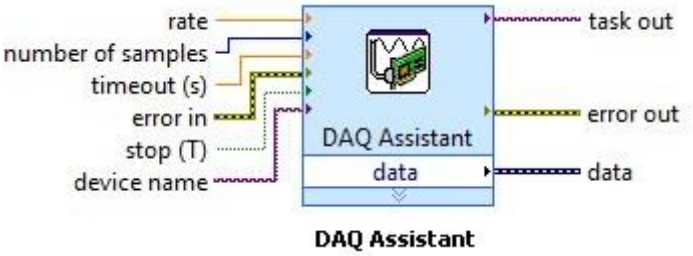
NI DAQmx Programming

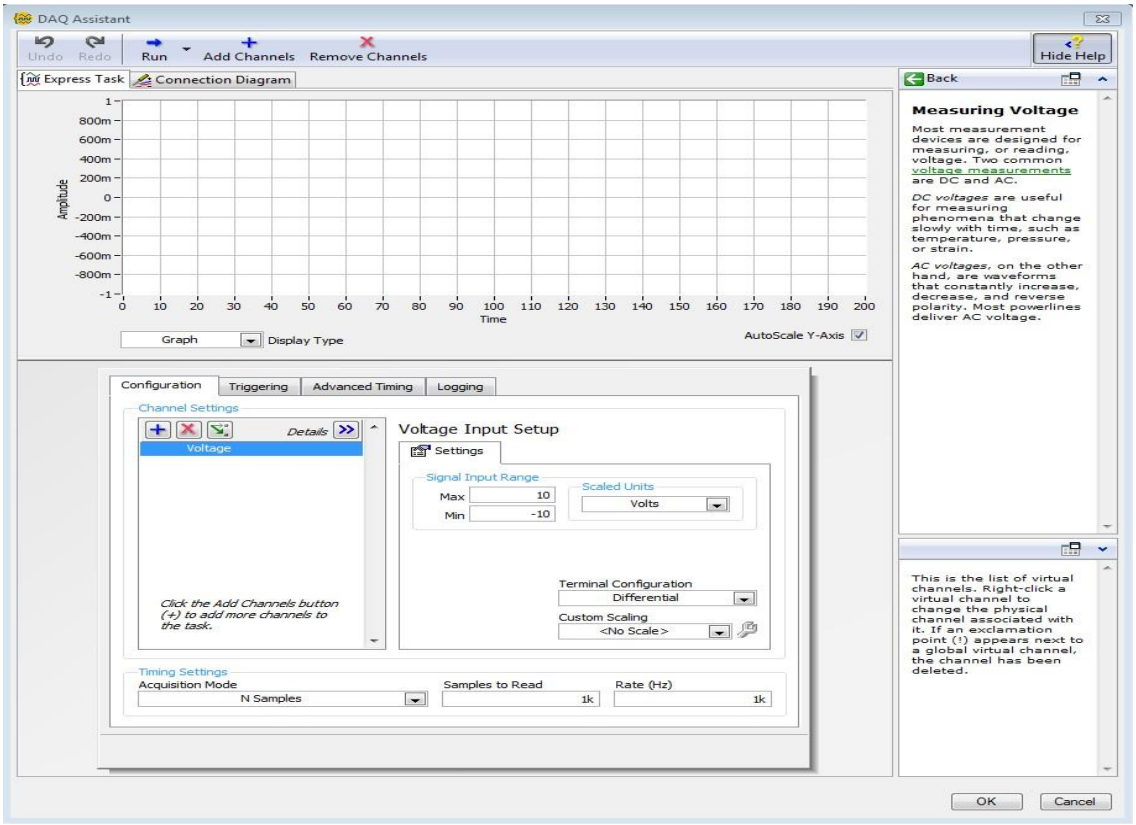
Corporate office: # 22, 2nd Floor, AKAI Plaza, Sanjaynagar, Bengaluru, Karnataka 560094.
Phone: 91-80-40948484 www.visolutions.org **Email:** support@visolutions.org

A. DAQ Assistant



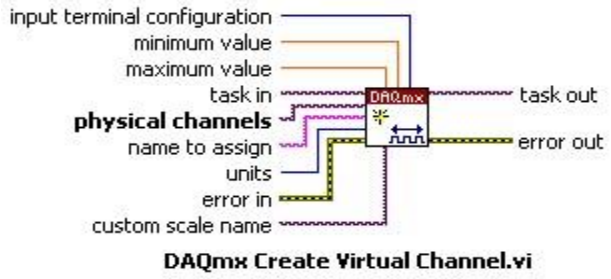
DAQ Assistant

DAQ Assistant is a graphical interface for interactively creating, editing, and running NI-DAQmx virtual channels and tasks. An NI-DAQmx virtual channel consists of a physical channel on a DAQ device and the configuration information for this physical channel, such as input range and custom scaling. An NI-DAQmx task is a collection of virtual channels, timing and triggering information, and other properties regarding the acquisition or generation.



B. DAQmx Programming

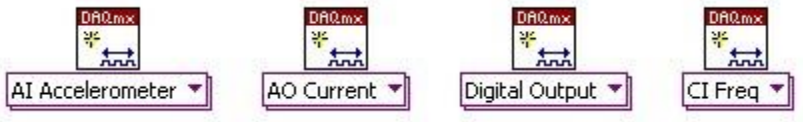
a) NI-DAQmx Create Virtual Channel



The NI-DAQmx Create Virtual Channel function creates a virtual channel and adds it to a task. It can also be used to create multiple virtual channels and add all of them to a task. When a task is not specified, the function creates a task. The NI-DAQmx Create Virtual Channel function has numerous instances. These instances correspond to the specific type of measurement or generation the virtual channel(s) perform.

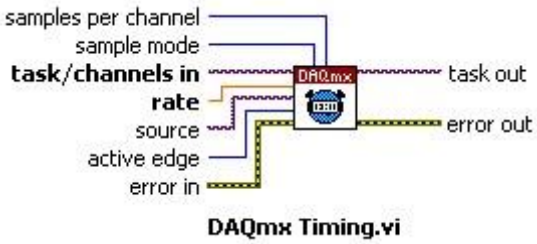
Creating a Channel in LabVIEW

The following figure shows four examples of different instances of the NI-DAQmx Create Virtual Channel VI.



The inputs to the NI-DAQmx Create Virtual Channel function differ for each instance of the function. However, certain inputs are common to most, if not all, of the function's instances. For example, an input is required to specify the physical channels (analog input and analog output), lines (digital), or counter that the virtual channel(s) will use. Additionally, analog input, analog output, and counter operations use minimum value and maximum value inputs to configure and optimize the measurements and generations based on the minimum and maximum expected values of the signals.

b) NI-DAQmx Timing



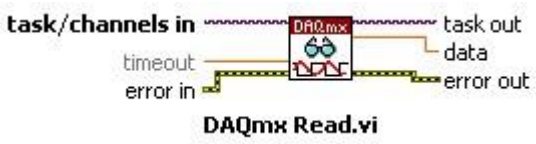
The NI-DAQmx Timing function configures the timing for hardware-timed data acquisition operations. This includes specifying whether the operation will be continuous or finite, selecting the number of samples to acquire or generate for finite operations, and creating a buffer when needed.

For operations that require sample timing (analog input, analog output, and counter), the Sample Clock instance of the NI-DAQmx Timing function sets both the source of the sample clock, which could be an internal or external source, and its rate. The sample clock controls the rate at which samples are acquired or generated. Each clock pulse initiates the acquisition or generation of one sample for each virtual channel included in the task.

c) NI-DAQmx Start task

The NI-DAQmx Start Task function explicitly transitions a task to the running state. In the running state, the task performs the specified acquisition or generation. A task will be implicitly transitioned to the running state and automatically started if the NI-DAQmx Start Task function is not used when the NI-DAQmx Read function executes. This implicit transition also occurs if the NI-DAQmx Start Task function is not used and the NI-DAQmx Write function executes with its auto start input specified accordingly.

d) NI-DAQmx Read

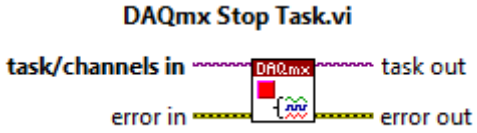


The NI-DAQmx Read function reads samples from the specified acquisition task. The different instances of the function allow for the type of acquisition (analog, digital, or counter), the number of virtual channels, the number of samples, and the data type to be selected.

e) NI-DAQmx Write

The NI-DAQmx Write function writes samples to the specified generation task. The different instances of the function allow for the type of generation (analog or digital), the number of virtual channels, the number of samples, and the data type to be selected.

f) NI-DAQmx Stop



The NI-DAQmx stop function writes stops the running task and returns it to the state the task was in before.

g) NI-DAQmx Clear Task

The NI-DAQmx Clear Task function clears the specified task. If the task is currently running, the function first stops the task and then releases all of its resources.

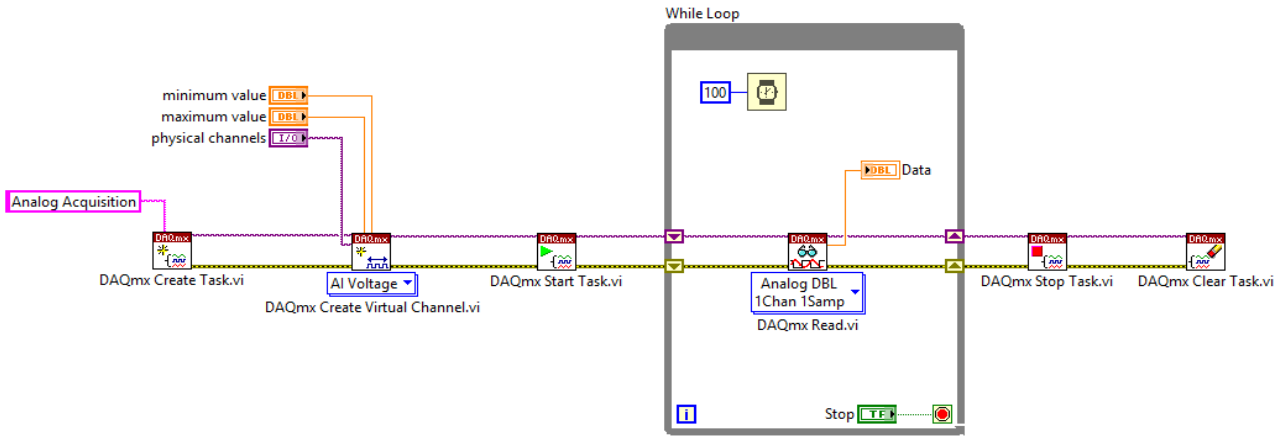
Once a task has been cleared, it cannot be used unless it is recreated. Thus, if a task will be used again, the NI-DAQmx Stop Task function should be used to stop the task but not to clear it.

C. Conclusion

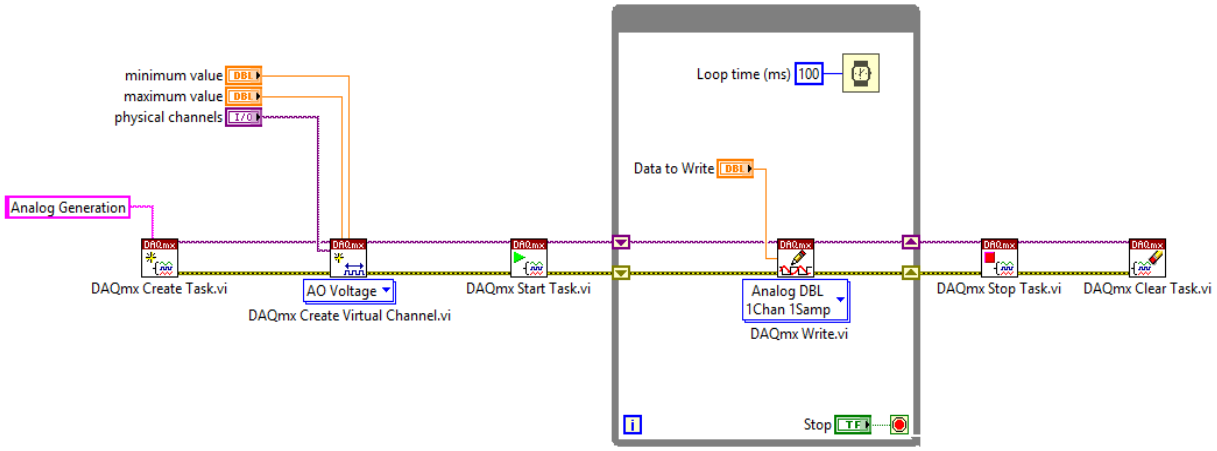
NI-DAQmx saves development time and improve the performance of data acquisition applications. One of the ways NI-DAQmx saves development time is by providing an API that requires only a small number of functions to expose the majority of its functionality.

D. Sample Programs

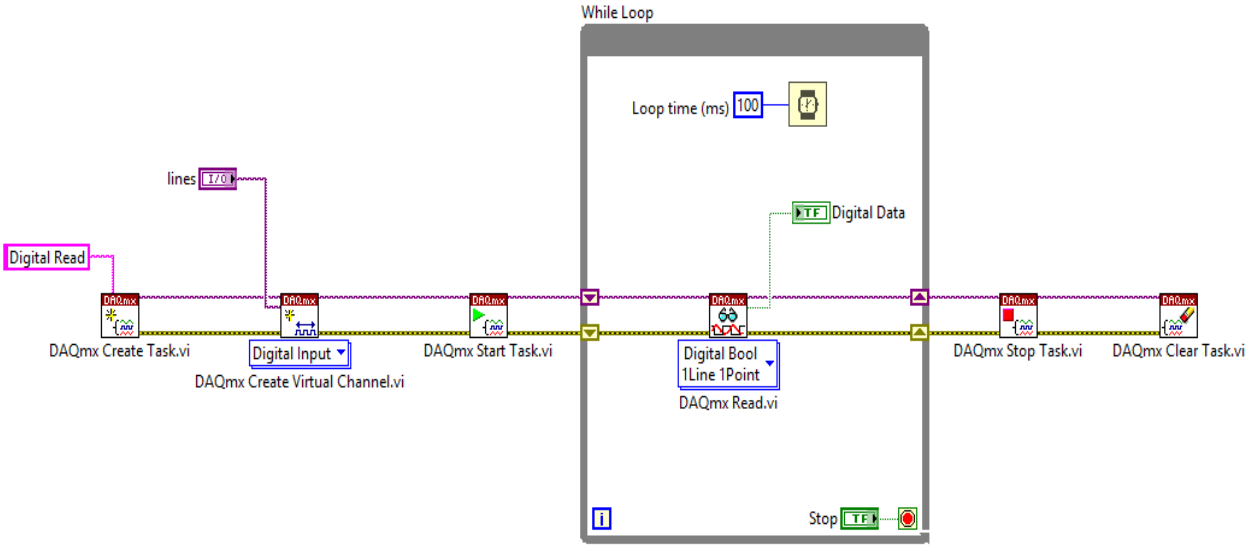
a) Analog Acquisition



b) Analog Generation



c) Digital Acquisition



d) Digital Generation