PSKCore DLL Wrapper for .NET Applications

December, 2010
Dave Cook, WAØTTN
http://www.netdave.com/wa0ttn

Introduction

PSKCoreWrap is a Microsoft .NET assembly that provides an interface between the Moe Wheatley (AE4JY) PSKCore DLL, allowing for easy development of digital amateur radio communications applications. This is a very brief overview of this utility. For more details, please read the API documentation and the review the example code included with this kit.

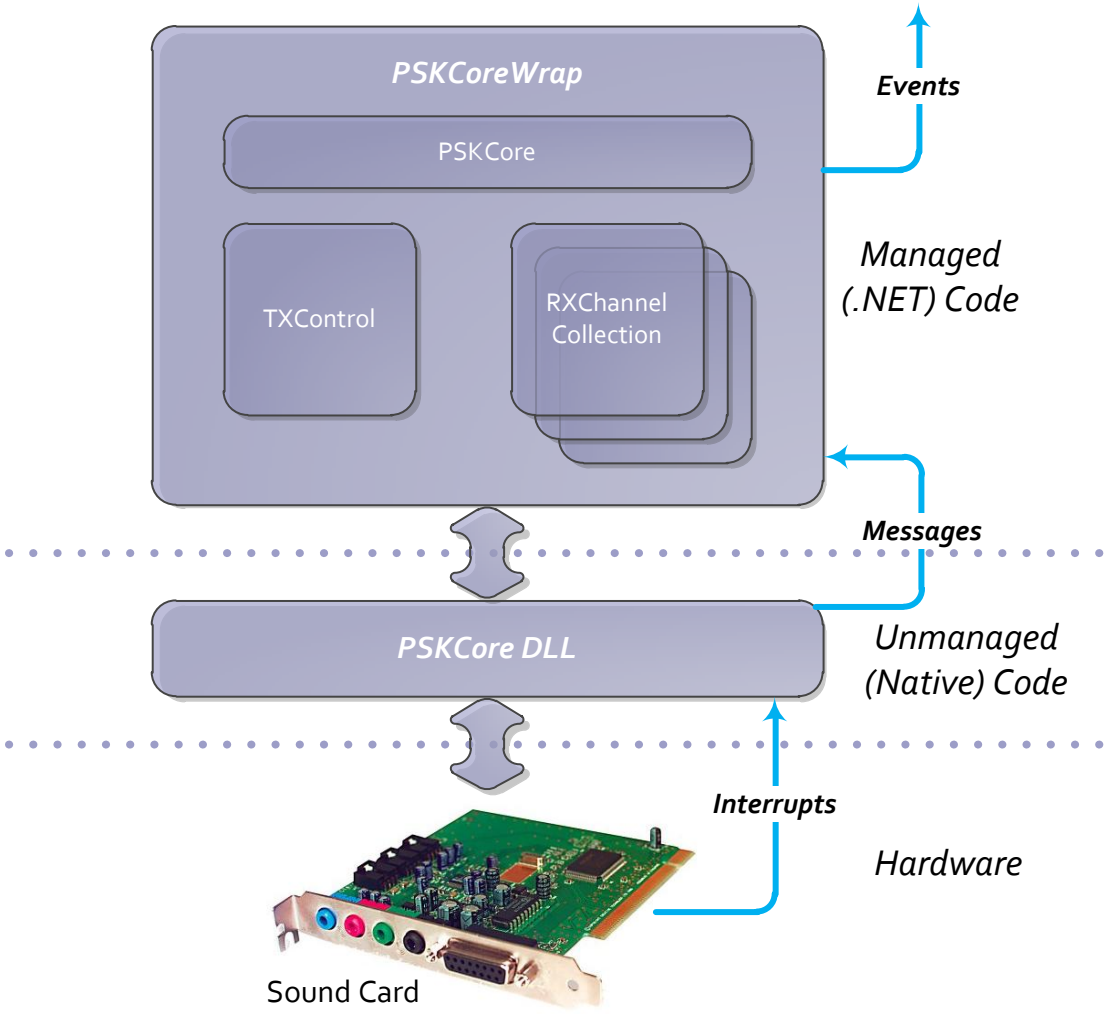
Manifest

This distribution consists of the following files:

File	Description
PSKCore.dll	The unmanaged code PSKCore DLL
PSKCoreDLL141.pdf	Moe Wheatley's original documentation for the PSKCore DLL
PSKCoreWrap.dll	The PSKCore .NET Interop wrapper assembly
PSKCoreWrap.chm	API reference for the PSKCoreWrap assembly
PSKCoreWrap.pdf	This document
PSKCoreDemo.exe	Executable for the PSKCoreWrap demonstration program
PSKCoreDemo folder	Source code for the PSKCoreWrap demonstration program

PSKCoreWrap Design

The PSKCore wrapper uses the .NET Interop services to provide access to the "native mode" interfaces of the PSKCore DLL. In addition, the wrapper encapsulates the functionality of the PSKCore DLL inside of an object-oriented component that provides a more logical structure than the flat APIs of the original DLL. The following diagram illustrates the architecture of the PSKCoreWrap system:



The PSKCore DLL provdes a set of "flat" APIs (application programming interfaces), meaning that there is no structure or organization to them. The PSKCoreWrap layer encapsulates the functionality of the flat APIs into a structured "object model" design. This model consists of the following primary objects:

Object	Description
PSKCore	This is the primary object in the PSKCore wrapper. It could also be considered the

Object	Description
	"Soundcard" object; however it also contains other functionality for managing the PSKCore operations.
RXChannel	Represents the parameters and data for a single receive data channel. The PSKCore object contains 50 RXChannel objects. The receive channels are accessed as an implicit indexed property of the PSKCore object.
TXControl	Represents the parameters and data for the transmit channel. There is only one transmitter, which is exposed as the **TXChannel** property of the PSKCore object.

An auxiliary object is defined for conveying vector data to the application:

Object	Description
PSKVector	A container object that is used to return signal vector data from a receive channel.

Several enumerations encapsulate the values for various parameters used by the underlying PSKCore DLL.

Enumeration	Description
PSKMode	Operating mode definitions. BPSK and QPSK are supported.
PSKSpeed	PSK speed definitions. PSK31, PSK63, and PSK125 are supported.
PTTMode	Serial port PTT mode definitions. RTS, DTR, and RST+DTR are supported.

Informational notifications are delivered to the application through Events. The following table lists the events that are delivered. A corresponding event argument object is delivered by the event notification that contains data details about the event.

Event	Description
DataReady	A new spectrum (FFT) or raw audio input data buffer is ready.
CharReady	A character is available from the receiver or, if in transmit mode, when a character has been sent.
IMDReady	IMD data is ready.
StatusChange	Status has changed.
ClockError	Clock error information.

How to Use the PSKCoreWrap Assembly

There are just a few essential elements to creating a PSKCore application. There are many more features that the PSKCore library provides that can be explored once you have a basic application working.

The following examples are explained using C#. If you are not familiar with C# then I recommend that you give it a try. Furthermore, these instructions assume you are using Microsoft Visual Studio to create your application. A free version, Visual Studio Express, is available from the Microsoft Web site.

Referencing the PSKCoreWrap Assembly

To add a reference to the PSKCoreWrap assembly to your project:

1. Right click on the References folder in the Solution Explorer window and choose "Add Reference…"
2. In the Add Reference dialog box, choose the Browse tab, browse to the directory where the PSKCoreWrap SDK is installed, and select the PSKCoreWrap.dll file.

On completion, there should be a new item named PSKCoreWrap under the References folder in the Solution Explorer window.

Note: *When you add the PSKCoreWrap reference to your project, a PSKCoreWrap Components category is added to the Visual Studio Toolbox. The PSKCore component contained in this category can be added to your application in Designer mode, however it references the PSKCore DLL which may not be available in the design time environment. The safe method for adding the PSKCore component to your application is to implement it programmatically as shown in the following implementation steps.*

PSKCore Object Instantiation

There are two steps for including a PSKCore object in your application:

1. Add a PSKCoreWrap namespace reference to the existing namespace declarations. This is not necessary if you prefer to use fully qualified object references in your code.
2. Add a global declaration for a PSKCore object to your application and instantiate the PSKCore object in your application startup code.

The following example shows how to instantiate a PSKCore object as an application global object:

```
private PSKCore _pskCore = new PSKCore();
```

Event Handling

You need to intercept the events that are fired by the PSKCore DLL. To do this, you connect an event handler to the PSKCore event as shown in the following example:

```
private void frmMain_Load(object sender, EventArgs e)
{
   _pskCore.DataReady += _pskCore_DataReady;
}
```

The following code demonstrates a handler that catches the DataReady event, which indicates that a new buffer of audio or spectrum data is available for processing.

```
private void _pskCore_DataReady(object sender, DataReadyArgs e)
{
    // handle data processing
}
```

Starting the Soundcard

You may wish to start the soundcard immediately during the application initialization, or allow the user to configure the soundcard options before starting it. The following code example shows how to start the default soundcard device with a single PSK receive channel.

```
_pskCore.StartSoundCard(-1, 1);
```

Accessing Receive Channels

There are 50 receive channels available in the PSKCore DLL. In the PSKCoreWrap object model, the receive channels are an implicit indexed property of the PSKCore object. The following example shows how to initialize some of the properties of the first receive channel (RXChannel[0]).

```
_pskCore[0].SquelchThreshold = 25;
_pskCore[0].Frequency = 1500;
_pskCore[0].SearchRange = 45;
_pskCore[0].AFCLimit = 100;
```

Running your Project

Before running your project, make sure you have placed the PSKCore.dll file in the directory where the application executable resides. You may also wish to place the PSKCore.dll file in the Windows\System32 directory for system-wide access; however Administrator privilege is required to place it there.

Demonstration Software

A simple project is included with this distribution that demonstrates the fundamentals of writing a .NET application that utilizes the PSKCoreWrap assembly, with the underlying PSKCore DLL, to display audio and spectrum data from the soundcard. Some of the features of this program are:

- The soundcard is started and stopped by means of an application menu control.
- A very simple data plotter is demonstrated that shows how to do off-screen drawing to avoid flicker.