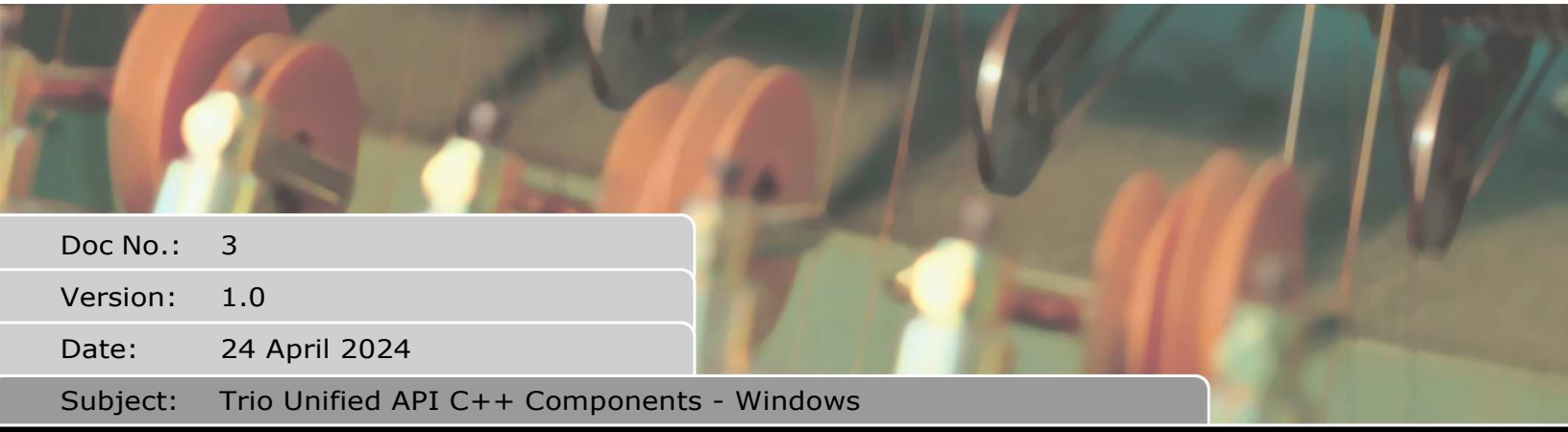
Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292333
Fax: +44 (0)1684 297929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412.968.9744
Fax: +1 412.968.9746

Tomson Centre
118 Zhang Yang Rd., B1701
Pudong New Area, Shanghai,
Postal code: 200122
P. R. CHINA
Tel/Fax: +86-21-58797659



Doc No.: 3
Version: 1.0
Date: 24 April 2024
Subject: Trio Unified API C++ Components - Windows

APPLICATION NOTE

www.triomotion.com

# Trio Unified API C++ Components - Windows

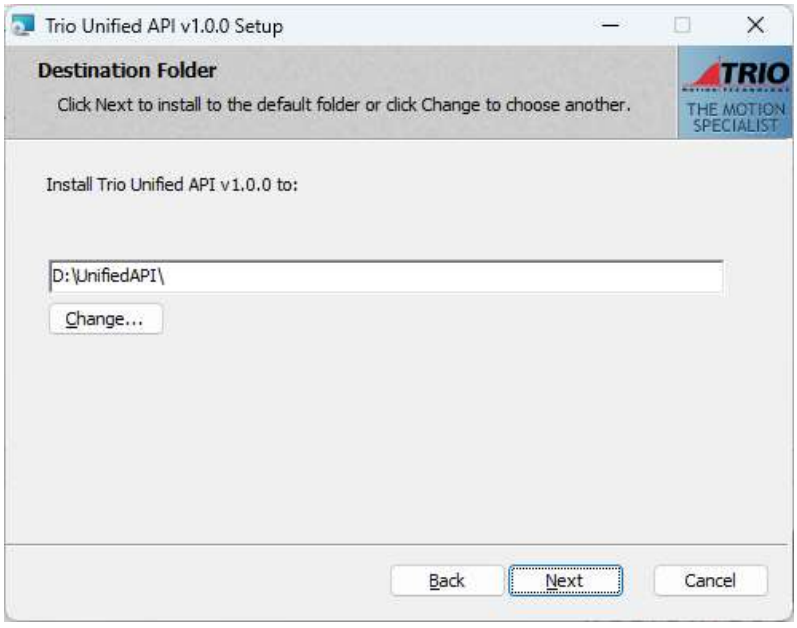
How to access controller via C++ using Unified API on Windows OS

## 1. Requirements

*1.1.* Windows10 system (recommended)
*1.2.* Visual Studio 2022 (required)
*1.3.* C++ Language Standard: ISO C++ Standard (/std:c++17)
*1.4.* Trio Unified API

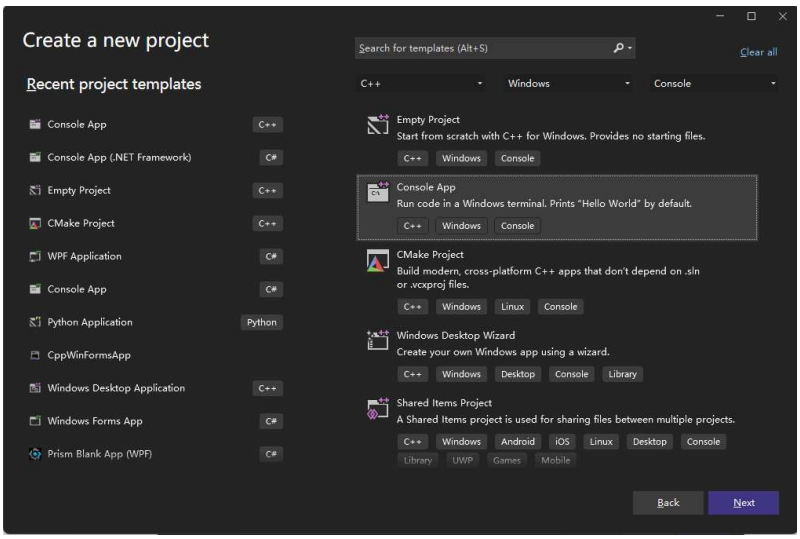
## 2. Project Setup

*2.1.* Install Trio Unified API in a folder on a local disk.

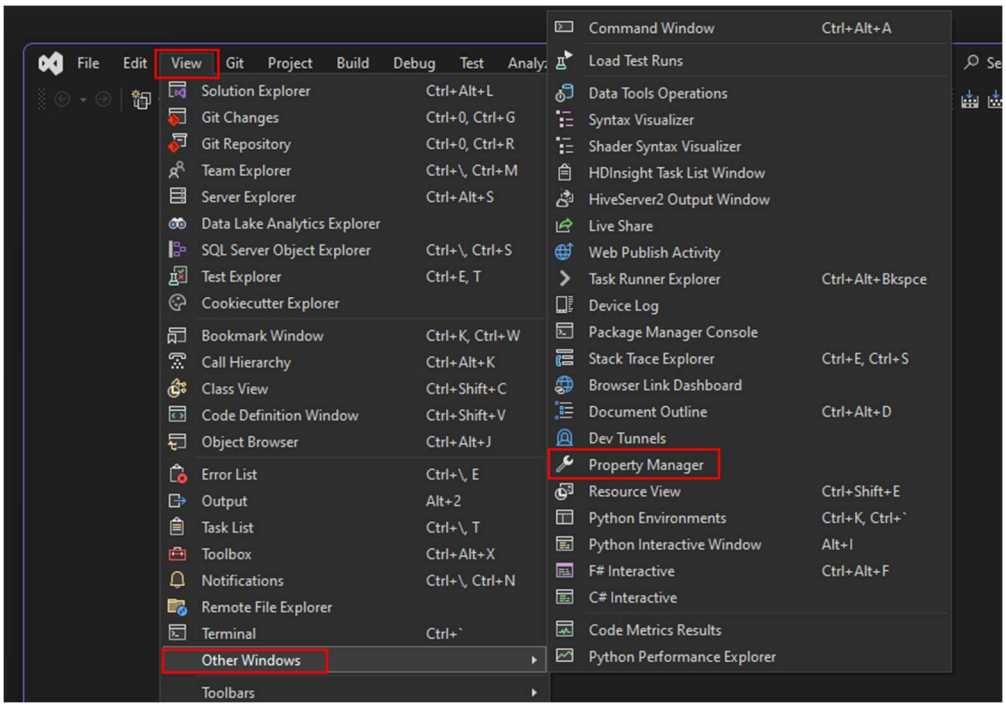


*2.2.* In this example will create a project in Visual Studio to test individual functions provided in Trio_UnifiedApi_CPP.h file.

*2.3.* Create a C++ Console App project with the name <UAPI_Test_C++> (for example) in a location on the windows disk.

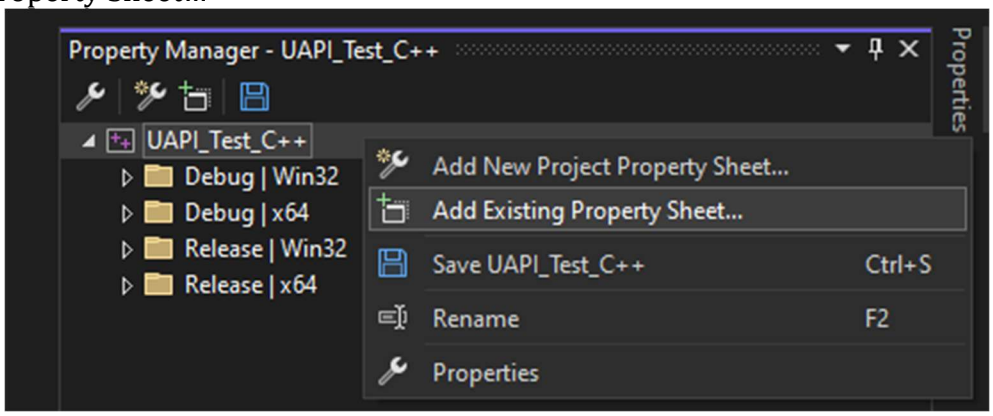



*2.4.* From Visual Studio menus navigate to 'View' -> 'Other Windows' and open 'Property Manager'.

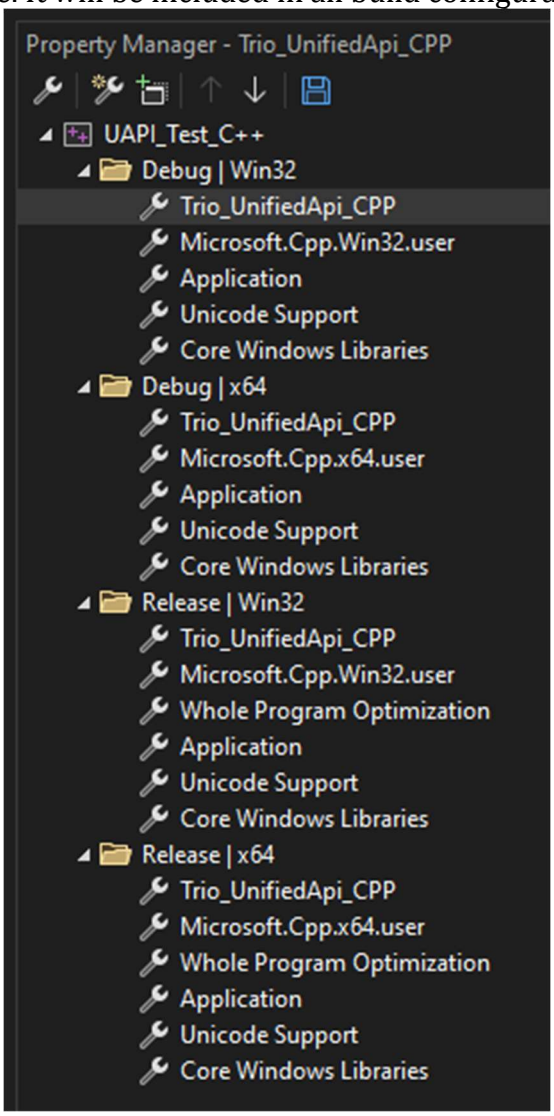



*2.5.* In 'Property Manager' right-click on 'UAPI_Test_C++' and select 'Add Existing

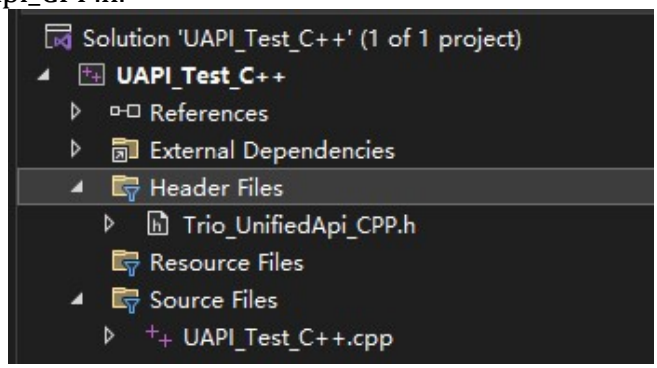
Property Sheet…’




*2.6.* Navigate to Trio Unified API local folder and select ‘Trio_UnifiedApi_CPP.props’ property sheet file. It will be included in all build configurations.



*2.7.* Add the Unified API header file to the project. Under Solution Explorer right click on Header Files and select “Add” -> “Existing Item…”. Browse and select Trio_UnifiedApi_CPP.h.




*2.8.* Include the Trio_UnifiedApi_CPP.h in UAPI_Test_C++.cpp.

## 3. Example code

```
#include <Trio_UnifiedApi_CPP.h>

namespace TUA = Trio::UnifiedApi; // To make the namespace shorter
```

*3.1.* Create the callback function.

```
// Connection callback
void connection_callback(TUA::EventType event_type, uint64_t int_value, const
std::string_view& str_value)
{
	switch (event_type)
	{
	case TUA::EventType::Error:
		std::cout << "Error [" << std::hex << int_value << "] occurred : " <<
str_value << std::endl;
		break;

	case TUA::EventType::Message:
		std::cout << "Msg: " << str_value << std::endl;
		break;

	default:
		break;
	}
}
```

*3.2.* Create the connection to the MC.

```
TUA::ITrioConnection* _connection;
std::string mcIP = "127.0.0.1"; // The IP address of the controller.
// Create connection context of TCP type
_connection = TUA::TrioConnectionFactory::CreateTCP(connection_callback, mcIP);
_connection->OpenConnection();
```

*3.3.* Writing and reading variables, including the VR/TABLE/AxisParameter/SystemParameter.

```
// Set VR value
uint32_t vrIndex = 100;
double vrValue = 1.25;
_connection->SetVrValue(vrIndex, vrValue);

//Get VR vlaue
double val = _connection->GetVrValue(vrIndex);
if (val != vrValue)
{
	std::cout << "GetVrValue returned error " << std::endl;
}

// Set TABLE value
uint32_t tbIndex = 100;
double tbValue = 2.34;
_connection->SetTableValue(tbIndex, tbValue);

//Get TABLE vlaue
double tbRd = _connection->GetTableValue(tbIndex);
if (tbRd != tbValue)
{
	std::cout << "GetTableValue returned error " << std::endl;
}

//Set AxisParamter - UNITS
uint32_t axisIndex = 0;
double unitsVal = 100;
_connection->SetAxisParameter_UNITS(axisIndex, unitsVal);

//Get AxisParameter - UNITS
double unitsRd = _connection->GetAxisParameter_UNITS(axisIndex);
if (unitsRd != unitsVal)
{
	std::cout << "GetAxisParameter_UNITS returned error " << std::endl;
}

//Set AxisParamter - SPEED
double speedVal = 1000;
_connection->SetAxisParameter_SPEED(axisIndex, speedVal);

//Get AxisParameter - SPEED
double speedRd = _connection->GetAxisParameter_SPEED(axisIndex);
if (speedRd != speedVal)
{
	std::cout << "GetAxisParameter_SPEED returned error " << std::endl;
}

//Set SystemParamter - WDOG
uint32_t wdog = 1;
_connection->SetSystemParameter_WDOG(wdog);
//Get SystemParameter - WDOG
uint32_t wdogRd = _connection->GetSystemParameter_WDOG();
if (wdogRd != wdog)
{
```

```
	std::cout << "GetSystemParameter_WDOG returned error " << std::endl;
}
```

### *3.4.* Digital IO operations.

```
//Set digital output
//Set digital output
_connection->SetDOut(0, true);

//Get digital output
bool doutRd = _connection->GetDOut(0);
if (!doutRd)
{
	std::cout << "GetDOut returned error " << std::endl;
}

//Get the input of channel 0.
bool dinRd = false;
dinRd = _connection->GetDIn(0);
```

### *3.5.* Analog IO operations

```
//Set analog output
int iOut = 100;
_connection->SetAOut(0, iOut);

//Get analog output
int aoutRd = _connection->GetAOut(0);
if (aoutRd != iOut)
{
	std::cout << "GetAOut returned error " << std::endl;
}

//Get the analog input of channel 0.
int ainRd = -1;
ainRd = _connection->GetAIn(0);
```

### *3.6.* Motion commands.

```
//Single axis move
double pos = 1000;
uint32_t axis = 0;
_connection->MoveRel(pos, axis); // Axis(0) moves the distance 1,000( relative
position)

//Two axes linear interpolation motion( relative position)
double positions[] = { 1000, 2000 };
uint32_t axes[] = { 0, 2 };
_connection->SetMultiAxisBase(axes, 2); // Set the base axes.
_connection->MoveRel(positions, 2, axis); // Axis(0) moves the distance 1,000, while
axis(2) moves 2,000

//Sets continuous forward movement.
int mode = 2;
_connection->Cancel(mode, axis); // Cancels all active and buffered moves
```

```
_connection->Forward(axis); // Axis(0) moves foward
sleep_ms(3000); // Waiting for 3 seconds.
mode = 0;
_connection->Cancel(mode, axis); // Cancel the currently executing move

//Sets continuous reverse movement.
_connection->Reverse(axis); // Axis(0) moves foward
sleep_ms(3000); // Waiting for 3 seconds.
_connection->Cancel(mode, axis); // Cancel the currently executing move

//Two axes circular interpolation motion
_connection->SetMultiAxisBase(axes, 2); // Set the base axes.
_connection->MoveCirc(0, 2000, 0, 1000, 0, 1); /* Semicircular trajectory(axis0 and
axis2, clockwise direction), finish point (2000,0), centre point (1000,0) */
sleep_ms(5000); // waiting for 5 seconds
```

*3.7.* Disconnect from the controller.

```
_connection->CloseConnection();
```