DENSO
SE1-HU-P プロバイダ
ユーザーズ ガイド

Version 1.0.0

June 26, 2020

備考：

【改版履歴】

バージョン	日付	内容
1.0.0	2020-06-26	初版.

【動作確認機種】

機種	バージョン	注意事項
SE1HUP	1.00	

目次

1. はじめに

本書は，DENSO Wave の SE1HUP を使い，RFID タグを読み込み及び書き込みをするプロバイダのユーザーズガイドです．図 1-1 が本プロバイダとデバイスの全体構成図になります．以降本プロバイダを"SE1HUP プロバイダ"と呼称します．

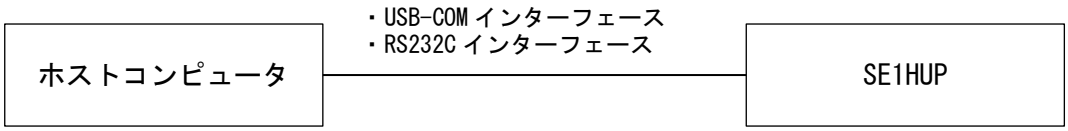


図 1-1 構成図

また，本プロバイダ及びデバイスそれぞれの対応を図 1-2に表します．
(※一例です．全てを表しているわけではありません．)

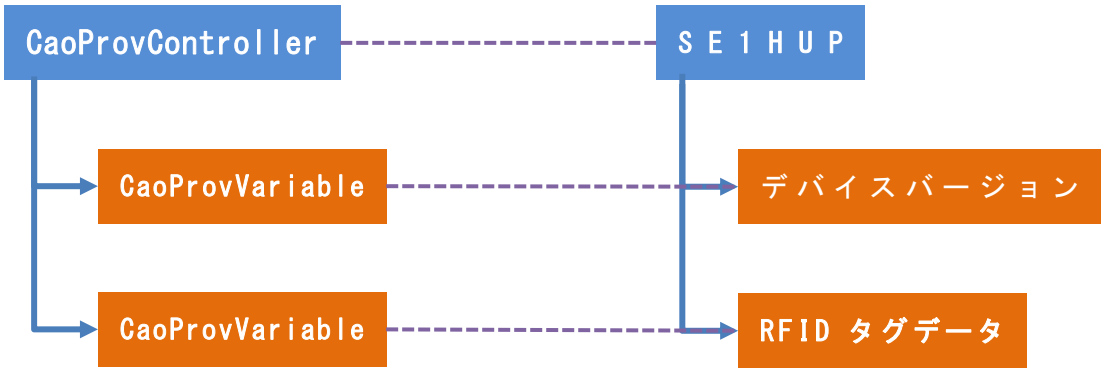


図 1-2 プロバイダの構成と SE1HUP との対応図

2. アプリケーション開発のための環境セットアップ

2.1. SE1HUP とクライアント PC との接続

SE1HUP とクライアント PC を接続するために USB 接続による仮想 COM 接続をする必要があります. 仮想 COM 接続を行うためには, DENSO WAVE のダウロードページから「ActiveUSBCOM」をクライアント PC にインストールする必要があります. インストール手順に関しては,「ActiveUSBCOM」をダウンロードした際に同梱されていますのでそちらを参照して下さい.

2.2. PC 開発環境のセットアップ

2.2.1. SE1HUP プロバイダの手動インストール

SE1HUP プロバイダを手動でインストールする場合は下記レジストリ登録を行う必要があります. レジストリ登録を行う場合は, 管理者権限でコマンドプロンプトを起動し, regsvr32 コマンドを実行してください. 実行する際には, ファイルのあるパスまで移動するか, ファイルパスを指定して実行してください.

表 2-1 SE1HUP プロバイダ

ファイル名	CaoProvDENSOSE1HUP.dll
ProgID	CaoProv.DENSO.SE1HUP
レジストリ登録	regsvr32 CaoProvDENSOSE1HUP.dll
レジストリ登録の抹消	regsvr32 /u CaoProvDENSOSE1HUP.dll

3. コマンドリファレンス

3.1. メソッド/プロパティー覧

表 3-1 メソッド/プロパティー覧

カテゴリ	メソッド/プロパティ[1]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.9
CaoController				
	VariableNames	M	接続可能な変数名リストの取得	P.10
	Variables	P	コントローラが保持する変数コレクションの取得	P.11
	AddVariable	M	変数オブジェクトの追加	P.11
	Execute	M	拡張コマンドの実行	P.11
	OnMessage	E	メッセージ受信イベント	P.19
CaoVariable				
	Value	P	値の取得/設定	P.19

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace クラス

3.2.1.1. AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．SE1HUP プロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当する SE1HUP と接続を行います．以下に，AddController メソッドの仕様を示します．

書式

```
AddController
(
        "<コントローラ名>",             // コントローラ名(任意)
        "CaoProv.DENSO.SE1HUP",         // プロバイダ名 (固定)
        "<マシン名>",                   // プロバイダ実行マシン名(未使用)
        "<オプション>"                  // オプション文字列(省略可能)
)
```

[1] M:メソッド，P:プロパティ，E:イベントをそれぞれ示します．

オプション

以下にオプション文字列に指定するオプションを示します．オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります．

オプション	必須	説明	値範囲
Conn=	○	SE1HUP と接続している COM ポート番号を指定します.	COM
Timeout=	--	コマンド送信から SE1HUP からデータ受信までのタイムアウト時間を指定します. デフォルト：2000	1-30000

使用例

```
// Engineオブジェクト
ORiN2.ManagedCAO.CCaoEngine engine = new ORiN2.ManagedCAO.CCaoEngine();
// Workspaceオブジェクト
ORiN2.ManagedCAO.CCaoWorkspace workspace = engine.AddWorkspace("NewWrks", "");
// Controllerオブジェクト
ORiN2.ManagedCAO.CCaoController controller = workspace.AddController("SE1HUP","CaoProv.DENSO.SE1HUP","","Conn = COM:1,Timeout = 5000");
```

3.2.1.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します．ここで角括弧("[]")内は省略可能なことを,各パラメータの解説中の下線部はオプションを指定しなかった時のデフォルト値をそれぞれ示します.

RS232C

"Conn=COM:<COM Port>[:BaudRate>[:<Parity>:<DataBits>:<StopBits>[:Flow]]]"

<COM Port> : COM ポート番号． '1' -COM1, '2' - COM2, ...

<BaudRate> : 通信速度． 4800, 9600, 19200, 38400, 57600, 115200

<Parity> : パリティ．'N'-NONE, 'E'-EVEN, 'O'-ODD

<DataBits> : データビット数．'7'-7bit, '8'-8bit

<StopBits> : ストップビット数．'1'-1bit, '2'-2bit

<Flow> : フロー制御．'0'-None, '1'-Xon/Xoff, '2'-ハードウェア制御 OR をとって指定できます.

3.2.2. CaoController クラス

3.2.2.1. VariableNames メソッド

接続可能な変数名リストを取得します．本メソッドで取得した変数名は，後述する AddVariable メソッドの第一引数に使用することができます.

使用例

```
    // 変数名リスト取得
```

```
String[] variableNames = controller.GetVariableNames("");
```

3.2.2.2. Variables プロパティ

コントローラが保持する，変数コレクションを取得します.

使用例

```
// 変数コレクション取得
ORiN2.ManagedCAO.CCaoVariables variables = controller.Variables;

// 変数取得
ORiN2.ManagedCAO.CCaoVariable variable = variables[0];
```

3.2.2.3. AddVariable メソッド

CaoController に変数オブジェクトを追加します.変数名には 3.3.1 に示すもののみ使用できます.
以下に，AddVariable の仕様を示します.

書式

```
AddVariable
(
        "<変数名>",             // 変数名
        "<オプション>"          // オプション文字列(省略可能)
)
```

3.2.2.4. Execute メソッド

CaoController の拡張コマンドを実行します. 以下に，Execute の仕様を示します.

書式

```
Execute
(
        "<拡張コマンド名>",             // 拡張コマンド名
        "<オプション文字列>"            // オプション文字列(省略可能)
)
```

以下に，Execute で指定できる拡張コマンド一覧を示します. 使用例は拡張コマンドの詳細で記述しています.

コマンド	説明	参照
SetRFIDReadTriggerWithBinary	バイナリ指定で RFID 読み込みモードをトリガーに設定します.	P.12
SetRFIDReadTrigerWithChar	Char 指定で RFID 読み込みモードをトリガーに設定します.	P.13

コマンド	説明	参照
ReadRFIDWithBinary	バイナリ指定で RFID タグを読み込みます.	P.13
ReadRFIDWithChar	Char 指定で RFID タグを読み込みます.	P.14
SetRFIDWriteTriggerWithBinary	バイナリ指定で RFID 書き込みモードをトリガーに設定します.	P.15
SetRFIDWriteTrigerWithChar	Char 指定で RFID 書き込みモードをトリガーに設定します.	P.15
WriteRFIDWithBinary	RFID タグにバイナリ指定で書き込みます.	P.16
WriteRFIDWithChar	RFID タグに Char 指定で書き込みます.	P.17
SetUIDReadTriger	UID 読み込みモードをトリガーに設定します.	P.17
ReadUID	UID を読み込みます.	P.17
ResetTrigger	トリガーモード設定をリセットします.	P.18
GetDeviceVersion	デバイスバージョンを取得します.	P.18

3.2.2.4.1. SetRFIDReadTriggerWithBinary コマンド

SE1HUP に対してバイナリ指定の読み込みコマンドをトリガースイッチモードに設定します. 設定したコマンドはトリガースイッチを押下することで実行され,実行結果は OnMessage イベントとして通知されます. イベント内容に関しては, SetRFIDReadTriggerWithBinary 実行結果を参照してください.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	読み込みをするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_BSTR	UID 情報を 16 文字の 16 進数文字で指定する. (省略可) 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

[2] 対象とする RFID によって指定できる範囲は異なります.

```
// SetRFIDReadTriggerWithBinary実行
object[] opt = new object[] { 0, 4, "0000000000000000" };

controller.Execute("SetRFIDReadTriggerWithBinary", opt);
```

3.2.2.4.2. SetRFIDReadTriggerWithChar コマンド

SE1HUP に対して ASCII 文字列指定の読み込みコマンドをトリガースイッチモードに設定します. 設定したコマンドはトリガースイッチを押下することで実行され,実行結果は OnMessage イベントとして通知されます. イベント内容に関しては, SetRFIDReadTriggerWithChar 実行結果を参照してください.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	読み込みをするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する. (省略可) 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

```
// SetRFIDReadTriggerWithChar実行
object[] opt = new object[] { 0, 4, "0000000000000000" };

controller.Execute("SetRFIDReadTriggerWithChar", opt);
```

3.2.2.4.3. ReadRFIDWithBinary コマンド

指定したアドレス位置から指定のデータ長 (Byte) 分のデータをバイナリで取得する.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	読み込みをするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400

項目	型説明		
	2	VT_BOOL	取得データ長が 1 の場合に VTARRAY で返却するか指定する. (省略可) TRUE : データ長が 1 の場合に戻り値の型を(VT_UI1 | VTARRAY)で返却する. FALSE:データ長が 1 の場合, 戻り値の型を (VT_UI1) で返却します. 省略時は"FALSE"がデフォルトで設定される.
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する. (省略可) 指定範囲 : "0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	VT_ARRAY | VT_UI		
	n	VT_UI1	指定したアドレスデータをバイナリで取得する.

使用例

```
// ReadRFIDWithBinary実行
object[] opt = new object[] { 0, 4, "0000000000000000" };

BYTE[] byte = controller.Execute("ReadRFIDWithBinary", opt) as byte[];
```

3.2.2.4.4. ReadRFIDWithChar コマンド

指定したアドレス位置から指定のデータ長 (Byte) 分のデータを ASCII 文字列で取得する.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	読み込みをするアドレスの開始位置を指定する. 指定範囲[2] : 1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2] : 1～400
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する. (省略可) 指定範囲 : "0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	VT_BSTR		指定したアドレスデータを ASCII 文字列で取得.

使用例

```
//ReadRFIDWithChar実行
object[] opt = new object[] { 0, 4, "0000000000000000" };

string strData = controller.Execute("ReadRFIDWithChar", opt) as string;
```

3.2.2.4.5. SetRFIDWriteTriggerWithBinary コマンド

SE1HUP に対してバイナリ指定の書き込みコマンドをトリガースイッチモードに設定します. 設定したコマンドはトリガースイッチを押下することで実行され, 実行結果は OnMessage イベントとして通知されます. イベント内容に関しては, SetRFIDWriteTriggerWithBinary 実行結果を参照してください.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	書き込みするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_ARRAY | VT_UI1	
		2.n VT_UI1	書き込みを行うデータをバイナリで指定する.
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する. (省略可) 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

```
// SetRFIDWriteTriggerWithBinary実行
byte[] writeData = new byte[] { 48, 49, 50, 51 }; //書き込むデータを作る
object[] opt = new object[] { 0, 4, writeData, "0000000000000000" };

controller.Execute("SetRFIDWriteTriggerWithBinary", opt);
```

3.2.2.4.6. SetRFIDWriteTriggerWithChar コマンド

SE1HUP に対してバイナリ指定の書き込みコマンドをトリガースイッチモードに設定します. 設定したコマンドはトリガースイッチを押下することで実行され, 実行結果は OnMessage イベントとして通知されます. イベント内容に関しては, SetRFIDWriteTriggerWithChar 実行結果を参照してください.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	書き込みするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)

項目	型説明		
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_BSTR	書き込みするデータを ASCII 文字列で指定する.
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する.（省略可） 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

```
// SetRFIDWriteTriggerWithChar実行
string writeData = "ABCD"; // 書き込むデータを作る
object[] opt = new object[] { 0, 4, writeData, "0000000000000000" };

controller.Execute("SetRFIDWriteTriggerWithChar", opt);
```

3.2.2.4.7. WriteRFIDWithBinary コマンド

指定したアドレス位置から指定のデータ長（Byte）分のデータをバイナリで書き込みをする.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	書き込みするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_ARRAY | VT_UI1	
	2.n	VT_UI1	書き込みを行うデータをバイナリで指定する.
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する.（省略可） 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

```
// WriteRFIDWithBinary実行
byte[] writeData = new byte[] { 48, 49, 50, 51 }; //書き込むデータを作る
object[] opt = new object[] { 0, 4, writeData, "0000000000000000" };

controller.Execute("WriteRFIDWithBinary", opt);
```

3.2.2.4.8. WriteRFIDWithChar コマンド

指定アドレス位置から RFID タグにデータを ASCII 文字列で書き込みをする.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	書き込みするアドレスの開始位置を指定する. 指定範囲[2]：1～999(1k バイトタグ) 0～1999(2k バイトタグ)
	1	VT_UI2	読み込むデータ長(Byte)を指定する. 指定範囲[2]：1～400
	2	VT_BSTR	書き込みするデータを ASCII 文字列で指定する.
	3	VT_BSTR	UID 情報を 16 文字の 16 進数文字列で指定する. (省略可) 指定範囲："0000000000000000"～"FFFFFFFFFFFFFFFF" 省略時は"0000000000000000"がデフォルトで設定される.
戻り値	なし		

使用例

```
// WriteRFIDWithChar実行
string writeData = "ABCD"; // 書き込むデータを作る
object[] opt = new object[] { 0, 4, writeData, "0000000000000000" };

controller.Execute("WritRFIDWithChar", opt);
```

3.2.2.4.9. SetUIDReadTrigger コマンド

SE1HUP に対して UID 読み込みコマンドをトリガースイッチモードに設定します. 設定したコマンドはトリガースイッチを押下することで実行され, 実行結果は OnMessage イベントとして通知されます. イベント内容に関しては, SetUIDReadTrigger 実行結果を参照してください.

項目	型説明
引数	なし
戻り値	なし

使用例

```
// SetUIDReadTrigger実行
controller.Execute("SetUIDReadTrigger", opt);
```

3.2.2.4.10. ReadUID コマンド

範囲内の RFID タグの UID データを取得します.

項目	型説明	
引数	なし	
戻り値	VT_BSTR | ARRAY or VT_BSTR	
	n	RFID タグの UID 情報を取得.

使用例

```
// ReadUID実行
object objUID = controller.Execute("ReadUID", opt);
// 取得データが配列か確認
if(objUID.GetType().IsArray)
{
    string[] strDatas = objUID as string[]; //stringの配列に変換
}
else
{
    String strData = objUID.ToString(); //stringに変換
}
```

3.2.2.4.11. ResetTrigger コマンド

SE1HUP をトリガースイッチモードに設定中の場合トリガースイッチモードを解除します.

項目	型説明
引数	なし
戻り値	なし

使用例

```
// ResetTrigger実行
controller.Execute("SetUIDReadTrigger" ,null); //トリガースイッチモードを設定する

controller.Execute("ResetTrigger" ,null); //設定したトリガースイッチモードを解除する
```

3.2.2.4.12. GetDeviceVersion コマンド

SE1HUP のデバイスバージョンを取得します.

項目	型説明	
引数	なし	
戻り値	VT_BSTR	デバイスバージョンを取得します.

使用例

```
' GetDeviceVersion実行
string strDeviceVersion = controller.Execute("GetDeviceVersion", null) as string;
```

3.2.2.5. OnMessage イベント

トリガースイッチモード中の受信結果を OnMessage イベントとして受け取ることが可能です. 受け取れるイベントについては 3.4 を参照してください.

使用例

```
// コントローラにメッセージイベントを登録
private void AddOnMessage()
{
    controller.OnMessage += new OnMesssageEventHandler(OnMessage);
}

// メッセージ受信イベント本体
private void OnMessage(object obj,OnMessageEventArgs arg)
{
    // メッセージ内容を取得
    object message = arg.Message.Value;
}
```

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

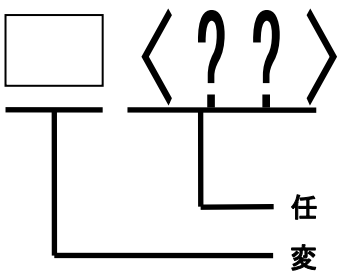
接続した SE1HUP からデータを取得/設定します. 変数名によって動作が異なります. 詳細は, 3.3. 変数一覧を参照してください.

3.3. 変数一覧

各クラスで使用可能な変数一覧を定義します. なお変数は, CaoVariable クラスのオブジェクトを指します. 複数変数を登録（オプションのみ変更したい場合等に有用）するために任意の文字列を付与することが可能です.

変数名に任意文字列を付与するための書式を以下に示します.

複数変数共通指定書式



3.3.1. CaoController クラス変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します.	○	-	P.20
@VERSION	DLL バージョンを取得します.	○	-	P.20
@DEVICE_VERSION	SE1HUP のデバイスバージョンを取得します.	○	○	P.20
@UID	範囲内の UID 情報を取得します.	○	-	P.21
RFID_ADDRESS_BINARY<??>	RFID タグデータをバイナリ指定で取得設定します.	○	○	P.21
RFID_ADDRESS_CHAR<??>	RFID タグデータを文字列指定で取得設定します.	○	○	P.22

3.3.1.1. @MAKER_NAME

メーカー名の取得をします.

データ型

型説明	
VT_BSTR	メーカー名を取得します.

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@MAKER_NAME","");
// 値取得
string value = var.Value.ToString();
```

3.3.1.2. @VERSION

DLL のバージョンの取得をします.

データ型

型説明	
VT_BSTR	DLL のバージョンを"*.*.*"形式で取得します.

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@VERSION","");
// 値取得
string value = var.Value.ToString();
```

3.3.1.3. @DEVICE_VERSION

DLL のバージョンの取得をします.

データ型

型説明	
VT_BSTR	SE1HUP のデバイスバージョンを"*.**"形式で取得します

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@DEVICE_VERSION","");
// 値取得
string value = var.Value.ToString();
```

3.3.1.4. @UID

DLL のバージョンの取得をします.

データ型

型説明	
VT_BSTR | ARRAY or VT_BSTR	
n	RFID タグの UID 情報を取得.

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var;
var = controller.AddVariable("@VERSION","");
// 値取得
object value = var.Value;
```

3.3.1.5. RFID_ADDRESS_BINARY<??>

RFID タグのデータをバイナリ指定で取得/設定します.RFID_ADDRESS_BINARY の後に任意の文字列を入力して変数名を指定してください.

オプション

オプション	必須	説明	値範囲 [2]
StartPos=	○	アドレスの開始位置を指定します.	1～999(1k バイトタグ) 0～1999(2k バイトタグ)
Elem=	○	アドレス開始位置から読み込む要素数を指定します.	1~400
UID=	--	UID情報を16文字の16進数文字列で指定する.(省略可) "0000000000000000"を指定された場合, 任意のタグからデータを取得設定します.	16 桁の 16 進数文字列 デフォルト値 : "0000000000000000"

データ型

型説明		
VT_ARRAY | VT_UI		
n	VT_UI1	指定したアドレスデータに対し，バイナリ配列で取得設定します.

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var;
var = controller.AddVariable("RFID_ADDRESS_BINARY","StartPos=0,Elem=2");
// 値取得
byte[] value = var.Value as byte[];
// 先頭アドレスに A の文字を入れる
byte[0] = 'A';
// 値設定
var Value = byte;
```

3.3.1.6. RFID_ADDRESS_CHAR<??>

RFID タグのデータをバイナリ指定で取得/設定します.RFID_ADDRESS_BINARY の後に任意の文字列を入力して変数名を指定してください.

オプション

オプション	必須	説明	値範囲 [2]
StartPos=	○	アドレスの開始位置を指定します.	1～999(1k バイトタグ) 0～1999(2k バイトタグ)
Elem=	○	アドレス開始位置から読み込む要素数を指定します.	1~400
UID=	--	UID情報を16文字の16進数文字列で指定する.(省略可) "0000000000000000"を指定された場合，任意のタグからデータを取得設定します.	16 桁の 16 進数文字列 デフォルト値： "0000000000000000"

データ型

型説明	
VT_BSTR	指定したアドレスデータに対し，ASCII 文字列で取得設定します.

使用例

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var;
var = controller.AddVariable("RFID_ADDRESS_CHAR","StartPos=0,Elem=2");
// 値取得
```

```
    string data = var.Value.ToString();
    // AB の文字列を設定する
    data = "AB";
    // 値の設定
    var.Value = data;
```

3.4. イベント一覧

以下に，OnMessage で受信できるトリガースイッチモード中の結果の一覧を示します.

メッセージ番号	説明	参照
0	SetRFIDReadTriggerWithBinary で設定したコマンドの実行結果	P.23
1	SetRFIDReadTriggerWithChar で設定したコマンドの実行結果	P.23
2	SetRFIDWriteTriggerWithBinary で設定したコマンドの実行結果	P.23
3	SetRFIDWriteTriggerWithChar で設定したコマンドの実行結果	P.24
4	SetUIDReadTrigger で設定したコマンドの実行結果	P.24

SE1HUP からの受信データの解析結果がエラーの場合は，メッセージ内容にエラーが格納されてイベントが通知されます. エラー内容に関しては，5.5 を参照ください.

3.4.1. SetRFIDReadTriggerWithBinary 実行結果

SetRFIDReadTriggerWithBinary コマンド実行後,SE1HUP 側でスイッチ押下した際の結果を通知します.

データ型

メッセージ内容		
VT_ARRAY | VT_UI1		
n	VT_UI1	受信した RFID タグのデータをバイナリで取得します.

3.4.2. SetRFIDReadTriggerWithChar 実行結果

SetRFIDReadTriggerWithChar コマンド実行後,SE1HUP 側でスイッチ押下した際の結果を通知します.

データ型

メッセージ内容	
VT_BSTR	受信した RFID タグのデータを ASCII 文字列で取得します.

3.4.3. SetRFIDWriteTriggerWithBinary 実行結果

SetRFIDWriteTriggerWithBinary コマンド実行後，SE1HUP 側でスイッチ押下した際の結果を通知します.

データ型

メッセージ内容
なし

3.4.4. SetRFIDWriteTriggerWithChar 実行結果

SetRFIDWriteTriggerWithChar コマンド実行後, SE1HUP 側でスイッチ押下した際の結果を通知します.

データ型

メッセージ内容
なし

3.4.5. SetUIDReadTrigger 実行結果

SetUIDReadTrigger コマンド実行後, SE1HUP 側でスイッチ押下した際の結果を通知します.

データ型

メッセージ内容		
VT_ARRAY | VT_BSTR or VT_BSTR		
n	VT_BSTR	受信した UID データを取得します.

4. SE1HUP プロバイダによるサンプルプログラミング

SE1HUP プロバイダでは，以下の手順でクライアント PC と SE1HUP を接続することができます.

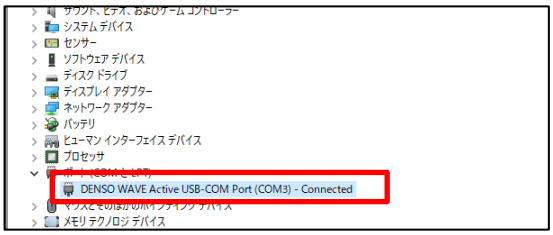
- CaoEngine の作成
- CaoWorkspace の作成
- CaoController の作成

SE1HUP に接続した後は，CaoController の Execute メソッドを使用する，もしくは，CaoVariabe オブジェクトを生成することで，SE1HUP の情報の取得や RFID タグに対してデータの取得設定を行うことができます.

4.1. UID 情報をタグデータの先頭アドレスに設定するサンプルプログラミング

ここでは例として最初に読み込んだ RFID タグの UID 情報を先頭アドレスに設定します. 表 4-1 にサンプルプログラムの要件を，図 4-1 にサンプルプログラムの流れをそれぞれ記述しています.

表 4-1 サンプルプログラムの要件

要件	説明
接続先	COM 接続
	接続先：DENSO WAVE Active USB-COM Port(ActiveUSBCOM) ※デバイスマネージャを開き,ActiveUSBCOM が割り当てられている COM ポート番号を指定します.（下図の赤枠が ActiveUSBCOM となる） 
処理内容	RFID タグの UID 情報を取得する.
	RFID タグの先頭 16 文字のデータを取得する.
	先頭アドレスから UID 情報を設定する.
	先頭アドレスから 16 文字のデータを取得し確認する.

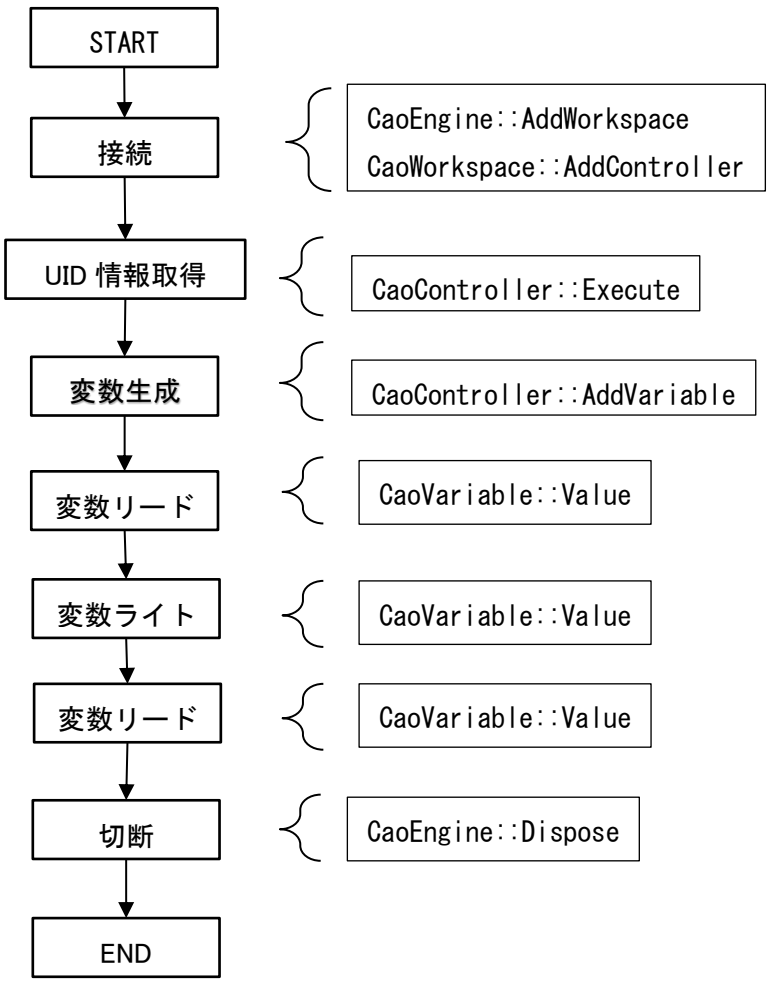


図 4-1 処理の流れ

以降の節から具体的なコードを示します.

4.1.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample SE1HUPSample.cs

```
// オブジェクト
private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
private ORiN2.ManagedCAO.CCaoController m_caoController = null;
private ORiN2.ManagedCAO.CCaoVariable m_rfidTagData = null;

public void Main()
{
    // 接続
    this.Connect();
    try
    {
```

```
            // UID情報を読み込む
            string uid = ReadUID();
            // UID情報を取得出来たら処理の実行
            if (!string.IsNullOrEmpty(uid))
            {
                // 先頭アドレスからUID情報分のデータサイズを指定したオプションの作成する
                string option = "StartPos=0,elem="+uid.Length.ToString()+",UID="+ uid;
                // UID情報を指定した変数を作成する
                this.m_rfidTagData = this.m_caoController.AddVariable("RFID_ADDRESS_CHAR",
option);
                 // データ書き込み前のデータを取得
                 string beforData = this.m_rfidTagData.Value.ToString();
                 // データ書き込み
                 this.m_rfidTagData.Value = uid;
                 // 書き込み後のデータを取得する
                 string afterData = this.m_rfidTagData.Value.ToString();

                 string DispString = "UID情報 = " + uid + Environment.NewLine + "書き込み前
 = " + beforData + Environment.NewLine + "書き込み後 = " + afterData;
                 // コンソールに結果を表示
                 Console.WriteLine(DispString);
             }
        }
        catch(Exception ex)
        {
              // エラー内容を結果を表示
              Console.WriteLine(ex.Message.ToString());
        }
        finally
        {
             // 切断
             this.Disconnect();
        }
}

// 接続メソッド
private void Connect()
{
    // CaoEngineオブジェクトの生成
    this.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
    // CaoWorkspaceオブジェクトの生成
    this.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
    // CaoControllerオブジェクトの生成
    this.m_caoController = this.m_caoWorkspace.AddController("SE1HUP","CaoProv.DENSO.SE1H
UP", "","Conn=COM:3,Timeout=2000");
    // OnMessageイベントを登録
    this.m_caoController.OnMessage += new OnMesssageEventHandler(OnMessage);
}
```

```
// 切断メソッド
private void Disconnect()
{
    this.m_caoEngine.Dispose();
    this.m_caoEngine = null;
}

// UID情報の取得
private string ReadUID()
{
     return this.m_caoController.Execute("ReadUID", null).ToString();
}
```

4.1.1.1. 接続

SE1HUP と接続するためには，以下の手順を取ります．

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngineオブジェクトとCaoWorkspaceオブジェクトとCaoControllerオブジェクトです．CaoWorkpaceオブジェクトは，CaoControllerオブジェクトをCaoWorkspacesから取得する場合には変数を用意する必要はありません．また変数にアクセスするためのCaoVariableオブジェクトも必要になります．以下にC#でのコード例を示します．

```
// CaoEngine オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
// CaoWorkspace オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
// CaoController オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoController m_caoController = null;
// CaoVariable オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoVariable m_rfidTagData = null;
```

(2) CaoEngineオブジェクトを生成します．CaoEngineオブジェクトはNewキーワードを使って生成します．

```
// CaoEngine オブジェクトの生成
this.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します．CaoEngineオブジェクトを生成すると，デフォルトでCaoWorkspacesオブジェクトとCaoWorkspaceオブジェクトを1つずつ生成しています．以下にCaoWorkspaceオブジェクトを新しく生成するコード例とデフォルトのCaoWorkspaceを示します．

```
    // CaoWorkspace オブジェクトの生成
    this.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
```

(4) CaoControllerオブジェクトを生成します. CaoControllerオブジェクトを生成するには, 使用するプロバイダ名と使用するためのパラメータを設定します. SE1HUPプロバイダでは, Connをオプションを COMで指定します. 以下にコード例を示します.

```
    // CaoController オブジェクトの生成
this.m_caoController = this.m_caoWorkspace.AddController("SE1HUP", "CaoProv.DENSO.SE1HUP",
                                                          "", "Conn=COM:3,Timeout=2000");
```

4.1.1.2. UID 情報を先頭アドレスから書き込む

UID 情報を取得してから先頭アドレスに書き込むためには以下の手順を取ります.

(1) UID 情報の取得するために CaoController::Execute メソッドを使います. Execute で取得したデータは object 型で取得されているため ToString()を使い string 型に変換します.

```
// Execute の実行
this.m_caoController.Execute("ReadUID", null).ToString();
```

(2) 取得した UID 情報を使い RFID_ADDRESS_CHAR<??>変数を生成します. 先頭アドレスを指定するため StartPos オプションに"0"を指定し, Elem オプションには取得した UID 情報の文字数, UID オプションには, UID 文字列を指定します.

```
// 先頭アドレスから UID 情報分のデータサイズを指定したオプションの作成する
string option = "StartPos=0,elem="+uid.Length.ToString()+",UID="+ uid;
// UID 情報を指定した Variable 変数を生成する
this.m_rfidTagData = this.m_caoController.AddVariable("RFID_ADDRESS_CHAR",
```

(3) 生成した Variable 変数を使用し, 指定したアドレス位置に UID 情報の書き込みを行います. Value プロパティを使い, 書き込み前のデータを一度取得します. その後 UID 情報を設定し, 最後に正常に書き込まれたことを確認するためにデータの取得をします.

```
// データ書き込み前のデータを取得
string beforData = this.m_rfidTagData.Value.ToString();
// データ書き込み
this.m_rfidTagData.Value = uid;
// 書き込み後のデータを取得する
string afterData = this.m_rfidTagData.Value.ToString();
```

4.1.1.3. 切断

コントローラと切断する場合には, 生成したオブジェクトを消去すると共に, オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します. ただし, ORiN.ManagedCAO を使用した場合は明示的に削除する必要はありません. 以下にコード例を示します.

```
    // CaoEngine からすべてのオブジェクトを削除
    this.m_caoEngine.Dispose();
    // CaoEngine の消去
    this.m_caoEngine = null;
```

5. SE1HUP プロバイダエラーコード

本プロバイダには，メソッド，プロパティ，イベントごとに独自エラーコードが存在します．
ORiN2 の共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

5.1. CaoWorkspace::AddController 時のエラーコード一覧

表 5-1 AddController 時のエラーコード表

エラー番号	説明
0x80111101	Connオプション指定がありません．
0x80111102	Connオプションの指定方法が間違っています．
0x80111103	Connオプションに有効範囲外の値が設定されています．
0x80111202	Timeoutオプションの指定方法が間違っています．
0x80111203	Timeoutオプションに有効範囲外の値が設定されています．

5.2. CaoController;;Execute 時のエラーコード一覧

表 5-2 CaoController::Execute 時のエラーコード表

エラー番号	説明
0x80121001	指定オプションの引数が足りていません．実行するコマンドのオプション引数をご確認ください．
0x80121005	想定外のデータを受信しました．通信環境等の見直しをしてください．
0x80121006	トリガーモード中にExecuteが実行されました．SE1HUPでトリガースイッチを押下してコマンドを実行するかRisetTriggerコマンドを実行してください．
0x80121102	第一引数のオプション指定方法が間違っています．実行するコマンドの第一引数のオプションをご確認ください．
0x80121103	第一引数のオプションに有効範囲外の値が設定されています．実行するコマンドの第一引数のオプションをご確認ください．
0x80121202	第二引数のオプション指定方法が間違っています．実行するコマンドの第二引数のオプションをご確認ください．
0x80121203	第二引数のオプションに有効範囲外の値が設定されています．実行するコマンドの第二引数のオプションをご確認ください．
0x80121302	第三引数のオプション指定方法が間違っています．実行するコマンドの第三引数のオプションをご確認ください．
0x80121303	第三引数のオプションに有効範囲外の値が設定されています．実行するコマンドの第三引数のオプションをご確認ください．
0x80120092	SE1HUPからRFIDタグ処理エラーが返却されました．

<table>
<tr><th>エラー番号</th><th>説明</th></tr>
<tr><td></td><td>※RFIDタグ処理エラーとは以下の内容があります. SE1HUPの有効エリアにRFIDタグが存在するかご確認ください.
<table>
<tr><th>RFIDタグエラー項目</th></tr>
<tr><td>タグからの応答がない</td></tr>
<tr><td>タグからの受信エラー</td></tr>
<tr><td>タグ送信時のエラー</td></tr>
<tr><td>タグからのエラー応答受信</td></tr>
<tr><td>同一UIDのタグがエリア内に存在する</td></tr>
</table></td></tr>
<tr><td>0x80120093</td><td>SE1HUPから読み取りデータ範囲外エラー返却されました.
※文字列指定のコマンド時に読み込むデータが0x20~0x7Eの範囲外の可能性があります. バイナリ指定でご確認ください.</td></tr>
<tr><td>0x80120280</td><td>SE1HUPのコマンドパラメータに設定範囲外の値が設定されました.</td></tr>
</table>

5.3. CaoController::AddVariable 時のエラーコード一覧

表 5-3 Controller::AddVariable 時のエラーコード表

エラー番号	説明
0x80131001	指定オプション数が足りていません. 追加する変数のオプション引数をご確認ください.
0x80131102	第一オプションの指定方法が間違っています. 追加する変数のオプションをご確認ください.
0x80131103	第一オプションに有効範囲外の値が設定されています. 実行するコマンドの第一オプションをご確認ください.
0x80131108	必須の第一オプションが指定されていません.
0x80131202	第二オプションの指定方法が間違っています. 実行するコマンドの第二引数のオプションをご確認ください.
0x80131203	第二オプションに有効範囲外の値が設定されています. 実行するコマンドの第二引数のオプションをご確認ください.
0x80131208	必須の第二オプションが指定されていません.
0x80131302	第三オプションの指定方法が間違っています. 実行するコマンドの第三引数のオプションをご確認ください.
0x80131303	第三オプションに有効範囲外の値が設定されています. 実行するコマンドの第三オプションをご確認ください.

5.4. CaoVariable::Value プロパティのエラーコード一覧

表 5-4 CaoVariable::Value プロパティのエラーコード表

エラー番号	説明
0x80141005	想定外のデータを受信しました. 通信環境等の見直しをしてください.
0x80141006	トリガーモード中にValueプロパティが使用されました. SE1HUPでトリガースイッチを押下してコマンドを実行するかRisetTriggerコマンドを実行してください.
0x80141303	RFID_ADDRESS_BINARY<??>またはRFID_ADDRESS_CHAR<??>の設定値がAddVariable時に設定したElemオプションと一致していません.
0x80140092	SE1HUPからRFIDタグ処理エラーが返却されました. ※RFIDタグ処理エラーとは以下の内容があります. SE1HUPの有効エリアにRFIDタグが存在するかご確認ください. RFIDタグエラー項目 タグからの応答がない タグからの受信エラー タグ送信時のエラー タグからのエラー応答受信 同一UIDのタグがエリア内に存在する
0x80140093	SE1HUPから読み取りデータ範囲外エラー返却されました. ※文字列指定のコマンド時に読み込むデータが0x20~0x7Eの範囲外の可能性があります. バイナリ指定でご確認ください.
0x80140280	SE1HUPのコマンドパラメータに設定範囲外の値が設定されました.

5.5. CaoController::OnMessage イベント時のエラーコード一覧

エラー番号	説明
0x80151007	想定外のデータを受信しました. 通信環境等の見直しをしてください.
0x80150092	SE1HUPからRFIDタグ処理エラーが返却されました. ※RFIDタグ処理エラーとは以下の内容があります. SE1HUPの有効エリアにRFIDタグが存在するかご確認ください. RFIDタグエラー項目 タグからの応答がない タグからの受信エラー タグ送信時のエラー タグからのエラー応答受信 同一UIDのタグがエリア内に存在する

エラー番号	説明
0x80150093	SE1HUPから読み取りデータ範囲外エラー返却されました. ※文字列指定のコマンド時に読み込むデータが0x20~0x7Eの範囲外の可能性があります. バイナリ指定でご確認ください.

付録A. 通信プロトコルコマンド対応表

CaoControllre::Execute メソッド	SE1HUP 通信プロトコルコマンド
SetRFIDReadTriggerWithBinary SetRFIDReadTriggerWithChar ReadRFIDWithBinary ReadRFIDWithChar	RFRO
SetRFIDWriteTriggerWithBinary SetRFIDWriteTriggerWithChar WriteRFIDWithBinary WriteRFIDWithChar	RFWO
SetUIDReadTrigger ReadUID	RFIO
ResetTrigger	RFSO
GetDeviceVersion	VERO
CaoVariable::Value プロパティ	SE1HUP 通信プロトコル
@DEVICE_VERSION::get	VERO
@UID::get	RFIO
RFID_ADDRESS_BINARY::get RFID_ADDRESS_CHAR::get	RFRO
RFID_ADDRESS_BINARY::set RFID_ADDRESS_CHAR::set	RFWO