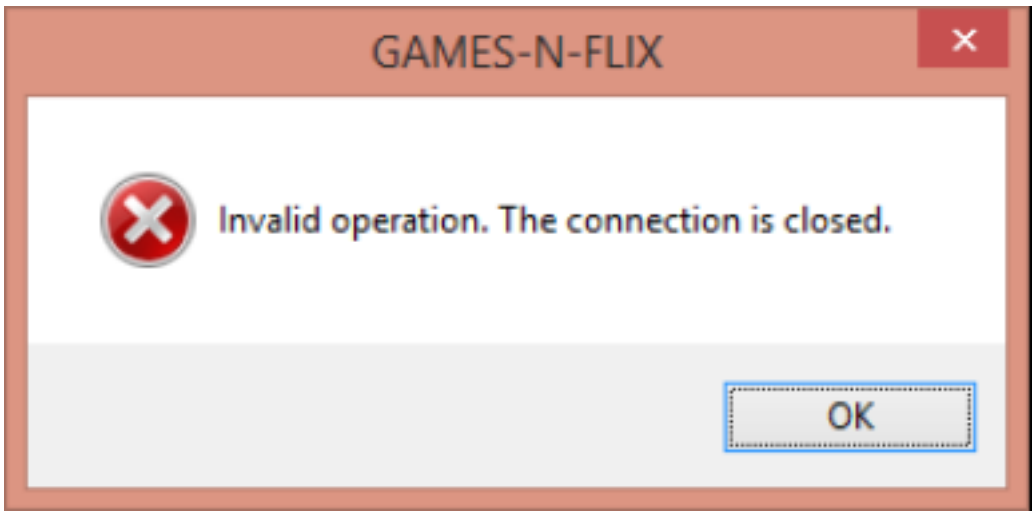
Floating-Point Exceptions

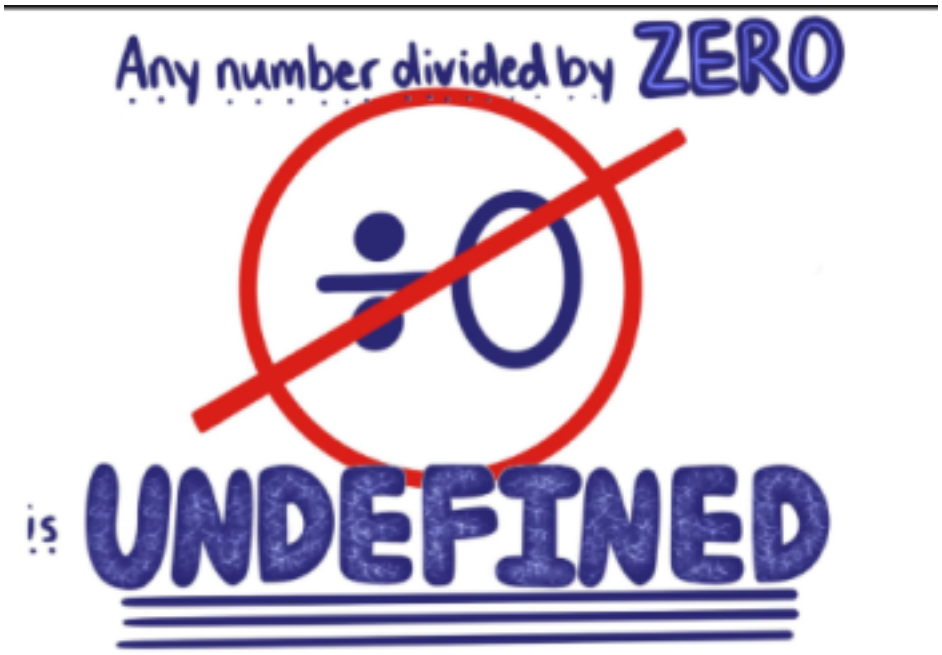
The passage below, describes floating-point exceptions and how the FPU handles them.

Floating-point exceptions are errors that can occur during floating-point calculations. The FPU recognizes and detects six types of floating-point exceptions:

Invalid operation (#I): This exception is raised when an invalid operation is attempted, such as dividing zero by zero or taking the square root of a negative number.

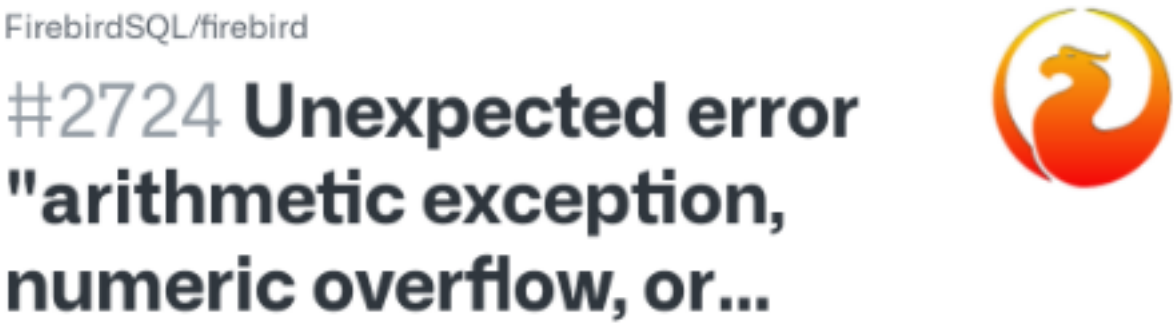


Divide by zero (#Z): This exception is raised when an attempt is made to divide by zero.



Denormalized operand (#D): This exception is raised when an attempt is made to operate on a denormalized number. A denormalized number is a floating-point number that is very close to zero.

Numeric overflow (#O): This exception is raised when the result of a floating-point operation is too large to be represented by the FPU.



Numeric underflow (#U): This exception is raised when the result of a floating-point operation is too small to be represented by the FPU.

Inexact precision (#P): This exception is raised when the result of a floating-point operation is not exact. This can happen because

floating-point numbers are represented by a finite number of bits.

Each exception type has a corresponding flag bit and mask bit. When a floating-point exception is detected, the processor sets the matching flag bit.

The mask bit controls whether or not the processor automatically handles the exception. If the mask bit is set, the processor automatically handles the exception and lets the program continue. If the mask bit is clear, the processor invokes a software exception handler.

The processor's masked (automatic) responses are generally acceptable for most programs. However, custom exception handlers can be used in cases where specific responses are required by the application.

A single instruction can trigger multiple exceptions. The processor keeps an ongoing record of all exceptions occurring since the last time exceptions were cleared. After a sequence of calculations completes, you can check to see if any exceptions occurred.

Here are some examples of how floating-point exceptions can occur:

• Dividing a number by zero will raise a divide by zero exception.
• Taking the square root of a negative number will raise an invalid operation exception.
• Adding two very large numbers together may raise a numeric overflow exception.
• Subtracting two very small numbers from each other may raise a numeric underflow exception.
• Multiplying two numbers together may raise an inexact precision exception if the result is too large to be represented by the FPU.

Floating-point exceptions can cause unexpected behavior in programs.

It is important to be aware of the different types of floating-point exceptions and how to handle them.

You can use the FPU's mask bits to control how the processor handles floating-point exceptions. You can also write custom exception handlers to handle floating-point exceptions in a specific way.