AAMotion API
Basic Setup

Application Note

agito

www.agito-akribis.com

Member of Akribis Systems group

Revision History

Version	Description	Date
1.0	Initial Release	18 October 2022

Contact Information

Manufacturer	Agito Akribis Systems Ltd., Member of Akribis Systems Group
Address	6 Yad-Harutsim St., P.O.Box 7172, Kfar-Saba 4464103
Telephone	+972-9-8909797
Website	www.agito-akribis.com

Copyright Notice

Products Rights

AGDx, AGCx, AGMx, AGAx, AGIx, and AGLx are products designed by Agito Akribis Systems Ltd. in Israel. Sales of the products are licensed to Akribis Systems Pte Ltd. under intercompany license agreement.

Agito Akribis Systems Ltd. has full rights to distribute above products worldwide.

Disclaimer

This product documentation was accurate and reliable at the time of its release.

Agito Akribis Systems Ltd. reserves the right to change the specifications of the product described in this manual without notice at any time.

Trademarks

Agito PCSuite is a trademark of Agito Akribis Systems Ltd..

Warranty

This product is warranted to be free of defects in material and workmanship and conforms to the specifications listed in this manual, for a period of 12 months from the shipment date from factory.

Contents

1 Introduction

1.1 Background

In system level applications, an industrial PC is often used to regulate the overall application. In a scanning application, for example, the PC will issue commands to the camera to take pictures, and commands to the motion controller to move motors.

API's are usually provided by the manufacturer to ease the integration of such communication between the PC and the motion controller. Agito motion controllers provide for a low-level API, 'AAComm', as well as a high-level API, 'AAMotion'.

The low-level API, 'AAComm' provides for fundamental access to the controller over the various communication protocol. To use the API at this level, the user will need to be familiar with the native commands and behavior of Agito controllers.

The high-level API, 'AAMotion' is built upon the low-level API, and provides for interfaces for axes and methods to perform functions such as point-to-point motion, homing and others. The high-level API shields the software developer for the native commands of the Agito driver, therefore, allowing for quicker implementation as the developer will not need to understand Agito commands in great detail. This also serves to abstract the role of the software developer from the application engineer.

1.2 Scope

This application note will cover the steps to install the C# AAMotion library in Visual Studios. This application note assumes that the user is already familiar with the Visual Studios IDE and C# language.

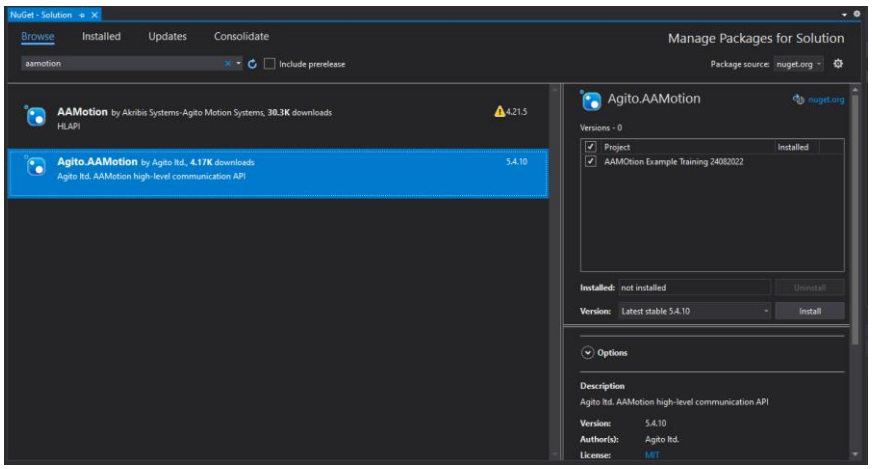
2 Setup

2.1 Installation

NuGet is a package manager for .NET where authors and consumers may produce or consume packages over a shared repository. AAMotion API is uploaded to this repository for easy accessibility and maintenance.

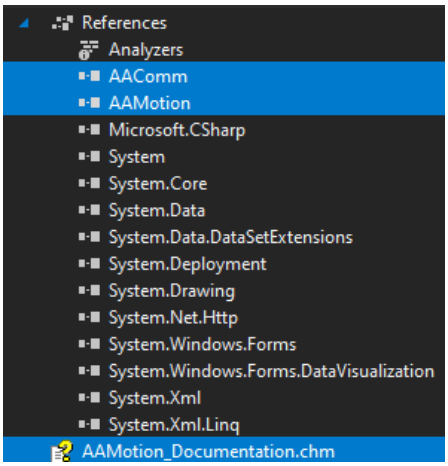
To install the API package, simply access the package manager in Visual Studios via Tools > NuGet Package Manager > Manage NuGet Packages for Solution...

Under the Browse tab, search for AAMotion. Select the package named “Agito.AAMotion” and click on Install to install.



Screenshot 1. NuGet Solution Manager

After installing the package, you will be able to see the changelog, along with help files on the API.



2.2 Basic Implementation

The AAMotion provides for Interfaces to be implemented upon for objects such as the motion controller, axes, group axes (CNC), IO modules.

The API also provides for the motion controller classes that implements the above interfaces based on the configuration of the product. For example, the AGD301 Class implements the motion controller interface, three instances of single axis, two instances of group axis and an instance of the IO module.

Functions and properties specific to a motion controller may be accessed through an instance of a motion controller class. For example, functions such as Connect(), Disconnect(), SendCommandString() can be accessed via an instance of these classes.

Functions and properties specific to the axis may be accessed via the axes instances in the motion controller object. For example, functions such as MoveAbs(), MotorOn() can be accessed via these objects.

Functions and properties related to IO are likewise accessed via instances of the IO module within the motion controller object.

For more information on the classes, methods and properties, please refer to the help file in the API.

2.3 Example

This segment explains the basic steps to instantiate a controller object and do a simple motion. Please refer to the code example while going through this segment.

Step 1. Reference the AAMotion library

```
using AAMotion;
```

Step 2. Instantiate an instance of the controller that you are using.

```
AGC301 _controller = new AGC301();
```

The constructor by default instantiates a controller object which polls for status updates every 50ms. If a different update interval is required, the constructor also provides for an override to specify the required update interval.

Do take note not to put too small a value as this might overload the communication bandwidth resulting in less resources for the controller to do other stuff, such as running the user program or processing other commands.

Step 3. Call for the Connect method to make a connection.

```
_controller.Connect("172.1.1.101");
```

The Connect() method takes in a string input to search for controllers over the ethernet that have the particular IP Address. If successful, the method returns True and the object starts polling for status updates from the controller and updates the internal properties.

Step 4. Read the current position of axis A

```
_aPos = controller.A.Pos;
```

Properties of the object are updated every interval as specified in the constructor. The user may also opt to call for the method UpdateInternalStatus() to make sure that he gets the latest value.

Step 5. Motor on, set motion profile and move to a position 5,000 counts.

```
_controller.A.MotorOn();
_controller.A.Speed = 5000;
_controller.A.Accel = 50000;
_controller.A.MoveAbs(5000);
```