Subject: Re: DOUBLE precision no precise??
Posted by William Clodius on Tue, 05 Mar 2002 16:11:54 GMT
View Forum Message <> Reply to Message

David Williams wrote:<snip>

```
>  To try and find where the problem is, we tried the following lines...
>
>  IDL> a = DOUBLE(42766.080001)
>  IDL> print,a,FORMAT='(F24.17)'
>
>   42766.07812500000000000
>
>  As you see, the number we get out isn't the same as the number we
>  entered. I'm guessing it's to do with the way IDL stores numbers in
>  memory, but my understanding of low-level computational processes isn't
>  great.
>
>  Can anybody help me understand what's going on, and/or if there's a way
>  around? I'd really appreciate whatever help is on offer, so thanks in
>  advance.
>
```

<snip>

All computers IDL is available for store numbers in memory using a binary representation. This representation comes in at least two forms, single (float) and double precision. Both representations can be thought of as typically representing a number by an integer multiplied by a scale factor (exponent) that is an integer power of two. Double uses twice as many bits as float to allow a larger range of integers and scale factors. Because of the finite range of the integers, and because the exponent is a power of two and not a power of ten, only an infinitesimal fraction of the numbers that can be written exactly in decimal can be represented exactly in a finite binary representation. This is a common source of confusion for users of most programming languages. (There are some languages that use less efficient representation such as decimal or rational arithmetic, but such languages, in addition to their inefficiencies, often provide only the simplest mathematical operations.)

In addition to this common source of confusion, your code has an additional problem that is almost as common among such languages. You apparently don't understand the lexical conventions used to distinguish between literals that represent single and double precision numbers. IDL iginores your DOUBLE in deciding this. Instead it interprets your 42766.08001 as a single precision literal, and finds the nearest representable value, which is only accurate to about 7 decimal places. If you want a literal to be interpretted as a double precision, it mus have D# (or d#) as a suffix,

where # is an appropriate decimal exponent, i.e. you could represent
42766.08001 as any of
42766.08001D0
42766.08001 d0
42.76608001 D3
4.276608001 D4
0.4276608001 D5
...
to have it interpretted as a double precision number.