OS-65DV3.0

OS-65D V3.0 Extensions

OS-65D V3.0	Permits the accessing of BASIC programs, assembler source files and data files by name.
#I/O DEVICE SPECIFIER	OS-65D V3.0 permits INPUT or PRINT statements to be directed to a specific I/O device. For example, PRINT #4, "THIS IS A LINE PRINTER" would output the message enclosed in quotes to the line printer. Also, INPUT #3,A$ would request INPUT from the cassette port contained on the 430B board.
OPEN	The OPEN command permits the OPENing of a data file for sequential (one after the other) or random (in any order) access.
CLOSE	The CLOSE command CLOSEs a file to permit the OPENing of another file.
I/O DEVICES #6 & 7	These devices are the CHANNELS under which DISK file I/O operates. Device #6 permits random access file operation while device #6 or #7 may be used in conjunction with sequential files.
DISK!	The DISK! command permits one to "send" commands* to the OS-65D V3.0 DOS (disk operating system). This command may be entered in BASIC's immediate mode or the command may be used in a BASIC program.
GET	The GET command brings a specific record from the disk to the work space and sets the INPUT and PRINT pointers (#6) to the beginning of the record. Random access records are set at 128 bytes but can be user modified via the CHANGE utility program.
PUT	PUT places a specific record back on the disk after it was obtained by the GET command.

Some additional features of OS-65D V3.0 BASIC include:

9 Digits of precision.
The option of declaring numeric variables to be integers.
Indirect command files.
INPUT and PRINT memory I/O.
and much more!

OS-65D 3.0 Utilities

BEXEC*	This BASIC program is automatically LOADed and executed when the system is started. This program may be modified to RUN any user program. This permits the automatic display of a menu on boot up.
CHANGE	A utility to permit the end user to set BASIC's terminal width. This utility also provides the end user with a means of modifying BASIC's and the assembler's work space. This permits the allocation of RAM for special purposes. Some examples of this being reserving memory for DISK buffers, allocating space in memory to hold machine code programs to be used in conjunction with a BASIC program, etc.
CREATE	Permits new files to be added to the OS-65D V3.0 directory. The user must specify the filename (1 to 6 characters), the starting and ending track of the file, and the number of pages per track. A full explanation may be found in the OS-65D V3.0 user's manual.
DELETE	Utility used to DELETE a filename from OS-65D 3.0 directory.
DIR	A BASIC program that when executed PRINTs an OS-65D 3.0 directory to the screen or to the printer. The directory provides the filename and the starting and ending track numbers.
DIRSRT	Utility provides same output as DIR but allows the user to specify the PRINT out to be in order by track number or by name (i.e. alphabetically).
RANLST	This utility provides the user with a means of outputting the contents of random access files to the screen. The end user specifies the filename and starting record number.
RENAME	Permits an entry in the OS-65D V3.0 directory to be RENAMEd.
SECDIR	This utility outputs a directory of the sectors contained on any given set of tracks. Simply put, if a given track contains more than one block (sector) of information, the user may find out exactly how many sectors exist on that track and their length.

SEQLST	A utility to permit the viewing of the contents of a sequential file.
TRACE	This utility sets the TRACE mode of BASIC. When any BASIC program is RUN after the execution of the TRACE, the line number of the line currently being executed will be PRINTed to the screen.
ZERO	Utility used to initialize (erase) the contents of a named file. This is useful for "clearing" garbage from data files. The user must specify the filename.

OS-65U and OS-65D V3.0

Disk BASIC Introduction

The 9 digit BASIC of both OS-65U and OS-65D V3.0 was written by Microsoft, Inc. and is very compatible with the numerous other BASICS written by Microsoft. The 6502 BASIC is considerably faster than 8080 and 6800 BASICS because of the 6502's superior instruction execution time.

This manual starts with features common to both versions of BASIC. It then follows with a short review of the special features of OS-65U and OS-65D V3.0. The user should then consult the corresponding disk operating system manual for further discussions of each of the system features reviewed here.

The following manual provides detailed information about features in the language which are unique in the particular version of BASIC. It further provides a handy reference for the standard syntax of Microsoft BASIC. It is not intended, however, to be a tutorial or teaching aid. The user is directed to several of any of the excellent texts on BASIC to learn the language such as BASIC AND THE PERSONAL COMPUTER by Dwyer and Crithfield which is available through most Ohio Scientific dealers.

SPECIAL CHARACTERS

Character	Use
@	Erases line being typed
(shift 0)	Erases last character being typed
Carriage return	Must be used after each line typed
Control C	Interrupts program execution or listing returns to command mode.
: (colon)	Allows multiple statements per line
Contol 0	Typing a control 0 once surpresses output until another control 0 is typed.
?	? can be used instead of print.

OSI 8K BASIC is a "standard" BASIC with additional string handling capability and I/O commands, as well as the following features.

OSI BASIC allows multiple statements per line via ":". Next without a variable can be used when FOR-NEXT statements are not nested. END statements are not necessary. Question marks can be used instead of "PRINT". "LET" is optional. No spaces are required in BASIC. These features allow highly efficient memory usage when necessary.

Variables can be two characters long. Longer variables can be used but only the first two characters will be utilized. The first character must be alphabetic, the second can be alphabetic or numeric. Long variables can not contain words used by BASIC such as NEW, SIN, and so on. Since spaces are not necessary BASIC would interpert a variable such as "ANEW" as a variable A and the command "NEW" and would erase the program.

EXAMPLES:

LEGAL	ILLEGAL
A	IA
A1	#B
AZ	TOO
BEQ	RGOTO
APPLE	NEW 1
TUESDAY	FREQUENCY

Note: that variables AZ1 and AZ2 would be treated the same since BASIC looks only at the first two characters.

COMMANDS

NAME	EXAMPLE	COMMENTS
LIST	LIST	Lists program
	LIST 100	Lists program from line 100. Control C stops program listing at end of current line.
NULL	NULL 3	Inserts 3 nulls at the start of each line to eliminate change return bounce problems. Null should be 0 when entering paper tapes from Teletype® readers. When punching tapes Null>=3. Higher settings are required on faster mechanical terminals.
RUN	RUN	Starts program execution at first line. All variables are reset. Use an immediate GOTO to start execution at a desired line.
	RUN 200	GOTO 200 with variables reset.
NEW	NEW	Deletes current program.
CONT	CONT	Continues program after Control C or STOP if the program has not been modified. For instance a STOP followed by manually printing out variables and then a CONT is a useful procedure in program debugging.
LOAD	LOAD	Used in cassette and Disk BASIC only.

OPERATORS

SYMBOL	EXAMPLE	COMMENTS
=	A=10	LET is optional
	LET B=10	
-	C=-B	Negation
↑ (Shift/n)	X↑4	X to the 4th power
		C↑D with C negative and D not an integer gives an FC error.
*	C=A*B	Multiplication
/	D=L/M	Division
+	Z=L+M	Addition

	J=255.1-X	Subtraction
<>	10 IF A<>B THEN 5	Not equal
>	B>A	B greater than A
<	B<A	B less than A
<=, =<	B<=A	B less than or equal to A
=>,>=	B=>A	B greater than or equal to A
AND	IF B>A AND A>C THEN 7	If both expressions are true then--.
OR	IF B>A OR A>C THEN 7	If either expression is true then--.
NOT	IF NOT B>A THEN 7	If B<=A then--.

AND, OR, and NOT can also be used in Bit manipulation mode for performing Boolean operations of 16 bit 2s complement numbers (-32768 to +32767)

EXAMPLES

EXPRESSION	RESULT
63 AND 16	16
-1 AND 8	8
4 OR 2	6
10 OR 10	10
NOT 0	-1
NOT 1	-2

OPERATOR EVALUATION RULES: Math statements evaluated from left to right with * and / evaluated before + and -. Parentheses explicitly determine order of evaluation.

Precedence for evaluation

1) By parenthese
2) ↑
3) Negation
4) * /
5) + -
6) =,<>, <, > , <=, >=
7) NOT
8) AND
9) OR

STATEMENTS

In the following examples

V or W is a numeric variable, X is a numeric expression, X$ is a string expression, I or J is a truncated integer.

NAME	EXAMPLE	COMMENTS
DATA	10 DATA 1,3,7	Data for READ statements must be in order to be read. Strings may be read in DATA statements.
DEF	10 DEF FNA (V)=V*B	User defined function of one argument.
DIM	110 DIM A (12)	Allocates space for Matrices and sets all matrix variables to zero. Non dimensioned variables default to 10.
END	999 END	Terminates program (optional)
FOR, NEXT	10 FOR x=.1 to 10 STEP.1 20_____ 30 NEXT X	STEP is needed only if X is not incremented by 1. NEXT X is needed only if FOR NEXT loops are nested if not NEXT alone can be used variables and functions can be used in FOR statements.
GOTO	50 GOTO 100	Jumps to line 100
GOSUB, RETURN	100 GOSUB 500 500 . . . . 600 RETURN	Goes to subroutine, RETURN goes back to next line number after the GOSUB
IF...THEN		
	10 If X=5 THEN 5	If the statement is true
	10 If x=5 THEN PRINT X	
	10 If X=5 THEN PRINT X:Y=Z	Then the following will be executed including multiple statements of that line.
IF...GOTO		
	10 IF X=5 GOTO5	Same as if THEN with line number
ON... GOTO		
	100 ON I GOTO 10, 20, 30	Computed GOTO If I=1 then 10 If I=2 then 20 If I=3 then 30

PRINT	10 PRINT X 20 PRINT "Test"	Prints value of expression Standard BASIC syntax with , ; " formats
READ	490 READ V, W	Reads data consecutively from DATA statements in program
REM	10 REM	This is a comment for non-executed comments.
RESTORE	500 RESTORE	Restores Intial values of all DATA statements
STOP	100 STOP	Stops program execution reports a BREAK. Program can be restarted via CONT.

FUNCTIONS

Function	Comment
ABS (X)	For X=>0 ABS(X)=X For X<0 ABS(X)=-X
INT (X)	INT (X) = largest integer less than X
RND (X)	Generates a random number between 0 and 1 RND (0) generates the same number always RND (X) with the same X always generates the same sequence of random numbers NOTE (B-A)* RND (1)+A generates a random number between B and A
SGN (X)	IF X>0 SGN(X)=1 IF X=<0 SGN(X)=0
SIN (X) COS (X) TAN (X) ATN (X)	Sine of X where X is in radians Same for COS, TAN, and ATN (ARC TAN)
SQR (X)	Square root
TAB (I)	Spaces the print head I.
USR (I)	See I/O section
EXP (X)	E↑X where E is 2.71828
FRE (X)	Gives number of Bytes left in the workspace.
LOG (X)	Natural LOG to obtain base 10 logs use LOG(X)/LOG (10)

POS (I)	Gives current location of terminal print head.
SPC (I)	Prints I spaces, can only be used in print statements.

STRINGS

Strings can be from 0 to 255 characters long. All string variables end in $ ex. A$, B9$, HELLO$.

Strings can be dimensioned equated, printed, read from Data statements, etc.

STRING FUNCTIONS

ASC (X$)	Returns ASCII value of first character in string.
CHR$ (I)	returns a I character string equivalent the ASCII value above.
LEFT$ (X$,I) RIGHT$(X$,I)	Gives left most I characters of string X$ Gives right most I character of string X$
MID $ (X$,I,J)	Gives string subset of string X$ starting at I^{th} character for J characters. If J is omitted, goes to end of string.
LEN (X$)	Gives length of string in bytes.
STR$ (X)	Gives a string which is the character representation of the numeric expression of X. Example X=3.1 X$=STR$(X) X$="3.1"
VAL (X$)	Returns string variable converted to number. Opposite of STR$(X)

I/O

The following features of OSI 8K BASIC are usefull primarily for I/O control. The user should be extremely careful with these statements and functions since they manipulate the memory of the computer directly. An improper operation with any of these commands can cause a system crash, wiping out BASIC and the users program, thus requiring a complete reload of the computer.

STATEMENT/FUNCTION	COMMENT
PEEK (I)	Returns the decimal value of the specified memory or I/O location. (Decimal) Example: X=PEEK (64256) Loads variable X with the 430 Board's A/D converter output. ($FB00_{hex}$)
POKE I,J	Loads memory location I (decimal) with J (Decimal) I must be between 0 and 65536 and J must be between 0 and 255 Example: 10 Poke 64256, 255 loads FB00 with FF (Hex) thus loads the 430 Board's D/A port 0 such that its output is +2 volts.
WAIT I,J,K	Reads status of memory location I (Decimal) exclusive OR's with K then AND's the result with J until a non zero result is obtained. If K is omitted, it is zero. Wait is used for fast service of input status flags. Example: Wait X,1 will wait until Bit zero of memory location X goes low then BASIC will continue.

The high speed servicing of flags via the WAIT command allows the programmer to service medium speed devices such as line printers or industrial equipment directly in BASIC.

USR: The USR function allows linkage to machine language routines such as ultra-fast device handlers, etc. The USR function calls only one machine language routine and can pass one integer value to the machine language routine so that 65,000 actual user routines are possible.

The beginning of the user subroutine must be poked into $23E_{hex}$ (low) and $23F_{hex}$ (high). The USR routine can use up to 8 levels of subroutines (16 stack locations) without page swapping.

The USR function can obtain the argument of the function by calling the routine pointed to by 6 (low) and 7 (high). This routine will

place the value of the argument in $AE_{(hex)}$ (high part) and $AF_{(hex)}$ (low part). To pass a value back to BASIC, the high part is placed in A and the low part is placed in Y and the subroutine pointed to by 8 and 9 should be called. If this function is not called USR (X) will equal X. An RTS returns from USR to BASIC. All registers can be modified by the user routine without affecting BASIC, however, no page zero locations can be modified! The POKE instruction can also be used to change the USR function call.

INTERRUPTS

For Interrupting routines of any significant length, page zero and page one should be swapped out to higher memory, or memory partitioning (A_{16} and A_{17} on late model OSI memory boards) should be used.

CONVERTING OTHER BASICS TO RUN ON OSI 6502 8K BASIC

MATRIX subscripts: Some BASICS use [ ]
OSI BASIC used ( ).

Strings:

OTHER	OSI
DIM A$(I,J)	DIM A$ (J)
A$ (I)	MID$ (A$,I,1)
AS (I,J)	MID$ (A$,I,J-I+1)

Multiple assignments: B=C=0 must be rewritten as B=0:C=0. Some BASICs use / to delimit multiple statements per line.Use ":". Some BASICs have MAT functions which will have to be rewritten with FOR-NEXT loops.

OSI 6502 8K BASIC ERROR MESSAGES

ERROR CODE	MEANING
BS	Bad Subscript: Matrix outside DIM statement range, etc.
DD	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 10.
FC	Function Call error: Parameter passed to function out of range.
ID	Illegal Direct: Input or DEFIN statements can not be used in direct mode.
NF	NEXT without FOR:
OD	Out of Data: More reads than DATA
OM	Out of Memory: Program too big or too many GOSUBs, FOR NEXT loops or variables.
OV	Overflow: Result of calculation too large for BASIC
SN	Syntax error: Typo, etc.
RG	RETURN without GOSUB.
US	Undefined Statement: Attempt to jump to non-existant line number.
/0	Division by Zero.
CN	Continue errors: Attempt to inappropriately continue from BREAK or STOP.
LS	Long String: String longer than 255 characters.
OS	Out of String Space: Same as OM
ST	String Temporaries: String expresssion too complex.
TM	Type Mismatch: String variable mismatched to numeric variable.
UF	Undefined Function.

OS-65U

OPEN

The OPEN command is used to OPEN a data file for access. The command has two possible formats. The formats are:

OPEN"FILENAME","PASSWORD", CHANNEL NUMBER and

OPEN"FILENAME",CHANNEL NUMBER

The channel number must be an integer between one and eight. After OPENing a file, the file is referred to by the channel it was OPENed under rather than by its name. If a file is OPENed with the correct password, the system may read or write to the file regardless of that file's access rights. If a file is OPENed without the correct password, the system may only access the file in the manner defined by its access rights (e.g. read only, write only, none, etc.).

CLOSE

The CLOSE command permits a file to be CLOSEd. CLOSEing a file frees the channel the file was OPENed under. Simply typing CLOSE, CLOSEs all channels currently OPENed. Typing CLOSE X, where X is a specific channel number, CLOSEs only that channel.

INDEX

The INDEX is a reserved system variable. This command takes two forms. The first being INDEX (channel number). An example of program usage would be X=INDEX(1). After the execution of this statement, X would equal the value of the INDEX for channel number one. This INDEX value is simply a relative pointer into the file OPENed under a particular channel. When a file is OPENed, the INDEX of the channel the file was OPENed under is set to zero.

The second form of the INDEX function is INDEX channel number = expression. This permits "bumping" the INDEX to point anywhere within the data file. Note the power of the INDEX function in permitting the user to define the file formats to be whatever is desired.

PRINT%

PRINT% channel number, variable expression where the variable expression may be string or numeric. This statement directs the variable to be PRINTed into the data file OPENed under the channel "channel number". The variable(s) will be PRINTed into the file starting at the current position of the INDEX of that channel.

INPUT% The syntax for this command is INPUT% channel number, variable(s). This command INPUTS data from the file OPENed under "channel number" and assigns the data to the variable(s) that appear in the INPUT% statement. The data is INPUT from the file starting at the current position of the INDEX. The INPUT% statement terminates upon the receipt of a carriage return.

FIND The FIND command syntax takes the form of FIND string expression, channel number. The FIND command executes a high speed search for the string expression in the file OPENed under "channel number". The search starts from the current position of the INDEX for that channel. If the string is found, the INDEX for that channel points at the string in the file. If the string is not found, the INDEX for that channel is set to 1,000,000,000. Note that the FIND command may be used to FIND subsets, e.g., FIND "ABCD",1 would FIND the subset "ABCD" in the string "ABCDEFG". The FIND command also permits "don't care" characters within the string it is searching for. That is, FIND "AB&HI" would FIND "ABCHI" or ABZHI", etc.

FLAG NN The FLAG command permits the end user to tailor his system toward a specific application. The syntax for the FLAG command is FLAG NN where NN corresponds to the FLAG number of the option desired. These options are listed below:

FLAG 1	Disables "close-files-on-error" feature
FLAG 2	Enables "close-files-on-error" feature
FLAG 3	Select video/keyboard as console
FLAG 4	Selects serial as console
FLAG 5	Enables user programmable "end-of-file-hit" error action
FLAG 6	Enables program abort on "end-of-file-hit" error
FLAG 7	Enables BASIC trace mode
FLAG 8	Disables BASIC trace mode
FLAG 9	Enables "GOTO 50000" on disk error
FLAG 10	Enables program abort on disk error

FLAG 11 Enables space suppression in numeric output to disk

FLAG 12 Disables the above

FLAG 19 Disables disk checksum error checking

FLAG 20 Enables disk checksum error checking

FLAG 21 Disables program termination on INPUT of carriage return only, in response to an INPUT statement

FLAG 22 Enables the above

#I/O DEVICE SPECIFICATION

OS-65U provides the user with the option of directing INPUT and PRINT statements to specific I/O devices. Some examples of this are:

PRINT#5, "THIS IS A TEST" - This would direct output to the line printer

INPUT#5, "EMPLOYEE'S NAME";#1,QAS - This would direct the message "EMPLOYEE'S NAME" to the line printer and would direct the INPUT to be from the serial console

MONEY-MODE

OS-65U permits MONEY-MODE output of numeric variables. Any numeric variable output in the MONEY-MODE is automatically rounded to two digits. For example, 3.141 would be output as 3.14. The MONEY-MODE also inherently provides left or right justification of the output. This simply means that the "screen" is broken up into fields. These fields are 14 spaces wide. When a number is output in the MONEY-MODE, that number is PRINTed at the next field boundary, e.g.,

X=3.142 : PRINT $R,X would PRINT as 3.14
^ right side of field

X=3.142 : PRINT $L,X would PRINT as 3.14
^ left side of field

BEXEC*	BASIC program which is automatically loaded and executed upon boot up - allows automatic execution of user's program
DIR	Program PRINTs OS-65U file directory on screen or printer. Directory contains filename, file type, file access rights, file starting address and file length.
CREATE	CREATE permits the creation of files under OS-65U. The user must specify: Filename (1 to 6 characters). File length (length in bytes). File type (BASIC, DATA, OTHER, i.e., machine code). File access rights (read/write, read only, write only, none). File password (if file access rights other than R/W).
RENAME	Permits an entry in the directory to be RENAMEd and a new password assigned.
DELETE	Allows the deletion of an entry (file) from the directory by name.
PACKER	Permits recovery of disk area currently occupied by deleted files.
COPIER	Permits copying of 65U system or 65U files.
COPYFI	Permits copying of one file (by name) to another file.
CHANGE	Change allows the end user to modify any portion of OS-65U stored on disk. Permits the user to modify the system or insert patches.
FDUMP	This utility allows one to examine 65U data files. The program displays a "core image" of the file. Thus, permitting the user to "see" carriage returns, spaces, null and all control codes in the file in addition to the data in the file.
FPRINT	FPRINT prints OS-65U data files to the screen or printer. The output may be in a sequential or "record" format.

LOAD32	Permits the user to transfer machine code stored on a WP-1A word processor diskette into OS-65U. LOAD32 is designed to run in a 32K system.
LOAD48	Same as LOAD32 except LOAD48 is designed to run in a 48K machine.
BUS1	Demo program illustrating index sequential file operation under OS-65U.
BUS2	This program illustrates sequential file handling. It also provides one example of how to append to sequential files.
BUS3	A demo program illustrating the use of mixed file types, i.e., index sequential and sequential files.
ACC1	Data file for BUS1.
ACC2	Data file for BUS2.
PHONE	Program example of a phone directory under OS-65U.
PHONDIR	Data file for PHONE.
LABLE	Programming example of a mailing label program.
TLABLE	Data file for LABLE.