START WITH P = 1000

Honeywell INC.
COMPUTER AND COMMUNICATIONS GROUP

PROGRAM SPECIFICATION

Series 16 Basic Compiler/Interpreter

1. SCOPE:

This document describes an interactive interpretive compiler for the BASIC language.

2. APPLICABLE DOCUMENTS:

The following documents (latest revisions) form a part of this specification to the extent specified herein:

70 110 010-397, H-316 Product Specification (includes ASR control Unit)
70 110 010-527, DDP-516 Product Specification
70 110 010-523, DDP-516 ASR Control Unit Product Specification
70 130 072-442, DAP-16 and DAP-16 Ver 2 Assembler Manual

3. DEFINITIONS:

3.1 Basic

Acronym for Beginners All-purpose Symbolic Instruction Code. BASIC is an engineering and science oriented programming language.

3.2 Interpretive Compiler

A compiler which compiles and executes statements without producing object code.

4. DESCRIPTION:

The Basic Interpretive Compiler provides an interactive environment in which a user can compile, edit, and execute programs written in the BASIC language. The user may assemble an entire program and call for its execution or may enter language statements to be executed immediately. Statements are checked for syntactic and logical errors during both compile and execution. All errors are reported to the user immediately.

5. REQUIREMENTS:

5.1 Environment.

5.1.1 Required Hardware.

1 - H316 or DDP-516 Central Processor Model 316/516-01

REV AUTH	WR	REL ENG	SIZE A	CODE IDENT 07573	DOC NO
	SPEC	DES ENG	SCALE	REV	SHEET OF

1 - 4K of Core Memory
1 - ASR-33 or ASR-35 Console Typewriter Model 316/516-53/55

5.1.2 Supported Hardware.

Up to 16K of Core Memory.

5.1.3 Required Software.

None.

5.1.4 Supported Software.

Will read in and execute any program written in BASIC. A method is included for linking to programs written in FORTRAN or DAP-16.

5.1.6 Special Relationships.

This system can be configured to execute under the RTX-16 Executive or the OP-16 Operating System.

5.2 Performance.

5.2.1 Design Goals.

This compiler will have as many of its features as possible operative in a 4K version. The following features will definitely be included in the 4K version: READ, PRINT, DATA, LET, GO TO, IF-THEN, FOR-TO-STEP, NEXT, GOSUB, RETURN, END, STOP, DEF, DIM, interactive operation, and source program editing.

This compiler is designed to be upward-compatible to a multi-user configuration.

A subset of this language will also be a subset of the 1648 Time Sharing BASIC.

Input/Output will be handled by an IOS package to allow expansion to new devices.

5.2.2 Timing Considerations.

This compiler is designed to be compatible with the multi-programming environment of OP-16.

5.3 Restrictions.

All programs must be written in BASIC as described herein. A subset of this language operates properly in a computer with 4K of memory.

6. PROCEDURES:

6.1 Use.

6.1.1 Stand-Alone Version.

The stand-alone version is loaded from self-loading paper tape and executed starting at a certain location. All further communication is via the ASR console.

6.1.2 OP-16 Version.

BASIC is called from the console using the RTX Keyboard program. All further communication is directly with BASIC via the console. A "QUIT" command is provided for terminating BASIC.

6.2 Data Formats.

6.2.1 Input Data Formats.

6.2.1.1 Statement Inputs

Detailed statement formats are described under Syntax (6.2.5). Statements must be terminated by a carriage return. A line feed is optional. The sequence X-OFF, RUBOUT may optionally follow the carriage return or carriage return, line feed. A null statement has a special meaning--see Section 6.2.2.1.

6.2.1.2 Data Inputs

The data must be in the format described for numbers under Syntax (6.2.5). Data must be terminated by a carriage return. A line feed is optional. The sequence X-OFF, RUBOUT may optionally follow the carriage return or carriage return, line feed.

6.2.1.3 Input Error Correction

A leftward arrow or succession of leftward arrows may be used to delete one or a succession of preceeding characters. A commercial at (@) may be used to delete the entire line.

6.2.2 Output Data Formats.

6.2.2.1 Request for Statement Input

BASIC outputs a question mark followed by an X-ON each time it is ready to receive a statement input. If the previous statement was a null statement, the question mark is output but the X-ON is not. This feature allows orderly termination of program input from paper tape.

6.2.2.2 Request for Data Input

BASIC outputs an exclamation point (!) followed by an X-ON each time it is ready to receive data input.

6.2.2.3 Listing Output

Statements are output by order of line numbers with all spaces except those in comments and messages deleted. Numbers are output in a standard format described in Section 6.2.6. Each statement is terminated with the sequence carriage return, line feed, X-OFF, RUBOUT. The program is terminated with a null statement.

6.2.2.4 Data Output

Data are output in the standard format described in sections 6.2.6 and 6.2.7.3.

6.2.3 Error Message Formats.

6.2.3.1 Stop/End Message

Whenever a STOP or END is encountered, the line number and the word EXIT is printed. If the program terminates by executing the highest numbered statement without encountering either a STOP or an END, line number 0000 and EXIT are printed.

6.2.3.2 Error Messages

The following message is printed for each error encountered during either statement input or execution:

ERROR AA LINE BBBBB

where AA stands for an error code and BBBBB for a line number.

6.2.4 Internal Data Formats.

Programs are stored internally as compressed statements. A line number table gives the starting byte location in core of each statement.

6.2.4.1 Variable Storage

All variables are stored as standard floating-point numbers (two words per variable). This format is described in the DAP-16 Mod 2 assembler manual.

6.2.4.2 Constant Storage

Constants are store as either one-word fixed-point numbers or two-word floating-point numbers. The preceeding byte indicates whether the constant occupies one word or two.

6.2.4.3 Control Bytes

Bytes with special values between 0 and 127 are used as compressions for the standard statement types, indication of constants and variables, etc.

6.2.4.4 Alphanumeric Bytes

Bytes with USASCII significance (values between 128 and 255) are used to hold variable names, arithmetic operators, etc.

6.2.5 Statement Syntax.

The following syntax is described in Backus Normal Form. Quantities enclosed within diamond brackets (< >) are metalinguistic variables representing a class of syntactic variables. A colon followed by an equal sign (:=) means "is defined as." A vertical line (|) connecting two elements means logical OR. An element or group of elements enclosed in square brackets followed by a subscript and superscript ($[\,]_a^b$) may be repeated any number of times within the inclusive range of the subscript and superscript. All letters and symbols not enclosed in diamond brackets are actual characters of the syntax. Blanks are ignored anywhere within any BASIC statement except a comment statement or a message. Thus GOTO, GO TO, and G OT O are all equivalent.

<alphabetic character> := A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<digit> := 0|1|2|3|4|5|6|7|8|9

<special character> := +|-|*|/|↑|=|(|)|<|> | .|,|;|Δ

<integer> := $[\langle digit\rangle]_1^9$

<decimal number> := $[\langle digit\rangle]_1^{N<9}$. $[\langle digit\rangle]_0^{9-N}$

<sign> := $[+|-]_0^1$

<exponent> := E <sign> $[\langle digit\rangle]_1^2$

<number> := $[\langle integer\rangle|\langle fraction\rangle|\langle decimal\ number\rangle]_1^1$ $[\langle exponent\rangle]_0^1$

<signed number> := <sign> <number>

<simple variable> := <alphabetic character> $[\langle digit\rangle]_0^1$

<fraction> := . <integer>

<subscripted variable> := <alphabetic character>(<expression> [,<expression>] $_0^\infty$)

<variable> := <simple variable>|<subscripted variable>

<function name> := SIN|COS|TAN|ATN|EXP|ABS|LOG|SQR|INT|RND|SGN|FN<alphabetic character>

<function term> := <function name> (<expression>)

<term> := <number>|<variable>|<function term>| (<expression>)

<involution factor> := <term>|<involution factor>↑<term>

<multiply factor> := <involution factor>|<multiply factor> [* | /] $_1^1$ <involution factor>

<expression> := <multiply factor>|<sign><expression>|<expression> [+|-] $_1^1$ <involution factor>

<assignment statement> := LET <variable> = <expression>|<variable> = <expression>

<READ statement> := READ <read list>

<INPUT statement> := INPUT <read list>

<read list> := <variable> [, <variable>] $_0^\infty$

<DATA statement> := DATA <number list>

<number list> := <expression> [, <expression>] $_0^\infty$

<RESTORE statement> := RESTORE

<PRINT statement> := PRINT[<print list>] $_0^1$

<print list> := <print item> [[,|;] $_1^1$ <print item>] $_0^\infty$ [,|;] $_0^1$

<print item> := <expression>|<message>|<message><expression>| TAB (<expression>)

<message> := "[<alphabetic character>|<digit>|<special character>] $_1^\infty$"

<comment> := REM [<alphabetic character>|<digit>| <special character>] $_0^\infty$

<GOTO statement> := GOTO <line number>

<GOSUB statement> := GOSUB <line number>

<RETURN statement> := RETURN

<ON statement> := ON<expression> GOTO <line number> [,<line number>] $_0^\infty$

<line number> := [<digit>] $_1^5$

<logical IF statement> := IF <expression> <relational operator> <expression> [THEN <statement body> [: <statement body>] $_0^\infty$ | THEN <line number>|GOTO <line number>] $_1^1$

SIZE	CODE IDENT	DOC NO
A	07583	
SCALE	REV	SHEET

<arithmetic IF statement> := IF <expression>,<line number>,<line number>,<line number>

<relational operator> := >|>=|=|<=|<|<>

<FOR statement> := FOR<simple variable>=<expression>[TO | ,] $_1^1$ <expression> [[STEP | ,] $_1^1$ <expression>] $_0^1$

<NEXT statement> := NEXT <simple variable>

<STOP statement> := STOP

<END statement> := END

<DIMENSION statement> := DIM <dimension variable>[,<dimension variable>] $_0^\infty$

<dimension variable> := <alphabetic character> (<size>)

<size> := <integer> [, <integer>] $_0^\infty$

<DEFINE statement> := DEFFN <alphabetic character>(<simple variable>)= <expression>

<CALL statement> := CALL(<subroutine identifier>[,<subroutine parameter>] $_0^\infty$)

<subroutine identifier> := <expression>

<subroutine parameter> := <expression>

<command request response> := <system command>|<BASIC statement>|<line deletion>|<immediate command>

<system command> := JOB|CLEAR|RUN|LIST|CONTINUE|QUIT

<line deletion> := <line number>

<immediate command> := <immediate statement>[: <immediate statement>] $_0^\infty$

<immediate statement> := <assignment statement>|<READ statement>|<INPUT statement>|<RESTORE statment>|<PRINT statement> <GOTO statement>|<logical IF statement>| <arithmetic IF statement>|<ON statement>|<FOR statement>|<NEXT statement>|<GOSUB statement>| <RETURN statement>|<CALL statement>|<comment statement>|<STOP statement>

<statement body> := <immediate statement>|<DATA statement>|<DIMENSION statement>|<DEFINE statement>

<BASIC statement> := <line number> <statement body> [: <statement body>] $_0^\infty$

<BASIC program> := [<BASIC statement>] $_1^\infty$ <line number><END statement>

6.2.6 Numerical Output Format in Lists.

All numbers printed in program lists are printed to take

up as little room as possible. Numbers with absolute values between 0.1 and 9999 are printed in the format XXXX.YYYYYY or -XXXX.YYYYYY with leading and trailing zeros suppressed. The decimal point is suppressed for integral values. Numbers outside this range are printed in the format .XXXXXXE±ZZ or -.XXXXXE±ZZ.

6.2.7 Description of Statement Types.

6.2.7.1 Assignment Statement

When executed, the variable on the left hand side of the = sign is given the value of the expression on the right hand side.

6.2.7.2 READ Statement

When executed, a READ statement causes the next available unused data items for DATA statements to be loaded into the list elements as they are encountered, scanning from left to right. If no unused data items remain, an error message is printed and a return is made to command mode.

6.2.7.3 INPUT Statement

When executed, an INPUT statement causes an exclamation point to be typed and the program waits for the user to type in the number of data items requested in the read list. If more than one item is typed in, the items must be separated from each other by commas. If there are more items typed than there are items in the list, the extra items are lost with no error indication. If fewer are typed, BASIC responds with ! again until the list is satisfied. A data item may be a numeric value or an expression.

6.2.7.4 DATA Statement

The DATA statements in a program form a pool of data items which may be read by READ statements. Data statements are ignored when executed.

6.2.7.5 RESTORE Statement

When executed, the RESTORE statement returns the DATA pointer to the first constant in the first DATA statement.

6.2.7.6 PRINT Statement

6.2.7.6.1 Use of TAB.

The word TAB in a PRINT statement moves the print head to the position specified in the expression.

6.2.7.6.1 Use of Comma.

A comma following a message or variable in a PRINT statement moves the print head to the next position of 14, 27, 40, or 53. A comma at the end of a PRINT statement suppresses a carriage return/line feed to the next line.

6.2.7.6.3 Use of Semi-Colon.

A semi-colon following a message of variable in a PRINT statement does not move the print head at all after that message or variable is printed. A semi-colon at the end of a PRINT statement suppresses a carriage return/line feed to the next line.

6.2.7.6.4 Print Format.

Variables with an absolute value between 0.1 and 9999 are printed in the format:bXXXX.YYYYYb or -XXXX.YYYYYb.

Other variables are printed in the format: [illegible] b.XXXXXXE+ZZb or -.XXXXXXE-ZZb.

6.2.7.7 GOTO Statement

When executed, a GOTO statement causes control to be transferred to the indicated line number.

6.2.7.8 GOSUB Statement

When executed, this statement transfers control to the indicated line number and saves the line number of the GOSUB statement for use with a later RETURN. GOSUB/RETURN loops may be nested to a depth of eight.

6.2.7.9 RETURN Statement

When executed, this statement transfers control to the statement following the last executed GOSUB statement. If no GOSUB has been executed, an error return is made to command mode.

6.2.7.10 ON Statement

The expression is evaluated and truncated to an integer. For expression = 1, control is transferred to the first line number in the list, for expression = 2, control is transferred to the second line number in the list, etc. If the expression truncates to less than one or more than the number of line numbers in the list, an error return is made to command mode.

6.2.7.11 Logical IF Statement

When executed, the two expressions are evaluated and compared. If the condition specified by the relational operator is true, the rest of the line is executed. If the condition is false, control passes to the next statement.

6.2.7.12 Arithmetic IF Statement

When executed, the expression is evaluated, and control is transferred to the first line number if the expression is negative, to the second line number if it is zero, and to the third line number if it is positive.

6.2.7.13 FOR Statement and NEXT Statement

When a FOR statement is encountered, the three expressions are evaluated and their values saved. The simple variable is then given the value of the first expression and control is transferred to the following statement in line.

When a NEXT statement is encountered the simple variable is incremented by the saved value of the third expression (or 1 if the third expression was not specified). If the new value is within the inclusive range of the first two saved expressions, control is transferred to the statement following the FOR. If not, the following NEXT statement is executed. FOR/NEXT loops may be nested to any depth.

6.2.7.14 STOP Statement

When executed, a STOP statement causes the line number and the word EXIT to be printed and BASIC returns to the command mode. A program need not have any STOP statements.

6.2.7.15 END Statement

An END statement is identical to a STOP statement. A program need not have any END statements.

6.2.7.16 DIMENSION Statement

This statement establishes array sizes. It is ignored during execution. DIMENSION statements may be anywhere in a program.

6.2.7.17 DEFINE Statement

This statement establishes a one-variable function.

Functions may be defined and redefined throughout a program. The last executed definition takes precedence.

6.2.7.18 CALL Statement

This statement causes a DAP-16 CALL through a location in a CALL Table maintained by BASIC. The addresses of DAP-16 or FORTRAN subroutines to be called must be entered in this table by the user. The subroutine identifier tells BASIC which table entry to use. Arguments can be transferred to the subroutine by BASIC, using the same calling sequence used by the FORTRAN compiler. This format is described in the DAP-16 Mod 2 assembler manual.

6.2.8 BASIC System Commands:

6.2.8.1 JOB Command

This command clears the user storage space, thereby deleting any old BASIC programs, and prepares the system to accept new information. It does not affect DAP-16 or FORTRAN subroutines nor the subroutine CALL table.

6.2.8.2 CLEAR Command

This command clears all user data space, but does not delete any programs.

6.2.8.3 RUN Command

This command initializes BASIC to run the current program. Execution starts with the lowest line number in the program. The program runs until an error, a STOP, an END, or the last defined statement is reached, or a control P command is entered.

6.2.8.4 Control-P Command-- Doesn't work, use sense switch 1 SS1 to set

This command may be entered at most times while a BASIC program is running in order to return the system to command mode. The statement number to be executed next is printed on the console.

6.2.8.5 CONTINUE Command

This command resumes processing where it left off after a control P. readies SS1

6.2.8.6 LIST Command

This command causes the entire BASIC program in core to

be listed in line number order.

6.2.8.7 QUIT Command

This command is undefined in the stand-alone version of BASIC. It terminates the BASIC processor in the OS-16 version.

6.2.9 Editing Feature.

The program may be edited indefinitely from command mode. Line may be deleted by typing their numbers with no contents. Lines may be added by typing them with line numbers that establish them where they belong. Lines may be replaced by typing the line number of the line to be replaced followed by the new line. The program is effectively stored in line number order. A completed program may be saved for future use by listing it with the ASR paper tape punch turned on.

6.3 Method.

6.3.1 Program Operation.

The RUN command initializes all array dimensions as specified in DIMENSION statements and starts interpreting the program at the lowest numbered statement. Each statement is interpreted and executed as it is encountered.

6.3.2 Immediate Operation.

Any statement (except DATA, DIMENSION, or DEFINE) entered without a line number is executed immediately. Any reference to line numbers (e.g. GOTO) refers to the program. Thus, immediate statements that transfer control may be used to start execution of the program.

: If a program has been modified in any way it must be started by RUN rather than GOTO or GOSUB.

7. NOTES:

None.

8. REFERENCE DOCUMENTS:

70130072090, H-1648 Time-Sharing Users Guide

PUNCH - Same as 'LIST' but punches Time Sharing [illegible]

LOAD - Loads tape punch by 'PUNCH' command [illegible]

SIZE	CODE IDENT	DOC NO
A	07573	
SCALE NONE	REV	SHEET 12 OF 12

[illegible]

- [illegible] [illegible] PRINT A
- [illegible]
- [illegible] IF A<0 GOTO 120
- [illegible] IF A>0 THEN LET Y=SQR(A)
- [illegible]
- [illegible]
- [illegible]
- LET [illegible] X=A: Y=X↑1.5
- [illegible]
- [illegible]
- [illegible] ("[illegible] READ)
- [illegible]
  [illegible](2)
  [illegible] 14150: PRINT [illegible]2.5↑2
  [illegible] 10: PRINT I; SQR(I): NEXT I
- [illegible] 22 [illegible] [illegible]

BASIC -16: FELKODER

Code	Error (explanation)
AS	Array subscript (index wrong)
DA	Data statement missing or no more data
DF	Deleted function (ATN; SIN, COS, TAN; SQR)
DL	Delimiter (ex comma instead of period or vice versa)
DN	Dimensioned variable name (1 letter only)
DP*	Too many decimal points
DV	Dummy variable (must not be indexed)
DZ	Division by zero
FD	FOR delimiter
FN	Function definition (FN missing or no letter after FN)
IC	IF condition
ID	Identifier (Neither variable nor reserved name)
IV	Indexed variable in FOR or NEXT
GS	GOSUB more than 8 levels
LG	LOG of zero or negative number
MO*	Memory overflow (Try CLEAR if in source mode)
Mx	Match not found or missing char x
NO	Numeric overflow or underflow
NX	NEXT (No FOR statement or undefined variable)
ON	ON (No GOTO or expression ≤ 0)
PD	PRINT delimiter
RT	RETURN without GOSUB
SN	Statement number (= 0 or > 9999)
SQ	SQR of number < 0
SS	Subroutine selection (undefined subroutine number in CALL)
TH	THEN missing in IF statement
TX*	Text string unclosed (missing ")
UF	Undefined own function
UM	Unary minus (ex. - - or +- or * - or /-)
US	Undefined statement
UV	Undefined variable

* Dessa felutskrifter erhålles vid inmatning av program, MO även vid exekvering.

CALL

CALL används för att i BASIC - 16 anropa egna assembler/ Fortran - kodade subprogram som utför exvis in/utmatning av analoga/digitala värden.

Anropsformat:

CALL (Subrutinnummer, parameter, parameter, . . . parameter) där subrutinnummer (1 - 10) istället för namn anger vilken rutin som avses.

Parameter kan vara ett variabelnamn, en konstant eller ett uttryck.

CALL - processorn i '3724 - '4037 bygger upp ett fortrankompatibelt anrop i buffertarean IBUF på sektor 0.

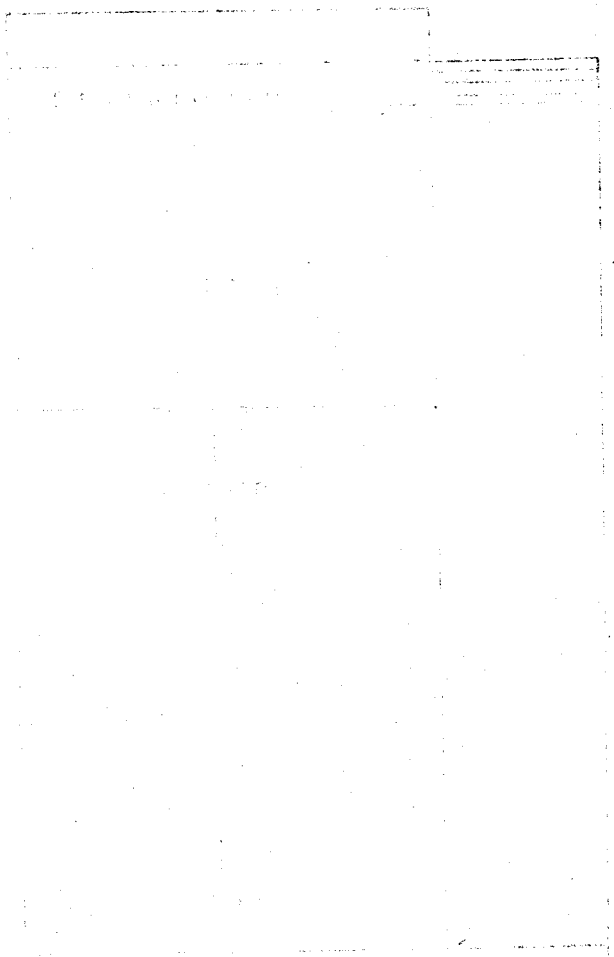
Ex.

'256	IBUF	JST *	'554 + subrutinnummer
'257		DAC	parameter 1
'260		DAC	parameter 2
		.	
		.	
		.	
		DAC	parameter n
		OCT	0 (om n > 1)
återhopp hit	⟶	JMP	'3765 (tillbaka till CALL - processorn)

I tabellen CJST i '555 - '566 skall alltså startadresser till subrutinerna läggas in.

Observera att alla parametrar är i flytande form (2 ord) dvs indata måste konverteras för att kunna användas av BASIC - 16.

CAT [illegible] – Minne layout och inkoding



[illegible] minnet med 4K loader

[illegible] 1555 och uppåt

[illegible] 0

[illegible] Snakes kan PAL-NP 6K