AsReader®

AsReaderP3xU SDK

C# SDK 開発マニュアル V1.2

Asterisk Inc.

修正履歴

No.	バージョン	修正内容	日付
1	1.0	新規作成	2023/3/21
2	1.1	SetHIDInventoryMode、GetHIDInventoryMode、HIDInventoryMode の追加	2024/5/9
3	1.2	CheckTagStatus、TagStatus の追加	2024/11/6

目 次

概 要

本マニュアルはSDKを使用するWindowsデスクトップアプリケーション開発者向けに以下の内容を提供します。

- 開発環境の構築方法。
- SDK ライブラリの各ファンクションの説明。

開発ツール：

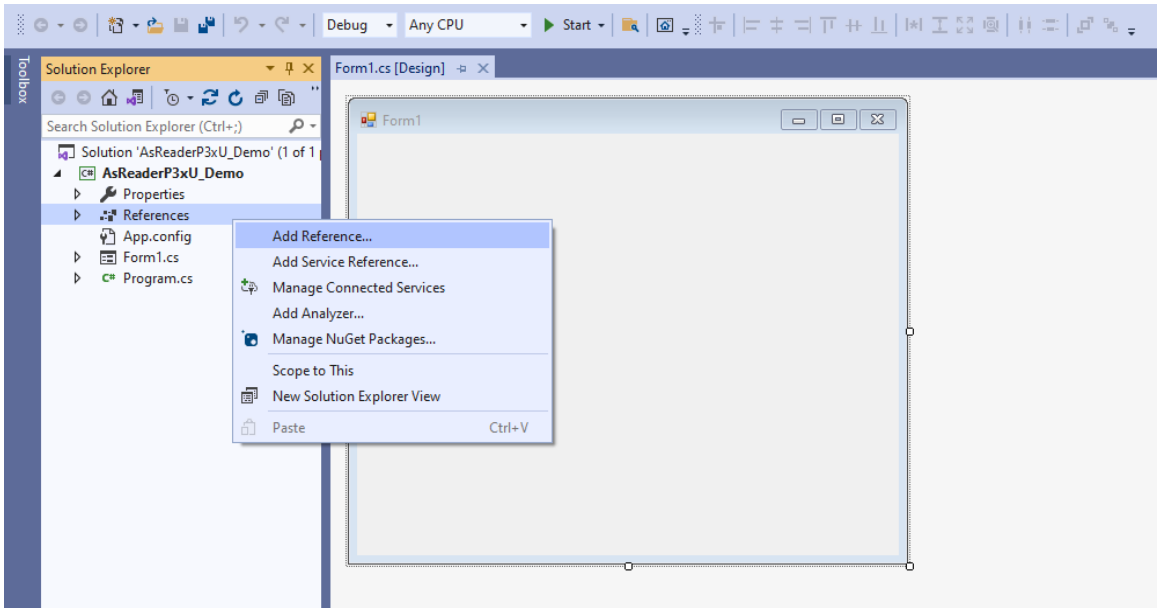
- Visual Studio 2012 以上

1. 開発環境の構築

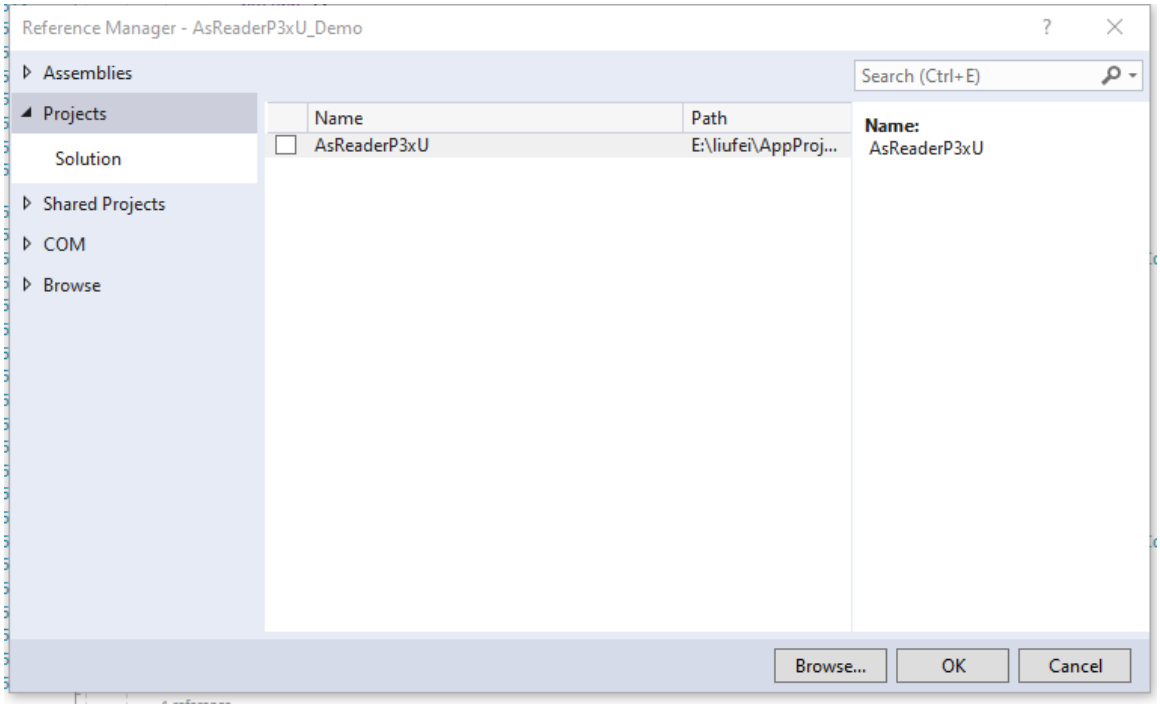
1.1. SDK の追加

1. Windows デスクトップアプリケーションの新規作成
   AsReaderP3xU.dll をプロジェクトフォルダ内にコピーします。
開発・デバッグ時は、プログラムのカレントディレクトリの bin/Debug フォルダの下にコピーします。
   リリース時は、プログラムのカレントディレクトリの bin/Release フォルダの下にコピーします。

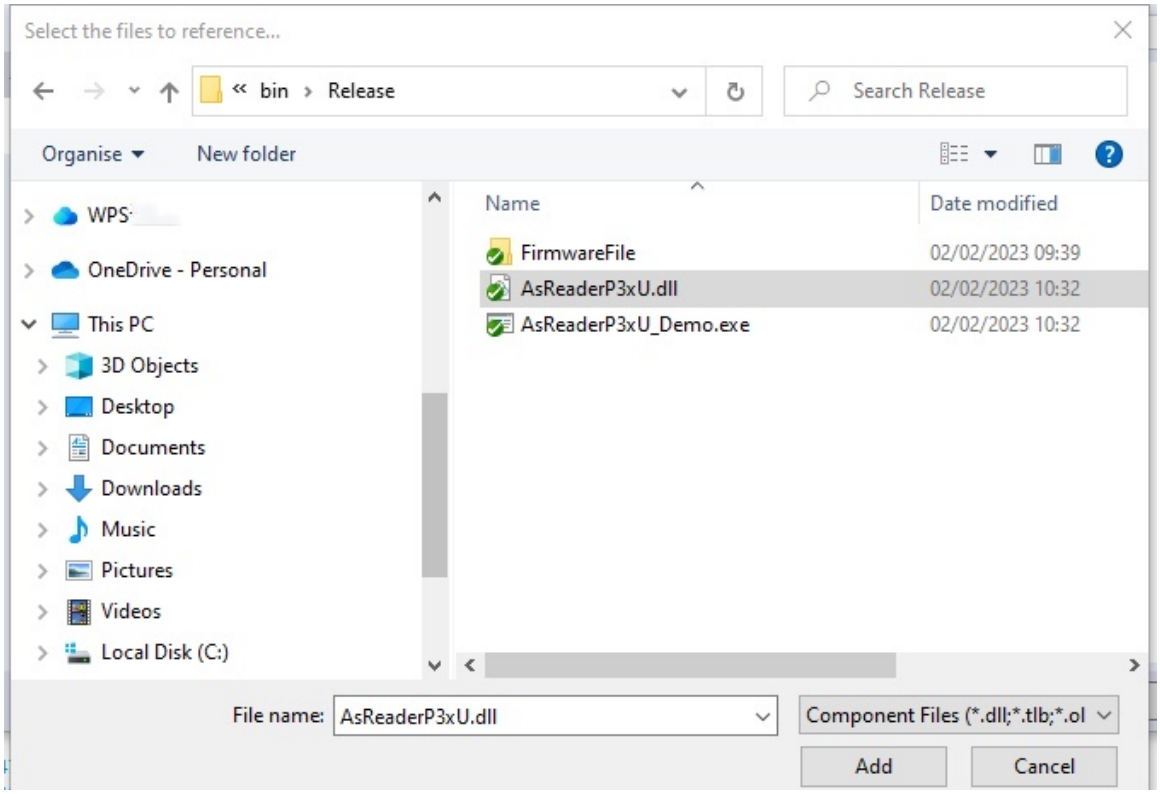
2. Reference に AsReaderPxU.dll を追加
   「References」を右クリックし、「Add Reference」を選択します。



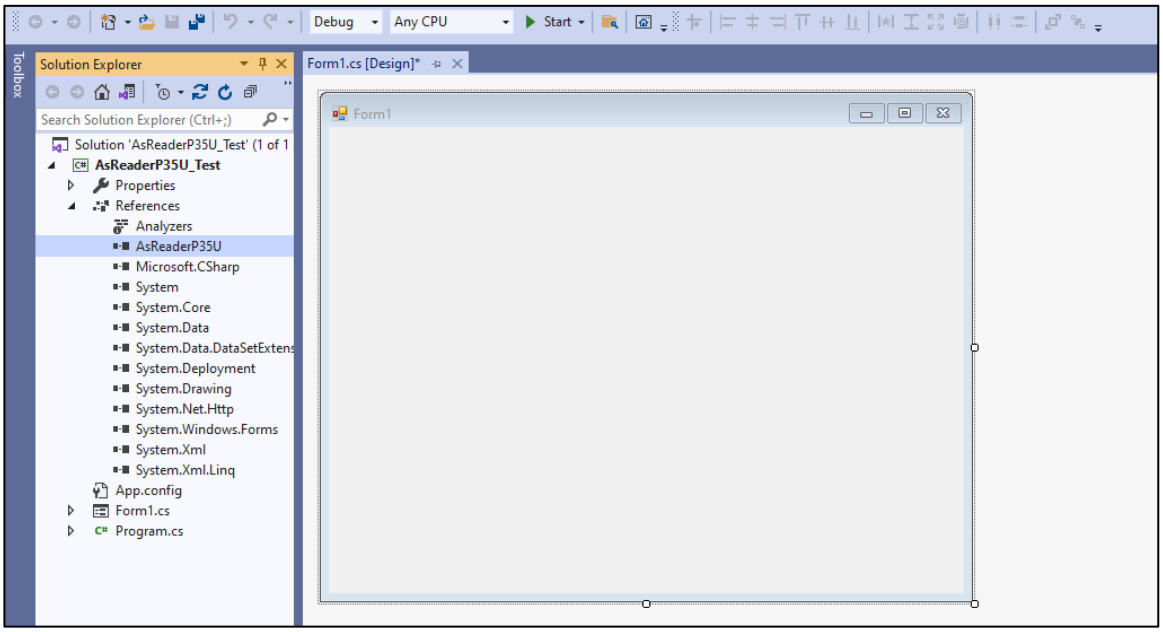
3. 「Browse」ボタンをクリックします。



4. 「Select the files reference」ダイアログボックスで、プロジェクトパスに AsReaderPxU.dll を選択し、「Add」ボタンをクリックします。



5. 正常に追加されると、Referense リストに表示されます。



1.2. 名前空間を参照する

```
using AsReaderP3xU;
```

1.3. SDK の使用方法

以下は AsReader クラスの関数を呼び出して、AsReader と接続する場合の使用例となります。

1.3.1. AsReader との接続

1. AsReader オブジェクトの準備

「AsReader」クラスのコンストラクタを呼び出して、AsReader のオブジェクトを取得します。

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

2. 関数 ConnectWithVCP（2.1.1 を参照）を実行して、AsReader との接続を行います。

注意：成功に実行する場合、0 を返します。実行には失敗する場合、1 を返します。

COM ポートをパラメータにセットして関数 asreader.ConnectServerWithRS232 を呼び出します。接続が成功すると、関数は 0 を返します。

```
UInt32 ret = asreader.ConnectWithVCP("COM1");
if(ret == 0){
    //接続成功時の処理。
}else{
    //接続失敗時の処理。
}
```

2. AsReader クラス

2.1. 関数

AsReader クラスでは RF タグのインベントリ、読取、書き込み、ロックなどの関数を提供します。

2.1.1. ConnectWithVCP

関数名	UInt32 ConnectWithVCP(string comPort)			
引数	引数名	IN/OUT	型	説明
	comPort	IN	string	AsReader の COM ポート番号
返り値	-	OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： AsReader と USB で接続する際に使用します。 COM ポートを設定し、AsReader と接続します。 **サンプルコード：** ConnectWithVCP(“COM1”);				

2.1.2. DisConnect

関数名	UInt32 DisConnect()			
引数	引数名	IN/OUT	型	説明
戻り値	-	Out	UInt32	実行成功：0 実行失敗：1
関数の説明： AsReader との接続を切断し、AsReader をリセットします。 **サンプルコード：** DisConnect();				

2.1.3. StartInventory

関数名	UInt32 StartInventory (bool rssiEn, byte mtnu, byte mtime, UInt16 rc, bool an1)

引数	引数名	IN/OUT	型	説明
	rssiEn	IN	bool	true: rssi を表示する false：rssi を表示しない
	mtnu	IN	byte	インベントリを停止する読み取りタグ数
	mtime	IN	byte	インベントリの継続時間（s）
	rc	IN	UInt16	インベントリラウンド回数
	an1	IN	bool	true: アンテナポートオン false：アンテナポートオフ
戻り値	-	OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

AsReader はインベントリを実行します。インベントリの停止条件（インベントリラウンド回数、読み取りタグ数、継続時間）、rssi の表示要否を設定可能です。複数の条件を設定した場合は、いずれかの条件を満たしたタイミングでインベントリが停止します。

サンプルコード：

インベントリラウンド回数を 10、インベントリを停止する読み取りタグ数を 100、インベントリの継続時間を 60s、rssi 表示しないに設定します。

```
StartInventory(false,100,60,10,true);
```

2.1.4. StopInventory

関数名	UInt32 StopInventory()			
引数	引数名	IN/OUT	型	説明
戻り値	-	Out	UInt32	実行成功：0 実行失敗：1

関数の説明：

インベントリを停止します。

サンプルコード：

```
StopInventory();
```

2.1.5. SetSelectMask

関数名	UInt32 SetSelectMask(Types.MemBankType memBank, Types.TargetType target, Types.ActionType action, UInt32 startAddressWord, byte[] selectMask)			
引数	引数名	IN/OUT	型	説明
	memBank	IN	Types.MemBankType	Selection Mask 対象のメモリバンク 3.1.6 を参照
	target	IN	Types.TargetType	Selection Mask の対象の Session 3.1.4 を参照
	action	IN	Types.ActionType	タグがマークされた後のアクション 3.1.5 を参照
	startAddressWord	IN	UInt32	スタートアドレス 単位：word
	selectMask	IN	byte[]	マスクの値 単位：word
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

マスクのパラメータを設定します。

対象のタグに対してのみインベントリ、読み取り、書き込み、ロックなどの操作をすることができます。

サンプルコード：

マスク条件：

メモリバンク： EPC；

Session ：SESSION_S0；

Action：ACTION_ASLINVA_DSLINVB；

スタートアドレス：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

関数名	UInt32 GetSelectMask(ref Types.MemBankType memBank, ref Types.TargetType target, ref Types.ActionType action, ref UInt32 startAddressWord, ref string selectMask)			
引数	引数名	IN/OUT	型	説明
	memBank	OUT	Types.MemBankType	Selection Mask 対象のメモリバンク 3.1.6 を参照
	target	OUT	Types.TargetType	Selection Mask の対象の Session （3.1.4 を参照）
	action	OUT	Types.ActionType	タグがマークされた後のアクション 3.1.5 を参照
	startAddressWord	OUT	UInt32	スタートアドレス 単位：word
	selectMask	OUT	string	マスクの値 単位：word
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
マスクのパラメータを取得します。

サンプルコード：
GetSelectMask(ref membank, ref target, ref action, ref startAddressWord, ref epc_string);

2.1.7. SetSelectionEnable

関数名	UInt32 SetSelectionEnable(Types.SelectionEnable selection_enable)			
引数	引数名	IN/OUT	型	説明
	selection_enable	IN	Types.SelectionEnable	選択されているマスクを使用するかどうか（3.1.18 を参照）
戻り値	-	OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
選択されているマスクを使用するかどうかを設定します。

サンプルコード：
SetSelectionEnable(ENABLE);

2.1.8. GetSelectionEnable

関数名	UInt32 GetSelectionEnable(ref Types.SelectionEnable selection_enable)			
引数	引数名	IN/OUT	型	説明
	selection_enable	OUT	Types.SelectionEnable	選択されているマスクを使用するかどうか（3.1.18 を参照）
戻り値	-	OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
選択されているマスクを使用するかどうかを取得します。

サンプルコード：
GetSelectionEnable(ref selection_enable);

2.1.9. WriteMemory

関数名	UInt32 WriteMemory(Types.MemBankType memBank, uint startAddressWord,uint accessPassword, byte[] writeData, byte[] epcData)			
引数	引数名	IN/OUT	型	説明
	memBank	IN	Types.MemBankType	書き込み対象のメモリーバンク 3.1.6 を参照
	startAddressWord	IN	uint	書き込み先メモリのオフセット 単位：word
	accessPassword	IN	uint	対象タグのアクセスパスワード(パスワードを設定してない場合、0)
	writeData	IN	byte[]	書き込みデータ
	epcData	IN	byte[]	書き込み対象のタグの EPC 値
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

1、epcData で対象タグを選定して、タグの対象メモリバングにデータを書き込みます。

2、書き込みデータの長さは 32 ワード（Words）／64 バイト（Bytes）が最大です。

サンプルコード：

メモリバンク：EPC

スタートアドレス：2

アクセスパスワード：0x12345678

書き込みデータ：byte[] writedata = {0x12,0x34};

書き込み対象タグの EPC 値：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

関数名	UInt32 ReadMemory(Types.MemBankType memBank, uint startAddressWord, uint lengthWord, uint accessPassword, byte[] epcData)			
引数	引数名	IN/OUT	型	説明
	memBank	IN	Types.MemBankType	読取対象タグのメモリーバンク 3.1.6 を参照
	startAddressWord	IN	uint	読取先のメモリのオフセット 単位：word
	lengthWord	IN	uint	読取メモリの長さ
	accessPassword	IN	uint	対象タグのアクセスパスワード(パスワードがない場合、0)
	epcData	IN	byte[]	タグの EPC データ
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

1、epcData で対象タグを選定して、タグの対象メモリバングのデータを読取します。

2、読み取りするデータの長さは 32 ワード（Words）／64 バイト（Bytes）が最大です。

サンプルコード：

メモリバンク：EPC

オフセット：2

長さ：2

アクセスパスワード：0x12345678

対象タグの EPC 値：

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

関数名	UInt32 Kill(uint killPassword, byte[] epcData)			
引数	引数名	IN/OUT	型	説明
	KillPassword	IN	uint	キル対象のタグのキルパスワード
	epcData	IN	byte[]	キル対象のタグの EPC 値
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

1、epcData で対象タグを選定して、キルします。

2、タグをキルする前に、パスワードを RESERVED バンクに書き込む必要があります。

パスワードはオフセット 00 から 2Word を書き込みます。

注：タグをキルすると、回復できません。

サンプルコード：

キルパスワード：0x12345678。

キル対象タグの EPC 値：

```
byte[] epcData=
{0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78};
Kill(0x12345678,epcData);
```

2.1.12. LockMemory

関数名	UInt32 LockMemory(TagMask tagMask, TagAction tagAction, uint accessPassword, byte[] epcData)			
引数	引数名	IN/OUT	型	説明
	tagMask	IN	TagMask	ロック操作のマスク設定値 付録 II を参照
	tagAction	IN	TagAction	ロック操作のアクション設定値 付録 II を参照
	accessPassword	IN	uint	ロック対象のタグのアクセスパスワード (パスワードを設定してない場合、0)
	epcData	IN	byte[]	ログ対象のタグの EPC 値
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

タグのメモリバンクに対してロック（Lock）、永久ロック（PermaLock）、アンロック（Unlock）永久アンロック（PermaUnlock）をします。

タグをロックする前に、アクセスパスワードの設定が必要です。

実際のタグのロック操作にはロックコマンドが使用されます。ロックコマンドは 20 桁の Payload を含んでいますが、マスク設定値は Payload の前 10 桁、アクション設定値は後 10 桁に該当します。マスク設定値に「True」を設定すると Payload に「1」が設定されロック対象になります。アクション設定値も「True」を設定すると Payload に「1」が設定されアンロック、永久的アンロック、ロック、永久ロックを行います。

ロックコマンドの Payload は以下通り：

Lock-Command Payload

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Kill Mask		Access Mask		EPC Mask		TID Mask		User Mask		Kill Action		Access Action		EPC Action		TID Action		User Action	

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

サンプルコード：

対象タグのアクセスパスワード：12345678。
対象タグの EPC 値は byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,
0x56x,0x78,0x12,0x34,0x56,0x78};
Lock 操作のマスクは TagMask tagMask={ false, false, false, false, true, true, false, false,
false, false}
Lock 操作のデータは TagAction tagAction={ false, false, false, false, true, false, false, false,
false, false}
LockMemory(tagMask, tagAction,0x12345678,epcData);

2.1.13. SetRegion

関数名	UInt32 SetRegion(Types.RegionType region)			
引数	引数名	IN/OUT	型	説明
	region	IN	Types.RegionType	設定する Region 3.1.2 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
Region を設定します。

サンプルコード：
Region を REGION_JAPAN に設定
SetRegion(REGION_JAPAN);

2.1.14. GetRegion

関数名	UInt32 SetRegion(Types.RegionType region)			
引数	引数名	IN/OUT	型	説明
	region	OUT	Types.RegionType	取得した Region 3.1.2 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： Region を取得します。 **サンプルコード：** GetRegion(ref region);				

2.1.15. GetTxPower

関数名	UInt32 GetTxPower(ref uint power, ref uint minPower, ref uint maxPower)			
引数	引数名	IN/OUT	型	説明
	power	OUT	uint	電波出力の設定値（JP：13～23；その他：13～27）
	minPower	OUT	uint	最小出力値（13）
	maxPower	OUT	uint	最大出力値（JP：23；その他：27）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： AsReader からの電波出力を取得します。 **サンプルコード：** GetTxPower(ref power,ref minPower,ref maxPower);				

2.1.16. SetTxPower

関数名	UInt32 SetTxPower(uint txPower)			
引数	引数名	IN/OUT	型	説明
	txPower	IN	uint	電波出力の設定値（JP：13～23；その他：13～27）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
AsReader からの電波出力を設定します。

サンプルコード：
電波出力を 22dBm に設定します。
SetTxPower(22);

2.1.17. SetSession

関数名	UInt32 SetSession(Types.SessionType session)			
引数	引数名	IN/OUT	型	説明
	session	IN	Types.SessionType	インベントリ処理実行時の Session 値 3.1.3 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
インベントリ処理実行時の Session 値を設定します。

サンプルコード：
Session 値を S0 に設定します。
SetSession(SESSION_S0);

2.1.18. GetSession

関数名	UInt32 GetSession(ref Types.SessionType session)			
引数	引数名	IN/OUT	型	説明
	session	OUT	Types.SessionType	インベントリ処理実行時の Session 値 3.1.3 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： インベントリ処理実行時の Session 値を取得します。 **サンプルコード：** GetSession(ref session);				

2.1.19. SetChannel

関数名	UInt32 SetChannel(uint channel)			
引数	引数名	IN/OUT	型	説明
	channel	IN	uint	タグ読取時の Channel 値 3.1.7 を参照 Channel 値の範囲は Region によって異なり
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： タグ読取時の Channel 値を設定します。 **サンプルコード：** Channel 値を 24 に設定します。 SetChannel(CHANNEL_24);				

2.1.20. GetChannel

関数名	UInt32 GetChannel(ref uint channel)			
引数	引数名	IN/OUT	型	説明
	channel	OUT	uint	タグ読取時の Channel 値 3.1.7 を参照 Channel 値の範囲は Region によって異なり
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
タグ読取時の Channel 値を取得します。

サンプルコード：
GetChannel(ref channel);

2.1.21. SetBasicTarget

関数名	UInt32 SetBasicTarget(Types.TargetABType target)			
引数	引数名	IN/OUT	型	説明
	target	IN	Types.TargetABType	タグ読取時の Session Flag 値 3.1.9 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
タグ読取時の Target 値を設定します。

サンプルコード：
Session Flag 値を TARGET_A に設定します。
SetBasicTarget(TARGET_A);

2.1.22. GetBasicTarget

関数名	UInt32 GetBasicTarget(ref Types.TargetABType target)			
引数	引数名	IN/OUT	型	説明
	target	OUT	Types.TargetABType	タグ読取時の Session Flag 値 3.1.9 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： タグ読取時の Target 値を取得します。 **サンプルコード：** GetBasicTarget(ref target);				

2.1.23. SetQuery

関数名	UInt32 SetQuery(Types.DRType dr, Types.MType m, Types.TRextType trext, Types.SelType sel, Types.SessionType session, Types.TargetABType target, Types.QType q)			
引数	引数名	IN/OUT	型	説明
	dr	IN	Types.DRType	DR 値（タグからの応答のサブキャリア周波数） 3.1.10 を参照
	m	IN	Types.MType	M 値（タグからの応答の符号化方式） 3.1.11 を参照
	trext	IN	Types.TRextType	TRext 値（3.1.12 を参照）
	sel	IN	Types.SelType	Sel 値（3.1.13 を参照）
	session	IN	Types.SessionType	Session 値（3.1.3 を参照）
	target	IN	Types.TargetABType	読取対象タグの Session Flag 値 3.1.9 を参照
	q	IN	Types.QType	Q 値（タグが応答を返すタイムスロット数） スロット数＝2 の Q 乗 3.1.14 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
インベントリ処理実行時の Query パラメータ値を設定します。

サンプルコード：
Query パラメータに DR 値=DR_8、M 値=M1、Sel 値=SEL_ALL、Session 値=SESSION_S0、Session Flag 値＝TARGET_A、インベントリ時のスロット=4 に設定します。

```
SetQuery(DR_8,M1,NO_Pilot_Tone,SEL_ALL,SESSION_S0,TARGET_A,Q4);
```

2.1.24. GetQuery

関数名	UInt32 GetQuery(ref Types.DRType dr, ref Types.MType m, ref Types.TRextType trext, ref Types.SelType sel, ref Types.SessionType session, ref 19 19 Types.TargetABType target, ref Types.QType q)			
引数	引数名	IN/OUT	型	説明
	dr	OUT	Types.DRType	DR 値（タグからの応答のサブキャリア周波数） 3.1.10 を参照
	m	OUT	Types.MType	M 値（タグからの応答の符号化方式） 3.1.11 を参照
	trext	OUT	Types.TRextType	TRext 値（3.1.12 を参照）
	sel	OUT	Types.SelType	Sel 値（3.1.13 を参照）
	session	OUT	Types.SessionType	Session 値（3.1.3 を参照）
	target	OUT	Types.TargetABType	読取対象タグの Session Flag 値 3.1.9 を参照
	q	OUT	Types.QType	Q 値（タグが応答を返すタイムスロット数） スロット数＝2 の Q 乗 3.1.14 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
インベントリ処理実行時の Query パラメータ値を取得します。

サンプルコード：
GetQuery(ref dr,ref m,ref trext,ref sel,ref session,ref target,ref q);

2.1.25. SetAntiCollisionMode

関数名	UInt32 SetAntiCollisionMode(Types.AntiCollisionMode anticollisionmode, Types.QType startq, Types.QType minq, Types.QType maxq)			
引数	引数名	IN/OUT	型	説明
	anticollisionmode	IN	Types.AntiCollisionMode	インベントリ処理実行時のアンチコリジョンモード（3.1.15 を参照）
	startq	IN	Types.QType	スタート Q 値（3.1.14 を参照）
	minq	IN	Types.QType	最小 Q 値（3.1.14 を参照）
	maxq	IN	Types.QType	最大 Q 値（3.1.14 を参照）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
インベントリ処理実行時のアンチコリジョンモードを設定します。

サンプルコード：
アンチコリジョンモード：FixedQ
スタート Q 値：Q1
最小 Q：Q2
最大 Q：Q8
SetAntiCollisionMode(FixedQ,Q1,Q2,Q8);

2.1.26. GetAntiCollisionMode

関数名	UInt32 GetAntiCollisionMode(ref Types.AntiCollisionMode anticollisionmode, ref Types.QType startq, ref Types.QType minq, ref Types.QType maxq)			
引数	引数名	IN/OUT	型	説明
	anticollisionmode	OUT	Types.AntiCollisionMode	インベントリ処理実行時のアンチコリジョンモード（3.1.15 を参照）
	startq	OUT	Types.QType	スタート Q 値（3.1.14 を参照）
	minq	OUT	Types.QType	最小 Q 値（3.1.14 を参照）
	maxq	OUT	Types.QType	最大 Q 値（3.1.14 を参照）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
インベントリ処理実行時のアンチコリジョンモードを取得します。

サンプルコード：
GetAntiCollisionMode(ref antiCollisionMode,ref startq,ref minq,ref maxq);

2.1.27. SetFH_LBT

関数名	UInt32 SetFH_LBT(uint readTime, uint idelTime, uint cst, uint rfl, Types.FHType fh, Types.LBTType lbt, Types.CWType cw)			
引数	引数名	IN/OUT	型	説明
	readTime	IN	uint	インベントリ時間（10～40000、1＝1ms）
	idelTime	IN	uint	読取間隔時間（ms）
	cst	IN	uint	キャリア検出時間（1＝1ms）
	rfl	IN	uint	電波出力レベル(-dBm x 10)
	fh	IN	Types.FHType	オン（0x01 以上）/オフ（0x00） 3.1.19 を参照
	lbt	IN	Types.LBTType	オン（0x01 以上）/オフ（0x00） 3.1.20 を参照
	cw	IN	Types.CWType	オン（0x01 以上）/オフ（0x00） 3.1.21 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
FH と LBT パラメータを設定します。

サンプルコード：
インベントリ時間：400
読取間隔時間：100
キャリア検出時間：10
電波出力レベル：-740
fh：DISABLE
lbt：DISABLE
cw：DISABLE
SetFH_LBT(400,100,10,-740,DISABLE,DISABLE,DISABLE)

2.1.28. GetFH_LBT

関数名	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)			
引数	引数名	IN/OUT	型	説明
	readTime	OUT	uint	インベントリ時間（10～40000、1＝1ms）
	idelTime	OUT	uint	読取間隔時間（ms）
	cst	OUT	uint	キャリア検出時間（1＝1ms）
	rfl	OUT	uint	電波出力レベル(-dBm x 10)
	fh	OUT	Types.FHType	オン（0x01 以上）/オフ（0x00） 3.1.19 を参照
	lbt	OUT	Types.LBTType	オン（0x01 以上）/オフ（0x00） 3.1.20 を参照
	cw	OUT	Types.CWType	オン（0x01 以上）/オフ（0x00） 3.1.21 を参照
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

FH と LBT パラメータを取得します。

サンプルコード：

GetFH_LBT(ref readTime,ref idelTime,ref cst,ref rfl,ref fh,ref lbt,ref cw)

2.1.29. SetFrequencyAutomatic

関数名	UInt32 SetFrequencyAutomatic(bool status)			
引数	引数名	IN/OUT	型	説明
	status	IN	bool	true：自動的に周波数を設定する false：自動的に周波数を設定しない
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： 自動的に周波数を設定するかどうかを設定します。 **サンプルコード：** SetFrequencyAutomatic(true);				

2.1.30. GetFrequencyAutomatic

関数名	UInt32 GetFrequencyAutomatic(bool status)			
引数	引数名	IN/OUT	型	説明
	status	OUT	bool	true：自動的に周波数を設定する false：自動的に周波数を設定しない
戻り値		OUT	UInt32	実行成功：0 実行失敗：1
関数の説明： 自動的に周波数を設定するかどうかを取得します。 **サンプルコード：** GetFrequencyAutomatic(ref status);				

2.1.31. SetReadTime

関数名	UInt32 SetReadTime(uint time_an1)			
引数	引数名	IN/OUT	型	説明
	status	IN	uint	インベントリ処理実行時の電波出力時間（10～40000、1＝1ms）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
AsReader のインベントリ処理実行時の電波出力時間を設定します。

サンプルコード：
インベントリ処理実行時の電波出力時間を 1000ms に設定します。
SetReadTime(1000);

2.1.32. GetReadTime

関数名	UInt32 SetReadTime(uint time_an1)			
引数	引数名	IN/OUT	型	説明
	status	OUT	uint	インベントリ処理実行時の電波出力時間（10～40000、1＝1ms）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
AsReader のインベントリ処理実行時の電波出力時間を取得します。

サンプルコード：
GetReadTime(ref time_an1);

2.1.33. Get SdkVersion

関数名	void GetSdkVersion(ref string sdkVersion)			
引数	引数名	IN/OUT	型	説明
	sdkVersion	OUT	string	SDK バージョン
戻り値		OUT	void	実行成功：0 実行失敗：1
関数の説明： SDK バージョンを取得します。 **サンプルコード：** GetSdkVersion(ref sdkVersion);				

2.1.34. SetIdelTime

関数名	uint SetIdelTime(uint idelTime)			
引数	引数名	IN/OUT	型	説明
	idelTime	IN	uint	インベントリ処理実行時の電波出力の停止時間（ms）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： AsReader のインベントリ処理実行時の電波出力の停止時間を設定します。 **サンプルコード：** 電波出力の停止時間を 10ms に設定します。 SetIdelTime(10);				

2.1.35. GetIdelTime

関数名	uint GetIdelTime(ref uint idelTime)			
引数	引数名	IN/OUT	型	説明
	idelTime	OUT	uint	インベントリ処理実行時の電波出力の停止時間（ms）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： AsReader のインベントリ処理実行時の電波出力の停止時間を取得します。 **サンプルコード：** GetIdelTime(ref idelTime);				

2.1.36. DefaultSetting

関数名	bool DefaultSetting()			
引数	引数名	IN/OUT	型	説明
戻り値		OUT	bool	実行成功：true 実行失敗: false
関数の説明： デフォルト設定に戻します。 **サンプルコード：** DefaultSetting();				

2.1.37. SetHIDWorkParams

関数名	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)			
引数	引数名	IN/OUT	型	説明
	hid_adr	IN	int	タグのスタートアドレス
	hid_len	IN	int	タグの長さ
	hid_inventory	IN	int	インベントリの時間間隔(s)
	hid_filter_time	IN	int	連続して同じ EPC/TID データのタグを読み取りする時間間隔(s) output_without が NO_CHECKED の場合のみ有効
	repeat_epc_tid	IN	byte	連続して読み取ったデータが同じかどうかのチェック対象エリア（3.1.26 を参照） output_without が NO_CHECKED の場合のみ有効
	epc_tid_user	IN	byte	読み取り対象エリア（3.1.22 を参照）
	output_suffix	IN	byte	読み取りデータの後に付加する値（3.1.23 を参照）
	output_without	IN	byte	連続して同じ EPC/TID データのタグを読み取りするかどうか（3.1.24 を参照）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

HID モードのパラメータを設定します。

サンプルコード：

スタートアドレス：0

長さ：0

インベントリの時間間隔：0

連続して同じ EPC/TID データのタグを読み取りする時間間隔：0

連続して読み取ったデータが同じかどうかのチェック対象エリア：IS_CHECKED_EPC

読み取り対象エリア：EPC

読み取りデータの後に付加する値：NO_CHECKED 連続して同じ EPC/TID データのタグを読み取りするかどうか：IS_CHECKED SetHIDWorkParams(0,0,0,0,IS_CHECKED_EPC,EPC,NO_CHECKED,IS_CHECKED)

2.1.38. GetHIDWorkParams

関数名	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)			
引数	引数名	IN/OUT	型	説明
	hid_adr	OUT	int	タグのスタートアドレス
	hid_len	OUT	int	タグの長さ
	hid_inventory	OUT	int	インベントリの時間間隔
	hid_filter_time	OUT	int	連続して同じ EPC/TID データのタグを読み取りする時間間隔(s) output_without が NO_CHECKED の場合のみ有効
	repeat_epc_tid	OUT	byte	連続して読み取ったデータが同じかどうかのチェック対象エリア（3.1.26 を参照） output_without が NO_CHECKED の場合のみ有効
	epc_tid_user	OUT	byte	読み取り対象エリア（3.1.22 を参照）
	output_suffix	OUT	byte	読み取りデータの後に付加する値（3.1.23 を参照）
	output_without	OUT	byte	連続して同じ EPC/TID データのタグを読み取りするかどうか（3.1.24 を参照）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：

HID モードに設定したパラメータを取得します。

サンプルコード：

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

関数名	UInt32 SetBuzzer(Types.Buzzer buzzer)			
引数	引数名	IN/OUT	型	説明
	Buzzer	IN	Types.Buzzer	ブザーの設定値（3.1.25 を参照）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： ブザーを設定します。 **サンプルコード：** ブザーをオフに設定します。 SetBuzzer(OFF);				

2.1.40. GetBuzzer

関数名	UInt32 GetBuzzer(Types.Buzzer buzzer)			
引数	引数名	IN/OUT	型	説明
	Buzzer	OUT	Types.Buzzer	ブザーの設定値（3.1.25 を参照）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： ブザーの設定を取得します。 **サンプルコード：** GetBuzzer(ref buzzer);				

2.1.41. GetFwVersion

関数名	UInt32 GetFwVersion(ref string fwVersion)			
引数	引数名	IN/OUT	型	説明
	fwVersion	OUT	string	ファームウェアバージョン
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： ファームウェアバージョンを取得します。 **サンプルコード：** GetFwVersion(ref fwVersion);				

2.1.42. GetHwVersion

関数名	UInt32 GetHwVersion(ref string hwVersion)			
引数	引数名	IN/OUT	型	説明
	hwVersion	OUT	string	ハードウェアバージョン
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： ハードウェアバージョンを取得します。 **サンプルコード：** GetHwVersion(ref hwVersion);				

2.1.43. GetRFIDFwVersion

関数名	UInt32 GetRFIDFwVersion(ref string rfidFwVersion)			
引数	引数名	IN/OUT	型	説明
	rfidFwVersion	OUT	string	RFID モジュールのファームウェアバージョン
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： RFID モジュールのファームウェアバージョンを取得します。 **サンプルコード：** GetRFIDFwVersion(ref rfidFwVersion);				

2.1.44. GetProductSN

関数名	UInt32 GetProductSN(ref string productSN)			
引数	引数名	IN/OUT	型	説明
	productSN	OUT	string	AsReader のシリアル番号
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： AsReader のシリアル番号を取得します。 **サンプルコード：** GetProductSN(ref productSN);				

2.1.45. SetRSSIThreshold

関数名	UInt32 SetRSSIThreshold(int rssi_threshold)			
引数	引数名	IN/OUT	型	説明
	rssi_threshold	IN	int	RSSI の閾値（−99～0）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： RSSI の閾値を設定します。 **サンプルコード：** RSSI 閾値を−40 に設定します。 SetRSSIThreshold(40);				

2.1.46. GetRSSIThreshold

関数名	UInt32 GetRSSIThreshold(ref int rssi_threshold)			
引数	引数名	IN/OUT	型	説明
	rssi_threshold	OUT	int	RSSI の閾値（−99～0）
戻り値		OUT	uint	実行成功：0 実行失敗：1
関数の説明： 設定した RSSI の閾値を取得します。 **サンプルコード：** GetRSSIThreshold(ref rssi_threshold);				

2.1.47. SendCommand

関数名	bool SendCommand(byte[] command)			
引数	引数名	IN/OUT	型	説明
	command	IN	byte[]	コマンド送信
戻り値		OUT	uint	実行成功：0 実行失敗：1

関数の説明：

コマンドを送信します。

サンプルコード：

```
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

関数名	Void SetDelegate(CallBackReadTagData readTagData, CallBackErrorCode errorCode, CallBackSuccessCode successCode, CallBackCommandData commandData, CallBackReadComplete completeStatus, CallBackTriggerHandler triggerHandler)			
引数	引数名	IN/OUT	型	説明
	readTagData	IN	CallBackReadTagData	読取データのコールバック関数
	errorCode	IN	CallBackErrorCode	処理失敗時のコールバック関数
	successCode	IN	CallBackSuccessCode	処理成功時のコールバック関数
	commandData	IN	CallBackCommandData	共通データを受け取るコールバック関数
	completeStatus	IN	CallBackReadComplete	RF タグ読取のステータスを返すコールバック関数
	triggerHandler	IN	CallBackTriggerHandler	トリガーキーのステータスを返すコールバック関数
戻り値		OUT	uint	実行成功：0 実行失敗：1

関数の説明：
デリゲート関数を設定します。

サンプルコード：

```
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);

読取データを処理するデリゲートと実行にエラーが出る場合の出力用デリゲートを定義します。
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：付録Ⅰを参照
error ：3.1.16を参照
success： 3.1.17を参照
```

2.1.49. SetHIDInventoryMode

関数名	UInt32 SetHIDInventoryMode(Types.HIDInventoryMode mode)			
引数	引数名	IN/OUT	型	説明
	mode	IN	Types.HIDInventoryMode	HID モードのインベントリモード（3.1.27 を参照） Manual：P3xU をモバイルデバイスまたは PC に接続してから、P3xU の「Scan」キーを押下すると、インベントリが開始されます。 Auto：P3xU をモバイルデバイスまたは PC に接続すると、自動でインベントリが開始されます。
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
HID モードのインベントリモードを設定します。

サンプルコード：
```
Types.HIDInventoryMode mode = Types.HIDInventoryMode.Manual;
SetHIDInventoryMode (mode);
```

2.1.50. GetHIDInventoryMode

関数名	UInt32 GetHIDInventoryMode(ref Types.HIDInventoryMode mode)			
引数	引数名	IN/OUT	型	説明
	mode	IN	Types.HIDInventoryMode	HID モードのインベントリモード（3.1.27 を参照）
戻り値		OUT	UInt32	実行成功：0 実行失敗：1

関数の説明：
HID モードのインベントリモードを取得します。

サンプルコード：
```
Types.HIDInventoryMode mode = Types.HIDInventoryMode.Manual;
GetHIDInventoryMode (ref mode);
```

2.1.51. CheckTagStatus

関数名	TagStatus CheckTagStatus(byte[] epcData)			
引数	引数名	IN/OUT	型	説明
epcData	IN	byte[]	epcData	タグの EPC データ
戻り値		OUT	TagStatus	返したタグのステータス（3.1.28 を参照）

関数の説明：

RF タグのステータスを取得します。

サンプルコード：

例：EPC データは 01010202 の R F タグのステータスを取得します。

```
byte[] epcData = {0x01,0x01,0x02,0x02};
TagStatus tagStatus = CheckTagStatus(epcData);
```

3. Types クラス

Types クラスは Region、RFID モード、Session、サーチモード、Selection Mask のタグ Session、タグの Session 状態、メモリバンク及びその他のパラメータ設定を定義します。

3.1. 列挙

3.1.1. InventoryType

インベントリタイプ	パラメータ
PC_EPC_RSSI	1
PC_EPC_TID	2
ONLY_PC_EPC	3

3.1.2. RegionType

Region	パラメータ
REGION_US	0x21
REGION_US_Narrow	0x22
REGION_Europe	0x31
REGION_JAPAN	0x41
REGION_CHINA2	0x52
REGION_BRAZIL	0x61

3.1.3. SessionType

Session	パラメータ
Session_S0	0x0
Session_S1	0x1
Session_S2	0x2
Session_S3	0x3

3.1.4. TargetType

Selection Mask を適用するタグの Session。

Target	パラメータ
Session_S0	0x0
Session_S1	0x1
Session_S2	0x2
Session_S3	0x3
SL_FLAG	0x4

3.1.5. ActionType

タグがマークされた後のアクション。

Selection Mask 使う時に、Selection Mask にマッチングする場合・しない場合のタグの Session と Session Flag のアクション。

Action	パラメータ
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

3.1.6. MemBankType

Selection Mask で設定するタグのメモリバンク。

MemBank	パラメータ
MEM_RESERVED	0x0
MEM_EPC	0x1
MEM_TID	0x2
MEM_USER	0x3

3.1.7. ChannelType

RF チャンネル	パラメータ
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

3.1.8. GainType

PA ゲインモード	パラメータ
HIGH_GAIN	0x00
LOW_GAIN	0x01

3.1.9. TargetABType

Session フラッグ	パラメータ
TARGET_A	0x00
TARGET_B	0x01
TOGGLE_INVENTORY_ROUBD	0x02

3.1.10. DRType

分周比	パラメータ
DR_8	0x00
DR_64_3	0x01

3.1.11. MType

エンコードタイプ	パラメータ
M1	0x00
M2	0x01
M4	0x02
M8	0x03

3.1.12. TRextType

パイロットトーン付加するかどうか	パラメータ
NO_Pilot_Tone	0x00
Use_Pilot_Tone	0x01

3.1.13. SelType

クエリに応答するタグを選択	パラメータ
SEL_ALL	0x00
SEL_SL_N	0x02
SEL_SL	0x03

3.1.14. QType

Q 値	パラメータ
Q0	0
Q1	1
Q2	2
Q3	3
Q4	4
Q5	5
Q6	6
Q7	7
Q8	8

3.1.15. AntiCollisionMode

アンチコ Region	パラメータ
FixedQ	0x0
DynamicQ	0x01

3.1.16. ErrorCode

エラーコード	パラメータ
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

3.1.17. SuccessCode

成功コード	パラメータ
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
ISP_DOWNLOAD	0xB1
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

3.1.18. SelectionEnable

Select Mask 使用できるかどうか	パラメータ
DISABLE	0x00
ENABLE	0x01

3.1.19. FHType

FH	パラメータ
DISABLE	0x00
ENABLE	0x01
WITH_LBT	0x02

3.1.20. LBTType

LBT	パラメータ
DISABLE	0x00
ENABLE	0x01
WITH_LBT	0x02

3.1.21. CWType

CW	パラメータ
DISABLE	0x00
ENABLE	0x01

3.1.22. HidEpcTidUser

HID の稼動範囲	パラメータ
EPC	0x00
TID	0x01
USER	0x02

3.1.23. HidOutputSuffix

操作キー	パラメータ
NO_CHECKED	0x00
ENTER	0x01
TAB	0x02
BACKSPACE	0x03
COMMA	0x04

3.1.24. HidOutputWithout

同じでない EPC/TID データを重複に入力するかどうか	パラメータ
IS_CHECKED	0x01
IS_TID_CHECKED	0x02
NO_CHECKED	0x00

3.1.25. Buzzer

Buzzer	パラメータ
OFF	0x00
LOW	0x01
HIGH	0x02

3.1.26. HidRepeatEpcTid

同じ EPC/TID データを重複に入力するかどうか	パラメータ
IS_CHECKED_EPC	0x00
IS_CHECKED_TID	0x01

3.1.27. HIDInventoryMode

HID_Inventory_Mode	パラメータ
Manual	0x00
Auto	0x01

3.1.28. TagStatus

TagStatus	パラメータ
UnLock	0x00
Lock	0x01
PermaLock	0x02
Unknown	0x03
Error	0x04

付録Ⅰ

1. InventoryResult

タグをインベントリする時に表示される項目
rssi
channel
phase
antenna
TagData tagData

2. TagData

タグのデータ
pc
epc
tid
data

付録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 開発マニュアル

2024 年 11 月第二版

株式会社アスタリスク

〒532-0013 大阪府大阪市淀川区木川西 2 丁目 2-1 AsTech Osaka Building