Timing Deterministic Applications

Because of the preemptive scheduling used by the real-time operating system (RTOS), higher priority threads can monopolize processor resources. A high priority thread might use all of the processor resources, not allowing lower priority threads to execute. After separating the deterministic tasks from non-deterministic tasks, you can use timing methods to ensure deterministic performance for deterministic tasks while allowing non-deterministic tasks to execute as needed.

You can use the Microsecond Sleep Functions in the Real-Time Utility Library and external timing sources to control loop execution rates.

Asynchronous timers inherently control loop execution rate based on the specified timer period. The period must be longer than the time required to execute the callback function so that time remains for lower priority threads to run before the next timer iteration.

Timing Loops Using the Microsecond Sleep Functions

Use the Microsecond Sleep Functions to time loops on RT targets. With these functions, you can use the microsecond timer of the RTOS to control the timing of loops. By configuring a sleep period for a portion of each loop, you allow time for lower priority threads to execute before the next loop iteration.

SleepUS

The SleepUS function causes a thread to sleep for the specified amount of time. For example, if the microsecond timer value is 112 µs when `SleepUS` executes and the **duration** parameter is 10, then the thread sleeps and does not return until the microsecond timer value equals 122 µs.

SleepUntilNextMultipleUS

The SleepUntilNextMultipleUS function causes a thread to sleep until the value of the microsecond timer equals equals a multiple of the **multiple** parameter. For example, if `SleepUntilNextMultipleUS` executes with a **multiple** input of 10 µs and the microsecond timer value is 112 µs, the function that calls `SleepUntilNextMultipleUS` sleeps until the microsecond timer value equals 120 µs because 120 µs is the first multiple of 10 µs after the execution of `SleepUntilNextMultipleUS`.

SleepUntilUS

The SleepUntilUS function causes a thread to sleep until the value of the microsecond timer equals the value of the **targetTime** parameter.

Timing Deterministic Applications Using External Timing Sources

The National Instruments drivers that run on RT targets support functions that can cause the current thread to sleep and then return when the driver detects a specific event. For example, you can use NI-DAQmx and NI data acquisition hardware to time real-time applications. Refer to the specific NI driver documentation for information about functions that you can use to sleep and wait for driver events.

Timing Real-Time Applications with NI-DAQmx

You can use NI data acquisition hardware with NI-DAQmx to match loop rates to match the rate of the hardware clock. With NI-DAQmx, you can use the following methods to time real-time applications:

- **Hardware-Timed Single-Point**—NI-DAQmx supports hardware-timed, single-point sample mode in which samples are acquired or generated continuously using hardware timing and no buffering. You can use hardware-timed, single-point mode for control applications that require input and/or output within a deterministic period of time. Refer to the *NI-DAQmx Single-Point Real-Time Applications* topic of the *NI-DAQmx Help* for information about using hardware-timed, single-point operations to time your deterministic application.
- **Counter Timers**—NI-DAQmx supports using hardware-timed counter input operations to drive a control loop. Use the `DAQmxCfgSampClkTiming` function to synchronize the counter operations with the counter sample clock. Refer to the *Hardware-Timed Counter Tasks* topic of the *NI-DAQmx Help* for information about using counter input operations to time deterministic applications.

Refer to the *NI-DAQmx Help* for more information about timing control loops using NI data acquisition hardware and NI-DAQmx software. Select **Start»All Programs»National Instruments»NI-DAQ»NI-DAQmx Help** to display the *NI-DAQmx Help*.