Part IV BASIC

- Contents -

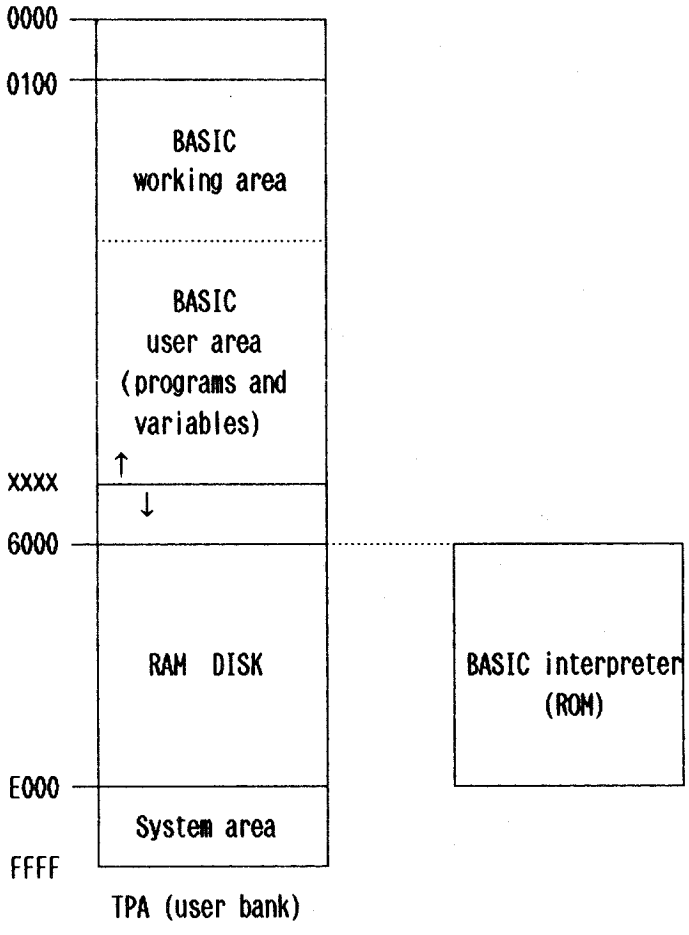
PINE BASIC

Chapter 1 Memory Management Under BASIC

Chapter 1 Memory Management Under BASIC

1.1 Outline

The BASIC interpreter is located in ROM, in a separate bank from the transient program area (TPA). After starting BASIC, memory is mapped as shown below.



In the figure above, "xxxx" indicates the upper address limit of the BASIC user area. This is either 6000H or the starting address of BDOS (RBDOS1), whichever is lower. A pointer to the BDOS entry address can be found in memory addresses 6 and 7.

To obtain the largest possible BASIC user area, set the RAM disk to the smallest possible size; this moves the BDOS entry address to a point above 6000H.

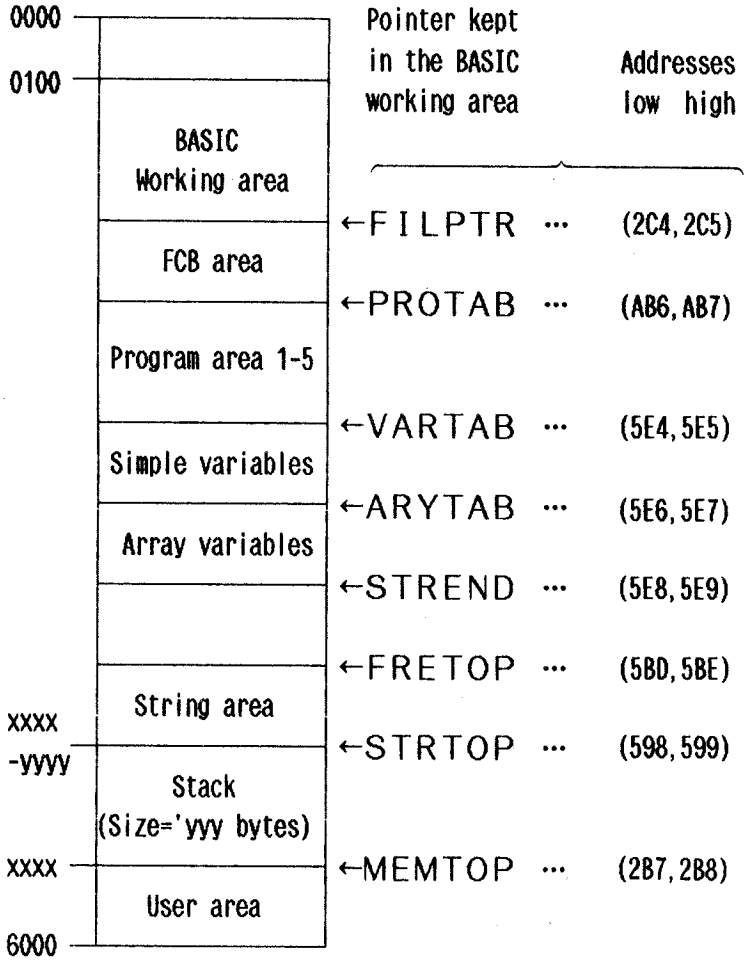
1.2 Memory Map of RAM

Transient program area addresses controlled by the BASIC interpreter are those in the range from 00H to 6000H. However, the upper address limit varies according to the starting address of BDOS, the maximum memory address specified with the /M: option when BASIC is started, and the upper memory limit (if any) specified with the CLEAR statement.

If the CLEAR statement is executed with optional parameters as follows,

CLEAR,xxxx,yyyy

the RAM memory map changes as shown below. (Here, "xxxx" specifies the upper memory limit, and "yyyy" specifies the size of the BASIC stack area.)



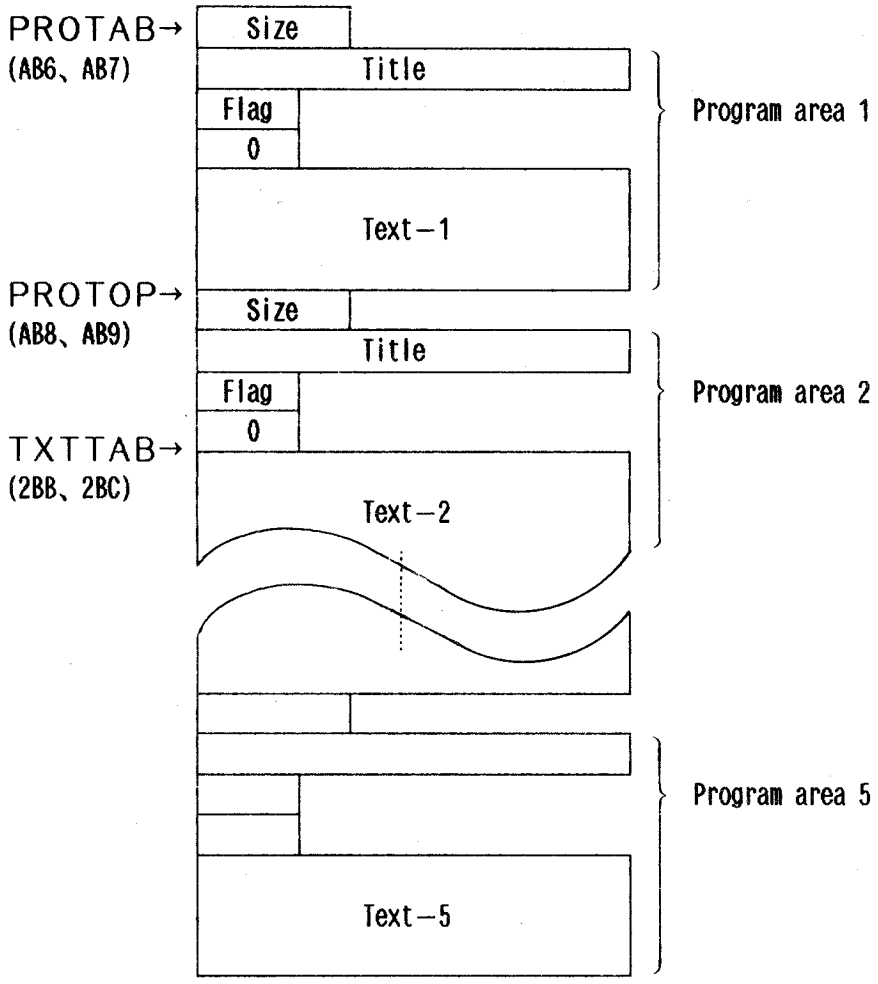
1.3 The File Control Block (FCB)

When a file number is specified as the parameter of the VARPTR function, the function returns the address of that file's file control block. The address returned varies according to whether the file specified is a sequential file (one opened in the "I" or "O" mode) or a random file (one opened in the "R" mode). The contents of the FCB are as follows.

Offset	Length (bytes)	Contents
Ø	1	Access mode 1: Indicates the "I" mode. 2: Indicates the "O" mode. 4: Indicates the "R" mode.
1	36	FCB used by BDOS.
37	2	With a sequential file, indicates the number of sectors read or written. With a random access file, indicates the last record number accessed plus 1.
39	1	Number of bytes in the sector being read or written.
40	1	Number of bytes remaining in the input buffer.
41	3	Reserved.
44	1	Device number FF: KYBD FE: SCRN FD: LPTØ FC: COMØ FB: COM1 FA: COM2 F9: COM3 F8: CASØ
45	1	Maximum number of output characters.
46	1	Output character counter.
47	2	Used internally.
49	128	Data buffer. Used as the DMA address by BDOS. When a file is opened as a sequential file, VARPTR returns the starting address of this buffer.
177	2	Buffer size for random files. The default is 128 bytes. Set according to the record length specification in the OPEN"R" statement.
179	2	The current physical record number.
181	2	The current logical record numb^r.
183	1	Reserved.
184	2	Output position for PRINT#, etc.
186	n	Buffer for random access. The size, n, is determined by the /S: option when BASIC is started; the default is 128 bytes. When a file is opened as a random access file, VARPTR returns the starting address of this buffer.

1.4 Program Areas

The PINE has five program areas, any one of which can be selected for use with the LOGIN statement. These areas are managed dynamically to prevent wasting memory space. The management scheme is illustrated in the figure below.

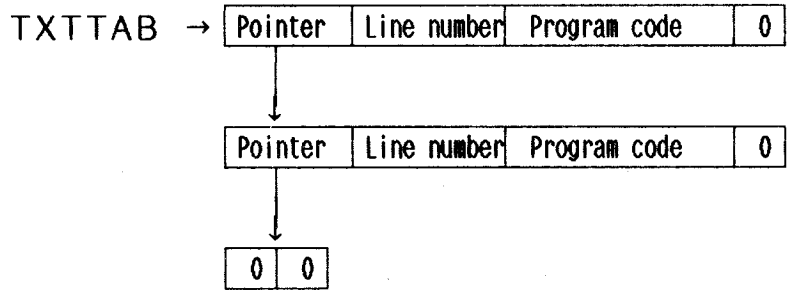


PROTOP points to the starting address of the currently selected program area, and TXTTAB points to the starting address of that program area's text area. The number of the currently selected program area is stored in PRONUM (at address AB5H). The figure above shows locations pointed to when program area 2 is selected.

The contents of each program area are as follows.

Offset	Length (bytes)	Contents
Ø	2	Size of the program area in bytes.
2	8	Program area name assigned with the TITLE statement.
1Ø	1	Program area protect flag. This flag is set with the TITLE statement's P option. When the flag is "1", the program area is protected and cannot be edited or changed with statements such as NEW or DELETE.
11	1	Reserved.
12	n	The text area. The size of this area varies according to program. The program area size (indicated at offsets Ø and 1) is equal to 12+n. The starting address of the next program area can be obtained by adding 12+n to the address indicated by PROTAB.

Format of the Text Area



The text has a chained structure, and is linked as follows.

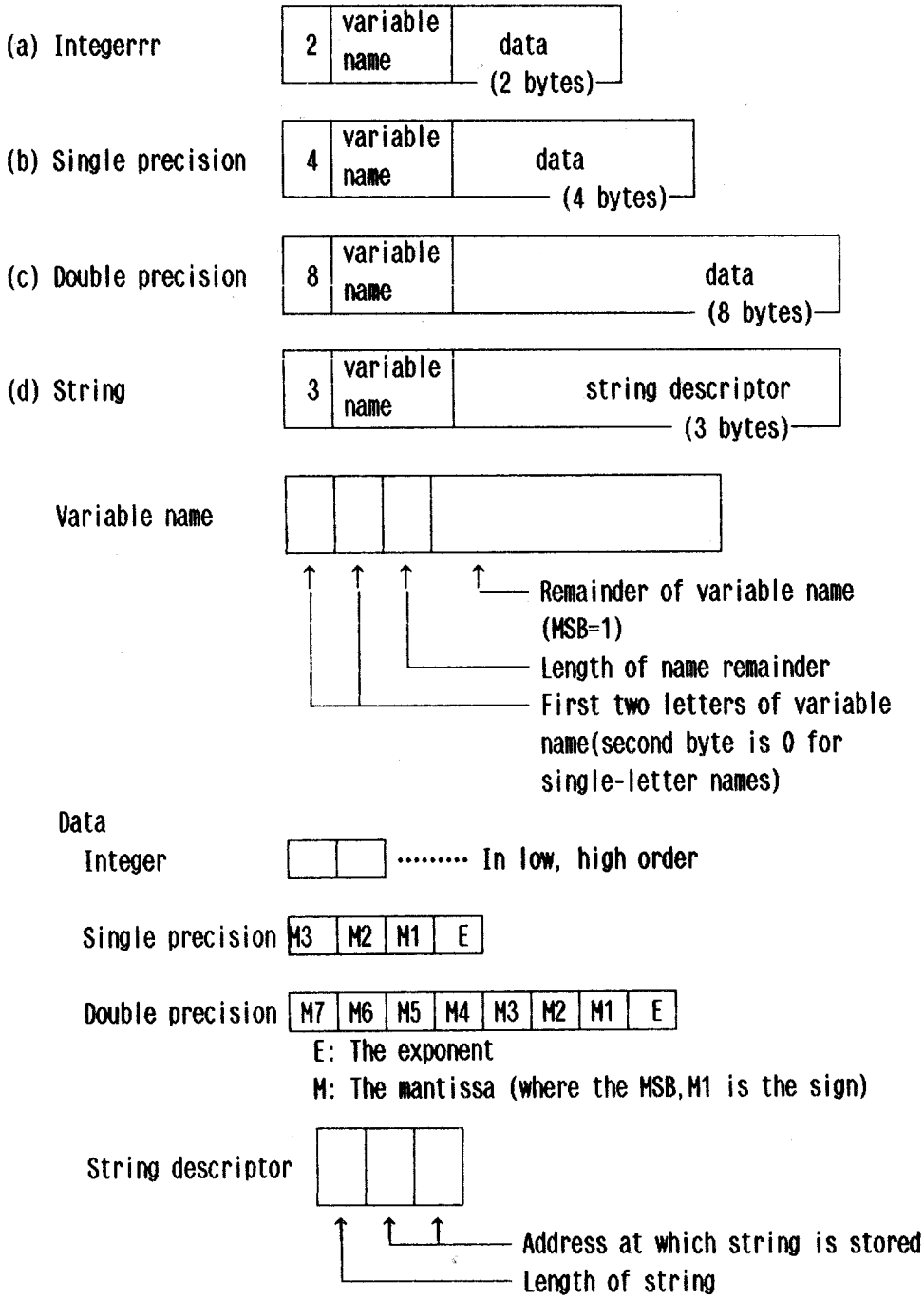
Each pointer indicates the address of the following line's pointer. Each pointer is followed by a line number, then by the program code for that line. The end of each line is indicated by Ø. The end of the entire text is indicated by a pointer consisting of two Ø bytes

1.5 Variables

The area used for variables is the same no matter what program area is currently selected.

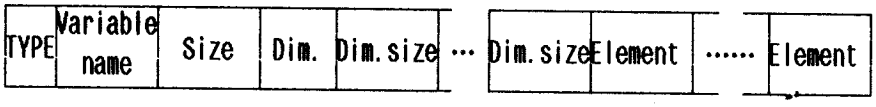
1.5.1 Simple Variables

When a program uses simple variables, they are registered in the simple program area in the order in which they are first used. Each variable is recorded in the format which corresponds to its type. The formats for each type are as follows.



1.5.2 Array Variables

Array variables are registered when they are defined by executing the DIM statement in a program, or when they are defined implicitly by using an array variable with a subscript of 10 or less. Array variables are registered in the following format in the order in which they are defined.



- Type ·····Same as for simple variables(2,3,4, or 8)
- Variable name ·····Same as for simple variables
- Size·····Two bytes indicating the number of bytes used following "Dim."
- Dim.·····One byte indicating the number of array dimensions
- Dim. size·····Two bytes indicating the size of one dimension
- Element ·····2,3,4, or 8 bytes, depending on the variable type.

1.6 Variables in the BASIC Working Area

	Address	
FRCINT	103-104	Entry point for the routine that gets integer value of the FAC into register pair HL.
MAKINT	105-106	Entry point of the routine that sets the integer value of register pair HL into the FAC.
VERS	107	The version number of BASIC. For version 1.0, contains the value 10H.
CHRSTT	110-111	Start code for user defined characters.
CHRADR	112-113	Address containing user defined character patterns.
BLDADR	114-115	Address at which machine language routines are loaded when BLOAD is executed.
BLDLNG	116-117	Number of bytes of data loaded when BLOAD is executed.
BLCHKF	118	Flag which determines whether addresses are checked (to ensure that they are within the machine language program area) when BLOAD or BSAVE is executed. Addresses are checked when this byte is set to 0; for any other value, addresses are not checked.
CUSIGN	119	The currency symbol used when PRINT USING "\\######" is executed.
LPWAIT	11A	The wait time for print not ready errors when LPRINT or LLIST is executed. The default is 30; thus, a DT error occurs if the printer does not become ready within 30 seconds of executing LPRINT or LLIST. For 0, the wait time is indefinite; for any other value, it is the specified number of seconds.
RSWAIT	11B	The wait time for communication interface not ready errors when COMn is opened or serial output is attempted with any of the control line check functions set to ON. The default value is 30. For 0, the wait time is indefinite; for any other value, it is the specified seconds.
DEVNAM	11C-11D	Pointer for expanded sequential devices. To expand sequential devices, rewrite this pointer and register the device name.
DCBTAB	11E-11F	Pointer to the DCB for expanded sequential devices.

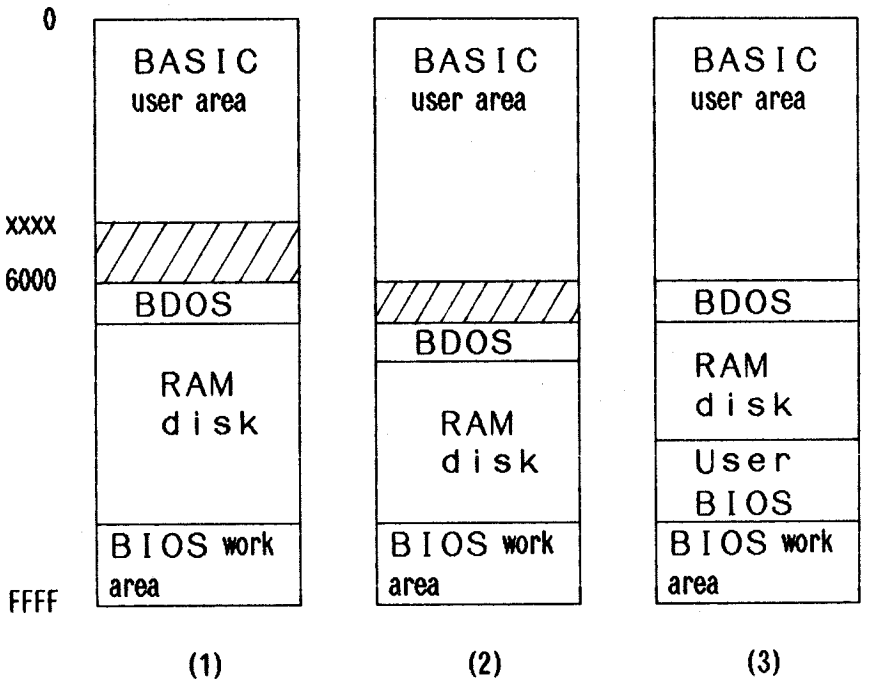
Chapter 2 Interfacing with Machine Language Programs

Chapter 2 Interfacing with Machine Language Programs

2.1 Reserving an Area for Machine Language Programs

There are three ways of reserving an area for machine language programs; these are as follows.

(1) Reduce the size of the BASIC user area with the CLEAR statement and use the area which is not accessed by BASIC.
(2) Change the BDOS entry point to a location above 6000H (by changing the size of CP/M) and use the area from 6000H to the BDOS entry point.
(3) Use a user BIOS area.



If the BDOS entry address is lower than 6000H, the first method allows addresses up to that preceding the BDOS entry address to be used.

2.2 The BLOAD and BSAVE Statements

The syntax of the BLOAD and BSAVE statements is as follows.

```
BLOAD <file descriptor>[,<load address>[,R]]
BSAVE <file descriptor>,<start address>,<length>
```

Any file created with BSAVE has a 5-byte header, and only files with this header can be loaded with BLOAD. The header format is as follows.

Offset	Data	
Ø	FD	Indicates a file which can only be accessed by the BLOAD or BSAVE statements.
1	Address (low)	The start address specified with BSAVE; when BLOAD is executed, this is the load address used unless another load address is explicitly specified.
2	Address (high)	
3	Length (low)	The number of bytes of memory saved by the BSAVE statement. This length does not include the 5-byte header.
4	Length (high)	
5 on	Data	

No matter what load address is specified, BLOAD cannot be used to load code into any area except the BASIC user area or the user BIOS area. (However, this restriction does not apply if a value other than Ø is set in BLCHKF, at address 118.)

Further, if BLOAD is executed with the R option attached, the load address and program length can be determined from within the machine language program by referencing BLDADR (addresses 114 and 115) and BLDLNG (addresses 116 and 117). This is handy when making machine language routines relocatable.

Chapter 3 Added Commands

The reserved word EXTD is provided to make it possible to add new statements or commands. When EXTD is to be used as a statement, use entry SEXTD; when it is to be used as a function, use entry FEXTD.

	Address	
FEXTD	142 - 143	EXTD function entry point
SEXTD	14A - 14B	EXTD statement entry point

The text pointer value is passed to the routines at these addresses via register pair HL. Initially, both entry points contain the address of the SN error. Since both routines are called directly from ROM by the BASIC interpreter, any addresses subsequently set in these entry points must be lower than 6000H if the added statement or function is to be intrepreted using the the syntax analysis routines which are described next.

3.1 Syntax Analysis Routines

This section describes some of the BASIC syntax analysis routines.

(1) SYNCHK

Address: 0008H (or C4BCH)
Input: HL=text pointer
Output: Same as CHRGET (see below)
Explanation: Checks statement syntax, and is called as follows.

```
CALL  SYNCHK
DB    XX
```

Here, XX is the character to be checked. If the character indicated by the text pointer is XX, the text pointer is advanced and the routine returns. If it is a different character, an "SN Error" occurs automatically.
SYNCHK is compatible with the following routine.

```
LD A, (HL) : HL= text pointer
EX (SP), HL
CMP (HL)
JP NZ, SNERR : give "SN Error"
INC HL
EX (SP), HL
JP CHRGET
```

(2) CHRGET

Address: 0010H (or 6CF5H)
Input: HL=text pointer
Output: A=character
HL=text pointer
Z flag=1 (when the end of a logical line is reached)
Explanation: Returns the character following that indicated by the text pointer in register A. Spaces are skipped, and the routine returns with the Z flag set to 1 when the text pointer reaches the end of a logical line.

(3) CHROUT

Address: 0018H (or AD50H)

Input: A=character

Explanation: Outputs one character to the currently selected device. (Ordinarily, this is the console.)

(4) COMPAR

Address: 0020H (or C4B6H)

Input: HL, DE

Output: Z flag, CY flag
When HL>DE ... Z=0, CY=0
When HL=DE ... Z=1, CY=0
When HL<DE ... Z=0, CY=1

Explanation: Compares the contents of the HL and DE registers. Contents of the A register are changed.

(5) GTTYPE

Address: 0028H (or 774FH)

Input: None

Output: Flags

	Z	CY	P	S
Integer	0	1	E	M
String	1	1	E	P
Single precision	0	1	O	P
Double precision	0	0	E	P

Explanation: Uses flags to return the type of the FAC contents.

(6) FORMUL

Address: 7409H

Input: HL=text pointer

Output HL=text pointer

Explanation: Evaluates an expression and sets the result in the FAC.

(7) FRCINT

Address: 83C9H

Output: HL=integer value

Explanation: Converts the contents of the FAC to an integer value and returns the result in register HL.

(8) BYTES

Address: 7A48H

Input: HL=text pointer

Output: E=A=value (0 to 255)
HL=text pointer
Z flag=1 ... When the end of a logical line is reached.

Explanation: This routine evaluates an expression and, when it is a numeric expression, returns the resulting value in register E. An "FC Error" results if it is other than a numeric expression or the result is 256 or more. BYTES is compatible with the following routine:

```
CALL FORMUL  ;evaluate formula
PUSH HL      ;save text pointer
CALL FRCINT  ;convert to integer value
EX   DE,HL   ;integer to DE
POP  HL      ;restore text pointer
MOV  A,D     ;get high order
OR   A,A     ;is it Ø?
JP   NZ,FCERR;no, give "FC Error"
DEC  HL      ;back text pointer
CALL CHRGET  ;set condition on terminator
MOV  A,E     ;return result in A and E
RET
```

(9) MAKINT

Address: 8426H
Input: HL=integer
Explanation: Sets an integer value in the FAC.

(1Ø) GETSTR

Address: C94FH
Output: HL=string descriptor
Explanation: Takes the string descriptor out of the FAC and places it in register HL. A "TM Error" automatically results if the data type is other than string.
Example: Evaluating an expression and placing the string length in A and the string address in DE.

```
CALL FORMUL  ;formula evaluate
PUSH HL      ;save text pointer
CALL GETSTR  ;get string string descriptor
MOV  A,(HL)  ;A=string length
INC  HL
MOV  E,(HL)
INC  HL
MOV  D,(HL)  ;DE=address
POP  HL      ;restore text pointer
RET
```

(11) ERROR

Address: 6648H
Input: E=error code
Explanation: Branches to the BASIC error handler.

Reserved words

A	AUTO	AND	ABS	ATN	ASC	ALARM		
B	BEEP	BLOAD	BSAVE					
C	CLOSE	CONT	CLEAR	CINT	CSNG	CDBL	CVI	CVS
	CVD	COS	CHR$	CALL	COMMON	CHAIN	CLS	COPY
	CSRLIN	COM						
D	DELETE	DATA	DIM	DEFSTR	DEFINT	DEFSNG	DEFDBL	DEF
	DAY	DATE	DSKF					
E	ELSE	END	ERASE	EDIT	ERROR	ERL	ERR	EXP
	EOF	EQV	EXTD					
F	FOR	FIELD	FILES	FN	FRE	FIX	FONT	
G	GOTO	GOSUB	GET					
H	HEX$							
I	INPUT	IF	INSTR	INT	INP	IMP	INKEY$	
J								
K	KILL	KEY						
L	LPRINT	LLIST	LPOS	LET	LINE	LOAD	LSET	LIST
	LOCATE	LOGIN	LOG	LOC	LEN	LEFT$	LOF	
M	MERGE	MOD	MKI$	MKS$	MKD$	MID$	MENU	MOUNT
	MOTOR							
N	NEXT	NAME	NEW	NOT				
O	OPEN	OUT	ON	OR	OCT$	OPTION	OFF	
P	PRINT	PUT	POKE	POS	PEEK	PSET	PRESET	POINT
	PCOPY	POWER						
Q								
R	RETURN	READ	RUN	REMOVE	REM	RESUME	RSET	RIGHT$
	RND	RENUM	RESET	RANDOMIZE				
S	STOP	SWAP	SAVE	SPC(	STEP	SGN	SQR	SIN
	STR$	STRING$	SPACE$	SYSTEM	SOUND	SCREEN	STAT	
T	THEN	TRON	TROFF	TAB(	TO	TAN	TIME	TITLE
	TAPCNT							
U	USING	USR						
V	VAL	VARPTR						
W	WIDTH	WAIT	WHILE	WEND	WRITE			
X	XOR							
Y								
Z								

Statements

80 -	A0 -	C0 - FIELD	E0 - USR
81 - END	A1 - WIDTH	C1 - GET	E1 - FN
82 - FOR	A2 - ELSE	C2 - PUT	E2 - SPC(
83 - NEXT	A3 - TRON	C3 - CLOSE	E3 - NOT
84 - DATA	A4 - TROFF	C4 - LOAD	E4 - ERL
85 - INPUT	A5 - SWAP	C5 - MERGE	E5 - ERR
86 - DIM	A6 - ERASE	C6 - FILES	E6 - STRING$
87 - READ	A7 - EDIT	C7 - NAME	E7 - USING
88 - LET	A8 - ERROR	C8 - KILL	E8 - INSTR
89 - GOTO	A9 - RESUME	C9 - LSET	E9 -
8A - RUN	AA - DELETE	CA - RSET	EA - VARPTR
8B - IF	AB - AUTO	CB - SAVE	EB - INKEY$
8C - RESTORE	AC - RENUM	CC - RESET	EC - OFF
8D - GOSUB	AD - DEFSTR	CD - CLS	ED -
8E - RETURN	AE - DEFINT	CE - LOCATE	EE -
8F - REM	AF - DEFSNG	CF - BEEP	EF - >
90 - STOP	B0 - DEFDBL	D0 - SOUND	F0 - =
91 - PRINT	B1 - LINE	D1 -	F1 - <
92 - CLEAR	B2 - BLOAD	D2 -	F2 - +
93 - LIST	B3 - BSAVE	D3 - PSET	F3 - -
94 - NEW	B4 - WHILE	D4 - PRESET	F4 - *
95 - ON	B5 - WEND	D5 -	F5 - /
96 -	B6 - CALL	D6 -	F6 - ^
97 - WAIT	B7 - WRITE	D7 -	F7 - AND
98 - DEF	B8 - COMMON	D8 -	F8 - OR
99 - POKE	B9 - CHAIN	D9 - COPY	F9 - XOR
9A - CONT	BA - OPTION	DA - KEY	FA - EQV
9B -	BB - RANDOM	DB - COM	FB - IMP
9C -	BC -	DC - TO	FC - MOD
9D - OUT	BD - SYSTEM	DD - THEN	FD -
9E - LPRINT	BE -	DE - TAB(	FE -
9F - LLIST	BF - OPEN	DF - STEP	FF - [function]

Functions

Functions are composed of two bytes. The first of these is always FFH, and the second is one of the codes from the list below.

80 -	A0 -	C0 -	E0 - ALARM
81 - LEFT$	A1 -	C1 -	E1 - WIND
82 - RIGHT$	A2 -	C2 -	E2 - EXTD
83 - MID$	A3 -	C3 -	E3 - MOTOR
84 - SGN	A4 -	C4 -	E4 - FONT
85 - INT	A5 -	C5 -	E5 -
86 - ABS	A6 -	C6 -	E6 -
87 - SQR	A7 -	C7 -	E7 -
88 - RND	A8 -	C8 -	E8 -
89 - SIN	A9 -	C9 -	E9 -
8A - LOG	AA -	CA -	EA -
8B - EXP	AB -	CB -	EB -
8C - COS	AC -	CC -	EC -
8D - TAN	AD -	CD -	ED -
8E - ATN	AE -	CE -	EE -
8F - FRE	AF -	CF -	EF -
90 - INP	B0 - LOC	D0 - CSRLIN	F0 -
91 - POS	B1 - LOF	D1 - POINT	F1 -
92 - LEN	B2 - MKI$	D2 - DAY	F2 -
93 - STR$	B3 - MDS$	D3 - DATE	F3 -
94 - VAL	B4 - MKD$	D4 - TIME	F4 -
95 - ASC	B5 -	D5 - SCREEN	F5 -
96 - CHR$	B6 -	D6 - DSKF	F6 -
97 - PEEK	B7 -	D7 - MENU	F7 -
98 - SPACE$	B8 -	D8 - LOGIN	F8 -
99 - OCT$	B9 -	D9 - TITLE	F9 -
9A - HEX$	BA -	DA - STAT	FA -
9B - LPOS	BB -	DB - PCOPY	FB -
9C - CINT	BC -	DC - MOUNT	FC -
9D - CSNG	BD -	DD - POWER	FD -
9E - CDBL	BE -	DE - REMOVE	FE -
9F - FIX	BF -	DF - TAPCNT	FF -

Chapter 4 Interfacing with Sequential Access Devices

Chapter 4 Interfacing with Sequential Access Devices

With sequential access devices (such as an external cassette recorder), file I/O is done based on information contained in an area called the device control block (DCB). One DCB is required for each device such as CASØ and COMØ.

To add sequential access devices, the user must set up an appropriate DCB and register the name under which the device is to be accessed. Registering the name requires rewriting DEVNAM, and setting up a DCB requires rewriting DCBTAB.

Name	Address	
DEVNAM	11C - 11D	Pointer to the expansion device's device name.
DCBTAB	11E - 11F	Pointer to the expansion device's DCB.

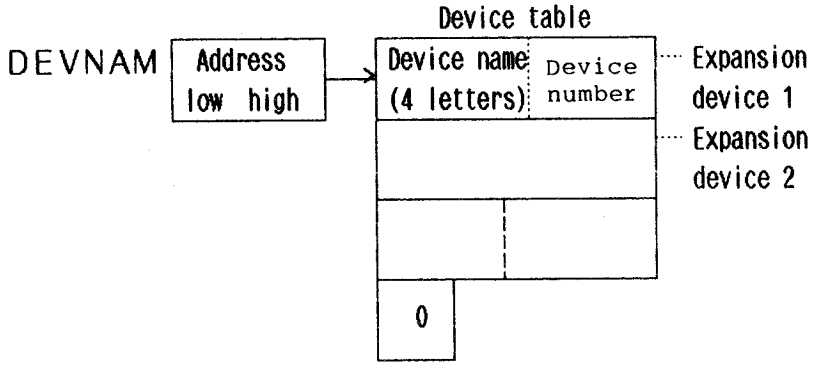
4.1 Device Table

The device table is the table in which device names and device numbers are registered. When adding a device, its device name and device number must be registered in this table.

BASIC supports eight sequential access devices; their device names and device numb r are as follows.

Device name	Device number
KYBD	FFH
SCRN	FEH
LPTØ	FDH
COMØ	FCH
COM1	FBH
COM2	FAH
COM3	F9H
CASØ	F8H

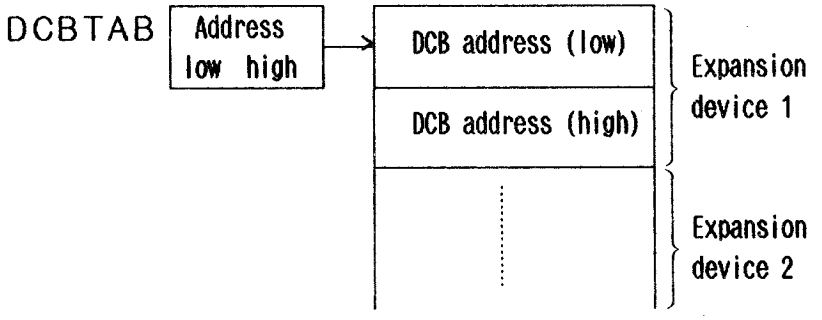
A 4-letter device name and a corresponding device number are required in order to register any extra device. Device numbers should be used in descending sequence, starting with F7H. BASIC recognizes the new device name when the starting address of the device table is stored in DEVNAM (at addresses 11C and 11D).



The device table must always end with "Ø".

4.2 The DCB Table

The DCB table contains addresses of DCBs for individual devices. When adding devices, the address of DCBs for each added device name must be stored in this table. Further, the starting address of the DCB table must be stored in DCBTAB (addresses 11EH and 11FH).



4.3 The DCB (Device Control Block)

The DCB is the table which contains entry points to routines for opening and closing various devices and accessing them for data I/O. Nine entry points are required for each device. These entry points and corresponding routines are as follows.

Offset (size)	Name	Contents
0, 1 (2 bytes)	OPEN	Contains the address of the entry point to the routine for opening the device.
2, 3 (2 bytes)	CLOSE	Contains the address of the entry point to the routine for closing the device.
4, 5 (2 bytes)	OUTPUT	Contains the address of the entry point to the routine which outputs 1 byte.
6, 7 (2 bytes)	INPUT	Contains the address of the entry point to the routine which inputs 1 byte.
8, 9 (2 bytes)	LOC	Contains the address of the entry point to the device's LOC routine.
10, 11 (2 bytes)	LOF	Contains the address of the entry point to the device's LOF routine.
12, 13 (2 bytes)	EOF	Contains the address of the entry point to the device's EOF routine.
14, 15 (2 bytes)	PUT	Contains the address of the entry point to the routine which saves preread data.
16, 17 (2 bytes)	WIDTH	Contains the address of the entry point to the routine which saves the maximum number of bytes which can be output to the device in one line.

4.3.1 OPEN (OPEN Statement Support)

Entry parameters: D=open mode (1 for the "I" mode, 2 for the "O" mode)
HL=FCB starting address
(SP)=BASIC text pointer

Processing:
1. Checks whether the OPEN mode is correct. (For example, if the "O" mode is specified for an input-only device, a "Bad file descriptor" error occurs.)
2. Opens access to the specified device. If an error occurs, branches to the error routine specified in the BASIC program.
3. If open processing is successful, initializes the FCB pointer and FCB.
   (1) Sets the FCB address in FCBPTR (addresses 2B5H and 2B6H).
   (2) Initializes a sequential device area for the FCB.
   FCB+0 Open mode
   +2D Maximum number of characters/line (see WIDTH).
   +2E Initial character position for output (ordinarily 0).
   The maximum characters/line and character position settings have no meaning in the "I" mode.
4. POPs the text pointer into register pair HL from the stack, then returns.

Note: Options can be specified when the OPEN statement is executed. For example, executing

```
OPEN"I",#1,"DEV1:(ABC)"
```

places the option string in parentheses into DSCOPT (the 10 bytes from address 7B1H to 7BAH).

4.3.2 CLOSE (CLOSE Statement)

Entry parameters: D=open mode
(SP)=FCB starting address

Processing:
1. Closes access to the device.
2. POPs the FCB address from the stack, clears the 49 bytes starting at that address to 0, then returns.

4.3.3 OUTPUT (PRINT# Statement, Etc.)

Entry parameters: (SP)=character to be output

Processing:
1. POPs the character to be output from the stack and outputs it to the device.
2. POPs subsequent bytes from the stack into the PSW, BC, DE, and HL (in that order), then returns.

4.3.4 INPUT (INPUT# Statement, INPUT$ Function, Etc.)

Entry parameters: None

Processing:
1. Before actually input from the device, checks whether any data has been saved by PUT. If so, returns that data.
2. Inputs 1 character from the device.

Return parameters: A=input data
CY flag=1 - Indicates that there is no data to input. This occurs when (EOF) is encountered or CTRL+STOP is pressed.
=Ø - Indicates data was input normally.

4.3.5 LOC (LOC Function)

Entry parameters: None
Return parameters: FAC=LOC value

4.3.6 LOF (LOF Function)

Entry parameters: None
Return parameters: FAC=LOF value

4.3.7 EOF (EOF Function)

Entry parameters: None
Return parameters: FAC=Ø or -1
Ø: Not EOF
-1: EOF

Note: MAKINT is used to set values in the FAC (floating point accumulator). The entry point of MAKING is located at addresses 1Ø5H and 1Ø6H. When an integer is set in register pair HL, calling MAKINT sets that value in the FAC.

4.3.8 PUT (INPUT# Statement)

Entry parameters: C=data to be saved

Processing:
Saves subsequent data for reading by the INPUT# statement. Data is saved in the PUT buffer (a byte reserved by the user).

4.3.9 WIDTH (WIDTH "device" Statement)

Entry parameters: (SP)=maximum number of characters

Processing:
Saves the maximum number of characters to be output in one line in an area reserved by the user. This value is copied into the FCB and becomes effective when the device is opened for output. After the maximum number of characters has been output, CR and LF codes are output automatically.

If the WIDTH "device" statement is not executed, this routine is not called; therefore, some other measure must be taken to set the initial value. (If the value set is FFH, line length is unlimited and no CR or LF codes are automatically output.)

4.3.10 Error Processing

When an error occurs, the error code is set in the E register; BASIC then detects the error upon jumping to the ERROR routine. The ERROR routine starts at address 10AH. See the BASIC Reference Manual for the error codes.

```
Example:    MVI   E, 68    ;DU error code
            JMP   ERROR    ;GOTO Error handler
```

Procedure for detecting CTRL+STOP
Depression of CTRL+STOP can be detected by checking CSTOPFLG (at address F01AH); the value of this flag becomes other than 0 when CTRL+STOP is pressed. It is recommended that machine language routines be written so that this flag is checked and control returns to BASIC when CTRL+STOP is pressed.