C NUMERIC FORMATS

The processor supports several numeric formats:

- IEEE Standard 754/854, 32-bit, single-precision floating-point format.
- An extended-precision version of the 32-bit, single-precision floating-point format that has eight additional bits in the mantissa (40 bits total).
- 32-bit, fixed-point formats that include both fractions and integers in signed (twos-complement) or unsigned formats.

Single-Precision Floating-Point Format

IEEE Standard 754/854 specifies a 32-bit, single-precision floating-point format, as shown in Figure C-1. A number in this format consists of a sign bit `s`, a 24-bit significant, and an 8-bit unsigned-magnitude exponent `e`.

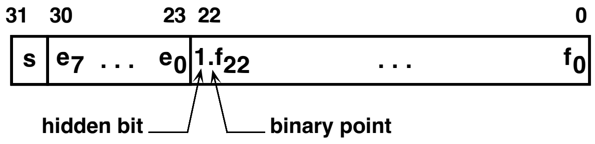

Figure C-1. IEEE 32-bit single-precision floating-point format

For normalized numbers, the significant consists of a 23-bit fraction `f` and a *hidden* bit `1` understood to precede `f22` in the significant. The binary point is understood to lie between the hidden bit and `f22`. The least significant bit (LSB) of the fraction is `f0`. The LSB of the exponent is `e0`.

The hidden bit effectively increases the precision of the floating-point significant to twenty-four bits from the twenty-three bits actually stored in the data format. It also ensures that the significant of any IEEE normalized number is always ≥1 and <2.

In the single-precision format, the unsigned exponent `e` ranges between $1 \leq e \leq 254$ for normal numbers. This exponent is biased by +127 (254 ÷ 2). To calculate the true unbiased exponent, you subtract 127 from `e`.

The IEEE standard also provides for several special data types in the single-precision floating-point format, as shown in Table C-1.

Table C-1. Supported single-precision, floating-point special data types

Type	Exponent	Fraction	Value	Notes
Infinity	255	0	$(-1)^s$ Infinity	Because the fraction is signed, can represent ±Infinity
NAN (Not-A-Number)	255 (all 1s)	nonzero	undefined	Typical uses are: Flags for data flow control Values of uninitialized variables Results of invalid operations (as 0 * ∞)
Normal	$1 \le e \le 254$	any	$(-1)^s\ (1.f_{22-0})2^{e-127}$	
Zero	0	0	$(-1)^s$ Zero	Represents ±Zero

Extended-Precision Floating-Point Format

The extended-precision floating-point format, as shown in Figure C-2, is the same as the single-precision format, except this format:

- Is forty bits wide
- Has a 32-bit significant

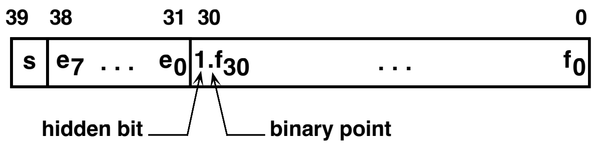

Figure C-2. 40-bit extended floating-point format

Short Word Floating-Point Format

The processor supports a 16-bit, floating-point data type and provides conversion instructions for it.

This format has an 11-bit mantissa, a 4-bit exponent, and a sign bit, as shown in Figure C-3. The 16-bit floating-point numbers reside in the lower sixteen bits of the 32-bit floating-point field.

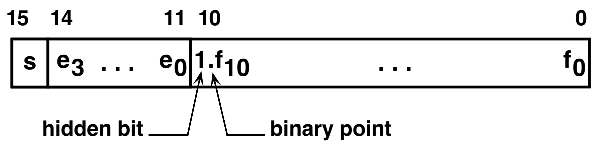

Figure C-3. 16-bit floating-point format

Two shifter instructions, FPACK and FUNPACK, perform the packing and unpacking conversions between 32- and 16-bit floating-point words.

The FPACK instruction converts a 32-bit IEEE floating-point number to a 16-bit floating-point number.

The FUNPACK instruction converts a 16-bit floating-point number to a 32-bit IEEE floating-point number.

Each instruction executes in a single cycle. Table C-2 lists and describes the results of the FPACK and FUNPACK operations.

Table C-2. Results of the FPACK and FUNPACK operations

Operation	Condition	Result
FPACK	135 < exp	Largest magnitude representation
	120 < exp ≤ 135	Exponent is MSB of source exponent concatenated with the three LSBs of source exponent. The packed fraction is the rounded upper 11 bits of the source fraction.
	109 < exp ≤ 120	Exponent=0. Packed fraction is the upper bits (source exponent -110) of the source fraction prefixed by zeros and the "hidden" 1. The packed fraction is rounded.
	exp < 110	Packed word is all zeros (*0*s).
exp = source exponent; sign bit remains the same in all cases		

Table C-2. Results of the FPACK and FUNPACK operations (Cont'd)

Operation	Condition	Result
FUNPACK	0 < exp < 15	Exponent is the 3 LSBs of the source exponent prefixed by the MSB of the source exponent and four copies of the complement of the MSB. The unpacked fraction is the source fraction with 12 zeros appended.
	exp = 0	Exponent is (120 – N), where N is the number of leading zeros in the source fraction. The unpacked fraction is the remainder of the source fraction with zeros (*0*s) appended to pad it and the *hidden* 1 stripped away.
exp = source exponent; sign bit remains the same in all cases		

The short float type supports gradual underflow, which sacrifices precision for dynamic range. When packing a number that would have underflowed, the processor sets the exponent to zero (0) and right shifts the mantissa (including the hidden 1) the appropriate amount. The packed result is a denormal, which you can unpack into a normal IEEE floating-point number.

During the FPACK operation, an overflow condition sets the SV flag, and a nonoverflow condition clears it. During the FUNPACK operation, the Shifter clears the SV flag. For both instructions, the Shifter clears the SZ and SS flags. For details, see Chapter 2, Computation Units, in *ADSP-21065L SHARC DSP User's Manual*.

Fixed-Point Formats

The processor supports two 32-bit fixed-point formats—fractional and integer—both of which include signed (twos-complement) and unsigned numbers. Figure C-4 shows the four possible combinations.

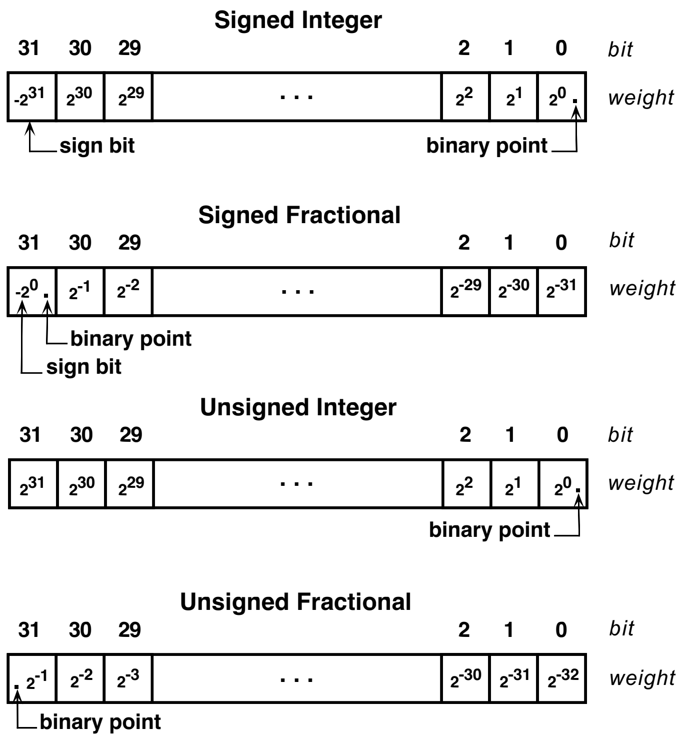
Signed Integer
31 30 29 2 1 0 bit
-2^{31} 2^{30} 2^{29} . . . 2^{2} 2^{1} 2^{0} weight
sign bit
binary point
Signed Fractional
31 30 29 2 1 0 bit
-2^{0} 2^{-1} 2^{-2} . . . 2^{-29} 2^{-30} 2^{-31} weight
binary point
sign bit
Unsigned Integer
31 30 29 2 1 0 bit
2^{31} 2^{30} 2^{29} . . . 2^{2} 2^{1} 2^{0} weight
binary point
Unsigned Fractional
31 30 29 2 1 0 bit
2^{-1} 2^{-2} 2^{-3} . . . 2^{-30} 2^{-31} 2^{-32} weight
binary point

Figure C-4. 32-bit fixed-point formats

The fractional format includes an implied binary point to the left of the most significant magnitude bit. The integer format includes an implied binary point to the right of the LSB. In signed (twos-complement) format, the sign bit is negatively weighted.

ALU outputs always have the same width and data format as the inputs.

The Multiplier, however, produces a 64-bit product from two 32-bit inputs. Multiplier results follow these rules:

- If both operands are unsigned integers, the result is a 64-bit unsigned integer.
- If both operands are unsigned fractions, the result is a 64-bit unsigned fraction.

  Figure C-5 shows both of these results.

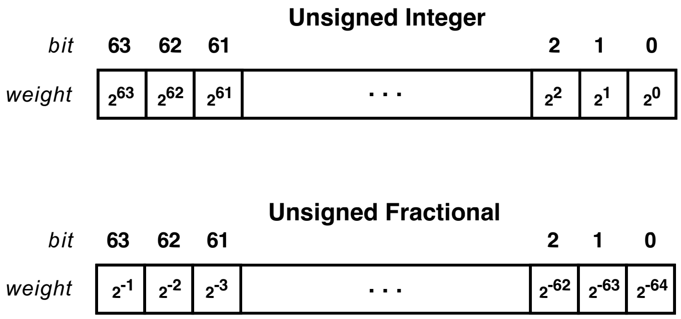

Figure C-5. 64-bit unsigned fixed-point products

- If one operand is signed and the other is unsigned, the result is signed.
- If both inputs are signed, the result is signed and automatically shifted left one bit.

  The LSB becomes zero (0) and bit 62 moves into the sign bit position.

  Normally bit 63 and bit 62 are identical when both operands are signed. (The only exception occurs when a full-scale negative is multiplied by itself.) So, the left shift normally removes a redundant sign bit, increasing the precision of the most significant product (MSP).

If the data format is fractional, a single-bit left shift renormalizes the MSP to a fractional format. Figure C-6 shows the signed formats with and without left shifting.

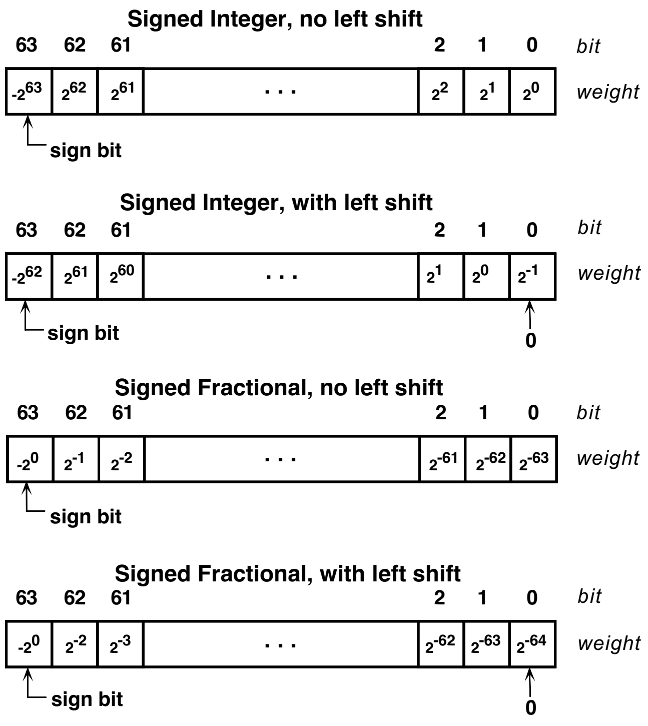
Signed Integer, no left shift
63 62 61 2 1 0 bit
-2^63 2^62 2^61 . . . 2^2 2^1 2^0 weight
sign bit
Signed Integer, with left shift
63 62 61 2 1 0 bit
-2^62 2^61 2^60 . . . 2^1 2^0 2^-1 weight
sign bit
0
Signed Fractional, no left shift
63 62 61 2 1 0 bit
-2^0 2^-1 2^-2 . . . 2^-61 2^-62 2^-63 weight
sign bit
Signed Fractional, with left shift
63 62 61 2 1 0 bit
-2^0 2^-2 2^-3 . . . 2^-62 2^-63 2^-64 weight
sign bit
0

Figure C-6. 64-bit signed, fixed-point product

The Multiplier has an 80-bit accumulator for accumulating 64-bit products. For details, see Chapter 2, Computation Units, in *ADSP-21065L SHARC DSP User's Manual.*