### ASCII Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Char</th>
<th>Code</th>
<th>Char</th>
<th>Code</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>NUL</td>
<td>2B</td>
<td>+</td>
<td>96</td>
<td>V</td>
</tr>
<tr>
<td>01</td>
<td>SOH</td>
<td>2C</td>
<td>:</td>
<td>57</td>
<td>W</td>
</tr>
<tr>
<td>02</td>
<td>STX</td>
<td>2D</td>
<td></td>
<td>58</td>
<td>X</td>
</tr>
<tr>
<td>03</td>
<td>ETX</td>
<td>2E</td>
<td></td>
<td>59</td>
<td>Y</td>
</tr>
<tr>
<td>04</td>
<td>EOT</td>
<td>2F</td>
<td>\</td>
<td>5A</td>
<td>Z</td>
</tr>
<tr>
<td>05</td>
<td>ENQ</td>
<td>30</td>
<td>0</td>
<td>5B</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>ACK</td>
<td>31</td>
<td>1</td>
<td>5C</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>BEL</td>
<td>32</td>
<td>2</td>
<td>5D</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>BS</td>
<td>33</td>
<td>3</td>
<td>5E</td>
<td>A</td>
</tr>
<tr>
<td>09</td>
<td>HT</td>
<td>34</td>
<td>4</td>
<td>5F</td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>LF</td>
<td>35</td>
<td>5</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>0B</td>
<td>VT</td>
<td>36</td>
<td>6</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>FF</td>
<td>37</td>
<td>7</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>CR</td>
<td>38</td>
<td>8</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>SO</td>
<td>39</td>
<td>9</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>SI</td>
<td>3A</td>
<td></td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DLE</td>
<td>3B</td>
<td></td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>DC1</td>
<td>3C</td>
<td>\</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DC2</td>
<td>3D</td>
<td>=</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>DC3</td>
<td>3E</td>
<td>&gt;</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>DC4</td>
<td>3F</td>
<td>?</td>
<td>6A</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>NAK</td>
<td>40</td>
<td>@</td>
<td>6B</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SYN</td>
<td>41</td>
<td>A</td>
<td>6C</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>ETB</td>
<td>42</td>
<td>B</td>
<td>6D</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CAN</td>
<td>43</td>
<td>C</td>
<td>6E</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>EM</td>
<td>44</td>
<td>D</td>
<td>6F</td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>SUB</td>
<td>45</td>
<td>E</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>1B</td>
<td>ESC</td>
<td>46</td>
<td>F</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>FS</td>
<td>47</td>
<td>G</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>1D</td>
<td>GS</td>
<td>48</td>
<td>H</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>1E</td>
<td>RS</td>
<td>49</td>
<td>I</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>1F</td>
<td>US</td>
<td>4A</td>
<td>J</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>SP</td>
<td>4B</td>
<td>K</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>4C</td>
<td>L</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>4D</td>
<td>M</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>23</td>
<td>#</td>
<td>4E</td>
<td>N</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>$</td>
<td>4F</td>
<td>O</td>
<td>7A</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>%</td>
<td>50</td>
<td>P</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>&amp;</td>
<td>51</td>
<td>Q</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>(</td>
<td>52</td>
<td>R</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>)</td>
<td>53</td>
<td>S</td>
<td>7E</td>
<td>+</td>
</tr>
<tr>
<td>29</td>
<td>*</td>
<td>54</td>
<td>T</td>
<td>7F</td>
<td>DEL</td>
</tr>
</tbody>
</table>

### OS-65D Disk Operating System

- **BASIC**
- **DOS**
- **ASSEMBLER**

#### DOS and BASIC Quick Reference

#### To Start Your Computer

1. Turn on the computer, disk drives and terminals switches are generally located on the back of the device cabinet.
2. Place an OS-65D disk in drive A (the drive whose red light is on or the top drive in dual drive cabinets). Close the disk drive door.
3. Depress the BREAK key on C1P and C4P systems (and hold for a few seconds). Depress the white reset button on C8P and serial systems.
4. When the "H/D/M?" ("D/C/W/M?" on C1P systems) message appears, respond by typing "D". In a few seconds a menu should appear on the screen.
5. To enter the BASIC immediate mode, respond UNLOCK to this menu in OS-65D V3.2; select option 9 in OS-65D V3.3.
65D BASIC

The entries are organized alphabetically according to Keywords used. Each entry consists of the general syntax, examples where appropriate, and a brief description. The following notation is used:
{n} see page of the OS-65D Tutorial and Reference Manual
{e} see page of the OSI BASIC Reference Manual
(*) cannot be used in the immediate (direct) mode; must be used with a program statement number.
(**) can only be used in the immediate (direct) mode; must not be used within a program.
(2) not available under OS-65D V3.3.
(3) available only under OS-65D V3.3.
ae a numeric constant or arithmetic expression (see {3})
re a logical constant or relational expression (see {4})
se a string constant or expression (see {4})
dos a 65D Disk Operating System (DOS) command.
e a constant or expression.
V a variable
c a constant
nv a numeric variable
iv an integer variable
sv a string variable
niv a numeric variable or integer variable
rre a relational expression or arithmetic expression
FILE a disk file name
loc a memory location address
sn a program statement number
dev an OS-65D device number. [54]

ABS
ABS(ae)
A function. Returns the absolute value of its argument. {19}

AND
re AND re
IF X < 15 AND X > = 0 THEN 100
A bitwise Boolean AND operator. re AND re will be TRUE only when both of the operands have the value TRUE. {4}

ASC
ASC(se)
ASC("BIG")
A function. Returns the ASCII value in decimal of the first character in the argument {20}

ATN
ATN(ae) (-1 < ae < 1)
ATN(0.431)
A function. Returns the arctangent of the argument {20} {2} {188}

CHR$
CHR$(ae)
CHR$(66)
A function. Returns the character whose decimal ASCII value is the greatest integer less than or equal to the argument. {21}

CLEAR
CLEAR
Clears the program variable table and restores the data pointer (*) {17}

CLOSE
DISK CLOSE, dev (dev = 6 or dev = 7)
Closes a disk file that has been previously opened. {28} {15}

CONT
CONT
Starts a program whose execution has been interrupted by a STOP or END statement or a CTRL-C. {15} (**)

COS
COS(ae)
A function. Returns the cosine of the argument. {20}

DATA
DATA c, c, c,
DATA 1.7, "BIG", 173, -812
Establishes a list of constants to be input by the program via the READ statement {6}

DEF FN
DEF FNv(nv) = ae
DEF FNv(X) = X**7 + 3
Defines a single variable function for future use within the program segment {23} (*)

DISK!
DISK! "dos"
DISK! "IO 5,6"
DISK! "LOAD FILE"
Permits 65D DOS commands to be used within a BASIC program. {202}

DISK CLOSE
See CLOSE

DISK FIND
See FIND

DISK GET
See GET

DISK OPEN
See OPEN

DISP PUT
See PUT

EDIT
EDIT sn
EDIT 100
Returns line sn for editing. The short form is E sn. (*{(71) 3}

END
Ends program execution {13}

EXIT
EXIT
Transfers control to the DOS mode {28} {53}

EXP
EXP(ae)
ae = 8.629619
EXP(41,692)
A function. Returns e = 2.71828...raised to the power equal to the value of the argument. {19}

FIND
DISK FIND, se
DISK FIND, "BIG"
Beginning at current file pointer location, the data file is searched for the specified string. The pointer is set to the end of the field in which it is found. An unsuccessful search results in a #D error. {36} {3}

FN
See DEF FN

FOR
FOR nv = ae TO ae
FOR nv = ae TO aeSTEP ae
FOR X = 15 TO 45 STEP 5
Opens program loop. End of the loop is indicated by the statement NEXT or NEXT nv. STEP is used to define an increment other than 1 for nv for each iteration of the loop. In the example, the loop is executed 7 times {12}

FRE
FRE(X) X is a dummy variable
A function. Returns the number of bytes of memory in the workspace that are unused. Save the program before using FRE. {17}

GET
DISK GET, nv
DISK GET, 15
Brings the record numbered nv from the disk to buffer #6 and sets the I/O pointers to the beginning of the record {28} {17}
GOSUB GOSUB sn
GOSUB 1000
Program control is transferred to statement number sn. When the statement RETURN is encountered, control goes back to the statement following sn. [23]

GOTO GOTO sn
GOTO 1000
Program control is transferred to statement number sn. [11]

IF IF rae GOTO sn
IF rae THEN sn
If the value of rae is TRUE (arithmetic expressions are considered to be TRUE if they have a value other than 0) program control is transferred to statement sn.
If rae THEN S (S is a program statement)
If the value of rae is TRUE, statement S is executed [11]

INPUT INPUT V, V, ...
INPUT X, Y, A$.
Prompts for keyboard input to the specified variables [6] [*]

INPUT# INPUT# V, V, ...
INPUT #6, A, B, Q$.
Input is from device number dev to the specified variables. [9] [13] [*]

INT INT (ae)
INT (16,8)
A function. Returns the greatest integer less than or equal to the argument [19]

LEFT$ LEFT$(ae, ae) ae > 0
LEFT$("ABCDEF", 3)
A function. Truncates ae to an integer and returns that leftmost number of characters from string se. In the example, "ABC" is returned. [21]

LEN LEN(se)
LEN(A$)
A function. Returns the length of the string se [21]

LET LET V = e
LET A$ = "BIG"
Assignment statement. Keyword LET is optional. [6]

LIST LIST
LIST sn-sn
LIST 100-200
LIST 1000
LIST 200
Lists the program in the workspace between the two specified statement numbers. If the first (second) statement number is omitted, the default is the beginning (end) of the program. [15]

LIST# LIST#dev
LIST#4
Same as LIST, but the listing is sent to device number dev. [9] [15] [54]

LOG LOG(ae) ae > 0
LOG10.4
A function. Returns the natural logarithm (log to the base e) of the argument. [19]

MIDS MIDS(se, ae, ae) first ae > 0, second ae > 0
MIDS("ABCDEF", 2, 3)
A function. In the example, A string of length 3 starting at position 2 is returned; i.e., "BCD". If the second ae is omitted, the string returned goes to the end of se. [21]

NEW NEW
Clears the workspace to prepare for creation of a new program [15]

NEXT see FOR
NOT NOT re
NOT (A > 5)
A bitwise Boolean NOT operator. Reverses the truth value of the operand re. [3]

NULL NULL iv 0 <= iv <= 8
Inserts iv zeros at the beginning of each line as it is stored on tape. [27] [2]

ON ON ae GOTO sn, sn...
ON ae GOSUB sn, sn...
ON X + 7 GOTO 100, 200
Depending upon the value of ae (truncated to an integer) program control passes to the ae-th statement in the list of statement numbers [12, 24]

OPEN DISK OPEN, dev, "FILE" (dev = 6 or 7)
Opens the disk file FILE for sequential (dev = 6 or 7) or random access (dev = 6 only) [28] [15]

OR re OR re
IF A > 5 OR A < 2 THEN 100
A bitwise Boolean OR operator. re OR re is FALSE only when both of the operands are FALSE. [3]

PEEK PEEK(loc)
A function. Returns the value stored in memory location loc [25]

POKE POKE loc, ae
ae is an integer.
POKE 11686, 17
The value ae is stored in memory location loc [25]

POS POS(X)
X is a dummy variable.
A function. In or following a PRINT statement, returns the current position (between 0 and 132) of the cursor [9]

PRINT PRINT e, e...
PRINT A, B$, C$
Outputs the values stored in the list of expressions. The keyword PRINT can be replaced by a question mark. [7]

PRINT# PRINT#dev, e, e...
Same as PRINT, but output is directed to device number dev instead of the screen. [7] [13] [54]

PRINT! PRINT!(HOC), e, e,... (HOC = Hazelrite Operation code-see [223])
PRINT!(28) X$, A, B, C
Depending on the value of HOC, certain screen characteristics and cursor positions are selected before beginning output of expression values; emulates certain Hazelrite terminal capabilities. [223] (3)

PRINT CHR$ see CHRS
PRINT(X, Y), e, e,...
PRINT(10, 20) A, B$.
Moves the screen cursor to screen
position (X, Y) (0, 0 = upper left corner)
before beginning output of expression
values. Identical to:
PRINT(17, X, Y), e, e,... [79] (3)

PRINT USING PRINT USING se, ae,...
PRINT USING "###.###" 6.87304
Used to format numeric output; ae must
be a string expression made up of a
decimal point and/or #s. In the example
the output format specified results in
printing 6.87 (with three leading blanks)
[73] (3)

PUT DISK PUT
Follows a previous DISK GET; places the
current record back to the disk. [28] [17]

READ READ V, V, V,...
READ A, B$, C
Inputs constants that are specified by
DATA statements in the same program
into the specified variables ($6) (7)

REM REM any remark
REM THIS IS A TEST PROGRAM
Used for program documentation. Everything
appearing after REM is ignored on
execution of that line [15]

RESTORE RESTORE
Resets the pointer in a program's DATA
list to the first item. [7]

RETURN See GOSUB

RIGHTS RIGHTS(se, ae) ae > 0
RIGHTS("ABCDEF", 2)
A function. Truncates se to an integer
and returns that number of rightmost
characters. In the example, "EF" is
returned. [21]

RND RND(ae)
RND(16)
A function. Returns a number between 0
and 1. Can be used repeatedly to
generate a sequence of pseudo-random
values. If ae = 0, the argument is a
dummy argument. If ae = 0, RND
returns the previous value again. If ae < 0,
ae functions as a "seed" and RND starts
a new sequence. The sequence repeats
after a certain period determined by the
seed. [19]

RUN RUN
Starts execution of the program in the
workspace at the first statement.
RUN sn
Starts execution of the program in the
workspace at statement number sn.
RUN "FILE"
Leads the program from disk file
FILE and starts execution.
RUN "TT" (TT = a disk track number)
Loads the program from the disk file
beginning at track TT and starts
execution. [15]

SGN SGN(ae)
A function. Returns +1 if ae > 0, 0 if
ae = 0, -1 if ae < 0 [19]

SIN SIN(ae)
A function. Returns the value of the sine
of the argument ae. [20]

SPC SPC(ae)
PRINT "A"; SPC(5); "B"
A function. Used to print ae spaces in a
PRINT sequence [9]

SQR SQR(ae) ae ≤ 0
A function. Returns the square root of
the argument ae. [20]

STEP See FOR
STOP STOP
Halts execution of a program and prints
a BREAK message indicating the state-
ment number of the STOP statement
[13]

STR$ STR$(ae)
STR$(6,71)
A function. Returns the value of the
argument ae as a string. [21]

TAB TAB(ae)
ae is an integer
TAB(10)
A function. Used in a PRINT statement
to move the print position for the next
character to position ae + 1 on the print
line. [8]

TAN TAN(ae)
A function. Returns the tangent of the
argument. [20]

THEN See IF
TO See FOR

TRAP TRAP sn
If an error is encountered in a program
after this statement, then control
transfers to statement sn.
TRAP 0 disables error trapping. [71] (3)

USR USR(ae)
y = USR(x)
Transfers control to a machine language
routine at a location determined
previously by appropriate POKEs. ae
may be an input parameter (and USR(ae)
an output parameter) or ae may be a
dummy parameter. [34]

VAL VAL(se)
VAL("6.31")
A function. It is the opposite of STR$;
returns the numeric value of the string
expression se if se represents a number.
Otherwise, 0 is returned.

WAIT WAIT loc, J
0 ≤ J ≤ 255
Halts program execution. Reads the
contents of location loc and AND's the
result (bitwise) until a nonzero result is
obtained, then resumes program
execution.
WAIT loc, J, K 0 ≤ J, K ≤ 255
Halts program execution, reads the
contents of location loc, exclusive OR's
that value (bitwise) with K, and then
AND's the result with J until a nonzero
result is obtained; then resumes
execution [25] [2]

SPECIAL V3.3 COMMANDS

Screen Display Commands:
(ESC) 1 Clears screen; homes cursor to upper left;
produces "wide character" display
(32x32 on C4P and C8P machines; 24x24 on
C1P)
PRINT Statement Commands
(These commands must be used in PRINT statements)

Display Size
!(20) Selects "wide character" display (32 x 32 on CP and C8P, 12 x 14 on C1P), clears screen and home cursor to upper left corner.
!(21) Selects "narrow character" display (32 x 64 on CP and C8P, 12 x 48 on C1P), clears screen, and home cursor to upper left corner.
!(22, w, h) Selects print window w characters wide and h characters high. Upper left window corner is at current cursor position; screen is not cleared.

Cursor Control

Single Step
CHR$(8) Back one space.
CHR$(16) Forward one space.
!(12) Up one space.
!(11) Down one space.
CHR$(10) Down one space.

Multistep
CHR$(13) Back to front of line.
CHR$(14) Forward to next eight space tab set (seven space for left-most field).

Anywhere
!(17, x, y) Relocate to x, y (@ 0 is upper left corner).
&(x, y) Relocate to x, y (80 is upper left corner).

Home
!(18) Relocate to 0, 0 (upper left corner).

Insert
!(26) Inserts line at cursor position; lower lines scroll down.

Line
!(15) Clears from cursor to end of line.
!(19) Clears entire line (lower lines move up).

Screen
!(24) Clears from cursor to end (lower right) of window.
!(28) Clears entire screen and home cursor in window.

Color Select
!(1) Selects color 0 as cell background.
!(25) Selects normal black/white display mode (i.e., black background, white character).
!(31, n) Selects color n as cell background.

Color Change
!(2, n, m) Changes all displayed cells of background color m to background color n.
!(29, n) Clears all displayed cells of background color n (i.e., cell background is changed to black and character is replaced with a blank).

Cursor Sensing
!(5) Sends information for current cursor position x, y, to string variable in following INPUT statement. Information is in the form of two characters for which (x + 85) is the ASCII code. Line feed follows the INPUT statement used with !(5).
!(33) Sends character at cursor position to string variable in following INPUT statement. Line feed follows the INPUT statement used with !(33).

Printer Control
!(67,FL) Initialize Epson Printer Drivers; set form length.
!(80) Send Video Screen to Epson Printer

** Note to Users of Serial Systems **
OS-65D V3.3 is only partially compatible with serial systems. If you are using a Hazeltine 1429 terminal, be sure switch 6 is set to the ESC position. Certain features that refer to color, screen size, or windowing are not operable on serial systems. Specifically,
1) The commands that use the ESC key are not operable.
2) The destructive backspace key is <DEL> instead of <SHIFT/O> or <RUB OUT> and the line delete is <-> instead of <SHIFT/P>.
3) The PRINT command !(26) inserts a line but not at the cursor position. The line always starts at the left margin.
4) The following PRINT commands should not be used:
    !(1) !(21) !(29, n) !(20)
    !(2, n, m) !(22, w, 1) !(31, n) !(67, FL)
    !(5) !(25) !(33) !(80)
    !(29) !(29)

V3.3 EDITOR COMMANDS

(CTRL)H Moves cursor one space to the left (non-destructively)
(CTRL)P Moves cursor one space to the right (non-destructively)
(CTRL)F Moves cursor to the front of the line
(CTRL)R Moves cursor to the rear of the line
(CTRL)I Moves the cursor (non-destructively) forward to the next tab position (i.e., positions 1, 8, 15, 22, 29, 36, 43, 50, 57, 64, 71)
(CTRL)T Retypes the line currently being edited (in its present edited form)
(SHIFT)P Clears screen of line currently being edited leaving the line in workspace as it was before calling it to be edited
(RUBOUT) Deletes the character flashing with the cursor. Line closes up from the right.
EDITnn or inn Calls line number nn for editing
EDIT or ! Calls next line in program for editing
EDITI or !! Recalls last edited line for re-editing
INDIRECT FILES

To merge two BASIC programs using indirect files:
1) determine the starting page number N of the indirect file.
2) load one program into the workspace.
3) move this program to the indirect file.
4) load the second program into the workspace.
5) move the first program back from the indirect file to the workspace.

If each of the programs has a line with the same number the line in the first program will be the one that appears in the merged program.

STARTING PAGE NUMBER OF INDIRECT FILE

The starting page number N of an indirect file can usually be set at 128 in OS-65D. If the program is quite large this value may not work. The indirect file must fit into memory above the program in the workspace. A value for N is given by:
N = highest page in memory — pages unused in memory

The highest page in memory can be obtained by:
?PEEK(133)

and the number of pages unused in memory can be obtained by:
?INT(FREXV/256), or if FREXV is negative, by:
?INT(USR5538 + FREXV/256)

The starting page of the program is:
page 58 (17E) for OS-65D V3.2 on an 8 inch disk.
pages 54 (27E) for OS-65D V3.2 on a 5 inch disk.
page 50 (2A7E) for V3.3 systems (see p. 40 of 65D Reference Manual)

The number of pages used by the program is:

The highest page — starting page — pages left.

If the number of pages used exceeds the number of pages left there is not enough memory available to put this program in an indirect file.

FROM WORKSPACE TO INDIRECT FILE

To move a program from the workspace to an indirect file:
1) enable the indirect file function with the following POKEs, where N is the starting page number.
POKE 9554,N

2) LIST the program between square brackets as follows: With the program in the workspace, type
LIST <RETURN> (wait for listing to end)
<SHIFT>M < > RETURN>
If the keyboard is a pooled keyboard use these commands instead:
LIST <SHIFT>K <RETURN>
wait for listing to end
<SHIFT> > RETURN>
The first bracket "[", "SHIFT"K will not appear on the video screen. The second bracket appears twice as "]"
If the end of the listing appears garbled the indirect file was not placed high enough in memory and the end of the program in the workspace has been overwritten.

FROM INDIRECT FILE TO WORKSPACE

To move a program from an indirect file to the workspace:
1) enter the appropriate POKEs where N is the starting page number of the indirect file
POKE 9368,N
2) enter the command:
<CTRL>U <RETURN>

A listing of the program in the indirect file will appear ending with the bracket closure "]". On some systems there will be a harmless error message before or after the listing. To see the contents of the workspace enter the command LIST.

MOVING PROGRAMS BETWEEN INCOMPATIBLE DISKS

To transfer a program between incompatible disks:
1) determine the starting page number N of the indirect file.
2) boot up BASIC and load the program into the workspace.
3) move the program to the indirect file using the POKEs for the system on this disk.
4) boot up BASIC on the other disk; clear the workspace with NEW.
5) move the program from the indirect file to the workspace using the POKEs for the system on this new disk.
6) PUT the program on the new disk.

(For additional details, see chapter 12 of the BASIC Reference Manual)

UTILITY PROGRAMS

A brief description of the utility program supplied with the OS-65D system (operating system restrictions are in parenthesis).

ASAMPL - Sample Assembly language program
ATNENB - Enables or disables arc tangent and print extensions (V3.3 only)
BEXEC* - Program which is run upon boot-up; displays menu.
BUFFER - Check the size of program buffers; add and delete buffers. (V3.3 only - Disk 2)

CHANGE - Permits adjustment of the following:
- Terminal width for BASIC.
- The highest page of memory available which is what BASIC and ASM use when loaded.
- The adjustment of the workspace limits for BASIC.
- The result is an empty workspace to the user specifications.

COLORS - Color adjustment program.
COMPAR - Utility for comparing diskettes. (V3.3 only)
COPIER - Utility for copying diskettes. (V3.3 only)
CREATE - Enter a file name into the directory and zero out the created file on disk.
DATTRAN - Copy data files. (V3.3 only - Disk 2)
DELETE - Remove a file name from directory.
DIR - Print unsorted disk directory.
DISASM - Generate an assembly language listing for machine code program. (V3.3 only - Disk 2)
GSOAND - Sort data files, including MDMS master files. (V3.3 only - Disk 2)
MODEM - Sets up a machine code modem routine for use with a standard RS-232 modem. (V3.3 only)
RANLIST - General random access file list utility.
RENAME - Rename a file name in directory.
REPACK - Remove REM statements and blank spaces from BASIC programs. (V3.3 only - Disk 2)
RSEQ - Change the numbering of statements in a BASIC program. (V3.3 only - Disk 2)
SECDIR - Print a sector map directory of disk.
SEQSMT - General sequential file list utility.
TRACE - Enable or disable statement number trace feature.
ZERO - Initialize contents of a data file to zeros.

SYSTEM MEMORY MAPS

<table>
<thead>
<tr>
<th>HEX</th>
<th>SYSTEM MEMORY MAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>V3.2 5&quot;</td>
<td>V3.2 8&quot;</td>
</tr>
<tr>
<td>32768</td>
<td>12576</td>
</tr>
<tr>
<td>00000</td>
<td>Buffer #1 (if used)</td>
</tr>
<tr>
<td>00510</td>
<td>Buffer #2 (if used)</td>
</tr>
<tr>
<td>00170</td>
<td>Buffer #3 (if used)</td>
</tr>
<tr>
<td>10000</td>
<td>Directory, Workspace File Header Information</td>
</tr>
<tr>
<td>10010</td>
<td>00500-00550 Extensions</td>
</tr>
<tr>
<td>10170</td>
<td>00550-00550 OFD 45-49 Extension</td>
</tr>
<tr>
<td>11070</td>
<td>00600-00650 Kernel</td>
</tr>
<tr>
<td>12482</td>
<td>00660-00660 Kernel</td>
</tr>
<tr>
<td>12512</td>
<td>00670-00670 Kernel</td>
</tr>
<tr>
<td>12580</td>
<td>00680-00680 Kernel</td>
</tr>
<tr>
<td>12696</td>
<td>00690-00690 Kernel</td>
</tr>
<tr>
<td>12780</td>
<td>00700-00700 Kernel</td>
</tr>
</tbody>
</table>

*Blank Line Indicates Normal Start of Workspace.
POKE AND PEEK LIST

As systems develop, different locations are committed to hold parameters. Many of these parameters have been mentioned in the text material. These parameters are collected here, along with some other useful parameters which may be needed by an advanced programmer. Users of the video systems and systems that include certain options and accessories (e.g., Home Security, Remote Control, High Resolution Graphics, etc.) may need to POKE or PEEK other parameter locations. These locations are fully documented in the appropriate User's Manuals. CAUTION: Care must be taken when POKEing any of these locations to avoid system errors.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>CONTENTS</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL</td>
<td>HEX (DEC)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>132</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>112</td>
</tr>
<tr>
<td>120-76</td>
<td>127</td>
<td>98 of BASIC work space (note 127 = $7F, 98 = $5C) Normal contents of Location 121 is 58 on V.3.3 and 49 on Serial Systems.</td>
</tr>
<tr>
<td>741</td>
<td>2E5</td>
<td>16</td>
</tr>
<tr>
<td>750</td>
<td>2EE</td>
<td>16</td>
</tr>
<tr>
<td>1707</td>
<td>7B6</td>
<td>32</td>
</tr>
<tr>
<td>2873</td>
<td>819</td>
<td>173</td>
</tr>
<tr>
<td>2200</td>
<td>86B</td>
<td>90</td>
</tr>
<tr>
<td>2288</td>
<td>B64</td>
<td>27</td>
</tr>
<tr>
<td>2903</td>
<td>B4D</td>
<td>55</td>
</tr>
<tr>
<td>2984</td>
<td>B4E</td>
<td>68</td>
</tr>
<tr>
<td>2972</td>
<td>B9C</td>
<td>58</td>
</tr>
<tr>
<td>2976</td>
<td>BA7</td>
<td>44</td>
</tr>
<tr>
<td>5784</td>
<td>2F9</td>
<td>41</td>
</tr>
<tr>
<td>5782</td>
<td>2212</td>
<td>27</td>
</tr>
<tr>
<td>5902</td>
<td>2206</td>
<td>99</td>
</tr>
<tr>
<td>8917</td>
<td>22D5</td>
<td></td>
</tr>
<tr>
<td>8954</td>
<td>22FA</td>
<td></td>
</tr>
<tr>
<td>8968</td>
<td>2308</td>
<td></td>
</tr>
<tr>
<td>8993</td>
<td>2321</td>
<td></td>
</tr>
<tr>
<td>8994</td>
<td>2322</td>
<td></td>
</tr>
<tr>
<td>8995</td>
<td>2323</td>
<td></td>
</tr>
<tr>
<td>8996</td>
<td>2324</td>
<td></td>
</tr>
</tbody>
</table>

LOCATION DECIMAL CONTENTS HEX (DEC) COMMENTS

(Note: Locations 8996 through 9005, 9123-9123, and 9155-9156 are used for Disk Buffer #0 (IO Flag Bit 5) device usage parameters.)

- 8996 2324 126 LO-HI byte address for the start of Buffer #6 (*contents vary: 58 on all V.3.3; 50 on 6 V.3.2; 49 on 8 V.3.2)
- 9001 2329 126 LO-HI byte address for the end of Buffer #6 (*contents vary: 66 for 5 V.3.3; 70 for 8 V.3.3; 58 for 5 V.3.2; 61 for 8 V.3.2)
- 9002 233A 126 First track of Buffer #6 File (BCD)
- 9003 232B 126 Last track of Buffer #6 File (BCD)
- 9004 232C 126 Current track in Buffer #6 (BCD)
- 9005 232D 126 Buffer #6 Dirty Flag (if contents is non-zero, then data has been written to the buffer, but has not yet been transferred to all other systems)

(Note: Locations 9006 through 9013, 9213-9214, 9238-9239 are used for Disk Buffer #7 (IO Flag Bit 6) device usage parameters.)

- 9006 233E 126 LO-HI byte address for the start of Buffer #7 (*contents vary: 56 on 5 V.3.2; 61 on 6 V.3.2; 66 on 5 V.3.3; 70 on 8 V.3.3)
- 9007 232F 126 LO-HI byte address for the end of Buffer #7 (*contents vary: 66 on 5 V.3.2; 73 on 6 V.3.2; 64 on 5 V.3.3; 82 on 8 V.3.3)
- 9008 2330 126 First track of Buffer #7 File (BCD)
- 9009 2331 126 Last track of Buffer #7 File (BCD)
- 9010 2334 126 Current track in Buffer #7 (BCD)
- 9011 2335 126 Buffer #7 Dirty Flag (= Clean; see comment for location 9006)
- 9012 236A 126 Pointer to Memory Storage Input (Lo and Hi Byte)
- 900D 23DF 126 Pointer to Memory Storage Output (Hi Byte)
- 9106 2391 126 Pointer to Memory Storage Output (Lo)
- 9107 23D5 126 LO-HI byte address of Buffer #6 *current input. (*50 on 5 V.3.2; 49 on all other systems)
- 9108 23C3 126 LO-HI byte address of Buffer #6 *current output. (*50 on 5 V.3.2; 49 on all other systems)
- 9113 23FD 126 LO-HI byte address of Buffer #7 *current input. (*62 on 5 V.3.2; 61 on all other systems)
- 9114 23FE 126 LO-HI byte address of Buffer #7 *current output. (*62 on 5 V.3.2; 61 on all other systems)
- 9115 2416 126 Indirect File Input Address (Hi Byte)
- 9116 2417 126 Indirect File Input Address (Lo Byte)
- 9117 2552 126 Pointer to Indirect File (Hi Byte) only for output (Lo = $00)
- 9118 25D2 126 Next Position for Cursor on video screen (Hi and LO Bytes) V.3.2 Video Systems only)
- 9119 25DC 126 Page Count for USRX(X)
- 911A 25DF 126 Pointer to memory for USRX(X).

- 911B 262A 64 Display control parameters. Single Space = 64; Double Space = 128; (V.3.2 Video Systems only)
- 911C 2644 126 Entry point to Keyboard Swap Routine
- 911D 265D 126 Sector for USRX(X) on Disk.
- 911E 265F 126 "Page Count for USRX(X)."
- 911F 2660 126 Pointer to memory for USRX(X). (Lo and Hi Bytes) USRX(X) will reside in location pointed to.
- 9120 2662 126 Contains track number for USRX(X) on disk (Decimal)
- 9121 2668 126 "Disable Ind." Terminator. See Location 2976 comments.
- 9122 26F8 126 Console terminal number. (*1 on Serial Systems; 2 on Video Systems)
- 9123 2700 126 Page 01 Swap Address
- 9124 2F2C 126 Sets record length for data file use
- 9125 2F0A 126 Sets Number of records per track for data file use

171 Selects Cursor character (V.3.3 only)
32 Selects Flashing cursor; 44 selects non-flashing cursor. (V.3.3 only)
ERROR MESSAGE CODES

1. Can't Read Sector (Parity Error).
2. Can't Write Sector (Reread Error).
3. Track Zero is Write Protected Against that Operation.
4. Diskette is Write Protected.
5. Seek Error (Track Header Doesn't Match Track).
6. Drive Not Ready.
7. Syntax Error in Command Line.
8. Bad Track Number.
9. Can't Find Track Header Within One Rev. of Diskette.
A. Can't Find Sector Before One Requested.
B. Bad Sector Length Value.
C. Can't Find that Name in Directory.
D. Read/Write Attempted Past End of Named File.

BASIC ERROR MESSAGE CODES

BS Bad subscript: Matrix outside DIM statement range, etc.
CN Continue Errors: Attempt to inappropriately continue from BREAK or STOP.
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 and DEF statements cannot be used in direct mode.
LS Long String: String longer than 255 characters.
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.
RG RETURN without GOSUB.
SN Syntax Error: Typo, etc.
ST String Temporaries: String expression too complex.
TM Type Mismatch: String variable mismatched to numeric variable.
UF Undefined Function.
US Undefined Statement: Attempt to jump to nonexistent line number.
/0 Division by Zero.
OS Out of String Space: Same as OM.

DOS COMMANDS

ASM Load the assembler and extended monitor. Transfer control to the assembler.

BASIC

CALL NNNN = TT,S Load contents of Track "TT", sector "S" to memory location "NNNN".

D9 Disable error 9. This is required to read some earlier version files (V1.5, V2.0) (on 8" systems only)

DIR TT Print sector map directory of track "TT". For each sector, the number of pages is given.

EM Load the assembler and extended monitor. Transfer control to the extended monitor.

EXAM NNNN = TT Examine track. Load entire track contents, including formatting information, into location "NNNN".

GO NNNN Transfer Control (GO) to location "NNNN".

HOME Reset track count to zero and HOME the current drive's head to track zero.

INIT INITIALIZE the entire disk. I.e. erase the entire diskette (except track 0) and write new formatting information on each track.

INIT TT Same as "INIT", but only operates on Track "TT".

I0, NN,MM Changes the Input I/O distributor flag to "NN", and the Output flag to "MM".

I0, MM Changes only the Output flag.

I0, NN Changes only the Input flag.

LOAD FILNAM Loads named source file, "FILNAM", into memory.

LOAD TT Loads source file into memory given starting track number "TT".

MEM NNNN, MM MM Sets the memory I/O device input pointer to "NNNN", and the Output pointer to "MMMM".

PUT FILNAM Saves source file in memory on the named disk file "FILNAM".

PUT TT Saves source file in memory on track "TT" and following tracks.

RET ASM Restart the assembler.

RET BAS Restart Basic.

RET EM Restart the extended monitor.

RET MON Restart the Prom monitor (via RSTVECTOR).

SAVE TT,S = NNNN,P Save memory from location "NNNN" on track "TT" sector "S" for "P" pages.

SELECT X Select disk drive "X" where "X" can be A, B, C, or D. Select enables the requested drive and homed the head to track 0.

XOT FILNAM Load the file, "FILNAM" as if it was an object file, and transfer control to location $3A7E (317E on 8" V3.2; 327E on 5" V3.2)

XOT TT Load the file beginning on track "TT" as if it was an object file and transfer control to location $3A7E (317E on 8" V3.2; 327E on 5" V3.2)

NOTES:
Only the first 2 characters are used in recognizing a DOS command. The rest up to the blank are ignored.
Commands can be used in the basic mode in the form DISK! "DOS" where DOS represents one of the commands above.
All memory locations should be in hex.