

Programming Guide

UnitechRFID Windows

unitech electronics Documentation
Issue4, revision 0
Jan 2024

Revision History

Release	Revision	Date	Changes
4	0	2024-01-24	Update TagReadEventArgs
3	0	2023-06-28	Add DisplayTags
2	0	2023-06-12	Add DisplayOutput Add AutoScreenOffTime
1	0	2023-03-30	Release

Table of Contents

TABLE OF CONTENTS.....	I
1. PREFACE	1
1.1 PREREQUISITES	2
1.2 CREATE PROJECT FOR UNITECHRFID WINDOWS	3
1.2.1 <i>Create Visual Studio Project</i>	<i>3</i>
1.2.2 <i>Add Reference.....</i>	<i>5</i>
2. TRANSPORT CLASSES	8
2.1 BASETRANSPORT	8
2.1.1 <i>Address</i>	<i>8</i>
2.1.2 <i>ConnectType</i>	<i>8</i>
2.1.3 <i>MacAddress</i>	<i>8</i>
2.2 TRANSPORTSERIAL	9
2.2.1 <i>BaudRate</i>	<i>9</i>
2.2.2 <i>Constructors.....</i>	<i>9</i>
2.2.3 <i>DataBits</i>	<i>10</i>
2.2.4 <i>ParityEvent.....</i>	<i>10</i>
2.2.5 <i>StopBits.....</i>	<i>10</i>
2.3 TRANSPORTBLUETOOTH	10
2.3.1 <i>Constructors.....</i>	<i>10</i>
3. UHFREADER.....	11
3.1 PROPERTIES	11
3.1.1 <i>ActionState</i>	<i>11</i>
3.1.2 <i>Address</i>	<i>11</i>
3.1.3 <i>AutoOffTime</i>	<i>11</i>
3.1.4 <i>AutoOffTimeList.....</i>	<i>11</i>
3.1.5 <i>AutoScreenOffTime.....</i>	<i>12</i>
3.1.6 <i>BaseTransport.....</i>	<i>12</i>
3.1.7 <i>BaseUHF</i>	<i>12</i>
3.1.8 <i>BatteryState.....</i>	<i>12</i>
3.1.9 <i>Beeper.....</i>	<i>12</i>

3.1.10	<i>CheckInterval</i>	13
3.1.11	<i>ConnectState</i>	13
3.1.12	<i>ConnectType</i>	13
3.1.13	<i>DataFormat</i>	13
3.1.14	<i>DataInfo</i>	13
3.1.15	<i>DataRange</i>	14
3.1.16	<i>DataTerminator</i>	14
3.1.17	<i>DetectDevice</i>	14
3.1.18	<i>DeviceName</i>	14
3.1.19	<i>DeviceType</i>	14
3.1.20	<i>DisplayOutput</i>	15
3.1.21	<i>DisplayTags</i>	15
3.1.22	<i>FindDevice</i>	15
3.1.23	<i>IsConnected</i>	15
3.1.24	<i>LogLevel</i>	16
3.1.25	<i>MacAddress</i>	16
3.1.26	<i>OperatingMode</i>	16
3.1.27	<i>ReadMode</i>	16
3.1.28	<i>SerialNumber</i>	16
3.1.29	<i>SKU</i>	17
3.1.30	<i>Time</i>	17
3.1.31	<i>Temperature</i>	17
3.1.32	<i>Version</i>	17
3.1.33	<i>Vibrator</i>	17
3.2	METHODS	18
3.2.1	<i>Constructors</i>	18
3.2.2	<i>Connect</i>	18
3.2.3	<i>Dispose</i>	18
3.2.4	<i>Disconnect</i>	18
3.2.5	<i>FactoryReset</i>	18
3.2.6	<i>StartDetect</i>	19
3.2.7	<i>StopDetect</i>	19
3.3	EVENTS	19
3.3.1	<i>ActionStateChangedEvent</i>	19
3.3.2	<i>BatteryStateEvent</i>	19

3.3.3	<i>ConnectStateChangedEvent</i>	20
3.3.4	<i>DetectDeviceEvent</i>	20
3.3.5	<i>DetectFinishedEvent</i>	20
3.3.6	<i>KeyStateEvent</i>	20
3.3.7	<i>LogEvent</i>	20
3.3.8	<i>TemperatureEvent</i>	21
4.	BASEUHF	22
4.1	PROPERTIES	22
4.1.1	<i>AlgorithmType</i>	22
4.1.2	<i>BLF</i>	22
4.1.3	<i>ContinuousMode</i>	22
4.1.4	<i>Encoding</i>	23
4.1.5	<i>FastMode</i>	23
4.1.6	<i>GlobalBand</i>	23
4.1.7	<i>IdleTime</i>	23
4.1.8	<i>InventoryTime</i>	24
4.1.9	<i>MaxQ</i>	24
4.1.10	<i>MinQ</i>	24
4.1.11	<i>ModuleProfile</i>	25
4.1.12	<i>Power</i>	25
4.1.13	<i>PowerMode</i>	25
4.1.14	<i>PowerRange</i>	25
4.1.15	<i>RFIDConfig</i>	26
4.1.16	<i>SelectMask6cMaxSize</i>	26
4.1.17	<i>Session</i>	26
4.1.18	<i>StartQ</i>	26
4.1.19	<i>Target</i>	27
4.1.20	<i>TARI</i>	27
4.1.21	<i>ToggleTarget</i>	27
4.2	METHODS	27
4.2.1	<i>GetSelectMask6cEnabled</i>	27
4.2.2	<i>SetSelectMask6cEnabled</i>	28
4.2.3	<i>GetSelectMask6cTarget</i>	28
4.2.4	<i>SetSelectMask6cTarget</i>	28
4.2.5	<i>GetSelectMask6cAction</i>	29

4.2.6	<i>SetSelectMask6cAction</i>	29
4.2.7	<i>GetSelectMask6cBank</i>	29
4.2.8	<i>SetSelectMask6cBank</i>	30
4.2.9	<i>GetSelectMask6cOffset</i>	30
4.2.10	<i>SetSelectMask6cOffset</i>	30
4.2.11	<i>GetSelectMask6cPattern</i>	31
4.2.12	<i>SetSelectMask6cPattern</i>	31
4.2.13	<i>SetSelectMask6c</i>	31
4.2.14	<i>GetSelectMask6c</i>	32
4.2.15	<i>Inventory6c</i>	32
4.2.16	<i>ReadMemory6c</i>	32
4.2.17	<i>WriteMemory6c</i>	33
4.2.18	<i>Lock6c</i>	33
4.2.19	<i>PermaLock6c</i>	34
4.2.20	<i>Kill6c</i>	34
4.2.21	<i>Stop</i>	34
4.3	EVENTS.....	34
4.3.1	<i>AccessResultEvent</i>	34
4.3.2	<i>TagReadEvent</i>	35
5.	EVENTARGS CLASSES	36
5.1	DETECTDEVICEEVENTARGS.....	36
5.1.1	<i>BTMACAddress</i>	36
5.1.2	<i>ConnectType</i>	36
5.1.3	<i>DeviceType</i>	36
5.1.4	<i>Name</i>	36
5.1.5	<i>Port</i>	36
5.1.6	<i>SerialNumber</i>	37
5.2	LOGEVENTARG.....	37
5.2.1	<i>DateTime</i>	37
5.2.2	<i>LogLevel</i>	37
5.2.3	<i>Tag</i>	37
5.2.4	<i>Log</i>	37
5.3	ACTIONSTATEEVENTARGS.....	38
5.3.1	<i>ActionState</i>	38
5.3.2	<i>Reader</i>	38

5.4	BATTERYSTATEEVENTARGS.....	38
5.4.1	Reader	38
5.4.2	Battery.....	38
5.5	KEYEVENTARGS.....	39
5.5.1	Reader	39
5.5.2	KeyState.....	39
5.5.3	KeyType.....	39
5.6	CONNECTSTATEEVENTARGS	39
5.6.1	Reader	39
5.6.2	ConnectState	39
5.7	TEMPERATUREEVENTARGS	40
5.7.1	Reader	40
5.7.2	Temperature	40
5.8	TAGREADEVENTARGS	40
5.8.1	Antenna	40
5.8.2	EPC.....	40
5.8.3	Frequency	40
5.8.4	PC.....	41
5.8.5	Q	41
5.8.6	RawEPC.....	41
5.8.7	RawPC.....	41
5.8.8	RawTID	41
5.8.9	RSSI.....	41
5.8.10	TID	42
5.9	ACCESSRESULTEVENTARGS	42
5.9.1	EPC.....	42
5.9.2	Data	42
5.9.3	ActionState	42
5.9.4	BankType	42
5.9.5	ResultCode.....	43
6.	FINDDEVICE	44
6.1	MODE	44
6.2	TIMEOUT	44
7.	POWERRANGE.....	45

7.1	MIN	45
7.2	MAX	45
8.	SELECTMASK6CPARAM	46
8.1	ISUSED	46
8.2	TARGET	46
8.3	ACTION	46
8.4	BANK	46
8.5	OFFSET	46
8.6	PATTERN	47
8.7	LENGTH	47
9.	RFIDCONFIG.....	48
9.1	CONTINUOUSMODE	48
9.2	TARGET	48
9.3	SESSION	48
9.4	ENCODING	48
9.5	TARITYPE	48
9.6	BLFTYPE.....	49
9.7	ALGORITHM.....	49
9.8	FASTMODE.....	49
9.9	TOGGLETARGET	49
9.10	MODULEPROFILE.....	49
9.11	POWER.....	50
9.12	INVENTORYTIME.....	50
9.13	IDLETIME	50
9.14	STARTQ	50
9.15	MAXQ.....	50
9.16	MINQ	51
10.	PERMALOCK6CPARAM	52
10.1	KILLPASSWORD	52
10.2	ACCESSPASSWORD	52
10.3	EPC	52
10.4	TID.....	52
10.5	USER.....	52

11.	LOCK6CPARAM.....	53
11.1	KILLPASSWORD	53
11.2	ACCESSPASSWORD	53
11.3	EPC	53
11.4	TID.....	53
11.5	USER.....	53
12.	MASK6CPATTERN	54
12.1	PATTERN	54
12.2	LENGTH	54
13.	DATARANGE	55
13.1	OFFSET	55
13.2	LENGTH	55
14.	ENUMS.....	56
14.1	ACTIONSTATE	56
14.2	ALGORITHMTYPE	56
14.3	BEEPERSTATE	57
14.4	BANKTYPE.....	57
14.5	BLFTYPE.....	57
14.6	BEEPANDVIBRATESTATE	58
14.7	CONNECTTYPE	58
14.8	CONNECTSTATE	59
14.9	DEVICETYPE.....	59
14.10	DATAFORMAT	59
14.11	DATAINFO	59
14.12	DATATERMINATOR.....	60
14.13	ENCODING	60
14.14	FINDDEVICEMODE.....	60
14.15	GLOBALBANDTYPE.....	60
14.16	KEYTYPE	61
14.17	KEYSTATE	61
14.18	LOCKSTATE	61
14.19	LOGLEVEL.....	62
14.20	MASK6cACTION.....	62

14.21	MASK6CTARGET.....	63
14.22	OPERATINGMODE.....	63
14.23	POWERMODE.....	64
14.24	RESULTCODE.....	64
14.25	READMODE.....	71
14.26	READONCESTATE.....	71
14.27	SESSION.....	71
14.28	SKUTYPE.....	71
14.29	TARGET.....	72
14.30	TARITYPE.....	72
14.31	VIBRATORSTATE.....	73

1. Preface

This document describes the functionalities of the RFID Windows SDK. The SDK is delivered through the C#-based library “unitechRFID.dll” for user with the supported portable RFID readers from unitech.

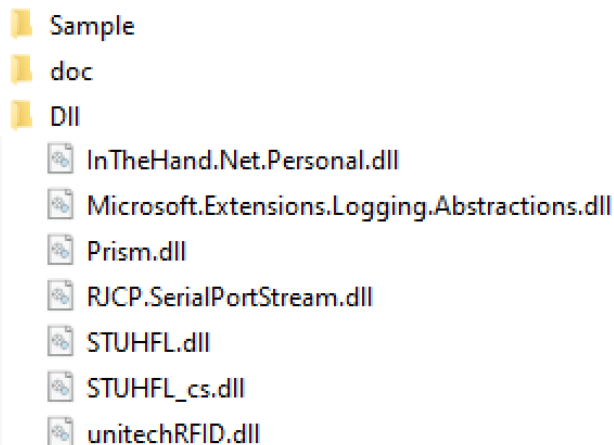
The “unitechRFID.dll” library is consisted of three key functional areas:

- **The TransportSerial/TransportBluetooth classes** – the unitech portable RFID readers use serial/Bluetooth connections to communicate with the PC. These classes manage the connection between the reader and the PC.
- **The BaseUHF class** –Developers can use these APIs for the read functions; tag operations; privacy and security; and the parameters for performance optimization.
- **The UHFReader class** – for simple and rapid setup and control of the supported unitech portal RFID readers. Developers can use these APIs for the read functions; tag operations; privacy and security; and the parameters for performance optimization.

The key work flow in getting the unitech portal RFID reader connected with your application consists of the following key steps:

- **Initialization Phase**
Initialize TransportSerial or TransportBluetooth and then pass it to UHFReader.
- **Connect Phase**
Use the API Connect() in UHFReader to make the connection with RFID reader.
- **Operational Phase**
Use the UHFReader APIs to setup and control the device or BashUHF in UHFReader to setup and control the RFID module on unitech portable RFID peripheral.

The figure below shows the structure of SDK. The *Dll* folder includes the main library unitechRFID and dependency libraries, and the *Sample* folder contains the source codes of demo application and the executable sample is included with the source codes.



Folders & Files	Description
/unitechRFID Windows	The SDK package
/ unitechRFID Windows/Sample/	Include the sample app project and code
/ unitechRFID Windows/Sample/C#/ unitechRFIDSample/bin/Release/netcoreapp3.1/unitechRFIDSample.exe	The built sample app for developer to quickly get started with sample code
/ unitechRFID Windows/ doc/unitechRFID_Windows_Programming_Guide_Issue1.pdf	To demonstrate how to import .dll to the project and SDK guide.
/ unitechRFID Windows/Dll/	The .dll for controlling UHF and reader.

1.1 Prerequisites

The SDK supports the following RFID Reader peripherals:

- RP902 with FW 1.0.12 or above

The Visual Studio:

- Visual Studio 2019 or above

The NuGet Packages:

- 32feet.NET 3.5.0 or above
- SerialPortStream 2.4.0 or above

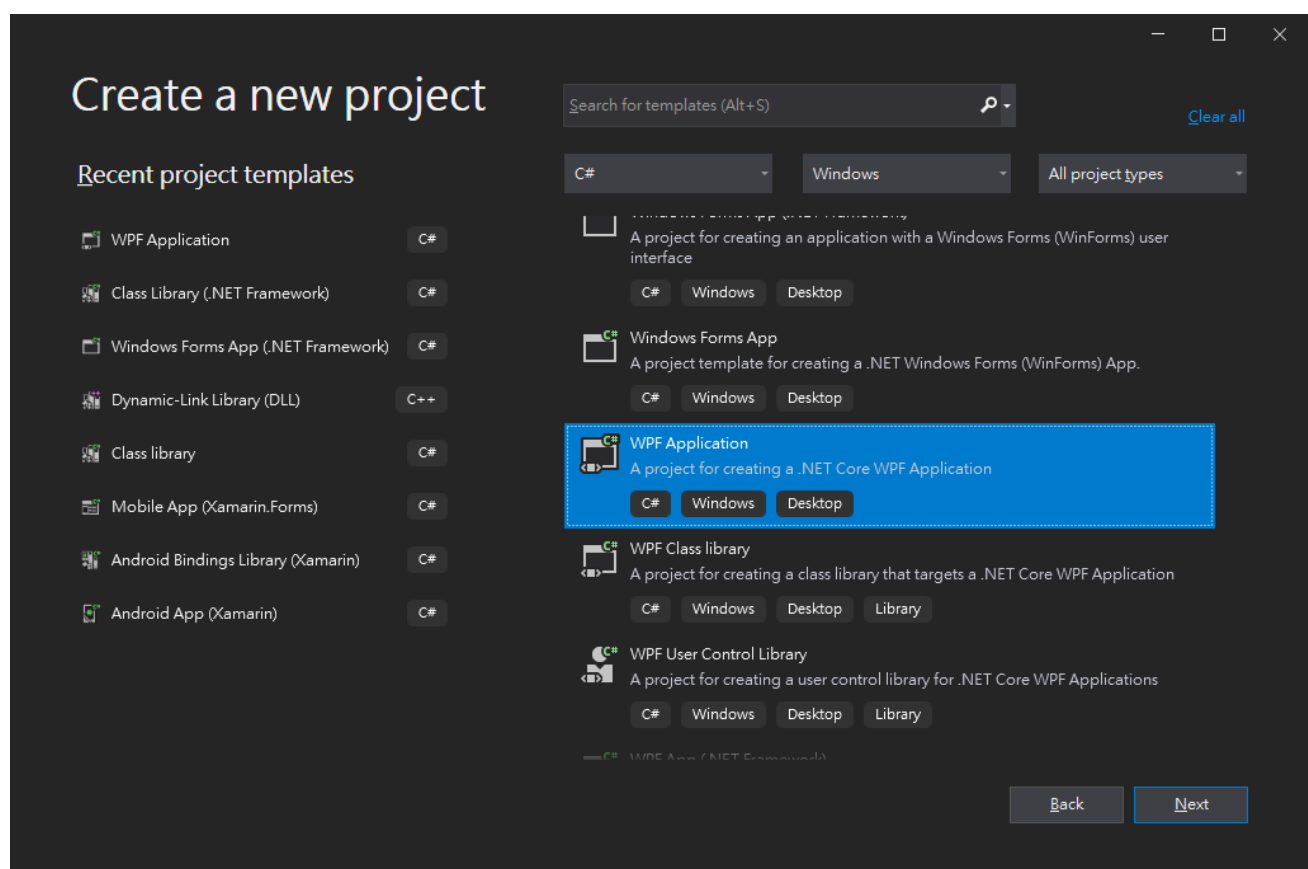
The target framework:

- .NET Framework 4.7.2 or above

1.2 Create Project for unitechRFID Windows

1.2.1 Create Visual Studio Project

Create a new WPF application in Visual Studio 2019.



Name your project

Configure your new project

WPF Application C# Windows Desktop

Project name

WpfApp1

Location

C:\Users\user\source\repos

Solution name ⓘ

WpfApp1

☒ Place solution and project in the same directory

Back Next

Setup target framework to .NET Core 3.1

Additional information

WPF Application C# Windows Desktop

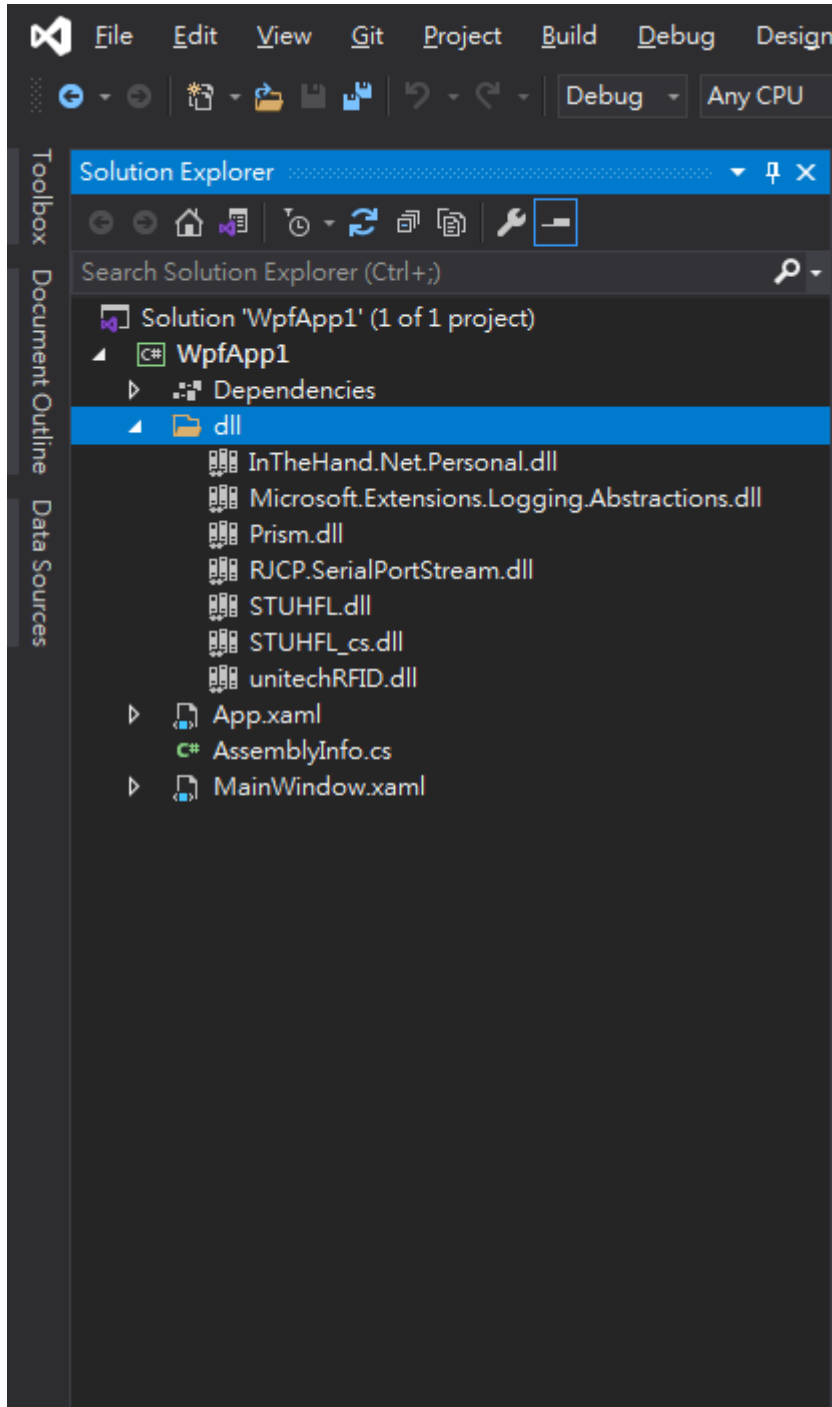
Target Framework ⓘ

.NET Core 3.1 (Out of support)

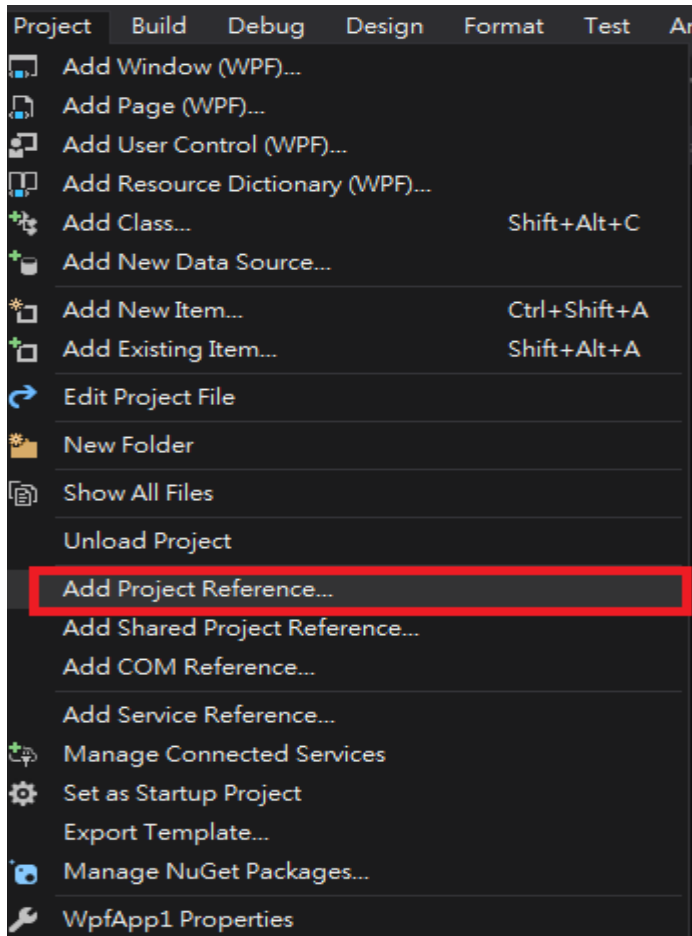
Back Create

1.2.2 Add Reference

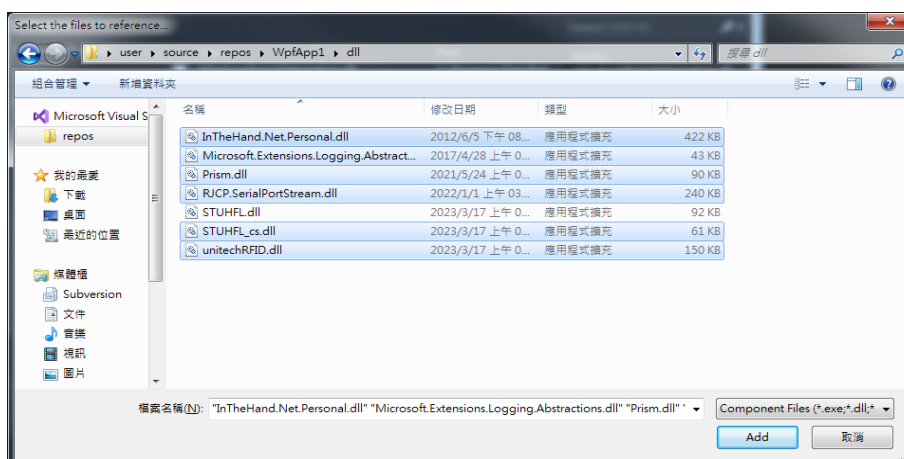
Create “dll” folder under the project and copy the libraries to it.



Project->Add Project Reference... to add the library for application

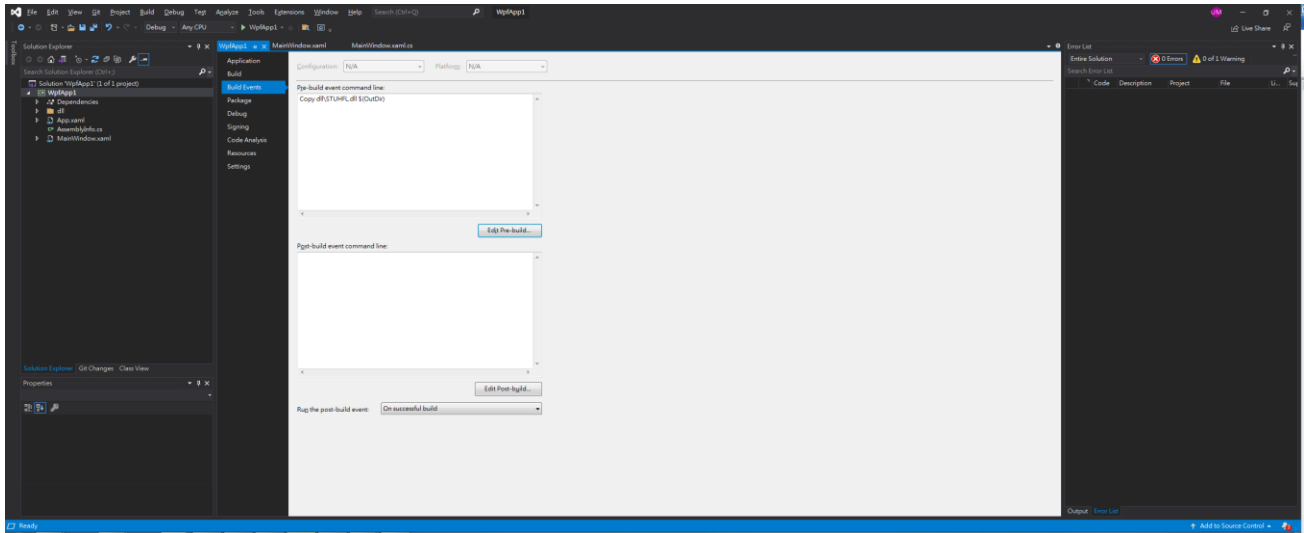


Select the dll files from SDK exclude STUHF.dll, because of it's C++ library.



In Project Properties->Build Events->Pre-build event command line:

Add "Copy dll\\STUHFL.dll \$(OutDir)" in Edit Pre-build to copy STUHFL.dll to output folder



2. Transport Classes

Transport class defines the communication with the RFID device for UHFReader. If we use the serial communication with RFID device, we need to use TransportSerial for UHFReader. If we use the Bluetooth communication, we need to use TransportBluetooth for UHFReader.

2.1 BaseTransport

BaseTransport is the parent of TransportSerial and TransportBluetooth.

2.1.1 Address

Definition	string Address
Description	Get/Set the serial port address of the device.
Example	baseTransport.Address = "COM3";

2.1.2 ConnectType

Definition	ConnectType ConnectType
Description	Get/Set the communication type with the device.
Example	baseTransport.ConnectType = ConnectType.Serial;

2.1.3 MacAddress

Definition	string MacAddress
Description	Get/Set the Bluetooth MAC address of the device.
Example	baseTransport.MacAddress = "AABBCCDDEEFF";

2.2 TransportSerial

TransportSerial is the class for UHFReader if the communication with RFID device is serial connection.

2.2.1 BaudRate

Definition	int BaudRate
Description	Get/Set the baud rate of serial communication.
Example	transportSerial.BaudRate = 115200;

2.2.2 Constructors

Definition	TransportSerial(DeviceType deviceType, string address, int baudRate, int dataBits, Parity parity, StopBits stopBits)
Parameters	deviceType: The type of target device. address: The serial port address of the device. baudRate: The baud rate of the device. dataBits: The data bits of the device. parity: The parity of the device. stopBits: The stop bits of the device.
Example	var transport = new TransportSerial(DeviceType.RP902, "COM3", 115200, 8, Parity.None, StopBits.One);
Note	The baud rate, data bits, parity and stop bits for RP902 is 115200, 8, None, One.

Definition	TransportSerial(DeviceType deviceType, string address, int baudRate)
Parameters	deviceType: The type of target device. address: The serial port address of the device. baudRate: The baud rate of the device.
Example	var transport = new TransportSerial(DeviceType.RP902, "COM3", 115200);
Note	The data bits, parity and stop bits are 8, None, One.

2.2.3 DataBits

Definition	int DataBits
Description	Get/Set the data bits of serial communication.
Example	transportSerial.DataBits = 8;

2.2.4 ParityEvent

Definition	Parity Parity
Description	Get/Set the parity of serial communication.
Example	transportSerial.Parity = Parity.None;

2.2.5 StopBits

Definition	StopBits StopBits
Description	Get/Set the stop bits of serial communication.
Example	transportSerial.StopBits = StopBits.One;

2.3 TransportBluetooth

TransportBluetooth is the class for UHFReader if the communication with RFID device is Bluetooth.

2.3.1 Constructors

Definition	TransportBluetooth(DeviceType deviceType, string btMACAddress)
Parameters	deviceType: The type of target device. btMACAddress: The Bluetooth MAC address of the device.
Example	var transport = new TransportBluetooth(DeviceType.RP902, "AABBCCDDEEFF");

3. UHFReader

UHFReader provides the methods to connect and control the RFID device.

3.1 Properties

3.1.1 ActionState

Definition	ActionState ActionState
Description	Get the state of the device.

3.1.2 Address

Definition	string Address
Description	Get the address of the device, if the communication is serial, it will be the name of serial port.

3.1.3 AutoOffTime

Definition	int AutoOffTime
Description	Get/Set the auto off time setting of the device.
Example	reader.AutoOffTime = 5;

3.1.4 AutoOffTimeList

Definition	int[] AutoOffTimeList
Description	Get the list of auto off time setting of the device. The unit is minute.
Note	RP902 supports 0, 1, 2, 3, 5, 10 and 0 is for disable.

3.1.5 AutoScreenOffTime

Definition	ScreenOffTime ScreenOffTime
Description	Get/Set the auto screen off time setting of the device.
Example	reader.ScreenOffTime = new ScreenOffTime(Minutes, Seconds);
Note	RP902 supports (0, 0), (0, 30), (1, 0), (2, 0), (3, 0), (5, 0), (10, 0) and (0, 0) is for disable.

3.1.6 BaseTransport

Definition	BaseTransport BaseTransport
Description	The BaseTransport of the UHFReader.

3.1.7 BaseUHF

Definition	BaseUHF BaseUHF
Description	The interface to control RFID module on the device.

3.1.8 BatteryState

Definition	int BatteryState
Description	Get the battery status of the device.

3.1.9 Beeper

Definition	BeeperState Beeper
Description	Get/Set the beeper setting of the device.
Example	reader.Beeper = BeeperState.Off;

3.1.10 CheckInterval

Definition	int CheckInterval
Description	Get/Set the time period for checking the battery and temperature status of the device. The unit is millisecond.
Example	reader.CheckInterval = 3000;
Note	The default value is 3000 for 3 seconds.

3.1.11 ConnectState

Definition	ConnectState ConnectState
Description	Get the state of connection with the device.

3.1.12 ConnectType

Definition	ConnectType ConnectType
Description	Get the communication type with the device.

3.1.13 DataFormat

Definition	DataFormat DataFormat
Description	Get/Set the data format setting of the device.
Example	reader.DataFormat = DataFormat.HEX

3.1.14 DataInfo

Definition	DataInfo DataInfo
Description	Get/Set the data info string of the device.
Example	reader.DataInfo = DataInfo.EPC;

3.1.15DataRange

Definition	DataRange DataRange
Description	Get/Set the data range setting of the device.
Example	reader.DataRange = new DataRange(0, 2);

3.1.16DataTerminator

Definition	DataTerminator DataTerminator
Description	Get/Set the data terminator setting of the device.
Example	reader.DataTerminator = DataTerminator.CR_LF;

3.1.17DetectDevice

Definition	bool DetectDevice
Description	Get the detect device state of the UHFReader

3.1.18DeviceName

Definition	string DeviceName
Description	Get the name of the device.

3.1.19DeviceType

Definition	DeviceType DeviceType
Description	Get the type of the device.

3.1.20 DisplayOutput

Definition	DisplayOutput DisplayOutput
Description	Set the display of the device output.
Example	reader.DisplayOutput = new DisplayOutput(Parameter, Data);
Note	Parameter: ● bit 0-3 character position (0~15), ● bit 5 clean screen before output (0:disable; 1:enable), ● bit 6-7 line position (1~3) Data: ASCII characters (MAX 16 bytes).

3.1.21 DisplayTags

Definition	DisplayTags DisplayTags
Description	Set the tags display of the device output.
Example	reader.DisplayTags= new DisplayTags(ReadOnceState .On, BeepAndVibrateState .On);

3.1.22 FindDevice

Definition	FindDevice FindDevice
Description	Set find device setting of the device.
Example	reader.FindDevice = new FindDevice(FindDeviceMode .VibrateBeep, 10);

3.1.23 IsConnected

Definition	bool IsConnected
Description	Get the connection state with the device.

3.1.24LogLevel

Definition	LogLevel LogLevel
Description	Get/Set the log level setting of the SDK.
Example	reader.LogLevel = LogLevel.Info;

3.1.25MacAddress

Definition	string MacAddress
Description	Get the Bluetooth MAC address of the device.

3.1.26OperatingMode

Definition	OperatingMode OperatingMode
Description	Get/Set the operating mode setting of the device.
Example	reader.OperatingMode = OperatingMode.USBSP; ;

3.1.27ReadMode

Definition	ReadMode ReadMode
Description	Get/Set the read mode of the device.
Example	reader.ReadMode = ReadMode.SingleRead;

3.1.28SerialNumber

Definition	string SerialNumber
Description	Get the serial number of the device.

3.1.29SKU

Definition	SKUType SKU
Description	Get the SKU setting of the device.

3.1.30Time

Definition	DateTime Time
Description	Get/Set the system time of the device.
Example	reader.Time = DateTime.Now;

3.1.31Temperature

Definition	double Temperature
Description	Get the temperature status of the device.

3.1.32Version

Definition	string Version
Description	Get the firmware version of the device.

3.1.33Vibrator

Definition	VibratorState Vibrator
Description	Get/Set the vibrator setting of the device.
Example	reader.Vibrator = VibratorState.Off;

3.2 Methods

3.2.1 Constructors

Definition	UHFReader(BaseTransport baseTransprot)
Parameters	baseTransport: The communication type of target device

3.2.2 Connect

Definition	bool Connect()
Description	Connect with device.

3.2.3 Dispose

Definition	void Dispose()
Description	Release the resource in UHFReader.

3.2.4 Disconnect

Definition	void Disconnect()
Description	Disconnect with device.

3.2.5 FactoryReset

Definition	ResultCode FactoryReset()
Description	Factory reset the device.

3.2.6 StartDetect

Definition	void StartDetect()
Description	If the device type is Unknown in TransportSerial, this method will start to detect the device type with the serial communication which defined in TransportSerial and trigger DetectDeviceEvent if the device is identified. DetectFinishedEvent will be triggered while the detection is finished.

3.2.7 StopDetect

Definition	void StopDetect()
Description	Stop the detect process in UHFReader.

3.3 Events

3.3.1 ActionStateChangedEvent

Definition	EventHandler<ActionStateEventArgs> ActionStateChangedEvent
Description	The event will be triggered while the action state of UHFReader is changed.

3.3.2 BatteryStateEvent

Definition	EventHandler<BatteryStateEventArgs> BatteryStateEvent
Description	The event used to receive the battery status.

3.3.3 ConnectStateChangedEvent

Definition	EventHandler<ConnectStateEventArgs> ConnectStateChangedEvent
Description	The event will be triggered while the connection state of the device is changed.

3.3.4 DetectDeviceEvent

Definition	EventHandler<DetectDeviceEventArgs> DetectDeviceEvent
Description	The event will be triggered while UHFReader identified the device.

3.3.5 DetectFinishedEvent

Definition	EventHandler<DetectDeviceEventArgs> DetectFinishedEvent
Description	The event will be triggered while the device detection is done in UHFReader.

3.3.6 KeyStateEvent

Definition	EventHandler<KeyEventArgs> KeyStateEvent
Description	The event will be triggered while the trigger key is pressed or released.

3.3.7 LogEvent

Definition	EventHandler<LogEventArgs> LogEvent
Description	The event used to receive the log from UHFReader.

3.3.8 TemperatureEvent

Definition	EventHandler<TemperatureEventArgs> TemperatureEvent
Description	The event used to receive the temperature status.

4. BaseUHF

BaseUHF provides the methods about UHF functions.

4.1 Properties

4.1.1 AlgorithmType

Definition	AlgorithmType AlgorithmType
Description	Get/Set the algorithm that the UHF module of the appliance uses to perform the Inventory.
Example	reader.BaseUHF.AlgorithmType = AlgorithmType.DynamicQ;

4.1.2 BLF

Definition	BLFType BLF
Description	Get/Set the status of the BLF that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.BLF = BLFType.BLF_256;

4.1.3 ContinuousMode

Definition	bool ContinuousMode
Description	Get/Set the UHF module will continue to perform Inventory operations.
Example	reader.BaseUHF.ContinuousMode = true;

4.1.4 Encoding

Definition	Encoding Encoding
Description	Get/Set the status of the Encoding that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.Encoding = Encoding.FM0;

4.1.5 FastMode

Definition	bool FastMode
Description	Get/Set the status of the fast mode that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.FastMode = true;

4.1.6 GlobalBand

Definition	GlobalBandType GlobalBand
Description	Get/Set the national orientation of the UHF module.
Example	reader.BaseUHF.GlobalBand = GlobalBandType.USA;

4.1.7 IdleTime

Definition	int IdleTime
Description	Get/Set the time at which radio waves are actually not output from the Inventory operation of the UHF module in the instrument. The unit is millisecond.
Example	reader.BaseUHF.IdleTime = 0;
Note	The range is 0 to 500.

4.1.8 InventoryTime

Definition	int InventoryTime
Description	Get/Set the time at which radio waves are actually output from the Inventory operation of the UHF module in the instrument. The unit is millisecond.
Example	reader.BaseUHF.InventoryTime = 200;
Note	The range is 40 to 500.

4.1.9 MaxQ

Definition	int MaxQ
Description	Get/Set the maximum Q value for algorithm of the UHF module. The value must be greader than the Min Q value.
Example	reader.BaseUHF.MaxQ = 15;
Note	The range is 0 to 15.

4.1.10MinQ

Definition	int MinQ
Description	Get/Set the minimum Q value for algorithm of the UHF module. The value msut be less than the Max Q value.
Example	reader.BaseUHF.MinQ = 0;
Note	The range is 0 to 15.

4.1.11ModuleProfile

Definition	int ModuleProfile
Description	Get/Set the status of the module's profile that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.ModuleProfile = 0;
Note	Not support on RP902.

4.1.12Power

Definition	int Power
Description	Get/Set the antenna output of the UHF module. The unit is dBm.
Example	reader.BaseUHF.Power = 20;
Note	The range of RP902 is 11 to 27.

4.1.13PowerMode

Definition	PowerMode PowerMode
Description	Get/Set the status of the power mode that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.PowerMode = PowerMode.Normal;
Note	Not support on RP902.

4.1.14PowerRange

Definition	PowerRange PowerRange
Description	Get the antenna output range of the UHF module. The unit is dBm.

4.1.15RFIDConfig

Definition	RFIDConfig RFIDConfig
Description	Get/Set the configs of UHF module.
Example	reader.BaseUHF.RFIDConfig = new RFIDConfig();

4.1.16SelectMask6cMaxSize

Definition	int SelectMask6cMaxSize
Description	Get the maximum size of select mask list status of the UHF module.

4.1.17Session

Definition	Session Session
Description	Get/Set the state of Session of the tag to be read by the Inventory operation.
Example	reader.BaseUHF.Session = Session.S0;

4.1.18StartQ

Definition	int StartQ
Description	Get/Set the start Q value for algorithm of the UHF module.
Example	reader.BaseUHF.StartQ = 4;
Note	The range is 0 to 15.

4.1.19 Target

Definition	Target Target
Description	Get/Set the status of the Target of the tag which will be read by the Inventory operation.
Example	reader.BaseUHF.Target = Target.A;

4.1.20 TARI

Definition	TARITYPE TARI
Description	Get/Set the status of the TARI that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.TARI = TARITYPE.T_6_25;

4.1.21 ToggleTarget

Definition	bool ToggleTarget
Description	Get/Set the status of the toggle target that the UHF module of the appliance uses to perform the Inventory operation.
Example	reader.BaseUHF.ToggleTarget = true;

4.2 Methods

4.2.1 GetSelectMask6cEnabled

Definition	bool GetSelectMask6cEnabled(int index);
Description	Returns whether to use the Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.2 SetSelectMask6cEnabled

Definition	void SetSelectMask6cEnabled(int index, bool enabled);
Description	Set whether or not to use the Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. enabled: A boolean specifying whether to apply the item.

4.2.3 GetSelectMask6cTarget

Definition	Mask6cTarget GetSelectMask6cTarget(int index);
Description	Returns the Flag of the tag to store the comparison result of Selection Mask item of UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.4 SetSelectMask6cTarget

Definition	void SetSelectMask6cTarget(int index, Mask6cTarget target);
Description	Set the Flag of the tag to save the comparison result of Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. target: The Mask6cTarget enumeration that specifies the Flag of the tag to store the result of the specified Selection Mask item comparison.

4.2.5 GetSelectMask6cAction

Definition	Mask6cAction GetSelectMask6cAction(int index);
Description	Returns the comparison behavior of the Selection Mask item of the RFID UHF module of the Device.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.6 SetSelectMask6cAction

Definition	void SetSelectMask6cAction(int index, Mask6cAction action);
Description	Set the comparison behavior of the Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. action: The Mask6cAction enumeration specifying how to compare the specified Selection Mask items.

4.2.7 GetSelectMask6cBank

Definition	BankType GetSelectMask6cBank(int index);
Description	Returns the memory bank to be compared with the specified Selection Mask item. Reserved bank is not used.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.8 SetSelectMask6cBank

Definition	void SetSelectMask6cBank(int index, BankType bank);
Description	Set the memory bank to be compared of the Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. bank: The BankType enumeration that specifies the memory bank to be compared with the specified Selection Mask item. Reserved bank is not used.

4.2.9 GetSelectMask6cOffset

Definition	int GetSelectMask6cOffset(int index);
Description	Returns the starting address at which the specified Selection Mask item will start the comparison. The unit specifying the start address of the Selection Mask is bit.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.10 SetSelectMask6cOffset

Definition	void SetSelectMask6cOffset(int index, int offset);
Description	Set the starting address at which the specified Selection Mask item will start the comparison. The unit specifying the start address of the Selection Mask is bit.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. offset: An integer specifying the starting address at which the specified Selection Mask item will start the comparison. The unit specifying the start address of the Selection Mask is bit.

4.2.11 GetSelectMask6cPattern

Definition	Mask6cPattern GetSelectMask6cPattern(int index);
Description	Returns an instance of Mask6cPattern specifying the comparison value and length of the specified Selection Mask item.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.12 SetSelectMask6cPattern

Definition	void SetSelectMask6cPattern(int index, Mask6cPattern pattern);
Description	Set the comparison value and length of the Selection Mask item of the UHF module.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. pattern: An instance of Mask6cPattern specifying the comparison value and length of the specified Selection Mask item.

4.2.13 SetSelectMask6c

Definition	void SetSelectMask6c(int index, SelectMask6cParam param);
Description	Set the Selection Mask item of the UHF module
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize. param: An instance of SelectMask6cParam that specifies the specified Selection Mask item.

4.2.14 GetSelectMask6c

Definition	SelectMask6cParam GetSelectMask6c(int index);
Description	Returns an instance of SelectMask6cParam that specifies the specified Selection Mask item.
Parameters	index: Integer specifying the radix of the item from 0 to GetSelectMask6cMaxSize.

4.2.15 Inventory6c

Definition	ResultCode Inventory6c();
Description	Device instructs to inventory the RFID UHF tag of ISO18000-6c Gen2 standard.

4.2.16 ReadMemory6c

Definition	ResultCode ReadMemory6c(BankType bankType, int offset, int length, string password);
Description	Device instructs to read specific memory of RFID UHF tag of ISO18000-6c Gen2 standard.
Parameters	bankType: The BankType enumeration that specifies the Bank for reading memory. offset: An integer specifying the starting address to start reading from the specified bank. Unit is WORD unit. length: An integer that specifies the length of data to read from the specified start address. Unit is WORD unit. password: If the tag is locked, it is a hex string specifying the Access Password set in the tag. Up to 8 characters (2 words) can be input.

4.2.17 WriteMemory6c

Definition	ResultCode WriteMemory6c(BankType bank, int offset, string data, string password);
Description	Device instructs to store data in a specific memory of RFID UHF tag of ISO18000-6c Gen2 standard.
Parameters	bank: The BankType enumeration that specifies the Bank for writing memory. offset: An integer specifying the starting address to start writing from the specified bank. Unit is WORD unit. data: An Hex type string that specifies the data to be stored in memory from the specified start address. Data must be specified in WORD units(4 characters). password :If the tag is locked, it is a hex string specifying the Access Password set in the tag. Up to 8 characters (2 words) can be input.

4.2.18 Lock6c

Definition	ResultCode Lock6c(Lock6cParam param, string password);
Description	Instructs the Device to lock or unlock the RFID UHF tags of the ISO18000-6c Gen2 standard.
Parameters	param: An instance of Lock6cParam that specifies the lock area of the tag memory. password: If the tag is locked, it is a hex string specifying the Access Password set in the tag.Up to 8 characters (2 words) can be input.

4.2.19 PermaLock6c

Definition	ResultCode PermaLock6c(PermaLock6cParam param, string password);
Description	Instructs the Device to permanent lock or unlock the RFID UHF tags of the ISO18000-6c Gen2 standard.
Parameters	param: An instance of PermaLock6cParam that specifies a permanent lock area for tag memory. password: If the tag is locked, it is a hex string specifying the Access Password set in the tag. Up to 8 characters (2 words) can be input.

4.2.20 Kill6c

Definition	ResultCode Kill6c(string password);
Description	Device instructs the RFID UHF tag of ISO18000-6c Gen2 specification to be no longer available.
Parameters	password: Hex type string that specifies the Kill Password set in the tag.

4.2.21 Stop

Definition	ResultCode Stop();
Description	Instructs the Device to stop the operation being performed.

4.3 Events

4.3.1 AccessResultEvent

Definition	EventHandler<AccessResultEventArgs> AccessResultEvent;
Description	The event used to receive the access result from the read/write/lock/kill operation.

4.3.2 TagReadEvent

Definition	EventHandler<TagReadEventArgs> TagReadEvent;
Description	The event used to receive the tag from the Inventory operation.

5. EventArgs Classes

5.1 DetectDeviceEventArgs

5.1.1 BTMACAddress

Definition	string BTMACAddress
Description	The Bluetooth MAC address of the detected device.

5.1.2 ConnectType

Definition	ConnectType ConnectType
Description	The communication type of the detected device.

5.1.3 DeviceType

Definition	DeviceType DeviceType
Description	The type of the detected device.

5.1.4 Name

Definition	string Name
Description	The name of the detected device.

5.1.5 Port

Definition	string Port
Description	The serial port of the detected device.

5.1.6 SerialNumber

Definition	string SerialNumber
Description	The serial number of the detected device.

5.2 LogEventArgs

5.2.1 DateTime

Definition	DateTime DateTime
Description	The created time of the log event.

5.2.2 LogLevel

Definition	LogLevel LogLevel
Description	The log level of the log event.

5.2.3 Tag

Definition	string Tag
Description	The tag information of the log event.

5.2.4 Log

Definition	string Log
Description	The log of the log event.

5.3 ActionStateEventArgs

5.3.1 ActionState

Definition	ActionState ActionState
Description	The current action state of the UHFReader.

5.3.2 Reader

Definition	BaseReader Reader
Description	The reader which to trigger the event.

5.4 BatteryStateEventArgs

5.4.1 Reader

Definition	BaseReader Reader
Description	The reader which to trigger the event.

5.4.2 Battery

Definition	int Battery
Description	The current status of battery.

5.5 KeyEventArgs

5.5.1 Reader

Definition	BaseReader Reader
Description	The reader which to trigger the event.

5.5.2 KeyState

Definition	KeyState KeyState
Description	The state of the key.

5.5.3 KeyType

Definition	KeyType KeyType
Description	The key type of the event.

5.6 ConnectStateEventArgs

5.6.1 Reader

Definition	BaseReader Reader
Description	The reader which to trigger the event.

5.6.2 ConnectState

Definition	ConnectState ConnectState
Description	The current state of connection.

5.7 TemperatureEventArgs

5.7.1 Reader

Definition	BaseReader Reader
Description	The reader which to trigger the event.

5.7.2 Temperature

Definition	double Temperature
Description	The current status of temperature.

5.8 TagReadEventArgs

5.8.1 Antenna

Definition	short Antenna
Description	The Antenna of the tag read.

5.8.2 EPC

Definition	string EPC
Description	The EPC of the tag.

5.8.3 Frequency

Definition	float Frequency
Description	The Frequency of the tag read.

5.8.4 PC

Definition	string PC
Description	The PC of the tag.

5.8.5 Q

Definition	short Q
Description	The Q value of the tag.

5.8.6 RawEPC

Definition	byte[] RawEPC
Description	The EPC of the tag.

5.8.7 RawPC

Definition	byte[] RawPC
Description	The PC of the tag.

5.8.8 RawTID

Definition	byte[] RawTID
Description	The TID of the tag.

5.8.9 RSSI

Definition	float RSSI
Description	The RSSI of the tag.

5.8.10 TID

Definition	string TID
Description	The TID of the tag.

5.9 AccessResultEventArgs

5.9.1 EPC

Definition	string EPC
Description	The EPC of the tag.

5.9.2 Data

Definition	string Data
Description	The data of the operation.

5.9.3 ActionState

Definition	ActionState ActionState
Description	The state of the operation.

5.9.4 BankType

Definition	BankType BankType
Description	The bank of the operation.

5.9.5 ResultCode

Definition	ResultCode ResultCode
Description	The result code of the operation.

6. FindDevice

6.1 Mode

Definition	FindDeviceMode Mode
Description	The mode of FindDevice function.

6.2 Timeout

Definition	int Timeout
Description	The timeout of FindDevice function. The unit is tenth of seconds.

7. PowerRange

7.1 Min

Definition	int Min
Description	The minimum of antenna output of the UHF module. The unit is dBm.

7.2 Max

Definition	int Max
Description	The maximum of antenna output of the UHF module. The unit is dBm.

8. SelectMask6cParam

8.1 IsUsed

Definition	bool IsUsed
Description	Indicating whether to use the condition of "Selection Mask".

8.2 Target

Definition	Mask6cTarget Target
Description	Indicating the flag of the tag to store the result of the "Selection Mask" condition.

8.3 Action

Definition	Mask6cAction Action
Description	Indicating how to save the results for the condition of "Selection Mask".

8.4 Bank

Definition	BankType Bank
Description	The BankType enumeration representing the memory bank of the tag to compare the pattern in "Selection Mask".

8.5 Offset

Definition	int Offset
Description	An integer representing the start address at which to begin comparing patterns in the "Selection Mask". The unit is bit.

8.6 Pattern

Definition	string Pattern
Description	A string representing the comparison pattern for "Selection Mask".

8.7 Length

Definition	int Length
Description	An integer representing the length of the pattern in "Selection Mask". The unit is bit.

9. RFIDConfig

9.1 ContinuousMode

Definition	bool ContinuousMode
Description	The UHF module will continue to perform Inventory operations.

9.2 Target

Definition	Target Target
Description	The status of the Target of the tag which will be read by the Inventory operation.

9.3 Session

Definition	Session Session
Description	The state of Session of the tag to be read by the Inventory operation.

9.4 Encoding

Definition	Encoding Encoding
Description	The status of the Encoding that the UHF module of the appliance uses to perform the Inventory operation.

9.5 TARIType

Definition	TARIType TARIType
Description	The status of the TARI that the UHF module of the appliance uses to perform the Inventory operation.

9.6 BLFType

Definition	BLFType BLFType
Description	The status of the BLF that the UHF module of the appliance uses to perform the Inventory operation.

9.7 Algorithm

Definition	AlgorithmType Algorithm
Description	The algorithm that the UHF module of the appliance uses to perform the Inventory.

9.8 FastMode

Definition	bool FastMode
Description	The status of the fast mode that the UHF module of the appliance uses to perform the Inventory operation.

9.9 ToggleTarget

Definition	bool ToggleTarget
Description	The status of the toggle target that the UHF module of the appliance uses to perform the Inventory operation.

9.10 ModuleProfile

Definition	int ModuleProfile
Description	The status of the module's profile that the UHF module of the appliance uses to perform the Inventory operation.
Note	Not support on RP902.

9.11 Power

Definition	int Power
Description	The antenna output of the UHF module. The unit is dBm.

9.12 InventoryTime

Definition	int InventoryTime
Description	The time at which radio waves are actually output from the Inventory operation of the UHF module in the instrument. The unit is millisecond.

9.13 IdleTime

Definition	int IdleTime
Description	The time at which radio waves are actually not output from the Inventory operation of the UHF module in the instrument. The unit is millisecond.

9.14 StartQ

Definition	int StartQ
Description	The start Q value for algorithm of the UHF module.

9.15 MaxQ

Definition	int MaxQ
Description	The maximum Q value for algorithm of the UHF module. The value must be greader than the Min Q value.
Note	The range is 0 to 15.

9.16 MinQ

Definition	int MinQ
Description	The minimum Q value for algorithm of the UHF module. The value must be less than the Max Q value.
Note	The range is 0 to 15.

10. PermaLock6cParam

10.1 KillPassword

Definition	LockState KillPassword
Description	TAG Access operation for 'Kill Password Area of Reserved Memory'.

10.2 AccessPassword

Definition	LockState AccessPassword
Description	TAG Access operation for 'Access Password Area of Reserved Memory'.

10.3 EPC

Definition	LockState EPC
Description	TAG Access action for 'EPC memory'.

10.4 TID

Definition	LockState TID
Description	TAG Access operation for 'TID memory'.

10.5 User

Definition	LockState User
Description	TAG Access action for 'User memory'.

11. Lock6cParam

11.1 KillPassword

Definition	LockState KillPassword
Description	TAG Access operation for 'Kill Password Area of Reserved Memory'.

11.2 AccessPassword

Definition	LockState AccessPassword
Description	TAG Access operation for 'Access Password Area of Reserved Memory'.

11.3 EPC

Definition	LockState EPC
Description	TAG Access action for 'EPC memory'.

11.4 TID

Definition	LockState TID
Description	TAG Access operation for 'TID memory'.

11.5 User

Definition	LockState User
Description	TAG Access action for 'User memory'.

12. Mask6cPattern

12.1 Pattern

Definition	string Pattern
Description	An Hex type string representing a mask pattern.

12.2 Length

Definition	int length
Description	An integer representing the length of the pattern. The length of the pattern is in bits.

13. DataRange

13.1 Offset

Definition	int Offset
Description	An integer specifying the starting address to start reading from the specified bank. Unit is WORD unit.

13.2 Length

Definition	int length
Description	An integer that specifies the length of data to read from the specified start address. Unit is WORD unit.

14. Enums

14.1 ActionState

```
public enum ActionState
{
    Stop = 0,
    Inventory6c = 0x0003,
    ReadMemory6c = 0x0005,
    WriteMemory6c = 0x0007,
    Lock = 0x0009,
    PermaLock = 0x000A,
    Kill = 0x000B,
    BlockErase = 0x000C,
    BlockWrite = 0x000D,
    Decoding = 0x10001,
    DefaultParameter = 0x20004,
    InventoryAndDecode = 0x30001
}
```

14.2 AlgorithmType

```
public enum AlgorithmType : byte
{
    FixedQ = 0,
    DynamicQ = 1
}
```

14.3 BeeperState

```
public enum BeeperState : byte
{
    Off = 0,
    Low = 1,
    Medium = 2,
    High = 3
}
```

14.4 BankType

```
public enum BankType : byte
{
    Reserved = 0,
    EPC = 1,
    TID = 2,
    User = 3
}
```

14.5 BLFType

```
public enum BLFType : byte
{
    /// <summary>
    /// Backscatter-link frequency 40
    /// </summary>
    BLF_40 = 0,
    /// <summary>
    /// Backscatter-link frequency 160
    /// </summary>
    BLF_160 = 6,
    /// <summary>
    /// Backscatter-link frequency 213
    /// </summary>
}
```

```
BLF_213 = 8,  
/// <summary>  
/// Backscatter-link frequency 256  
/// </summary>  
BLF_256 = 9,  
/// <summary>  
/// Backscatter-link frequency 320  
/// </summary>  
BLF_320 = 12,  
/// <summary>  
/// Backscatter-link frequency 640  
/// </summary>  
BLF_640 = 15  
}
```

14.6 BeepAndVibrateState

```
public enum BeepAndVibrateState : byte  
{  
    Off = 0,  
    On = 1  
}
```

14.7 ConnectType

```
public enum ConnectType : byte  
{  
    Unknown = 0,  
    Bluetooth = 1,  
    BluetoothLe = 2,  
    Serial = 5  
}
```

14.8 ConnectState

```
public enum ConnectState : byte
{
    Disconnected = 0,
    Listen = 1,
    Connecting = 2,
    Connected = 3
}
```

14.9 DeviceType

```
public enum DeviceType : byte
{
    Unknown = 0,
    RP902 = 1
}
```

14.10 DataFormat

```
public enum DataFormat : byte
{
    HEX = 0,
    ASCII = 1
}
```

14.11 DataInfo

```
public enum DataInfo : byte
{
    EPC = 0,
    User = 1,
    TID = 2
}
```

14.12 DataTerminator

```
public enum DataTerminator : byte
{
    None = 0,
    CR_LF = 1,
    Tab = 2
}
```

14.13 Encoding

```
public enum Encoding : byte
{
    FM0 = 0,
    M2 = 1,
    M4 = 2,
    M8 = 3
}
```

14.14 FindDeviceMode

```
public enum FindDeviceMode : byte
{
    Off = 0,
    Vibrate = 1,
    Beep = 2,
    VibrateBeep = 3
}
```

14.15 GlobalBandType

```
public enum GlobalBandType : byte
{
    Unknown = 0,
    Europe = 1,
```

```
    USA = 2,  
    China = 5,  
    Taiwan = 6,  
    Japan = 3,  
    Japan250mW = 7,  
    Australia = 8  
}
```

14.16 KeyType

```
public enum KeyType : byte  
{  
    Trigger = 0,  
    RightKey = 1,  
    LeftKey = 2  
}
```

14.17 KeyState

```
public enum KeyState : byte  
{  
    KeyUp = 0,  
    KeyDown = 1  
}
```

14.18 LockState

```
public enum LockState : byte  
{  
    NoChanged = 0,  
    Unlock = 1,  
    Lock = 2  
}
```

14.19 LogLevel

```
public enum LogLevel
{
    Trace = 0,
    Debug = 1,
    Info = 2,
    Warning = 3,
    Error = 4,
    Fatal = 5
}
```

14.20 Mask6cAction

```
public enum Mask6cAction : byte
{
    /// <summary>
    /// Matching is Inventoried A or Assert SL, Not-Matching is Inventoried B or
Deassert SL.
    /// </summary>
    AB = 0,
    /// <summary>
    /// Matching is Inventoried A or Assert SL, Not-Matching is Do Nothing.
    /// </summary>
    AN = 1,
    /// <summary>
    /// Matching is Do Nothing, Not-Matching is Inventoried B or Deassert SL.
    /// </summary>
    NB = 2,
    /// <summary>
    /// Matching is Invert Session or Negate SL, Not-Matching is Do Nothing.
    /// </summary>
    MN = 3,
    /// <summary>
    /// Matching is Inventoried B or Deassert SL, Not-Matching is Inventoried A or
```


Assert SL.

```

    /// </summary>
    BA = 4,
    /// <summary>
    /// Matching is Inventoried B or Deassert SL, Not-Matching is Do Nothing.
    /// </summary>
    BN = 5,
    /// <summary>
    /// Matching is Do Nothing, Not-Matching is Inventoried A or Assert SL.
    /// </summary>
    NA = 6,
    /// <summary>
    /// Matching is Do Nothing, Not-Matching is Invert Session or Negate SL.
    /// </summary>
    NM = 7
}

```

14.21 Mask6cTarget

```

public enum Mask6cTarget : byte
{
    S0 = 0,
    S1 = 1,
    S2 = 2,
    S3 = 3,
    SL = 4
}

```

14.22 OperatingMode

```

public enum OperatingMode : byte
{
    USBSP = 0,
    BTSP = 1,
    BTHID = 2,
    Buffer = 3,
}

```

```
        BLEHID = 4  
    }
```

14.23 PowerMode

```
public enum PowerMode  
{  
    /// <summary>  
    /// Normal process.  
    /// </summary>  
    Normal = 0,  
    /// <summary>  
    /// Refer to battery level to change duty cycle dynamically.  
    /// </summary>  
    Optimized = 1  
}
```

14.24 ResultCode

```
public enum ResultCode  
{  
    /// <summary>  
    /// No Error  
    /// </summary>  
    NoError = 0x0000,  
  
    /// <summary>  
    /// Other Error  
    /// </summary>  
    OtherError = 0x0001,  
  
    /// <summary>  
    /// Undefined  
    /// </summary>  
    Undefined = 0x0002,
```

/// <summary>
/// Memory Overrun
/// </summary>
MemoryOverrun = 0x0003,

/// <summary>
/// Memory Locked
/// </summary>
MemoryLocked = 0x0004,

/// <summary>
/// Insufficient Power
/// </summary>
InsufficientPower = 0x000B,

/// <summary>
/// Non-Specific Error
/// </summary>
NonSpecificError = 0x000F,

/// <summary>
/// Invalid Module Object
/// </summary>
InvalidModule = 0x4000,

/// <summary>
/// Not Supported
/// </summary>
NotSupported = 0x4001,

/// <summary>
/// In Operation
/// </summary>
InOperation = 0x4002,

/// <summary>
/// Out of Range
/// </summary>
OutOfRange = 0x4003,

/// <summary>
/// Not Supported Beep Function
/// </summary>
NotSupportedBeep = 0x4010,

/// <summary>
/// Not Supported Vibration Function
/// </summary>
NotSupportedVibrate = 0x4011,

/// <summary>
/// Not Supported Light Function
/// </summary>
NotSupportedLight = 0x4012,

/// <summary>
/// Not Supported Normal Mode
/// </summary>
NotSupportedNormalMode = 0x4013,

/// <summary>
/// Not Supported Barcode Mode
/// </summary>
NotSupportedBarcodeMode = 0x4014,

/// <summary>
/// Not Supported Barcode Mode
/// </summary>
NotSupportedKeyMode = 0x4015,

/// <summary>

/// Invalid Response

/// </summary>

InvalidResponse = 0x5000,

/// <summary>

/// Invalid Response Index

/// </summary>

InvalidResponseIndex = 0x5001,

/// <summary>

/// Invalid Parameter

/// </summary>

InvalidParameter = 0x5002,

/// <summary>

/// Invalid Response Data

/// </summary>

InvalidResponseData = 0x5003,

/// <summary>

/// Invalid Response Extend Parameter

/// </summary>

InvalidResponseExParam = 0x5004,

/// <summary>

/// Failed to process Parameter

/// </summary>

FailedParameter = 0x5005,

/// <summary>

/// Not Found Parameter

/// </summary>

NotFoundParameter = 0x5006,

/// <summary>

/// Send Failed

/// </summary>

SendFail = 0x6000,

/// <summary>

/// Receive Failed

/// </summary>

ReceiveFail = 0x6001,

/// <summary>

/// Send Timeout

/// </summary>

SendTimeout = 0x6002,

/// <summary>

/// Receive Timeout

/// </summary>

ReceiveTimeout = 0x6003,

/// <summary>

/// Invalid Send Packet

/// </summary>

InvalidSendPacket = 0x6004,

/// <summary>

/// Invalid Receive Packet

/// </summary>

InvalidReceivePacket = 0x6005,

/// <summary>

/// Connect Failed

/// </summary>

ConnectFail = 0x6006,

/// <summary>

/// Timeout

/// </summary>

Timeout = 0x7000,

/// <summary>

/// Unknown Error

/// </summary>

UnknownError = 0x7FFE,

/// <summary>

/// System Error

/// </summary>

SystemError = 0x7FFF,

/// <summary>

/// Handle Mismatch

/// </summary>

HandleMismatch = 0xF001,

/// <summary>

/// CRC error on tag response

/// </summary>

CRCErrror = 0xF002,

/// <summary>

/// No tag reply

/// </summary>

NoTagReply = 0xF003,

/// <summary>

/// Invalid password

/// </summary>

InvalidPassword = 0xF004,

/// <summary>

/// Zero kill password

/// </summary>

ZeroKillPassword = 0xF005,

/// <summary>

/// Tag lost

/// </summary>

TagLost = 0xF006,

/// <summary>
/// Command format error
/// </summary>
CommandFormatError = 0xF007,
/// <summary>
/// Read count invalid
/// </summary>
ReadCountInvalid = 0xF008,
/// <summary>
/// Out of retries
/// </summary>
OutOfRetries = 0xF009,

/// <summary>
/// Parameter error
/// </summary>
ParamError = 0xFFFB,

/// <summary>
/// Busy Device
/// </summary>
BusyDevice = 0xFFFC,

/// <summary>
/// Invalid command
/// </summary>
InvalidCommand = 0xFFFD,

/// <summary>
/// Low battery
/// </summary>
LowBattery = 0xFFFE,

/// <summary>
/// Operation failed
/// </summary>


```
        OperationFailed = 0xFFFF  
    }
```

14.25 ReadMode

```
public enum ReadMode : byte  
{  
    SingleRead = 0,  
    MultiRead = 1,  
    CounterRead = 2,  
    RapidRead = 3  
}
```

14.26 ReadOnceState

```
public enum ReadOnceState : byte  
{  
    Off = 0,  
    On = 1  
}
```

14.27 Session

```
public enum Session : byte  
{  
    S0 = 0,  
    S1 = 1,  
    S2 = 2,  
    S3 = 3  
}
```

14.28 SKUType

```
public enum SKUType : byte  
{  
    Unknown = 0,
```

```
EU = 1,  
US = 2,  
JP = 3,  
USJP = 4  
}
```

14.29 Target

```
public enum Target : byte  
{  
    A = 0,  
    B = 1  
}
```

14.30 TARIType

```
public enum TARIType : byte  
{  
    /// <summary>  
    /// 6.25  
    /// </summary>  
    T_6_25 = 0,  
    /// <summary>  
    /// 12.5  
    /// </summary>  
    T_12_50 = 1,  
    /// <summary>  
    /// 25  
    /// </summary>  
    T_25_00 = 2  
}
```

14.31 VibratorState

```
public enum VibratorState : byte
{
    Off = 0,
    On = 1
}
```