Subject: Conversion of MS 64bit timestamps to JD
Posted by andrewcool777 on Fri, 11 Nov 2016 02:23:54 GMT
View Forum Message <> Reply to Message

Hi All,

I'm converting some astronomical files from SER format to FITS.

The SER format contains timestamps as Micrsoft 64bit integer, described as :-

“Holds IEEE 64-bit (8-byte) values that represent dates ranging from January 1 of the year 0001 through December 31 of the year 9999, and times from 12:00:00 AM (midnight) through 11:59:59.9999999 PM. Each increment represents 100 nanoseconds of elapsed time since the beginning of January 1 of the year 1 in the Gregorian calendar. The maximum value represents 100 nanoseconds before the beginning of January 1 of the year 10000.”

Does anyone have a prebuilt function to convert these 64bit integers to Julian Day?

Andrew

Subject: Re: Conversion of MS 64bit timestamps to JD
Posted by Dick Jackson on Sat, 12 Nov 2016 02:35:43 GMT
View Forum Message <> Reply to Message

On Thursday, 10 November 2016 18:23:56 UTC-8, andrew...@gmail.com wrote:
> Hi All,
>
> I'm converting some astronomical files from SER format to FITS.
>
> The SER format contains timestamps as Micrsoft 64bit integer, described as :-
>
> “Holds IEEE 64-bit (8-byte) values that represent dates ranging from January 1 of the year 0001 through December 31 of the year 9999, and times from 12:00:00 AM (midnight) through 11:59:59.9999999 PM. Each increment represents 100 nanoseconds of elapsed time since the beginning of January 1 of the year 1 in the Gregorian calendar. The maximum value represents 100 nanoseconds before the beginning of January 1 of the year 10000.”
>
>
> Does anyone have a prebuilt function to convert these 64bit integers to Julian Day?
>
>
> Andrew

Hi Andrew,

I don't have a prebuilt function, but if you need to build one, here's what I see as the tricky parts:

Going with the naive assumption that the SER value for Jan. 1, 1 CE is 0, and that the values are integers (not IEEE 64-bit floating point, which IDL doesn't support directly*—can you confirm that?) and using the Gregorian calendar as required (not Julian):

```
IDL> help,y,mo,d,h,m,s
Y               LONG      =        2016
MO              LONG      =          11
D               LONG      =          11
H               LONG      =          20
M               LONG      =          16
S               DOUBLE    =       24.000000

IDL> Long64((Greg2Jul(mo,d,y,h,m,s) - Greg2Jul(1,1,1,0,0,0)) * 24LL * 60 * 60 * 1D7)
   636144921839999872
```

Adding 1D-7 to s makes no effect:

```
IDL> s=24D + 1D-7
IDL> s
      24.000000100000001
IDL> Long64((Greg2Jul(mo,d,y,h,m,s) - Greg2Jul(1,1,1,0,0,0)) * 24LL * 60 * 60 * 1D7)
   636144921839999872
```

The 0.0000001-second increment is lost in Double-precision of large day number,
so try a formula with day computed separately from fraction of day:

```
IDL> s=24
IDL> Long64((Greg2Jul(mo,d,y) - Greg2Jul(1,1,1)) * 24LL * 60 * 60 * 1D7) + Long64(((h * 60D +
m) * 60D + s) * 1D7)
   636144921840000000
IDL> s=24D + 1D-7
IDL> Long64((Greg2Jul(mo,d,y) - Greg2Jul(1,1,1)) * 24LL * 60 * 60 * 1D7) + Long64(((h * 60D +
m) * 60D + s) * 1D7)
   636144921840000001
```

So, a function could compute this from the six parameters:
serDateTime = Long64((Greg2Jul(mo,d,y) - Greg2Jul(1,1,1)) * 24LL * 60 * 60 * 1D7) + Long64(((h * 60D + m) * 60D + s) * 1D7)

Just to push the limits, what if we do Nov. 11, 9999 (with fraction of a second):

```
IDL> y=9999
IDL> Long64((Greg2Jul(mo,d,y) - Greg2Jul(1,1,1)) * 24LL * 60 * 60 * 1D7) + Long64(((h * 60D +
m) * 60D + s) * 1D7)
   3155335641840000001
```

Can Long64 handle another factor of 10?

```
IDL> y=99990
IDL> Long64((Greg2Jul(mo,d,y) - Greg2Jul(1,1,1)) * 24LL * 60 * 60 * 1D7) + Long64(((h * 60D +
m) * 60D + s) * 1D7)
  -9223371307014775807
% Program caused arithmetic error: Floating illegal operand
```

Nope.
So indeed, 64-bit works to number the 100 ns intervals from year 1 to year 9999.

Working the other way, to turn serDateTime into six parameters:

```
IDL> serDateTime = 636144921840000001 ; 2016-11-11T20:16:24.0000001 from above

IDL> Jul2Greg,(serDateTime / (24LL * 60 * 60 * 10000000)) + Greg2Jul(1,1,1,0,0,0),mo,d,y
IDL> increments = serDateTime MOD (10000000LL * 24 * 60 * 60)
IDL> s = increments MOD (10000000LL * 60) / 1D7
IDL> m = increments / (10000000LL * 60) MOD 60
IDL> h = increments / (10000000LL * 60 * 60)
```

This seems to have worked:

```
IDL> help,y,mo,d,h,m
Y               LONG      =        2016
MO              LONG      =          11
D               LONG      =          11
H               LONG64    =                  20
M               LONG64    =                  16
IDL> s
      24.000000100000001
```

Hope this helps!

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

*: https://en.wikipedia.org/wiki/Double-precision_floating-poin t_format

---

Subject: Re: Conversion of MS 64bit timestamps to JD
Posted by andrewcool777 on Sat, 12 Nov 2016 23:25:32 GMT
View Forum Message <> Reply to Message

Hi Dick,

This is Gold Star stuff! Maybe even an Elephant stamp too.

Of course, having the ability to record 100nanosecond intervals assunes that the Windows clock is up to that int eh first place, and well calibrated, but that's another story.

A friend and I recorded and occultation by the Centaur asteroid Chariklo
on Oct 1st, using SER format. I've written a converter to get individual FITS
files from the SER file, but it needs those 64bit timestamps converted too.

Many Thanks.

Andrew

Working the other way, to turn serDateTime into six parameters:

IDL> serDateTime = 636144921840000001 ; 2016-11-11T20:16:24.0000001 from above

IDL> Jul2Greg,(serDateTime / (24LL * 60 * 60 * 10000000)) + Greg2Jul(1,1,1,0,0,0),mo,d,y
IDL> increments = serDateTime MOD (10000000LL * 24 * 60 * 60)
IDL> s = increments MOD (10000000LL * 60) / 1D7
IDL> m = increments / (10000000LL * 60) MOD 60
IDL> h = increments / (10000000LL * 60 * 60)

This seems to have worked:

```
IDL> help,y,mo,d,h,m
Y               LONG      =        2016
MO              LONG      =          11
D               LONG      =          11
H               LONG64    =                  20
M               LONG64    =                  16
```

> Andrew

Subject: Re: Conversion of MS 64bit timestamps to JD
Posted by andrewcool777 on Sun, 13 Nov 2016 11:48:15 GMT
View Forum Message <> Reply to Message

Hi Dick,

Using your method, here are the decodes of the 64bit timestamps
embedded in a short SER image that a friend supplied.

Cheers,

Andrew

2016.0000	11.000000	13.000000	6.0000000	38.000000	16.072238
2016.0000	11.000000	13.000000	6.0000000	38.000000	16.332335
2016.0000	11.000000	13.000000	6.0000000	38.000000	16.593015
2016.0000	11.000000	13.000000	6.0000000	38.000000	16.851370
2016.0000	11.000000	13.000000	6.0000000	38.000000	17.114491
2016.0000	11.000000	13.000000	6.0000000	38.000000	17.456318
2016.0000	11.000000	13.000000	6.0000000	38.000000	17.633305
2016.0000	11.000000	13.000000	6.0000000	38.000000	17.891857
2016.0000	11.000000	13.000000	6.0000000	38.000000	18.152162
2016.0000	11.000000	13.000000	6.0000000	38.000000	18.412077
2016.0000	11.000000	13.000000	6.0000000	38.000000	18.671080
2016.0000	11.000000	13.000000	6.0000000	38.000000	18.932369
2016.0000	11.000000	13.000000	6.0000000	38.000000	19.191133
2016.0000	11.000000	13.000000	6.0000000	38.000000	19.451026

On Sunday, 13 November 2016 09:55:34 UTC+10:30, andrew...@gmail.com wrote:
> Hi Dick,
>
> This is Gold Star stuff! Maybe even an Elephant stamp too.
>
> Of course, having the ability to record 100nanosecond intervals assunes that the Windows clock is up to that int eh first place, and well calibrated, but that's another story.
>
> A friend and I recorded and occultation by the Centaur asteroid Chariklo
> on Oct 1st, using SER format. I've written a converter to get individual FITS
> files from the SER file, but it needs those 64bit timestamps converted too.
>
> Many Thanks.
>
> Andrew
>
> Working the other way, to turn serDateTime into six parameters:
>
> IDL> serDateTime = 636144921840000001 ; 2016-11-11T20:16:24.0000001 from above
>
> IDL> Jul2Greg,(serDateTime / (24LL * 60 * 60 * 10000000)) + Greg2Jul(1,1,1,0,0,0),mo,d,y
> IDL> increments = serDateTime MOD (10000000LL * 24 * 60 * 60)
> IDL> s = increments MOD (10000000LL * 60) / 1D7
> IDL> m = increments / (10000000LL * 60) MOD 60
> IDL> h = increments / (10000000LL * 60 * 60)
>
> This seems to have worked:
>

> IDL> help,y,mo,d,h,m
> Y LONG = 2016
> MO LONG = 11
> D LONG = 11
> H LONG64 = 20
> M LONG64 = 16
>
>
>> Andrew

Subject: Re: Conversion of MS 64bit timestamps to JD
Posted by Dick Jackson on Mon, 14 Nov 2016 16:09:00 GMT
View Forum Message <> Reply to Message

Hi Andrew,

That looks promising, but do those dates/times seem at least roughly correct? With the calendar systems and time zone issues, I'd want to know that we're not a couple of days or hours off. (I'd implicitly trust the minutes and seconds, even the tenth-of-a-microseconds! :-)

I notice that your output is only giving microsecond precision. If you were at all concerned about that, I'd want to see a couple of zeroes to reassure that the conversion is good. I'm guessing that you're using Float for these values, but that is not enough to allow the 'seconds' to go up to 60 and allow seven accurate places after the decimal. So, if this matters to you, I'd use Double when computing seconds (at least) and print with eight or nine places, expecting to see some zeroes.

Thanks for the stickers. :-)

Cheers,
-Dick

On Sunday, 13 November 2016 03:48:19 UTC-8, andrew...@gmail.com wrote:
> Hi Dick,
>
> Using your method, here are the decodes of the 64bit timestamps
> embedded in a short SER image that a friend supplied.
>
> Cheers,
>
> Andrew
>
>
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 16.072238
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 16.332335
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 16.593015
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 16.851370
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 17.114491

> 2016.0000 11.000000 13.000000 6.0000000 38.000000 17.456318
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 17.633305
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 17.891857
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 18.152162
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 18.412077
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 18.671080
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 18.932369
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 19.191133
> 2016.0000 11.000000 13.000000 6.0000000 38.000000 19.451026
>
>
> On Sunday, 13 November 2016 09:55:34 UTC+10:30, andrew...@gmail.com wrote:
>> Hi Dick,
>>
>> This is Gold Star stuff! Maybe even an Elephant stamp too.
>>
>> Of course, having the ability to record 100nanosecond intervals assunes that the Windows clock is up to that int eh first place, and well calibrated, but that's another story.
>>
>> A friend and I recorded and occultation by the Centaur asteroid Chariklo
>> on Oct 1st, using SER format. I've written a converter to get individual FITS
>> files from the SER file, but it needs those 64bit timestamps converted too.
>>
>> Many Thanks.
>>
>> Andrew
>>
>> Working the other way, to turn serDateTime into six parameters:
>>
>> IDL> serDateTime = 636144921840000001 ; 2016-11-11T20:16:24.0000001 from above
>>
>> IDL> Jul2Greg,(serDateTime / (24LL * 60 * 60 * 10000000)) + Greg2Jul(1,1,1,0,0,0),mo,d,y
>> IDL> increments = serDateTime MOD (10000000LL * 24 * 60 * 60)
>> IDL> s = increments MOD (10000000LL * 60) / 1D7
>> IDL> m = increments / (10000000LL * 60) MOD 60
>> IDL> h = increments / (10000000LL * 60 * 60)
>>
>> This seems to have worked:
>>
>> IDL> help,y,mo,d,h,m
>> Y LONG = 2016
>> MO LONG = 11
>> D LONG = 11
>> H LONG64 = 20
>> M LONG64 = 16
>>
>>
>>> Andrew