* TAPE ----- RELEASE TAPE FROM MIL OR RELGEN/MASTER
* NEWTAPE --- TAPE GENERATED BY RELGEN
* PRINTER --- LISTINGS GENERATED BY RELGEN
* PUNCH ----- PUNCHED OBJECT DECK GENERATED BY RELGEN
* CODE ----- OBJECT CODE DISK FILE CREATED BY RELGEN
* SOURCE ----- SOURCE CODE DISK FILE CREATED BY RELGEN

RELGEN/MASTER

VERSION 1.0

RELGEN/MASTER IS USED TO CREATE A MASTER RELEASE TAPE FROM ONE OR MORE (A MAXIMUM OF 16) RELEASE TAPES AS PRODUCED BY MIL. THE PROGRAM READS CARDS FROM FILE "CARDS", RELEASE TAPES FROM "RELEASE" AND WRITES A MASTER RELEASE TAPE TO A TAPE FILE WITH MULTIFILE, ID = "RELEASE." INPUT ON FILE CARDS CONTAINS INFORMATION ABOUT THE TAPES TO BE READ. INFORMATION IS PROVIDED, ONE 80 COLUMN CARD PER RELEASE TAPE. EACH CARD HAS THREE KEYWORD PARAMETERS, ALL REQUIRED. THEY MAY APPEAR IN ANY ORDER, SEPARATED BY ONE COMMA AND/OR ONE OR MORE BLANKS. EACH KEYWORD IS SEPARATED FROM ITS VALUE BY AN EQUAL SIGN AND/OR ONE OR MORE BLANKS. THE THREE KEYWORDS ARE:

NAME = <FILE.IDENTIFIER> (WHERE <FILE.IDENTIFIER> MAY INCLUDE A <FILE.ID> OR A <MULTI.FILE.ID> / <FILE.ID>)

REL.LEVEL = <4 CHARACTER RELEASE LEVEL>

PROG.LEVEL = <4 CHARACTER PROGRAM LEVEL>

THE INFORMATION CONTAINED ON THE CARDS IS MAINTAINED IN A DIRECTORY AS THE FIRST FILE OF THE MASTER RELEASE TAPE.

AFTER READING ALL OF THE CARDS, THE PROGRAM WILL ISSUE SPO MESSAGES REQUESTING THE OPERATOR TO PROVIDE EACH OF THE RELEASE TAPE SPECIFIED ON THE CARDS.

EXAMPLE

?EX RELGEN/MASTER

?DATA CARDS

NAME = COBOL/INTERP314 REL.LEVEL=0331,PROG.LEVEL=0401

REL.LEVEL= 0330, PROG.LEVEL =1709 , NAME SDL/INTERP

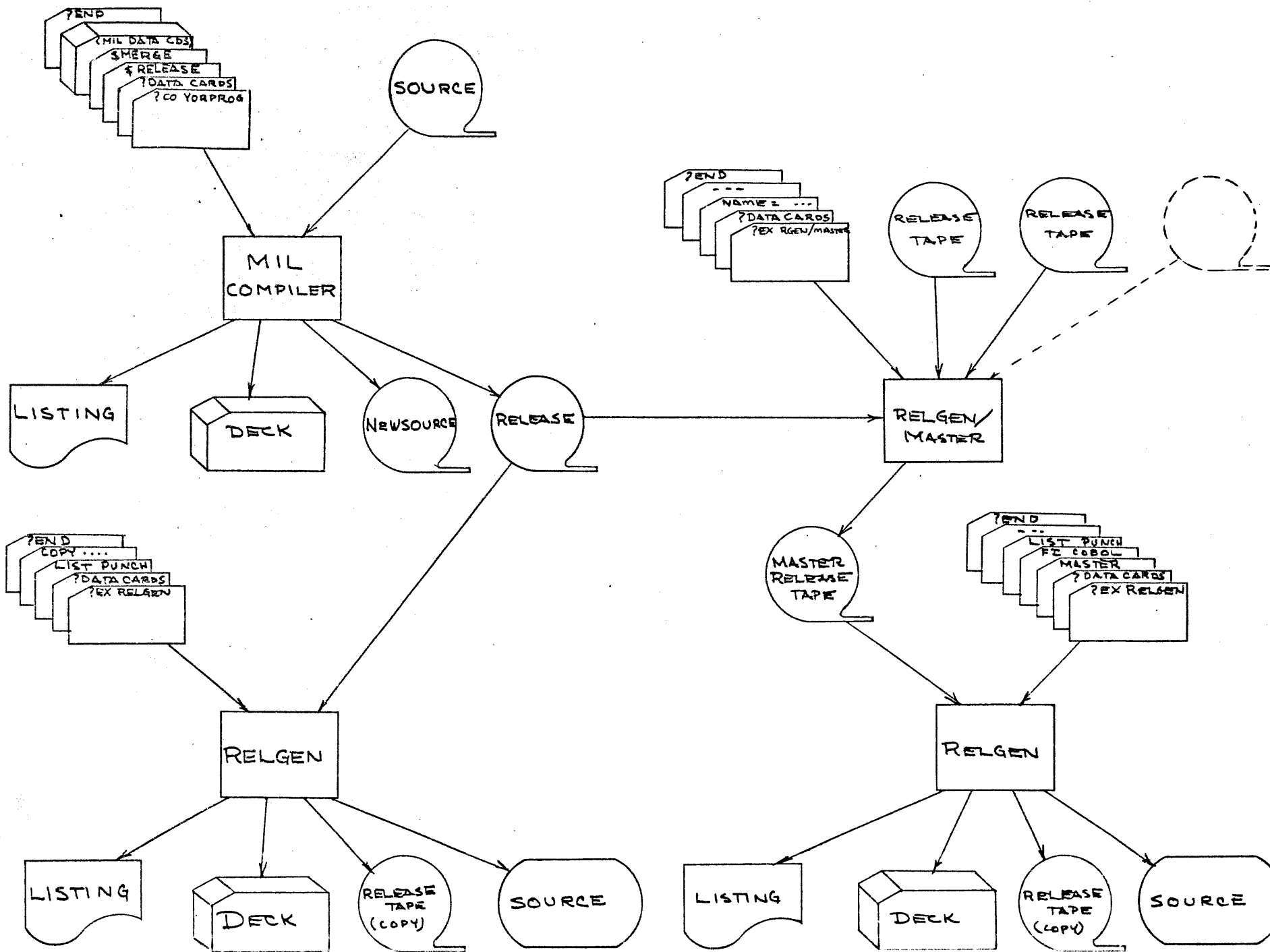
?END

FILES USED BY RELGEN/MASTER

CARDS ---- FILE INFORMATION INPUT TO RELGEN/MASTER

TAPE ----- FILE FROM WHICH INDIVIDUAL RELEASE TAPES ARE READ

RELEASE -- WILL CONTAIN THE GENERATED MASTER RELEASE TAPE



DISK CARTRIDGE INITIALIZER

GENERAL

The DISK CARTRIDGE INITIALIZER must be used to initialize a disk cartridge before it can be used for the system software. The initializer assigns addresses to the appropriate segments and writes a random pattern in the segments to ensure the segment is not defective. When a segment is found to be bad the entire track in which it resides will be removed from the disk available table on that disk cartridge. If a segment is found to be bad within the first 64 segments of a disk cartridge, that cartridge cannot be used due to the system requirements.

OPERATION INSTRUCTIONS

The disk initializer program does not operate under the control of the MCP and must be loaded and executed through the cassette reader on the control panel.

Information will be supplied to the initializer through the SPO as to which cartridges are to be initialized, label name, type, etc. There must be information entered for each disk cartridge to be initialized. To stop the Disk Cartridge Initializer enter a blank after the last cartridge has been initialized.

The following messages will be displayed:

1. WHICH CARTRIDGE? - DC<X> OR LEAVE BLANK TO TERMINATE
2. VERIFICATION ONLY? - <YES OR NO>
3. ENTER 6 DIGIT SERIAL NUMBER
4. ENTER PACK.ID
5. ENTER CARTRIDGE TYPE - <V, S, OR R>
6. ENTER JULIAN DATE - <YYDDD>
7. ENTER OWNER'S NAME

DISK PACK INITIALIZER

GENERAL

The DISK PACK INITIALIZER must be used to initialize a disk pack before it can be used for the system software. The initializer assigns addresses to the appropriate segments and writes a random pattern in the segments to ensure the segment is not defective. When a segment is found to be bad, it will be relocated into a spare segment. If more than five segments are bad in the same cylinder then the remaining bad segments will be removed from the disk available table on that pack. If six bad segments are found to be bad within the first 64 then the pack cannot be used due to system requirements.

OPERATION INSTRUCTIONS

The disk initializer program does not operate under the control of the MCP and must be loaded and executed through the cassette reader on the control panel.

Information will be supplied to the initializer through the card reader as to which cartridges are to be initialized, label name, type, etc. There must be one input card for each disk cartridge to be initialized followed by an ?END card. The following is a description of the input card.

<u>Card Columns</u>	<u>Description</u>
1	Drive character (A-P)
2	V = Verify only bInak = initialize and verify
3-8	Disk cartridge serial number
10-19	Label
21	TYPE S = System U = Unrestricted R = Restricted
23-27	Julian date (YYDDD)
29-42	Remarks (Owner's name)

QF, QP

The Program Parameter Parser of the Control Language Processor in the MCP has been extended slightly in the MCP II 4.1 version.

"CO", "EX", "BI", "MO", "DY" control statements permit modification of Program parameters. "QF" and "QP" (standing for, but not substitutable by "QUERYFILE" and "Query Program") permit interrogation of Program Parameters.

QF<CODE.FILE.NAME><PROGRAM.PARAMETER.INTERROGATION.STATEMENT.LIST> will interrogate mother copy parameters.

QP<JOB.NUMBER><P.P.I.S.L> will interrogate working (log) copy parameters.

The <P.P.I.S.L> is much like the <P.P.MODIFICATION.S.L> employed in the five PROGRAM.PARAMETER.MODIFICATION.STATEMENTS, except: where a P.P.MODIFICATION was achieved by using a "NAME/VALUE" pair (e.g., INTERP=GLUMPH), the P.P.INTERROGATION will use only the NAME of the parameter. (e.g. "INTERP")

Where a P.P.MODIFICATION used only a value (e.g. "DISK"), the P.P.INTERROGATION used ANY POSSIBLE VALUE (e.g. "PAPER.TAPE.PUNCH").

Examples:

QF<FN> IN PR FIQ NAME DISK; ID INTERP

Will cause to be typed on the SPO (with appropriate preambles, etc.):

1. INTERP NAME
2. RUN TIME PRIORITY
3. FILE Q'S EXTERNAL NAME
4. FILE Q'S DEVICE TYPE
5. NAME OF INTRINSIC DIRECTORY
6. INTERP NAME (AGAIN)

as represented in the mother copy of code file "FN" on the disk.

"QP<JOB.NO>" will do the same thing but with respect to a "working copy" of program parameters associated with the particular instance of the CODE-File's execution.

Note:

Out of contest interrogations (e.g. "QFN LI") will cause the MCP to complain.

It makes sense to interrogate certain parameters which it does not make sense to modify. (e.g. "COMPILER NAME") hopefully the syntax to provide these will be added soon (but I doubt in 4.1).

Burroughs Corporation

INTER-OFFICE CORRESPONDENCE

TO :

CORPORATE UNIT Computer Systems Group		LOCATION Santa Barbara	DEPT.
NAME Programming Activity		DATE October 7, 1974	
FROM Ben Patterson		DEPT. & LOCATION Operating Systems	
SUBJECT:		C.C.	

HARDWARE TYPES

Due to resent changes, I am publishing an up-to-date list of Hardware Types:

<u>DEVICE</u>	<u>HDWR TYPE</u>
80 Col. READER-PUNCH-PRINTER	01
80 Col. PUNCH	02
96 Col. READER-PUNCH	03
MFCU	04
96 Col. READER-PUNCH-PRINTER	05
PAPER TAPE READER	06
PAPER TAPE READER-1	07
PRINTER	08
READER SORTER-2	09
READER SORTER	10
DISK FILE (ANY HEAD-PER-TRACK DFC-1, DFC-3)	11
DFC-1 (1C, 1A, SYSTEM MEMORY)	12
DPC-3 (DCC-2)	13
DPC-1 (DCC-1)	14
DPC-2 (DISK PACK)	15
DISK-PACK (DPC-1, DPC-2, DPC-3)	16
DISK (ANY DISK)	17
DFC-3 (5-N, 5-R)	18
96 Col. READER	19
PAPER TAPE PUNCH	20
80 Col. READER	21
SPO	22
TAPE 9 TRK. NRZ	24
TAPE 7 TRK.	25
TAPE PHASE ENCODED	26
TAPE (ANY AVAILABLE TAPE)	27
TAPE (ANY 9 TRK. OR PHASE ENCODED)	28
CASSETTE	30
QUEUE FILE	62
REMOTE	63

B. Patterson
Operating Systems

jg

8 OCT 1974