Subject: Re: axis problem
Posted by David Fanning on Sat, 19 Jul 2003 18:44:28 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> I am not sure if this is a bug.
>
> If I use axis with _extra xticks=0 the idl automatic is used to draw quite
> good major ticks.
> But if I use axis with _extra (xticks=4,xminor=0) the major ticks are the
> same as the other example but no automatic for minor ticks.
> The same result gives axis with _extra (xticks=4,xminor=1)

I don't thing this is a bug. The documentation describes
the MINOR keyword as the number of minor tick *intervals*,
not tick marks. I wouldn't expect any visual difference between
an axis with 0 minor tick intervals and 1 minor tick interval
between major tick marks. They should look exactly the same,
as they do in this example.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: axis problem
Posted by R.Bauer on Sun, 20 Jul 2003 07:52:02 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Reimar Bauer writes:
>
>> I am not sure if this is a bug.
>>
>> If I use axis with _extra xticks=0 the idl automatic is used to draw
>> quite good major ticks.
>> But if I use axis with _extra (xticks=4,xminor=0) the major ticks are the
>> same as the other example but no automatic for minor ticks.
>> The same result gives axis with _extra (xticks=4,xminor=1)

>
> I don't thing this is a bug. The documentation describes
> the MINOR keyword as the number of minor tick *intervals*,
> not tick marks. I wouldn't expect any visual difference between
> an axis with 0 minor tick intervals and 1 minor tick interval
> between major tick marks. They should look exactly the same,
> as they do in this example.
>
> Cheers,
>
> David
>

But if I don't use _extra it looks different.

I should have submitted this example too. Then xminor=0 looks different.

```
pro test_axis_minor_error
erase
window,0
tek_color
plot,findgen(15),findgen(15),/nodata,xstyle=5,ystyle=5

axis,xaxis=0,xticks=4,xminor=0,color=2
axis,xaxis=1,xticks=4,xminor=1,color=2

axis,yaxis=0,yticks=4,yminor=0,color=2
axis,yaxis=1,yticks=4,yminor=1,color=2

end
```

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: axis problem
Posted by David Fanning on Sun, 20 Jul 2003 15:26:18 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> But if I don't use _extra it looks different.

>
> I should have submitted this example too. Then xminor=0 looks different.
>
> pro test_axis_minor_error
> erase
> window,0
> tek_color
> plot,findgen(15),findgen(15),/nodata,xstyle=5,ystyle=5
>
> axis,xaxis=0,xticks=4,xminor=0,color=2
> axis,xaxis=1,xticks=4,xminor=1,color=2
>
> axis,yaxis=0,yticks=4,yminor=0,color=2
> axis,yaxis=1,yticks=4,yminor=1,color=2
>
> end

Ah, yes. I see now. Well, I *still* don't think this
is a bug. Here is why.

With *most* IDL system variables (the ![XYZ].MARGIN
system variable is an exception), setting the system
variable to 0 is equivalent to setting it to it's
default value. This is good, because otherwise
you couldn't ever restore the system variable to its
default value without knowing what it was.

I would argue that the MINOR keyword is a local way
of setting the ![XYZ].MINOR system variable. (Or something
like that. I don't really know what goes on under the hood.)
In any case, setting MINOR=0 is equivalent to saying to IDL
"do whatever the default thing is for minor tick marks". In
this case, you get the usual five tick intervals.

But when you put MINOR=0 into an _EXTRA keyword this
"default" behavior mechanism is circumvented, and the
MINOR keyword then is treated in a more literal sense.
I think this is proper and correct. You certainly can't
expect IDL to "process" keywords in an _EXTRA structure,
since finding the keywords would be a monumental task
and would slow IDL down terribly (for one thing, you
would have to process numerous spellings of the keyword,
since keywords can be shortened to their shortest
unambiguous spelling).

I think this is a case (there are many) in which
IDL has been programmed in a far-sighted and sensitive
way, and that it is doing *exactly* what it is

suppose to be doing: making life easier for the
programmer. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: axis problem
Posted by R.Bauer on Sun, 20 Jul 2003 16:23:54 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Reimar Bauer writes:
>
>
>> But if I don't use _extra it looks different.
>>
>> I should have submitted this example too. Then xminor=0 looks different.
>>
>> pro test_axis_minor_error
>> erase
>> window,0
>> tek_color
>> plot,findgen(15),findgen(15),/nodata,xstyle=5,ystyle=5
>>
>> axis,xaxis=0,xticks=4,xminor=0,color=2
>> axis,xaxis=1,xticks=4,xminor=1,color=2
>>
>> axis,yaxis=0,yticks=4,yminor=0,color=2
>> axis,yaxis=1,yticks=4,yminor=1,color=2
>>
>> end
>
>
> Ah, yes. I see now. Well, I *still* don't think this
> is a bug. Here is why.
>
> With *most* IDL system variables (the ![XYZ].MARGIN
> system variable is an exception), setting the system
> variable to 0 is equivalent to setting it to it's

> *default* value. This is good, because otherwise
> you couldn't ever restore the system variable to its
> default value without knowing what it was.
>
> I would argue that the MINOR keyword is a local way
> of setting the ![XYZ].MINOR system variable. (Or something
> like that. I don't really know what goes on under the hood.)
> In any case, setting MINOR=0 is equivalent to saying to IDL
> "do whatever the default thing is for minor tick marks". In
> this case, you get the usual five tick intervals.
>
> But when you put MINOR=0 into an _EXTRA keyword this
> "default" behavior mechanism is circumvented, and the
> MINOR keyword then is treated in a more literal sense.
> I think this is proper and correct. You certainly can't
> expect IDL to "process" keywords in an _EXTRA structure,
> since finding the keywords would be a monumental task
> and would slow IDL down terribly (for one thing, you
> would have to process numerous spellings of the keyword,
> since keywords can be shortened to their shortest
> unambiguous spelling).
>
> I think this is a case (there are many) in which
> IDL has been programmed in a far-sighted and sensitive
> way, and that it is doing *exactly* what it is
> suppose to be doing: making life easier for the
> programmer. :-)
>
> Cheers,
>
> David
>

David I can't agree

It can't be that the user program has to test what keywords values are set as default by a routine and if it has this value then it must be killed from the _extra structure if it is there.

With all the other keywords it works as supposed. It would be very bad if it is somewhere defined that's the user is not able to pass default values by _extra. It must be possible to switch back to the whatever default value by submitting 0 for example.

At the moment I believe there is a bug with the querying of xyz minor. They used keyword_set() instead of n_elements() and ...

Is someone out there with a version before 5.0 and could test the examples?

regards
Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
---------- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
========== =======

Subject: Re: axis problem
Posted by David Fanning on Sun, 20 Jul 2003 17:08:58 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> David I can't agree
>
> It can't be that the user program has to test what keywords values are
> set as default by a routine and if it has this value then it must be
> killed from the _extra structure if it is there.
>
> With all the other keywords it works as supposed. It would be very bad
> if it is somewhere defined that's the user is not able to pass default
> values by _extra. It must be possible to switch back to the whatever
> default value by submitting 0 for example.
>
> At the moment I believe there is a bug with the querying of xyz minor.
> They used keyword_set() instead of n_elements() and ...

I'm not so sure of my answer that I would bet a
whole lot of money on the "no bug" theory, but still...

Think of how you would do this. A keyword has a value of 5 by
default. If the keyword is set to 0, you which to set the
value to 5. You would write the program like this:

```
  PRO MyPlot, KEY=key, _Extra=extra
  IF N_Elements(key) EQ 0 THEN key = 5
  IF key EQ 0 THEN key = 5

  PLOTSOMETHING, Key=key, _Extra=extra
  END
```

Now, if you pass a value in with the keyword, you encounter
the "processing".

```
  IDL> MyPlot, KEY=0
```

If you pass it in via the _EXTRA mechanism, you bypass
the processing:

```
  IDL> MyPlot, _Extra={KEY:0}
```

This seems quite reasonable to me. The alternative would
be to put something like this into your program:

```
 IF N_ELements(extra) NE 0 THEN BEGIN
   tagnames = Tag_Names(extra)
   index = WHERE(tagnames EQ 'K', count)
   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
   index = WHERE(tagnames EQ 'KE', count)
   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
   index = WHERE(tagnames EQ 'KEY', count)
```

```
   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
  ENDIF
```

Think what would happen if you wrote a long keyword name,
or if you had multiple keywords defined that you had to
chase down like this. You would spend all your time writing
code and no time at all drinking beer. :-(

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: axis problem
Posted by R.Bauer on Sun, 20 Jul 2003 20:42:36 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Reimar Bauer writes:
>
>> David I can't agree
>>
>> It can't be that the user program has to test what keywords values are
>> set as default by a routine and if it has this value then it must be
>> killed from the _extra structure if it is there.
>>
>> With all the other keywords it works as supposed. It would be very bad
>> if it is somewhere defined that's the user is not able to pass default
>> values by _extra. It must be possible to switch back to the whatever
>> default value by submitting 0 for example.
>>
>> At the moment I believe there is a bug with the querying of xyz minor.
>> They used keyword_set() instead of n_elements() and ...
>
> I'm not so sure of my answer that I would bet a
> whole lot of money on the "no bug" theory, but still...
>
> Think of how you would do this. A keyword has a value of 5 by
> default. If the keyword is set to 0, you which to set the
> value to 5. You would write the program like this:
>

>   PRO MyPlot, KEY=key, _Extra=extra
>   IF N_Elements(key) EQ 0 THEN key = 5
>   IF key EQ 0 THEN key = 5
>
>   PLOTSOMETHING, Key=key, _Extra=extra
>   END
>
> Now, if you pass a value in with the keyword, you encounter
> the "processing".
>
>   IDL> MyPlot, KEY=0
>
> If you pass it in via the _EXTRA mechanism, you bypass
> the processing:
>
>   IDL> MyPlot, _Extra={KEY:0}
>
> This seems quite reasonable to me. The alternative would
> be to put something like this into your program:
>
>   IF N_ELements(extra) NE 0 THEN BEGIN
>   tagnames = Tag_Names(extra)
>   index = WHERE(tagnames EQ 'K', count)
>   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
>   index = WHERE(tagnames EQ 'KE', count)
>   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
>   index = WHERE(tagnames EQ 'KEY', count)
>   IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5
>   ENDIF
>
> Think what would happen if you wrote a long keyword name,
> or if you had multiple keywords defined that you had to
> chase down like this. You would spend all your time writing
> code and no time at all drinking beer. :-(
>
> Cheers,
>
> David

Dear David,

I think they have probably written a function like this,

FUNCTION is_keyword,keyword,names

ix=where(strpos(names,keyword) eq 0 ,count_ix)
if count_ix eq 1 then return,names[ix]$
 else message,'Ambiguous keyword abbreviation '+keyword,/cont

end

And with routine_info you get the names of the keywords of the routine
But I can't do this myself with plot

print,is_keyword('xmin',['xminor','xaxis'])
xminor
print,is_keyword('x',['xminor','xaxis'])
print,is_keyword('x',['xminor','xaxis'])
% IS_KEYWORD: Ambiguous keyword abbreviation x
       0

Let us have the next beer

Prost

Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 ========== ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: axis problem
Posted by JD Smith on Mon, 21 Jul 2003 15:25:45 GMT
View Forum Message <> Reply to Message

On Sun, 20 Jul 2003 13:42:36 -0700, Reimar Bauer wrote:

> David Fanning wrote:
>
>> Reimar Bauer writes:
>>
>>> David I can't agree
>>>
>>> It can't be that the user program has to test what keywords values are
>>> set as default by a routine and if it has this value then it must be
>>> killed from the _extra structure if it is there.
>>>
>>> With all the other keywords it works as supposed. It would be very bad
>>> if it is somewhere defined that's the user is not able to pass default
>>> values by _extra. It must be possible to switch back to the whatever

>>> default value by submitting 0 for example.
>>>
>>> At the moment I believe there is a bug with the querying of xyz minor.
>>> They used keyword_set() instead of n_elements() and ...
>>
>> I'm not so sure of my answer that I would bet a whole lot of money on
>> the "no bug" theory, but still...
>>
>> Think of how you would do this. A keyword has a value of 5 by default.
>> If the keyword is set to 0, you which to set the value to 5. You would
>> write the program like this:
>>
>>   PRO MyPlot, KEY=key, _Extra=extra
>>   IF N_Elements(key) EQ 0 THEN key = 5
>>   IF key EQ 0 THEN key = 5
>>
>>   PLOTSOMETHING, Key=key, _Extra=extra
>>   END
>>
>> Now, if you pass a value in with the keyword, you encounter the
>> "processing".
>>
>>   IDL> MyPlot, KEY=0
>>
>> If you pass it in via the _EXTRA mechanism, you bypass the processing:
>>
>>   IDL> MyPlot, _Extra={KEY:0}
>>
>> This seems quite reasonable to me. The alternative would be to put
>> something like this into your program:
>>
>>   IF N_ELements(extra) NE 0 THEN BEGIN
>>   tagnames = Tag_Names(extra)
>>   index = WHERE(tagnames EQ 'K', count) IF count GT 0 THEN IF
>>   extra.(index) EQ 0 THEN key = 5 index = WHERE(tagnames EQ 'KE',
>>   count) IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5 index
>>   = WHERE(tagnames EQ 'KEY', count) IF count GT 0 THEN IF
>>   extra.(index) EQ 0 THEN key = 5
>>   ENDIF
>>
>> Think what would happen if you wrote a long keyword name, or if you had
>> multiple keywords defined that you had to chase down like this. You
>> would spend all your time writing code and no time at all drinking
>> beer. :-(
>>
>> Cheers,
>>
>> David

>
> Dear David,
>
> I think they have probably written a function like this,
>
> FUNCTION is_keyword,keyword,names
>
> ix=where(strpos(names,keyword) eq 0 ,count_ix) if count_ix eq 1 then
> return,names[ix]$
> else message,'Ambiguous keyword abbreviation '+keyword,/cont
>
> end
>
> And with routine_info you get the names of the keywords of the routine
> But I can't do this myself with plot
>
> print,is_keyword('xmin',['xminor','xaxis']) xminor
> print,is_keyword('x',['xminor','xaxis'])
> print,is_keyword('x',['xminor','xaxis']) % IS_KEYWORD: Ambiguous keyword
> abbreviation x
> 0
>

That's quite a creative set of theories, but the explanation is much
simpler:

Compare:

axis,xaxis=0,xticks=4,color=2,xminor=0

and

axis,xaxis=0,xticks=4,color=2,xminor=0,xstyle=1

XSTYLE=1, combined with XTICKS=4, is the source of your trouble.
You're forcing IDL to divide the axis range into non-whole units,
which means that the minor units cannot be simply specified. Rather
than approximate minor tick positions, IDL just skips minor ticks
altogether. You can specify the minor tickmark count yourself with,
e.g., XMINOR=5, to get the effect you want. You might consider this a
bug that IDL doesn't do this for you. I'd probably just drop the
XTICKS specification.

The inherited vs. direct issue was a red herring, since you didn't
compare apples to apples, specifying many more parameters in the
_EXTRA version. As far as I know, all _EXTRA processing, abbreviation
expansion, keyword ambiguity checking, etc., is done at a higher
level, such that individual routines have no information on how their

keywords arrived. This is a good thing, since otherwise you'd have
all kinds of differing treatments of abbreviations, keyword
inheritance, etc.

JD

Subject: Re: axis problem
Posted by David Fanning on Mon, 21 Jul 2003 15:57:23 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> That's quite a creative set of theories, but the explanation is much
> simpler:

I thought it was getting too complicated. Occum's Razor and all...

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: axis problem
Posted by R.Bauer on Wed, 23 Jul 2003 16:06:42 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Sun, 20 Jul 2003 13:42:36 -0700, Reimar Bauer wrote:
>
>
>> David Fanning wrote:
>>
>>
>>> Reimar Bauer writes:
>>>
>>>
>>>> David I can't agree
>>>>
>>>> It can't be that the user program has to test what keywords values are

>>>> set as default by a routine and if it has this value then it must be
>>>> killed from the _extra structure if it is there.
>>>>
>>>> With all the other keywords it works as supposed. It would be very bad
>>>> if it is somewhere defined that's the user is not able to pass default
>>>> values by _extra. It must be possible to switch back to the whatever
>>>> default value by submitting 0 for example.
>>>>
>>>> At the moment I believe there is a bug with the querying of xyz minor.
>>>> They used keyword_set() instead of n_elements() and ...
>>>
>>> I'm not so sure of my answer that I would bet a whole lot of money on
>>> the "no bug" theory, but still...
>>>
>>> Think of how you would do this. A keyword has a value of 5 by default.
>>> If the keyword is set to 0, you which to set the value to 5. You would
>>> write the program like this:
>>>
>>>   PRO MyPlot, KEY=key, _Extra=extra
>>>   IF N_Elements(key) EQ 0 THEN key = 5
>>>   IF key EQ 0 THEN key = 5
>>>
>>>   PLOTSOMETHING, Key=key, _Extra=extra
>>>   END
>>>
>>> Now, if you pass a value in with the keyword, you encounter the
>>> "processing".
>>>
>>>   IDL> MyPlot, KEY=0
>>>
>>> If you pass it in via the _EXTRA mechanism, you bypass the processing:
>>>
>>>   IDL> MyPlot, _Extra={KEY:0}
>>>
>>> This seems quite reasonable to me. The alternative would be to put
>>> something like this into your program:
>>>
>>>   IF N_ELements(extra) NE 0 THEN BEGIN
>>>   tagnames = Tag_Names(extra)
>>>   index = WHERE(tagnames EQ 'K', count) IF count GT 0 THEN IF
>>>   extra.(index) EQ 0 THEN key = 5 index = WHERE(tagnames EQ 'KE',
>>>   count) IF count GT 0 THEN IF extra.(index) EQ 0 THEN key = 5 index
>>>   = WHERE(tagnames EQ 'KEY', count) IF count GT 0 THEN IF
>>>   extra.(index) EQ 0 THEN key = 5
>>>   ENDIF
>>>
>>> Think what would happen if you wrote a long keyword name, or if you had
>>> multiple keywords defined that you had to chase down like this. You

>>> would spend all your time writing code and no time at all drinking
>>> beer. :-(
>>>
>>> Cheers,
>>>
>>> David
>>
>> Dear David,
>>
>> I think they have probably written a function like this,
>>
>> FUNCTION is_keyword,keyword,names
>>
>> ix=where(strpos(names,keyword) eq 0 ,count_ix) if count_ix eq 1 then
>> return,names[ix]$
>> else message,'Ambiguous keyword abbreviation '+keyword,/cont
>>
>> end
>>
>> And with routine_info you get the names of the keywords of the routine
>> But I can't do this myself with plot
>>
>> print,is_keyword('xmin',['xminor','xaxis']) xminor
>> print,is_keyword('x',['xminor','xaxis'])
>> print,is_keyword('x',['xminor','xaxis']) % IS_KEYWORD: Ambiguous keyword
>> abbreviation x
>>   0
>>
>
>
> That's quite a creative set of theories, but the explanation is much
> simpler:
>
> Compare:
>
>   axis,xaxis=0,xticks=4,color=2,xminor=0
>
> and
>
>   axis,xaxis=0,xticks=4,color=2,xminor=0,xstyle=1
>
> XSTYLE=1, combined with XTICKS=4, is the source of your trouble.
> You're forcing IDL to divide the axis range into non-whole units,
> which means that the minor units cannot be simply specified. Rather
> than approximate minor tick positions, IDL just skips minor ticks
> altogether. You can specify the minor tickmark count yourself with,
> e.g., XMINOR=5, to get the effect you want. You might consider this a
> bug that IDL doesn't do this for you. I'd probably just drop the

> XTICKS specification.
>
> The inherited vs. direct issue was a red herring, since you didn't
> compare apples to apples, specifying many more parameters in the
> _EXTRA version. As far as I know, all _EXTRA processing, abbreviation
> expansion, keyword ambiguity checking, etc., is done at a higher
> level, such that individual routines have no information on how their
> keywords arrived. This is a good thing, since otherwise you'd have
> all kinds of differing treatments of abbreviations, keyword
> inheritance, etc.
>
> JD

Yes, it's simple. I have accepted now that it is this way defined.

I will do a feature request to add this decription to the help manuals.

thanks

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
---------- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
========== =======