

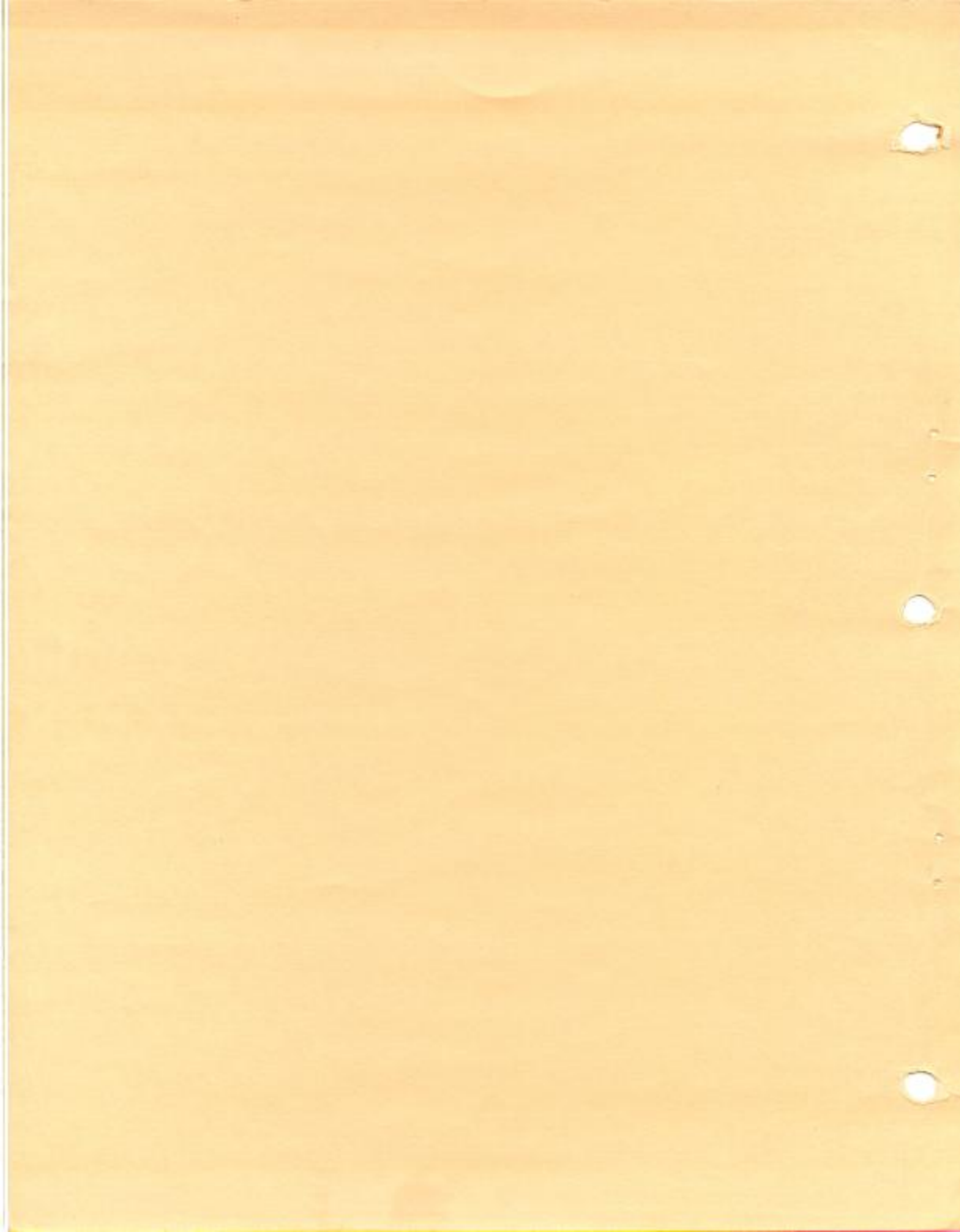
CROMEMCO
32K Structured Basic
Instruction Manual
Errata/Addendum

CROMEMCO, INC.
280 Bernardo Avenue
Mountain View, CA 94040

Part No. 023-0094

December 1979

COPYRIGHT © 1979
BY CROMEMCO, INC.
ALL RIGHTS RESERVED



ERRATA/ADDENDUM

Manual : Cromemco 32K Structured Basic
Publication Date : September 1979
Part No. : 023-0080

A reminder: 64K of memory is required to run Structured Basic.

BASICGEN

Basicgen is a powerful tool which allows the programmer to generate a version of Structured Basic which will occupy less than the 32K bytes required for the entire Basic interpreter. The space which is not used by Basic can be used for longer programs, larger arrays and strings, additional Basic-KSAM and normal disk buffers, etc.

The space is saved by not implementing those features of Structured Basic which are not required for a specific application. One system may not require Basic-KSAM access while another may be a run time only package which does not require the PRINT USING instruction.

Notice that a program with large arrays may be developed using shorter arrays and after the program runs properly, the DIMension statements may be changed to reflect the needed array size and the program then run under a run time only version of Sbasic.

The Sbasic which is generated by Basicgen will occupy approximately 2 to 3 K bytes more as a disk file than as a running program. This is because Basicgen saves this amount of the User Area on the disk. As an example, when Basicgen is used to create a full fledged Sbasic, the disk file will occupy 35K bytes while the same program (as supplied on your disk from Cromemco) can occupy 32K bytes. This difference is accounted for by space within Sbasic which may be used by a program (User Area.)

The following files must be present on the disk in the current drive in order for Basicgen to function properly:

B1.sbr	C2.sbr	C8.sbr
B2.sbr	C2a.sbr	C9.sbr
B3.sbr	C3.sbr	C9a.sbr
B4.sbr	C3a.sbr	C9b.sbr
B5.sbr	C4.sbr	Sbasic.sbr
Basicgen.com	C5.sbr	Sbasicio.sbr
Baslib.sbr	C6.sbr	
C1.sbr	C7.sbr	

If Basicgen cannot find any one of these files an error message will be generated and control will be returned to CDOS. If this happens the missing file must be transferred onto the disk and Basicgen executed again.

The most likely cause of this type of error is the failure of the user to rename the re-assembled Sbasicio routine from Sbasicio.rel to Sbasicio.sbr. Refer to the section of this Addendum on Modifying the Sbasic I/O Drivers for more information on this error.

To generate a new version of Sbasic enter the following command line in response to the CDOS prompt (not in Sbasic):

BASICGEN filename

where filename is the name of the file which is to be generated. Notice that no file name extension should be used with the file name, it will automatically receive an extension of COM. This is the name which will be used to call the newly generated version of Sbasic.

The Basicgen program will ask questions pertaining to the customized version of Sbasic which is to be generated. The proper response to each question is a Y or N for YES or NO respectively.

1. The first question is:

Will this be an interactive version
of SBASIC??

Notice that Basicgen prompts with a double question mark. If the user responds with an N a run-time only version of Sbasic will be generated. This will save about 5500 bytes of memory.

The only way to execute a program under a run time version of Sbasic is to declare the name of a saved Sbasic file in the CDOS command line. If RTBASIC.COM is the name of the generated run time version of Sbasic and INVENT.SAV is the name of a saved Sbasic file, the following CDOS command line will execute the program and return control to CDOS:

```
RTBASIC INVENT.SAV
```

Because the listing, editing, entering, and manual loading features are not implemented in a run time Sbasic the prompt (>>) is never displayed. The user must enter and edit a program using a version of Sbasic which is not designed for run time only use.

Notice that in this prompt the term interactive refers to the Sbasic interpreter and not to the saved program. A program running under a run-time only version of Sbasic may prompt the user and may be an interactive program.

2. The second question is:

Do you wish KSAM file access capability??

If the user responds with an N no Basic-KSAM instructions will be included in the version of Sbasic which is generated. This will save about 6500 bytes of memory.

The only instructions which are affected by the answer to this question are those listed in chapter 22 of the 32K Structured Basic Manual.

3. The third question is:

Do you wish the full text of error messages displayed??

If the user responds with an N the Sbasic which is generated will display only the appropriate error number (as listed in the appendix to the 32K Structured Basic Manual) when an error is generated. This will save about 1350 bytes of memory.

4. The fourth question is:

Do you wish editing capability??

This question will only be asked if the answer to question number one indicated that the generated Sbasic was not to be a run time only version. If the user responds with an N the In Line Basic Editor (as described on pages 74 through 78 of the 32K Structured Basic Manual) will not be included in this version of Sbasic. This will save about 770 bytes of memory.

5. The fifth question is:

Do you wish to include the PRINT USING instruction??

If the response to this question is an N, Print Using will not be implemented in the generated version of Sbasic. This will save about 900 bytes of memory.

6. The sixth question is:

Do you wish to allow user defined functions??

An N in response to this question will not allow user defined functions to be used in the generated version of Sbasic. No standard Sbasic functions will be affected by the response to this question. This will save about 190 bytes of memory.

7. The seventh question is:

Do you want to include the LOG and EXP functions and the power operator??

If the question is answered with an N, the Logarithm and Exponent functions will be inoperative and the power operator (** or ^) will not be available in the version of Sbasic which is being generated. Also, if the question is answered negatively, the square root and trigonometric functions will not be able to be implemented because they call subroutines common to the Log and Exp functions. This will save about 890 bytes of memory.

8. The eighth question is:

Do you wish to include the square root function??

An N in response to this question will disable the square root (SQR) function in Sbasic. This will save about 175 bytes of memory. Note that this question will not be asked if question number seven was answered negatively.

9. The ninth question is:

Do you wish to include the trigonometric functions??

N in response to this question will disable the trigonometric functions (SIN, COS, TAN, and ATN). This will save about 510 bytes of memory. Note that this question will not be asked if question number seven was answered negatively.

10. The last question is:

Do you wish to include the HEX, VALC, DATE,
and TIME functions and the EXPAND
instruction??

If the user types an N in response to this, the
last question, the named functions and instruction
will not be included in the generated version of
Sbasic. This will save about 400 bytes of memory.

ADDITIONAL FEATURES

File attributes may now be established or changed from Sbasic. The format of the attribute instruction is:

Attr file-ref, "parameters"

or

Atrib file-ref, "parameters"

In the above instructions file-ref is a file reference to the file whose attributes are to be altered. File-ref may be a string or a string literal enclosed in quotation marks. Notice that ambiguous file references may be used. Parameters is one or more of the permissible parameters (+, E, R, and W), or no parameter (a null string) at all. Notice that there is no difference between the two forms of the instruction.

Please refer to the CDOS Manual, Intrinsic Commands section for further information.

In order to maintain compatibility with CDOS, an additional file renaming facility has been added to Sbasic. The new instruction reverses the two parameters of the Sbasic RENAME instruction. As in CDOS, the new instruction is called REN. The format of the instruction is:

REN new-file-ref, old-file-ref

where the file-refs are strings or string literals enclosed in quotation marks.

The DSK command now has two additional permissible parameters:

DSK "*"

and

DSK "-dr"

If the command is given in the first form, the disk

drive motors will be turned off. They will automatically be turned on again when a disk is accessed.

In the second form, dr represents one of the allowable disk drive specifiers as described in the 32K Structured Basic Manual. When dr is preceded by a dash (minus sign) the disk in the specified drive will be ejected.

Two new functions have been added to take advantage of the time and date features of CDOS and the Cromemco 3102 terminal. To set the date and time give the instructions:

```
Dummy$ = Date$("yymmdd")
```

and

```
Dummy$ = Time$("hhmmss")
```

where yymmdd are digits representing the year, month, and day and hhmmss are digits representing the hour, minute, and second.

To cause the functions to return the date and time (which will only be accurate if the system includes a Cromemco 3102 terminal) they can be called as:

```
Print Date$("")
```

and

```
Print Time$("")
```

Note that the functions may be assigned to string variables, printed (as above), or called in any other manner a standard function may be called.

Input to Sbasic from the terminal now takes advantage of the INPUT BUFFERED LINE system call. This will not affect most programs. Sbasic programs using the input timer must use the old (character by character) input routine. This can be invoked by giving the command:

```
SET 14,1
```

A new driver has been added to Sbasic with the name:

```
$COnsole
```

If this file is OPENed, a PUT instruction can invoke the CDOS call number 142 (8E hex), SET SPECIAL CRT FUNCTION:

```
OPEN\1\"$CONSOLE"  
PUT\1,d,e\ [list]
```

where d is the contents of the D register and e is the contents of the E register prior to the system call. List is the optional list of information to be displayed on the console after the system call. If e is omitted, it assumes a default value of zero.

Note that PRINT, GET, and INPUT instructions may also be used with this file following the format described above. Please refer to the CDOS manual, Programmer's Guide, system call 142, for more information.

Two additional instructions are now available for use with Basic-KSAM files:

```
KGET\pfn\var-1,...,var-n  
KPUT\pfn\exp-1,...,exp-n
```

where pfn is a Primary Data file number, var-1 through var-n is a list of variables, and exp-1 through exp-n is a list of expressions or variables.

These instructions may be used after the Current Record Pointer (CRP) has been positioned and

(optionally) after part of the current record has been written (KPUT) or read (KGET). They will cause additional fields (variables) in a long record to be written to or read from the same (current) record.

For example, the following two sets of instructions are equivalent:

```
100 Kgetfwd\1\Id'number, Year, Amount, Value
```

and

```
100 Kgetfwd\1\  
110 Kget\1\Id'number, Year  
120 Kget\1\Amount  
130 Kget\1\Value
```

A source listing of the Sbasic Input/Output routines is included at the end of this addendum. A copy of the Z80 source code for these I/O drivers is included on the Sbasic diskette under the file name SBASICIO.Z80.

Please make the following corrections to the manual:

Page : 64

Add : Add the following line to the Statistical Analysis Program:

225 If Variance < 0 Then Variance = 0

This will compensate for round-off error in the event that all numbers which are entered are the same.

Page : 73, second example

From : Dir "A:*,*"

To : Dir "A:"

It is no longer necessary to follow a drive designation in this command by *.*. The drive specification alone will cause all files on that drive to be cataloged.

Page : 86, middle of page

From : (if L2 is not specified, L1=L1)

To : (if L2 is not specified, L2=L1)

Page : 184, Example Program, Line 110

From : 110 Dim A\$(21)

To : 110 Dim Text\$(21)

Page : 151, paragraph 4

From : All normal PRINT functions (such as TAB, SPC, semicolon, and comma) are overridden by the PRINT USING instruction.

To : All normal PRINT functions (such as TAB, SPC, and comma) are overridden by the PRINT USING instruction. The terminating semicolon will still suppress the generation of a new line.

Page : 235, 2nd line

From : format: SET aexp-1, aexp-2

To : format: [Ln] Set aexp-1, aexp-2

Note that this instruction may be used as a statement as well as a command (i.e., it may be used with a line number.)

Page : 240, 1st and 5th lines

From : instruction

To : function

Page : 242, 2nd line

From : format: OUT m,b

To : format: [Ln] OUT m,b

Note that this instruction may be used as a statement as well as a command (i.e., it may be used with a line number.)

Page : 244, 2nd line

From : format: POKE m,b

To : format: [Ln] POKE m,b

Note that this instruction may be used as a statement as well as a command (i.e., it may be used with a line number.)

Page : 254, 5th paragraph, 5th line

From : reference to

To : definition of

Page : 254, 6th paragraph, 3rd line

From : referenced.

To : defined.

Page : 263, first paragraph of Note, next to last line

From : Procedures.

To : files.

Page : 306, next to last line

From : 2304

To : 2560

Page : 328, last 3 lines

From : 40 Kopen \1\ Voll\$,Vol2\$,Vol3\$,Vol4\$

Because Vol3\$ is a null string a 2
volume file (Voll\$ and Vol2\$) are
opened.

To : 40 Kaltopen \1\ Voll\$,Vol2\$,Vol3\$,Vol4\$

Because Vol3\$ is a null string a 2
volume file (Voll\$ and Vol2\$) is
opened.

MODIFYING THE SBASIC I/O DRIVERS

A source listing of the Sbasic Input/Output routines follows. A copy of the Z80 source code for these I/O drivers is included on the Sbasic diskette under the file name SBASICIO.Z80.

Users who have the need and desire to modify these routines may modify this code, assemble it (using the Cromemco Z80 Macro Assembler), and replace the file named SBASICIO.SBR on the Sbasic diskette with the modified version.

When Basicgen is executed the new file will be used to supply the I/O routines and the user's modified drivers will automatically be incorporated into the generated Sbasic.

Notice that after assembly of the file SBASICIO.Z80 the file SBASICIO.REL will be created on the disk. This file must be renamed before executing Basicgen so that it may be properly incorporated into the generated version of Sbasic. This may be done from CDOS as follows:

```
Ren SBASICIO.SBR=SBASICIO.REL
```

Please refer to the following source listing itself for additional information.

THE FOLLOWING ROUTINES AND TABLES CONSTITUTE SBASIC'S INTERFACE
TO THE VARIOUS PERIPHERAL DEVICES.

FROM SBASIC A PERIPHERAL DEVICE IS TREATED AS IF IT WERE AN
ORDINARY DISK FILE WITH THE FOLLOWING EXCEPTIONS:

1. The 'file' (peripheral) name must start with a dollar sign.
2. Peripheral names must be unique within the next two characters.
(i.e., "\$LPT" and "\$LPR" are seen as the same name.)
3. Many functions within SBASIC can only refer to disk files;
examples include RENAME, ERASE, and CREATE (it doesn't
really seem logical to be able to ERASE a line printer,
anyway).
4. Specific peripherals may or may not support all I/O operations;
one cannot GET data from a line printer, for example.

EXTERNAL ROUTINES WHICH MAY BE USED BY THE I/O ROUTINES

```
0027     EXT    PSHALL ; SAVE ALL REGISTERS (INCLUDING IX,IY,HL',ETC.)
0028           ;     ON CPU STACK
0029     EXT    POPALL ; RESTORE ALL REGISTERS
0030           ;     NOTE: THESE ROUTINES MUST BE ACCESSED BY A 'CALL'
0031           ;     DO NOT 'JP' TO THESE ROUTINES
0032     EXT    UMULT  ; UNSIGNED MULTIPLY
0033           ;     BC * DE => HL,DE (16 BIT RESULT IN DE)
0034           ;     IF HL>0 (32 BIT RESULT) ROUTINE RETURNS 'NZ'
0035           ;     [USE CDOS CALL FOR DIVIDE, IF NEEDED.]
0036     EXT    ERTIME ; TIME-OUT ERROR FOR TIMED INPUT. DO NOT USE.
0037     EXT    ERRIO  ; GENERAL I/O ERROR. TO USE, PLACE DESIRED ERROR
0038           ;     CODE IN THE A REGISTER AND CALL ERRIO.
0039           ;     NOTE: IF ERROR CODE IS 0-127, THE RESULTANT
0040           ;     ERROR WILL NOT BE TRAPPABLE FROM THE
0041           ;     SBASIC PROGRAM.
0042           ;     NOTE: ALSO SEE BELOW FOR SPECIAL ERROR RETURNS
0043           ;     FROM SOME FUNCTIONS.
```

ENTRY POINTS WITHIN THIS MODULE

```
*****
0050     ENTRY  DDLIST ; MASTER DEVICE DRIVER TABLE
0052     ENTRY  SYGETC ; GET-CHARACTER-FROM SYSTEM CONSOLE
0053           ;     NOTE: NOT NORMALLY USED FOR PROGRAM EDITING
0054           ;     AND 'INPUT'. USED BY DRIVER "$CONSOLE"
0055           ;     AND BY 'GET' STATEMENT ('GET\0\...')
0056           ;     AND ALWAYS USED IF 'SET 14,1' USED IN SBASIC.
0058     ENTRY  SYPUTC ; PUT-A-CHARACTER TO SYSTEM CONSOLE. USED
0059           ;     FOR ALL CONSOLE OUTPUT.
0060     ENTRY  CHSTAT ; CHECK CONSOLE STATUS. USED FOR CHECKING TO SEE
0061           ;     IF A KEY (PARTICULARLY AN ESCAPE) HAS BEEN PUSHED.
```

FIXED OFFSETS AND RAM TABLE LOCATIONS WITHIN SBASIC WHICH ARE USED
BY I/O ROUTINES.

THE EXTENDED FILE CONTROL BLOCK (EFCB)
EACH OPEN FILE IS ASSIGNED AN EFCB (AND ALL CALLS RECEIVE ADDRESS
OF THIS EFCB IN THE IY REGISTER). CERTAIN PARAMETERS ARE PASSED
TO THE I/O ROUTINES VIA THESE BLOCKS.

(0000)	0076	EFCB	EQU	0	; THE BASE ADDRESS
(0000)	0077	EFCBUS	EQU	0	; EFCB-IN-USE FLAG: SET AND RESET BY SYSTEM!
	0078				; DON'T TOUCH!
(0001)	0079	EFCBDA	EQU	1	; ADDRESS OF THE DEVICE DRIVER CURRENTLY USING
	0080				; THIS EFCB. NOTE THAT THIS IS THE ADDRESS
	0081				; OF THE HEAD OF ITS TABLE (SEE BELOW).
(0003)	0082	EFCBDD	EQU	3	; DEVICE DEPENDENT INFORMATION -- THESE TWO BYTES
	0083				; ARE COPIED FROM THE DEVICE'S ENTRY IN DDLIST.
(0005)	0084	EFCBP1	EQU	5	; PARAMETER 1 ... PASSED AT 'OPEN' TIME VIA
	0085				; 'OPEN\CHANNEL,PARM1,PARM2\...' FROM SBASIC.
(0007)	0086	EFCBP2	EQU	7	; PARAMETER 2 ... PASSED BY 'OPEN', AS ABOVE.
(0009)	0088	EFCBS1	EQU	9	; STATUS PARAMETER 1 ... PASSED AT I/O TIME VIA
	0089				; GET, PUT, PRINT, INPUT ... FOR EXAMPLE:
	0090				; 'GET\CHANNEL,STATUS1,STATUS2\...' FROM SBASIC.
(000B)	0091	EFCBS2	EQU	11	; STATUS PARAMETER 2 ... SIMILAR TO EFCBS1.
(000D)	0093	EFCBFREE	EQU	13	
(00BF)	0094	EFCBTOP	EQU	191	; THE REST OF THE 192 BYTES OF THE EFCB IS AVAILABLE
	0095				; TO THE DRIVER FOR TEMPORARY STORAGE, BUFFERED I/O,
	0096				; OR WHATEVER IS NECESSARY.

Equates for CDOS system calls, etc.

(0002)	0101	SYSPUTC	EQU	2	; 'PUT C' CHARACTER TO SYSTEM CONSOLE
(000B)	0102	SYSRDY	EQU	11	; CHECK IF CHARACTER READY AT SYSTEM CONSOLE
(0080)	0103	SYSRNE	EQU	12B	; READ ONE CHARACTER FROM SYSTEM CONSOLE, WITHOUT ECHO
(008E)	0104	SYS CRT	EQU	142	; Special CRT Function (cursor addressing, ffields, etc.)
(0003)	0106	SYSRDR	EQU	3	; GET BYTE FROM 'READER' DEVICE
(0004)	0107	SYS PUN	EQU	4	; PUT BYTE TO 'PUNCH' DEVICE
(0005)	0108	SYS LPT	EQU	5	; PUT BYTE TO 'LIST' OR 'PRINTER' DEVICE

FIXED RAM LOCATIONS

CAUTION! VALID ONLY FOR VERSION 03.XX SBASIC.

	0118	ABS			
(0005)	0119 CDOS:	EQU	5		; SBASIC ALWAYS USES STANDARD CDOS ENTRY POINT!
0000	0122	ORG	0230H		
	0123 SETSYS:				; THE VARIOUS SET/SYS() PARAMETERS !
	0124				; SEE SBASIC MANUAL FOR EXPLANATION OF USAGE FROM
	0125				; SBASIC...SOME OF THESE MAY BE USEFUL TO
	0126				; AN I/O DRIVER. PARAMETERS THAT SHOULD NOT
	0127				; BE CHANGED HAVE NOT BEEN NAMED HERE.
0230 (0002)	0128 PAGESIZE	DS	2		; MAX CHARACTERS PER LINE FOR OUTPUT...IF MSB
	0129				; IS SET, PAGESIZE IS INFINITE.
0232 (0002)	0130 TABSIZE	DS	2		; NUMBER OF COLUMNS BETWEEN PRINT POSITIONS WHEN
	0131				; USING 'PRINT A,B,...'
0234 (0002)	0132 LASTCHAR	DS	2		; LAST CHARACTER OUTPUT TO ANY DEVICE (LSB ONLY)
	0133				;
0236 (0002)	0134 LASTERR	DS	2		; LAST RUNTIME ERROR NUMBER IS IN LSB. MSB NOT USED
	0135				; BY SYSTEM, BUT MOST USER PROGRAMS EXPECT
	0136				; IT TO BE ZERO.
0238 (0002)	0137 COLUMN	DS	2		; CURRENT PRINT COLUMN. RESET TO ZERO ONLY BY CR
	0138				; (BUT I/O ROUTINES COULD CHANGE IF WANTED)
023A (0002)	0139 TIMER	DS	2		; INPUT TIMEOUT COUNTER.
	0140				;

THE FOLLOWING LOCATIONS REFERENCE INTERNAL SBASIC FLAGS, ETC.
\$CONSOLE USES THEM TO PROVIDE I/O TRULY COMPATIBLE WITH 'SYGETC'
(SEE BELOW). NO DOCUMENTATION OF USAGE IS PROVIDED. USER-WRITTEN
DRIVERS SHOULD NOT USE THESE LABELS.

(031D)	0148 FLGECB	EQU	031DH
(0260)	0149 MODECH	EQU	0260H
(0267)	0150 MODFIL	EQU	0267H
(021F)	0151 MODRUN	EQU	021FH

0154 REL ; BEGIN ACTUAL CODE GENERATION

DDLIST -- a table of names, entry point addresses, etc. of all peripheral device drivers available to SBASIC users.

DDLIST consists of a series of 8-byte entries, each defining a separate device driver. The format of each entry must follow certain rules. The offsets of required information within each entry is shown below:

OFFSET (bytes)	INFORMATION
0 (2)	the device name...without the '\$' that the SBASIC user needs.
2 (2)	the address of the driver that will handle I/O to this named device.
4 (2)	device dependent ... each driver may designate its own usage for these bytes. NOTE: SBASIC will move these bytes into the bytes labeled EFCBDD within the appropriate EFCB before calling the driver's OPEN routine.
6 (2)	RESERVED for future use. Should be zero.

the actual DDLIST:

0186 DDLIST:

first entry

0000' 434P	0190	db	'CO'	; "\$Console" -- the system console driver
0002' 3300'	0191	dw	DRCONSOLE	; address of the driver for this device
0004' 0000	0192	db	0,0	; driver does not use device dependent info
0006' 0000	0193	dw	0	; --reserved--

next entry

0008' 5359	0197	db	'SY'	; "\$System console" -- historical name
	0198			; NOTE: this driver differs from \$CONSOLE in that
	0199			; it directly accesses the 4PDC/TUART port
	0200			; rather than going thru CDOS. CAUTION!!
000A' AC00'	0201	dw	DRTUART	; address of the general purpose TUART driver
000C' 00	0202	db	00H	; first EFCBDD byte: port address !
000D' 00	0203	db	0	; second EFCBDD byte: do not set baud rate on OPEN
000E' 0000	0204	dw	0	; --reserved--

next entry

0010' 5435	0208	db	'T5'	; "\$T50" -- tuart port 50H -- used in some systems
	0209			; as a serial line printer.
0012' AC00'	0210	dw	DRTUART	; note that the driver address is the same as that

```
0014' 50      0211      db      50H      ;      used by "SSY" !!!
0015' 00      0212      db      0          ; first EFCBDD byte: port address is 50H.
0016' 0000    0213      db      0          ; second EFCBDD byte: do not set baud rate on OPEN
                                dw      0          ; --reserved--

next entry

0018' 4C50    0218      db      'LP'      ; "$LPT" -- the line printer
001A' 0401'   0219      dw      DRLPT     ; address of its driver
001C' 0000    0220      db      0,0       ; no EFCBDD info
001E' 0000    0221      dw      0          ; --reserved--

next entry

0020' 5244    0225      db      'RD'      ; "$RDR" -- the CDOS 'reader device' driver
0022' 3901'   0226      dw      DRRDR     ; address of driver
0024' 0000    0227      db      0,0
0026' 0000    0228      dw      0

next entry

0028' 5055    0232      db      'PU'      ; "$PUNCH" -- the CDOS 'punch device' driver
002A' 2201'   0233      dw      DRPUNCH   ; address of driver
002C' 0000    0234      db      0,0
002E' 0000    0235      dw      0

no more entries

0030' 00      0238      db      0          ; *** this zero byte must be here to signal
                                dw      0239      ; the end of DDLIST ***
```

Just as the DDLIST must adhere to a certain format, a peripheral device driver must conform to SBASIC's rules.

The driver address specified at OFFSET 2 within any DDLIST entry must, in turn, point to a table of eight (8) two-byte addresses. Each address within this new table (known as the 'DRIVER ROUTINES TABLE') points to a sub-driver which implements a particular primitive I/O function, according to SBASIC's definition thereof.

The following shows the offset within the table (in bytes) of each subroutine address, a description of the necessary routine, and a list of what parameters are passed and where. The notation 'ERROR:' indicates what error message results if the A register contains a non-zero value upon return from the routine. Be sure to zero A if no error is encountered!!!!

OFFSET (bytes)	DESCRIPTION
0 (2)	* OPEN. This routine is called when the user OPENS the device from SBASIC. It should perform any needed initialization. EFCBDD contains the device dependent bytes from the DDLIST entry. EFCBP1 and EFCBP2 contain the parameters passed by the user's OPEN request (see above). The A ¹ register contains 0,1, or 2 -- a count of the parameters passed. (note: EFCBP1 and EFCBP2 will contain OFFFHH if the user did not specify a value.) ERROR: 134, Cannot open file.
2 (2)	* CLOSE. called when the file channel (number) is closed either implicitly or explicitly. The routine should perform any necessary cleanup (buffer flushes, etc.). NO parameters are passed. ERROR: 142, Cannot CLOSE file.
4 (2)	* SET STATUS OR POSITION. called at each usage of any file I/O statement (GET, PUT, PRINT, or INPUT). The parameters specified by the user (as in 'PUT \channel,param1,param2\ ...') are passed to SET STATUS for usage on a device-dependent basis. EFCBS1 and EFCBS2 contain the parameters (or OFFFHH if not specified). Register A ¹ contains the parameter count. ERROR: 140, File position/status.
6 (2)	GET STATUS OR POSITION. called by the IOSTAT() function. Usage is device dependent, and the second value of the function may be used to select one of several returned values. Register A ¹ contains the second value passed

to the IOSTAT function (e.g., "n" in
'IOSTAT(channel, n)'.
ERROR: 140, File position/status.

- 8 (2) OUTPUT ONE BYTE or character. Currently all I/O
to peripheral devices is performed on a
byte-at-a-time basis. If buffering is desired,
the driver should provide it.
The byte to output is passed in the A register.
(No error return exists for this function --
use ERRIO as noted above to force an error.)
- 10 (2) INPUT ONE BYTE or character. The byte is NOT masked
to 7-bits by GET, but is so masked by INPUT.
The input byte should be returned in the
A register.
(No error return exists for this function --
use ERRIO as noted above to force an error.)
- 12 (2) --RESERVED-- not currently in use. should be zero.
14 (2) --RESERVED-- not currently in use. should be zero.

NOTES: an asterisk denotes routines which MUST be included in
each driver!
for the optional routines, an omitted routine should be
be marked by an address of zero. A user attempt
to access such a routine (e.g., by trying to INPUT
from a line printer) will generate an ERROR: 130,
Invalid command for device.

dummy -- A Special Routine

Several of the device drivers have 'missing' routines...that is
they do not implement one or more of the 'required' routines
(open, close, and set status). However, SBASIC will not run
correctly with a driver if these routines are not supplied.
Since these routines are expected to return a zero in the
A register if no errors are encountered, the following routine
serves this purpose.

	0343 .dummy:			
0031' AF	0344	XOR	A	; this zeroes the A register...
0032' C9	0345	RET		; and this ensures a no-error routine to SBASIC.

the driver for \$CONSOLE includes the routines used for SBASIC's
primary console I/O.
EXCEPT: program entry from the keyboard and responses to the
INPUT statement use the CDOS read-a-buffered-line
system call. Usage of the CDOS routine may be
overridden by the user via a 'SET 14,1' command.
NOTE that the command sequence
 OPEN \n\ "\$CONSOLE"
 INPUT \n\ ...
will NOT use the CDOS routine, but will instead use
this driver.

0364 DRCONSOLE:

0033' 3100'	0366	dw	.dummy	; the CONSOLE does not require or use an OPEN routine
0035' 3100'	0367	dw	.dummy	; nor does it use a CLOSE routine
	0368			; but NOTE that the routines must exist,
	0369			; even if they are dummies !!!
0037' 9000'	0370	dw	CONSET	; the SET STATUS routine may be used to change
	0371			; cursor position, etc.
0039' 0000	0372	dw	0	; currently, the CONSOLE does not support a status
	0373			; read (cursor read?)
	0374			; NOTE that this routine may be omitted, and
	0375			; hence the use of the zero 'address'.
003B' 4300'	0376	dw	SYPUTC	; master system output-a-character routine
	0377			;
003D' 5100'	0378	dw	CONGETC	; get-a-character from console: almost identical
	0379			; to master system getc...see below.
003F' 0000	0380	dw	0	; --reserved--
0041' 0000	0381	dw	0	; --reserved--

end of driver table for \$CONSOLE

start of driver routines:

put-a-character

```
0394 SYPUTC:
0043' D5      0395   PUSH  DE      ; (NORMALLY NOT REQUIRED OF A DRIVER --
0044' C5      0396   PUSH  BC      ; SYSTEM DRIVER IS AN EXCEPTION -- SAVES TIME
0045' 3A3402  0397   LD    A,(LASTCHAR) ; AND SPACE MANY PLACES TO HAVE IT DO ITS
                   0398   ; OWN HOUSEKEEPING.)
0048' 5F      0399   LD    E,A    ; CDOS EXPECTS THE CHARACTER HERE
0049' 0E02    0400   LD    C,SYPUTC ; SYSTEM CALL FOR PUTC
004B' CD0500  0401   CALL  CDOS   ; OUTPUT THE CHARACTER

004E' C1      0403   POP   BC      ; RECOVER OUR OWN REGS...
004F' D1      0404   POP   DE
0050' C9      0405   RET                    ;AND QUIT.
```

GET-A-CHARACTER

CONGETC is specially designed to 'simulate' console I/O and hence must foo
SBASIC's get-a-line routine into believing that it called SYGETC
directly instead of via \$CONSOLE.

```
0416 CONGETC:
0051' 3A1D03  0417   LB    A,(FLGRCH)
0054' 326002  0418   LD    (MODECH),A
0057' 3EFP    0419   LD    A,0FFH
0059' 326702  0420   LD    (MODFIL),A ; GET-A-LINE IS PROPERLY POOLED
```

NOW CONGETC SIMPLY FALLS THRU TO SYGETC...SBASIC'S STANDARD GET-A-BYTE-PRO
CONSOLE ROUTINE (SEE NOTE ABOVE ABOUT 'SET 14,1').

NOTE: SYGETC DOES SEVERAL THINGS IN ITS OWN WAY: NOT THE LEAST OF
WHICH IS HANDLING THE INPUT TIMER TIMEOUT.

```
0429 SYGETC:
0430
(00AF) 0431 .TIME EQU 175 ; THIS IS THE FUNDAMENTAL WAIT-FOR-A-TENTH-OF-A-SECOND
                   0432 ; COUNTER VALUE...CHANGE THIS TO FIT YOUR SYSTEM
                   0433 ; IF YOU WISH (DIFFERENT MEMORY SPEEDS, DIFFERENT
                   0434 ; VERSION OF CDOS CAN AFFECT THIS).

005C' E5      0436   PUSH  HL    ; AGAIN, SYGETC DOES ITS OWN HOUSEKEEPING
005D' D5      0437   PUSH  DE
005E' C5      0438   PUSH  BC

005F' 3A1F02  0440   LD    A,(MODRUN) ; FLAG: >0 IF A PROGRAM IS RUNNING!
0062' B7      0441   OR    A
0063' 2B1F    0442   JR    Z,CHARRDY ; PROGRAM ENTRY...NO TIMED INPUT
0065' FA8     0443   JP    M,CHARRDY ; DIRECT STATEMENT EXECUTION...TO TIMING
```

<<< the CONSOLE driver >>>

```

0068' 2A3A02    0445    LD      HL,(TIMER)
0068' 7C        0446    LD      A,H
006C' B5        0447    OR      L      ; CHECK CURRENT VALUE OF TIMER
006D' 2E15     0448    JR      Z,CHARDY ; THE TIMER IS ALREADY ZERO.  MEANS ITS NOT ACTIVE.

```

READY TO DO TIMED INPUT

```

006F' 06AF     0453 TIMED:
0454    LD      B,.TIME

0456 SGCLOOP:      ; WAIT LOOP
0071' CDA600'   0457    CALL   CHSTAT ; CHECK FOR CHARACTER READY...SEE BELOW
0074' D7        0458    OR      A
0075' 200D     0459    JR      NZ,CHARDY ; STOP TIMING...A CHARACTER IS READY
0077' 10F8     0460    DJNZ   SGCLOOP ; OTHERWISE KEEP WAITING

```

MINOR TIMER (REG B == .TIME) HAS TIMED OUT...TICK OFF 1/10TH OF A SECOND

```

0079' 2B        0465    DEC     HL      ; COUNT DOWN BY ONE
007A' 223A02   0466    LD      (TIMER),HL ; AND PUT IT BACK FOR NEXT TIME
007D' 7C        0467    LD      A,H
007E' B5        0468    OR      L      ; AND CHECK ONCE AGAIN...
007F' CC0000#  0469    CALL   Z,ERTIME ; TIMER RAN OUT.  GENERATE USER-TRAPPABLE
0470      ; ERROR...NOTE THAT SYS(5) IS ALREADY SET TO 0 NOW
0471      ; SO FURTHER INPUTS WON'T BE TIMED.
0082' 18EB     0472    JR      TIMED  ; AND GO TRY FOR ANOTHER TIMER TICK

```

 TO HERE WHEN 1) CHARACTER IS READY OR 2) WHEN WE ARE NOT DOING TIMED INP

```

0477 CHARDY:
0084' 0E80     0478    LD      C,SYSRNE ; READ FROM CONSOLE WITH NO ECHO
0086' CD0500   0479    CALL   CDOS      ; (BECAUSE SBASIC DOES ITS OWN SPECIAL ECHOING)
0089' 323402   0480    LD      (LASTCHAR),A ; HISTORICAL
008C' C1        0481    POP    BC
008D' D1        0482    POP    DE
008E' E1        0483    POP    HL      ; CLEAN UP OUR HOUSE
008F' C9        0484    RET      ; AND BACK TO CALLER

```

 CONSOLE SET STATUS (SET CURSOR POSITION)

```

0490 CONSET:
0090' 08        0491    EX      AF,AF' ; GET COUNT OF PARAMETERS TO A REGISTER
0091' B7        0492    OR      A      ; IS COUNT ZERO?
0092' C8        0493    RET     Z      ; YES...SO WE DON'T DO A SET STATUS. NOTE THAT
0494      ; WE ARE RETURNING WITH A=0...IMPLYING
0495      ; THAT WE FOUND NO ERROR!

0093' 1E00     0497    LD      E,0    ; ( IN CASE THERE IS NO SECOND PARAMETER )

0095' FD5609   0499    LD      D,(IY+EPCBS1) ; D REGISTER GETS LSB OF FIRST PARAMETER...
0500      ; NOTE THAT NO CHECK IS PERFORMED
0501      ; AS TO LEGALITY OF THE VALUE...SEE

```

```
                                CDOS MANUAL FOR SIDE-EFFECTS.
0090' FE02      0502                ;
                                ; NOW CHECK: DID USER PASS A SECOND PARAMETER?
009A' 2003      0503      CP      2
                                ;
009C' FD5E0B    0504      JR      NZ,CSET2; NO...SO WE USE THE ZERO IN E-REGISTER AS PARM2
                                ;
                                ; E,(IY+EPCBS2) ; AND THIS IS LSB OF SECOND PARAMETER.
                                ;
                                ;
                                ;
0508 CSET2:
009F' 0E8E      0509      LD      C,SYSCRT ; REQUEST SPECIAL CRT FUNCTION
00A1' CD0500    0510      CALL   CDOS   ; ...FROM CDOS
                                ;
                                ;
00A4' AF        0512      XOR     A      ; ENSURE THAT A=0 (NO ERROR ENCOUNTERED)
00A5' C9        0513      RET     ; AND BACK TO USER
```

THE FOLLOWING ROUTINE IS REQUIRED BY SBASIC. IT IS THROUGH THIS
ROUTINE THAT SBASIC CHECKS FOR AN ESCAPE-KEY WHILE A PROGRAM
IS RUNNING. MODIFY AT YOUR OWN RISK!

```
                                ;
0520 CHSTAT:
00A6' 0E0B      0521      LD      C,SYSCRDY
00AB' CD0500    0522      CALL   CDOS   ; WE LET CDOS DO ALL THE WORK!
00AB' C9        0523      RET
```

DRTUART -- this is a fairly complete driver for serial I/O via
 a Cromemco TUART (or the serial port of the 4FDC).

FEATURES:

- One driver may be used to access any TUART port.
- The DDLIST may specify a default baud rate (to 9600 baud) or
 it may specify that the currently set baud rate remain
 unchanged.
- The SBASIC user may override any baud rate specified via the DDLIST.

```

0537 DRTUART:                                ; the driver table comes first
00AC' BC00'      0539      dw      TUOPEN      ; open a tuart port
00AE' 3100'      0540      dw      .dummy      ; no CLOSE routine needed
00B0' 3100'      0541      dw      .dummy      ; set status not supported
00B2' 0000'      0542      dw      0           ; get status not legal
00B4' E800'      0543      dw      TUPUTC      ; put-a-character to TUART port
00B6' F700'      0544      dw      TUGETC      ; get-a-character from TUART port
00B8' 0000'      0545      dw      0           ; --reserved--
00BA' 0000'      0546      dw      0           ; --reserved--
  
```

 OPEN a TUART serial port

```

0551 TUOPEN:
00BC' FD4E03     0552      LD      C,(IY+EPCBDD) ; THE FIRST BYTE OF THE DEVICE DEPENDENT INFO
0553            ; SPECIFIES THE TUART PORT ADDRESS.
00BF' FD7E04     0554      LD      A,(IY+EPCBDD+1) ; THE SECOND BYTE OF THE DEVICE DEPENDENT
0555            ; ENTRY...NOTE THAT SBASIC HAS MOVED IT FROM
0556            ; THE DDLIST INTO OUR OWN PERSONAL EPCB!
00C2' B7         0557      OR      A           ; DID THE DDLIST SPECIFY A DEFAULT BAUD RATE?
00C3' 280A      0558      JR      Z,TUOP2      ; NO...LEAVE THE BAUD RATE ALONE

00C5' ED79      0560      OUT     (C),A        ; SEND OUT BAUD RATE TO TUART...NOTE THAT
0561            ; THE DDLIST ENTRY IS EXPECTED TO CONTAIN
0562            ; THE CORRECT BIT PATTERN FOR THE DESIRED
0563            ; BAUD RATE !!! CAUTION !!!
00C7' 3B00      0564      LD      A,0           ; THIS IS A JUST-IN-CASE...
00C9' 0C         0565      INC     C
00CA' 0C         0566      INC     C           ; WE ADDRESS PORT n2H...THE TUART COMMAND
0567            ; REGISTER
00CB' ED79      0568      OUT     (C),A        ; AND ENSURE THAT WE ARE NOT IN 'HIGH BAUD'
0569            ; (OCTUPLED BAUD RATE) MODE.
00CD' 0D         0570      DEC     C
00CE' 0D         0571      DEC     C           ; BACK TO PRIMARY PORT ADDRESS
  
```

DDLIST-SPECIFIED BAUD RATE HAS BEEN SET

```

0575 TUOP2:
00CF' 08        0576      EX      AF,AF'      ; GET COUNT OF USER-SPECIFIED PARAMETERS
00D0' B7        0577      OR      A
00D1' C8        0578      RET     Z           ; NONE SPECIFIED...SO WE ARE DONE WITH 'OPEN'
  
```

USER REQUESTED A SPECIAL BAUD RATE/COMMAND FOR THE TUART

```

00D2' 0600      0582      LD      B,0           ; A DEFAULT VALDE: NO OCTUPLED BAUD RATE
  
```



```

00D4' FE02      0583      CP      2      ; DID USER SPECIFY 2 PARAMETERS?
00D6' 2003      0584      JR      NZ,TUOP3      ; NO...JUST ONE.
00D8' FD460B    0585      LD      B,(IY+EPCBS2) ; YES...RETRIEVE IT...
                   0586      TUOP3:
00DB' 0C        0587      INC     C
00DC' 0C        0588      INC     C
00DD' ED41      0589      OUT    (C),B      ; THE 2ND PARAM (OR DEFAULT) IS THE TUART
                   0590      ; COMMAND REGISTER SPECIFIER!
00DF' 0D        0591      DEC     C
00E0' 0D        0592      DEC     C      ; BACK TO THE BAUD-RATE PORT
00E1' FD7E09    0594      LD      A,(IY+EFCBS1) ; GET USER-SPECIFIED BAUD RATE
00E4' ED79      0595      OUT    (C),A      ; AND NOW THE TUART BAUD RATE AND COMMAND ARE
                   0596      ; COMPLETE!
00E6' AF        0598      XOR     A
00E7' C9        0599      RET
  
```

OUTPUT ONE CHARACTER TO TUART

```

00EB' FD4E03    0604      TUPUTC:
                   0605      LD      C,(IY+EPCBDD) ; GET PORT ADDRESS FROM EPCB (SEE TUOPEN)
00ED' 47        0607      LD      B,A      ; SAVE BYTE TO BE OUTPUT HERE FOR NOW
                   0609      TPCLOOP:
00EC' ED78      0610      IN      A,(C)      ; GET STATUS OF THIS TUART
00EE' E680      0611      AND     080H      ; CHECK: IS TRANSMIT BUFFER EMPTY?
00F0' 28FA      0612      JR      Z,TPCLOOP  ; NO...WE MUST WAIT FOR IT TO FINISH LAST
                   0613      ; CHARACTER TRANSMISSION.
00F2' 0C        0615      INC     C      ; TO THE DATA PORT OF THE TUART
00F3' 78        0616      LD      A,B      ; SO THAT BYTE RETURNS STILL IN A
00F4' ED79      0617      OUT    (C),A      ; AND OUTPUT ONE BYTE TO TUART!
00F6' C9        0618      RET
  
```

INPUT ONE CHARACTER FROM TUART

```

00F7' FD4803    0624      TUGETC:
                   0625      LD      C,(IY+EPCBDD) ; THE PORT ADDRESS FROM EPCB
00FA' ED78      0628      INGCLOOP:
00FC' E640      0629      AND     040H      ; CHECK TUART STATUS PORT
00FE' 28FA      0630      JR      Z,TGCLOOP  ; NO...WAIT FOR IT.
0100' 0C        0632      INC     C      ; TO THE DATA PORT OF THIS TUART
0101' ED78      0633      IN      A,(C)      ; GET THE BYTE
0103' C9        0634      RET      ; AND QUIT.
  
```


These interfaces are simple. They support only unidirectional I/O, do not allow any status passing, and (with the exception of the printer driver) do nothing at OPEN or CLOSE time.

 PRINTER or LIST device --- "\$LPT" from SBASIC

```
*****
0646 DRLPT: ; first, the driver address table

0104' 1401' 0648 dw LPTOPEN ; a simple open routine
0106' 3100' 0649 dw .dummy ; no CLOSE routine
0108' 3100' 0650 dw .dummy ; and no SET STATUS routine
010A' 0000 0651 dw 0 ; GET STATUS is illegal here
010C' 1B01' 0652 dw LPTPUTC ; output 1 character to the printer
010E' 0000 0653 dw 0 ; trying to get a character from a printer is illegal
0110' 0000 0654 dw 0 ; --reserved--
0112' 0000 0655 dw 0 ; --reserved--
```

```
*****
0658 LPTOPEN:
0114' 3E0C 0659 LD A,0CH ; THIS IS AN ASCII FORM FEED...
0116' CD1B01' 0660 CALL LPTPUTC ; ...WHICH WE OUTPUT EACH TIME THE PRINTER IS OPENED

0119' AF 0662 XOR A ; ZERO TO A
011A' C9 0663 RET ; ...SAYS THAT OPEN WAS SUCCESSFUL I
```

```
*****
0666 LPTPUTC:
011B' 5F 0667 LD E,A ; MOVE CHARACTER TO WHERE CDOS EXPECTS IT
011C' 0E05 0668 LD C,SYSLPT
011E' CD0500 0669 CALL CDOS ; AND OUTPUT ONE CHARACTER VIA CDOS
0121' C9 0670 RET ; AND THAT IS ALL
```

 PUNCH device -- "\$PUNCH"

```
*****
0676 DRPUNCH: ; the device address table

0122' 3100' 0678 dw .dummy ; no OPEN routine
0124' 3100' 0679 dw .dummy ; no CLOSE routine
0126' 3100' 0680 dw .dummy ; no SET STATUS routine
0128' 0000 0681 dw 0 ; GET STATUS is illegal
012A' 3201' 0682 dw PUNPUTC ; output 1 character to punch
012C' 0000 0683 dw 0 ; get-a-byte from punch is illegal
012E' 0000 0684 dw 0 ; --reserved--
0130' 0000 0685 dw 0 ; --reserved--
```

```
*****
0688 PUNPUTC: ; OUTPUT ONE CHARACTER TO PUNCH
0132' 5F 0689 LD E,A ; WHERE CDOS EXPECTS THE CHARACTER
0133' 0E04 0690 LD C,SYSPUN
0135' CD0500 0691 CALL CDOS ; OUTPUT VIA CDOS SYSTEM CALL
0138' C9 0692 RET
```

```
*****  
READER driver -- *$RDR"  
*****  
0699 DRRDR: ; device driver address table  
0139' 3100' 0701 dw .dummy ; no OPEN  
013B' 3100' 0702 dw .dummy ; no CLOSE  
013D' 3100' 0703 dw .dummy ; no SET STATUS  
013F' 0000 0704 dw 0 ; illegal to GET STATUS  
0141' 0000 0705 dw 0 ; can't PUT a character to the reader  
0143' 4901' 0706 dw RDRGETC ; get 1 character (byte) from reader  
0145' 0000 0707 dw 0 ; --reserved--  
0147' 0000 0708 dw 0 ; --reserved--  
*****  
0711 RDRGETC: ; GET 1 BYTE FROM READER DEVICE  
0149' 0E03 0712 LD C,SYSRDR  
014B' CD0500 0713 CALL CDOS ; LET CDOS DO ALL THE WORK!  
014E' C9 0714 RET  
*****  
END OF SBASIC DRIVERS...ADD YOUR OWN  
014F' (0000) 0719 END  
Errors 0  
Program Length 014F (335)
```