Create a Robot Skill

Synthiam ARC is the leading distribution channel for a sensor, cloud service, or peripheral used by robot builder customers. Our customers include R&D PoC, DIY, enterprise businesses, manufacturers, and educational institutions. Synthiam's platform focuses on democratizing complicated technologies so that robot builders can quickly prototype and use them.

Get Robot Builders Using Your Product

Become a Technology Creator so that Robot Builders can use your product. Synthiam has a large and growing customer base of robot builders that want to use your technology as a robot skill in their robots using ARC.

How do you make a robot skill?

Follow this guide to publish sensors, cloud services, or peripheral products in the skill store as modular packages for robot builders to purchase and use in ARC.

What is a Robot Skill?

If you are new to ARC, it is recommended to read the Robot Skills overview. The overview will explain how Robot Skills are added to a robot project, and how the communication model works.

Learn what a Robot Skill is

ARC apps consist of Robot Skill Controls. Each skill is a behavior for the robot, similar to a process (or node). There are skills for Cameras, Speech Recognition, Machine Learning, and hundreds more. Skills can be added to a project workspace using the Add Skill option located in the Project tab of the main menu.

By combining multiple skills, robots can perform advanced and complex tasks. Robot skills come in two flavors...

1. **Built-In Robot Skills -** These are skills developed by Synthiam that are included with ARC installation and cannot be removed.
2. **Plugin Robot Skills -** Developed by third parties and may be added/removed from the ARC software through the Synthiam Skill Store.

Here is a screenshot of an example ARC project demonstrating multiple robot skill controls. Each skill control is performing a specific function of the robot. Pro user's projects may contain an unlimited number of skills (as PC memory allows). Read more about robot skills here.

ARC Stack Overview

Alright, Let's Do It!

This document section contains the resources you need to start building a robot skill for distribution. The side navigation menu lists the steps and resources to build an example skill control.

1. Download ARC

ARC is the software which you will be creating the robot skill plugin for. Your skill plugin is a graphical library that runs within the ARC open framework. The open framework allows your robot skill plugin the ability to use ARC's pre-built functions, rather than creating an entire program starting completely from scratch. This allows robot builder users to install your robot skill from the skill store to add new features to their robot.

Download and install the latest version of ARC (*Pro is recommended*) from here.

2. Install Visual Studio

Visual Studio is a software development IDE from Microsoft. The community edition can be downloaded for free and gives you the power to create .Net applications. In this step, we'll use ARC to detect an existing installation of Visual Studio and download the latest.

Visual Studio 2022 Warning

As of 2022/02/01, there is a bug with Microsoft Visual Studio 2022 edition. We recommend installing the Visual Studio 2019 Community edition as a replacement. Microsoft is aware of the bug and has not decided to fix it yet. This message will be removed when/if Microsoft fixes the bug. You can read more about the bug HERE. If this is an inconvenience, feel free to contribute your feedback to the bug report on the bug report link.

1) Load ARC

2) Press **Project -> Create Skill**

ï»¿
3) The Create Robot Skill window will check for an existing installation of Visual Studio. If there is not one detected, a *DOWNLOAD* button will be presented.

ï»¿
4) Press the Download link if present, or get the *Community Edition* of Visual Studio by clicking here.

Dependency
Ensure you have .Net 4.7.2 framework installed. If you keep up-to-date with Windows Updates and ARC runs without problems, then you must have .Net 4.7.2. Many programs are built with .Net 4.7.2 and therefore, it's uncommon for a computer not to have it. So, if ARC works, then you don't need to download and install .Net 4.7.2

3. Create Project

***Note:** If you are using C++ instead of C#, you will have to create the project manually for a managed C++ DLL library.

ARC will automatically generate an example C# .Net project with details about your plugin. To do so, first load the Create New Skill Control dialog box.

1) Load ARC

2) Press *Project -> Controls -> Create Skill*

3) Enter the name of your new control, company name, and a short description of the control.

4) Select a location for your project. Otherwise, the default location is usually best to keep them easily accessible.

5) Press *Create & Open Project*

6) A folder will open with the location of your project solution. Double click on the SLN file to open the project in Visual Studio.

4. Example Project Overview

The example project will add the references and configuration necessary to begin developing.

Running The Example Project
Let us first take a moment to load the example project in ARC and see it in action.

1) Shutdown any instances of ARC, as they aren't necessary during debugging.

2) Load the Visual Studio solution file for your plugin.

3) Press **F5** in Visual Studio, or select *Debug -> Start Debugging* from the top menu

4) ARC will launch

5) We'll now add your plugin to the workspace. Press *Project -> Skills -> Add Skill*

6) Navigate to the *Beta* tab, and your plugin will be visible. Click the icon and the project will load.

7) View your new skill control on the workspace. The example project even has a configuration menu configured. You can press the configuration option (3 little dots next to the X) to view the example configuration menu.

8) Close ARC to stop debugging the robot skill. You must stop ARC when you want to recompile and test the robot skill after new code changes.

Notes
Visual Studio is running as the debugger for the project. For experienced programmers, you can step through and debug the skill control plugin in real-time.

5. Add Some Buttons

Visual Studio 2022 Warning

As of 2022/02/01, there is a bug with Microsoft Visual Studio 2022 edition. We recommend installing the Visual Studio 2019 Community edition as a replacement. Microsoft is aware of the bug and has not decided to fix it yet. This message will be removed when/if Microsoft fixes the bug. You can read more about the bug HERE. If this is an inconvenience, feel free to contribute your feedback to the bug report on the bug report link.

Now that the **MainForm** has been created with the example project, it will be the default form that users see when using your plugin. The form is currently only containing a label with some text, making it a boring plugin. In this step we will add two buttons to move servos between different positions.

1) Locate the **MainForm** in the Solution Explorer and double click.

2) When the MainForm designer loads, click on the text and press Del to remove the label from the form. This will leave you with a blank form.

3) Locate the **Button** under *All Windows Forms* in the Toolbox. Drag two buttons anywhere onto your **MainForm**.

3) Give the buttons readable text that tells the user what they will do. In this tutorial example, we will be programming the buttons to move a servo between two positions. Click on each button and locate the **Text** field in the properties window.

4) Double click on each button in the Designer and code will automatically be generated for the Click event of each button. This means that when a user clicks on the button, the code within the function will be executed. The functions are automatically inserted into your code when you double click on them from the designer.

5) Insert code into each of the button click events to move a servo. The command to move a servo is located within an EZB class. Because ARC allows more than one EZB connection, the list of EZB's available is an array. It is safe to assume that the first EZB is used the most. Here is the code which will move the servo connected on the EZ-B port D0 between position 10 degrees and 170 degrees when the buttons are pressed.

Code:

```
private void button1_Click(object sender, EventArgs e) {
  ARC.EZBManager.EZBs[0].Servo.SetServoPosition(EZ_B.Servo.ServoPortEnum.D0, 10);
}

private void button2_Click(object sender, EventArgs e) {
  ARC.EZBManager.EZBs[0].Servo.SetServoPosition(EZ_B.Servo.ServoPortEnum.D0, 170);
}
```

6) When your code has been entered, it will now look like this.

7) Let's compile your project to ensure there are no errors before continuing to the next step. Press **CTRL-SHIFT-B** and watch the Output window for any error messages. If everything compiles okay, you will see a similar message result to the screenshot below.

8) Assuming you have no errors and everything compiles fine, press F5, the project will compile and load ARC. Add the plugin to your workspace and test it out!

6. Publish Plugin To Skill Store

When you are ready to share the robot skill plugin with the world, it will be published to the technology store on Synthiam's website.

Create Package
1) Build a fresh copy of your skill plugin in Visual Studio.

2) Navigate to the plugin folder. This will be located in *C:\ProgramData\ARC\Pluginsî»¿\[guid]*. Where is the guid of your plugin in the Plugin.XML file.

3) Select all files *(CTRL-A)* and *Right-Click* with the mouse. Select *Send To -> Compressed (zipped) Folder*

4) A .ZIP file will be created containing all of the necessary skill control plugins and sub-folders. This is the file that will be uploaded to Synthiam.com

Upload Package To Synthiam
The zip file created in the above step will be uploaded to the synthiam website.

1) Visit synthiam.com and login.
2) Press the Account button on the top right.
3) Press the My Content sub-menu.

4) Locate your plugin by the title and click to select it. The plugin statistic page will be displayed. When your skill control plugin is live, you may return here to view download and usage performance. Press the Details button.

5) By default, the details page will introduce your plugin as Private and display a number of requirements to make it public.

6) Scroll down and locate the Control Archive File option. This is where we will upload the skill control Zip file package that was created earlier.

7) Scroll up and press Save Changes. The package file will upload.

Notes
Before your plugin can be visible to the public, some requirements must be met. Review the list of requirements on the right of the details page. Once your requirements have been met, check the Public checkbox and your plugin will be published to the Synthiam Technology Store.

UI Components

Visual Studio 2022 Warning

As of 2022/02/01, there is a bug with Microsoft Visual Studio 2022 edition. We recommend installing the Visual Studio 2019 Community edition as a replacement. Microsoft is aware of the bug and has not decided to fix it yet. This message will be removed when/if Microsoft fixes the bug. You can read more about the bug HERE. If this is an inconvenience, feel free to contribute your feedback to the bug report on the bug report link.

The ARC Plugin Framework is very powerful because it is entirely open. That means you can access every component and resource within the ARC.exe application or EZ_B.DLL library. There are hundreds of custom .Net user UI components exposed in the ARC application to assist with your plugin. Including joysticks, buttons, camera canvases, and more.

Theming
All plugin components will be themed using the ARC standard theme renderer. This happens automatically, so the control colors, look and feel will be adjusted accordingly. There is an Example in another tutorial step regarding themes and the available commands to interact with the theme engine.

The Visual Studio IDE has a Tool Box, which is used for designer mode when customizing a form. The toolbox can have custom controls added to it. Below is a screenshot of a default toolbox.

Create New Toolbox Tab
To keep the new ARC component controls organized, we will create a new category in the toolbox. This is done by scrolling to the bottom of the toolbox list and RIGHT CLICK -> ADD TAB

The new tab will be named ARC.

Add Controls To ARC Tab
Now we will right click on the ARC tab and select CHOOSE ITEMS

A Choose Items Dialog box will be displayed. It may take a few moments for the dialog to load while it scans and organizes the existing controls within the toolbox. Press the BROWSE button.

Now locate and select the ARC.exe application in C:\Program Files (x86)\Synthiam Inc\ARC. All controls of ARC will now be selected for the toolbox. Press OK

When editing/creating a Windows Form, the ARC tab in the toolbox will now have many new controls that you can begin using. Have fun!

More Publishing Information

Additional Open-Source Code Examples
Some developers can find additional open-source code examples in the Synthiam OpenSource GitHub repository by clicking hereï»¿.

Adding Custom ARC Icons
Your control will use the default icon if there isn't another specified. To add an icon, include either a transparent PNG in your project output folder named **title.png**. This file will be used in the Add Control window of ARC as an icon for your plugin.

Changing ARC Category
You may wish to change the category of your plugin from BETA to an appropriate category when publishing to the public. Specify the robot skill's category in the Plugin.xml file. The category is must match one of the categories from the ARC Add Control menu.

Share Unpublished Plugin (User Testing)
It is a good idea to share your newly created plugin with others before publishing it to the public. You can copy the URL of the DOWNLOAD option on your Plugin Definition page and send it to others. The DOWNLOAD option will not display until you have uploaded a valid plugin file.

To share the plugin for testing, copy the URL of the DOWNLOAD option or send the <xxxxx.EZPLUGIN> file to your test group.

ARC Classes, Methods & Events

This outlines the ARC API, which has Events, Methods, and Fields. You can search for any of those with the browser using CTRL-F to find all events, for example. Many classes and respective methods may not be documented here because they are less commonly used. These are the most common API calls for robot skills.

Jump To...

- Common
- Constants
- EZBManager
- FormMain
- InterfaceBuilder
- Invokers
- MessagingService
- MovementManager
- Renderer
- Scripting
- Services
- TimerSmart
- URLServiceManager
- VAD
- WebServiceWrappers

InterfaceBuilder

Event: ARC.InterfaceBuilder.RemoteUIServer.OnConnection

Event risen when for handleCustomEvent is true and a new connection is established
Method: ARC.InterfaceBuilder.RemoteUIServer.Start(*System.Int32*)

Start the TCP Server and beginning listening on the specified port.
Method: ARC.InterfaceBuilder.RemoteUIServer.Stop

Stop the TCP Server listener
Method: ARC.InterfaceBuilder.RemoteUIServer.DisconnectClients

Disconnect all clients
Type: ARC.InterfaceBuilder.ResourceImages

A strongly-typed resource class, for looking up localized strings, etc.
Type: ARC.InterfaceBuilder.ResourceTemplates

A strongly-typed resource class, for looking up localized strings, etc.

MessagingService

Type: ARC.MessagingService.MessagingService

The messaging framework which allows controls to broadcast data to all other controls which are listening for relevant data.
Event: ARC.MessagingService.MessagingService.OnMessageReceived

Subscribe to receive messages sent from other controls that may contain relevant data for your control.
Method: ARC.MessagingService.MessagingService.BroadcastMessage(*System.String, System.Object*)

Broadcast data to all controls that are listening for relevant data
Event: ARC.MessagingService.Navigation2DV1.Messenger.OnNewLocationPointScanPoints

Combined navigation (positioning) and scanning data event.
This is ideal if you're requiring both data groups for your navigation/mapping solution.
The event is fired on every navigation update. All scanning data is collected between the last navigation update and this one.
Event: ARC.MessagingService.Navigation2DV1.Messenger.OnNewScan

When new scan points are received by a level #3 Group #1 sensor.
If you are creating a navigator, it is recommended to use teh OnNewLocationPointScanPoints becaue it contains both the combined location and scan points in one event

Event: ARC.MessagingService.Navigation2DV1.Messenger.OnNewLocation

When a new location is updated by a Lavel #3 Group #2 sensor.
If you are creating a navigator, it is recommended to use teh OnNewLocationPointScanPoints becaue it contains both the combined location and scan points in one event
Event: ARC.MessagingService.Navigation2DV1.Messenger.OnNavigationStatusChanged

When a Level #3 Group #1 sensor wishes to alert the navigator that it sould pause or continue navigating based on obstacles
Field: ARC.MessagingService.Navigation2DV1.Messenger._scanPoints

Stores the scan points if someone has subscribed to the OnNewLocationPointScanPoints event
Field: ARC.MessagingService.Navigation2DV1.Messenger.Location

The last reported position of the robot relative to the starting position
Field: ARC.MessagingService.Navigation2DV1.Messenger.DegreesHeading

The last reported heading degree the robot is facing relative to the starting degrees
Method: ARC.MessagingService.Navigation2DV1.Messenger.Clear

Clear the stored scanpoints
Method: ARC.MessagingService.Navigation2DV1.Messenger.ChangeNavigationStatus(*ARC.MessagingService.Navigation2DV1.Messenger.NavigationStatusEnum*)

Change the navigation status for the navigator (level #1) if we should be navigating or not
Method: ARC.MessagingService.Navigation2DV1.Messenger.UpdateLocation(*ARC.MessagingService.Navigation2DV1.LocationPoint*)

Update the location of the robot.
The navPoint is an offset from the starting position (0, 0) in cm
Method: ARC.MessagingService.Navigation2DV1.Messenger.UpdateLocation(*System.Single, System.Single, System.Byte, System.Single*)

Update the location of the robot.
The navPoint is an offset from the starting position (0, 0) in cm
Method: ARC.MessagingService.Navigation2DV1.Messenger.UpdateScan(*ARC.MessagingService.Navigation2DV1.ScanPoint[]*)

Update the scan of the current location with a list of scan points
Each scan point is a different degree of the scan with a distance in cm of the deteced object.
If no object detected, a distance of 0 cm is specified
Method: ARC.MessagingService.Navigation2DV1.Messenger.UpdateScan(*ARC.MessagingService.Navigation2DV1.ScanPoint*)

Update the scan of the current location with a list of scan points
*Performance note: If you are updating with many scan points of many degrees, do not use this override. Use the array override
Each scan point is a different degree of the scan with a distance in cm of the deteced object.
If no object detected, a distance of 0 cm is specified
Method: ARC.MessagingService.Navigation2DV1.Messenger.UpdateScan(*System.Single, System.Byte, System.Single*)

Update the scan of the current location with a list of scan points
*Performance note: If you are updating with many scan points of many degrees, do not use this override. Use the array override
Each scan point is a different degree of the scan with a distance in cm of the deteced object.
If no object detected, a distance of 0 cm is specified
Field: ARC.MessagingService.Navigation2DV1.LocationPointScanPoints.LocationPoint

The nav point
Field: ARC.MessagingService.Navigation2DV1.LocationPointScanPoints.ScanPoints

A list of scan points
Field: ARC.MessagingService.Navigation2DV1.ScanPoint.Distance

The distance in cm of any detected object from the center of the robot, otherwise 0
Field: ARC.MessagingService.Navigation2DV1.ScanPoint.Confidence

Confidence of the detected object (0 = not confident, 255 = very confident)
Field: ARC.MessagingService.Navigation2DV1.ScanPoint.Degree

The degree of where the object was detected
Field: ARC.MessagingService.Navigation2DV1.LocationPoint.X

The distance in CM from the originating X position since tracking began
Field: ARC.MessagingService.Navigation2DV1.LocationPoint.Y

The distance in CM from the originating Y position since tracking began
Field: ARC.MessagingService.Navigation2DV1.LocationPoint.DegreesHeading

When initialized, the robot is determined to be facing 0 degrees.
This is the degrees the robot is currently facing relative to the initialization reference angle
If a robot were to strafe, this angle doesn't change.
If a robot rotates left, the angle begins to increase
Field: ARC.MessagingService.Navigation2DV1.LocationPoint.Confidence

Confidence of the location (0 = not confident, 255 = very confident)
Method: ARC.MessagingService.Navigation2DV1.LocationPoint.ToPoint

Returns the X/Y cordinate as a point object

Method: ARC.MessagingService.Navigation2DV1.LocationPoint.ToPoint(*System.Single*)

Returns the X/Y cordinate as a point object and increases by the offset value
Method: ARC.MessagingService.Navigation2DV1.LocationPoint.ToPoint(*System.Single, System.Single*)

Returns the X/Y cordinate as a point object and increases by the offset value
Method: ARC.MessagingService.Navigation2DV1.LocationPoint.Equals(*System.Object*)

Determines if the current object cordinates equal the specified object cordinates

MovementManager

Event: ARC.MovementManager.OnMovement

Event risen when for movement action
Event: ARC.MovementManager.OnMovement2

Event risen when for movement action with speed
Event: ARC.MovementManager.OnSpeedChanged

Event risen when for speed changed
Field: ARC.MovementManager.LocomotionStyle

The type of locomotion that this robot uses to move in physical space
Type: ARC.MovementManager.LocomotionStyleEnum

List of locomotion styles
Field: ARC.MovementManager.LocomotionStyleEnum.Undefined

Locomotion type has not been specified by the movement panel, or a movement panel doesn't exist
Check the EZBManager.MovementPanel to see if it is null. If it is null, a movement panel doesn't exist
If it is not null, then the movement panel did not specify a locomotion type
Field: ARC.MovementManager.LocomotionStyleEnum.Wheeled_Steering

The robot has 2 wheels in the front that steer like an automobile
Field: ARC.MovementManager.LocomotionStyleEnum.Wheeled_Tracked

The robot has tracks like a tank for steering
Field: ARC.MovementManager.LocomotionStyleEnum.Drone

The robot flies like a drone with 4 or more blades
Field: ARC.MovementManager.LocomotionStyleEnum.GAIT

The robot walks with a GAIT like a hexapod or humanoid
Field: ARC.MovementManager.LocomotionStyleEnum.Helicopter

The robot flies like a helicopter (similar to a drone but not)
Field: ARC.MovementManager.LocomotionStyleEnum.Plane

The robot is a plane and flies with wings
Field: ARC.MovementManager.LocomotionStyleEnum.Rocket

The robot is a jet propelled rocket
Field: ARC.MovementManager.LocomotionStyleEnum.Submarine

The robot is a submarine
Field: ARC.MovementManager.LocomotionStyleEnum.Boat

The robot is a boat with a rudder, steerable jet or steerable motor
Type: ARC.MovementManager.MovementDirectionEnum

directions supported by movement manager
Field: ARC.MovementManager.MovementDirectionEnum.Stop

The robot is stopping
Field: ARC.MovementManager.MovementDirectionEnum.Forward

The robot is moving forward
Field: ARC.MovementManager.MovementDirectionEnum.Reverse

The robot is reversing
Field: ARC.MovementManager.MovementDirectionEnum.Left

The robot is turning left
Field: ARC.MovementManager.MovementDirectionEnum.Right

The robot is turning right
Field: ARC.MovementManager.MovementDirectionEnum.Up

The robot is moving up
Field: ARC.MovementManager.MovementDirectionEnum.Down

The robot is moving down
Field: ARC.MovementManager.MovementDirectionEnum.RollRight

The robot is rolling right
Field: ARC.MovementManager.MovementDirectionEnum.RollLeft

The robot is rolling left
Field: ARC.MovementManager.MovementDirectionEnum.Takeoff

Takeoff the robot (if flying)
Field: ARC.MovementManager.MovementDirectionEnum.Land

Land the robot (if flying)
Field: ARC.MovementManager.MovementDirectionEnum.Emergency

Handle an emergency situation where the robot needs to stop everything
Field: ARC.MovementManager.MovementDirectionEnum.Custom

A custom movement was specified
Check the GetCustomMovementID to see what custom movement you specified and react accordingly
Method: ARC.MovementManager.GetSpeed

Get the global speed
Method: ARC.MovementManager.GetSpeedLeft

Get the global speed for Left wheel
Method: ARC.MovementManager.GetSpeedRight

Get the global speed for Right wheel
Method: ARC.MovementManager.SetSpeed(*System.Byte*)

Set the speed for both wheels to be the same value
Method: ARC.MovementManager.SetSpeed(*System.Byte, System.Byte*)

Set the speed for each wheel
Method: ARC.MovementManager.SetSpeedLeft(*System.Byte*)

Set the left wheel speed
Method: ARC.MovementManager.SetSpeedRight(*System.Byte*)

Set the left wheel speed
Method: ARC.MovementManager.GoStop

Stops the robot if moving
Method: ARC.MovementManager.GoForward(*System.Byte*)

Moves robot forward at specified speed
Method: ARC.MovementManager.GoForward

Moves robot forward
Method: ARC.MovementManager.GoForward(*System.Byte, System.Byte*)

Moves robot forward
Method: ARC.MovementManager.GoReverse(*System.Byte*)

Moves robot backward at specified speed
Method: ARC.MovementManager.GoReverse

Moves robot backward
Method: ARC.MovementManager.GoReverse(*System.Byte, System.Byte*)

Moves robot backward
If FORCE=TRUE then the command will be set again even though the movement may already be the same direction.
Also, FORCE=TRUE will not raise the OnMovement event
Method: ARC.MovementManager.GoLeft(*System.Byte*)

Turns robot left at specified speed
Method: ARC.MovementManager.GoLeft

Turns robot left
Method: ARC.MovementManager.GoLeft(*System.Byte, System.Byte*)

Turns robot left
Method: ARC.MovementManager.GoRight(*System.Byte*)

Turns robot right at specified speed
Method: ARC.MovementManager.GoRight

Turns robot right at specified speed
Method: ARC.MovementManager.GoRight(*System.Byte, System.Byte*)

Turns robot right
Method: ARC.MovementManager.Takeoff

Robot Takes off (Flying robots)
Method: ARC.MovementManager.Land

Robot Lands (Flying robots)

Method: ARC.MovementManager.GoUp

Robot Goes Up (Drone flying robots)
Method: ARC.MovementManager.GoDown

Robot Goes Down (Drone flying robots)
Method: ARC.MovementManager.GoRollRight

Robot Rolls Right (Drone flying robots)

Method: ARC.MovementManager.GoRollLeft

Robot Rolls Left (Drone flying robots)
Method: ARC.MovementManager.GoEmergency

Instruct robot to execute the emergency mode (if supported)
For example, in a drone this will shut off the motors (EAK!)
Method: ARC.MovementManager.GoCustom(*System.String, System.Byte, System.Byte*)

To use a custom movement with your own Movement ID
The movement ID from the OnMovement events can be read with GetCustomMovementId
This is if you wanted to have your own movement handled by the movement manager
Method: ARC.MovementManager.GoCustom(*System.String, System.Byte*)

To use a custom movement with your own Movement ID
The movement ID from the OnMovement events can be read with GetCustomMovementId
This is if you wanted to have your own movement handled by the movement manager
Method: ARC.MovementManager.GoCustom(*System.String*)

To use a custom movement with your own Movement ID
The movement ID from the OnMovement events can be read with GetCustomMovementId
This is if you wanted to have your own movement handled by the movement manager

Scripting

Event: ARC.Scripting.JavaScript.JavascriptEngine.OnSetValues

Event to set custom values, methods, etc to the JavaScript engine
Event: ARC.Scripting.JavaScript.JavascriptEngine.OnUnsetValues

Event to remove custom values, methods, etc from the JavaScript engine
Method: ARC.Scripting.JavaScript.JavascriptEngine.GetMethodsFromReflectionForIntellisense(*System.String*)

Get all intellisense
Method: ARC.Scripting.ParseUtilities.ToBoolean(*System.Object*)

Checks if the inObj is a boolean.
Looks for "true", "false", "0", or "1"
This is useful when parsing parameters for the ControlCommand (i.e. SendCommand override)
Method: ARC.Scripting.Python.PythonEngine.GetMethodsFromReflectionForIntellisense(*System.String*)

Get all intellisense
Event: ARC.Scripting.ScriptEngineBase.OnStart

Event executed when a script has been started
Event: ARC.Scripting.ScriptEngineBase.OnDone

Event executed when the script has completed executing
Event: ARC.Scripting.ScriptEngineBase.OnResult

Event executed for the output of the script. For example the PRINT() statement
Event: ARC.Scripting.ScriptEngineBase.OnError

Event executed when a script has been started
Method: ARC.Scripting.EZScript.ScriptEngineEZScript.StartScriptASync(*System.String*)

Start the script in the background
Method: ARC.Scripting.EZScript.ScriptEngineEZScript.StartScriptBlocking(*System.String*)

Execute the script and block until it has completed
Event: ARC.Scripting.ScriptManager.OnAdded

Event raised when a new executor is created
Event: ARC.Scripting.ScriptManager.OnRemoved

Event raised when an executor is removed.
The executor is disposed after this event.
Method: ARC.Scripting.ScriptManager.AddCompiler(*ARC.Scripting.Executor*)

Add an executor to the collection manually
Method: ARC.Scripting.ScriptManager.DoesExecutorExist(*System.String*)

Returns true/false if the executor name exists in the collection
Method: ARC.Scripting.ScriptManager.GetExecutor(*System.String*)

Gets a reference to the executor specified by name, otherwise creates a new one and adds to the collection
Method: ARC.Scripting.ScriptManager.ClearAllExecutors

Stops scripts, clears all executors in the collection and disposes them

Method: ARC.Scripting.ScriptManager.RemoveExecutor(*ARC.Scripting.Executor*)

Removes the executor from the collection and disposes it
Method: ARC.Scripting.ScriptManager.RemoveExecutor(*System.String*)

Removes the executor from the collection and disposes it
Type: ARC.Scripting.VariableManager

Variable Manager is a static instance which stores all variables in the environment.

Variables come in two types:
1) Normal variable (i.e. $x = 1)
2) Array (i.e. $x[2] = 1)

Variables accept two data types for values:
1) String ("hello world")
2) Numeric (-2.31)

When storing a string variable value, never specify start/end quotes on the string. This module will take care of that for you.

Variable values may contain escape characters:
1) \r
2) \n
3) \"
Event: ARC.Scripting.VariableManager.OnVariableChanged

Event raised when a value of a variable has been changed.
If the variable is an array, the index will be populated with a value greater than 0
If the variable is not an array, the index will be -1
If you want the variable type, use the OnVariableChanged2 event
Event: ARC.Scripting.VariableManager.OnVariableChanged2

Event raised when a value of a variable has been changed.
If the variable is an array, the index will be populated with a value greater than 0
If the variable is not an array, the index will be -1
Type: ARC.Scripting.VariableManager.VariableTypeCls

This is a variable entry, which contains the variable name and the value.
The value can either be String or Numeric, which is determined inside of SetValue().
SetValue() will check to see if the value is a number, if so, it will be stored as such and no quotes will be wrapped around it
If the value is a string, the SetValue() will automatically wrap quotes around it.
*Note: Do not ever wrap your own strings in quotes!
Method: ARC.Scripting.VariableManager.VariableTypeCls.SetValue(*System.Object*)

*Note: Never wrap the value in quotes. This function will wrap the string in quotes for you.
Also note that your string must have escaped characters (\r, \n, \")
Method: ARC.Scripting.VariableManager.SubsituteWithValues(*System.Object*)

If you pass a string into here, it will replace all instances of known $variables with their appropiate value
Method: ARC.Scripting.VariableManager.GetVariable(*System.String*)

Get the value of a variable
Method: ARC.Scripting.VariableManager.GetVariable(*System.String, System.Int32*)

Get the value of a variable array by the specified index
Method: ARC.Scripting.VariableManager.SetVariable(*System.String, System.Boolean*)

Set the value of a variable
Method: ARC.Scripting.VariableManager.IsVariableReservedWord(*System.String*)

Throws an esception if the variableName is a reserved word (i.e. $date, $month, $year, etc)
This will also throw an exception if the variable does not start with a $
Method: ARC.Scripting.VariableManager.DoesVariableExist(*System.String*)

Check if a variable has been defined in memory
Method: ARC.Scripting.VariableManager.IsVariableArray(*System.String*)

Is the variable an array?
Method: ARC.Scripting.VariableManager.SetVariable(*System.String, System.Object*)

Set the value of a variable
Method: ARC.Scripting.VariableManager.CreateVariableArray(*System.String, System.Int32*)

Create an array with empty values in every position of Size
Method: ARC.Scripting.VariableManager.CreateVariableArray(*System.String, System.Object, System.Int32*)

Create an array with default value in every position of Size
Method: ARC.Scripting.VariableManager.CreateVariableArray``1( *System.String, ``0[]*)

Create an array with specified data
Method: ARC.Scripting.VariableManager.FillVariableArray(*System.String, System.Object*)

Fill an entire array with the specified value
Method: ARC.Scripting.VariableManager.GetArraySize(*System.String*)

Returns the length of an array
Method: ARC.Scripting.VariableManager.AppendToVariableArray(*System.String, System.Object*)

Grows the size of an array by 1 and adds this value to it.
You will need to CreateArray() before this can be used
Method: ARC.Scripting.VariableManager.SetVariable(*System.String, System.Object, System.Int32*)

Set the value of a variable within an array at the specified index
You will need to CreateArray before this can be called on a variable
Method: ARC.Scripting.VariableManager.DumpVariablesToString

Write all variables to a string for debugging purposes
Method: ARC.Scripting.VariableManager.ClearVariable(*System.String*)

Clear specified variable and value from memory
Method: ARC.Scripting.VariableManager.ClearVariables

Clears all variables and associated values from memory
Sets the default variables (i.e. movement, ezbsound, navigation status)
Method: ARC.Scripting.VariableManager.IsVariableValid(*System.String*)

Throws an exception with the formatting error of a variable name
Field: ARC.Scripting.Executor._textBox

For logging
Event: ARC.Scripting.Executor.OnStart

Event executed when a script has been started
Event: ARC.Scripting.Executor.OnDone

Event executed when the script has completed executing
Event: ARC.Scripting.Executor.OnResult

Event executed for the output of the script. For example the PRINT() statement
Event: ARC.Scripting.Executor.OnError

Event executed when a script has been started
Method: ARC.Scripting.Executor.BindOutputToTextBox(*System.Windows.Forms.TextBox*)

Binds the output (debug, errors, start, end, result) to a textbox

Renderer

Method: ARC.Renderer.ThemeRenderer.ApplyTheme(*System.Drawing.Color, System.Windows.Forms.Control[]*)

Apply the theme to the control. This can only be called once.
There's no need for you to call this, because it's called by the ARC control manager automatically.
You can provide a list of controls to ignore, and the theme won't be added to them
You can also add SkipTheme to the TAG element of any WinForms control to skip theming of that control and children.
Method: ARC.Renderer.ThemeRenderer.ApplyTheme(*System.Windows.Forms.Control[]*)

Apply the theme to the control. This can only be called once.
There's no need for you to call this, because it's called by the ARC control manager automatically.
You can provide a list of controls to ignore, and the theme won't be added to them
You can also add SkipTheme to the TAG element of any WinForms control to skip theming of that control and children.
Method: ARC.Renderer.ThemeRenderer.ApplyTheme

Apply the theme to the control/form using the default settings of the registry theme.
Method: ARC.Renderer.ThemeRenderer.ApplyTheme(*System.Drawing.Color*)

Apply the theme to the control. This can only be called once.
There's no need for you to call this, because it's called by the ARC control manager automatically.
You can also add SkipTheme to the TAG element of any WinForms control to skip theming of that control and children.

Services

Type: ARC.Services.AutoPosition.AutoPositionService

This is auto position engine which the Auto Position control uses. You can define servos, frames and actions for this.
This will run the auto position action in a background thread.
Event: ARC.Services.AutoPosition.AutoPositionService.OnComplete

Event risen when movement is complete
Event: ARC.Services.AutoPosition.AutoPositionService.OnStartAction

Event risen when an action is started
Event: ARC.Services.AutoPosition.AutoPositionService.OnStartFrame

Event risen when a frame is started
Field: ARC.Services.AutoPosition.AutoPositionService.Config

Set/Get the current configuration of Frames and Actions
Field: ARC.Services.AutoPosition.AutoPositionService.Name

Unique name for this auto position instance
Method: ARC.Services.AutoPosition.AutoPositionService.SetSpeed(*System.Byte*)

Value between 0-255 (0=slow, 255=fast)
Method: ARC.Services.AutoPosition.AutoPositionService.RequireInit

Set the require init flag so that the next time a transition is requested, it initializes the servos first.
Method: ARC.Services.AutoPosition.AutoPositionService.Stop

Stops the current movement. Blocks until stop is successful.
Method: ARC.Services.AutoPosition.AutoPositionService.MoveImmediate(*System.String*)

Move to the specified frame
Method: ARC.Services.AutoPosition.AutoPositionService.MoveToFrame(*ARC.Services.AutoPosition.AutoPositionActionFrame*)

Move into the selected position from the current position
Method: ARC.Services.AutoPosition.AutoPositionService.MoveToFrame(*System.String, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32*)

Move into the selected position from the current position
Method: ARC.Services.AutoPosition.AutoPositionService.ExecAction(*System.String*)

Execute the Action
Method: ARC.Services.AutoPosition.AutoPositionService.ExecAction(*ARC.Services.AutoPosition.AutoPositionAction.ActionTypeEnum*)

Execute the Action
Method: ARC.Services.AutoPosition.AutoPositionService.AddPauseToNewAction(*System.String, System.Int32*)

Add a pause frame to the specified action title with the number of milliscond delay.
If the action doesn't exist, it will be created for you.
This returns new action if it was created, or the old action that matched the name.
Method: ARC.Services.AutoPosition.AutoPositionService.AddAllServoPositionsToNewAction(*System.String*)

Read all of the servo positions that are defined in this auto position and add those positions to a new action as a new frame.
This returns the new action or the existing action that matched the title.
Method: ARC.Services.AutoPosition.AutoPositionService.AddAllServoPositionsToNewAction(*System.String, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32*)

Read all of the servo positions that are defined in this auto position and add those positions to a new action as a new frame.
This returns the new action or the existing action that matched the title.
Method: ARC.Services.AutoPosition.AutoPositionAction.ToString

Returns the title of this action as a string
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddFrame(*ARC.Services.AutoPosition.AutoPositionFrame*)

Add a frame to the list of frames. Returns the GUID of the frame
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddFrame(*System.String*)

Add a frame to the list of frames. Returns the GUID of the frame
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddFrame(*System.String, System.String*)

Add a frame to the list of frames. Returns the GUID of the frame
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddFrame(*System.String, System.String, System.Int32[]*)

Add a frame to the list of frames. Returns the GUID of the frame
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddFrame(*System.String, System.Int32[]*)

Add a frame to the list of frames. Returns the GUID of the frame
Method: ARC.Services.AutoPosition.AutoPositionConfig.GetUniqueFrameTitle(*System.String*)

Get a unique frame title by adding a number to the end
Method: ARC.Services.AutoPosition.AutoPositionConfig.AddAction(*ARC.Services.AutoPosition.AutoPositionAction*)

Add an action to the list of actions. Returns the GUID of the action

TimerSmart

Event: ARC.TimerSmart.Elapsed

Synthiam smart timer
- only allow once execution of an elapsed event at one time
- offers an IsEventRunning to know if an evnet is currently being executed from another thread
- offers IsTimerActive to know if the timer is actively running
Method: ARC.TimerSmart.Start

Starts the timer. If timer is already running, it will not run another.
This will gracefully exit if a timer is already running.

Common

Method: ARC.Common.SerializeObjectString(*System.Object*)

Serialize the object to an xml string
Method: ARC.Common.SerializeObjectString(*System.Object, System.Type*)

Serialize object to an xml string
Method: ARC.Common.SerializeObjectCompressed(*System.Object*)

Serialize object to an xml string compressed with gzipStream
Method: ARC.Common.SerializeObjectCompressed(*System.Object, System.Type*)

Serialize object to an xml string compressed with gzipStream
Method: ARC.Common.SerializeObjectCompressed(*System.String, System.Object*)

Serialize object to an xml file compressed with gzipStream
Method: ARC.Common.SerializeObjectFile(*System.String, System.Object*)

Serialize object to an xml file compressed with gzipStream
Method: ARC.Common.DeserializeObjectFile(*System.String, System.Type*)

Return the specified type of object deserialized from a file containing xml string
Method: ARC.Common.DeserializeObjectCompressed(*System.String, System.Type*)

Return the specified type of object deserialized from a gzip file containing xml string
Method: ARC.Common.DeserializeObjectCompressed(*System.Byte[], System.Type*)

Return the specified type of object deserialized from a gzip byte array
Method: ARC.Common.DeserializeObjectString(*System.String, System.Type*)

Deserialize xml string to specified object type

Method: ARC.Common.IsAlpha(*System.Object*)

Is the value a string by containing letters or anything other than digits
Method: ARC.Common.IsNumeric(*System.Object*)

Is the value a number
Will also check for decimal point and minus sign
Method: ARC.Common.IsApplicationAlreadyRunning

Check if ARC is already running
Method: ARC.Common.GetNumbersFromString(*System.String*)

Get only the numbers from a string
Returns 0 if no numbers are detected
Method: ARC.Common.GetNumbersFromString(*System.String, System.Int32*)

Get only the numbers from a string
Returns default if no numbers are detected
Method: ARC.Common.GetDecimalFromString(*System.String*)

Extract the decimal value from a string
Method: ARC.Common.GetListFromArray(*System.String[]*)

Joins the inArray strings into a comma separated string
Method: ARC.Common.IsUpdateAvailable(*System.String*)

Compares the newVersion against the current version of the software and returns whether they're the same
Method: ARC.Common.IsUpdateAvailable

Checks the synthiam.com server to see if there is a newer version of software available.
Method: ARC.Common.LatestUpdateVersion

Returns the most recent software version available from the synthiam.com website.
Method: ARC.Common.StrRemoveToEnd(*System.String, System.Char, System.Boolean*)

Removes all characters from the last instance of the specified string in the input string to the end of input string
Method: ARC.Common.StrRemoveToStart(*System.String, System.Char, System.Boolean*)

Removes all characters from the first instance of the specified character in the input string to the beginning of the input string
Method: ARC.Common.StartsWithAlphaCharacter(*System.String*)

Returns true if the first character of the input data is an alphabet character (i.e. A-Z or a-z)
Method: ARC.Common.StrStartsWithWord(*System.String, System.String*)

Returns true if the string starts with the specified word. Checks for Word by checking if the next character after the word are valid characters
Method: ARC.Common.RenameRegistrySubKey(*Microsoft.Win32.RegistryKey, System.String, System.String*)

Renames a subkey of the passed in registry key since
the Framework totally forgot to include such a handy feature.
Method: ARC.Common.CopyRegistryKey(*Microsoft.Win32.RegistryKey, System.String, System.String*)

Copy a registry key. The parentKey must be writeable.
Method: ARC.Common.RegistryRecurseCopyKey(*Microsoft.Win32.RegistryKey, Microsoft.Win32.RegistryKey*)

Copy a key and it's children to another key recursively
Method: ARC.Common.BitmapClone(*System.Drawing.Bitmap*)

Makes a deep clone of the bitmap image
Method: ARC.Common.BitmapGetFromByteArray(*System.Byte[], System.Boolean*)

Convert an array of bytes into a bitmap image.
The conversion is done with a memorystream. You can specify to dispose of that stream.
If you dispose of the stream, bewarned of the issues it will cause.
The original stream should stay with the lifetime of the bitmap.
Type: ARC.Common.WebResponseCls

Response from the v2 instances of Common.GetWebResponseV2 and Async version
Field: ARC.Common.WebResponseCls.Response

The response from the server
Field: ARC.Common.WebResponseCls.Message

Message the describes the error exception (if one exists)
Field: ARC.Common.WebResponseCls.Status

The status of the response
Method: ARC.Common.GetWebResponseAsyncV2(*System.String*)

Get the data from a web request using default ARC timeout
Method: ARC.Common.GetWebResponseAsyncV2(*System.String, System.Int32*)

Get the data from a web request
Method: ARC.Common.GetWebResponseAsync(*System.String*)

Get the data from a web request using default ARC timeout
Method: ARC.Common.GetWebResponseAsync(*System.String, System.Int32*)

Get the data from a web request
Method: ARC.Common.GetWebResponseV2(*System.String*)

Get the data from a web request using ARC default timeout
Method: ARC.Common.GetWebResponseV2(*System.String, System.Int32*)

Get the data from a web request
Method: ARC.Common.GetWebResponse(*System.String*)

Get the data from a web request
Method: ARC.Common.GetWebResponse(*System.String, System.Int32*)

Get the data from a web request
Method: ARC.Common.PostWebResponse(*System.String, System.String*)

Post data to a web url as content type application/x-www-form-urlencoded
Method: ARC.Common.PostWebResponse(*System.String, System.String, System.Int32*)

Post data to a web url as content type application/x-www-form-urlencoded
Method: ARC.Common.PostWebResponse(*System.String, System.String, System.Int32, System.String[], System.String[]*)

Post data to a web url as content type application/x-www-form-urlencoded
Method: ARC.Common.IsInternetConnected

Check if there is a valid connection to the synthiam website
Method: ARC.Common.IsInternetConnectedAsync

Check if there is a valid connection to the synthiam website
Method: ARC.Common.GetRemoteIPAddress

Query the web service to get the ip address of this machine
Method: ARC.Common.SplitOnlyOnce(*System.String, System.Char*)

Splits only the first instance of the character
Method: ARC.Common.GetTextBetween(*System.String, System.String@, System.Char, System.Char*)

Returns the text between the start and end token
Method: ARC.Common.Min(*System.Double[]*)

Returns the minimal value of the list
Method: ARC.Common.MaxOrDefault(*System.Object, System.Int32*)

Returns the higher of the two values
Returns the default value if InVal is invalid type of INT
You don't need to cast the inVal, it'll be done within this method
So you can pass a string and it'll parse it (or try to)
If it can't parse it, it'll return the defaultVal
Method: ARC.Common.Max(*System.Double[]*)

Returns the maximum value of the list
Method: ARC.Common.DecompressToStr(*System.Byte[]*)

decompress array of bytes to string
Method: ARC.Common.DecompressToFile(*System.Byte[], System.String*)

decompress array of bytes
Method: ARC.Common.Decompress(*System.Byte[]*)

decompress array of bytes
Method: ARC.Common.CompressToFile(*System.IO.Stream, System.String*)

compress array of bytes
Method: ARC.Common.CompressFromFile(*System.String*)

compress array of bytes
Method: ARC.Common.Compress(*System.Byte[]*)

compress array of bytes
Method: ARC.Common.Compress(*System.IO.Stream*)

compress array of bytes
Method: ARC.Common.GetEnumeratorTypeFromString(*System.Type, System.String*)

Returns the enumerator type that matches the string
Method: ARC.Common.GetBytes(*System.String*)

Converts a string into a byte array. The sytem.encoding will remove any bytes above 127, this will not
Method: ARC.Common.GetString(*System.Byte[]*)

Converts an array of bytes into a string - but uses extended ascii (8 bit) instead of 7 bit
Method: ARC.Common.GetTextUntilOrEmpty(*System.String, System.String, System.Boolean*)

Get text from beginning of string up until the stopAt character/string
Returns empty string if the character is not found
Method: ARC.Common.TextEncode(*System.String*)

Performs an HtmlEncode on the text
Method: ARC.Common.TextDecode(*System.String*)

PErforms an HtmlDecode on the text
Method: ARC.Common.GetValueFromHexString(*System.String*)

Returns the byte value of the hex string (ie 0xa0), or numeric. Size if of byte
Method: ARC.Common.GetInvertedColorToGreyScale(*System.Drawing.Color, System.Int32, System.Int32*)

Return the invert of the specified color
minBrightness and maxBrightness are between 0-255 and that's the hard limit for the returned value
Method: ARC.Common.GetInvertedColorToGreyScale(*System.Drawing.Color, System.Boolean, System.Int32, System.Int32*)

Return the invert of the specified color
If absolute is specified, the value will be either black or white but no transisition
minBrightness and maxBrightness are between 0-255 and that's the hard limit for the returned value
Method: ARC.Common.ChangeColorBrightness(*System.Drawing.Color, System.Single*)

Brighten the color by percentage (-1 to 0 to +1)
Negative value is darker, positive value is lighter
Method: ARC.Common.ChangeControlsRecursiveColor(*System.Windows.Forms.Control, System.Drawing.Color, System.Drawing.Color, System.Windows.Forms.Control[]*)

Recursive change the top control and all children controls of the specified topControl to the specified colors.
If you do not wish to set a color, set it to null and it will be skipped
This will not suspend the layout
Method: ARC.Common.ChangeChildrenControlsRecursiveColor(*System.Windows.Forms.Control, System.Drawing.Color, System.Drawing.Color, System.Windows.Forms.Control[]*)

Recursive change only children controls of the specified topControl to the specified colors.
This does not change the color of the topControl, only the children.
If you do not wish to set a color, set it to null and it will be skipped
This will not suspend the layout
Method: ARC.Common.ChangeColorBrightness2(*System.Drawing.Color, System.Single*)

Brighten the color by percentage (-1 to 0 to +1)
Negative value is darker, positive value is lighter
Method: ARC.Common.Quote(*System.Object*)

Quote the input value if it's a string. Keep it alone if it's a number
Internal quotes are escaped
If there are starting and ending quotes, additional quoting will be ignored
Method: ARC.Common.CopyDirectory(*System.String, System.String, System.Boolean*)

Copy the contents of source directory into the dest directory
Method: ARC.Common.IsFileLocked(*System.String*)

Is the file locked (i.e. used elsewhere)
Method: ARC.Common.IsFileLocked(*System.IO.FileInfo*)

Is the file locked (i.e. used elsewhere)
Method: ARC.Common.DeepCopy``1( ``*0*)

Copies/clones a serializable class as a deep copy. Meaning, the two are unreleated.
Method: ARC.Common.MD5Encrypt(*System.String*)

Return md5 hashed string of input string
Method: ARC.Common.MD5Encrypt(*System.Byte[]*)

Return md5 hashed string of input string
Method: ARC.Common.MD5Encrypt(*System.IO.Stream*)

Return md5 hashed string of input string
Method: ARC.Common.ShutDownPC

Shuts down the computer without prompting to save project or anything
Method: ARC.Common.GetSizeOfFolder(*System.IO.DirectoryInfo, System.Boolean*)

Get size of folder in bytes

Constants

Field: ARC.Constants.MOUSE_DRAG_SENSITIVITY_NORMAL

The default mouse drag sensitivity for the slide up and down number boxes
Field: ARC.Constants.MOUSE_DRAG_SENSITIVITY_SHIFT

The default mouse drag sensitivity for the slide up and down number boxes when shift is held
Field: ARC.Constants.GET_WEB_RESPONSE_TIMEOUT_MS

Default timeout value for get and post web calls. This can be overriden in the registry with key 'GetWebResponseTimeoutMS'
Field: ARC.Constants.WEB_SERVICE_TIMEOUT

The default timeout for web service calls in the web service wrapper.
Can be overridden with registery key 'WebServiceTimeoutMS'
Field: ARC.Constants.DEFAULT_FOLDER

The My Documents\ARC
Field: ARC.Constants.DEFAULT_APPLICATION_FOLDER

The ProgramData\ARC
Field: ARC.Constants.DEFAULT_PROJECT_FOLDER

My Documents\ARC\My Projects
Field: ARC.Constants.PROJECT_BACKUP_FOLDER

My Documents\ARC\Projects Backups
Field: ARC.Constants.EZ_BITS_STL_SAVE_FOLDER

My Documents\ARC\My STL Files
Field: ARC.Constants.DEFAULT_USER_LOGS_FOLDER

My Documents\ARC\Logs
Field: ARC.Constants.SERVO_PROFILES_FOLDER

My Documents\ARC\Servo Profiles
Field: ARC.Constants.AUTO_POSITIONS_FOLDER

My Documents\ARC\Auto Positions
Field: ARC.Constants.HTTP_SERVER_CUSTOM_FOLDER

My Documents\ARC\HTTP Server Root
Field: ARC.Constants.PROJECT_HISTORY_FILE

My Documents\ARC\ProjectHistory.txt
Field: ARC.Constants.RECENT_SKILLS_FILE

My Documents\ARC\SkillHistory.txt
Field: ARC.Constants.BEHAVIOR_CONTROL_PROJECT_FOLDER

My Documents\ARC\Behavior Control Projects
Field: ARC.Constants.ARC_INSTALLATION_FOLDER

Where ARC is installed (ie Program Files (x86)\Synthiam Inc.\ARC by Synthiam
Field: ARC.Constants.TEMP_FOLDER

C:\Users\[current user]\AppData\Local\Temp\ARC
Field: ARC.Constants.DEFAULT_PROJECT_EXAMPLES_FOLDER

ProgramData\ARC\Examples
Field: ARC.Constants.EZ_BITS_CACHE_FOLDER

ProgramData\ARC\EZ-Bits v2
Field: ARC.Constants.DEFAULT_LOGS_FOLDER

ProgramData\ARC\Logs
Field: ARC.Constants.FIRMWARE_XML_FILE

ProgramData\ARC\firmwares.xml
Field: ARC.Constants.TEAMS_KNOWN_ISSUES_FILE

Copy of the known issues file from the web. Updated for Teams during start-up if there is an internet connection.
Field: ARC.Constants.MEDIA_SAVE_FOLDER

My Pictures\My Robot Pictures
Field: ARC.Constants.EZ_ROBOT

The project configuration. This is the only part of the project that is static and can be referenced
while a project is loaded. All other parts of a project configuration are serialized from the controls.
This holds the theme inforamtion, such as wallpaper, project filename, ezbits, etc..

EZBManager

Type: ARC.EZBManager

The main manager for communicating with EZ-B's using many provided helpers.
Event: ARC.EZBManager.OnConnectionChange

Event risen when connection to any ezb changes
Field: ARC.EZBManager.EZBuilderLoadedDateTime

Time that EZ-Builder was first loaded.
Field: ARC.EZBManager.AudioManager

The audio manager provides audio functions for recording and playing back audio.
To be friendly on resources, controls can share these methods and events.
Method: ARC.EZBManager.Log(*System.Object, System.Object[]*)

The global log is useful to notify the user of an error or message.
This popups up a dialog at the bottom of the workspace for a few seconds with the debug message.
This also logs the message to the log file in the user's My Documents\EZ-Builder folder.
Field: ARC.EZBManager.MovementManager

The movement panels control physical locomotion of the robot. The locomotion is handled through this movement class.
The locomotion of a robot is _ALWAYS_ the first EZ-B index in the list.
Your custom movement panel will obtain movement direction changes from this class.
Method: ARC.EZBManager.ClearEZBs

Clear all EZBs from the EZBs list above 5. This will not clear the EZ-B's on index 0-4 (the first 5). There will always be at least 5 EZ-B's.

Method: ARC.EZBManager.AddEZBToList(*System.String*)

Add an ezb and return the new index. The EZBs can be referenced from the EZBs list.
The ID should be unique
Method: ARC.EZBManager.RemoveLastEZBInList

Remove the last ezb from the list
Method: ARC.EZBManager.GetPing(*System.String, System.String*)

Get the distance returned by the ping distance sensor.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetPWM(*System.String, System.Int32*)

Sets the PWM output on the specified port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.GetServoSpeed(*System.String*)

Get the current servo speed that was set for the specified port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.GetPWM(*System.String*)

Get the current PWM of the specified port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetServoSpeed(*System.String, System.Int32*)

Sets the servo speed on the specified port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetServoPosition(*System.String, System.Int32*)

Sets the servo absolutely position on the specified.
This accepts a servo port
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetServoMaxLimit(*System.String, System.Int32*)

Sets the maximum servo limit of the specified servo port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetServoMinLimit(*System.String, System.Int32*)

Sets the mininum servo limit on the specified port.
This accepts a string port in the format of D2, D3, etc. Or 1.D3 or 3.D5 if specifying the EZ-B, such as with EZ-Script.
Method: ARC.EZBManager.SetReleaseServo(*System.String*)

Release servo from it's holdling position.
This accepts the port by string such as D0, D3 or 3.D5 or 5.D2

Method: ARC.EZBManager.GetServoPosition(*System.String*)

Get the position of the servo specified by the port as a string.
For example, this can be D3, D5, 2.D2, 3.D5, etc..
This is the last position the servo was instructed to move to, not the position of the servo if you manually moved it with your hand.
OR you can also use GetServoPositionRealtime() if the servo is a smart servo, such as Dynamixel.
Method: ARC.EZBManager.GetServoPositionRealtime(*System.String*)

Get the position of the Smart servo specified by the port as a string.
For example, this can be V3, V5, 2.V5 or 4.V34, etc...
This only works on Virtual servos and will throw an exception otherwise.
The virtual servo must support this ability otherwise an exception will be thrown.
Method: ARC.EZBManager.SetDigital(*System.String, System.Boolean*)

Sets the digital port output to the specified value.
This supports the port to be in the string format, such as D3, D6, 3.D8, etc...
Method: ARC.EZBManager.SetDigital(*System.Int32, EZ_B.Digital.DigitalPortEnum, System.Boolean*)

Sets the digital port to the specified value.
Method: ARC.EZBManager.GetDigital(*System.String*)

Get the digtial port status.
This supports the port to be in the string format, such as D3, D6, 3.D8, etc...
Method: ARC.EZBManager.GetADC12Bit(*System.String*)

Get the ADC in 12 bit resolution
This supports the port to be in the string format, such as A1, A2, 3.A3, etc...
Method: ARC.EZBManager.GetADC(*System.String*)

Get the ADC in 8 bit resolution
This supports the port to be in the string format, such as A1, A2, 3.A3, etc...
Method: ARC.EZBManager.SendSerial(*System.String, EZ_B.Uart.BAUD_RATE_ENUM, System.Byte[]*)

Transmit serial data out of the specified digital port, if supported by the hardward
This supports the port to be in the string format, such as A1, A2, 3.A3, etc...

Invokers

Type: ARC.Invokers

It's standard to run intensive processes in separate thread than the UI. This means you can't simply modify the UI thread objects from another thread.
This class contains a lot of threadsafe functions to Get/Set properties of UI objects.

FormMain

Type: ARC.FormMain

The FormMain is the user interface controller for the control framework. In here are commands to find controls, details of the current project, access the desktop manager, and more.
Field: ARC.FormMain._ProjectFilename

The current project that is loaded.
Field: ARC.FormMain._LastOpenFolder

The last folder that was opened from the FormOpenEZB
Field: ARC.FormMain._IS_CLOSING_PROJECT

Is the project or application closing? This is mostly used for controls to close without prompting the user. You can handle this in your code if you have any loops or timers that need to know if the project or application is closing.
Field: ARC.FormMain._IS_CLOSING_APP

Is the application closing?
Field: ARC.FormMain.MovementPanel

The current movement panel registered with the project. There can only be one movement panel, so this should be null if there isn't one, or have a reference to it if there is one. If you create a movement panel, you must register it here, and you must set this to null when your control closes.
Field: ARC.FormMain.ucControls

This is the object that hosts the virtual desktops and all of the controls.
Type: ARC.FormMain.OnProjectLoadCompletedHandler

Event raised after a project has completely loaded all skill controls.
If you're control is looking to bind to another control, find the control in this event.
If you attempt to look for a control (i.e. camera) during constructor or SetConfiguration, the other control
may not have loaded from the config yet. This event is raised after all of the controls have been loaded to the workspace
Event: ARC.FormMain.OnProjectLoadCompleted

Event raised after a project has completely loaded all skill controls.
If you're control is looking to bind to another control, find the control in this event.
If you attempt to look for a control (i.e. camera) during constructor or SetConfiguration, the other control
may not have loaded from the config yet. This event is raised after all of the controls have been loaded to the workspace
Type: ARC.FormMain.OnBehaviorControlAddedHandler

Event raised when a control is added to the workspace. This could be during the project load event, or if a user uses the Add Control menu.
If you're wanting to keep track of new controls added to the workspace, this is how to do it.
However, if you're expecting a control to exist when a project is loaded, look into OnProjectLoadCompleted event.
Look into OnBehaviorControlRemoved as well

Event: ARC.FormMain.OnBehaviorControlAdded

Event raised when a control is added to the workspace. This could be during the project load event, or if a user uses the Add Control menu.
If you're wanting to keep track of new controls added to the workspace, this is how to do it.
However, if you're expecting a control to exist when a project is loaded, look into OnProjectLoadCompleted event.
Look into OnBehaviorControlRemoved as well
Event: ARC.FormMain.OnPanicReleaseServosPressed

There is a button on the Options tab of the main menu ribbon bar for PANIC RELEASE/STOP SERVOS
If that button is pressed, this event is triggered. Your control can respond to it if necessary.
Such as, stop the functions or stop what you're doing. It's a panic stop all type scenario
triggered by the user.
Type: ARC.FormMain.OnPanicReleaseServosPressedHandler

There is a button on the Options tab of the main menu ribbon bar for PANIC RELEASE/STOP SERVOS
If that button is pressed, this event is triggered. Your control can respond to it if necessary.
Such as, stop the functions or stop what you're doing. It's a panic stop all type scenario
triggered by the user.
Method: ARC.FormMain.SoundV4_OnStopPlaying

Sets the $EZBPlayingAudio variable to false
Method: ARC.FormMain.Movement_OnMovement(*ARC.MovementManager.MovementDirectionEnum*)

Sets the $Direction movement variable to the specified direction.
Method: ARC.FormMain.NewProject

Closes the current project and creates a new empty blank project
Method: ARC.FormMain.MenuClick_Open(*System.Object*)

Opens the FormOpenEZB dialog at the location of the _LastOpenFolder. If a project is specified, it loads the project.
Method: ARC.FormMain.OpenProject(*System.String*)

Open the specified project and close the existing project
Method: ARC.FormMain.SendFormCommand(*System.String, System.String, System.String[]*)

Send a ControlCommand() to the specified control on any desktop.
Method: ARC.FormMain.GetFormValue(*System.String, System.String, System.Object*)

Returns a value if the control supports it. The control must have an override for GetValue(string value). If so, you can query the value from a window with this.
Method: ARC.FormMain.SetupProject(*System.String*)

Loads the project into the existing project and does not clear the existing project. This means it merely sets up the projects by loading the controls and configuration. If you want to load a project, then use the LoadProject() method.
The filename will be loaded as and parsed into a project.
Method: ARC.FormMain.DoesFormTypeAlreadyExist(*System.Type[]*)

Does the type of form or control exist? Use this if you're looking to see if a control has already been added to a form. For example, you can check if a typeof(FormCameraDevice) exists.

Method: ARC.FormMain.DoesMovementPanelExist

Becuase only one movement panel can exist on a project at once, this returns if a movement panel has been registered in MovementPanel.
Method: ARC.FormMain.NewConnection

Add a connection control or Show() the connection control that is already on the project.
Only one connection control should exist in a project. Having two connection controls will break-down the universe into a obvlivion
Method: ARC.FormMain.PanicReleaseStopServos

Send a panic/stop/release servos command to all controls that bind to the associated event.
If you call this method, the event will be raised and any controls that bind to the event will
be responsible to stop their execution. This is a panic mode scenario generated by the user.
Method: ARC.FormMain.SetBackgroundColor(*System.Drawing.Color*)

Sets the background color of the workspace. This won't work if there is already an Image on the workspace. You will need to SetBackgroundImage(null) to see the background color.
Method: ARC.FormMain.SetBackgroundImage(*System.Drawing.Image*)

Sets the background to an image on the workspace. This overrides the background color with the image.
Method: ARC.FormMain.SuspendLayoutAllTabs

Suspends the layout of all controls in all tabs. This is used when significant layout changes are about to occur. (i.e. moving windows or resizing them, etc).
The ResumeLayout MUST be called after you've completed re-arranging the controls.
Method: ARC.FormMain.ResumeLayoutAllTabs

Resumes the layout event that was suspended with the SuspendLayoutAllTabs()
Method: ARC.FormMain.SuspendLayoutCurrentTab

Suspends the layout of all controls in the current tab. This is used when significant layout changes are about to occur. (i.e. moving windows or resizing them, etc).
The ResumeLayout MUST be called after you've completed re-arranging the controls.
Method: ARC.FormMain.ResumeLayoutCurrentTab

Resumes the layout event that was suspended with the SuspendLayoutCurrentTabs()
Method: ARC.FormMain.AddControl(*System.Windows.Forms.Control*)

Adds the skill control to the current tab.
Method: ARC.FormMain.AddControl(*System.Windows.Forms.Control, System.Int32*)

Adds a skill control to the specified page tab. The first virtual desktop is page 0 and so on...
Method: ARC.FormMain.AddControl(*System.Windows.Forms.Control[], System.Int32*)

Adds skill controls to the specified page tab. The first virtual desktop is page 0 and so on...
Method: ARC.FormMain.SmartArrangeSelectedPage(*System.Boolean, System.Boolean*)

Arranges the skill controls on current page to optimize available screen space effeciently. You can order large to small or opposite.
If you're calling this while already suspending the layout, you can also override the suspend layout option with the variable.
If you do not ignore the suspend layout yourself, set the variable to false and this method will handle the layout suspension and resume.
You may be handling the suspend layout yourself because you're performing a number of workspace changes, in which case you'd want to ignore the suspend layout.
Method: ARC.FormMain.SmartArrangeAll(*System.Boolean, System.Boolean*)

Arranges the skill controls on all pages to optimize available screen space effeciently. You can order large to small or opposite.
If you're calling this while already suspending the layout, you can also override the suspend layout option with the variable.
If you do not ignore the suspend layout yourself, set the variable to false and this method will handle the layout suspension and resume.
You may be handling the suspend layout yourself because you're performing a number of workspace changes, in which case you'd want to ignore the suspend layout.
Method: ARC.FormMain.SmartArrange(*System.Int32, System.Boolean, System.Boolean*)

Arranges the skill controls on specified page to optimize available screen space effeciently. You can order large to small or opposite.
Desktop is either 0, 1, or 2 (Desktops supporting controls)
If you're calling this while already suspending the layout, you can also override the suspend layout option with the variable.
If you do not ignore the suspend layout yourself, set the variable to false and this method will handle the layout suspension and resume.
You may be handling the suspend layout yourself because you're performing a number of workspace changes, in which case you'd want to ignore the suspend layout.
Method: ARC.FormMain.SmartArrange(*System.Windows.Forms.Panel, System.Boolean, System.Boolean*)

Arranges the skill controls on specified page to optimize available screen space effeciently. You can order large to small or opposite.
If you're calling this while already suspending the layout, you can also override the suspend layout option with the variable.
If you do not ignore the suspend layout yourself, set the variable to false and this method will handle the layout suspension and resume.
You may be handling the suspend layout yourself because you're performing a number of workspace changes, in which case you'd want to ignore the suspend layout.
Method: ARC.FormMain.GetSkillCount(*System.Type*)

Return the number of all skills that have been added to the project workspace
Method: ARC.FormMain.GetSkillCount

Return the number of all skills that have been added to the project workspace
Method: ARC.FormMain.GetControlCountByNameAllPages(*System.String*)

Get the number of controls that have the title. This is NOT case sensitive. Meaning, it will count controls and ignore the case
Method: ARC.FormMain.GetControlByNameAllPages(*System.String*)

This will get the control (or first control if there are more than one) with the specified title.
This is NOT case sensitive, meaning it any case will be returned.
Method: ARC.FormMain.GetControlByType(*System.Type*)

Get a list of controls that match the type. This is helpful if you're looking for a FormCameraDevice, for example.
This searchs all tabs.
Method: ARC.FormMain.GetControlsAllPages

Returns all controls in the project on all tabs
Method: ARC.FormMain.GetControls(*System.Windows.Forms.Control*)

Gets the controls on the specified tabpage. The page comes from the ucControls workspace manager.
Method: ARC.FormMain.GetControls(*System.Int32*)

Gets the controls by the specified page index. The first virtual workspace index is 0.

Method: ARC.FormMain.GetControlsCurrentPage

Gets all controls on the current active workspace.
Method: ARC.FormMain.LoadInterfaceBack

The last virtual desktop workspace is a full screen mobile app view. This will load the previous workspace in the breadcrumb trail.
Essentially it Pops the breadcrum from the stack. When a new interface is displayed with LoadInterfaceByName() or LoadInterfaceByConfiguration(), the current interface is added to the stack.
This is like a BACK button to go to the previous interface.
Method: ARC.FormMain.LoadInterfaceByName(*System.String*)

The last virtual desktop workspace is a full screen mobile app view.
This will load the mobile interface by the window title into that window and add it to the stack.
The previous interface on the stack can be restored with LoadInterfaceBack()
Method: ARC.FormMain.CloseFormByType(*System.Type[]*)

Close the control form by the type. This is useful if you wish to close all forms of a specified type. For example, close all FormCameraDevice.
Method: ARC.FormMain.CloseControls(*System.Boolean*)

Close all controls on all workspace tabs.
You can ignore the suspend layout if you're code is handling the layout suspension and resume manually.
Method: ARC.FormMain.CloseIntroControls(*System.Boolean*)

When the software first loads, a number of introduction controls are added to the workspace.
These controls are things like Bookmarks, Check for plugins, Check for updates, etc..
This will close those controls.
Method: ARC.FormMain.UpdateVirtualDesktopShots

The virtual desktop screenshots are displayed in the ribbon menu for accessibility.
If the workspace has changed, you acn manually update the screenshot with this method.
Method: ARC.FormMain.updateVirtualDesktopButtonsAndScreenshots

Make sure the buttons in the menu bar match the count of virtual desktops (adds or removes buttons based on desktop count)
Also updates the thumbnails and workspace titles

URLServiceManager

Method: ARC.URLServiceManager.ViewUrl(*System.Windows.Forms.Control, System.String*)

Opens the url in the default browser. Displays a message to the user if no internet connection is discovered
Method: ARC.URLServiceManager.ViewUrl(*System.Windows.Forms.Control, ARC.URLServiceManager.URLEnum*)

Opens the url in the default browser. Displays a message to the user if no internet connection is discovered
Returns true if internet otherwise false if no internet

VAD

Type: ARC.VAD.ByteBuffer

Class for converts among different array types. Inspired with NAudio WaveBuffer class https://github.com/naudio/NAudio
Type: ARC.VAD.FFT2

Fft implementation from https://gerrybeauregard.wordpress.com/2011/04/01/an-fft-in-c/
Field: ARC.VAD.VoiceActivityDetector.REQUIRED_BUFFER_SIZE

160 samples * 2 bytes per sample

WebServiceWrappers

Method: ARC.WebServiceWrappers.Tools.IsProxySetByUser

Returns true if the user has configured a proxy server for their installation.
Method: ARC.WebServiceWrappers.Tools.GetWebProxy

Returns the proxy object that can be used for a webservice or WebRequest
Method: ARC.WebServiceWrappers.WebRequest.Create(*System.Uri*)

Returns an instance of a web request with the proxy server information, if specifeid
Method: ARC.WebServiceWrappers.WebRequest.Create(*System.String*)

Returns an instance of a web request with the proxy server information, if specifeid

EZB Classes, Methods, & Events

This outlines the EZB API, which has Events, Methods, and Fields. You can search for any of those with the browser using CTRL-F to find all events, for example. Many classes and respective methods may not be documented here because they are less commonly used. These are the most common API calls for robot skills.

Jump To...

- ADC
- ARDrone
- AudioEffects
- AvgHistogramCls
- AVM
- BlinkM
- BV4615
- Camera
- CameraDetection
- CameraDummyDevice

- Classes
- ConfigurationManager
- Digital
- Extensions
- EZ430
- EZB
- EZBv4Manager
- EZBv4Sound
- EZBv4Video
- EZTaskScheduler
- FFMPEGUtils
- Firmware
- FormConnectionResources
- Functions
- HC_SR04
- HT16K33
- HT16K33Animator
- I2C
- Imaging
- Joystick
- JPEGStream
- MMA7455
- MP3Trigger
- MusicSynth
- NeoPixelController8
- ObjectLocation
- PWM
- RandomUnique
- Resource1
- RGB8x8
- RGB8x8Animator
- RGBAnimator
- RGBEyes
- RSS
- Servo
- SoundTouch
- Speakjet
- SpeechSynth
- Sphero
- SureDualAxisCompass
- TCPServer
- TellyMate
- Twitter
- Uart
- UARTVideo
- UCCameraCanvas
- UCComboBoxTextBox
- UCEZB_Connect
- Video
- VideoPlayer
- Vision
- Vuzix

ADC

Type: EZ_B.ADC.ADCPortEnum

List of ADC Ports
Method: EZ_B.ADC.GetADCValue(*EZ_B.ADC.ADCPortEnum*)

Get an integer from 0-255 (8 bits) representing the relative voltage of a specified ADC port (Between 0 and 5 volts)
Method: EZ_B.ADC.GetADCValue12Bit(*EZ_B.ADC.ADCPortEnum*)

Get an integer from 0-4096 (12 bits) representing the relative voltage of a specified ADC port (Between 0 and 5 volts)
Method: EZ_B.ADC.GetADCVoltageFromValue(*System.Int32*)

Returns the voltage relative to the inputted value. If you want to display the Value and Voltage, you can pass the value to this function rather then executing a new command. This saves bandwidth over the line.
Method: EZ_B.ADC.GetADCVoltage(*EZ_B.ADC.ADCPortEnum*)

Get the voltage from 0-5v of a specified ADC port

AvgHistogramCls

Method: EZ_B.AvgHistogramCls.Add(*System.Double*)

Add a new item to the histogram and return the current average

CameraDetection

Method: EZ_B.CameraDetection.CustomYCbCrColorDetection.GetObjectLocationByColor(*System.Boolean, System.Boolean, System.Int32, System.Single, System.Single, System.Single, System.Single, System.Single, System.Single*)

Check for an object of the specified color. Returns a class that references its location.
The searchObjecTSizePixels is the number of pixels for the minimum detected object size and is suggested around 25.
hueMin and hueMax is a range of Hue for the color.
SaturatioMin and Saturation Max is the range of Saturation for the color.
LuminanceMin and LuminanceMax is the range of Luminance for the color.
Method: EZ_B.CameraDetection.CustomYCbCrColorDetection.GetObjectLocationByColor(*System.Boolean, EZ_B.Classes.CustomYCbCrColorConfig*)

Check for an object of the specified color. Returns a class that references its location.
Method: EZ_B.CameraDetection.AVMObjectDetection.LoadObjects(*EZ_B.AVM.TrainedObjectsContainer*)

Load any trained objects from an XML container of a saved file
Method: EZ_B.CameraDetection.AVMObjectDetection.GetDetectedObjects(*System.Boolean, System.Boolean*)

Method: EZ_B.CameraDetection.CustomColorDetection.GetObjectLocationByColor(*System.Boolean, System.Boolean, System.Int32, System.Int32, System.Int32, System.Single, System.Single, System.Single, System.Single*)

Check for an object of the specified color. Returns a class that references its location.
The searchObjecTSizePixels is the number of pixels for the minimum detected object size and is suggested around 25.
hueMin and hueMax is a range of Hue for the color.

SaturatioMin and Saturation Max is the range of Saturation for the color.
LuminanceMin and LuminanceMax is the range of Luminance for the color.
Method: EZ_B.CameraDetection.CustomColorDetection.GetObjectLocationByColor(*System.Boolean, EZ_B.Classes.CustomColorConfig*)

Check for an object of the specified color. Returns a class that references its location.
Method: EZ_B.CameraDetection.QRCodeDetection.GetObjectLocationByQRCode

Check for a QRCode. Returns a class that references its location.
Type: EZ_B.CameraDetection.ColorDetection.ColorEnum

A list of colors used for searching for objects.
Method: EZ_B.CameraDetection.ColorDetection.GetObjectLocationByColor(*System.Boolean, EZ_B.CameraDetection.ColorDetection.ColorEnum, System.Int32, System.Byte*)

Check for an object of the specified color. Returns a class that references its location.
You can use the returned class information to determine what direction to move your robots head.
minBrightness will need to be adjusted for the environment. Higher number is for brighter images. Lower number is for darker environments.
The searchObjecTSizePixels is the number of pixels for the minimum detected object size
Method: EZ_B.CameraDetection.CustomHaarDetection.LoadHaarCascade(*System.String*)

Load a custom Haar Cascade XML file to be detected
Method: EZ_B.CameraDetection.CustomHaarDetection.GetCustomDetection

Get the location of a detected object
Method: EZ_B.CameraDetection.FaceDetection.GetFaceDetection(*System.Int32, System.Int32, System.Int32*)

Get the location of a detected face
The smallest size and largest size is a limit in pixels (width or height) of the detected face
The mininumDetectionCount is how many frames to detect before returning a positive detection. This is used to filter false positives
Field: EZ_B.CameraDetection.GlyphDetection.Glyph1Overlay

Glyph Overlay Image for Augmented Reality. Set this image and it will be overlayed on top of the actual glyph.
Field: EZ_B.CameraDetection.GlyphDetection.Glyph2Overlay

Glyph Overlay Image for Augmented Reality. Set this image and it will be overlayed on top of the actual glyph.
Field: EZ_B.CameraDetection.GlyphDetection.Glyph3Overlay

Glyph Overlay Image for Augmented Reality. Set this image and it will be overlayed on top of the actual glyph.
Field: EZ_B.CameraDetection.GlyphDetection.Glyph4Overlay

Glyph Overlay Image for Augmented Reality. Set this image and it will be overlayed on top of the actual glyph.
Method: EZ_B.CameraDetection.GlyphDetection.GetGlyphDetection

Get the location of shapes
Method: EZ_B.CameraDetection.MotionDetection.GetMotionDetection(*System.Int32, System.Int32, System.Int32*)

Return an object that describes the location of the change in motion.
Suggested values are: ColorFunctions.Difference=30, CountLimit=80
The searchObjecTSizePixels is the number of pixels for the minimum detected object size
skipFrames value determines how many frames to wait before checking the difference. In most cases, this value can be a 0. However, you can use this value to wait until the robot has moved from the last update before checking for motion.

CameraDummyDevice

Type: EZ_B.CameraDummyDevice

This is a dummy camera device that is called "Custom" in the camera device list
This is used for when a robot skill wants to push a video stream into the camera device
You can bind to the EZ_B.Camera.OnStart, EZ_B.Camera.OnStop events to start and stop your robot skill
You can also send the image to EZ_B.Camera.SetCaptureImage();

Classes

Field: EZ_B.Classes.ServoAccelerationItem.Port

The affected port
Field: EZ_B.Classes.ServoAccelerationItem.Acceleration

The acceleration for the servo. -1 means ignore
Field: EZ_B.Classes.ServoVelocityItem.Port

The affected port
Field: EZ_B.Classes.ServoVelocityItem.Velocity

The velocity of the servo. -1 means ignore

Field: EZ_B.Classes.ServoSpeedItem.Port

The affected port
Field: EZ_B.Classes.ServoSpeedItem.Speed

The speed. -1 means ignore
Field: EZ_B.Classes.ServoPositionItem.Port

The affected port
Field: EZ_B.Classes.ServoPositionItem.Position

The position for the servo
Field: EZ_B.Classes.ServoPositionItem.Speed

The speed for the servo. -1 means ignore
Field: EZ_B.Classes.ServoPositionItem.Velocity

The velocity for the servo. -1 means ignore.
Field: EZ_B.Classes.ServoPositionItem.Acceleration

The acceleration for the servo. -1 means ignore.
Method: EZ_B.Classes.ServoPositionItem.Clone

Make a clone of this object
Field: EZ_B.Classes.GPSData.IsValid

Is the data valid (i.e. is there a satellite lock)
Field: EZ_B.Classes.GPSData.GPRMCRaw

The RMC Sentence from the gps
Field: EZ_B.Classes.GPSData.GPGSVRaw

The GSV Sentence from the gps
Field: EZ_B.Classes.GPSData.GPGSARaw

The GSA sentence from the gps
Field: EZ_B.Classes.GPSData.GPGGARaw

The GGA sentence from the gps
Field: EZ_B.Classes.GPSData.LastUpdated

The last timestamp of data
Field: EZ_B.Classes.GPSData.EarthLocationNS

Your position location of the earth (north/south)
Field: EZ_B.Classes.GPSData.EarthLocationEW

Your position location of the earth (east/west)
Field: EZ_B.Classes.GPSData.SatellitesUsed

Number of satellites used to obtain the data
Field: EZ_B.Classes.GPSData.SpeedKnots

The speed your robot is moving in knots
Field: EZ_B.Classes.GPSData.Course

Course over ground in degrees
Field: EZ_B.Classes.GPSData.Altitude

Your current altitude in meters

AudioEffects

Method: EZ_B.AudioEffects.IEffect.Init

Should be called on effect load, sample rate changes, and start of playback
Method: EZ_B.AudioEffects.IEffect.Slider

will be called when a slider value has been changed
Method: EZ_B.AudioEffects.IEffect.Block(*System.Int32*)

called before each block is processed
Method: EZ_B.AudioEffects.IEffect.Sample(*System.Single@, System.Single@*)

called for each sample

Extensions

Method: EZ_B.Extensions.MemoryStreamEx.Clear(*System.IO.MemoryStream*)

Reset the memory stream to position 0 and clear all of the data in the buffer. This allows reusing a memorystream object

Firmware

Type: EZ_B.Firmware.BoardImages

A strongly-typed resource class, for looking up localized strings, etc.
Field: EZ_B.Firmware.FirmwareCls.Capabilities

The list of capabilities supported by this firmware
Field: EZ_B.Firmware.FirmwareCls.FirmwareName

The friendly name of this firmware
Field: EZ_B.Firmware.FirmwareCls.Board

The board that this firmware supports
Field: EZ_B.Firmware.FirmwareCls.FirmwareId

The ID of this firmware (the content ID from synthiam.com)

Field: EZ_B.Firmware.FirmwareCls.Description

A friendly short description of this firmware
Method: EZ_B.Firmware.FirmwareCls.CapabilityRequired(*EZ_B.Firmware.CapabilityCls*)

Throw an exception if the specified capability is not supported by the loaded firmware
Method: EZ_B.Firmware.FirmwareCls.CapabilityRequired(*System.String*)

Throw an exception if the specified capability is not supported by the loaded firmware
Method: EZ_B.Firmware.FirmwareCls.IsCapabilitySupported(*EZ_B.Firmware.CapabilityCls*)

Is the specified capability supported by the loaded firmware?
Method: EZ_B.Firmware.FirmwareCls.IsCapabilitySupported(*System.String*)

Is the specified capability supported by the loaded firmware?
Method: EZ_B.Firmware.FirmwareCls.GetCapabilityDetails(*System.String*)

Get the capability details by the capability guid ID
Method: EZ_B.Firmware.FirmwareManager.GetFirmwareById(*System.UInt32*)

Retrieve the details of the firmware by the firmware id (content ID from synthiam.com)
Method: EZ_B.Firmware.FirmwareManager.LoadFirmwareLibraryXML(*System.String*)

Load the firmware XML file
Field: EZ_B.Firmware.XMLFirmwareSimulator.DefaultFirmware

The default firmware, also the first firmware in the Firmwares list

FormConnectionResources

Type: EZ_B.FormConnectionResources

A strongly-typed resource class, for looking up localized strings, etc.

UARTVideo

Type: EZ_B.UARTVideo

This connects to an EZ-B v4 Video Codec Camera over UART.
If you wish to connect to the camera over TCP, there is the EZBv4Video class that can be used instead.
Event: EZ_B.UARTVideo.OnImageIRReady

Event raised when an infrared image is ready. This image must be disposed after use.
Event: EZ_B.UARTVideo.OnImageReady

Event raised when the image is ready. This image must be disposed after use.
Event: EZ_B.UARTVideo.OnImageIRDataReady

Event raised when an infrared image is ready.
Event: EZ_B.UARTVideo.OnImageDataReady

Event raised when the image is ready.
Event: EZ_B.UARTVideo.OnStart

Event raised when the JPEGStream has started
Event: EZ_B.UARTVideo.OnStop

Event raised when the JPEGStream has stopped
Method: EZ_B.UARTVideo.Start(*EZ_B.EZB, System.String, System.Int32*)

** This method is deprecated and not maintained. Use the other Start()
Connect and begin receiving the camera stream
Method: EZ_B.UARTVideo.Start(*System.String, System.Int32*)

Connect and begin receiving the camera stream
Method: EZ_B.UARTVideo.Stop

Stop the camera from streaming and receiving frames

EZTaskScheduler

Type: EZ_B.EZTaskScheduler

This scheduler ensures tasks are executed on a background threads with queuing. Tasks are added to the queue and when completed, the next task runs.
Event: EZ_B.EZTaskScheduler.OnEventError

Raised if the event throws an exeception on the same thread as the event was executed.
*Note: If an error is raised in the task, the OnEventCompleted will not be raised so you'll need to check this event for an error of why it wasn't completed.
Event: EZ_B.EZTaskScheduler.OnEventCompleted

Raised when the event has completed on the same thread as the event executed.
*Note: If an error is thrown while the task is running, this won't be called because the OnEventError will be called instead.
Event: EZ_B.EZTaskScheduler.OnEventStart

Raised before the work event is started on the same thread as the work event will execute.
Event: EZ_B.EZTaskScheduler.OnEventToRun

The work event/task that will run for every instance.
Event: EZ_B.EZTaskScheduler.OnQueueCompleted

Raised when all events/tasks in the queue have completed executing. Executes on the same thread that the last task ran on.
Method: EZ_B.EZTaskScheduler.IsCancelRequested(*System.Int32*)

Has the specified taskid been cancelled or requested to cancel?
Method: EZ_B.EZTaskScheduler.ResetCancellation

If cancel was called, you can reset it here. It's not necessary but there may be specific circumstances where you need to
Method: EZ_B.EZTaskScheduler.CancelCurrentTask

Cancel the current running task and keeps processing any queued tasks
Method: EZ_B.EZTaskScheduler.StopAllTasks

Clear the task queue and stop current task
Method: EZ_B.EZTaskScheduler.ClearAllQeuedTasks

Clear all tasks in the queue, but do not stop the current running task.
Method: EZ_B.EZTaskScheduler.AddToQueue(*System.Object*)

Add item to the queue
This does not start the task scheduler. You will have to run Start() after adding items to the queue.
Ideally you should be using StartNew() unless you have a bunch of items to load into the queue before starting.
Method: EZ_B.EZTaskScheduler.StartNew

Add items to the queue and start the scheduler to begin processing them
*Note: This is the method that you should always be using unless you wish to prep items ahead of time, then use AddtoQueue() and Start(), respectively.
Method: EZ_B.EZTaskScheduler.StartNew(*System.Object*)

Add items to the queue and start the scheduler to begin processing them
*Note: This is the method that you should always be using unless you wish to prep items ahead of time, then use AddtoQueue() and Start(), respectively.
Method: EZ_B.EZTaskScheduler.Start

Start processing the tasks in the queue
Method: EZ_B.EZTaskScheduler.Abort

Aborts the task manager thread.
Never call this. Calling this will stop additional tasks to run because they're called within the thread.

ARDrone

Type: EZ_B.ARDrone.Commands.LedAnimationEnum

Indicates the LED animation to perform.
Field: EZ_B.ARDrone.Commands.VideoChannelEnum.Horizontal

Captured images are coming from the horizontal (forward) camera.
Field: EZ_B.ARDrone.Commands.VideoChannelEnum.Vertical

Captured images are coming from the vertical (downward) camera.
Field: EZ_B.ARDrone.Commands.VideoChannelEnum.VerticalInHorizontal

Captured images are coming from both the vertical and horizontal camera. The vertical image is shown in upper left corner.
Field: EZ_B.ARDrone.Commands.VideoChannelEnum.HorizontalInVertical

Captured images are coming from both the vertical and horizontal camera. The horizontal image is shown in upper left corner.
Field: EZ_B.ARDrone.Commands.VideoChannelEnum.Next

Captured images are coming the next videochannel determined by this enumeration.
Field: EZ_B.ARDrone.Commands.SwitchVideoChannel

This AT command is used to switch between different camera views.

Field: EZ_B.ARDrone.Commands.SetFlyingValue

This AT Command is used for take off/land and emergency reset.
Field: EZ_B.ARDrone.Commands.SetFlatTrim

This AT command sets a reference of the horizontal plane for the drone internal control system.
Field: EZ_B.ARDrone.Commands.SetConfiguration

This AT Command sets an configurable option on the drone.
Field: EZ_B.ARDrone.Commands.SetControlMode

This AT Command is used when communicating with the control communication channel.
Field: EZ_B.ARDrone.Commands.PlayLedAnimation

This AT Command makes the ARDrone animate its LED's according to a selectable pattern.
Field: EZ_B.ARDrone.Commands.SetProgressiveInputValues

This AT Command is used to provide the ARDrone with piloting instructions.
Field: EZ_B.ARDrone.Commands.SetTagDetection

This AT Command activates/deactivates the detection of coloured patterns.
Field: EZ_B.ARDrone.Commands.ResetCommunicationHub

This AT Command resets the internal ARDrone communication system.
Method: EZ_B.ARDrone.ARDrone.Connect(*EZ_B.ARDrone.ARDrone.ARDroneVersionEnum*)

Establish connection to drone
Method: EZ_B.ARDrone.ARDrone.Disconnect

Disconnect from the Drone
Method: EZ_B.ARDrone.ARDrone.StartVideo

Start receiving video from Drone.
Image can be obtained from OnImage event
Method: EZ_B.ARDrone.ARDrone.StopVideo

Stop receiving video from drone
Method: EZ_B.ARDrone.ARDrone.SendDefaultValues

This uploads default values to the drone for easy flying
Method: EZ_B.ARDrone.ARDrone.SetIsOutside(*System.Boolean*)

Set true if you are flying outside
Method: EZ_B.ARDrone.ARDrone.SetIsFlyingWithoutShell(*System.Boolean*)

Set to TRUE if you are flying with the outside shell
Method: EZ_B.ARDrone.ARDrone.SetWiFiNetworkName(*System.String*)

Set the WiFi network name for the AR Drone. Changes are applied on reboot
Method: EZ_B.ARDrone.ARDrone.SetYaw(*System.Single*)

Maximum yaw (spin) speed of the AR.Drone, in radians per second.
Recommanded values goes from (0.7) 40/s to (6.11) 350/s. Others values
may cause instability.
Default: 3.0
Method: EZ_B.ARDrone.ARDrone.SetVZMax(*System.Int32*)

Maximum vertical speed of the AR.Drone, in milimeters per second.
Recommanded values goes from 200 to 2000. Others values may cause instability.
Default: 1000
Method: EZ_B.ARDrone.ARDrone.SetEulerAngleMax(*System.Single*)

Set maximum bending angle for drone in radians for pitch and roll.
I.E. Maximum angle for going forward, back, left or right
This does not affect YAW (spin)
Floating point between 0 (0 deg) and 0.52 (32 deg)
Default: 0.25
Method: EZ_B.ARDrone.ARDrone.SetAltitudeMax(*System.Int32*)

Maximum drone altitude in millimeters.
Give an integer value between 500 and 5000 to prevent the drone from flying above this limit,
or set it to 10000 to let the drone fly as high as desired.
Default: 3000
Method: EZ_B.ARDrone.ARDrone.SetAltitudeMin(*System.Int32*)

Minimum drone altitude in millimeters.
Should be left to default value, for control stabilities issues
Default: 50
Method: EZ_B.ARDrone.ARDrone.SetProgressiveInputValues(*System.Single, System.Single, System.Single, System.Single*)

Move the drone. Values are between -1f and +1f
Method: EZ_B.ARDrone.ARDrone.NextVideoChannel

Cycle through the video channels. Go to next.
Method: EZ_B.ARDrone.ARDrone.Hover

Call this method to stop moving and hover in one place
Method: EZ_B.ARDrone.ARDrone.Land

Land the drone
Method: EZ_B.ARDrone.ARDrone.TakeOff

Take off/Start Engines
Method: EZ_B.ARDrone.ARDrone.Emergency

Emergency Stop the drone. Cuts power to motors
Method: EZ_B.ARDrone.ARDrone.SetFlatTrim

Must be called before take-off (start engines).
Must be called on a flat surface. This flattens the trim values for the surface.
Method: EZ_B.ARDrone.ARDrone.PlayLedAnimation(*EZ_B.ARDrone.Commands.LedAnimationEnum, System.Int32, System.Int32*)

Makes the ARDrone animate its LED's.
Field: EZ_B.ARDrone.VideoImage.PictureFormats.Cif

176px x 144px
Field: EZ_B.ARDrone.VideoImage.PictureFormats.Vga

320px x 240px

NeoPixelController8

Method: EZ_B.NeoPixelController8.SetColor(*EZ_B.EZB, EZ_B.Digital.DigitalPortEnum, System.Byte, System.Byte, System.Byte, System.Byte*)

Set the color of the first led on the bus on the specified output port on the controller
Method: EZ_B.NeoPixelController8.SetColor(*EZ_B.EZB, EZ_B.Digital.DigitalPortEnum, System.Byte, EZ_B.NeoPixelController8.ColorStruct[]*)

Set the color of leds on the specified output port on the controller. The first LED is the first color specified, second LED is the second color specified, etc...

RGB8x8Animator

Event: EZ_B.RGB8x8Animator.OnComplete

Event risen when movement is complete
Event: EZ_B.RGB8x8Animator.OnStartAction

Event risen when an action is started
Field: EZ_B.RGB8x8Animator.Name

Unique name for this auto position instance
Method: EZ_B.RGB8x8Animator.Stop

Stops the current movement. Blocks until stop is successful.
Method: EZ_B.RGB8x8Animator.ExecAction(*EZ_B.Classes.RGB8x8AnimatorAction*)

Execute the Action

HT16K33Animator

Event: EZ_B.HT16K33Animator.OnComplete

Event risen when movement is complete
Event: EZ_B.HT16K33Animator.OnStartAction

Event risen when an action is started
Field: EZ_B.HT16K33Animator.Name

Unique name for this auto position instance
Method: EZ_B.HT16K33Animator.Stop

Stops the current movement. Blocks until stop is successful.
Method: EZ_B.HT16K33Animator.ExecAction(*EZ_B.Classes.HT16K33AnimatorAction*)

Execute the Action

RGBAnimator

Event: EZ_B.RGBAnimator.OnComplete

Event risen when movement is complete
Event: EZ_B.RGBAnimator.OnStartAction

Event risen when an action is started
Event: EZ_B.RGBAnimator.OnStartFrame

Event risen when an action is started
Field: EZ_B.RGBAnimator.Name

Unique name for this auto position instance
Method: EZ_B.RGBAnimator.Stop

Stops the current movement. Blocks until stop is successful.

Method: EZ_B.RGBAnimator.ExecAction(*EZ_B.Classes.RGBAnimatorAction*)

Execute the Action

AVM

Field: EZ_B.AVM.CvAssociativeMemory32S.hAVM

A handle of AVM API
Method: EZ_B.AVM.CvAssociativeMemory32S.Finalize

Destructor
Method: EZ_B.AVM.CvAssociativeMemory32S.Create(*System.Drawing.Size, System.Int16, System.Int32, System.Boolean*)

Creating of associative memory

Notes: The parameter aKeyImgSize is depended on sequence 40, 80, 160, 320, 640, 1280... 2^n*10.
Use of this numbers for sizing gives more accuracy in recognition.
The parameter aLevelMax set a maximal level for associative tree.
If set to zero then a maximal level will be computed as optimal for image key size.
aTreeTotal - total number of independent associative trees.
aClustering - flag of using cluster tree
Method: EZ_B.AVM.CvAssociativeMemory32S.Destroy

Destroying of associative memory
Method: EZ_B.AVM.CvAssociativeMemory32S.SetActiveTree(*System.Int32*)

Set the active associative tree
Method: EZ_B.AVM.CvAssociativeMemory32S.ClearTreeData

Removing all data out from associative tree
Method: EZ_B.AVM.CvAssociativeMemory32S.Save(*System.String, System.Boolean*)

Saving of recognition data
Method: EZ_B.AVM.CvAssociativeMemory32S.Load(*System.String*)

Loading of recognition data
Method: EZ_B.AVM.CvAssociativeMemory32S.GetPackedDataSize

Get size of packed recognition data
Method: EZ_B.AVM.CvAssociativeMemory32S.WritePackedData

Writing a packed recognition data to memory
Method: EZ_B.AVM.CvAssociativeMemory32S.ReadPackedData(*System.Byte[]*)

Reading a packed recognition data from memory
Method: EZ_B.AVM.CvAssociativeMemory32S.OptimizeAssociativeTree

Optimization of associative tree
Method: EZ_B.AVM.CvAssociativeMemory32S.RestartTimeForOptimization

Restart time counter of optimization event
Method: EZ_B.AVM.CvAssociativeMemory32S.EstimateOpportunityForTraining(*System.Drawing.Rectangle*)

The estimation of an opportunity for training
Method: EZ_B.AVM.CvAssociativeMemory32S.SetImage(*AForge.Imaging.UnmanagedImage*)

Set an image for processing
Method: EZ_B.AVM.CvAssociativeMemory32S.SetImage(*System.Drawing.Bitmap@*)

Set an image for processing
Method: EZ_B.AVM.CvAssociativeMemory32S.PtrToArray(*System.Type, System.IntPtr, System.Int32*)

Convert structure pointer into array
Method: EZ_B.AVM.CvAssociativeMemory32S.ObjectRecognition

Object recognition (result will be returned as sequence)
Method: EZ_B.AVM.CvAssociativeMemory32S.ObjectTracking(*System.Boolean, System.Double, System.Double*)

Object recognition and tracking (result will be returned as sequence)
Method: EZ_B.AVM.CvAssociativeMemory32S.Read(*System.Drawing.Rectangle, System.Drawing.Rectangle@, System.IntPtr@, System.UInt64@, System.UInt64@, System.Double@, System.Boolean*)

Reading from associative memory cell (associative base)

Notes: aInterestArea - interest area for object searching;
apObjRect - rectangle where object found;
appData - pointer to the data of associative cell;
apIndex - unique index of associative base;
apHitCounter - number of hitting to associative base;
apSimilarity - similarity of the interest area to an object (0 ... 1);
aTotalSearch - flag of the total search until end level of associative memory.
Method: EZ_B.AVM.CvAssociativeMemory32S.Write(*System.Drawing.Rectangle, System.Int32, System.Boolean*)

Writing to associative memory cell

Method: EZ_B.AVM.CvAssociativeMemory32S.GetTotalABases

Get total number of associative bases
Method: EZ_B.AVM.CvAssociativeMemory32S.GetTotalLevels

Get total number of memory levels
Method: EZ_B.AVM.CvAssociativeMemory32S.GetCurIndex

Get current index of associative base
Method: EZ_B.AVM.CvAssociativeMemory32S.GetWrRdCounter

Get read-write counter
Method: EZ_B.AVM.CvAssociativeMemory32S.GetKeyImageSize

Get size of key image of associative memory
Method: EZ_B.AVM.CvAssociativeMemory32S.GetBaseKeySize

Get base key size
Method: EZ_B.AVM.CvAssociativeMemory32S.SetParam(*EZ_B.AVM.CvAM_ParamType, System.Double*)

Set the value of parameter
Field: EZ_B.AVM.avmInvoke.AVM_LIBRARY

The file name of the AVM library
Method: EZ_B.AVM.avmInvoke.avmOpen(*System.Int16*)

Open AVM API
Method: EZ_B.AVM.avmInvoke.avmClose(*EZ_B.AVM.avmHandle*)

Close AVM API
Method: EZ_B.AVM.avmInvoke.avmCreate(*EZ_B.AVM.avmHandle, System.Drawing.Size, System.Int16, System.Int32, System.Boolean*)

Creating of associative memory
Method: EZ_B.AVM.avmInvoke.avmDestroy(*EZ_B.AVM.avmHandle*)

Destroying of associative memory
Method: EZ_B.AVM.avmInvoke.avmSetActiveTree(*EZ_B.AVM.avmHandle, System.Int32*)

Set the active associative tree
Method: EZ_B.AVM.avmInvoke.avmClearTreeData(*EZ_B.AVM.avmHandle*)

Removing all data out from associative tree
Method: EZ_B.AVM.avmInvoke.avmSave(*EZ_B.AVM.avmHandle, System.Char[], System.Boolean*)

Saving of recognition data
Method: EZ_B.AVM.avmInvoke.avmLoad(*EZ_B.AVM.avmHandle, System.Char[]*)

Loading of recognition data
Method: EZ_B.AVM.avmInvoke.avmGetPackedDataSize(*EZ_B.AVM.avmHandle*)

Get size of packed recognition data
Method: EZ_B.AVM.avmInvoke.avmWritePackedData(*EZ_B.AVM.avmHandle, System.IntPtr*)

Writing a packed recognition data to memory
Method: EZ_B.AVM.avmInvoke.avmReadPackedData(*EZ_B.AVM.avmHandle, System.IntPtr*)

Reading a packed recognition data from memory
Method: EZ_B.AVM.avmInvoke.avmOptimizeAssociativeTree(*EZ_B.AVM.avmHandle*)

Optimization of associative tree
Method: EZ_B.AVM.avmInvoke.avmRestartTimeForOptimization(*EZ_B.AVM.avmHandle*)

Restart time counter of optimization event
Method: EZ_B.AVM.avmInvoke.avmEstimateOpportunityForTraining(*EZ_B.AVM.avmHandle, System.Drawing.Rectangle*)

The estimation of an opportunity for training
Method: EZ_B.AVM.avmInvoke.avmSetImageAsArray(*EZ_B.AVM.avmHandle, System.Drawing.Size, System.IntPtr, System.IntPtr*)

Set an image for processing where image is presented as 2D pixel (byte) array
Method: EZ_B.AVM.avmInvoke.avmObjectRecognition_sq_32S(*EZ_B.AVM.avmHandle, System.Int32@*)

Object recognition (result will be returned as sequence)
Method: EZ_B.AVM.avmInvoke.avmObjectTracking_sq_32S(*EZ_B.AVM.avmHandle, System.Int32@, System.Boolean, System.Double, System.Double*)

Object recognition and tracking (result will be returned as sequence)
Method: EZ_B.AVM.avmInvoke.avmRead_32S(*EZ_B.AVM.avmHandle, System.Drawing.Rectangle, System.Drawing.Rectangle@, System.IntPtr@, System.UInt64@, System.UInt64@, System.Double@, System.Boolean*)

Reading from associative memory cell (associative base)
Method: EZ_B.AVM.avmInvoke.avmWrite_32S(*EZ_B.AVM.avmHandle, System.Drawing.Rectangle, System.Int32@, System.Boolean*)

Writing to associative memory cell
Method: EZ_B.AVM.avmInvoke.avmGetTotalABases(*EZ_B.AVM.avmHandle*)

Get total number of associative bases
Method: EZ_B.AVM.avmInvoke.avmGetTotalLevels(*EZ_B.AVM.avmHandle*)

Get total number of memory levels
Method: EZ_B.AVM.avmInvoke.avmGetCurIndex(*EZ_B.AVM.avmHandle*)

Get current index of associative base
Method: EZ_B.AVM.avmInvoke.avmGetWrRdCounter(*EZ_B.AVM.avmHandle*)

Get read-write counter
Method: EZ_B.AVM.avmInvoke.avmGetKeyImageSize(*EZ_B.AVM.avmHandle*)

Get size of key image of associative memory
Method: EZ_B.AVM.avmInvoke.avmGetBaseKeySize(*EZ_B.AVM.avmHandle*)

Get base key size
Method: EZ_B.AVM.avmInvoke.avmSetParam(*EZ_B.AVM.avmHandle, EZ_B.AVM.CvAM_ParamType, System.Double*)

Set the value of parameter
Type: EZ_B.AVM.avmHandle

Wrapper for handle of AVM API
Type: EZ_B.AVM.CvAM_ParamType

Definition of parameter types
Type: EZ_B.AVM.CvAM_State

Definition of the state constants for recognition function
Type: EZ_B.AVM.CvTrcInfo

Data structure of tracking

Field: EZ_B.AVM.CvTrcInfo.cAM_TrcLen

Length of tracking
Field: EZ_B.AVM.CvTrcInfo.Pnt

Trajectory points
Field: EZ_B.AVM.CvTrcInfo.Age

Age of tracking
Type: EZ_B.AVM.CvObjDsr32S

Definition of object descriptor for type "long"
Field: EZ_B.AVM.CvObjDsr32S.State

State of recognition function
Field: EZ_B.AVM.CvObjDsr32S.ObjRect

Rectangle where object found
Field: EZ_B.AVM.CvObjDsr32S.Similarity

Similarity of the interest area to an object (0 ... 1)
Field: EZ_B.AVM.CvObjDsr32S.Data

Associated data
Field: EZ_B.AVM.CvObjDsr32S.Trj

Trajectory
Type: EZ_B.AVM.TrainedObjectsContainer

Recognition data structure

EZBv4Manager

Method: EZ_B.EZBv4Manager.SetLipoBatteryProtection(*System.Boolean, System.Decimal*)

Disable or Enable the battery monitor for the EZ-B v4. If the battery monitor is disabled, the EZ-B will continue to operate I/O if the voltage is low.
You can also adjust the lowest voltage value to one decimal place.
Method: EZ_B.EZBv4Manager.SetLipoBatteryLowestVoltage(*System.Decimal*)

Sets the lowest voltage that the EZ-B will operate with for the battery monitor. This is useful to Lipo batteries.
This feature is enabled by default on the EZ-B v4.
Method: EZ_B.EZBv4Manager.SetLipoBatteryProtectionState(*System.Boolean*)

Disable or Enable the battery monitor for the EZ-B v4. If the battery monitor is disabled, the EZ-B will continue to operate I/O if the voltage is low.
Method: EZ_B.EZBv4Manager.GetCPUTemperature

Returns the cpu core temperature in degrees celcuis
Method: EZ_B.EZBv4Manager.GetBatteryVoltage

Returns the battery voltage

HT16K33

Field: EZ_B.HT16K33.I2C_ADDRESS

Default I2C Address of the HT16K33 Module (0x70)
Field: EZ_B.HT16K33.BRIGHTNESS_MAX

The maximum brightness that can be sent to the LED (15)
Field: EZ_B.HT16K33.BRIGHTNESS_MIN

The minimium brightness that can be sent to the LED (0)
Method: EZ_B.HT16K33.Init

Initialize the HT16K33 by enabling the oscillator and setting the brightness to 15
Method: EZ_B.HT16K33.SetAllStatus(*System.Boolean*)

Sets all of the LED's to the specific color.
Method: EZ_B.HT16K33.SetLED(*System.Int32, System.Int32, System.Boolean*)

Set the LED status in the array
*Note: This will not actually change the physical LED. You must call Update() to update the array
Method: EZ_B.HT16K33.GetLED(*System.Int32, System.Int32*)

Return the status of the LED in the array
Method: EZ_B.HT16K33.UpdateLEDs(*System.Boolean[0:, 0:]*)

Update the LEDs with the current matrix. Also sets the current matrix to this value
Method: EZ_B.HT16K33.UpdateLEDs

Update the LEDs with the current matrix

RGB8x8

Field: EZ_B.RGB8x8.I2C_ADDRESS

Default I2C Address of the RGB 8x8 Module (0xa2)
Field: EZ_B.RGB8x8.INDEX_MAX

The number of RGB LEDs is referenced by the index
Method: EZ_B.RGB8x8.ChangeI2CAddress(*System.Byte, System.Byte*)

Change the I2C address of the device.
Send 0x58, New Address
Method: EZ_B.RGB8x8.ClearDisplay

Clear the display
Method: EZ_B.RGB8x8.ClearDisplay(*System.Byte*)

Clear the display
Method: EZ_B.RGB8x8.ClearDisplay(*System.Boolean, System.Boolean, System.Boolean*)

Clear the display
Method: EZ_B.RGB8x8.ClearDisplay(*System.Byte, System.Boolean, System.Boolean, System.Boolean*)

Clear the display
Method: EZ_B.RGB8x8.SetAllColor(*System.Boolean, System.Boolean, System.Boolean*)

Sets all of the LED's to the specific color.
Method: EZ_B.RGB8x8.SetAllColor(*System.Boolean, System.Boolean, System.Boolean, System.Byte*)

Sets all of the LED's to the specific color.
Method: EZ_B.RGB8x8.SetColor(*System.Int32, System.Boolean, System.Boolean, System.Boolean*)

Sets the color of the specified index.
Method: EZ_B.RGB8x8.SetColor(*System.Int32, System.Boolean, System.Boolean, System.Boolean, System.Byte*)

Sets the color of the specified index.
Method: EZ_B.RGB8x8.SetColor(*System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean*)

Sets the color of the specified index.
Method: EZ_B.RGB8x8.SetColor(*System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean, System.Byte*)

Sets the color of the specified index.
Method: EZ_B.RGB8x8.SetColors(*EZ_B.RGB8x8.RGBDef[]*)

Sets the LED's to the specific color.
The RGBDef must be 64 items or less.
Method: EZ_B.RGB8x8.SetColors(*EZ_B.RGB8x8.RGBDef[], System.Byte*)

Sets the LED's to the specific color.
The RGBDef must be 64 items or less.
Method: EZ_B.RGB8x8.CanvasRectangle(*System.Int32, System.Int32, System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean*)

Draw a rectangle on the display
Method: EZ_B.RGB8x8.CanvasLine(*System.Int32, System.Int32, System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean*)

Draw a line on the display
Method: EZ_B.RGB8x8.CanvasClear

Clear the canvas
Method: EZ_B.RGB8x8.CanvasClear(*System.Boolean, System.Boolean, System.Boolean*)

Clear the canvas with the specified color
Method: EZ_B.RGB8x8.CanvasUpdate

Send the current canvas to the display
Method: EZ_B.RGB8x8.CanvasUpdate(*System.Byte*)

Send the current canvas to the display
Method: EZ_B.RGB8x8.CanvasSetPixel(*System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean*)

Set the pixel color on the canvas
Method: EZ_B.RGB8x8.CanvasSetPixel(*System.Int32, System.Int32, System.Boolean, System.Boolean, System.Boolean, System.Byte*)

Set the pixel color on the canvas
Method: EZ_B.RGB8x8.SetColors(*EZ_B.RGB8x8.CanvasColorStruct[0:, 0:], System.Byte*)

Sets the LED's to the specific color.
The RGBDef must be 64 items or less.

RGBEyes

Field: EZ_B.RGBEyes.I2C_ADDRESS

Default I2C Address of the RGB Eyes Module (0xa0)
Field: EZ_B.RGBEyes.BRIGHTNESS_MAX

The maximum brightness that can be sent to the RGB LED (7)
Field: EZ_B.RGBEyes.BRIGHTNESS_MIN

The minimium brightness that can be sent to the RGB LED (0)

Field: EZ_B.RGBEyes.INDEX_MAX

The number of RGB LEDs is referenced by the index
Method: EZ_B.RGBEyes.ChangeI2CAddress(*System.Byte*)

Change the I2C address of the device. Will send the command to the default address.
Method: EZ_B.RGBEyes.ChangeI2CAddress(*System.Byte, System.Byte*)

Change the I2C address of the device.
Method: EZ_B.RGBEyes.SetAllColor(*System.Byte, System.Byte, System.Byte*)

Set all of the LED's to the specific color. Sends the command to the default I2C address
Method: EZ_B.RGBEyes.SetAllColor(*System.Byte, System.Byte, System.Byte, System.Byte*)

Sets all of the LED's to the specific color.
Method: EZ_B.RGBEyes.SetColor(*System.Byte, System.Byte, System.Byte, System.Byte*)

Set the color of the specified index. Sends the command to the default I2C address
Method: EZ_B.RGBEyes.SetColor(*System.Byte, System.Byte, System.Byte, System.Byte, System.Byte*)

Sets the color of the specified index.
Method: EZ_B.RGBEyes.SetColor(*System.Byte[], System.Byte, System.Byte, System.Byte*)

Sets the color of the specified indexes within the array. Sends the command to the default I2C Addres
Method: EZ_B.RGBEyes.SetColor(*System.Byte, System.Byte[], System.Byte, System.Byte, System.Byte*)

Sets the color of the specified indexes within the array.

EZBv4Video

Type: EZ_B.EZBv4Video

This connects to an EZ-B v4 Video Codec Camera over TCP.
If you wish to connect to the camera over UART, there is the UARTVideo class that can be used instead.
Event: EZ_B.EZBv4Video.OnImageIRReady

Event raised when an infrared image is ready. This image must be disposed after use.
Event: EZ_B.EZBv4Video.OnImageReady

Event raised when the image is ready. This image must be disposed after use.
Event: EZ_B.EZBv4Video.OnImageIRDataReady

Event raised when an infrared image is ready.
Event: EZ_B.EZBv4Video.OnImageDataReady

Event raised when the image is ready.
Event: EZ_B.EZBv4Video.OnStart

Event raised when the JPEGStream has started
Event: EZ_B.EZBv4Video.OnStop

Event raised when the JPEGStream has stopped
Event: EZ_B.EZBv4Video.OnDebugLog

Event raised with any debug information
Method: EZ_B.EZBv4Video.Start(*EZ_B.EZB, System.String, System.Int32*)

* This method Deprecated. Use the other Start()
Connect and begin receiving the camera stream
Method: EZ_B.EZBv4Video.Start(*System.String, System.Int32*)

Connect and begin receiving the camera stream
Method: EZ_B.EZBv4Video.Stop

Stop the camera from streaming and receiving frames

MusicSynth

Field: EZ_B.MusicSynth.random

Random provider for noise generator

EZBv4Sound

Type: EZ_B.EZBv4Sound

The EZ-B v4 Sound codec stream driver.
You can pass an audio stream to have it play out of the EZ-B v4 that supports the audio streaming capability.
There are readonly values that you can reference which specifies the bit rate that the audio stream expects.
It also expects the audio to be 8 bit and mono.
Field: EZ_B.EZBv4Sound.RECOMMENDED_PACKET_SIZE

The recommended size of the the audio packets
Field: EZ_B.EZBv4Sound.RECOMMENDED_PREBUFFER_SIZE

The recommended size of the prebuffer before playing the audio
Field: EZ_B.EZBv4Sound.AUDIO_SAMPLE_BITRATE

The sample rate at which the data is played back on the EZ-B
Field: EZ_B.EZBv4Sound.PACKET_SIZE

The size of each packet which is transmitted over the wire to the EZ-B.
Field: EZ_B.EZBv4Sound.PREBUFFER_SIZE

The ammount of data to prebuffer to the EZ-B before playing the audio. The EZ-B has a 50k buffer, so this value cannot be any higher than that.
Type: EZ_B.EZBv4Sound.OnAudioDataChangedHandler

Event exceuted when new data is being sent to the EZ-B
Type: EZ_B.EZBv4Sound.OnVolumeChangedHandler

Event executed when the volume value has changed
Type: EZ_B.EZBv4Sound.OnStopPlayingHandler

Event executed when the audio has stopped playing
Type: EZ_B.EZBv4Sound.OnStartPlayingHandler

Event executed when the audio has begun playing
Type: EZ_B.EZBv4Sound.OnClippingStatusHandler

Event executed when the audio level is clipping. This means the volume value is amplifying the audio past the limits
Type: EZ_B.EZBv4Sound.OnProgressHandler

Event executed with the playing progress by sample position. The resolution of this event can be specified with Play method.
In summary, you set the Play Positions by the sample index and this event will execute when the playing reaches that particular sample point.
If you simply want an update of the curernt play time, use the OnPlayTime event.
Type: EZ_B.EZBv4Sound.OnPlayTimeHandler

Event executed with the playing progress by sample position with 1000ms resolution.
Method: EZ_B.EZBv4Sound.PlayDataWait(*System.Byte[]*)

Play the Audio Data out of the EZ-B.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
Method: EZ_B.EZBv4Sound.PlayData(*System.Byte[]*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate

Method: EZ_B.EZBv4Sound.PlayData(*System.Byte[], System.Decimal*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
Method: EZ_B.EZBv4Sound.PlayData(*System.Byte[], System.Decimal, System.Int32[]*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
Method: EZ_B.EZBv4Sound.PlayDataWait(*System.Byte[], System.Decimal*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream, System.Int32[]*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream, System.Int32[], System.Int32*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayDataWait(*System.IO.Stream*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream, System.Decimal*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream, System.Decimal, System.Int32[]*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayData(*System.IO.Stream, System.Byte[], System.Decimal, System.Int32[], System.Int32*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.PlayDataWait(*System.IO.Stream, System.Byte[], System.Decimal, System.Int32[], System.Int32*)

Stream raw audio data to the EZ-B v4's speakers.
0 is silent, 100 is normal, 200 is 2x gain, 300 is 3x gain, etc.
The audio must be RAW 8 Bit at 18 KHZ Sample Rate
*Note: You must dispose of the memory stream yourself after calling this
Method: EZ_B.EZBv4Sound.Stop

Stop the audio which is being played
Method: EZ_B.EZBv4Sound.NormalizeAbsolute(*System.Byte[]*)

Normalizing the audio stream. This expects the audio stream to match the required stream parameters of this class.
Mono 8-Bit PCM
Method: EZ_B.EZBv4Sound.NormalizeDynamic(*System.Byte[]*)

Normalizing the audio stream. This expects the audio stream to match the required stream parameters of this class.
Mono 8-Bit PCM

Joystick

Field: EZ_B.Joystick.JoystickDevice.ID

Joystick ID, [0..15].
Type: EZ_B.Joystick.ButtonEnum

Flags enumeration of joystick buttons.
Field: EZ_B.Joystick.ButtonEnum.Button1

1st button.
Field: EZ_B.Joystick.ButtonEnum.Button2

2nd button.
Field: EZ_B.Joystick.ButtonEnum.Button3

3rd button.
Field: EZ_B.Joystick.ButtonEnum.Button4

4th button.
Field: EZ_B.Joystick.ButtonEnum.Button5

5th button.
Field: EZ_B.Joystick.ButtonEnum.Button6

6th button.
Field: EZ_B.Joystick.ButtonEnum.Button7

7th button.
Field: EZ_B.Joystick.ButtonEnum.Button8

8th button.
Field: EZ_B.Joystick.ButtonEnum.Button9

9th button.
Field: EZ_B.Joystick.ButtonEnum.Button10

10th button.
Field: EZ_B.Joystick.ButtonEnum.Button11

11th button.
Field: EZ_B.Joystick.ButtonEnum.Button12

12th button.
Field: EZ_B.Joystick.ButtonEnum.Button13

13th button.
Field: EZ_B.Joystick.ButtonEnum.Button14

14th button.
Field: EZ_B.Joystick.ButtonEnum.Button15

15th button.
Field: EZ_B.Joystick.ButtonEnum.Button16

16th button.
Method: EZ_B.Joystick.JoystickStatus.IsButtonPressed(*EZ_B.Joystick.ButtonEnum*)

Check if certain button (or combination of buttons) is pressed.
Type: EZ_B.Joystick.Joystick.OnJoystickMoveHandler

Event risen when for joystick action
Event: EZ_B.Joystick.Joystick.OnControllerAction

Event risen when for joystick action
Method: EZ_B.Joystick.Joystick.GetCurrentStatus

Get Current Joystick Status
Method: EZ_B.Joystick.Joystick.RefreshState

Call this to refresh the state of the joystick
You may call this in a timer every 100 or 200ms
Method: EZ_B.Joystick.Joystick.StartEventWatcher

Start the watcher for joystick event notifications
Method: EZ_B.Joystick.Joystick.StopEventWatcher

Stop the watcher for joystick evet notifications
Method: EZ_B.Joystick.Joystick.ButtonPressed(*System.Int32*)

Returns True if a button is pressed, False if it is not
Method: EZ_B.Joystick.Joystick.ButtonStateChanged(*System.Int32*)

Returns true if the state of the button has changed since last update
Method: EZ_B.Joystick.Joystick.AxisXStateChanged

Returns true if the X Axis location has changed since last update
Method: EZ_B.Joystick.Joystick.AxisYStateChanged

Returns True if the Y Axis has changed since last update
Method: EZ_B.Joystick.Joystick.AxisZStateChanged

Returns True if the Z Axis has changed since last update.
Z Axis is the second analog joystick (if equiped)
Method: EZ_B.Joystick.Joystick.AxisRzStateChanged

Returns True if the Rz Axis has changed since last update.
Rz Axis is the second analog joystick (if equiped)
Method: EZ_B.Joystick.Joystick.AxisUStateChanged

Returns True if the U Axis has changed since last update.
U Axis is the second analog joystick (if equiped)
Method: EZ_B.Joystick.Joystick.AxisVStateChanged

Returns True if the V Axis has changed since last update.
V Axis is the second analog joystick (if equiped)

Camera

Type: EZ_B.Camera

Class that it used to connect to many camera devices (usb, ez-b v4 uart, ez-b v4 TCP, etc.)
This class also holds the detection methods and allows a custom detection method to be used with the provided overrides.
You may also record video to a file.
This is essentially one massive class that provides a ton of function for camera recognition.
Field: EZ_B.Camera._originalUnmanagedBitmap

This is the unmanaged image of _OriginalManagedBitmap.
It is the resized and corrected pixel format image because the image from the capture device may
not match the output dimensions and pixel format.
Field: EZ_B.Camera._workerUnmanagedBitmap

Image that is used as a worker to the display. We draw on this bitmap with the tracking details and grid lines
Each tracking type will draw it's squares on this bitmap and then we copy this to the output bitmap.
A worker is required because if we modify the bitmap memory then windows might invalidate the control while
we're in the middle of rendering the output bitmap. So this results in a flicker of varying states of the image.
So we work with the bitmap here, and copy it to the output bitmap when we're done with it
Field: EZ_B.Camera._outputUnmanagedBitmap

This is the bitmap data that we use to update the preview windows with.
This contains tracking data.
Type: EZ_B.Camera.VideoCodec

Type of Video File to save
Field: EZ_B.Camera.CAMERA_NAME_EZBv4

The default address for the EZ-B v4. The EZ-B TCP camera will always start with an EZB://(address)
If you're connecting to an EZ-B v4 camera via UART, specify the address as COM1 or COM2, COMx etc...
Field: EZ_B.Camera.CAMERA_NAME_CUSTOM

Select when a robot skill is sending a video stream into the camera device using SetCaptureImage()
Event: EZ_B.Camera.OnInitCustomTracking

Event raised for custom tracking initializers.
If you have a custom tracking, use this to initialize your tracking engine if needed.
This happens when the camera is started.
Event: EZ_B.Camera.OnDeinitCustomTracking

Event raised for a custome tracking type when it has been uninitialized.
This happens when the camera is stopped.
Event: EZ_B.Camera.OnStart

Event raised when the Image capture has started
Event: EZ_B.Camera.OnStop

Event raised when the Image capture has stopped
Event: EZ_B.Camera.OnNewFrame

Event risen when a new video frame is received.
The actual bitmap is not drawn until after this event is raised.
In this event, you can use GetOutputBitmap to draw your tracking information and GetOriginalBitmap for detecting the tracking
This event is risen BEFORE OnNewFrameProcessed, which means there are no grid lines drawn on the bitmap at this time.
This is the event where you will use object detection. And you can draw on the output bitmap with the detected objects
Event: EZ_B.Camera.OnNewFrameProcessed

Event risen when a new video frame has been processed after OnNewFrame tracking.
If you're wanting to display the video to the UCCameraCanvas control, use it from here.
This event is raised after the grid lines are drawn, which is also after OnNewFrame.
This event is raised after OnNewFrame.
Generally, OnNewFrame is where the tracking types are executed, which means the OutputBitmaps will be modified with detected objects
Field: EZ_B.Camera.DisableDetectedRectangleDisplay

If true, the rectangle around the detect object is not displayed. The objectLocation is still returned, but there is no visual identifier of where the object was detected.
Field: EZ_B.Camera.AVIIntroText

Set the title for the intro text for video recording
Field: EZ_B.Camera.AVIIntroBGColor

Set the intro title background color for video recording
Field: EZ_B.Camera.AVIIntroFGColor

Set the intro title foreground color for video recording
Field: EZ_B.Camera.AVIShowIntro

Enable/Disable intro title in video recording

Field: EZ_B.Camera.AVIIntroLength

Length of time the intro title will display for video recording
Field: EZ_B.Camera.AVIPauseRecording

Set to true to pause the current recording
Field: EZ_B.Camera.GridTransparency

Set the transparency of the camera grid. 0 is transparent, 255 is solid.
Field: EZ_B.Camera.CameraBasicColorDetection

Camera Basic Color Detection
Field: EZ_B.Camera.CameraFaceDetection

Camera Face Detection
Field: EZ_B.Camera.CameraMotionDetection

Camera Motion Detection
Field: EZ_B.Camera.CameraCustomColorDetection

Camera Custom Color Detection
Field: EZ_B.Camera.CameraCustomYCbCrColorDetection

Camera Custom Color Detection
Field: EZ_B.Camera.CameraGlyphDetection

Camera Glyph Detection
Field: EZ_B.Camera.CameraQRCodeDetection

Camera QR Code Detection
Field: EZ_B.Camera.CameraCustomHaarDetection

Camera Custom Haar Detection
Field: EZ_B.Camera.CameraAVMObjectDetection

Camera Custom Object Detection
Field: EZ_B.Camera.QuadLeftX

The X cordinate of the Left quadrant
Field: EZ_B.Camera.QuadRightX

The X cordinate of the Right quadrant
Field: EZ_B.Camera.QuadTopY

The Y cordinate of the Top quadrant
Field: EZ_B.Camera.QuadBottomY

The Y cordinate of the Bottom quadrant
Field: EZ_B.Camera.Brightness

Set the brightness correction of the image.
Range is between -255 and +255.
Positive values increase brightness.
Negative values decrease brightness.
Field: EZ_B.Camera.Saturation

Set the saturation between -1f and 1f
Field: EZ_B.Camera.SharpenImage

Set the sharpening enhancement
Field: EZ_B.Camera.Contrast

Set the contrast correction of the image.
Range is between -255 and +255.
Positive values increase contrast.
Negative values decrease contrast.
Field: EZ_B.Camera.SnapshotQuality

The quality for the snapshot jpeg file
Field: EZ_B.Camera.RotateType

The type of rotation for the video stream
Method: EZ_B.Camera.RegisterOriginalPreview(*EZ_B.UCCameraCanvas*)

Register the canvas object for the video preview of the original camera image.
Ensure to unregister this before you dispose of your camera canvas object
Method: EZ_B.Camera.UnregisterOriginalPreview(*EZ_B.UCCameraCanvas*)

Unregister the canvas object from the video preview
Method: EZ_B.Camera.RegisterOutputPreview(*EZ_B.UCCameraCanvas*)

Register the canvas object for the video preview of the output camera image that contains tracking information
Ensure to unregister this before you dispose of your camera canvas object
Method: EZ_B.Camera.UnregisterOutputPreview(*EZ_B.UCCameraCanvas*)

Unregister the canvas object from the video preview
Method: EZ_B.Camera.CopyManagedBitmapToUnmanaged(*System.Drawing.Bitmap, AForge.Imaging.UnmanagedImage*)

Copy the data from a managed bitmap to an unmanaged image. This does not create the destination image.
This checks the dimensions and pixel format are the same
Your dst image must have already been created with the same dimemsions and pixel format as the src
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.CopyManagedBitmapToUnmanagedUnsafe(*System.Drawing.Bitmap, AForge.Imaging.UnmanagedImage*)

Copy the data from a managed bitmap to an unmanaged image. This does not create the destination image.
This does NOT check the dimensions and pixel format are the same (for performance)
Your dst image must have already been created with the same dimemsions and pixel format as the src
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.CopyUnmanagedImageToBitmap(*AForge.Imaging.UnmanagedImage, System.Drawing.Bitmap*)

Copy the data from an unmanaged image to a bitmap. This does not create a bitmap.
This checks the dimensions and pixel format are the same
Your dst bitmap must have already been created with the same dimemsions and pixel format as the src
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.CopyUnmanagedImageToBitmapUnsafe(*AForge.Imaging.UnmanagedImage, System.Drawing.Bitmap*)

Copy the data from an unmanaged image to a bitmap. This does not create a bitmap.
This does NOT check if your dimensions and pixel format is the same (for performance)
Your dst bitmap must have already been created with the same dimemsions and pixel format as the src
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.CopyBitmapMemory(*System.Drawing.Bitmap, System.Drawing.Bitmap*)

Copies the memory of one bitmap to another.
This checks if the dimensions and pixel type are the same
The source and destination bitmaps must have the same size and pixel format, otherwise this won't work and may produce unmanaged code errors
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.CopyBitmapMemoryUnsafe(*System.Drawing.Bitmap, System.Drawing.Bitmap*)

Copies the memory of one bitmap to another.
This does NOT check if the dimensions and pixel type are the same (for performance)
The source and destination bitmaps must have the same size and pixel format, otherwise this won't work and may produce unmanaged code errors
This is a whole memory copy, which is a duplicate. The images will not share the same memory space
Method: EZ_B.Camera.StartCamera(*EZ_B.ValuePair, System.Int32, System.Int32*)

Initialize a camera in preperation for object detection.
The VideoCaptureDevice must be a ValuePair, where the Key is what contains either the Moniker String for the Video Driver.
Method: EZ_B.Camera.StopCamera

Disable camera, if enabled. Free the scanning resources.
Method: EZ_B.Camera.SetCaptureImage(*System.Drawing.Bitmap*)

Manually set a bitmap as the capture image to be processed. You can manually set a bitmap instead of using a hardware device.
You may also add your own hardware device supporting by setting the bitmap within this function.
This method will not dispose your bitmap, so it's up to you to dispose once it has completed.
Method: EZ_B.Camera.AVIStartRecording(*System.String, EZ_B.Camera.VideoCodec*)

Save the incoming video stream to a video file
Method: EZ_B.Camera.AVIStartRecording(*System.String*)

Save the incoming video stream to a video file
Method: EZ_B.Camera.AVIStopRecording

Stop the AVI video recording
Method: EZ_B.Camera.SaveImageAsJPEG(*System.String, System.Byte*)

Save the next frame to the specified file as a jpg
Method: EZ_B.Camera.SaveImageAsJPEG(*System.String*)

Save the next frame to the specified file as a jpg
Method: EZ_B.Camera.GetHorizontalLocation(*System.Int32*)

This returns the quadarant that the xPos value is.
Because the user can specify the quadarant values, you can use this to identify what quadarant the position is in.
Method: EZ_B.Camera.GetVerticalLocation(*System.Int32*)

This returns the quadarant that the yPos value is.
Because the user can specify the quadarant values, you can use this to identify what quadarant the position is in.

BlinkM

Method: EZ_B.BlinkM.StopScript(*System.Byte*)

Stop Script with 7 bit address
Method: EZ_B.BlinkM.ChangeToColor(*System.Byte, System.Byte, System.Byte, System.Byte*)

Change the BlinkM to the specified Red/Green/Blue color
Method: EZ_B.BlinkM.FadeToColor(*System.Byte, System.Byte, System.Byte, System.Byte*)

Fade the BlinkM to the specified Red/Green/Blue color

Method: EZ_B.BlinkM.GetCurrentColor(*System.Byte*)

Returns the current colors on the BlinkM

BV4615

Method: EZ_B.BV4615.GetFirmware

Return the firmware of the device
Method: EZ_B.BV4615.GetData

Returns a response object with the data from the buffer

ConfigurationManager

Method: EZ_B.ConfigurationManager.SetConfiguration(*EZ_B.ConfigurationManager.ConfigurationEnum, System.Byte*)

Write configuration data to the EZ-B v3
Method: EZ_B.ConfigurationManager.SetConfiguration(*EZ_B.ConfigurationManager.StringConfigurationEnum, System.Byte, System.String*)

Write configuration data to the EZ-B v3
Method: EZ_B.ConfigurationManager.GetConfiguration(*EZ_B.ConfigurationManager.ConfigurationEnum*)

Return data at a memory location from the EZ-B v3 eeprom
Method: EZ_B.ConfigurationManager.GetConfiguration(*EZ_B.ConfigurationManager.StringConfigurationEnum, System.Byte*)

Return data at the memory location and the length of bytes in the EZ-B v3 eeprom

JPEGStream

Event: EZ_B.JPEGStream.OnImageReady

Event raised when the image is ready. This image must be disposed after use.
Event: EZ_B.JPEGStream.OnStart

Event raised when the JPEGStream has started
Event: EZ_B.JPEGStream.OnStop

Event raised when the JPEGStream has stopped
Method: EZ_B.JPEGStream.SetURL(*System.String*)

Set a new URL for streaming
Method: EZ_B.JPEGStream.Start(*System.Int32*)

Start the JPEG Streamer with the specified FPS.
The FPS is dependent on the internet connection speed.
We usually use an FPS of 10.
Method: EZ_B.JPEGStream.Start(*System.String, System.Int32*)

Start the JPEG Streamer with the specified URL and FPS.
The FPS is dependent on the internet connection speed.
We usually use an FPS of 10. Specify a value of -1 for unlimited FPS control
Method: EZ_B.JPEGStream.Stop

Stop the JPEG Streamer

RandomUnique

Method: EZ_B.RandomUnique.GetRandomNumber(*System.Int32, System.Int32*)

Return a random number within specified range.
Using this random number generating function will provide a common seed.
Method: EZ_B.RandomUnique.GetRandomUniqueNumber(*System.Int32, System.Int32*)

Return a random number and tries to make the returned value unique from the last time this function was called.

EZ430

Field: EZ_B.EZ430.Address

Once connected is successfully established, this returns the address of the chrono watch. Four bytes, seperated by a dash
Method: EZ_B.EZ430.Start

Start the eZ430
Method: EZ_B.EZ430.Stop

Stop the eZ430
Method: EZ_B.EZ430.Ping

Send a ping to the eZ430 and wait for a response
Method: EZ_B.EZ430.GetAccData

Get the accelometer data from the eZ430

RSS

Method: EZ_B.RSS.GetRSSMessage(*System.String, EZ_B.RSS.SortDirectionEnum, System.Int32*)

Gets an RSS feed and only returns the specified story index. Returns the last if the specified storyIndex is greater than the index count.
The storyIndex is a zero based number.
Method: EZ_B.RSS.GetRSSMessages(*System.String, EZ_B.RSS.SortDirectionEnum*)

Get RSS feed from the specific URL

Sphero

Method: EZ_B.Sphero.Roll(*System.Int32, System.Byte*)

The headingDegrees follows the 360 degrees on a circle, relative to the ball: 0 is straight ahead, 90 is to the right, 180 is back and 270 is to the left. The valid range is 0..359
Method: EZ_B.Sphero.SetHeading(*System.Int32*)

Adjusts the orientation of Sphero by commanding a new reference heading in degrees, which ranges from 0 to 359. You will see the ball respond immediately to this command if stabilization is enabled.

Twitter

Method: EZ_B.Twitter.GetTwitterMessage(*System.String, EZ_B.RSS.SortDirectionEnum, System.Int32*)

Get latest messages from specified twitter account
Method: EZ_B.Twitter.GetTwitterMessages(*System.String, EZ_B.RSS.SortDirectionEnum*)

Get latest messages from specified twitter account

Speakjet

Field: EZ_B.Speakjet.Baud

Specify the baud rate for the connection to the SpeakJet
Field: EZ_B.Speakjet.soundCodes

Sound codes offset starts at 128 to 254
Method: EZ_B.Speakjet.SpeakCode(*EZ_B.Digital.DigitalPortEnum, System.Byte[]*)

Speak by phonetic codes
Method: EZ_B.Speakjet.Reset(*EZ_B.Digital.DigitalPortEnum*)

Reset the Speakjet
Method: EZ_B.Speakjet.SetDistortion(*EZ_B.Digital.DigitalPortEnum, System.Int32*)

Set the global distortion between 0-255
Method: EZ_B.Speakjet.SetVolume(*EZ_B.Digital.DigitalPortEnum, System.Int32*)

Set the global volume between 0-255
Method: EZ_B.Speakjet.SetEnvelope(*EZ_B.Digital.DigitalPortEnum, EZ_B.Speakjet.EnvelopeType, System.Int32, System.Boolean, System.Boolean*)

Set the global envelope. Frequency is between 0-3999
Method: EZ_B.Speakjet.PlayNote(*EZ_B.Digital.DigitalPortEnum, EZ_B.Speakjet.OscillatorEnum, System.Int32, System.Int32*)

Play a note on the selected oscillator. Frequency is between 0-3999. Volume is between 0-31
Method: EZ_B.Speakjet.SpeakString(*EZ_B.Digital.DigitalPortEnum, System.String*)

Speak by string and reference the internal dictionary to pronounce words.
If words are not found in dictionary, custom \PHONETICs can be used.

PWM

Field: EZ_B.PWM.PWM_MAX

The maximum value for a PWM (100)
Field: EZ_B.PWM.PWM_MIN

The minimum value of a PWM (0)
Method: EZ_B.PWM.SetPWM(*EZ_B.Digital.DigitalPortEnum, System.Int32*)

Set the PWM Duty Cycle. The speed can be between PWM_MIN and PWM_MAX
Method: EZ_B.PWM.SetPWM(*EZ_B.Classes.PWMItem[]*)

Set the PWM Duty Cycle. The speed can be between PWM_MIN and PWM_MAX
Method: EZ_B.PWM.GetPWM(*EZ_B.Digital.DigitalPortEnum*)

Get the PWM
Method: EZ_B.PWM.StopPWM(*EZ_B.Digital.DigitalPortEnum*)

Stop PWM.
Method: EZ_B.PWM.IsPWMStopped(*EZ_B.Digital.DigitalPortEnum*)

Return true if the specified pwm port is in a stopped state

Resource1

Type: EZ_B.Resource1

A strongly-typed resource class, for looking up localized strings, etc.

TCPServer

Event: EZ_B.TCPServer.OnByteReceived

This event returns the bytes.
Event: EZ_B.TCPServer.OnCommandReceived

Event risen when for handleCustomEvent is true and a connected user presses the Enter key.
This event returns the line of text entered by the user.
Event: EZ_B.TCPServer.OnConnection

Event risen when for handleCustomEvent is true and a new connection is established
Method: EZ_B.TCPServer.Start(*System.Int32*)

Start the TCP Server and beginning listening on the specified port.
Method: EZ_B.TCPServer.Stop

Stop the TCP Server listener
Method: EZ_B.TCPServer.GetConnectedClients

Receive a list of the connected clients and their respective terminal id's
Method: EZ_B.TCPServer.CloseClientById(*System.Int32*)

Disconnect a client by it's terminal id
Method: EZ_B.TCPServer.DisconnectClients

Disconnect all clients

UCCameraCanvas

Method: EZ_B.UCCameraCanvas.OnPaint(*System.Windows.Forms.PaintEventArgs*)

We ignore painting because all we care about is the background painting
Method: EZ_B.UCCameraCanvas.OnPaintBackground(*System.Windows.Forms.PaintEventArgs*)

Override the background painting because we don't want the base to paint if we're busy.
This is because the direct memory copy to the backgroundimage locks the bitmap, and so does the onPaintBackground
Method: EZ_B.UCCameraCanvas.SetDefaultCameraImage

Set the camera image to the default camera stub.
If the camera is going to be stopped, this lets the user know
This is threadsafe

UCComboBoxTextBox

Event: EZ_B.UCComboBoxTextBox.OnSelectedItemChanged

Event that is raised when reading the positions of servos that support reading positions
Contains the servo of the requesting position
Method: EZ_B.UCComboBoxTextBox.Add(*System.String*)

Adds an item to the list and selects that item
Does not raise SelectedItemChanged event
Method: EZ_B.UCComboBoxTextBox.Add(*EZ_B.ValuePair*)

Adds an item to the list and selects that item.
Does not raise SelectedItemChanged event
Method: EZ_B.UCComboBoxTextBox.AddRange(*System.String[]*)

Adds items to the list and selects that item
Does not raise SelectedItemChanged event
Method: EZ_B.UCComboBoxTextBox.AddRange(*EZ_B.ValuePair[]*)

Adds items to the list and selects that item
Does not raise SelectedItemChanged event

Video

Type: EZ_B.Video.DirectShow.CameraControlProperty

The enumeration specifies a setting on a camera.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Pan

Pan control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Tilt

Tilt control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Roll

Roll control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Zoom

Zoom control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Exposure

Exposure control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Iris

Iris control.
Field: EZ_B.Video.DirectShow.CameraControlProperty.Focus

Focus control.
Type: EZ_B.Video.DirectShow.CameraControlFlags

The enumeration defines whether a camera setting is controlled manually or automatically.
Field: EZ_B.Video.DirectShow.CameraControlFlags.None

No control flag.
Field: EZ_B.Video.DirectShow.CameraControlFlags.Auto

Auto control Flag.
Field: EZ_B.Video.DirectShow.CameraControlFlags.Manual

Manual control Flag.
Type: EZ_B.Video.DirectShow.FilterInfo

DirectShow filter information.
Method: EZ_B.Video.DirectShow.FilterInfo.CompareTo(*System.Object*)

Compare the object with another instance of this class.
Method: EZ_B.Video.DirectShow.FilterInfo.CreateFilter(*System.String*)

Create an instance of the filter.
Type: EZ_B.Video.DirectShow.FilterInfoCollection

Collection of filters' information objects.
Type: EZ_B.Video.DirectShow.Internals.IAMCameraControl

The IAMCameraControl interface controls camera settings such as zoom, pan, aperture adjustment,
or shutter speed. To obtain this interface, query the filter that controls the camera.
Method: EZ_B.Video.DirectShow.Internals.IAMCameraControl.GetRange(*EZ_B.Video.DirectShow.CameraControlProperty, System.Int32@, System.Int32@, System.Int32@, System.Int32@, EZ_B.Video.DirectShow.CameraControlFlags@*)

Gets the range and default value of a specified camera property.
Method: EZ_B.Video.DirectShow.Internals.IAMCameraControl.Set(*EZ_B.Video.DirectShow.CameraControlProperty, System.Int32, EZ_B.Video.DirectShow.CameraControlFlags*)

Sets a specified property on the camera.
Method: EZ_B.Video.DirectShow.Internals.IAMCameraControl.Get(*EZ_B.Video.DirectShow.CameraControlProperty, System.Int32@, EZ_B.Video.DirectShow.CameraControlFlags@*)

Gets the current setting of a camera property.
Type: EZ_B.Video.DirectShow.Internals.IAMCrossbar

The IAMCrossbar interface routes signals from an analog or digital source to a video capture filter.
Method: EZ_B.Video.DirectShow.Internals.IAMCrossbar.get_PinCounts(*System.Int32@, System.Int32@*)

Retrieves the number of input and output pins on the crossbar filter.
Method: EZ_B.Video.DirectShow.Internals.IAMCrossbar.CanRoute(*System.Int32, System.Int32*)

Queries whether a specified input pin can be routed to a specified output pin.
Method: EZ_B.Video.DirectShow.Internals.IAMCrossbar.Route(*System.Int32, System.Int32*)

Routes an input pin to an output pin.
Method: EZ_B.Video.DirectShow.Internals.IAMCrossbar.get_IsRoutedTo(*System.Int32, System.Int32@*)

Retrieves the input pin that is currently routed to the specified output pin.
Method: EZ_B.Video.DirectShow.Internals.IAMCrossbar.get_CrossbarPinInfo(*System.Boolean, System.Int32, System.Int32@, EZ_B.Video.DirectShow.PhysicalConnectorType@*)

Retrieves information about a specified pin.
Type: EZ_B.Video.DirectShow.Internals.IAMStreamConfig

This interface sets the output format on certain capture and compression filters,
for both audio and video.

Method: EZ_B.Video.DirectShow.Internals.IAMStreamConfig.SetFormat(*EZ_B.Video.DirectShow.Internals.AMMediaType*)

Set the output format on the pin.
Method: EZ_B.Video.DirectShow.Internals.IAMStreamConfig.GetFormat(*EZ_B.Video.DirectShow.Internals.AMMediaType@*)

Retrieves the audio or video stream's format.
Method: EZ_B.Video.DirectShow.Internals.IAMStreamConfig.GetNumberOfCapabilities(*System.Int32@, System.Int32@*)

Retrieve the number of format capabilities that this pin supports.

Method: EZ_B.Video.DirectShow.Internals.IAMStreamConfig.GetStreamCaps(*System.Int32, EZ_B.Video.DirectShow.Internals.AMMediaType@, EZ_B.Video.DirectShow.Internals.VideoStreamConfigCaps*)

Retrieve a set of format capabilities.
Type: EZ_B.Video.DirectShow.Internals.IAMVideoControl

The interface controls certain video capture operations such as enumerating available
frame rates and image orientation.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.GetCaps(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.VideoControlFlags@*)

Retrieves the capabilities of the underlying hardware.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.SetMode(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.VideoControlFlags*)

Sets the video control mode of operation.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.GetMode(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.VideoControlFlags@*)

Retrieves the video control mode of operation.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.GetCurrentActualFrameRate(*EZ_B.Video.DirectShow.Internals.IPin, System.Int64@*)

The method retrieves the actual frame rate, expressed as a frame duration in 100-nanosecond units.
USB (Universal Serial Bus) and IEEE 1394 cameras may provide lower frame rates than requested
because of bandwidth availability. This is only available during video streaming.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.GetMaxAvailableFrameRate(*EZ_B.Video.DirectShow.Internals.IPin, System.Int32, System.Drawing.Size, System.Int64@*)

Retrieves the maximum frame rate currently available based on bus bandwidth usage for connections
such as USB and IEEE 1394 camera devices where the maximum frame rate can be limited by bandwidth
availability.
Method: EZ_B.Video.DirectShow.Internals.IAMVideoControl.GetFrameRateList(*EZ_B.Video.DirectShow.Internals.IPin, System.Int32, System.Drawing.Size, System.Int32@, System.IntPtr@*)

Retrieves a list of available frame rates.
Type: EZ_B.Video.DirectShow.Internals.IBaseFilter

The IBaseFilter interface provides methods for controlling a filter.
All DirectShow filters expose this interface
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.GetClassID(*System.Guid@*)

Returns the class identifier (CLSID) for the component object.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.Stop

Stops the filter.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.Pause

Pauses the filter.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.Run(*System.Int64*)

Runs the filter.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.GetState(*System.Int32, System.Int32@*)

Retrieves the state of the filter (running, stopped, or paused).
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.SetSyncSource(*System.IntPtr*)

Sets the reference clock for the filter or the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.GetSyncSource(*System.IntPtr@*)

Retrieves the current reference clock.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.EnumPins(*EZ_B.Video.DirectShow.Internals.IEnumPins@*)

Enumerates the pins on this filter.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.FindPin(*System.String, EZ_B.Video.DirectShow.Internals.IPin@*)

Retrieves the pin with the specified identifier.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.QueryFilterInfo(*EZ_B.Video.DirectShow.Internals.FilterInfo@*)

Retrieves information about the filter.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.JoinFilterGraph(*EZ_B.Video.DirectShow.Internals.IFilterGraph, System.String*)

Notifies the filter that it has joined or left the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IBaseFilter.QueryVendorInfo(*System.String@*)

Retrieves a string containing vendor information.
Type: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2

This interface builds capture graphs and other custom filter graphs.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.SetFiltergraph(*EZ_B.Video.DirectShow.Internals.IGraphBuilder*)

Specify filter graph for the capture graph builder to use.

Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.GetFiltergraph(*EZ_B.Video.DirectShow.Internals.IGraphBuilder@*)

Retrieve the filter graph that the builder is using.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.SetOutputFileName(*System.Guid, System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@, System.IntPtr@*)

Create file writing section of the filter graph.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.FindInterface(*System.Guid, System.Guid, EZ_B.Video.DirectShow.Internals.IBaseFilter, System.Guid, System.Object@*)

Searche the graph for a specified interface, starting from a specified filter.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.RenderStream(*System.Guid, System.Guid, System.Object, EZ_B.Video.DirectShow.Internals.IBaseFilter, EZ_B.Video.DirectShow.Internals.IBaseFilter*)

Connect an output pin on a source filter to a rendering filter, optionally through a compression filter.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.ControlStream(*System.Guid, System.Guid, EZ_B.Video.DirectShow.Internals.IBaseFilter, System.Int64, System.Int64, System.Int16, System.Int16*)

Set the start and stop times for one or more streams of captured data.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.AllocCapFile(*System.String, System.Int64*)

Preallocate a capture file to a specified size.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.CopyCaptureFile(*System.String, System.String, System.Boolean, System.IntPtr*)

Copy the valid media data from a capture file.
Method: EZ_B.Video.DirectShow.Internals.ICaptureGraphBuilder2.FindPin(*System.Object, EZ_B.Video.DirectShow.Internals.PinDirection, System.Guid, System.Guid, System.Boolean, System.Int32, EZ_B.Video.DirectShow.Internals.IPin@*)

Type: EZ_B.Video.DirectShow.Internals.ICreateDevEnum

The ICreateDevEnum interface creates an enumerator for devices within a particular category,
such as video capture devices, audio capture devices, video compressors, and so forth.
Method: EZ_B.Video.DirectShow.Internals.ICreateDevEnum.CreateClassEnumerator(*System.Guid@, System.Runtime.InteropServices.ComTypes.IEnumMoniker@, System.Int32*)

Creates a class enumerator for a specified device category.
Type: EZ_B.Video.DirectShow.Internals.IEnumFilters

This interface is used by applications or other filters to determine
what filters exist in the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IEnumFilters.Next(*System.Int32, EZ_B.Video.DirectShow.Internals.IBaseFilter[], System.Int32@*)

Retrieves the specified number of filters in the enumeration sequence.
Method: EZ_B.Video.DirectShow.Internals.IEnumFilters.Skip(*System.Int32*)

Skips a specified number of filters in the enumeration sequence.
Method: EZ_B.Video.DirectShow.Internals.IEnumFilters.Reset

Resets the enumeration sequence to the beginning.
Method: EZ_B.Video.DirectShow.Internals.IEnumFilters.Clone(*EZ_B.Video.DirectShow.Internals.IEnumFilters@*)

Makes a copy of the enumerator with the same enumeration state.
Type: EZ_B.Video.DirectShow.Internals.IEnumPins

Enumerates pins on a filter.
Method: EZ_B.Video.DirectShow.Internals.IEnumPins.Next(*System.Int32, EZ_B.Video.DirectShow.Internals.IPin[], System.Int32@*)

Retrieves a specified number of pins.
Method: EZ_B.Video.DirectShow.Internals.IEnumPins.Skip(*System.Int32*)

Skips a specified number of pins in the enumeration sequence.
Method: EZ_B.Video.DirectShow.Internals.IEnumPins.Reset

Resets the enumeration sequence to the beginning.
Method: EZ_B.Video.DirectShow.Internals.IEnumPins.Clone(*EZ_B.Video.DirectShow.Internals.IEnumPins@*)

Makes a copy of the enumerator with the same enumeration state.
Type: EZ_B.Video.DirectShow.Internals.IFileSourceFilter

The interface is exposed by source filters to set the file name and media type of the media file that they are to render.
Method: EZ_B.Video.DirectShow.Internals.IFileSourceFilter.Load(*System.String, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Loads the source filter with the file.
Method: EZ_B.Video.DirectShow.Internals.IFileSourceFilter.GetCurFile(*System.String@, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Retrieves the current file.
Type: EZ_B.Video.DirectShow.Internals.IFilterGraph

The interface provides methods for building a filter graph. An application can use it to add filters to
the graph, connect or disconnect filters, remove filters, and perform other basic operations.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.AddFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter, System.String*)

Adds a filter to the graph and gives it a name.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.RemoveFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter*)

Removes a filter from the graph.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.EnumFilters(*System.IntPtr@*)

Provides an enumerator for all filters in the graph.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.FindFilterByName(*System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Finds a filter that was added with a specified name.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.ConnectDirect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Connects two pins directly (without intervening filters).
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.Reconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Breaks the existing pin connection and reconnects it to the same pin.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.Disconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Disconnects a specified pin.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph.SetDefaultSyncSource

Sets the reference clock to the default clock.
Type: EZ_B.Video.DirectShow.Internals.IFilterGraph2

This interface extends the and
interfaces, which contain methods for building filter graphs.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.AddFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter, System.String*)

Adds a filter to the graph and gives it a name.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.RemoveFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter*)

Removes a filter from the graph.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.EnumFilters(*EZ_B.Video.DirectShow.Internals.IEnumFilters@*)

Provides an enumerator for all filters in the graph.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.FindFilterByName(*System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Finds a filter that was added with a specified name.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.ConnectDirect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Connects two pins directly (without intervening filters).
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.Reconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Breaks the existing pin connection and reconnects it to the same pin.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.Disconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Disconnects a specified pin.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.SetDefaultSyncSource

Sets the reference clock to the default clock.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.Connect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.IPin*)

Connects two pins. If they will not connect directly, this method connects them with intervening transforms.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.Render(*EZ_B.Video.DirectShow.Internals.IPin*)

Adds a chain of filters to a specified output pin to render it.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.RenderFile(*System.String, System.String*)

Builds a filter graph that renders the specified file.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.AddSourceFilter(*System.String, System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Adds a source filter to the filter graph for a specific file.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.SetLogFile(*System.IntPtr*)

Sets the file for logging actions taken when attempting to perform an operation.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.Abort

Requests that the graph builder return as soon as possible from its current task.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.ShouldOperationContinue

Queries whether the current operation should continue.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.AddSourceFilterForMoniker(*System.Runtime.InteropServices.ComTypes.IMoniker, System.Runtime.InteropServices.ComTypes.IBindCtx, System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.ReconnectEx(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Breaks the existing pin connection and reconnects it to the same pin,
using a specified media type.
Method: EZ_B.Video.DirectShow.Internals.IFilterGraph2.RenderEx(*EZ_B.Video.DirectShow.Internals.IPin, System.Int32, System.IntPtr*)

Render an output pin, with an option to use existing renderers only.
Type: EZ_B.Video.DirectShow.Internals.IGraphBuilder

This interface provides methods that enable an application to build a filter graph.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.AddFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter, System.String*)

Adds a filter to the graph and gives it a name.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.RemoveFilter(*EZ_B.Video.DirectShow.Internals.IBaseFilter*)

Removes a filter from the graph.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.EnumFilters(*EZ_B.Video.DirectShow.Internals.IEnumFilters@*)

Provides an enumerator for all filters in the graph.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.FindFilterByName(*System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Finds a filter that was added with a specified name.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.ConnectDirect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Connects two pins directly (without intervening filters).
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.Reconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Breaks the existing pin connection and reconnects it to the same pin.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.Disconnect(*EZ_B.Video.DirectShow.Internals.IPin*)

Disconnects a specified pin.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.SetDefaultSyncSource

Sets the reference clock to the default clock.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.Connect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.IPin*)

Connects two pins. If they will not connect directly, this method connects them with intervening transforms.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.Render(*EZ_B.Video.DirectShow.Internals.IPin*)

Adds a chain of filters to a specified output pin to render it.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.RenderFile(*System.String, System.String*)

Builds a filter graph that renders the specified file.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.AddSourceFilter(*System.String, System.String, EZ_B.Video.DirectShow.Internals.IBaseFilter@*)

Adds a source filter to the filter graph for a specific file.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.SetLogFile(*System.IntPtr*)

Sets the file for logging actions taken when attempting to perform an operation.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.Abort

Requests that the graph builder return as soon as possible from its current task.
Method: EZ_B.Video.DirectShow.Internals.IGraphBuilder.ShouldOperationContinue

Queries whether the current operation should continue.
Type: EZ_B.Video.DirectShow.Internals.IMediaControl

The interface provides methods for controlling the flow of data through the filter graph.
It includes methods for running, pausing, and stopping the graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.Run

Runs all the filters in the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.Pause

Pauses all filters in the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.Stop

Stops all the filters in the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.GetState(*System.Int32, System.Int32@*)

Retrieves the state of the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.RenderFile(*System.String*)

Builds a filter graph that renders the specified file.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.AddSourceFilter(*System.String, System.Object@*)

Adds a source filter to the filter graph, for a specified file.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.get_FilterCollection(*System.Object@*)

Retrieves a collection of the filters in the filter graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.get_RegFilterCollection(*System.Object@*)

Retrieves a collection of all the filters listed in the registry.
Method: EZ_B.Video.DirectShow.Internals.IMediaControl.StopWhenReady

Pauses the filter graph, allowing filters to queue data, and then stops the filter graph.
Type: EZ_B.Video.DirectShow.Internals.IMediaEventEx

The interface inherits contains methods for retrieving event notifications and for overriding the
filter graph's default handling of events.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.GetEventHandle(*System.IntPtr@*)

Retrieves a handle to a manual-reset event that remains signaled while the queue contains event notifications.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.GetEvent(*EZ_B.Video.DirectShow.Internals.DsEvCode@, System.IntPtr@, System.IntPtr@, System.Int32*)

Retrieves the next event notification from the event queue.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.WaitForCompletion(*System.Int32, System.Int32@*)

Waits for the filter graph to render all available data.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.CancelDefaultHandling(*System.Int32*)

Cancels the Filter Graph Manager's default handling for a specified event.

Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.RestoreDefaultHandling(*System.Int32*)

Restores the Filter Graph Manager's default handling for a specified event.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.FreeEventParams(*EZ_B.Video.DirectShow.Internals.DsEvCode, System.IntPtr, System.IntPtr*)

Frees resources associated with the parameters of an event.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.SetNotifyWindow(*System.IntPtr, System.Int32, System.IntPtr*)

Registers a window to process event notifications.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.SetNotifyFlags(*System.Int32*)

Enables or disables event notifications.
Method: EZ_B.Video.DirectShow.Internals.IMediaEventEx.GetNotifyFlags(*System.Int32@*)

Determines whether event notifications are enabled.
Type: EZ_B.Video.DirectShow.Internals.IMediaFilter

The interface provides methods for controlling the flow of data through the filter graph.
It includes methods for running, pausing, and stopping the graph.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.Stop

This method informs the filter to transition to the new state.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.Pause

This method informs the filter to transition to the new state.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.Run(*System.Int64*)

This method informs the filter to transition to the new (running) state. Passes a time value to synchronize independent streams.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.GetState(*System.Int32, EZ_B.Video.DirectShow.Internals.FilterState@*)

This method determines the filter's state.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.SetSyncSource(*EZ_B.Video.DirectShow.Internals.IReferenceClock*)

This method identifies the reference clock to which the filter should synchronize activity.
Method: EZ_B.Video.DirectShow.Internals.IMediaFilter.GetSyncSource(*EZ_B.Video.DirectShow.Internals.IReferenceClock@*)

This method retrieves the current reference clock in use by this filter.
Type: EZ_B.Video.DirectShow.Internals.IPersist

Provides the CLSID of an object that can be stored persistently in the system. Allows the object to specify which object
handler to use in the client process, as it is used in the default implementation of marshaling.
Method: EZ_B.Video.DirectShow.Internals.IPersist.GetClassID(*System.Guid@*)

Retrieves the class identifier (CLSID) of the object.
Type: EZ_B.Video.DirectShow.Internals.IPin

This interface is exposed by all input and output pins of DirectShow filters.
Method: EZ_B.Video.DirectShow.Internals.IPin.Connect(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Connects the pin to another pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.ReceiveConnection(*EZ_B.Video.DirectShow.Internals.IPin, EZ_B.Video.DirectShow.Internals.AMMediaType*)

Makes a connection to this pin and is called by a connecting pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.Disconnect

Breaks the current pin connection.
Method: EZ_B.Video.DirectShow.Internals.IPin.ConnectedTo(*EZ_B.Video.DirectShow.Internals.IPin@*)

Returns a pointer to the connecting pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.ConnectionMediaType(*EZ_B.Video.DirectShow.Internals.AMMediaType*)

Returns the media type of this pin's connection.
Method: EZ_B.Video.DirectShow.Internals.IPin.QueryPinInfo(*EZ_B.Video.DirectShow.Internals.PinInfo@*)

Retrieves information about this pin (for example, the name, owning filter, and direction).
Method: EZ_B.Video.DirectShow.Internals.IPin.QueryDirection(*EZ_B.Video.DirectShow.Internals.PinDirection@*)

Retrieves the direction for this pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.QueryId(*System.String@*)

Retrieves an identifier for the pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.QueryAccept(*EZ_B.Video.DirectShow.Internals.AMMediaType*)

Queries whether a given media type is acceptable by the pin.
Method: EZ_B.Video.DirectShow.Internals.IPin.EnumMediaTypes(*System.IntPtr*)

Provides an enumerator for this pin's preferred media types.
Method: EZ_B.Video.DirectShow.Internals.IPin.QueryInternalConnections(*System.IntPtr, System.Int32@*)

Provides an array of the pins to which this pin internally connects.
Method: EZ_B.Video.DirectShow.Internals.IPin.EndOfStream

Notifies the pin that no additional data is expected.
Method: EZ_B.Video.DirectShow.Internals.IPin.BeginFlush

Begins a flush operation.
Method: EZ_B.Video.DirectShow.Internals.IPin.EndFlush

Ends a flush operation.
Method: EZ_B.Video.DirectShow.Internals.IPin.NewSegment(*System.Int64, System.Int64, System.Double*)

Specifies that samples following this call are grouped as a segment with a given start time, stop time, and rate.
Type: EZ_B.Video.DirectShow.Internals.IPropertyBag

The IPropertyBag interface provides an object with a property bag in
which the object can persistently save its properties.
Method: EZ_B.Video.DirectShow.Internals.IPropertyBag.Read(*System.String, System.Object@, System.IntPtr*)

Read a property from property bag.
Method: EZ_B.Video.DirectShow.Internals.IPropertyBag.Write(*System.String, System.Object@*)

Write property to property bag.
Type: EZ_B.Video.DirectShow.Internals.IReferenceClock

The IReferenceClock interface provides the reference time for the filter graph.

Filters that can act as a reference clock can expose this interface. It is also exposed by the System Reference Clock.
The filter graph manager uses this interface to synchronize the filter graph. Applications can use this interface to
retrieve the current reference time, or to request notification of an elapsed time.
Method: EZ_B.Video.DirectShow.Internals.IReferenceClock.GetTime(*System.Int64@*)

The GetTime method retrieves the current reference time.
Method: EZ_B.Video.DirectShow.Internals.IReferenceClock.AdviseTime(*System.Int64, System.Int64, System.IntPtr, System.Int32@*)

The AdviseTime method creates a one-shot advise request.
Method: EZ_B.Video.DirectShow.Internals.IReferenceClock.AdvisePeriodic(*System.Int64, System.Int64, System.IntPtr, System.Int32@*)

The AdvisePeriodic method creates a periodic advise request.
Method: EZ_B.Video.DirectShow.Internals.IReferenceClock.Unadvise(*System.Int32*)

The Unadvise method removes a pending advise request.

Type: EZ_B.Video.DirectShow.Internals.ISampleGrabber

The interface is exposed by the Sample Grabber Filter. It enables an application to retrieve
individual media samples as they move through the filter graph.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.SetOneShot(*System.Boolean*)

Specifies whether the filter should stop the graph after receiving one sample.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.SetMediaType(*EZ_B.Video.DirectShow.Internals.AMMediaType*)

Specifies the media type for the connection on the Sample Grabber's input pin.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.GetConnectedMediaType(*EZ_B.Video.DirectShow.Internals.AMMediaType*)

Retrieves the media type for the connection on the Sample Grabber's input pin.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.SetBufferSamples(*System.Boolean*)

Specifies whether to copy sample data into a buffer as it goes through the filter.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.GetCurrentBuffer(*System.Int32@, System.IntPtr*)

Retrieves a copy of the sample that the filter received most recently.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.GetCurrentSample(*System.IntPtr*)

Not currently implemented.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabber.SetCallback(*EZ_B.Video.DirectShow.Internals.ISampleGrabberCB, System.Int32*)

Specifies a callback method to call on incoming samples.
Type: EZ_B.Video.DirectShow.Internals.ISampleGrabberCB

The interface provides callback methods for the method.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabberCB.SampleCB(*System.Double, System.IntPtr*)

Callback method that receives a pointer to the media sample.
Method: EZ_B.Video.DirectShow.Internals.ISampleGrabberCB.BufferCB(*System.Double, System.IntPtr, System.Int32*)

Callback method that receives a pointer to the sample bufferÃ¾
Type: EZ_B.Video.DirectShow.Internals.ISpecifyPropertyPages

The interface indicates that an object supports property pages.
Method: EZ_B.Video.DirectShow.Internals.ISpecifyPropertyPages.GetPages(*EZ_B.Video.DirectShow.Internals.CAUUID@*)

Fills a counted array of GUID values where each GUID specifies the
CLSID of each property page that can be displayed in the property
sheet for this object.
Type: EZ_B.Video.DirectShow.Internals.IVideoWindow

The interface sets properties on the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Caption(*System.String*)

Sets the video window caption.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Caption(*System.String@*)

Retrieves the video window caption.

Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_WindowStyle(*System.Int32*)

Sets the window style on the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_WindowStyle(*System.Int32@*)

Retrieves the window style on the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_WindowStyleEx(*System.Int32*)

Sets the extended window style on the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_WindowStyleEx(*System.Int32@*)

Retrieves the extended window style on the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_AutoShow(*System.Boolean*)

Specifies whether the video renderer automatically shows the video window when it receives video data.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_AutoShow(*System.Boolean@*)

Queries whether the video renderer automatically shows the video window when it receives video data.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_WindowState(*System.Int32*)

Shows, hides, minimizes, or maximizes the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_WindowState(*System.Int32@*)

Queries whether the video window is visible, hidden, minimized, or maximized.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_BackgroundPalette(*System.Boolean*)

Specifies whether the video window realizes its palette in the background.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_BackgroundPalette(*System.Boolean@*)

Queries whether the video window realizes its palette in the background.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Visible(*System.Boolean*)

Shows or hides the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Visible(*System.Boolean@*)

Queries whether the video window is visible.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Left(*System.Int32*)

Sets the video window's x-coordinate.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Left(*System.Int32@*)

Retrieves the video window's x-coordinate.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Width(*System.Int32*)

Sets the width of the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Width(*System.Int32@*)

Retrieves the width of the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Top(*System.Int32*)

Sets the video window's y-coordinate.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Top(*System.Int32@*)

Retrieves the video window's y-coordinate.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Height(*System.Int32*)

Sets the height of the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Height(*System.Int32@*)

Retrieves the height of the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_Owner(*System.IntPtr*)

Specifies a parent window for the video windowÃ¾
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_Owner(*System.IntPtr@*)

Retrieves the video window's parent window, if anyÃ¾
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_MessageDrain(*System.IntPtr*)

Specifies a window to receive mouse and keyboard messages from the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_MessageDrain(*System.IntPtr@*)

Retrieves the window that receives mouse and keyboard messages from the video window, if any.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_BorderColor(*System.Int32@*)

Retrieves the color that appears around the edges of the destination rectangle.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_BorderColor(*System.Int32*)

Sets the color that appears around the edges of the destination rectangle.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.get_FullScreenMode(*System.Boolean@*)

Queries whether the video renderer is in full-screen mode.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.put_FullScreenMode(*System.Boolean*)

Enables or disables full-screen mode.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.SetWindowForeground(*System.Int32*)

Places the video window at the top of the Z order.

Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.NotifyOwnerMessage(*System.IntPtr, System.Int32, System.IntPtr, System.IntPtr*)

Forwards a message to the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.SetWindowPosition(*System.Int32, System.Int32, System.Int32, System.Int32*)

Sets the position of the video windowÃ¾
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.GetWindowPosition(*System.Int32@, System.Int32@, System.Int32@, System.Int32@*)

Retrieves the position of the video window.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.GetMinIdealImageSize(*System.Int32@, System.Int32@*)

Retrieves the minimum ideal size for the video image.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.GetMaxIdealImageSize(*System.Int32@, System.Int32@*)

Retrieves the maximum ideal size for the video image.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.GetRestorePosition(*System.Int32@, System.Int32@, System.Int32@, System.Int32@*)

Retrieves the restored window position.
Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.HideCursor(*System.Boolean*)

Hides the cursor.

Method: EZ_B.Video.DirectShow.Internals.IVideoWindow.IsCursorHidden(*System.Boolean@*)

Queries whether the cursor is hidden.
Type: EZ_B.Video.DirectShow.Internals.PinDirection

This enumeration indicates a pin's direction.
Field: EZ_B.Video.DirectShow.Internals.PinDirection.Input

Input pin.
Field: EZ_B.Video.DirectShow.Internals.PinDirection.Output

Output pin.
Type: EZ_B.Video.DirectShow.Internals.AMMediaType

The structure describes the format of a media sample.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.MajorType

Globally unique identifier (GUID) that specifies the major type of the media sample.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.SubType

GUID that specifies the subtype of the media sample.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.FixedSizeSamples

If true, samples are of a fixed size.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.TemporalCompression

If true, samples are compressed using temporal (interframe) compression.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.SampleSize

Size of the sample in bytes. For compressed data, the value can be zero.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.FormatType

GUID that specifies the structure used for the format block.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.unkPtr

Not used.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.FormatSize

Size of the format block, in bytes.
Field: EZ_B.Video.DirectShow.Internals.AMMediaType.FormatPtr

Pointer to the format block.
Method: EZ_B.Video.DirectShow.Internals.AMMediaType.Finalize

Destroys the instance of the class.
Type: EZ_B.Video.DirectShow.Internals.PinInfo

The structure contains information about a pin.
Field: EZ_B.Video.DirectShow.Internals.PinInfo.Filter

Owning filter.
Field: EZ_B.Video.DirectShow.Internals.PinInfo.Direction

Direction of the pin.
Field: EZ_B.Video.DirectShow.Internals.PinInfo.Name

Name of the pin.
Field: EZ_B.Video.DirectShow.Internals.FilterInfo.Name

Filter's name.
Field: EZ_B.Video.DirectShow.Internals.FilterInfo.FilterGraph

Owning graph.
Type: EZ_B.Video.DirectShow.Internals.VideoInfoHeader

The structure describes the bitmap and color information for a video image.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.SrcRect

structure that specifies the source video window.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.TargetRect

structure that specifies the destination video window.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.BitRate

Approximate data rate of the video stream, in bits per second.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.BitErrorRate

Data error rate, in bit errors per second.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.AverageTimePerFrame

The desired average display time of the video frames, in 100-nanosecond units.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader.BmiHeader

structure that contains color and dimension information for the video image bitmap.
Type: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2

The structure describes the bitmap and color information for a video image (v2).
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.SrcRect

structure that specifies the source video window.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.TargetRect

structure that specifies the destination video window.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.BitRate

Approximate data rate of the video stream, in bits per second.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.BitErrorRate

Data error rate, in bit errors per second.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.AverageTimePerFrame

The desired average display time of the video frames, in 100-nanosecond units.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.InterlaceFlags

Flags that specify how the video is interlaced.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.CopyProtectFlags

Flag set to indicate that the duplication of the stream should be restricted.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.PictAspectRatioX

The X dimension of picture aspect ratio.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.PictAspectRatioY

The Y dimension of picture aspect ratio.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.Reserved1

Reserved for future use.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.Reserved2

Reserved for future use.
Field: EZ_B.Video.DirectShow.Internals.VideoInfoHeader2.BmiHeader

structure that contains color and dimension information for the video image bitmap.
Type: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader

The structure contains information about the dimensions and color format of a device-independent bitmap (DIB).
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.Size

Specifies the number of bytes required by the structure.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.Width

Specifies the width of the bitmap.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.Height

Specifies the height of the bitmap, in pixels.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.Planes

Specifies the number of planes for the target device. This value must be set to 1.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.BitCount

Specifies the number of bits per pixel.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.Compression

If the bitmap is compressed, this member is a FOURCC the specifies the compression.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.ImageSize

Specifies the size, in bytes, of the image.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.XPelsPerMeter

Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap.

Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.YPelsPerMeter

Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.ColorsUsed

Specifies the number of color indices in the color table that are actually used by the bitmap.
Field: EZ_B.Video.DirectShow.Internals.BitmapInfoHeader.ColorsImportant

Specifies the number of color indices that are considered important for displaying the bitmap.
Type: EZ_B.Video.DirectShow.Internals.RECT

The structure defines the coordinates of the upper-left and lower-right corners of a rectangle.
Field: EZ_B.Video.DirectShow.Internals.RECT.Left

Specifies the x-coordinate of the upper-left corner of the rectangle.
Field: EZ_B.Video.DirectShow.Internals.RECT.Top

Specifies the y-coordinate of the upper-left corner of the rectangle.
Field: EZ_B.Video.DirectShow.Internals.RECT.Right

Specifies the x-coordinate of the lower-right corner of the rectangle.
Field: EZ_B.Video.DirectShow.Internals.RECT.Bottom

Specifies the y-coordinate of the lower-right corner of the rectangle.
Type: EZ_B.Video.DirectShow.Internals.CAUUID

The CAUUID structure is a Counted Array of UUID or GUID types.
Field: EZ_B.Video.DirectShow.Internals.CAUUID.cElems

Size of the array pointed to by pElems.
Field: EZ_B.Video.DirectShow.Internals.CAUUID.pElems

Pointer to an array of UUID values, each of which specifies UUID.
Method: EZ_B.Video.DirectShow.Internals.CAUUID.ToGuidArray

Performs manual marshaling of pElems to retrieve an array of Guid objects.
Type: EZ_B.Video.DirectShow.Internals.DsEvCode

Enumeration of DirectShow event codes.
Type: EZ_B.Video.DirectShow.Internals.FilterState

Specifies a filter's state or the state of the filter graph.
Field: EZ_B.Video.DirectShow.Internals.FilterState.State_Stopped

Stopped. The filter is not processing data.
Field: EZ_B.Video.DirectShow.Internals.FilterState.State_Paused

Paused. The filter is processing data, but not rendering it.
Field: EZ_B.Video.DirectShow.Internals.FilterState.State_Running

Running. The filter is processing and rendering data.
Type: EZ_B.Video.DirectShow.Internals.Tools

Some miscellaneous functions.
Method: EZ_B.Video.DirectShow.Internals.Tools.GetPin(*EZ_B.Video.DirectShow.Internals.IBaseFilter, EZ_B.Video.DirectShow.Internals.PinDirection, System.Int32*)

Get filter's pin.
Method: EZ_B.Video.DirectShow.Internals.Tools.GetInPin(*EZ_B.Video.DirectShow.Internals.IBaseFilter, System.Int32*)

Get filter's input pin.
Method: EZ_B.Video.DirectShow.Internals.Tools.GetOutPin(*EZ_B.Video.DirectShow.Internals.IBaseFilter, System.Int32*)

Get filter's output pin.
Type: EZ_B.Video.DirectShow.Internals.Clsid

DirectShow class IDs.
Field: EZ_B.Video.DirectShow.Internals.Clsid.SystemDeviceEnum

System device enumerator.
Field: EZ_B.Video.DirectShow.Internals.Clsid.FilterGraph

Filter graph.
Field: EZ_B.Video.DirectShow.Internals.Clsid.SampleGrabber

Sample grabber.
Field: EZ_B.Video.DirectShow.Internals.Clsid.CaptureGraphBuilder2

Capture graph builder.
Field: EZ_B.Video.DirectShow.Internals.Clsid.AsyncReader

Async reader.
Type: EZ_B.Video.DirectShow.Internals.FormatType

DirectShow format types.
Field: EZ_B.Video.DirectShow.Internals.FormatType.VideoInfo

VideoInfo.
Field: EZ_B.Video.DirectShow.Internals.FormatType.VideoInfo2

VideoInfo2.
Type: EZ_B.Video.DirectShow.Internals.MediaType

DirectShow media types.
Field: EZ_B.Video.DirectShow.Internals.MediaType.Video

Video.
Field: EZ_B.Video.DirectShow.Internals.MediaType.Interleaved

Interleaved. Used by Digital Video (DV).
Field: EZ_B.Video.DirectShow.Internals.MediaType.Audio

Audio.
Field: EZ_B.Video.DirectShow.Internals.MediaType.Text

Text.

Field: EZ_B.Video.DirectShow.Internals.MediaType.Stream

Byte stream with no time stamps.
Type: EZ_B.Video.DirectShow.Internals.MediaSubType

DirectShow media subtypes.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.YUYV

YUY2 (packed 4:2:2).
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.IYUV

IYUV.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.DVSD

A DV encoding format. (FOURCC 'DVSD')
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB1

RGB, 1 bit per pixel (bpp), palettized.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB4

RGB, 4 bpp, palettized.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB8

RGB, 8 bpp.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB565

RGB 565, 16 bpp.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB555

RGB 555, 16 bpp.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB24

RGB, 24 bpp.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.RGB32

RGB, 32 bpp, no alpha channel.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.Avi

Data from AVI file.
Field: EZ_B.Video.DirectShow.Internals.MediaSubType.Asf

Advanced Streaming Format (ASF).
Type: EZ_B.Video.DirectShow.Internals.PinCategory

DirectShow pin categories.
Field: EZ_B.Video.DirectShow.Internals.PinCategory.Capture

Capture pin.
Field: EZ_B.Video.DirectShow.Internals.PinCategory.StillImage

Still image pin.
Field: EZ_B.Video.DirectShow.Internals.FindDirection.UpstreamOnly

Equals to LOOK_UPSTREAM_ONLY.
Field: EZ_B.Video.DirectShow.Internals.FindDirection.DownstreamOnly

Equals to LOOK_DOWNSTREAM_ONLY.
Type: EZ_B.Video.DirectShow.Internals.Win32

Some Win32 API used internally.
Method: EZ_B.Video.DirectShow.Internals.Win32.CreateBindCtx(*System.Int32, System.Runtime.InteropServices.ComTypes.IBindCtx@*)

Supplies a pointer to an implementation of IBindCtx (a bind context object).
This object stores information about a particular moniker-binding operation.
Method: EZ_B.Video.DirectShow.Internals.Win32.MkParseDisplayName(*System.Runtime.InteropServices.ComTypes.IBindCtx, System.String, System.Int32@, System.Runtime.InteropServices.ComTypes.IMoniker@*)

Converts a string into a moniker that identifies the object named by the string.
Method: EZ_B.Video.DirectShow.Internals.Win32.memcpy(*System.Byte*, System.Byte*, System.Int32*)

Copy a block of memory.
Method: EZ_B.Video.DirectShow.Internals.Win32.OleCreatePropertyFrame(*System.IntPtr, System.Int32, System.Int32, System.String, System.Int32, System.Object@, System.Int32, System.IntPtr, System.Int32, System.Int32, System.IntPtr*)

Invokes a new property frame, that is, a property sheet dialog box.
Type: EZ_B.Video.DirectShow.PhysicalConnectorType

Specifies the physical type of pin (audio or video).
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.Default

Default value of connection type. Physically it does not exist, but just either to specify that connection type should not be changed (input) or was not determined (output).
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoTuner

Specifies a tuner pin for video.

Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoComposite

Specifies a composite pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoSVideo

Specifies an S-Video (Y/C video) pin.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoRGB

Specifies an RGB pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoYRYBY

Specifies a YRYBY (Y, Râ€“Y, Bâ€“Y) pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoSerialDigital

Specifies a serial digital pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoParallelDigital

Specifies a parallel digital pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoSCSI

Specifies a SCSI (Small Computer System Interface) pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoAUX

Specifies an AUX (auxiliary) pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.Video1394

Specifies an IEEE 1394 pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoUSB

Specifies a USB (Universal Serial Bus) pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoDecoder

Specifies a video decoder pin.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoEncoder

Specifies a video encoder pin.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoSCART

Specifies a SCART (Peritel) pin for video.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.VideoBlack

Not used.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioTuner

Specifies a tuner pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioLine

Specifies a line pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioMic

Specifies a microphone pin.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioAESDigital

Specifies an AES/EBU (Audio Engineering Society/European Broadcast Union) digital pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioSPDIFDigital

Specifies an S/PDIF (Sony/Philips Digital Interface Format) digital pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioSCSI

Specifies a SCSI pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioAUX

Specifies an AUX pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.Audio1394

Specifies an IEEE 1394 pin for audio.
Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioUSB

Specifies a USB pin for audio.

Field: EZ_B.Video.DirectShow.PhysicalConnectorType.AudioDecoder

Specifies an audio decoder pin.
Type: EZ_B.Video.DirectShow.FilterCategory

DirectShow filter categories.
Field: EZ_B.Video.DirectShow.FilterCategory.AudioInputDevice

Audio input device category.
Field: EZ_B.Video.DirectShow.FilterCategory.VideoInputDevice

Video input device category.
Field: EZ_B.Video.DirectShow.FilterCategory.VideoCompressorCategory

Video compressor category.
Field: EZ_B.Video.DirectShow.FilterCategory.AudioCompressorCategory

Audio compressor category
Type: EZ_B.Video.DirectShow.VideoCapabilities

Capabilities of video device such as frame size and frame rate.
Field: EZ_B.Video.DirectShow.VideoCapabilities.FrameSize

Frame size supported by video device.
Field: EZ_B.Video.DirectShow.VideoCapabilities.AverageFrameRate

Average frame rate of video device for corresponding frame size.
Field: EZ_B.Video.DirectShow.VideoCapabilities.MaximumFrameRate

Maximum frame rate of video device for corresponding frame size.
Field: EZ_B.Video.DirectShow.VideoCapabilities.BitCount

Number of bits per pixel provided by the camera.
Method: EZ_B.Video.DirectShow.VideoCapabilities.Equals(*System.Object*)

Check if the video capability equals to the specified object.
Method: EZ_B.Video.DirectShow.VideoCapabilities.Equals(*EZ_B.Video.DirectShow.VideoCapabilities*)

Check if two video capabilities are equal.
Method: EZ_B.Video.DirectShow.VideoCapabilities.GetHashCode

Get hash code of the object.
Method: EZ_B.Video.DirectShow.VideoCapabilities.op_Equality(*EZ_B.Video.DirectShow.VideoCapabilities, EZ_B.Video.DirectShow.VideoCapabilities*)

Equality operator.
Method: EZ_B.Video.DirectShow.VideoCapabilities.op_Inequality(*EZ_B.Video.DirectShow.VideoCapabilities, EZ_B.Video.DirectShow.VideoCapabilities*)

Inequality operator.
Type: EZ_B.Video.DirectShow.VideoInput

Video input of a capture board.
Field: EZ_B.Video.DirectShow.VideoInput.Index

Index of the video input.
Field: EZ_B.Video.DirectShow.VideoInput.Type

Type of the video input.

FFMPEGUtils

Field: EZ_B.FFMPEGUtils.VideoCodec.Default

Default video codec, which FFmpeg library selects for the specified file format.
Field: EZ_B.FFMPEGUtils.VideoCodec.MPEG4

MPEG-4 part 2.
Field: EZ_B.FFMPEGUtils.VideoCodec.WMV1

Windows Media Video 7.
Field: EZ_B.FFMPEGUtils.VideoCodec.WMV2

Windows Media Video 8.
Field: EZ_B.FFMPEGUtils.VideoCodec.MSMPEG4v2

MPEG-4 part 2 Microsoft variant version 2.
Field: EZ_B.FFMPEGUtils.VideoCodec.MSMPEG4v3

MPEG-4 part 2 Microsoft variant version 3.
Field: EZ_B.FFMPEGUtils.VideoCodec.H263P

H.263+ / H.263-1998 / H.263 version 2.
Field: EZ_B.FFMPEGUtils.VideoCodec.FLV1

Flash Video (FLV) / Sorenson Spark / Sorenson H.263.
Field: EZ_B.FFMPEGUtils.VideoCodec.MPEG2

MPEG-2 part 2.
Field: EZ_B.FFMPEGUtils.VideoCodec.Raw

Raw (uncompressed) video.

VideoPlayer

Event: EZ_B.VideoPlayer.OnPlayingFrame

Event risen for every frame. Returns the current frame number
Event: EZ_B.VideoPlayer.OnBeginPlaying

Event risen from a video begins playing
Event: EZ_B.VideoPlayer.OnCompleted

Event risen when video is completed

Vuzix

Event: EZ_B.Vuzix.OnConnected

Event when connected to device
Event: EZ_B.Vuzix.OnDisconnected

Event when disconnected from device

MMA7455

Method: EZ_B.MMA7455.WhoAmI

Return the firmware of the device
Method: EZ_B.MMA7455.Init(*EZ_B.MMA7455.SensitivityEnum*)

Send initialization
Method: EZ_B.MMA7455.GetMode

Return the current configuration
Method: EZ_B.MMA7455.GetX

Get X
Method: EZ_B.MMA7455.GetY

Get Y
Method: EZ_B.MMA7455.GetZ

Get Z

SureDualAxisCompass

Field: EZ_B.SureDualAxisCompass.MinPoolTimeMS

To prevent requests from flooding the communication channel, this limit prevents too many calls. Best to leave it alone.
Method: EZ_B.SureDualAxisCompass.SetCoil

Init the coil. Should be called as init one time
Method: EZ_B.SureDualAxisCompass.ResetCoil

Reset the Compass Coil
Method: EZ_B.SureDualAxisCompass.Update

Updates CompassData object with the current magnetic co-ordinates of the DC-SS503 Compass Module

MP3Trigger

Field: EZ_B.MP3Trigger.CommunicationPort

Specify the communication port that the MP3 Trigger is connected with
Field: EZ_B.MP3Trigger.BaudRate

Specify the baud rate that the MP3 Trigger is connected with.
Default is 38400
Method: EZ_B.MP3Trigger.Reverse

Play previous track
Method: EZ_B.MP3Trigger.Forward

Play next track
Method: EZ_B.MP3Trigger.SetVolume(*System.Byte*)

Specify volume.
0 - Loud.
255 - Quiet.
Method: EZ_B.MP3Trigger.StartStop

Start/Stop

Method: EZ_B.MP3Trigger.PlayTrack(*System.Byte*)

Play specified track number

I2C

Method: EZ_B.I2C.WriteBinary(*System.Byte, System.Byte, System.Byte, System.Byte, System.Byte, System.Byte, System.Byte, System.Byte, System.Byte*)

Write binary to the specified 7 bit address.
Example: WriteBinary(0x1D, 0, 0, 1, 1, 1, 0, 1, 0);
Method: EZ_B.I2C.Write(*System.Byte, System.Byte[]*)

Write data to the specified device 7 bit address.
Example: Write(0x1D, new byte [] { 127, 64 });
Method: EZ_B.I2C.ReadByte(*System.Byte*)

Read data from the specified i2c device address.
Example: byte ret = ReadByte(0x1D);
Method: EZ_B.I2C.Read(*System.Byte, System.Byte*)

Read data from the specified i2c device address.
Example: byte [] ret = Read(0x1D, 1);
Method: EZ_B.I2C.SetClockSpeed(*System.UInt32*)

Set the clock speed of the i2c interface

ObjectLocation

Type: EZ_B.ObjectLocation

This class is returned by the GetObjectLocation method.
This class will contain information regarding the location of an object, if found.
Field: EZ_B.ObjectLocation.ShapeType

The type of shape detected
Field: EZ_B.ObjectLocation.TrackingType

The type of tracking for this object
Field: EZ_B.ObjectLocation.Glyph

If Shape detect was used, this contains the glyph found
Field: EZ_B.ObjectLocation.Rotation

If supported, the rotation of the detected object
Field: EZ_B.ObjectLocation.VerticalLocation

The vertical location of the object, if found.
Field: EZ_B.ObjectLocation.ObjectName

The name of the object from the custom object detection
Field: EZ_B.ObjectLocation.QRCodeText

The QR Code text if detected
Field: EZ_B.ObjectLocation.HorizontalLocation

The horizontal location of the object, if found.
Field: EZ_B.ObjectLocation.ColorName

If custom color tracking is used, this will contain the name of the color that was detected
Field: EZ_B.ObjectLocation.Rect

Contains the rectangle of the detected object

EZB

Type: EZ_B.EZB

Communication protocol driver to EZ-B's
Type: EZ_B.EZB.ProtocolVersionEnum

The protocol version being used.
Field: EZ_B.EZB.IgnoreProtocolVersionError

If there is a protocol version error? This allows the connection routine to ignore it.
Field: EZ_B.EZB.ConnectedEndPointAddress

The address of the current connection.
If this is an EZ-B v3, this will contain the com port.
If this is an EZ-B v4, this will contain the physical address.
Type: EZ_B.EZB.OnConnectionChangeHandler

Event risen when there is a connection change
Type: EZ_B.EZB.OnConnectionChangeHandler2

Event risen when there is a connection change and sends itself
Event: EZ_B.EZB.OnConnectionChange2

Event risen when there is a connection change

Event: EZ_B.EZB.OnConnectionChange

Event risen when there is a connection change
Type: EZ_B.EZB.OnDataSendHandler

Event risen when data is sent to the ez-b
Event: EZ_B.EZB.OnDataSend

Event risen when data is sent to the ez-b
Type: EZ_B.EZB.OnDataReturnHandler

Event risen when data is received from the ez-b
Event: EZ_B.EZB.OnDataReceive

Event risen when data is received from the ez-b
Type: EZ_B.EZB.OnLogHandler

Event risen when there is debug data
Event: EZ_B.EZB.OnLog

Event risen when there is debug data
Field: EZ_B.EZB._Serial

This is the underlying bluetooth connection to the EZ-B.
This is for advanced users.
Field: EZ_B.EZB.Uart

Send serial commands from any digital port
Field: EZ_B.EZB.Servo

Servo commands. Control regular and modified servos.
Field: EZ_B.EZB.ADC

Analog To Digital Convertor (ADC) commands. Read voltages and values from the ADC Ports of the EZ-B
Field: EZ_B.EZB.Digital

Commands to read and write digital ports on the EZ-B
Field: EZ_B.EZB.SpeechSynth

Commands to have the computer speak and recognize voice commands
Field: EZ_B.EZB.HC_SR04

Commands to get the distance from a HC-SR04 Ping Sensor
Field: EZ_B.EZB.I2C

Send a I2C command out of the I2C interface
Field: EZ_B.EZB.Recorder

Allows recording and replaying of communication between the computer and EZ-B
Field: EZ_B.EZB.ConfigurationManager

Set hardware device settings
Field: EZ_B.EZB.TCPServer

Allows remote connectivity from other EZ_B DLL instances
Field: EZ_B.EZB.PWM

Control PWM (Pulse Wave Modulation) output
Field: EZ_B.EZB.SoundV4

Sound beep test for the v4
Field: EZ_B.EZB.EZBv4Manager

Manages settings specific to the EZ-B v4
Field: EZ_B.EZB.RGBEyes

Helper Class for controlling the RGB LED Eyes that ships with JD, and can be purchased optionally seperate
Field: EZ_B.EZB.MusicSynth

Helper class for making synthesized music on the ez-b v4 speaker
Method: EZ_B.EZB.GetDateTimeAsFormattedString(*System.DateTime*)

Return the specified date time as a formatted string which is standard across all ARC log usages
Method: EZ_B.EZB.GetDateTimeAsFormattedString

Return the current date time as a formatted string which is standard across all ARC log usages
Method: EZ_B.EZB.Log(*System.Boolean, System.String, System.Object[]*)

Manually send text to the log event
Method: EZ_B.EZB.GetAvailableCommunicationPorts(*System.Boolean*)

Get all communication ports. One of these should be connected to the EZ-B via Bluetooth
Method: EZ_B.EZB.GetProtocolVersionRaw

Returns the raw byte that reflects the firmware returned by the current connected EZ-B.
Do not use this, use the GetFirmwareVersion() instead.

Method: EZ_B.EZB.GetProtocolVersion

Return the firmware version in a string of the EZ-B
Method: EZ_B.EZB.GetProtocolVersionEnum

Return the firmware version as the enum
Method: EZ_B.EZB.PingController

Sends a ping request to the EZ-B to see if it's still responding. Returns a True if so, false if it isn't
Method: EZ_B.EZB.StopServer

Stop the server which is listening for incoming TCP connections from an EZ-B

Method: EZ_B.EZB.StartServer(*System.Int32*)

Start the server which listens for incoming TCP connections from an EZ-B
Method: EZ_B.EZB.Connect(*System.String*)

Connect to an EZ-B.
Remote can be a PORT: Get the port name from GetAvailableCommunicationPorts()
Remote can be an IP Address, example: 192.168.1.5:23
Method: EZ_B.EZB.Connect(*System.String, System.Int32*)

Connect to an EZ-B.
1) Hostname can be a communication PORT. Get the port name from GetAvailableCommunicationPorts()
2) Hostname can be an IP Address, example: 192.168.1.5:23
3) Baudrate is not used for TCP connections
Method: EZ_B.EZB.Disconnect

Disconnect from the EZ-B
Method: EZ_B.EZB.SendCommandData(*System.Int32, System.Byte[]*)

Send raw data to the ez-b
Method: EZ_B.EZB.ShowDebugWindow

Opens a debug window with diagnostic information
Method: EZ_B.EZB.GetRandomNumber(*System.Int32, System.Int32*)

Return a random number within specified range.
Using this random number generating function will provide a common seed.
Method: EZ_B.EZB.GetRandomUniqueNumber(*System.Int32, System.Int32*)

Return a random number and tries to make the returned value unique from the last time this function was called.
Method: EZ_B.EZB.GetUniqueIDBytes

Returns a byte array unique ID of the EZ-B v4
Method: EZ_B.EZB.LogFirmwareDetails(*EZ_B.Firmware.FirmwareCls*)

Send the firmware details and capabilities to the log method of this ezb.
Any events assigned to listen to the log update will receive the firmware details.
Method: EZ_B.EZB.GetUniqueIDString

Returns a byte array unique ID of the EZ-B v4

HC_SR04

Field: EZ_B.HC_SR04.MinPoolTimeMS

To prevent ADC requests from flooding the communication channel, this limit prevents too many calls. Best to leave it alone.
Method: EZ_B.HC_SR04.GetValue(*EZ_B.Digital.DigitalPortEnum, EZ_B.Digital.DigitalPortEnum*)

Get the value received from the HC-SR04 Ping Sensor

Digital

Type: EZ_B.Digital.DigitalPortEnum

List of Digital Ports
Method: EZ_B.Digital.SetDigitalPort(*EZ_B.Classes.DigitalItem[]*)

Set the status of many digital ports. TRUE will output +5 on v3, and +3.3 on v4, FALSE will short to GND
Method: EZ_B.Digital.SetDigitalPort(*EZ_B.Digital.DigitalPortEnum, System.Boolean*)

Set the status of a digital ports. TRUE will output +5 on v3, and +3.3 on v4, FALSE will short to GND
Method: EZ_B.Digital.GetLastDigitalPortSet(*EZ_B.Digital.DigitalPortEnum*)

Does not query the EZ-B Controller. This returns the status of the port after you had SetDigitalPort().
Method: EZ_B.Digital.Toggle(*EZ_B.Digital.DigitalPortEnum*)

Toggles the status of a digital port and returns the new status
Method: EZ_B.Digital.GetDigitalPort(*EZ_B.Digital.DigitalPortEnum*)

Query the status of a digital port.
Method: EZ_B.Digital.GetDigitalPortAsInt(*EZ_B.Digital.DigitalPortEnum*)

Query the status of a digital port as an Integer (0 false, 1 true)

UCEZB_Connect

Type: EZ_B.UCEZB_Connect

User Control for connecting to an EZ-B
Event: EZ_B.UCEZB_Connect.OnConnection

Event executed when Connection to EZ-B is established.
Event: EZ_B.UCEZB_Connect.OnDisconnect

Event executed when Connection to EZ-B is lost.

Method: EZ_B.UCEZB_Connect.InitButtonColors

Change the color of the connect button based on the connection status
Method: EZ_B.UCEZB_Connect.RefreshPortList

Refresh the list of ports if not connected
Method: EZ_B.UCEZB_Connect.Connect(*System.Boolean*)

Manually connect to the specified communication port. The optional parameter allows you to override if an error dialog is displayed
Method: EZ_B.UCEZB_Connect.Disconnect

Manually disconnect

Functions

Method: EZ_B.Functions.DisplayBitSequence(*System.Int32*)

Displays the bit sequence of an integer value.
Method: EZ_B.Functions.SetBitValue(*System.Int32, System.Int32*)

Sets the bit in an integer value at the requested position.
Method: EZ_B.Functions.ClearBitValue(*System.Int32, System.Int32*)

Clears the bit in an integer value at the requested position.
Method: EZ_B.Functions.FlipBitValue(*System.Int32, System.Int32*)

Flips the bit in an integer value at the requested position.
Method: EZ_B.Functions.ConvertStringToByteArray(*System.String*)

Converts the string to a byte array containing the ASCII values of each char.
Method: EZ_B.Functions.ByteArrayToHexString(*System.Byte[]*)

Converts an array of byte to a string of hex separated by whitespace (i.e. 0x55 0x82 0x20)
Method: EZ_B.Functions.ConvertByteArrayToString(*System.Byte[]*)

Converts the byte array to a string.
Method: EZ_B.Functions.ConvertByteArrayToString(*System.Byte[], System.Int32*)

Converts the byte array to a string.
Method: EZ_B.Functions.ConvertToDecimal(*System.Object*)

Convert ascii object to a decimal value
Method: EZ_B.Functions.Chunk``1( ``*0[], System.Int32*)

Returns an IEnumerable of input list split into the number of specified parts
Method: EZ_B.Functions.IsByte(*System.Object*)

Returns true if the InObj is a byte value
Method: EZ_B.Functions.IsNumeric(*System.Object*)

Returns true if the InObj is a numerical value (including int and floating point)
Method: EZ_B.Functions.IsLargerThan(*System.Int32, System.Int32[]*)

Returns true if the mainValue is larger than all other values
Method: EZ_B.Functions.IsEqualToo(*System.Int32, System.Int32[]*)

Returns true if the mainValue is equal to any other values
Method: EZ_B.Functions.IsEqualToo(*System.Char, System.Char[]*)

Returns true if the mainValue is equal to any other values
Method: EZ_B.Functions.CompareColors(*System.Drawing.Color, System.Drawing.Color*)

Compares Color A from Color B and returns the difference
Method: EZ_B.Functions.WithinRange(*System.Int32, EZ_B.Functions.Range*)

Returns true if the number falls within the high and low range
Method: EZ_B.Functions.WithinRange(*System.Decimal, EZ_B.Functions.Range*)

Returns true if the number falls within the high and low range
Method: EZ_B.Functions.WithinRange(*System.Double, EZ_B.Functions.Range*)

Returns true if the number falls within the high and low range
Method: EZ_B.Functions.EndsWith(*System.Boolean, System.Object, System.String[]*)

Extension of the String.EndsWith but allows an array of items to check for rather than just one.

Method: EZ_B.Functions.EndsWithNumber(*System.String*)

Check if the last characters of the input string are numeric
Method: EZ_B.Functions.GetNumberAtEndOfString(*System.String*)

Get the numbers at the end of a string
Method: EZ_B.Functions.Diff(*System.Int32, System.Int32, System.Int32*)

Returns true if the difference between Master and Compare is greater then Diff
Method: EZ_B.Functions.Diff(*System.Decimal, System.Decimal, System.Decimal*)

Returns true if the difference between Master and Compare is greater then Diff
Method: EZ_B.Functions.IsBitSet(*System.Int32, System.Int32*)

Returns true if the specified bit in the byte is 1. false if not. 0 is LSB, 7 is MSB
Method: EZ_B.Functions.ByteToBinaryString(*System.Byte*)

Converts a byte to a binary string
Method: EZ_B.Functions.ByteToBinaryString(*System.Byte, System.String*)

Converts a byte to a binary string
Method: EZ_B.Functions.ToByteFromBinary(*System.Boolean, System.Boolean, System.Boolean, System.Boolean, System.Boolean, System.Boolean, System.Boolean, System.Boolean*)

Returns a byte from specified binary. LSB is val0. MSB is val7
Method: EZ_B.Functions.ToByteFromBinary(*System.Int32, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32*)

Returns a byte out of the binary. The inputs for each bit an either be a 0 or a 1. The LSB is val0. MSB is val7
Method: EZ_B.Functions.GetScalarFromRange(*System.Int32, System.Single, System.Single*)

Returns a scalar. Used for converting one range into another range. (i.e. Wii Input Remote X/Y/Z to Servo Positions)
Method: EZ_B.Functions.GetScalarFromRange(*System.Int32, System.Int32, System.Int32*)

Returns a scalar. Used for converting one range into another range. (i.e. Wii Input Remote X/Y/Z to Servo Positions)
Method: EZ_B.Functions.GetScalarFromRange(*System.Byte, System.Byte, System.Byte*)

Returns a scalar. Used for converting one range into another range. (i.e. Wii Input Remote X/Y/Z to Servo Positions)
Method: EZ_B.Functions.SingleToInt32Bits(*System.Single*)

Converts a Float to an IEEE754 Compliant Integer
Method: EZ_B.Functions.GetShortestAngle(*System.Int32, System.Int32*)

Returns the shortest angle between two angles (Absolute, no negatives)
Method: EZ_B.Functions.GetAngle(*System.Int32, System.Int32, System.Int32, System.Int32*)

Get the angle of the second point relative to the first point
Method: EZ_B.Functions.GetAngle(*System.Decimal, System.Decimal, System.Decimal, System.Decimal*)

Get the angle of the second point relative to the first point
Method: EZ_B.Functions.GetAngle(*System.Double, System.Double, System.Double, System.Double*)

Get the angle of the second point relative to the first point
Method: EZ_B.Functions.GetDistance(*System.Int32, System.Int32, System.Int32, System.Int32*)

Returns the distance between two points on a 2d vector
Method: EZ_B.Functions.StripHTML(*System.String*)

Remove all html tags
Method: EZ_B.Functions.CopyTo(*System.Object, System.Object*)

Copy object to object
Method: EZ_B.Functions.HsvToRgb(*System.Double, System.Double, System.Double*)

Convert HSV to RGB Color
h is from 0-360
s,v values are 0-1
r,g,b values are 0-255
Method: EZ_B.Functions.ClampDouble(*System.Double, System.Double, System.Double*)

Clamp a value
Method: EZ_B.Functions.Clamp(*System.Int32, System.Int32, System.Int32*)

Clamp a value to range
Method: EZ_B.Functions.Clamp(*System.Single, System.Single, System.Single*)

Clamp a value to range
Method: EZ_B.Functions.Clamp(*System.Int32*)

Clamp a value to 0-255
Method: EZ_B.Functions.DegX(*System.Double, System.Double*)

Returns the degree X co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Cos(deg + 90 / (180 / System.Math.PI)) * radius
Method: EZ_B.Functions.DegX2(*System.Double, System.Double*)

Returns the degree X co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Cos(deg - 90 * Math.PI / 180) * radius
Method: EZ_B.Functions.DegX2(*System.Single, System.Single*)

Returns the degree X co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Cos(deg -90 * Math.PI / 180) * radius;

Method: EZ_B.Functions.DegY(*System.Double, System.Double*)

Returns the degree Y co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Sin(deg + 90 / (180 / System.Math.PI)) * radius;
Method: EZ_B.Functions.DegY2(*System.Double, System.Double*)

Returns the degree Y co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Sin(deg -90 * Math.PI / 180) * radius
Method: EZ_B.Functions.DegY2(*System.Single, System.Single*)

Returns the degree Y co-ordinate for a circle
i.e. Plot(DegX(10, 20), DegY(10, 20));
System.Math.Sin(deg -90 * Math.PI / 180) * radius
Method: EZ_B.Functions.GetNetworkIPAddressCount(*System.String*)

Returns how many network instances of IP Address or the first part of it exist (i.e. are more than one network interface using the same IP network)
This uses StartsWith(specified network)
So you can search for 192.168.1. to see if any ip address exists within that network

Servo

Event: EZ_B.Servo.OnServoGetPosition

Event that is raised when reading the positions of servos that support reading positions
Contains the servo of the requesting position
Event: EZ_B.Servo.OnServoMove

Event that is raised when a servo is moved.
Contains the servo positions specified by the user/control and limited by MIN/MAX limits
Event: EZ_B.Servo.OnServoSpeed

Event that is raised when a servo speed setting is changed
Event: EZ_B.Servo.OnServoVelocity

Event that is raised when a servo velocity setting is changed
Event: EZ_B.Servo.OnServoAcceleration

Event that is raised when a servo acceleration setting is changed
Event: EZ_B.Servo.OnServoRelease

Event that is raised when a servo is released
Field: EZ_B.Servo.SERVO_SPEED_FASTEST

The slowest speed for a servo (0)
Field: EZ_B.Servo.SERVO_SPEED_SLOWEST

The slowest speed for a servo (20)
Field: EZ_B.Servo.SERVO_MAX

The maximum value for a servo (default: 180)
Field: EZ_B.Servo.SERVO_MIN

The minimum value of a servo (1)
Field: EZ_B.Servo.SERVO_OFF

The value of a servo to disable
Type: EZ_B.Servo.ServoPortEnum

List of Servo Ports
Method: EZ_B.Servo.ResetServoFineTune

Reset the fine tuning values to 0 for each servo
Method: EZ_B.Servo.GetServoFineTune(*EZ_B.Servo.ServoPortEnum*)

Return the fine tunign value of the specified servo
Method: EZ_B.Servo.SetServoFineTune(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the fine tuning value for the specified servo. This means that if the fine tune value for a servo is set to 1, then every position that is specified will be incremented by 1.
This allows you to fine tune a servo position across the entire application.
Method: EZ_B.Servo.ResetServoMinLimits

Reset the servo min value
Method: EZ_B.Servo.GetServoMin(*EZ_B.Servo.ServoPortEnum*)

Return the min value that this servo will ever move
Method: EZ_B.Servo.SetServoMin(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the mininum servo value that this servo will ever be able to go
Method: EZ_B.Servo.ResetServoMaxLimits

Reset the servo max value
Method: EZ_B.Servo.GetServoMax(*EZ_B.Servo.ServoPortEnum*)

Return the max value that this servo will ever move
Method: EZ_B.Servo.SetServoMax(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the mininum servo value that this servo will ever be able to go
Method: EZ_B.Servo.SetServoPosition(*EZ_B.Servo.ServoPortEnum, System.Int32, System.Int32*)

Set the speed and position of a servo
Method: EZ_B.Servo.SetServoPosition(*EZ_B.Servo.ServoPortEnum, System.Int32, System.Int32, System.Int32*)

Set the speed and velocity and position of a servo
Method: EZ_B.Servo.SetServoPosition(*EZ_B.Servo.ServoPortEnum, System.Int32, System.Int32, System.Int32, System.Int32*)

Set the speed and velocity and acceleaation and position of a servo
Method: EZ_B.Servo.SetServoPositionScalar(*EZ_B.Servo.ServoPortEnum, System.Int32, System.Int32, System.Int32, System.Int32, System.Int32, System.Boolean*)

Move the servo to the position based on the servo min and max position related/mapped to the client width min and max position position.
Method: EZ_B.Servo.SetServoPositionScalar(*EZ_B.Servo.ServoPortEnum, System.Int32, System.Int32, System.Single, System.Single, System.Single, System.Boolean*)

Move the servo to the position based on the servo min and max position related/mapped to the client width min and max position position.
Method: EZ_B.Servo.SetServoPosition(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the position of a servo
Method: EZ_B.Servo.SetServoPosition(*EZ_B.Classes.ServoPositionItem[]*)

Set the position of a servo.
Raises OnServoMoveDetailed, OnServoMoveDetailed2 and OnServoMove events.
Method: EZ_B.Servo.SetServoVelocity(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the velocity of a servo
Raises OnServoVelocity event unless a -1 (skip) is passed
Method: EZ_B.Servo.SetServoVelocity(*EZ_B.Servo.ServoPortEnum[], System.Int32*)

Set the velocity of multiple servos.
Raises OnServoVelocity event unless a -1 (skip) is passed
Method: EZ_B.Servo.GetServoVelocity(*EZ_B.Servo.ServoPortEnum*)

Return the current velocity of a servo
Method: EZ_B.Servo.SetServoAcceleration(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the acceleration of a servo
Raises OnServoAcceleration event unless a -1 (skip) is passed
Method: EZ_B.Servo.SetServoAcceleration(*EZ_B.Servo.ServoPortEnum[], System.Int32*)

Set the acceleration of multiple servos
Raises OnServoAcceleration event unless a -1 (skip) is passed
Method: EZ_B.Servo.GetServoAcceleration(*EZ_B.Servo.ServoPortEnum*)

Return the current acceleration of a servo
Method: EZ_B.Servo.SetServoSpeed(*EZ_B.Servo.ServoPortEnum, System.Int32*)

Set the speed of a servo
Raises OnServoSpeed event unless a -1 (skip) is passed
Method: EZ_B.Servo.SetServoSpeed(*EZ_B.Servo.ServoPortEnum[], System.Int32*)

Set the speed of multiple servos
Raises OnServoSpeed event unless a -1 (skip) is passed
Method: EZ_B.Servo.GetServoSpeed(*EZ_B.Servo.ServoPortEnum*)

Return the current speed of a servo
Method: EZ_B.Servo.GetServoPositionRealtime(*EZ_B.Servo.ServoPortEnum*)

Get the realtime position of a servo
If a servo supports bi-direction communication, this queries the servo
If the servo does not respond with success, the last known servo position is returned
Raises OnServoGetPosition event
Method: EZ_B.Servo.GetServoPosition(*EZ_B.Servo.ServoPortEnum*)

Get the position of a servo
Method: EZ_B.Servo.ReleaseServo(*EZ_B.Servo.ServoPortEnum*)

Release servo. Release a servo from holding its position.
Raises OnServoRelease event
Method: EZ_B.Servo.ReleaseServo(*EZ_B.Servo.ServoPortEnum[]*)

Release servo. Release a servo from holding its position.
Raises OnServoRelease event
Method: EZ_B.Servo.ReleaseAllServos

When servos have been used, they will hold their position until the EZ-B power is cycled or until they are told to release.
This will send a command to the EZ-B to release all servos
Raises OnServoRelease event
Method: EZ_B.Servo.ResetAllServoSpeeds

Reset all the servo speeds to their default of 0 (fastest)
Raises OnServoRelease event
Method: EZ_B.Servo.IsServoReleased(*EZ_B.Servo.ServoPortEnum*)

Return true if the specified servo port is in a released state
Method: EZ_B.Servo.GetNumberOfSecondsSinceLastMove(*EZ_B.Servo.ServoPortEnum*)

How long has it been since the last move of a servo

SpeechSynth

Type: EZ_B.SpeechSynth.SpeakingOutputTypeEnum

The type of output that the audio is being sent to.
This is because the OnSpeakingCompleted event is not executed when speaking to a stream (ie EZB).
Field: EZ_B.SpeechSynth.SpeakingOutputTypeEnum.Stream

This is outputing to an EZB or file. Therefore, the OnSpeakingCompleted event will not be raised
Field: EZ_B.SpeechSynth.SpeakingOutputTypeEnum.PC

This is outputing to a soundcard speaker. The OnSpeakingCompleted event will be raised
Event: EZ_B.SpeechSynth.OnAudioSignalProblem

When the audio stream has a problem
Event: EZ_B.SpeechSynth.OnPauseListeningStatusChanged

When the pause listening status has changed by calling PauseListening or ResumeListening
Event: EZ_B.SpeechSynth.OnPhraseRecognized

Event thrown when text is recognized.
The confidence value is between 0.00 and 1.00. The higher the number, the more confidence.
It's usually safe to trust confidence > 0.80
Text will be returned in lowercase!
Event: EZ_B.SpeechSynth.OnSpeaking

Event executed before text to speech is executed
The OnSpeakingCompleted event is not raised when Stream is the output type.
Use OnSpeaking2 to get the output type
Event: EZ_B.SpeechSynth.OnSpeaking2

Event executed before text to speech is executed that includes the output device type of the audio.
This is because the OnSpeakingCompleted event is not raised when Stream is the output type.
Event: EZ_B.SpeechSynth.OnSpeakingCompleted

Event executed when the speaking has completed if outputing to a soundcard
This event is not raised if outputing to an EZB
Event: EZ_B.SpeechSynth.OnAudioLevelChanged

Event is thrown when the audio level has changed from the default input audio device
Field: EZ_B.SpeechSynth.AudioLevel

The normalized level of the spoken audio from the input device
Method: EZ_B.SpeechSynth.PauseListening

Pause listening for speech recognition.
Any events assigned to recognized phrases will not be raised.
To re-enable listening and recognized events, call ResumeListening()
The OnPauseListeningStatusChanged event will raise when this is called
Method: EZ_B.SpeechSynth.SetVoice(*System.String*)

Set the active voice to be used when speaking
Method: EZ_B.SpeechSynth.ResumeListening

Resume listening for speech recognition if previously paused with PauseListening()
The OnPauseListeningStatusChanged event will raise when this is called
Method: EZ_B.SpeechSynth.GetInstalledCultures

Get the list of installed languages for recognition
Method: EZ_B.SpeechSynth.GetInstalledVoices

Get the list of installed voices within the system
Method: EZ_B.SpeechSynth.Say(*System.String*)

Say MSG to the default audio device
Method: EZ_B.SpeechSynth.SayWait(*System.String*)

Say MSG to the default audio device
Method: EZ_B.SpeechSynth.SayToStream(*System.String*)

Say MSG to memory stream in 8 bit, 16000, mono PCM.
Does not dispose the returned memorystream, so it's up to you.
Method: EZ_B.SpeechSynth.SayToFile(*System.String, System.String*)

Say MSG to the specified filename
Method: EZ_B.SpeechSynth.SayToBytes(*System.String*)

Say MSG and return the raw bytes in 8 bit, 16000, mono PCM
Method: EZ_B.SpeechSynth.SayStop

Stop speaking the current spoken message
Method: EZ_B.SpeechSynth.RaiseOnCompletedEvent

This will raise the On Speaking Completed Event to any robot skills that have subscribed to it.
Generally, speech recognition robot skills will use this event to upause listening after the
robot has spoken.
Use this method to resume all speech recognition robot skills to continue listening
by unpausing themselvers.
Method: EZ_B.SpeechSynth.RaiseOnSpeakingEvent

This will raise the On Speaking Event to any robot skills that have subscribed to it.
Generally, speech recognition robot skills will use this event to pause listening while
the robot is speaking.
Use this method to pause all speech recognition robot skills from listening.
Method: EZ_B.SpeechSynth.SetDictionaryOfPhrases(*System.String[]*)

Load the dictionary with custom recgonized phrases.
Method: EZ_B.SpeechSynth.AppendDictionaryOfPhrases(*System.String[]*)

Append to the dictionary with custom recgonized phrases.
Method: EZ_B.SpeechSynth.ClearDictionary

Clear the grammar dictionary. Use this before appending to the grammar dictionary for complex speech interaction.
Method: EZ_B.SpeechSynth.AppendDictionaryFromGrammar(*System.Speech.Recognition.GrammarBuilder*)

Appends to the existing grammar dictionary. Use this to build your own grammar rather for complex speech interaction
Method: EZ_B.SpeechSynth.SetDictionaryFromGrammar(*System.Speech.Recognition.GrammarBuilder*)

Sets the grammar dictionary. Use this to build your own grammar rather for complex speech interaction
Method: EZ_B.SpeechSynth.SetDictionaryToAllLocale

Load the dictionary with all known words for your locale.
Remember, there will be a lot of recognition errors if you do this.
It's always best to populate your own list of phrases using SetDictionaryOfPhrases() method.
Method: EZ_B.SpeechSynth.ListenForSpeechCommand(*System.Int32*)

Blocks and listens for a speech command.
Returns the text of the recognized speech.
Returns string.empty if timeout occures.
Requires Windows 7 or higher.
Method: EZ_B.SpeechSynth.StartListening

Start listening for voice recognition. PhraseRecognized event will be called with success.
Requires Windows 7 or higher
Method: EZ_B.SpeechSynth.StopListening

Disable listening for voice recognition.

TellyMate

Type: EZ_B.TellyMate.CmdEnum

List of TellyMate Commands
Type: EZ_B.TellyMate.FontAttribEnum

List of TellyMate Font Attributes
Method: EZ_B.TellyMate.SendText(*System.String*)

Send the text to a Tellymate on port D0 with optional carriage return
Method: EZ_B.TellyMate.SendText(*System.String, System.Boolean*)

Send the text to a Tellymate on port D0 with optional carriage return
Method: EZ_B.TellyMate.SendCommand(*EZ_B.TellyMate.CmdEnum*)

Sent a command to the TellyMate
Method: EZ_B.TellyMate.MoveCursor(*System.Int32, System.Int32*)

Move the cursor to specified position
Method: EZ_B.TellyMate.SetFontAttrib(*EZ_B.TellyMate.FontAttribEnum*)

Set the font attribute

Uart

Field: EZ_B.Uart.MaxUARTReceiveBuffer

This is the maximum size that the receive buffer from the EZB -> WIFI will be
Method: EZ_B.Uart.SetBaudClock(*EZ_B.Uart.BAUD_RATE_ENUM, System.Int32*)

Specify the clock delay between bytes in cycles of the EZ-B's 120mhz 32 Bit ARM processor. This would only need to be used to fine tune the baudrate timing if the connected device is not very accurate or requires a diffference in timing.
For example, some open-source hardware platforms use Software Serial drivers, which sometimes need a little bit of tweaking. Generally, you should never need to change these values.
However, there is a Custom labelled baudrate which you can change for specific speeds.
Anyone adjusting these speeds will need a logic analyzer, such as the Saleae Logic16 or Logic32
Method: EZ_B.Uart.SendSerial(*EZ_B.Digital.DigitalPortEnum, EZ_B.Uart.BAUD_RATE_ENUM, System.String*)

Send text over serial specified serial port at baud rate
Method: EZ_B.Uart.SendSerial(*EZ_B.Digital.DigitalPortEnum, EZ_B.Uart.BAUD_RATE_ENUM, System.Char[]*)

Send text over serial specified serial port at baud rate
Method: EZ_B.Uart.SendSerial(*EZ_B.Digital.DigitalPortEnum, EZ_B.Uart.BAUD_RATE_ENUM, System.Byte*)

Send text over serial specified serial port at baud rate
Method: EZ_B.Uart.SendSerial(*EZ_B.Digital.DigitalPortEnum, EZ_B.Uart.BAUD_RATE_ENUM, System.Byte[]*)

Send text over serial specified serial port at baud rate
Method: EZ_B.Uart.UARTExpansionInit(*System.Int32, System.UInt32*)

Initialize the ez-b v4 UART. this must be called before any other UARTExpansion function.
Method: EZ_B.Uart.UARTExpansionWrite(*System.Int32, System.Byte[]*)

Write to the EZ-B v4 Uart
Method: EZ_B.Uart.UARTExpansionAvailableBytes(*System.Int32*)

Read the number of bytes available in the EZ-B v4 uart buffer
Method: EZ_B.Uart.UARTExpansionRead(*System.Int32, System.Int32*)

return the specified number of bytes from the ez-b v4 uart input buffer
Method: EZ_B.Uart.UARTExpansionReadAvailable(*System.Int32*)

Read all available bytes from the UART on the EZ-B v4

SoundTouch

Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_createInstance

Create a new instance of SoundTouch processor.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_destroyInstance(*System.IntPtr*)

Destroys a SoundTouch processor instance.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_getVersionString2(*System.Text.StringBuilder, System.Int32*)

Get SoundTouch library version string - alternative function for
environments that can't properly handle character string as return value
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_getVersionId

Get SoundTouch library version Id
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setRate(*System.IntPtr, System.Single*)

Sets new rate control value. Normal rate = 1.0, smaller values
represent slower rate, larger faster rates.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setTempo(*System.IntPtr, System.Single*)

Sets new tempo control value. Normal tempo = 1.0, smaller values
represent slower tempo, larger faster tempo.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setRateChange(*System.IntPtr, System.Single*)

Sets new rate control value as a difference in percents compared
to the original rate (-50 .. +100 %);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setTempoChange(*System.IntPtr, System.Single*)

Sets new tempo control value as a difference in percents compared
to the original tempo (-50 .. +100 %);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setPitch(*System.IntPtr, System.Single*)

Sets new pitch control value. Original pitch = 1.0, smaller values
represent lower pitches, larger values higher pitch.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setPitchOctaves(*System.IntPtr, System.Single*)

Sets pitch change in octaves compared to the original pitch
(-1.00 .. +1.00);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setPitchSemiTones(*System.IntPtr, System.Single*)

Sets pitch change in semi-tones compared to the original pitch
(-12 .. +12);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setChannels(*System.IntPtr, System.UInt32*)

Sets the number of channels, 1 = mono, 2 = stereo
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setSampleRate(*System.IntPtr, System.UInt32*)

Sets sample rate.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_flush(*System.IntPtr*)

Flushes the last samples from the processing pipeline to the output.
Clears also the internal processing buffers.

Note: This function is meant for extracting the last samples of a sound
stream. This function may introduce additional blank samples in the end
of the sound stream, and thus it's not recommended to call this function
in the middle of a sound stream.

Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_putSamples(*System.IntPtr, System.Single[], System.Int32*)

Adds 'numSamples' pcs of samples from the 'samples' memory position into
the input of the object. Notice that sample rate _has_to_ be set before
calling this function, otherwise throws a runtime_error exception.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_clear(*System.IntPtr*)

Clears all the samples in the object's output and internal processing
buffers.

Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_setSetting(*System.IntPtr, VarispeedDemo.SoundTouch.SoundTouchSettings, System.Int32*)

Changes a setting controlling the processing system behaviour. See the
'SETTING_...' defines for available setting ID's.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_getSetting(*System.IntPtr, VarispeedDemo.SoundTouch.SoundTouchSettings*)

Reads a setting controlling the processing system behaviour. See the
'SETTING_...' defines for available setting ID's.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_numUnprocessedSamples(*System.IntPtr*)

Returns number of samples currently unprocessed.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_receiveSamples(*System.IntPtr, System.Single[], System.UInt32*)

Adjusts book-keeping so that given number of samples are removed from beginning of the
sample buffer without copying them anywhere.

Used to reduce the number of samples in the buffer when accessing the sample buffer directly
with 'ptrBegin' function.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_numSamples(*System.IntPtr*)

Returns number of samples currently available.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop32.soundtouch_isEmpty(*System.IntPtr*)

Returns nonzero if there aren't any samples available for outputting.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_createInstance

Create a new instance of SoundTouch processor.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_destroyInstance(*System.IntPtr*)

Destroys a SoundTouch processor instance.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_getVersionString2(*System.Text.StringBuilder, System.Int32*)

Get SoundTouch library version string - alternative function for
environments that can't properly handle character string as return value
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_getVersionId

Get SoundTouch library version Id
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setRate(*System.IntPtr, System.Single*)

Sets new rate control value. Normal rate = 1.0, smaller values
represent slower rate, larger faster rates.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setTempo(*System.IntPtr, System.Single*)

Sets new tempo control value. Normal tempo = 1.0, smaller values
represent slower tempo, larger faster tempo.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setRateChange(*System.IntPtr, System.Single*)

Sets new rate control value as a difference in percents compared
to the original rate (-50 .. +100 %);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setTempoChange(*System.IntPtr, System.Single*)

Sets new tempo control value as a difference in percents compared
to the original tempo (-50 .. +100 %);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setPitch(*System.IntPtr, System.Single*)

Sets new pitch control value. Original pitch = 1.0, smaller values
represent lower pitches, larger values higher pitch.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setPitchOctaves(*System.IntPtr, System.Single*)

Sets pitch change in octaves compared to the original pitch
(-1.00 .. +1.00);
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setPitchSemiTones(*System.IntPtr, System.Single*)

Sets pitch change in semi-tones compared to the original pitch
(-12 .. +12);

Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setChannels(*System.IntPtr, System.UInt32*)

Sets the number of channels, 1 = mono, 2 = stereo
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setSampleRate(*System.IntPtr, System.UInt32*)

Sets sample rate.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_flush(*System.IntPtr*)

Flushes the last samples from the processing pipeline to the output.
Clears also the internal processing buffers.

Note: This function is meant for extracting the last samples of a sound
stream. This function may introduce additional blank samples in the end
of the sound stream, and thus it's not recommended to call this function
in the middle of a sound stream.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_putSamples(*System.IntPtr, System.Single[], System.Int32*)

Adds 'numSamples' pcs of samples from the 'samples' memory position into
the input of the object. Notice that sample rate _has_to_ be set before
calling this function, otherwise throws a runtime_error exception.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_clear(*System.IntPtr*)

Clears all the samples in the object's output and internal processing
buffers.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_setSetting(*System.IntPtr, VarispeedDemo.SoundTouch.SoundTouchSettings, System.Int32*)

Changes a setting controlling the processing system behaviour. See the
'SETTING_...' defines for available setting ID's.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_getSetting(*System.IntPtr, VarispeedDemo.SoundTouch.SoundTouchSettings*)

Reads a setting controlling the processing system behaviour. See the
'SETTING_...' defines for available setting ID's.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_numUnprocessedSamples(*System.IntPtr*)

Returns number of samples currently unprocessed.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_receiveSamples(*System.IntPtr, System.Single[], System.UInt32*)

Adjusts book-keeping so that given number of samples are removed from beginning of the
sample buffer without copying them anywhere.

Used to reduce the number of samples in the buffer when accessing the sample buffer directly
with 'ptrBegin' function.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_numSamples(*System.IntPtr*)

Returns number of samples currently available.
Method: VarispeedDemo.SoundTouch.SoundTouchInterop64.soundtouch_isEmpty(*System.IntPtr*)

Returns nonzero if there aren't any samples available for outputting.
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.UseAaFilter

Available setting IDs for the 'setSetting' and 'get_setting' functions.
Enable/disable anti-alias filter in pitch transposer (0 = disable)
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.AaFilterLength

Pitch transposer anti-alias filter length (8 .. 128 taps, default = 32)
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.UseQuickSeek

Enable/disable quick seeking algorithm in tempo changer routine
(enabling quick seeking lowers CPU utilization but causes a minor sound
quality compromising)
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.SequenceMs

Time-stretch algorithm single processing sequence length in milliseconds. This determines
to how long sequences the original sound is chopped in the time-stretch algorithm.
See "STTypes.h" or README for more information.
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.SeekWindowMs

Time-stretch algorithm seeking window length in milliseconds for algorithm that finds the
best possible overlapping location. This determines from how wide window the algorithm
may look for an optimal joining location when mixing the sound sequences back together.
See "STTypes.h" or README for more information.
Field: VarispeedDemo.SoundTouch.SoundTouchSettings.OverlapMs

Time-stretch algorithm overlap length in milliseconds. When the chopped sound sequences
are mixed back together, to form a continuous sound stream, this parameter defines over
how long period the two consecutive sequences are let to overlap each other.
See "STTypes.h" or README for more information.

Imaging

Type: FaceDetect.Imaging.IntegralImage2

Joint representation of both Integral Image and Squared Integral Image.
Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*System.Drawing.Bitmap, System.Int32*)

Constructs a new Integral image from a Bitmap image.

Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*System.Drawing.Bitmap, System.Int32, System.Boolean*)

Constructs a new Integral image from a Bitmap image.
Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*System.Drawing.Imaging.BitmapData, System.Int32*)

Constructs a new Integral image from a BitmapData image.
Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*System.Drawing.Imaging.BitmapData, System.Int32, System.Boolean*)

Constructs a new Integral image from a BitmapData image.
Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*AForge.Imaging.UnmanagedImage, System.Int32*)

Constructs a new Integral image from an unmanaged image.
Method: FaceDetect.Imaging.IntegralImage2.FromBitmap(*AForge.Imaging.UnmanagedImage, System.Int32, System.Boolean*)

Constructs a new Integral image from an unmanaged image.
Method: FaceDetect.Imaging.IntegralImage2.GetSum(*System.Int32, System.Int32, System.Int32, System.Int32*)

Gets the sum of the pixels in a rectangle of the Integral image.
Method: FaceDetect.Imaging.IntegralImage2.GetSum2(*System.Int32, System.Int32, System.Int32, System.Int32*)

Gets the sum of the squared pixels in a rectangle of the Integral image.
Method: FaceDetect.Imaging.IntegralImage2.GetSumT(*System.Int32, System.Int32, System.Int32, System.Int32*)

Gets the sum of the pixels in a tilted rectangle of the Integral image.
Method: FaceDetect.Imaging.IntegralImage2.Finalize

Releases unmanaged resources and performs other cleanup operations
before the is reclaimed by garbage collection.
Method: FaceDetect.Imaging.Tools.IsEqual(*System.Drawing.Rectangle, System.Drawing.Rectangle, System.Int32*)

Compares two rectangles for equality, considering an acceptance threshold.

Vision

Type: FaceDetect.Vision.Detection.Cascades.FaceHaarCascade

Default Face Haar Cascade for using with Haar Classifiers.
Type: FaceDetect.Vision.Detection.Cascades.NoseHaarCascade

Automatic transcription of haar cascade definitions
for facial features by Modesto Castrillon-Santana.
Type: FaceDetect.Vision.Detection.HaarCascade

Cascade of Haar-like features' weak classification stages.
Method: FaceDetect.Vision.Detection.HaarCascade.checkTiltedFeatures(*FaceDetect.Vision.Detection.HaarCascadeStage[]*)

Checks if the classifier contains tilted (rotated) features
Method: FaceDetect.Vision.Detection.HaarCascade.Clone

Creates a new object that is a copy of the current instance.
Method: FaceDetect.Vision.Detection.HaarCascade.FromXml(*System.IO.Stream*)

Loads a HaarCascade from a OpenCV-compatible XML file.
Method: FaceDetect.Vision.Detection.HaarCascade.FromXml(*System.String*)

Loads a HaarCascade from a OpenCV-compatible XML file.
Method: FaceDetect.Vision.Detection.HaarCascade.FromXml(*System.IO.TextReader*)

Loads a HaarCascade from a OpenCV-compatible XML file.
Method: FaceDetect.Vision.Detection.HaarCascade.ToCode(*System.String, System.String*)

Saves a HaarCascade to C# code.
Method: FaceDetect.Vision.Detection.HaarCascade.ToCode(*System.IO.TextWriter, System.String*)

Saves a HaarCascade to C# code.
Type: FaceDetect.Vision.Detection.HaarCascadeStage

Haar Cascade Classifier Stage.
Method: FaceDetect.Vision.Detection.HaarCascadeStage.Classify(*FaceDetect.Imaging.IntegralImage2, System.Int32, System.Int32, System.Double*)

Classifies an image as having the searched object or not.
Method: FaceDetect.Vision.Detection.HaarCascadeStage.Clone

Creates a new object that is a copy of the current instance.
Type: FaceDetect.Vision.Detection.HaarCascadeSerializationObject

Haar Cascade Serialization Root. This class is used
only for XML serialization/deserialization.

Type: FaceDetect.Vision.Detection.HaarCascadeWriter

Automatic transcriber for Haar cascades.
Method: FaceDetect.Vision.Detection.HaarCascadeWriter.Write(*FaceDetect.Vision.Detection.HaarCascade, System.String*)

Writes the specified cascade.
Type: FaceDetect.Vision.Detection.HaarClassifier

Strong classifier based on a weaker cascade of
classifiers using Haar-like rectangular features.
Method: FaceDetect.Vision.Detection.HaarClassifier.Compute(*FaceDetect.Imaging.IntegralImage2, System.Drawing.Rectangle*)

Detects the presence of an object in a given window.
Type: FaceDetect.Vision.Detection.HaarFeature

Rectangular Haar-like feature container.
Method: FaceDetect.Vision.Detection.HaarFeature.GetSum(*FaceDetect.Imaging.IntegralImage2, System.Int32, System.Int32*)

Gets the sum of the areas of the rectangular features in an integral image.
Method: FaceDetect.Vision.Detection.HaarFeature.SetScaleAndWeight(*System.Single, System.Single*)

Sets the scale and weight of a Haar-like rectangular feature container.
Method: FaceDetect.Vision.Detection.HaarFeature.Clone

Creates a new object that is a copy of the current instance.
Type: FaceDetect.Vision.Detection.HaarFeatureNode

Haar Cascade Feature Tree Node.
Method: FaceDetect.Vision.Detection.HaarFeatureNode.Clone

Creates a new object that is a copy of the current instance.
Type: FaceDetect.Vision.Detection.HaarRectangle

Scalable rectangular area.
Method: FaceDetect.Vision.Detection.HaarRectangle.ScaleRectangle(*System.Single*)

Scales the values of this rectangle
Method: FaceDetect.Vision.Detection.HaarRectangle.ScaleWeight(*System.Single*)

Scales the weight of this rectangle.
Method: FaceDetect.Vision.Detection.HaarRectangle.Parse(*System.String*)

Converts from a string representation.
Method: FaceDetect.Vision.Detection.HaarRectangle.Clone

Creates a new object that is a copy of the current instance.
Type: FaceDetect.Vision.Detection.ObjectDetectorSearchMode

Object detector options for the search procedure.
Field: FaceDetect.Vision.Detection.ObjectDetectorSearchMode.Default

Entire image will be scanned.
Field: FaceDetect.Vision.Detection.ObjectDetectorSearchMode.Single

Only a single object will be retrieved.
Field: FaceDetect.Vision.Detection.ObjectDetectorSearchMode.NoOverlap

If a object has already been detected inside an area,
it will not be scanned twice for inner/overlapping objects.
Type: FaceDetect.Vision.Detection.ObjectDetectorScalingMode

Object detector options for window scaling.
Field: FaceDetect.Vision.Detection.ObjectDetectorScalingMode.GreaterToSmaller

Will start with a big search window and
gradually scale into smaller ones.
Field: FaceDetect.Vision.Detection.ObjectDetectorScalingMode.SmallerToGreater

Will start with small search windows and
gradually scale into greater ones.
Type: FaceDetect.Vision.Detection.HaarObjectDetector

Viola-Jones Object Detector based on Haar-like features.
Method: FaceDetect.Vision.Detection.HaarObjectDetector.ProcessFrame(*System.Drawing.Bitmap*)

Performs object detection on the given frame.
Method: FaceDetect.Vision.Detection.HaarObjectDetector.ProcessFrame(*AForge.Imaging.UnmanagedImage*)

Performs object detection on the given frame.
Type: FaceDetect.Vision.Detection.IObjectDetector

Object detector interface.
Method: FaceDetect.Vision.Detection.IObjectDetector.ProcessFrame(*AForge.Imaging.UnmanagedImage*)

Process a new image scene looking for objects.

Events

An event allows a method to be executed when an event is raised. You subscribe to events in the code, and the method will be executed every time the event is called. We'll show you where to subscribe and unsubscribe from event handlers in ARC.

Subscribe to Events

When your robot skill loads, you can subscribe to events. It's best to subscribe to events in the FormLoad event for your form. Here are the steps to create a FormLoad event for your robot skill. Once the form has been loaded and rendered to the screen, this method will be called. Notice that the subscription has a +=, which means subscribe.

1. Select the FormMain, so the designer views it. This is when you can see the buttons, etc.
2. Navigate to the Events for the form in the properties panel
3. Locate the Form Load event
4. Double click in the empty area next to the Form Load event
5. A new event will be created
6. This is the method where you will place the subscribe commands for your robot skill.

Unsubscribe from Event

The events must be unsubscribed when the robot skill is removed from the project. The robot skill is gone, so the event methods no longer exist, or their resources won't. The best place to unsubscribe from events is in the FormClosing event. This is similar to the process you executed in the Subscribe to Events instructions. The only difference is a -= means unsubscribe.

1. Select the FormMain, so the designer views it. This is when you can see the buttons, etc.
2. Navigate to the Events for the form in the properties panel
3. Locate the Form Closing event
4. Double click in the empty area next to the Form Closing event
5. A new event will be created
6. This is the method where you will place the unsubscribe commands for your robot skill.

Troubleshooting

This is a list of troubleshooting conversations with solutions from the community forum regarding creating plugins...

Blank Space In GUID Folder Name
https://synthiam.com/Question/1465

Plugin.XML Not Copying to Output Folder
https://synthiam.com/Question/1465

Plugin.XML file with invalid characters produces error
https://synthiam.com/Question/4186

Here are some items to verify that you may have overlooked during the previous tutorial steps. Please check that these steps have been taken:

1) Have the ARC and EZ_B references been added and configured for "not to copy"?

2) Is the Output Folder of the plugin specified to the correct location? (i.e. *c:\ProgramData\Arc\plugins\[guid]*)

3) Is the plugin.xml configured to be copied to the output folder? Setting should be set for "Copy Always" in solution explorer properties.

4) Is the latest version of ARC installed?

5) Is the main form of your plugin inheriting the correct PluginMaster class rather than Form class?

6) If you cannot execute the plugin for debugging, is the "Debug With External Application" configured to use ARC.exe?

7) Is the correct DLL filename of your plugin specified in the Plugin.xml file?

8) Is the correct GUID specified in the plugin.xml file?

To verify any of these questions, revisit the tutorial steps to ensure that you have completed the tutorial entirely.

Plugin Compliance

Overview
Synthiam is committed to offer users a secure and dependable robot development environment. Due to the high expectation of efficient, dependable and secure from users, there are a few dependencies and restrictions which we enforce in your plugins during review.

Avoid Length GUI Thread Processing
It is very convenient to throw a bunch of code into an event raised by a GUI widget, such as a button or checkbox. When code is executed in an event raised by the user interface, it runs in the thread of that object. This means lengthy code will delay/pause the user interface experience until the code has completed and processing is returned to the GUI thread. This behavior must be reduced at all cost by using events, threading or background workers.

Events
.Net framework provides a number of events for controls, forms and such. Additionally, thr ARC framework provides events for various activities. It is highly preferred to use Events rather than Timers. This includes servo movements, new camera frames, adding/removing controls to workspace, and more. It's good etiquette to unsubscribe from events when your plugin is being closed in the OnClosing() event. If you do not unsubscribe from events, the .Net framework will not know your plugin and respective controls have been disposed, and therefore it may still attempt to execute the method. Notice that an unsubscribe from event is a -= and a subscribe to event is +=

Code:

```
public MyPluginForm() {

  // Subscribe to events to monitor control activity on the workspace
  OnBehaviorControlAdded += FormMain_OnBehaviorControlAdded;
  OnBehaviorControlRemoved += FormMain_OnBehaviorControlRemoved;
}

private void FormMain_OnBehaviorControlAdded(object newControl, int page) {

  MessageBox(string.Format("{0} has been added to the workspace #{1}", newControl.Text, page));
}

private void FormMain_OnBehaviorControlRemoved(object removedControl) {

  MessageBox(string.Format("{0} has been removed from the workspace", removedControl.Text);
}

private void MyPluginForm_FormClosing(object sender, FormClosingEventArgs e) {

  // Unsubscribe from events that I subscribed to while my plugin is going away
  OnBehaviorControlAdded -= FormMain_OnBehaviorControlAdded;
  OnBehaviorControlRemoved -= FormMain_OnBehaviorControlRemoved;
}
```

Timers
One of the most common timers that is used from convenience is *System.Windows.Forms.Timer*, which is heavily frowned upon and will always have your plugin revoked from public access. An alternative and accepted timer for your background worker is *System.Timers.Timer*.

The difference between these two timers is trivial programatically, but vast in their operational behavior. The System.Windows.Forms.Timer will raise the elapsed event in the user interface thread, while the System.Timers.Timer will raise the event in a background thread.

As a new programmer, you may be familiar with the behavior of System.Windows.Forms.Timer not raising the elapsed event until the previous event has completed. With System.Timers.Timer, if your last elapsed event has not completed, a new one will still be raised. Avoid using a Lock() statement for this behavior, and instead use a boolean variable shown in this example...

Code:

```
    System.Timers.Timer _timer;
    bool                _isRunning = false; // variable to ensure timer runs once at a time

    public FormMaster() {

      InitializeComponent();

      _timer = new System.Timers.Timer();
      _timer.Elapsed += _timer_Elapsed;
      _timer.Interval = 100;
      _timer.Start();
    }

    void _timer_Elapsed(object sender, System.Timers.ElapsedEventArgs e) {

        // Check if another copy of the time event is running, if so get out
      if (_isRunning)
        return;

      _isRunning = true;
```

```
    try {

      // Do some work
    } catch (Exception ex) {

      // Uh oh!
    } finally {

      _isRunning = false;
    }
  }
```

Cross-Thread Invoking
While new programmers may feel comfort placing code in events owned by GUI widgets in the user interface thread, there is a different experience when working in a background thread. A background thread will not be able to modify parameters of a GUI object that exists on a different thread. This is called a Cross-Threading Exception. ARC has a helper class to make life easy for you, which is ARC.*Invokers*. You will need to use Invokers when updating UI components from System.Timers.Timer or most events that aren't triggered by UI. The Invokers class will check if an invoke is required.

Code:

```
  void _timer_Elapsed(object sender, System.Timers.ElapsedEventArgs e) {

    if (_isRunning)
      return;

    _isRunning = true;

    try {

      EZ_Builder.Invokers.SetText(textbox1, "Here is a number: {0}", 5);

      EZ_Builder.Invokers.SetChecked(checkbox1, false);

      EZ_Builder.Invokers.SetBackColor(button1, System.Drawing.Color.Red);
    } catch (Exception ex) {

      // Uh oh!
    } finally {

      _isRunning = false;
    }
  }
```

User Configuration Settings
Never under any circumstances save user configuration settings in a separate project file. Always save user configuration settings in the ARC Project File using the tutorial step for Saving/Loading Configuration. Plugins that save configuration data locally to the drive will be revoked from public status. This is to ensure a seamless user experience within the ARC environment. If you have questions about saving custom user data, please inquire on the community forum for assistance.

Exception Handling
Always wrap code in Try {} Catch {} to avoid unhandled exceptions, which will exit ARC. Your plugin will execute under the ARCmaster assembly. If your plugin throws an error that is not handled within an exception, the ARC instance may close.

Code:

```
    try {

      // do some work

    } catch (Exception ex) {

      EZ_Builder.EZBManager.Log("Error in control '{0}'. Message: {1}", this.Text, ex.Message);
    }
```

Examples

Custom Movement Action

Create a custom movement action. Movements, such as forward, left, right, stop, etc. are actions that can be called from any skill. If you wish to create a custom action for a movement panel, it can be done here.

Code:

```
using System;
using System.Windows.Forms;
using ARC;

namespace Custom_Movement_Example {

  public partial class MainForm : ARC.UCForms.FormPluginMaster {

    readonly string MOVEMENT_JUMP_ID = "Jump";

    public MainForm() {

      InitializeComponent();

      // show a config button in the title bar. Set this to false if you do not have a config form.
      ConfigButton = true;

      // Bind to the movement event for every movement that is specified this will raise
      EZBManager.MovementManager.OnMovement2 += MovementManager_OnMovement2;
    }

    private void MainForm_FormClosing(object sender, FormClosingEventArgs e) {

      // Unbind from the movement event when the form is going away
      EZBManager.MovementManager.OnMovement2 += MovementManager_OnMovement2;
    }

    private void MovementManager_OnMovement2(MovementManager.MovementDirectionEnum direction, byte speedLeft, byte speedRight) {

      if (direction == MovementManager.MovementDirectionEnum.Custom && EZBManager.MovementManager.GetCustomMovementId == MOVEMENT_JUMP_ID) {

        // Handle the code here for the custom movement type
      }
    }

    private void btnCustomMovement_Click(object sender, EventArgs e) {

      // This will execute the custom movement action

      EZBManager.MovementManager.GoCustom(MOVEMENT_JUMP_ID);
    }
  }
}
```

Custom JavaScript Extension

A custom object can be assigned to the ARC JavaScript engine. This allows you to create custom methods that are exposed globally to the ARC framework. Once your robot skill is added to the project, the method will be available. This includes using the IntelliSense features within the ARC editor.

Download Example Source

Download the example source project for this robot skill here: Custom JavaScript Method.zip (updated 2025/02/25)

The Robot Skill Code

Here is the main plugin code that subscribes to the OnSetValues event of the javascript engine. This event is raised every time the ARC JavaScript engine initializes. Your code is then responsible to assign an instance of your extension class to the engine. Lastly, when your robot skill is closed (removed from the project), you must unsubscribe from the OnSetValues event.

```
using System.Windows.Forms;

namespace CustomJavaScriptMethod {

  public partial class MainForm : ARC.UCForms.FormPluginMaster {

    public MainForm() {

      InitializeComponent();

      // Do not show the config button because this form has no user options
      ConfigButton = false;

      // Assign an event to apply the custom method(s) for the javascript engine
      ARC.Scripting.JavaScript.JavascriptEngine.OnSetValues += JavascriptEngine_OnSetValues;
    }

    ///

    /// Event raised when the robot skill plugin is closed (i.e. removed from the ARC project)
    ///

    private void MainForm_FormClosing(object sender, FormClosingEventArgs e) {

      // Remove the event assignment when our robot skill form is closing
      ARC.Scripting.JavaScript.JavascriptEngine.OnSetValues -= JavascriptEngine_OnSetValues;
    }

    ///

    /// This method is called by the OnSetValues event.
    /// Every time the ARC JavaScript engine initializes, this method is called to assign our custom
    /// extension method to the javascript engine.
    ///

    private void JavascriptEngine_OnSetValues(ARC.Scripting.JavaScript.JavascriptEngine javascriptEngine) {

      // Create an instance of our custom extension class
      var extensionClass = new ExampleCustomJavaScriptExtension(javascriptEngine);

      // Assign the extension class to the engine and give it the javascript object class name
      javascriptEngine.JintEngine.SetValue(extensionClass.JavaScriptObjectName, extensionClass);
```

```
    }
  }
}
```

The Extension Code

As noticed in the robot skill plugin code above, the custom object is initialized and passed to the ARC JavaScript engine. While you can pass any object to the JavaScript engine, only extending the "DynamicCommandTemplate" will include a parent object (i.e. MyCustom.xxx) and Intellisense compatibility. You will notice this user-defined extension method defines itself in the MyCustom parent object namespace.

```
using System;
using System.Reflection;
using System.Text;
using ARC.Scripting.JavaScript;

namespace CustomJavaScriptMethod {

  [Obfuscation(Exclude = true, ApplyToMembers = true)]
  internal class ExampleCustomJavaScriptExtension : DynamicCommandTemplate {

    ///

    /// Array of characters we will use for the GetRandomCharacters() method
    ///

    const string _chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

    ///

    /// This is the intializer which configures the extension to the engine
    /// The parameters passed to the base class are...
    /// 1) Engine (this is the instance of the arc javascript engine. Leave it as "engine"
    /// 2) Object Name (this is the parent object name that your methods will be avaible under (i.e. MyCustom.getRandomCharacters)
    /// 3) Friendly Description (the friendly description that is displayed to the user in the intellisense syntax editor within ARC)
    ///

    public ExampleCustomJavaScriptExtension(JavascriptEngine engine) : base(engine, "MyCustom", "My Custom Methods") {

    }

    ///

    /// If you need to dispose anything, do it in here
    ///

    protected override void DisposeOverride() {

    }

    ///

    /// One of the custom methods that is exposed to the engine under the MyCustom namespace
    ///

    public string GetRandomCharacters(int count) {

      StringBuilder sb = new StringBuilder(count);

      var r = new Random();

      for (int x = 0; x < count; x++)
        if (JavaScriptEngine.CancelRequested)
          return string.Empty;
        else
          sb.Append(_chars[r.Next(_chars.Length)]);

      return sb.ToString();
    }
  }
}
```

Running The Code

Once you have successfully created a robot skill with the above example, you may now test it. In order for your custom JavaScript extension method(s) to be registered, the robot skill must be added to the ARC project.

Now you can load a script robot skill and use the JavaScript tab to write JavaScript code. In the above example, you may call the custom user-defined method and see the results.

Extend EZB Protocol Custom Command

The EZB protocol contains built-in commands for setting servo positions, reading adc/digital data, etc. However, you can add custom commands by extending the protocol. This is done using the EZBâ€"SendCommand () function in your C# robot skill. The function will send some data and optionally return data in an array. Your firmware needs to wait, listen for the command, and return the specified number of bytes. In this example, the EZB firmware will wait for a custom command and return some example data.

***Note:** This ability is more detailed in the EZB Protocol definition section HERE.

1. EZB Arduino Firmware

First, we must program the Arduino with firmware with a custom command and return some data.

```
#include
#include "SendOnlySoftwareSerial.h"

// The first digital port that is usable on this controller
#define _PortStart 4

// The last digital port on this controller that is usable
#define _PortCnt 14

// The number of analog ports
#define _AnalogPorts 6

// The firmware version that is reported to EZ-Builder to notify of capabilities
#define _FIRMWARE_ID 0x0000000B

// The communication baud rate
#define _BAUD_RATE 57600

// The primary communication interface between EZ-Builder and this controller
#define COMMUNICATION_PORT Serial

// The amount of RX buffer on the communication interface for EZ-Builder
#define _BUFFER_SIZE 1024

// ---------------------------------------------------------------------------

byte         _INPUT_BUFFER[_BUFFER_SIZE];
unsigned int _WRITE_POSITION = 0;
unsigned int _READ_POSITION = 0;
```

```
int _BAUD_RATES [] = {
  4800,
  9600,
  19200,
  38400,
  57600,
  115200,
  115200
};

Servo Servos[_PortCnt];

#define CmdOurCustomCmds      0
#define CmdReleaseAllServos   1
#define CmdGetUniqueID        2
#define CmdEZBv3              3
#define CmdEZBv4              4
#define CmdSoundBeep          5
#define CmdEZServo            6
#define CmdI2CWrite           10
#define CmdI2CRead            11
#define CmdBootLoader         14
#define CmdSetPWMSpeed        15
#define CmdSetServoSpeed      39
#define CmdPing               0x55
#define CmdSetDigitalPortOn   100
#define CmdSetDigitalPortOff  124
#define CmdGetDigitalPort     148
#define CmdSetServoPosition   172
#define CmdGetADCValue        196
#define CmdSendSerial         204
#define CmdHC_SR04            228
#define CmdGetFirwareID       253
#define CmdSoundStreamCmd     254

// CmdEZBv4 Commands
// ------------------------------------------------------------------------------
#define CmdV4SetLipoBatteryProtectionState 0
#define CmdV4SetBatteryMonitorVoltage      1
#define CmdV4GetBatteryVoltage             2
#define CmdV4GetCPUTemp                    3

#define CmdV4UARTExpansion0Init            5
#define CmdV4UARTExpansion0Write           6
#define CmdV4UARTExpansion0AvailableBytes  7
#define CmdV4UARTExpansion0Read            8

#define CmdV4UARTExpansion1Init            9
#define CmdV4UARTExpansion1Write           10
#define CmdV4UARTExpansion1AvailableBytes  11
#define CmdV4UARTExpansion1Read            12

#define CmdV4UARTExpansion2Init            13
#define CmdV4UARTExpansion2Write           14
#define CmdV4UARTExpansion2AvailableBytes  15
#define CmdV4UARTExpansion2Read            16

#define CmdV4I2CClockSpeed                 17
#define CmdV4UARTClockSpeed                18
#define CmdV4ResetToDefaults               19

// CmdSoundStreamCmd Commands
// ------------------------------------------------------------------------------
#define CmdSoundInitStop 0
#define CmdSoundLoad     1
#define CmdSoundPlay     2

bool IsAvail() {

  return _WRITE_POSITION != _READ_POSITION || COMMUNICATION_PORT.available();
}

byte ReadByte() {

  while (_WRITE_POSITION == _READ_POSITION && COMMUNICATION_PORT.available() == 0);

  while (COMMUNICATION_PORT.available()) {

    _WRITE_POSITION++;

    _INPUT_BUFFER[_WRITE_POSITION % _BUFFER_SIZE] = COMMUNICATION_PORT.read();
  }

  _READ_POSITION++;

  return _INPUT_BUFFER[_READ_POSITION % _BUFFER_SIZE];
}

void setup() {

  COMMUNICATION_PORT.begin(_BAUD_RATE);
}

void loop() {

  doEZProtocol();
}

void Write32(long val) {

  COMMUNICATION_PORT.write((byte)(val & 0xff));
  COMMUNICATION_PORT.write((byte)((val >> 8) & 0xff));
  COMMUNICATION_PORT.write((byte)((val >> 16) & 0xff));
  COMMUNICATION_PORT.write((byte)((val >> 24) & 0xff));
}

void Write16(int val) {

  COMMUNICATION_PORT.write((byte)(val & 0xff));
  COMMUNICATION_PORT.write((byte)((val >> 8) & 0xff));
}

#define UNKNOWN_PIN 0xFF

uint8_t getPinMode(uint8_t pin) {

  uint8_t bit = digitalPinToBitMask(pin);
  uint8_t port = digitalPinToPort(pin);

  // Megan't see an option for mega to return this, but whatever...
  if (NOT_A_PIN == port)
    return UNKNOWN_PIN;

  // Is there a bit we can check?
  if (0 == bit)
    return UNKNsingle-bit  // Is there only a single bit set?
  if (bit & bit - 1)
    return UNKNOWN_PIN;
```

```
  volatile uint8_t *reg, *out;
  reg = portModeRegister(port);
  out = portOutputRegister(port);

  if (*reg & bit)
    return OUTPUT;
  else if (*out & bit)
    return INPUT_PULLUP;
  else
    return INPUT;
}

void doEZProtocol() {

  if (IsAvail()) {

    byte cmd = ReadByte();

    if (cmd == CmdPing) {

      // return as a "Capability Controller"
      COMMUNICATION_PORT.write(222);
    } else if (cmd == CmdGetFirwareID) {

      Write32(_FIRMWARE_ID);
    } else if (cmd == CmdReleaseAllServos) {

      for (int port = _PortStart; port < _PortCnt; port++)
        if (Servos[port].attached())
          Servos[port].detach();
    } else if (cmd >= CmdSetServoPosition && cmd <= CmdSetServoPosition + 23) {

      byte port = cmd - CmdSetServoPosition;
      byte pos = ReadByte();

      if (port >= _PortStart && port <= _PortCnt) {

        if (pos == 0 && Servos[port].attached()) {

          Servos[port].detach();
        } else {

          if (!Servos[port].attached())
            Servos[port].attach(port);

          Servos[port].write(pos);
        }
      }
    } else if (cmd >= CmdSetPWMSpeed && cmd <= CmdSetPWMSpeed + 23) {

      byte port = cmd - CmdSetPWMSpeed;
      byte pos = ReadByte();

      if (port >= _PortStart && port <= _PortCnt) {

        if (Servos[port].attached())
          Servos[port].detach();

        if (getPinMode(port) != OUTPUT)
          pinMode(port, OUTPUT);

        analogWrite(port, map(pos, 0, 100, 0, 255));
      }
    } else if (cmd >= CmdSetDigitalPortOn && cmd <= CmdSetDigitalPortOn + 23) {

      byte port = cmd - CmdSetDigitalPortOn;

      if (port >= _PortStart && port <= _PortCnt) {

        if (Servos[port].attached())
          Servos[port].detach();

        if (getPinMode(port) != OUTPUT)
          pinMode(port, OUTPUT);

        digitalWrite(port, HIGH);
      }
    } else if (cmd >= CmdSetDigitalPortOff && cmd <= CmdSetDigitalPortOff + 23) {

      byte port = cmd - CmdSetDigitalPortOff;

      if (port >= _PortStart && port <= _PortCnt) {

        if (Servos[port].attached())
          Servos[port].detach();

        if (getPinMode(port) != OUTPUT)
          pinMode(port, OUTPUT);

        digitalWrite(port, LOW);
      }
    } else if (cmd >= CmdGetDigitalPort && cmd <= CmdGetDigitalPort + 23) {

      byte port = cmd - CmdGetDigitalPort;

      if (port >= _PortStart && port <= _PortCnt) {

        if (Servos[port].attached())
          Servos[port].detach();

        if (getPinMode(port) != INPUT)
          pinMode(port, INPUT);

        COMMUNICATION_PORT.write(digitalRead(port));
      } else {

        COMMUNICATION_PORT.write(0);
      }
    } else if (cmd >= CmdGetADCValue && cmd <= CmdGetADCValue + 7) {

      byte port = cmd - CmdGetADCValue;

      if (port >= 0 && port <= _AnalogPorts)
        Write16(analogRead(port));
      else
        Write16(0);
    } else if (cmd >= CmdSendSerial && cmd <= CmdSendSerial + 23) {

      // Send Serial
      uint8_t port = cmd - CmdSendSerial;
      uint8_t baud = ReadByte(); // Baud rate
      uint8_t size = ReadByte(); // Size

      if (port >= _PortStart && port <= _PortCnt) {

        if (Servos[port].attached())
          Servos[port].detach();

        SendOnlySoftwareSerial tmpSerial(port);
```

```
        tmpSerial.begin(_BAUD_RATES[baud]);

        for (int x = 0; x < size; x++)
          tmpSerial.write(ReadByte());

        tmpSerial.end();
      }
    } else if (cmd == CmdOurCustomCmds) {

      byte cmd2 = ReadByte();

      if (cmd2 == 0x00) {

        // Our first custom command sends two 16 bit ints (4 bytes total)
        Write16(1234);
        Write16(4321);
      } else if (cmd2 == 0x01) {

        // Our second custom command sends two different 16 bit ints (4 bytes total)
        Write16(5678);
        Write16(8765);
      }
    }
  }
}
```

2. C# Robot Skill

In the C# robot skill, we need to send the command and receive the data in an array. Here, we provide examples for both custom commands we created in the Arduino EZB firmware example above.

*Note: In this example, the functions use the EZBManager.PrimaryEZB is the 0 (first) index of the EZB in the connection control. Alternatively, you can use the *EZBManager.EZBs[0].SendCommandData()* to specify the EZB index manually or programmatically. You can have an option in the configuration menu of the robot skill to specify which EZB connection index is hosting your custom command firmware.

```
    void readValues1() {

      try {

        if (!EZBManager.PrimaryEZB.IsConnected)
          return;

        // We'll get the data from the first custom command (should return four bytes, which will be 1234 and 4321)
        var response = EZBManager.PrimaryEZB.SendCommandData(4, 0x00, 0x00);

        // Decode the data returned
        UInt16 val1 = BitConverter.ToUInt16(response, 0);
        UInt16 val2 = BitConverter.ToUInt16(response, 2);

        // Set the data into global ARC variables so all robot skills have access to it
        ARC.Scripting.VariableManager.SetVariable("$val1", val1);
        ARC.Scripting.VariableManager.SetVariable("$val2", val2);

        // Display the values to the textbox on this robot skill
        Invokers.SetAppendText(textBox1, true, "val1: {0}, val2: {1}", val1, val2);
      } catch (Exception ex) {

        Invokers.SetAppendText(textBox1, true, ex.ToString());
      }
    }

    void readValues2() {

      try {

        if (!EZBManager.PrimaryEZB.IsConnected)
          return;

        // We'll get the data from the second custom command (should return four bytes, which will be 5678 and 8765)
        var response = EZBManager.PrimaryEZB.SendCommandData(4, 0x00, 0x01);

        // Decode the data returned
        UInt16 val1 = BitConverter.ToUInt16(response, 0);
        UInt16 val2 = BitConverter.ToUInt16(response, 2);

        // Set the data into global ARC variables so all robot skills have access to it
        ARC.Scripting.VariableManager.SetVariable("$val3", val1);
        ARC.Scripting.VariableManager.SetVariable("$val4", val2);

        // Display the values to the textbox on this robot skill
        Invokers.SetAppendText(textBox1, true, "val3: {0}, val4: {1}", val1, val2);
      } catch (Exception ex) {

        Invokers.SetAppendText(textBox1, true, ex.ToString());
      }
    }
```

EZB Manager

The ARC.EZBManager is a helper class that makes accessing I/O very easy, by doing the complicated work for you. Within the EZBManager are methods to read adc, read digital, set digital, set servo positions, etc..

The ARC.EZBManager also has an array of EZB objects. Each EZB object is a physical EZB. This is an array because the userâ€™s project can have up to 128 EZ-Bâ€™s defined. Itâ€™s always safe to assume that the first (index zero) EZ-B is default, which is why EZBManager.EZBs[0] is commonly used for raw commands that are not using the EZBManager helper methods.

It is highly recommended to use the ARC.EZBManager instead of raw commands directly to the EZB. Here are some examples of using the ARC methods, to demonstrate how much more convenient they are for moving servos, specifically...

Code:

```
ARC.EZBManager.SetServoPosition(_cf.SERVOS[ConfigurationDictionary._VERTICAL_SERVOS], 20);

ARC.EZBManager.SetServoIncrement(_cf.SERVOS[ConfigurationDictionary._HORIZONTAL_SERVOS], -1);
```

Saving/Loading Configuration

In all plugins, there is a _cf variable which is provided by the inherited class which defines the Form. The _cf (configuration file) contains the data saved with your project. The _cf.STORAGE[key] is where your plugin configuration is kept. However, due to ARC providing the UCServoSelection user control, there is also a _cf.SERVOS[key] to store servo configuration.

The UCServoSelection user control is the common displayed control across all configuration screens in ARC. This user control allows configuration of servo ports, min and max distances. As well as multi servo support, etc.. So, if your plugin is to use the UCServoSelection, we highly recommend doing so. To move servos with UCServoSelection configuration data, the example source code in the above mentioned project should be reviewed. In short, it's easy to pass UCServoSelection.Config() data to the ARC.EZBManager.SetServoPosition(), which will take care of the magic - including relationship scaling between multiple servos, min, max and inverted options from UCServoSelection.

Configuration settings must be initialized with default values in the SetConfiguration override. This will ensure that when your plugin is loaded, there is some default configuration data assigned to the keys. Because the data is initialized using the _cf.STORAGE.AddIfNotExist(), the default data will only be added if the key doesn't exist. If the key does exist, following the user loading a project with saved configuration data, the default configuration will not be applied to the key. As demonstrated...

Code:

```
  public override void SetConfiguration(EZ_Builder.Config.Sub.PluginV1 cf) {

    cf.SERVOS.AddIfNotExist(ConfigurationDictionary._HORIZONTAL_SERVOS, new EZ_Builder.Config.Sub.ServoDescriptor[] { });
    cf.SERVOS.AddIfNotExist(ConfigurationDictionary._VERTICAL_SERVOS, new EZ_Builder.Config.Sub.ServoDescriptor[] { });

    cf.STORAGE.AddIfNotExist(ConfigurationDictionary._HORIZONTAL_DEGREES, 0.14m);
    cf.STORAGE.AddIfNotExist(ConfigurationDictionary._VERTICAL_DEGREES, 0.14m);
    cf.STORAGE.AddIfNotExist(ConfigurationDictionary._EDGE_ENABLED, true);
    cf.STORAGE.AddIfNotExist(ConfigurationDictionary._EDGE_SIZE, 10);

    base.SetConfiguration(cf);
  }
```

The base.SetConfiguration in the above example is necessary to set the configuration with the inherited base. This ensures the initialized configuration is applied.

Also in the above example, the ConfigurationDictionary is a class which contains read-only static strings to reference the resource keys. We use these strings so the configuration data can be referenced without worrying about spelling mistakes where ever the data is to be accessed. Here is a look at the above example ConfigurationDictionary...

Code:

```
public class ConfigurationDictionary {

  public static readonly string _HORIZONTAL_SERVOS = "horizontal servos";
  public static readonly string _VERTICAL_SERVOS = "vertical servos";

  public static readonly string _HORIZONTAL_DEGREES = "horizontal degrees";
  public static readonly string _VERTICAL_DEGREES = "vertical degrees";

  public static readonly string _EDGE_SIZE = "edge size";
  public static readonly string _EDGE_ENABLED = "edge enabled";
}
```

Now to use the configuration data from the _cf in your project, simply cast the object from the _cf.STORAGE to the correct type. For _cf.SERVOS, pass the key result directly into the ARC.EZBManager helper methods. Like so...

Code:

```
    if (Convert.ToBoolean(_cf.STORAGE[ConfigurationDictionary._EDGE_ENABLED])) {

      int edgeSize = Convert.ToInt16(_cf.STORAGE[ConfigurationDictionary._EDGE_SIZE]);
      var servosX = _cf.SERVOS[ConfigurationDictionary._HORIZONTAL_SERVOS];
      var servosY = _cf.SERVOS[ConfigurationDictionary._VERTICAL_SERVOS];

      if (_mousePoint.X  _cameraControl.Camera.CaptureWidth - 10)
        ARC.EZBManager.SetServoIncrement(servosX, 1);

      if (_mousePoint.Y  _cameraControl.Camera.CaptureHeight - 10)
        ARC.EZBManager.SetServoIncrement(servosY, 1);
    }
```

Any data in the _cf.STORAGE and _cf.SERVOS will be automatically saved with the users project. And when that project is loaded again in the future, the _cf data will be reloaded as well. This allows your plugin to save the custom configuration applied by the user.

Moving Robot With Movement Panels

If your plugin is to make a robot move, it doesn't need to know anything about the robot.

ARC uses a concept of â€œMovement Panelsâ€. Users can only add one movement panel per project, and this is how their robot moves. When a movement panel is added to a project, it binds itself to a common class that is exposed to the plugin system. This allows your program to easily tell the robot to move forward, left, right, stop, etc. regardless of the movement panel the user has added. This means if the robot is a humanoid, it will walk forward using the project's own method of walking. If the project is a hexapod, it will do the same. If the robot is a drone, it will also do the same. So for your plugin, it doesnâ€™t matter what kind of robot it is â€“ you simply tell it to move a direction. And hereâ€™s a few examples...

Code:

```
ARC.EZBManager.MovementManager.GoForward();
ARC.EZBManager.MovementManager.GoReverse();
ARC.EZBManager.MovementManager.GoRight();
ARC.EZBManager.MovementManager.GoRollLeft();
ARC.EZBManager.MovementManager.GoUp();

Etc...
```

Within that Movement class, there are methods for moving in all directions. Up, Down, Left, Right, Stop, etc.. Not all movement panels will support Up, Down, RollLeft, RollRight as those are generally reserved for used with flying drone robots - but you get the point

Note: It is important to note that all native ARC Movement Panels use the movement class, as it is standard practice to put the movement servos/hbridges on the first EZB when designing a robot.

Move Servos

If you wish to move a servo, it can be done one of two ways. You can specify the servo to move directly, or take advantage of ARC built in servo helpers.

The servo helpers are what you find throughout the ARC interface, when prompting users to select a servo in configuration screens. The user can always select one or more servos that have a relationship to each other, including inverting. It also allows the user to specify a MIN and MAX for each servo.

The Servo Helper in ARC makes moving servos a breeze - and that will make your plugin interact with servos much easier, without having to perform all the relationship math and limits yourself.

Servo Movements (Raw)
To move a servo with raw commands, not using the servo helper, here is path to the class:

â€¢ ARC.EZBManager.EZBs[0].Servo

It is non recommended to set servo positions using this method, unless they are non user-configurable servos. If your plugin expects users to configure the servos, use the UCServoSelection object and the helper methods below.

Servo Movements (UCServoSelection)
The ARC UI has the ability for users to specify multiple servos and have the positions relative to each other with invert options, etcâ€¦ This can be viewed in the sourcecode for the Click Servo plugin here: https://synthiam.com/Software/Manual/16147

You can install that plugin to see how it works, and watch the video on how to set it up. The example in that source code demonstrates how to use the UCServoSelection

user control for users to specify multiple servos, if you choose to go this route. You will notice that the array collection of servos is passed to helper commands in the ARC.EZBManager directly.

By using the UCServoSelection and ARC.EZBManager helper methods, the servo configuration can be easily stored in the project file and passed to helper methods for executing. For example, to extract and store the user's servo configuration from a UCServoSelection into the project file can be like so...

Code:

```
_cf.SERVOS.AddOrUpdate(ConfigurationDictionary._HORIZONTAL_SERVOS, ucServoSelectionX.Config);

_cf.SERVOS.AddOrUpdate(ConfigurationDictionary._VERTICAL_SERVOS, ucServoSelectionY.Config);
```

Now to move the servos from the configuration _cf.SERVOS file, simply do the following...

Code:

```
ARC.EZBManager.SetServoIncrement(_cf.SERVOS[ConfigurationDictionary._HORIZONTAL_SERVOS], -1);

ARC.EZBManager.SetServoIncrement(_cf.SERVOS[ConfigurationDictionary._VERTICAL_SERVOS], 1);
```

Theme Renderer

Overview
When a behavior control is added to an ARC workspace by a user, the control form UI will be manipulated to provide a standard look and feel. Research has demonstrated the benefits of providing users with an unified graphical experience that promotes creativity within ARC. This is done by relieving cognitive load of the user by giving them less to *think* about during their robot programming sessions.

Some Theme Examples
Users configure theme colors for their ARC instance on a per user basis. This configuration is stored in the current logged-in user's registry. The color theme can be customized in the top ARC menu *Options -> Preferences -> Window Theme*.

When Is A Control Form Themed?
The theme engine is called against forms when they're added to the project automatically by the ARC workspace manager. This is the same manager that allows changing desktops, smart arranging windows, loading configurations, etc.. As far as when the theme renderer is called in code, it's after the form's constructor and before the OnLoad() event.

Code:

```
YourPlugin.Constructor()
       |
       V
Theme Renderer
       |
       V
YourPlugin.OnLoad()
```

Skip Themeing of Controls
There's two ways to have a control skip the theme process. This means the specified controls, and respective child controls will be skipped when the theme is applied.Â

1) ThemeRenderer - The *FormPluginMaster* is the base form that your plugin must inherit. Within this *FormPluginMaster* is a *ThemeRenderer* object. You can access the ThemeRenderer.ControlsToSkipThemingï»¿ in the plugin form's constructor following InitializeComponent()Â Â For example, if you had a btnSave that wished to not be themed...

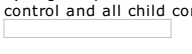
Code:

```
public MyPlugin() {

  // Define and initialize components on plugin form
  InitializeComponent();

  // Skip the following components from the theme renderer on form
  ThemeRenderer.ControlsToSkipTheming.Add(btnSave);
ï»¿}
```

2) Tag: SkipThemeï»¿ - In the properties of a control on your plugin form, you can specify *SkipTheme* as the *Tag* value. This will instruct the ThemeRenderer to skip the control and all child controls.



Dependencies, Files and Sub Folders

Your plugin may require dependencies, such as images, fonts, or other various files.

DLL Files
There are two types of DLL files, managed and unmanaged. A managed DLL is the type that can be added as a resource in the .Net project. The unmanaged DLL is a C or C++ library that is used with Interop calls. If you use either of these, you will be aware of which one is being used.

Any managed DLL file that was added as a resource to your project, should be included to be copied as a resource during compile time. Using managed DLL files does not require any path specifications in your code, as the assembly will be loaded when needed. Visual Studio will take care of copying the file, long as it is marked to be copied, during compile, into the root plugin folder. You should not need to worry about managed DLL location, as the visual studio will take care of it, but it must be in the same folder as your Plugin DLL.

The unmanaged DLL usage is a different story. When your plugin references an unmanaged DLL, it also specifies the path. If the path is not specified in the interop declaration, the default directory of your plugin is assumed. The unmanaged DLL must be added to your project as a content type and included to be copied when compiled. This is because unmanaged DLL files are not added as resources in .Net Visual Studio projects, and therefore you must take care of specifying the file to be copied yourself, during compile time. This is easy by simply adding the file to the project as content, and selecting Copy Always from its property settings.

Files & Sub Folders
If your plugin requires reading files from the plugin folder, here's how you can get the path to the plugin root folder, by combing the guid and windows environment settings for the public path. The location of the plugin folder in the Public Documents is defined by the Windows Operating System. The Public Documents and Plugin path is assembled with the method *EZ_Builder.Constants.PLUGINS_FOLDER*, as seen in code examples below.

This code example combines the user's public ez-builder plugin folder, which comes from the windows environment settings, the guide, and the filename.

Code:

```
using System.IO;
```

```
string OculusRift_Fx_File = ARC.Common.CombinePath(
        ARC.Constants.PLUGINS_FOLDER,
        _cf._pluginGUID,
        "OculusRift.fx");

if (!File.Exists(OculusRift_Fx_File)) {

    MessageBox.Show("Unable to find the OculusRift.fx file that should have installed with this plugin. Cancelling");

    return;
}
```

So if you have a fonts subfolder in the plugin folder or some sub in the plugin folder, you can do this...

Code:

```
      string FontFile = ARC.Common.CombinePath(
        ARC.Constants.PLUGINS_FOLDER,
        _cf._pluginGUID,
        "Fonts",
        "SomeFont.ttf");
```

Or to read the files from a subfolder of the plug directory...

Code:

```
      using System.IO;

      string fontFolder = ARC.Common.CombinePath(
        ARC.Constants.PLUGINS_FOLDER,
        _cf._pluginGUID,
        "Fonts");

      foreach (string fontFile in Directory.GetFiles(fontFolder)) {

           Console.WriteLine("File: {0}", fontFile);
      }
```

ControlCommand() Binding

Overview
The ControlCommand() is a scripting function which enables users to send commands and parameters to supporting controls from another control. It is how controls communicate. Each control broadcasts a list of commands it supports. These commands are displayed in the Cheat Sheet while users are editing scripts (JavasScript or EZ-Script). When a ControlCommand() is executed that has a destination name matching your plugin, an event is triggered in which you will respond.

An Example
An example of a popular ControlCommand() is starting the video feed in the camera control. Users will add Script Control to their program which instructs the Camera Control to begin streaming video from the specified video source.

Code:

```
# JavaScript example to start the video feed on a camera control
controlCommand("Camera Control", "CameraStart");
```

You may notice that some ControlCommand() will accept optional parameters. The CameraStart also has an optional parameter, which is the device name.

Code:

```
# JavaScript example to start the video feed on a camera control specifying device name
controlCommand("Camera Control", "CameraStart", "EZB://192.168.1.1:24");
```

Bind To ControlCommand()
Now that the user is aware of the supported options available in your Cheat Sheet, we will bind to the script engine for any calls directed to your control. This is done through an override method which will be raised in the event that a ControlCommand() matches your control.

Some facts to note in this example code...

1) Comparison is case insensitive. We have no idea what case the text will be entered by the user.

2) If no commands match your syntax, the Base() method will notify the script engine.

3) If expected parameters are missing or incorrect, you may throw an exception which will be caught by the parent script engine.

4) To avoid cross-threading exceptions, there is a fancy helper class ARC.Invokers which contains methods to manipulate user controls from different threads. The SendCommand() event will *always* be called from a background thread. This is because the script engine will never execute threads on a GUI thread.

Code:

```
    public override void SendCommand(string windowCommand, params string[] values) {

      if (windowCommand.Equals("PauseOn", StringComparison.InvariantCultureIgnoreCase)) {

        ARC.Invokers.SetChecked(checkbox1, true);

        if (values.Length == 1)
          ARC.Invokers.SetText(checkbox1, values[0]);
        else if (values.Length > 1)
          throw new Exception(string.Format("Only 0 or 1 parameters expected. You passed {0}", values.Length));

      } else if (windowCommand.Equals("PauseOff", StringComparison.InvariantCultureIgnoreCase)) {

        ARC.Invokers.SetChecked(checkbox1, false);

        if (values.Length == 1)
          ARC.Invokers.SetText(checkbox1, values[0]);
        else if (values.Length > 1)
          throw new Exception(string.Format("Only 0 or 1 parameters expected. You passed {0}", values.Length));

      } else {
```

```
        base.SendCommand(windowCommand, values);
      }
    }
```

Conclusion
By using the ControlCommand(), users can send commands to your plugin or configure settings, all from scripts. This gives your plugin the ability to be better customized for the users needs programmatically.

Blockly
ControlCommand() are usable in Blockly UI, with one exception. Because the Blockly UI does not contain the ability for user defined parameters of the ControlCommand() feature, they are limited to commands with no user parameters. This means that a ControlCommand() with parameters will not display in the blockly UI.

The ControlCommand() for blockly is found in the Utility category.

Viewing the available ControlCommand()'s within blockly, you will see that commands accepting user parameters are not displayed..

To further the example, here are two control commands in which one will be displayed, and one will not be

Code:

```
// This will be displayed in blockly
controlCommand("My Control", "SetColorRed");
controlCommand("My Control", "SetColorGreen");

// These will not be displayed in blockly because it accepts a user parameter
controlCommand("My Control", "SetColor", "Red");
controlCommand("My Control", "SetColor", "Green");
```

Camera Control

Your plugin can access the camera by attaching itself to an existing camera control. Attaching to an existing camera control is a friendly way of sharing the camera video stream with other plugins and maintaining the camera control's features.

This is demonstrating one method of finding a control within the workspace. There are a number of methods that make this easy. Take a look at the *Example: Finding Other Behavior Control* section of this tutorial for a more detailed explanation.

In this example, we will search for an existing camera control and attach to frame event.

Step 1 - Declare a global camera control variable

Your plugin will need to search the project to find a camera control. When it finds the camera control, it will need to keep a reference to it for future actions. Specifically, your plugin will need to keep the camera control reference so it can detach from the video frame event during dispose or close.

Code:

```
namespace Camera_Example {

  public partial class FormMain : EZ_Builder.UCForms.FormPluginMaster {

    // Global variable within your plugin class
    ARC.UCForms.FormCameraDevice _cameraControl;

    public FormMain() {

      InitializeComponent();
    }
  }
}
```

Step 2 - Create an attach() method with Frame Event

This method will search the project for an existing camera control, add a reference to it and attach a method to the frame event.

Code:

```
void attach() {

  // detach in case there is already an attachment to an existing camera control
  detach();

  // get all camera controls in the project
  Control [] cameras = ARC.EZBManager.FormMain.GetControlByType(typeof(ARC.UCForms.FormCameraDevice));

  // if there are no camera controls, inform the user
  if (cameras.Length == 0) {

    MessageBox.Show("There are no camera controls in this project.");

    return;
  }

  // get a reference to the first camera control we find, in the case there are many
  _cameraControl = (ARC.UCForms.FormCameraDevice)cameras[0];

  // attach to the New Frame event
  _cameraControl.Camera.OnNewFrame += Camera_OnNewFrame;

  ARC.EZBManager.Log("Attached to: {0}", _cameraControl.Text);
}

// We will add code to this method later in this example
void Camera_OnNewFrame() {

}
```

Step 3 - Create a detach() method to detach from the camera control

Now that we have created an attach() method, there will need to be a method to detach as well. The detach is necessary specifically for your plugin's Closing event. Be sure to add the detach to the Closing event as in this example.

Code:

```
private void FormMain_FormClosing(object sender, FormClosingEventArgs e) {

  // detach the event handler from this plugin when it closes
  detach();
}

void detach() {

  // only perform detach if there is a reference to a camera control
  if (_cameraControl != null) {

    ARC.EZBManager.Log("Detaching from {0}", _cameraControl.Text);

    // Detach the from the camera event handler
    _cameraControl.Camera.OnNewFrame -= Camera_OnNewFrame;

    // Set the camera control reference to null
    _cameraControl = null;
  }
}
```

Step 4 - Create a button to Attach()

Your plugin will need some way to let the user attach to the camera control. You could do this when the plugin is initiated, but you may wish to give the user the ability to attach/detach on their own. Just in case they do not wish for it to be active right away. Create a button on your plugin which will have this code in the OnClick event. This code will attach() if there is no reference to a camera control, and detach if there is. The code will work as a toggle between the two states.

Code:

```
private void btnAttach_Click(object sender, EventArgs e) {

  if (_cameraControl == null)
    attach();
  else
    detach();
}
```

Step 5 - Create ControlCommand() attach/detach

Be friendly to your users by giving them the ability to attach and detach your plugin via ez-script ControlCommand() syntax. These two override methods will do just that...

Code:

```
// Override is called when ControlCommand() is sent to this control
// If an applicable command is passed, execute it. Otherwise execute the base which throws an exception to the user
public override void SendCommand(string windowCommand, params string[] values) {

  if (windowCommand.Equals("attach", StringComparison.InvariantCultureIgnoreCase))
    attach();
  else if (windowCommand.Equals("detach", StringComparison.InvariantCultureIgnoreCase))
    detach();
  else
    base.SendCommand(windowCommand, values);
}

// Return a list of available ControlCommands() to the ï»¿ARC UI
// This list is presented to the user when they are editing EZ-Script and viewing the Cheat Sheet
public override object[] GetSupportedControlCommands() {

  List cmds = new List();

  cmds.Add("Attach");
  cmds.Add("Detach");

  return cmds.ToArray();
}
```

Step 6 - Do something in the video frame event

Now that you have successfully created attach and detach methods to the camera control, let's do something with the video frame event. The video frame event was created earlier in this project, and was empty. Now let's simply add some code to get the image and draw on it.

Add a reference to the *aforge.dll* in the ARC program folder. This is the same folder which you added the ARC.exe and EZ_B.dll during the plugin setup. The aforge.dll library is a fantastic open-source project with direct memory access to image buffer for manipulation, which is faster than using GDI (System.Drawing).

In this example frame event code, we will draw a solid square around the object which was detected during tracking. For testing, you will need to enable a tracking type.

Code:

```
void Camera_OnNewFrame() {

  // Exit this method if there is no reference to a camera control
  // This may only happen during a dipose/detach if the frame is already executing
  if (_cameraControl == null)
    return;

  // If no object is detected, display a message to the user in the camera image
  if (ARC.Scripting.VariableManager.GetVariable("$CameraIsTracking") == "0") {

    using (Graphics g = Graphics.FromImage(_cameraControl.Camera.GetOutputBitmap.ToManagedImage(false)))
      g.DrawString("No object detected", SystemFonts.CaptionFont, Brushes.Red, 0, 0);

    return;
  }

  // If we got this far, an object must be detected
  // Get the camera control variables for the detected object location and size
  int objectX = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectX"));
  int objectY = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectY"));
  int objectCenterX = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectCenterX"));
  int objectCenterY = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectCenterY"));
  int objectWidth = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectWidth"));
  int objectHeight = Convert.ToInt32(ARC.Scripting.VariableManager.GetVariable("$CameraObjectHeight"));

  // Draw a Crosshair using the aforge library fast draw functions
  AForge.Imaging.Drawing.Line(_cameraControl.Camera.GetOutputBitmap, new AForge.IntPoint(objectCenterX - 8, objectCenterY), new AForge.IntPoint(objectCenterX + 8, objectCenterY),
  AForge.Imaging.Drawing.Line(_cameraControl.Camera.GetOutputBitmap, new AForge.IntPoint(objectCenterX, objectCenterY - 8), new AForge.IntPoint(objectCenterX, objectCenterY + 8),

  // draw a filled rectangle with opacity
  AForge.Imaging.Drawing.FillRectangle(_cameraControl.Camera.GetOutputBitmap, new Rectangle(objectX, objectY, objectWidth, objectHeight), Color.FromArgb(100, 50, 50, 250));
```

```
}
```

You're Done!

You have successfully created methods required to attach/detach from existing camera controls and perform drawing on the frame image!

Finding Other Behavior Controls

Similar to this tutorial's Example: Camera Control step, you can search for other controls added to the project workspace. There are several *EZBManager.FormMain* methods that make it easy. Additionally, there are a few *EZBManager.FormMain* events that allow your plugin to watch for added and removed controls. We'll discuss and provide examples of accessing other controls in the current project.

Control Add/Remove Events

EZBManager.FormMain.OnProjectLoadCompleted ()
This event is raised when a project has completed loading all controls to the workspace. This is a handy method if you're looking for a control when the project is loaded. This event will ensure all other behavior controls have loaded before raising. The event raised after a project has completely loaded all behavior controls. If your control is looking to bind to another control, find the control in this event. If you attempt to look for control (i.e., camera) during constructor or SetConfiguration, the other control may not have loaded from the config yet. This event is raised after all of the controls have been loaded into the workspace.

EZBManager.FormMain.OnBehaviorControlAdded(object newControl, int page)
The event is raised when a new control is added to the project workspace. The control and virtual desktop page will be returned. The event is raised when a control is added to the workspace. This could be during the project load event or if a user uses the Add Control menu. If you want to keep track of new controls added to the workspace, this is how to do it. However, if you're expecting a control to exist when a project is loaded, look into the OnProjectLoadCompleted event.

EZBManager.FormMain.OnBehaviorControlRemovedHandler(object removedControl)
The event is raised when an existing control is removed from the project workspace. This is not raised when the application is closing, or the project is closing. Contrary to OnBehaviorControlAdded, this event is raised when a user removes a control from the current workspace. The control is passed as a parameter, but it won't be closed until you release it from the completion of the event. This means don't expect the control to exist once your method attached to this event has been completed.

Find a Control

You may need to search the project for a specific control, perhaps bind to a camera frame event. This example will grab the first camera that it finds.

```
    void detach() {

      if (_cameraControl != null) {

        if (!_isClosing)
          ARC.Invokers.SetAppendText(tbLog, true, "Detaching from {0}", _cameraControl.Text);

        _cameraControl.Camera.OnNewFrame -= Camera_OnNewFrame;

        _cameraControl = null;
      }

      if (!_isClosing)
        ARC.Invokers.SetText(btnAttach, "Attach");
    }

    void attach() {

      detach();

      Control [] cameras = ARC.EZBManager.FormMain.GetControlByType(typeof(ARC.UCForms.FormCameraDevice));

      if (cameras.Length == 0) {

        ARC.Invokers.SetAppendText(tbLog, true, "There are no camera controls in this project.");

        return;
      }

      _cameraControl = (ARC.UCForms.FormCameraDevice)cameras[0];

      _cameraControl.Camera.OnNewFrame += Camera_OnNewFrame;

      ARC.Invokers.SetAppendText(tbLog, true, "Attached to: {0}", _cameraControl.Text);

      ARC.Invokers.SetText(btnAttach, "Detach");
    }
```

Camera Custom Tracking Type

As you've seen in the previous tutorial step about detecting and attaching to the camera, there are a bunch of events that you can use. One of the events allows you to create a custom tracking type as a plugin, which is real cool!Â

Uses for creating a custom tracking type is if you want to experiment with OpenCV or any other vision libraries. Because ARC leverages .Net, we recommend the x86 nuget install of EMGUCV (https://github.com/emgucv/emgucv). Installing from NUGET is the easiest and most convenient.

The camera events that we'll use for creating a custom tracking type are...

Code:

```
        // assign an event that raises when the camera wants to initialize tracking types
        _camera.Camera.OnInitCustomTracking += Camera_OnInitCustomTracking;

        // assign an event that raises with a new frame that you can use for tracking
        _camera.OnCustomDetection += Camera_OnCustomDetection;
```

Once inside the OnCustomDetection() event, you have access to a bunch of different bitmaps throughout the flow of the detection process. They are...

Code:

```
// **********************
// From the EZ_B.Camera class
// **********************

    ///
    /// This is a temporary bitmap that we can use to draw on but is lost per tracking type
    ///
    public volatile Bitmap _WorkerBitmap;

    ///
    /// This is the resized original bitmap that is never drawn on. Each tracking type uses this as the main source image for tracking, and then draws on the _OutputBitmap for tra
    ///
    public volatile AForge.Imaging.UnmanagedImage _OriginalBitmap; // resized image that we process

    ///
```

```
    /// Image that is outputted to the display. We draw on this bitmap with the tracking details
    ///
    public volatile AForge.Imaging.UnmanagedImage _OutputBitmap;

    ///
    /// Raw image unsized directly from the input device
    ///
    public volatile AForge.Imaging.UnmanagedImage _RawUnsizedBitmap;

    ///
    /// Last image for the GetCurrentImage
    ///
    public volatile AForge.Imaging.UnmanagedImage _RawUnsizedLastBitmap;
```

Understanding the images available, the ones we care about for creating a tracking type of our own are...

Code:

```
    ///
    /// This is the resized original bitmap that is never drawn on. Each tracking type uses this as the main source image for tracking, and then draws on the _OutputBitmap for tra
    ///
    public volatile AForge.Imaging.UnmanagedImage _OriginalBitmap; // resized image that we process

    ///
    /// Image that is outputted to the display. We draw on this bitmap with the tracking details
    ///
    public volatile AForge.Imaging.UnmanagedImage _OutputBitmap;
```

This is because we can use the _OriginalBitmap for our detection, and then draw on the _OutputBitmap where our detection was.

Example
This is an example that fakes detection by drawing a rectangle on the _OutputBitmap that bounces around the screen. It moves with every frame in the CustomDetection event.

Code:

```
    // faking an object being tracked
    int _xPos = 0;
    int _yPos = 0;
    bool _xDir = true;
    bool _yDir = true;

    private EZ_B.ObjectLocation[] Camera_OnCustomDetection(EZ_Builder.UCForms.FormCameraDevice sender) {

      if (_isClosing)
        return new ObjectLocation[] { };

      if (!_camera.Camera.IsActive)
        return new ObjectLocation[] { };

      List objectLocations = new List();

      try {

        // This is demonstrating how you can return if an object has been detected and draw where it is
        // The camera control will start tracking when more than one ObjectLocation is returned
        // We're just putting fake bouncing rectable of a detected rect which will be displayed as a tracked object on the screen in the camera device

        if (_xDir)
          _xPos += 10;
        else
          _xPos -= 10;

        if (_yDir)
          _yPos += 10;
        else
          _yPos -= 10;

        var r = new Rectangle(_xPos, _yPos, 50, 50);

        if (r.Right > _camera.Camera._OutputBitmap.Width)
          _xDir = false;
        else if (r.Left  _camera.Camera._OutputBitmap.Height)
          _yDir = false;
        else if (r.Top <= 0)
          _yDir = true;

        var objectLocation = new ObjectLocation(ObjectLocation.TrackingTypeEnum.Custom);
        objectLocation.Rect = r;
        objectLocation.HorizontalLocation = _camera.Camera.GetHorizontalLocation(objectLocation.CenterX);
        objectLocation.VerticalLocation = _camera.Camera.GetVerticalLocation(objectLocation.CenterY);
        objectLocations.Add(objectLocation);

        AForge.Imaging.Drawing.Rectangle(_camera.Camera._OutputBitmap, r, Color.MediumSeaGreen);
      } catch (Exception ex) {

        EZ_Builder.EZBManager.Log(ex);
      }

      return objectLocations.ToArray();
    }
```

Global Script Variables

The ARC variable manager stores and retrieves global variables that are available within ARC plugins and controls. This allows you to Set and Get variables that other controls may be configuring or using.

Data Types
The script variable stores dynamic data types. This means that there is no declaration between numeric or string values. For example in JavaScript you would type...

Code:

```
setVar("$a", 3);
setVar("$b", 3.123);
setVar("$c", "This is a string");
setVar("$d", 0x55);
```

And in C# Plugin you would type...

Code:

```
ARC.Scripting.VariableManager.SetVariable("$a", 3);
ARC.Scripting.VariableManager.SetVariable("$b", 3.123);
ARC.Scripting.VariableManager.SetVariable("$c", "This is a string");
ARC.Scripting.VariableManager.SetVariable("$d", 0x055);
```

Single Variables
ARC.Scripting.VariableManager.ClearVariable(string variableName)
Clear the variable and remove it from memory.

ARC.Scripting.VariableManager.ClearVariables()
Clear the entire variable memory.

ARC.Scripting.VariableManager.DoesVariableExist(string variableName)
Returns a Boolean if the variable has been defined in memory.

ARC.Scripting.VariableManager.DumpVariablesToString()
Returns a string with each variable and the corresponding value (one per line). This is similar to the Variable Manager control found in ARC->Add Control->Scripting->Variable Manager.

ARC.Scripting.VariableManager.GetVariable(string variableName)
Get the value stored in the specified variable.

ARC.Scripting.VariableManager.GetVariable(string variableName, int index)
Get the value stored in the specified index of the array variable.

ARC.Scripting.VariableManager.SetVariable(string variableName, object value)
Set the value into the memory location of the specified variable. If the variable does not exist, this will create the variable and assign the value.

Array Variables
ARC.Scripting.VariableManager.AppendToVariableArray(string variableName, object value)
Append the value to the existing array.

ARC.Scripting.VariableManager.CreateVariableArray(string variableName, byte[] values)
ARC.Scripting.VariableManager.CreateVariableArray(string variableName, string[] values)
ARC.Scripting.VariableManager.CreateVariableArray(string variableName, object defaultValue, int size)
Create an array variable and specify the default values. Use AppendToVariableArray() to add more items to this array.

ARC.Scripting.VariableManager.FillVariableArray(string variableName, object defaultValue)
If a variable array already exists, this will fill every defined index of the array with the specified value.

ARC.Scripting.VariableManager.GetArraySize(string variableName)
Returns an integer that is the index size of the specified array variable.

ARC.Scripting.VariableManager.IsVariableArray(string variableName)
Returns a Boolean if the specified variable is an array.

ARC.Scripting.VariableManager.SetVariable(string variableName, object value, int index)
Set the value into the index of the specified array variable. The array size must already be defined. If you attempt to set a value to an index outside of the index size, an exception will be thrown.

Monitoring Variable Changes
The variable manager has an event which will be raised for any variable changes. If your plugin has a variable that you wish to watch for changes, subscribe to the event. Remember to unsubscribe from the event when your plugin closes, as per the Plugin Compliance section of this tutorial. Below is an example of subscribing, checking for a variable change, and unsubscribing when the plugin closes.

If the variable is an array, the index will be populated with the array index that has changed. If the variable is not an array, the index will equal -1.

Code:

```
private MyForm() {

Â  InitializeComponent();

Â  // Subscribe to variable change event
Â  ARC.Scripting.VariableManager.OnVariableChanged += VariableManager_OnVariableChanged;
}

private void FormMain_FormClosing(object sender, FormClosingEventArgs e) {

Â  // Unsubscribe to variable change event
Â  ARC.Scripting.VariableManager.OnVariableChanged -= VariableManager_OnVariableChanged;
}

private void VariableManager_OnVariableChanged(string variableName, object value, int index) {

Â  if (!variableName.Equals("$myVariable", StringComparison.InvariantCultureIgnoreCase))
Â  Â  return;

Â  // Do something because the variable has changed
}
```

Script/Blockly Edit Control

Throughout ARC controls, you will notice an edit user control that allows single-line, multi-line and blockly editing for a script. The user control looks like this...

When holding a single line of Script, it looks like this...

Multi-line script is generated by either syntax editor or blockly editor, both of which are accessible by pressing the edit button on the control. When holding multi-lines of the script, generated from either Blockly or Syntax editing UI, it looks like this...

This control can be found as a component within the ARC.exe, and it is called "UCScriptEditInput"

When the edit button of the UCScriptEditInput is pressed, it will display a new window containing both script syntax editor and blockly code creator. The default view (blockly/syntax) is based on 1) the user's preferred editor from preferences menu, 2) the last saved edit mode. ARC will decide which view to display to the user.

Blockly Tab Of Editor

Script Syntax Tab Of Editor

Loading Saved Code Into UCScriptEditInput
Your plugin should be saving code that was created by the user to the project file via the cf.STORAGE option demonstrated in an earlier step of this tutorial. The UCScriptEditInputrequires the loaded data in two formats, the VALUE, and XML. The VALUE property contains the raw SCRIPT that was edited using the Syntax Editor. The XML is the Blockly configuration. In the code example below, the UCScriptEditInput is populated by the cf.STORAGE.

Code:

```
public void SetConfiguration(PluginV1 cf) {

      ucScriptEdit1.Value = cf.STORAGE["MyCodeValue"].ToString();
      ucScriptEdit1.XML = cf.STORAGE["MyCodeXML"].ToString();
    }
```

Saving Code from UCScriptEditInput
As you saw from the earlier step, the user code is loaded in two parts, the XML and script. Both of those properties contain the data which will be saved to the cf.STORAGE as well.

Code:

```
    public PluginV1 SaveCode() {

      PluginV1 cf = new PluginV1();

      cf.STORAGE["MyCodeValue"] = ucScriptEdit1.Value;
      cf.STORAGE["MyCodeXML"] = ucScriptEdit1.XML;

      return cf;
    }
```

Script Editing in DataGridView
Having multiple saved scripts in a DataGridView is possible as well. This allows users to create multiple scripts that could be triggered based on an input, for example. Reference of this behavior is the Speech Recognition or Twitter Recognition controls. They each allow the user to add custom scripts that are triggered on an input (Phrase) value.

When defining an script column for the DataGridView, select the UCScriptEditColumn for the type.

Populating Saved Code
Loading user saved code from the project configuration file to UCScriptEditColumn is a little different than loading to a single UCScriptEditInput. The difference is that a UCScriptEditColumn accepts the CODE and XML to be provided in a single class (UCForms.UC.FormScriptEdit.ConfigurationCls) and passed to the VALUE property. In the example below, the saved data is stored as a CustomObjectv2 in the project configuration. As you learned in an earlier step of this tutorial, the CustomObjectv2 will save your custom class using reflection into the project file.

The code below will foreach loop through each item of the saved code class, which contains the Phrase, Code and XML of each user added item in the DataGridView. Notice how the Code and XML are populated to the UCScriptEditColumn within the UCForms.UC.FormScriptEdit.ConfigurationCls class.

Code:

```
using System.Windows.Forms;

namespace EZ_Builder {

  public partial class Testform : Form {

    public class SavedCodeCls {

      public SavedCodeItemCls [] UserCodes = new SavedCodeItemCls[] { };

      public class SavedCodeItemCls {

        public string Phrase = string.Empty;
        public string Code = string.Empty;
        public string XML = string.Empty;
      }
    }

    public Testform() {

      InitializeComponent();
    }

    public void LoadSavedData(Config.Sub.PluginV1 cf) {

      SavedCodeCls savedCode = (SavedCodeCls)cf.GetCustomObjectV2(typeof(SavedCodeCls));

      foreach (var codeItem in savedCode.UserCodes) {

        // Add a new row and get the index of it
        int rowIndex = dataGridView1.Rows.Add();

        // Assign the saved Phrase to column 0
        dataGridView1.Rows[rowIndex].Cells[0].Value = codeItem.Phrase;

        // Assign the saved Code and XML to column 1
        dataGridView1.Rows[rowIndex].Cells[1].Value = new UCForms.UC.FormScriptEdit.ConfigurationCls(codeItem.Code, codeItem.XML);
      }
    }
  }
}
```

Saving Code
We will now expand on the above code example to include a function for saving the user-defined rows and code from the DataGridView to the project file.

Code:

```
using System.Windows.Forms;
using System.Collections.Generic;

namespace ARC {

  public partial class Testform : Form {

    public class SavedCodeCls {

      public SavedCodeItemCls [] UserCodes = new SavedCodeItemCls[] { };

      public class SavedCodeItemCls {

        public string Phrase = string.Empty;
        public string Code = string.Empty;
        public string XML = string.Empty;
      }
```

```
    }

    public Testform() {

      InitializeComponent();
    }

    public void LoadSavedData(Config.Sub.PluginV1 cf) {

      SavedCodeCls savedCode = (SavedCodeCls)cf.GetCustomObjectV2(typeof(SavedCodeCls));

      foreach (var codeItem in savedCode.UserCodes) {

        // Add a new row and get the index of it
        int rowIndex = dataGridView1.Rows.Add();

        // Assign the saved Phrase to column 0
        dataGridView1.Rows[rowIndex].Cells[0].Value = codeItem.Phrase;

        // Assign the saved Code and XML to column 1
        dataGridView1.Rows[rowIndex].Cells[1].Value = new UCForms.UC.FormScriptEdit.ConfigurationCls(codeItem.Code, codeItem.XML);
      }
    }

    public Config.Sub.PluginV1 GetSavedData() {

      var cf = new Config.Sub.PluginV1();

      List items = new List();

      foreach (DataGridViewRow dgvr in dataGridView1.Rows) {

        // Check if this row is valid by seeing if any necessary columsn are null
        if (dgvr.Cells[0].Value == null)
          continue;
        else if (dgvr.Cells[1].Value == null)
          continue;

        // Get the values of each column
        var phrase = dgvr.Cells[0].Value.ToString();
        var code = (UCForms.UC.FormScriptEdit.ConfigurationCls)dgvr.Cells[1].Value;

        // Assign the values (code, xml and phrase) to the item
        var codeItem = new SavedCodeCls.SavedCodeItemCls();
        codeItem.Code = code.Code;
        codeItem.XML = code.XML;
        codeItem.Phrase = phrase;

        // Add the item to the list
        items.Add(codeItem);
      }

      // Assign the list of code items to the array within the master clas
      SavedCodeCls savedCode = new SavedCodeCls();
      savedCode.UserCodes = items.ToArray();

      // Add the master config class to the project configuration as a custom object
      cf.SetCustomObjectV2(savedCode);

      return cf;
    }
  }
}
```

Script Executor

ARC supports a number of programming languages built-in (JavaScript, EZ-Script & Python). The scripting languages contain many robot-specific commands. The power of scripting also enables robot skill controls to interact with each other using the ControlCommand().

This part of the tutorial will demonstrate how to execute a user-defined script within your plugin. An example of how this is useful is if your plugin provides configuration for the user to configure custom code for a specific action. If you review DJ's Tic Tac Toe example, you will notice that the configuration dialog allows the user to create a custom script for Winning, Losing, and Tie.

The namespace for all scripting methods is *ARC.Scripting*.

Simple Example
This code example demonstrates how to run a script piece of code that will speak a phrase out of the speaker when the button is pressed. You will need to add a Button to your form and assign the Click event for it to work.

Code:

```
  public partial class MainForm : ARC.UCForms.FormPluginMaster {

    ARC.Scripting.Executor _executor;

    public MainForm()
      : base() {

      InitializeComponent();

      _executor = new ARC.Scripting.Executor();
    }

    private void button1_Click(object sender, EventArgs e) {

      _executor.StartScriptASync("Say(\"Hello, I am script\")");
    }
  }
```

Example with Events
There are many events that can be associated with the Executor, such as OnDone, OnError, etc.. This allows your program to accommodate behaviors. In this example, the Executor will display a message box when the script has completed executing.

Code:

```
  public partial class MainForm : ARC.UCForms.FormPluginMaster {

    ARC.Scripting.Executor _executor;

    public MainForm()
      : base() {

      InitializeComponent();

      _executor = new ARC.Scripting.Executor();
      _executor.OnDone += _executor_OnDone;
    }

    private void button1_Click(object sender, EventArgs e) {

      _executor.StartScriptASync("Say(\"Hello, I am a script\")");
```

```
    }

    void _executor_OnDone(string compilerName, TimeSpan timeTook) {

      MessageBox.Show(string.Format("Script has completed and took {0} milliseconds", timeTook.TotalMilliseconds));
    }
  }
```

Starting A New Script When Another Is Running
If the executor is currently running an ASync script in the background while you launch another, the first script will be cancelled and the most recent script will begin executing. Each instance of an Executor can run only one script. To run multiple scripts at the same time, use an Executor per script.

Inside the Executor
The executor has many methods and events.

ARC.Scripting.Executor.ExecuteScriptSingleLine(string line);
Executes only one line of script and blocks until it has completed. This method returns the result of the single line of code. For example, if the code was a function to return the value of a servo (example GetServo(d0)), the value of the servo d0 will be returned.

ARC.Scripting.Executor.Resume();
There is a script command to "Pause" the current running script. If the command is ever executed in the script, this will resume the execution.

ARC.Scripting.Executor.StartScriptASync(Command[] compiled);
This executes the compiled Command Opcode array in the background. If you precompile the script into an array of Command Opcodes, this method can be used. The advantage to pre-compiling the source into Command Opcodes is that the script will execute faster because it will not need to be compiled each time. There is a compiler cache in the Executor, however. This means that if you run the same script twice that is not compiled, the last Opcode cache will be used.

ARC.Scripting.Executor.StartScriptASync(string script);
This method compiles the plain text script and executes it in the background. There is a compiler cache in the Executor, however. This means that if you run the same script consecutive times, the last Opcode cache will be used.

ARC.Scripting.Executor.StartScriptBlocking(string script);
This method compiles the plain text script and executes it on the current thread. There is a compiler cache in the Executor, however. This means that if you run the same script consecutive times, the last Opcode cache will be used.

ARC.Scripting.Executor.StopScript();
Stops the current ASync running script on this Executor.

Events
These events can be assigned to an Executor and will be raised at their appropriate function. The "CompilerName" parameter will include the optional compiler name that can be provided when the Executor class is initiated. In the above examples, the Executor class is created without any parameters. If a parameter was supplied, it would be the name of this compiler. If your program has many Executors that share these events, the CompilerName parameter will come in handy to identify what executor it originated from.

event OnCmdExecHandler(string compilerName, int lineNumber, string execTxt)
Event is raised for each line that is executed in the script. This will dramatically slow the execution of the script, but is great for debugging.

event OnDoneHandler(string compilerName, TimeSpan timeTook)
Event is raised when the script has completed.

event OnPausedHandler(string compilerName)
Event is raised when the script calls the Pause method. The Executor.Resume() function is necessary to continue, or StopScript().

event OnResumeHandler(string compilerName)
Event is raised when the script is instructed to resume after a Pause.

event OnStartHandler(string compilerName)
Event is raised when the script begins to execute.

Custom EZ-Script Function

The EZ-Script engine has two event to add your own custom EZ-Script functions for users to use. The two different events will call before (Override) and after (Additional) the EZ-Script executor. That means you can either override existing commands or add new commands.

Additional Commands
Adding a command extends the existing commandset of EZ-Script. The example below demonstrates adding a new command to EZ-Script called "SetColor()".

Code:

```
SetColor(20, 100, 20)
```

The *ExpressionEvaluation.FunctionEval.AdditionalFunctionEvent* event is raised. Your plugin may attach to this event and process functions for the script.

Here is an example plugin that creates a user defined function called **SetColor()**: https://synthiam.com/Software/Manual/User-Defined-Function-Example-15864

Code:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace User_Defined_Function_Example {

  public partial class FormMaster : EZ_Builder.UCForms.FormPluginMaster {

    public FormMaster() {

      InitializeComponent();
    }

    private void FormMaster_Load(object sender, EventArgs e) {

      // Intercept all unknown functions called from any ez-script globally.
      // If a function is called that doesn't exist in the ez-script library, this event will execute
      ExpressionEvaluation.FunctionEval.AdditionalFunctionEvent += FunctionEval_AdditionalFunctionEvent;
    }

    private void FormMaster_FormClosing(object sender, FormClosingEventArgs e) {

      // Disconnect from the function event
      ExpressionEvaluation.FunctionEval.AdditionalFunctionEvent -= FunctionEval_AdditionalFunctionEvent;
    }

    ///
    /// This is executed when a function is specified in any ez-scripting that isn't a native function.
    /// You can check to see if the function that was called is your function.
    /// If it is, do something and return something.
    /// If you don't return something, a default value of TRUE is returned.
    /// If you throw an exception, the ez-script control will receive the exception and present the error to the user.
    ///
    private void FunctionEval_AdditionalFunctionEvent(object sender, ExpressionEvaluation.AdditionalFunctionEventArgs e) {

      // Check if the function is our function (SetColor)
      if (!e.Name.Equals("setcolor", StringComparison.InvariantCultureIgnoreCase))
```

```
        return;

      // Check if the correct number of parameters were passed to this function
      if (e.Parameters.Length != 3)
        throw new Exception("Expects 3 parameters. Usage: SetColor(red [0-255], green [0-255], blue [0-255]). Example: SetColor(20, 200,100)");

      // Convert the parameters to datatypes
      byte red = Convert.ToByte(e.Parameters[0]);
      byte green = Convert.ToByte(e.Parameters[1]);
      byte blue = Convert.ToByte(e.Parameters[2]);

      // Do something
      EZ_Builder.Invokers.SetBackColor(label1, Color.FromArgb(red, green, blue));

      // Return something. Good idea to return TRUE if your function isn't meant to return anything
      e.ReturnValue = true;
    }
  }
}
```

Override Commands
Override commands will replace the existing EZ-Script command with your own version. This means you can intercept the command and handle the logic yourself. An example of this functionality is in the Ultrasonic Ping Sensor control, which replaces the GetPing() command with it's own version.

*Note: This event has OverrideFunctionEventArgs() which has a IsHandled(bool) that must be set if you hanlded the return object.

The *ExpressionEvaluation.FunctionEval.OverrideFunctionEvent* event and your logic must exist in there to override the existing command. If you expect a different number of parameters, you can still return and let the main EZ-Script handle it by not setting IsHandled in the OverrideFunctionEventArgs.

Code:

```
using ExpressionEvaluation;
using EZ_B;
using EZ_Builder.Config;
using EZ_Builder.Scripting;

namespace EZ_Builder.UCForms {

  public partial class FormPing : FormMaster {

    public FormPing() {

      InitializeComponent();

      progressBar1.Maximum = EZ_B.HC_SR04.MAX_VALUE;

      // Set the override event
      ExpressionEvaluation.FunctionEval.OverrideFunctionEvent += FunctionEval_OverrideFunctionEvent;
    }

    private void FormPing_FormClosing(object sender, FormClosingEventArgs e) {

      // detach from the override event
      ExpressionEvaluation.FunctionEval.OverrideFunctionEvent -= FunctionEval_OverrideFunctionEvent;
    }

    private void FunctionEval_OverrideFunctionEvent(object sender, ExpressionEvaluation.OverrideFunctionEventArgs e) {

      if (e.Name.Equals(OpCodeReadEnum.getping.ToString(), StringComparison.InvariantCultureIgnoreCase)) {

        FunctionEval.CheckParamCount(e.Name, e.Parameters, 2);

        string triggerStr = e.Parameters[0].ToString();
        string echoStr = e.Parameters[1].ToString();
        HelperPortParser cmdTrig = new HelperPortParser(triggerStr);
        HelperPortParser cmdEcho = new HelperPortParser(echoStr);

        if (cmdTrig.DigitalPort != _cf.Ping.PingTriggerPort ||
            cmdTrig.BoardIndex != _cf.Ping.EZBIndex ||
            cmdEcho.DigitalPort != _cf.Ping.PingEchoPort ||
            cmdEcho.BoardIndex != _cf.Ping.EZBIndex)
          return;

        if (!Invokers.GetCheckedValue(cbPause))
          return;

        if (cmdTrig.BoardIndex != cmdEcho.BoardIndex)
          throw new Exception("Trigger and Echo ports must be on the same EZ-B");

        if (!EZBManager.EZBs[cmdTrig.BoardIndex].IsConnected)
          throw new Exception(string.Format("Not connected to EZ_B {0}", cmdTrig.BoardIndex));

        int value = EZBManager.EZBs[cmdTrig.BoardIndex].HC_SR04.GetValue(cmdTrig.DigitalPort, cmdEcho.DigitalPort);

        updateDisplayData(value);

        e.ReturnValue = value;
        e.IsHandled = true;

        return;
      }
    }
  }
}
```

Custom Movement Panel

The ARC software uses controls that register themselves as a movement panel. This allows your plugin to listen to movement requests from other controls. When any control or ez-script calls for a movement direction (i.e., Forward, Left, Stop, etc.), your plugin can be responsible for moving the robot. There can only be one movement panel per robot project. This is because there is only one method of locomotion for a robot, which this movement panel would be responsible for.

To understand more about how Movement Panels work in ARC, read this tutorial.

What's In A Movement Panel?
A movement panel will have buttons that let the user specify directions that the robot should move. Your movement panel will be responsible for the robot moving. This means that anywhere a direction is specified, your control will be responsible for moving the robot. Generally, a movement panel has speed controls in the form of trackbars of some sort.

Code Example
Want to make your movement panel? Here is an example of how to implement code that will respond to movement requests:

Code:

```
public FormMain()
  : base() {

  InitializeComponent();

  // Assign this control as a movement panel so the software knows who owns the movements
  EZ_Builder.FormMain.MovementPanel = this;
```

```
  // Assign the movement locomotion style for this control.
  // There are different kind of locomotion for your robot, and this helps other
  // controls know what to expect when a movement is happening.
  // Check out the ENUM to see what other locomotion styles there are that
  // fits your movement panel type.
  EZBManager.MovementManager.LocomotionStyle = LocomotionStyleEnum.GAIT;

  // assign the movement event
  // this event is raised when another ARC control requests movement
  EZBManager.MovementManager.OnMovement2 += Movement_OnMovement2;

  // assign the speed change event
  // this event is raised when another control or user changes the speed
  EZBManager.MovementManager.OnSpeedChanged += Movement_OnSpeedChanged;
}

private void FormModifiedServoMovementPanel_FormClosing(object sender, FormClosingEventArgs e) {

  // Remove this control as a movement panel
  EZ_Builder.FormMain.MovementPanel = null;

  EZBManager.MovementManager.OnSpeedChanged -= Movement_OnSpeedChanged;

  EZBManager.MovementManager.OnMovement2 -= Movement_OnMovement2;
}

private void Movement_OnSpeedChanged(int speedLeft, int speedRight) {

  // do something with the speed change
}

private void Movement_OnMovement2(MovementManager.MovementDirectionEnum direction, byte speedLeft, byte speedRight) {

  // **
  // do something based on the speed
  // handle speed change here
  // **

  // Now do something based on the new movement direction

  if (direction == MovementManager.MovementDirectionEnum.Forward) {

    // handle custom Forward movement

  } else if (direction == MovementManager.MovementDirectionEnum.Reverse) {

    // handle custom Reverse movement

  } else if (direction == MovementManager.MovementDirectionEnum.Right) {

    // handle custom Right movement

  } else if (direction == MovementManager.MovementDirectionEnum.Left) {

    // handle custom Left movement

  } else if (direction == MovementManager.MovementDirectionEnum.Stop) {

    // handle custom Stop movement

  }
}
```

Output Audio from EZ-B

This is an example with source code about how to play audio out of the EZ-B. The EZ-B will output audio as a stream or byte array.

Source code as a plugin project is here:
OutputAudioFromEZ-BSource.zip

*Dependency: Additional to adding ARC.exe and EZ-B.DLL, this plugin requires NAudio.DLL library to be added as a project reference. Remember to UNSELECT "Copy Files"! The naudio, arc.exe, ezb.dll are located in the c:\program files (x86) Synthiam arc installation folder.

This plugin provides the following examples:

1) Load audio from MP3 or WAV file

Code:

```
// MP3
NAudio.Wave.Mp3FileReader mp3 = new NAudio.Wave.Mp3FileReader(openFileDialog1.FileName);

// WAV
NAudio.Wave.WaveStream wav = new NAudio.Wave.WaveFileReader(openFileDialog1.FileName);
```

2) Convert audio file to uncompressed PCM data to supported EZ-B sample rate and sample size

Code:

```
NAudio.Wave.WaveFormatConversionStream pcm = new NAudio.Wave.WaveFormatConversionStream(new NAudio.Wave.WaveFormat(EZ_B.EZBv4Sound.AUDIO_SAMPLE_BITRATE, 8, 1), mp3);
```

3) Compress PCM data with gzip to be stored in project STORAGE

Code:

```
                using (MemoryStream ms = new MemoryStream()) {

                  using (GZipStream gz = new GZipStream(ms, CompressionMode.Compress))
                    pcm.CopyTo(gz);

                  _cf.STORAGE[ConfigTitles.COMPRESSED_AUDIO_DATA] = ms.ToArray();
                }
```

4) Play audio data from compressed project STORAGE

Code:

```
        using (MemoryStream ms = new MemoryStream(compressedAudioData))
        using (GZipStream gz = new GZipStream(ms, CompressionMode.Decompress))
          EZBManager.EZBs[0].SoundV4.PlayData(gz);
```

5) Supports ControlCommand() for Play and Stop of audio to be used in external scripts

Code:

```
public override object[] GetSupportedControlCommands() {

  List items = new List();

  items.Add(ControlCommands.StartPlayingAudio);
  items.Add(ControlCommands.StopPlayingAudio);

  return items.ToArray();
}

public override void SendCommand(string windowCommand, params string[] values) {

  if (windowCommand.Equals(ControlCommands.StartPlayingAudio, StringComparison.InvariantCultureIgnoreCase))
    playStoredAudio();
  else if (windowCommand.Equals(ControlCommands.StopPlayingAudio, StringComparison.InvariantCultureIgnoreCase))
    stopPlaying();
  else
    base.SendCommand(windowCommand, values);
}
```

6) Changes the status of the button when audio is playing globally from anywhere in ARC on EZ-B #0

Code:

```
public FormMain() {

  InitializeComponent();

  EZBManager.EZBs[0].SoundV4.OnStartPlaying += SoundV4_OnStartPlaying;
  EZBManager.EZBs[0].SoundV4.OnStopPlaying += SoundV4_OnStopPlaying;
}

private void FormMain_FormClosing(object sender, FormClosingEventArgs e) {

  EZBManager.EZBs[0].SoundV4.OnStartPlaying -= SoundV4_OnStartPlaying;
  EZBManager.EZBs[0].SoundV4.OnStopPlaying -= SoundV4_OnStopPlaying;
}

private void SoundV4_OnStopPlaying() {

  Invokers.SetText(btnPlayAudio, "Play");
}

private void SoundV4_OnStartPlaying() {

  Invokers.SetText(btnPlayAudio, "Stop");
}
```

Output Text to Speech
You can output text to speech easily as well, using the following code example...

Code:

```
using (MemoryStream s = EZBManager.EZBs[0].SpeechSynth.SayToStream("I am speaking out of the EZ-B"))
  EZBManager.EZBs[0].SoundV4.PlayData(s);
```