Subject: Re: U16 bit question
Posted by Peter Mason on Tue, 09 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

On Mon, 8 Jul 1996, Dave Johnson wrote:
> I would be grateful if anyone could answer this question. Is it
> possible to represent a number or an array in a true unsigned
> 16 bit way?
> It would be nice to do this in IDL rather than C/C++ or Fortran
> or another language.
> The problem arises from reading U16 bit words from a cdrom and
> placing the data into an array. Due to the way IDL represents
> integer words, anything greater than 7FFF(hex) gets its bits
> shifted around.

No, IDL doesn't have an unsigned integer type. But you can work around this to some extent by using longs:
Going from int to long:
long_val=int_val & if (long_val lt 0L) then long_val=65536L+long_val
Going from long to int:
int_val=long_val ;just takes the lower 16 bits as they are
The idea is that you'd read your data into an INT array; convert as above (but array-wise) to LONGs, process/view/etc; convert back to INTs if you want to save processed data.
Depending on the sort of processing you want to do, you could probably emulate U16 behaviour by trapping negatives (as above), and by ANDing with 65535L (after each processing step) to keep the range restricted.
Graceless, I know, but a way out.

But your message gives a hint that you might have an additional problem - byte ordering. Data is not actually altered (bit shifted etc) by virtue of being read into a signed int instead of an unsigned one. It is only the interpretation of the data (the way arithmetic works on it, etc) that changes.
But if the dataset was created on a "big endian" platform and is then read by a "small endian" one, or vice-versa, then the 2 bytes in each INT will end up reversed, and you have to put things right with:
byteorder,INT_array,/sswap

You can tell what order your own platform uses with:
big_endian=byte('0100'x,0,1)
A result of 1b indicates big endian, 0b indicates small endian.

You also have to know or guess the order of the platform on which the data was created. Some I know:
. Intel PC or compat (dos, win3.1x, win95, winnt) small endian
. Any DEC computer (AXP/NT, OSF, VMS, Ultrix) small endian
. SGI, SUN, 680x0, the rest of the world, basically usually big endian

(In practice, of course, one just tries out the data as-is, and attempts
byte reversal if the data appears meaningless :)

Peter Mason
