REX-PCI70D

RS-422A/485・デジタル I/O PCI ボード

ユーザーズマニュアル

2015 年 10 月

第 4.0 版

ラトックシステム株式会社

第1章 はじめに ---------- 1- 1
(1-1) 製品仕様 1- 1
(1-2) 添付品 1- 3
(1-3) ブラケットの取替え 1- 3
(1-4) コネクタピンアサイン 1- 4
(1-5) RS-422A/485 の切換えと終端抵抗について 1- 6
(1-6) レジスタセット 1-10
(1-7) DIO ポートについて 1-11
(1-8) DIP スイッチについて 1-12
(1-9) RS-422A/485 の特性について 1-13

第 2 章 Windows セットアップ ---------- 2- 1
(2-1) Windows 10/8.1/8/7/Vista/2000 Server2008/2000Server セットアップ 2- 1
(2-2) Windows XP/Server2003 セットアップ 2- 3
(2-3) インストールの確認 2- 8
(2-4) COM ポート番号の変更と設定について 2- 9
(2-5) ドライバのアンインストール 2-10

第 3 章 DIO 制御・設定ポート確認用ライブラリ関数 ---------- 3- 1
(3-1) ライブラリ関数について 3- 1
(3-2) 関数仕様 3- 1
(3-3) DIO 制御・設定ポート確認サンプルアプリケーションの構成 3-13
(3-4) ライブラリ関数の呼び出し 3-14
(3-5) DIO 制御サンプルアプリケーションについて 3-19
(3-6) 設定ポート確認サンプルアプリケーションについて 3-20

第 4 章 通信サンプルアプリケーション ---------- 4- 1
(4-1) 通信サンプルアプリケーションの構成について 4- 1
(4-2) 通信サンプルアプリケーションについて 4- 1

安全にご使用いただくために

本製品は安全に充分配慮して設計を行っていますが、誤った使い方をすると火災や感電などの事故につながり大変危険です。ご使用の際は、警告/注意事項を必ず守ってください。

表示について

この取扱説明書は、次のような表示をしています。表示の内容をよく理解してから本文をお読みください。

⚠警告　この表示を無視して誤った取扱いをすると、火災や感電などにより、人が死亡または重傷を負う可能性がある内容を示しています。

⚠注意　この表示を無視して誤った取扱いをすると、感電やその他の事故により、人が負傷または物的損害が発生する可能性がある内容を示しています。

⚠警告

- 製品の分解や改造などは、絶対に行わないでください。
- 無理に曲げる、落とす、傷つける、上に重い物を載せることは行わないでください。
- 製品が水・薬品・油などの液体によって濡れた場合、ショートによる火災や感電の恐れがあるため使用しないでください。

⚠注意

- 本製品は電子機器ですので、静電気を与えないでください。
- ラジオやテレビ、オーディオ機器の近く、モーターなどのノイズが発生する機器の近くでは誤動作することがあります。必ず離してご使用ください。
- 高温多湿の場所、温度差の激しい場所、チリやほこりの多い場所、振動や衝撃の加わる場所、スピーカなどの磁気を帯びた物の近くで保管しないでください。
- 煙が出たり異臭がする場合は、直ちにパソコンや周辺機器の電源を切り、電源ケーブルもコンセントから抜いてください。
- 本製品は、医療機器、原子力機器、航空宇宙機器、輸送機器など人命に関わる設備や機器、及び高度な信頼性を必要とする設備や機器での使用は意図されておりません。これらの設備、機器制御システムに本製品を使用し、本製品の故障により人身事故/火災事故/その他の障害が発生した場合、いかなる責任も負いかねます。
- 取り付け時、鋭い部分で手を切らないように、十分注意して作業を行ってください。
- 配線を誤ったことによる損失、逸失利益などが発生した場合でも、いかなる責任も負いかねます。

その他のご注意

- 本書の内容に関して、将来予告なしに変更することがあります。
- 本書の内容につきましては万全を期して作成しておりますが、万一不審な点や誤りなどお気づきになりましたらご連絡お願い申し上げます。
- 本製品の運用を理由とする損失、逸失利益などの請求につきましては、いかなる責任も負いかねますので、予めご了承ください。
- 製品改良のため、将来予告なく外観または仕様の一部を変更する場合があります。
- 本製品は日本国内仕様となっており、海外での保守及びサポートは行っておりません。
- 本製品を廃棄するときは地方自治体の条例に従ってください。条例の内容については各地方自治体にお問い合わせください。
- 本製品の保証や修理に関しましては、添付の保証書に内容を明記しております。必ず内容をご確認の上、大切に保管してください。
- “REX”は株式会社リコーが商標権を所有しておりますが、弊社はその使用許諾契約により本商標の使用が認められています。
- Windowsは米国マイクロソフト社の米国およびその他の国における登録商標です。その他本書に記載されている商品名/社名などは、各社の商標または登録商標です。 なお本書では、™、Ⓡ マークは明記しておりません。

【電波障害自主規制について】

この装置は、クラスＢ情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。
取扱説明書に従って正しい取り扱いをして下さい。

ＶＣＣＩ－Ｂ

第1章 はじめに

(1-1) 製品仕様

REX-PCI70D は、シリアルコントローラに 16550 互換 UART を搭載し、D-Sub15 ピン(メス)コネクタを 2 ポート採用した RS-422A/485 PCI ボードです。
また、基板上に DIO ポート(6 ビット)と複数(4 枚まで)の基板を識別する DIP スイッチも実装しています。

ハードウェア仕様

項　　目	仕　　様　　内　　容
バスインターフェース	PCI Local Bus Rev.3.0(32 ビットバスマスタ) 3.3V/5V ユニバーサル PCI ボード
シリアルコントローラ	メモリマップ方式 16550 互換 UART
接続コネクタ	D-Sub 15Pin(メス)×2
入出力レベル	EIA/TIA-422B/485A レベル
通信速度	300/600/1200/2400/4800/9600/19200/38400/ 57600/115200/230400/460800 bps ※実際に実行可能な最大通信速度はパソコンの仕様に依存します。
通信パラメータ	ビット長：7/8　　スタートビット：1 ストップビット：1/2　　パリティ：偶数/奇数/なし
ドライバ・レシーバ	【ドライバ】Intersil ISL32274E（または相当品） 【レシーバ】Intersil ISL32275E（または相当品）
静電気放電保護	気中放電：±16.5KV　接触放電：±8KV
ボード ID	REX-PCI60D/PE60D/PCI64D/PE64D/PCI70D/PE70D を複数枚使用した場合、各ボードを特定するための DIP スイッチを基板上に設置。(4 枚まで識別可能)
DIO 端子	基板上に 6 ビットの DIO 端子を装備 0V～+5V（TTL レベル・正論理）
伝送距離	1200m 以内 ※高速でのデータ伝送をおこなう場合、ケーブル長や通信環境により正常な通信ができない場合があります。

外形寸法	約 119.91mm(W)×54.4(H)[mm] (PCI ブラケット含まず)
重量	約 80g(標準 PCI ブラケットを含む)
電源電圧	+3.3V/+5V(PCI バスより供給)
動作環境	温度:0～55℃　湿度:20～80%(ただし結露しないこと)

ソフトウェア仕様

項　　目	仕　　様　　内　　容
通信サンプルプログラム	RS-422A/485 通信サンプルプログラム (VC++2010,VB2010,VC++6.0,VB6.0)
ポート認識用サンプルプログラム	ボード上 DIP スイッチおよびポート読み取りサンプルプログラム (VC++2010,VB2010,VC++6.0,VB6.0)
DIO 制御サンプルプログラム	DIO を制御するサンプルプログラム (VC++6.0,VB6.0)
シリアル通信ドライバ DIO 制御ドライバ	REX-PCI70D 用デバイスドライバ

本製品の制限事項

本製品はメモリマップ方式のため、OADG ハードウェア仕様で定められた I/O ベースアドレスへのマッピングは行われません。
したがいまして、直接 I/O ポートアドレスにアクセスしているアプリケーションはご使用になれません。

本製品は RS-422A のドライバを使用しておりますので、RS-485 のドライブ能力は RS-422A となります。

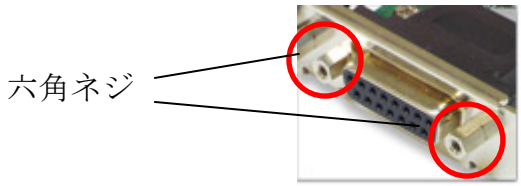
(1-2) 添付品

ご使用前に下記添付品が添付されているかをご確認願います。

- ☑ RS-422A/485 PCI ボード本体(標準 PCI ブラケット付)
- ☑ Low profile PCI ブラケット
- ☑ ソフトウェア CD-ROM
- ☑ インストールガイド
- ☑ 保証書

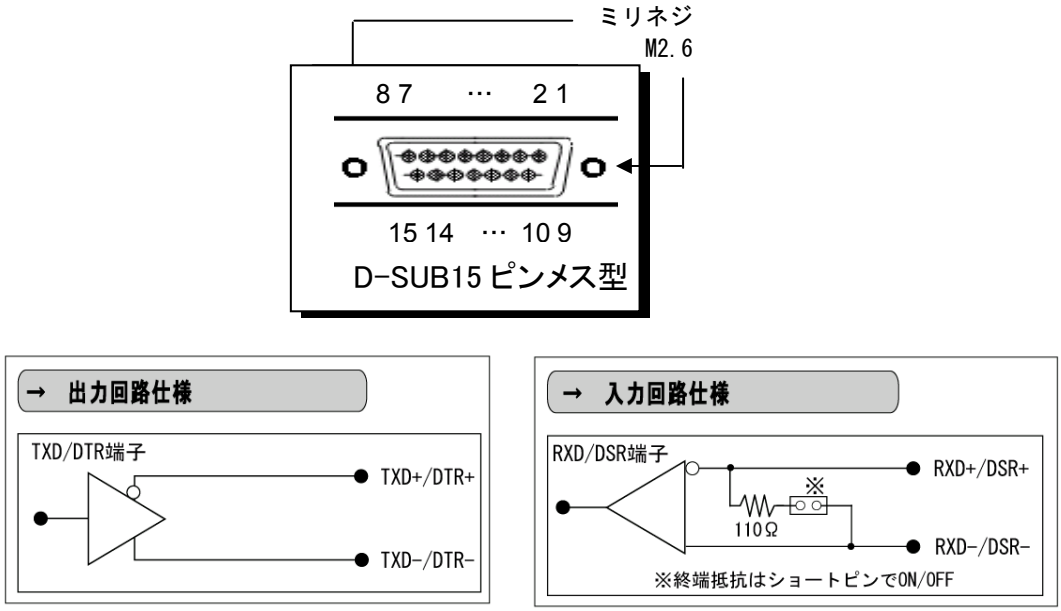
(1-3) PCI ブラケットの取替え

ロープロファイル PCI スロットでご使用の場合は、ブラケットの取替えが必要となります。
PCI ボード上の六角ネジを取外してブラケットを取替えます。



(1-4) コネクタピンアサイン

各信号のコネクタピンアサインと入出力回路は下図のようになります。



次ページは、各信号と機能についての表となります。
RS-422A の場合、本製品の基板上で RTS/CTS が処理されており、ドライバ・レシーバは常に ON となっています。
また RS-485 の場合、本製品の基板上で RTS/CTS DTR/DSR が処理されており、ドライバ・レシーバの ON/OFF 制御に RTS と DTR を次のように設定します。

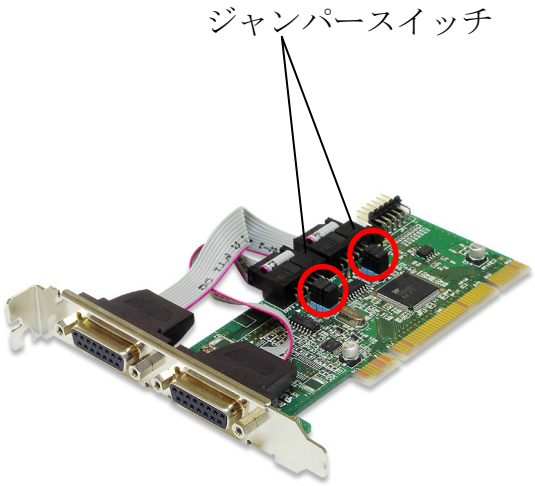
- RS-485(半二重)の場合
  送信時 -- RTS を ON。　受信時 -- RTS を OFF。DTR を ON。
- RS-485(全二重)の場合
  送信時 -- RTS を ON。　受信時 -- DTR を ON。

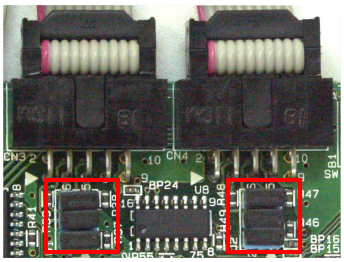
	RS-422A ピンアサイン		RS-485 ピンアサイン	
ピン番	信号名	機能	信号名	機能
1	NC	－	NC	－
2	SD+(TXD+)	送信データ	SD+(TXD+)	送信データ
3	NC	－	NC	－
4	RD+(RXD+)	受信データ	RD+(RXD+)	受信データ
5	DR+(DSR+)	データセットレディ	NC	－
6	NC	－	NC	－
7	ER+(DTR+)	端末レディ	NC	－
8	SG	信号用接地	SG	信号用接地
9	SD-(TXD-)	送信データ	SD-(TXD-)	送信データ
10	NC	－	NC	－
11	RD-(RXD-)	受信データ	RD-(RXD-)	受信データ
12	DR-(DSR-)	データセットレディ	NC	－
13	NC	－	NC	－
14	ER-(DTR-)	端末レディ	NC	－
15	NC	－	NC	－

(1-5) RS-422A/485 の切換えと終端抵抗について

本製品の基板上に搭載されているジャンパースイッチでは、ポートごとに次の設定を行うことができます。

- RS-422A と RS-485 の切換え。
- 終端抵抗の設定。





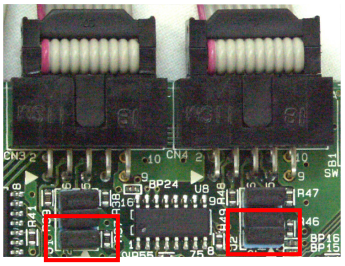
[J1] [J2]

6	5	6	5
4	3	4	3
2	1	2	1

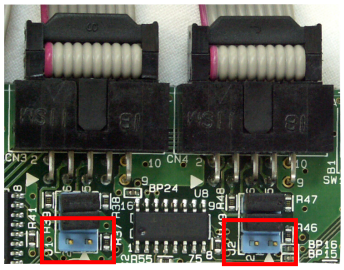
［ジャンパー図］

・ RS-422A/RS485 の切換え

ジャンパー図の[1・2]がショートの場合は RS-422A、
オープンの場合は RS-485 の設定となります。



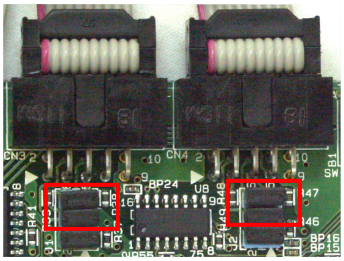
[RS-422A 設定]
(ジャンパー[1・2]ショート)



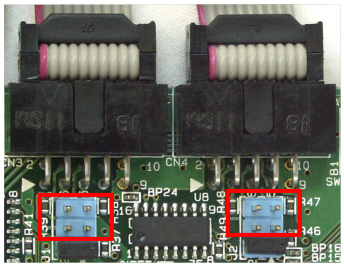
[RS-485 設定]
(ジャンパー[1・2]オープン)

・ 終端抵抗の設定

終端抵抗を取り外す場合は、ジャンパー図の[3・4][5・6]をオープンにします。



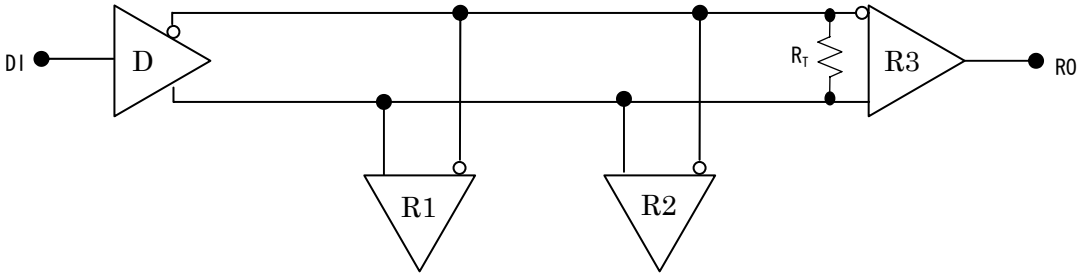
[終端抵抗あり設定]
(ジャンパー[3・4] [5・6]ショート)



[終端抵抗なし設定]
(ジャンパー[3・4] [5・6]オープン)

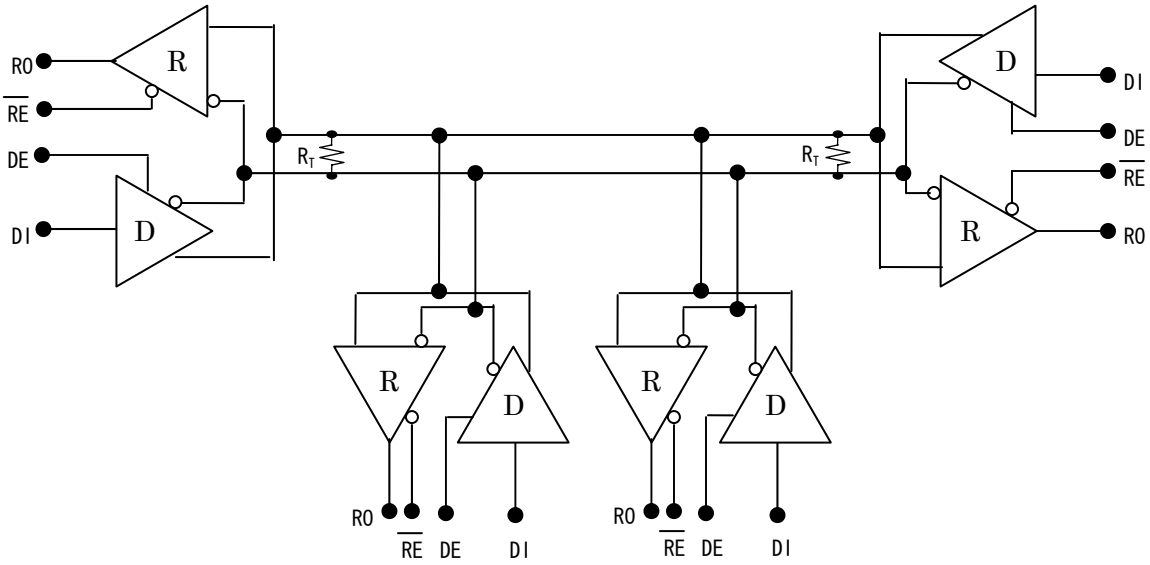
・ RS-422A 回路例

1 つのドライバが、最大 10 個のレシーバを接続することができます。



・ 半二重 RS-485 回路例

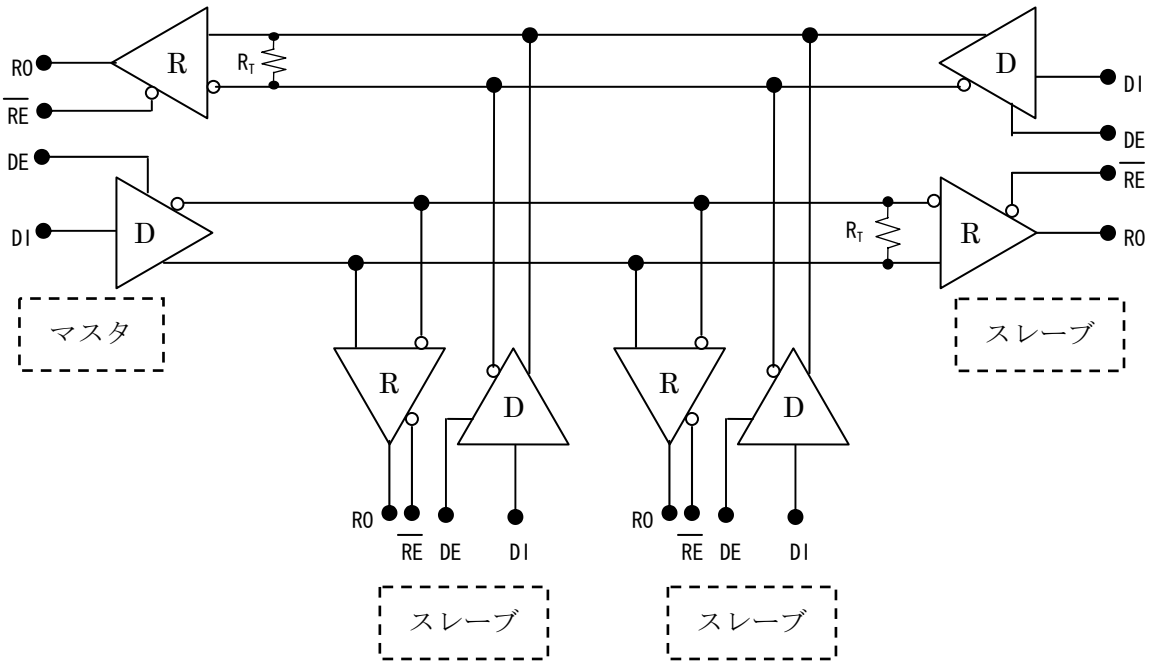
同じバス上に複数のドライバとレシーバを持ち、双方向でのデータ伝送が可能ですが、同時には一方向のみとなります。



※ RS-485 の場合は、配線長が一番長くなる両端に終端抵抗を取り付け、それ以外の終端抵抗は取り外してください。

・ 全二重 RS-485 回路例

マスタノードとスレーブノードとの間で、双方向の同時通信が可能です。



※ RS-485 の場合は、配線長が一番長くなる両端に終端抵抗を取り付け、それ以外の終端抵抗は取り外してください。

(1-6) レジスタセット

シリアルコントローラはメモリマップ方式16C550互換UARTが搭載されています。
詳細につきましては、 EXAR XR17D152 のデータシートをご参照ください。

TABLE 11: UART CHANNEL CONFIGURATION REGISTERS DESCRIPTION. SHADED BITS ARE ENABLED BY EFR BIT-4.

ADDRESS A3-A0	REG NAME	READ/ WRITE	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	COMMENT
0 0 0 0	RHR	R	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	LCR[7]=0
0 0 0 0	THR	W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	LCR[7]=0
0 0 0 0	DLL	R/W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	LCR[7]=1
0 0 0 1	DLM	R/W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	LCR[7]=1
0 0 0 1	IER	R/W	0/ CTS/ DSR# Int. Enable	0/ RTS/ DTR# Int. Enable	0/ Xon/Xoff/ Sp. Char. Int. Enable	0	Modem Status Int. Enable	RX Line Status Int. Enable	TX Empty Int. Enable	RX Data Int. Enable	
0 0 1 0	ISR	R	FIFOs Enable	FIFOs Enable	0/ Delta-Flow Cntl	0/ Xoff/special char	INT Source Bit-3	INT Source Bit-2	INT Source Bit-1	INT Source Bit-0	
0 0 1 0	FCR	W	RXFIFO Trigger	RXFIFO Trigger	0/ TXFIFO Trigger	0/ TX FIFO Trigger	DMA Mode	TX FIFO Reset	RX FIFO Reset	FIFOs Enable	
0 0 1 1	LCR	R/W	Divisor Enable	Set TX Break	Set Parity	Even Par-ity	Parity Enable	Stop Bits	Word Length Bit-1	Word Length Bit-0	
0 1 0 0	MCR	R/W	0/ BRG Prescaler	0/ IR Enable	0/ XonAny	Internal Lopback Enable	(OP2)[1]	(OP1)[1] RTS/DTR Flow Sel	RTS# Pin Control	DTR# Pin Control	
0 1 0 1	LSR	R/W	RX FIFO ERROR	TSR Empty	THR Empty	RX Break	RX Fram-ing Error	RX Parity Error	RX Over-run	RX Data Ready	
0 1 1 0	MSR	R	CD	RI	DSR	CTS	Delta CD#	Delta RI#	Delta DSR#	Delta CTS#	
	MSR	W	RS485 DLY-3	RS485 DLY-2	RS485 DLY-1	RS485 DLY-0	Reserved	Reserved	Reserved	Reserved	
0 1 1 1	SPR	R/W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	User Data
1 0 0 0	FCTR	R/W	TRG Table Bit-1	TRG Table Bit-0	Auto RS485 Enable	Invert IR RX Input	RTS/DTR Hyst Bit-3	RTS/DTR Hyst Bit-2	RTS/DTR Hyst Bit-1	RTS/DTR Hyst Bit-0	
1 0 0 1	EFR	R/W	Auto CTS/DSR Enable	Auto RTS/DTR Enable	Special Char Select	Enable IER [7:5], ISR [5:4], FCR[5:4], MCR[7:5,2] MSR[7:4]	Software Flow Cntl Bit-3	Software Flow Cntl Bit-2	Software Flow Cntl Bit-1	Software Flow Cntl Bit-0	
1 0 1 0	TXCNT	R	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
1 0 1 0	TXTRG	W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
1 0 1 1	RXCNT	R	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
1 0 1 1	RXTRG	W	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	

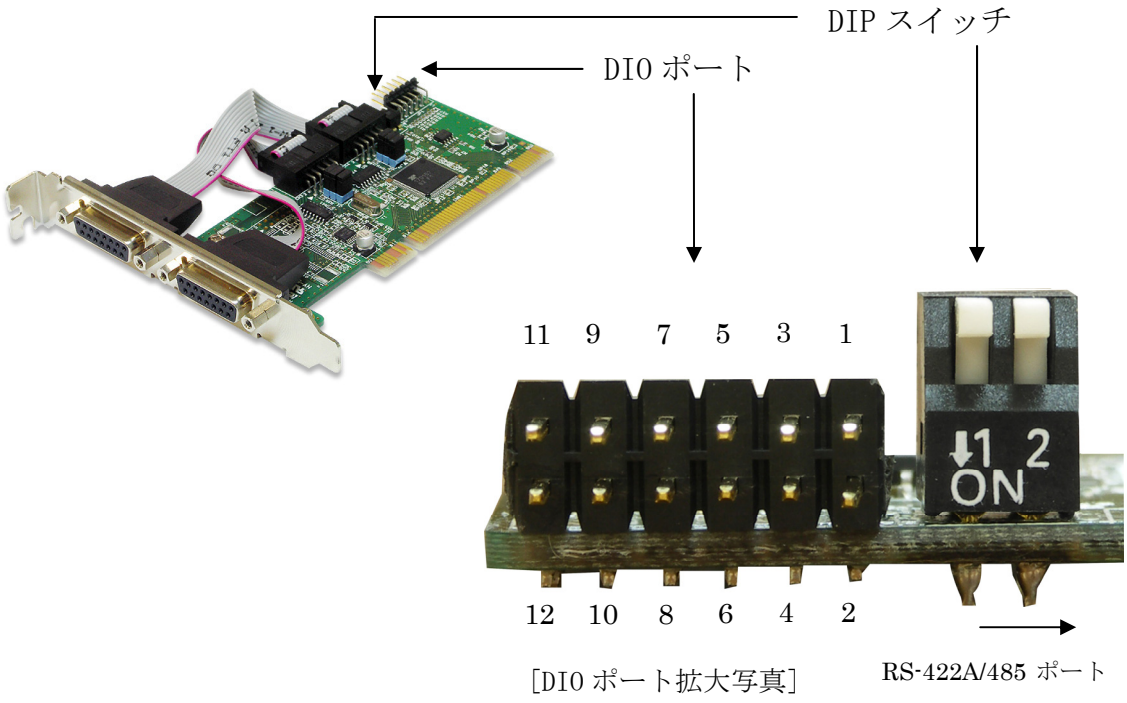
(1-7) DIO ポートについて

本製品の基板上には、DIO 制御するためのポート(6 ビット)が実装されています。
DIO ポートを利用することにより、次の機能を使用することができます。

・ ビット単位での 0V～+5V のデジタル入出力。

・ 指定したビットへの割り込み入力の検出。(エッジを指定)
(参照:「第 3 章 (3-5) DIO 制御サンプルアプリケーションについて」)

基板上の DIO ポートは下図のようになります。(横から見た図)



[DIO ポート拡大写真]

上図のピン番号 1～6 が DIO bit0～5 に対応しています。
※ DIO ポートは 6 ビットが有効となります。

ピン番号	12	11	10	9	8	7	6	5	4	3	2	1
DIO bit	--						bit5	bit4	bit3	bit2	bit1	bit0
機能	--	GND	Power (5V)	--			I/O	I/O	I/O	I/O	I/O	I/O
							INT	INT	INT	INT	INT	INT

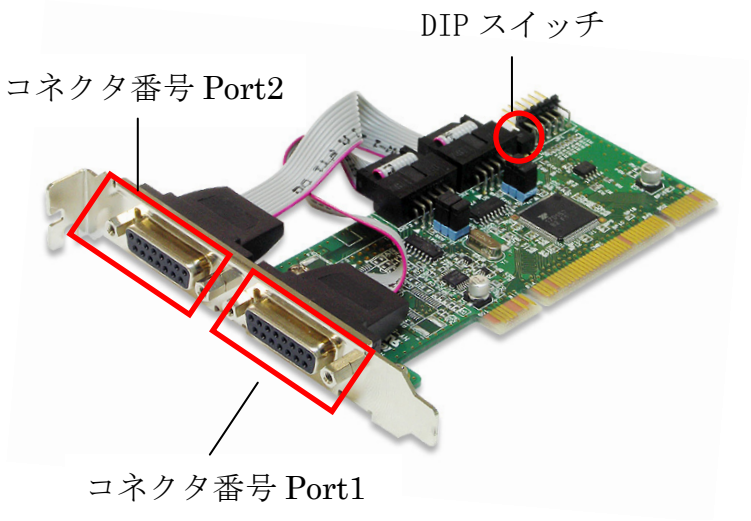
(1-8) DIP スイッチについて

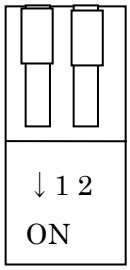
本製品の基板上には、複数枚のボードをソフトウェアで識別するための DIP スイッチが実装されています。
DIP スイッチで ID 番号(ID:0～ID:3)を設定することにより、次の機能を使用することができます。

・ 複数枚(4 枚まで)同時装着時に各コネクタ番号(下図参照：基板側が Port1、拡張側が Port2)に割り当てられている COM ポート番号が簡単に確認できます。

・ ソフトウェアから各ボードを識別することが可能。
(参照:「第 3 章 (3-6) 設定ポート確認サンプルアプリケーションについて」)

本機能を使用するには、下図の DIP スイッチで、ID 番号が重複しないように設定し、ボードを装着します。
※添付の設定ポート確認サンプルアプリケーションを使用する場合は、必須となります。
※重複した場合は上記機能が使用できませんが、COM ポート番号がアサインされていれば正常にご使用いただけます。





DIP スイッチ

DIP スイッチを矢印の方向に設定すると ON となります。
各 DIP スイッチの設定に対応する ID 番号は右表の通りです。
(※ 出荷時は ID：0 となっています。)

1	2	ID 番号
OFF	OFF	0
ON	OFF	1
OFF	ON	2
ON	ON	3

(1-9) RS-422A/485 の特性について

EIA 規格の RS-422A 及び RS-485 は RS-232C と異なり、信号の名称・タイミング・プロトコル・コネクタピン配置に関する定義はなく、ドライバ・レシーバの電気的特性・ケーブル及び終端抵抗についてのみ定義されています。

RS-422A は、1 ドライバ・10 レシーバの単一方向のマルチドロップ方式で規定されています。

これに対し RS-485 は、32 ドライバ・32 レシーバの双方向のマルチドロップ形式でパーティライン構成として規定されています。

RS-422A 及び RS-485 の電気的特性は下表のようになっています。

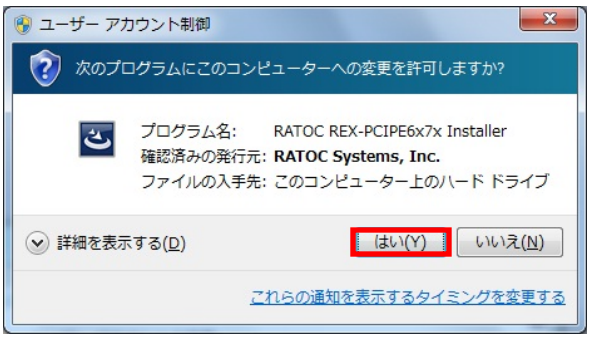
項目		RS-422A	RS-485
動作モード		平衡型	平衡型
接続可能台数		1 ドライバ 10 レシーバ	32 ドライバ 32 レシーバ
最大ケーブル長		1200m	1200m
最大伝送速度	12m	10Mbit/s	10Mbit/s
	120m	1Mbit/s	1Mbit/s
	1200m	90Kbit/s	90Kbit/s
最大同相電圧		+6V -0.25V	+12V -7V
ドライバ出力電圧	負荷時	±2V	±1.5V
ドライバ負荷抵抗		100Ω	54Ω
ドライバ出力抵抗 (ﾊｲｲﾝﾋﾟｰﾀﾞﾝｽ状態)	パワーON	規定無し	±100μA(Max) -7V≦Vcom≦+12V
	パワーOFF	±100μA(Max) -0.25V≦Vcom≦+6V	
レシーバ同相入力電圧範囲		-7V から+7V	-12V から+12V
レシーバ入力感度		±200mV	±200mV
レシーバ入力抵抗		＞4KΩ	＞12KΩ

第2章 Windowsセットアップ

(2-1) Windows 10 / 8.1 / 8 / 7 / Vista / 2000 Server2008 / 2000Server セットアップ

本製品を接続する前に製品添付 CD-ROM 内のドライバをセットアップします。
以下の手順でインストールを行ってください。

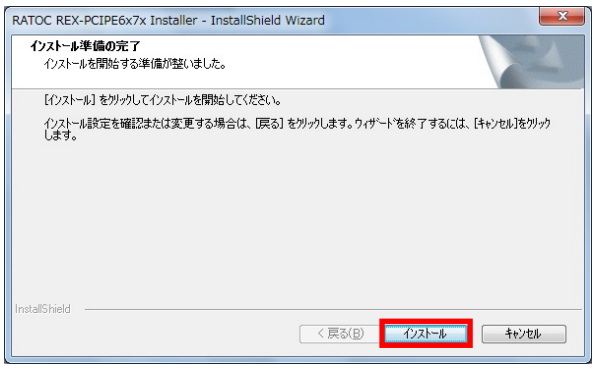
製品添付 CD-ROM 内の PCIPE6x7x_Setup.exe をダブルクリックして実行します。
ユーザーアカウント制御の画面が表示された場合は、「はい(Y)」ボタンをクリックします。



「RATOC REX-PCIPE6x7x Installer セットアップへようこそ」の画面で「次へ(N)」ボタンをクリックします。



「インストール準備の完了」の画面で「インストール」ボタンをクリックします。

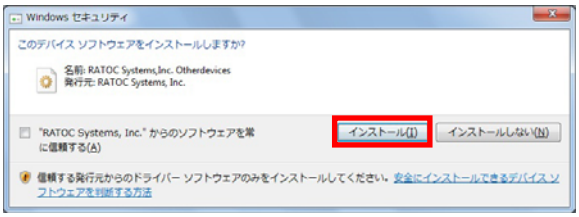


Windows セキュリティの確認画面が 3 回表示されますので、「インストール(I)」ボタンをクリックします。

(※ Windows 2000/2000Server ではこれらの画面は表示されません。)

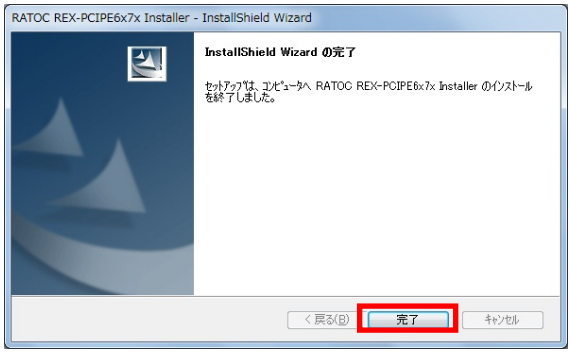






以上でドライバのセットアップは完了です。

PC の電源を切り本製品を装着してください。



PC を起動後はセットアップしたドライバが自動的にインストールされます。

「(2-3) インストールの確認」へ進み、正常にインストールされていることを確認してください。

(2-2) Windows XP / Server2003 セットアップ

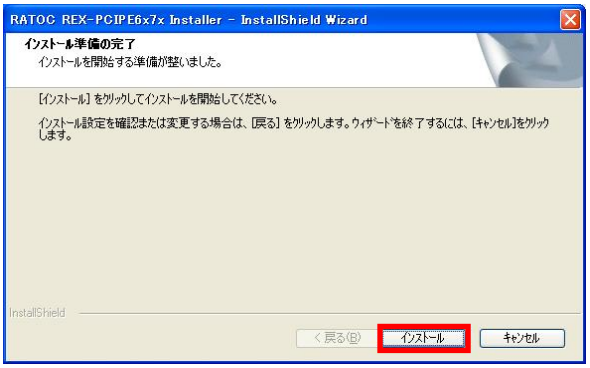
本製品を接続する前に製品添付 CD-ROM 内のドライバをセットアップします。以下の手順でインストールを行ってください。

< ドライバのセットアップ>

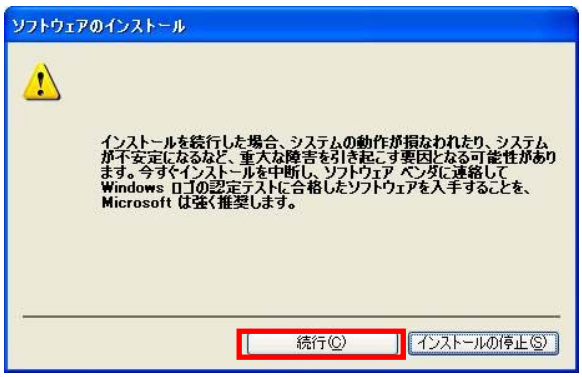
「RATOC REX-PCIPE6x7x Installer セットアップへようこそ」の画面で「次へ(N)」ボタンをクリックします。

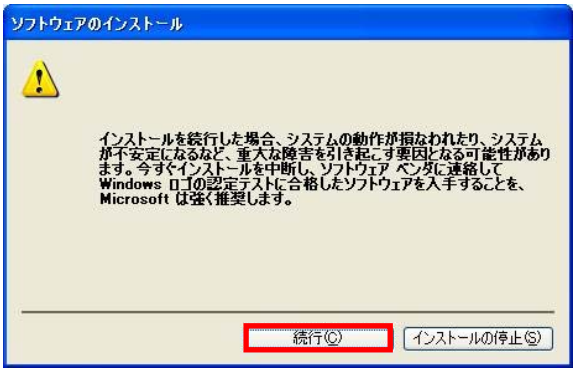


「インストール準備の完了」の画面で「インストール」ボタンをクリックします。



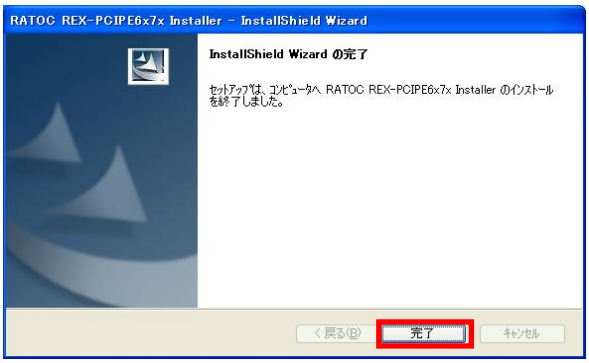
警告画面が 2 回表示されますが「続行(C)」ボタンをクリックします。





以上でドライバのセットアップは完了です。

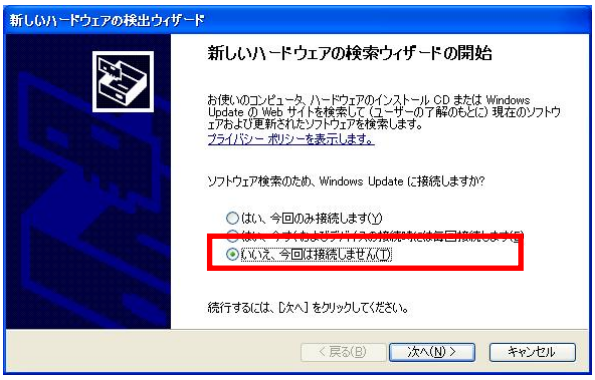
PC の電源を切り本製品を装着してください。



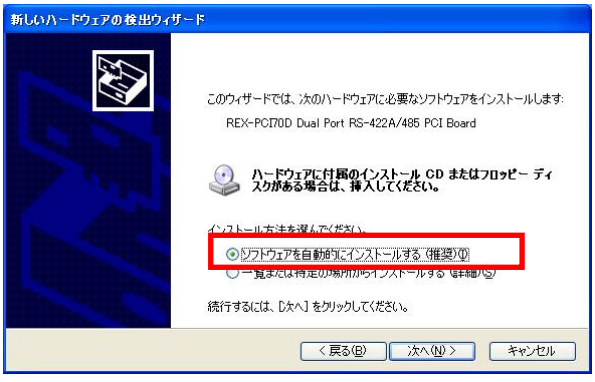
PC を起動後は以下の手順にてインストールを行います。

< REX-PCI70D Dual Port RS-422A/485 PCI Board のインストール>

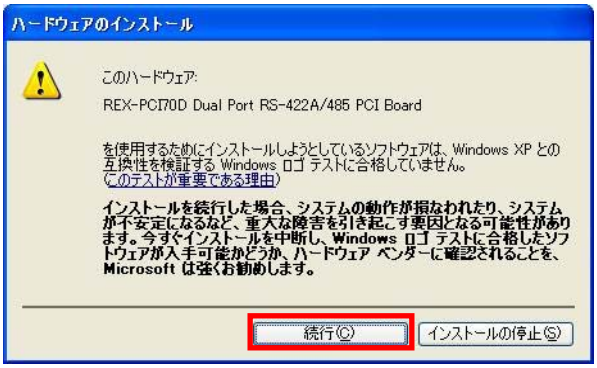
「新しいハードウェアの検索ウィザードの開始」で、「いいえ、今回は接続しません(T)」を選択し「次へ(N)」ボタンをクリックします。



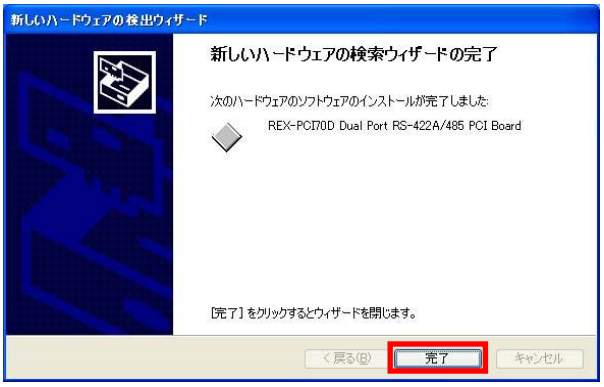
「ソフトウェアを自動的にインストールする(推奨)(I)」が選択されていることを確認し「次へ(N)」ボタンをクリックします。



「ロゴテストに合格していません」と表示されますが、「続行(C)」ボタンをクリックします。

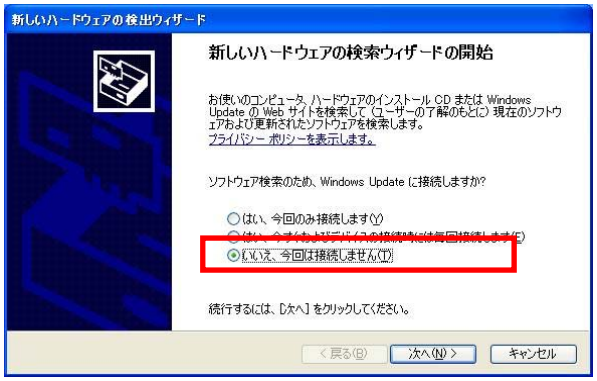


以上で REX-PCI70D Dual Port RS-422A/485 PCI Board のインストールは完了です。
次に
「REX-PCI70D Digital I/O Port」のインストールウィザード
が自動的に起動します。

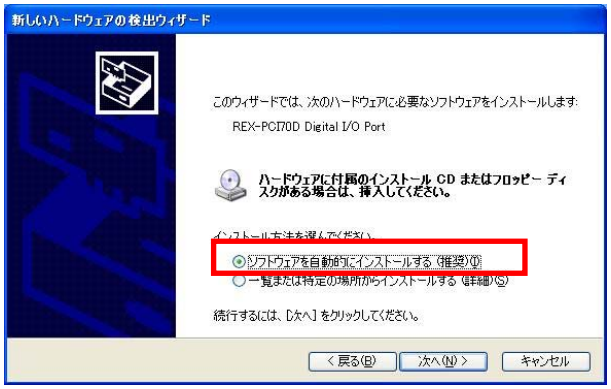


<REX-PCI70D Digital I/O Port のインストール>

「新しいハードウェアの検索ウィザードの開始」で、「いいえ、今回は接続しません(T)」を選択し「次へ(N)」ボタンをクリックします。



「ソフトウェアを自動的にインストールする(推奨)(I)」が選択されていることを確認し「次へ(N)」ボタンをクリックします。

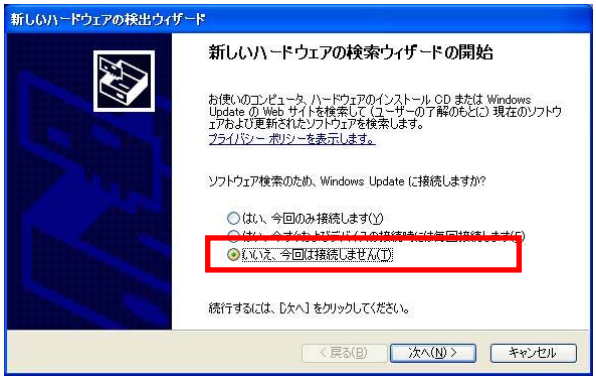


以上で REX-PCI70D Digital I/O Port のインストールは完了です。
次に
「REX-PCI70D Communications Port」のインストールウィザードが自動的に起動します。

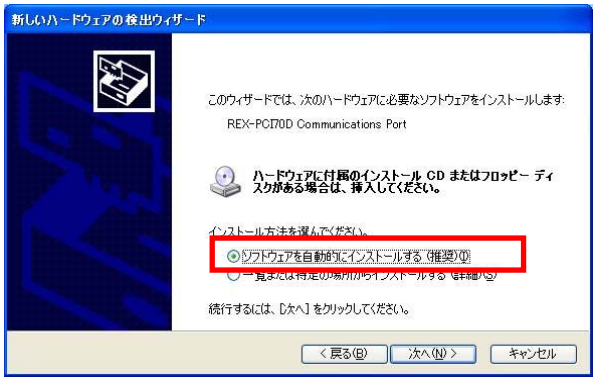


<REX-PCI70D Communications Port のインストール>

「新しいハードウェアの検索ウィザードの開始」で、「いいえ、今回は接続しません(T)」を選択し「次へ(N)」ボタンをクリックします。



「REX-PCI70D Communications Port」と表示されていることを確認してください。
「ソフトウェアを自動的にインストールする(推奨)(I)」が選択されていることを確認し「次へ(N)」ボタンをクリックします。



「ロゴテストに合格していません」と表示されますが、「続行(C)」ボタンをクリックします。



ドライバのコピーが開始され、完了のメッセージが表示されます。
「完了」ボタンをクリックします。



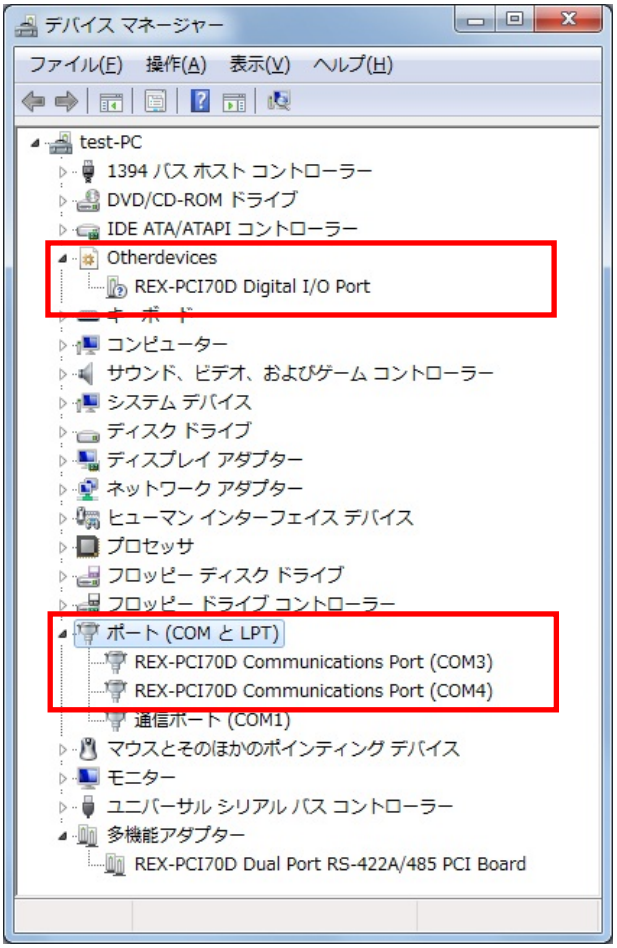
2 ポート分についてのインストール作業が必要となりますので、同様の手順で行なってください。

以上で REX-PCI70D のインストールは終了です。
「(2-3) インストールの確認」へ進み、正常にインストールされていることを確認してください。

(2-3) インストールの確認

コントロールパネルの「デバイスマネージャ」を起動します。
「ポート(COM と LPT)」をクリックして新しくポートが追加されていることを確認してください。
また、Otherdevices に
「REX-PCI70D Digital I/O Port」が追加されていることを確認してください。

(Windows10/8.1/8/7/Vista/Server2008 以外の OS では、コントロールパネルのシステムを起動し、
「システムのプロパティ」の「ハードウェア」タブから「デバイスマネージャ」ボタンをクリックします。)

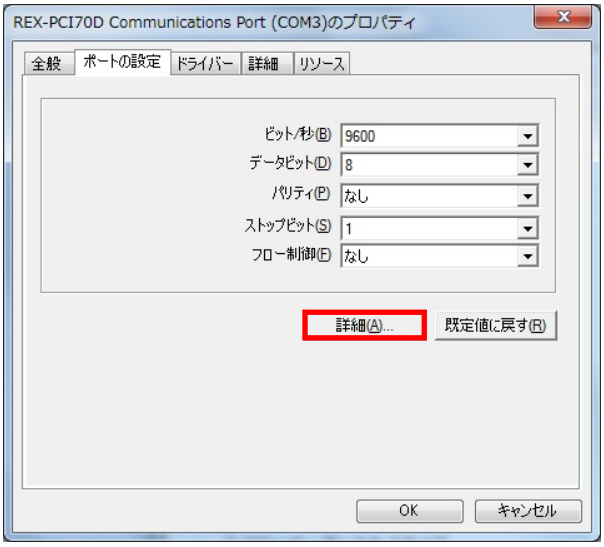


画面では「COM3」「COM4」となっておりますが、ご使用の環境により COMx の x の数字が異なりますのでご注意ください。

(2-4) COM ポート番号の変更と設定について

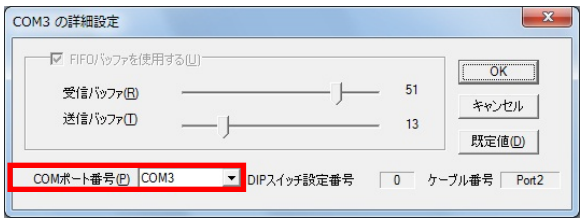
本製品に割り当てられた COM ポート番号の変更はデバイスマネージャ上より行うことができます。

「(2-3) インストールの確認」と同様にポートのプロパティ画面を開き「ポートの設定」タブをクリックします。
「詳細」ボタンをクリックするとポートの詳細設定ダイアログが表示されます。



【ポートのプロパティ画面】

COM ポート番号を変更するには「COM ポート番号」コンボボックスより変更先 COM 番号を選択後に「OK」ボタンをクリックしてください。



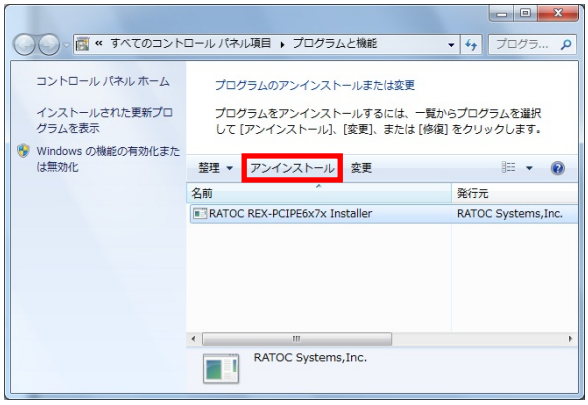
【ポートの詳細設定ダイアログ】

※ ご使用される環境や通信設定によっては FIFO バッファがオーバーフローする場合があります。
この場合、上記「ポートの詳細設定ダイアログ」にて受信バッファ(受信時の割り込みトリガレベル)を小さくすることで改善されることがあります。

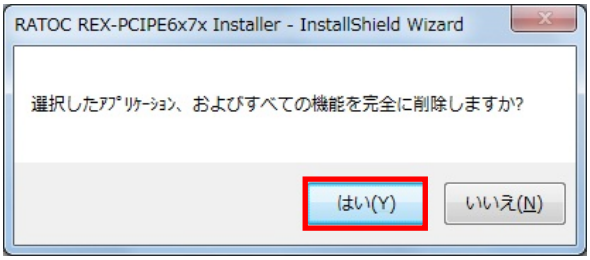
(2-5) ドライバのアンインストール

コントロールパネルの「プログラムと機能」を起動し、セットアップされたドライバをアンインストールします。
(Windows XP/2000/Server2003 では「プログラムの追加と削除」を起動します。)

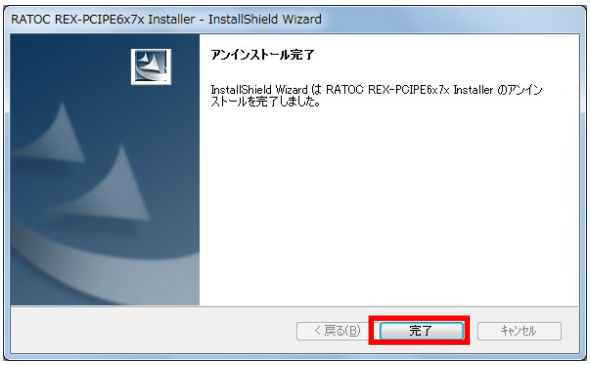
「 RATOC REX-PCIPE6x7x Installer」を選択し、「アンインストール」をクリックします。



アンインストールの確認画面が表示されますので「はい(Y)」ボタンをクリックします。



以上で
REX-PCI70D のアンインストールは完了です。



第3章 DIO制御・設定ポート確認用ライブラリ関数

(3-1) ライブラリ関数について

本製品に添付しているライブラリ関数を利用すると、「6bit の DIO 制御」「DIP スイッチ設定とコネクタ番号から割り当てられている COM ポート番号を取得」することができます。
また、アサインされている COM ポート番号から DIP スイッチ設定とコネクタ番号を知ることができます。

(3-2) 関数仕様

ライブラリ関数と機能の一覧は次の通りとなります。

関数名	機能
DIO 制御	
DioOpen	DIO ポートをオープン
DioClose	DIO ポートをクローズ
SetDirection	DIO ポートの入出力方向を設定
ReadPort	入力方向に設定した DIO ポートの読み取り
WritePort	出力方向に設定した DIO ポートへの書き込み
IntPostMessage	DIO ポートの指定したビットに指定した入力が検出されると、ポストメッセージで通知される
設定ポート確認	
GetComPort	本製品に割り当てられているポート数および COM ポート番号の取得
GetBoardIdCableNo	指定した COM ポート番号の DIP スイッチ番号とコネクタ番号を取得
GetBoardId	本製品の枚数および DIP スイッチ番号を取得
GetCableNo	各コネクタに割り当てられている COM ポート番号を取得

※ ライブラリ関数仕様の詳細につきましては次ページ以降をご参照ください。

DIO 制御

DioOpen　　DIO ポートをオープン

書式　HANDLE DioOpen(Int BoardID)

引数　Int BoardID　　制御する製品の DIP スイッチの値を指定

戻値　INVALID_HANDLE_VALUE 以外：成功(DIO ポートのハンドルが返されます。)
INVALID_HANDLE_VALUE　　：失敗

機能　DIO 制御するボード上の DIP スイッチ番号を指定し、DIO ポートをオープンします。

VB6.0 での書式

```
Function DioOpen( ByVal BoardID As Long ) As Long
```

VB2010 での書式

```
Function DioOpen( ByVal BoardID As Integer ) As Integer
```

使用例

```
BYTE   BoardId; // DIP スイッチの値
HANDLE hSys;    // DIO ドライバのハンドル

// ボード ID をコンボボックスから取得
BoardId = m_BoardID.GetCurSel();
// ドライバオープン
hSys = DioOpen( BoardId );
```

DioClose DIO ポートをクローズ

書式 BOOL DioClose(HANDLE hSys)

引数 HANDLE hSys オープンした DIO ポートのハンドル

戻値 TRUE ：成功
FALSE：失敗

機能 オープンしている DIO ポートのハンドルを指定し、クローズします。

VB6.0 での書式

```
Function DioClose( ByVal hSys As Long ) As Byte
```

VB2010 での書式

```
Function DioClose( ByVal hSys As Integer ) As Byte
```

使用例

```
HANDLE hSys; // DIO ドライバのハンドル

// ドライバクローズ
DioClose( hSys );
```

SetDirection　　DIO ポートの入出力方向を設定

書式　INT SetDirection(HANDLE hSys, UCHAR Direction)

引数　HANDLE hSys　　オープンした DIO ポートのハンドル
UCHAR Direction　　ビット単位での方向設定(0:出力 1:入力)

戻値　0 ：成功
-2 ：ドライバ呼び出しエラー

機能　ビット単位で入出力方向を設定します。
bit0～5 のみ有効となり、上位 2 ビットは無視されます。

例）下位 4 ビットを入力する場合。

bit7	Bit6	bit5	bit4	bit3	bit2	bit1	bit0
Direction = 0x0f							
無効	無効	0	0	1	1	1	1

Direction に 0x3f を指定すると全ビット入力(bit0-bit5)、0x00 を指定すると全ビット出力(bit0-bit5)となります。

VB6.0 での書式

```
Function SetDirection( ByVal hSys As Long,
                       ByVal Direction As Byte ) As Long
```

VB2010 での書式

```
Function SetDirection( ByVal hSys As Integer,
                       ByVal Direction As Byte ) As Integer
```

使用例

```
HANDLE hSys;        // DIO ドライバのハンドル
BYTE bDirection;    // 方向設定

// 指定ビット出力方向にする
SetDirection( hSys, bDirection );
```

ReadPort　　入力方向に設定した DIO ポートの読み取り

書式　INT ReadPort(HANDLE hSys, PUCHAR pReadData)

引数　HANDLE hSys　　オープンした DIO ポートのハンドル
PUCHAR pReadData　　読み取った値が格納されるバッファのアドレス

戻値　0 ：成功
-1 ：方向設定エラー
（SetDirection 関数で、どのビットも入力方向に設定されていない。）
-2 ：ドライバ呼び出しエラー

機能　入力方向に設定した DIO ポートの値を読み取ります。

VB6.0 での書式

```
Function ReadPort( ByVal hSys As Long, pReadData As Byte ) As Long
```

VB2010 での書式

```
Function ReadPort( ByVal hSys As Integer,
                   ByRef pReadData As Byte ) As Integer
```

使用例

```
HANDLE hSys;     // DIO ドライバのハンドル
UCHAR  ReadData; // リードバッファ

// 現在のデータ取得
ReadPort( hSys, &ReadData );
```

WritePort　出力方向に設定した DIO ポートへの書き込み

書式　INT WritePort(HANDLE hSys, UCHAR WriteData)

引数　HANDLE hSys　　オープンした DIO ポートのハンドル
UCHAR WriteData　　書き込む値

戻値　0 ：成功
-1 ：方向設定エラー
（SetDirection 関数で、どのビットも出力方向に設定されていない。）
-2 ：ドライバ呼び出しエラー

機能　出力方向に設定した DIO ポートへ値を書き込みます。

VB6.0 での書式

```
Function WritePort( ByVal hSys As Long,
                    ByVal WriteData As Byte ) As Long
```

VB2010 での書式

```
Function WritePort( ByVal hSys As Integer,
                    ByVal WriteData As Byte ) As Integer
```

使用例

```
HANDLE hSys;      // DIO ドライバのハンドル
UCHAR  WriteData; // 書き込みデータ

// 書き込み
WritePort( hSys, WriteData );
```

IntPostMessage　　**DIO ポートの指定したビットに指定した入力が検出されると、ポストメッセージで通知される**

書式　INT IntPostMessage(HANDLE hSys, UCHAR IntBit, UCHAR IntEdge,
HWND hwnd)

引数

HANDLE hSys	オープンした DIO ポートのハンドル
UCHAR IntBit	割り込みを検出するビットを指定 (1：有効　0：無効)
UCHAR IntEdge	割り込みを検出するエッジを設定 (0:立下り High→Low　1:立上がり Low→High)
HWND hwnd	メッセージを受取るウィンドウのハンドル

戻値　0 ：成功
-1 ：方向設定エラー
(SetDirection 関数で、どのビットも入力方向に設定されていない。)
-2 ：ドライバ呼び出しエラー

機能　DIO ポートの指定したビットに指定した入力が検出されると、ポストメッセージでアプリケーションに通知されます。

例) bit0 の Low→High(立上り)と bit5 の High→Low(立下り)の
割り込みを検出する場合

bit7	Bit6	bit5	bit4	bit3	bit2	bit1	bit0
IntBit = 0x21							
無効	無効	1	0	0	0	0	1
IntEdge = 0x01							
無効	無効	0	--	--	--	--	1

VB6.0 での書式

```
Function IntPostMessage( ByVal hSys As Long,
                         ByVal IntBit As Byte,
                         ByVal IntEdge As Byte,
                         ByVal hwnd As Long ) As Long
```

VB2010 での書式

```
Function IntPostMessage( ByVal hSys As Integer,
                         ByVal IntBit As Byte,
                         ByVal IntEdge As Byte,
                         ByVal hwnd As Integer ) As Integer
```

使用例

```
HANDLE hSys;   // DIO ドライバのハンドル
UCHAR  bInt;   // 割り込みビットを指定
UCHAR  bEdge;  // 割り込みエッジを指定

// 割り込み設定をする
IntPostMessage( hSys, bInt, bEdge, m_hWnd );
```

設定ポート確認

GetComPort　　**本製品に割り当てられている COM ポート数および COM ポート番号の取得**

書式　CHAR GetComPort(PCHAR pComPort, CHAR ComPortSize)

引数　PCHAR pComPort　　COM ポート番号を格納する配列へのポインタ
(NULL をセットした場合、本製品にアサインされている COM ポートのトータル数のみが戻り値として返されます)
CHAR　ComPortSize　　第一引数で確保された配列のサイズ

戻値　本製品にアサインされている COM ポート数が返されます。

機能　ComPortSize に本製品にアサインされているポート数を指定すると、COM ポート番号の値が pComPort[0], pComPort[1]…にセットされます。

VB6.0 での書式

```
Function GetComPort(ComPort As Any, ByVal ComPortSize As Byte) As Byte
```

VB2010 での書式

```
Function GetComPort(ByVal ComPort As String,
                    ByVal ComPortSize As Byte) As Byte
```

使用例

```
// 本製品にアサインされている COM ポート数を取得
ComNum = GetComPort( NULL, 0 );
if( ComNum != 0 )  {
   // COM ポート分のメモリ確保
   pCom = (PCHAR)LocalAlloc( LPTR, ComNum );
   // アサインされた全ての COM ポートの番号を取得
   GetComPort( pCom, ComNum );
   // COM ポート番号を用いた処理
    ……..
   // メモリ解放
   LocalFree( pCom ); }
```

GetBoardIdCableNo　指定した COM ポート番号の DIP スイッチの値とコネクタ番号を取得

書式 BOOL GetBoardIdCableNo(PCHAR pComPort, PCHAR pBoardId, PCHAR pCableNo)

引数

PCHAR pComPort	COM ポートの番号を文字列でセット
PCHAR pBoardId	対応する DIP スイッチの値を格納する配列へのポインタ
PCHAR pCableNo	対応するコネクタ番号を格納する配列へのポインタ

戻値 pComPort で指定された COM ポート番号が本製品にアサインされている場合は TRUE が返されます。
その他のシリアルポートにアサインされている番号、または存在しないポート番号を指定した場合は FALSE が返されます。

機能 指定した COM ポート番号の DIP スイッチの値とコネクタ番号を取得します。

VB6.0 での書式

```
Function GetBoardIdCableNo(ByVal ComPort As String, BoardId As Any,
                           CableNo As Any) As Boolean
```

VB2010 での書式

```
Function GetBoardIdCableNo(ByVal ComPort As String,
                           ByVal BoardId As String,
                           ByVal CableNo As String) As Boolean
```

使用例

```
CHAR  BoardId;
CHAR  CableNo;
// 指定した COM ポートにアサインされた BoardId,CableNo を取得
if( GetBoardIdCableNo( “COM5”, &BoardId, &CableNo ) ) {
    // DIP スイッチ又はコネクタ番号を用いた処理
    …….. }
```

GetBoardId	本製品の枚数および DIP スイッチの値を取得

書式 CHAR GetBoardId(PCHAR pBoardId, CHAR BoardIdSize)

引数 PCHAR pBoardId DIP スイッチの値を格納する配列へのポインタ
(NULL をセットして本関数を呼び出すと、PC 上で認識されている本製品の枚数のみが戻り値として返されます)
CHAR BoardIdSize 第一引数で確保された配列のサイズ

戻値 PC 上で認識されている本製品の枚数が返されます。複数枚の本製品が認識されていて、DIP スイッチの値が重複している場合はエラーとなります。
VC の場合は-1、VB の場合は 0xFF が返されます。

機能 PC 上で認識されている複数枚の本製品についての DIP スイッチの値を列挙し第一引数に格納します。
(例) 2 枚認識されていて、DIP スイッチの値が“1”, “2”と設定されていた場合、BoardId[0] = 1, BoardId[1] = 2 とセットされます。

VB6.0 での書式

```
Function GetBoardId (BoardId As Any,
                     ByVal BoardIdSize As Byte) As Byte
```

VB2010 での書式

```
Function GetBoardId (ByVal BoardId As String,
                     ByVal BoardIdSize As Byte) As Byte
```

使用例

```
BoardNum = GetBoardId ( NULL, 0 ); // 本製品の接続数取得
if(BoardNum != 0 ) {
   pBoard = (PCHAR)LocalAlloc( LPTR, BoardNum ); // メモリ確保
   // 接続した全ての DIP スイッチの値を取得
   GetBoardId ( pBoard, BoardNum );
   // DIP スイッチの値を用いた処理
    ……
   LocalFree( pBoard ); } // メモリ解放
```

GetCableNo　各コネクタに割り当てられている COM ポート番号を取得

書式　BOOL GetCableNo(CHAR BoardId, PCHAR pComPort, CHAR ComPortSize)

引数

CHAR	BoardId	DIP スイッチの値をセット
PCHAR	pComPort	COM ポート番号を格納する配列へのポインタ
CHAR	ComPortSize	第二引数で確保された配列のサイズ

戻値　指定した DIP スイッチの値が設定されたボードの検出と、そのボードに対応する COM ポート番号を取得できた場合は TRUE が返されます。
指定した DIP スイッチの値が設定されたボードが検出されなかった場合は FALSE が返されます。

機能　指定した DIP スイッチの値が設定されているボードについて、
コネクタ番号が Port1、Port2 の順に、第二引数で指定された配列に COM ポート番号がセットされます。

VB6.0 での書式

```
Function GetCableNo(ByVal BoardId As Byte, ComPort As Any,
                    ByVal ComPortSize As Byte) As Byte
```

VB2010 での書式

```
Function GetCableNo(ByVal BoardId As Byte, ByVal ComPort As String,
                    ByVal ComPortSize As Byte) As Byte
```

使用例

```
CHAR   BoardId;
CHAR   ComPort[2];
BoardId = 0;
// 指定した BoardId のコネクタ番号 Port1、Port2 に割り当てられた
// COM 番号を取得
If( GetCableNo( BoardId, ComPort, sizeof(ComPort) ) ) {
   //COM ポート番号を用いた処理
     …… }
```

(3-3) DIO 制御・設定ポート確認サンプルアプリケーションの構成

各サンプルアプリケーションは製品添付 CD-ROM の以下のフォルダ内に収録されています。

・ DIO 制御サンプルアプリケーション [DioSample]
フォルダ内の構成は次のようになります。
- VB6 フォルダ・・・VisualBasic6.0 サンプル(OCX 未使用)
- VB6_OCX フォルダ・・・VisualBasic6.0 サンプル(OCX 使用)
- VC6 フォルダ・・・VisualC++6.0 サンプル

・ 設定ポート確認サンプルアプリケーション [ViewSample]
フォルダ内の構成は次のようになります。
- VB6 フォルダ・・・VisualBasic6.0 サンプル
- VC6 フォルダ・・・VisualC++6.0 サンプル
- VB2010 フォルダ・・・VisualBasic2010 サンプル
- VC2010 フォルダ・・・VisualC++2010 サンプル

(3-4) ライブラリ関数の呼び出し

VC からの DIO 制御ライブラリ呼び出し

Visual C/C++のアプリケーションから、製品に添付された DLL(RsDio.dll)の API を呼び出すには、以下の関数についての呼び出し宣言を行い、宣言したファイルをプロジェクトにインクルードする必要があります。
(RsDio.dll は本製品のドライバインストール時に、..¥Windows¥System32 フォルダへコピーされます。)

```
HANDLE DioOpen( Int BoardID )
BOOL DioClose( HANDLE hSys )
INT SetDirection( HANDLE hSys, UCHAR Direction )
INT ReadPort( HANDLE hSys, PUCHAR pReadData )
INT WritePort( HANDLE hSys, UCHAR WriteData )
INT IntPostMessage( HANDLE hSys, UCHAR IntBit, UCHAR IntEdge, HWND hwnd )
```

VC6.0 サンプルでの呼び出し例 (抜粋)

```
// RsDio.h 内
// RsDio.dll 関数呼び出し宣言
DllImport HANDLE APIENTRY DioOpen( int BoardID );
DllImport BOOL APIENTRY DioClose( HANDLE hSys );
DllImport INT APIENTRY ReadPort( HANDLE hSys, PUCHAR pReadData );
DllImport INT APIENTRY WritePort( HANDLE hSys, UCHAR WriteData );
DllImport INT APIENTRY SetDirection( HANDLE hSys, UCHAR Direction );
DllImport INT APIENTRY IntPostMessage( HANDLE hSys, UCHAR IntBit, UCHAR IntEdge, HWND hwnd );

// DioIntDlg.cpp 内
// 宣言したヘッダファイルをインクルード
#include "RsDio.h"
```

VC からのポート確認ライブラリ呼び出し

Visual C/C++のアプリケーションから、製品に添付された DLL(rsportui.dll ※64bit 版 OS の場合は rsportui64.dll)の API を呼び出すには、以下の関数についての呼び出し宣言を行い、それらを参照する必要があります。
(rsportui.dll は本製品のドライバインストール時に、..¥Windows¥System32 フォルダへコピーされます。)

```
CHAR GetComPort( PCHAR pComPort, CHAR ComPortSize )
BOOL GetBoardIdCableNo( PCHAR pComPort, PCHAR pBoardId, PCHAR pCableNo )
CHAR GetBoardId( PCHAR pBoardId, CHAR BoardIdSize )
BOOL GetCableNo( CHAR BoardId, PCHAR pComPort, CHAR ComPortSize )
```

VC6.0 サンプルでの呼び出し例 (抜粋)

```
// rsportui.dll 関数呼び出し宣言
CHAR (__stdcall*fnGetComPort)( CHAR*, CHAR );
BOOL (__stdcall*fnGetBoardIdCableNo)( CHAR*, CHAR*, CHAR* );
CHAR (__stdcall*fnGetBoardId)( CHAR*, CHAR );
BOOL (__stdcall*fnGetCableNo)( CHAR, CHAR*, CHAR );

BOOL CSampleDlg::OnInitDialog()
{
  // rsportui.dll をロード
  hRsui = LoadLibrary( "rsportui.dll" );
  if( !hRsui ){
    index = GetLastError();
    MessageBox( "LoadLibrary ｴﾗｰ", "失敗", MB_OK );
    return FALSE;
  }
  // rsportui.dll 内のエクスポート済み関数のアドレスを取得
  fnGetComPort = ( CHAR (__stdcall*)( CHAR*, CHAR ) )GetProcAddress( hRsui, "GetComPort" );
  fnGetBoardIdCableNo = ( BOOL (__stdcall*)( CHAR*, CHAR*, CHAR* ) )GetProcAddress( hRsui,
                                "GetBoardIdCableNo" );
  fnGetBoardId = ( CHAR (__stdcall*)( CHAR*, CHAR ) )GetProcAddress( hRsui, "GetBoardId" );
  fnGetCableNo = ( BOOL (__stdcall*)( CHAR, CHAR*, CHAR ) )GetProcAddress( hRsui,
                          "GetCableNo" );
```

VB からの DIO 制御ライブラリ呼び出し

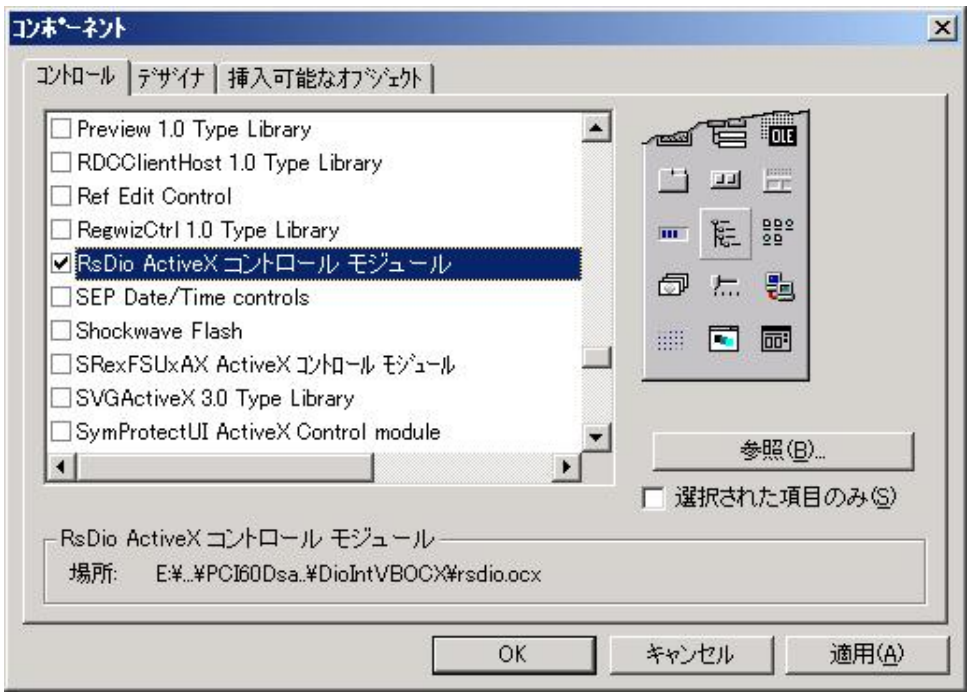
Visual Basic のアプリケーションから、製品に添付された DLL(RsDio.dll)の API を呼び出すには、関数についての呼び出し宣言を行い、それらを参照する方法と、OCX から呼び出す方法があります。

VB6.0 で DLL から直接呼び出す場合（RsDio.bas）

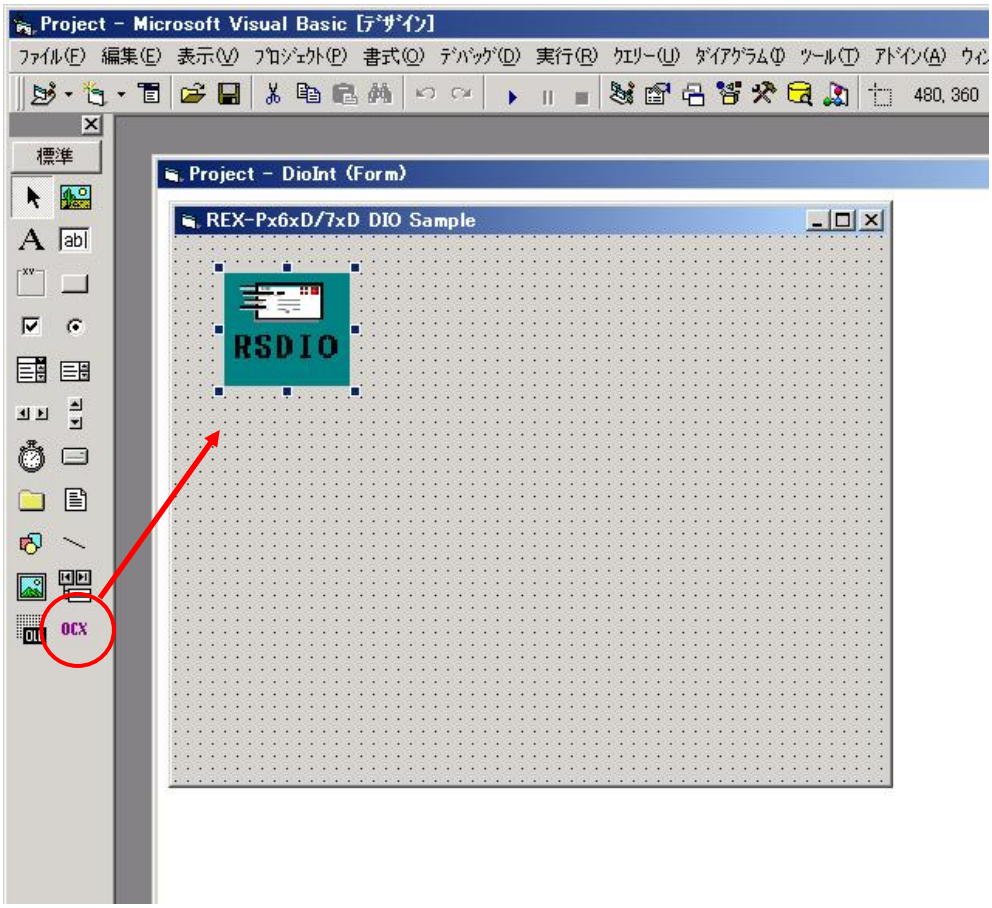
```
Declare Function DioOpen Lib "RsDio.dll" (ByVal BoardID As Long) As Long
Declare Function DioClose Lib "RsDio.dll" (ByVal hSys As Long) As Byte
Declare Function SetDirection Lib "RsDio.dll" (ByVal hSys As Long, ByVal Direction As Byte)
                                        As Long
Declare Function ReadPort Lib "RsDio.dll" (ByVal hSys As Long, pReadData As Byte) As Long
Declare Function WritePort Lib "RsDio.dll" (ByVal hSys As Long, ByVal WriteData As Byte) As Long
Declare Function IntPostMessage Lib "RsDio.dll" (ByVal hSys As Long, ByVal IntBit As Byte,
                                        ByVal IntEdge As Byte, ByVal hwnd As Long) As Long
```

VB6.0 で OCX を使用して呼び出す場合

VB6.0 の[プロジェクト]-[コンポーネント]より「RsDio ActiveX コントロールモジュール」にチェックを入れます。



フォーム上に OCX コントロールを貼り付けます。



VB からのポート確認ライブラリ呼び出し

Visual Basic のアプリケーションから、製品に添付された DLL(rsportui.dll ※64bit 版 OS の場合は rsportui64.dll)の API を呼び出すには、関数についての呼び出し宣言を行い、それらを参照する必要があります。

VB6.0 サンプルでの宣言 (RSPORTUI.bas) [32bit 用]

```
Declare Function GetComPort Lib "rsportui.dll" (ComPort As Any, ByVal ComPortSize As Byte)
                                                As Byte
Declare Function GetBoardIdCableNo Lib "rsportui.dll" (ByVal ComPort As String, BoardId As Any,
                                                CableNo As Any) As Boolean
Declare Function GetBoardId Lib "rsportui.dll" (BoardId As Any, ByVal BoardIdSize As Byte)
                                                As Byte
Declare Function GetCableNo Lib "rsportui.dll" (ByVal BoardId As Byte, ComPort As Any,
                                                ByVal ComPortSize As Byte) As Boolean
```

VB2010 サンプルでの宣言 (rsportui.vb) [64bit 用]

```
Module rsportui
Declare Function GetComPort Lib "rsportui64.dll" (ByVal ComPort As String ,
                                ByVal ComPortSize As Byte) As Byte
Declare Function GetBoardIdCableNo Lib "rsportui64.dll" (ByVal ComPort As String,
                                ByVal BoardId As String, ByVal CableNo As String) As Boolean
Declare Function GetBoardId Lib "rsportui64.dll" (ByVal BoardId As String ,
                                ByVal BoardIdSize As Byte) As Byte
Declare Function GetCableNo Lib "rsportui64.dll" (ByVal BoardId As Byte, ByVal ComPort As String,
                                ByVal ComPortSize As Byte) As Byte
End Module
```

(3-5) DIO 制御サンプルアプリケーションについて

製品添付の DIO 制御サンプルアプリケーションでは、各ボードの DIO ポートでビット単位の入出力を行うことができます。

また、ビット単位での割り込み検出を行い、割込み検出時の DIO ポートに入力されているデータを読み取ります。



制御する製品のボード ID(DIP スイッチ設定番号)を選択します。

【入出力設定】

[ライト] -- DioOpen()で選択したボード ID の DIO ポートをオープンし、SetDirection()で方向設定を行い、出力設定したビットへ指定した値を WritePort()で書き込みます。

[リード] -- DioOpen()で選択したボード ID の DIO ポートをオープンし、SetDirection()で方向設定を行い、入力設定したビットの値を ReadPort()で読み込みます。

【割り込み設定】

DioOpen()で選択したボード ID の DIO ポートをオープンし、SetDirection()ですべてのビットを入力方向に設定し、IntPostMessage()で割り込みの検出を開始します。

指定した割り込みが検出されると、メッセージがポストされ ReadPort()で全ポートに入力されている値を読み取ります。

(3-6) 設定ポート確認サンプルアプリケーションについて

製品添付の設定ポート確認サンプルアプリケーションでは、製品に割当たっている COM ポート番号から「DIP スイッチ設定番号」と「コネクタ番号」を取得することができます。
また、認識している製品の DIP スイッチ設定番号から各コネクタ番号に割り当たっている COM ポート番号を取得することができます。



【COM ポート番号から情報取得】
GetComPort()を用いて本製品に割り当たっている COM ポート番号を全て列挙します。
GetBoardIdCableNo()を用いてコンボボックスに列挙された COM ポート番号を指定することで、DIP スイッチ設定番号の値とコネクタ番号を呼び出します。

【DIP スイッチ設定番号から情報取得】
GetBoardId()を用いて認識している製品のボード ID の値を全て列挙します。
GetCableNo()を用いてコンボボックスに列挙された DIP スイッチ設定番号の値を指定することで、割り当てられているコネクタ番号 Port1、Port2 の COM ポート番号を列挙します。

(注意)
関数を機能させるためには、すべてのポートが有効になっている必要がありますので、デバイスマネージャ上で無効としないでください。

第4章 通信サンプルアプリケーション

(4-1) 通信サンプルアプリケーションの構成について

製品添付の CD-ROM の[CommSample]フォルダ内の構成は次のようになります。

- VB6 フォルダ・・・VisualBasic6.0 サンプル
- VC6 フォルダ・・・VisualC++6.0 サンプル
- VB2010 フォルダ・・・VisualBasic2010 サンプル
- VC2010 フォルダ・・・VisualC++2010 サンプル

※ 製品添付のサンプルソースコードは RS-232C 用となっております。
RS-422A/485 でご使用の場合は、次のソースコード内での定義を変更してください。

[VB6] -- Serial.bas
[VC6] -- WIN32COM.C
[VB2010] -- Serial.vb
[VC2010] -- WINCOM.C

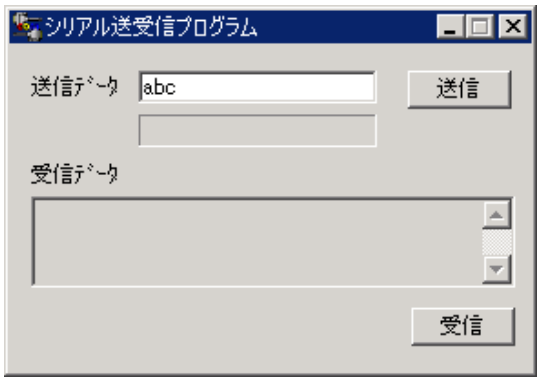
(4-2) 通信サンプルアプリケーションについて

通信サンプルアプリケーションは ASCII 文字列を送受信する簡易プログラムです。

使用するポートを選択します。



入力した ASCII 文字列の送信、および接続先から送信されたデータの受信を行います。



通信サンプルプログラム抜粋(VC)

(Visual Basic についてはサンプルソース内をご参照ください)

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMessage, WPARAM wParam, LPARAM lParam)
{
    switch (uMessage)
    {
         case WM_CREATE:
                  // COMﾎﾟｰﾄｵｰﾌﾟﾝ
                  hComPort = CreateFile( lpszComName,
                                       GENERIC_READ|GENERIC_WRITE,
                                       0,
                                       NULL,
                                       OPEN_EXISTING,
                                       FILE_FLAG_OVERLAPPED,
                                       NULL);
                  if ( hComPort == INVALID_HANDLE_VALUE) {
                           // ﾊﾝﾄﾞﾙｴﾗｰ
                           ShowError("COM Open Error.");
                           return FALSE;
                  }
                  // DCB 設定
                  memset(&dcb,0,sizeof(dcb));
                  dcb.DCBlength = sizeof(dcb);
                  dcb.BaudRate = 9600;
                  dcb.fBinary = 1;
                  dcb.fDtrControl = DTR_CONTROL_ENABLE;
                  dcb.fOutxCtsFlow = 1;
                  dcb.fRtsControl = DTR_CONTROL_HANDSHAKE;
                  dcb.Parity = NOPARITY;
                  dcb.StopBits = ONESTOPBIT;
                  dcb.ByteSize = 8;
                  // 新たに通信ﾊﾟﾗﾒｰﾀを設定する
                  if ( !SetCommState( hComPort, &dcb) )
                           ShowError("Set COM parameter error.");
                  break;

         case WM_COMMAND:
         switch (wParam)
         {
                  case IDB_TXDATA:
                           // 送信ﾃﾞｰﾀ取得
                           memset( TxBuf, 0x00, sizeof( TxBuf ) );
                           GetDlgItemText( hWnd, IDE_TXDATA, TxBuf, sizeof(TxBuf) );
                           SetDlgItemText( hWnd, IDS_TXDATA, TxBuf );
                           SetDlgItemText( hWnd, IDE_TXDATA, "" );
                           nToWrite = strlen(TxBuf);
                           // COMﾎﾟｰﾄにﾃﾞｰﾀ送信
                           iRet = WriteFile ( hComPort, TxBuf, nToWrite, &dwBytesWrote, &ov);
                           if ( iRet == 0 ){
                                    WaitForSingleObject(ov.hEvent ,1000);
                           }
                           break;
/* 次ページに続く */
```

```
                    case IDB_RXDATA:
                              // 受信ｽﾚｯﾄﾞを作成します
                              hThread = CreateThread( NULL,
                                                      0,
                                                      (LPTHREAD_START_ROUTINE)ReadThread,
                                                      hWnd,
                                                      0,
                                                      &ThreadId);
                              break;
          }
          break;
          case WM_DESTROY:
                    if(hThread != NULL){
                              CloseHandle( hThread );
                              fReadThread = FALSE;
                    }
                    PostQuitMessage(0);
                    break;
          default:
                    return DefWindowProc(hWnd, uMessage, wParam, lParam);
          }return 0;
}

DWORD WINAPI ReadThread( LPVOID lpParameter )
{
    // ﾊﾞｲﾄ受信ｲﾍﾞﾝﾄを待って受信ﾃﾞｰﾀを取り出し格納
    while( fReadThread ){
      // ｲﾍﾞﾝﾄを待つ
      WaitCommEvent(hComPort,&dwEvent, &ov) ;
      if ( WaitForSingleObject(ov.hEvent,INFINITE) == WAIT_OBJECT_0){
        do{
          memset(RxBuf,0,sizeof(RxBuf));
          if( !ReadFile( hComPort,RxBuf,sizeof(RxBuf),&dwBytesRead,&ov) ){
            if( (WinError = GetLastError()) == ERROR_IO_PENDING ){
              if( !GetOverlappedResult(hComPort,&ov,&dwBytesRead,TRUE) ){
                ShowError("GetOverlappedResult failed");
                break;
              }
            }
            else{
             if(WinError != ERROR_INVALID_HANDLE){
               ShowError("ReadFile failed");
               break;
             }
            }
          }
          if ( dwBytesRead > 0 ){
            // 受信ﾃﾞｰﾀ表示
            RxBuf[dwBytesRead] = 0x00;
            SetDlgItemText( hWnd, IDS_RXDATA, RxBuf );
          }
        }while (dwBytesRead > 0 && fReadThread != FALSE);
      }
    }return 0L;
}
```

ラトックシステム株式会社

製品に対するお問い合わせ

REX-PCI70D の技術的なご質問やご相談の窓口を用意していますのでご利用ください。

ラトックシステム株式会社
I&L サポートセンター
〒556-0012
大阪市浪速区敷津東 1-6-14　朝日なんばビル
TEL. 06-6633-6741
FAX. 06-6633-8285

〈サポート受付時間〉
月曜－金曜（祝祭日は除く）AM 10:00 - PM 1:00
PM 2:00 - PM 5:00

また、インターネットのホームページでも受け付けています。
HomePage ➲ http://www.ratocsystems.com

個人情報取り扱いについて
ご連絡いただいた氏名、住所、電話番号、メールアドレス、その他の個人情報は、お客様への回答など本件に関わる業務のみに利用し、他の目的では利用致しません。

ご注意

☑本書の内容については、将来予告なしに変更することがあります。
☑本書の内容につきましては万全を期して作成しましたが、万一ご不審な点や誤りなどお気づきになられましたらご連絡願います。
☑本製品および本製品添付のマニュアルに記載されている会社名および製品名は、各社の商品または登録商標です。
☑運用の結果につきましては、責任を負いかねますので、予めご了承願います。

FAX：06-6633-8285

REX-PCI70D 質問用紙

●下記情報をご記入願います。

法人登録の方のみ	会社名・学校名			
	所属部署			
ご担当者名				
E-Mail				
住所	〒			
TEL		FAX		
製品型番		シリアルNo.		
ご購入情報	販売店名		ご購入日	

●下記運用環境情報とお問い合わせ内容をご記入願います。

【パソコン/マザーボードのメーカ名と機種名】
【ご利用の OS】
【接続機器】
【お問合せ内容】
【添付資料】

⚠ 個人情報取り扱いについて

ご連絡いただいた氏名、住所、電話番号、メールアドレス、その他の個人情報は、お客様への回答など本件に関わる業務のみに利用し、他の目的では利用致しません。



http://www.ratocsystems.com/

ⓒRATOC Systems, Inc. All rights reserved.