```
#include <NIDAQmx.h>

#include <iostream>
#include <string>

void HandleError(TaskHandle handle)
{
	char errBuff[2048] = { '\0' };
	DAQmxGetExtendedErrorInfo(errBuff, 2048);

	std::cout << "Error " << std::string(errBuff) << std::endl;

	system("PAUSE");

	if (handle)
	{
		DAQmxStopTask(handle);
		DAQmxClearTask(handle);
	}

	exit(EXIT_FAILURE);
}

int main(int argc, char *argv[])
{

	int32 returnvalue = 0;

	TaskHandle  taskHandle = nullptr;

	const int noOfCHannels = 4;
	float64     data[noOfCHannels];
	std::string chIDs[noOfCHannels] = { "ai1", "ai5", "ai9", "ai13" };     //
physikal names of the ports

	// DAQmx analog voltage channel and timing parameters

	returnvalue = DAQmxCreateTask("TestName", &taskHandle);

	if (DAQmxFailed(returnvalue))
		HandleError(taskHandle);


	for (int i = 0; i < noOfCHannels; ++i)
	{

		returnvalue = DAQmxCreateAIVoltageChan(taskHandle, ("Dev1/" +
chIDs[i]).c_str(), "", DAQmx_Val_NRSE, -10.0, 10.0, DAQmx_Val_Volts, NULL);
		if (DAQmxFailed(returnvalue))
			HandleError(taskHandle);
	}

	returnvalue = DAQmxCfgSampClkTiming(taskHandle, "", 1.0, DAQmx_Val_Rising,
DAQmx_Val_ContSamps, 1);      // continuous measurement with 1 Hz

	if (DAQmxFailed(returnvalue))
		HandleError(taskHandle);


	// DAQmx Start Code

	returnvalue = DAQmxStartTask(taskHandle);

	if (DAQmxFailed(returnvalue))
```

```
            HandleError(taskHandle);


      // DAQmx Read Code
      int32       read;

      while (true)
      {
            // Read data
            returnvalue = DAQmxReadAnalogF64(taskHandle, 1, 10.0,
DAQmx_Val_GroupByChannel, data, noOfCHannels, &read, NULL);

            if (DAQmxFailed(returnvalue))
                  HandleError(taskHandle);

            std::cout << std::endl;

            for (int i = 0; i < noOfCHannels; ++i)
            {
                  std::cout << "Ch " << chIDs[i] << ": " << data[i] <<
std::endl;
            }

      }

      // Stop and clear task
      DAQmxStopTask(taskHandle);
      DAQmxClearTask(taskHandle);

      system("PAUSE");
      return EXIT_SUCCESS;
}
```