



Decimal Floating Point

Ian McIntosh
February 6, 2008

www.ibm.com

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

What is Floating Point?

- **Floating Point values have a fraction, exponent and a sign. 42 is an integer value; 0.42e2 is a floating point value.**
- **There have been many floating point formats. In early days almost every computer manufacturer had one. Or two, or more. Compatibility was rare.**
- **Some formats used binary, some decimal, some hexadecimal. The precision and exponent range varied.**
- **In recent years, the IEEE 754 Binary Floating Point formats have been dominant, but others still exist.**
- **Over the past five years, IEEE / IBM / Intel etc. have developed new formats called IEEE 754 Decimal Floating Point.**

IEEE = Institute of Electrical and Electronics Engineers

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

First, what is Binary Floating Point?

- **Binary Floating Point values have a binary (base 2) fraction, an exponent which scales binary digits (bits), and a sign.**
- **The bits in the binary fraction represent binary digits (powers of 2): 1, 2, 4, 8, 16 . . . and 1/2, 1/4, 1/8, 1/16, . . .**
- **Each format has a maximum precision, in bits.**
- **Integers below some limit can be exactly represented; eg, 12345.**
- **Some integers above that limit can only be approximated; eg, 1234567890123456789012345678901234567890.**
- **Others are exact; eg, 2^{50} . If x is exact, $x * 2^n$ is also exact.**
- **Some fractions can be exactly represented; eg, 3/16.**
- **Other fractions can only be approximated; eg, 1/3 or 1/10.**
- **C/C++/Java `float`, `double` and `long double` usually use BFP.**

What is Decimal Floating Point?

- **Decimal Floating Point values have a decimal (base 10) fraction, an exponent which scales decimal digits, and a sign.**
- **The digits in the decimal fraction represent decimal digits.**
- **Each format has a maximum precision, in decimal digits.**
- **Integers below some limit can be exactly represented; eg, 12345.**
- **Some integers above that limit can only be approximated; eg, 1234567890123456789012345678901234567890.**
- **Others are exact; eg, 10^{50} . If x is exact, $x \cdot 10^n$ is also exact.**
- **Some fractions can be exactly represented; eg, $1/10$.**
- **Other fractions can only be approximated; eg, $1/3$.**
- **Most languages (C/C++, etc.) rarely or never use DFP. Some like C#, REXX and the Java `BigDecimal` class do.**

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

What is IEEE Decimal Floating Point?

- **Some old computer designs had Decimal Floating Point hardware. New designs generally don't.**
- **Some applications need DFP. Examples include calculators and cash registers.**
- **Generally DFP is implemented via a subroutine library, so is hundreds of times slower than Binary Floating Point hardware. In some applications that's fast enough.**
- **Some computer designs have instructions to speed DFP up.**
- **Sometimes scaled integers are used instead, sometimes scaled decimal fixed point software or hardware.**
- **Over the past five years, IEEE / IBM / Intel etc. have developed new formats called IEEE Decimal Floating Point, to have a *standardized* approach to DFP.**

What is IEEE Decimal Floating Point?

There are two sets ☹ of IEEE Decimal Floating Point formats:

- **Densely Packed Decimal (DPD):**
 - designed by IBM.
 - optimized for a hardware implementation.
- **Binary Integer Decimal (BID):**
 - designed by Intel.
 - optimized for a software implementation.

Both give you 7, 15 or 34 digit precisions.

Exponent ranges, sizes, and operations on them are identical. They are equivalent but internally different, *not to be intermixed without conversion.*

What is IEEE Decimal Floating Point?

Type Name	Precision	Exponent Range	Size
decimal32 (single precision)	7 digits	-101 to +90	4 bytes (32 bits)
decimal64 (double precision)	16 digits	-398 to +369	8 bytes (64 bits)
decimal128 (quadruple precision)	34 digits	-6176 to +6111	16 bytes (128 bits)

What is IEEE Decimal Floating Point?

In addition to the existing BFP rounding modes:

- Round to nearest, ties to even (the default).**
- Round toward zero (used by casts to int).**
- Round towards +Infinity.**
- Round towards –Infinity.**

IEEE DFP adds:

- Round towards nearest, ties away from zero (the most intuitive).**



IBM adds:

- Round towards nearest, ties toward zero.**
- Round away from zero.**
- Round to prepare for Reround (used for special precisions).**

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

When does DFP or BFP not matter?

- Neither BFP nor DFP is inherently more accurate than the other. It depends on the value.
- BFP can exactly represent very large or very small powers of 2 exactly. DFP cannot.
- DFP can exactly represent very large or very small powers of 10 exactly. BFP cannot.
- Neither can represent values like $1/3$, $1/17$ or π exactly.
- Both IEEE BFP and IEEE DFP have special values for +/-Infinity or for “Not a Number”.
(NaN is the result of $0/0$ or  - .)

Why avoid Decimal Floating Point?

- **It's still new and different (but it's easier and faster than decimal library alternatives).**
- **Not all languages include it yet (but it's coming to some).**
- **Not many compilers support it yet, so portability is limited (but IBM AIX XL C / XL C++ and gcc support it now).**
- **Tools and library support will take time (but IBM AIX debuggers and gdb support DFP types now).**
- **IBM Power6 hardware DFP is not as fast as BFP, because there's only one non-pipelined functional unit (but when accurate decimal results are needed it is much faster than decimal libraries).**
- **Software emulation DFP is much slower (about as slow as decimal libraries).**

Why use Decimal Floating Point?

- As long as all of your values can be represented exactly, binary versus decimal doesn't matter.
- When your values must be approximated, they will be rounded up or down. Then it may matter.
- Since BFP can't exactly represent $1/10$, $1/100$, $1/1000$ etc, the rounded values can be confusing.
- Since DFP can exactly represent $1/10$, $1/100$, $1/1000$ etc, the rounded values are what you would expect.
- Binary Floating Point can only approximate many values important to human calculations, that Decimal Floating Point can represent exactly. Obvious examples are multiples of $1/10$, $1/100$, $1/1000$, etc. In BFP, $1/10$ is 0.099999999, not 0.1.

Why use Decimal Floating Point?

- **Unlike Binary Floating Point and older DFP formats, IEEE Decimal Floating Point automatically keeps track of the number of significant digits in every calculation.**
 - **You can ask how many significant digits a value has.**
 - **You can print the significant digits without knowing how many there are.**
 - **You can round to fewer significant digits.**
 - **You can also adjust one number to have the same “quantization” (the same least significant digit position) as another; eg, “quantize” 1234.56 to match 98.765, giving 1234.560.**
 - **Keeping track of significance can be useful to scientific and engineering calculations, not just financial.**

Why use Decimal Floating Point?

- **Some languages / environments require some form of DFP.**
- **IEEE DFP is the new standard, with IEEE, C and C++ standards in the approval process and Fortran underway.**
- **Some applications require rounding to decimal digits.**
- **IEEE DFP also has a new more obvious rounding mode.**
- **Math libraries can be more consistent.**
- **Your program's calculations and output will make more sense to you and their users.**
- ***If you need accurate decimal digits, you need some form of DFP.***

Why use Decimal Floating Point?

Consider a sales tax calculation, that legally must be right:

```
double price = 0.70;  
double tax_rate = 0.05;  
double tax = price * tax_rate;  
double total = price * (1. + tax_rate);  
printf ("%2f + %2f = %2f\n", price, tax, total);
```

0.70 * 1.05 = 0.735, which should round up to 0.74,

**but in BFP it is 0.73499999999999999866, which rounds
down to 0.73, giving “.70 + .04 = .73”.**

When 3 cents tax is collected instead of 4 cents, who pays?

The programmer? The business? The government? You?

Why use Decimal Floating Point?

- **Excel appears to use DFP for some values and BFP for others, or maybe some format(s) of their own.**

Try 0.1, and try =1.95-1.85.

The right answers are that both are exactly 0.1.

What do you get?

What do you get if you set the cell format to Scientific?

What if you set the precision to 14 digits? To more?

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

What are the standards?

- **The IEEE 754 standard is being revised (currently known as 754R). The revision adds new functionality and new BFP types as well as DFP.**
- **The next C standard will incorporate ISO/ANSI C Technical Report N1154 (close to being approved) which adds new DFP types to C.
IBM has implemented this in both XL C and in GNU gcc C.**
- **The next C++ standard will incorporate ISO/ANSI C++ TR24733 (also close to approval), which adds new DFP types and classes.
IBM xLC has implemented this in XL C++.**

What are the standards?

- **The IBM PowerPC Architecture now includes DFP (DFP). The pSeries Power6 (running AIX and Linux) has DFP instructions.**
- **Future z/Architecture mainframes may include DFP.**
- **No other hardware support has been announced.**

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

How do you use DFP in C?

Overview:

- **Since DFP isn't officially part of C yet, you may need a compiler option to enable it.**
- **There are 3 new built in types.**
- **There are 3 new literal suffixes.**
- **The standard floating point operators work.**
- **There are new printf / scanf format codes.**
- **There are new library functions.**

How do you use DFP in C++?

Overview:

- **Since DFP isn't officially part of C++ yet, you may need a compiler option to enable it.**
- **There are 3 new built in types and 3 new library classes.**
- **There are 3 new literal suffixes.**
- **The standard floating point operators work.**
- **There are new `cin>>` and `cout<<` operators.**
- **There are new library functions.**

How do you use DFP in IBM xlc / xlc?

- **Since DFP isn't officially part of C++ yet, you need the “-qdfp” compiler option to enable recognition.**
- **The AIX / Linux Power6 has hardware DFP, so `-qarch=pwr6` generates hardware instructions. Other `-qarch` options use software library calls.**
- **There is a new compile-time rounding mode parameter: `-yd_` like the binary `-y_`.**
- **You need the latest AIX operating system. On Linux, DFP is a technology preview without operating system support, so you also need to download several libraries.**

How do you use DFP in IBM xlc / xlC?

- In future C and the latest IBM C / C++ there are 3 new built in types:
 - `_Decimal32` - single precision – 7 digits
 - `_Decimal64` - double precision – 16 digits
 - `_Decimal128` - quad precision – 34 digits
- In future C++ and in IBM alphaworks there are 3 new classes:
 - `decimal32`
 - `decimal64`
 - `decimal128`
- In the future C++ standard the built in types are optional.

How do you use DFP in IBM xlc / xlc?

- In future C and IBM C / C++ there are 3 new literal suffixes:
 - df or DF – single precision
 - dd or DD – double precision
 - dl or DL – quad precisioneg, 12.34dd
- In IBM C / C++ these are built in.

In the future C++ standard the built in literals will be optional, but the new “User Defined Literals” feature will let them be defined in the classes.

How do you use DFP in IBM xlc / x1C?

- There are new printf / scanf format codes:
 - The printf / scanf / etc functions use a modifier **H**, **D** or **DD** with the existing **%e / %E**, **%f / %F** and **%g / %G**; eg, “%**D**7.2f”.
- In C++ there are new **cin>>** and **cout<<** operators.
- There are new library functions, mostly with suffixes **d32**, **d64** and **d128**; eg, **sqrtd64** or **sind128**.
- There are new library functions to set / get the decimal rounding mode (which is separate from the BFP one).
- IBM also supplies new built-in functions for all the DFP hardware instructions otherwise unavailable (mostly for those writing math library functions).

How do you use DFP in IBM xlc / xlcC?

```
int main (void) {
    _Decimal128 value;
    _Decimal64 rate;
    int periods;
    int period;
    printf ("Enter initial value, interest rate and periods:\n");
    scanf ("%DDg %Dg %d\n", &value, &rate, &periods);
    for (period = 1; period <= periods; ++period) {
        value = value * (1.dd + rate);
        printf ("after period %d value is %.2DDf\n", period, value);
    }
}
```

```
xlc interest.c -qdfp
```

What if you need non-xlc/xlC and non-gcc portability?

- The **decNumber** library is available via download from IBM's Hursley lab or www.alphaworks.com. It is also supplied in the latest AIX operating system.
- The **DFPAL** library is available via download from www.alphaworks.com. It is easier to use than decNumber, and much faster on Power6 because it uses the Power6 DFP instructions instead of decNumber.
- The C++ **decNumber++** class library is available via download from www.alphaworks.com.

IEEE / C / C++ Decimal Floating Point Summary

Type Name Class Name	Precision	Exponent Range	Size	Literal Suffix	Format Modifier	Function Suffix
<code>_Decimal32</code> <code>decimal32</code>	7 digits	-101 to +90	4 bytes (32 bits)	<code>df / DF</code>	<code>HD</code>	<code>d32</code>
<code>_Decimal64</code> <code>decimal64</code>	16 digits	-398 to +369	8 bytes (64 bits)	<code>dd / DD</code>	<code>D</code>	<code>d64</code>
<code>_Decimal128</code> <code>decimal128</code>	34 digits	-6176 to +6111	16 bytes (128 bits)	<code>dl / DL</code>	<code>DD</code>	<code>d128</code>

IEEE Decimal Floating Point Summary

- **In the early days of computing, many computers used decimal integer arithmetic, scaled decimal fixed point, and decimal floating point (not the IEEE format !).**
- **Manufacturers moved to binary (and hexadecimal) to simplify circuitry and especially for speed.**
- **IEEE Binary Floating Point offered speed, portability and extra features like NaNs and Infinities.**
- **Now IEEE Decimal Floating Point trades a little of that circuit size and speed for better usability.**

Discussion and Questions

For the really curious . . .

Outline

- **What is Floating Point?**
- **What is Decimal Floating Point?**
- **What is IEEE Decimal Floating Point?**
- **Why use DFP? Why not?**
- **What are the DFP standards?**
- **How do you use DFP?**
- **What is Decimal Floating Point, really?**

What is IEEE Decimal Floating Point, Really?

There are two sets ☹ of IEEE Decimal Floating Point formats:

- **Densely Packed Decimal (DPD):**
 - designed by IBM.
 - optimized for a hardware implementation.
- **Binary Integer Decimal (BID):**
 - designed by Intel.
 - optimized for a software implementation.

Precisions, exponent ranges, sizes, and operations on them are identical. They are equivalent but internally different, *not to be intermixed without conversion.*

Here only DPD will be described in detail.

What is IEEE Decimal Floating Point, Really?

The Decimal Floating Point format has:

- a sign -- 1 bit, 0=+ve, 1=-ve.
- a combination field -- 5 bits, containing 2 bits of the exponent and in DPD the leftmost digit.
- an exponent continuation field -- 6 / 8 / 12 bits, containing the rest of the exponent.
- a fraction continuation field -- 20 / 50 / 110 bits
 - In DPD: divided into 2 or 5 or 11 Densely Packed Decimal 3 digit blocks, containing the rest of the digits (6 or 15 or 33).
 - In BID: a binary integer.

What is IEEE Decimal Floating Point, Really?

a



What is IEEE Decimal Floating Point, Really?

The 5 bit Combination Field:

- **11111 = NaN**

first exponent bit 0=Quiet, 1=Signaling

- **11110 = Infinity (+ve or -ve)**

- **Other = 2 exponent bits + 1 decimal digit**

00### / 01### / 10 ### =

– leftmost exponent bits = 00 / 01 / 10

– leftmost decimal digit = ### (0 to 7)

- 11xx0 = digit 8, exponent bits = xx (00 / 01 / 10)
- 11xx1 = digit 9, exponent bits = xx (00 / 01 / 10)

What is IEEE Decimal Floating Point, Really?

The Exponent Continuation Field:

- 6 / 8 / 12 bits, depending on precision.
- Concatenated to first 2 bits from Combination Field.
- Exponent is biased integer.
- Exponent counts decimal digits.
- Exponent ranges are slightly asymmetrical:
 - single precision -101 to +90 (bias 101)
 - double precision -398 to +369 (bias 398)
 - quad precision -6176 to +6111 (bias 6176)

What is IEEE Decimal Floating Point, Really?

The Fraction Continuation Field:

- **20 / 50 / 110 bits divided into 2 or 5 or 11
Densely Packed Decimal 3 digit (10 bit) blocks
(6 or 15 or 33 digits).**
- **Concatenated to first digit from Combination Field.**

What is IEEE Decimal Floating Point, Really?

DPD Densely Packed Decimal:

- **Each 10 bit block holds 0-1023, representing 3 digits.**
- **There are 24 redundant values.**
- **The format is complicated but can be expanded to 12 bits (3 4 bit digits) quickly, and 3 4 bit digits can be compressed to a 10 bit block quickly ($< \frac{1}{2}$ pipeline stage each at 4.7 MHz) with minimal circuitry.**

What is IEEE Decimal Floating Point, Really?

How to pack 12 bits into 10, using Densely Packed Decimal:

- **For 12 bits (3 digits x 4 bits each):**
- **If no digits are 8 or 9, simply store 3 bits of each.**
- **If one digit is 8 or 9, store which one, and whether 8 or 9.**
- **If two digits are 8 or 9, store which two, and which are 8 or 9.**
- **If all 3 digits are 8 or 9, store which are 8 and which are 9.**
- **Also store which of those four cases is true, using flag bits.**
- **This can be done in 10 bits, not 12.**

What is IEEE Decimal Floating Point, Really?

Densely Packed Decimal:

- For 12 bits (3 digits) ABCD EFGH IJKL:
 - If no digits are 8 or 9, simply store 3 bits of each.

A==0 & E==0 & I==0 (no 8/9) => DPD: BCD FGH 0 JKL

What is IEEE Decimal Floating Point, Really?

Densely Packed Decimal:

- For 12 bits (3 digits) ABCD EFGH IJKL:
 - If no digits are 8 or 9, simply store 3 bits of each.
 - If one digit is 8 or 9, store which one, and whether 8 or 9.

A==0 & E==0 & I==0 (no 8/9) => DPD: BCD FGH 0 JKL

A==0 & E==0 & I==1 (3rd 8/9) => DPD: BCD FGH 1 00L

A==0 & E==1 & I==0 (2nd 8/9) => DPD: BCD JKH 1 01L

A==1 & E==0 & I==0 (1st 8/9) => DPD: JKD FGH 1 10L

What is IEEE Decimal Floating Point, Really?

Densely Packed Decimal:

- For 12 bits (3 digits) ABCD EFGH IJKL:
 - If no digits are 8 or 9, simply store 3 bits of each.
 - If one digit is 8 or 9, store which one, and whether 8 or 9.
 - If two digits are 8 or 9, store which two, and which are 8 or 9.

A==0 & E==0 & I==0	(no 8/9)	=>	DPD:	BCD	FGH	0	JKL
A==0 & E==0 & I==1	(3rd 8/9)	=>	DPD:	BCD	FGH	1	00L
A==0 & E==1 & I==0	(2nd 8/9)	=>	DPD:	BCD	JKH	1	01L
A==1 & E==0 & I==0	(1st 8/9)	=>	DPD:	JKD	FGH	1	10L
A==1 & E==1 & I==0	(1&2 8/9)	=>	DPD:	JKD	00H	1	11L
A==1 & E==0 & I==1	(1&3 8/9)	=>	DPD:	FGD	01H	1	11L
A==0 & E==1 & I==1	(2&3 8/9)	=>	DPD:	BCD	10H	1	11L

What is IEEE Decimal Floating Point, Really?

Densely Packed Decimal:

- For 12 bits (3 digits) ABCD EFGH IJKL:
 - If no digits are 8 or 9, simply store 3 bits of each.
 - If one digit is 8 or 9, store which one, and whether 8 or 9.
 - If two digits are 8 or 9, store which two, and which are 8 or 9.
 - If all 3 digits are 8 or 9, store which are 8 and which are 9.

A==0 & E==0 & I==0 (no 8/9) => DPD: BCD FGH 0 JKL

A==0 & E==0 & I==1 (3rd 8/9) => DPD: BCD FGH 1 00L

A==0 & E==1 & I==0 (2nd 8/9) => DPD: BCD JKH 1 01L

A==1 & E==0 & I==0 (1st 8/9) => DPD: JKD FGH 1 10L

A==1 & E==1 & I==0 (1&2 8/9) => DPD: JKD 00H 1 11L

A==1 & E==0 & I==1 (1&3 8/9) => DPD: FGD 01H 1 11L

A==0 & E==1 & I==1 (2&3 8/9) => DPD: BCD 10H 1 11L

A==1 & E==1 & I==1 (all 8/9) => DPD: ~~xxD~~ 11H 1 11L

What is IEEE Decimal Floating Point, Really?

Densely Packed Decimal:

- For 12 bits (3 digits) ABCD EFGH IJKL:
 - If no digits are 8 or 9, simply store 3 bits of each.
 - If one digit is 8 or 9, store which one, and whether 8 or 9.
 - If two digits are 8 or 9, store which two, and which are 8 or 9.
 - If all 3 digits are 8 or 9, store which are 8 and which are 9.

A==0 & E==0 & I==0 (no 8/9) => DPD: BCD FGH 0 JKL

A==0 & E==0 & I==1 (3rd 8/9) => DPD: BCD FGH 1 00L

A==0 & E==1 & I==0 (2nd 8/9) => DPD: BCD JKH 1 01L

A==1 & E==0 & I==0 (1st 8/9) => DPD: JKD FGH 1 10L

A==1 & E==1 & I==0 (1&2 8/9) => DPD: JKD 00H 1 11L

A==1 & E==0 & I==1 (1&3 8/9) => DPD: FGD 01H 1 11L

A==0 & E==1 & I==1 (2&3 8/9) => DPD: BCD 10H 1 11L

A==1 & E==1 & I==1 (all 8/9) => DPD: xxD 11H 1 11L

Discussion and Questions