
Subject: Re: fsc_Inputfield feature request
Posted by [markb77](#) on Fri, 30 Nov 2012 10:40:39 GMT
[View Forum Message](#) <> [Reply to Message](#)

OK, I looked into the code and it seems like a pretty simple fix. The routine adding the 'd' is the DBLTOSTR.pro from the Coyote library, and the d is there for the general case when scientific notation is used. I think I've fixed the problem with the unusual output when there is no exponent by adding one line of code just before the final output string is generated. I added the line:

```
if indx eq " then typeExt = "
```

which removes the type extension (d) when there is no exponent text. The modified DBLTOSTR routine is below. David, what do you think?

```
FUNCTION DBLTOSTR, value
```

```
    ; Error handling.
    On_Error, 2
    IF N_Elements(value) EQ 0 THEN Message, $
        'Double precision or floating value must be passed to the
function.'

    ; Get the data type.
    theType = Size(value, /Type)
    IF theType NE 4 AND theType NE 5 THEN BEGIN
        value = Double(value)
        theType = 5
    ENDIF

    ; Data extension.
    typeExt = theType EQ 4 ? 'e' : 'd'

    ; Create a string, using the full-width G format.
    rawstr = StrTrim(String(value, Format = '(g)'), 2)

    ; Extract the sign from the string and remove it for the moment.
    sign = StrMid(rawstr, 0, 1) EQ '-' ? '-' : ""
    rawstr = sign EQ "" ? rawstr:StrMid(rawstr, 1)

    ; Is there an exponent in the string?
    ; If so, remove that for the moment.
    epos = StrPos(rawstr, 'e')
    indx = epos gt -1 ? StrMid(rawstr, epos+1) : ""
    rawstr = indx EQ "" ? rawstr:StrMid(rawstr, 0, epos)
```

```

; Find the position of the decimal point.
dpos = StrPos(rawstr, '.')

; Rounding process (assumes 14 significant digits).
outstr = StrArr(15)
FOR i = 0, 14 DO ostr[i] = StrMid(rawstr, i, 1)
aux = Fix(StrMid(rawstr, 16, 1)) GE 5?1:0
FOR i = 14, 0, -1 DO BEGIN
    IF i NE dpos then BEGIN
        sumstr = StrTrim(String(aux+fix(outstr[i])), 2)
        sumlen = StrLen(sumstr)
        ostr[i] = StrMid(sumstr, sumlen-1, 1)
        aux = sumlen EQ 1 ? 0 : 1
    ENDIF
ENDFOR

; Throw away '0's at the end.
ii = 14
WHILE ostr[ii] EQ '0' DO BEGIN
    ii = ii-1
ENDWHILE

; Reconstruct the string.
saux = aux NE 0 ? '1' : ""

if indx eq "" then typeExt = ""

ostr = sign + saux + StrJoin(ostr[0:ii]) + typeExt + indx

; Return it.
RETURN, ostr

END

```

Subject: Re: fsc_Inputfield feature request
Posted by [David Fanning](#) on Fri, 30 Nov 2012 13:20:50 GMT
[View Forum Message](#) <> [Reply to Message](#)

markbates writes:

```

> OK, I looked into the code and it seems like a pretty simple fix. The
> routine adding the 'd' is the DBLTOSTR.pro from the Coyote library,
> and the d is there for the general case when scientific notation is
> used. I think I've fixed the problem with the unusual output when
> there is no exponent by adding one line of code just before the final
> output string is generated. I added the line:
>

```

> if indx eq " then typeExt = "
>
> which removes the type extension (d) when there is no exponent text.
> The modified DBLTOSTR routine is below. David, what do you think?

Works for me. Thanks for doing the leg work. :-)

Generally, when I have to open these old programs up for some reason, I retire them, convert the documentation to IDL-Doc style, and reincarnate them with a "cg" prefix to mark them as a Coyote Library routine. I'll do that with this one as well, so it might take a day or so to get back into the Library.

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Sepore ma de ni thue. ("Perhaps thou speakest truth.")
