Instructions

for the



EXTENDED BENTON HARBOR BASIC (Version 10.02.00)

Model HC8-13

Table of Contents

ntroduction	1
ystem Data Format	2
New Command Mode Statements	4
rror Table	7
Optional Patches	8
asic Utility Routines 1	(
Intry Points To Basic Utility Routines	(

INTRODUCTION

This Manual describes the new and different functions of the Extended Benton Harbor BASIC Program (version 10.02.00). It is not meant to replace the H8 Software Reference Manual, only to supplement it. Read through this Manual to become familiar with the new commands.

It is a good idea to make notes or references to this Manual in your Software Manual to remind you of the new and different commands contained in version 10.02.00 of BASIC. Keep this Manual with the H8 Software Reference Manual.

Copyright © 1978 Heath Company All Rights Reserved Printed in the United States of America

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022



SYSTEM DATA FORMAT

Extended Benton Harbor BASIC (version 10.02.00) contains the three original file types (001, 002, and 003) plus three new file types (004, 005, and 006).

BASIC Programs using Extended Benton Harbor BASIC version 10.01.00.
Compressed Text.
BASIC Programs using Extended Benton Harbor BASIC version 10.02.00.
BASIC Data using Extended Benton Harbor BASIC version 10.02.00.
BASIC Programs and Data using Extended Benton Harbor BASIC version 10.02.00.

The three original file types (001, 002, and 003), which are explained in the "Tape Files" section of the H8 Software Reference Manual, do not change. The three new file types are explained in the following paragraphs.

BASIC PROGRAMS (Type = 004)

001

Memory Image.

This file type is used by BASIC (version 10.02.00) when you load and dump **programs**. It is the same as file type 002 except that it uses a different format to dump and load the program. This file always has a label record (#0), a table record (#1) that contains a table describing what data has been stored, and a data record (#2) containing the actual data (program text).



BASIC DATA (Type = 005)

This file type is the same as file type 004 except that it is used to load and dump data (program variables).

BASIC PROGRAM AND DATA (Type = 006)

This file type is the same as file type 004 except that it is used to load and dump both the **program text** and **program variables** together.

READING THE DISPLAYS

When the H8 computer is reading or writing data on a tape transport, the front panel displays are continually displaying data about the tape operation. The information contained in the "Reading The Displays" section of the H8 Software Reference Manual explains how to read the displays. However, the data-type LED (the one on the right) displays the type of data being read or written. This information is displayed as:

DISPLAY	DATA/TYPE
1	Memory Image
2	BASIC Program Text (version 10.01.00)
3	Compressed Text
4	BASIC Program Text (version 10.02.00)
5	BASIC Program Variables (version 10.02.00)
6	BASIC Program Text and Program Variables (version 10.02.00)



NEW COMMAND MODE STATEMENTS

The command mode statements in this version (10.02.00) of BASIC either replace or supplement those listed in the H8 Software Reference Manual. Refer to the statements in this Manual when you run version 10.02.00 of Extended Benton Harbor BASIC.

DUMP

The DUMP statement is the same as the dump statement for Extended Benton Harbor BASIC version 10.01.00 except that it produces a file type 4. A program dumped using the DUMP statement can only be loaded using the LOAD statement.

LOAD

The LOAD statement is the same as the load statement for Extended Benton Harbor BASIC version 10.01.00 except that the variables in memory are not destroyed when the program (file type 4) is loaded. However, the variables will be cleared to zero if you use the RUN command to execute the program after it is loaded. Therefore, use the CONTINUE statement when you want to run the program without destroying the variables. The LOAD command enters the LOCK mode after it has loaded the program text.

OLDLOAD

The OLDLOAD statement lets you load programs (file type 2) that were dumped using the old version (10.01.00) of Extended Benton Harbor BASIC. The OLDLOAD statement, unlike the new LOAD statement, destroys the program variables currently in memory. The form of the OLDLOAD statement is:

*OLDLOAD "name" @

PUT

The PUT statement is a form of dump statement, in that it saves the program variables on tape (file type 5). It does not save the program, only the variables. The "name" that you give to the variables is written on the tape so you can reload the variables in the future using the specified name. The form of the PUT statement is:

*PUT "name" @



The string "name" may consist of up to 80 ASCII characters. Any normal ASCII character string is permitted. Make sure the tape drive is ready before you enter the PUT statement. BASIC starts the drive, writes the data, and stops the drive. You can use CONTROL-C to abort the PUT routine; however, the file will not be complete. A program dumped using the PUT statement can only be loaded using the GET statement.

GET

The GET statement loads variables (file type 5), previously stored on tape, into memory. The current variables in memory are destroyed, but the program text is not affected. Since the variables are stored in mass storage under a specified name, you can load them from storage using their specified name. The form of the GET statement is:

*GET "name" @

SURE?

A Y reply to the question "SURE?" causes BASIC to scan the mass storage device until it finds a variable (file type 5) whose name matches the specified string "name." It then destroys current variables in memory and loads the new variables. Any other response cancels the GET routine. If the name in mass storage device is longer than the specified name you enter, a match on the supplied characters in the string "name" is valid. Thus, a program may be dumped (PUT) with extra information entered in the name (such as program version number). This lets you load a program without entering the extra information. The GET command enters the LOCK mode after it has loaded the program variables.

FDUMP

The FDUMP statement is a combination of the DUMP and PUT statements in that it dumps both the program text and the variables (file type 6). The "name" that you give to the program is written on the tape so that you can reload the program and variables in the future using the specified name. The form of the FDUMP statement is:

*FDUMP "name" 🕏

The string "name" may consist of up to 80 ASCII characters. Any normal ASCII character string is permitted. Make sure the tape drive is ready before you enter the FDUMP statement. BASIC starts the drive, writes the data, and stops the drive. You can use the CONTROL-C to abort the FDUMP routine; however, the file will be incomplete. A program dumped using the FDUMP statement can only be loaded using the FLOAD statement.



FLOAD

The FLOAD statement is a combination of the LOAD and GET statements in that it loads both the program and its variables (file type 6) at the same time. Since the program and its variables are stored in mass storage under a specified name, you can load them from storage using the specified name. The form of the FLOAD statement is:

*FLOAD "name" ®
SURE?

A Y reply to the question "SURE?" causes BASIC to scan the mass storage device until it finds a program (file type 6) whose name matches the specified string "name." It then destroys the current program in memory and loads the new program text and program variables. Any other response cancels the FLOAD routine. If the name in the mass storage device is longer than the specified name you enter, a match on the supplied characters in the string "name" is valid. Thus, a program may be dumped (FDUMPed) with extra information entered in the name (such as program version). This lets you load a program without entering the extra information. The FLOAD command enters the LOCK mode after it has loaded the program text and the program variables.

LOCK

The LOCK statement protects your program by preventing the execution of the following command mode statements:

BUILD	SCRATCH
DELETE	CLEAR
LOAD	GET
RIIN	FLOAD



It also prevents the entry or deletion of program text. Variables can be changed, but not deleted. The form of the LOCK statement is:

*LOCK @

A lock error (LOCK) is generated if you attempt to enter a "locked out" command mode statement, such as RUN. Use the UNLOCK statement to abort the LOCK mode.

UNLOCK

The UNLOCK statement aborts the LOCK mode and restores the use of all command mode statements. The form of the UNLOCK statement is:

*UNLOCK @

ERROR TABLE

The following new error listing is in addition to those errors given in the "BASIC Error Table" in the H8 Software Reference Manual.

BASIC EXTENDED COMMENTS

BASIC (Cause of Error)

LOCK
 Data LOCK engaged. Attempting to change data

using the RUN, CLEAR, SCRATCH, BUILD, LOAD, DELETE, GET, or FLOAD commands. Or, attempting to add or delete lines of BASIC text.



OPTIONAL PATCHES

An optional patch is supplied with Extended Benton Harbor BASIC version 10.02.00. This option is for systems that use a terminal device that requires two stop bits (such as the Teletype Model ASR 33). Refer to the "Installing A Patch" section of the H8 Software Reference Manual if you need to use either of these optional patches.

	EXIENDED BENIUN HARBOR BASIC 10.02.xx.
	OPTION PATCH #1
	.2.SIOP. BILS.
.USE.	THIS PAICH IS INSERTED FOR SYSTEMS WHICH USE A LERMINAL DEVICE. REQUIRING 2 STOP BITS. THIS SHOULD NOT BE USED FOR DEVICES WHICH CAN RUN WITH DNLY ONE STOP BIL.
.NOTES.	. NONE ♣
041010) 316 L001.
	:

BASIC UTILITY ROUTINES

The following pages contain a description of several utility routines included in Extended Benton Harbor BASIC version 10.02.00. You can use them with userwritten machine language routines called by the USR function. Refer to the section "Entry Points to BASIC Utility Routines" in this Manual.

0000.000				
# # # # # # # # # # # # # # # # # # #		•	: :	
# # # # # # # # # # # # # # # # # # #				
#3 *		*	- *WINTEK	INTERPRETER
### ### ### ### ### ### ### ### ### ##			J. G. L., 09/76, FI	OR *WINTEK* CORPORATION.
# # # # # # # # # # # # # # # # # # #				LAFAYETTE, IN.
## COPYRIGHT 09/1976, ##IN 90		•	3	: 4
90 ** 000		•	•	ž :
### COPYRIGHT 09/1976, *#INJEK* 90 * 91 * 91 * 92 * 93 * 94 * 94 * 95 * 97 *** COPYFJGHT 01/1976, ##INJEK* 94 * 95 * 99				
91 * LAFAYETTE 92 * COPYFIGHT 01/1978, HEATH COM 94 * COPYFIGHT 01/1978, HEATH COM 94 * COPYFIGHT 01/1978, HEATH COM 95 * COPYFIGHT 01/1978, HEATH COM 95 * COPYFIGHT 01/1978, HEATH COM 96 * COPYFIGHT 01/1978, HEATH COM 97 ** COPYFIGHT 01/1978, HEATH COM 97 ** COPYFIGHT 01/1978, HEATH COM 91 ** COPYFIGHT 01/1978, HEATH COM 91 ** COPYFIGHT 01/1978, HEATH COM 91 ** COPYFIGHT 01/1978, HEATH COM 92 ** COPYFIGHT 01/1978, HEATH COM 93 ** COPYFIGHT 01/1978, HEATH COM 94 ** COPYFIGHT 01/1978, HEATH COM 95 ** ACCX 96 ** ACCX 96 ** ACCX 96 ** ACCX 96 ** ACCX 97 ** ACCX 97 ** ACCX 97 ** ACCX 97 ** ACCX 98 ** ACCX 98 ** ACCX 98 ** ACCX 99 ** ACCX 90 ** ACCX 91 ** ACCX 92 ** ACCX 93 ** ACCX 94 ** ACCX 95 ** ACCX 96 ** ACCX 97 ** ACCX 97 ** ACCX 97 ** ACCX 98 ** A		*:	COPYRIGHT 09/1976.	*WINTEK*
93 * COPYRIGHT 01/1978, 94 * 94 * 95 * 100 08				902 N. 91H ST.
93 * COPYFIGHT 01/1978, HEATH COMPANY 94 * 94 * 95 * 100 100 100 100 100 100 100 100 100 10		:		- MONTO IN TOTAL MONTO CONTINUE CONTINU
94 * 95 * 100		:	COPYRIGHT 01/1978.	Z.
97. ** LON-WEMORY. CELLS. USFD. BY 94 94 99 99 99 99 99 99 99 946 7*34.01%EC. 100 100 05 4 103 05 4 104 ACCY 0S 4 4 4		:	HE.	≥ :
99 100 101 101 08 2 102 ACX 08 4 103 ACY 08 4		*	LON-MEMORY CELLS U	SFD RY BASIC
101 103 103 103 104 104 104 105 106 106 107 107 108 108 108	7 0 0 0		247111 +2+L 500	
101 102 ACCX 0S 4 103 ACCY 0S 4 104 ACCY 0S 4		100		
102 ACCX 0S 4 103 ACCY 0S 4	040.064	1.01		ACCX, TYPE
104 ACCY 05 4	040.066	102		
	040.074	:		



JASIC - HEATH HASIC INTERPRET	ER.	FUNC11UN.	HEATH HBASM V1.2 10/20/77	PAGE
10	** 40	USR - CALL USF ASSEMRY LANGUAGE FUNCTION		
10	x.	סבי מסבי שמחות בשומים במינים	•	
	:	THE *USP* FUCTION IS ACTUALLY A CALL TO A USER-WRITTEN ROUTINE WHICH MUST HAVE REEN PREVIOUSLY LOADED INTO MEMORY BY THE USER	USER-WRITTEN ROUTINE TO MEMORY BY THE USER.	:
	- ~	THE ADDRESS OF THE FUNCTION'S ENIBY BOINT WIST OF	MINIST DE 17 MINISTER	:
11	13		* K Z J J J J J J J J J J J J J J J J J J	
	* 511	FINATION WESTDES IN MEMORY AROVE THE STACK POINTER (HIGH MEMORY)	D SO THAT THE USER K POINTFR (HIGH MEMORY)	:
	117 ×	OF ELSE BASIC FILL OVERLAY THE FUNCTION W	ІТН ОАТА.	
	* * *	FIGATING POINTS ENTERED WITH A POINTER TO	THE SINGLE ARGUMENT (IN	
	* 0.2	IF NO RETURN VALUE IS PLACED IN *ACCX*, THEN THE ORIGINAL ARGUMENT	G POINT OR STRING ARBUMENT. HEN THE ORIGINAL ARGUMENT	
	* * <<	REMAINS THERE, AND USR (RETURNS ITS ARGUM	ENT AS ITS VALUE.	
15.0	* * *	ENTRY (BC) = #ACCX		
	25 *	USES ALL		
	27			
	ar:			
T 000 000	30 *	DW 0 ADDRESS FO USER F	FUNCTION ENTRY POINT NOT LEGAL	

197					
1350 ** 136 ** 139 139 141 142 143 144 145 145 146 146 147 148 148 148 148 149 149 149 149 149 149 149 149 149 149		•	THESE FRAUR	R PROCESSORS ARE ENTERED WHEN AN ERROR IS DETECTED.	
139 ERR, CC EQU ** 140 ERR, CB EQU ** 141 ERR, DD EQU ** 145 ERR, DD EQU ** 146 ERR, DD EQU ** 151 ERR, NV EQU ** 152 ERR, NV EQU ** 154 ERR, NV EQU ** 155 ERR, NV EQU ** 156 ERR, NV EQU ** 157 ERR, SR EQU ** 158 ERR, SR EQU ** 159 ERR, SR EQU ** 150 ERR, SR EQU ** 171 ERR, NO EQU ** 172 ERR, NO EQU ** 174 ERR, NO EQU ** 175 ERR, NO EQU ** 176 ERR, NO EQU ** 177 ERR, NO EQU ** 177 ERR, NO EQU ** 178 ERR, NO EQU ** 178 ERR, NO EQU ** 179 ERR, NO EQU ** 170 ERR, NO EQU ** 171 ERR, NO EQU ** 171 ERR, NO EQU ** 172 ERR, NO EQU ** 174 ERR, NO EQU ** 175 ERR, NO EQU ** 176 ERR, NO EQU ** 177 ERR, NO EQU ** 177 ERR, NO EQU ** 178 ERR, NO EQU ** 178 ERR, NO EQU ** 179 ERR, NO EQU **		•	CONTROL PAS	SSES DIRECTLY BACK TO COMMAND MODE.	
142 ERR.CB EQU ** 143 ERR.DE EQU ** 146 ERR.DO EQU ** 146 ERR.DO EQU ** 151 ERR.TIV EQU ** 152 ERR.NY EQU ** 154 ERR.NY EQU ** 155 ERR.NY EQU ** 156 ERR.SN EQU ** 157 ERR.SR EQU ** 167 ERR.SR EQU ** 168 ERR.SR EQU ** 169 ERR.SR EQU ** 160 ERR.SR EQU ** 161 ERR.SR EQU ** 171 ERR.NO EQU ** 172 ERR.NO EQU ** 174 ERR.NO EQU ** 175 ERR.LC EQU ** 176 ERR.LC EQU ** 177 ERR.LC EQU ** 178 ERR.LC EQU ** 179 ERR.TR EQU ** 179 ERR.TR EQU ** 170 ERR.TR EQU ** 170 ERR.TR EQU ** 171 ERR.LC EQU ** 171 ERR.LC EQU ** 172 ERR.TR EQU ** 174 ERR.LC EQU **		; w	:	CONTROL-C	
144 ERR, DE EQU ** 146 ERR, DO EQU ** 147 ERR, IN EQU ** 151 ERR, IN EQU ** 152 ERR, NV EQU ** 154 ERR, NV EQU ** 156 ERR, SN EQU ** 157 ERR, SN EQU ** 158 ERR, SN EQU ** 159 ERR, SN EQU ** 150 ERR, SN EQU ** 151 ERR, SN EQU ** 152 ERR, SN EQU ** 153 ERR, SN EQU ** 154 ERR, SN EQU ** 155 ERR, SN EQU ** 156 ERR, SN EQU ** 177 ERR, SN EQU ** 178 ERR, ID EQU ** 179 ERR, ID EQU **	1	1		CONTROL-B	
1445 147 147 147 147 148 159 159 154 158 158 158 158 159 159 160 160 160 160 160 160 160 160 160 160		. ш.	: :	DATA EXHAUSTED	
147 147 149 149 150 151 151 153 153 158 158 158 158 159 160 161 162 163 164 164 164 167 169 169 169 169 171 171 171 171 172 188.10 173 174 175 178 178 178 178 178 178 178 178 178 178	1	ш.	:	0/	
140 EPR.1U FQU * 151	1	ш.	- :	ILLEGAL NUMBER	
152 FRP.LK EQU * 153 FRP.NV FQU * 154 FRP.NV FQU * 156 FRP.SK EQU * 156 ERP.SK EQU * 163 ERP.SK EQU * 164 ERP.SK EQU * 165 ERP.SK EQU * 167 ERP.SK EQU * 170 ERP.SK EQU * 171 ERP.LC EQU * 173 ERP.LC EQU * 174 ERP.K EQU * 175 ERP.LC EQU * 176 ERP.LC EQU * 177 ERP.LC EQU * 177 ERP.LC EQU * 178 ERP.LC EQU * 178 ERP.LC EQU * 179 ERP.LC EQU * 179 ERP.LC EQU * 170 ERP		ш.		ILLEGAL USAGE	
153 ERR, NV FOU * 154 ERR, RE EQU * 156 ERR, RE EQU * 157 ERR, SR EQU * 163 ERR, SR EQU * 164 ERR, SR EQU * 165 ERR, TC EQU * 165 ERR, TC EQU * 167 ERR, ND EQU * 173 ERR, ND EQU * 174 ERR, ND EQU * 175 ERR, TR EQU * 177 ERR, TR EQU *		:سا		DATA LOCK ENGAGED	
155 FRR.0V EQU * 157 ERR.RE EQU * 150 ERR.RE EQU * 160 ERR.SN EQU * 161 ERR.SN EQU * 162 ERR.TC EQU * 164 ERR.TC EQU * 167 ERR.TC EQU * 171 ERR.NO EQU * 173 ERR.NO EQU * 174 ERR.NO EQU * 175 ERR.NO EQU * 176 ERR.TC EQU * 177 ERR.NO EQU * 178 ERR.NO EQU * 179 ERR.NO EQU *		ш,	:	NEXT VARIABLE MISSING	
158 ERR, RE EQU ** 169 ERR, SL EQU ** 162 ERR, SN EQU ** 164 EPR, SY EQU ** 165 ERR, TC EQU ** 166 ERR, TC EQU ** 177 ERR, SR EQU ** 173 ERR, NO EQU ** 174 ERR, NO EQU ** 175 ERR, LO EQU ** 177 LTA ERR, LO EQU ** 178 ERR, LO EQU ** 179 LTA ERR, LO EQU ** 179 LTA ERR, LO EQU ** 170 EQU **		u.		WO FRED	
159 160 161 165 164 164 164 167 169 169 170 170 171 172 173 174 175 175 179 179 179 179 179 179 179 179 179 179		: ш	:	DE TILBA CODOD	
162 ERR, SN EQU * 163 ERR, SN EQU * 164 ERR, SY EQU * 165 ERR, TC EQU * 167 ERR, TC EQU * 170 ERR, SR EQU * 171 ERR, ND EQU * 175 ERR, ND EQU * 175 LTA ERR, ND EQU * 177 LTA ERR, ND EQU * 177 LTA ERR, ND EQU * 178 ERR, ND EQU *		. 4	•	THE COLUMN TO SERVICE STATE OF THE COLUMN TO SERVICE STATE STATE OF THE COLUMN TO SERVICE STATE S	
163		1: 4	:	בורפת בור ביים ביים ביים ביים ביים ביים ביים ביי	
164 EPR.SY EQU * 165 EPR.TC EQU * 168 EPR.TC EQU * 169 EPR.SE EQU * 171 EPR.SC EQU * 173 ERR.ND EQU * 174 EPR.ND EQU * 175 EPR.LC EQU * 177 EPR.LC EQU * 177 EPR.LC EQU * 177 EPR.LC EQU *		u; i	*	STATEMENT NUMBER	
166. EFR, IC EQU. * 167. 168. EPR, IQ EQU. * 170. ERR, SR EQU. * 173. ERR, NO EQU. * 175. ERR, NO EQU. * 175. 174. ERR, IQ EQU. * 177. 178. ERR, IQ EQU. * 179. 180. ERR, IP EQU. *		Ψ.	:	SYNTAX, ERROR	
168 EPR.10 EQU. * 169 170 ERR.SR EQU. * 171 172 ERR.SC EQU. * 173 ERR.NO EQU. * 175 ERR.LD EQU. * 177 ERR.LD EQU. * 179 ERR.LD EQU. * 179 ERR.LD EQU. *		ш:		TYPE CONFLICT	
170 ERR.SR EQU. * 171 172 ERR.SC EQU. * 173 ERR.NO EQU. * 174 ERR.NO EQU. * 175 ERR.UO EQU. * 177 ERR.UO EQU. * 179 ERR.UO EQU. *		œ:	Equ	TABLE QVERFLOW	
173 ERR.SC EQU. * 173 ERR.NO EQU. * 174 ERR.NO EQU. * 175 ERR.LO EQU. * 177 ERR.LO EQU. * 179 ERR.LO EQU. *		щ	EqU	SUBSCRIPT RANGE	
173		14.	EQU. *	SUBSCRIPT COUNT	
175 176 ERR.IC EQU. * 177 179 179 180 ERR.IP EQU. *		Ψ,	Equ. *	NOT DIMENSIONED	
177 179 188.10. EQU. *		w		TLLFGAL CHARACTER	
179 180 ERR-IP EQU. *		بعا	Fau	LINDESTACO CINCATON	
		, w	Equ. *	TAPE ERROR	

	183	:	- × ^ C	CVX = COPY VALUE INTO "XX" ACCUMULATOR.
	184	* *	O. XVO	CVX COPIES A 4 BYTE VALUE INTO THE X ACCUMULATOR.
	186	:		:
	188		EXIT	COPIED
	0.01	:	USES	:
	161	:	17370	
•		\$. 2:)))	
			:	
	194	:	- XXO	- COPY (ACCX) TO (ACCY)
	196	:	ENTRY	
	197	* *	EXIT	NONE A.F.D.F
	199	:		:
	200	C × √	FAU	*
	20 <i>5</i> 206	* *	CXV COPI	CXV COPIES THE CONTENTS OF THE 'X' ACCUMUALTOR INTO A MEMORY
	207	:		•
	000 000 010	* * *	EXIT USES	((OF) = TARGET ADDRESS COPIED A,F
	211	: :		:
	213	> ×	EOU	
		:		
	615	*	× :	- SPLII NUMBER INTO INTEGER AND FRACTION.
	217.	:	IF.1X.F	JEIX FIXES ((DE)), INTO IN INTEGER.
	218	* *	ENTRY	(DE) # ADDRESS OF NUMBER
	220	:	EXIT	(DE) = INTEGRAL PART OF 0<=N<=65535
	222	*		TO ERRITO OTHERMISE
	223			
	757	IF IX	E 0.0	*

226 ** FELT - FLOAT NUMBER - 227 ** FELT	UTILITY SUBROUTINES	INTERPRETER			HEATH H8ASM VI.2" TO/20/77 PAGE 13:12:47 15-JAN-78
229 * 230 *		:N 0	* •	"YFUT"	
230 * 231 233		u:∩ ∩	* *	ENTRY	(66) = VALUE
232 235 236 237 237 237 237 241 241 241 241 242 243 243 243 243 243 243 243 243 243		7:W W.	. *	- 1 - 1	(DE) = WACCX-1
235 ** 237 * 238 * 239 * 239 * 239 * 241 * 242 * 243 * 243 * 244 * 245 * 246 * 246 * 246 * 246 * 247 * 247 * 248 *		ww:	IFLT		*
236 ** 238 ** 240 ** 241 ** 241 ** 242 ** 243 ** 244 ** 244 ** 245 ** 245 ** 251 ** 252 ** 252 ** 254 ** 255 ** 255 ** 256 ** 264 ** 265 ** 265 ** 266 ** 26		235	*	- 101	TYPE DECIMAL INTEGER.
239 * 240 * 241 *		236	* *	TOI TY	YPES AN INTEGER AS A 5 PLACE NUMBER, LEADING ZEROS ARE
240 * ENTRY (DE) = 241 * EXIT TYPED 243 244 245 TDI EQU * PENTRY NONE 248 250 * ENTRY NONE 251 * USES ArF DI EQU * EXIT TYPED 252 * ENTRY NONE 252 254 XCY EQU * ENTRY (HL) = 254 XCY EQU * ENTRY (HL) = 254 XCY EQU * ENTRY (HL) = 255 XCY EQU * EQU * ENTRY (HL) = 255 XCY EQU * EQU * EQU * ENTR		23.8	* 1	SUPPRE	FSSED.
241 * EXIT TYPED 242 * USES A,F,D, 243 244		240	* *	ENTRY	:
243 244 245 247 247 247 248 248 248 248 248 248 248 248 248 248		241	* *	USES	
245 TDI EGU * 247 ** XCY - EXCHANGE 248 * ENTRY NONE 250 * EXIT NONE 251 * USES A F 252 * COU * 253 * ZRO - ZERO MEM 254 * ZRO - ZERO MEM 255 * ZRO - ZERO MEM 256 * ENTRY (HL) = 256 * ENTRY (OE) = 256 * ZRO - ZERO MEM 265 * ZRO - ZERO MEM 265 * ZRO - ZERO MEM 265 * ZRO - ZRO - ZERO MEM 265 * ZRO - ZERO MEM 265 * ZRO - ZRO		243			
248 * ENIRY NONE 248 * ENIRY NONE 250 * ENIRY NONE 251 * USFS A F 252 * ENI * NONE 253 * ENIRY (HL) # 254 * ENIRY (HL) # 255 * ENIRY (HL) # 265 * ENIRY (HL) # 266 * ENIRY (HL) # 267 * ENIRY (HL) # 268 *		245	TOI	EGU	***************************************
249 * 250 * 251 * 251 * 252 * 252 * 253 * 254 * 254 * 254 * 254 * 254 * 254 * 254 * 255 *		247	* 1	XCY.	EXCHANGE
251 * 251 * 251 * 251 * 255 * 257 *		249	*	ENTRY	:
252 253 254 xCY 254 x 255 x 25		251	* *	USES	:
254 xCY 256 ** 257 * 257 * 257 * 257 * 261 * 262 * 263 * 263 * 263 * 263 * 264 * 265 *		252			
256. ** 257. * 258. * 259. * 260. * 261. * 263. * 264. * 264. * 264. 280	•	524	×C×	EQU	*
257 * 258 * 259 * 259 * 261 * 261 * 263 * 263 * 263 * 264 * 265 * 264 * 265 *		256.		ZR0 =	. ZERO. MEMORY.
259 * 260. x 261. x 261. x 264. x 264. x 264. x 264. x 265. x 265		257 258.	* *	ZR0.ZE	EROS. A. FIELD. OF MEMORY.
261 * EXII 262 * EXII 263 * USES 264 * USES 265 260 EAU.		259	* +	2	
263 * USES 264 * USES 264 265 265 280 EAU.		261	< *		:
264 265 266. ZRD. E.B.U.		262	* *	EXI.T USES	:
		264.			
	-	266.	. ZKD	E.QU	**************************************

EXPONENT: EXPONENT:
272 * 89 A 4-897F VALUE. THE NUMBER C 274 * EXPONENT, THE NUMBER FORMAT IS: 276 * N+0

FLOATING PUINT ROUTINES.			3547 17.03701 35.1 TS.05.1 TS.
35		FPADD	FPADD - FLOATING POINT ADD.
325	:	ACCX =	ACCX + (DE)
V. C.	:	ENTRY	(DE) = POINTER TO 4 BYTE FP VALUE
35.	* *	ExIT	ACCX # REGULT
33	:	USES	A THE CALCANGED
55 55			
33	3 FPADD	EOU	**************************************
K. K.	**	FPSUB	FPSUM - FLOATING POINT SUBTRACT.
337.	:	FPSUB	FPSUB COMPUTES (DE) - ACCX
33			
330	* * 0	ENTRY	(OE) = POINTER TO 4 BYTE FP VALUE
1 P.			SUPPLIED VALUE UNCHANGED
342	*	USES	A,F
34.	د. د		
34	FPSUR	Egu	*
34	7 **	FPNRM	- FLOATING POINT NORMALIZE.
341	*		
349		FPNRM	FPNRM NORMALIZES THE CONTENTS OF (ACCX).
35		FNTRY	VONE CACCA MODIAN 1755
35.	* *	USES	A, F
35	7.		
35(F.PNRM	E.U	
358	:	FPNEG	- FLNATING POINT NEGATE.
360	:	FPNEG	NEGATES THE CONIENTS OF ACCX.
362		ENTRY	NON
	:	Ex1.T.	(ACCX) = = (ACCX)
364		USES	ApF
366	d e		

10 10 10 10 10 10 10 10	FPRIL - FLOATING POINT BLUE. ENITY (DE) = ACCE. * Y USES		692			15:16:47	13:12:47 15-JAN-78
FPDIV = FLOATING POINT DIVIDE, FPDIV = FLOATING POINT FOR FEBUL FALSA FEBUL FEBUL FALSA FEBUL F	### (OF = ANDRESS OF Y USES 4,F FPOIV - FLOATING POINT DIVIDE. ACCX = ACCX Y EXIT (ACCX) = MESULT USES 4,F EXIT (ACCX) = MESULT USES 4,F ATF - ASCIT TO FLOATING TWIO X FLOATING POINT VALUE EXIT (ACCX) = MESULT WANN [A, WORRS SA OF TEXT FEAT (ACCX) = MESULT ATF CONVERTS AN ASCIT STRING TWIO X FLOATING POINT VALUE EXIT (ACX) = AASCIT STRING TWIO X FLOATING POINT VALUE EXIT (ACX) = AASCIT STRING TWIO X FLOATING POINT VALUE EXIT (ACX) = AASCIT STRING TWIO X FLOATING POINT VALUE EXIT (ACX) = AAADE EXIT (ACX) = AAADE SECTION ACCX = AAADE FEAT (ACX) = AAADE SECTION ACCX = AAADE FEAT (ACX) = AAADE SECTION ACCX = AAADE FEAT (ACX) = AAADE			*	•	FLOATING POINT MULTIPLY	
WILL GCK = ACCK + Y WILL GCK = ACCK + Y	#UL FOU * FPOIV - FLOATING POINT DIVIDE, ACCX = ACCX = ACCX Y ENTRY (DE) = POINTER TO Y EXT (ACCX) = HESULI USES A,F ATF - ASCII TO FLOATING. * ATF - ASCII TO FLOATING.		570 371	* *	ENTRY	(DE) = ADDRESS OF Y	
### FOUL ** FEDIV - FLOATING POINT DIVIDE. FEDIV - FLOATING POINT DIVIDE. FEDIV - FLOATING POINT DIVIDE. FACK = ACCX/Y EXIL (ACCX) = MESULI USES A,F ATF - ASCII TO FLOATING. 1311247 15-JAN-76 13	FEDIT	r	372	*	EXII	ACCX = ACCX + Y	
#### FDDIV - FLOATING POINT DIVIDE. ACCX = ACCX/Y ENTRY (OE) = POINTEN TO Y ISLOAM	#### #################################		374	*	0.55.5	A	
### ##################################	### ##################################		375 376		FOU	*	
### ##################################	### ##################################						
ACCX = ACCX/Y ENTRY (ACCX) = RESULT USES A,F EXIT (ACCX) = RESULT HEATH HBASH VI.2 10/20/77 13:12:47 15-JAN-78 S. 13:12:47 15-JAN-78 NAMIN 1. WINN 1E (+-) NY ENTRY (H.) = ADDRESS OF TEXT EXIT (H.) = ADDRESS OF TEXT IN ACCX.	ACCX = ACCX/Y ENINY (GCX) = POINTER TO Y ENINY (ACCX) = POINTER TO Y ENINY (ACCX) = POINTER TO Y INTERPRETATION PAINTER ATF = ASCIT TO FLOATING. ATT = ASCIT TO FLOATING		378	*	FPDIV -	FLOATING POINT DIVIDE.	
ACCX = ACCX/Y EXIT (ACCX) = FESULT USES A.F TO SELECT A.F A.F A.F A.F A.F A.F A.F A.	ACCX = ACCX		379	*			
E-VIPY (OE) = POINTER TO Y USES 4,F 101V FOU. * * ATF - ASCIT TO FLOATING. * ATF - ASCIT STAING INTO A FLOATING POINT VALUE * ATF - ADORESS OF IEXT * ALT - (ALL) PROATED * ASCIT - ADORESS OF IEXT * ATF - ADORESS OF IEXT * ATT - ADORESS OF IEX	E ENIRY (DE) = POINTER TO Y USES AFF	,-1	380	* *		ACCX/Y	
DIV fau. * ATF - ASCII TO FLOATING. ATF CONVERTS AN ASCII STRING INTO A FLOATING POINT VALUE IN ACCX. SYNTAX NNNN [,NNN] [E (+-) NN] EXIT (HL) UPDATED USES A,F,H,L 17. EQU. *	011V		382	*	ENTRY	(DE) = POINTER TO Y	
917. £90. * * ATF - ASCII TO FLOATING. ATF CONVERTS AN ASCII SYRING INTO A FLOATING POINT VALUE IN ACC.* SYMIAX MININ [F. ML] = ADORESS OF TEXT EXIT (HL) UPDATED (ACC.) = VALUE USES A,F,H,L	917 £90. * ** AYF - ASCII TO FLOATING. ** AND		583 384	* *	USES	(ACCX) = RESULT	
	######################################		385				
S. * ATF - ASCII IN FLOATING. ATF CONVERTS AN ASCII STRING INTO A FLOATING POINT VALUE SYNTAX NNNN [NN] [E. (+) NN] ENIRY (HL) = ADDRESS OF TEXT EXIT (AL) = ALUE USES A,F,H,L.	S. * ATF - ASCII TO FLOATING. ATF CONVERTS AN ASCII STRING INTO A FLOATING POINT VALUE IN ACC.*. SYNTAX MININ [F. MIL) = ADORESS OF TEXT EXIT (HL) UPDATED (ACC.*) = "VALUE USES A.F.H.L. T. E. G.U. *	::		FPDIV	Fau	•	
390 ** 391 * 392 * 392 * 392 * 394 * 395 * 396 * 400 * 400 * 402 * 402 *	390 ** 391 ** 392 ** 392 ** 392 ** 396 ** 396 ** 400 ** 402 ** 402 ** 405 ** 40	IC - HEATH BASIC INTERPR II/FLOATING CONVERSION R	ETER IOUTI	: ທ:		HEATH HBASH 13:12:47 15	
392 * 393 * 394 * 394 * 395 * 396 * 396 * 397 * 398 *	3992 * 3943 * 3945 * 39		390	* *	ATF - 1	ASCII TO FLOATING.	
394 * 37NTAX 395 * 37NTAX 396 * NNNN I 398 * ENTRY 400 * EXIT 401 * USES 403 404 404 405 ATF FQU	394 * SYNTAX 396 * NNNN [396 * ENTRY 400 * EXIT 401 * USES 402 * USES 403 404 404 404 405 ATF EQU		392		ATF COV	NVERTS AN ASCII STRING INTO A FLOATING POINT V	ALUE
396 * NNNN F 398 * NNNN F 398 * ENTRY 400 * EXIT 400 * EXIT 402 * USES 403 * USES 404 * EQU	397 * NUNN F 397 * NUNN F 398 * ENTRY 400 * EXIT 401 * USES 402 * USES 404 * EQU		394		SYNTAX		
398 * 399 * ENTRY 400 * EXIT 401 * 402 * USES 403 * 404 * 405 AJF FQU	398 * ENTRY 400 * EXIT 401 * EXIT 401 * USES 403 * USES 404 * USES 404 * EQU		396		ZZZ	NNN	
400 * EXIT 401 * USES 402 * USES 404 405 AJF EQU	400 * EXIT 401 * USES 402 * USES 404 404 405 ATF F.0U		398 399	* *	ENTRY	(HL) = ADDRESS OF TEXT	
402 * USES A 403 404 405 ATF EQU *	402 * USES A 403 404 405 ATF EQU.		400	* *	ExIT	(HL) UPDATED (ACCX) = VALUE	
404 405 ATF F.9U	405 ATF F.QU.		402	*	USES	A F H L	
			404	AJF	FOU	*	

	-	*	FIA - FI	FLUATING TO ASCII.	
	40 B	*			
	400	* *	REPRESENTATION.	FLOATING POINT	NUMBER INTO AN ASCII
	411	*			
	7 7	* 1	ENTRY	(ACCX) = VALUE	
	415		F Y 1 1	(AL) # ADDRESS TO STORE LEXT	
	415	*	:	(DE) = ADDRESS OF LAST BYTE	
	416	*	SE.S	AFFORE	
	417				
	419	FTA	FOU		
BASIC - HEATH BASIC INTERPRETE! FLOATING POINT CONSTANTS.	.07	•			HEATH HBASM V1.2 10/20/77 PAGE 13:12:47 15-14N-76
	422	*	FLOATING	FLNATING POINT VALUES.	
	423	*			
	424		LON	G LIST GENERATED	ATED BYTES
100	426	FP1.0		,0,1000,2010	
201	428		· · · · · · ·		
000 000 120	429	FP10.	DR	0,0,1200,2040	
	430				
000	7).).	2	0.00	
022 170 233	432	NP1.2	90	0220,1700,2330,2010 -PI/2	1/2
033 170 313	434	0.00	2		C+1
:	0			1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
022 170 233 202	437	I dv	DH	0220,1700,2330,2020 -PI	1
022 170 233	438 439	4.14v	Æ	0929,1700,2330,2000	7/Id-
002	077				
356 207 144	177	p.I.d	90	3560,2070,1440,2000 PI	P1/4
	443	: : : : :			
	777	: : : : : :	FND		
444 STATEMENTS					
41178 AYTES FREE					



ENTRY POINTS TO BASIC UTILITY ROUTINES

		ADDRESSE	S FOR					
	EXTENDE	D BENTON	HARBOR BASIC					
		• • • • • • • • • • • • • • • • • • • •						
ACCX CXV ERR.DO	040066 .0.70.023 .065136	CXY	040074 070007 065125	ATF ERR.CH ERR.IC	101070 065115 065333	CVX FRH.CC FRK.IA	067374 065105 065143	
ERR.IU ERR.OV ERR.SN ERR.TO	065175	ERR.RE ERR.SR ERR.TP	065157 065206 065274 065355	ERR.ND ERR.SC ERR.SY ERR.UD	065324 065311 065237 065346	FRR.NV FRR.SL FPR.TC FPO.O	055166 065214 065245 106135	
FPQ.1 FPDIV FPSUB NPI	106131 100025 076333 106151	FPMUL FTA	106121 077070 101355 106141	FP10. FPNEG IFIX NPI.4	106125 077047 070340 106155	FPADD FPADA IFLI MPIZ	076125 076347 070376 106145	
PI.4 ZRO	106161 074373		074046	USRECN	106106	×C.Y	074341	
3								
**************	•••••							
•••••	•••••	• • • • • • • • • • • • • • • • • • • •		•••••				
					• • • • • • • • • • • • • • • • • • • •			
`	• • • • • • • • • • • • • • • • • • • •		• • • • • • • • • • • • • • • • • • • •				• • • • • • • • • • • • • • • • • • • •	
				•••••			• • • • • • • • • • • • • • • • • • • •	
					•••••	•••••		
:								