



Long Range Reader Modul UHF – M700
Module API for C#

iDTRONIC GmbH
Ludwig-Reichling-Straße 4
67059 Ludwigshafen
Germany/Deutschland

Phone: +49 621 6690094-0
Fax: +49 621 6690094-9
E-Mail: info@idtronic.de
Web: idtronic.de

Issue 1.3
– 19. May 2025 –

Subject to alteration without prior notice.
© Copyright iDTRONIC GmbH 2025
Printed in Germany

Contents

1	Introduction	5
2	Namespaces	5
2.1	MODULETECH.GEN2	5
2.1.1	Session Enumeration	5
2.1.2	MemBank Enumeration	5
2.1.3	Gen2LockAct Enumeration	5
2.1.4	Gen2LockAction Class	6
2.1.5	Gen2TagFilter Class	6
2.1.6	Target Enumeration	7
2.1.7	WriteMode Enumeration	7
2.2	MODULELIBRARY	7
2.2.1	ModuleException Class	7
2.2.2	OpFaidedException Class	7
2.2.3	HardwareAlertException Class	7
2.3	MODULETECH	7
2.3.1	Reader Class	8
2.3.2	ByteFormat Class	8
2.3.3	SimpleReadPlan Class	9
2.3.4	Region Enumeration	9
2.3.5	ReaderIPInfo Class	9
2.3.6	ReaderIPInfo_Ex.WifiSetting Class	10
2.3.7	ReaderIPInfo_Ex Class	10
2.3.8	TagData Class	10
2.3.9	TagFilter Interface	11
2.3.10	TagLockAction Class	11
2.3.11	TagProtocol Enumeration	11
2.3.12	TagReadData Class	11
2.3.13	EmbeddedCmdData Class	11
2.3.14	AntPower Class	12
2.3.15	ReaderConfiguration Class	12
2.3.16	BackReadOption Class	12
2.3.17	BackReadOption.FastReadTagMetaData Class	13
3	Reader Life Cycle	13
3.1	Create Method	13
3.2	Disconnect Method	14
4	Reader Parameters	14
4.1	ParamGet Method	14
4.2	ParamSet Method	14
4.3	Parameters	14
4.3.1	Read-only parameters	15
5	Synchronous Inventory	15
5.1	Read Method	16
6	Asynchronous Inventory	16
6.1	Start Inventory	17

6.2	Stop Inventory	18
7	Tag Access	18
7.1	Read Tag Memory	18
7.2	Write Tag Memory	19
7.3	Write Tag EPC.....	19
7.4	Lock Tag	19
7.5	Kill Tag	20
8	Peripheral Functions	20
8.1	Get GPI.....	20
8.2	Set GPO.....	21
9	Error Handling.....	21

1 Introduction

ModuleAPI for .net environment, to help customers to quickly develop a variety of applications.

2 Namespaces

2.1 MODULETECH.GEN2

There are some classes and enumeration about Gen2 protocol to be used when inventory, reading, writing, locking and destroying a tag. They are mainly defined in this namespace.

2.1.1 Session Enumeration

Define four Session of the GEN2 protocol.

Member	Description
Session0	Gen2 Session0
Session1	Gen2 Session1
Session2	Gen2 Session2
Session3	Gen2 Session3

2.1.2 MemBank Enumeration

Define four banks of GEN2 tag.

Member	Description
RESERVED	Reserve bank
EPC	EPC bank
TID	TID bank
USER	User bank

2.1.3 Gen2LockAct Enumeration

Define the memory area and type of tag locking operation.

Member	Description
ACCESS_LOCK	Temporarily lock access password
ACCESS_PERMALOCK	Permanently lock access password
ACCESS_UNLOCK	Unlock access password
EPC_LOCK	Temporarily lock EPC bank
EPC_PERMALOCK	Permanently lock EPC bank
EPC_UNLOCK	Unlock EPC bank
KILL_LOCK	Temporarily kill the lock password
KILL_PERMALOCK	Permanently kill the lock password
KILL_UNLOCK	Unlock kill password
TID_LOCK	Temporarily lock TID bank
TID_PERMALOCK	Permanently lock TID bank
TID_UNLOCK	Unlock TID bank
USER_LOCK	Temporarily lock USER bank
USER_PERMALOCK	Permanently lock USER bank
USER_UNLOCK	Unlock USER bank

2.1.4 Gen2LockAction Class

The action of locking a tag.

```
1. public Gen2LockAction(ModuleTech.Gen2.Gen2LockAct[] acts)
```

Parameter	Description
Acts	Store the locking actions

Example

If you want to temporarily lock EPC, unlock USER and permanently lock access password the code would be as following:

```
1. Gen2LockAct[] acts = new Gen2LockAct[3];
2. acts[0] = Gen2LockAct.EPC_LOCK;
3. acts[1] = Gen2LockAct.TID_UNLOCK;
4. acts[2] = Gen2LockAct.ACCESS_PERMALOCK;
5. Gen2LockAction lockact = new Gen2LockAction(acts);
```

2.1.5 Gen2TagFilter Class

Users can add a filter option when executing inventory, reading, writing, locking and kill tag operations so that only the tags which match the filter criteria would be operated on.

```
1. public Gen2TagFilter(int filterlen, byte[] filterdata, MemBank filterbank, int filteraddr, bool isInvert)
```

Parameter	Description
Filterlen	The length of data to be compared, in bits.
Filterdata	Data to compare, when filterlen is not the integral multiple of 8, the length of filterdata is (filterlen-1)/8 + 1
Filterbank	The bank to filter.
Filteraddr	The starting address of filtering, in bits
isInvert	Rule of filter, false means matching the filter criteria, true means not matching the filter criteria

Example

There are several tags in the field of an antenna. Among them there is only one tag whose EPC ID start with 1101. Now if we want to read the TID of this tag, we can use the parameter Gen2TagFilter to ensure that the reading operation is only on the right tag. The code is as following:

```
1. int bitcnt = 0;
2. string binarystr = "1101";
3. byte[] filterbytes = new byte[(binarystr.Length-1)/8+1];
4.
5. for (int c = 0; c < filterbytes.Length; ++c)
6.     filterbytes[c] = 0;
7.
8. foreach (Char ch in binarystr) {
9.     if (ch == '1')
10.         filterbytes[bitcnt / 8] |= (byte)(0x01 << (7 - bitcnt % 8));
11.     bitcnt++;
12. }
13. Gen2TagFilter filter = new Gen2TagFilter(binarystr.Length, filterbytes, ModuleTech.Gen2.MemBank.EPC,
14.    32, false);
```

2.1.6 Target Enumeration

Define the target of the GEN2 protocol.

Member	Description
A	Search for tags in State A
B	Search for tags in State B
AB	Search for tags in State A, then switch to B
BA	Search for tags in State B, then switch to A

2.1.7 WriteMode Enumeration

Define the write mode of a tag write operation.

Member	Description
WORD_ONLY	Use the Write command of Gen2 air interface for tag write operation
BLOCK_ONLY	Use the BlockWrite command of Gen2 air interface for tag write operation
BLOCK_FALLBACK	Temporarily not supported

2.2 MODULELIBRARY

The reader may have an abnormal situation during the working process, and the physical communication link between the reader and the external system where the problem occurs most. No matter what happens, the exception needs to be detected in time. Tag operations such as read, write, lock and destroy may also fail. These situations are thrown by an instance of the exception class, prompting the upper system.

2.2.1 ModuleException Class

The base class of other exception classes.

Member	Description
ErrCode	Error code

2.2.2 OpFaidedException Class

Non-fatal error, this only means that the operation failed.

2.2.3 HardwareAlertException Class

When the reader detects any condition that may cause damage to hardware, this exception would be thrown. The actions that possibly cause damage to hardware are the following:

- Transmit power through a port without antenna connection.
- High environmental temperature.
- Antennas stay too close to metal plate when the reader is reading tags. If the transmit power is very high, it would be highly possible that a lot of the power is reflected into the reader through the antennas causing damage.

Therefore, when detecting these errors, users should stop the reader from working and check whether there is any kind of maloperation as listed above.

2.3 MODULETECH

Defines some base classes associated with tag operations and the Reader class, the Reader class contains various methods of tag operation and methods related to configuring reader parameters.

2.3.1 Reader Class

This is the most important class, encapsulates all the functions for tag operations and manages the life cycle of the reader.

Main Methods

Method name	Description
Create	Create an instance of the Reader class
Disconnect	Free the resource of the instance of the reader class
ParamGet	Get parameters of the reader
ParamSet	Set parameters of the reader
Read	Synchronous inventory operation
StartReading	Start asynchronous inventory operation
StopReading	Stop asynchronous inventory operation
ReadTagMemWords	Read the data block of the tag's one storage area
WriteTagMemWords	Write data block to the tag's one storage area
WriteTag	Write the EPC code of a tag
LockTag	Lock the tag's storage areas
KillTag	Kill a tag
GPIGet	Get GPI state
GPOSet	Set GPO state

2.3.2 ByteFormat Class

Convert among hexadecimal string, ushort array and byte array.

FromHex

Convert hexadecimal string to byte array.

```
1. public static byte[] FromHex(string hex)
```

Parameter	Description
Hex	The hexadecimal string to convert

Example

```
1. byte[] readdata = ByteFormat.FromHex("1234abcd");
```

ToHex

Convert ushort array to hexadecimal string.

```
1. public static string ToHex(ushort[] words)
```

Parameter	Description
words	The ushort array to convert

Example

```
1. string readdata = ByteFormat.ToHex(new ushort[] {1,2,4});
```

Convert bytes array to hexadecimal string.

```
1. public static string ToHex(byte[] bytes)
```

Parameter	Description
bytes	The bytes array to convert

Example

```
1. string readdata = ByteFormat.ToHex(new byte[] {1,2,4});
```

2.3.3 SimpleReadPlan Class

Specify the antennas and tag protocol for inventory. `ModuleTech.ReadPlan` is the base class which would not be used in practice.

```
1. public SimpleReadPlan(int[] antennas)
```

Parameter	Description
antennas	The working antennas when inventory

Example

Use No.1, No.2 and No.4 antenna.

```
1. int[] ants = new int[]{1, 2, 4};
2. SimpleReadPlan plan = new SimpleReadPlan(ants);
```

2.3.4 Region Enumeration

The working frequency regulatory of the reader, it is set to `ModuleTech.Region.NA` by default.

Member	Description
UNSPEC	Unspecific area
CN	Canada
EU	Europe version 1 (LBT)
EU2	Europe version 2
EU3	Europe version 3 (no LBT)
IN	India
KR	Korea
JP	Japan
NA	North America
PRC	China (920-925)
PRC2	China (840-845)
OPEN	Full frequency (860-960)

2.3.5 ReaderIPInfo Class

The IP information of the reader, using `ReaderIPInfo`. Create create `ReaderIPInfo`.

Attribute	Description
GATEWAY	Read only, the gateway of the reader.
IP	Read only, the IP address of the reader.
SUBNET	Read only, the subnet mask of the reader.

```
1. public static ModuleTech.ReaderIPInfo Create(string ip, string subnet, string gateway)
```

Parameter	Description
-----------	-------------

ip	IP address
subnet	Subnet mask
gateway	Gateway

2.3.6 ReaderIPInfo_Ex.WifiSetting Class

The wifi settings.

Attribute	Description
Auth	Read only, WiFi authentication mode
SSID	Read only, WiFi SSID
KEY	Read only, WiFi password

```
1. WifiSetting(AuthMode auth, string ssid, KeyType ktype, string key)
```

Parameter	Description
auth	WiFi authentication mode: AuthMode_Open, AuthMode_Open_Wep, AuthMode_Shared_Wep, AuthMode_Wpa_Psk, AuthMode_Wpa2_Psk
ssid	SSID of WiFi setting
ktype	WiFi password type: KeyType_HEX means hexadecimal character, KeyType_ASC2 means ASC2 character
key	Password

2.3.7 ReaderIPInfo_Ex Class

The reader IP information including the wifi parameters.

Attribute	Description
IPInfo	Basic IP information
NType	Read only, net type
Wifi	Read only, WiFi setting parameter

```
1. ReaderIPInfo_Ex(ReaderIPInfo ip, NetType type, WifiSetting wifi)
```

Parameter	Description
ip	IP address, subnet mask, gateway
type	Net type: NetType_Ethernet, NetType_Wifi
wifi	WiFi setting parameter

2.3.8 TagData Class

Used for function WriteTag.

```
1. public TagData(byte[] epc)
```

Parameter	Description
epc	EPC data

```
1. public TagData(string epc)
```

Parameter	Description
epc	EPC data represented by hexadecimal string

2.3.9 TagFilter Interface

Used for filtering tags during tag operations.

2.3.10 TagLockAction Class

The base class of `ModuleTech.Gen2.Gen2LockAction`.

2.3.11 TagProtocol Enumeration

Air interface protocol

Member	Description
EPC0_IMPINJ	EPC0 of IMPINJ
EPC0_MATRICS	EPC0 of MATRICS
EPC1	EPC1 protocol
GEN2	GEN2 protocol
ISO18000_6B	ISO 18000-6B protocol
NONE	None
UCODE	UCODE protocol
IPX64	IPX64 protocol
IPX256	IPX256 protocol

2.3.12 TagReadData Class

The result of inventory.

Attribute	Description
Antenna	Read only, the antenna that read the tag
EPC	Read only, EPC data
EPCString	Read only, the hexadecimal form of EPC
ReadCount	Read only, the number of times read
Tag	Read only, the tagdata form of EPC
Time	Read only, the time when the tag was read
EdbData	Read only, if a read operation was embedded in inventory, then this attribute would contain the read data of another bank. If no read operation was embedded or the read operation failed, the attribute would be null.
Rssi	Read only, Received Signal Strength Indicator

2.3.13 EmbededCmdData Class

In the inventory operation, users can additionally read data of another bank at the same time. This class is used to specify the details of the read operation.

Attribute	Description
Bank	Read only, indicates the bank to be read
StartAddr	Read only, the starting address in the bank [in blocks]
ByteCnt	Read only, the number of bytes to read

```
1. public EmbededCmdData(Membank bank_, UInt32 addr, byte datacnt)
```

Parameter	Description
-----------	-------------

bank_	The bank to read during inventory
addr	Read only, the starting address in the bank [in blocks]
datacnt	Read only, the number of bytes to read

2.3.14 AntPower Class

Transmit power setting of a single antenna.

Attribute	Description
AntId	Read and write, the id of the antenna
ReadPower	Read and write
WritePower	Read and write

```
1. public AntPower(byte aid, UInt16 rpwr, UInt16 wpwr)
```

Parameter	Description
aid	ID of antenna, starting from 1
rpwr	Read power of the antenna [centi-dBm]
wpwr	Write power of the antenna [centi-dBm]

2.3.15 ReaderConfiguration Class

The setting to be used when permanently saving the parameters of the reader.

Attribute	Description
ReaderConfCode	Read and write

```
1. public ReaderConfiguration(SaveConfCode sccode)
```

Parameter	Description
sccode	SaveConf_Save indicates permanently saving configurations; SaveConf_Erase indicates deleting the configuration permanently saved, the reader would then use the default parameter setting.

2.3.16 BackReadOption Class

This parameter specifies all the details of the asynchronous inventory.

Attribute	Description
ReadDuration	Inventory cycle in milliseconds when using common asynchronous inventory mode. When using high-speed asynchronous inventory mode, the attribute is meaningless.
ReadInterval	The idle interval in milliseconds between two inventory cycles when using common asynchronous inventory mode. When using high-speed asynchronous inventory mode, the attribute is meaningless.
IsFastRead	Using high speed asynchronous inventory mode if true, otherwise using common asynchronous inventory mode.
FastReadDutyRation	The upper 4 bits must be 0, the lower 4 bits are valid and the lower 4 bits are used to indicate the proportion of time spent resting during the reader's inventory. If the FastReadDutyRation is 0-9, the proportion is FastReadDutyRation * 5%. If the FastReadDutyRation is 10-15 the proportion is 50%.

FRTMetadata	Specifies which metadata information the tag carries when using the high-speed asynchronous inventory mode. When using common asynchronous inventory mode the attribute is meaningless.
-------------	---

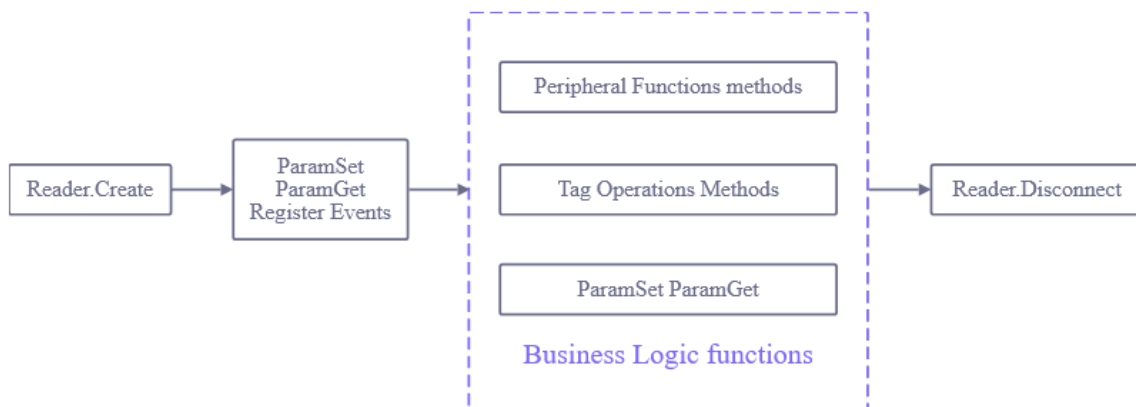
2.3.17 BackReadOption.FastReadTagMetaData Class

Specifies which metadata information the tag carries when using the high-speed asynchronous inventory mode.

Attribute	Description
IsReadCnt	The read count of the tag
IsRssi	Received Signal Strength Indication of the tag
IsAntennaID	The antenna ID on which the tag is inventoried
IsFrequency	The frequency on which the tag is inventoried
IsTimestamp	The time stamp when the tag is inventoried
IsRFU	The reserved field
IsEmdData	The bank data of the tag, during inventory the reader can read tag data of another bank depending on the EmbeddedCmdOfInventory parameter

3 Reader Life Cycle

Call the `Reader.Create` method to initialize the reader. Afterwards you may call the `Reader.ParamSet` or `Reader.ParamGet` methods to configure the reader. Now you can call tag operation and peripheral function methods. Call `Reader.Disconnect` if you no longer use the reader.



3.1 Create Method

Create an instance of the Reader class and initialize the reader.

```
1. public static Reader Create(string uriString, Region region, int antsnum)
```

Parameter	Description
uriString	IP address or serial port number of the reader (e.g. com1)
region	Working frequency band area of the reader
antsnum	The number of physical ports of the reader

Example:

```

1. try {
2.     Reader rdr = Reader.Create("192.168.0.250", Region.NA, 4);
3. } catch (Exception ex) {
4.     MessageBox.Show("Create failed: " + ex.ToString());
5. }
  
```

3.2 Disconnect Method

Disconnect from the reader and release related resources.

```
1. public abstract void Disconnect()
```

Example:

```
1. rdr.Disconnect();
```

4 Reader Parameters

The `Reader.ParamGet` and `Reader.ParamSet` methods are to get and set the reader's parameters. Every parameter of the reader is represented by a key value which is a string. Some of the parameters are read-only while others are read- and writeable. A parameter of the reader will consistently until it is set to another value.

4.1 ParamGet Method

This method gets a parameter of the reader and can be called any time during the lifetime of the reader instance.

```
1. public object ParamGet(string key)
```

Parameter	Description
key	The name of the parameter

Return value: A value of object type. Should be cast to the right parameter type.

Example: Get the firmware version of the reader.

```
1. string firmware = (string) rdr.ParamGet("SoftwareVersion");
```

4.2 ParamSet Method

This method sets a parameter of the reader and can be called any time during the lifetime of the reader instance. Once a parameter was set, it will be in effect until the reader is disconnected or the parameter is modified again.

```
1. public void ParamSet(string key, object value)
```

Parameter	Description
key	The name of the parameter
value	The value of the parameter

Example: Set the Gen2Session as Session0.

```
1. rdr.ParamSet("Gen2Session", Session.Session0);
```

4.3 Parameters

Key string	Parameter meaning	Value type	Description
IPAddress	Ip address	ReaderIPInfo	The reader will restart after the IP is set successfully; hence it is necessary to reconnect to the reader.
Gen2Session	Gen2 Session	ModuleTech.Gen2.Session	For small amounts but fast-moving tags use Session0; for large amounts but slow-moving tags use Session1.
Gen2Qvalue	Gen2 Q value	int	Value ranges from -1 to 15.

ReadPlan	The antenna setting during inventory operation	ReadPlan	Do not execute any tag operations on antenna ports without antenna connection.
Region	Frequency regulatory	Region	Uses different frequencies depending on the set region.
TagopAntenna	Used antenna for tag operations (except inventory)	int	Only one antenna can be specified before invoking tag operations.
TagopProtocol	Protocol to use for tag operations (except inventory)	TagProtocol	The protocol must be specified before invocation of tag operations.
AccessPassword	The access password	uint	Specify the password for access if needed.
Singulation	Tag filter	TagFilter	Selection criteria for tags during inventory operation. May be set to null.
OpTimeout	Operation timeout (except for inventory)	ushort	The maximum duration an operation blocks (in milliseconds).
EmbeddedCmdOfInventory	Embedded operation during inventory operation	EmbeddedCmdData	Set as null to disable.
BackReadOption	Detailed settings for asynchronous inventory	BackReadOption	Specifies details for asynchronous inventory. Must be set before starting inventory, else an exception will be thrown.
AntPowerConf	Configuration of all antennas tx powers	AntPower[]	The default antenna power after startup is two third of the maximum power.
FrequencyHopTable	Frequency hopping table of the reader	uint[]	The frequencies must be within the range of region regulatory.
Gen2Target	Gen2 target	ModuleTech.Gen2.Target	The parameter of the Gen2 air interface.
Gen2WriteMode	Gen2 writing mode	ModuleTech.Gen2.WriteMode	Only supports WORD_ONLY and BLOCK_ONLY modes.
IsTagDataUniqueByAnt	Filter criterium	bool	Use antenna port as unique filter
IsTagDataUniqueByEmddata	Filter criterium	bool	Use tag data as unique filter
IsTagDataRecordHighestRssi	Filter criterium	bool	Whether to record the highest RSSI

4.3.1 Read-only parameters

Key string	Parameter meaning	Value type	Description
BootloaderVersion	Bootloader version	string	
ConnectedAntennas	The connected antennas	int[]	Not all kinds of antennas can be detected.
HardwareVersion	Hardware version	string	
SoftwareVersion	Firmware version	string	
RfPowerMax	Max power	int	The maximum power in centi-dBm.
RfPowerMin	Min power	int	The minimum power in centi-dBm.

5 Synchronous Inventory

Synchronous inventory is implemented using the `Reader.Read` method, which blocks until the end of the inventory. It is suitable for short inventory operations with a few tags.

5.1 Read Method

Before calling this method, the `ReadPlan` parameter must be set, otherwise an exception will be thrown.

```
1. public abstract ModuleTech.TagReadData[] Read(int milliseconds)
```

Parameter	Description
milliseconds	The execution duration in milliseconds. The method will block for at least this time duration.

Returns the inventoried tags, an instance of the `TagReadData` class contains all the information.

Example: Use antennas 1, 3 and 4, only read tags with an EPC code starting with 0x1234 and an embedded reading instruction which reads data of two blocks from the second block of the TID area.

```
1. EmbeddedCmdData ecd = new EmbeddedCmdData(MemBank.TID, 2, 4);
2. rdr.ParamSet("EmbeddedCmdOfInventory", ecd);
3. byte[] fdata = ByteFormat.FromHex("1234");
4. Gen2TagFilter filter = new Gen2TagFilter(fdata.Length*8, fdata, ModuleTech.Gen2.MemBank.EPC, 32, false);
5. rdr.ParamSet("Singulation", filter);
6. int[] ants = new int[]{1, 3, 4};
7. SimpleReadPlan searchPlan = new SimpleReadPlan(ants);
8. rdr.ParamSet("ReadPlan", searchPlan);
9. TagReadData[] tags = rdr.Read(500);
```

6 Asynchronous Inventory

Asynchronous inventory can be implemented in two ways. The first option, the common asynchronous inventory mode, is to use a thread to call the synchronous inventory method (Read method). The other option, the true asynchronous inventory mode, sets the reader in a continuous inventory state. This way, the inventory performance of the reader is optimal. Hence this mode should be used for applications with higher inventory performance requirements. It is also called high-speed asynchronous inventory mode.

Attention: Not all device types support asynchronous high-speed inventory mode, only those readers built with R2000 RF chips can support this mode.

During the high-speed asynchronous inventory, the `TagsRead` event will be triggered if tags are inventoried to pass the tag data to the user program. If the reader has an abnormal condition, the `ReadException` event will be triggered to pass the exception details to the user program. When the asynchronous inventory is configured to start or stop with a GPI trigger, the `GpiTrigger` and `GpiTriggerBoundary` events are triggered when the GPI condition is met.

Method	Description
Reader.StartReading	Start asynchronous inventory
Reader.StopReading	Stop asynchronous inventory

Event	Description
Reader.TagsRead	Fires when tags are inventoried
Reader.ReadException	Fires when an exception occurs
Reader.GpiTrigger	Fires when the GPI trigger condition is met, and the asynchronous inventory is configured to start by <code>GpiTrigger</code> .
Reader.GpiTriggerBoundary	Fires when starting or stopping asynchronous inventory if it is configured to start by <code>GpiTrigger</code> .

6.1 Start Inventory

To start the asynchronous inventory, call the `StartReading` method. Before starting inventory, it must be ensured that the following parameters have been set correctly and the events related to asynchronous inventory have been registered, else an exception will be thrown.

- The `ReadPlan` parameter of `SimpleReadPlan` must be set to specify the air interface protocol and the working antenna.
- The `BackReadOption` parameter must be set to specify the detailed configuration of the inventory operation.
- The `TagsRead` event must be registered, otherwise the tags inventoried cannot be passed to the user program.
- The `ReadException` event must be registered, otherwise the reader exception state cannot be passed to the user program.

If the reader is already performing asynchronous inventory, calling the `StartReading` method again will throw an exception. In case of doubt about the current state of the reader, the `StopReading` method can be called before starting the inventory.

StartReading Method

Starts the asynchronous inventory.

```
1. public void StartReading()
```

TagsRead Event

Fires when tags are inventoried to pass the tag data to the user program by the parameter of type `TagsReadEventArgs`.

ReadException Event

Fires when an exception occurs to pass the exception details to the user program by the parameter of type `ReadExceptionEventArgs`.

GpiTrigger Event

If asynchronous inventory is configured to start by `GpiTrigger`, it fires an event when the GPI trigger condition is met to pass the current GPI states to the user program by the parameter of type `GpiTriggerEventArgs`.

GpiTriggerBoundary Event

If asynchronous inventory is configured to start by `GpiTrigger`, it fires an event when starting or stopping asynchronous inventory to pass the causes of starting or stopping inventory to the user program by the parameter of type `GpiTriggerBoundaryEventArgs`.

TagsReadEventArgs Class

Parameter of the `TagsRead` event to pass the inventoried tags' data.

Attribute	Description
Tags	The inventoried tags' data.

ReadExceptionEventArgs Class

Parameter of the `ReadException` event to pass the exception details.

Attribute	Description
ReaderException	The reader exception.

GpiTriggerEventArgs Class

Parameter of the `GpiTrigger` event to pass the GPI status and trigger condition number.

Attribute	Description
GpiStates	GPI status when triggering start or stop of asynchronous inventory.
TriggerId	Condition number that triggers the start or stop of the asynchronous inventory.

GpiTriggerBoundaryEventArgs Class

Parameter of the GpiTriggerBoundary event to pass the reason for starting or stopping asynchronous inventory.

Attribute	Description
Reason	Cause of starting or stopping inventory.
BoundaryType	Starting or stopping.

6.2 Stop Inventory

Call the StopReading method to stop the asynchronous inventory.

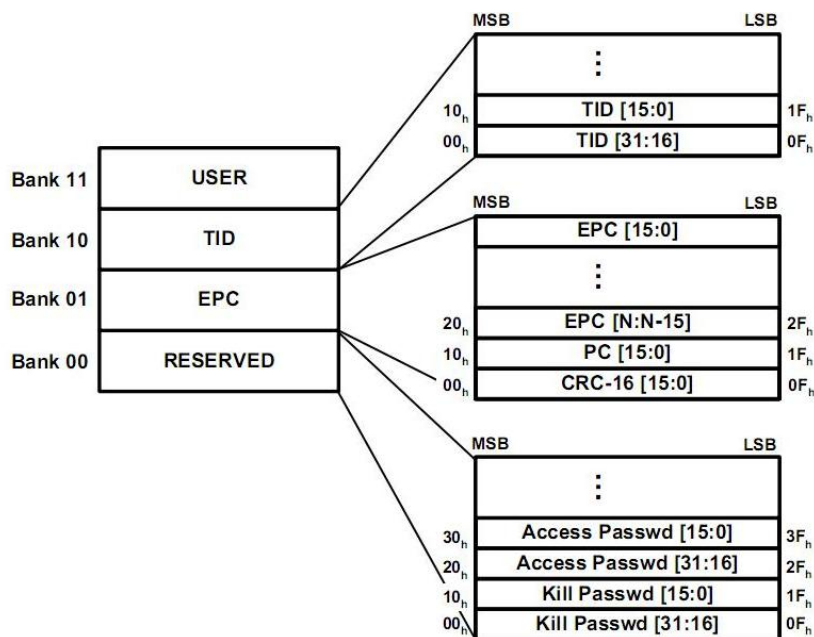
```
1. public void StopReading()
```

7 Tag Access

Tag access operations include read, write, write EPC, lock and kill operations and are applied to a single tag. In case of multiple tags in the antenna field, it is undetermined on which tag the operation is performed. To act on a specific tag, pass the TagFilter parameter to the tag access method.

The operations can be performed on one antenna at a time only. Therefore, the TagopAntenna parameter must be set before the tag access operation.

A Gen2 tag is divided into four banks which are bank 0, bank 1, bank 2 and bank 3. Bank 0 is also called the reserved bank which contains the access password and the kill password where each password has 32 bits. Bank 1 is called EPC bank, which contains a CRC field (16 bits), PC field (Protocol Control, 16 bits) and EPC field (Electronic Product Code, maximum length is 496 bits and the common length is 96 bits). Bank 2 also known as TID bank, which contains tag- and vendor-specific data (for example, a tag serial number). Bank 3 is called user bank which allows user-specific data storage. Lots of tags do not have bank 3.



7.1 Read Tag Memory

Read a memory area of a bank.

```
1. public abstract ushort[] ReadTagMemWords(ModuleTech.TagFilter target, ModuleTech.Gen2.MemBank bank,
2. int wordAddress, int wordcount)
```

Parameter	Description
-----------	-------------

target	Tag filter, set to null if not needed.
bank	The bank to read.
wordAddress	The starting address in the bank, in blocks.
wordCount	The number of blocks to read.

Example: Read data of two blocks from the second block of the TID bank.

```
1. ushort[] readdata = rdr.ReadTagMemWords(null, 2, Membank.TID, 2);
```

7.2 Write Tag Memory

Write data into a memory area of a bank.

```
1. public abstract void WriteTagMemWords(ModuleTech.TagFilter target, ModuleTech.Gen2.MemBank bank,
2.                                     int address, ushort[] data)
```

Parameter	Description
target	Tag filter, set to null if not needed.
bank	The bank to write.
address	The starting address in the bank, in blocks.
data	The data to write.

Example: Write two blocks of data "0x44445555" from the third block into the USER bank.

```
1. string datastr = "0x44445555";
2. ushort[] writedata = new ushort[2];
3. for (int a = 0; a < writedata.Length; ++a)
4.     writedata[a] = ushort.Parse(datastr.Substring(a*4, 4),
                                   System.Globalization.NumberStyles.AllowHexSpecifier);
5. rdr.WriteTagMemWords(null, Membank.USER, 3, writedata);
```

7.3 Write Tag EPC

Write the EPC of a tag. One could also write the EPC through the WriteTagMemWords method. However the WriteTag method also changes the EPC length field of the PC field.

```
1. public abstract void WriteTag(ModuleTech.TagFilter target, ModuleTech.TagData epc)
```

Parameter	Description
target	Tag filter, set to null if not needed.
epc	The EPC code.

Example: Write EPC 0x0086000400004100000000280 into a tag.

```
1. TagData epccode = new TagData("0086000400004100000000280");
2. rdr.WriteTag(null, epccode);
```

7.4 Lock Tag

Lock the tag's storage areas.

```
1. public abstract void LockTag(ModuleTech.TagFilter target, ModuleTech.TagLockAction action)
```

Parameter	Description
target	Tag filter, set to null if not needed.
action	What areas to lock or unlock.

Example: For a tag with TID starting with 0x4321123 and access password 0x33332222, temporarily lock the EPC bank, unlock the user bank and permanently lock the access password.

```
1. byte[] fdata = ByteFormat.FromHex("4321123");
2. Gen2TagFilter filter = new Gen2TagFilter(fdata.Length*8, fdata, ModuleTech.Gen2.MemBank.TID, 0, false);
3. string accesspwdstr = "33332222";
4. uint passwd = uint.Parse(accesspwdstr, System.Globalization.NumberStyles.AllowHexSpecifier);
5. rdr.ParamSet("AccessPassword", passwd);
6. Gen2LockAct[] acts = new Gen2LockAct[3];
7. acts[0] = Gen2LockAct.EPC_LOCK;
8. acts[1] = Gen2LockAct.TID_UNLOCK;
9. acts[2] = Gen2LockAct.ACCESS_PERMALOCK;
10. Gen2LockAction lockact = new Gen2LockAction(acts);
11. rdr.LockTag(filter, lockact);
```

7.5 Kill Tag

To kill a tag.

```
1. public abstract void KillTag(ModuleTech.TagFilter target, uint password)
```

Parameter	Description
target	Tag filter, set to null if not needed.
password	Kill password.

Example: Kill a tag with EPC starting with 0x1234 and kill password 0x11112222.

```
1. string kpasswdstr = "11112222";
2. uint kpasswd = uint.Parse(kpasswdstr, System.Globalization.NumberStyles.AllowHexSpecifier);
3. byte[] fdata = ByteFormat.FromHex("1234");
4. Gen2TagFilter filter = new Gen2TagFilter(fdata.Length*8, fdata, ModuleTech.Gen2.MemBank.EPC, 32, false);
5. rdr.KillTag(filter, kpasswd);
```

8 Peripheral Functions

Peripheral functions currently only support getting GPI status and setting GPO status.

8.1 Get GPI

There are two methods to get the GPI state, one gets only one GPI pin's state at a time, the other gets all the reader's GPI pins' states at a time.

Get the state of one GPI pin.

```
1. public abstract bool GPIGet(int pin)
```

Parameter	Description
pin	GPI-ID, one-based numbering.

Example: Get the state of GPI pin 1.

```
1. bool gpi1 = rdr.GPIGet(1);
```

Get the states of all GPI pins of the reader.

```
1. public GPIState[] GPIGet()
```

Example:

```
1. GPIState[] gstates = rdr.GPIGet();
```

8.2 Set GPO

Set the state of a GPO pin.

```
1. public abstract void GP0Set(int pin, bool state)
```

Parameter	Description
pin	GPO-ID, one-based numbering.
state	The state to set.

Example: Set the state of GPO pin 1 to true.

```
1. rdr.GP0Set(1, true);
```

9 Error Handling

All methods of the Reader class may throw an exception. Hence calling the methods should include code that catches the exception. If the `Exception System.IO.IOException` is caught, the instance of the Reader class should be re-created after calling the `Disconnect` method.

If the `Exception OpFaidedException` is caught upon calling the methods `Read`, `KillTag`, `LockTag`, `ReadTagMemWords`, `ParamGet` or `ParamSet`, the operation failed. Execution of tag operations may still be continued.

The `HardwareAlertException` indicates improper operations which can damage the hardware. These operations include:

- Transmit power through antenna ports without an antenna connection.
- The use of unqualified antennas.
- High ambient temperature.
- High return loss caused by a metal plate in front of the antennas.

It is best to turn off the reader and check the working conditions and environment for these errors.

All the methods, except for the `Create` method of the Reader class, of the current version of the development kit are not thread-safe for the same reader instance. For different reader instances there is no such limit.