AsReaderP3xU SDK

C# SDK Reference Guide V1.1

Asterisk Inc.

Revision History

Version	Description	Date
V1.0	Initial version.	2024/5/9
V1.1	1. Add function CheckTagStatus. 2. Add enumeration TagStatus.	2024/11/20

Contents

Introduction

This manual provides the following information for the developers developing Windows desktop applications using the SDK:

- How the development environment is built
- Description of various SDK library functions

Development tool:

- Visual Studio 2012 or later

1 Building the Development Environment

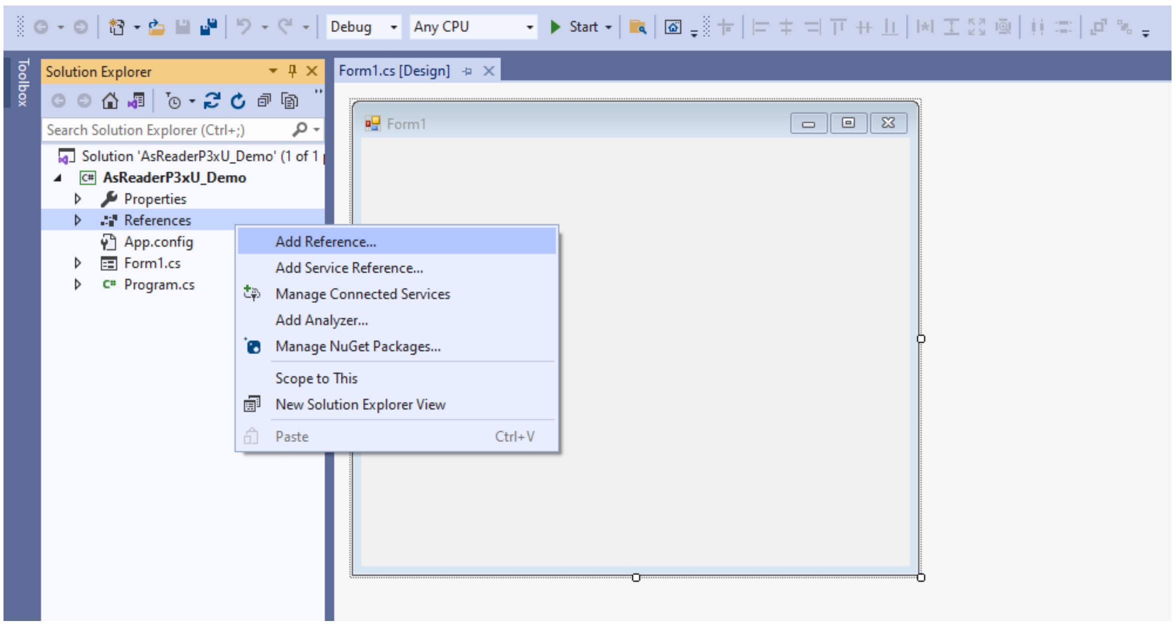
1.1 Importing the SDK

1. Create a Windows desktop application

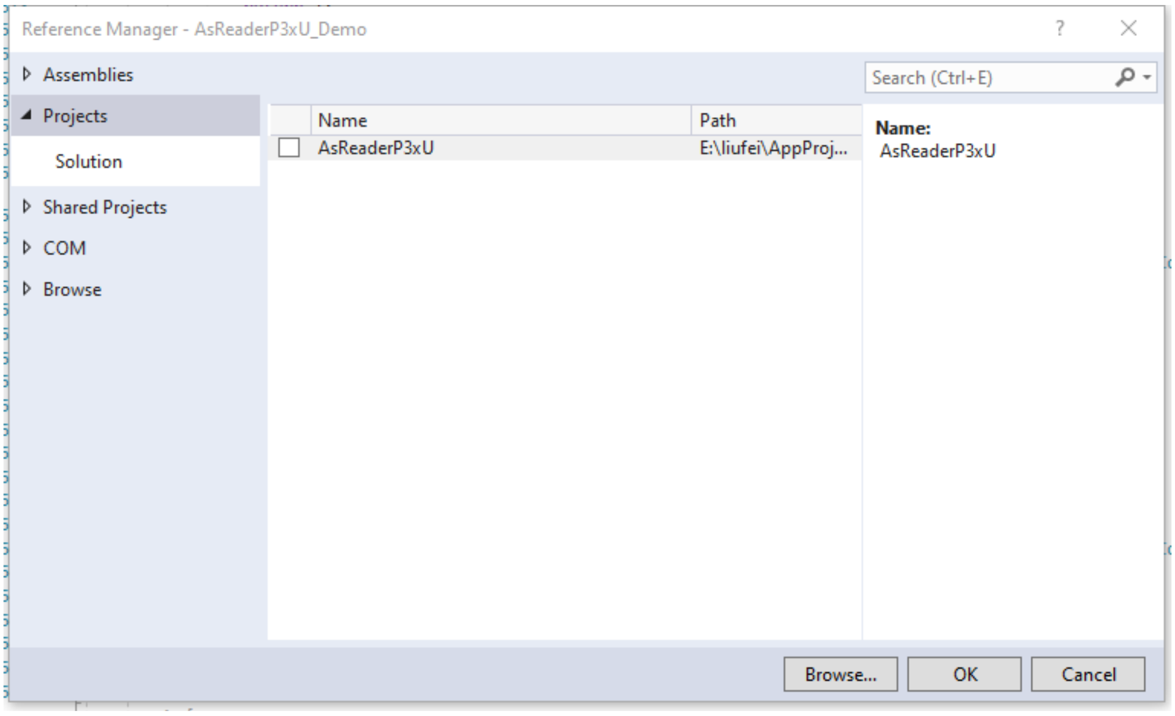
   Copy the AsReaderP3xU.dll file into the project folder.
   For developing or debugging, copy it to the bin/Debug folder in the program's current directory.
   For releasing, copy to the bin/Release folder in the program's current directory.

2. Add Reference to AsReaderP3xU.dll

   Right click “References” and choose “Add Reference”.



3. Click the “Browse” button.

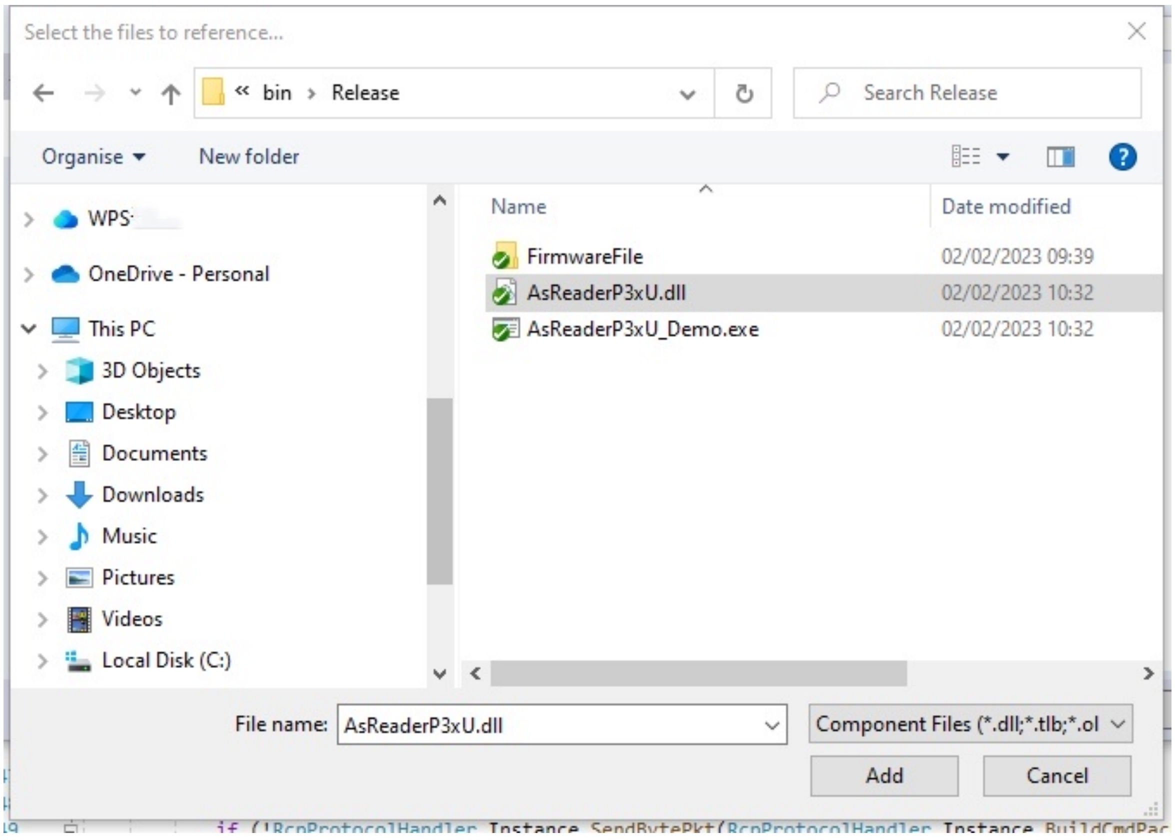


Reference Manager - AsReaderP3xU_Demo

	Name	Path
☐	AsReaderP3xU	E:\liufei\AppProj...

Name:
AsReaderP3xU

4. Select the AsReaderP3xU.dll file under the project path in the “Select the Files Reference” dialog box and click the “Add” button.



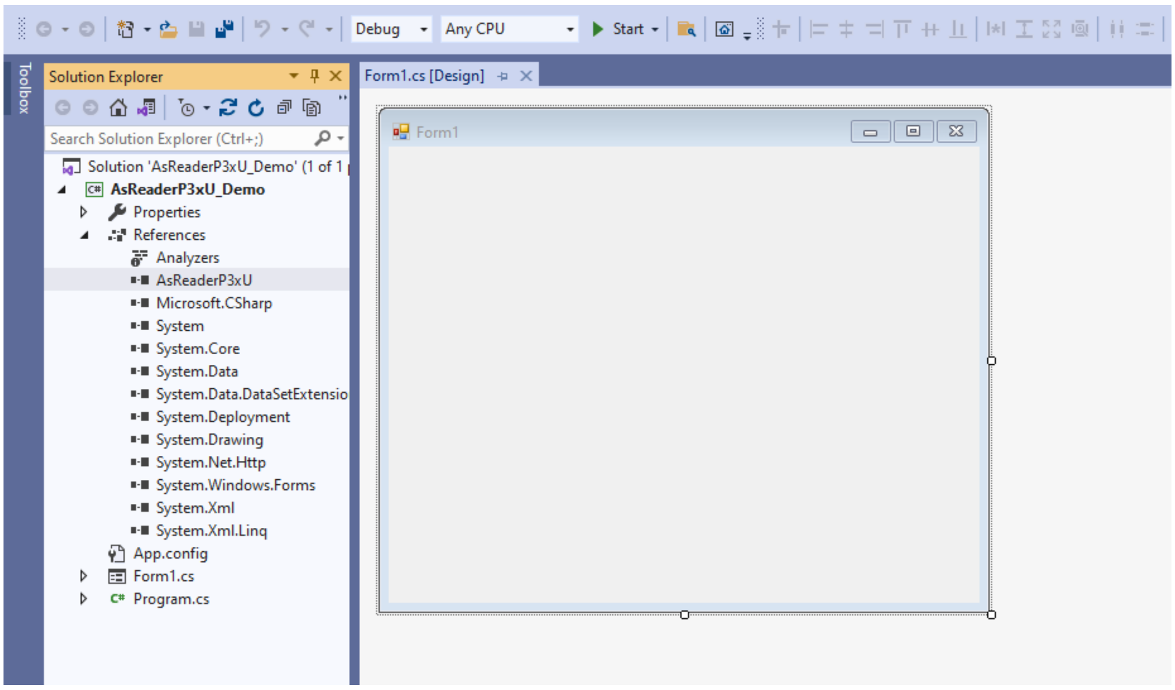
Select the files to reference...

Name	Date modified
FirmwareFile	02/02/2023 09:39
AsReaderP3xU.dll	02/02/2023 10:32
AsReaderP3xU_Demo.exe	02/02/2023 10:32

File name: AsReaderP3xU.dll

Component Files (*.dll;*.tlb;*.ol

5. The reference is displayed in the reference list after being added successfully.



Solution 'AsReaderP3xU_Demo' (1 of 1
AsReaderP3xU_Demo
Properties
References
Analyzers
AsReaderP3xU
Microsoft.CSharp
System
System.Core
System.Data
System.Data.DataSetExtensio
System.Deployment
System.Drawing
System.Net.Http
System.Windows.Forms
System.Xml
System.Xml.Linq
App.config
Form1.cs
Program.cs

1.2 Referring the Namespace

```
using AsReaderP3xU;
```

1.3 Using the SDK

The following are examples of calling functions in the AsReader class such as connecting or disconnecting from AsReader and other operations.

1. Get the AsReader object

Call the constructor of the “AsReader” class to get the AsReader object.

```
AsReader asreader = new AsReader();
```

2. Run the ConnectWithVCP function (See 2.1.1) to connect to the AsReader device.

Call the function "asreader.ConnectWithVCP" with setting the COM port as the parameter.
If the connection is successful, the function returns 0.

```
UInt32 ret = asreader.ConnectWithVCP("COM1");
if(ret == 0){
    //Processing after connection.
}else{
    //Processing if the connection fails.
}
```

2 AsReader Class

2.1 Functions

The AsReader class provides functions for RFID tag inventory, read, write, lock, and other operations.

2.1.1 ConnectWithVCP

Function	UInt32 ConnectWithVCP(string comPort)		
Parameter	**IN/OUT**	**Type**	**Description**
comPort	IN	string	COM port of the AsReader device
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Used when connecting via USB. Sets the COM port to connect to the AsReader device.			
Code example: ConnectWithVCP(“COM1”);			

2.1.2 DisConnect

Function	UInt32 DisConnect()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Disconnects the AsReader device from the host device.			
Code example: DisConnect();			

2.1.3 StartInventory

Function	UInt32 StartInventory (bool rssiEn, byte mtnu, byte mtime, UInt16 rc, bool an1)		
Parameter	**IN/OUT**	**Type**	**Description**
rssiEn	IN	bool	true: Display RSSI false: Do not display RSSI.
mtnu	IN	byte	Maximum number of tags read
mtime	IN	byte	Maximum time to inventory (sec)
rc	IN	UInt16	The maximum number of inventorying cycles
an1	IN	bool	true: enable the antenna false: disable the antenna
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:

The AsReader device starts an inventory of RFID tags with the conditions for stopping the inventory (number of inventory cycles, number of read tags, duration of inventory). Whether to display the RSSI can also be set. If multiple conditions are set, the AsReader device stops the inventory when one of the conditions is met.

Code example:

```
//Number of inventory cycles: 10
//Number of read tags: 100
//Duration of inventory: 60s
//rssi: Do not display
StartInventory(false,100,60,10,true);
```

2.1.4 StopInventory

Function	UInt32 StopInventory()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:

Stops the inventory of RFID tags.

Code example:

```
StopInventory();
```

2.1.5 SetSelectMask

Function	UInt32 SetSelectMask(Types.MemBankType memBank, Types.TargetType target, Types.ActionType action, UInt32 startAddressWord, byte[] selectMask)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	Types.MemBankType	The memory bank of the tag on which the Selection Mask operation is performed See 2.2.1.6
target	IN	Types.TargetType	The Session value of the tag on which the Selection Mask operation is performed See 2.2.1.4
action	IN	Types.ActionType	The action after the tag is marked See 2.2.1.5
startAddressWord	IN	UInt32	Offset of the tag memory bank (starting address) Unit: word
selectMask	IN	byte[]	Mask value Unit: word
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:

Sets the Mask parameters.

The function can only inventory, read, write, and lock to the selected tag.

Code example:

```
//Memory bank: EPC
//Session: SESSION_S0
//Action: ACTION_ASLINVA_DSLINVB
//Offset: 2
byte[] selectMask= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};
SetSelectMask(MEM_EPC,SESSION_S0,ACTION_ASLINVA_DSLINVB,0x02,selectMask);
```

2.1.6 GetSelectMask

Function	UInt32 GetSelectMask(ref Types.MemBankType memBank, ref Types.TargetType target, ref Types.ActionType action, ref UInt32 startAddressWord, ref string selectMask)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	OUT	Types.MemBankType	The memory bank of the tag on which the Selection Mask operation is performed See 2.2.1.6.
target	OUT	Types.TargetType	The Session value of the tag on which the Selection Mask operation is performed See 2.2.1.4.
action	OUT	Types.ActionType	The action after the tag is marked See 2.2.1.5.
startAddressWord	OUT	UInt32	Offset of the tag memory bank (starting address) Unit: word
selectMask	OUT	string	Mask value Unit: word
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the current values of the Mask parameters.			
Code example: GetSelectMask(ref membank, ref target, ref action, ref startAddressWord, ref epc_string);			

2.1.7 SetSelectionEnable

Function	UInt32 SetSelectionEnable(Types.SelectionEnable selection_enable)		
Parameter	**IN/OUT**	**Type**	**Description**
selection_enable	IN	Types.SelectionEnable	Whether to use the current mask See 2.2.1.18
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets whether to use the current mask.			

Code example:
//Enable the mask that is currently selected.
SetSelectionEnable(ENABLE);			

2.1.8 GetSelectionEnable

Function	UInt32 GetSelectionEnable(ref Types.SelectionEnable selection_enable)		
Parameter	**IN/OUT**	**Type**	**Description**
selection_enable	OUT	Types.SelectionEnable	Whether to use the current mask (See 2.2.1.18)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets whether to use the current mask.			
Code example: GetSelectionEnable(ref selection_enable);			

2.1.9 WriteMemory

Function	UInt32 WriteMemory(Types.MemBankType memBank, uint startAddressWord, uint accessPassword, byte[] writeData, byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	Types.MemBankType	The memory bank of the tag to which the writing operation is performed See 2.2.1.6
startAddressWord	IN	uint	The offset of the memory bank to be written Unit: word
accessPassword	IN	uint	The access password for the target tag (if no password is set, the value is 0.)
writeData	IN	byte[]	The data to be written to the tag
epcData	IN	byte[]	The EPC value of the target tag
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: 1. Specify a tag using epcData and write data to the target memory bank of the tag. 2. The maximum length of data written to the tag's memory bank is 32 Words / 64 Bytes.			

Code example:
//Target memory bank: EPC
//Offset: 2
//Access password: 0x12345678
//Data to be written:
byte[] writedata = {0x12,0x34};
//EPC value of the target tag:
byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};
WriteMemory(MEM_EPC,0x02,0x12345678,writeData,epcData);			

2.1.10 ReadMemory

Function	UInt32 ReadMemory(Types.MemBankType memBank, uint startAddressWord, uint lengthWord, uint accessPassword, byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	Types.MemBankType	The memory bank of the tag to which the Reading operation is performed See 2.2.1.6
startAddressWord	IN	uint	The offset of the memory bank to be read Unit: Word
lengthWord	IN	uint	The length of the memory bank to be read Unit: Word
accessPassword	IN	uint	The access password for the target tag (if no password is set, the value is 0.)
epcData	IN	byte[]	The EPC value of the target tag.
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:

1. Specify a tag using epcData and read data of the target memory bank of the tag.
2. The maximum length of the memory bank's data to be read is 32 Words / 64 Bytes.

Code example:

```
//Target memory bank: EPC bank
//Offset: 2
//Length: 2
//Access password: 0x12345678
//EPC value of the target tag:
byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};
ReadMemory(MEM_EPC,0x02,0x02,0x12345678,epcData);
```

2.1.11 Kill

Function	UInt32 Kill(uint killPassword, byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
KillPassword	IN	uint	The Kill password of the target tag
epcData	IN	byte[]	The EPC value of the target tag
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:

1. Specify a tag using epcData and kill it.
2. Before killing the tag, write the Kill password into the RESERVED field (two Words starting from the offset 00).

Note: Once the Kill operation is performed, it is irreversible.

Code example:

```
//Kill password: 0x12345678
//EPC value of the target tag:
byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};
Kill(0x12345678,epcData);
```

2.1.12 LockMemory

Function	UInt32 LockMemory(TagMask tagMask, TagAction tagAction, uint accessPassword, byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
tagMask	IN	TagMask	The value for which the Mask of Lock operation is set (See Appendix II)
tagAction	IN	TagAction	The value for which the Action of Lock operation is set (See Appendix II)
accessPassword	IN	uint	The access password of the target tag (if no password is set, the value is 0.)
epcData	IN	byte[]	The EPC value of the target tag
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

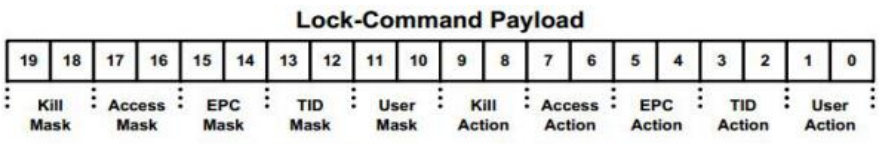
Description:

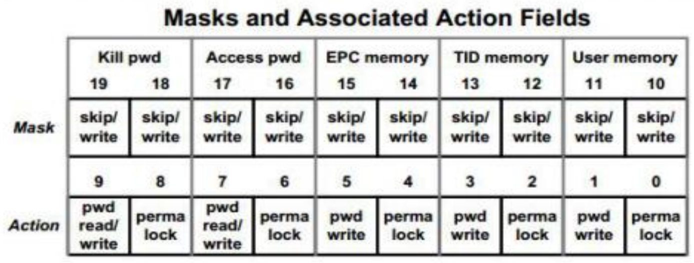
Lock, PermaLock, Unlock, or PermaUnlock the memory bank of a tag.

It is necessary to set an Access password before locking a tag.

The Lock command is required to Lock a tag. The Lock command contains a 20-bit payload. The first ten places are Masks and the last ten places are Actions. Only the Action whose Mask bit is 1 is valid. Each data area has 2bit, 00 to 11, corresponding to unlock, permanently unlock, lock, and permanently lock.

The meaning of each bit of Mask and Action is as follows:





Masks and Associated Action Fields

	Kill pwd		Access pwd		EPC memory		TID memory		User memory	
	19	18	17	16	15	14	13	12	11	10
Mask	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write
	9	8	7	6	5	4	3	2	1	0
Action	pwd read/ write	perma lock	pwd read/ write	perma lock	pwd write	perma lock	pwd write	perma lock	pwd write	perma lock

Code example: //Lock the EPC area of the tag.

//The access password of the target tag: 12345678

//The EPC value of the target tag:

byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};

//The Mask of Lock operation:

TagMask tagMask={ false, false, false, false, true, true, false, false, false, false}

//The Action of Lock operation:

TagAction tagAction={ false, false, false, false, true, false, false, false, false, false}

LockMemory(tagMask, tagAction,0x12345678,epcData);

2.1.13 SetRegion

Function	UInt32 SetRegion(Types.RegionType region)		
Parameter	**IN/OUT**	**Type**	**Description**
region	IN	Types.RegionType	Current country or region (See 2.2.1.2)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the current country or region.			

Code example: //Sets the current country or region to REGION_JAPAN.
SetRegion(REGION_JAPAN);			

2.1.14 GetRegion

Function	UInt32 GetRegion(ref Types.RegionType region)		
Parameter	**IN/OUT**	**Type**	**Description**
region	OUT	Types.RegionType	The country or region (See 2.2.1.2)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the current country or region.			
Code example: GetRegion(ref region);			

2.1.15 GetTxPower

Function	UInt32 GetTxPower(ref uint power, ref uint minPower, ref uint maxPower)		
Parameter	**IN/OUT**	**Type**	**Description**
power	OUT	uint	The value of power [Japan: 13-23, Others: 13-27]
minPower	OUT	uint	Parameter value of the minimum power [13]
maxPower	OUT	uint	Parameter value of the maximum power [Japan: 23, Others: 27]
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the power of the AsReader device.			
Code example: GetTxPower(ref power,ref minPower,ref maxPower);			

2.1.16 SetTxPower

Function	UInt32 SetTxPower(uint txPower)		
Parameter	**IN/OUT**	**Type**	**Description**
txPower	IN	uint	The value of Power [Japan: 13-23, Others: 13-27]
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the power of the AsReader device.			

Code example: //Sets the power to 22dBm
SetTxPower(22);			

2.1.17 SetSession

Function	UInt32 SetSession(Types.SessionType session)		
Parameter	**IN/OUT**	**Type**	**Description**
session	IN	Types.SessionType	The session value of the inventory (See 2.2.1.3)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the session value of the inventory.			

Code example: //Sets the session to S0
SetSession(SESSION_S0);			

2.1.18 GetSession

Function	UInt32 GetSession(ref Types.SessionType session)		
Parameter	**IN/OUT**	**Type**	**Description**
session	OUT	Types.SessionType	The session value of the inventory (See 2.2.1.3)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the session value of the inventory.			
Code example: GetSession(ref session);			

2.1.19 SetChannel

Function	UInt32 SetChannel(uint channel)		
Parameter	**IN/OUT**	**Type**	**Description**
channel	IN	uint	The channel value when reading tags (See 2.2.1.7) The range of channel numbers depends on the Region.
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the channel value when reading tags.			

Code example: //Set the channel value to CHANNEL_24
SetChannel(CHANNEL_24);			

2.1.20 GetChannel

Function	UInt32 GetChannel(ref uint channel)		
Parameter	**IN/OUT**	**Type**	**Description**
channel	OUT	uint	The channel value when reading tags (See 2.2.1.7) The range of channel numbers depends on the Region.
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the channel value when reading tags.			
Code example: GetChannel(ref channel);			

2.1.21 SetBasicTarget

Function	UInt32 SetBasicTarget(Types.TargetABType target)		
Parameter	**IN/OUT**	**Type**	**Description**
target	IN	Types.TargetABType	The target value when reading tags (See 2.2.1.9)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the target value when reading tags.			

Code example: //Set the RFID session tag bit TARGET_A.
SetBasicTarget(TARGET_A);			

2.1.22 GetBasicTarget

Function	UInt32 GetBasicTarget(ref Types.TargetABType target)		
Parameter	**IN/OUT**	**Type**	**Description**
target	OUT	Types.TargetABType	The target value when reading tags (See 2.2.1.9)
Return Value	OUT	UInt32	Succeed: 0 Failed: 1
Description: Gets the target value when reading tags.			
Code example: GetBasicTarget(ref target);			

2.1.23 SetQuery

Function	UInt32 SetQuery(Types.DRType dr, Types.MType m, Types.TRextType trext, Types.SelType sel, Types.SessionType session, Types.TargetABType target, Types.QType q)		
Parameter	**IN/OUT**	**Type**	**Description**
dr	IN	Types.DRType	The data rate of the reader (See 2.2.1.10)
m	IN	Types.MType	Encoding type (See 2.2.1.11)
trext	IN	Types.TRextType	Whether the preamble contains a pilot signal (See 2.2.1.12)
sel	IN	Types.SelType	Tag bit of the tag (See 2.2.1.13)
session	IN	Types.SessionType	Session (See 2.2.1.3)
target	IN	Types.TargetABType	Session flag bit (See 2.2.1.9)
q	IN	Types.QType	Q sets the number of slots in the round. Slot counts=2^q (See 2.2.1.14)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets all query parameter values in the inventory operation.			

Code example: //Set the dived ratio= DR_8, encoding mode= M1, no pilot tone.
Sel= SEL_ALL, Session= SESSION_S0, Session flag= TARGET_A, Slots=4.
SetQuery(DR_8,M1,NO_Pilot_Tone,SEL_ALL,SESSION_S0,TARGET_A,Q4);			

2.1.24 GetQuery

Function	UInt32 GetQuery(ref Types.DRType dr, ref Types.MType m, ref Types.TRextType trext, ref Types.SelType sel, ref Types.SessionType session, ref Types.TargetABType target, ref Types.QType q)		
Parameter	**IN/OUT**	**Type**	**Description**
dr	OUT	Types.DRType	The data rate from the tag to the reader (See 2.2.1.10)
m	OUT	Types.MType	Encoding type (See 2.2.1.11)
trext	OUT	Types.TRextType	Whether a tag prepends a preamble with a pilot tone. (See 2.2.1.12)
sel	OUT	Types.SelType	Tags to respond to the Query. (See 2.2.1.13)
session	OUT	Types.SessionType	Session for the inventory rounds. (See 2.2.1.3)
target	OUT	Types.TargetABType	Whether tags whose inventoried flag is A or B participate in the inventory rounds. Tags may change their inventoried flag from A to B (or vice versa) as a result of being singulated. (See 2.2.1.9)
q	OUT	Types.QType	The number of slots in the round. Slot counts=2^q. (See 2.2.1.14)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets all parameter values in the tag query operation.			
Code example: GetQuery(ref dr,ref m,ref trext,ref sel,ref session,ref target,ref q);			

2.1.25 SetAntiCollisionMode

Function	UInt32 SetAntiCollisionMode(Types.AntiCollisionMode anticollisionmode, Types.QType startq, Types.QType minq, Types.QType maxq)		
Parameter	**IN/OUT**	**Type**	**Description**
anticollisionmode	IN	Types.AntiCollisionMode	Anti-Collision Mode (See 2.2.1.15)
startq	IN	Types.QType	The start value of Q (See 2.2.1.14)
minq	IN	Types.QType	The minimum value of Q (See 2.2.1.14)
maxq	IN	Types.QType	The maximum value of Q (See 2.2.1.14)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets which anti-collision algorithm to use.			

Code example:
//Anti-Collision Mode: FixedQ
//The start value of Q: Q1
//The minimum value of Q: Q2
//The maximum value of Q: Q8
SetAntiCollisionMode(FixedQ,Q1,Q2,Q8);			

2.1.26 GetAntiCollisionMode

Function	UInt32 GetAntiCollisionMode(ref Types.AntiCollisionMode anticollisionmode, ref Types.QType startq, ref Types.QType minq, ref Types.QType maxq)		
Parameter	**IN/OUT**	**Type**	**Description**
anticollisionmode	OUT	Types.AntiCollisionMode	Anti-Collision Mode (See 2.2.1.15)
startq	OUT	Types.QType	The start value of Q (See 2.2.1.14)
minq	OUT	Types.QType	The minimum value of Q (See 2.2.1.14)
maxq	OUT	Types.QType	The maximum value of Q (See 2.2.1.14)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets which anti-collision algorithm is used.			
Code example: GetAntiCollisionMode(ref antiCollisionMode,ref startq,ref minq,ref maxq);			

2.1.27 SetFH_LBT

Function	UInt32 SetFH_LBT(uint readTime, uint idelTime, uint cst, uint rfl, Types.FHType fh, Types.LBTType lbt, Types.CWType cw)		
Parameter	**IN/OUT**	**Type**	**Description**
readTime	IN	uint	Query time (10~40000, 1=1ms)
idelTime	IN	uint	idle time (ms)
cst	IN	uint	Carrier detection time (1 = 1ms)
rfl	IN	uint	Target RF power level (-dBm x 10)
fh	IN	Types.FHType	Enable (0x01 or above) / Disable (0x00) (See 2.2.1.19)
lbt	IN	Types.LBTType	Enable (0x01 or above) / Disable (0x00) (See 2.2.1.20)
cw	IN	Types.CWType	Enable (0x01) / Disable (0x00) (See 2.2.1.21)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:
Sets FH and LBT parameters.

Code example:

```
//Inventory time: 400
//Idle time: 100
//Carrier detection time: 10
//Target RF power level: -740
//fh: DISABLE
//lbt: DISABLE
//cw: DISABLE
SetFH_LBT(400,100,10,-740,DISABLE,DISABLE,DISABLE)
```

2.1.28 GetFH_LBT

Function	UInt32 GetFH_LBT(ref uint readTime, ref uint idelTime, ref uint cst, ref uint rfl, ref Types.FHType fh, ref Types.LBTType lbt, ref Types.CWType cw)		
Parameter	**IN/OUT**	**Type**	**Description**
readTime	OUT	uint	Query time (10~40000, 1=1ms)
idelTime	OUT	uint	idle time (ms)
cst	OUT	uint	Carrier detection time (1 = 1ms)
rfl	OUT	uint	Target RF power level (-dBm x 10)
fh	OUT	Types.FHType	Enable (0x01 or above) / Disable (0x00) (See 2.2.1.19)
lbt	OUT	Types.LBTType	Enable (0x01 or above) / Disable (0x00) (See 2.2.1.20)
cw	OUT	Types.CWType	Enable (0x01) / Disable (0x00) (See 2.2.1.21)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets FH and LBT parameters.			
Code example: GetFH_LBT(ref readTime,ref idelTime,ref cst,ref rfl,ref fh,ref lbt,ref cw)			

2.1.29 SetFrequencyAutomatic

Function	UInt32 SetFrequencyAutomatic(bool status)		
Parameter	**IN/OUT**	**Type**	**Description**
status	IN	bool	true: Automatic frequency false: Not automatic frequency
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets whether to set the frequency automatically.			
Code example: SetFrequencyAutomatic(true);			

2.1.30 GetFrequencyAutomatic

Function	UInt32 GetFrequencyAutomatic(bool status)		
Parameter	**IN/OUT**	**Type**	**Description**
status	OUT	bool	true: Automatic frequency false: Not automatic frequency
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets whether to set the frequency automatically.			
Code example: GetFrequencyAutomatic(ref status);			

2.1.31 SetReadTime

Function	UInt32 SetReadTime(uint time_an1)		
Parameter	**IN/OUT**	**Type**	**Description**
time_an1	IN	uint	The time for output power when performing inventory processing (10~40000, 1=1ms)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Set the time for power output when performing inventory processing for the AsReader device.			
Code example: Set the time for power output when performing inventory processing to 1000ms. SetReadTime(1000);			

2.1.32 GetReadTime

Function	UInt32 GetReadTime(ref uint time_an1)		
Parameter	**IN/OUT**	**Type**	**Description**
time_an1	OUT	uint	The time for output power when performing inventory processing (10~40000, 1=1ms)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets the time for power output when performing inventory processing for the AsReader device.			
Code example: GetReadTime(ref time_an1);			

2.1.33 GetSdkVersion

Function	void GetSdkVersion(ref string sdkVersion)		
Parameter	**IN/OUT**	**Type**	**Description**
sdkVersion	OUT	string	The SDK version
Return Value	OUT	void	
Description: Gets the SDK version information.			
Code example: GetSdkVersion(ref sdkVersion);			

2.1.34 SetIdelTime

Function	uint SetIdelTime(uint idelTime)		
Parameter	**IN/OUT**	**Type**	**Description**
idelTime	IN	uint	Idle time (ms)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Sets the read interval for AsReader device to perform inventory processing.			
Code example: Set the read interval for AsReader to perform inventory processing as 10ms. SetIdelTime(10);			

2.1.35 GetIdelTime

Function	uint GetIdelTime(ref uint idelTime)		
Parameter	**IN/OUT**	**Type**	**Description**
idelTime	OUT	uint	Idle time (ms)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the read interval for AsReader device to perform inventory processing.			
Code example: GetIdelTime(ref idelTime);			

2.1.36 DefaultSetting

Function	bool DefaultSetting()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	bool	Returns true on success and false on failure.
Description: Restores default Settings.			
Code example: DefaultSetting();			

2.1.37 SetHIDWorkParams

Function	UInt32 SetHIDWorkParams(int hid_adr,int hid_len, int hid_inventory, int hid_filter_time, byte repeat_epc_tid, byte epc_tid_user, byte output_suffix, byte output_without)		
Parameter	**IN/OUT**	**Type**	**Description**
hid_adr	IN	int	The start address of the area to be read
hid_len	IN	int	The length of the area to be read
hid_inventory	IN	int	HID idle time
hid_filter_time	IN	int	The time interval for reading tags with the same EPC/TID data consecutively. This parameter is valid only when output_without is NO_CHECKED.
repeat_epc_tid	IN	byte	Check whether the data read consecutively is the same target area. (See 2.2.1.26) This parameter is valid only when output_without is NO_CHECKED.
epc_tid_user	IN	byte	The target area to read (See 2.2.1.22)
output_suffix	IN	byte	Suffix of read data (See 2.2.1.23)
output_without	IN	byte	Whether to read the tag with the same EPC/TID value repeatedly during continuous scan (See 2.2.1.24)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1

Description:
Sets parameters for HID mode.

Code example:
```
//Start address: 0
//Length: 0
//Idle time: 0
//Time interval between reading tags with the same EPC/TID data consecutively: 0
//Check whether the data read consecutively is the same target area: IS_CHECKED_EPC
//The target area to read: EPC
```

```
//Suffix of read data: NO_CHECKED
//Whether to read the tag with the same EPC/TID value repeatedly during continuous scan: IS_CHECKED
SetHIDWorkParams(0,0,0,0,IS_CHECKED_EPC,EPC,NO_CHECKED,IS_CHECKED)
```

2.1.38 GetHIDWorkParams

Function	UInt32 GetHIDWorkParams(int hid_adr,int hid_len, int hid_inventory, int hid_filter_time, byte repeat_epc_tid, byte epc_tid_user, byte output_suffix, byte output_without)		
Parameter	**IN/OUT**	**Type**	**Description**
hid_adr	OUT	int	The start address of the area to be read
hid_len	OUT	int	The length of the area to be read
hid_inventory	OUT	int	HID idle time
hid_filter_time	OUT	int	Time interval between reading tags with the same EPC/TID data consecutively This parameter is valid only when output_without is NO_CHECKED.
repeat_epc_tid	OUT	byte	Check whether the data read consecutively is the same target area. (See 2.2.1.26) This parameter is valid only when output_without is NO_CHECKED.
epc_tid_user	OUT	byte	The target area to read (See 2.2.1.22)
output_suffix	OUT	byte	Suffix of read data (See 2.2.1.23)
output_without	OUT	byte	Whether to read the tag with the same EPC/TID value repeatedly during continuous scan (See 2.2.1.24)
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Gets parameters for HID mode.			
Code example: GetHIDWorkParams(ref hid_adr,ref hid_len,ref hid_inventory,ref hid_filter_time,ref repeat_epc_tid,ref epc_tid_user,ref output_suffix,ref output_without)			

2.1.39 SetBuzzer

Function	UInt32 SetBuzzer(Types.Buzzer buzzer)		
Parameter	**IN/OUT**	**Type**	**Description**
buzzer	IN	Types.Buzzer	Buzzer status value (See 2.2.1.25)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Sets the buzzer status.			
Code example: Set the buzzer status to OFF. SetBuzzer(OFF);			

2.1.40 GetBuzzer

Function	UInt32 GetBuzzer(Types.Buzzer buzzer)		
Parameter	**IN/OUT**	**Type**	**Description**
buzzer	OUT	Types.Buzzer	Buzzer status value (See 2.2.1.25)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the buzzer status.			
Code example: GetBuzzer(ref buzzer);			

2.1.41 GetFwVersion

Function	UInt32 GetFwVersion(ref string fwVersion)		
Parameter	**IN/OUT**	**Type**	**Description**
fwVersion	OUT	string	Firmware version
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the firmware version.			
Code example: GetFwVersion(ref fwVersion);			

2.1.42 GetHwVersion

Function	UInt32 GetHwVersion(ref string hwVersion)		
Parameter	**IN/OUT**	**Type**	**Description**
hwVersion	OUT	string	Hardware version
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the hardware version.			
Code example: GetHwVersion(ref hwVersion);			

2.1.43 GetRFIDFwVersion

Function	UInt32 GetRFIDFwVersion(ref string rfidFwVersion)		
Parameter	**IN/OUT**	**Type**	**Description**
rfidFwVersion	OUT	string	The firmware version of the RFID module
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the firmware version of the RFID module.			
Code example: GetRFIDFwVersion(ref rfidFwVersion);			

2.1.44 GetProductSN

Function	UInt32 GetProductSN(ref string productSN)		
Parameter	**IN/OUT**	**Type**	**Description**
productSN	OUT	string	Serial number of the AsReader device
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the serial number of the AsReader device.			
Code example: GetProductSN(ref productSN);			

2.1.45 SetRSSIThreshold

Function	UInt32 SetRSSIThreshold(int rssi_threshold)		
Parameter	**IN/OUT**	**Type**	**Description**
rssi_threshold	IN	int	RSSI threshold (-99~0)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Sets the RSSI threshold.			
Code example: Set the RSSI threshold to -40. SetRSSIThreshold(40);			

2.1.46 GetRSSIThreshold

Function	UInt32 GetRSSIThreshold(ref int rssi_threshold)		
Parameter	**IN/OUT**	**Type**	**Description**
rssi_threshold	OUT	int	RSSI threshold (-99~0)
Return Value	OUT	uint	Succeeded: 0 Failed: 1
Description: Gets the RSSI threshold.			
Code example: GetRSSIThreshold(ref rssi_threshold);			

2.1.47 SendCommand

Function	bool SendCommand(byte[] command)		
Parameter	**IN/OUT**	**Type**	**Description**
command	IN	byte[]	command content
Return Value	OUT	bool	Returns true on success and false on failure.
Description: Sends commands.			

Code example:
byte[] command = new byte[9];
command[0] = 0xBB;
command [1] = 0x00;
command [2] = 0x5B;
command [3] = 0x00;
command [4] = 0x01;
command [5] = 0x00;
command [6] = 0x7E;
command [7] = 0x5F;
command [8] = 0xB4;
SendCommand(command);			

2.1.48 SetDelegate

Function	Void SetDelegate(CallBackReadTagData readTagData, CallBackErrorCode errorCode, CallBackSuccessCode successCode, CallBackCommandData commandData, CallBackReadComplete completeStatus, CallBackTriggerHandler triggerHandler)		
Parameter	**IN/OUT**	**Type**	**Description**
readTagData	IN	CallBackReadTagData	The callback function for data processing.
errorCode	IN	CallBackErrorCode	The callback function for processing failed
successCode	IN	CallBackSuccessCode	The callback function for processing succeeded
commandData	IN	CallBackCommandData	The callback parameters for receiving common data
completeStatus	IN	CallBackReadComplete	Returns the status of RFID automatic stop and scan completion
triggerHandler	IN	CallBackTriggerHandler	Returns the trigger key status
Return Value	OUT	Void	

Description:
Sets the delegate function.

Code example:

```
CallBackReadTagData Rec = null;
CallBackErrorCode Rec1 = null;
CallBackSuccessCode Rec2 = null;
CallBackCommandData Rec3 = null;
CallBackReadComplete Rec4 = null;
CallBackTriggerHandler Rec5 = null;
Void test(InventoryResult ReadTagStruct);
Void test1(uint error);
Void test2(uint success);
Void test3(byte[] commandCallBackData);
Void test4(bool completeStatus);
Void test5(int keyStatus);

Rec = test;
Rec1 = test1;
Rec2 = test2;
Rec3 = test3;
Rec4 = test4;
Rec5 = test5;
SetDelegate(Rec,Rec1,Rec2,Rec3,Rec4,Rec5);
```

```
//Note: Defines the delegate for data processing and the delegate for performing error output.
public delegate void CallBackReadTagData(InventoryResult tagcallbackdata);
public delegate void CallBackErrorCode (uint error);
public delegate void CallBackErrorCode (uint success);
public delegate void CallBackCommandData (byte[] commandCallBackData);
public delegate void CallBackReadComplete (bool completeStatus);
public delegate void CallBackTriggerHandler (int keyStatus);

//InventoryResult:   See Appendix I.
//error: See 2.2.1.16.
//success: See 2.2.1.17.
```

2.1.49 SetHIDInventoryMode

Function	UInt32 SetHIDInventoryMode(Types.HIDInventoryMode mode)		
Parameter	**IN/OUT**	**Type**	**Description**
mode	IN	Types.HIDInventoryMode	The inventory mode of the HID mode (See 2.2.1.27) Manual: Pressing the SCAN button of P3xU will start reading after connecting the P3xU to the mobile device (or the PC). Auto: The reading will start automatically after P3xU is connected to the mobile device (or the PC).
Return Value	OUT	UInt32	Succeeded: 0 Failed: 1
Description: Sets the inventory mode of the HID mode.			

Code example:
Types.HIDInventoryMode mode = Types.HIDInventoryMode.Manual;
SetHIDInventoryMode (mode);			

2.1.50 GetHIDInventoryMode

Function	UInt32 GetHIDInventoryMode(ref Types.HIDInventoryMode mode)		
Parameter	**IN/OUT**	**Type**	**Description**
mode	IN	Types.HIDInventoryMode	The inventory mode of the HID mode (See 2.2.1.27)
Return Value	OUT	UInt32	Succeed: 0 Failed: 1
Description: Gets the inventory mode of the HID mode.			

Code example:
Types.HIDInventoryMode mode = Types.HIDInventoryMode.Manual;
GetHIDInventoryMode (ref mode);			

2.1.51 CheckTagStatus

Function	TagStatus CheckTagStatus(byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
epcData	IN	byte[]	The EPC value of the tag
Return Value	OUT	TagStatus	The status of the tag (See 2.2.1.28)
Description: Gets the status of the tag.			

Code example: Gets the status of the tag with EPC 01010202.
byte[] epcData = {0x01,0x01,0x02,0x02};
TagStatus tagStatus = CheckTagStatus(epcData);			

2.2 Types Class

The Types class is used to define the region types, RFID mode types, session types, search mode types, Selection Mask target session types, action types, memory bank types, and other parameter types.

2.2.1 Enumeration Types

2.2.1.1 InventoryType

Query Parameter	**Value**
PC_EPC_RSSI	1
PC_EPC_TID	2
ONLY_PC_EPC	3

2.2.1.2 RegionType

Region	**Value**
REGION_US	0x21
REGION_US_Narrow	0x22
REGION_Europe	0x31
REGION_JAPAN	0x41
REGION_CHINA2	0x52
REGION_BRAZIL	0x61

2.2.1.3 SessionType

Session types:

Session	**Value**
SESSION_S0	0x0
SESSION_S1	0x1
SESSION_S2	0x2
SESSION_S3	0x3

2.2.1.4 TargetType

Sets the target session for the Selection Mask.

Target	**Value**
SESSION_S0	0x0
SESSION_S1	0x1
SESSION_S2	0x2
SESSION_S3	0x3
SL_FLAG	0x4

2.2.1.5 ActionType

Sets the session status of the tag. ActionType enumeration type that returns what happens to the Session and Session Flag of a tag that matches and does not match with a Selection Mask when using the Selection Mask function.

Action	Value
ACTION_ASLINVA_DSLINVB	0x0
ACTION_ASLINVA_NOTHING	0x1
ACTION_NOTHING_DSLINVB	0x2
ACTION_NSLINVS_NOTHING	0x3
ACTION_DSLINVB_ASLINVA	0x4
ACTION_DSLINVB_NOTHING	0x5
ACTION_NOTHING_ASLINVA	0x6
ACTION_NOTHING_NSLINVS	0x7

2.2.1.6 MemBankType

Sets memory bank of tag which mask data of Selection Mask will be compared to.

MemBank	**Value**
MEM_RESERVED	0x0
MEM_EPC	0x1
MEM_TID	0x2
MEM_USER	0x3

2.2.1.7 ChannelType

RF Channel	**Value**
CHANNEL_24	24
CHANNEL_25	25
CHANNEL_26	26
CHANNEL_27	27
CHANNEL_28	28
CHANNEL_29	29
CHANNEL_30	30
CHANNEL_31	31
CHANNEL_32	32

2.2.1.8 GainType

PA Gain	**Value**
HIGH_GAIN	0x00
LOW_GAIN	0x01

2.2.1.9 TargetABType

Session Flag	Value
TARGET_A	0x00
TARGET_B	0x01
TOGGLE_INVENTORY_ROUBD	0x02

2.2.1.10 DRType

TRcal Divide Ratio	Value
DR_8	0x00
DR_64_3	0x01

2.2.1.11 MType

M (cycles per symbol) sets the T=>R data rate and modulation format.

M	Value
M1	0x00
M2	0x01
M4	0x02
M8	0x03

2.2.1.12 TRextType

TRext chooses whether a Tag prepends the T=>R preamble with a pilot tone.

TRext	Value
NO_Pilot_Tone	0x00
Use_Pilot_Tone	0x01

2.2.1.13 SelType

Sel chooses which Tags respond to the Query.

Sel	Value
SEL_ALL	0x00
SEL_SL_N	0x02
SEL_SL	0x03

2.2.1.14 QType

Q sets the number of slots in the round.

Q	Value
Q0	0
Q1	1
Q2	2
Q3	3
Q4	4
Q5	5
Q6	6
Q7	7
Q8	8

2.2.1.15 AntiCollisionMode

Anti-Collision Mode	Value
FixedQ	0x0
DynamicQ	0x01

2.2.1.16 ErrorCode

Error Code	Value
OTHER_ERROR	0x0
NOT_SUPPORTED	0x1
INSUFFICIENT_PRIVILEGES	0x2
MEMORY_OVERRUN	0x3
MEMORY_LOCKED	0x4
CRYPTO_SUITE_ERROR	0x5
COMMAND_NOT_ENCAPSULATED	0x6
RESPONSEBUFFER_OVERFLOW	0x7
SECURITY_TIMEOUT	0x8
INSUFFICIENT_POWER	0xB
NON_SPECIFIC_ERROR	0xF
SENSOR_SCHEDULING_CONFIGURATION	0x11
TAG_BUSY	0x12
MEASUREMENT_TYPE_NOT_SUPPORTED	0x13
NO_TAG_DETECTED	0x80
HANDLE_ACQUISITION_FAILURE	0x81
ACCESS_PASSWORD_FAILURE	0x82
KILL_PASSWORD_FAILURE	0x83
CRC_ERROR	0x90

RX_TIMEOUT	0x91
REGISTRY_UPDATE_FAILURE	0xA0
REGISTRY_ERASE_FAILURE	0xA1
REGISTRY_WRITE_FAILURE	0xA2
REGISTRY_NOT_EXIST	0xA3
UART_FAILURE	0xB0
SPI_FAILURE	0xB1
I2C_FAILURE	0xB2
GPIO_FAILURE	0xB3
NOT_SUPPORTED_COMMAND	0xE0
UNDEFINED_COMMAND	0xE1
INVALID_PARAMETER	0xE2
TOO_HIGH_PARAMETER	0xE3
TOO_LOW_PARAMETER	0xE4
FAILURE_AUTOMATIC_READ_OPERATION	0xE5
NOT_AUTOMATIC_READ_MODE	0xE6
FAILURE_TO_GET_LAST_RESPONSE	0xE7
FAILURE_TO_CONTROL_TEST	0xE8
FAILURE_TO_RESET_READER	0xE9
RFID_BLOCK_CONTROL_FAILURE	0xEA
PR9200_BUSY	0xEB
COMMAND_FAILURE	0xF0
VERIFY_FAILURE	0xF1
ABNORMAL	0xFC
ERROR_NONE	0xFF

2.2.1.17 SuccessCode

Success Code	Value
SET_READER_POWER_CONTROL	0x0
GET_READER_INFORMATION	0x03
GET_REGION	0x06
SET_REGION	0x07
SET_SYSTEM_RESET	0x08
GET_TYPE_C_AI_SELECT_PARAMETERS	0xB
SET_TYPE_C_AI_SELECT_PARAMETERS	0xC
GET_TYPE_C_AI_QUERY_RELATED_PARAMETERS	0xD
SET_TYPE_C_AI_QUERY_RELATED_PARAMETERS	0xE
GET_CURRENT_RF_CHANNEL	0x11
SET_CURRENT_RF_CHANNEL	0x12
GET_FH_AND_LBT_PARAMETERS	0x13

SET_FH_AND_LBT_PARAMETERS	0x14
GET_TX_POWER_LEVEL	0x15
SET_TX_POWER_LEVEL	0x16
RF_CW_SIGNAL_CONTROL	0x17
GET_MULTIPLE_POWER	0x18
SET_MULTIPLE_POWER	0x19
GET_READ_TIME	0x1e
SET_READ_TIME	0x1f
SET_ANTENNA	0x1B
READ_TYPE_C_UII	0x22
READ_TYPE_C_UII_RSSI	0x23
READ_TYPE_C_USER_DATA	0x24
READ_TYPE_C_UII_TID	0x25
START_AUTO_READ	0x27
STOP_AUTO_READ	0x28
READ_TYPE_C_TAG_DATA	0x29
READ_TYPE_C_TAG_DATA2	0x2A
GET_SESSION	0x2E
SET_SESSION	0x2F
GET_FREQUENCY_HOPPING_TABLE	0x30
SET_FREQUENCY_HOPPING_TABLE	0x31
GET_MODULATION	0x32
SET_MODULATION	0x33
GET_ANTICOLLISION_MODE	0x34
SET_ANTICOLLISION_MODE	0x35
START_AUTO_READ2	0x36
STOP_AUTO_READ2	0x37
START_AUTO_READ_RSSI	0x38
STOP_AUTO_READ_RSSI	0x39
START_AUTO_READ_EX2	0x3A
WRITE_TYPE_C_TAG_DATA	0x46
BLOCKWRITE_TYPE_C_TAG_DATA	0x47
BLOCKERASE_TYPE_C_TAG_DATA	0x48
ISP_DATA	0x57
KILL_RECOM_TYPE_C_TAG	0x65
SET_GAIN	0x66
GET_GAIN	0x67
LOCK_TYPE_C_TAG	0x82
BLOCKPERMALOCK_TYPE_C_TAG	0x83
SET_MODEM_REGISTER	0xA6

SET_RF_REGISTER	0xA7
GET_MODEM_REGISTER	0xA8
GET_RF_REGISTER	0xA9
GET_RSSI	0xC5
SCAN_RSSI	0xC6
UPDATE_REGISTRY	0xD2
ERASE_REGISTRY	0xD3
GET_REGISTRY_ITEM	0xD4
SET_REGISTRY_ITEM	0xD5
SET_OPTIMUM_FREQUENCY_HOPPING_TABLE	0xE4
GET_FREQUENCY_HOPPING_MODE	0xE5
SET_FREQUENCY_HOPPING_MODE	0xE6
GET_TX_LEAKAGE_RSSI_LEVEL_FOR_SMART_HOPPING_MODE	0xE7
SET_TX_LEAKAGE_RSSI_LEVEL_FOR_SMART_HOPPING_MODE	0xE8
START_READ_WITH_FAST_LEAKAGE_CAL	0xEC
REQUEST_FAST_LEAKAGE_CAL	0xED
SET_HID_WORK_PARAMS	0x51
GET_HID_WORK_PARAMS	0x52
SET_BUZZER	0x53
GET_BUZZER	0x54
SET_HID_VIBRATOR	0x55
GET_HID_VIBRATOR	0x56
GET_DEVICE_MODE	0x57
GET_UPDATE_ADDRESS	0x58
TRANSFER_FILE	0x59
TRANSFER_COMPLETE	0x5A
DEVICE_REBOOT	0x5B
GET_FW_VERSION	0x5C
DEFAULT_SETTING	0x5D
TRIGGER_HANDLER	0x5E
GET_SELECTION_MASK	0xAE
SET_SELECTION_MASK	0xAF
GET_SELECTION_ENABLE	0x8E
SET_SELECTION_ENABLE	0x8F
SET_RSSI_THRESHOLD	0x5F
GET_RSSI_THRESHOLD	0x61
GET_HW_VERSION	0x62
GET_PRODUCT_SN	0x63

2.2.1.18 SelectionEnable

Whether the Selection mask is enabled	Value
DISABLE	0x00
ENABLE	0x01

2.2.1.19 FHType

FH	Value
DISABLE	0x00
ENABLE	0x01
WITH_LBT	0x02

2.2.1.20 LBTType

LBT	Value
DISABLE	0x00
ENABLE	0x01
WITH_LBT	0x02

2.2.1.21 CWType

CW	Value
DISABLE	0x00
ENABLE	0x01

2.2.1.22 HidEpcTidUser

epc_tid_user	Value
EPC	0x00
TID	0x01
USER	0x02

2.2.1.23 HidOutputSuffix

output_suffix	Value
NO_CHECKED	0x00
ENTER	0x01
TAB	0x02
BACKSPACE	0x03
COMMA	0x04

2.2.1.24 HidOutputWithout

output_without	**Value**
IS_CHECKED	0x01
IS_TID_CHECKED	0x02
NO_CHECKED	0x00

2.2.1.25 Buzzer

buzzer	**Value**
OFF	0x00
LOW	0x01
HIGH	0x02

2.2.1.26 HidRepeatEpcTid

repeat_epc_tid	**Value**
IS_CHECKED_EPC	0x00
IS_CHECKED_TID	0x01

2.2.1.27 HIDInventoryMode

HID_Inventory_Mode	**Value**
Manual	0x00
Auto	0x01

2.2.1.28 TagStatus

TagStatus	**Value**
UnLock	0x00
Lock	0x01
PermaLock	0x02
Unknown	0x03
Error	0x04

Appendix I

1. InventoryResult

Items displayed when inventory tags
rssi
channel
phase
antenna
TagData tagData

2. TagData

Tag Data
pc
epc
tid
data

Appendix II

TagMask
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2

TagAction
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2

AsReaderP3xU SDK

C# SDK Reference Guide

Nov. 2024 2nd Edition

AsReader Inc.

111 SW 5th Ave., Ste 3150

Portland, OR 97204-3656　U.S.A.

Tel.: (503) 770-2777 x102

Asterisk Inc.

AsTech Osaka Building 5F,

2-2-1, Kikawa-nishi,

Yodogawa-ku, Osaka, 532-0013, JAPAN