HLA Exceptions Appendix G

The HLA Standard Library provides the following exception types[1]:

ex.StringOverflow

HLA raises this exception if you attempt to store too many characters into preallocated string variable. The following standard library routines can raise this exception:

pat.extract, strrealloc, stdin.gets, str.cpy, str.setstr, str.cat, str.substr, str.insert, console.gets, cStrToStr, dToStr, e80ToStr, hToStr, u64ToStr, i64ToStr, qToStr, r80ToStr, tbToStr, wToStr, date.print, date.toString, and date.a_toString.

ex.StringIndexError

HLA raises this exception if a routine attempts to use an index that is beyond the last valid character in a string. The following standard library routines can raise this exception:

str.span2, str.rspan2, str.brk2, str.rbrk2, str.substr, str.a_substr, strToFlt, StrToi8, StrToi16, StrToi32, StrToi64, StrTou8, StrTou16, StrTou32, StrTou64, StrToh, StrTow, StrTod, and StrToq.

ex.ValueOutOfRange

HLA raises this exception if an arithmetic overflow occurs, if an input parameter is out of range, or if user input is too great for the destination variable. The following standard library routines can raise this exception:

arg.v, arg.delete, rand.urange, rand.range, stdin.geti8, stdin.geti16, stdin.geti32, stdin.geti64, stdin.getu8, stdin.getu16, stdin.getu32, stdin.getu64, stdin.geth, stdin.getw, stdin.getd, stdin.getq, stdin.getf, table.create, console.a_getRect, console.fillRect, console.fillRectAttr, console.getc, console.getRect, console.gets, console.gotoxy, console.putRect, console.scrollDnRect, console.scrololUpRect, atof, atoh, atoi8, atoi16, atoi32, atoi64, atou8, atou16, atou32, atou64, e80ToStr, r80ToStr, StrToi8, StrToi16, StrToi32, StrToi64, StrTou8, StrTou16, StrTou32, StrTou64, StrToh, StrTow, StrTod, StrToq, fileio.getd, fileio.geth, fileio.getw, fileio.getq, fileio.geti8, fileio.geti16, fileio.geti32, fileio.geti64, fileio.getu8, fileio.getu16, fileio.getu32, fileio.getu64, fileio.pute80pad, fileio.pute64pad, fileio.pute32pad, fileio.putr32Pad, fileio.putr64Pad, and fileio.putr80Pad.

ex.IllegalChar

Several HLA routines raise this exception if they encounter a non-ASCII character (character code $80..$FF) where a delimiter character is expected. Generally, you can treat this error as though it were a conversion error. Routines that raise this exception include:

atoh, atoi8, atoi16, atoi32, atoi64, atou8, atou16, atou32, and atou64.

ex.ConversionError

Routines in the HLA Standard Library raise this exception if there is an error convertion data from one format to another. Typically this occurs when converting strings to numeric data. Routines that raise this exception include:

1. Please note that the HLA Standard Library is under constant revision and the list appearing in this chapter may be slightly out of date. Please consult the HLA Standard Library documentation for an up-to-date listing of exceptions and the routines that raise them.

stdin.geti8, stdin.geti16, stdin.geti32, stdin.geti64, stdin.getu8, stdin.getu16, stdin.getu32, stdin.getu64, stdin.geth, stdin.getw, stdin.getd, stdin.getq, stdin.getf, atof, atoh, atoi8, atoi16, atoi32, atoi64, atou8, atou16, atou32, atou64, fgetf, fileio.geti8, fileio.geti16, fileio.geti32, fileio.geti64, fileio.getu8, fileio.getu16, fileio.getu32, fileio.getu64, fileio.geth, fileio.getw, fileio.getd, and fileio.getq.

ex.BadFileHandle

The file class method file.handle raises this exception if the handle associated with a file class object is illegal (has not been initialized).

ex.FileOpenFailure

The fileio.Open, fileio.OpenNew, file.Open, and file.OpenNew procedures raise this exception is there is an error opening a file.

ex.FileCloseError

The fileio.Close and file.Close procedures raise this exception if there is some sort of error when attempting to close a file.

ex.FileWriteError

Those routines that write data to a file (e.g., fputi8) raise this exception if there is an error writing data to the output file.

ex.FileReadError

Those routines that read data from a file (e.g., fileio.geti8) raise this exception if there is a physical error reading the data from the file.

ex.DiskFullError

Those routines that write data to a file will raise this exception if an attempt is made to write data to a full disk.

ex.EndOfFile

Those routines that read data from a file will raise this exception if your program attempts to read data beyond the end of the file.

ex.MemoryAllocationFailure

Routines that allocation storage (e.g., malloc and realloc) will raise this exception if Windows cannot satisify the memory allocaiton requestion. Several routines in the standard library may raise this exception since they indirectly call the HLA *malloc* routine. Examples include *stdin.a_gets* and almost any other Standard Library routine that has "a_" as a prefix to the name.

ex.AttemptToDerefNULL

Several routines in the Standard Library that expect a string pointer will raise this exception if the string pointer (or other pointer) contains NULL (zero). Examples include:

getf, str.cpy, str.a_cpy, str.setstr, str.cat, str.a_cat, str.index, str.rindex, str.chpos, str.rchpos, str.span, str.span2, str.rspan, str.rspan2, str.brk, str.brk2, str.rbrk, str.rbrk2, str.eq, str,ne, str.lt, str.le, str,gt, str.ge, str.substr, str.a_substr, str.insert, str.a_insert, str.delete, str.a_delete, str.ieq, str.ine, str.ilt, str.ile, str.igt, str.ige, str.upper, str.a_upper, str.lower, str.a_lower, str.delspace, str.a_delspace, str.trim, str.a_trim, str.tokenize, str.tokenize2, atof, atoi8, atoi16, atoi32, atoi64, atou8, atou16, atou32, atou64, dtostr, htostr, wtostr, qtostr, strToFlt, StrToi8, StrToi16, StrToi32, StrToi64, StrTou8, StrTou16, StrTou64, StrToh, StrTow, StrTod, StrToq, and tbtostr.

ex.WidthTooBig

HLA routines that print a numeric value within a field width will raise this exception if the specified width exceeds 1,024 characters. Routines that raise this exception include *stdout.pu*t, *stdout.puti8Size*, *fputu32Size*, and all other *xxxxSize* output routines.

ex.TooManyCmdLnParms

The *arg.CmdLn* procedure raises this exception if it determines that there are more than 256 command line parameters on the command line (a virtual impossibility since command lines are generally limited to 128 characters). Since all of the other routines in the *args* module can call arg.CmdLn, it is possible for any of the routines in this module to raise this exception.

ex.ArrayShapeViolation

Routines in the arrays module raise this exception if the dimension on some array are inappropriate for the specified operation. For example, the array.cpy code will raise this exception if you attempt to copy a source array to a destination whose dimensions don't exactly match the source array.

ex.InvalidDate

The routines in the datetime module will raise this exception if you pass them an illegal date value as a parameter.

ex.InvalidDateFormat

The date output routines (date.print and date.toString) raise this exception if you attempt convert a date to a string but the current (internal) date format variable contains an invalid value. This usually implies that you've passed an incorrect parameter to the *date.SetFormat* procedure.

ex.TimeOverflow

The time.secsToHMS procedure raises this overflow if there is an error converting time in seconds to hours, minutes, and seconds (very rare, only occurs above two billion seconds).

ex.AccessViolation

This is a hardware exception that Windows raise if your program attempts to access an illegal memory location (generally a NULL reference or an uninitialized pointer).

ex.Breakpoint

This exception is raised by Windows for debugger programs; you should never see this exception unless you are writing a debugger program.

ex.SingleStep

This is another exception that is raised by Windows for debugger programs; you should never see this exception unless you are writing a debugger program.

ex.PrivInstr

Windows raises this instruction if you attempt to execute a special instruction that is illegal in “user mode” (that is, can only be executed by Windows). Since HLA only compiles a few priviledged instructions, and this book doesn’t discuss them at all, the only way you’ll probably see this exception occur is if your program jumps off into (non-code) memory somewhere and begins executing data as instructions.

ex.IllegalInstr

Windows raises this exception if the CPU attempts to execute some code that is not a valid instruction. This generally implies that your program has jumped off into data memory and is attempting to execute data as machine instructions.

ex.BoundInstr

Windows raises this exception if you execute the BOUND instruction (see “Some Additional Instructions: INTMUL, BOUND, INTO” on page 393) and the register value is outside the specified range.

ex.IntoInstr

Windows raises this exception if you execute the INTO instruction and the overflow flag is set (see “Some Additional Instructions: INTMUL, BOUND, INTO” on page 393).

ex.DivideError

Windows raises this exception if you attempt an integer division by zero, or if the quotient of a division is too large to fit within the destination operand(AL, AX, or EAX).

ex.fDenormal

ex.fDivByZero

ex.fInexactResult

ex.fInvalidOperation

ex.fOverflow

ex.fStackCheck

ex.fUnderflow

Windows raises one of these exceptions if you’ve enabled exceptions on the FPU and one of the specified conditions occurs (e.g., a floating point division by zero will raise the *ex.fDivByZero* exception).

InvalidHandle

Windows will raise this exception if you pass an uninitialized or otherwise invalid handle value to a Windows API (application programmer’s interface) routine. Many of the HLA Standard Library routines pass handles to Windows, so this exception could occur as a result of a call to an input/output routine.

StackOverflow

Windows raises this exception if the stack exceeds the storage allocated to it (16Mbytes in a typical HLA program).

ControlC

HLA raises this exception if the user presses control-C or control-Break during program execution.