This chapter describes the parts of the floating-point environment that you can control. The IEEE standard specifies that users should be able to control the rounding direction, floating-point exceptions, and in some instances the rounding precision. PowerPC Numerics implementations provide utilities (called **environmental controls**) with which you can set, clear, and test the rounding direction and floating-point exception flags. (See Parts 2 and 3 for the exact names of functions and instructions that control the floating-point environment.) This chapter describes the four rounding direction modes and the five floating-point exception flags that you can set, clear, and test in PowerPC Numerics. You should read it to learn more about the floating-point environment.

Rounding Direction Modes

The available **rounding direction modes** are

- to nearest
- upward (toward +∞)
- downward (toward −∞)
- toward zero

The rounding direction affects all conversions, except conversions between decimal structures and decimal strings (described in Chapter 5, “Conversions”), and all arithmetic operations except remainder. All operations are calculated without regard to the range and precision of the data type in which the result is to be stored. That is, an operation first produces a result that is infinitely precise, or exact. If the destination data type cannot represent this number exactly, the result is rounded in the direction specified by the rounding mode.

The default rounding direction is to nearest. In this mode, floating-point expressions deliver the value nearest to the exact result that the destination data type can represent. If two representable values are equally close to the exact result, the expression delivers the one whose least significant bit is zero. Hence, halfway cases (for example, 1.5) round to even when the destination is an integer type or when the round-to-integer operation is used. If the magnitude of the exact result is greater than the data type’s largest value (by at least one half unit in the last place), then the Infinity with the corresponding sign is delivered.

The other rounding directions are upward, downward, and toward zero. When rounding upward, the result is the representable value (possibly +∞) closest to, and not less than, the exact result. When rounding downward, the result is the representable value (possibly −∞) closest to, and not greater than, the exact result. When rounding toward zero, the result is the representable value closest to, and not greater in magnitude than, the exact result. Toward-zero rounding truncates a number to an integer (when the destination is an integer type). Table 4-1 shows some values rounded to integers using different rounding modes.

Table 4-1 Examples of rounding to integer in different directions

Floating-point number	Rounded to nearest	Rounded toward 0	Rounded downward	Rounded upward
1.5	2	1	1	2
2.5	2	2	2	3
–2.2	–2	–2	–3	–2

Rounding Precision

A rounding precision mode specifies a precision to which all numeric operation results are rounded. For example, if the rounding precision mode were set to single, the results of all operations would be rounded to single precision until the rounding precision mode changed.

The IEEE standard requires rounding precision modes only on systems that always deliver results to double or extended format destinations. With a rounding precision mode set to a narrower format, such systems round to the precision of that format regardless of the destination. Thus, they can use rounding precision modes to emulate other systems that deliver results in narrower formats.

Because PowerPC Numerics delivers results in any of its three data formats, it does not support dynamic rounding precision modes. Instead, a PowerPC Numerics implementation may support static narrowing of rounding precision at translation time through pragmas, compiler options, or narrower expression evaluation methods.

Exception Flags

Floating-point exceptions are signaled with **exception flags.** When an application begins, all floating-point exception flags are cleared and the default rounding direction (to nearest) is in effect. This is the **default environment.** When an exception occurs, the appropriate exception flag is set, but the application continues normal operation. Floating-point exception flags merely indicate that a particular event has occurred; they do not change the flow of control for the application. An application can examine or set individual exception flags and can save and retrieve the entire environment (rounding direction and exception flags).

Note
The Exception Manager, described in the book *Inside Macintosh: PowerPC System Software*, does not report floating-point exceptions in the first version of the system software for PowerPC processor-based Macintosh computers. ◆

The numerics environment supports five exception flags:

- invalid operation (often called simply *invalid*)
- underflow
- overflow
- divide-by-zero
- inexact

These are discussed in the paragraphs that follow.

Invalid Operation

The **invalid exception** (or invalid-operation exception) occurs if an operand is invalid for the operation being performed. The result is a quiet NaN for all destination formats (single, double, or double-double). The invalid conditions for the different operations are

Operation	Invalid condition
Addition or subtraction	Magnitude subtraction of Infinities, for example, $(+\infty) + (-\infty)$
Multiplication	$0 \times \infty$
Division	$0/0$ or ∞/∞
Remainder	x rem y, where y is 0 or x is infinite
Square root	A negative operand
Conversion	See Chapter 5, “Conversions”
Comparison	With predicates involving less than or greater than, but not unordered, when at least one operand is a NaN

In addition, any operation on a signaling NaN except the class and sign inquiries and, on some implementations, sign manipulations (absolute value and copysign) produce an invalid exception.

Underflow

The **underflow exception** occurs when a floating-point result is both tiny and inexact (and therefore is perhaps significantly less accurate than if there were no limit to the exponent range). A result is considered **tiny** if it must be represented as a denormalized number.

Overflow

The **overflow exception** occurs when the magnitude of a rounded floating-point result is greater than the largest finite number that the floating-point destination data format can represent. (Invalid exceptions, rather than overflow exceptions, flag the production of an out-of-range value for an integer destination type.)

Divide-by-Zero

The **divide-by-zero exception** occurs when a finite, nonzero number is divided by zero. It also occurs, in the more general case, when an operation on finite operands produces an exact infinite result; for example, `logb(0)` returns $-\infty$ and signals divide-by-zero. (Overflow exceptions, rather than divide-by-zero exceptions, flag the production of an inexact infinite result.)

Inexact

The **inexact exception** occurs if the rounded result of an operation is not identical to the exact (infinitely precise) result. Thus, an inexact exception always occurs when an overflow or underflow occurs. Valid operations on Infinities are always exact and therefore signal no exceptions. Invalid operations on Infinities are described at the beginning of this section.