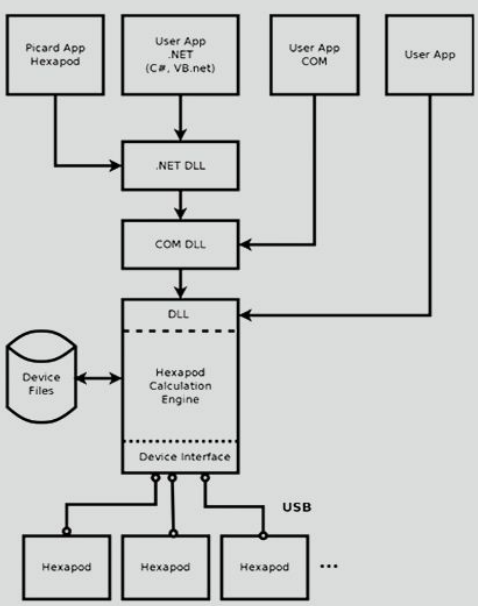
MHP Software API

The MHP software can be accessed as a library (or API - Applications Programming Interface) for controlling a Picard Industries Mini Hexapod System.

Users can use the API to write their own programs to monitor and control (move) the platform. The user interface we provide uses this API and can be used as an example to follow its use.

- .Net component for all Visual Studio .Net languages.
- COM (ActiveX) component for most languages including scripting languages.
- DLL component for any language that can call functions in a standard Windows DLL.
- 32-bit and 64-bit components.
- Windows 7, and higher.
- Control multiple hexapod devices.
- Device interface layer for all Picard Industries Hexapod (MHP) devices.
- Device files to configure for different hexapod devices.
- Simulation capability.
- API documentation as Windows Help file.
- Sample programs in C++ and C#.



Picard Industries

4960 Quaker Hill Road, Albion, NY 14411

(585) 589-0358

www.Picard-Industries.com

MHP Software API

API Overview

The API provides a complete set of functions for monitoring, controlling, and managing an MHP system.

Setup: The setup functions allow you to search for attached devices, and find their address, serial number, and status. Once you have identified the device, you can open the device and connect to it.

Coordinate Systems and Named Positions: You can create, modify, and delete Coordinate Systems and Named Positions. Coordinate Systems are the basic building blocks of hexapod positioning. A coordinate system defines a position and an orientation (direction) in space. You specify a coordinate system using a 6-element array, which represents translation in X, Y, and Z, and rotation about the X, Y, and Z axes. Alternately, you can specify a coordinate system using a 4x4 transformation matrix. There are several built-in coordinate systems, including the "World" coordinate system that serves as the master coordinate system. You can define as many additional coordinate systems as you wish.

Any coordinate system can be used as a "Virtual Pivot Point" – a point in space about which the system rotates. For example, you can define a pivot location 3 meters above the base, and have the platform pivot about that location.

Named Positions are a simple way to remember a particular position and orientation in space, allowing you to easily move to a memorized or trained location.

Coordinate Systems and Named Positions are stored in the device files associated with each device, so they can be shared among applications. For example, you can use the Picard MHP application to define Coordinate Systems and/or Positions, and then reference those Coordinate Systems and Positions in your own applications.

Motion: The motion functions allow you to translate and rotate the platform. Motions can be relative to any coordinate system. You can set the velocity and acceleration to be used during subsequent moves.

Status: You can check the status of a device to see if it is moving or homing, and whether it is physically present or simulated. You can get the current position relative to any coordinate system.

Simulation: The library provides simulation for all supported MHP devices. This allows you to develop and test software independently from the hardware. The software can simulate the actual speed of the Hexapod device, or use an "instantaneous motion" feature for rapid software testing.

Components

The Hexapod Library consists of three components:

1. Interop.HexapodCOMLib.dll. Use this component if you are writing a managed application in a .NET language such as C# or Visual Basic.Net. This component is a wrapper for the COM component.

2. HexapodCOM.dll. Use this component if you are writing an un-managed application in a language that can use COM (ActiveX) components. This includes unmanaged Visual C++ and most other languages on Windows, such as Python, Java, etc.

3. HexapodDLL.dll. Use this component for any other language that can call a standard windows DLL, or if you need to avoid using COM components.

All components are provided in both 32-bit and 64-bit versions, allowing you to build 32-bit or 64-bit applications.

Documentation

The MHP Library API is described in a Windows Help (chm) file.

Sample Programs

The installation includes two sample programs. One is a .Net application written in C# and the other is an unmanaged application written in C++. These provide an introduction to using the library and can be used as the starting point for your own application.