Using the Timed Loop to Write Multirate Applications in LabVIEW™

Introduction

This application note describes the features of the Timed Loop and how to use the Timed Loop to develop multirate applications. For information about using the Timed Loop with specific hardware devices, such as DAQ and FPGA devices, refer to the documentation for those devices.

Applicable Platforms

You can use the Timed Loop on Windows 2000/XP, PharLap ETS, and LabVIEW Real-Time Module on Mac OS 10.x targeted to a supported real-time device.

Refer to the `labview\examples\general\timedloop.llb` for examples of using the Timed Loop.

Timed Loops

A Timed Loop executes an iteration of the loop at the period you specify. Use the Timed Loop when you want to develop VIs with multi-rate timing capabilities, precise timing, feedback on loop execution, timing characteristics that change dynamically, or several levels of execution priority. Use the VIs and functions on the **Timed Loop** and the **DAQmx** palettes with the Timed Loop to develop VIs that control timing sources.

The Timed Loop includes the (1) Input, (2) Left Data, (3) Right Data, and (4) Output nodes, as shown in the following illustration.

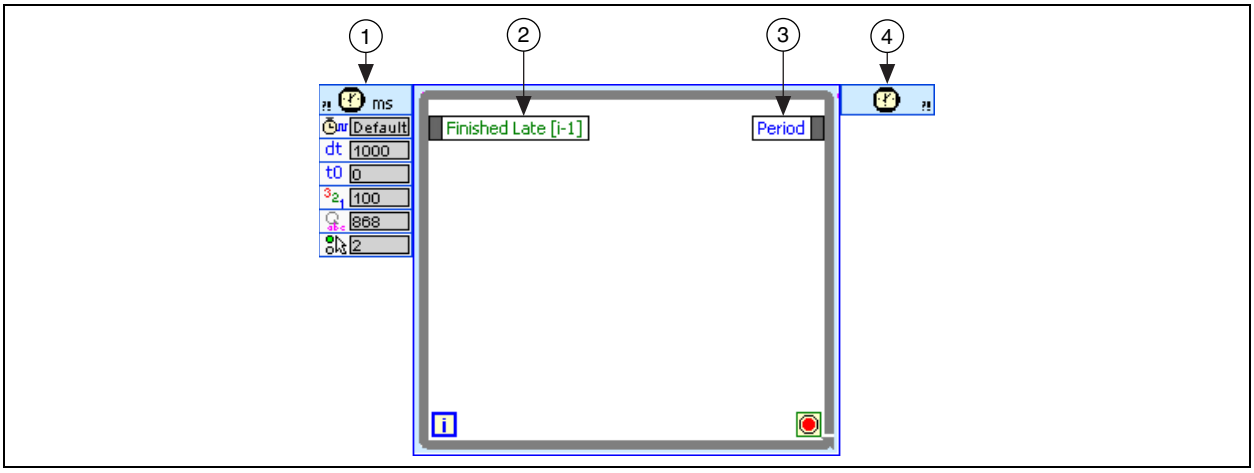

Figure 1. Timed Loop

You can set configuration options of the Timed Loop by wiring values to the inputs of the Input Node, or you can use the **Loop Configuration** dialog box to enter values for the options. By default, the inputs of the Input Node appear as icons with the values you specified in the **Loop Configuration** dialog box. Refer to the *Configuring Timed Loops* section for more information about configuring a Timed Loop.

The Left Data Node of the Timed Loop provides timing and status information about the previous loop iteration, such as if the iteration executed late, the time the iteration actually started executing, and when the iteration should have executed. You can wire data from the Left Data Node to the Right Data Node to configure future iterations of the Timed Loop. Right-click the border of the Timed Loop and select **Show Right Data Node** from the shortcut menu to display the Right Data Node. You can resize the Left Data and Right Data nodes. Refer to the *Changing Timed Loop Input Node Values Dynamically* section for more information on using the Left Data and Right Data nodes.

The Output Node returns error information the Timed Loop receives in the **error in** input of the Input Node or error information the Timed Loop generates during an iteration. For example, if an external source aborts the execution of the Timed Loop or if more than 128 Timed Loops run at one time, the Output Node returns an error. The Output Node does not return error information from any subdiagram that executes within the Timed Loop.

Configuring Timed Loops

Use the **Loop Configuration** dialog box to configure how the Timed Loop executes. Double-click the Input Node or right-click the Input Node and select **Configure Timed Loop** from the shortcut menu to display the **Loop Configuration** dialog box.

You can use this dialog box to specify a timing source, period, offset timing, and other options, or you can use the dialog box to configure options so they appear as inputs on the Input Node.

Place a checkmark in the **Use terminal** checkbox for any dialog box option you want to appear as an input on the Input Node. Any option you configure in the **Loop Configuration** dialog box to appear as an input on the Input Node appears as an icon without the configuration values. Right-click the Input Node and select **Name Format»Names** from the shortcut menu to display the names of the inputs. Only the options for which you select **Use terminal** on the **Loop Configuration** dialog box appear as inputs when you select **Name Format»Names** from the shortcut menu.

You also can use the Right Data Node to dynamically adjust the period, offset, priorities, and mode values for the Timed Loop. The updates take effect the next iteration of the loop. Refer to the *Changing Timed Loop Input Node Values Dynamically* section for more information about dynamically adjusting the Timed Loop values.

Selecting Timing Sources

A timing source determines when a Timed Loop executes a loop iteration. By default, the Timed Loop uses the 1 kHz clock of the operating system as the timing source and can execute only once every 1 ms because that is the fastest speed at which the operating system timing source operates. If the system does not include a supported hardware device, the 1 kHz clock is the only timing source available. If the system does include a supported hardware device, you can select from other timing sources, such as the 1 µs in CPU cycles available on some real-time hardware; or events, such as the rising edge of a DAQ-STC counter input or output; or the end-of-scan interrupt of an E-Series board AI engine.

Use the **Source type** listbox in the **Loop Configuration** dialog box to select a timing source or use the Create Timing Source VI to programmatically select a timing source.

Setting the Period and the Offset

The period is the length of time between loop executions. The offset is the length of time the Timed Loop waits to execute the iterations. The timing source determines the time unit of the period and the offset. If the timing source is a 1 kHz clock, the unit of time for the period and the offset is in milliseconds. If the timing source is a 1 MHz clock on a LabVIEW Real-TIme target with a Pentium processor, the unit of time for the period and the offset is in microseconds.

The time the first timing source starts determines the start time of the offset.

The Timed Loops in the following block diagram use the default 1 kHz timing source and have offset (**t0**) values of 500, so the Timed Loops wait 500 ms to execute their iterations. The period (**dt**) for Loop A is 1,000 ms, and the period for Loop B is 2,000 ms, which means Loop A executes every second and Loop B executes every 2 seconds. Both Timed Loops stop executing after six iterations. Loop A stops executing after 6 seconds because its period is 1 second, and Loop B stops executing after 12 seconds because its period is 2 seconds.

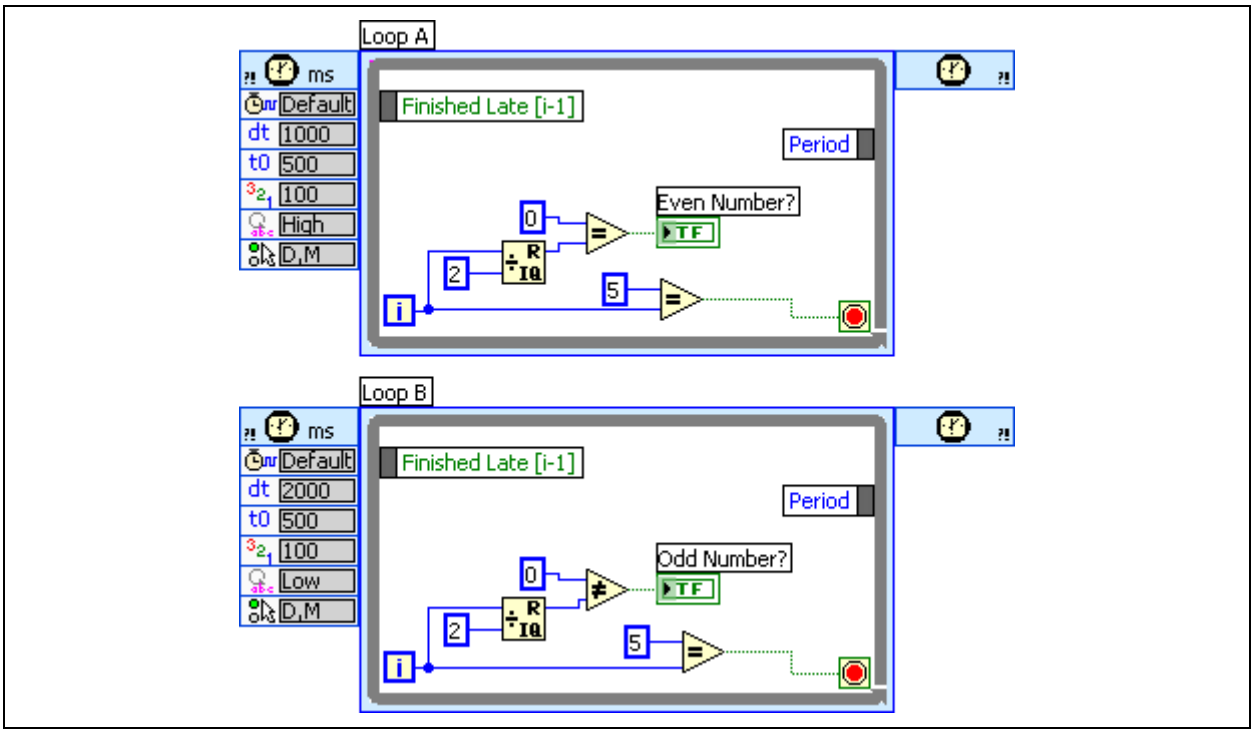

Figure 2. Setting Timed Loop Period

Setting Priorities

Each Timed Loop on the block diagram creates and runs in its own execution system that contains a single thread, so no parallel tasks can occur. The priority of a Timed Loop specifies when the loop executes on the block diagram relative to other Timed Loops. Use the priority setting of a Timed Loop to write applications with multiple tasks that can preempt each other in the same VI. The higher the value you enter in **Priority** of the Timed Loop, the higher the priority the Timed Loop has relative to other Timed Loops on the block diagram. The value you enter in the Priority input must be a positive integer between 1 and 2,147,480,000.

In the following block diagram, the Priority (**321**) value of Loop B (1,000) is higher than the priority of Loop A (100). Both loops are configured with an offset (**t0**) value of 30, so they execute at the same time. Because Loop B has a higher priority, it executes first.

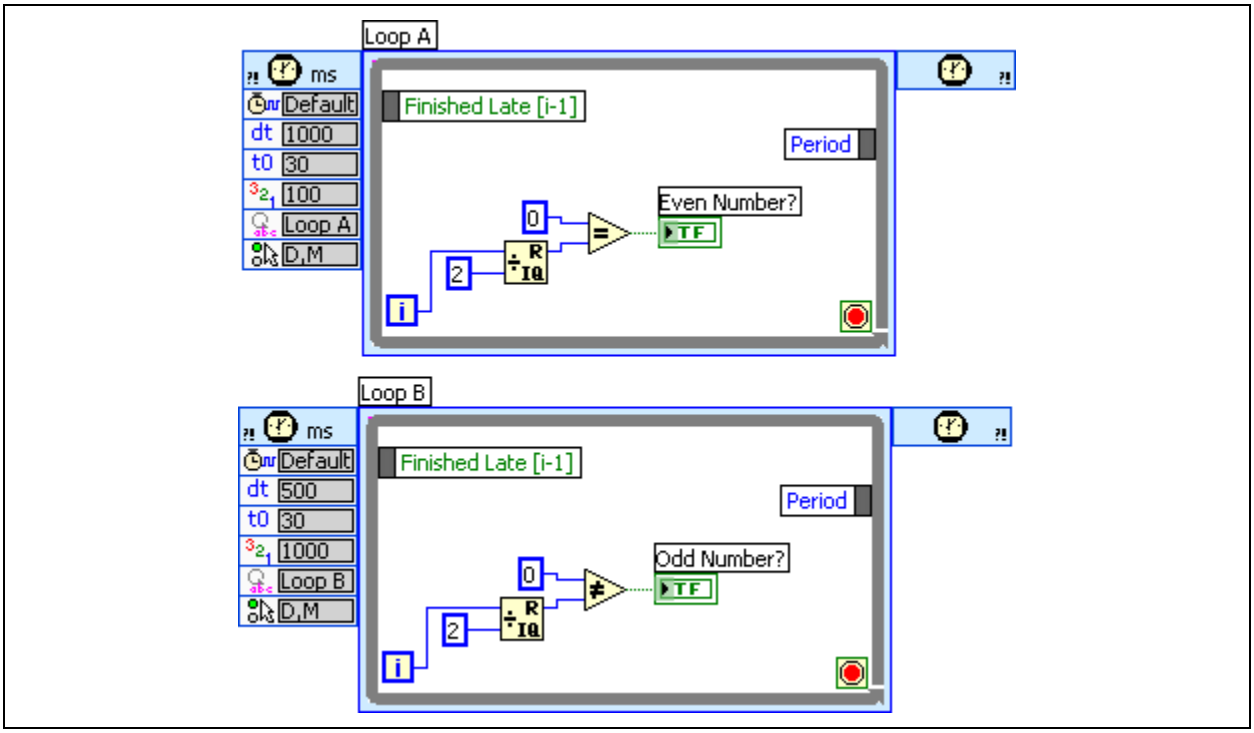

Figure 3. Setting Timed-Loop Priority

If two Timed Loops have the same priority, the first Timed Loop that executes completes its execution before the other Timed Loop starts its execution.

Timed Loops execute at a priority below the time-critical priority of any VI but above high priority, which means that Timed Loops execute in the data flow of a block diagram ahead of any VI not configured to run at a time-critical priority.

Naming Timed Loops

By default, LabVIEW automatically uniquely identifies each Timed Loop you place on the block diagram with a name, which appears in the **Loop name** text box of the **Loop Configuration** dialog box. You can rename the Timed Loop by entering a name in this text box. You can use the unique name of the Timed Loop with the VIs on the **Timed Loop** palette to programmatically stop the Timed Loop and to synchronize a group of Timed Loops to use the same start time.

If a reentrant VI includes a Timed Loop and you use two or more instances of that reentrant VI as subVIs on a block diagram, you must programmatically change the name of the Timed Loop for each instance of the reentrant VI. Ensure that the reentrant VI that includes the Timed Loop has an input terminal on the connector pane connected to a string control wired to the **Loop name** input of the Timed Loop on the block diagram. On the block diagram where two or more instances of the reentrant VI are used as a subVI, wire unique string values to the **Loop name** input on the reentrant subVI to uniquely identify each Timed Loop within each instance of the reentrant subVI. For example, the following block diagram includes two instances of the same reentrant VI as subVIs. The **Loop name** string constants wired to the instances of the subVIs pass two different names to the Timed Loop in the block diagram of the reentrant VI each time the block diagram executes. If the Timed Loops were not programmatically renamed, LabVIEW would return an error.

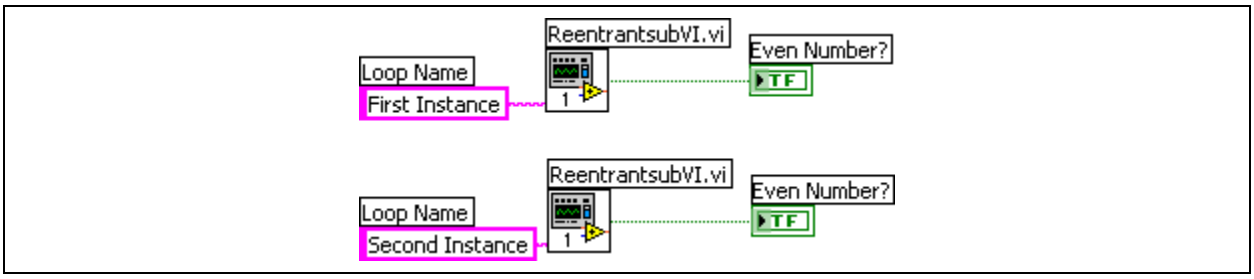

Figure 4. Naming Timed Loops in Reentrant VIs

Refer to the *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability*, Application Note 114, for more information about using reentrant VIs.

Timed Loop Modes

Occasionally, an iteration of a Timed Loop might execute later than the time you specified. The mode of the Timed Loop determines how the loop handles any late executions. Use the options in the **Timed Loop action to take on late iterations** section of the **Loop Configuration** dialog box or the **Mode** input of the Input Node to specify the mode a Timed Loop uses to handle the late execution of a Timed Loop iteration.

You can handle the late execution of a Timed Loop in the following ways:

- The LabVIEW Timed Loop Scheduler can align the execution with the original established schedule.
- The LabVIEW Timed Loop Scheduler can define a new schedule that starts at the current time.
- The Timed Loop can process the missed iterations.
- The Timed Loop can skip any missed iterations.

For example, if you set a Timed Loop with a period of 100 ms and an offset of 30 ms, you expect the first loop iteration to execute 30 ms after the first timing source starts running and in multiples of 100 ms after that at 130 ms, 230 ms, 330 ms, and so on. However, the first execution of the Timed Loop might occur after 240 ms have elapsed. Because other Timed Loops or hardware devices might already be running at the schedule you specified, you might want to align the late Timed Loop with the already running global schedule, which means the Timed Loop should align itself as quickly as possible with the schedule you specified. In this case, the next Timed Loop iteration would run at 330 ms and continue to run in multiples of 100 at 430 ms, 530 ms, and so on. If aligning the Timed Loop with other Timed Loops or other hardware devices is not important, the Timed Loop can run immediately and use the current time as its actual offset. In this case, the subsequent loop iterations would run at 240 ms, 340 ms, 440 ms, and so on.

If the Timed Loop is late, it might miss data other Timed Loops or hardware devices generate. For example, if the Timed Loop missed two iterations and some of the data from the current period, a buffer could hold the data from the missed iterations. You might want the Timed Loop to process the missed data before it aligns with the schedule you specified. However, a Timed Loop that processes the missed iterations causes jitter, which is the amount of time that a loop cycle time varies from the time you specified. If you do not want to process the missed data, the Timed Loop can ignore older data in the buffer the loop iterations missed and process only the latest data, such as the data available at the next period and the subsequent iterations.

Configuring Modes Using the Loop Configuration Dialog Box

By default, the Timed Loop discards any data generated during missed iterations and maintains the original schedule. For this mode setting, the icon for the **Mode** input of the Input Node appears with a D for discard and an M for maintain. Remove the checkmark from the **Discard missed periods** checkbox in the **Loop Configuration** dialog box to process data any missed or late loop iterations generate. For this mode setting, the icon for the **Mode** input of the Input Node includes a P for process. Remove the checkmark from the **Maintain original phase** checkbox in the **Loop Configuration** dialog box to configure the Timed Loop to execute on a new schedule based on the first iteration of the Timed Loop.

Configuring Modes Using the Input Node

Use the **Mode** input of the Timed Loop Input Node to programmatically change the mode of the Timed Loop or maintain the current configuration of the modes you configured in the **Loop Configuration** dialog box. Open the **Loop Configuration** dialog box and place a checkmark in the **Use terminal** checkbox in the **Timed Loop action on late iterations** section. Close the dialog box, right-click the **Mode** input, and select **Create»Constant** or **Create»Control** from the shortcut menu to create an enumerated constant or control you can use to select a mode. Select **No Change** in the enumerated constant or control to maintain the current mode configuration or select the option you want to take on any late or missed iteration of the Timed Loop.

Changing Timed Loop Input Node Values Dynamically

Use the Left Data Node to acquire information about the execution of the Timed Loop, such as if the timing source executed late or if the offset or period values changed. You can wire the values the Left Data Node returns to the Right Data Node or to nodes in the subdiagram within the Timed Loop.

Use the Right Data Node to dynamically change the input values of the Timed Loop on the next loop iteration. The Timed Loop in the following block diagram runs 1 second (1,000 ms) longer each time the loop iterates until the loop has executed six times.

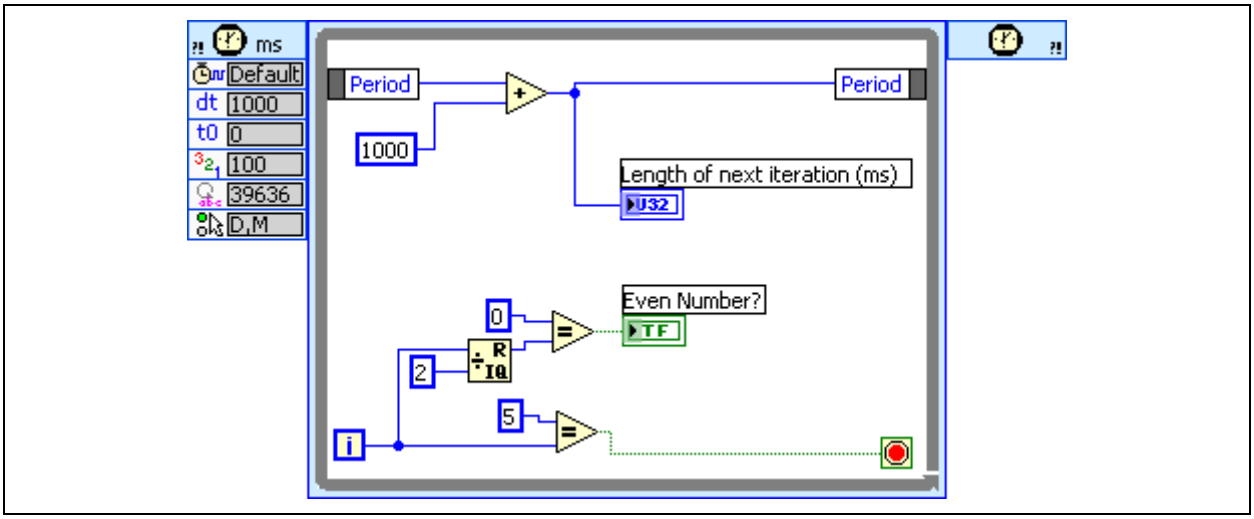

Figure 5. Changing Timed Loop Values Dynamically Using the Left Data and Right Data Nodes

If you dynamically change the offset of the Timed Loop by wiring a value to the **Offset** input terminal of the Right Data Node, you also must specify a mode with the **Mode** input terminal of the Right Data Node. To set the mode, right-click the **Mode** input terminal of the Right Data Node and select **Create»Constant** or **Create»Control** from the shortcut menu to create an enumerated constant or control you can use to select a mode.

Aborting a Timed Loop Execution

Use the Stop Timed Loop VI to abort the execution of a Timed Loop programmatically. Specify the name of the Timed Loop you want to abort by wiring that name in a string constant or control to the **name** input of the Stop Timed Loop VI. For example, in the following block diagram, the Low Timed Loop includes the Stop Timed Loop VI. The High Timed Loop runs and displays the number of iterations it has completed. If the user clicks the **Abort Time Critical Loop** button on the front panel, the **Wakeup reason** output of the Left Data Node returns a value of Aborted, a dialog box appears, and when the user clicks **OK** in the dialog box, the VI stops running.

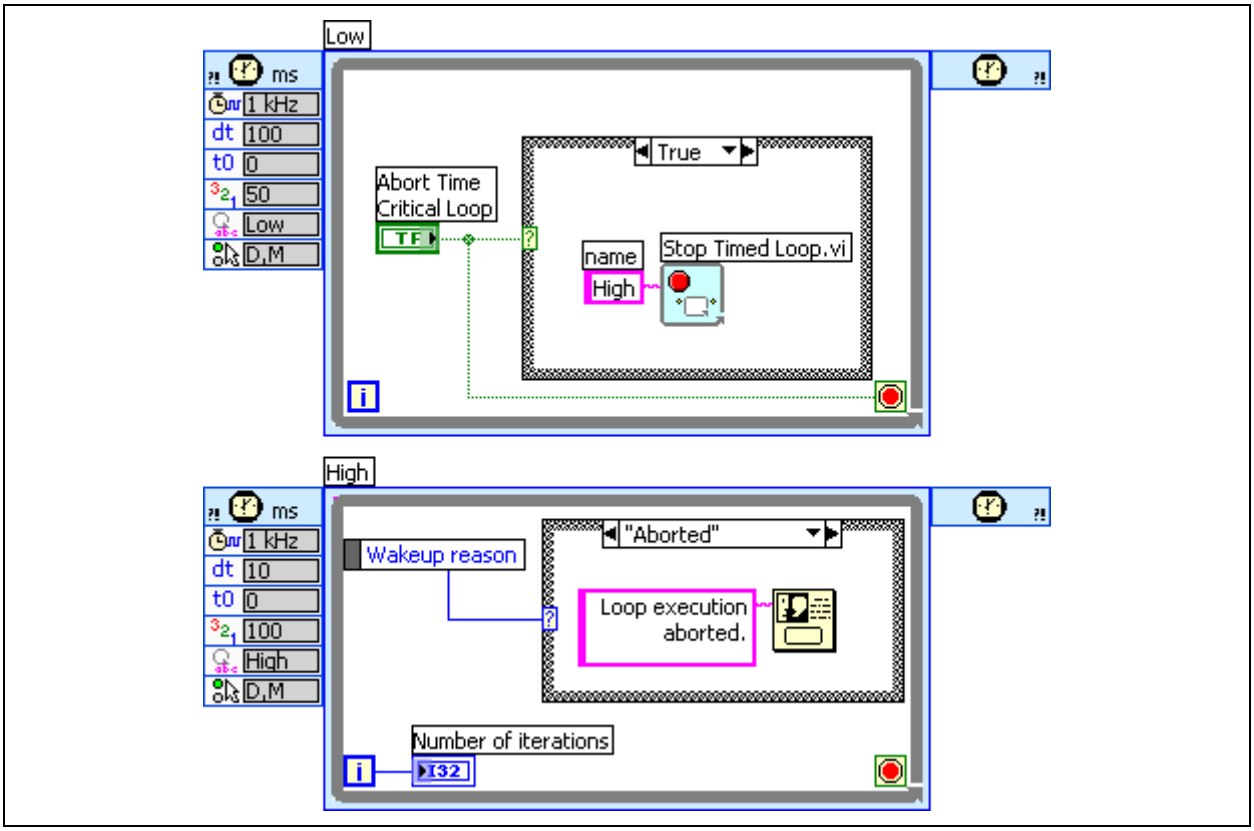

Figure 6. Aborting Timed Loop Execution

Synchronizing Timed Loops

Use the Synchronize Timed Loop Starts VI to ensure all the Timed Loops on a block diagram use the same start time and the same timing source. For example, you might have two Timed Loops and you want to ensure that they execute on the same schedule relative to each other. You might want the first Timed Loop to execute first and generate data, then have the second Timed Loop process that data when the Timed Loop execution finishes. To ensure that both Timed Loops use the same start time as the basis for their execution, you create a Timed Loop group by wiring a name for the group to the **synchronization group name** input and wiring an array of Timed Loop names to the **loop names** input.

The Synchronize Timed Loop Starts VI in the following block diagram creates a synchronization group name and synchronizes Timed Loops A and B to use the same start time.

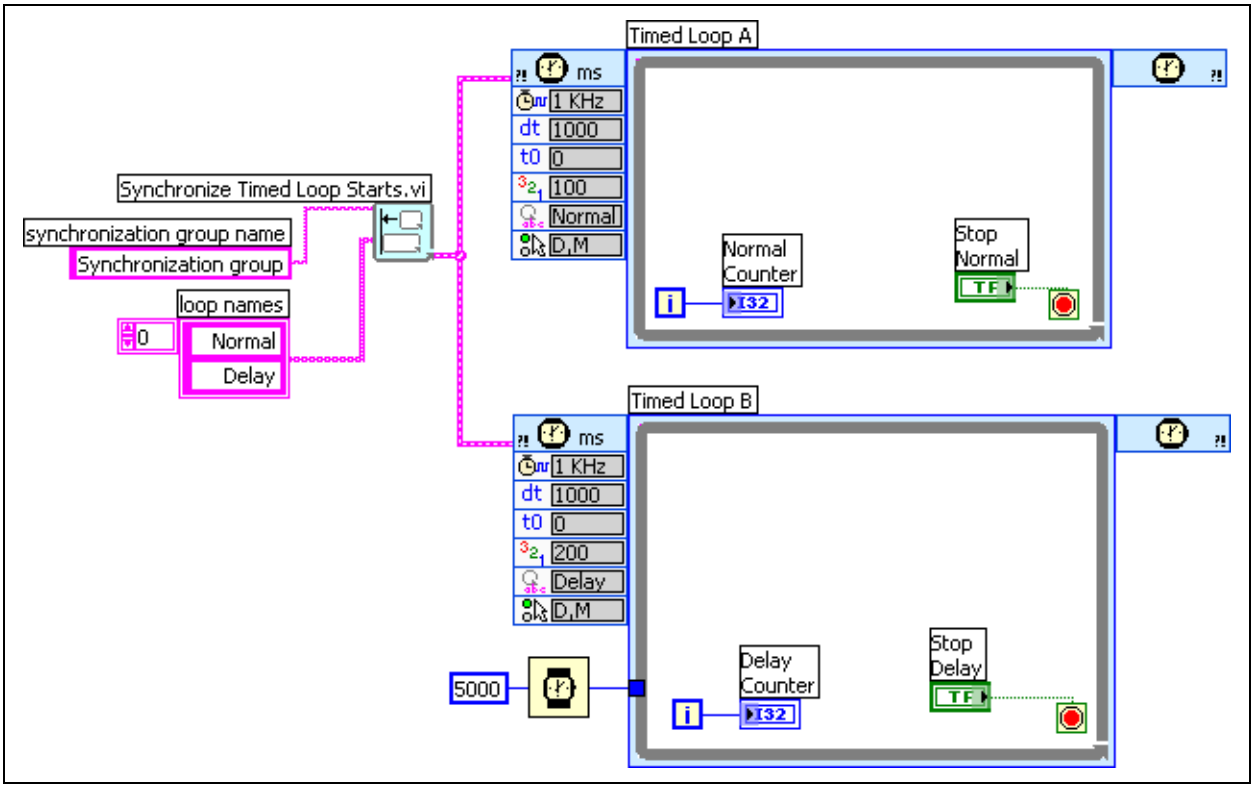

Figure 7. Synchronizing Timed Loops

Using Hardware Timing to Control the Timed Loop

Use the VIs on the hardware-specific palettes, such as the DAQmx - Data acquisition VIs, to configure timing sources that use hardware timing capabilities of devices you have installed.

Timed Loop Execution Overview

The LabVIEW Timed Loop Scheduler, which is part of the LabVIEW Run-Time Engine, manages the execution of Timed Loops. Predefined events within the timing source represent a tick of the timing source. Based on the timing source type and hardware capability, all or some of the events generate interrupts that notify the LabVIEW Timed Loop Scheduler that an event or a timeout occurred. The LabVIEW Timed Loop Scheduler handles the interrupt, checks the list of Timed Loops associated with the timing source, and triggers a new iteration of a Timed Loop if necessary. The LabVIEW Timed Loop Scheduler uses the value of the **Priority** input of the Timed Loop to determine if the loop should preempt the currently running loop iteration. The scheduler also controls which Timed Loop continues its execution after the currently running Timed Loop finishes its iteration. Finally, the scheduler reconfigures the hardware component of the timing source for the next notification.