Subject: Re: Extracting bits with IDL
Posted by stl on Mon, 18 Jul 1994 08:58:21 GMT
View Forum Message <> Reply to Message

In article <Ct2GM1.1780@yuma.ACNS.ColoState.EDU>
dean%phobos.dnet@sirius.cira.colostate.edu writes:
> .....
> contains some time information. Extracting individual bits from data with IDL
> is difficult. At first I consider building an IDL structure, but IDL is
> unable to break down to bits.
kinda sorta not/really (see below)
> The information can fit into LONARR(2). If anyone has any suggestion that
> would allow IDL to extract the information from these LONARR(2), please
> forward your comments to me or post them.

Hi,

okay, to strip a long apart bit by bit, simple test each bit. For
instance if you wish to check if the second bit was set (binary value of
0000 0000 0000 0010) AND it with 2, if the result is 2, the bit was set,
otherwise it was not. This can then be done for each bit, or you could
check groups of bits this way.
I am sure there are other methods of doing this, but this should be
pretty functional method. In IDL do something like the following:

```
x = 3L  ;sample long data, lets test it to see if the second
   ; bit is set
if (x and 2) eq 2 then print,"second bit set" else $
 print,"second bit wasn't set"
```

-hope this helps, and is what your looking for.

-stephen
--
Stephen C Strebel / SKI TO DIE
strebel@sma.ch / and
Swiss Meteorological Institute, Zuerich / LIVE TO TELL ABOUT IT
01 256 93 85 / (and pray for snow)

Subject: Re: Extracting bits with IDL
Posted by gumley on Mon, 18 Jul 1994 13:51:49 GMT
View Forum Message <> Reply to Message

In article <Ct2GM1.1780@yuma.ACNS.ColoState.EDU>,
dean@phobos.cira.colostate.edu wrote:

> We are set up to routinely collect GOES-8 GVAR data. Like with most data
> from satellites, they put information in the individual bits. Below is a C
> structure that extracts the individual bits from part of the data header which
> contains some time information. Extracting individual bits from data with IDL
> is difficult. At first I consider building an IDL structure, but IDL is
> unable to break down to bits.

IDL can actually extract bits quite easily. You just need to use the ISHFT and AND functions. ISHFT shifts byte, integer, or longword arguments to the left or right, and fills any vacant positions with zeros. AND performs a bitwise AND on byte, integer, or longword arguments. For example, let's say I have an eight bit word, and I want to extract the first 4 bits and the last 4 bits as two separate values. To extract the leftmost 4 bits, you would use

value = ishft(bytval, -4)

which shifts to the right 4 positions, and fills in the space created on the left with zeros. To get the rightmost 4 bits, you would use

value = bytval and 15

which does a bitwise AND with any bits to set to 1 in the rightmost 4 positions. You can also use shift/mask(and) operations to extract other bit fields. For example if you wanted to extract bits 4 and 3 (bits are typically numbered left to right in descending order i.e. bit 7 to bit 0 in an 8 bit word), you would use

value = ishft(bytval, -3) and 3

I hope this helps. It is usually instructive to write a short test program to convince yourself that it works.

Cheers,
Liam.

--
Liam E. Gumley
NASA/GSFC Climate and Radiation Branch
Greenbelt MD, USA

Subject: Re: Extracting bits with IDL
Posted by chase on Mon, 18 Jul 1994 20:47:36 GMT
View Forum Message <> Reply to Message

>>>> > "dean" == dean <dean@phobos.cira.colostate.edu> writes:

dean> We are set up to routinely collect GOES-8 GVAR data. Like with
dean> most data from satellites, they put information in the
dean> individual bits. Below is a C structure that extracts the
dean> individual bits from part of the data header which contains some
dean> time information. Extracting individual bits from data with IDL
dean> is difficult. At first I consider building an IDL structure, but
dean> IDL is unable to break down to bits.

dean> The information can fit into LONARR(2). If anyone has any
dean> suggestion that would allow IDL to extract the information from
dean> these LONARR(2), please forward your comments to me or post
dean> them. I hope to use this information to build a widget to
dean> navigate through this new and fine data set coming from GOES-8.

There was a discussion in this group some time ago about obtaining bit information in IDL. I wrote a very general function for arbitrary base decoding called decode.pro which I include for your use below. One of its applications can be to decode an array of scalars into base 2. For example, with geos = lonarr(2), decode(geos,2) returns an array containing the binary expansion for geos which you can then test.

I am not exactly sure which bits of your LONARR(2) correspond to your C struct definition because the C ANSI standard says that bit-fields are implementation dependent, i.e., non-portable. For example, whether fields are assigned left to right or right to left and/or can overlap word boundaries can vary between machines.

Good luck,
Chris

--- begin included file ---
```
FUNCTION Decode, scl, dim, help=help
;+
; $Id: decode.pro,v 1.1 1994/07/18 16:05:32 chase Exp $
;
; NAME:
;
;   DECODE
;
; PURPOSE:
;
;   Decode a vector of scalars into the dimensions d1,...,dn
;   in order of increasing significance.
;
; CATEGORY:
;
;   Mathematical Functions
```

```
;
; CALLING SEQUENCE:
;
;   Result = DECODE(Scl, Dim)
;
; INPUTS:
;
;   Scl - Vector of scalars to be decoded. Will be converted to
;         integers first, truncating if necessary.
;
;   Dim - If a scalar, then it is converted to an integer base for the
;         decoding, i.e., D1,...,DN=Dim.
;         If a 1 dimensional vector, then it is converted to the
;         integer dimensions D1,...,DN.
;         If > 1 dimensional array, then the dimensions of the array
;         are used for D1,...,DN.
;         The dimensions increase in significance, i.e., the first
;         dimension is the least significant and the last dimension is
;         the most significant.
;
; KEYWORD PARAMETERS:
;
;   HELP - Provide help (this information). No other action is performed.
;
; OUTPUTS:
;
;   Result - Array of size NxM where M is dimension of Scl.
;            Result(*,i) is the decoding of the scalar Scl(i). If
;            Scl(i) is larger then the dimensioned size,
;            i.e. D1x...xDN, then the modulus, Scl(i) mod D1x...xDN,
;            is decoded.  Result(j-1,i) corresponds to dimension Dj
;            with Result(N-1,i) the most significant digit of the
;            decoding.
;
; PROCEDURE:
;
;   Let b1,...,bN be the decoding. Then Scl can be represented as:
;
;   Scl = D1(D2(...(D[N-1]*bN + b[N-1])...+ b3 ) + b2) + b1
;
;   with 0 <= bi < Di
;
; EXAMPLE:
;
;   scl = [20,63]
;   ; Conversion to base 16
;   print,decode(scl,16)
;
```

```
;   ; Conversion to binary (base 2)
;   print,decode(scl,2)
;   ; Invert the decoding
;   print,2^indgen(5)#decode(scl,2)
;
;   ; Arbitrary decoding.  Generates a warning for decoding 63 in
;   ; which case (63 mod 3*4*5) = 3 is decoded.
;   print,decode(scl,[3,4,5])
;   print,[1,3,3*4]#decode(scl,[3,4,5])
;   print,[1,3,3*4,3*4*5]#decode(scl,[3,4,5,6])
;
;   ; Convert 1D index into a multi-dimensional index
;   w=dist(20,20)
;   a=max(w,i)
;   ; Get 2D index for max
;   print,decode(i,w)
;
; MODIFICATION HISTORY:
;
;       Mon Jul 18 15:58:18 1994, Chris Chase S1A <chase@jackson>
;  Fixed/cleaned up.
;
;       Mon Jul 26 12:17:56 1993, Chris Chase <chase@aphill>
;               Created.  Named for similar APL function.
;
;-
if keyword_set(help) then begin
    doc_library, 'decode'
    return, ''
endif
s = size(dim)
if (s(0) eq 0) then begin
    d = replicate(long(dim), long(alog(max(scl))/alog(dim)) + 1)
endif else begin
    if (s(0) gt 1) then d = s(1:s(0)) $
    else d = dim
endelse

d = long(d)
nd = n_elements(d)
v = long(scl)
index = lonarr(nd, n_elements(v))
for i = 0, nd-1 do begin
    f = v/d(i)
    index(i, *) = v-f*d(i)
    v = f
endfor
if (max(v) ne 0) then begin
```

```
    print, "Warning - function DECODE: scalar outside dimension " + $
      "bounds, decode of modulus returned."
endif
return, index
end
```

--

==========

Bldg 24-E188
The Applied Physics Laboratory
The Johns Hopkins University
(301)953-6000 x8529
chris_chase@jhuapl.edu

Subject: Re: Extracting bits with IDL
Posted by dball on Mon, 18 Jul 1994 22:32:41 GMT
View Forum Message <> Reply to Message

In article <Ct2GM1.1780@yuma.ACNS.ColoState.EDU>, dean@phobos.cira.colostate.edu writes:

> We are set up to routinely collect GOES-8 GVAR data. Like with most data
> from satellites, they put information in the individual bits. Below is a C
> structure that extracts the individual bits from part of the data header which
> contains some time information. Extracting individual bits from data with IDL
> is difficult. At first I consider building an IDL structure, but IDL is
> unable to break down to bits.
>
> The information can fit into LONARR(2). If anyone has any suggestion that
> would allow IDL to extract the information from these LONARR(2), please
> forward your comments to me or post them. I hope to use this information to
> build a widget to navigate through this new and fine data set coming from
> GOES-8.
>
> ==========

```
>
> typedef struct
> {
>     unsigned year_100   : 4;
>     unsigned year_1000  : 4;
>
>     unsigned year_1     : 4;
>     unsigned year_10    : 4;
>
>     unsigned day_10     : 4;
>     unsigned day_100    : 3;
>     unsigned time_code  : 1;
>
>     unsigned hour_10    : 4;
```

```
>     unsigned day_1      : 4;
>
>     unsigned min_10     : 4;
>     unsigned hour_1     : 4;
>
>     unsigned sec_10     : 4;
>     unsigned min_1      : 4;
>
>     unsigned msec_100   : 4;
>     unsigned sec_1      : 4;
>
>     unsigned msec_1     : 4;
>     unsigned msec_10    : 4;
>  } BCD_TIME_T;
>
>  ******************************************************  \      *****
>  Kelly Dean                                              \
>  E-MAIL: DEAN@PHOBOS.CIRA.COLOSTATE.EDU                      |  \
>   ************************************************************ **************
```

You can use a combination of AND (which is a bitwise logical operator) and
ISHFT (which shifts bits left or right) to extract these values.
Its probably easier if you read it into a BYTE array. e.g.:

```
b=BYTARR(8)
READU,1,b
year_100 = ISHFT( b(0), -4 )          ; shift top 4 bits down
year_1000 = b(0) AND 'F'XB             ; mask off bottom 4 bits
day_100 = ISHFT( b(2), -1 ) AND '7'XB  ; shift bits down one, and mask off 3
```

'n'XB is just a byte size hex constant. 'F'XB = 15B and '7'XB = 7B

-- Dave