
**Application Note: Using SENSR Instruments in LabVIEW**

## SENSR Background

SENSR manufactures innovative systems for capturing dynamic measurements, assessing performance and managing risk. Our unique blend of advanced technology and core application experience empowers us to make systems that elevate our customer's abilities.

Our high-value solutions are used to inspect and monitor critical infrastructure, watch over national treasures, safeguard facilities, prosecute and defend legal cases, break world records, engineer better products and advance scientific research.

The systems that we design and manufacture reduce the complexity and costs associated with dynamic monitoring.

You may visit SENSR on the web at http://www.sensr.com.

## Overview

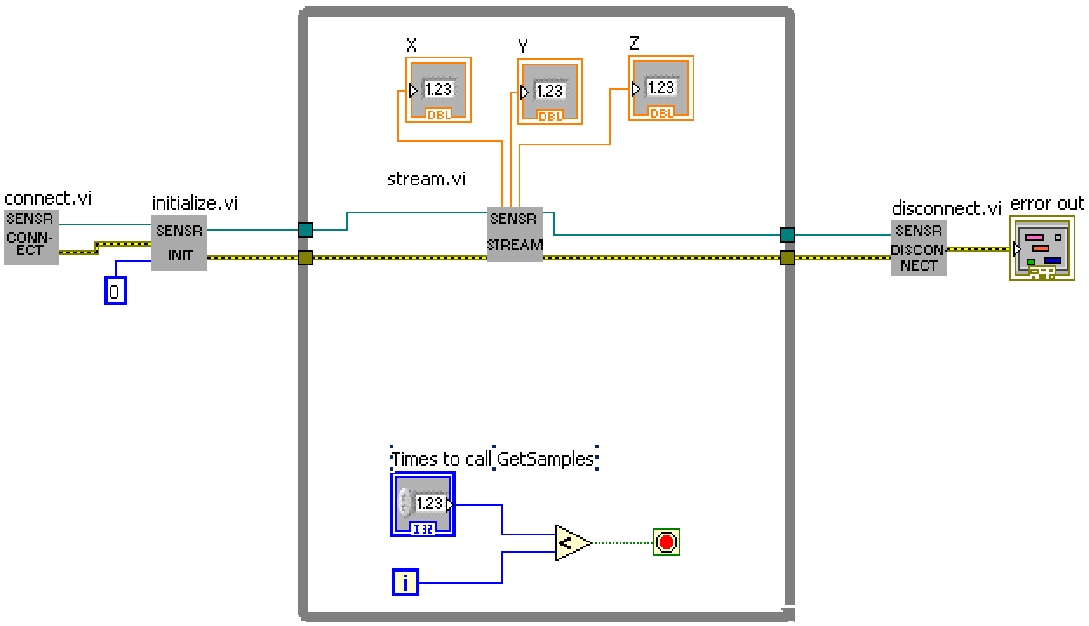
The majority of SENSR instrument users make measurements with the software bundled with each instrument, but SENSR also can provide a device API (application-programming-interface) for the GP series and the CX series of instruments. These APIs are provided in the form of Microsoft .Net Assembly DLL (dynamic-link library) files[1]. SENSR also provides source code for example applications to help users writing custom software for SENSR instruments. This document and the associated LabVIEW files may assist a user in interfacing a SENSR

A LabVIEW user may find using the DLL files more difficult without an example application. In order to help LabVIEW users get started quickly, SENSR provides a set of example LabVIEW VIs (virtual instruments) and sub-VIs. This documentation explains how the example application was constructed.

[1] The DLL files are .Net assemblies.

**Example Application**

SENSR provides an example application that uses sub-VIs to read measurements from a SENSR GP Series instrument. This application is stored in a file called "SENSR LabVIEW stream example.vi" and requires the DLL files and sub-VIs that are also stored in the "GP Series" directory. The LabVIEW block diagram for the test application looks like this:

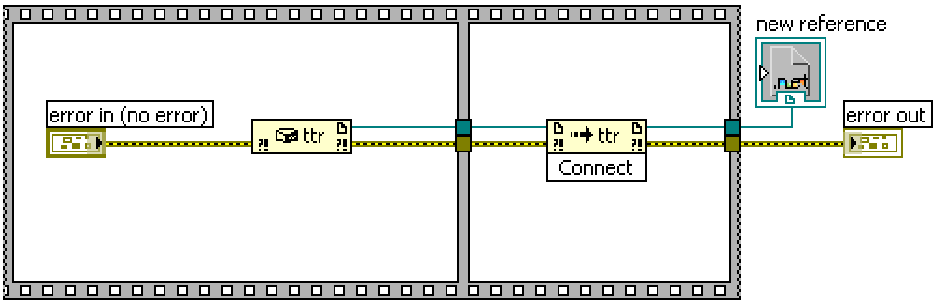



The majority of the interaction with the SENSR DLL files is contained in the sub-VIs. The steps executed are:

1) The first VI, which is named "connect.vi," connects to the SENSR device.
2) The second VI, which is named "initialize.vi," initializes the SENSR device and selects the G-range to be measured.
3) The third VI, which is named "stream.vi," gathers streaming data from the SENSR device. This VI is inside a while-loop structure so it may be called repeatedly to provide measurements. The front panel indicators for the test application are contained in the while-loop so the measurements may be updated.
4) The fourth VI, which is named "disconnect.vi," closes the connection to the SENSR device.

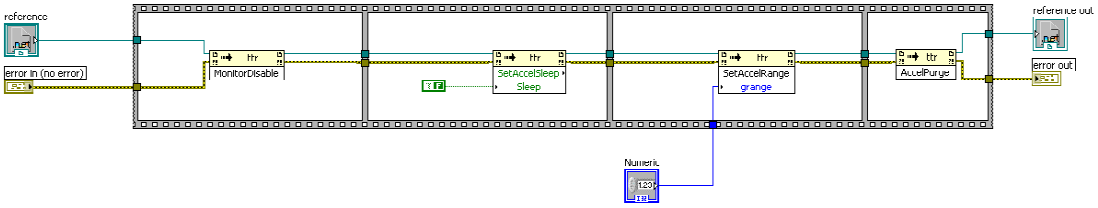
In general, each step must be both completed and in the above order for device communication to work properly.

**Connecting to the SENSR Device**



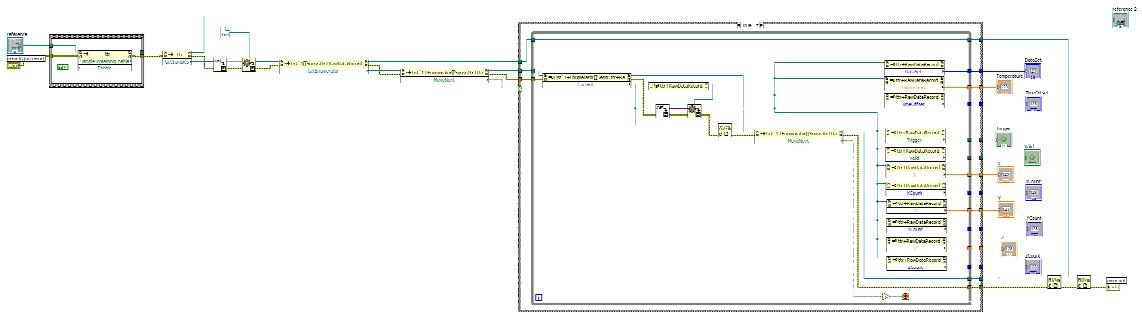

The connect.vi first uses a constructor node to create a ttr object and then uses an invoke node to call the Connect method of the ttr object. The output terminals are connected to a reference to the ttr object and the error out.

**Initializing the SENSR Device**



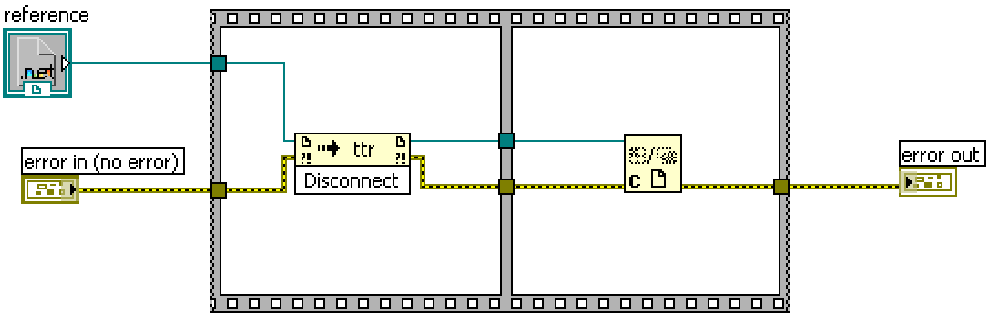

The initialize.vi requires a ttr object reference connected to an input terminal and has a terminal to accept the error in. First, the MonitorDisable method of the ttr object is called. Next, a false constant is input to the SetAccelSleep method of the ttr object to ensure the accelerometer is not sleeping. Next, a numeric constant is input to the SetAccelRange method of the ttr object to select the acceptable limit for acceleration measurement used inside the instrument. Finally, the AccelPurge method of the ttr object must be called to clear out any readings stored in the instrument's buffer. The output terminals are connected to a reference to the ttr object and the error out.

**Retrieving Streaming Data from the SENSR Device**



The stream.vi requires a ttr object reference connected to an input terminal and has a terminal to accept the error in. First, a true constant is input to the SampleStreamingEnable method of the ttr object. Next, the GetSamples method of the ttr object is called and the ttr object returns a list of RawDataRecords as defined in the ttr.dll. A case structure and while loop collect each RawDataRecord in the list and Property nodes break each RawDataRecord down into the data produced by the instrument. Each property is linked to an output terminal. The error out and ttr object reference are also linked to output terminals.

**Closing the connection to the SENSR Device**




The disconnect.vi requires a ttr object reference connected to an input terminal and has a terminal to accept the error in. First, the disconnect method of the ttr object is called. Next, the ttr object reference is closed. The output terminal is connected to the error out.