

Spider-80 API Product Brochure

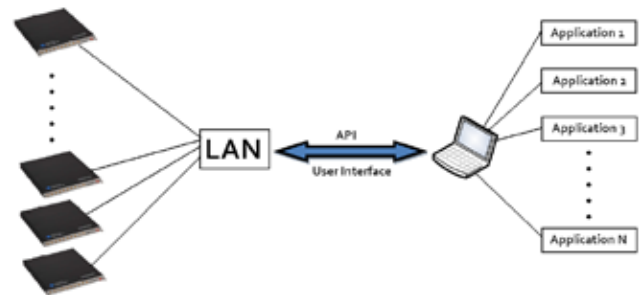
Crystal Instruments (CI) Application Programming Interface (API) for the Spider-80 provides scientists and engineers a powerful tool to develop custom applications that interface directly with CI's industry leading hardware. With over 15 years of experience, Crystal Instruments is dedicated to developing advanced DSP technologies and signal processing solutions. The Spider-80 is a highly modular, distributed, scalable dynamic measurement system. It is ideal for a wide range of industries including automotive, aviation, aerospace, electronics, and military use. The Spider-80 excels in applications requiring easy, quick, and accurate data recording and real-time signal processing.



With the Spider-80 API, users can focus on the interface of their applications and leave the hardware design to Crystal Instruments. While the Spider system is running, user have access to the real-time signal data in both the time and frequency domain. Additionally, with the long-time data recording function the Spider-80 can record up to 4 GB of continuous time data to its own internal memory.

API Basics

The Spider-80 API is implemented in the Microsoft Visual Studio development environment and .NET framework. It provides a high level interface accessible in Visual C++, C#, and Basic. In addition, it can be used by any programming language that supports Dynamic-Linked Libraries (DLLs). Command can configure the Spider front end, control data acquisition, check the status of the processor, and retrieve DSP data.



Run API in Simulation Mode

The Spider-80 API includes a Simulation Mode. This useful tool allows the user to develop applications even if the hardware is not connected. This tool also makes training and development much more efficient, especially when hardware is not continuously available.

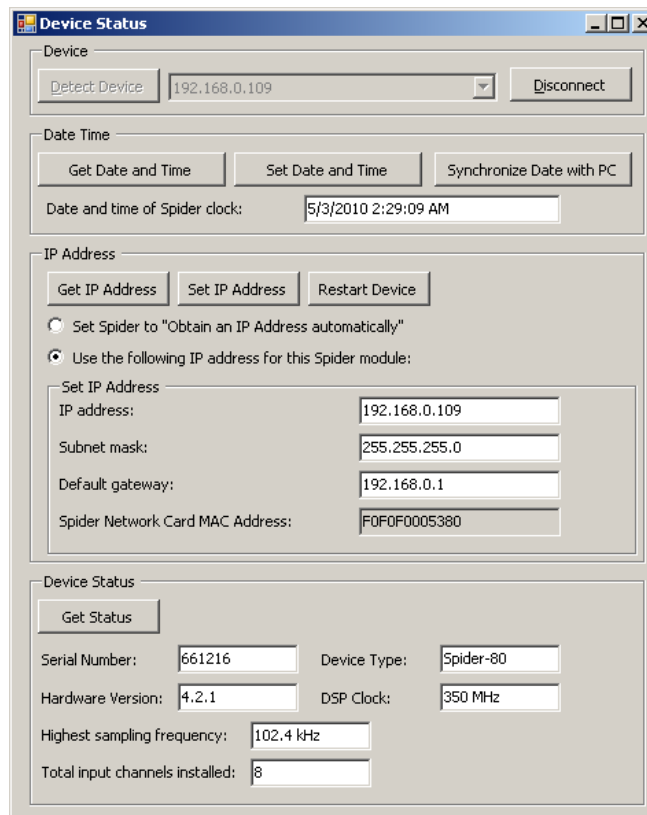
What is in the API Package?

- One or More Calibrated Spider-80 Systems
- Spider-80 API Libraries
- Three API Samples
- Source Code
- Spider-80 API Product Brochure
- Spider-80 API User's Guide

With these tools offered by Crystal Instruments, users are able to easily build customized solutions. The Spider-80 system provides high quality data capturing and real-time processing. The Spider-80 API libraries provide useful function calls to control the powerful hardware. The API samples help the user verify the connection and demonstrate how to set up front-end parameters and triggers. Source code helps programmers understand how the API works internally and reduces workload.

API Samples

Get Status



The screenshot shows the 'Device Status' application window. It contains several sections for configuring and monitoring the device:

- Device:** Includes a 'Detect Device' button, a dropdown menu showing '192.168.0.109', and a 'Disconnect' button.
- Date Time:** Includes buttons for 'Get Date and Time', 'Set Date and Time', and 'Synchronize Date with PC'. Below these, it shows 'Date and time of Spider clock: 5/3/2010 2:29:09 AM'.
- IP Address:** Includes buttons for 'Get IP Address', 'Set IP Address', and 'Restart Device'. It has two radio button options: 'Set Spider to "Obtain an IP Address automatically"' (unselected) and 'Use the following IP address for this Spider module:' (selected). Below this, there are input fields for 'IP address: 192.168.0.109', 'Subnet mask: 255.255.255.0', 'Default gateway: 192.168.0.1', and 'Spider Network Card MAC Address: F0F0F0005380'.
- Device Status:** Includes a 'Get Status' button and a table of device information:

Serial Number:	661216	Device Type:	Spider-80
Hardware Version:	4.2.1	DSP Clock:	350 MHz
Highest sampling frequency:	102.4 kHz		
Total input channels installed:	8		

Measurement Sample

The screenshot shows the 'Measurement Sample' software interface. It includes sections for Device, Test, Test Status, DSA Parameter, Output, Channel Status, Channel Table, Record Files, and Test Data.

Channel Table:

Measurement Quantity	Unit (EU)	Sensitivity (mV/EU)	Input Mode	High-Pass Filter Fc
1 Acceleration	m/s ²	2.047	IEPE	2
2 Velocity	m/s	3600	DC-Single End	1
3 Pressure	Pa	0.0425	DC-Differential	0
4 Force	Newton	1625	IEPE	1.35
5 Frequency	Hz	0.026	AC-Differential	0
6 Sound Press...	Pa	0.06	AC-Single End	0.5

Record Files:

Index	File Name	Size	Created Date
1	CAE1ACE7	1.09 MB	5/18/2010 5:13:27 PM
2	PS2CC06	12.94 MB	5/18/2010 5:19:29 PM
3	1DCAE0E8	7.76 MB	6/7/2010 1:57:09 AM
4	AP05AE87	2.90 MB	6/9/2010 10:16:44 AM
5	302D6F3	2.51 MB	6/21/2010 3:39:22 PM

Test Data:

Channel ID: PT1 | Save Signal | Get Signal Stat | Block Size: 1024

Trigger Sample

The screenshot shows the 'Trigger Sample' software interface. It includes sections for Device, Test, Test Status, DSA Parameter, Trigger Parameter, Channel Table, Test Data, and Trigger.

Trigger Parameter:

Mode: Manual-Arm Trigger | Source: PT1
 Delay Point: 256 pt | Delay Time: 60 ms
 High Threshold: 50.52 st | Low Threshold: 0.0000 st
 Condition: Trigger Level > High Edge

Channel Table:

Measurement Quantity	Unit (EU)	Sensitivity (mV/EU)	Input Mode	High-Pass Filter Fc
1 Acceleration	m/s ²	2.047	IEPE	2
2 Voltage	V	1000	DC-Single End	2
3 Voltage	V	1000	DC-Single End	2
4 Voltage	V	1000	DC-Single End	2
5 Voltage	V	1000	DC-Single End	2
6 Voltage	V	1000	DC-Single End	1

Test Data:

Channel ID: PT1 | Save Signal | Get Signal Stat | Block Size: 1024

Trigger:

Accept | Reject

Full Range of Support

The most challenging part of any development project is the beginning. But Crystal Instruments is here to help. We can work with users and help to define the fundamental requirements, such as defining parameters, commands, and control settings. CI can even deliver an alpha version of the user's application which includes the basic interfaces and commands needed to interact with the Spider-80 hardware.

Crystal Instruments provides a one-year hardware warranty and comprehensive tech support for each purchase. When it is time to recalibrate the Spider-80 hardware, specialized software is available allowing the user to calibrate the system, or the system can be shipped back to Crystal Instruments for calibration.

Data Capturing and Processing

The Spider-80 API can control the hardware to function as both a data recorder and dynamic signal analyzer at the same time. All time stream signals can be simultaneously recorded and displayed.

Acquisition mode controls how the data is acquired block-by-block and processed with signal analyzer functions. These time blocks can be either gap free, with gaps, or overlapped depending on the acquisition mode selection. Real-time processing has a 46 kHz spectral bandwidth with all inputs enabled (102.4 kHz sampling rate). The sampling rate can be set in 54 increments.

Analysis functions include time capture, APS, FRF, and correlation with many windowing options available. Output source waveforms include sine, triangle, square, white noise, DC, chirp, and swept sine. Averaging can be applied to the frequency data with linear, exponential, overlap, or peak hold options. Triggering includes free-run, continuous after trigger, single shot by user, auto/manual-arm trigger.

Front end setup includes the following steps:

- Type of test (Time Stream, Block, APS, FRF)
- DSA parameters (Analysis Frequency, Block Size/Line, Window Type, Overlap Ratio, Average Mode, and Average Number)
- Output Channels (Channel Selection, Type, Amplitude, Frequency)
- Input Channels (Location ID, Measurement Quantity, Engineering Unit, Sensitivity, Input Mode, High-Pass Filter)
- Trigger Parameters (Mode, Source, Delay Point, Delay Time, High Threshold, Low Threshold, Condition)

API Specification

Spider API Methods			
Connection Methods		Trigger Methods	
GetDeviceList	Get all available Spiders information	SetTriggerParameter	Set parameters for trigger
Connect	Connect to device	GetTriggerParameter	Get trigger parameters
Disconnect	Disconnect from device	TriggerArm	Trigger Arm
GetLastError	Get last error info	TriggerNext	Trigger Next
		TriggerAbort	Trigger Abort
Command Methods		Trigger Accept	
SendCommand	Send commands		
		Signal Methods	
Test Methods		GetSignalStatus	Get status parameters of all signals
CreateTest	Create new test	GetSignalData	Get signal data
CreateTriggerTest	Create trigger test		
CreateFRFTest	Create FRF test	Record	
GetTestStatus	Get current test status	StartRecord	Start recording
GetChannelTable	Get channel table parameters	StopRecord	Stop recording
SetChannelTable	Set parameters for channel table	SaveSignal	Save a frame of data
SetDSAParameter	Set parameters for DSA	GetFileList	Get the list of files
GetDSAParameter	Get DSA Parameters	DownloadFile	Download data file
SetOutputParameter	Set parameters for output	RemoveLastFile	Delete the last data file
GetOutputParameter	Get output parameters	RemoveAll	Delete all data files on hardware
GetChannelStatus	Get channel status		
GetTestStatus	Get test status	Simulation Mode	
GetSpiderTime	Get hardware system time	SetSimulationMode	Enter simulation mode without Spider
SetSpiderTime	Set hardware system time	GetSimulationMode	Get simulation mode status
GetSpiderConfig	Get hardware parameter (IP address)		
SetSpiderConfig	Set hardware parameter (IP address)		
CheckLicenseKey	Check the status of license key file		
LoadLKFile	Load license key file		

Callback and Events

```

event DeviceDSPMessageHandler DeviceDSPMessageReceived;
    // Called after DSP instruction message received
event DeviceNotifyMessageHandler DeviceNotifyMessageReceived;
    // Called after device status received
event TestRunStatHandle TestRunStatChanged;
    // Called after status changed
event DeviceDataIsReadyHandler DeviceDataIsReady;
    // Called after data collected
event DeviceConnectedHandler DeviceConnected;
    // Called after connected
event DeviceDisconnectedHandler DeviceDisconnected;
    // Called after disconnected
event DeviceReadyHandler DeviceReady;
    // Called after test created
event DeviceStoppedHandler DeviceStopped;
    // Called after device stopped
event TriggerArmedHandler TestTriggerArmedChanged;
    // Called after trigger armed
event TriggerDisarmedHandler TestTriggerDisarmedChanged;
    // Called after trigger disarmed
event TriggerFiredHandler TestTriggerFiredChanged;
    // Called after trigger fired
  
```