App Note 3429: Datalogger Start Delay Issue in DS2422, DS1923, DS1922L, and DS1922T

The new datalogger family of 8k-byte Thermochron and Hygrochron chips has a design flaw which causes the first mission sample to be delayed by as much as one sample period. This application note describes the issue so that customers can characterize the delay in their applications. A workaround is described which can assist in reducing the worst-case delay.

The new datalogger family of 8k-byte Thermochron and Hygrochron chips has a design flaw which causes the first mission sample to be delayed by as much as one sample period. This includes the temperature dataloggers (DS1922L and DS1922T), the humidity/temperature datalogger (DS1923), and the stand-alone chip datalogger (DS2422). There are 2 cases to consider for examining the impact of this problem. The first case is for a datalogger in an application where the current mission may be stopped and a new mission 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, if using a 10 minute sample rate with a mission that 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 the first sample (records mission timestamp, etc) and internally copies the Sample Rate register value to a 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.

Work-Around

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 as many as 10 seconds for the new mission to start. The real problem, where the delay is noticed drastically, occurs 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 best workaround for a datalogger that is delayed in its mission start time when using minute resolution is to 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). See the pseudo-code in Listing 1

for an example of how this could be implemented.

Changing Resolution to Accelerate First Sample

Listing 1

```
# Start by initializing the datalogger with the desired mission parameters.
# For this example, the sample rate is 60 minutes, with no start delay
SET_SAMPLE_RATE( 60 );
SET_EHSS( 0 ); # Disable high-speed sampling

SET_START_DELAY( 0 );
SET_TEMPERATURE_SAMPLING( 0.0625C ); # enable sampling in desired resolution

# Initialize and start the Real-time clock
SET_RTC( CurrentTimeAndDate );

START_MISSION();

# Since resolution is minutes, first sample will only be taken on RTC's
# minute boundary, so delay to give the device time to take the sample
DELAY_ONE_MINUTE();

# Check to make sure the Mission Sample Count register is a non-zero value
If ( GET_MISSION_SAMPLE_COUNT() == 0 ) Then

   # Mission did not begin right away, so a 1 second mission will be
   # used to clear the internal counter of the device
   STOP_MISSION();

   SET_SAMPLE_RATE( 1 );
   SET_EHSS( 1 ); # Enable high-speed sampling

   START_MISSION();

   # Now counter is counting down in 1 second increments instead of
   # 1 minute increments.  Wait for first sample to be recorded.
   While( GET_MISSION_SAMPLE_COUNT() == 0 )
      DELAY_ONE_MINUTE();
   EndWhile;

   # Mission sample has been recorded and the internal countdown register
   # contains a value of either 1 or 0.  Reset the original parameters.
   STOP_MISSION();

   SET_SAMPLE_RATE( 60 );
   SET_EHSS( 0 ); # Disable high-speed sampling

   START_MISSION();

   # Mission will now start in less than 2 minutes reliably.  The time could be
```

```
   # 2 minutes because there is up to one minute of delay if the RTC is not
   # near the minute boundary and up to one minute of delay from the internal
   # counter.  Delay 2 minutes and double-check, for sanity's sake.
   DELAY_ONE_MINUTE();
   DELAY_ONE_MINUTE();

   If ( GET_MISSION_SAMPLE_COUNT == 0 ) THEN
      SEVERE_ERROR(); # There is another unrelated failure with this device
   EndIf

EndIf;
```

In the case of stopping a mission and starting a new mission 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 (stopping a mission, changing mission parameters, and starting a new mission is much 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 forthcoming DS1922L/T, DS1923, and DS2422 Revision B1. This revision is scheduled for summer of 2005.

More Information

DS1922L: QuickView -- Full (PDF) Data Sheet

DS1922T: QuickView -- Full (PDF) Data Sheet

DS1923: QuickView -- Full (PDF) Data Sheet -- Free Samples

DS2422: QuickView -- Full (PDF) Data Sheet -- Free Samples