Release Notes

RobotStudio SDK

5.15.01

Revision: -

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission, and contents thereof must not be imparted to a third party nor be used for any unauthorized purpose. Contravention will be prosecuted.

Additional copies of this document may be obtained from ABB at its then current charge.

© Copyright 2013 ABB All rights reserved.
ABB AB
Robotics Products
SE-721 68 Västerås
Sweden

Table of Contents

Overview 5

1 Release Information 6

1.1 General 6

1.2 Contents 6

2 Information to all users of Visual Studio Tools for Applications (VSTA) 7

2.1 VSTA will be phased out 7

2.2 Switch to SmartComponents or Visual Studio 2012 7

3 What's New in 5.15.01? 8

3.1 Project templates for Visual Studio 2012 8

3.2 Application Id 8

3.3 Loading Add-Ins downloaded from the Web 8

3.4 SmartComponents installed under Program Files are now trusted 8

3.5 Improved validation of .rsaddin file validation 8

3.6 Launch RAPID Editor with module from controller memory 9

3.7 All new types and methods 9
3.7.1 ABB.Robotics.RobotStudio.Environment 9
3.7.2 ABB.Robotics.RobotStudio 9
3.7.3 ABB.Robotics.RobotStudio.Controllers 9
3.7.4 ABB.Robotics.RobotStudio.Stations 10
3.7.5 ABB.Robotics.RobotStudio.Forms 10

4 What's New in 5.15.00.01? 11

5 What's New in 5.15? 12

5.1 RobotApps™ integrated in Document Window 12

5.2 RobotStudio 64-bit edition 13
5.2.1 Add-In target platform support 13
5.2.2 Add-In search paths 13

5.3 Support for asynchronous programming 14
5.3.1 Fire and forget 14
5.3.2 Waiting for the return value 14

5.4 Online and Offline tabs merged 15
5.4.1 Station controller, virtual controllers and real controllers 16
5.4.2 Selected controller object 16
5.4.3 Recent controllers list 16
5.4.4 Adding a controller connection 16
5.4.5 Launching the RAPID Editor and Configuration Editor 16

5.5 Two rows of document tabs 17

5.6 Denavit- Hartenberg parameters for mechanisms 17

5.7 Volume intersection check on Part 17

5.8 All new types and methods 17
5.8.1 ABB.Robotics.RobotStudio.Environment 18
5.8.1 ABB.Robotics.RobotStudio 19
5.8.2 ABB.Robotics.RobotStudio.Controllers 19
5.8.3 ABB.Robotics.RobotStudio.Documents 20
5.8.4 ABB.Robotics.RobotStudio.Stations 21
5.8.5 ABB.Robotics.RobotStudio.Forms 22

6 Late-breaking information 24

6.1 Overview 24

7 Problems Corrected 25

7.1 Solved Product Defect Documents (PDD) since RobotStudio SDK 5.15 25

8 Known Limitations 26

8.1 Development Environment 26

8.2 Visual Studio Tools for Applications (VSTA) 26

9 Installation Information 27

9.1 Hardware and Software requirements 27

9.2 Compatibility 27

Overview

Description

This document provides information about the new features, problems corrected, and installation of RobotStudio SDK.

Product Overview

RobotStudio SDK is included in the RobotStudio installation and can be used for free by anyone who wants to develop on the RobotStudio platform. It could be scripts, Add-Ins or applications that can provide new functionality and/or customized user interfaces in RobotStudio.

You can also combine RobotStudio SDK with PC SDK to communicate with real or virtual IRC5 controllers.

Custom Smart Components with Code Behind can also be developed using the RobotStudio SDK.

Visit our web site at http://www.abb.com/roboticssoftware for more information and updates.

RobotStudio, including RobotStudio SDK, can be downloaded from the Download section.

Here you will also find the User Forum, where developers discuss software problems and solutions online.

To get started, or to learn more, visit our Developer Center: http://developercenter.robotstudio.com.

1 Release Information

1.1 General

Release Name

The release name is RobotStudio SDK 5.15.01 and the build number is 5.15.5335.1091.

For information about RobotStudio, please refer to the document *'Release Notes RobotStudio 5.15.01.pdf'*.

Release Date

The release date is April 19th, 2013.

1.2 Contents

Overview

The installation includes software, documentation and tools as specified below.

Software

ABB.Robotics.Math.dll

ABB.Robotics.Environment.dll

ABB.Robotics.RobotStudio.dll

ABB.Robotics.RobotStudio.Controllers.dll

ABB.Robotics.RobotStudio.Documents.dll

ABB.Robotics.RobotStudio.Stations.dll

ABB.Robotics.RobotStudio.Stations.Forms.dll

Documentation

After installation the help file can be launched from the RobotStudio Help Menu, and Windows Start Menu\Programs\ABB Industrial IT\Robotics IT\Robot Studio 5.15\SDK.

Developer Center - http://developercenter.robotstudio.com.

- API reference
- Sample projects
- Walkthroughs

Reference Manual – RobotStudio SDK (Html Help).

Tools

LibraryCompiler.exe – A tool for batch creation for RobotStudio libraries such as SmartComponents and Mechanisms.

Visual Studio Project Templates (C#)

RobotStudio Add-In – template for a standard Add-In

RobotStudio Smart Component – template for a Smart Component with Code Behind and XML description.

2 Information to all users of Visual Studio Tools for Applications (VSTA)

2.1 VSTA will be phased out

From the next major release, VSTA will no longer be available in RobotStudio.

This has the fillowing implications:

- The IDE will not be available.
- Macros will not be executed.
- Stations and libraries with macros can be loaded, but the macros will not be executed.

2.2 Switch to SmartComponents or Visual Studio 2012

The C# or VB.NET code in a VSTA macro can be migrated to a Visual Studio 2012 Express C# project, and you can continue development using Visual Studio.

In case you have a macro that is used in together with the EventManager, you also have the option to implement the same functionality using RobotStudio SmartComponents, instead of using Visual Studio and C#.

If you want to migrate another macro, ABB recommends using Visual Studio 2012 Express.

3 What's New in 5.15.01?

Overview

This section contains information on the new features of RobotStudio SDK 5.15.01.

3.1 Project templates for Visual Studio 2012

RobotStudio SDK now contains Visual Studio 2012 project templates for Add-Ins and SmartComponent projects.

3.2 Application Id

A new optional element `<ApplicationId></ApplicationId>` has been introduced in the .rsaddin file.

This is useful if multiple versions of the same Add-In is installed on the same machine. If the application id is not specified, all versions of the Add-In will be loaded. When the application id is specified, RobotStudio will recognize that it has found several instances of the same Add-In, and will only load the first one, and inform the user that more Add-Ins with the same application id was found.

It is recommened that all Add-Ins specify a unique application id of the format "com.organization.department.product".

It is also recommended to prefix your ProjectObject attribute keys with your application name in order to avoid collisions with attributes managed by other Add-Ins.

3.3 Loading Add-Ins downloaded from the Web

For security reasons the .NET 4.0 runtime prevents loading of assemblies downloaded from the Web under ceratin circumstances, which results in an exception being thrown.

This exception was not handled properly by RobotStudio, and may have caused problems for users downloading Add-Ins from RobotApps for example.

RobotStudio now gives you the ability to either unblock the assembly if you trust it, and continue load it, or skip loading it, if you do not trust it.

This article on MSDN contains more information about the security model change between .NET 3.5 and 4.0.

3.4 SmartComponents installed under Program Files are now trusted

A SmartComponent with Code-Behind, installed under *Program File/Program Files x86*, is now trusted by RobotStudio. This means that when loading a non signed SmartComponent from such trusted location, a warning dialog will not be displayed.

If you create an installer for your application and the users installs it, she trusts your entire application, including the Code-Behind, hence it is not necessary to sign it.

3.5 Improved validation of .rsaddin file validation

A message will be displayed in the output window if RobotStudio fails to load a PowerPac because the correct entrypoint is not specified in its .rsaddin file.

3.6 Launch RAPID Editor with module from controller memory

The method `ControllerManager.ShowControllerUserInterface(string url)` has been improved to accept paths to RAPID modules and Configuration types in the controller memory.

For example you can launch an instance of the RAPID Editor, displaying a RAPID module in a connected real or virtual controller;

```
ControllerManager.ShowControllerUserInterface("{0CE62073-0B82-4232-
B048-D93FED20A606}/RAPID/T_ROB1/Module_1")
```

To open a module on the PCs file system;

```
ControllerManager.ShowControllerUserInterface("C:\\Users\\user\\Module
1.mod")
```

Note: It is also possible to specify the path to a configuration domain, which will open the Configuration Editor instead.

3.7 All new types and methods

Overview

This section contains information about all the new types in the API, and existing types which has been extended with new methods. Each updated namespace has a separate chapter. For extended types, the name of the type is in bold face, followed by its new methods.

3.7.1 ABB.Robotics.RobotStudio.Environment

Extended types

```
class ExecuteCommandEventArgs
public CancellationToken CancellationToken
```

3.7.2 ABB.Robotics.RobotStudio

Extended types

```
class DataRecorderBase
void Start()
void Stop()
class ProjectSelection
void AddRange(IEnumerable<Object> selections)
```

3.7.3 ABB.Robotics.RobotStudio.Controllers

New types

```
class BuiltInOnlineControllerSourceSignals
class DataRecorderOnlineControllerSource
class OnlineDataRecorder
enum BuiltInOnlineDataRecorderMotionSignal
```

Extended types

```
class ControllerManager
BuiltInDataRecorderSignals BuiltInDataRecorderSignals
OnlineDataRecorder DataRecorder
```

3.7.4 ABB.Robotics.RobotStudio.Stations

New types

```
enum MappingFailure
```

Extended types

```
class GraphicControl
void Clear(Boolean disposeChildren)
void Remove(GraphicComponent graphicComponent, Boolean dispose)
class RsIrc5Controller
MappingResult AssociateMechanismWithMechanicalUnits(Mechanism mechanism, RsMechanicalUnit[] mechanicalUnits, Boolean updateTransform)
MappingResult ChangeMechanismInMechanicalUnits(Mechanism mechanism, RsMechanicalUnit[] mechanicalUnits)
```

3.7.5 ABB.Robotics.RobotStudio.Forms

Extended types

```
DirectionControl
RefCoordSys RefCoordSys
class GraphicControl
Bitmap ScreenShot(ProjectObject obj, Int32 width, Int32 height, ScreenshotOptions options)
class GraphicPicker
Boolean MultiJogEnabled
```

4 What’s New in 5.15.00.01?

Overview

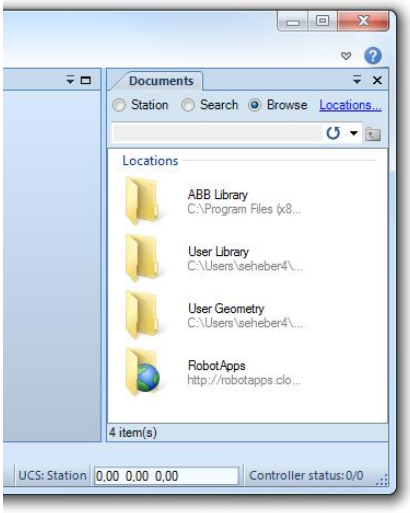
There are no new features in 5.15.00.01.

5 What's New in 5.15?

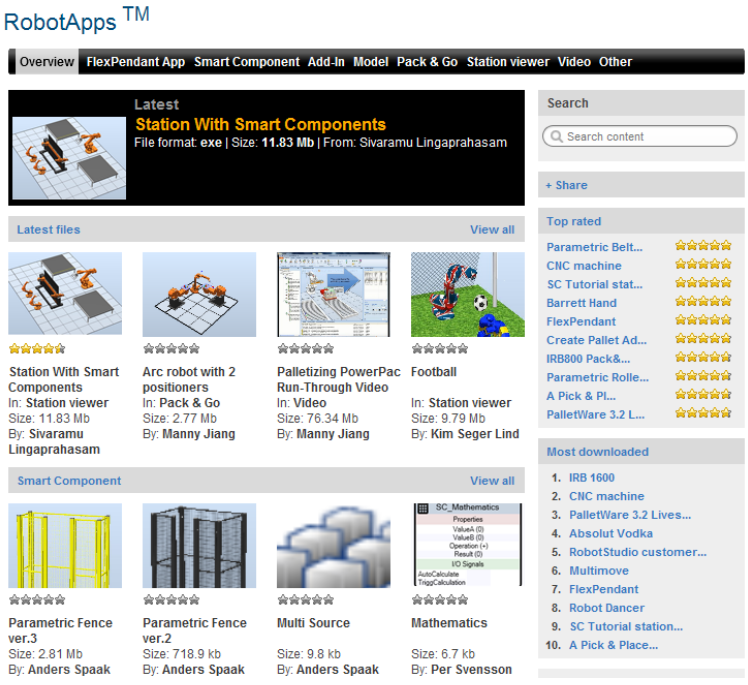
Overview

This section contains information about new features and corrected problems in 5.15. A selected set of the most important new APIs are explained. For a complete list of all new types and extended types, see 5.6.

5.1 RobotApps™ integrated in Document Window



Library files, including SmartComponents, and CAD models published on RobotApps™, is now accessible directly from the `DocumentManager` in the API and the Document Window in the RobotStudio user interface. You can publish content to the RobotApps web site under http://www.abb.com/roboticssoftware



5.2 RobotStudio 64-bit edition

RobotStudio is now available in a 64-bit version, and you can develop applications that runs in the 64-bit version.

5.2.1 Add-In target platform support

Add-ins built with the Visual Studio project setting *Platform Target* set to *Any CPU*, can be loaded in both the 32-bit and 64-bit edition of RobotStudio.

If the add-in has references to other assemblies they must be built for *Any CPU* as well.

Note: A native 64-bit process may not load native 32-bit dll's. As result hereof, an add-in referencing any native 32-bit assembly cannot be loaded in the 64-bit edition of RobotStudio.

It is the responsibility of the developer to verify that add-ins are compatible with the 64-bit edition of RobotStudio, and by default add-ins located outside the 64-bit bin folder, will not be loaded;

```
C:\Program Files (x86)\ABB Industrial IT\Robotics IT\RobotStudio 5.15\Bin64\Addins
```

You can specify which platform your add-in supports, by using the new `<Platform/>` element in your `.rsaddin` file.

For example, to specify that an add-in may only be loaded in the 32-bit edition;

`<Platform>x86</Platform>`.

Value	Description
x86	The add-in and its dependencies may only be loaded in the 32-bit edition.
x64	The add-in and its dependencies may only be loaded in the 64-bit edition.
Any	The add-in and its dependencies may be loaded in both the 32-bit and the 64-bit edition.

If the `<Platform>` element is not present, the default value is `x86`.

5.2.2 Add-In search paths

Add-ins is loaded from the directories, and in the order, as specified in the table below.

32-bit edition	
1	RobotStudio\Bin\Addins
2	Program Files (x86)\Common Files\ABB Industrial IT\Robotics IT\RobotStudio\Addins

64-bit edition	
1	RobotStudio\Bin64\Addins
2	RobotStudio\Bin\Addins *****
3	Program Files (x86)\Common Files\ABB Industrial

	IT\Robotics IT\RobotStudio\Addins *

***** Remark: The 64-bit edition of RobotStudio will assume that add-ins outside the Bin64\Addins folder are 32-bit, and will hence not be loaded, unless otherwise stated by the `<Platform>` element in an `.rsaddin` file.

As before, an add-in can also be placed in a subdirectory with the same name as the add-in.

5.3 Support for asynchronous programming

The RobotStudio API now supports asynchronous programming using the `await` keyword for some methods.

These methods may have both a synchronous and an asynchronous variant. In this case the asynchronous variant has the suffix “Async”.

For example the method `Task RsIrc5Controller.StartAsync()` returns immediately to the caller and returns a `Task` object.

Using the new `await` keyword in .NET 4.5 you can call several asynchronous methods after each other and even have branches between them in your code, without having to take care of waiting for the completion of each asynchronous operation.

Refer to MSDN for more information on `Task` and `await`.

Note: The `await` keyword was introduced in .NET 4.5, and the `Task` class in .NET 4.0.

5.3.1 Fire and forget

The simplest scenario is when you are calling an asynchronous method and is not interested in the return value – fire and forget.

```
void StartMyTwoControllers()
{
   ctrl1.StartAsync(…)
   ctrl2.StartAsync(…)
}
```

5.3.2 Waiting for the return value

If your code needs to know the return value from an asynchronous method call, in order to know what do to next you can use the `await` keyword.

By using `await` you can write asynchronous code that has a sequential flow and looks pretty much as it if is calling plain synchronous methods.

A simple use case is illustrated by the following code snippet;

```
DocumentInfo docInfo = null;
docInfo = await DocumentInfo.FromFileAsync(fileName);
// This line will not be executed until docInfo
// has been returned.
String author = docInfo.Author;
```

In the below scenario we are implementing a list of recently used station files where a thumbnail shall be displayed using the DocumentInfo class.

The code is called from a user interface event handler. In order to get a responsive user interface it is important to return as fast as possible from the event handler.

If the document info for our file is not already cached it has to be loaded from the station or file on disk, which can take any amount of time.

Maybe the file is not a network share. It may end up with a failure after 10 seconds, and that is a very long time for an event handler.

We have a command list where the thumbnails shall be displayed.

```
BackstageCommandList _recentListBox;
```

The track event is subscribed to.

```
_recentListBox.TrackedItemChanged += new
EventHandler(_recentListBox_TrackedItemChanged);
```

In the event handler `FromFileAsync` is called. Code which is not relevant for the example has been omitted.

```
async void _recentListBox_TrackedItemChanged(object sender, EventArgs e)
{
	…
	…
	if (fileName != null && File.Exists(fileName))
	{
		DocumentInfo docInfo = null;
		if (!_cachedDocInfo.TryGetValue(fileName, out docInfo))
		{
			docInfo = await DocumentInfo.FromFileAsync(fileName);
			_cachedDocInfo[fileName] = docInfo;
			…
			…
	}
}
```

The return value is stored in a variable and can be used on the next line, without having to take care of waiting for the `FromFileAsync` to return.

5.4 Online and Offline tabs merged

In RobotStudio 5.15 the Offline and Online tabs have been merged into the Controller tab.

This gives a unified user experience and a tab which is controller centric and allows the user to access all kinds of controllers whether they are part of a station or found on the network.

This opens up new possibilities for developers to create applications where the user can work with virtual as well as real controllers in a seamless way.

Note 1: If your existing add-in contains code that relies on the existence of the Online or Offline tab it must be modified.

Note 2: Each ribbon tab has a string identifier which may be used to programmatically access the tab. The identifiers are not published in the API documentation and ABB does not promise that they are constant between product releases. They are used at your own risk.

5.4.1 Station controller, virtual controllers and real controllers

The class `ControllerManager` is responsible for keeping track of and provides information about referenced real controllers on the network, referenced virtual controllers on the network, and virtual controllers being part of a RobotStudio station.

The following properties can be used to distinguish between different types of controller references.

`ControllerObjectReference.IsServicePortController`

`ControllerObjectReference.IsStationController`

`and`

`ConrollerTypeControllerObjectReference.ControllerType`

The `ControllerType` specifies three types of controllers;

`StationVC` - A virtual controller that is part of a RobotStudio station
`VC` – A virtual controller on the network
`RC` - A real controller on the network

5.4.2 Selected controller object

The currently selected controller reference can be retrieved using the `ControllerManager.SelectedControllerObject` property.

Virtual as well as real controllers can be selected.

5.4.3 Recent controllers list

The list of recent referenced controllers can be accessed using the

`ControllerManager.RecentControllers` property.

5.4.4 Adding a controller connection

Connections to controllers on the network can be added and removed using

```
Task<ControllerObjectReference>
ControllerManager.ControllerReferences.AddAsync(String url)
```

and

```
Task ControllerManager.ControllerReferences.RemoveAsync(String url)
```

5.4.5 Launching the RAPID Editor and Configuration Editor

Applications can lunch the RAPID Editor or the Configuration Editor, given the URL to the RAPID module, or configuration domain.

```
ControllerManager.ShowControllerUserInterface(String url)
```

5.5 Two rows of document tabs

Document windows can now be organized in two levels of tabs.

All `DocumentWindow` instances with the same value of the new `Category` property will be grouped together.

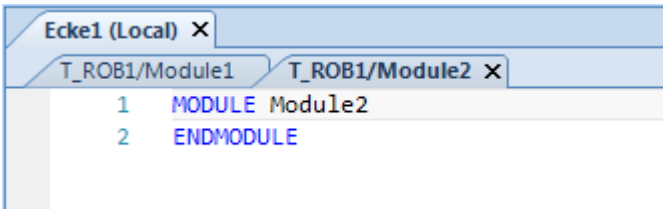


Figure 1 Two document windows with the same category.

5.6 Denavit- Hartenberg parameters for mechanisms

The new method `DenavitHartenbergParameters[] GetDenavitHartenbergParameters()` can be used to retrieve DH-parameters for the specified mechanism.

DH-parameters are available for mechanisms with closed loop kinematics.

The typical use case is to retrieve DH-parameters from RobotStudio and pass them to another kinematic solver software package.

5.7 Volume intersection check on Part

With `Part.IntersectVolume (BoundingBox box, Matrix4 boxTransform)` you can check if a part intersects with a bounding box.

One use case is a SmartComponent gripper that needs to sense objects to pick, using a volume instead of a line.

5.8 All new types and methods

Overview

This section contains information about all the new types in the API, and existing types which has been extended with new methods. Each namespace has a separate chapter. For extended types, the name of the type is bold face, followed by its new methods.

5.8.1 ABB.Robotics.RobotStudio.Environment

New types

```
enum CloseButtonBehavior
interface IZoomableWindow
```

Extended types

```
class CommandBarPopup
void Show(Int32 x, Int32 y)
class CommandGroupExecuteCommandEventArgs
Task CompletionTask
class DisplayCommandGroupEventArgs
void AddSeparator()
class DocumentWindow
String Category
class ExecuteCommandEventArgs
Task CompletionTask
enum RibbonControlLayout
Hidden
StatusBarPane
AutoSizeMode AutoSizeMode
HorizontalAlignment TextAlignment
class ToolBarControl
Boolean Horizontal
class ToolWindow
CloseButtonBehavior CloseButtonBehavior
class UIEnvironment
Boolean GroupDocumentWindows
void DisableCommands(Task task)
class Window
Boolean ActiveTab
class WindowCollection
void AddDockedOrTabbed(ToolWindow window,
DockStyle dockStyle)
event
EventHandler<WindowCollectionChangedEventArgs>
Added
event
EventHandler<WindowCollectionChangedEventArgs>
Removed
```

5.8.1 ABB.Robotics.RobotStudio

New types

```
class DataRecorderSourceBaseAsync
class DataRecorderSinkBase2
interface IHasSystemId
interface IProgressCallback
```

Extended types

```
class DataRecorderSinkBase
Boolean UIVisible
class DataRecorderSourceBase2
String GetImageKey
class DocumentInfo
Task FromFileAsync(void fileName)
class Logger
void AddMessage(LogMessage msg, Boolean
bringToFront)
void AddMessage(String message, Boolean
bringToFront)
class Options
void SetFileName(String filename)
void RemoveSection(String section)
class ProjectObject
IEnumerable<ProjectObject>
FindObjects(Predicate<ProjectObject> filter,
Predicate<ProjectObject> recurse)
enum ProjectObjectChangeType
UIVisible
class RobotStudioAPI
SynchronizationContext SyncContext
```

5.8.2 ABB.Robotics.RobotStudio.Controllers

New types

```
enum ControllerType
enum RecentControllerAvailability
class RecentControllerCollection
class RecentControllerInfo
```

Extended types

```
class ControllerManager
```

```
ControllerObjectReference
SelectedControllerObject
RecentControllerCollection RecentControllers
Boolean IsStationController(String systemId)
Boolean ShowControllerUserInterface(String
url, Object data)
Boolean ShowControllerUserInterface(String
url)
class ControllerObjectReference
Boolean IsServicePortController
Boolean IsStationController
ControllerObjectReference Root
ControllerType ControllerType
String Name
String RelativeUrl
String SystemIdString
String Tag
String Url
enum ControllerObjectType
RAPID
Module
Routine
class ControllerReferenceCollection
ControllerObjectReference this[String
systemId]
Task Add(void systemId)
Task Add(void systemId)
Task Remove(ControllerObjectReference
controller)
```

5.8.3 ABB.Robotics.RobotStudio.Documents

Extended types

```
class DocumentManager
Task GetLocalCopyAsync(void documentInfo)
void Initialize()
```

5.8.4 ABB.Robotics.RobotStudio.Stations

New types

```
enum ControllerMappingState
struct DenavitHartenbergParameters
enum IntersectionType
enum PackAndGoFailureReason
enum PackAndGoLibraryCopyOptions
class PackAndGoResult
enum ScreenshotOptions
enum SimulationStopwatchCollection
class StationServices
struct StopwatchTrigger
enum VirtualControllerRestartMode
```

Extended types

```
class GraphicComponentCollection
void Clear(Boolean disposeChildren)
void Remove(GraphicComponent graphicComponent,
Boolean dispose)
class Mechanism
Boolean CalculateInverseKinematics(RsRobTarget
robTarget, RsWorkObject workObject, RsToolData
tool, Int32[] cfg, out Double[]
resultJointVector)
CanReach(RsRobTarget robTarget, RsWorkObject
workObject, RsToolData tool)
DenavitHartenbergParameters[]
GetDenavitHartenbergParameters()
Boolean SetJointValues(Double[] jointValues,
Boolean updateController, Boolean notify)
class Part
IntersectionType IntersectVolume(BoundingBox
box, Matrix4 boxTransform)
Part Load(String fileName, IProgressCallback
progressCallback, Boolean surfaceModel,
Boolean translateHidden, Boolean healing,
DetailLevels detail)
class RsIrc5Controller
RsIrc5Controller(String systemPath)
ControllerMappingState MappingState
StartAsync(VirtualControllerRestartMode
restartMode, IEnumerable<Mechanism>
mechanismsToMap)
StartAsync(VirtualControllerRestartMode
restartMode, IEnumerable<Mechanism>
mechanismsToMap, Boolean checkBaseFrame)
```

```
event EventHandler SystemStateChanged
class RsIrc5ControllerCollection
void Add(RsIrc5Controller ctrl)
void Remove(RsIrc5Controller ctrl)
class RsJointTarget
void Highlight(Color color)
void ResetHighlight()
class RsMechUnit
Task SetBaseFrameAsync(Matrix4 baseFrame,
Boolean restart)
event EventHandler ActivationModeChanged
class RsMoveInstruction
void Highlight(Color color)
void ResetHighlight()

class RsPathProcedure
Task MoveAlongAsync()

class RsTask
Boolean SetExternalAxisJointValues(Double[]
jointValues, Boolean notify)

class RsTaskCollection
RsTask this[String name]

class SimulationConfiguration
SimulationStopwatchCollection Stopwatches

class Simulator
Task StartAsync()

class SmartComponent
Boolean IsProtected

class Station
Station(Boolean setAsActive)

enum SystemState
Connecting
```

5.8.5 ABB.Robotics.RobotStudio.Forms

New types

```
class DirectionControl
```

Extended types

```
class NumericTextBoxArray
Boolean OverrideBugInSetBoundsCore
class GraphicControl
Point ProjectPointToScreen(Vector3 point)
Bitmap ScreenShot(ScreenshotOptions options)
Bitmap ScreenShot(Int32 width, Int32 height,
ScreenshotOptions options)
```

6 Late-breaking information

6.1 Overview

This section contains late-breaking information that will be included in the appropriate documents in the subsequent releases.

7 Problems Corrected

Overview

This section describes the problems solved in the RobotStudio SDK 5.15.01 release.

7.1 Solved Product Defect Documents (PDD) since RobotStudio SDK 5.15

No solved PDDs.

Other defects solved

WI ID	Title
8887	Ability to load addins directly on startup
8279	RobotStudio SDK templates not copied to VisualStudio from 64-bit exe
8050	Performance to GetJointSignals is very slow

8 Known Limitations

8.1 Development Environment

The RobotStudio API is not thread safe

Access to the RobotStudio API is not inherently thread safe. Only access the API from the thread that your Add-In was called from by RobotStudio. If multiple threads manipulate the object model it can be left in an inconsistent state.

8.2 Visual Studio Tools for Applications (VSTA)

Breakpoints are not supported in VSTA

It is not possible to debug VSTA projects using brekpoints. When selecting Start Debugging from the Debug menu of VSTA, then the added breakpoints will be disabled.

Workaround: *Create an add-in using Visual Studio instead of VSTA.*

Properties and methods that use the type System.Drawing.Color will not work in VSTA.

This is a limitation on the Visual Studio Tools for Applications (VSTA) environment.

Note*: The VSTA-class VSTABridge can be used to work around this problem, see API documentation.*

Static events cannot be called from applications developed in VSTA.

This affects for example the Simulation – Tick event.

Workaround*: Create a standard add-in if static events are to be used. Alternatively, use the VSTABridge class that can workaround this problem, see API documentation.*

Debugging of VSTA Applications

When debugging a VSTA application that adds menu items to the RobotStudio environment, then the menu will not be removed when the program execution stops. This may cause multiple entries of the same menu to be added in RobotStudio. This only affects VSTA add-ins being debugged and not completed VSTA add-ins.

Workaround*: Restart RobotStudio to remove the extra menus.*

VSTA Library add-ins not available

In the Add-ins browser there is a folder for so-called VSTA Library add-ins. This feature is not available.

Use Visual Studio 2010 or 2012 Express for advanced add-in

The purpose of VSTA is to write custom actions and minor utilities. For advanced add-in development use Visual Studio 2010 or 2012 Express that can be downloaded free of charge from *http://www.microsoft.com/express/*

RsLoadData does not work from VSTA

VSTA limitation

The 'FindDataDeclarationsByType' method used in VB.net throw an exception
Limitation added to API doc.

9 Installation Information

Overview

To install RobotStudio SDK click *RobotStudio* on the RobotWare & RobotStudio DVD.

If you select the default installation option **Complete**, RobotStudio SDK will be installed.

In order for the Visual Studio project templates to be installed, Visual Studio 2010 or 2012 needs to be installed on the computer before you install RobotStudio SDK.

If you do not want to install the assemblies and templates, select the installation option Custom and uncheck the feature RobotStudio/SDK.

9.1 Hardware and Software requirements

Software requirements

Operating system:

Same as RobotStudio – Please refer to the RobotStudio Release Notes

Supported development environment

Microsoft Visual Studio 2010

Microsoft Visual Studio 2012

The Visual Studio project templates works only for the English version of Visual Studio.

NOTE!

As RobotStudio 5.15 is built on .NET Framework 4.0, Microsoft Visual Studio 2008 can not be used for RobotStudio 5.15.xx application development.

It is possible, but not supported, to develop RobotStudio Add-Ins with any .NET development environment, such as Visual Studio 2010 or 2012 Express or SharpDevelop, but there will be no project templates available. The RobotStudio API documentation also assumes that Visual Studio 2010 or 2012 is used, which means that information about VisualStudio project settings etc has to be adopted to the other development environment.

9.2 Compatibility

The APIs in the RobotStudio SDK are backwards compatible.