DeviceNet プロバイダ

OMRON　デバイスネット PCI ボード

Version 1.1.0

ユーザーズ　ガイド

July 10, 2013

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2006-02-24	初版.
1.1.0.0	2008-07-10	リトライオプション追加.
1.1.0.1	2010-02-12	エラーコード追加
1.1.0.2	2011-03-11	プロバイダ登録ツールに関する追記
1.1.0	2012-07-17	ドキュメントのバージョンルールを変更
	2013-07-10	誤植を修正

【対応機器】

機種	バージョン	注意事項

目次

1. はじめに

本書は，OMRON 製 DeviceNet PCI ボードにアクセスするためのプロバイダである，DeviceNet プロバイダのユーザーズガイドです．

詳細については，OMRON 社 DeviceNet PCI ボードスキャナのユーザーズマニュアルを参照してください．

> 注意：DeviceNet プロバイダを使用するには，DeviceNet PCI ボードのドライバをインストールしなければなりません．ドライバインストール後にプロバイダをレジストリ登録する必要があります．レジストリ登録の方法は表 2-1 を参照してください．

2. プロバイダの概要

2.1. 概要

DeviceNet プロバイダのファイル形式は DLL(Dynamic Link Library)となっており, その詳細は表 2-1 の通りです.

表 2-1 DeviceNet プロバイダ

ファイル名	CaoProvDNet.dll
ProgID	CaoProv.OMRON.DNet
レジストリ登録[1]	regsvr32 CaoProvDNet.dll
レジストリ登録の抹消	regsvr32 /u CaoProvDNet.dll

[1] プロバイダの登録は regsvr32.exe または RegCOM.exe ([スタート]→[ORiN2]→[Tools])で実行できます. DeviceNet PCI ボードのドライバをインストールしていないと, DeviceNet プロバイダの登録はできません.

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

DeviceNet プロバイダでは Controller オブジェクトの生成時に DeviceNet ボードとの接続処理を行います. 接続時にオプション文字列でデバイスを指定します.

```
AddController
(
    "<コントローラ名>",          // コントローラ名.
    " CaoProv.OMRON.DNet",      // プロバイダ名. 固定.
    "<マシン名>",                // プロバイダの実行マシン名.
    "<オプション>"               // オプション文字列.
)
```

指定できるオプションの一覧を示します. また, 指定できるオプションの種類は, マスタ/スレーブの指定によって変わります. マスタ/スレーブの指定は Function オプションで指定してください.

表 2-2 AddController のオプション文字列

オプション	意味	マスタ	スレーブ
Function=<0｜0 以外>	マスタ/スレーブの指定(必須) 0:マスタ 0 以外:スレーブ	○	○
BoardId=<BoardId>	接続先ボード番号の指定(必須)	○	○
Online=<MacId>:<BaudRate>	オンライン情報の指定(必須) MacId:ノードアドレス BaudRate:通信速度(kbps) 0:125kbps 1:250kbps 2:500kbps	○	○
Retry[=<リトライ回数>]	接続時リトライ回数の指定(デフォルト:0)	○	○
SlaveDevice[=<設定文字列>]	スキャンリスト登録時情報の指定. 詳細は 2.2.1.1 を参照して下さい.	○	○
SlaveList[=<FilePath>]	スキャンリスト複数/詳細情報ファイルの指定. FilePath:ファイルパス スレーブを複数, もしくは詳細に設定したい場合に使用します.	○	-

SlaveCSV[=<FilePath>]	スキャンリスト複数情報 CSV ファイルの指定. FilePath:ファイルパス スレーブを複数設定したい場合に使用します. 1行につき1スレーブの情報を記述します. 各スレーブ情報は以下のように指定します. <OutSize>,<InSize>,<MacID>	○	-
ErrorOutData =<True|False>]	エラー発生時の処理選択 True:通信異常時停止 False:停止しない(デフォルト)	○	-

2.2.1.1. SlaveDevice オプション詳細

以下にマスタ/スレーブ時, SlaveDevice オプション文字列に指定するリストを示します.

・ マスタ時

SlaveDevice=<OutSize>:<InSize>:<MacId>

表 2-3 マスタ時の SlaveDevice オプション

オプションメンバ	意味
OutSize	出力データサイズ(Byte)(必須)
InSize	入力データサイズ(Byte)(必須)
MacId	ノードアドレス(必須)

・ スレーブ時

SlaveDevice=<OutSize1>:<InSize1>:
[<OutSize2>]:[<InSize2>]:
[<ScanType>]:[<ConnectAccept>]:[<ErrorOutData>]
スキャンリスト登録時情報の指定(必須)

各メンバの詳細は表 2-4 を参照してください.

表 2-4 スレーブ時の SlaveDevice オプション

オプションメンバ	意味
OutSize1	出力データ 1 サイズ(Byte)(必須)
InSize1	入力データ 1 サイズ(Byte)(必須)
OutSize2	出力データ 2 サイズ(Byte)(デフォルト:0)
InSize2	入力データ 2 サイズ(Byte) (デフォルト:0)

ScanType	スキャンタイプの指定. (デフォルト:32768) 32768(0x8000)を指定すると自動選択します.
ConnectAccept	接続状態の指定. 非接続:65535 接続:上記以外(デフォルト)
ErrorOutData	通信異常時出力データ処理の指定. 出力データ保持:65535 クリア:上記以外(デフォルト)

2.2.2. CaoController::AddCommand メソッド

使用できるコマンド名と詳細は表 2-5 を参照してください.

指定するオプション文字列はありません.

```
AddCommand
(
    "<コマンド名>",                    // コマンド名.
    "<オプション>"                     // オプション文字列. (未使用)
)
```

表 2-5 コマンドクラス 予約語一覧

コマンド予約語	パラメータデータ型	パラメータ設定	マスタ	スレーブ
ISEXISTCARD	VT_I2	<ボード番号>	○	○
RESET	なし	なし	○	○
REGDEVICE	VT_ARRAY | VT_I2	<出力データサイズ>, <入力データサイズ> <ノードアドレス>,	○	-
REMOVEDEVICE	VT_I2	<ノードアドレス>	○	-
SETSCANLIST	VT_BSTR	<ファイルパス>	○	-
CLEARSCANLIST	なし	なし	○	-
STORESCANLIST	なし	なし	○	○
LOADSCANLIST	なし	なし	○	○
STORESCANTIME	なし	なし	○	-
LOADSCANTIME	なし	なし	○	-
CONNECTDEVICE	VT_I2	<ノードアドレス>	○	-
DISCONNECTDEVICE	VT_I2	<ノードアドレス>	○	-
IOREFRESH	なし	なし	○	○

SENDCOS	VT_I2	<ノードアドレス>	○	○

2.2.3. CaoController::AddVariable メソッド

使用できるシステム変数名と詳細は 2.3.1 を参照してください.

```
AddVariable
(
    "<変数名>",                    // 変数名.
    "<オプション>"                 // オプション文字列. (未使用)
)
```

2.2.4. CaoController::get_CommandNames プロパティ

コマンドクラスで使用できる予約語リストを出力します.

指定するオプションはありません.

詳しくは表 2-5 を参照してください.

2.2.5. CaoController::get_VariableNames プロパティ

変数クラスで使用できるシステム変数名リストを出力します.

指定するオプションはありません.

詳しくは 2.3.1 を参照してください.

2.2.6. CaoCommand::put_Parameterプロパティ

パラメータの設定を行います.

設定できるパラメータについては表 2-5 を参照してください.

2.2.7. CaoCommand::get_Parameter プロパティ

パラメータの設定を行います.

設定できるパラメータについては表 2-5 を参照してください.

2.2.8. CaoController::get_Value プロパティ

各システム変数名に対応した値取得を行います.

詳細は 2.3.1 を参照して下さい.

2.2.9. CaoController::put_Value プロパティ

各システム変数名に対応した値設定を行います.

詳細は 2.3.1 を参照して下さい.

2.3. 変数一覧

2.3.1. コントローラクラス

表 2-6 コントローラクラス マスタ指定時ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
IN<ノード番号>	VT_ARRAY | VT_VARIANT（VTARRAY | VT_UI1）	スレーブ入力データの読み出しを行います. 変数オブジェクトの ID プロパティの値によって取得データが変わります. ID=0 : 要素 0 に IN1 の値を取得 (VT_ARRAY | VT_UI1) 要素 1 に IN2 の値を取得します. (VT_ARRAY | VT_UI1) ID=1 : IN1 の値を取得します. (VT_ARRAY | VT_UI1) ID=2 : IN2の値を取得します. (VT_ARRAY | VT_UI1) SCAN_GetInData 関数を実行します.	○	-

OUT<ノード番号>	VT_ARRAY | VT_VARIANT	スレーブ出力データの書き込みを行います. 要素 0 に OUT1 の値を設定/取得します. (VT_ARRAY | VT_UI1) 要素 1 に OUT2 の値を設定/取得します. (VT_ARRAY | VT_UI1) SCAN_SetOutData 関数を実行します. Put_Value 時の注意: VT_EMPTY を要素に格納したときは, 対応するデータの値をすべて0にします. また, Variant 配列ではなく, バイト配列(VT_ARRAY | VT_UI1)の場合は, ID プロパティの値によって設定箇所が変わります. ID=0,1 : OUT1 の値を設定します. ID=2 : OUT2の値を設定します. Get_Value 時の注意: 最後に Put_Value をした値を取得します. 変数オブジェクト作成後に一度も Put_Value を実行していないとき, 又は VT_EMPTY を設定したときはすべてのバイトが0に設定されたデータを取得します.	○	○

表 2-7 コントローラクラス マスタ指定時システム変数一覧

変数名	データ型	説明	属性	
			get	put
@VERSION	VT_I4	DLL のバージョンを取得します. SCAN_GetVersion 関数を実行します.	○	-
@DRVVERSION	VT_I4	ドライバのバージョンを取得します. SCAN_GetDriverVersion 関数を実行します.	○	-
@SLAVEDEVICE <ノード番号>	VT_ARRAY | VT_I4	指定スレーブの情報を取得します. [2] 要素 0 に出力データサイズを格納します. 要素 1 に入力データサイズを格納します. SCAN_GetSlaveDevice 関数を実行します.	○	-
@SLAVEDEVICEEX <ノード番号>	VT_ARRAY | VT_VARIANT	指定スレーブの情報を取得します. [2] 引数の詳細は表 2-10 @SLAVEDEVICEEX 引数データ詳細を参照して下さい. SCAN_GetSlaveDeviceEx 関数を実行します.	○	-

@SCANTIME	VT_I2	スキャンタイムの設定/取得を行います. SCAN_GetScanTimeValue 関数, SCAN_SetScanTimeValue 関数を実行します.	○	○
@ACTUALDEVICE <ノード番号>	VT_ARRAY | VT_I2	指定スレーブの情報を取得します. [2] 引数の詳細は表 2-10 を参照して下さい SCAN_GetActualSlaveDevice 関数を実行します.	○	-

表 2-8 コントローラクラス スレーブ指定時ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
IN	VT_ARRAY | VT_VARIANT	スレーブ入力データの読み出しを行います. 変数オブジェクトの ID プロパティの値によって取得データが変わります. ID=0 : 要素 0 に IN1 の値を取得 (VT_ARRAY | VT_UI1) 要素 1 に IN2 の値を取得します. (VT_ARRAY | VT_UI1) ID=1 : IN1 の値を取得します. (VT_ARRAY | VT_UI1) ID=2 : IN2の値を取得します. (VT_ARRAY | VT_UI1) SCAN_GetSlaveOutData 関数を実行します	○	-

※ [2]取得するスレーブ情報の詳細度は 次の通りです. @SLAVEDEVICEEX > @ACTUALDEVICE > @SLAVEDEVIDE

OUT	VT_ARRAY | VT_VARIANT	スレーブ出力データの書き込みを行います. 要素 0 に OUT1 の値を設定/取得します. (VT_ARRAY | VT_UI1) 要素 1 に OUT2 の値を設定/取得します. (VT_ARRAY | VT_UI1) SCAN_SetSlaveInData 関数を実行します Put_Value 時の注意: VT_EMPTY を要素に格納したときは, 対応するデータの値をすべて0にします. また, Variant 配列ではなく, バイト配列(VT_ARRAY | VT_UI1)の場合は, ID プロパティの値によって設定箇所が変わります. ID=0,1 : OUT1 の値を設定します. ID=2 : OUT2の値を設定します. Get_Value 時の注意: 最後に Put_Value をした値を取得します. 変数オブジェクト作成後に一度も Put_Value を実行していないとき, 又は VT_EMPTY を設定したときはすべてのバイトが0に設定されたデータを取得します.	○	○

表 2-9 コントローラクラス スレーブ指定時システム変数一覧

変数名	データ型	説明	属性	
			get	put
@VERSION	VT_I4	DLL のバージョンを取得します. SCAN_GetVersion 関数を実行します.	○	-
@DRVVERSION	VT_I4	ドライバのバージョンを取得します. SCAN_GetDriverVersion 関数を実行します.	○	-
@SLAVEDEVICE	VT_ARRAY | VT_I4	スレーブ(自分自身)の情報取得を行います. 要素 0 にスキャンタイプを格納します. 要素 1 に出力データ 1 サイズを格納します. 要素 2 に入力データ 1 サイズを格納します. 要素 3 に出力データ 2 サイズを格納します. 要素 4 に入力データ 2 サイズを格納します. 要素 5 に接続状態を格納します. 要素 6 に通信異常時出力データを格納します. SCAN_GetSelfSlaveDevice 関数を実行します.	○	-

表 2-10 @SLAVEDEVICEEX 引数データ詳細

要素番号	データ型	説明
0	VT_I2	ベンダ ID
1	VT_I2	プロダクトタイプ
2	VT_I2	プロダクトコード
3	VT_I2	スキャンタイプ
4	VT_I2	出力データ 1 サイズ(Byte)
5	VT_I2	入力データ 1 サイズ(Byte)
6	VT_I2	出力データ 2 サイズ(Byte)
7	VT_I2	入力データ 2 サイズ(Byte)
8	VT_I2	接続/非接続
9	VT_I2	COS/サイクリック送信感覚時間(ms)
10	VT_I2	出力データ 1 コネクションパスサイズ
11	VT_ARRAY | VT_I2	出力データ 1 コネクションパス
12	VT_I2	入力データ 1 コネクションパスサイズ
13	VT_ARRAY | VT_I2	入力データ 1 コネクションパス
14	VT_I2	出力データ 2 コネクションパスサイズ
15	VT_ARRAY | VT_I2	出力データ 2 コネクションパス
16	VT_I2	入力データ 2 コネクションパスサイズ
17	VT_ARRAY | VT_I2	入力データ 2 コネクションパス
18	VT_I2	予約エリア
19	VT_I2	予約エリア 2

表 2-11 @ACTUALDEVICE 引数データ詳細

要素番号	説明
0	ベンダ ID
1	プロダクトタイプ
2	プロダクトコード
3	スキャンタイプ

4	出力データ 1 サイズ(Byte)
5	入力データ 1 サイズ(Byte)
6	出力データ 2 サイズ(Byte)
7	入力データ 2 サイズ(Byte)

2.3.2. CAO-DeviceNet API 対応表

DeviceNet プロバイダは，コマンドの実行方法として CaoCommand::Execute，CaoVariable による 2 通りの方法を提供しています．

CaoCommand::Execute メソッドは，動作を行う API 関数を実行します．

CaoVariable は，値の設定/取得を行う API 関数を実行します．

表 2-12 コマンドクラス，変数クラスと DeviceNet API 対応表

CAO API		DeviceNet API	
クラス	オブジェクト名	マスタ	スレーブ
CaoCommand	ISEXISTCARD	SCAN_IsExistCard	SCAN_IsExistCard
	RESET	SCAN_Reset	SCAN_Reset
	REGDEVICE	SCAN_RegisterSlaveDevice	－
	REMOVEDEVICE	SCAN_RemoveDevice	－
	SETSCANLIST	SCAN_SetScanlist	－
	CLEARSCANLIST	SCAN_ClearScanlist	－
	STORESCANLIST	SCAN_StoreScanlist	SCAN_StoreScanlist
	LOADSCANLIST	SCAN_LoadScanlist	SCAN_LoadScanlist
	CONNECTDEVICE	SCAN_ConnectSlaveDevice	－
	DISCONNECTDEVICE	SCAN_DisconnectSlaveDevice	－
	IOREFRESH	SCAN_IoRefresh	SCAN_SlaveIoRefresh
	SENDCOS	SCAN_SendMasterCosToSlave	SCAN_SendSlaveCosToMaster
CaoVariable	@VERSION	SCAN_GetVersion	SCAN_GetVersion
	@DRVVERSION	SCAN_GetDrvVersion	SCAN_GetDrvVersion
	@SLAVEDEVICE	SCAN_GetSlaveDevice	SCAN_GetSelfSlaveDevice
	@SLAVEDEVICEEX	SCAN_ GetSlaveDeviceEx	－
	@ACTUALDEVICE	SCAN_GetActualSlaveDevice	－
	@SCANTIME	SCAN_GetScanTimeValue	－
		SCAN_SetScanTimeValue	－
	IN	SCAN_GetInData	SCAN_GetSlaveOutData
	OUT	SCAN_SetOutData	SCAN_SetSlaveInData

2.4. エラーコード

Devicenet プロバイダでは，固有のエラーコードはありません．ORiN2 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．

3. サンプルプログラム

DeviceNet のスレーブに接続し, Reset の実行とスレーブモードで値の設定, 取得をするコードを示します.

List 3-1 Sample.frm

```
Dim eng As CaoEngine
Dim ctrl As CaoController
Dim valIN As CaoVariable
Dim valOUT As CaoVariable

Private Sub Form_Load()

    ' CAO エンジンの生成
    Set eng = New CaoEngine

    'DeviceNet への接続
    Set ctrl = eng.Workspaces(0).AddController("DNet", "CaoProv.OMRON.DNet", "", _
                                        "Function=1,BoardId=0,Online=63:2,SlaveDevice=2:2")

    ' 変数オブジェクトの生成
    Set valIN = ctrl.AddVariable("IN", "ID=1")
    Set valOUT = ctrl.AddVariable("OUT", "ID=1")

End Sub

Private Sub Command1_Click()

    Dim Data(1) As Byte
    Data(0) = 1
    Data(1) = 0

    ' 値の設定
    valOUT.Value = Data

End Sub

Private Sub Command2_Click()

    ' 値の取得
    Dim vntVal() as Variant
    vntVal = valIN.Value

End Sub
```

3.1. API 実装状況

3.1.1. ボード管理用 API

表 3-1 ボードサービス API 実装状況

API 関数名	実装済	実装内容
SCAN_GetVersion	○	変数クラスにて実装
SCAN_GetDriverVersion	○	変数クラスにて実装
SCAN_IsExistCard	○	コマンドクラスにて実装
SCAN_Open	△	AddController 時に内部的に実行
SCAN_Close	△	Controller 削除時に内部的に実行
SCAN_Online	△	AddController 時に内部的に実行
SCAN_Offline	△	Controller 削除時に内部的に実行
SCAN_Reset	○	コマンドクラスにて実装

表 3-2 PC ボード割り込みサービス API 実装状況

API 関数名	実装済	実装内容
SCAN_GetIrqControl	－	－
SCAN_SetIrqControl	－	－
SCAN_RegIrqEvtNotifyMessage	－	－
SCAN_UnRegIrqEvtNotifyMessage	－	－
SCAN_PeekIrqEvent	－	－
SCAN_ClearIrqEvent	－	－

3.1.2. マスタ機能用 API

表 3-3 マスタ:スキャンリスト操作 API 実装状況

API 関数名	実装済	実装内容
SCAN_RegisterSlaveDevice	○	コマンドクラスにて実装
SCAN_GetSlaveDevice	○	変数クラスにて実装
SCAN_RemoveDevice	○	コマンドクラスにて実装
SCAN_ClearScanlist	○	コマンドクラスにて実装
SCAN_StoreScanlist	○	コマンドクラスにて実装
SCAN_LoadScanlist	○	コマンドクラスにて実装
SCAN_RegisterSlaveDeviceEx	―	代わりに SCAN_SetScanlist を用いて対応
SCAN_GetSlaveDeviceEx	○	変数クラスにて実装
SCAN_SetScanlist	○	コマンドクラスにて実装

表 3-4 マスタ:I/O 通信サービス API 実装状況

API 関数名	実装済	実装内容
SCAN_StartScan	△	AddController 時に内部的に実行
SCAN_StopScan	△	Controller 削除時に内部的に実行
SCAN_SetScanTimeValue	○	変数クラスにて実装
SCAN_GetScanTimeValue	○	変数クラスにて実装
SCAN_StoreScanTimeValue	○	コマンドクラスにて実装
SCAN_LoadScanTimeValue	○	コマンドクラスにて実装
SCAN_GetActualSlaveDevice	○	変数クラスにて実装
SCAN_ConnectSlaveDevice	○	コマンドクラスにて実装
SCAN_DisconnectSlaveDevice	○	コマンドクラスにて実装

表 3-5 マスタ:I/O データアクセスサービス API 実装状況

API 関数名	実装済	実装内容
SCAN_IoRefresh	○	コマンドクラスにて実装
SCAN_GetInData	○	変数クラスにて実装
SCAN_SetOutData	○	変数クラスにて実装
SCAN_SendMasterCosToSlave	○	コマンドクラスにて実装

3.1.3. スレーブ機能用 API

表 3-6 スレーブスキャンリスト操作 API

API 関数名	実装済	実装内容
SCAN_RegisterSlaveDevice	○	AddController 時に内部的に実行
SCAN_RemoveSelfSlaveDevice	○	CaoController 削除時に内部的に実行
SCAN_GetSelfDevice	○	変数クラスにて実装
SCAN_StoreSlaveScanlist	○	コマンドクラスにて実装
SCAN_LoadSlavelist	○	コマンドクラスにて実装

表 3-7 I/O 通信サービス API

API 関数名	実装済	実装内容
SCAN_ConnectMasterDevice	○	AddController 時に内部的に実行
SCAN_DisconnectMasterDevice	○	CaoController 削除時に内部的に実行

表 3-8 I/O データアクセスサービス API

API 関数名	実装済	実装内容
SCAN_SlaveIoRefresh	○	コマンドクラスにて実装
SCAN_GetSlaveOutData	○	変数クラスにて実装
SCAN_SetSlaveInData	○	変数クラスにて実装
SCAN_SendSlaveCosToMaster	○	コマンドクラスにて実装

3.1.4. Explicit メッセージ用 API

表 3-9 メッセージ管理タイマサービス API

API 関数名	実装済	実装内容
SCAN_SetMessageTimerValue	―	―
SCAN_GetMessageTimerValue	―	―
SCAN_StoreMessageTimerValueList	―	―
SCAN_LoadMessageTimerValueList	―	―

表 3-10 クライアントメッセージサービス API

API 関数名	実装済	実装内容
SCAN_RegClientEvtNotifyMessage	―	―
SCAN_UnRegClientEvtNotifyMessage	―	―
SCAN_PeekClientEvent	―	―
SCAN_GetClientEventLength	―	―
SCAN_SendClientExplicit	―	―
SCAN_ReceiveClientExplicit	―	―

表 3-11 サーバメッセージサービス API

API 関数名	実装済	実装内容
SCAN_RegObjectClass	―	―
SCAN_UnRegObjectClass	―	―
SCAN_RegServerEvtNotifyMessage	―	―
SCAN_UnRegServerEvtNotifyMessage	―	―
SCAN_PeekServerEvent	―	―
SCAN_GetServerEventLength	―	―
SCAN_SendServerExplicit	―	―
SCAN_ReceiveServerExplicit	―	―

3.1.5. 保守用 API

表 3-12 ステータスサービス API

API 関数名	実装済	実装内容
SCAN_GetNetworkStatus	―	―
SCAN_GetScannerStatus	―	―
SCAN_GetMasterModeStatus	―	―
SCAN_GetSlaveModeStatus	―	―
SCAN_IsScanlistSlaveDeviceRegist	―	―
SCAN_IsDeviceConnection	―	―
SCAN_GetCycleTime	―	―
SCAN_GetMaxCycleTime	―	―
SCAN_GetMinCycleTime	―	―
SCAN_ClearCycleTime	―	―
SCAN_GetSlaveModeStatus	―	―

表 3-13 異常履歴アクセスサービス API

API 関数名	実装済	実装内容
SCAN_GetErrorLog	―	―
SCAN_ClearErrorLog	―	―

表 3-14 PC ウォッチドッグタイマサービス API

API 関数名	実装済	実装内容
SCAN_EnablePCWDTTimer	―	―
SCAN_RefreshPCWDTTimer	―	―