**Using the LabView to SDK Program**

**Software Requirements**: The software requires LabView 2010 or later, and Cortex\_SDK.dll (64-bit) version 1.4.0 or later. It was developed using LabView 2010 Professional software.

**Description of Sample Code**: The file name for the main program is *Main\_RT Feedback.vi*. The program extracts streamed data from Cortex to drive a 3D cylinder model composed of a torso and arms. It uses the SDK provided by Motion Analysis to interact with the dynamic link libraries that provide access to the motion capture data as it is collected. The components of the program that are used to create the cylinder model can easily be adapted to other uses, and there is substantial documentation in the program to assist the user in identifying the sections of code that can be adapted.

**Files included with the Sample**: The following LabView subVi’s are needed to run *Main\_RT Feedback.vi* and should all be contained in the same directory:

[R]\_ZYX.vi

Access DLL.vi

AP Position.vi

Create 3D Axes.vi

Create 3D Cylinder.vi

Create 3D Sphere.vi

Create Animation.vi

Create Arm.vi

Create Body.vi

EventCallBack.vi

Feedback [R]s.vi

HELICAL.VI

L Arm Initial Position.vi

Main\_RT Feedback.vi

R Arm Initial Position.vi

RT Animation.vi

RT Positions.vi

Write TRC.vi

The sample .TRB and .PRJ files included with this program are:

Test\_3D\_Feedback.trb

Test\_3D\_Feedback.prj

In addition, Cortex\_SDK.dll (64-bit version) is also required to be present on the machine running the LabView code.

**Running the Program**: Before starting the program, make sure that the machine running the LabView code is networked to the machine running Cortex. Also, make sure that the project file (Test\_3D\_Feedback.prj) and the data file (Test\_3D\_Feedback.trb) are loaded into Cortex. The “Post Process” tab should be selected, and “Run” should be pressed so that the stick figure is animated in the Cortex 3-D View.

To start *Main\_RT Feedback.vi*, enter the IP address of the machine running LabView into the box labeled “Local IP” (Figure 1). Next, enter the IP address of the machine running Cortex into the box labeled “Host IP”. Finally, click on the folder icon to the right of the box labeled “DLL Path”, then browse to and select the file labeled “Cortex\_SDK.dll”. For the sample TRB file provided with this program, the subject moved their right shoulder, so the button labeled “Shoulder” should be set to “Right”. If “Write Data to TRC” is set to “On”, then the program will write out the last frame collected once the user presses the “Stop” button.

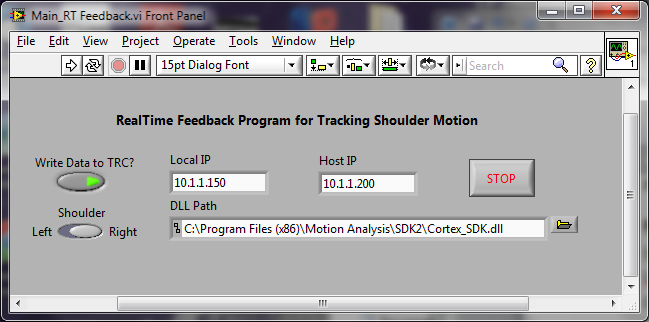


Figure 1. Front panel of Main\_RT Feedback.vi.

When run (by pressing ) a window containing the animated figure will open (Figure 2). The figure will follow the exact same movement pattern as the corresponding stick figure in Cortex.

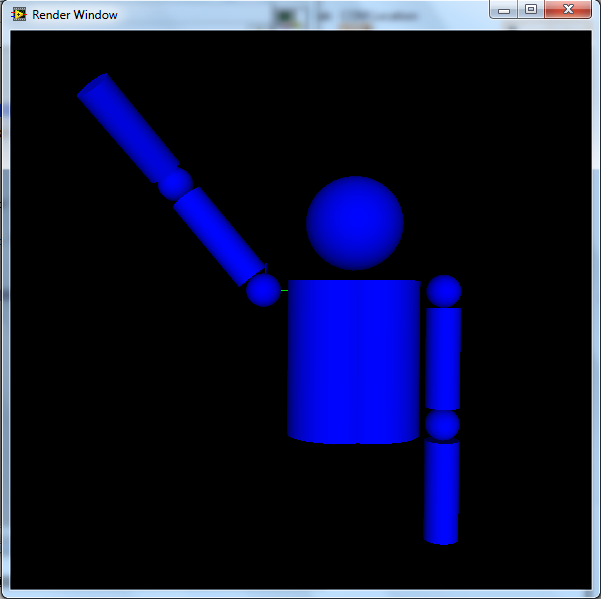


Figure 2. Animated figure that displays arm motion. This figure can be rotated inside of the window by moving the mouse inside of the window while pressing the left button.

**Main\_RT Feedback.vi Documentation**: The program is organized by frames contained in a sequence structure. The code contained in Frame 0 executes first, followed by the code contained in Frame 1, then the code in Frame 2, and so forth.

The code in Frame 0 (Figure 3) establishes the parameters that are used for the creation of the 3D Picture. This frame is not necessary to access the data stream from Cortex. It is included to illustrate how the LabView 3D Picture tools can be used in conjunction with the code used to access Cortex\_SDK.dll.

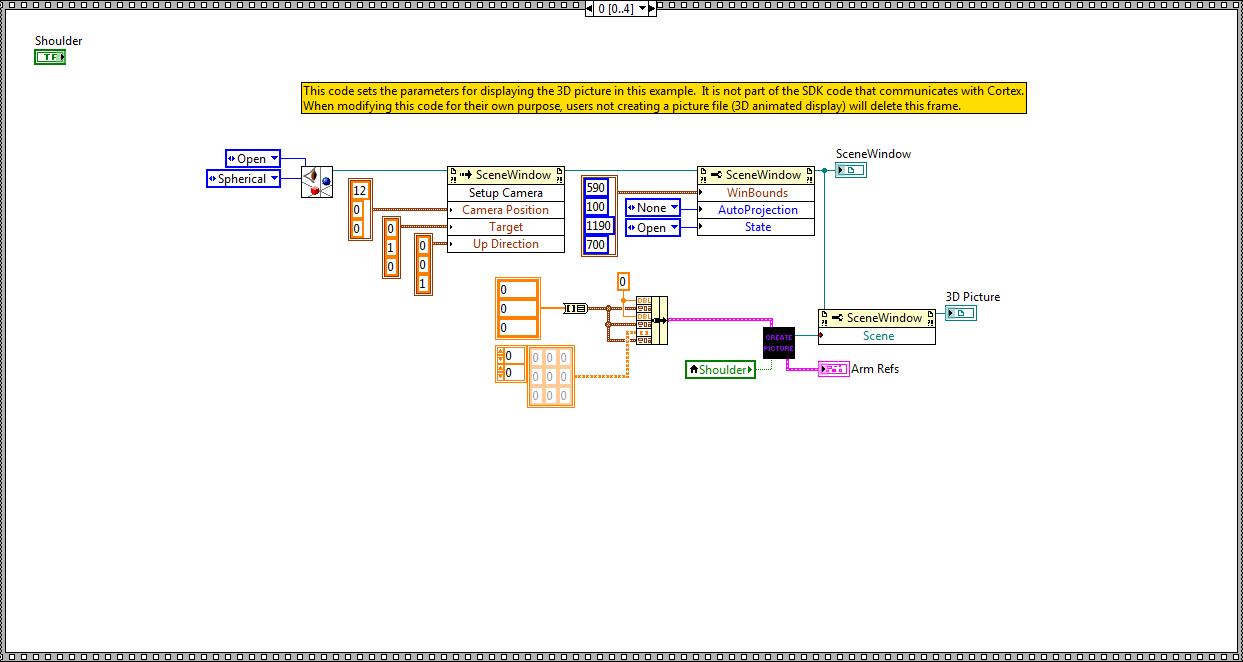


Figure 3. Frame 0 of *Main\_RT Feedback.vi.*

The code in Frame 1 (Figure 4) initializes the connection between the machine running Cortex (Host IP) and the machine running the LabView code (Local IP). This is the first step required to access the data stream from Cortex.

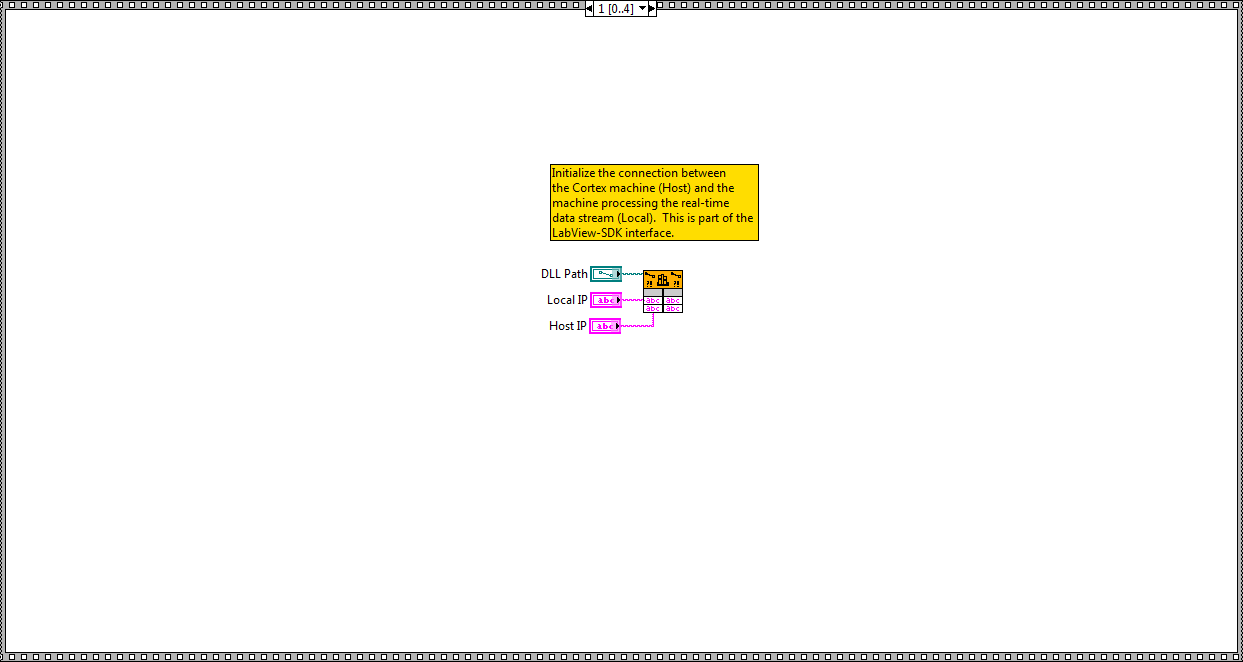


Figure 4. Frame 1 of *Main\_RT Feedback.vi.*

The code in Frame 2 (Figure 5) creates the structure for sBodyDefs as defined in the Cortex.h documentation, then retrieves the sBodyDefs data. Multiple bodies may be defined within a project. This code was designed to access the number of markers and the corresponding marker names from the first body definition only. This code in this frame is recommended to access the data stream from Cortex.

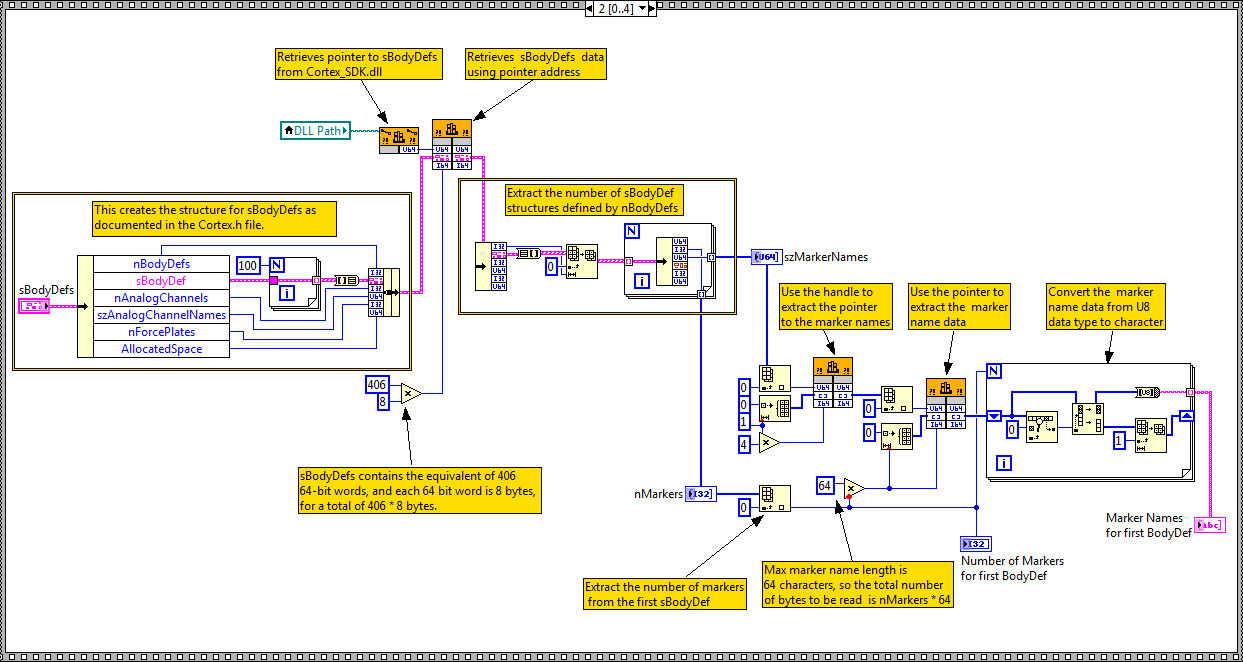


Figure 5. Frame 2 of *Main\_RT Feedback.vi.*

The code in Frame 3 (Figure 6) contains a subVi called Access DLL.vi which retrieves the data for each frame collected in Cortex. The code immediately to the right of Access DLL.vi transforms the coordinate data stream into a 2D array of marker coordinates. Finally, the third (rightmost) block of code inside of the WHILE loop is used to create the 3D display. Of the code inside of the WHILE loop, only the Access DLL.vi subVi is necessary to stream data from Cortex, although the code that transforms the data stream into a 2D array is helpful.

The code in the block below the WHILE loop extracts the coordinate data from the unnamed markers. It is currently outside of the WHILE loop since this example saves only the data from the last frame collected before the user presses the STOP button on the front panel. If the user wishes to stream the unnamed markers in every frame, this block should be placed inside of the WHILE loop.

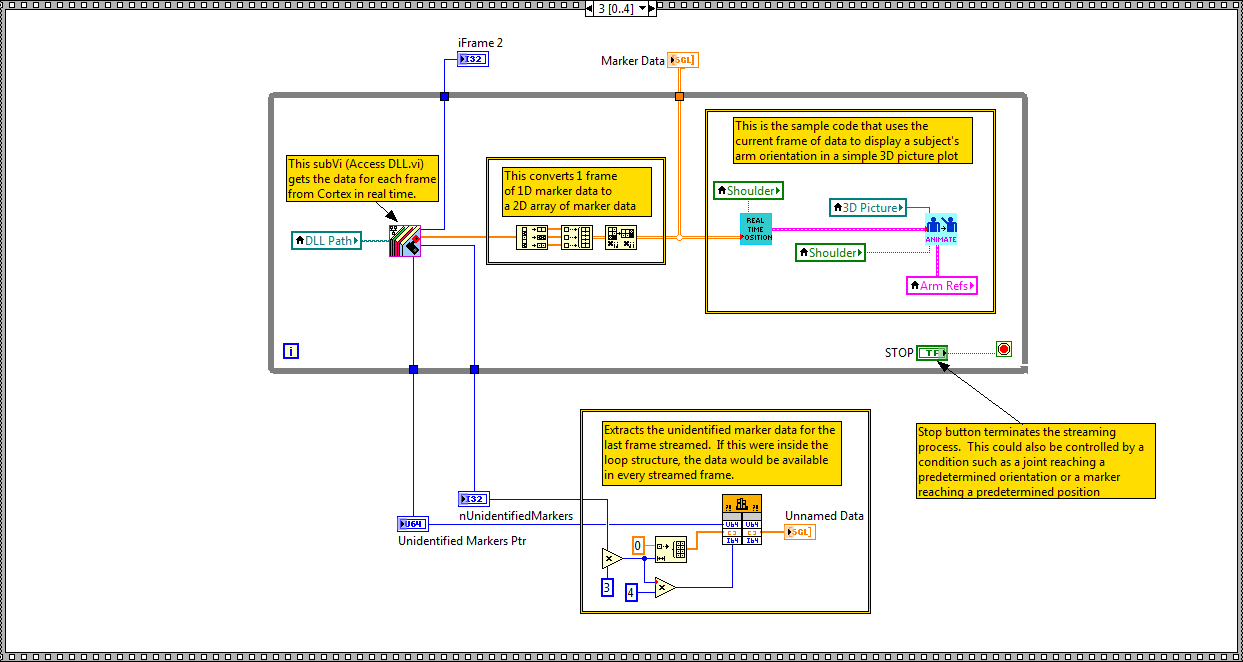


Figure 6. Frame 3 of *Main\_RT Feedback.vi.*

The code in Frame 4 of *Main\_RT Feedback.vi* (Figure 7) has 2 functions: the top block terminates the connection with Cortex, and this is a required part of the streaming code. The bottom block of code writes the last frame of data collected to a text file in TRC format.

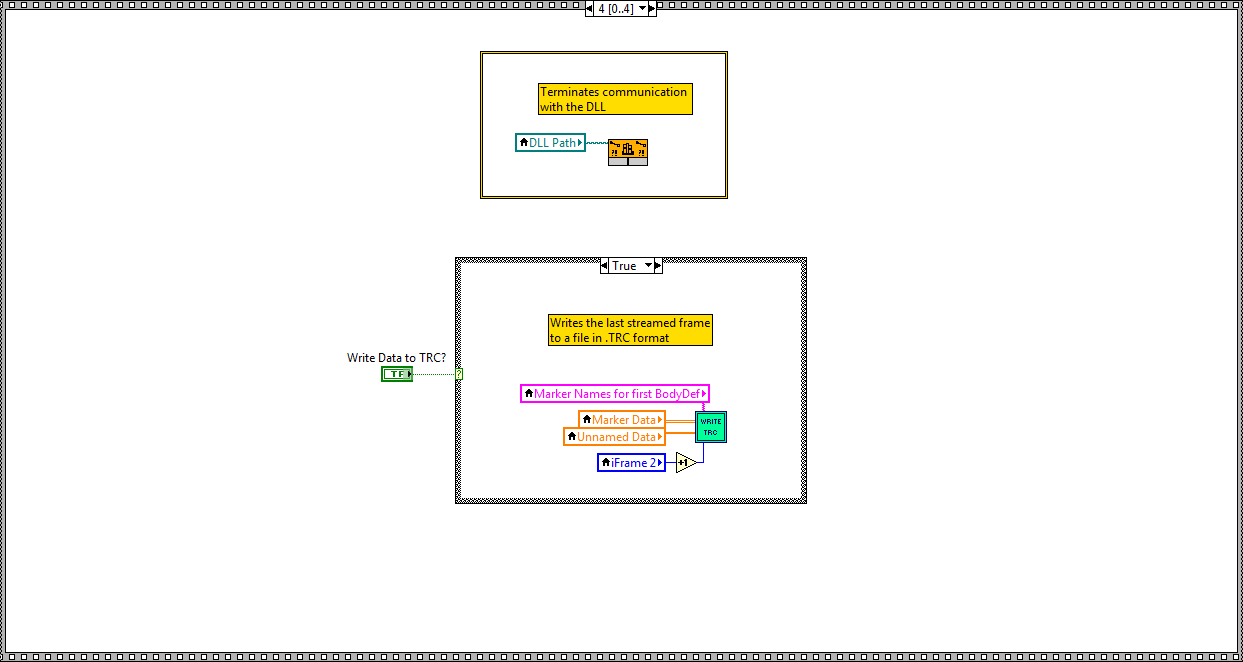


Figure 7. Frame 4 of *Main\_RT Feedback.vi.*

**Access DLL.vi Documentation:** *Access DLL.vi* is responsible for extracting the data from each frame collected in Cortex. Frame 0 of the *Access DLL.vi* code (Figure 8) simply instructs Cortex to provide a pointer to the data collected for the most recent frame. This code must be included as part of the streaming process.

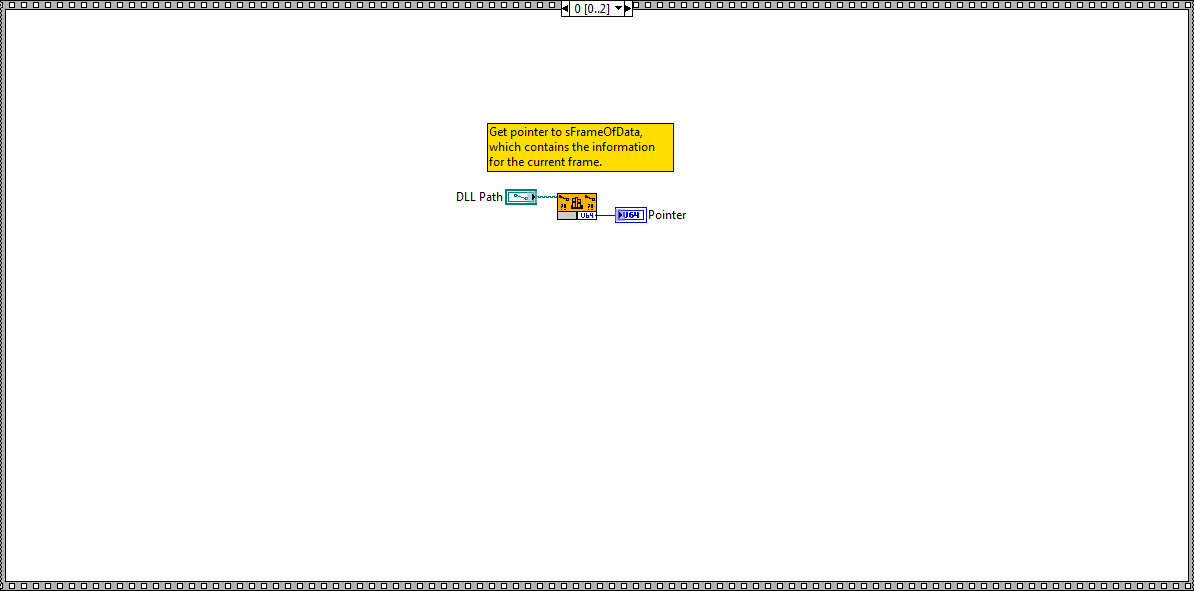


Figure 8. Frame 0 of *Access DLL.vi.*

Frame 1 of *Access DLL.vi* (Figure 9) defines the structure for a frame of data as specified in the Cortex.h documentation. This is accomplished in the code inside of the box in the left half of Frame 1. The code then uses the pointer obtained in Frame 0 to retrieve the data for the current frame. The code inside of the box on the right side of the frame then extracts the number of markers and the pointer to the marker data for the first defined body (identified by the constant 0 inside of the box). If more bodies were defined, this segment of code would be placed inside of a loop. All of the code in Frame 1 of Access DLL.vi must be included as part of the streaming process.

While the analog data has not been extracted for use in this example code, its implementation would parallel that used for extracting the marker data. Specifically, nAnalogChannels and nAnalogSamples would be used to calculate the number of bytes to extract using the pointer AnalogSamples. These values would be applied in the same manner as the MarkersPtr and nMarkers values in Frame 2 of *Access DLL.vi.*

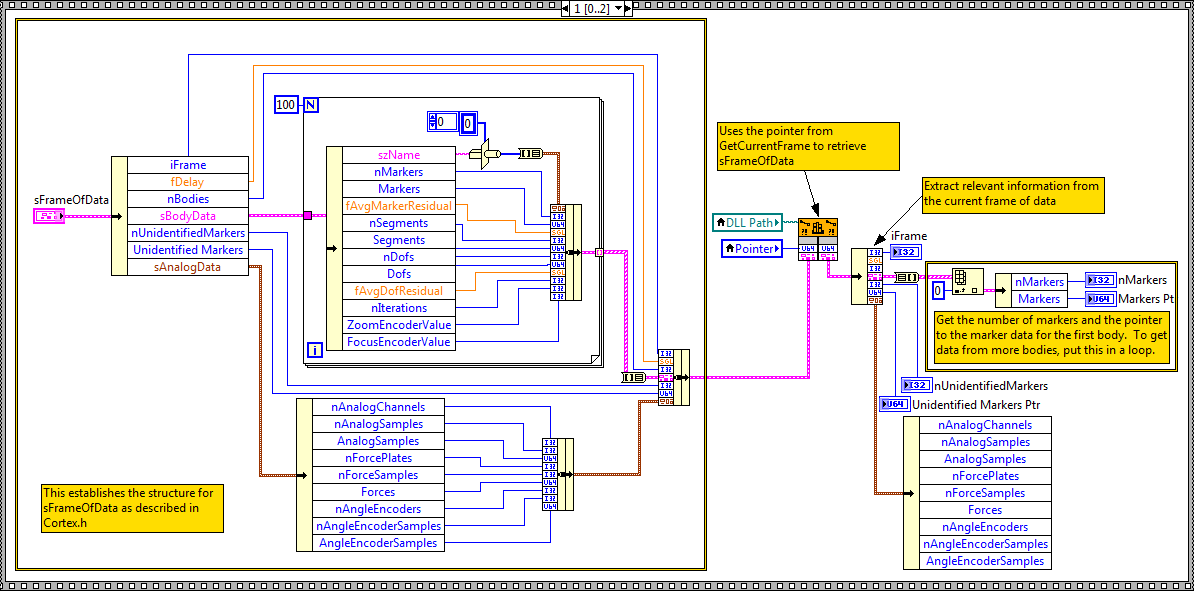


Figure 9. Frame 1 of *Access DLL.vi.*

Frame 2 of *Access DLL.vi* (Figure 10) uses the number of markers and the pointer to the data obtained in Frame 1 of Access DLL.vi to retrieve the data. This code is also necessary for streaming the data from Cortex.

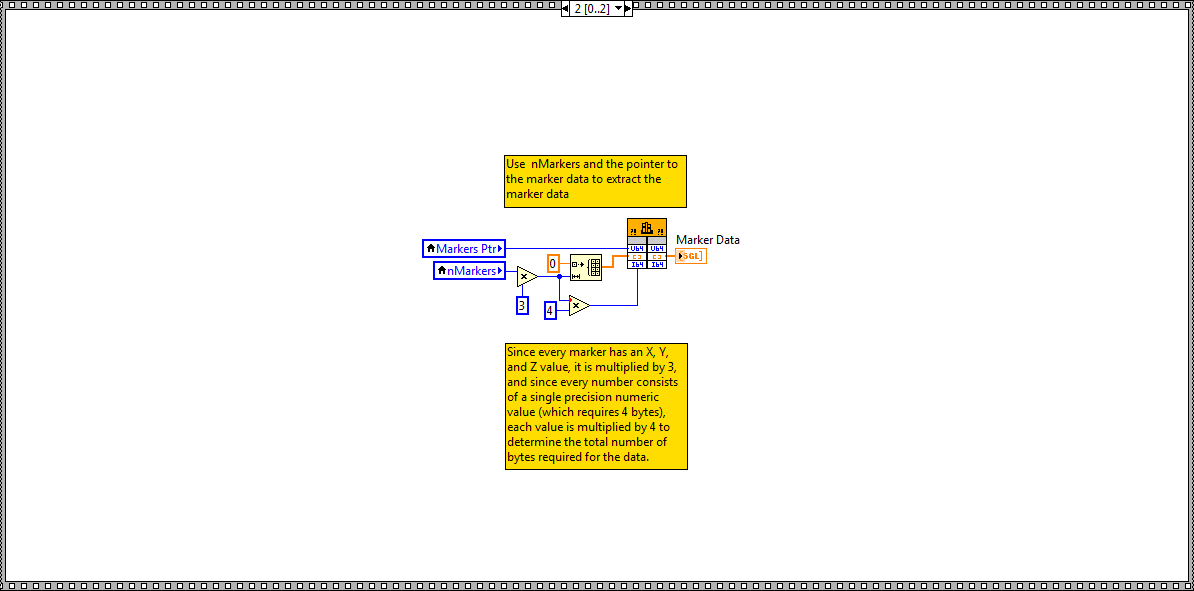


Figure 10. Frame 2 of *Access DLL.vi.*