
Subject: Re: storing in array

Posted by [Spon](#) on Sun, 11 May 2008 08:11:04 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 11, 12:48 am, kishore1...@gmail.com wrote:

> Hi,
>
> I am new in IDL language. For reading the CALIPSO satellite data sets
> in that what type of cloud type for that they provided read routine.
> If I use this routine it is printing on IDL log window. How to store
> this information into an string array.
> This is the routine
> vfm_feature_flags,36282
> The output is like this:
> Bit set: 2
> Bit set: 4
> Bit set: 5
> Bit set: 6
> Bit set: 8
> Bit set: 9
> Bit set: 11
> Bit set: 12
> Bit set: 16
> Feature Type : cloud
> Feature Subtype : cirrus (transparent)
> Feature Type QA : high
> Ice/Water Phase : ice
> Ice/Water Phase QA: high
> Cloud/Aerosol/PSC Type QA : not confident
> Horizontal averaging required for detection: 20 km
>
> The above information i want to store in string array.
>
> *****routine*****
> ;
> pro vfm_feature_flags,val
> ; this routine demonstrates how to read and extract values from a
> feature
> ; classification flag 16-bit integer value in CALIPSO Level 2
> Vertical
> ; Feature Mask files
> ;
> ; INPUT:
> ; val - the feature classification flag value to be decoded
> ;
> ; OUTPUT:
> ; all information is printed into the IDL log window
>
> print, val

```

>
> feature_type = 0
> feature_type_qa = 0
> ice_water_phase = 0
> ice_water_phase_qa = 0
> feature_subtype = 0
> cloud_aerosol_psc_type_qa = 0
> horizontal_averaging = 0
>
> for i=0,15 do begin
>   if ((val and 2L^i) NE 0) then begin
>     print,'Bit set: ',i+1
>     case i+1 of
>       1 : feature_type = feature_type + 1
>       2 : feature_type = feature_type + 2
>       3 : feature_type = feature_type + 4
>       4 : feature_type_qa = feature_type_qa + 1
>       5 : feature_type_qa = feature_type_qa + 2
>       6 : ice_water_phase = ice_water_phase + 1
>       7 : ice_water_phase = ice_water_phase + 2
>       8 : ice_water_phase_qa = ice_water_phase_qa + 1
>       9 : ice_water_phase_qa = ice_water_phase_qa + 2
>      10 : feature_subtype = feature_subtype + 1
>      11 : feature_subtype = feature_subtype + 2
>      12 : feature_subtype = feature_subtype + 4
>      13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
>      14 : horizontal_averaging = horizontal_averaging + 1
>      15 : horizontal_averaging = horizontal_averaging + 2
>      16 : horizontal_averaging = horizontal_averaging + 4
>     else:
>       endcase
>   endif
> endfor
>
> case feature_type of
> 0 : print,"Feature Type : invalid (bad or missing data)"
> 1 : print,"Feature Type : clear air"
> 2 : begin
>   print,"Feature Type : cloud"
>   case feature_subtype of
>   0 : print, "Feature Subtype : low overcast, transparent"
>   1 : print, "Feature Subtype : low overcast, opaque"
>   2 : print, "Feature Subtype : transition stratocumulus"
>   3 : print, "Feature Subtype : low, broken cumulus"
>   4 : print, "Feature Subtype : altocumulus (transparent)"
>   5 : print, "Feature Subtype : altostratus (opaque)"
>   6 : print, "Feature Subtype : cirrus (transparent)"
>   7 : print, "Feature Subtype : deep convective (opaque)"

```

```

>     else : print,"*** error getting Feature Subtype"
>     endcase
> end
> 3 : begin
>     print,"Feature Type : aerosol"
>     case feature_subtype of
>     0 : print, "Feature Subtype : not determined"
>     1 : print, "Feature Subtype : clean marine"
>     2 : print, "Feature Subtype : dust"
>     3 : print, "Feature Subtype : polluted continental"
>     4 : print, "Feature Subtype : clean continental"
>     5 : print, "Feature Subtype : polluted dust"
>     6 : print, "Feature Subtype : smoke"
>     7 : print, "Feature Subtype : other"
>     else : print,"*** error getting Feature Subtype"
>     endcase
> end
> 4 : begin
>     print,"Feature Type : stratospheric feature--PSC or
> stratospheric aerosol"
>     case feature_subtype of
>     0 : print, "Feature Subtype : not determined"
>     1 : print, "Feature Subtype : non-depolarizing PSC"
>     2 : print, "Feature Subtype : depolarizing PSC"
>     3 : print, "Feature Subtype : non-depolarizing aerosol"
>     4 : print, "Feature Subtype : depolarizing aerosol"
>     5 : print, "Feature Subtype : spare"
>     6 : print, "Feature Subtype : spare"
>     7 : print, "Feature Subtype : other"
>     else : print,"*** error getting Feature Subtype"
>     endcase
> end
> 5 : print,"Feature Type : surface"
> 6 : print,"Feature Type : subsurface"
> 7 : print,"Feature Type : no signal (totally attenuated)"
> else : print,"*** error getting Feature Type"
> endcase
>
> case feature_type_qa of
> 0 : print,"Feature Type QA : none"
> 1 : print,"Feature Type QA : low"
> 2 : print,"Feature Type QA : medium"
> 3 : print,"Feature Type QA : high"
> else : print,"*** error getting Feature Type QA"
> endcase
>
> case ice_water_phase of
> 0 : print,"Ice/Water Phase : unknown/not determined"

```

```

> 1 : print,"Ice/Water Phase : ice"
> 2 : print,"Ice/Water Phase : water"
> 3 : print,"Ice/Water Phase : mixed phase"
> else : print,"*** error getting Ice/Water Phase"
> endcase
>
> case ice_water_phase_qa of
> 0 : print,"Ice/Water Phase QA: none"
> 1 : print,"Ice/Water Phase QA: low"
> 2 : print,"Ice/Water Phase QA: medium"
> 3 : print,"Ice/Water Phase QA: high"
> else : print,"*** error getting Ice/Water Phase QA"
> endcase
>
> if (cloud_aerosol_psc_type_qa eq 0) then begin
>   print,"Cloud/Aerosol/PSC Type QA : not confident"
> endif else begin
>   print,"Cloud/Aerosol/PSC Type QA : confident"
> endelse
>
> case horizontal_averaging of
> 0 : print,"Horizontal averaging required for detection: not
>   applicable"
> 1 : print,"Horizontal averaging required for detection: 1/3 km"
> 2 : print,"Horizontal averaging required for detection: 1 km"
> 3 : print,"Horizontal averaging required for detection: 5 km"
> 4 : print,"Horizontal averaging required for detection: 20 km"
> 5 : print,"Horizontal averaging required for detection: 80 km"
> else : print,"*** error getting Horizontal averaging"
> endcase
> end

```

Rather than a string array, I would use an anonymous structure for this sort of information:

e.g.

```
Data = {Type:"", Subtype:"", QA:"", Phase:"", PhaseQA:"", TypeQA:""} ;
```

etc

This creates a set of empty string fields.

then you can read in your strings like this:

```
data.type = 'cloud'
```

and retrieve your information like this:

```
help, data, /struct
```

```
IDL> ** Structure <ff8a18>, 6 tags, length=72, data length=72, refs=1:
```

```
  TYPE          STRING  'cloud'
```

```
  SUBTYPE       STRING  "
```

```
QA      STRING  "  
PHASE   STRING  "  
PHASEQA STRING  "  
TYPEQA  STRING  "
```

Reading in your strings could probably be done into a string array

either if you really want to:

```
Result = STRARR[n]  
FOR i = 0, n-1 DO BEGIN  
Result[i] = 'Cloud'  
ENDFOR ; etc.
```

Also, you can probably use format codes to convert your bytes/integers to set bits, something along the lines of:

```
CloudBits = STRING(CloudByte, FORMAT='(B0)')  
NCloudBits = STRLEN(CloudBits)
```

Which I can imagine would make determining your string contents a bit easier. I can't remember exactly, and I'm stuck with demo-mode only today(!), so you'll have to play around with it yourself, unless someone more knowledgeable jumps in with a fuller explanation.

Good luck!

Chris

Subject: Re: storing in array

Posted by [kishore1818](#) on Sun, 11 May 2008 21:26:37 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 11, 5:11 am, Spon <christoph.b...@gmail.com> wrote:

> On May 11, 12:48 am, kishore1...@gmail.com wrote:

>

>

>

>> Hi,

>

>> I am new in IDL language. For reading the CALIPSO satellite data sets

>> in that what type of cloud type for that they provided read routine.

>> If I use this routine it is printing on IDL log window. How to store

>> this information into an string array.

>> This is the routine

>> vfm_feature_flags,36282

>> The output is like this:

>> Bit set: 2

>> Bit set: 4

>> Bit set: 5

>> Bit set: 6

```

>> Bit set:      8
>> Bit set:      9
>> Bit set:     11
>> Bit set:     12
>> Bit set:     16
>> Feature Type : cloud
>> Feature Subtype : cirrus (transparent)
>> Feature Type QA : high
>> Ice/Water Phase : ice
>> Ice/Water Phase QA: high
>> Cloud/Aerosol/PSC Type QA : not confident
>> Horizontal averaging required for detection: 20 km
>
>> The above information i want to store in string array.
>
>> ;*****routine*****
>> pro vfm_feature_flags,val
>> ; this routine demonstrates how to read and extract values from a
>> feature
>> ; classification flag 16-bit integer value in CALIPSO Level 2
>> Vertical
>> ; Feature Mask files
>> ;
>> ; INPUT:
>> ; val - the feature classification flag value to be decoded
>> ;
>> ; OUTPUT:
>> ; all information is printed into the IDL log window
>
>> print, val
>
>> feature_type = 0
>> feature_type_qa = 0
>> ice_water_phase = 0
>> ice_water_phase_qa = 0
>> feature_subtype = 0
>> cloud_aerosol_psc_type_qa = 0
>> horizontal_averaging = 0
>
>> for i=0,15 do begin
>>   if ((val and 2L^i) NE 0) then begin
>>     print,'Bit set: ',i+1
>>     case i+1 of
>>       1 : feature_type = feature_type + 1
>>       2 : feature_type = feature_type + 2
>>       3 : feature_type = feature_type + 4
>>       4 : feature_type_qa = feature_type_qa + 1
>>       5 : feature_type_qa = feature_type_qa + 2

```

```

>> 6 : ice_water_phase = ice_water_phase + 1
>> 7 : ice_water_phase = ice_water_phase + 2
>> 8 : ice_water_phase_qa = ice_water_phase_qa + 1
>> 9 : ice_water_phase_qa = ice_water_phase_qa + 2
>> 10 : feature_subtype = feature_subtype + 1
>> 11 : feature_subtype = feature_subtype + 2
>> 12 : feature_subtype = feature_subtype + 4
>> 13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
>> 14 : horizontal_averaging = horizontal_averaging + 1
>> 15 : horizontal_averaging = horizontal_averaging + 2
>> 16 : horizontal_averaging = horizontal_averaging + 4
>> else:
>>     endcase
>> endif
>> endfor
>
>> case feature_type of
>> 0 : print,"Feature Type : invalid (bad or missing data)"
>> 1 : print,"Feature Type : clear air"
>> 2 : begin
>>     print,"Feature Type : cloud"
>>     case feature_subtype of
>> 0 : print, "Feature Subtype : low overcast, transparent"
>> 1 : print, "Feature Subtype : low overcast, opaque"
>> 2 : print, "Feature Subtype : transition stratocumulus"
>> 3 : print, "Feature Subtype : low, broken cumulus"
>> 4 : print, "Feature Subtype : altocumulus (transparent)"
>> 5 : print, "Feature Subtype : altostratus (opaque)"
>> 6 : print, "Feature Subtype : cirrus (transparent)"
>> 7 : print, "Feature Subtype : deep convective (opaque)"
>>     else : print,"*** error getting Feature Subtype"
>>     endcase
>>     end
>> 3 : begin
>>     print,"Feature Type : aerosol"
>>     case feature_subtype of
>> 0 : print, "Feature Subtype : not determined"
>> 1 : print, "Feature Subtype : clean marine"
>> 2 : print, "Feature Subtype : dust"
>> 3 : print, "Feature Subtype : polluted continental"
>> 4 : print, "Feature Subtype : clean continental"
>> 5 : print, "Feature Subtype : polluted dust"
>> 6 : print, "Feature Subtype : smoke"
>> 7 : print, "Feature Subtype : other"
>>     else : print,"*** error getting Feature Subtype"
>>     endcase
>>     end
>> 4 : begin

```

```

>>   print,"Feature Type : stratospheric feature--PSC or
>> stratospheric aerosol"
>>   case feature_subtype of
>>   0 : print, "Feature Subtype : not determined"
>>   1 : print, "Feature Subtype : non-depolarizing PSC"
>>   2 : print, "Feature Subtype : depolarizing PSC"
>>   3 : print, "Feature Subtype : non-depolarizing aerosol"
>>   4 : print, "Feature Subtype : depolarizing aerosol"
>>   5 : print, "Feature Subtype : spare"
>>   6 : print, "Feature Subtype : spare"
>>   7 : print, "Feature Subtype : other"
>>   else : print,"*** error getting Feature Subtype"
>>   endcase
>> end
>> 5 : print,"Feature Type : surface"
>> 6 : print,"Feature Type : subsurface"
>> 7 : print,"Feature Type : no signal (totally attenuated)"
>> else : print,"*** error getting Feature Type"
>> endcase
>
>> case feature_type_qa of
>> 0 : print,"Feature Type QA : none"
>> 1 : print,"Feature Type QA : low"
>> 2 : print,"Feature Type QA : medium"
>> 3 : print,"Feature Type QA : high"
>> else : print,"*** error getting Feature Type QA"
>> endcase
>
>> case ice_water_phase of
>> 0 : print,"Ice/Water Phase : unknown/not determined"
>> 1 : print,"Ice/Water Phase : ice"
>> 2 : print,"Ice/Water Phase : water"
>> 3 : print,"Ice/Water Phase : mixed phase"
>> else : print,"*** error getting Ice/Water Phase"
>> endcase
>
>> case ice_water_phase_qa of
>> 0 : print,"Ice/Water Phase QA: none"
>> 1 : print,"Ice/Water Phase QA: low"
>> 2 : print,"Ice/Water Phase QA: medium"
>> 3 : print,"Ice/Water Phase QA: high"
>> else : print,"*** error getting Ice/Water Phase QA"
>> endcase
>
>> if (cloud_aerosol_psc_type_qa eq 0) then begin
>>   print,"Cloud/Aerosol/PSC Type QA : not confident"
>> endif else begin
>>   print,"Cloud/Aerosol/PSC Type QA : confident"

```



```

>> endelse
>
>> case horizontal_averaging of
>> 0 : print,"Horizontal averaging required for detection: not
>> applicable"
>> 1 : print,"Horizontal averaging required for detection: 1/3 km"
>> 2 : print,"Horizontal averaging required for detection: 1 km"
>> 3 : print,"Horizontal averaging required for detection: 5 km"
>> 4 : print,"Horizontal averaging required for detection: 20 km"
>> 5 : print,"Horizontal averaging required for detection: 80 km"
>> else : print,"*** error getting Horizontal averaging"
>> endcase
>> end
>
> Rather than a string array, I would use an anonymous structure for
> this sort of information:
>
> e.g.
> Data = {Type:"", Subtype:"", QA:"", Phase:"", PhaseQA:"", TypeQA:""} ;
> etc
> This creates a set of empty string fields.
>
> then you can read in your strings like this:
> data.type = 'cloud'
>
> and retrieve your information like this:
> help, data, /struct
> IDL> ** Structure <ff8a18>, 6 tags, length=72, data length=72, refs=1:
>  TYPE      STRING  'cloud'
>  SUBTYPE    STRING  "
>  QA         STRING  "
>  PHASE      STRING  "
>  PHASEQA    STRING  "
>  TYPEQA     STRING  "
>
> Reading in your strings could probably be done into a string array
> either if you really want to:
> Result = STRARR[n]
> FOR i = 0, n-1 DO BEGIN
>   Result[i] = 'Cloud'
> ENDFOR ; etc.
>
> Also, you can probably use format codes to convert your bytes/integers
> to set bits, something along the lines of:
> CloudBits = STRING(CloudByte, FORMAT='(B0)')
> NCloudBits = STRLEN(CloudBits)
>
> Which I can imagine would make determining your string contents a bit

```

> easier. I can't remember exactly, and I'm stuck with demo-mode only
> today(!), so you'll have to play around with it yourself, unless
> someone more knowledgeable jumps in with a fuller explanation.
>
> Good luck!
> Chris

Hi Chris,

Thanks for your quick reply.

I am using this routine in my program "vfm_feature_flags,36282"

This one is changing every time. Just I want to store o/p information into a string.

Your program is simple, but I could not understand clearly. For example I want to store "Feature Subtype : cirrus (transparent) " and this one is changing every time. Some times cirrus and some times convective ..

Could you give me little clearly then I can follow.

Thanking you,

Kishore

Subject: Re: storing in array

Posted by [Spon](#) on Mon, 12 May 2008 10:03:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 11, 10:26 pm, kishore1...@gmail.com wrote:

> On May 11, 5:11 am, Spon <christoph.b...@gmail.com> wrote:

>

>

>

>> On May 11, 12:48 am, kishore1...@gmail.com wrote:

>

>>> Hi,

>

>>> I am new in IDL language. For reading the CALIPSO satellite data sets

>>> in that what type of cloud type for that they provided read routine.

>>> If I use this routine it is printing on IDL log window. How to store

>>> this information into an string array.

>>> This is the routine

>>> vfm_feature_flags,36282

>>> The output is like this:

>>> Bit set: 2

>>> Bit set: 4

>>> Bit set: 5

>>> Bit set: 6

```

>>> Bit set:      8
>>> Bit set:      9
>>> Bit set:     11
>>> Bit set:     12
>>> Bit set:     16
>>> Feature Type : cloud
>>> Feature Subtype : cirrus (transparent)
>>> Feature Type QA : high
>>> Ice/Water Phase : ice
>>> Ice/Water Phase QA: high
>>> Cloud/Aerosol/PSC Type QA : not confident
>>> Horizontal averaging required for detection: 20 km
>
>>> The above information i want to store in string array.
>
>>> ;*****routine*****
>>> pro vfm_feature_flags,val
>>> ; this routine demonstrates how to read and extract values from a
>>> feature
>>> ; classification flag 16-bit integer value in CALIPSO Level 2
>>> Vertical
>>> ; Feature Mask files
>>> ;
>>> ; INPUT:
>>> ; val - the feature classification flag value to be decoded
>>> ;
>>> ; OUTPUT:
>>> ; all information is printed into the IDL log window
>
>>> print, val
>
>>> feature_type = 0
>>> feature_type_qa = 0
>>> ice_water_phase = 0
>>> ice_water_phase_qa = 0
>>> feature_subtype = 0
>>> cloud_aerosol_psc_type_qa = 0
>>> horizontal_averaging = 0
>
>>> for i=0,15 do begin
>>>   if ((val and 2L^i) NE 0) then begin
>>>     print,'Bit set: ',i+1
>>>     case i+1 of
>>>       1 : feature_type = feature_type + 1
>>>       2 : feature_type = feature_type + 2
>>>       3 : feature_type = feature_type + 4
>>>       4 : feature_type_qa = feature_type_qa + 1
>>>       5 : feature_type_qa = feature_type_qa + 2

```

```

>>> 6 : ice_water_phase = ice_water_phase + 1
>>> 7 : ice_water_phase = ice_water_phase + 2
>>> 8 : ice_water_phase_qa = ice_water_phase_qa + 1
>>> 9 : ice_water_phase_qa = ice_water_phase_qa + 2
>>> 10 : feature_subtype = feature_subtype + 1
>>> 11 : feature_subtype = feature_subtype + 2
>>> 12 : feature_subtype = feature_subtype + 4
>>> 13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
>>> 14 : horizontal_averaging = horizontal_averaging + 1
>>> 15 : horizontal_averaging = horizontal_averaging + 2
>>> 16 : horizontal_averaging = horizontal_averaging + 4
>>> else:
>>> endcase
>>> endif
>>> endfor
>
>>> case feature_type of
>>> 0 : print,"Feature Type : invalid (bad or missing data)"
>>> 1 : print,"Feature Type : clear air"
>>> 2 : begin
>>>     print,"Feature Type : cloud"
>>>     case feature_subtype of
>>>     0 : print, "Feature Subtype : low overcast, transparent"
>>>     1 : print, "Feature Subtype : low overcast, opaque"
>>>     2 : print, "Feature Subtype : transition stratocumulus"
>>>     3 : print, "Feature Subtype : low, broken cumulus"
>>>     4 : print, "Feature Subtype : altocumulus (transparent)"
>>>     5 : print, "Feature Subtype : altostratus (opaque)"
>>>     6 : print, "Feature Subtype : cirrus (transparent)"
>>>     7 : print, "Feature Subtype : deep convective (opaque)"
>>>     else : print, "*** error getting Feature Subtype"
>>>     endcase
>>> end
>>> 3 : begin
>>>     print,"Feature Type : aerosol"
>>>     case feature_subtype of
>>>     0 : print, "Feature Subtype : not determined"
>>>     1 : print, "Feature Subtype : clean marine"
>>>     2 : print, "Feature Subtype : dust"
>>>     3 : print, "Feature Subtype : polluted continental"
>>>     4 : print, "Feature Subtype : clean continental"
>>>     5 : print, "Feature Subtype : polluted dust"
>>>     6 : print, "Feature Subtype : smoke"
>>>     7 : print, "Feature Subtype : other"
>>>     else : print, "*** error getting Feature Subtype"
>>>     endcase
>>> end
>>> 4 : begin

```

```

>>>     print,"Feature Type : stratospheric feature--PSC or
>>> stratospheric aerosol"
>>>     case feature_subtype of
>>>     0 : print, "Feature Subtype : not determined"
>>>     1 : print, "Feature Subtype : non-depolarizing PSC"
>>>     2 : print, "Feature Subtype : depolarizing PSC"
>>>     3 : print, "Feature Subtype : non-depolarizing aerosol"
>>>     4 : print, "Feature Subtype : depolarizing aerosol"
>>>     5 : print, "Feature Subtype : spare"
>>>     6 : print, "Feature Subtype : spare"
>>>     7 : print, "Feature Subtype : other"
>>>     else : print,"*** error getting Feature Subtype"
>>>     endcase
>>> end
>>> 5 : print,"Feature Type : surface"
>>> 6 : print,"Feature Type : subsurface"
>>> 7 : print,"Feature Type : no signal (totally attenuated)"
>>> else : print,"*** error getting Feature Type"
>>> endcase
>
>>> case feature_type_qa of
>>> 0 : print,"Feature Type QA : none"
>>> 1 : print,"Feature Type QA : low"
>>> 2 : print,"Feature Type QA : medium"
>>> 3 : print,"Feature Type QA : high"
>>> else : print,"*** error getting Feature Type QA"
>>> endcase
>
>>> case ice_water_phase of
>>> 0 : print,"Ice/Water Phase : unknown/not determined"
>>> 1 : print,"Ice/Water Phase : ice"
>>> 2 : print,"Ice/Water Phase : water"
>>> 3 : print,"Ice/Water Phase : mixed phase"
>>> else : print,"*** error getting Ice/Water Phase"
>>> endcase
>
>>> case ice_water_phase_qa of
>>> 0 : print,"Ice/Water Phase QA: none"
>>> 1 : print,"Ice/Water Phase QA: low"
>>> 2 : print,"Ice/Water Phase QA: medium"
>>> 3 : print,"Ice/Water Phase QA: high"
>>> else : print,"*** error getting Ice/Water Phase QA"
>>> endcase
>
>>> if (cloud_aerosol_psc_type_qa eq 0) then begin
>>>   print,"Cloud/Aerosol/PSC Type QA : not confident"
>>> endif else begin
>>>   print,"Cloud/Aerosol/PSC Type QA : confident"

```

```

>>> endelse
>
>>> case horizontal_averaging of
>>> 0 : print,"Horizontal averaging required for detection: not
>>> applicable"
>>> 1 : print,"Horizontal averaging required for detection: 1/3 km"
>>> 2 : print,"Horizontal averaging required for detection: 1 km"
>>> 3 : print,"Horizontal averaging required for detection: 5 km"
>>> 4 : print,"Horizontal averaging required for detection: 20 km"
>>> 5 : print,"Horizontal averaging required for detection: 80 km"
>>> else : print,"*** error getting Horizontal averaging"
>>> endcase
>>> end
>
>> Rather than a string array, I would use an anonymous structure for
>> this sort of information:
>
>> e.g.
>> Data = {Type:"", Subtype:"", QA:"", Phase:"", PhaseQA:"", TypeQA:""} ;
>> etc
>> This creates a set of empty string fields.
>
>> then you can read in your strings like this:
>> data.type = 'cloud'
>
>> and retrieve your information like this:
>> help, data, /struct
>> IDL> ** Structure <ff8a18>, 6 tags, length=72, data length=72, refs=1:
>>  TYPE      STRING  'cloud'
>>  SUBTYPE    STRING  "
>>  QA         STRING  "
>>  PHASE      STRING  "
>>  PHASEQA    STRING  "
>>  TYPEQA     STRING  "
>
>> Reading in your strings could probably be done into a string array
>> either if you really want to:
>> Result = STRARR[n]
>> FOR i = 0, n-1 DO BEGIN
>> Result[i] = 'Cloud'
>> ENDFOR ; etc.
>
>> Also, you can probably use format codes to convert your bytes/integers
>> to set bits, something along the lines of:
>> CloudBits = STRING(CloudByte, FORMAT='(B0)')
>> NCloudBits = STRLEN(CloudBits)
>
>> Which I can imagine would make determining your string contents a bit

```

```

>> easier. I can't remember exactly, and I'm stuck with demo-mode only
>> today(!), so you'll have to play around with it yourself, unless
>> someone more knowledgeable jumps in with a fuller explanation.
>
>> Good luck!
>> Chris
>
> Hi Chris,
> Thanks for your quick reply.
> I am using this routine in my program "vfm_feature_flags,36282"
> This one is changing every time. Just I want to store o/p information
> into a string.
> Your program is simple, but I could not understand clearly. For
> example I want to store "Feature Subtype : cirrus (transparent) " and
> this one is changing every time. Some times cirrus and some times
> convective ..
>
> Could you give me little clearly then I can follow.
>
> Thanking you,
>
> Kishore

```

First, define your output structure to include every possible field
and subfield,
For example, after this:

```

feature_type = 0
feature_type_qa = 0
ice_water_phase = 0
ice_water_phase_qa = 0
feature_subtype = 0
; etc.

```

add this:

```

Output = { feature_type:"",    $
           feature_type_qa:"", $
           ice_water_phase:"", $
           ice_water_phase_qa:"", $
           feature_subtype:"", $
           ; etc.             , $
           }

```

Then, fill your output structure with strings wherever you've used
PRINT.
For example, change this:

```

case feature_type of
0 : print,"Feature Type : invalid (bad or missing data)"
1 : print,"Feature Type : clear air"
2 : begin
    print,"Feature Type : cloud"
    case feature_subtype of
    0 : print, "Feature Subtype : low overcast, transparent"
    1 : print, "Feature Subtype : low overcast, opaque"

```

To this:

```

case feature_type of
0 : Output.FeatureType = 'invalid (bad or missing data)'
1 : Output.FeatureType = 'clear air'
2 : begin
    Output.FeatureType = 'cloud'
    case feature_subtype of
    0 : Output.FeatureSubType = 'low overcast, transparent'
    1 : Output.FeatureSubType = 'low overcast, opaque'

```

...

HELP, Output, /STRUCTURE

RETURN, Output

END

This will give you an 'anonymous structure' (look it up in the documentaton), rather than a string array, as your output. To me it's a much more managable piece of information, though it takes slightly longer to get your head around how to juggle with it...

Plus, if you want you can probably concatenate all you structures into a structure array and you'll have one single output file for all your string data.

Regards,
Chris

Subject: Re: storing in array

Posted by [Spon](#) on Mon, 12 May 2008 11:18:51 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 12, 11:03 am, Spon <christoph.b...@gmail.com> wrote:

```

> Output = { feature_type:"",    $
>           feature_type_qa:"",  $
>           ice_water_phase:"",  $
>           ice_water_phase_qa:"", $
>           feature_subtype:"",  $

```



```

>         ; etc.         , $
>         }

> case feature_type of
> 0 : Output.FeatureType = 'invalid (bad or missing data)'
> 1 : Output.FeatureType = 'clear air'
> 2 : begin
>     Output.FeatureType = 'cloud'
>     case feature_subtype of
>     0 : Output.FeatureSubType = 'low overcast, transparent'
>     1 : Output.FeatureSubType = 'low overcast, opaque'

```

Obviously, your variable names have to match up (including structure field names), which they don't in my example, sorry.

```

> 0 : Output.FeatureType = 'invalid (bad or missing data)'
> 1 : Output.FeatureSubType = 'low overcast, opaque'

```

and so on
should be

```

> 0 : Output.Feature_Type = 'invalid (bad or missing data)'
> 1 : Output.Feature_SubType = 'low overcast, opaque'

```

to make my example work.

Subject: Re: storing in array
 Posted by [Spon](#) on Mon, 12 May 2008 14:09:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

On May 11, 12:48 am, kishore1...@gmail.com wrote:

```
> vfm_feature_flags,36282
```

Also, be careful: the above number is converted to a longword integer by IDL. This might cause headaches later on if you're expecting a 16-bit input. You could use:

```
    vfm_feature_flags,36282u
```

to ensure the number is defined as a 16-bit unsigned integer.

I would protect my code like this:

```
---
```

```
function vfm_feature_flags, input
```

```

; Check input
if input gt 65535L or input lt -32767L then $
    message, 'Input not a 16-bit flag.'

```

```

; Convert input to unsigned integer

```

```
val = uint(input)
```

You've already written a loop to convert your flag to sub-flags, so this may be superfluous, but I've had a dig around the format codes and come up with this:

```
; Convert to left-zero-padded 16-char string
bval = string(val, format = '(B016)')
```

```
; And then you can extract substrings to get
; the sub-flag values using ReadS:
ftflag = strmid(bval, 0, 3) ; Position 0, length 3
feature_type = 0u
reads, ftflag, format = '(B)', feature_type
```

```
ftqflag = strmid(bval, 3, 2) ; Position 3, length 2
feature_type_qa = 0u
reads, ftqflag, format = '(B)', feature_type_qa
; and so on.
```

I have to say, the fact that ReadS is a procedure and not a function, plus the fact that you can't pass type codes to it for your output does seem a bit counter-intuitive to me. Your FOR loop will in theory be slower, but probably not noticeably; and at least it's pretty easy to follow compared to messing about with format codes. But it can be done without the loop.

Regards,
Chris

Subject: Re: storing in array

Posted by [kishore1818](#) on Mon, 12 May 2008 15:35:26 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 12, 8:18 am, Spon <christoph.b...@gmail.com> wrote:

> On May 12, 11:03 am, Spon <christoph.b...@gmail.com> wrote:

>

```
>> Output = { feature_type:"",      $
>>           feature_type_qa:"",    $
>>           ice_water_phase:"",    $
>>           ice_water_phase_qa:"", $
>>           feature_subtype:"",    $
>>           ; etc.                , $
>>                                }
```

```

>> case feature_type of
>> 0 : Output.FeatureType = 'invalid (bad or missing data)'
>> 1 : Output.FeatureType = 'clear air'
>> 2 : begin
>>     Output.FeatureType = 'cloud'
>>     case feature_subtype of
>>     0 : Output.FeatureSubType = 'low overcast, transparent'
>>     1 : Output.FeatureSubType = 'low overcast, opaque'
>
> Obviously, your variable names have to match up (including structure
> field names), which they don't in my example, sorry.
>
>> 0 : Output.FeatureType = 'invalid (bad or missing data)'
>>     1 : Output.FeatureSubType = 'low overcast, opaque'
>
> and so on
> should be
>
>> 0 : Output.Feature_Type = 'invalid (bad or missing data)'
>>     1 : Output.Feature_SubType = 'low overcast, opaque'
>
> to make my example work.

```

Hi,
 Thanks for your suggestions, I modified according your suggestions.
 Pl. go through my program
 pro vfm_feature_flags1, val

```

; this routine demonstrates how to read and extract values from a
feature
; classification flag 16-bit integer value in CALIPSO Level 2
Vertical
; Feature Mask files
;
; INPUT:
; val - the feature classification flag value to be decoded
;
; OUTPUT:
; all information is printed into the IDL log window

```

print, val

```

feature_type = 0
feature_type_qa = 0
ice_water_phase = 0
ice_water_phase_qa = 0
feature_subtype = 0

```

```
cloud_aerosol_psc_type_qa = 0
horizontal_averaging = 0
```

```
Output = { feature_type:"",      $
           feature_type_qa:"",   $
           ice_water_phase:"",   $
           ice_water_phase_qa:"", $
           feature_subtype:"",   $
           cloud_aerosol_psc_type_qa:"", $
           horizontal_averaging:""}
```

```
for i=0,15 do begin
  if ((val and 2L^i) NE 0) then begin
    print,'Bit set: ',i+1
    case i+1 of
      1 : feature_type = feature_type + 1
      2 : feature_type = feature_type + 2
      3 : feature_type = feature_type + 4
      4 : feature_type_qa = feature_type_qa + 1
      5 : feature_type_qa = feature_type_qa + 2
      6 : ice_water_phase = ice_water_phase + 1
      7 : ice_water_phase = ice_water_phase + 2
      8 : ice_water_phase_qa = ice_water_phase_qa + 1
      9 : ice_water_phase_qa = ice_water_phase_qa + 2
     10 : feature_subtype = feature_subtype + 1
     11 : feature_subtype = feature_subtype + 2
     12 : feature_subtype = feature_subtype + 4
     13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
     14 : horizontal_averaging = horizontal_averaging + 1
     15 : horizontal_averaging = horizontal_averaging + 2
     16 : horizontal_averaging = horizontal_averaging + 4
    else:
      endcase
    endif
  endfor
```

```
case feature_type of
0 : Output.Feature_Type = 'invalid (bad or missing data)'
1 : Output.Feature_Type = 'clear air'
2 : begin
  Output.Feature_Type = 'cloud'
  case feature_subtype of
0 : Output.Feature_Subtype = 'low overcast, transparent'
1 : Output.Feature_Subtype = 'low overcast, opaque'
2 : Output.Feature_Subtype = 'transition stratocumulus'
3 : Output.Feature_Subtype = 'low, broken cumulus'
4 : Output.Feature_Subtype = 'altocumulus (transparent)'
5 : Output.Feature_Subtype = 'altostratus (opaque)'
```

```

    6 : Output.Feature_Subtype = 'cirrus (transparent)'
    7 : Output.Feature_Subtype = 'deep convective (opaque)'
    else : Output,'*** error getting Feature Subtype'
    endcase
end
3 : begin
    Output.Feature_Type = 'aerosol'
    case feature_subtype of
    0 : Output.Feature_Subtype = 'not determined'
    1 : Output.Feature_Subtype = 'clean marine'
    2 : Output.Feature_Subtype = 'dust'
    3 : Output.Feature_Subtype = 'polluted continental'
    4 : Output.Feature_Subtype = 'clean continental'
    5 : Output.Feature_Subtype = 'polluted dust'
    6 : Output.Feature_Subtype = 'smoke'
    7 : Output.Feature_Subtype = 'other'
    else : Output,'*** error getting Feature Subtype'
    endcase
end
4 : begin
    Output.Feature_Type = 'stratospheric feature--PSC or
stratospheric aerosol'
    case feature_subtype of
    0 : Output.Feature_Subtype = 'not determined'
    1 : Output.Feature_Subtype = 'non-depolarizing PSC'
    2 : Output.Feature_Subtype = 'depolarizing PSC'
    3 : Output.Feature_Subtype = 'non-depolarizing aerosol'
    4 : Output.Feature_Subtype = 'depolarizing aerosol'
    5 : Output.Feature_Subtype = 'spare'
    6 : Output.Feature_Subtype = 'spare'
    7 : Output.Feature_Subtype = 'other'
    else : Output,'*** error getting Feature Subtype'
    endcase
end
5 : Output.Feature_Type = 'surface'
6 : Output.Feature_Type = 'subsurface'
7 : Output.Feature_Type = 'no signal (totally attenuated)'
else : Output,'*** error getting Feature Type'
endcase

case feature_type_qa of
0 : Output.Feature_Type_QA = 'none'
1 : Output.Feature_Type_QA = 'low'
2 : Output.Feature_Type_QA = 'medium'
3 : Output.Feature_Type_QA = 'high'
else : Output,'*** error getting Feature Type QA'
endcase

```

```

case ice_water_phase of
0 : Output.Ice_Water_Phase = 'unknown/not determined'
1 : Output.Ice_Water_Phase = 'ice'
2 : Output.Ice_Water_Phase = 'water'
3 : Output.Ice_Water_Phase = 'mixed phase'
else : Output, '*** error getting Ice/Water Phase'
endcase

```

```

case ice_water_phase_qa of
0 : Output.Ice_Water_Phase_QA = 'none'
1 : Output.Ice_Water_Phase_QA = 'low'
2 : Output.Ice_Water_Phase_QA = 'medium'
3 : Output.Ice_Water_Phase_QA = 'high'
else : Output, '*** error getting Ice/Water Phase QA'
endcase

```

```

if (cloud_aerosol_psc_type_qa eq 0) then begin
    Output.Cloud_Aerosol_PSC_Type_QA = 'not confident'
endif else begin
    Output.Cloud_Aerosol_PSC_Type_QA = 'confident'
endelse

```

```

case horizontal_averaging of
0 : Output.Horizontal_averaging = 'not applicable'
1 : Output.Horizontal_averaging = '1/3 km'
2 : Output.Horizontal_averaging = '1 km'
3 : Output.Horizontal_averaging = '5 km'
4 : Output.Horizontal_averaging = '20 km'
5 : Output.Horizontal_averaging = '80 km'
else : Output, '*** error getting Horizontal averaging'
endcase
help, Output, /structure
;Return, Output
end

```

Using above routine I wrote a small program
data1=[44474, 36282, 28602, 28090]

```

for i=0,3 do begin
    ;vfm_feature_flags,data1(i)
    vfm_feature_flags1,data1(i)
    pause=" & read,pause

```

```

endfor
end

```

Output

```
FEATURE_SUBTYPE STRING 'cirrus (transparent)'
FEATURE_SUBTYPE STRING 'cirrus (transparent)'
FEATURE_SUBTYPE STRING 'deep convective (opaque)'
FEATURE_SUBTYPE STRING 'cirrus (transparent)'
```

Only, problem is how to recall output parameters.
For example, I want to recall all output.FEATURE_SUBTYPE.
These output I have to calculate how many cirrus and deep convective
system around the globe at each day at different Longitude latitude
bands.

Once again I am very much thankful to you.

Regards,

Kishore

Subject: Re: storing in array
Posted by [Spon](#) on Mon, 12 May 2008 16:06:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

On May 12, 4:35 pm, kishore1...@gmail.com wrote:

>
> Hi,
> Thanks for your suggestions, I modified according your suggestions.
> Pl. go through my program

> pro vfm_feature_flags1,val
FUNCTION VFM_FEATURE_FLAGS1, Input

>
> ; this routine demonstrates how to read and extract values from a
> feature
> ; classification flag 16-bit integer value in CALIPSO Level 2
> Vertical
> ; Feature Mask files
> ;
> ; INPUT:
> ; val - the feature classification flag value to be decoded
> ;
> ; OUTPUT:
> ; all information is printed into the IDL log window

; Make sure input is usable
IF Input GT 65535L OR Input LT -32767L THEN \$
MESSAGE, 'Input not a 16-bit flag.'

```
; Convert input to 16-bit value
Val = UINT(Input)
```

```
> print, val
>
> feature_type = 0
> feature_type_qa = 0
> ice_water_phase = 0
> ice_water_phase_qa = 0
> feature_subtype = 0
> cloud_aerosol_psc_type_qa = 0
> horizontal_averaging = 0
>
> Output = { feature_type:"",          $
>           feature_type_qa:"",        $
>           ice_water_phase:"",        $
>           ice_water_phase_qa:"",     $
>           feature_subtype:"",        $
>           cloud_aerosol_psc_type_qa:"", $
>           horizontal_averaging:""}
>
> for i=0,15 do begin
>   if ((val and 2L^i) NE 0) then begin
>     print,'Bit set: ',i+1
>     case i+1 of
>       1 : feature_type = feature_type + 1
>       2 : feature_type = feature_type + 2
>       3 : feature_type = feature_type + 4
>       4 : feature_type_qa = feature_type_qa + 1
>       5 : feature_type_qa = feature_type_qa + 2
>       6 : ice_water_phase = ice_water_phase + 1
>       7 : ice_water_phase = ice_water_phase + 2
>       8 : ice_water_phase_qa = ice_water_phase_qa + 1
>       9 : ice_water_phase_qa = ice_water_phase_qa + 2
>      10 : feature_subtype = feature_subtype + 1
>      11 : feature_subtype = feature_subtype + 2
>      12 : feature_subtype = feature_subtype + 4
>      13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
>      14 : horizontal_averaging = horizontal_averaging + 1
>      15 : horizontal_averaging = horizontal_averaging + 2
>      16 : horizontal_averaging = horizontal_averaging + 4
>     else:
>       endcase
>   endif
> endfor
>
> case feature_type of
```



```

> 0 : Output.Feature_Type = 'invalid (bad or missing data)'
> 1 : Output.Feature_Type = 'clear air'
> 2 : begin
>     Output.Feature_Type = 'cloud'
>     case feature_subtype of
>     0 : Output.Feature_Subtype = 'low overcast, transparent'
>     1 : Output.Feature_Subtype = 'low overcast, opaque'
>     2 : Output.Feature_Subtype = 'transition stratocumulus'
>     3 : Output.Feature_Subtype = 'low, broken cumulus'
>     4 : Output.Feature_Subtype = 'altocumulus (transparent)'
>     5 : Output.Feature_Subtype = 'altostratus (opaque)'
>     6 : Output.Feature_Subtype = 'cirrus (transparent)'
>     7 : Output.Feature_Subtype = 'deep convective (opaque)'
>     else : Output, '*** error getting Feature Subtype'
ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'

>     endcase
> end
> 3 : begin
>     Output.Feature_Type = 'aerosol'
>     case feature_subtype of
>     0 : Output.Feature_Subtype = 'not determined'
>     1 : Output.Feature_Subtype = 'clean marine'
>     2 : Output.Feature_Subtype = 'dust'
>     3 : Output.Feature_Subtype = 'polluted continental'
>     4 : Output.Feature_Subtype = 'clean continental'
>     5 : Output.Feature_Subtype = 'polluted dust'
>     6 : Output.Feature_Subtype = 'smoke'
>     7 : Output.Feature_Subtype = 'other'
>     else : Output, '*** error getting Feature Subtype'
ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'

>     endcase
> end
> 4 : begin
>     Output.Feature_Type = 'stratospheric feature--PSC or
> stratospheric aerosol'
>     case feature_subtype of
>     0 : Output.Feature_Subtype = 'not determined'
>     1 : Output.Feature_Subtype = 'non-depolarizing PSC'
>     2 : Output.Feature_Subtype = 'depolarizing PSC'
>     3 : Output.Feature_Subtype = 'non-depolarizing aerosol'
>     4 : Output.Feature_Subtype = 'depolarizing aerosol'
>     5 : Output.Feature_Subtype = 'spare'
>     6 : Output.Feature_Subtype = 'spare'
>     7 : Output.Feature_Subtype = 'other'
>     else : Output, '*** error getting Feature Subtype'
ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'

```

```

>     endcase
> end
> 5 : Output.Feature_Type = 'surface'
> 6 : Output.Feature_Type = 'subsurface'
> 7 : Output.Feature_Type = 'no signal (totally attenuated)'
> else : Output,'*** error getting Feature Type'
ELSE : Output.Feature_Type = '*** error getting Feature Type'

> endcase
>
> case feature_type_qa of
> 0 : Output.Feature_Type_QA = 'none'
> 1 : Output.Feature_Type_QA = 'low'
> 2 : Output.Feature_Type_QA = 'medium'
> 3 : Output.Feature_Type_QA = 'high'
> else : Output,'*** error getting Feature Type QA'
ELSE : Output.Feature_Type_QA = '*** error getting Feature Type QA'

> endcase
>
> case ice_water_phase of
> 0 : Output.Ice_Water_Phase = 'unknown/not determined'
> 1 : Output.Ice_Water_Phase = 'ice'
> 2 : Output.Ice_Water_Phase = 'water'
> 3 : Output.Ice_Water_Phase = 'mixed phase'
> else : Output,'*** error getting Ice/Water Phase'
ELSE : Output.Ice_Water_Phase = '*** error getting Ice/Water Phase'

> endcase
>
> case ice_water_phase_qa of
> 0 : Output.Ice_Water_Phase_QA = 'none'
> 1 : Output.Ice_Water_Phase_QA = 'low'
> 2 : Output.Ice_Water_Phase_QA = 'medium'
> 3 : Output.Ice_Water_Phase_QA = 'high'
> else : Output,'*** error getting Ice/Water Phase QA'
ELSE : Output.Ice_Water_Phase_QA = '*** error getting Ice/Water Phase
QA'

> endcase
>
> if (cloud_aerosol_psc_type_qa eq 0) then begin
>   Output.Cloud_Aerosol_PSC_Type_QA = 'not confident'
> endif else begin
>   Output.Cloud_Aerosol_PSC_Type_QA = 'confident'
> endelse
>

```

```

> case horizontal_averaging of
> 0 : Output.Horizontal_averaging = 'not applicable'
> 1 : Output.Horizontal_averaging = '1/3 km'
> 2 : Output.Horizontal_averaging = '1 km'
> 3 : Output.Horizontal_averaging = '5 km'
> 4 : Output.Horizontal_averaging = '20 km'
> 5 : Output.Horizontal_averaging = '80 km'
> else : Output,'*** error getting Horizontal averaging'
ELSE : Output.Ice_Water_Phase = '*** error getting Ice/Water Phase'

```

```

> endcase
> help,Output,/structure
> ;Return, Output
RETURN, Output
> end

```

```

> Using above routine I wrote a small program
> data1=[44474, 36282, 28602, 28090]
>
> for i=0,3 do begin
>   ;vfm_feature_flags,data1(i)
>   vfm_feature_flags1,data1(i)
>   pause=" & read,pause
>
> endfor
> end
>

```

> Only, problem is how to recall output parameters.

Replace this very first line:
 pro vfm_feature_flags,val

with

```
function vfm_feature_flags,val
```

then, down at the very end of the programme, remove the comment-out
 from this line
 ;Return, Output

IDL has two types of programmes, 'procedures' and 'functions'.
 Functions will return an output. Procedures can only alter input
 parameters (although this can be used to effectively give output too).
 Functions are called slightly differently to procedures, though the

code language in them is identical apart from the 'Return, A_Value' thing.

To call the function, use:

```
result = vfm_feature_flags(value)
```

e.g.

```
myoutput = vfm_feature_flags(36282u)
```

Then you can look at the content of the output like this:

```
help, myoutput, /structure
```

or something like this:

```
print, myoutput.feature_type
```

to get a single output field from the structure.

Regards,
Chris

Subject: Re: storing in array

Posted by [kishore1818](#) on Mon, 12 May 2008 17:29:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 12, 1:06 pm, Spon <christoph.b...@gmail.com> wrote:

> On May 12, 4:35 pm, kishore1...@gmail.com wrote:

>

>

>

>> Hi,

>> Thanks for your suggestions, I modified according your suggestions.

>> Pl. go through my program

>> pro vfm_feature_flags1,val

>

> FUNCTION VFM_FEATURE_FLAGS1, Input

>

>

>

>> ; this routine demonstrates how to read and extract values from a
>> feature

>> ; classification flag 16-bit integer value in CALIPSO Level 2

>> Vertical

>> ; Feature Mask files

>> ;

>> ; INPUT:

>> ; val - the feature classification flag value to be decoded

>> ;

>> ; OUTPUT:

```

>> ; all information is printed into the IDL log window
>
> ; Make sure input is usable
> IF Input GT 65535L OR Input LT -32767L THEN $
>   MESSAGE, 'Input not a 16-bit flag.'
>
> ; Convert input to 16-bit value
> Val = UINT(Input)
>
>
>
>> print, val
>
>> feature_type = 0
>> feature_type_qa = 0
>> ice_water_phase = 0
>> ice_water_phase_qa = 0
>> feature_subtype = 0
>> cloud_aerosol_psc_type_qa = 0
>> horizontal_averaging = 0
>
>> Output = { feature_type:"",          $
>>             feature_type_qa:"",      $
>>             ice_water_phase:"",      $
>>             ice_water_phase_qa:"",   $
>>             feature_subtype:"",      $
>>             cloud_aerosol_psc_type_qa:"", $
>>             horizontal_averaging:""}
>
>> for i=0,15 do begin
>>   if ((val and 2L^i) NE 0) then begin
>>     print,'Bit set: ',i+1
>>     case i+1 of
>>       1 : feature_type = feature_type + 1
>>       2 : feature_type = feature_type + 2
>>       3 : feature_type = feature_type + 4
>>       4 : feature_type_qa = feature_type_qa + 1
>>       5 : feature_type_qa = feature_type_qa + 2
>>       6 : ice_water_phase = ice_water_phase + 1
>>       7 : ice_water_phase = ice_water_phase + 2
>>       8 : ice_water_phase_qa = ice_water_phase_qa + 1
>>       9 : ice_water_phase_qa = ice_water_phase_qa + 2
>>      10 : feature_subtype = feature_subtype + 1
>>      11 : feature_subtype = feature_subtype + 2
>>      12 : feature_subtype = feature_subtype + 4
>>      13 : cloud_aerosol_psc_type_qa = cloud_aerosol_psc_type_qa + 1
>>      14 : horizontal_averaging = horizontal_averaging + 1
>>      15 : horizontal_averaging = horizontal_averaging + 2

```

```

>> 16: horizontal_averaging = horizontal_averaging + 4
>> else:
>> endcase
>> endif
>> endfor
>
>> case feature_type of
>> 0 : Output.Feature_Type = 'invalid (bad or missing data)'
>> 1 : Output.Feature_Type = 'clear air'
>> 2 : begin
>>     Output.Feature_Type = 'cloud'
>>     case feature_subtype of
>>     0 : Output.Feature_Subtype = 'low overcast, transparent'
>>     1 : Output.Feature_Subtype = 'low overcast, opaque'
>>     2 : Output.Feature_Subtype = 'transition stratocumulus'
>>     3 : Output.Feature_Subtype = 'low, broken cumulus'
>>     4 : Output.Feature_Subtype = 'altocumulus (transparent)'
>>     5 : Output.Feature_Subtype = 'altostratus (opaque)'
>>     6 : Output.Feature_Subtype = 'cirrus (transparent)'
>>     7 : Output.Feature_Subtype = 'deep convective (opaque)'
>>     else : Output, '*** error getting Feature Subtype'
>>
> ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'
>
>> endcase
>> end
>> 3 : begin
>>     Output.Feature_Type = 'aerosol'
>>     case feature_subtype of
>>     0 : Output.Feature_Subtype = 'not determined'
>>     1 : Output.Feature_Subtype = 'clean marine'
>>     2 : Output.Feature_Subtype = 'dust'
>>     3 : Output.Feature_Subtype = 'polluted continental'
>>     4 : Output.Feature_Subtype = 'clean continental'
>>     5 : Output.Feature_Subtype = 'polluted dust'
>>     6 : Output.Feature_Subtype = 'smoke'
>>     7 : Output.Feature_Subtype = 'other'
>>     else : Output, '*** error getting Feature Subtype'
>>
> ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'
>
>> endcase
>> end
>> 4 : begin
>>     Output.Feature_Type = 'stratospheric feature--PSC or
>> stratospheric aerosol'
>>     case feature_subtype of
>>     0 : Output.Feature_Subtype = 'not determined'

```

```

>> 1 : Output.Feature_Subtype = 'non-depolarizing PSC'
>> 2 : Output.Feature_Subtype = 'depolarizing PSC'
>> 3 : Output.Feature_Subtype = 'non-depolarizing aerosol'
>> 4 : Output.Feature_Subtype = 'depolarizing aerosol'
>> 5 : Output.Feature_Subtype = 'spare'
>> 6 : Output.Feature_Subtype = 'spare'
>> 7 : Output.Feature_Subtype = 'other'
>> else : Output,'*** error getting Feature Subtype'
>
> ELSE : Output.Feature_Subtype = '*** error getting Feature Subtype'
>
>> endcase
>> end
>> 5 : Output.Feature_Type = 'surface'
>> 6 : Output.Feature_Type = 'subsurface'
>> 7 : Output.Feature_Type = 'no signal (totally attenuated)'
>> else : Output,'*** error getting Feature Type'
>
> ELSE : Output.Feature_Type = '*** error getting Feature Type'
>
>> endcase
>
>> case feature_type_qa of
>> 0 : Output.Feature_Type_QA = 'none'
>> 1 : Output.Feature_Type_QA = 'low'
>> 2 : Output.Feature_Type_QA = 'medium'
>> 3 : Output.Feature_Type_QA = 'high'
>> else : Output,'*** error getting Feature Type QA'
>
> ELSE : Output.Feature_Type_QA = '*** error getting Feature Type QA'
>
>> endcase
>
>> case ice_water_phase of
>> 0 : Output.Ice_Water_Phase = 'unknown/not determined'
>> 1 : Output.Ice_Water_Phase = 'ice'
>> 2 : Output.Ice_Water_Phase = 'water'
>> 3 : Output.Ice_Water_Phase = 'mixed phase'
>> else : Output,'*** error getting Ice/Water Phase'
>
> ELSE : Output.Ice_Water_Phase = '*** error getting Ice/Water Phase'
>
>> endcase
>
>> case ice_water_phase_qa of
>> 0 : Output.Ice_Water_Phase_QA = 'none'
>> 1 : Output.Ice_Water_Phase_QA = 'low'
>> 2 : Output.Ice_Water_Phase_QA = 'medium'

```

```

>> 3 : Output.Ice_Water_Phase_QA = 'high'
>> else : Output,'*** error getting Ice/Water Phase QA'
>
> ELSE : Output.Ice_Water_Phase_QA = '*** error getting Ice/Water Phase
> QA'
>
>
>
>> endcase
>
>> if (cloud_aerosol_psc_type_qa eq 0) then begin
>>   Output.Cloud_Aerosol_PSC_Type_QA = 'not confident'
>> endif else begin
>>   Output.Cloud_Aerosol_PSC_Type_QA = 'confident'
>> endelse
>
>> case horizontal_averaging of
>> 0 : Output.Horizontal_averaging = 'not applicable'
>> 1 : Output.Horizontal_averaging = '1/3 km'
>> 2 : Output.Horizontal_averaging = '1 km'
>> 3 : Output.Horizontal_averaging = '5 km'
>> 4 : Output.Horizontal_averaging = '20 km'
>> 5 : Output.Horizontal_averaging = '80 km'
>> else : Output,'*** error getting Horizontal averaging'
>
> ELSE : Output.Ice_Water_Phase = '*** error getting Ice/Water Phase'
>
>
>
>
>> endcase
>> help,Output,/structure
>> ;Return, Output
> RETURN, Output
>> end
>> Using above routine I wrote a small program
>> data1=[44474, 36282, 28602, 28090]
>
>> for i=0,3 do begin
>>   ;vfm_feature_flags,data1(i)
>>   vfm_feature_flags1,data1(i)
>>   pause=" & read,pause
>
>> endfor
>> end
>
>> Only, problem is how to recall output parameters.

```


>
> Replace this very first line:
> pro vfm_feature_flags,val
>
> with
>
> function vfm_feature_flags,val
>
> then, down at the very end of the programme, remove the comment-out
> from this line
> ;Return, Output
>
> IDL has two types of programmes, 'procedures' and 'functions'.
> Functions will return an output. Procedures can only alter input
> parameters (although this can be used to effectively give output too).
> Functions are called slightly differently to procedures, though the
> code language in them is identical apart from the 'Return, A_Value'
> thing.
>
> To call the function, use:
> result = vfm_feature_flags(value)
>
> e.g.
> myoutput = vfm_feature_flags(36282u)
>
> Then you can look at the content of the output like this:
> help, myoutput, /structure
>
> or something like this:
> print, myoutput.feature_type
> to get a single output field from the structure.
>
> Regards,
> Chris

Hi Chris,

Now, its working fine, thanks a lot for your kind help.
This forum is very good for new IDL guys.

Regards,

Kishore
