CHAPTER 14

PDOS ERROR DEFINITIONS

PDOS errors are divided into three groups: BASIC errors (1-49), PDOS system errors (50-99), and Device Service Routine errors (100+). Errors are returned through register R0 on all assembly primitives. Event 126 enables (0) or disables (1) error printouts from the file 'HLPTX'.

14.1 BASIC ERROR NUMBERS

ERROR 01 SYNTAX ERROR - Syntax errors are the most common error encountered with BASIC. Syntax errors result from non-parsable BASIC statements.

```
10 A=10**C              ;A(0#1)
*ERROR 01               *ERROR 01
10 A=10**C              ;A(0#1)
        ^                   ^
```

ERROR 02 UNMATCHED PARENTHESIS - Incorrect parenthesis enclosure errors occur only during the editing of a statement as parentheses are not stored in the pseudo source.

```
10 A=(B+C))
*ERROR 02
10 A=(B+C))
          ^
```

ERROR 03 NO SUCH LINE NUMBER - During statement editing, ERROR 3 is the result of an invalid line number being used by the statement. Invalid line numbers are floating point numbers or out of range. Valid line numbers range from -32767 to 32767. During program execution, ERROR 3 means that no program line exists with the line number.

```
99.9 A=10
*ERROR 03
99.9 A=10
    ^
GOSUB 1.5
*ERROR 03
GOSUB 1.5
         ^
```

ERROR 04 TOO MANY VARIABLES - Only 160 variables can be defined at any one time in a BASIC program. This includes simple, dimensioned, label, and function names.

```
10 A=B+C
*ERROR 04
10 A=B+C
     ^
```

ERROR 05 ILLEGAL CHARACTER - Various ASCII characters are illegal in any BASIC line (literals excluded). ERROR 5 also results from the ASCII byte assignment when a non-hexadecimal character is encountered. (Blanks are legal.)

```
10 A=_B
*ERROR 05
10 A=_B
     ^
$A=%"ABCDEFGH"
*ERROR 5
```

ERROR 06 MISSING ASSIGNMENT OPERATOR - An assignment operator ("=") must be found after FOR, DEFN, LET, and implied LET statements.

```
I*10
*ERROR 06
I*10
 ^
```

ERROR 07 SUBSCRIPT OUT OF RANGE - An array is dimensioned with the argument values of the first execution occurrence. Any reference to the array with larger arguments will result in error 7. This error also occurs when the byte offset is non-positive.

```
DIM A(10)
;A(20)
*ERROR 7
;A(0;0)
*ERROR 7
```

(14.1 BASIC ERROR NUMBERS continued)

ERROR 08 TOO FEW SUBSCRIPTS - When an array is referenced with less subscripts than when dimensioned, an error 8 occurs.

```
DIM B(10,20)
;B(10)
*ERROR 8
```

ERROR 09 TOO MANY SUBSCRIPTS - Only 7 subscripts are allowed for any function or variable.

```
DIM B(10)
;B(10,10,10)
*ERROR 9
;A(1,2,3,4,5,6,7,8)
*ERROR 09
;A(1,2,3,4,5,6,7,8)
                  ^

;SIN(1,2,3);
*ERROR 9
```

ERROR 10 STORAGE OVERFLOW - A storage overflow results when the heap pointer and next variable storage pointer cross. This usually occurs during a dimension statement or function call.

```
DIM C(10000)
*ERROR 10
```

ERROR 11 STACK OVERFLOW - The GOSUB and FOR/NEXT stacks are fixed in length and any nesting exceeding the stack limits result in a stack overflow error.

```
10 GOSUB 10
RUN
*ERROR 11 AT 10
```

ERROR 12 STACK UNDERFLOW - If a RETURN or NEXT statement is executed without a proper stack entry having been made (ie. GOSUB or FOR), a stack underflow occurs.

```
10 RETURN
RUN
*ERROR 12 AT 10
```

ERROR 13 ILLEGAL DELIMITER - An illegal delimiter results during program execution if a unexpected characters are used to delimit expressions.

```
FILE 3?A
*ERROR 13
EQUATE A;B
*ERROR 13
FOR I=10;20
*ERROR 13
```

ERROR 14 EXPECTING DELIMITER OR OPERATOR - Various delimiters are expected by the BASIC interpreter, such as relational delimiters must separate strings.

```
IF "AB"-"CD"
*ERROR 14
```

ERROR 15 EXPECTING VARIABLE - BASIC commands which return values require BASIC variables. An expecting variable error results if the parameter is on the evaluation stack.

```
$A=#"00";N          EQUATE A,10
*ERROR 15           *ERROR 15

EVENT A,10          PDOS 10
*ERROR 15           *ERROR 15
```

(14.1 BASIC ERROR NUMBERS continued)

ERROR 16 EXPECTING SIMPLE VARIABLE - A simple variable is required in the syntax of FOR and EQUATE statements.

```
FOR I(0)=1 TO 10          EQUATE A(1),B
*ERROR 16                 *ERROR 16
```

ERROR 17 EXPECTING DIMENSIONED VARIABLE - When equating an array to an address, a single dimensioned variable is required.

```
EQUATE T(0,0),A
*ERROR 17
```

ERROR 18 EXPECTING STRING - Commands which use string arguments require string parameters.

```
FILE 5,A                  IF ;: GOTO 10
*ERROR 18                 *ERROR 18
```

ERROR 19 EXPECTING STRING VARIABLE - A string variable is required for all string assignments.

```
"AB"=$A                   FILE 6,I
*ERROR 19                 *ERROR 19
```

ERROR 20 PARAMETER ERROR - Error 20 is a miscellaneous error for symatic discrepancies.

```
FILE 10                   PRINT @"10;"
*ERROR 20                 *ERROR 20
```

ERROR 21 READ OUT OF DATA - A READ command returns error 21 if the end of program is reached without a data item or an invalid delimiter is encountered in a DATA statement.

```
READ I
*ERROR 21
LIST
 10  READ I: PRINT I;: GOTO 10
 20  DATA 1,2;3,4
RUN
 1 2
*ERROR 21 AT 10
```

ERROR 22 READ TYPE DIFFERS FROM DATA TYPE - BASIC requires the DATA and READ types to match. (ie. Strings to string variables, numbers to numeric variables.)

```
LIST
10 READ I
20 DATA "HELLO"
RUN
*ERROR 22 AT 10
```

ERROR 23 FOR W/O NEXT - A NEXT statement is required for all FOR statements. The NEXT statement must be the first statement of the line if the start value is greater than the end value.

```
10 FOR I=10 TO 1
RUN
*ERROR 23 AT 10
```

ERROR 24 NEXT W/O FOR - An error 24 results if the FOR/NEXT stack does not contain an entry with the same simple variable argument of the NEXT statement.

```
LIST
 10  FOR I=1 TO 10
 20  NEXT J
RUN
*ERROR 24 AT 20
```

ERROR 25 ILLEGAL FUNCTION NAME - A function name consists of the letters 'FN' followed by a valid variable name.

```
I=FN10(10)
*ERROR 25
I=FN10(10)
    ^
```

(14.1 BASIC ERROR NUMBERS continued)

ERROR 26	ILLEGAL FUNCTION OR LOCAL ARGUMENT - A DEFN requires a valid function name. LOCAL parameters must be simple variables.	DEFN A(10) *ERROR 26 LOCAL A(10) *ERROR 26
ERROR 27	UNDEFINED FUNCTION OR FUNCTION W/O FNEND - Functions are defined with a prepass of the program code when 'RUN' is executed. Any functions not defined or entered after the 'RUN' command result in error 27. All function definitions require a FNEND as the first command of a statement at the foot of the definition.	10 I=FNFANC(10) RUN *ERROR 27 AT 10 LOCAL A *ERROR 27 LIST 10 DEFN FNA 20 FNA=1: FNEND RUN *ERROR 27 AT 10
ERROR 28	DIVISION BY ZERO - Zero is not a legal divisor.	;1/0; *ERROR 28
ERROR 29	FLOATING POINT OVERFLOW - Floating point overflows result from numbers greater than approximately 1E75 and less than 1E-78.	;1E99; *ERROR 29
ERROR 30	FIX ERROR - Fix errors result from numbers which cannot be integerized to one word. The range is from -32767 to 32767.	;MEMW(68000) *ERROR 30
ERROR 31	SQUARE ROOT OF NEGATIVE NUMBER - The square root of a negative number is undefined.	;SQR -1 *ERROR 31
ERROR 32	LOG OF NON-POSITIVE NUMBER - The natural log of a non-positive number is undefined.	;LOG -1 *ERROR 32 ;LOG 0 *ERROR 32
ERROR 33	INVALID SYS FUNCTION ARGUMENT - Error 33 results from too large of a SYS argument or trying to write to a read-only SYS value.	;SYS(50); *ERROR 33 SYS(20)=10 *ERROR 33
ERROR 34	UNIMPLEMENTED BASIC COMMAND - Disk and interpreter commands are illegal in the run module and result in error 34 if executed. This error is reported on the system LED.	Run module error

14.2 PDOS ERROR NUMBERS

ERROR 50 INVALID FILE NAME. Valid file names consist of an alpha character followed by up to 7 alpha-numeric characters. An optional extension and disk number may follow. An extension consists of a colon followed by 1 to 3 characters. A disk number consists of a slash and a number ranging from 0 to 127.

```
.DKDKDKDKF
PDOS ERR 50
.
```

ERROR 51 FILE ALREADY DEFINED. Each file name is unique to a disk file directory. There is one directory per disk number.

```
.DF FILE1
.DF FILE1
PDOS ERR 51
.
```

ERROR 52 FILE NOT OPEN. An attempt to access a file which has not been opened, results in error 52.

```
.EX
FILE 1,1;3,I
*ERROR 52
```

ERROR 53 FILE NOT DEFINED. If the file name does not exist in the disk directory, an error 53 occurs.

```
.SF FILE2
PDOS ERR 53
.
```

ERROR 54 INVALID FILE TYPE. Valid file types are AC, BN, OB, SY, BX, EX, TX, UD, *, and **. All others result in error.

```
.SA FILE1,TR
PDOS ERR 54
.
```

ERROR 55 NOT ENOUGH CONTIGUOUS SECTORS. Error 55 results from attempting to define a contiguous file on a disk unit which does not have enough room or is fragmented such that there is not a big enough contiguous block of sectors.

```
.DF FILE2,10000
PDOS ERR 55
.
```

ERROR 56 END-OF-FILE. Error 56 comes from an attempt to read past the END-OF-FILE index of a file.

```
.EX
*READY
OPEN "#PAUL",F
FILE 1,F;3,I
*ERROR 56
```

ERROR 57 FILE DIRECTORY FULL. The file directory size is set when the file is initialized. Any attempt to define another file after the directory has been filled, results in error 57.

```
.DF FILE3
PDOS ERR 57
.
```

ERROR 58 FILE DELETE PROTECTED. An attempt to delete a file with a delete or write protect flag results in error 58.

```
.SA TEMP,*
.DL TEMP
PDOS ERR 58
.
```

(14.2 PDOS ERROR NUMBERS continued)

ERROR 59	INVALID SLOT #. A valid file slot number is returned from PDOS on all open commands. A file slot consists of the the disk number in the left byte and the slot index in the right byte.	.EX *READY FILE 1,F;3,I *ERROR 59
ERROR 60	FILE SPACE FULL. An attempt to extend a file or define a file after the disk space is filled results in error 60.	.CF TEMP,LIST PDOS ERR 60 .
ERROR 61	NO START ADDRESS. An object (OB) or system (SY) must have a start address. This is generated by an address parameter for the 'END' statement in the assembly source.	.TEMP PDOS ERR 61 .
ERROR 62	FILE ALREADY OPEN. A file can be opened only once in sequential (XSOP) and random (XROP) modes. Read only open (XROO) and shared random open (XNOP) can be executed more than once on the same file.	.EX *READY OPEN "LIST",F OPEN "LIST",F *ERROR 62
ERROR 63	ILLEGAL OBJECT TAG. Only TI object tags 0 (program IDT), 2 (relocatable entry), 7 (checksum), 8 (ignore checksum), A (relocatable load address), B (absolute data), D (relocatable data), and F (end of record). All others are illegal.	.SA TEST:SR,OB .TEST:SR PDOS ERR 63 .
ERROR 64	ILLEGAL PORT NUMBER OR BAUD RATE. Only 1 through 9 are legal ports. Valid baud rates are 110, 300, 600, 1200, 2400, 4800, 9600, and 19200.	.BP 2,1250 PDOS ERR 64 .BP 10,9600 PDOS ERR 64 .
ERROR 65	EXCEEDS TASK SIZE. Each data entry of an object file is checked against the task upper limit. Overflows are reported as error 65.	.ADVENT PDOS ERR 65 .
ERROR 66	FILE NOT LOADABLE. Only files typed 'OB', 'SY', 'EX', and 'BX' are loadable by the monitor loader.	.SA ASM,UD .ASM PDOS ERR 66 .
ERROR 67	INVALID PARAMETER. Most monitor commands check parameters for valid ranges and types.	.IM 0 PDOS ERR 67 .

(14.2 PDOS ERROR NUMBERS continued)

ERROR 68	DISK NOT FORMATTED. A initialized PDOS disk has the constant >A55A at location >0028 of the header sector (sector 0). If not found on a disk read, error 68 results.	.LS /2 PDOS ERR 68 .
ERROR 69	NOT ENOUGH FILE SLOTS. A maximum of 32 files can be open at a time. These correspond to the 32 file slots.	.CF TEMP,TEMP1 PDOS ERR 69 .
ERROR 70	POSITION ERROR. Error 70 results from a position command beyond the end-of-file index. A position error also occurs if a position or rewind command is executed on a file not opened for random access.	.EX *READY OPEN "#PAUL",F FILE 1,F;4,0 *ERROR 70
ERROR 71	SYSTEM FILE ERROR	
ERROR 72	TOO MANY TASKS. The task list is defined when the PDOS system is generated. For 101/MA systems, the limit is 16 tasks. For 102 systems, the limit is 11.	.@CF LIST,$TTA PDOS ERR 72 .
ERROR 73	NOT ENOUGH MEMORY. An attempt to create a task with more memory than the current task or available memory in the system memory bit maps, results in error 73.	.CT ,40,,1 PDOS ERR 73 .
ERROR 74	NO SUCH TASK. Error 74 occurs when referencing a task not in the task list or task 0.	.KT 5 PDOS ERR 74 .
ERROR 75	FILE LOCKED. Once a file has been locked (XLKF), it cannot be accessed until unlocked (XULF).	.CF FDATA,TEMP PDOS ERR 75 .
ERROR 76	TASK LOCKED. Once a task has been locked (XLKT), it cannot be killed until unlocked (XULT).	.KT 5 PDOS ERR 76 .
ERROR 77	PROCEDURE NOT MEMORY RESIDENT. If PDOS BASIC has been dropped or replaced, all 'BX' and 'EX' files do not execute. Also, the interpreter cannot be entered with the 'EX' command.	.RECOVER >3000 .EX PDOS ERR 77
ERROR 78	MESSAGE BUFFER FULL. There are only 8 message buffers currently in the PDOS system. Too many messages results in error 78.	.SENDM 4,ANOTHER MESSAGE PDOS ERR 78 .

(14.2 PDOS ERROR NUMBERS continued)

ERROR 79 MEMORY PAGE ERROR. Only pages 0 through 7 are valid in a paged memory system. Any other page reference results in error 79.

```
.IMP 9
PDOS ERR 79
.
```

ERROR 80 CHECKSUM ERROR. Error 80 results from memory between >0080 and >2000 summing to a nonzero value. The checksum location is at memory address >0082.

ERROR 81 PDOS PRIMITIVE NOT IMPLEMENTED

```
.EX
*READY
MEM(092H)=0
BYE
.CS
PDOS ERR 80
Run module error
```

14.3 DEVICE SERVICE ROUTINE ERRORS

ERROR 100 ILLEGAL DISK #. It is the responsibility of the boot EPROMs to detect and report illegal disk numbers. Disk numbers range from 0 to 127. The disk number is passed in register R0.

```
.LS /10
PDOS ERR 100
.
```

ERROR 101 SECTOR # TOO LARGE. It is the responsibility of each individual disk DSR to test and report sector access too large for the particular device. The sector number is passed in register R1.

ERROR 102 NOT READY. A not ready error is a device timeout condition. DSR's need to ensure that a failure of a physical device does not hang the system.

ERROR 103 WRITE PROTECT. Each DSR should monitor DSR write protect signals and report them to PDOS as error 103.

ERROR 109 PAGE BOUNDARY ERROR (102). DSR DMA routines generally cannot cross memory mapped boundaries. Hence, error 109 reports buffers which are not on 256 byte boundaries.

ERROR 110 SINGLE/DOUBLE SIDED DISKETTE CONFLICT. Devices which report the type of diskette in the drive can be used to report diskette incompatibilities. The double sided flag in the header sector (sector 0) at location >0030 should match the drive type.