

AN300046: Using Adlos Communication Hardware Abstract API [HAL]

Introduction

The KannMOTION-API serves the easy connection of KannMOTION Devices with your application.

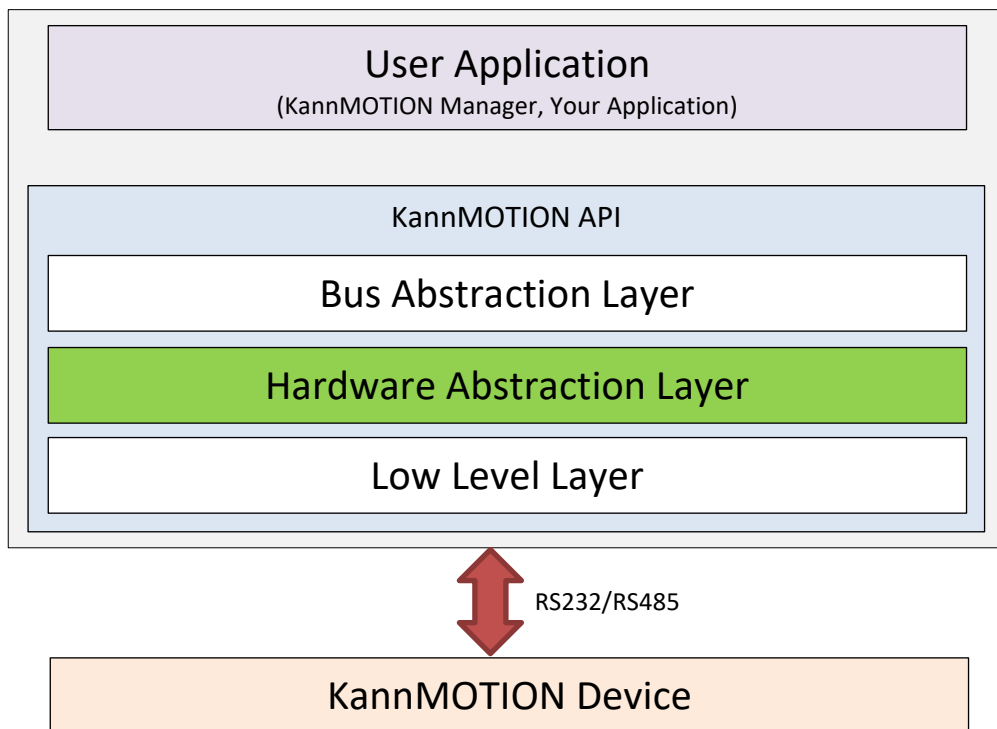
The API is split into 3 layers, the Low Level Layer (LLL), the Hardware Abstraction Layer (HAL) and the Bus Abstraction Layer (BAL).



This particular document introduces the user to the HAL-API. The HAL library allows you to work with some functions as search devices, call commands and so on.



Hardware Abstraction Layer

Common Infos



	The HAL is ONLY available as a .NET Core library.
	This application note is for HAL with version $\geq 0.9.0.0$

API

With this API you can search and connect devices, read information and send commands. With the **SerialCommunicator**-Object or the **ISerialCommunicator**-Interface, you can communicate with your KannMOTION device. Below a quick overview.

```
namespace Adlos.KM.Framework.Interfaces.Devices.Serial
{
    public interface ISerialCommunicator
        : ICommunicator<int>, ICommunicator, IArticle, IDisposable, IComparable<ICommunicator<int>>
    {
        int SerialTimeout { get; set; }
        List<IDataTable> TrackingList { get; }
        List<IDataTable> OnlineList { get; }
        List<IEventTable> EventsList { get; }
        List<ICommandTable> CommandsList { get; }
        ISettingsTable Settings { get; }

        void ActivateLogMode(string path = null);
        void CloseConnection();
        void OpenConnection();
        List<byte> Send(string command);
        List<byte> Send(string command, int respBytes);
    }
}
```

Usage

In this app note it is described, how to use the HAL DLL in a Visual Studio Project.

For this, we create a new Visual Studio Project (Console App .NET Core) to call some basic commands with API methods.

To use the complete HAL DLL, some other DLLs are needed too. Following the needed DLLs. Version number of used DLL in this examples are shown in brackets.

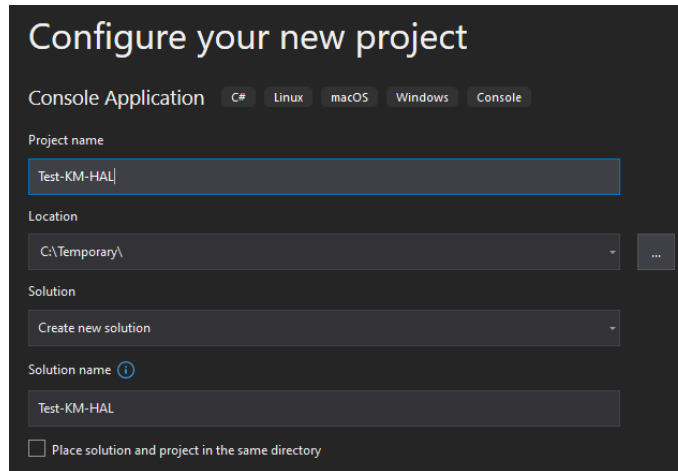
- Adlos.Formulator.dll (v1.0.0.1)
- Adlos.Framework.dll (v0.0.2.0)
- Adlos.KM.Framework.dll (v1.0.0.0)
- Adlos.KMAPI.HAL.dll (v0.9.0.0)
- Adlos.KMAPI.LLL.dll (v0.9.4.14)
- Adlos.Loader.dll (v1.2.1.0)

We will create a SerialCommunicator, read some device information, call some commands and subscribe to events.

This example project is available as *Test-KM-HAL*, and can be opened with Visual Studio.

Additional infos for protocol, API, etc. can be found under *Documentation & Links*.

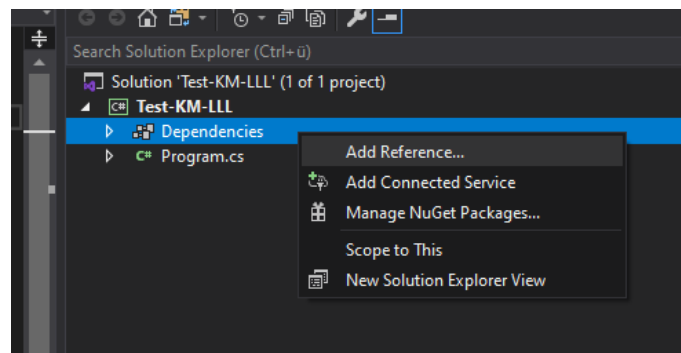
Create new Visual Studio Project



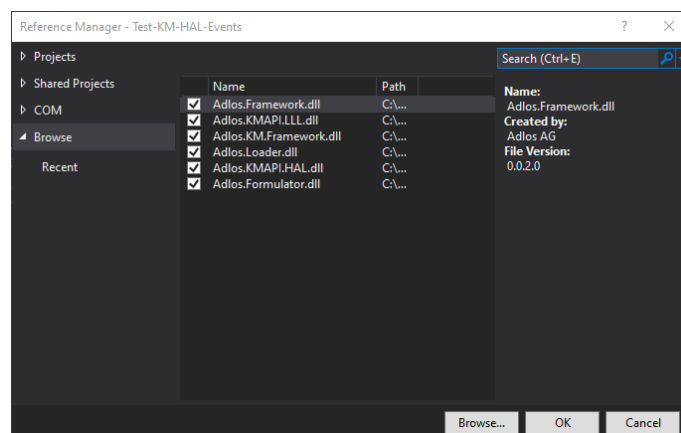
Then, click on **Next** and **Create**.

Include HAL and KM Framework library

In your project map on the right side, right click on **Dependencies** and **Add Reference**. In other versions of Visual Studio it can be called **Add Shared Project Reference**.



Next, go to **Browse** on the left side, click on **Browse** and select needed (as listed in Usage) then click **Add**.



Install needed NuGet

The LLL needs the *System.IO.Ports* NuGet. Again right click on **Dependencies** and click **Manage NuGet Packages...** Search for *System.IO.Ports* and install it.

Examples

Here you can find some coded examples to get to work with the HAL API library. You can find these examples in the *Test-KM-HAL* Visual Studio Project.

Code: Using directive

Allow the use of types in the namespace so that you do not have to qualify the use of the type in that namespace. These are the ones used in the example project.

```
using Adlos.KMAPI.HAL;  
using Adlos.KMAPI.HAL.Model.DataPoints;  
using Adlos.KM.Framework.Enumerations;  
using Adlos.KM.Framework.Interfaces.ComObjects.Serial;  
using Adlos.KM.Framework.Interfaces.Devices.Serial;
```

Code: Prevent downloading

```
// Flag, to either prevent or allow downloading of most recent Communication XML needed for devices  
// Has to be downloaded at least once!  
HALManager.PreventDownloading = false;
```

Code: Get HAL Version

```
// Get version of HAL library  
Version HalVersion = typeof(HALManager).Assembly.GetName().Version;
```

Code: Connect to a KannMOTION device

Either, you look for all available KannMOTION devices...

```
// Get KannMOTION devices  
List<ISerialCommunicator> serialCommunicators = HALManager.SearchDevices();
```

... or you search for explicit devices with COM-Ports as parameter.

```
// Get KannMOTION device on com port 7 and 16  
string[] comPorts = new string[2] { "COM7", "COM16" };  
List<ISerialCommunicator> explicitCommunicators = HALManager.SearchDevices(comPorts);  
  
// Get serial device on com port 7  
ISerialCommunicator serialCommunicator = explicitCommunicators.Find(d => d.Identifier == 7);
```

Code: Get SerialCommunicator informations

The *SerialCommunicator*-Object holds some information about the device directly available.

```
// Get hw article number (controller)
```

```
string hardwareArtNumber =
${SerialCommunicator.ArticleNumber}.{SerialCommunicator.ArticleIndex:000}";
// Serialnumber
int serialNumber = (int)serialCommunicator.SerialNumber;
// Firmware info
string fwArtNr =
${SerialCommunicator.Firmware.ArticleNumber}.{SerialCommunicator.Firmware.ArticleIndex:000}";
Version fwVersion = serialCommunicator.Firmware.Version;
```

Code: Get settings data

```
// Get settings information
string positioningUnit = "";
if (serialCommunicator.Settings.First(s => s.KMMTag == KMMTag.PositioningUnit) is ISettingPoint sp
&& sp != null)
{
    switch (sp.DeviceValue)
    {
        case 0: positioningUnit = "um"; break; // Mikrometer

        case 1: positioningUnit = "°"; break; // Grad

        default: positioningUnit = "Å"; break; // Unbekannte Einheit (Å)
    }
}
else
{
    // Unbekannte Einheit (Å)
    positioningUnit = "Å";
}
```

Code: Get information from TrackingData

TrackingData can be found with KMMTag, updated with Update() and if available, formatted value is calculated.

```
// Get more information from TrackingData
double systemArticleNumber = 0;
if (serialCommunicator.TrackingList.Any(trackingTable => trackingTable.Any(trackingPoint =>
trackingPoint.KMMTag == KMMTag.DUArticleNumber)))
{
    IDataTable trackingTable = serialCommunicator.TrackingList.First(tt => tt.Any(tp =>
tp.KMMTag == KMMTag.DUArticleNumber));
    trackingTable.Update();
    systemArticleNumber = ((UIntegerDataPoint)trackingTable.First(tp =>
tp.KMMTag == KMMTag.DUArticleNumber)).GetFormattedValue();
}

double totalRunTime = 0;
if (serialCommunicator.TrackingList.Any(tt => tt.Any(tp => (tp.KMMTag == KMMTag.TotalRuntime)
|| tp.Name.Contains("TotalRunTime"))))
{
    IDataTable trackingTable = serialCommunicator.TrackingList.First(tt => tt.Any(tp =>
(tp.KMMTag == KMMTag.TotalRuntime) || tp.Name.Contains("TotalRunTime")));
    trackingTable.Update();
    totalRunTime = ((UIntegerDataPoint)trackingTable.First(tp => (tp.KMMTag == KMMTag.TotalRuntime)
|| tp.Name.Contains("TotalRunTime"))).GetFormattedValue();
}
```

Get information from OnlineData

OnlineData can be found with KMMTag and updated with Update(). For easy use they can be gathered in a list and being updated as wanted.

Define and populate update list.

```
// Get more information from OnlineData
List<IDataTable> updateList = new List<IDataTable>();
string appState = "";
double temperature = 0;
double actualPosition = 0;
string analogInput = "";

foreach(IDataTable table in serialCommunicator.OnlineList)
{
    if (table.Any(p => p.KMMTag == KMMTag.Status || p.KMMTag == KMMTag.Temperature
        || p.KMMTag == KMMTag.Position || p.KMMTag == KMMTag.AnalogInput))
    {
        updateList.Add(table);
    }
}
```

Next, the tables are updated and data points can be used.

```
foreach (IDataTable table in updateList)
{
    if (table.Update())
    {
        foreach (IDataPoint dataPoint in table.Where(dp => dp.KMMTag != null))
        {
            switch (dataPoint.KMMTag)
            {
                case KMMTag.Status:
                    appState = (dataPoint as EnumDataPoint).GetParsedValue().Item2 ?? "N/A";
                    break;

                case KMMTag.Temperature:
                    temperature = (dataPoint as UByteDataPoint).GetFormattedValue();
                    break;

                case KMMTag.Position:
                    actualPosition = positioningUnit == "o" // else micrometer
                    ? ((dataPoint as SIntegerDataPoint).GetFormattedValue() / 10.0) // 0.1° in °
                    : (dataPoint as SIntegerDataPoint).GetFormattedValue();
                    break;

                case KMMTag.AnalogInput:
                    analogInput = $"{(dataPoint as UShortDataPoint).GetFormattedValue().ToString("0.0")}"
                    {dataPoint.Unit}";
                    break;
            };
        }
    }
    else
    {
        Console.WriteLine("Can't update selected device!");
    }
}
```

Code: Commands

Look for and use different commands.

```
// Rotate with -100rpm
serialCommunicator.CommandsList.FirstOrDefault(c => c.KMMTag == KMMTag.RotateCommand)
    .Send(new string[] {"-100"});

// Stop
serialCommunicator.CommandsList.FirstOrDefault(c => c.KMMTag == KMMTag.RotateCommand)
    .Send(new string[] { "0" });
```

```
// Get commands
ICommandTable goToPos = serialCommunicator.CommandsList.FirstOrDefault(c =>
    c.KMMTag == KMMTag.GoToPositionCommand);
ICommandTable homing = serialCommunicator.CommandsList.FirstOrDefault(c =>
    c.KMMTag == KMMTag.HomingCommand);

// Go to position -1000
goToPos.Send(new string[] { "1", "-1000" });

// Set zero position
homing.Send(new string[] { "0", "0", "0" });
```

Alternative handling of command in c#.

```
// Alternative handling of rotate command
Action<string[]> rotateAction = serialCommunicator.CommandsList.FirstOrDefault(c =>
    c.KMMTag == KMMTag.RotateCommand).Send;
Action<short> rotateFunction = new Action<short>(speed =>
    rotateAction.Invoke(new string[] { speed.ToString() }));

rotateFunction.Invoke(100);
```



Commands (GoToPos, Rotate, etc.) can only be used on drives, as long as there is no user sequence programmed.

Code: Close connection

Clode the connection.

```
// Close connection
KMConnection.Close();
```

Or close the connection and free all used system resources.

```
// ...or close connection and free all used system resources
KMConnection.Dispose();
```

Code: Subscribe to event (See example project Test-KM-HAL-Events)

Some drives provide events, here's how they can be used.

```
// Get desired event table
IEventTable eventTable = serialCommunicator.EventsList.First(et => et.Command == "DF#0D");

// Subscribe to event
eventTable.Subscribe();

// Register callback
eventTable.EventReceived += SerialCommunicator_EventReceived;

// Check if subscribed
Console.WriteLine($"Event is subscribed: {eventTable.Subscribed}");

// Unsubscribe from event
eventTable.Unsubscribe();

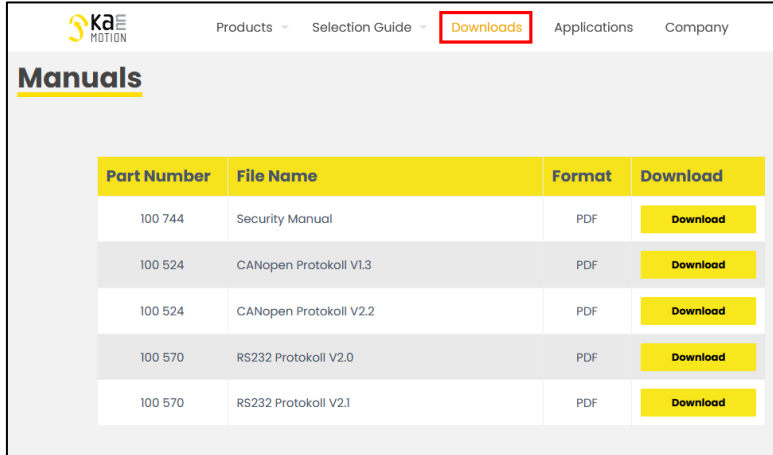
// Check if unsubscribed
Console.WriteLine($"Event is subscribed: {eventTable.Subscribed}");
```

Callback function, where events get in.

```
private static void SerialCommunicator_EventReceived(object sender, List<byte> args)
{
    byte[] data = { args[0], args[1], args[2], args[3] };
    Console.WriteLine("Received:\t{0}", BitConverter.ToUInt32(data));
}
```


Documentation & Links

API, Protocols and additional information can be found on kannmotion.com under downloads tab.



Part Number	File Name	Format	Download
100 744	Security Manual	PDF	Download
100 524	CANopen Protokoll V1.3	PDF	Download
100 524	CANopen Protokoll V2.2	PDF	Download
100 570	RS232 Protokoll V2.0	PDF	Download
100 570	RS232 Protokoll V2.1	PDF	Download

Art. Nr.	Description
100570	RS232 Protocol
100524	CANopen Protocol
300054	Protocol Simplification
190073	LLL API (Low Level Layer)
190074	HAL API (Hardware Abstraction Layer)
190080	BAL API (Bus Abstraction Layer)
300045	LLL Application Note
300046	HAL Application Note
300047	BAL Application Note

Additional Files	
HAL Sample Code	https://kannmotion.li/download/ANs/300046/AN300046_SampleCode.zip

Contact information

Adlos AG
Föhrenweg 14
FL-9496 Balzers

Thomas Vogt
Thomas.Vogt@adlos.com
Tel: +423 263 63 63

Countries: CH, A, LI, SK, IT
www.adlos.com

KOCO MOTION GmbH
Niedereschacher Straße 54
D-78083 Dauchingen

Olaf Kämmerling
O.Kaemmerling@kocomotion.de
Tel: +49 7720/995858-0

Countries: DE, BE, NL, LU
www.kocomotion.de