13 Exceptions Module (excepts.hhf)

The exceptions module contains several things of interest. First, it defines the ExceptionValues enumerated data type that lists out all the standard exceptions in the HLA Standard Library. The second thing provided in the excepts unit is the ex.PrintExceptionError procedure which prints a string associated with the exception number in EAX. Next, the excepts.hhf header file defines the "assert(expr)" macro. Finally, the excepts.hhf header file defines some procedures that the HLA run-time system uses to maintain the exception handling system; these procedures are of interest only to those who want to override the default HLA exception handling mechanisms.

13.1 The Exceptions Module

To use the exceptions functions in your application, you will need to include one of the following statements at the beginning of your HLA application:

```
#include( "excepts.hhf" )
or
#include( "stdlib.hhf" )
```

13.2 Exception Resource Reduction

By default, if you include excepts.hhf or stdlib.hhf in your HLA main program, HLA will automatically link in a set of strings that describe, in detail, each of the possible exceptions. This string data consumes a considerable amount of space and may not be necessary if you're not taking advantage of HLA's exception-handling system.

HLA will link in these strings if the compile-time variable @exceptions contains true when HLA encounters the **begin** associated with the main program. If you would like HLA to link in a single (small) string in place of this huge table of strings, just set the @exceptions compile-time variable to false after include excepts.hhf (or stdlib.hhf), e.g,

```
#include( "excepts.hhf" )
?@exceptions := false;
```

This will reduce the size of your executable. Note, however, that you'll get a single "unhandled exception" error message if an unhandled error ever comes along. So during debugging, it's probably best to leave the exception strings in the program and remove them only for a release of your program.

13.3 Exception Constants

The following paragraphs describe each of the standard HLA exception constants and describe the conditions that lead to the Standard Library routines raising these exceptions. The "excepts.hhf" header file defines these constants. Since this list changes frequently, please refer to the excepts.hhf header file for the most recent list of exception names. HLA and the HLA Standard Library only raise these exceptions; user applications, however, may define other exceptions in addition to these. Of course, user applications may also raise exceptions using these exception constants. Note that the following numeric constants for the exception names are subject to change.

```
/*  0 */UnknownException,

  // String related exceptions:

/*  1 */StringOverflow,
/*  2 */StringIndexError,
/*  3 */StringOverlap,
/*  4 */StringMetaData,
/*  5 */StringAlignment,
/*  6 */StringUnderflow,
/*  7 */IllegalStringOperation,

  // General exceptions:
```

```
    /*  8 */ValueOutOfRange,
    /*  9 */IllegalChar,
    /* 10 */AttemptToDerefNULL,
    /* 11 */TooManyCmdLnParms,
    /* 12 */AssertionFailed,
    /* 13 */ExecutedAbstract,
    /* 14 */BadObjPtr,
    /* 15 */InvalidAlignment,
    /* 16 */InvalidArgument,
    /* 17 */BufferOverflow,
    /* 18 */BufferUnderflow,
    /* 19 */IllegalSize,

      // Formatting and conversion errors:

    /* 20 */ConversionError,
    /* 21 */WidthTooBig,
    /* 22 */FractionTooBig,

      // File-related exceptions:

    /* 23 */BadFileHandle,
    /* 24 */FileNotFound,
    /* 25 */FileOpenFailure,
    /* 26 */FileCloseError,
    /* 27 */FileWriteError,
    /* 28 */FileReadError,
    /* 29 */FileSeekError,
    /* 30 */DiskFullError,
    /* 31 */AccessDenied,
    /* 32 */EndOfFile,

      // FileSys-related exceptions:

    /* 33 */CannotCreateDir,
    /* 34 */CannotRemoveDir,
    /* 35 */CannotRemoveFile,
    /* 36 */CDFailed,
    /* 37 */CannotRenameFile,

      // Memory management exceptions:

    /* 38 */MemoryAllocationFailure,
    /* 39 */MemoryFreeFailure,
    /* 40 */MemoryAllocationCorruption,
    /* 41 */AttemptToFreeNULL,
    /* 42 */BlockAlreadyFree,
    /* 43 */CannotFreeMemory,
    /* 44 */PointerNotInHeap,

      // Array exceptions:

    /* 45 */ArrayShapeViolation,
    /* 46 */ArrayBounds,

      // Time/date exceptions:

    /* 47 */InvalidDate,
    /* 48 */InvalidDateFormat,
    /* 49 */TimeOverflow,
    /* 50 */InvalidTime,
```

```
    /* 51 */InvalidTimeFormat,

      // Socket Errors:

    /* 52 */SocketError,

      // Thread Errors:

    /* 53 */ThreadError,

      // Hardware/OS related exceptions

    /* 54 */AccessViolation,
    /* 55 */InPageError,
    /* 56 */NoMemory,
    /* 57 */InvalidHandle,
    /* 58 */ControlC,
    /* 59 */StackOverflow,
    /* 60 */Breakpoint,
    /* 61 */SingleStep,
    /* 62 */PrivInstr,
    /* 63 */IllegalInstr,
    /* 64 */BoundInstr,
    /* 65 */IntoInstr,
    /* 66 */DivideError,
    /* 67 */fDivByZero,
    /* 68 */fInexactResult,
    /* 69 */fInvalidOperation,
    /* 70 */fOverflow,
    /* 71 */fUnderflow,
    /* 72 */fStackCheck,
    /* 73 */fDenormal,

      // Blob related exceptions

    /* 74 */BlobOverflow
```

ex.UnknownException

This is a reserved value that HLA's Standard Library functions do not raise. The HLA run-time system displays this exception value if it cannot figure out the source of the interrupt. ex.PrintExceptionError calls also use this value to display an appropriate message for unhandled user exceptions.

ex.StringOverflow

The string functions in the HLA Standard Library raise this exception if the caller attempts to store too many characters into a string variable (causing a string overflow error).

ex.StringIndexError

Some string functions require a parameter that supplies an index into a string. If those functions require that the index be within the range 0..length-1, they will raise this exception to denote an index out of range error.

ex.ValueOutOfRange

Several HLA Standard Library routines raise this exception if an integer calculation overflows. The best examples are the integer input routines (e.g., stdin.geti8) that will raise this exception if the user's input is otherwise legal but out of range for the specific data type (i.e., -128..+127 for stdin.geti8).

ex.IllegalChar

Certain input and conversion routines raise this exception if an unexpected character comes along. An unexpected character is usually a non-ASCII character (character codes in the range $80..$FF). Note that the conversion and input routines do not raise this exception if a non-digit character comes along. See ex.ConversionError to see how the HLA Standard Library handles that exception.

ex.AttemptToDerefNULL

Many HLA Standard Library routines expect a pointer to some object as a parameter. If they do not allow a NULL pointer value (zero) the routines may explicitly test for a NULL value and raise this exception if the user inadventently passes in a NULL pointer. Also see the *ex.AccessViolation* exception.

ex.TooManyCmdLnParms

The args.hhf module raises this exception if you specify too many command line parameters. The exact maximum value may vary between versions of the HLA Standard Library, but it's typically a value like 64 or 128.

ex.AssertionFailed

The HLA *assert* statement raises this expression if the value of the assertion expression evaluates false. See the section on assertions later in this section for more details.

ex.ExecutedAbstract

The HLA run-time system raises this exception if you attempt to execute an abstract class method that has not been overridden and defined.

ex.BadObjPtr

The HLA run-time system raises this exception if you attempt to execute a class method using an illegal pointer.

ex.InvalidAlignment

HLA raises this exception if you specify an illegal alignment value for an allocation operation.

ex.ConversionError

HLA raises this exception whenever there is some sort of error converting data from one from to another (usually, this exception occurs when converting string data to numeric data). For example, when converting a string to an integer value, the HLA Standard Library will raise this exception if it encounters a character that is not legal for that numeric type and is not a delimiter character.

ex.WidthTooBig

Certain numeric conversion and output functions let you specify a field width value for the conversion. Those routines raise this exception if that field width value is too large (this is nominally 256, but the exact value may be different).

ex.BadFileHandle

The file class and fileio library modules raise this exception if you attempt to read from or write to a file with an illegal file handle (i.e., the file has not been opened or has already been closed).

ex.FileNotFound

HLA raises this exception if you attempt to open (or otherwise access by name) a file and the system could not find the path/filename you specified.

ex.FileOpenFailure

The HLA file open routines raise this error if there was a catastrophic error opening a file.

ex.FileCloseError

The HLA file close routines raise this error if there was an error closing a file.

ex.FileWriteError

The HLA Standard Library file output routines raise this exception if there is an error while attempting to write data to a file. This is usually a catastrophic error such as file I/O or some hardware error.

ex.FileReadError

The HLA Standard Library file output routines raise this exception if there is an error while attempting to read data from a file. This is usually a catastrophic error such as file I/O or some hardware error.

ex.FileSeekError

The HLA Standard Library file routines raise this exception if there was an error while attempting to seek to some new position in a file.

ex.DiskFullError

The HLA Standard Library raises this exception if you attempt to write data to a disk that is full.

ex.AccessDenied

The HLA Standard Library raises this exception if you attempt to access a file for which you do not have proper access permission.

ex.EndOfFile

The HLA Standard Library file I/O routines raise this exception if you attempt to read data from a file after you've reached the end of file. Note that HLA does not raise this exception upon reaching the EOF. You must actually attempt to read beyond the end of the file.

ex.CannotCreateDir

The HLA Standard Library mkdir function raises this exception if you attempt to create a subdirectory and the system returns an error.

ex.CannotRemoveDir

The HLA Standard Library rmdir function raises this exception if you attempt to remove a subdirectory and the system returns an error.

ex.CannotRemoveFile

The HLA Standard Library rmdir function raises this exception if you attempt to remove a file and the system returns an error.

ex.CDFailed

The HLA Standard Library raises this exception if you attempt to switch to a new working directory and the system could not find that directory or the change working directory operation otherwise failed.

ex.CannotRenameFile

The HLA Standard Library raises this exception if you attempt to rename a file and the operation failed.

ex.MemoryAllocationFailure

HLA raises this exception if a function attempts to allocate storage and the memory allocation operation fails (because of insufficent storage).

ex.MemoryFreeFailure

HLA raises this exception if you attempt to free storage and the request could not be satisfied (see also: CannotFreeMemory).

ex.MemoryFreeFailure

HLA raises this exception if you attempt to free storage and the request could not be satisfied (see also: CannotFreeMemory).

ex.AttemptToFreeNULL

HLA raises this exception if you attempt to free storage storage but you pass a NULL pointer to be freed.

ex.BlockAlreadyFree

HLA raises this exception if you attempt to free storage storage that has already been freed.

ex.CannotFreeMemory

The HLA memory free routines raise this exception if there is an error deallocating memory that was (presumably) allocated earlier.

ex.PointerNotInHeap

The HLA memory management routines raise this exception if you pass a pointer to an object that is supposed to be on the heap, but the pointer does not reference any object on the heap.

ex.ArrayShapeViolation

The arrays.hhf module raise this exception if you attempt to copy data from one array to another or otherwise operate on two arrays with incompatible "shapes." The "shape" of an array is the number of dimensions and the bounds on each dimension of that array. Compatible arrays typically have the same number of dimensions and the same bounds on each dimensions (though there are some exceptions to this rule).

ex.ArrayBounds

The arrays.hhf module raises this exception if you attempt to supply the wrong number of array dimensions or one of the array indices is out of bounds for that array.

ex.InvalidDate

The HLA datetime.hhf module raises this expression if you supply an illegal date to a date function. Note that legal dates must fall between Jan 1, 1600 and Dec 31, 9999 and must have valid day and month values (depending on the month and year).

ex.InvalidDateFormat

The HLA date conversion routines raise this exception if the internal date format value is illegal.

ex.TimeOverflow

The HLA datetime.hhf module raises this exception if, during a time calculation, an overflow occurs.

ex.InvalidTime

The HLA datetime.hhf module raises this expression if you supply an illegal time to a time function. Note that legal times must fall between 00:00:00 and 23:59:59.

ex.InvalidTimeFormat

The HLA time conversion routines raise this exception if the internal time format value is illegal.

ex.AccessViolation

This is a hardware exception that the CPU raises if you attempt to access an illegal memory or I/O location.

ex.InPageError

This is a hardware exception that the CPU raises if you attempt to access an illegal memory or I/O location.

ex.NoMemory

This is an exception that the OS raises if it cannot provide memory for the requested operation.

ex.InvalidHandle

This is an exception that the OS raises if it you pass it an invalid handle value for some operation.

ex.ControlC

If control-C checking is enabled, Windows will raise this exception whenever the user presses control-C on the console device.

ex.StackOverflow

The OS raises this exception if the hardware (80x86) stack exceeds the bounds set by the linker.

ex.Breakpoint

This is a hardware exception that the CPU raises if you execute an INT 3 (breakpoint) instruction.

ex.SingleStep

This is a hardware exception that the CPU raises after each instruction if the trace flag is set in the EFLAGs register.

ex.PrivInstr

This is a hardware exception that the CPU raises if you attempt to execute a priviledged instruction while in user (non-kernel) mode.

ex.IllegalInstr

This is a hardware exception that the CPU raises if you attempt to execute an opcode that is not a legal 80x86 instruction.

ex.BoundInstr

This is a hardware exception that the CPU raises if you execute a BOUND instruction and the register value is not within the bounds specified by the BOUND memory operand(s).

ex.IntoInstr

This is a hardware exception that the CPU raises if you execute an INTO instruction and the overflow flag is set.

ex.DivideError

This is a hardware exception that the CPU raises if you attempt to divide by zero or if the quotient will not fit in the destination operand.

ex.fDivByZero

This is a hardware exception that the FPU raises if you've enable floating point exceptions and a floating point division by zero occurs.

ex.fInexactResult

This is a hardware exception that the FPU raises if you've enable floating point exceptions and a floating point operation produces an inexact result.

ex.fInvalidOperation

This is a hardware exception that the FPU raises if you've enable floating point exceptions and you attempt an illegal operation on the FPU.

ex.fOverflow

This is a hardware exception that the FPU raises if you've enable floating point exceptions and a floating point operation produces an overflow (see *ex.fDenormal* and *ex.fUnderflow* for underflows).

ex.fUnderflow

This is a hardware exception that the FPU raises if you've enable floating point exceptions and an underflow occurs.

ex.fStackCheck

This is a hardware exception that the FPU raises if you've enable floating point exceptions and an FPU stack overflow occurs.

ex.fDenormal

This is a hardware exception that the FPU raises if you've enable floating point exceptions and a floating point operation produces a demormalized result.

13.4 Exception Messages

The exceptions module provides two functions for converting exception numbers into meaningful messages.

procedure ex.exceptionMsg(exceptionCode:dword; msg:string);

ex.exceptionMsg converts the exception code passed in the exceptionCode parameter to a string message and stores the resulting string into the string pointed at by the msg parameter. The *msg* string **must** be large enough to hold the result (128 characters should be sufficient). Note that this function cannot raise any exceptions because it may be called from inside an exception handler, hence the requirement that *msg* be of sufficient size to hold the string.

Note: no "ex.a_exceptionMsg" function exists because the error code resulting in the call to the function might be an "out of memory" error and it wouldn't do to have this function produce an error.

If the exception code is outside the range of the valid exception codes, this function returns the message associated with the "unknown exception" code.

HLA high-level calling sequence examples:

```
static
    msg:str.strvar(256);
        .
        .
        .
    ex.exceptionMsg( someCode, msg );
    stdout.put( msg );
```

procedure ex.printExceptionError;

ex.printExceptionError displays the message associated with the exception code in EAX in a form appropriate to the OS (e.g., under Windows this brings up a dialog box, under Linux this prints the message to the standard error device).