Subject: Re: UNsigned Integer Data
Posted by davidf on Wed, 05 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Marc Kippen <marc.kippen@msfc.nasa.gov> writes:

> Can anyone tell me how to convert between signed and unsigned
> short integer data formats in IDL?

There is no unsigned integer format in IDL. :-)

> e.g., how can I decode a byte array into an unsigned short
> integer? The FIX function automatically interprets the sign
> bit as a sign, rather than a data bit.

Marc, I'm going to assume that what you mean by this is that you have unsigned 16-bit integers in a binary or unformated data file. And you want to know how to interpret this data properly in IDL.

Suppose your have a 256 by 256 array of these 16-bit integers in your binary data file. You will do something like this:

```
    ; Read the data as IDL signed integers (16-bit)

  array = INTARR(256, 256)
  OPENR, lun, unsignedIntDataFile, /GET_LUN
  READU, lun, array
  FREE_LUN, lun

    ; Convert the SIGNED integers to UNSIGNED values.

  array = LONG(array) AND 'FFFF'x
```

Now you have an array of LONG integers, but they have the correct unsigned values. There is no way to get around the requirement for LONG integers unless your data is always between 0 and 2^31-1 or 2147483647.

(If you have a 256 by 256 byte array [another way to interpret your question], then you will want to make array = INTARR(128,128) to read the data correctly.)

Cheers!

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: UNsigned Integer Data
Posted by Struan Gray on Thu, 06 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning, davidf@dfanning.com writes:

> array = LONG(array) AND 'FFFF'x
>
> Now you have an array of LONG integers, but they
> have the correct unsigned values. There is no way to get
> around the requirement for LONG integers unless your data
> is always between 0 and 2^31-1 or 2147483647

If you have unsigned, 2-byte integers that go:

0000 0000 0000 0000 : 0
1000 0000 0000 0000 : 32768
1111 1111 1111 1111 : 65535

All you really need to do to get signed integers is twiddle the top bit:

1000 0000 0000 0000 : 2s-comp -32768
0000 0000 0000 0000 : 2s-comp 0
0111 1111 1111 1111 : 2s-comp 32767

Any routines that calculate a real-world value (ie to label and axis on a graph) need to have an offset part that accounts for the fact that 'zero' has moved, but for a lot of datasets that's a more convenient way of handling things anyway.

Thus you can avoid the memory and time penalty of converting to longs with something like this:

```
array = temporary(array)  xor fix(-32768)
```

It definitely works faster on our macs, and since some of our data files are pushing the memory limits anyway it saves

a lot of disk-swapping for the big ones.

Struan
