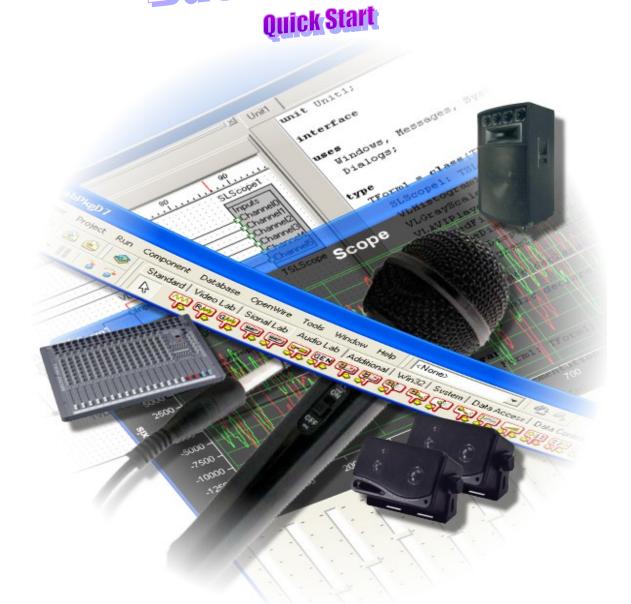
BASICITIED 75



www.openwire.org www.mitov.com

Copyright Boian Mitov 2004 - 2014

Index

Installation	.3
Where is BasicAudio?	.3
Why some of the examples don't work?	.3
Creating a simple audio player using Win32API Components	
Creating a simple audio player using DirectShow components	
Creating a simple Audio Recorder Win32API Components	.8
Using the TSLCRealBuffer in C++ Builder and Visual C++	10
Distributing your application.	13
Deploying your 32 bit application with the IPP DLLs	
Deploying your 64 bit application	

Installation

AudioLab comes with an installation program. Just start the installation by doubleclicking on the Setup.exe file and follow the installation instructions.

Where is BasicAudio?

After the installation, start your Delphi or C++ Builder. Scroll the "Component Palette", until you see the last three tabs:

🕻 Delphi 7 - Project1	
File Edit Search View Project Run Component Database OpenWire Tools Window Help 😽 (search component) 🚺 👫 🕞 🐏	
🔭 🕼 🕶 🗐 🎒 🚰 😰 😰 🥔 Standard Sianal Lab Audio Lab Plot Lab Additional Win32 Sustem Data Access Data Controls dbExpress DataSnap BDE ADD	

If the installation was successful, they should be named "Signal Lab", "AudioLab" and "Plot Lab". On the SignalLab palette you will have only a subset of the SignalLab components. SignalLab is a separated product, and will not be shipped as full with AudioLab.

Only the following two components of SignalLab will be available:

I	Component Database OpenWire Tools Window Help	
Τ	Standard Signal Lab Audio Lab Plot Lab Control Lab Additional Win32 System Data Access Data Controls dbExpress DataSnap	į.

The following two PlotLab components will be available.

1	Component Database OpenWire Tool	Window Help
Τ	Standard Signal Lab Audio Lab Video Lab	Plot Lab Additional Win32 System Data Access Data Controls dbExpress DataSnap

Why some of the examples don't work?

Video lab is a unique library that supports both the Win32 API's AVIFile (VFW) functions (ACM) and DirectShow. You as a developer have the ultimate choice to use either the Win32 API or DirectShow components or both at the same time.

The advantage of the Win32 API components is that hey will work on any Windows 95 and up system out of the box, however they are much less capable than the DirectShow components, and should be avoided if not necessary.

The advantage of the DirectShow components is that they will use the latest and greatest capability of DirectShow, the latest video camera devices, and TV Tuners, but they require DirectShow 9.0 or higher to be installed in order to work.

If you don't have DirectX 9.0 or higher installed on your system, you will not be able to use see the DirectShow examples working.

Creating a simple audio player using Win32API Components

From the Delphi/C++Builder menu select | File | New | Application |.

2	🌶 Delphi 7							
]	File Edit Search View Project Run Component Database OpenWire Tools Window Help 🛛 🕄 🖳 🛃 🗐							
T		New 🕨	(T	Application	udio Lab Sianal Lab Additional Win32 System Data Access Data Controls dbExpress 💶 🕨			
Ť	3	Open	<u>و</u>	CLX Application	IT 🖁 A 💵 📄 🗷 🗵 💿 📕 📑 🚥 🗋 📄 🔜			
1	2	Open Project Ctrl+F11		Data Module				
		Reopen >		Form				
		Save Ctrl+S		Frame				
) E (-	Save As	P	Unit				
		Save Project As	*	Other				
		Save All Shift+Ctrl+S	_					
	₿ ₩₽	Close						
	900 1900	Close All						
	Ъ	Use Unit Alt+F11						
	8	Print						
	Ē.	Exit						

An empty form will appear on the screen.

From the "Component Palette" select the "Audio Lab" tab:

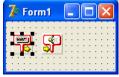
Iomponent Database OpenWire Tools Window Help 🔣 KNone> 💽 🛃 🐴
Standard Sianal Lab Audio Lab Video Lab Vision Lab Plot Lab Additional Win32 System Data Access Data Controls dbE 💶 🕨
· • • 🛒 42 32 32 32 32 32 39 39 39 37 32 39 39 39 39 39 39 39 39 39 39 39 39 39

select and drop on the form the following two components:

One TALWavePlayer

One **I** - TALAuduioOut

Make the form small and select the ALDSAudioOut1 component:



In the Object Inspector select the FileName property and click the 🛄 button.

Object Inspector 🛛 🔀					
ALWavePlayer1 TALWavePlayer 💌					
Properties Eve	ents				
Enabled	True 🔼				
EnablePin	(Disconnected)				
FileName					
FileNamePin	(Disconnected)				
Lood	False				

November 13, 2014

BasicAudio 7.5

A File selection dialog will appear:

Open						? 🗙
Look <u>i</u> n:	🗀 AVIFiles		•	🗢 🔁 (
My Recent Documents Desktop	Demo.wav					
My Documents						
My Computer						
My Network Places	File <u>n</u> ame:	Demo.wav			•	<u>O</u> pen
	Files of type:	Audio files (*.wav;*.wave;*wm	ia;*mp\$	3)	•	Cancel

Select a file to play and click "Open".

In the Object inspector select the OutputPin property and click the 🛄 button.

5	1				
Object Inspector 🛛 🛛 🔀					
ALWavePlayer1 TALWavePlayer 💌					
Properties Eve	ents				
Name	ALWavePlayer1 🔨				
OutputPin	(Disconnected) ····				
Paused	False				
PausePin	(Disconnected)				

You should see the Pin Editor:

🅻 Connections - Source Pin : ALWavePlayer1.OutputPin								
Form	Form : Form1 (Current)							
	Sink pin Component Connected to Connection Typ							
	InputPin	ALAudioOut1			OpenWire			
					🔁 <u>R</u> estore			

Click on the check box to make it look as in the picture, and then click OK.

Compile and run the application.

You should see the form and hear the audio playing:



Congratulations! You have just created your first AudioLab application. Here are the OpenWire connections in this application:

ALWavePlaye	ALAudioOut1	
FileNamePin ClockPin EnablePin PausePin	OutputPing ProgressPing	

TALWavePlayer

Creating a simple audio player using DirectShow components

WARNING: In order to run the application in this example you must have DirectX 9.0 or higher installed!

7	🛛 De	lphi 7			
]	File	Edit Search View Project	RL	in Component	Database OpenWire Tools Window Help
		New 🕨	(T	Application	udio Lab Sianal Lab Additional Win32 System Data Access Data Controls dbExpress 💶 🕨
Ť		Open	<pre>pi</pre>	CLX Application	I 🖁 A 💵 📄 🗷 🗵 💿 🛃 🚍 🚥 🗋 🗎 🗌 🏭
1	2	Open Project Ctrl+F11		Data Module	
		Reopen 🕨	—	Form	
		Save Ctrl+S		Frame	
) () (;	Save As		Unit	
		Save Project As	°	Other	
		Save All Shift+Ctrl+S	-		-
	日中華	Close			
		Close All			
	ъ	Use Unit Alt+F11			
	8	Print			
	i.	Exit			

From the Delphi/C++Builder menu select | File | New | Application |.

An empty form will appear on the screen.

From the "Component Palette" select the "Audio Lab" tab:

<none></none>
p Plot Lab Additional Win32 System Data Access Data Controls dbE 💶 🕨

select and drop on the form the following two components:

One 🔛 - TALDSAudioPlayer

One **I** - TALDSAudioOut

Make the form small and select the ALDSAudioOut1 component:

🌈 Form1 👘]	2	×	
	÷	1	1	1	1	1	÷	÷	÷	l
	·	·	·	·	·	·	·	·	·	÷
	·	·	·	·	·	·	·	·	·	•
	·	·	·	·	·	·	·	·	·	
	•	·	·	·	·	·	·	·	•	

In the Object Inspector select the FileName property and click the 🛄 button.

Object Inspect	or 🔀
ALDSAudioPlaye	r1 TALDSAudioPlaye
Properties Eve	ents
Enabled	True 🔥
EnablePin	(Disconnected)
FileName	
FileNamePin	(Disconnected)

A File selection dialog will appear:

Open						? 🔀
Look <u>i</u> n:	🗀 AVIFiles		•	(- 🔁	* 💷 *	
My Recent Documents Desktop	Demo.wav					
My Documents						
My Computer						
San	5 1	Derror		1	-	0
My Network Places	File <u>n</u> ame: Files of <u>type</u> :	Demo.wav Audio files (*.wav;*.wave;*wm	na;*mp(3)	- -	<u>O</u> pen Cancel

Select a file to play and click "Open".

In the Object inspector select the OutputPin property and click the 🛄 button.

5	L
Object Inspect	or 🔀
ALDSAudioPlaye	r1 TALDSAudioPlaye
Properties Eve	ents
⊞ Graph	(TDSGraph) 🛛 🔥
Loop	False
Name	ALDSAudioPlayer1
OutputPin	(Disconnected) 💻 📥
Paused	False
PausePin	(Disconnected)

You should see the Pin Editor:

7 0 c	onnection	s - Source Pin :	ALDSA u dioP la	yer1.OutputPi	🛛
Form	: Form1 (Cu	rrent)		•	6)
₽	Sink pin	Component	Connected to	Connection Typ	OpenWire
	InputPin	ALDSAudioOut1			opennie
					🔁 <u>R</u> estore

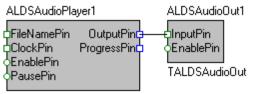
Click on the check box to make it look as in the picture, and then click OK.

Compile and run the application.

You should see the form and hear the audio playing:



Congratulations! You have just created your first DirectShow BasicAudio application. Here are the OpenWire connections in this application:



TALDSAudioPlayer

Creating a simple Audio Recorder Win32API Components

From the Delphi/C++Builder menu select | File | New | Application |.

7	De	phi 7			
	File	Edit Search View Project	Ru	n Component	Database OpenWire Tools Window Help
		New 🕨	P	Application	udio Lab Sianal Lab Additional Win32 System Data Access Data Controls dbExpress 💶 🕨
Ť		Open	C	CLX Application	IT 🔧 A 💵 📄 🗷 🗵 💿 🛃 🗃 🚥 🗋 🖹 🗌 🏭
4	2	Open Project Ctrl+F11		Data Module	
		Reopen 🕨		Form	
		Save Ctrl+S		Frame	
) iii (-	Save As		Unit	
		Save Project As	° b	Other	-
		Save All Shift+Ctrl+S	-		
		Close			
	in Lynge	Close All			
	Ъ	Use Unit Alt+F11			
	۲	Print			
	i.	Exit			

An empty form will appear on the screen.

From the "Component Palette" select the "Audio Lab" tab:

	JX
Iomponent Database OpenWire Tools Window Help	
Standard Signal Lab Audio Lab Video Lab Vision Lab Plot Lab Additional Win32 System Data Access Data Controls dbf	
* 🛒 XI	•

select and drop on the form the following three components:

One 🔀 - TALAudioIn

One TALWaveLogger

Select the ALAudioIn component:

🅻 Form1			[]	2	×	
			÷	1	ĵ.	1	÷.	1
·· [•••] · [•••=]	• •	• •	÷	÷	÷	÷	÷	1
ה הפרי	1 · ·	• •	1	1	1	1	1	
· · · · · · · · · · · · · · · · · · ·		• •		1	1	1	1	
				1	1	1	1	

In the Object Inspector select the OutputPin property and click the 🛄 button.

Object Inspect	or 🗵
ALAudioIn1	TALAudioln 💽
Properties Ev	ents
⊞AudioFormat	(TALAudioFormat)
⊞Device	(TALAudioCaptureDev
Enabled	True
EnablePin	(Disconnected)
Name	ALAudioIn1
OutputPin	(Disconnected)
PumpPriority	0
Tan	0

You should see the Pin Editor:

700	Connection	ns - Source Pin :	ALAudioIn1.0	DutputPin		
Form	: Form1 (Cu	urrent)			-	()
₽	Sink pin	Component	Connected to	After Pin	C	OpenWire
	InputPin	ALWaveLogger1				opennie
						🔁 <u>R</u> estore
<u> </u>						A Link to all

Click on the check boxes as shown in the picture, and then click OK.

On the form select the ALWaveLogger1 component:

🅻 Form1		×
	 11	
- <u>es</u> s - <u>um</u>	 	
	 1.1	111

In the Object Inspector select the FileName property and set it to RecordedAudio.wav:

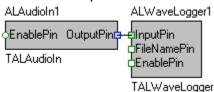
Object Inspector 🛛 🔀		
ALWaveLogger1 TALWaveLogger 💌		
Properties Eve	ents	
Compression	(TALAudioCompressio	
Enabled	True	
EnablePin	(Disconnected)	
FileName	RecordedAudio.wav 😐	
FileNamePin	(Disconnected)	

Compile and run the application. You should see the form:



A file named RecordedAudio.wav will be created containing the recorded audio.

Here are the OpenWire connections in this application:



You have just learned how to create audio recorder with BasicAudio.

Using the TSLCRealBuffer in C++ Builder and Visual C++

The C++ Builder version of the library comes with a powerful data buffer class, called TSLCRealBuffer.

The TSLCRealBuffer is capable of performing basic math operations over the data as well as some basic signal processing functions. The data buffer also uses copy on write algorithm improving dramatically the application performance.

The TSLCRealBuffer is an essential part of the SignalLab generators and filters, but it can be used independently in your code.

You have seen already some examples of using TSLCRealBuffer in the previous chapters. Here we will go into a little bit more details about how TSLCRealBuffer can be used.

In order to use TSLCRealBuffer you must include SLCRealBuffer.h directly or indirectly (trough another include file):

#include <SLCRealBuffer.h>

Once the file is included you can declare a buffer:

Here is how you can declare a 1024 samples buffer: TSLCRealBuffer Buffer(1024);

Version 4.0 and up does not require the usage of data access objects. The data objects are now obsolete and have been removed from the library.

You can obtain the current size of a buffer by calling the GetSize method: Int ASize = Buffer.GetSize(); // Obtains the size of the buffers

You can resize (change the size of) a buffer: Buffer.Resize(2048); // Changes the size to 2048

You can set all of the elements (samples) of the buffer to a value: Buffer.Set(30); // Sets all of the elements to 30.

You can access individual elements (samples) in the buffer: Buffer [5] = 3.7; // Sets the fifth elment to 3.7 Double AValue = Buffer [5]; // Assigns the fifth element to a variable

You can obtain read, write or modify pointer to the buffer data:

```
const double *data = Buffer.Read() // Starts reading only
double *data = Buffer.Write()// Starts writing only
double *data = Buffer.Modify()// Starts reading and writing
```

Sometimes you need a very fast way of accessing the buffer items. In this case, you can obtain a direct pointer to the internal data buffer. The buffer is based on copy on write technology for high performance. The mechanism is encapsulated inside the buffer, so when working with individual items you don't have to worry about it. If you want to access the internal buffer for speed however, you will have to specify up front if you are planning to modify the data or just to read it. The TSLCRealBuffer has 3 methods for accessing the data Read(), Write(), and Modify (). Read() will return a constant pointer to the data. You should use this method when you don't intend to modify the data and just need to read it. If you want to create new data from scratch and don't intend to preserve the existing buffer data, use Write(). If you need to modify the data you should use Modify (). Modify () returns a non constant pointer to the data, but often works slower than Read() or Write(). Here are some examples:

```
const double *pcData = Buffer.Read(); // read only data pointer
double Value = *pcData; // OK!
*pcData = 3.5; // Wrong!
double *pData = Buffer.Write(); // generic data pointer
double Value = *pData; // OK!
*pData = 3.5; // OK!
```

You can assign one buffer to another:

Buffer1 = Buffer2;

You can do basic buffer arithmetic:

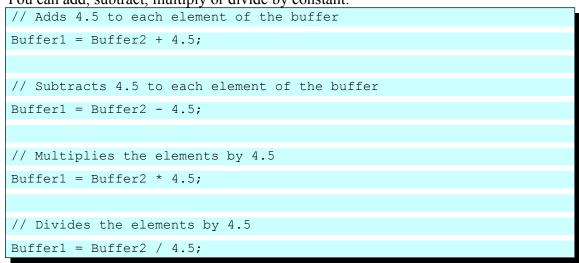
TSLCRealBuffer Buffer1(1024);		
TSLCRealBuffer Buffer2(1024);		
TSLCRealBuffer Buffer3(1024);		
Buffer1.Set(20.5);		
Buffer2.Set(5);		
Buffer3 = Buffer1 + Buffer2;		

November 13, 2014

BasicAudio 7.5

```
Buffer3 = Buffer1 - Buffer2;
Buffer3 = Buffer1 * Buffer2;
Buffer3 = Buffer1 / Buffer2;
```

In this example the elements of the Buffer3 will be result of the operation (+,-,* or /) between the corresponding elements of Buffer1 and Buffer2. You can add, subtract, multiply or divide by constant:



You can do "in place" operations as well:

Buffer1	+=	Buffer2;
Buffer1	+=	4.5;
Buffer1	-=	Buffer2;
Buffer1	-=	4.5;
Buffer1	*=	Buffer2;
Buffer1	*=	4.5;
Buffer1	/=	Buffer2;
Buffer1	/=	4.5;

Those are just some of the basic buffer operations provided by SignalLab.

If you are planning to use some of the more advanced features of TSLCRealBuffer please refer to the online help.

SignalLab also provides TSLCComplexBuffer and TSLCIntegerBuffer. They work similar to the TSLCRealBuffer but are intended to be used with Complex and Integer data. For more information on TSLCComplexBuffer and TSLCIntegerBuffer please refer to the online help.

Distributing your application

Once you have finished the development of your application you most likely will need to distribute it to other systems. In order for some AudioLab built application to work, you will have to include a set of DLL files together with the distribution. The necessary files can be found under the [install path]\DLL directory([install path] is the location where the AudioLab was installed).

On 32 bit windows systems, you can distribute them to the [Windows]\System32 directory, or to the distribution directory of your application([Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS).

On 64 bit windows systems, you can distribute them to the [Windows]\SysWOW64 directory, or to the distribution directory of your application([Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS).

Not all of the components in the library require additional DLLs. Please check if the DLLs are needed by the application before including them in the install.

Deploying your 32 bit application with the IPP DLLs

The compiled applications can be deployed to the target system by simply copying the executable. The application will work, however the performance can be improved by also copying the Intel IPP DLLs provided with the library.

The DLLs are under the [install path]\LabPacks\IppDLL\Win32 directory([install path] is the location where the library was installed).

In 32 bit Windows to deploy IPP, copy the files to the [Windows]\System32 directory on the target system.

In 64 bit Windows to deploy IPP, copy the files to the [Windows]\SysWOW64 directory on the target system.

[Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS This will improve the performance of your application on the target system.

Deploying your 64 bit application

The current version of the library requires when deploying 64 bit applications, the Intel IPP DLLs to be deployed as well.

The DLLs are under the [install path]\LabPacks\IppDLL\Win64 directory([install path] is the location where the library was installed).

To deploy IPP, copy the files to the [Windows]\System32 directory on the target system. [Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS This will improve the performance of your application on the target system.