Mission Start Delayed On Ds1922l/t, Ds1923, Ds2422

Oct 27 2004, 08:51 AM

A design flaw was found in the DS1922 which causes the first mission sample to be delayed. This problem affects all flavors of this part: DS1922L, DS1922T, DS1923, and DS2422. There are 2 cases to consider for examining the impact of this problem. The first case is for a DS1922 in an application where the mission may be stopped and started regularly during its use. In this case, each time a new mission is started the first sample could be delayed by as much as one sample period. For example, when using a 10 minute sample rate and the mission is re-started at 1:00, the first sample could be recorded anywhere between 1:00 and 1:10. Every sample thereafter will be recorded at the regular 10 minute rate.

The second case to consider is when prototyping and experimenting with different values for the Sample Rate and the "Enable High-Speed Sampling" bit. The impact in this case can be a much larger delay, but the delay is predictable and can be shortened easily. What happens is that after a new mission is started, the device records its first sample (records mission timestamp, etc) and internally copies the Sample Rate register value to an internal countdown register. When this count reaches zero, the next sample is taken. The Clear Memory command should have erased this internal register to zero, but it does not and that is the cause of the flaw. When the mission is stopped, the countdown is halted in some arbitrary value between 0 and the Sample Rate value. This means that when another new mission is started, instead of taking the first sample right away, this internal countdown register must finish the previously halted countdown.

When using high-speed sampling (EHSS=1), this problem is hardly noticeable. When stopping and starting new missions at 10 second intervals, there could be a as many as 10 seconds for the new mission to start. The real problem, where the delay is noticed drastically, when switching from 10 second intervals to 1 minute intervals. In this case, the internal countdown register has a value between 0 and 10, but it is now counting down in minutes, instead of in seconds. So the delay could be as long as 10 minutes for the first sample, then 1 minute for each sample thereafter. The OneWireViewer app makes it easy to get caught by this issue since the field for starting new missions is in seconds. If the mission Sample Rate is evenly divisible by 60 (or greater than the max value), then the Sample Rate is divided down and the EHSS bit is cleared. So, if for example a mission was started with a Sample Rate of 601 seconds, the EHSS bit would be set and the Sample Rate register would contain the value "601". If subsequently a new mission was started with the value 600 seconds, the EHSS bit would be cleared and the Sample Rate register would have contained the value "10". But, before the iButton started taking samples at the 10 minute rate, it would first have to countdown somewhere between 0-601 minutes.

The best workaround right now is that when a DS1922 is delayed in its mission start time when using minute resolution, stop and start a new mission with "1 second" resolution (EHSS=1, SR=1). After that new mission starts, stop it and start a new mission containing your desired mission parameters. This reduces the delay before the mission start significantly (and doesn't reduce the lifetime of the part, since no samples are taken during that interval while the internal countdown register finishes).

Note that when performing a Start Upon Temperature Alarm (SUTA) mission, the device samples the temperature for the alarming condition at the same rate as your Sample Rate, which means it is effected by the same flaw. The first sample of the temperature and check for alarm will occur after the internal countdown is complete.

In the general case of stopping and starting new missions frequently, reading the RTC immediately after halting the mission (or halt the mission and RTC at approximately the same time) allows the system to predict very accurately what the value of the internal countdown register is by measuring the time between the last recorded sample's timestamp and the current value of the RTC. In this manner, it's possible to calculate how many minutes/seconds delay are needed so that the mission can be started at a pre-decided time. Since this problem doesn't conceivably affect most applications (stop/start and change mission parameters seems more likely during prototyping), it is recommended that prototyping be done with smaller values and EHSS=1. If longer sample rates are need, on the order of minutes, it is recommended to proceed with caution and use the RTC to predict the value of this hidden internal register.

This cause of this problem has been identified and fixed by Design Engineering and will be included in the next revision of the part. This revision is scheduled for Summer of 2005.

Scott Hughes, shughes aht dalsemi daut com
Engineer, Maxim/Dallas Semiconductor

Dec 7 2004, 10:50 AM

An appnote is now available which describes the problem and provides a pseudo-code description of the proposed workaround.

http://www.maxim-ic.com/appnotes.cfm/appnote_number/3429

Note that the maximum delay you could experience (though it would be highly unlikely to occur) is 16,384 minutes. This assumes the value 0x3FFF was loaded into the sample rate register and, subsequently, loaded into the internal countdown register. This equates to about 11 days of start delay before the first mission sample occurs. Using the proposed "1-second mission" workaround, it could be reduced to 4.5 hours. There is no workaround that actually avoids the delay altogether, it is only possible to reduce the time by a factor of 60 only when minute resolution is the desired resolution. If second resolution is desired, the "1-second mission" workaround will not aid in reducing the delay, but will aid in acquiring a more precise start time.

Scott Hughes, shughes aht dalsemi daut com
Engineer, Maxim/Dallas Semiconductor