

様

## 仕様書

PCardRW64.DLL

シリアル通信制御 API ライブラリー

NOTE:

Ver. 1.0.0.0

受領印欄

パナソニックシステムネットワークス株式会社  
ターミナルシステムビジネスユニット

福岡市博多区美野島4丁目1-62  
電話：(092) 477-1592 (〒812-8531)

REF		REV
No.		A

発 行		
日付	2013. 4. 5	
確 認		



- ・改定履歴

[illegible]



## 【目 次】

<b>1.</b>	<b>適用範囲.....</b>	<b>1</b>
<b>2.</b>	<b>概要 .....</b>	<b>1</b>
2.1.	動作対象環境 .....	1
2.2.	適用機種 .....	1
<b>3.</b>	<b>使い方.....</b>	<b>1</b>
3.1.	Visual C#、Visual Basic.NET からの使用.....	1
<b>4.</b>	<b>APIリファレンス.....</b>	<b>2</b>
4.1.	シリアルポート名取得関数 crwGetPortNames .....	5
4.2.	COMポート情報取得関数 crwGetComDevInfo .....	6
4.3.	ポートオープン関数 crwOpenPort.....	7
4.4.	ポートクローズ関数 crwClosePort.....	8
4.5.	コマンド送受信関数 crwSendCommandRR .....	9
4.6.	コマンド送信関数 crwSendCommand .....	11
4.7.	レスポンス受信関数 crwReceiveResponse .....	12
4.8.	PI Image Data 作成&出力関数 crwPIDataSend .....	14
4.9.	IB Image Data 作成&出力関数 crwIBBinarySend.....	15
4.10.	データ送信関数（応答あり） crwSendDataWR .....	17
4.11.	データ送信関数（応答なし） crwSendDataQR.....	18
4.12.	データ受信関数 crwReceiveData .....	19
4.13.	電文データ送受信関数 crwSendDataRR.....	20
4.14.	電文データ送信関数 crwSendDataBCC .....	21
4.15.	電文データ受信関数 crwReceiveDataBCC.....	22
4.16.	受信バッファークリア関数 crwComRXBufferClear .....	23
4.17.	バイナリデータ送信関数QR crwSendBinaryDataQR.....	24
4.18.	バイナリデータ受信関数 crwReceiveBinaryData .....	25
4.19.	レスポンス受信イベントハンドラ登録関数 crwSetResponseEventProc .....	26
4.20.	電文データ受信イベントハンドラ登録関数 crwSetReceiveDataBCCEventProc .....	29
4.21.	通信イベントハンドラ登録関数 crwSetCommEventProc .....	32
4.22.	通信中DSR/CTS信号監視設定関数 crwComDSRCTSsw.....	34
4.23.	文字コード変換関数 crwCharCodeConv.....	35
4.24.	JIS/Shift-JIS漢字コード変換関数 crwJISconvSJIS .....	35
<b>5.</b>	<b>iniファイルの構成.....</b>	<b>36</b>
<b>6.</b>	<b>ログファイルの構成.....</b>	<b>37</b>
<b>7.</b>	<b>DLL使用上の注意事項.....</b>	<b>39</b>
<b>8.</b>	<b>補足資料.....</b>	<b>40</b>
8.1.	マルチスレッドの基本使用方法.....	40
8.2.	マルチスレッドの動作説明.....	41
8.2.1.	シングルスレッド（1ポート使用）の場合 .....	41
8.2.2.	マルチスレッド（同一ポート使用）の場合.....	42
8.2.3.	マルチスレッド（別ポート使用）の場合.....	43
8.3.	データ送受信処理について.....	44
8.3.1.	送信処理 .....	44
8.3.2.	受信処理 .....	45



## 1. 適用範囲

本仕様書はパナニック システムネットワークズ株式会社が提供する機器の上位アプリケーション（以降 APL と記す）のための RS-232C 通信制御ドライバー（PCardRW64.dll）が提供するアプリケーションインターフェイス（API）仕様について記述します。

## 2. 概要

ドライバー PCardRW64.dll はパナニック システムネットワークズ株式会社が提供する機器を、Win64 から利用するための API を提供します。

### 2.1. 動作対象環境

本 DLL の動作対象環境は以下のとおりです。

**OS : Windows XP / Windows Server 2003 / Windows Vista / Server 2008 / Windows 7 / Windows 8 / Windows Server2012**

※ すべての64ビット版のみ動作確認。

### 2.2. 適用機種

下記の機種でパナニック システムネットワークズ株式会社の標準通信プロトコルをサポートする機種

- ・ JT-KP41U / JT-KU4x シリーズ
- ・ KU-R3000 / KU-A3000シリーズ
- ・ KU-R40000シリーズ
- ・ KU-R10000 / KU-R18000 / KU-R13000
- ・ KU-R28000 / KU-A9000シリーズ
- ・ KU-Z28000 / KU-Z9000シリーズ
- ・ KU-A1400シリーズ

注記）コマンド仕様に関しては各機種のコマンド仕様書を参照してください。

## 3. 使い方

PCardRW64.dll は Visual Studio 2005(C++,C#,VB)、Visual Studio 2008(C++,C#,VB)  
Visual Studio 2010(C++,C#,VB)、Visual Studio 2012(C++,C#,VB)  
で開発されるプロジェクトから利用可能です。

### 3.1. Visual C#、Visual Basic.NET からの使用

Visual Studio 2005、Visual Studio 2008、Visual Studio 2010、Visual Studio 2012

DllImportAttributeクラスを使用して、DLLから関数をインポートしてください。(サンプルソース参照)

## 4. APIリファレンス

ここでは、PCardRW64.dll が提供する API について説明します。各関数、パラメータは C++ 言語に準拠した形式で記述されています。また、パラメータの型と API の戻り値の型は、Windows SDK に基づいたデータ型で記述されています。

### 【APIの概要】

PCardRW64.dll が提供する API は以下のとおりです。

No	関数名	機能
1	crwGetPortNames()	シリアルポート名取得関数
2	crwGetComDevInfo()	COM ポート情報取得関数
3	crwOpenPort()	ポートオープン関数
4	crwClosePort()	ポートクローズ関数
5	crwSendCommandRR()	コマンド送受信関数
6	crwSendCommand()	コマンド送信関数
7	crwReceiveResponse()	レスポンス受信関数
8	crwSendDataWR()	データ送信(応答あり)関数
9	crwSendDataQR()	データ送信(応答なし)関数
10	crwReceiveData()	データ受信関数
11	crwSendDataRR()	データ送受信関数
12	crwSendDataBCC()	電文データ送信関数
13	crwReceiveDataBCC()	電文データ受信関数
14	crwComRXBufferClear()	受信バッファクリア関数
15	crwSendBinaryDataQR()	バイナリデータ送信関数 QR
16	crwReceiveBinaryData()	バイナリデータ受信関数
17	crwComDSRCTSsw()	通信中 DSR/CTS 信号監視設定関数
18	crwPIDataSend	PI Image Data 作成&出力関数
19	crwIBBinarySend	IB Image Data 作成&出力関数
20	crwCharCodeConv	文字コード変換関数
21	crwJISconvSJIS	JIS/Shift-JIS 漢字コード変換関数

また、各 API における戻り値の定数は以下のとおりです。

定数名	値
RET_CRW_SUCCESS RET_CRW_ACK RET_CRW_RESPONSE	0
RET_CRW_FAILURE RET_CRW_NAK RET_CRW_READY	1
RET_CRW_BUSY RET_CRW_NAK_BINARY	2
RET_CRW_ERR_PARAM	100
RET_CRW_ERR_PORT	101
RET_CRW_ERR_RES_TIMEOUT	110
RET_CRW_ERR_CTS_TIMEOUT	111
RET_CRW_ERR_DSR	112
RET_CRW_ERR_DSR_TIMEOUT <sup>1</sup>	112
RET_CRW_ERR_BUSY_TIMEOUT	113
RET_CRW_ERR_BCC_RETRY	120
RET_CRW_ERR_BUFFEROVER	130
RET_CRW_ERR_BUSY	140
RET_CRW_ERR_COMM_WRITE	200
RET_CRW_ERR_COMM_READ	201

各定数の説明は、各 API の説明の部分をご参照してください。

### 1.1.1.

<sup>1</sup>下位バージョンとの互換性の為、記載されている戻り値となります。



# 【Win SDKとC++に基づいたパラメータの型と戻り値の型】

型	意味
<b>BOOL</b>	TRUE(0 以外) と FALSE(0) を有効値とする整数
<b>LPCSTR</b>	定数文字列への 32 ビットポインタ
<b>short int</b>	16 ビット長の符号付き整数
<b>LONG</b>	32 ビット長の符号付き整数
<b>BSTR*</b>	32 ビットの文字ポインタ
<b>LPCVOID</b>	あらゆる型のデータへのポインタ。(内容が変更されない)
<b>LPARAM</b>	32 ビットメッセージパラメータ

## 注記)

PCardRW64.dll では、基本的に BSTR 型の文字ポインタを用いて文字列の受け渡しを行っています。BSTR 型のメモリ領域の確保には、SysAllocString / SysFreeString API を使用します。

引数として BSTR 型のポインタを受け取った場合、DLL 内部で一度、領域が解放され、領域を再確保し、APL に処理を返します。つまり、関数に使用した変数は必ず領域を確保した状態で APL に返ってくることになります。

使用した領域を確実に解放するために、BSTR 型の変数は使用した後に SysFreeString API を呼び出して領域を解放するようにしてください。

## 【イベントの概要】

PCardRW64.DLL が提供するイベントは以下の通りです。

No	関数名	機能
1	<b>crwSetResponseEventProc()</b>	レスポンス受信イベントハンドラ登録関数
2	<b>crwSetReceiveDataBCCEventProc()</b>	電文データ受信イベントハンドラ登録関数
3	<b>crwSetCommEventProc()</b>	通信イベントハンドラ登録関数

## 機種別に使用する PCardRW64.dll API 関数の一覧表

### 【機種別使用関数一覧】

機種	処理形式	関数名	内容	頁
全機種共通	通信系	crwGetPortNames	シリアルポート名取得	5
		crwGetComDevInfo	COM ポート情報取得関数	6
		crwOpenPort	ポートオープン関数	7
		crwClosePort	ポートクローズ関数	8
		crwComRXBufferClear	受信バッファークリア関数	23
		crwComDSRCTSsw	通信中 DSR/CTS 信号監視設定関数	34
	イベント受信	crwSetCommEventProc	通信イベントハンドラ登録	32
JT-KP41U シリーズ KU-R3000/A3000 KU-Z28000/Z9000 KU-R28000/A9000 KU-A1400 シリーズ	ポーリング処理	crwSendCommandRR	コマンド送受信関数	9
		crwSendCommand	コマンド送信関数	11
		crwReceiveResponse	レスポンス受信関数	12
	イメージ転送	crwPIDataSend	PI Image Data 作成&出力関数	14
	イベント受信	crwSetResponseEventProc	レスポンス受信 イベントハンドラ登録関数	26
JT-KP41U シリーズ	イメージ転送	crwIBBinarySend	IB Image Data 作成&出力関数	15
KU-R40000 シリーズ KU-R10000 シリーズ KU-R13000 シリーズ KU-R18000 シリーズ (簡易 POS 接続仕様)	コマンド・ レスポンス処理	crwSendDataRR	電文データ送受信関数	20
		crwSendDataBCC	電文データ送信関数	21
		crwReceiveDataBCC	電文データ受信関数	22
	イベント受信	crwSetReceiveDataBCCEventProc	電文データ受信イベントハンドラ登録関数	29

### 【その他関数一覧】

機種	処理形式	関数名	内容	頁
自由度を持たせる為 の関数群	データ無加工 処理	crwSendDataWR	データ送信関数 (ACK/NAK 応答待ちあり)	17
		crwSendDataQR	データ送信関数 (ACK/NAK 応答待ちなし)	18
		crwReceiveData	データ受信関数	19
	バイナリデータ 無加工処理	crwSendBinaryDataQR	バイナリデータ送信関数 QR	24
		crwReceiveBinaryData	バイナリデータ受信関数	25
	文字コード 変換処理	crwCharCodeConv	文字コード変換関数	35
		crwJISconvSJIS	JIS/Shift-JIS 漢字コード変換関数	

#### 4.1. シリアルポート名取得関数 `crwGetPortNames`

##### 【機能】

利用可能なシリアルポート名を取得し、CSV形式で返却します。

##### 【形式】

```
BOOL crwGetPortNames (
    BSTR*   bpPortNames,
    short int* piNumOfPort
);
```

##### 【パラメータ】

名前	説明
<i>bpPortNames</i>	使用可能なシリアルポート名を CSV 形式で返却するポインタ
<i>piNumOfPort</i>	使用可能なシリアルポートの数を返却するポインタ

##### 【戻り値】

値	説明
TRUE	関数成功
FALSE	関数失敗

##### 【解説】

COM1,COM2 が実行対象 PC に存在する場合、"COM1,COM2"が *bpPortNames* に、*piNumOfPort* には 2 が設定されます。

使用可能なシリアルポートが存在しない場合、空文字列が *bpPortNames* に、*piNumOfPort* には 0 が設定されます。

*bpPortNames*、*piNumOfPort* パラメータに NULL を指定した場合は、FALSE を返します。

本関数と他の関数をマルチスレッドで実行した場合、以下の動作となります。

- ・ 本関数を実行中に他の関数を実行した場合
  - 他の関数は実行できず待機し本関数の完了とともに処理を開始します。
- ・ 本関数以外の関数を実行中に本関数を実行した場合
  - 他の関数の完了を待たずに処理可能です。

## 4.2. COMポート情報取得関数 `crwGetComDevInfo`

### 【機能】

指定した COM ポートドライバのプロパティ情報を取得する。

### 【形式】

```
BOOL crwGetComDevInfo (
    short int  iPortNo,
    BSTR*      bpDriverDesc,
    BSTR*      bpProviderName,
    BSTR*      bpDriverDate,
    BSTR*      bpDriverVersion
);
```

### 【パラメータ】

名前	説明
<i>iPortNo</i>	取得したいシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>bpDriverDesc</i>	指定したシリアルポートのディスクリプター名を返却するエリアのポインタ
<i>bpProviderName</i>	指定したシリアルポートドライバのプロバイダーを返却するエリアのポインタ
<i>bpDriverDate</i>	指定したシリアルポートドライバの日付を返却するエリアのポインタ
<i>bpDriverVersion</i>	指定したシリアルポートドライバのバージョンを返却するエリアのポインタ

### 【戻り値】

値	説明
TRUE	関数成功
FALSE	関数失敗

### 【解説】

指定したCOMポートドライバのプロパティ情報を取得する。

※この情報はログ取得時に、`crwOpenPort()` 関数実行すると追加されます。

本関数と他の関数をマルチスレッドで実行した場合、以下の動作となります。

- ・本関数を実行中に他の関数を実行した場合  
→ 他の関数は実行できず待機し本関数の完了とともに処理を開始します。
- ・本関数以外の関数を実行中に本関数を実行した場合  
→ 他の関数の完了を待たずに処理可能です。

### 4.3. ポートオープン関数 crwOpenPort

#### 【機能】

シリアルポートをオープンします。

#### 【形式】

```
short int crwOpenPort(  
    short int  iPortNo,  
    LONG       lBaudRate,  
    LPCSTR     lpDataBit,  
    LPCSTR     lpParity  
);
```

#### 【パラメータ】

名前	説明
<i>iPortNo</i>	オープンするシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>lBaudRate</i>	シリアルポートの通信速度（単位：bps） （2400、4800、9600、19200、38400、57600、115200）
<i>lpDataBit</i>	シリアルポートのデータビット長（7または8）
<i>lpParity</i>	シリアルポートのパリティチェック（N：なし / E：偶数 / O：奇数）

#### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	オープン完了
RET_CRW_FAILURE(1)	指定したシリアルポートは既にオープンされている
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	オープン不可（他のプロセスがポートを使用中）

#### 【解説】

シリアルポートをオープンし、接続されている機器を使用可能な状態にします。  
これ以降、このポート番号を経由した機器との送受信が可能となります。

※ マルチスレッド環境下で使用する場合

各ポート別のオープンは、メインスレッドでコールされるようにお願いします。

8.1 マルチスレッドの基本使用方法 をご理解の上ご使用ください。

#### 4.4. ポートクローズ関数 crwClosePort

##### 【機能】

シリアルポートをクローズします。

##### 【形式】

```
short int crwClosePort(  
    short int  iPortNo  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	クローズするシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	クローズ完了
RET_CRW_FAILURE(1)	指定したシリアルポートは既にクローズされている
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中

##### 【解説】

シリアルポートをクローズし、再度オープンできる状態にします。

このAPIが成功した後は、crwOpenPort()関数を実行しないと使用できません。

## 4.5. コマンド送受信関数 crwSendCommandRR

### 【機能】

渡されたパラメータから、コマンド電文を生成し、シリアルポートに出力後、レスポンスデータの受信を行います。

### 【形式】

```
short int crwSendCommandRR(  
    short int iPortNo,  
    short int iRecMode,  
    LONG      ITimeOut,  
    LONG      IBusyTimeOut,  
    short int iDSRCheck,  
    LPCSTR lpCommandCode,  
    LPCSTR lpCommandParameter,  
    short int iParameterLength,  
    BSTR* bpCOMSTS,  
    BSTR* bpERRSEN,  
    BSTR* bpResDATA  
);
```

### 【パラメータ】

名前	説明
<i>iPortNo</i>	送信するシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>iRecMode</i>	ビジステータス受信の時にポーリング再送を行なうかどうかのフラグ （0以外：する / 0：しない）
<i>ITimeOut</i>	ACK/NAK待ち及びCTSオン待ちタイマー値（単位：msec）
<i>IBusyTimeOut</i>	ビジステータスの終了待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR信号線をチェックするかどうかのフラグ（0：しない / 0以外：する）
<i>lpCommandCode</i>	機器のコマンドコード（2バイト）
<i>lpCommnadParameter</i>	機器のコマンドのパラメータ パラメータ無しの際はNULL
<i>iParameterLength</i>	コマンドパラメータの長さ
<i>bpCOMSTS</i>	レスポンスのコマンドコード(=COM\$)・レディステータスのステータス 情報(=STS\$)を返却するエリアのポインタ（2バイト）
<i>bpERRSEN</i>	レスポンスのエラーコード(=ERR\$)・レディステータスのセンサー情報 (=SEN\$)を返却するエリアのポインタ（2バイト）
<i>bpResDATA</i>	レスポンスのデータ部分(=DATA\$)を返却するエリアのポインタ (センサー情報以外の追加情報がある場合は、以降の情報は "bpResDATA" に格納される)

### 【戻り値】

値	説明
RET_CRW_RESPONSE(0)	レスポンスデータ("A")受信
RET_CRW_NAK(1)	否定 (NAK) 応答
RET_CRW_BUSY(2)	レスポンスデータ("B")受信
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	機器からの応答待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTSオン待ちタイムアウト
RET_CRW_ERR_DSR_TIMEOUT(112)	DSRオフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY_TIMEOUT(113)	機器からのビジステータス終了待ちタイムアウト
RET_CRW_ERR_BCC_RETRY(120)	BCCエラーリトライオーバー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時のWIN64 APIエラー発生
RET_CRW_ERR_COMM_READ(201)	受信時のWIN64 APIエラー発生

### 【解説】

渡されたコマンドコード、コマンドパラメータから次の形式でコマンド電文を生成します。

電文形式 : [EOT] [STX] [*lpCommandCode*] [*lpCommnadParameter*] [ETX] [BCC]

これをシリアルポートに出力し、機器からのACK応答を受信したら、次のポーリングコマンドを送信します。

電文形式 : [EOT] [ENQ]

注記) [*lpCommnadParameter*]にはコマンド2文字を除いたテキストを設定します。

これにより機器からの**ビジーステータス**、**コマンドレスポンス**のいずれかのデータを受信します。

応答がない場合、*lTimeOut*時間待ち、制御をAPLに返します。

**ビジーステータス**が受信された場合は、*iRecMode*を参照してポーリングするかどうか判断し、レディステータスかコマンドレスポンスが受信されるまでポーリングを繰り返します。ビジーステータス以外の状態に遷移するか、*lBusyTimeOut*時間経過した場合、APLに通知します。

**コマンドレスポンス**が受信された場合は、それぞれの受信電文形式から、対象とするデータ部分を抽出し、返却用パラメータにセットして、制御をAPLに返します。

また、機器からの応答電文のBCC（水平パリティ）を計算し、正しくない場合には、機器にNAKを送信する。この動作を3回繰り返しても正常なBCCが得られないときは、エラーを返します。

### 【注記】

#### ① *lBusyTimeOut* の設定値

機器動作系のコマンドの場合は、動作処理時間の2倍程度の値設定を行ってください。

例1 : RD・WR・VR = 処理時間 × (Retry回数 + 1) × 2

例2 : PO・PR・ER・CO = 処理時間 × 2

・消去印字に関わるPO・PR・ER コマンドは環境温度に大きく影響されますので、30000ms以上値設定が必要ことがあります。(複数のリライトカード媒体種で運用している場合など)

#### ② “CI” コマンドでの取り扱い

JT-KP41U等の“CI” コマンド送信時では *iRecMode=0*(しない) に設定し“CL” コマンドで、カード挿入待ちキャンセルを可能な状態に実装してください。

*crwReceiveResponse(iRecMode=0)*を繰り返し発行することで機器のステータスを取得することによりカードが挿入されたかどうかを検知することができます。

下記にいくつかある実装パターン例を記載いたします。

方法1 : 上記の方法

*crwSendCommandRR*(“CI” *iRecMode=0*) + *crwReceiveResponse(iRecMode=0)*を繰り返し発行することで、カード挿入を検知。

方法2 : *crwReceiveResponse(iRecMode=0)*を繰り返し発行し、ポーリングステータス確認後、*crwSendCommandRR(iRecMode=1)* で“CI” コマンド発行。

方法3 : *crwSendCommandRR*(“ST” *iRecMode=1*) でコマンド繰り返し発行にてセンサー状態(1)を検知し“CI” コマンド発行

(1) JT-KP41UではAセンサーでの判定になりますが、シャッターユニット搭載機種はSセンサーでの判定になることもありますので、お使いの機種の仕様書をご確認ください。

(2) “ST”コマンドのアンサー情報での全比較判定は行わないでください。

Aセンサー又はSセンサーでの情報のみで判定するよう実装してください。

方法4 : *crwSendCommand*(“CI”) 発行後、*crwResponseEventProc(iRecMode=1)*コールバック関数登録でカード挿入を待つ。

カード挿入待ちをキャンセルする場合は、  
*crwResponseEventProc()*コールバック関数登録解除したあと  
*crwSendCommnadRR*(“CL” *iRecMode=1*)にて、キャンセル要求を行う。



## 4.6. コマンド送信関数 `crwSendCommand`

### 【機能】

渡されたパラメータから、コマンド電文を生成し、シリアルポートに出力します。

### 【形式】

```
short int crwSendCommand(  
    short int  iPortNo,  
    LONG      ITimeOut,  
    short int  iDSRCheck,  
    LPCSTR     lpCommandCode,  
    LPCSTR     pCommnadParameter,  
    short int  iParameterLength  
);
```

### 【パラメータ】

名前	説明
<i>iPortNo</i>	コマンドを送信するシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>ITimeOut</i>	ACK/NAK待ち及びCTSオン待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR信号線をチェックするかどうかのフラグ（0以外：する / 0：しない）
<i>lpCommandCode</i>	機器のコマンドコード（2バイト）
<i>lpCommnadParameter</i>	機器のコマンドのパラメータ（パラメータ無しの時はNULL）
<i>iParameterLength</i>	コマンドパラメータの長さ

### 【戻り値】

値	説明
RET_CRW_ACK(0)	肯定（ACK）応答
RET_CRW_NAK(1)	否定（NAK）応答
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	ACK/NAK待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTSオン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSRオフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時のWIN64 APIエラー発生
RET_CRW_ERR_COMM_READ(201)	受信時のWIN64 APIエラー発生

### 【解説】

渡されたコマンドコード、コマンドパラメータから次の形式でコマンド電文を生成します。

形式：[EOT] [STX] [*lpCommandCode*] [*lpCommnadParameter*] [ETX] [BCC]

これをシリアルポートに出力し、機器からの応答を*ITimeOut*時間まで待ち、制御をAPLに返します。

注記） [*lpCommnadParameter*]にはコマンド2文字を除いたテキストを設定します。

コマンドパラメータで指定した文字列長とコマンドパラメータの長さで指定した値が異なる場合、RET\_CRW\_ERR\_PARAM(100)が返ります。

送信時のWIN64APIエラーとは、WriteFile APIの戻り値がFALSEだった場合を表しています。

受信時のWIN64APIエラーとは、ReadFile APIの戻り値がFALSEだった場合を表しています。

#### << 信号線のチェック >>

データの送受信時には、シリアル信号線(DSR /CTS)のチェックを行います。

DSR線はチェックフラグ*iDSRCheck*がONの時にチェックし、これはCTS線のチェックよりも先に行い、その方法は以下のとおりです。

- ・送信時                   : 送信前にDSRをチェックし、オフであればタイムアウトまで待ちます。
- ・受信時                   : 受信データなしであればDSRをチェックし、オフであればすぐエラーとします。

以下のAPIについても信号線のチェックはこの方法で行います。

## 4.7. レスポンス受信関数 `crwReceiveResponse`

### 【機能】

ポーリングを送信し、機器からのレスポンスデータを受け取ります。

### 【形式】

```
short int crwReceiveResponse(  
    short int  iPortNo,  
    short int  iRecMode,  
    LONG       ITimeOut,  
    short int  iDSRCheck,  
    BSTR*      bpCOMSTS,  
    BSTR*      bpERRSEN,  
    BSTR*      bpResDATA  
);
```

### 【パラメータ】

名前	説明
<i>iPortNo</i>	レスポンスを受信するシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>iRecMode</i>	ビジステータス受信時にポーリング再送を行うかどうかのフラグ （0 以外：する / 0：しない）
<i>ITimeOut</i>	ポーリングに対する機器からの応答待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR 信号線のチェックを行うかどうかのフラグ（0 以外：する / 0：しない）
<i>bpCOMSTS</i>	ステータス情報、またはコマンドコードを返却するエリアのポインタ（2 バイト）
<i>bpERRSEN</i>	センサー情報、またはエラーコードを返却するエリアのポインタ（2 バイト）
<i>bpResDATA</i>	受信データを返却するエリアのポインタ（可変長） （センサー情報以外の追加情報がある場合は、以降の情報は “bpResDATA” に格納される）

### 【戻り値】

値	説明
RET_CRW_RESPONSE(0)	レスポンスデータ("A")受信
RET_CRW_READY(1)	レスポンスデータ("R")受信
RET_CRW_BUSY(2)	レスポンスデータ("B")受信
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	機器からの応答待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BCC_RETRY(120)	BCC エラーリトライオーバー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

### 【解説】

シリアルポートに対して、次のポーリングコマンドを送信します。

形式：[EOT] [ENQ]

これにより機器から以下のいずれかのデータを受信します。

レディステータス：

[STX][‘R’][ステータス情報(2バイト)][センサー情報(2バイト)][ETX][BCC]

ビジステータス：

[STX][‘B’][コマンドコード(2バイト)][センサー情報(2バイト)][ETX][BCC]

コマンドレスポンス：

[STX][‘A’][コマンドコード(2バイト)][エラーコード(2バイト)][受信データ(可変長)][ETX][BCC]

それぞれの受信形式から、対象とするデータ部分を抽出し、返却用パラメータにセットして、制御を APL に返します。

ビジステータスが受信された場合、*iRecMode*を参照してポーリングするかどうか判断し、レディステータスかコマンドレスポンスが受信されるまでポーリングを繰り返します。

このとき、*iRecMode*を0以外に設定し、ビジステータス受信時にDLL内でポーリング再送を行うようにした場合、ビジステータスを受信している限り、呼び出し側に制御が戻らなくなりますので、注意が必要です。他のスレッドから同一ポートを指定して他の関数を呼び出した場合、*crwReceiveResponse()*関数の終了を待ちます。

機器からの応答電文のBCC（水平パリティ）を計算し、正しくない場合には、機器にNAKを送信します。

この動作を3回繰り返しても正常なBCCが得られないときは、*RET\_CRW\_ERR\_BCC\_RETRY*を返します。

#### 【注記】

##### ① “C I” コマンドでの取り扱い

JT-KP41U等の“CI”コマンド送信時では *iRecMode=0*(しない) に設定し“CL”コマンドで、カード挿入待ちキャンセルを可能な状態に実装してください。

*crwReceiveResponse(iRecMode=0)*を繰り返し発行することで機器のステータスを取得することによりカードが挿入されたかどうかを検知することができます。

下記にいくつかある実装パターン例を記載いたします。

方法1：上記の方法

*crwSendCommandRR*(“CI” *iRecMode=0*) + *crwReceiveResponse(iRecMode=0)*を繰り返し発行することで、カード挿入を検知。

方法2：*crwReceiveResponse(iRecMode=0)*を繰り返し発行し、ポーリングステータス確認後、*crwSendCommandRR(iRecMode=1)*で“CI”コマンド発行。

方法3：*crwSendCommandRR*(“ST” *iRecMode=1*)でコマンド繰り返し発行にてセンサー状態(1)を検知し“CI”コマンド発行

(1) JT-KP41UではAセンサーでの判定になりますが、シャッターユニット搭載機種はSセンサーでの判定になることもありますので、お使いの機種の仕様書をご確認ください。

(2) “ST”コマンドのアンサー情報での全比較判定は行わないでください。

Aセンサー又はSセンサーでの情報のみで判定するよう実装してください。

方法4：*crwSendCommand*(“CI”)発行後、*crwResponseEventProc(iRecMode=1)*コールバック関数登録でカード挿入を待つ。

カード挿入待ちをキャンセルする場合は、

*crwResponseEventProc()*コールバック関数登録解除したあと

*crwSendCommandRR*(“CL” *iRecMode=1*)にて、キャンセル要求を行う。

## 4.8. PI Image Data 作成&出力関数 crwPIDataSend

### 【機能】

機器専用のイメージデータを PI コマンド形式のデータを作成し、シリアルポートへ出力します。

### 【形式】

```
short int crwPIDataSend(  
    short int  iPortNo,  
    short int  iRetryCount,  
    short int  iDSRCheck,  
    short int  iCTSCheck,  
    short int  iWidth,  
    short int  iHeigth,  
    short int  iStartX,  
    short int  iStartY,  
    LONG       ITimeout,  
    LONG       IComBufferSize,  
    LPCSTR     strPathFileName,  
    BSTR*      bpCOMSTS,  
    BSTR*      bpERRSEN,  
);
```

### 【パラメータ】

名前	説明
<i>iPortNo</i>	シリアルポートの番号
<i>iRetryCount</i>	データ送信、NAK 受信、及び SUN エラー受信時のリトライ回数
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ (0 : しない / 0 以外 : する)
<i>iCTSCheck</i>	CTS 信号線をチェックするかどうかのフラグ (0 : しない / 0 以外 : する)
<i>iWidth</i>	印字エリアの横幅 (機器の仕様書参考) 例 : 319
<i>iHeigth</i>	印字エリアの縦幅 (機器の仕様書参考) 例 : 459
<i>iStartX</i>	印字開始位置(X 座標) 例 : 20
<i>iStartY</i>	印字開始位置(Y 座標) 例 : 100
<i>ITimeout</i>	データの受信待ちタイマー値(単位:msec)
<i>IComBufferSize</i>	COM ポート通信バッファサイズ (機器の通信バッファサイズ合わせてデータを作成し送信します)
<i>strPathFileName</i>	イメージデータファイル名[ BMP, JPG, GIF ] 形式 (フルパス指定可能)
<i>bpCOMSTS</i>	ステータス情報、またはコマンドコードを返却するエリアのポインタ
<i>bpERRSEN</i>	センサー情報、またはエラーコードを返却するエリアのポインタ

### 【戻り値】

値	説明
RET_CRW_RESPONSE(0)	レスポンスデータ("A")受信
RET_CRW_NAK(1)	否定 (NAK) 応答
RET_CRW_BUSY(2)	レスポンスデータ("B")受信
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	機器からの応答待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー (端末未接続 / 電源オフ)
RET_CRW_ERR_BCC_RETRY(120)	BCC エラーリトライオーバー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

### 【解説】

- ・転送が正常に終了すると、API90にて終了します。
- ・転送が正常終了するかエラーが発生しない限り、関数からの応答はありません。
- ・画像ファイルの形式は[ BMP, JPG, GIF ] のいずれかにてご指定ください。
- ・イメージデータフォーマットはコマンド仕様書をご参照ください。

#### 4.9. IB Image Data 作成&出力関数 crwIBBinarySend

##### 【機能】

機器専用のイメージデータを IB コマンド形式のバイナリデータを作成し、  
シリアルポートへ出力します。

##### 【形式】

```
short int crwIBBinarySend(
    short int iPortNo,
    short int iRetryCount,
    short int iDSRCheck,
    short int iCTSCheck,
    short int iWidth,
    short int iHeight,
    short int iStartY,
    LONG lTimeout,
    LONG lComBufferSize,
    LPCSTR strPathFileName,
    BSTR* bpCOMSTS,
    BSTR* bpERRSEN,
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	シリアルポートの番号
<i>iRetryCount</i>	データ送信、NAK 受信、及び SUM エラー受信時のリトライ回数
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ (0: しない/0 以外: する)
<i>iCTSCheck</i>	CTS 信号線をチェックするかどうかのフラグ (0: しない/0 以外: する)
<i>iWidth</i>	印字エリアの横幅 (固定値: 319) ※ 機器の仕様上 319 固定
<i>iHeight</i>	印字エリアの縦幅 (機器の仕様書参考) 例: 459
<i>iStartY</i>	印字開始位置(Y 座標) 例: 100
<i>lTimeout</i>	データの受信待ちタイマー値(単位:msec)
<i>lComBufferSize</i>	COM ポート通信バッファサイズ (機器の通信バッファサイズ合わせてデータを作成し送信します)
<i>strPathFileName</i>	イメージデータファイル名[ BMP, JPG, GIF ] 形式 (フルパス指定可能)
<i>bpCOMSTS</i>	ステータス情報、またはコマンドコードを返却するエリアのポインタ
<i>bpERRSEN</i>	センサー情報、またはエラーコードを返却するエリアのポインタ

##### 【戻り値】

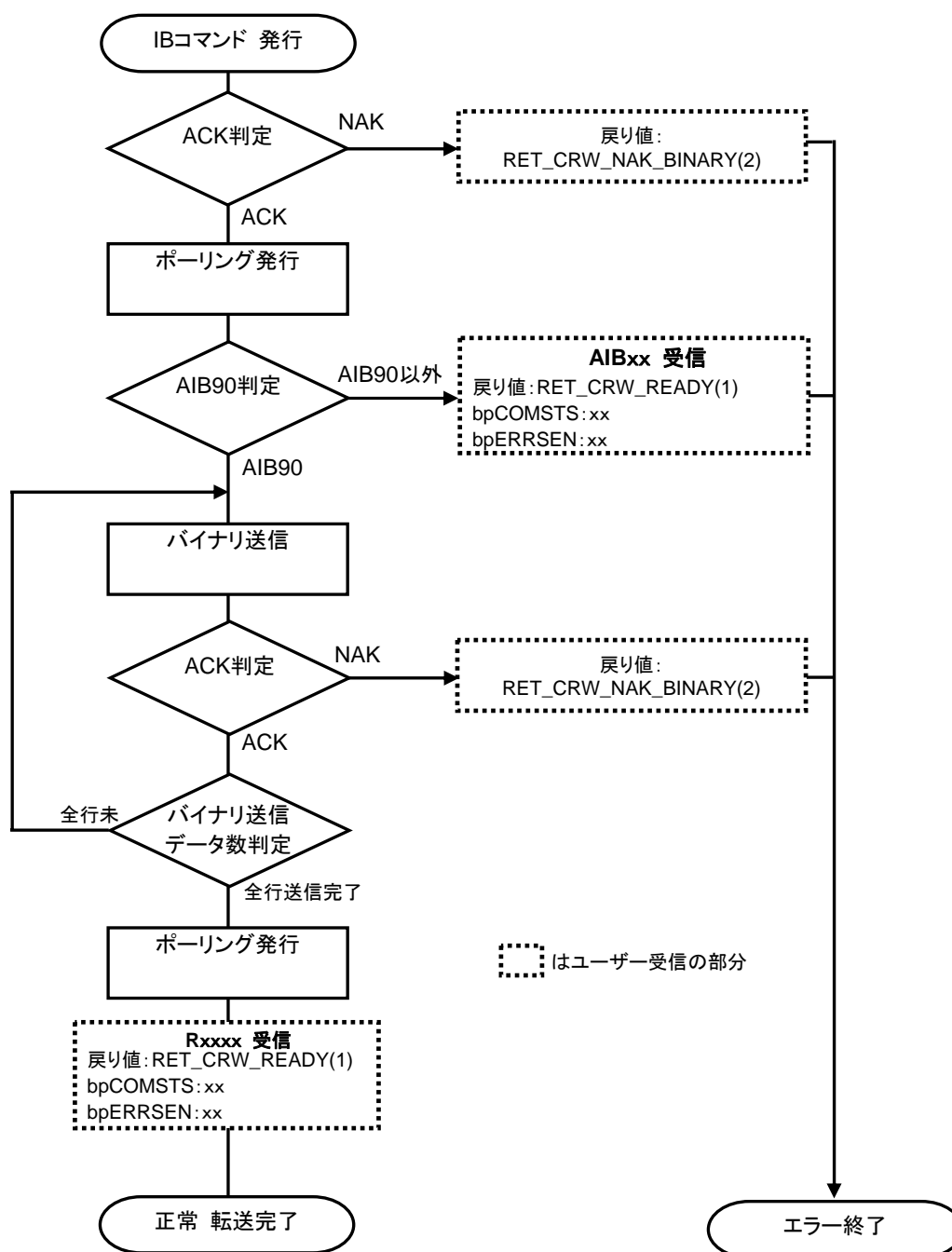
値	説明
RET_CRW_RESPONSE(0)	レスポンスデータ("A")受信
RET_CRW_READY(1)	レスポンスデータ("R")受信
RET_CRW_NAK_BINARY(2)	否定 (NAK) 応答
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	機器からの応答待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー (端末未接続/電源オフ)
RET_CRW_ERR_BCC_RETRY(120)	BCC エラーリトライオーバー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

# 【解説】

“IB” コマンド発行後、“AIB90”を受信判定後、バイナリ送信を開始し全行分送信後ポーリングにて終了いたします。  
 その間NAK応答を受信した場合は、RET\_CRW\_NAK\_BINARY(2)を返却し終了します。

- ・ *iWidth*(印字エリア幅)は機器の仕様上 319 固定でお願いします。
- ・ 転送が正常終了するかエラーが発生しない限り、関数からの応答はありません。
- ・ 画像ファイルの形式は[ BMP, JPG, GIF ] のいずれかにてご指定ください。
- ・ バイナリデータフォーマットはコマンド仕様書をご参照ください。

## ・ IB Image Data 作成&出力関数 (crwIBBinarySend) のフローとエラー時の戻り値



#### 4.10. データ送信関数（応答あり） `crwSendDataWR`

##### 【機能】

渡されたデータを加工せずにシリアルポートへ出力し、機器からの応答を受信します。

##### 【形式】

```
short int crwSendDataWR(  
    short int  iPortNo,  
    LONG      ITimeOut,  
    short int  iDSRCheck,  
    LPCSTR     lpData,  
    short int  iDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	データを送信するシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>ITimeOut</i>	ACK/NAK 待ち及び CTS オン待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ（0 以外：する / 0：しない）
<i>lpData</i>	送信するデータ
<i>iDataLength</i>	送信するデータの長さ

##### 【戻り値】

値	説明
RET_CRW_ACK(0)	肯定（ACK）応答
RET_CRW_NAK(1)	否定（NAK）応答
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	ACK / NAK 待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

##### 【解説】

渡された送信データを一切加工せず、そのままの形で機器に送信します。

その後、機器からの応答を *ITimeOut* 時間まで待ち、制御を APL に返します。

#### 4.11. データ送信関数（応答なし） `crwSendDataQR`

##### 【機能】

渡されたデータを加工せずにシリアルポートへ出力します。ACK/NAK応答は待ちません。

##### 【形式】

```
short int crwSendDataQR(  
    short int  iPortNo,  
    LONG      ITimeout,  
    short int  iDSRCheck,  
    LPCSTR     lpData,  
    short int  iDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	データを送信するシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>ITimeout</i>	CTS オン待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ（0 以外：する / 0：しない）
<i>lpData</i>	送信するデータ
<i>iDataLength</i>	送信するデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR (112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生

##### 【解説】

渡された送信データを一切加工せず、そのままの形で機器に送信します。  
その後、機器からの応答は待たず、送信処理が成功した時点で、制御を APL に返します。



#### 4.12. データ受信関数 `crwReceiveData`

##### 【機能】

シリアルポート受信バッファに格納されているデータを読み込みます。

##### 【形式】

```
short int crwReceiveData(  
    short int  iPortNo,  
    short int  iDSRCheck,  
    BSTR*      bpResDATA  
    short int* piDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	受信するシリアルポートの番号
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ (0 以外:する / 0:しない)
<i>bpResDATA</i>	レスポンスのデータ部分を返却するエリアのポインタ
<i>piDataLength</i>	受信したデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	データ受信あり
RET_CRW_FAILURE(1)	データ受信なし
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_DSR(112)	DSR オフエラー (端末未接続／電源オフ)
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

##### 【解説】

指定されたシリアルポートの受信バッファを読み込んで、何らかのデータが受信されていればそのデータとデータ長を APL に返します。データがバッファに何も無いときは、API の戻り値として RET\_CRW\_FAILURE(1)を返します。

#### 4.13. 電文データ送受信関数 `crwSendDataRR`

##### 【機能】

渡されたデータに STX・ETX を付加し算出した BCC データを付加し、シリアルポートに出力後、レスポンスデータの受信を行う。

電文形式 [STX]+"lpData"+[ETX]+[BCC]

##### 【形式】

```
short int crwSendDataRR(  
    short int  iPortNo,  
    short int  iRetryCount,  
    LONG       ITimeOut,  
    short int  iDSRCheck,  
    LPCSTR     lpData,  
    short int  iDataLength,  
    BSTR*      bpResDATA,  
    short int* piDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	受信するシリアルポート番号
<i>iRetryCount</i>	データ送信・応答データ受信タイムアウト時のリトライ回数
<i>ITimeOut</i>	リトライ時の待ち時間（単位:msec）
<i>iDSRCheckDSR</i>	信号線をチェックするかどうかのフラグ（0 以外:する / 0:しない）
<i>lpData</i>	送信するデータ
<i>iDataLength</i>	送信するデータの長さ
<i>bpResDATA</i>	レスポンスのデータ部分を返却するエリアポインタ
<i>piDataLength</i>	受信したデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	データ受信あり
RET_CRW_FAILURE(1)	データ受信なし
RET_CRW_BUSY(2)	他関数実行中
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	ACK/NAK 待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BCC_RETRY(120)	BCC エラーリトライオーバー
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

##### 【解説】

渡された送信データから次の形式のデータ電文を作成します。

形式：[STX][lpData][ETX][BCC]

これをシリアルポートに出力し、機器からの ACK 応答を受信します。

*ITimeOut* 時間だけ応答がない場合、及び NAK が受信された場合、電文再送リトライを *iRetryCount* 回数行います。

リトライ後もステータスが変化しない場合、最新のステータスを返却します。

ACK を受信したら、機器からの次の形式のデータを受信します。

形式：[STX][bpResDATA][ETX][BCC]

機器からの応答電文の BCC（水平パリティ）を計算し、正しくない場合には、機器に NAK を送信します。

この動作を 3 回繰り返しても正常な BCC が得られない場合、RET\_CRW\_ERR\_BCC\_RETRY(120)を返します。

正常なデータが受信された場合、データ部分を抽出し、機器にACKを送信します。  
その後、データを返却用パラメータにセットして、制御をAPLに返します。

#### 4.14. 電文データ送信関数 crwSendDataBCC

##### 【機能】

渡されたデータにSTX・ETXを付加し算出したBCCデータを付加し、シリアルポートへ出力します。  
ACK/NAKを待ちます  
電文形式 [STX]+"*lpData*"+[ETX]+[BCC]

##### 【形式】

```
short int crwSendDataBCC(  
    short int iPortNo,  
    short int iACKCheck,  
    short int iRetryCount,  
    LONG lTimeOut,  
    short int iDSRCheck,  
    LPCSTR lpData,  
    short int iDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	データを送信するシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>iACKCheck</i>	ACK/NAK を待つかどうかのフラグ（0 以外：する / 0：しない）
<i>iRetryCount</i>	データ送信・ACK/NAK 受信タイムアウト時のリトライ回数
<i>lTimeOut</i>	ACK/NAK 待ち及び CTS ON 待ちタイマー（単位:msec） <i>iACKCheck</i> =0 のときは ACK/NAK 待ちタイマーのみ無効
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ（0 以外：する / 0：しない）
<i>lpData</i>	送信するデータ
<i>iDataLength</i>	送信するデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_NAK(1)	否定（NAK）応答
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	ACK/NAK待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTSオン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSRオフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時のWIN64 APIエラー発生
RET_CRW_ERR_COMM_READ(201)	受信時のWIN64 APIエラー発生

##### 【解説】

渡された送信データから次の形式のデータ電文を作成します。

形式：[STX][*lpData*][ETX][BCC]

これをシリアルポートに出力後、*iACKCheck*=0 以外のとき機器からの ACK を  
*lTimeOut* 時間まで待ち制御を APL に返します。

*iACKCheck* =0 以外の時、機器より NAK を受信したならば、*iRetryCount* 指定数リトライする。

#### 4.15. 電文データ受信関数 `crwReceiveDataBCC`

##### 【機能】

シリアルポート受信バッファに格納されているデータ(STX~ETX+BCC の電文)を読み込みます。  
ACK/NAK を返却します。

##### 【形式】

```
short int crwReceiveDataBCC(  
    short int iPortNo,  
    short int iACKResponse,  
    LONG      ITimeOut,  
    short int iDSRCheck,  
    BSTR*     bpResDATA,  
    short int* piDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	レスポンスを受信するシリアルポートの番号（例：COM1 なら"1"） COM1~COM256 までのシリアルポートをサポート
<i>iACKResponse</i>	ACK/NAK を返却するかどうかのフラグ（0 以外：する / 0：しない）
<i>ITimeOut</i>	CTS ON 待ちタイマー（単位:msec） / <i>iACKResponse</i> =0 のときは無効
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ（0 以外:する / 0:しない）
<i>bpResDATA</i>	レスポンスのデータ部分を返却するエリアのポインタ
<i>piDataLength</i>	受信したデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	データ正常受信あり
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	レスポンス待ちタイムアウト
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BCC_RETRY(120)	BCC エラー / BCC エラーリトライオーバー
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー発生
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー発生

##### 【解説】

受信電文形式： : [STX][*bpResDATA*][ETX][BCC]

指定されたシリアルポートの受信バッファを読み込み、受信電文からデータ部分を抽出（先頭の[STX]、及び末尾の[[ETX][BCC]を除き、*bpResDATA*に格納して制御をアプリケーションに戻します。

*iACKResponse*=0以外のとき機器からの応答電文のBCC（水平パリティ）を計算し、正しくない場合には、機器にNAKを送信します。この動作を3回繰り返しても正常なBCCが得られないときは、エラーを返します。

正常なデータが受信された場合は、データ部分を抽出し、機器にACKを送信し、制御をAPLに戻します。

注-1) 通信バッファにデータが複数たまっていた場合は、  
最後の電文のみを抽出返却し残りは破棄します。

注-2) *iACKResponse*=0（しない）の時、RET\_CRW\_ERR\_BCC\_RETRY(120) にて終了すると  
*bpResDATA*・*piDataLength*にデータを格納します。

#### 4.16. 受信バッファークリア関数 `crwComRXBufferClear`

##### 【機能】

シリアルポート受信バッファに格納されているすべてのデータを初期化（破棄）します。

##### 【形式】

```
short int crwComRXBufferClear(  
                                short int iPortNo  
                                );
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	受信バッファをクリアするシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_FAILURE(1)	クリア失敗
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー

##### 【解説】

シリアルポートの受信バッファのデータを、初期化（破棄）し次回の受信に備えます。  
通常のポーリング通信方式では使用の必要ありません。

#### 4.17. バイナリデータ送信関数QR `crwSendBinaryDataQR`

##### 【機能】

渡されたバイナリデータを一切加工せずシリアルポートへ出力します。（ACK/NAK応答は待たない。）

##### 【形式】

```
short int crwSendBinaryDataQR (  
    short int iPortNo,  
    short int ITimeOut,  
    short int iDSRCheck,  
    short int iCTSCheck,  
    LPCVOID lpData,  
    short int iDataLength  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	シリアルポートの番号
<i>ITimeOut</i>	CTS オン待ちタイマー値（単位：msec）
<i>iDSRCheck</i>	DSR 信号線をチェックするかどうかのフラグ（0：しない／0 以外：する）
<i>iCTSCheck</i>	CTS 信号線をチェックするかどうかのフラグ（0：しない／0 以外：する）
<i>lpData</i>	送信するデータ
<i>iDataLength</i>	送信するデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_CTS_TIMEOUT(111)	CTS オン待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_WRITE(200)	送信時の WIN64 API エラー

##### 【解説】

データをシリアルポートに出力し、制御をAPLに返します。

#### 4.18. バイナリデータ受信関数 `crwReceiveBinaryData`

##### 【機能】

シリアルポート受信バッファに格納されているデータを読み込みます。

##### 【形式】

```
short int  crwReceiveBinaryData(  
                                short int  iPortNo,  
                                LONG        ITimeOut,  
                                short int  iDSRChech,  
                                LPVOID     lpData,  
                                short int  iDataLength,  
                                short int* piDataLength  
                                );
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	送信するシリアルポートの番号
<i>ITimeOut</i>	データの受信待ちタイマー値（単位：msec）
<i>iDSRChech</i>	DSR 信号線をチェックするかどうかのフラグ（0：しない／0 以外：する）
<i>lpData</i>	受信データを返却するエリアのポインタ
<i>iDataLength</i>	受信データを返却するエリアのサイズ
<i>piDataLength</i>	受信したデータの長さ

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー
RET_CRW_ERR_RES_TIMEOUT(110)	データの受信待ちタイムアウト
RET_CRW_ERR_DSR(112)	DSR オフエラー（端末未接続／電源オフ）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中
RET_CRW_ERR_COMM_READ(201)	受信時の WIN64 API エラー

##### 【解説】

指定されたシリアルポートの受信バッファを読み込み、*iDataLength*で指定したバイト数分受信データを抽出して、*lpData*に格納して、制御をアプリケーションに戻します。

`crwReceiveBinaryData`は、最初に*piDataLength*の指す変数を0に設定し、受信データのバッファへの格納が完了すると、実際に読み取ったサイズが設定されます。

*iDataLength*で指定したバイト数受信するか、*ITimeOut*で指定した時間待ってレスポンスを返却することができます。1度に全てのデータが格納できない場合は、RET\_CRW\_ERR\_BUFFEROVERが返却されます。このとき、再度、関数を実行することで格納できなかったデータを格納することができます。

#### 4.19. レスポンス受信イベントハンドラ登録関数 `crwSetResponseEventProc`

##### 【機能】

ポーリングレスポンス受信イベント用のコールバック関数を登録します。

##### 【形式】

```
short int  crwSetResponseEventProc (
                                short int iPortNo,
                                short int iRecMode,
                                LONG      ITimeOut,
                                RES_EVENT_PROC pResponseEventProc,
                                LPARAM    pParam,
                                LONG      IEvent
                                );
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	受信するシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>iRecMode</i>	ビジーステータス受信時にポーリング再送を行うかどうかのフラグ （0以外：する / 0：しない）
<i>ITimeOut</i>	データの受信待ちタイマー値(単位:msec) （0 msec指定で無限待ち）
<i>pResponceEventProc</i>	<i>crwResponseEventProc</i> 関数のポインタ(NULLを指定:登録解除)
<i>pParam</i>	アプリケーションによって定義された値
<i>IEvent</i>	コールバック関数で受け取るイベントの論理和

##### 【戻り値】

値	説明
RET_CRW_SUCCES(0)	登録正常完了／登録正常解除
RET_CRW_FAILURE(1)	同ポート登録済
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	未オープン又はポートが存在しない（登録不可）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中

##### 【IEventの設定値】

値	説明	値(16進数)
EVT_RSP_CRW_RESPONSE	レスポンスデータ("A")受信	0x00000001
EVT_RSP_CRW_READY	レスポンスデータ("R")受信	0x00000002
EVT_RSP_CRW_BUSY	レスポンスデータ("B")受信が継続している	0x00000004
EVT_RSP_CRW_CANCELED	登録解除によるキャンセル通知	0x00000008
EVT_RSP_CRW_ERR_RES_TIMEOUT	デバイスからの応答待ちタイムアウト	0x00000100
EVT_RSP_CRW_ERR_CTS_TIMEOUT	CTS オン待ちタイムアウト	0x00000200
EVT_RSP_CRW_ERR_DSR	DSR オフエラー（端末未接続／電源オフ）	0x00000400
EVT_RSP_CRW_ERR_BCC_RETRY	BCC エラーリトライオーバー	0x00000800
EVT_RSP_CRW_ERR_COMM_WRITE	送信時の WIN64 API エラー発生	0x00001000
EVT_RSP_CRW_ERR_COMM_READ	受信時の WIN64 API エラー発生	0x00002000
EVT_RSP_CRW_ALL	全イベント指定	0xFFFFFFFF



## 【解説】

対象機器から受信したレスポンスを通知し処理するため、コールバック関数の登録／解除を行います。

登録するコールバック関数は、`crwSetResponseEventProc`関数の形式に従うこととします。

コールバック関数の解除は、`pResponseEventProc`にNULLポインタを設定します。

登録解除するとポーリングを中止します。

`pParam`には`crwSetResponseEventProc`関数に実引数として渡す値を指定します。

シリアルポートに対して、次のポーリングコマンドを送信します。

形式：[EOT] [ENQ]

これにより機器から以下のいずれかのデータを受信します。

コマンドレスポンス：

[STX][‘A’][コマンドコード(2バイト)][エラーコード(2バイト)][受信データ(可変長)][ETX][BCC]

レディステータス：

[STX][‘R’][ステータス情報(2バイト)][センサー情報(2バイト)][ETX][BCC]

ビジステータス：

[STX][‘B’][コマンドコード(2バイト)][センサー情報(2バイト)][ETX][BCC]

それぞれの受信形式から、対象とするデータ部分を抽出し、受信データエリアにセットして、それぞれの受信イベントを通知します。

受信したデータは [STX][ETX][BCC] を除いた部分が `lpData` の領域に格納され、データ長は `iDataLength` 領域に格納されます。

受信電文形式：[STX][ `lpData` ][ETX][BCC]

ポーリング再送指定を行った場合に、ビジステータスが継続している状態で、処理を中断し、新たな処理を行いたい場合にはコールバック関数の登録解除にて中断します。

レスポンス受信メソッドで `iRecMode` を指定した場合にポーリング再送を行い、**レディステータス**が**コマンドレスポンス**を受信した場合、またはエラーが発生した場合にイベントを通知します。

また、対象機器からの応答電文のBCC（水平パリティ）を計算し、正しくない場合には、機器にNAKを送信します。この動作を3回繰り返しても正常なBCCが得られないときは、`EVT_RSP_CRW_ERR_BCC_RETRY`を通知します。

`lTimeOut` で指定した時間応答が得られなかった場合、`EVT_RSP_CRW_ERR_RES_TIMEOUT` イベント通知し、CTSがオンにならなかった場合、`EVT_RSP_CRW_ERR_CTS_TIMEOUT`を通知します。

DSRオフエラー（`EVT_RSP_CRW_ERR_DSR`）のイベントを指定しなかった場合、DSRのチェックは行いません。

コールバック関数登録後、イベント受信報告を受信すると、コールバック関数は自動にて解除となります。

`CrwSendCommand`(`crwSendDataQR`)実行後に、登録してください。

例： `crwSendCommsnd` → `crwSetResponseEventProc`

### 【ユーザコールバック関数のプロトタイプ】

```
short int crwResponseEventProc (short int  iPortNo,
                                LONG  IEvent,
                                BSTR  bstrData,
                                short int iDataLength,
                                LPARAM pParam
                                )
{
    switch(IEvent){
        case EVT_RSP_CRW_RESPONSE:
            // レスポンスデータ("A")を受信した。
            break;
        case EVT_RSP_CRW_READY:
            // レスポンスデータ("R")を受信した。
            break;
    }
    return 0;
}
```

### 【ユーザコールバック関数に渡されるパラメータ】

名前	説明
<i>iPortNo</i>	イベントが発生したポートを返す
<i>IEvent</i>	コールバック関数で受け取るイベントの定数
<i>bstrData</i>	受信データを返却するエリアのポインタ
<i>iDataLength</i>	受信したデータの長さ
<i>pParam</i>	アプリケーションによって定義された値

### 【IEventの値】

値	説明	値(16進数)
EVT_RSP_CRW_RESPONSE	レスポンスデータ("A")受信	0x00000001
EVT_RSP_CRW_READY	レスポンスデータ("R")受信	0x00000002
EVT_RSP_CRW_BUSY	レスポンスデータ("B")受信が継続している	0x00000004
EVT_RSP_CRW_CANCELED	登録解除によるキャンセル通知	0x00000008
EVT_RSP_CRW_ERR_RES_TIMEOUT	デバイスからの応答待ちタイムアウト	0x00000100
EVT_RSP_CRW_ERR_CTS_TIMEOUT	CTS オン待ちタイムアウト	0x00000200
EVT_RSP_CRW_ERR_DSR	DSR オフエラー（端末未接続／電源オフ）	0x00000400
EVT_RSP_CRW_ERR_BCC_RETRY	BCC エラーリトライオーバー	0x00000800
EVT_RSP_CRW_ERR_COMM_WRITE	送信時の WIN64 API エラー発生	0x00001000
EVT_RSP_CRW_ERR_COMM_READ	受信時の WIN64 API エラー発生	0x00002000

#### 4.20. 電文データ受信イベントハンドラ登録関数 `crwSetReceiveDataBCCEventProc`

##### 【機能】

電文形式 [STX]+[lpData]+[ETX]+[BCC] 受信イベント用のコールバック関数を登録します。

##### 【形式】

```
short int  crwSetReceiveDataBCCEventProc (
                                short int  iPortNo,
                                short int  iACKResponse,
                                LONG        ITimeOut,
                                RES_EVENT_PROC pReceiveBCCEventProc,
                                LPARAM      pParam,
                                LONG        IEvent
                                );
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	受信するシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>iACKResponse</i>	ACK/NAKを返却するかどうかのフラグ（0：しない / 0以外：する）
<i>ITimeOut</i>	データの受信待ちタイマー値(単位:msec)（0 msec指定で無限待ち）
<i>pReceiveBCCEventProc</i>	<code>crwSetReceiveDataBCCEventProc</code> 関数のポインタ (NULLを指定:登録解除)
<i>pParam</i>	アプリケーションによって定義された値
<i>IEvent</i>	コールバック関数で受け取るイベントの論理和

##### 【戻り値】

値	説明
RET_CRW_SUCCES(0)	登録正常完了／登録正常解除
RET_CRW_FAILURE(1)	同ポート登録済
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	未オープン又はポートが存在しない（登録不可）
RET_CRW_ERR_BUSY(140)	非同期処理を実行中

##### 【IEventの設定値】

値	説明	値(16進数)
EVT_BCC_CRW_RESPONSE	データ正常受信	0x00000001
EVT_BCC_CRW_CANCELED	登録解除による受信キャンセル通知	0x00000008
EVT_BCC_CRW_ERR_RES_TIMEOUT	デバイスからの受信待ちタイムアウト	0x00000100
EVT_BCC_CRW_ERR_CTS_TIMEOUT	CTS オン待ちタイムアウト	0x00000200
EVT_BCC_CRW_ERR_DSR	DSR オフエラー（端末未接続／電源オフ）	0x00000400
EVT_BCC_CRW_ERR_BCC_RETRY	BCC エラーリトライオーバー	0x00000800
EVT_BCC_CRW_ERR_COMM_WRITE	送信時の WIN64 API エラー発生	0x00001000
EVT_BCC_CRW_ERR_COMM_READ	受信時の WIN64 API エラー発生	0x00002000
EVT_BCC_CRW_ALL	全イベント指定	0xFFFFFFFF

## 【解説】

対象機器から受信したレスポンスを通知し処理するため、コールバック関数の登録／解除を行います。

登録するコールバック関数は、`crwSetReceiveDataBCCEventProc`関数の形式に従うこととします。

コールバック関数の解除は、`pReceiveBCCEventProc`にNULLポインタを設定します。

登録解除すると受信街を中止します。

`pParam`には`crwSetReceiveDataBCCEventProc`関数に実引数として渡す値を指定します。

シリアルポートに対して、機器より電文形式のデータ受信します。

これにより機器から以下のいずれかのデータを受信します。

[STX] [lpData 受信データ(可変長)] [ETX] [BCC]

受信データ(可変長)部分を抽出し、受信データエリアにセットして、それぞれの受信イベントを通知します。

受信したデータは [STX][ETX][BCC] を除いた部分が `lpData` の領域に格納され、データ長は `iDataLength` 領域に格納されます。

登録時に `iACKResponse` を “0 以外” に指定した場合にデータ正常受信後ACKを機器に返却します。

また、対象機器からの応答電文のBCC（水平パリティ）を計算し、正しくない場合には、機器にNAKを送信します。この動作を3回繰り返しても正常なBCCが得られないときは、`EVT_RSP_CRW_ERR_BCC_RETRY`を通知します。

`lTimeout` で指定した時間応答が得られなかった場合、`EVT_RSP_CRW_ERR_RES_TIMEOUT` イベント通知し、CTSがオンにならなかった場合、`EVT_RSP_CRW_ERR_CTS_TIMEOUT`を通知します。

DSRオフエラー（`EVT_BCC_CRW_ERR_DSR`）のイベントを指定しなかった場合、DSRのチェックは行いません。

コールバック関数登録後、イベント受信報告を受信すると、コールバック関数は自動にて解除となります。

`crwSendDataBCC` 実行後に、登録してください。

例： `crwSendDataBCC` → `crwSetReceiveDataBCCEventProc`

### 【ユーザコールバック関数のプロトタイプ】

```
short int crwResponseEventProc (short int  iPortNo,
                                LONG  IEvent,
                                BSTR  lpData,
                                short int iDataLength,
                                LPARAM pParam
                                )
{
    switch(IEvent){
        case EVT_BCC_CRW_RESPONSE:
            // データを正常に受信しました。
            break;
        case EVT_BCC_CRW_ERR_CANCELED:
            // ユーザーによる中断。
            break;
    }
    return 0;
}
```

### 【ユーザコールバック関数に渡されるパラメータ】

名前	説明
<i>iPortNo</i>	イベントが発生したポートを返す
<i>IEvent</i>	コールバック関数で受け取るイベントの定数
<i>bstrData</i>	受信データを返却するエリアのポインタ
<i>iDataLength</i>	受信したデータの長さ
<i>pParam</i>	アプリケーションによって定義された値

### 【IEventの値】

値	説明	値(16進数)
EVT_BCC_CRW_RESPONSE	正常データ受信	0x00000001
EVT_BCC_CRW_CANCELED	登録解除による受信キャンセル中止	0x00000008
EVT_BCC_CRW_ERR_RES_TIMEOUT	デバイスからの応答待ちタイムアウト	0x00000100
EVT_BCC_CRW_ERR_CTS_TIMEOUT	CTS オン待ちタイムアウト	0x00000200
EVT_BCC_CRW_ERR_DSR	DSR オフエラー（端末未接続／電源オフ）	0x00000400
EVT_BCC_CRW_ERR_BCC_RETRY	BCC エラーリトライオーバー	0x00000800
EVT_BCC_CRW_ERR_COMM_WRITE	送信時の WIN64 API エラー発生	0x00001000
EVT_BCC_CRW_ERR_COMM_READ	受信時の WIN64 API エラー発生	0x00002000

#### 4.21. 通信イベントハンドラ登録関数 `crwSetCommEventProc`

##### 【機能】

通信イベント用のコールバック関数を登録します。

##### 【形式】

```
short int  crwSetCommEventProc (
                                short int  iPortNo,
                                COMM_EVENT_PROC pCommEventProc,
                                LPARAM      pParam,
                                LONG         lEvent
                                );
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	監視するシリアルポートの番号（例：COM1なら"1"） COM1～COM256までのシリアルポートをサポート
<i>pCommEventProc</i>	crwCommEventProc関数のポインタ(NULLを指定:登録解除)
<i>pParam</i>	アプリケーションによって定義された値
<i>lEvent</i>	コールバック関数で受け取るイベントの論理和

##### 【戻り値】

値	説明
RET_CRW_SUCCES(0)	登録正常完了／登録正常解除
RET_CRW_FAILURE(1)	同ポート登録済
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポートが存在しない（登録不可）

##### 【lEventの設定値】

値	説明	値(16進数)
EVT_CRW_RXCHAR	入力バッファにデータを受信した。	0x00000001
EVT_CRW_TXEMPTY	出力バッファの最後の文字を送信した。	0x00000004
EVT_CRW_ERR_FRAME	フレーミングエラーが発生した。	0x00000100
EVT_CRW_ERR_OVERRUN	オーバーランエラーが発生した。	0x00000200
EVT_CRW_ERR_PARITY	パリティエラーが発生した。	0x00000400
EVT_CRW_CTS	CTS (送信可) 信号の状態が変わった。	0x00001000
EVT_CRW_DSR	DSR (データセットレディ) 信号の状態が変わった。	0x00002000
EVT_CRW_RLSD	RLSD (受信線信号検出)信号の状態が変わった。	0x00004000
EVT_CRW_BREAK	Break 信号を検出した。	0x00008000
EVT_CRW_RING	呼び出し信号を検出した。	0x00010000
EVT_CRW_ALL	全イベント指定	0xFFFFFFFF

##### 【解説】

操作対象の機器内で発生したデータ通信イベント報告を処理するため、コールバック関数の登録／解除を行います。

登録するコールバック関数は、`crwCallBackCommEventProc`関数の形式に従うこととします。コールバック関数の解除は、`pCommEventProc`にNULLポインタを設定します。`pParam`には`crwCommEventProc`関数に実引数として渡す値を指定します。

コールバック関数登録後は、解除するまで、イベント監視を続行します。

### 【ユーザコールバック関数のプロトタイプ】

```
short int crwCallBackCommEventProc(short int  iPortNo,
                                   LONG      IBufferLength,
                                   LONG      IEvent,
                                   LPARAM    pParam
                                   )
{
    switch(IEvent){
        case EVT_CRW_RXCHAR:
            // 入力バッファにデータを受信。
            break;
        case EVT_CRW_CTS:
            // CTS信号の状態が変化。
            break;
    }
    return 0;
}
```

### 【ユーザコールバック関数に渡されるパラメータ】

名前	説明
<i>iPortNo</i>	イベントが発生したポートを返す
<i>IBufferLength</i>	EVT_CRW_RXCHARイベントが発生したときの入力バッファのサイズ
<i>IEvent</i>	コールバック関数で受け取るイベントの定数
<i>pParam</i>	アプリケーションによって定義された値

### 【IEventの値】

値	説明	値(16進数)
EVT_CRW_RXCHAR	入力バッファにデータを受信した。	0x00000001
EVT_CRW_TXEMPTY	出力バッファの最後の文字を送信した。	0x00000004
EVT_CRW_ERR_FRAME	フレーミングエラーが発生した。	0x00000100
EVT_CRW_ERR_OVERRUN	オーバーランエラーが発生した。	0x00000200
EVT_CRW_ERR_PARITY	パリティエラーが発生した。	0x00000400
EVT_CRW_CTS	CTS (送信可) 信号の状態が変わった。	0x00001000
EVT_CRW_DSR	DSR (データセットレディ) 信号の状態が変わった。	0x00002000
EVT_CRW_RLSD	RLSD (受信線信号検出)信号の状態が変わった。	0x00004000
EVT_CRW_BREAK	Break 信号を検出した。	0x00008000
EVT_CRW_RING	呼び出し信号を検出した。	0x00010000

#### 4.22. 通信中DSR/CTS信号監視設定関数 `crwComDSRCTSsw`

##### 【機能】

通信中の DSR・CTS の監視を行うかどうかの設定を行います(初期値：する)

##### 【形式】

```
short int crwComDSRCTSsw(  
    short int iPortNo ,  
    short int iDSRCheck ,  
    short int iCTSCheck  
);
```

##### 【パラメータ】

名前	説明
<i>iPortNo</i>	DSR・CTS フローを監視するシリアルポートの番号（例：COM1 なら"1"） COM1～COM256 までのシリアルポートをサポート
<i>iDSRCheck</i>	通信中の DSR 信号線をチェックするかどうかのフラグ (0:しない／0 以外:する) 初期値：0 以外:する
<i>iCTSCheck</i>	通信中の CTS 信号線をチェックするかどうかのフラグ (0:しない／0 以外:する) 初期値：0 以外:する

##### 【戻り値】

値	説明
RET_CRW_SUCCESS(0)	正常終了
RET_CRW_ERR_PARAM(100)	パラメータエラー
RET_CRW_ERR_PORT(101)	ポート未オープンエラー

##### 【解説】

通信中のDSR/CTSの監視のON/OFFを行います。

他の関数は、関数実行時のみにチェックを行いますが、通信中チェックをおこないません。

`CrwOpenPort()`関数、実行直後の設定を推奨します。

##### 【注記】

- ・ 通常は初期値のままでご使用をお願いします。  
(本仕様書での適用機種では、初期値：する のままご使用ください)
- ・ 物理的または信号線が、真に存在しない機器のみチェックを（0:しない）の選択をお願いします。
- ・ 他の各関数とのチェックの同期は取れておりません。  
(本関数でチェックを(0:しない)する必要がある場合は、他の通信関数でも、DSR・CTSとも(0:しない)に選択しないと関数実行エラーで返却されます。)

関連関数：`crwOpenPort`



#### 4.23. 文字コード変換関数 `crwCharCodeConv`

##### 【機能】

文字列⇔ASCII / 文字列⇔UTF-16 コードの文字への変換を行います。

##### 【形式】

```
BSTR crwCharCodeConv(  
                                short int  iConvMode,  
                                LPCSTR     csQConvStr  
                                );
```

##### 【パラメータ】

名前	説明
<i>iConvMode</i>	0 = char ⇒ ASCII ("0" -> "30") 1 = ASCII ⇒ char ("30" -> "0") 2 = char ⇒ UTF-16 ("A" -> "0041") 3 = UTF-16 ⇒ char ("0041" -> "A")
<i>csQConvStr</i>	変換するデータ

##### 【戻り値】

値	説明
変換されたデータ	変換できないときはNullが返却されます。

#### 4.24. JIS/Shift-JIS漢字コード変換関数 `crwJISconvSJIS`

##### 【機能】

漢字コード JIS ⇔ Shift-JIS の変換を行います。

##### 【形式】

```
BSTR crwJISconvSJIS (  
                                short int  iJConvMode,  
                                LPCSTR     csQConvStr  
                                );
```

##### 【パラメータ】

名前	説明
<i>iJConvMode</i>	0 = JIS ⇒ Shift-JIS("4267" -> "91E5") = "大" 0 以外 = Shift-JIS ⇒ JIS("91E5" -> "4267") = "大"
<i>csQConvStr</i>	変換するデータ

##### 【戻り値】

値	説明
変換されたデータ	変換できないときはNullが返却されます。

## 5. iniファイルの構成

iniファイルはアプリケーションとDLLと同じフォルダに置いてください。  
iniファイルに記述されている項目は以下の通りです。

```
[Log]
Logging = -1
LogFileName = .¥¥PcardRW64_¥03d.log
[Delay]
Delay = 0
[TimeOut]
WriteFileTimeOut = 5000
```

Logging	<p>ログファイルを出力するかどうかを決定します。</p> <p>-1: ログを出力しない 0: ログファイルを出力する (エラーと各種情報) 1: ログファイルを出力する (エラーのみ) 2: ログをデバッガーに出力する (エラーと各種情報) 3: ログをデバッガーに出力する (エラーのみ)</p> <p>※デフォルトは -1 -1, 0, 1, 2, 3 以外の値が設定されている場合は、-1 が指定されているものとします。 ※デバッガー出力 OutputDebugString APIにて出力します。DebugView等で取得してください。</p>
LogFileName	<p>出力するログファイル名のフォーマット</p> <p>LogFileName = [ Path ]+[ FileName ]+[ ¥03d ]+[ ¥. ]+[ 拡張子 ] [ ¥03d ] = 通信ポート出力指定 (3桁出力の場合)</p> <p>例: LogFileName = C:¥¥Log¥¥PcDLLtest_¥02d.logで指定した場合 ポートCOM8にて通信を行うと、ログがCドライブのLogフォルダのPcDLLtest_08.logに出力されます。エスケープシーケンス¥¥は、プログラム内では文字として扱われないので、¥¥と指定します。</p> <p>※デフォルトは “.¥¥PcardRW64_¥03d.log” (実行アプリケーションと同フォルダに出力されます)</p>
Delay	<p>データ送信前に挿入する遅延時間 (単位はmsec)</p> <p>※デフォルトは 0 数値以外の値が設定されている場合は、0 が指定されているものとします。</p>
WriteFileTimeOut	<p>WriteFile API を実行した際に処理が戻るまでの最大待ち時間 (単位はmsec)</p> <p>※デフォルトは 5000 数値以外の値が設定されている場合は、5000 が指定されているものとします。</p>

※1 iniファイルが存在しない場合は、デフォルトの値が設定されているものとして動作します。

※2 iniファイルは以下の優先順位で読み込みます。いずれにもiniファイルが存在している場合は優先順位の高い方のiniファイルを読み込みます。

優先順位

1. カレントディレクトリ (アプリケーションと同じフォルダ)
2. Windowsのディレクトリ(%WinDir% 以下)

## 6. ログファイルの構成

出力されるログファイルは以下のようになります。

```

2013:01:13:11:08:37: 90: 3232: 3292:Inf:Info    : DLL Version = 1, 0, 0, 0
2013:01:13:11:08:37:100: 3232: 3292:Inf:Enter   : crwOpenPort
2013:01:13:11:08:37:100: 3232: 3292:Inf:CSLock   : crwOpenPort
2013:01:13:11:08:37:131: 3232: 3292:Inf:Info    : PortName is COM1
2013:01:13:11:08:37:131: 3232: 3292:Inf:Info    : PortDesc   : 通信ポート
2013:01:13:11:08:37:131: 3232: 3292:Inf:Info    : PortDriver : Microsoft 7-1-2012 1.1.2600.0
2013:01:13:11:08:37:141: 3232: 3292:Inf:Info    : Return Value is 0
2013:01:13:11:08:37:141: 3232: 3292:Inf:CSUnlock: crwOpenPort
2013:01:13:11:08:37:141: 3232: 3292:Inf:Exit    : crwOpenPort
2013:01:13:11:08:37:151: 3232: 3292:Inf:Enter   : crwReceiveResponse
2013:01:13:11:08:37:161: 3232: 3292:Inf:CSLock   : crwReceiveResponse
  Addr : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
-----
0000 : 04 05                                     J |

  Addr : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
-----
0000 : 02 52 30 30 31 30 03 50                   7 R0010 LP
  
```

2013:01:11:11:08:37:100: 3232: 3292:Inf:Enter : crwOpenPort  
                   (1)                  (2)                  (3)          (4)          (5)          (6)                  (7)

- (1) 日付 (yyyy:mm:dd)  
ログファイルに書き込まれた日付が記録されています。
- (2) 時刻 (hh:mm:ss:SSS)  
ログファイルに書き込まれた時刻が記録されています。
- (3) プロセスID  
ログファイルに書き込んだプロセスのIDが記録されています。
- (4) スレッドID  
ログファイルに書き込んだスレッドのIDが記録されています。
- (5) ログの種類1 (Inf: 情報 / Err: エラー)  
書き込まれたログの種類が記録されています。  
 Inf        通常情報 (関数進入、関数脱出、パラメータ出力など)  
 Err       エラー情報 (エラーコードの出力)
- (6) ログの種類2  
書き込まれたログの詳細が記録されています。  
 Enter        関数進入時  
 Exit        関数脱出時  
 CSLock      クリティカルセクションのロックを取得時  
 CSUnlock    クリティカルセクションのロックを解放時  
 Param      パラメータ出力  
 Info        各種情報  
 ErrInfo     エラー情報
- (7) その他  
進入 (脱出) したときの関数名、パラメータの値、エラーコードとエラー内容などがここに記述されます。

(7-1) イベントログ詳細

DLL内にて発生時のイベントのログについて説明します。

Event Notification: 0x00000004 (0,0)

① ② ③

① イベントコード詳細

イベントコード		説明
0x00000001	EV_RXCHAR	Any Character received
0x00000002	RXFLAG	Received certain character
0x00000004	TXEMPTY	Transmitt Queue Empty
0x00000008	EV_CTS	CTS changed state
0x00000010	EV_DSR	DSR changed state
0x00000020	EV_RLSD	RLSD changed state
0x00000040	EV_BREAK	BREAK received
0x00000080	EV_ERR	Line status error occurred
0x00000100	EV_RING	Ring signal detected
0x00000200	EV_PERR	Printer error occured
0x00000400	EV_RX80FULL	Receive buffer is 80 percent full
0x00008000	EV_EVENT1	Provider specific event 1
0x00010000	EV_EVENT2	Provider specific event 2

② イベント発生前の受信バッファ中(Windows管理)残データ数

③ イベント発生後の受信バッファ中(Windows管理)残データ数

※注記

ログが書き込まれるタイミングとして、関数を呼び出した段階でEnterログが書き込まれます。そのため、マルチスレッド環境において、Enterログが続けて出力されることがあります。

また、不正なポート番号（1～256以外の番号）が指定された場合、ログファイルは出力されません。

## 7. DLL使用上の注意事項

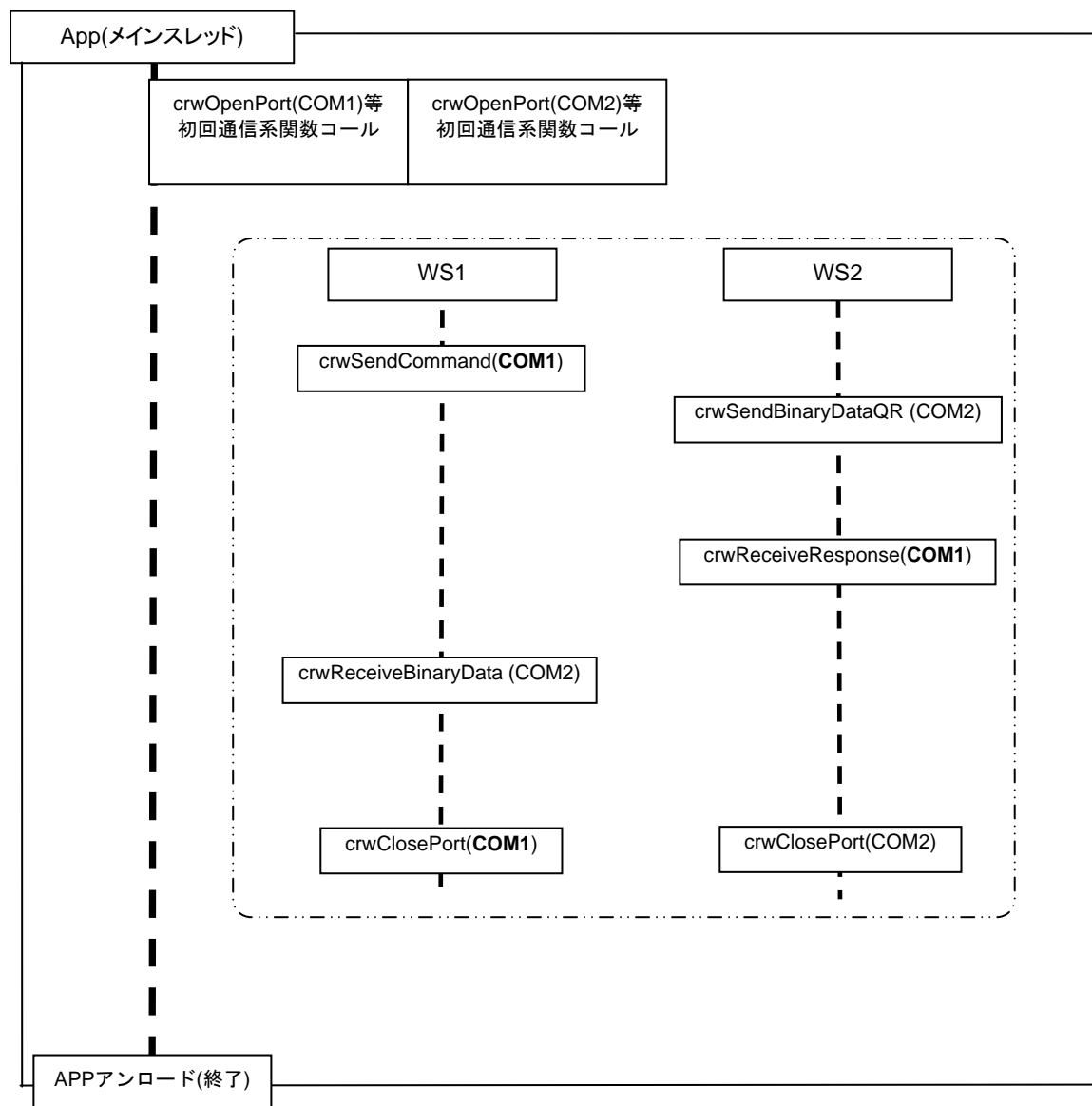
本DLLを使用するにあたり、特に注意が必要な点については以下のとおりです。

1. 本DLLの送受信で扱えるデータはテキストデータ(キャラクタコード0x00から0x1fまでの文字および0x7fの文字を含まないデータ)であり、バイナリデータを含むデータの送受信を行った場合、送受信したデータが化ける場合があります。  
ここでの「送受信で扱えるデータ」とは、本DLLと上位アプリケーションがやり取りするデータであり、メソッド内部あるいは外部機器のやり取りで使用する、ACK、NAK、EOT、STX、ETX、ENQ、BCCなどのデータについては問題ありません。  
上位アプリケーションからDLLを呼び出すときに渡した送信データにテキストデータ以外のコードを含む場合及び、上位アプリケーションがDLLを呼び出して受信したデータにテキストデータ以外のデータを含む場合にデータが化ける場合があります。
2. 本DLLにて1度に2Kバイト以上のデータを送受信しようとすると、データ落ちが発生する可能性があります。(さらに1度に送受信可能なデータサイズは機器側のコマンド使用によって制限されます。)
3. コマンド送信時の伝送エラーに関するリトライ処理は、本DLLのcrwSendCommand()関数内ではサポートされておりません。crwSendCommand()、crwSendCommandRR()関数で、NAK応答やタイムアウトエラーが発生した場合は、上位ソフトにてレスポンス受信crwReceiveResponse()関数でレディ状態を再度確認した後、コマンドの再送を3回程度行うようにプログラムを作成してください。
4. 本DLLを使用するアプリケーションにおいては、BSTR\*型の文字列の取り扱いにご注意ください。本DLLではBSTR型の文字コードにANSIコードを使用しています。
5. 同一のプロセスにおいて、複数のシリアルポートを同時にオープンすることができます。
6. 同一のプロセスにおいて、複数の関数を非同期に(並行に)呼び出すことが可能です。同一ポートへのそれぞれの処理(ポートのオープン、データの送受信、ポートのクローズ)を異なるスレッドで行うことも可能です。ただし同一ポートへの同時関数呼び出しは、そのポートにおける実行中の関数がある場合はその処理が終わるまで待たされます。
7. 弊社、仮想COMドライバー(PSN Virtual COM)と併用する場合には、下記の点を踏まえ設計願います。
  - ・ ポートオープン中はUSBケーブルの抜き差しは基本的に行わないでください。  
(通信が必要ないときは、crwClosePort()関数にてクローズ処理を行うことをお勧めします)
  - ・ ポートオープン中USBケーブルが突然切断された状態になった時に、“PSN Virtual COM”はBreak信号検出”の信号を返却してきますので、  
通信イベントハンドラ登録関数(crwSetCommEventProc)にて“EVT\_CRW\_BREAK”を取得許可するよう実装し、このときにポートクローズ関数(crwClosePort)にて、即座に対象ポートをクローズするように実装してください。  
(ポートがオープンのままでは、ドライバーがアンロード又はリロードできない場合があります)  
処理手順：  

USBケーブル切断	⇒	ブレーク信号発生(イベント取得)	⇒	ポートクローズ処理
-----------	---	------------------	---	-----------
  - ・ 通信イベントハンドラ登録関数を使用しない場合、ポートのオープン中にUSBケーブルの挿抜等が発生した場合  
RET\_CRW\_ERR\_DSR(112)、またはRET\_CRW\_ERR\_RES\_TIMEOUT(110)が発生するようになり、送信・受信共にできなくなってしまう。  
この状態を解消するためには、一旦ポートをクローズする必要があります。  
(仮想COMのドライバーバージョンは、最新版で設計してください。)
8. データ送信時には、COM受信バッファはクリアされません。  
ごみデータが残っていると思われる時には、受信バッファークリア関数(crwComRXBufferClear)にて、データ送信前にクリアするか、ダミー受信を行うことを推奨いたします。

## 8. 補足資料

### 8.1. マルチスレッドの基本使用方法



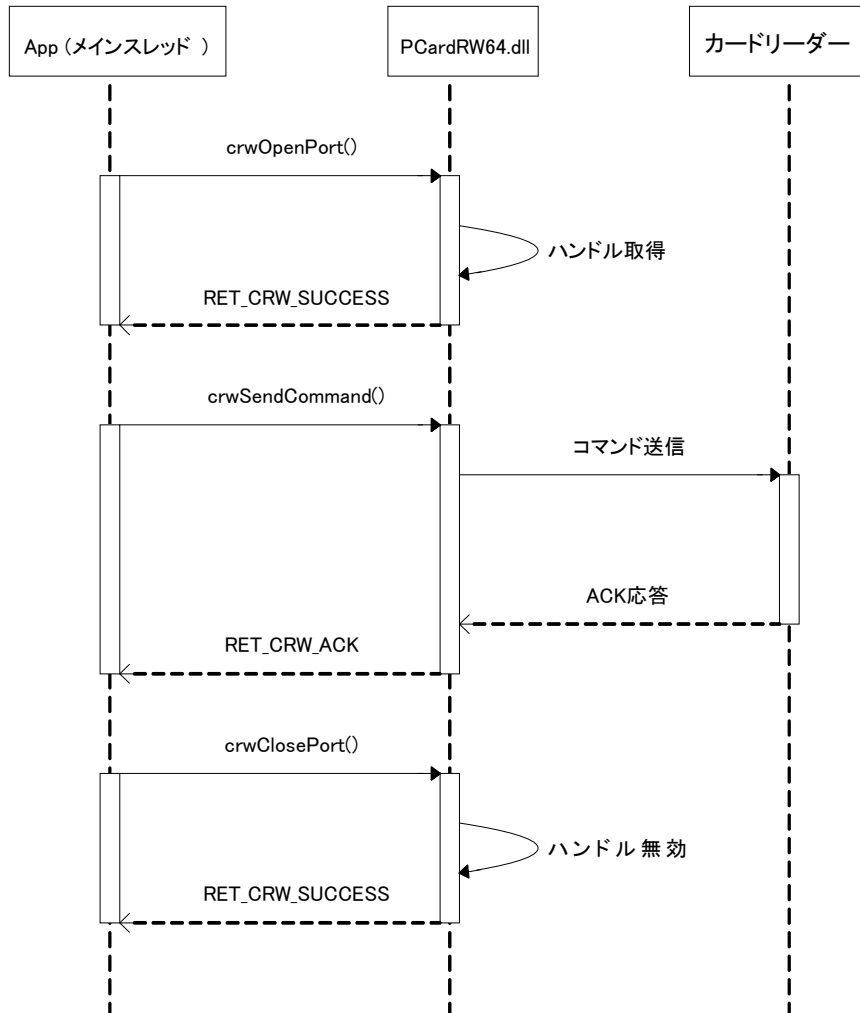
注意:各ポートに対して初回の PCardRW64.DLL 関数呼び出し時に MainThread にて、通信系関数(通常は crwOpenPort)を、コールするように設計してください。

## 8.2. マルチスレッドの動作説明

マルチスレッドの前提として、以下の二つがあります。

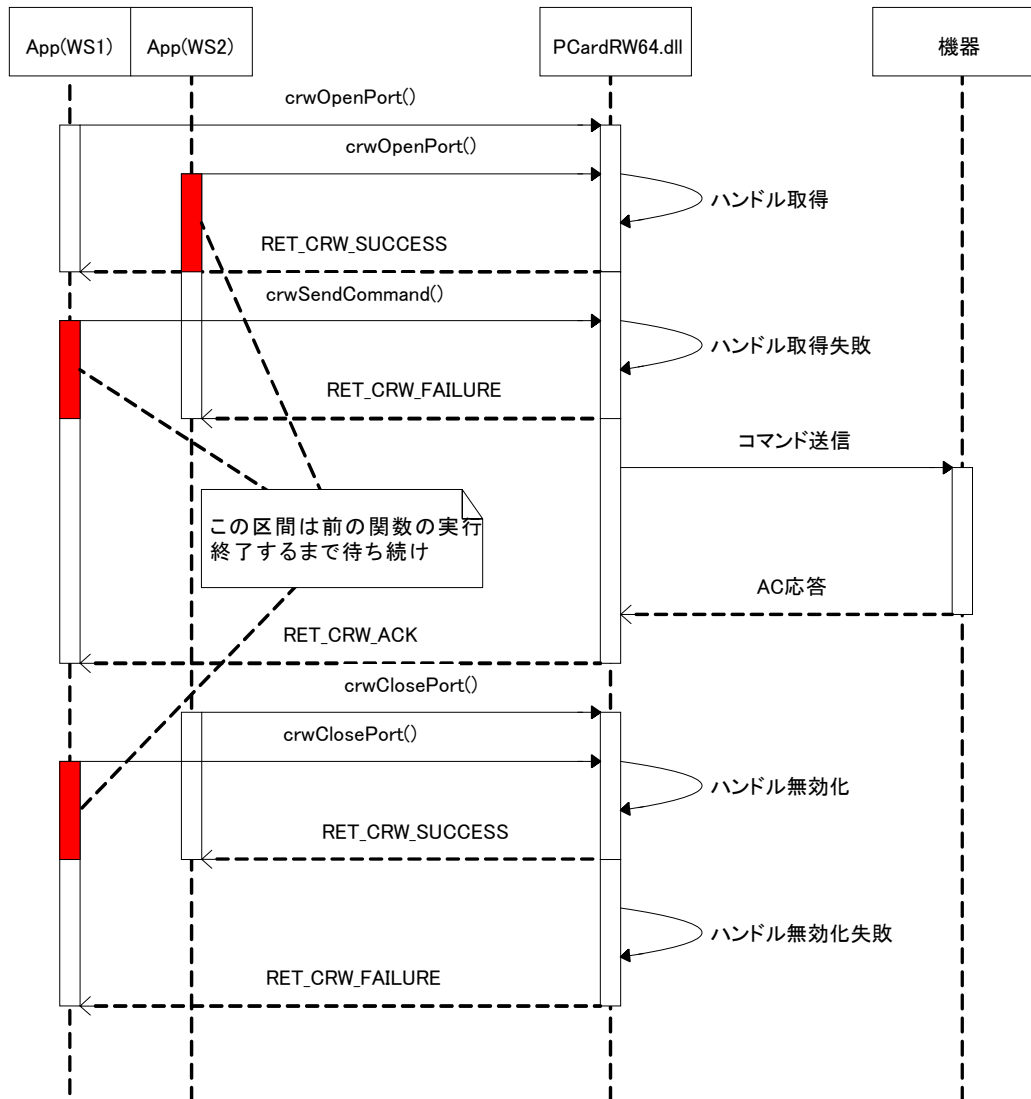
- (1) 同ポートに関する関数呼び出しは、一つの関数処理が終了するまで、別の関数の実行を待機します。
  - (2) 別ポートに関する関数呼び出しは、並列に動作します。
- 以下の項目では、それぞれの事例について説明します。

### 8.2.1. シングルスレッド（1ポート使用）の場合



各関数は同期的に実行されます。

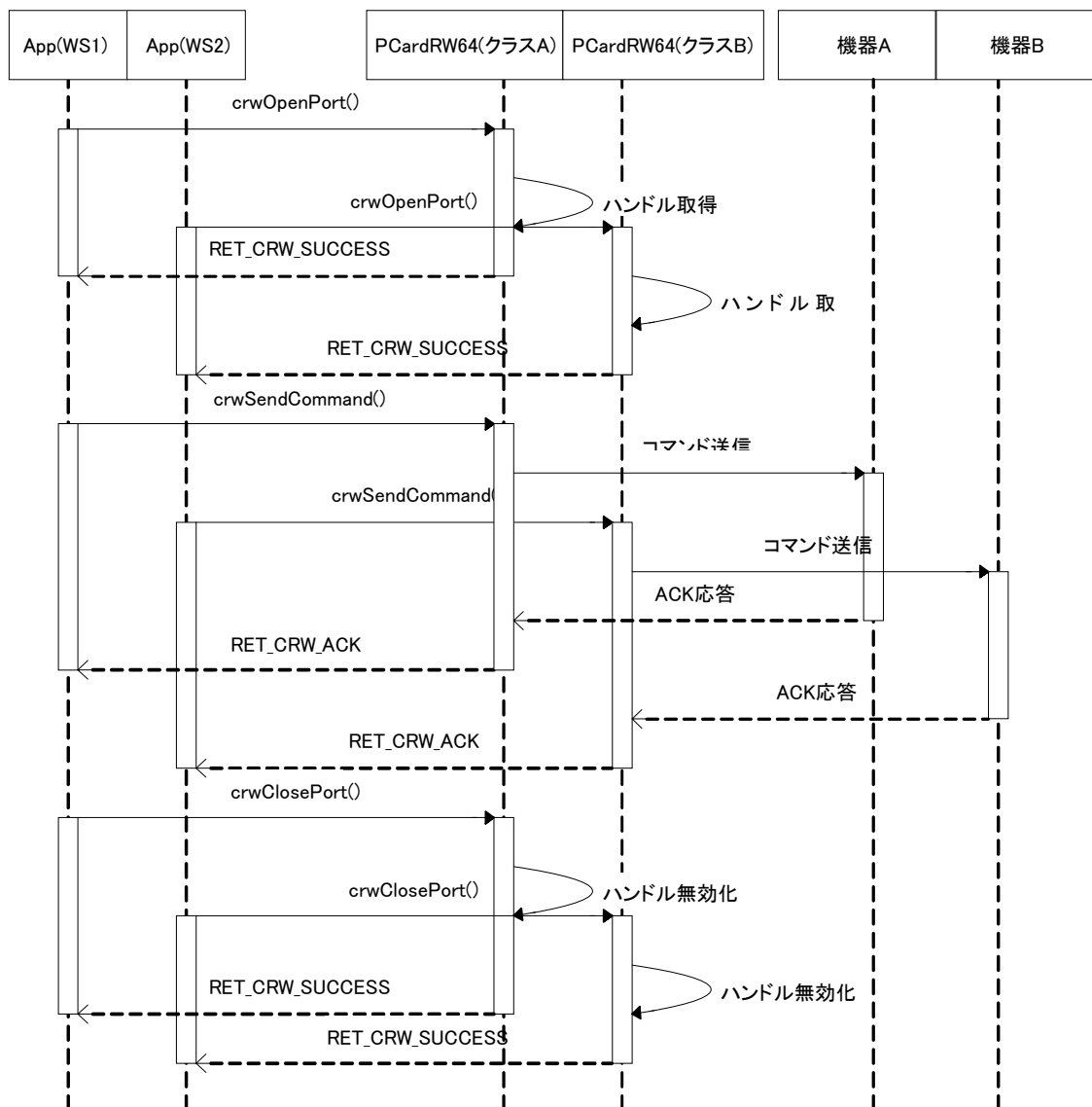
## 8.2.2. マルチスレッド（同一ポート使用）の場合



複数のワーカースレッド（WS）から関数の呼び出しが行われますが、同じポートに対して、関数は同時実行することができないため、呼び出しから実行まで待ち時間が発生する場合があります。図の例では、WS1から`crwOpenPort`関数が最初に呼ばれて実行されます。このとき、WS2から同一ポートに対して、`crwOpenPort`関数を実行した場合、先に実行していた関数の終了を待ってから関数を実行するため、待ち時間が発生します。



### 8.2.3. マルチスレッド（別ポート使用）の場合



複数のワーカースレッド（WS）から関数の呼び出しが行われますが、それぞれが別のポートにアクセスしている場合は、それぞれの関数を並列に実行することができます。<sup>2</sup>  
 図の例では、WS1とWS2がそれぞれcrwOpenPort、crwSendCommand、crwClosePort関数を実行していますが、アクセスしているポートが異なるため、並列に実行することができます。

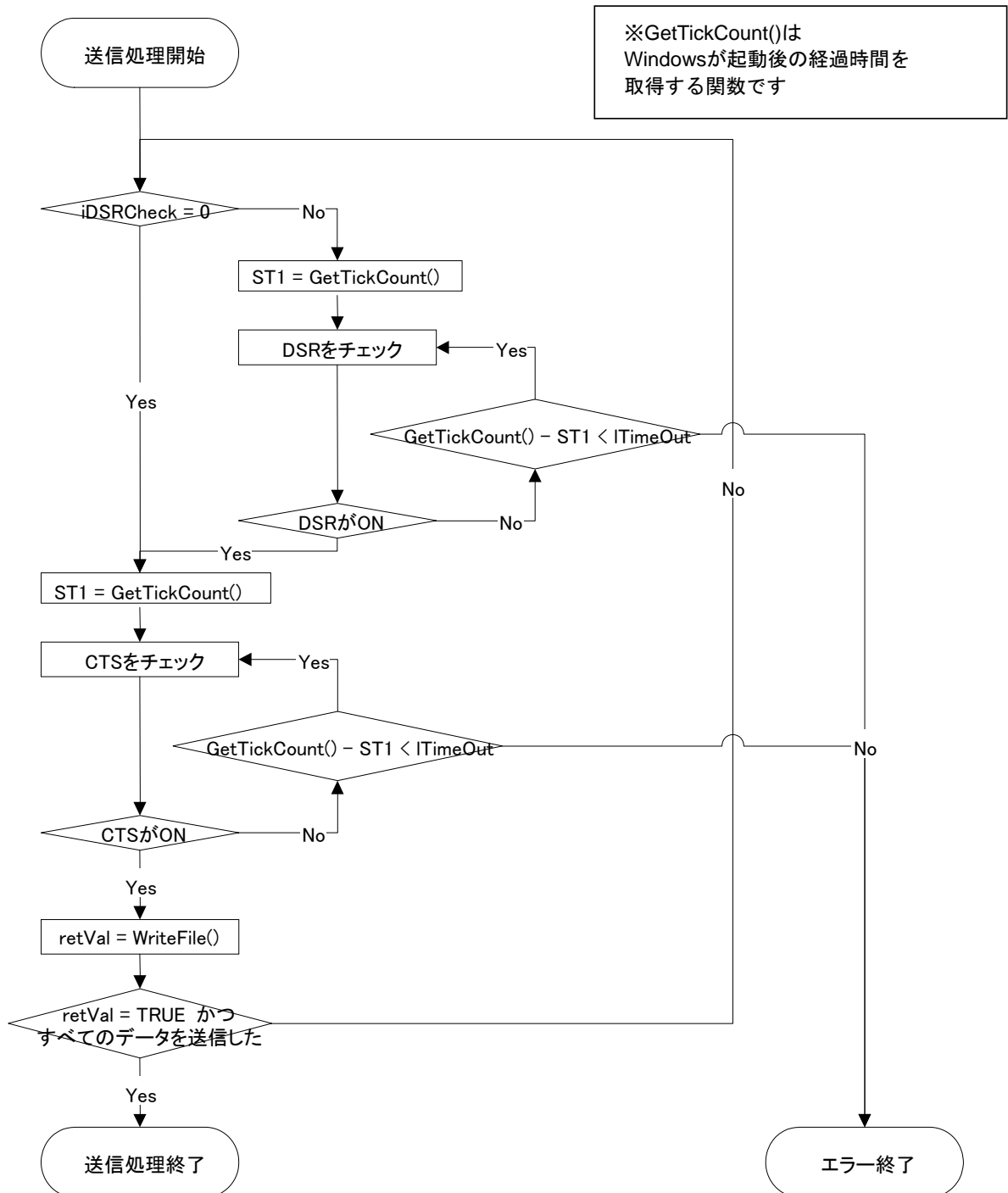
#### 1.1.1.

<sup>2</sup> DLL内部でポート別に並列処理が可能な実装となっております。図内のDLLクラスA,Bはあくまで内部の実装の記載ですので、DLL呼び出し側は通常の間数呼び出しを行うだけです。

### 8.3. データ送受信処理について

ここでは、crwSendCommand()関数を例に送信処理・受信処理のフローチャートを次に示します。

#### 8.3.1. 送信処理



### 8.3.2. 受信処理

