DISTRIBUTION LIST

B1700 SOFTWARE PRODUCT SPECIFICATIONS

Detroit — Single Copy

J. Cox - Prod. Mgmt.
B. Gould - International
H. R. Hayde - International
D. Kosinski - Prod. Mgmt.
K. Stokes - International
J. Lambke - BMG
W. Varns - BMG
L. Atkins - BMG

D. Hill - TC, BM & SS
V. Morton - GPS, BM & SS

J. Shifman - CSG
C. Nash - International
J. G. Cleary - SSG
~~T. J. Allison~~ - Int'nl F. E. P.E. PLEMING
G. Parchinski - BMG

U.S. and Europe — Single Copy

D. O. Calkins (Plymouth)
D. R. Bookwalter (Plymouth)
[illegible] Berta (Downingtown)
K. Minarcik (Paoli)
G. Smolnik (Paoli)
A. Kosla (McLean)
A. Lacaneta - F&SSG (McLean)
B. Bell (Wayne)
L. DeBartello (Irvine)
R. Solt (Pasadena)
H. M. Townsend (Pasadena)
D. B. Prout - Pat. Atty. (Pasadena)
E. Sweaney (Mission Viejo)
J. J. Dowling (Westlake)

J. C. Allan (Glenrothes)
W. McKee (Cumbernauld)
I. J. Carradine (Cumbernauld)
Mgr, NPSGrp (Ruislip)
P. R. Evans (Middlesex)
J. Gerain (Pantin)
A. Isola (Gennevieliers)
P. Cornil (Seneffe)
J. C. Wery (Liege)
A. W. Fell (Liege)
R. Bouvier (Liege)
J. Cazanove (Villers)
B. Hammersley (Croydon)

Santa Barbara Plant — Single/Multiple

R. S. Bunker ✓
J. Hale
~~A. Goodman~~ R. Shobe
K. Meyers
R. Bauerle
A. van der Linden
E. Yardi
Quality Assurance
[illegible] Ross-Smith

E. Munsch - 2
G. Hammond - 2
J. Casey - 2

Distribution current as of ~~3/21/77~~ 5/20/77.

Burroughs Corporation

COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

P.S. 2212 5280

BASIC COMPILER

PRODUCT SPECIFICATION

REV LTR	REVISION ISSUE DATE	APPROVED BY	REVISIONS	
A	1/23/75	Hale	ORIGINAL ISSUE	
B	1/20/76	Hale	MAJOR REVISION: Supports restructuring of MARK V.1 Basic Compiler	
C	6/10/77	Hale	MARK V1.1 CHANGES:	
			1-1	References to B1700 changed to B1800/B1700
			1-1	Statement maximum length changed from eighty to 243 characters
			1-3	Memory requirements: 8K bytes of memory changed to 10K
			2-1 to 2-3	All references to INTRN2 changed to INTRN3. Added BAS.INTRN3/#MRM, BAS.INTRN3/#RDM, BAS.INTRN3/#WRM, BAS.INTRN3/#WRT

"THE INFORMATION CONTAINED IN THIS DOCUMENT IS CONFIDENTIAL AND PROPRIETARY TO BURROUGHS CORPORATION AND IS NOT TO BE DISCLOSED TO ANYONE OUTSIDE OF BURROUGHS CORPORATION WITHOUT THE PRIOR WRITTEN RELEASE FROM THE PATENT DIVISION OF BURROUGHS CORPORATION"

TABLE OF CONTENTS

GENERAL

The B1800/B1700 BASIC Compiler program generates, from valid BASIC source statements, object code files and S-instructions that can be processed by the BASIC interpreter. After a brief preliminary discussion of the compiler's mode of operation, this product specification discusses the essential operating requirements of the compiler: file structure, memory size, $-card options, and intrinsics.

Legitimate statements in BASIC are defined in the Burroughs publication: Burroughs Standard BASIC (CSG). A BASIC statement consists of a line number followed by a verb followed by the remainder of the statement, e.g., 100 LET A = B + C. The statement begins in column one and its maximum length is 243 characters. Code files are discussed, in general, in the product specifications for MCPI and MCPII, and BASIC S-instructions are described in product specification #2210 0135, B1800/B1700 BASIC S-language.

RELATED PUBLICATIONS

NAME	NUMBER
Burroughs BASIC Language Standard (CSG)	1955 2783
B1800/B1700 BASIC S-language	P.S. 2210 0135
MCPII	P.S. 2212 5462
Software Operational Guide	1068731

STRATEGY OF COMPILATION

After performing the necessary initialization of various tables and variables, the compiler enters a loop that processes all of the source statements in the input file in three procedure calls. The first procedure reads the statements and associates the line numbers and their current locations in the object code by pushing both their values into a stack. If the list option is on and compilation was not initiated from a remote terminal, the statement and the current location in the object code is printed. The second procedure gets the verb (or keyword) that introduces the statement. The third procedure contains a set of procedure calls, one for each of the possible BASIC statements.

When the last source statement has been processed, some clean-up work, e.g., filling in unresolved branch addresses and checking for unfinished function definitions and for-loops, is done. If there were no compilation errors, a code file is built. The code file contains the Program Parameter Block, Run Structure, and the other features that the MCP requires to run the program. The code file is written onto disk and compilation terminates. If there were compilation errors, no code file is built.

ERROR CONDITIONS

The list of error conditions and associated messages are found in the Burroughs BASIC Language Standard.

COMPILER FILES

During compilation, the following three files are used:

INTERNAL FILE NAME	EXTERNAL FILE IDENTIFIER	USE
CARDS	CARDS	Source statements
LINE	LINES	Output listing
CODE	CODE	Compiled S-code

MEMORY REQUIREMENTS

In its released version, the compiler requires 10K bytes of memory.

PERFORMANCE

The performance of BASIC object programs which use arrays or character string manipulation and which run with the state light on can usually be improved greatly by increasing the amount of dynamic memory available for these arrays or strings. The optimum amount of memory varies depending on the size of the program, the other jobs in the mix, and the amount of memory on the system on which the program is running.

DOLLAR-SIGN CONTROL CARDS

The compiler recognizes the following $-card options, entered after the line number and separated by a space. The $-sign itself must precede the options and be separated from them by a space. For example, the correct format to request a single-spaced listing of the source statements only would be:

```
1000 $ NO CODE LIST SINGLE
```

Compiler options are as follows:

OPTION	DEFAULT	USE
CARD	ON	Documentation only.
LIST	ON	Output listing
SINGLE	ON	Single-spaced output listing
DOUBLE	OFF	Double-spaced output listing
CODE	OFF	Output compiled S-code
NO	OFF	Turns off option which follows
STRINGSPACE	8	Number of data pages available for string concatenation, input, and other operations which generate new strings. A string space data page holds 512 characters.
STACK	100	Number of words available for the numeric/string stack. The word size is 48 bits.

OBJECT PROGRAM FILES

During execution, one or more files are used as required by the program (e.g., the print file is always present). Compiler file names are:

INTERNAL	EXTERNAL	
FILE NAME	FILE IDENTIFIER	USE
PRINT	PRINT	Output from print-statement
INPUT	INPUT	Inout to input-statement
FILE01	AS DECLARED	Disk File
FILE02	AS DECLARED	Disk file
*	*	*
*	*	*
*	*	*

INTRINSICS

During execution, the following intrinsics can be called by the program to perform input, output, and mathematical functions:

BAS.INTRN3/#CAT — Concatenate two strings.

BAS.INTRN3/#CON — Constant one. Generates a numeric array, all elements are one.

BAS.INTRN3/#DCT — Date, clock and time. Called at the beginning of execution to get the start time. Also implements the functions DAT$, CLK$, TIM, CHR$, IDA, and BCL.

BAS.INTRN3/#EOJ — End of job. Called at the end of execution. Also implements the statements STOP, END, and CHAIN.

BAS.INTRN3/#ERR — Error. Handles execution errors.

BAS.INTRN3/#EXP — Antilogarithm (Base e).

BAS.INTRN3/#EVI — Evaluates image.

BAS.INTRN3/#FAP — File append.

BAS.INTRN3/#FBS — File backspace.

BAS.INTRN3/#FDE — File delimit.

BAS.INTRN3/#FEM — If end file and if more file.

BAS.INTRN3/#FMA — File margin.

BAS.INTRN3/#FMT — Format numeric and string output according to the associated image.

BAS.INTRN3/#FOC — File open and close.

BAS.INTRN3/#FRE — File restore.

BAS.INTRN3/#FSC — File scratch.

BAS.INTRN3/#FUN — File functions HPS, LIN, and VPS.

BAS.INTRN3/#IDN — Identity. Generates an identity matrix.

BAS.INTRN3/#INT	Greatest integer function.
BAS.INTRN3/#INV	Invert a matrix.
BAS.INTRN3/#IOB	Input-output begin. Initiates all I/O statements.
BAS.INTRN3/#LOG	Natural logarithm (Base e).
BAS.INTRN3/#MAD	Matrix add.
BAS.INTRN3/#MAS	Matrix assignment.
BAS.INTRN3/#MMY	Matrix multiply.
BAS.INTRN3/#MOD	MOD function.
BAS.INTRN3/#MRI	Matrix read and input.
BAS.INTRN3/#MRM	Matrix read a memory image file.
BAS.INTRN3/#MSB	Matrix subtract.
BAS.INTRN3/#MSM	Matrix scalar multiply.
BAS.INTRN3/#NUL	Null. Generates a string array, all of whose elements are null.
BAS.INTRN3/#PRM	Print a numeric or string array. Implements MAT PRINT and MAT WRITE statements.
BAS.INTRN3/#PRT	Print a numeric or string value. Implements PRINT and WRITE statements and the TAB function.
BAS.INTRN3/#PWR	Power routine (X**Y).
BAS.INTRN3/#RDM	Read a memory image file.
BAS.INTRN3/#REP	Replace. Implements the REP$ function.
BAS.INTRN3/#RIA	Read or input a numeric or string value.
BAS.INTRN3/#RND	Random number generator.
BAS.INTRN3/#RNI	Random number initializer.
BAS.INTRN3/#SCN	Scan. Implements the SCN function.

BAS.INTRN3/#SGN Sign function.

BAS.INTRN3/#SQR Square root.

BAS.INTRN3/#STR String. Implements the STR$ function.

BAS.INTRN3/#TRG Trig routines. Implements the SIN, COS, TAN, COT, and ATN functions.

BAS.INTRN3/#TRM Trim an array. Implements trimmers in MAT statements.

BAS.INTRN3/#TRN Transpose an array.

BAS.INTRN3/#VAL Value. Implements the VAL function.

BAS.INTRN3/#WRM Write a numeric or string array to a memory image file.

BAS.INTRN3/#WRT Write a numeric or string value to a memory image file.

BAS.INTRN3/#ZER Zero. Generates a numeric array, all of whose elements are zero.

INDEX