



REX-5055

16Bit Digital I/O PC CARD

ユーザーズマニュアル

2002年3月

第7.0版

 **RATOC**
Systems, Inc.
ラトックシステム株式会社

■ ■ ■ ■ ■ ■ ■ ■		
■ ■ ■ ■ ■ ■ ■ ■		
■ ■ ■ ■ ■ ■ ■ ■		
● ■ ■ ■ ■ ■ ■ ■	第1章 はじめに -----	1- 1
● ■ ■ ■ ■ ■ ■ ■	(1-1) 製品概要	1- 1
● ■ ■ ■ ■ ■ ■ ■	(1-2) 添付品	1- 2
● ■ ■ ■ ■ ■ ■ ■	(1-3) ハードウェア仕様	1- 3
● ■ ■ ■ ■ ■ ■ ■	(1-3-1) コネクタピンアサイン	1- 3
● ■ ■ ■ ■ ■ ■ ■	(1-3-2) 入出力回路仕様	1- 4
● ■ ■ ■ ■ ■ ■ ■	(1-3-3) レジスタ仕様	1- 5
● ■ ■ ■ ■ ■ ■ ■	第2章 Windows95/98/Me 解説 -----	2- 1
● ■ ■ ■ ■ ■ ■ ■	(2-1) インストール	2- 1
● ■ ■ ■ ■ ■ ■ ■	(2-2) PC カード設定内容の確認	2- 6
● ■ ■ ■ ■ ■ ■ ■	(2-3) アンインストール	2- 8
● ■ ■ ■ ■ ■ ■ ■	(2-4) C 言語 API ライブラリ解説	2- 9
● ■ ■ ■ ■ ■ ■ ■	(2-4-1) Visual C によるアプリケーション開発	2- 9
● ■ ■ ■ ■ ■ ■ ■	(2-4-2) Visual C サンプルプログラム	2-17
● ■ ■ ■ ■ ■ ■ ■	(2-5) BASIC 言語 API ライブラリ解説	2-22
● ■ ■ ■ ■ ■ ■ ■	(2-5-1) Visual BASIC によるアプリケーション開発	2-22
● ■ ■ ■ ■ ■ ■ ■	(2-5-2) Visual BASIC サンプルプログラム	2-28
● ■ ■ ■ ■ ■ ■ ■	第3章 Windows2000/XP 解説 -----	3- 1
● ■ ■ ■ ■ ■ ■ ■	(3-1) インストール	3- 1
● ■ ■ ■ ■ ■ ■ ■	(3-2) PC カード設定内容の確認	3- 5
● ■ ■ ■ ■ ■ ■ ■	(3-3) アンインストール	3- 6
● ■ ■ ■ ■ ■ ■ ■	(3-4) C 言語 API ライブラリ解説	3- 8
● ■ ■ ■ ■ ■ ■ ■	(3-4-1) Visual C によるアプリケーション開発	3- 8
● ■ ■ ■ ■ ■ ■ ■	(3-4-2) Visual C サンプルプログラム	3-13
● ■ ■ ■ ■ ■ ■ ■	(3-5) BASIC 言語 API ライブラリ解説	3-19
● ■ ■ ■ ■ ■ ■ ■	(3-5-1) Visual BASIC によるアプリケーション開発	3-19
● ■ ■ ■ ■ ■ ■ ■	(3-5-2) カスタムコントロール	3-24
● ■ ■ ■ ■ ■ ■ ■	(3-5-3) Visual BASIC サンプルプログラム	3-28
● ■ ■ ■ ■ ■ ■ ■	第4章 MS-DOS/Windows3.1 解説 -----	4- 1
● ■ ■ ■ ■ ■ ■ ■	(4-1) MS-DOS/Windows3.1 でのインストール	4- 1
● ■ ■ ■ ■ ■ ■ ■	(4-1-1) イネーブラのインストール	4- 1
● ■ ■ ■ ■ ■ ■ ■	(4-1-2) DOS/V 版カードサービス対応イネーブラを使用する場合	4- 2
● ■ ■ ■ ■ ■ ■ ■	(4-1-3) PC-9800 シリーズ版カードサービス対応イネーブラを使用する場合	4- 5
● ■ ■ ■ ■ ■ ■ ■	(4-1-4) DOS/V 版ポイントイネーブラを使用する場合	4- 7
● ■ ■ ■ ■ ■ ■ ■	(4-2) MS-DOS ライブラリ解説	4- 9
● ■ ■ ■ ■ ■ ■ ■	(4-2-1) MS-DOS ライブラリ	4- 9
● ■ ■ ■ ■ ■ ■ ■	(4-2-2) MS-DOS サンプルプログラム	4-10
● ■ ■ ■ ■ ■ ■ ■	(4-3) Windows3.1 16Bit ライブラリ解説	4-20
● ■ ■ ■ ■ ■ ■ ■	(4-3-1) Windows3.1 アプリケーション	4-20
● ■ ■ ■ ■ ■ ■ ■	(4-3-2) 16Bit DLL 関数仕様	4-23
● ■ ■ ■ ■ ■ ■ ■	第5章 WindowsNT 解説 -----	5- 1
● ■ ■ ■ ■ ■ ■ ■	(5-1) インストール	5- 1
● ■ ■ ■ ■ ■ ■ ■	(5-2) Visual C によるアプリケーション開発	5- 3
● ■ ■ ■ ■ ■ ■ ■	(5-2-1) DLL 関数仕様	5- 4
● ■ ■ ■ ■ ■ ■ ■	(5-2-2) Visual C サンプルプログラム	5- 8
● ■ ■ ■ ■ ■ ■ ■	(5-3) Visual BASIC によるアプリケーション開発	5-11
● ■ ■ ■ ■ ■ ■ ■	(5-3-1) Visual BASIC サンプルプログラム	5-12
● ■ ■ ■ ■ ■ ■ ■	第6章 オプションアイソレーションユニット -----	6- 1
● ■ ■ ■ ■ ■ ■ ■	(6-1) REX-IPI16 製品概要	6- 1
● ■ ■ ■ ■ ■ ■ ■	(6-2) REX-IPO16 製品概要	6- 2
● ■ ■ ■ ■ ■ ■ ■	(6-3) RCL-TRM48 製品概要	6- 3

第1章 はじめに

(1-1) 製品概要

REX-5055 は DOS/V・PC-9800 シリーズ対応の TTL レベル 16 ビットパラレル入出力 PC カードです。PC Card Standard (Release 2.1 クラスの 16Bit PC Card) に準拠した仕様になっており、PC/AT 互換機及び NEC PC-9800 シリーズで使用することができます。本製品の主なハードウェア仕様及びソフトウェア仕様を下表に示します。

ハードウェア仕様

項目	内容				
入出力ビット数	16 ビット(8 ビット単位で入出力方向設定) 割り込み入力:入力ポートの 1 ポートを割り当て可				
入出力仕様	TTL レベル(74HCT245 相当)				
	<table border="1"> <tr> <td>最大入出力電流</td> <td>入力: ±20mA 出力: ±35mA</td> </tr> <tr> <td>標準入出力電圧</td> <td>0V/5V</td> </tr> </table>	最大入出力電流	入力: ±20mA 出力: ±35mA	標準入出力電圧	0V/5V
	最大入出力電流	入力: ±20mA 出力: ±35mA			
標準入出力電圧	0V/5V				
(注) 上記定格は 1 ポート当たりの仕様になります。					
入出力コネクタ	セントロニクス 24 ピンコネクタ付き ケーブル長: 約 50 cm (参考) PC カード側コネクタ: NX-25P ヒロセ電機製相当品使用				

ソフトウェア仕様

項目	内容
カードサービス版イネーブラ	DOS/V 版・PC-9800 シリーズ版
ポイントイネーブラ	DOS/V 版
MS-DOS 対応ソフト	Microsoft C 対応ライブラリ サンプルソフト
Windows3.1 対応ソフト	Visual C/Visual BASIC 対応 16Bit DLL Visual C/Visual BASIC サンプルソフト
Windows95/98/Me 対応ソフト	セットアップ用カード情報ファイル Visual C/Visual BASIC 対応 32Bit DLL Visual C/Visual BASIC サンプルソフト
WindowsNT 対応ソフト	セットアッププログラム Visual C/Visual BASIC 対応 32Bit DLL Visual C/Visual BASIC サンプルソフト
Windows2000 /XP 対応ソフト	セットアップ用カード情報ファイル Visual C/Visual BASIC 対応 32Bit DLL Visual C/Visual BASIC サンプルソフト

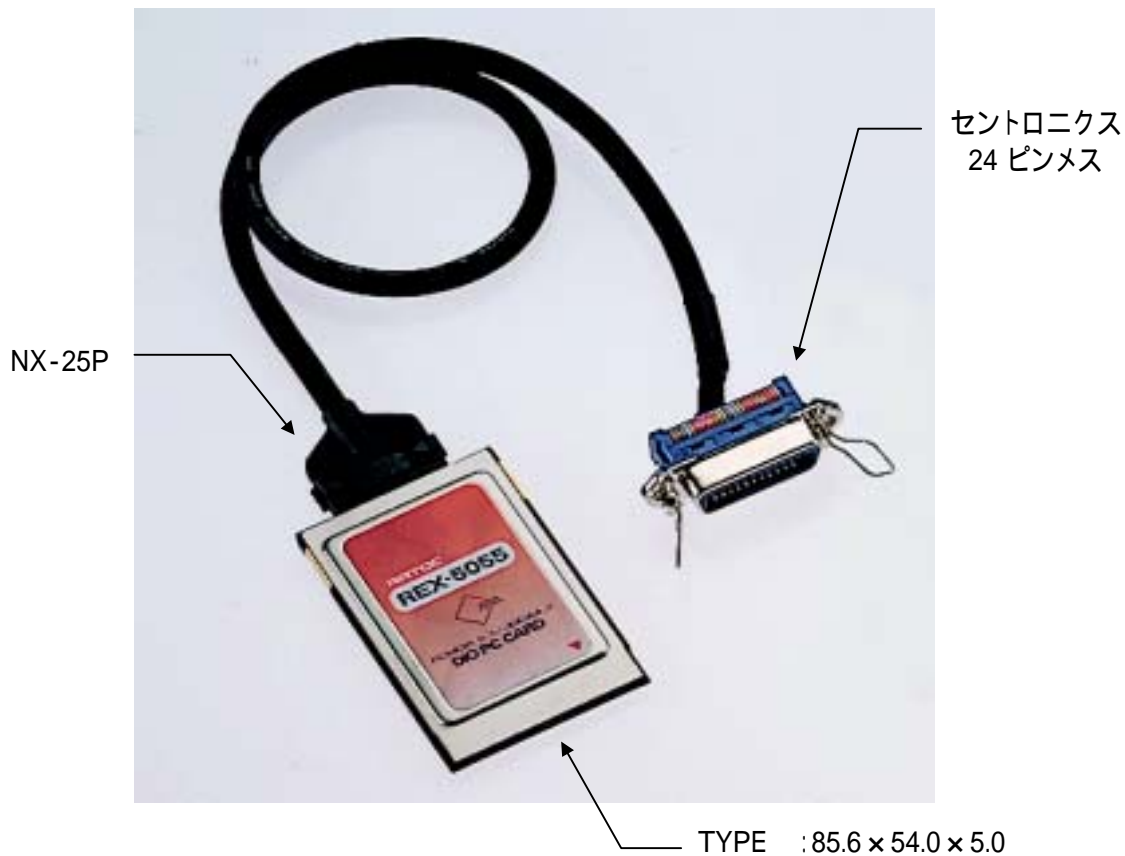
(1-2) 添付品

製品には PC カードと下記添付品が添付されています。ご使用前にご確認願います。

- セントロニクス 24 ピンコネクタ付きケーブル
- セントロニクス 24 ピンオス側コネクタ
- セットアップ・ライブラリディスク 1.44MB 3.5" (3 枚)
- ユーザーズマニュアル
- ご愛用者登録はがき・保証書

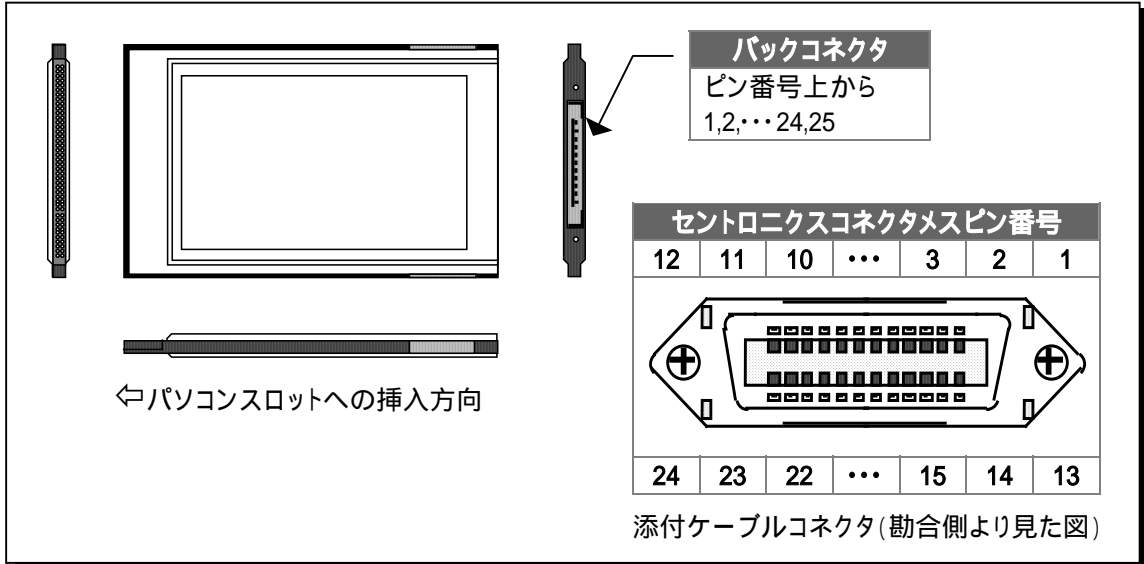
◆*注意*◆

ご愛用者カードは保証書を切り離した後、必要事項を記入の上、必ずご返送ください。ご返送頂けない場合、弊社からのバージョンアップ等のサポートサービスは受けられなくなりますのでご注意ください。



(1-3) ハードウェア仕様

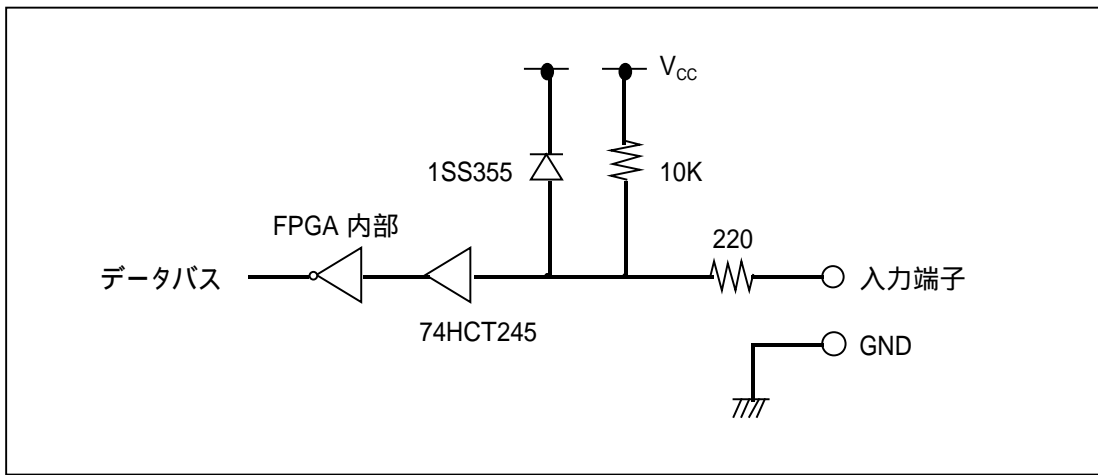
(1-3-1) コネクタピンアサイン



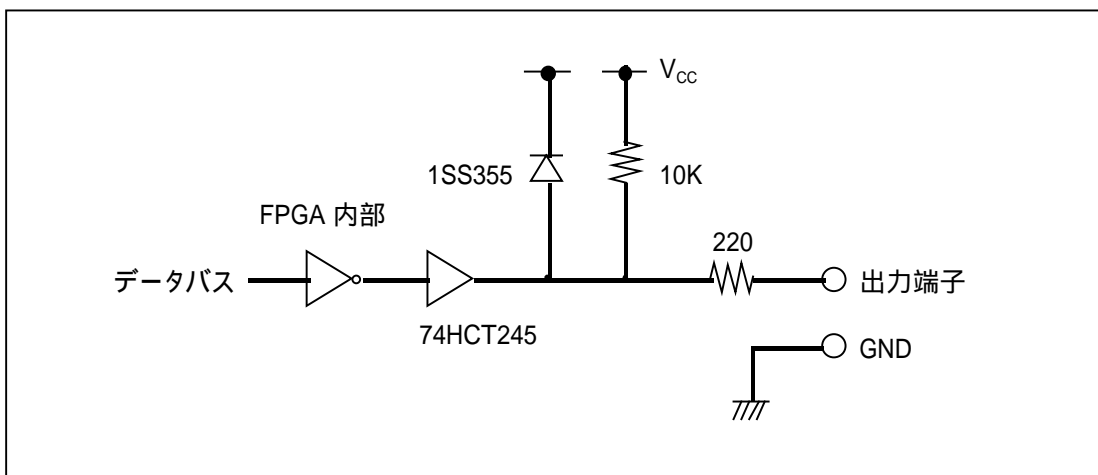
コネクタピン番号	信 号 名	
	カードバックコネクタ側	セントロニクスコネクタ側
1	GND	V _{CC}
2	PIO 0	NC
3	PIO 1	PIO 15
4	PIO 2	PIO 13
5	PIO 3	PIO 11
6	GND	PIO 9
7	GND	V _{CC}
8	PIO 4	NC
9	PIO 5	PIO 7
10	PIO 6	PIO 5
11	PIO 7	PIO 3
12	GND	PIO 1
13	GND	GND
14	PIO 8	NC
15	PIO 9	PIO 14
16	PIO 10	PIO 12
17	PIO 11	PIO 10
18	GND	PIO 8
19	GND	GND
20	PIO 12	NC
21	PIO 13	PIO 6
22	PIO 14	PIO 4
23	PIO 15	PIO 2
24	GND	PIO 0
25	V _{CC}	

(1-3-2) 入出力回路仕様

PIO0-PIO15 入力回路 (TTL レベル負論理)



PIO0-PIO15 出力回路 (TTL レベル負論理)



(1-3-3) レジスタ仕様

REX-5055 DIO PC カードは、ベースアドレスから連続した4バイトのレジスタがマッピングされます。コントロールレジスタ及び割込マスクレジスタは、必ずバイトアクセスしてください。データレジスタは、バイトアクセスまたはワードアクセスのどちらでも可能です。

OFFSET	REX-5055 レジスタ仕様																																																																																												
Base+0	コントロールレジスタ [WRITE]																																																																																												
	B7	B6	B5	B4	B3	B2	B1	B0																																																																																					
	IMD	IR3	IR2	IR1	IR0	-	DIRH	DIRL																																																																																					
	ビット	内 容																																																																																											
	DIRL	PIO0-PIO7 のデータ入出力方向を設定します。 0:入力 1:出力																																																																																											
	DIRH	PIO8-PIO15 のデータ入出力方向を設定します。 0:入力 1:出力																																																																																											
	IR0-IR3	割り込みソースビットを指定します。 IR0-IR3 で指定された PIO ポートに IMD で指定された信号の立ち下がり・立ち上がりがあった場合、割込ステータスレジスタの IntFlag ビットがセットされます。																																																																																											
		<table border="1"> <thead> <tr> <th>割り込みソース信号</th> <th>IR3</th> <th>IR2</th> <th>IR1</th> <th>IR0</th> </tr> </thead> <tbody> <tr><td>PIO 0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>PIO 1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>PIO 2</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>PIO 3</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>PIO 4</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>PIO 5</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>PIO 6</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>PIO 7</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>PIO 8</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>PIO 9</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>PIO 10</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>PIO 11</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>PIO 12</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>PIO 13</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>PIO 14</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>PIO 15</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>							割り込みソース信号	IR3	IR2	IR1	IR0	PIO 0	0	0	0	0	PIO 1	0	0	0	1	PIO 2	0	0	1	0	PIO 3	0	0	1	1	PIO 4	0	1	0	0	PIO 5	0	1	0	1	PIO 6	0	1	1	0	PIO 7	0	1	1	1	PIO 8	1	0	0	0	PIO 9	1	0	0	1	PIO 10	1	0	1	0	PIO 11	1	0	1	1	PIO 12	1	1	0	0	PIO 13	1	1	0	1	PIO 14	1	1	1	0	PIO 15	1	1	1	1
割り込みソース信号	IR3	IR2	IR1	IR0																																																																																									
PIO 0	0	0	0	0																																																																																									
PIO 1	0	0	0	1																																																																																									
PIO 2	0	0	1	0																																																																																									
PIO 3	0	0	1	1																																																																																									
PIO 4	0	1	0	0																																																																																									
PIO 5	0	1	0	1																																																																																									
PIO 6	0	1	1	0																																																																																									
PIO 7	0	1	1	1																																																																																									
PIO 8	1	0	0	0																																																																																									
PIO 9	1	0	0	1																																																																																									
PIO 10	1	0	1	0																																																																																									
PIO 11	1	0	1	1																																																																																									
PIO 12	1	1	0	0																																																																																									
PIO 13	1	1	0	1																																																																																									
PIO 14	1	1	1	0																																																																																									
PIO 15	1	1	1	1																																																																																									
	IMD	割込トリガーモードを設定します。 0:立下りエッジ 1:立上りエッジ																																																																																											

OFFSET	REX-5055 レジスタ仕様																												
Base+1	割り込みマスクレジスタ[WRITE] <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>B7</th> <th>B6</th> <th>B5</th> <th>B4</th> <th>B3</th> <th>B2</th> <th>B1</th> <th>B0</th> </tr> </thead> <tbody> <tr> <td>IrEbl</td> <td>-</td> <td>-</td> <td>-</td> <td>ICLRW3</td> <td>ICLRW2</td> <td>ICLRR3</td> <td>ICLRR2</td> </tr> </tbody> </table> <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>ビット</th> <th>内 容</th> </tr> </thead> <tbody> <tr> <td>B0:ICLRR2</td> <td>割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトをリードした時クリア</td> </tr> <tr> <td>B1:ICLRR3</td> <td>割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをリードした時クリア</td> </tr> <tr> <td>B2:ICLRW2</td> <td>割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトにライトした時クリア</td> </tr> <tr> <td>B3:ICLRW3</td> <td>割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをライトした時クリア</td> </tr> <tr> <td>B7:IrEbl</td> <td>割込モードを設定します。 0 ディセーブル(リセット時のデフォルト値) 1 コントロールレジスタの IR0-IR3 で指定された PIO ポートに入力があった時、CPU に割込要求を発生します。</td> </tr> </tbody> </table>	B7	B6	B5	B4	B3	B2	B1	B0	IrEbl	-	-	-	ICLRW3	ICLRW2	ICLRR3	ICLRR2	ビット	内 容	B0:ICLRR2	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトをリードした時クリア	B1:ICLRR3	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをリードした時クリア	B2:ICLRW2	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトにライトした時クリア	B3:ICLRW3	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをライトした時クリア	B7:IrEbl	割込モードを設定します。 0 ディセーブル(リセット時のデフォルト値) 1 コントロールレジスタの IR0-IR3 で指定された PIO ポートに入力があった時、CPU に割込要求を発生します。
B7	B6	B5	B4	B3	B2	B1	B0																						
IrEbl	-	-	-	ICLRW3	ICLRW2	ICLRR3	ICLRR2																						
ビット	内 容																												
B0:ICLRR2	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトをリードした時クリア																												
B1:ICLRR3	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをリードした時クリア																												
B2:ICLRW2	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの下位バイトにライトした時クリア																												
B3:ICLRW3	割込ステータスレジスタの IntFlag ビットクリア条件を設定します。 1 データレジスタの上位バイトをライトした時クリア																												
B7:IrEbl	割込モードを設定します。 0 ディセーブル(リセット時のデフォルト値) 1 コントロールレジスタの IR0-IR3 で指定された PIO ポートに入力があった時、CPU に割込要求を発生します。																												
Base+1	割り込みステータスレジスタ[READ] <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>B7</th> <th>B6</th> <th>B5</th> <th>B4</th> <th>B3</th> <th>B2</th> <th>B1</th> <th>B0</th> </tr> </thead> <tbody> <tr> <td>IntFlag</td> <td>-</td> <td>-</td> <td>-</td> <td>ICLRW3</td> <td>ICLRW2</td> <td>ICLRR3</td> <td>ICLRR2</td> </tr> </tbody> </table> <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>ビット</th> <th>内 容</th> </tr> </thead> <tbody> <tr> <td>B3-B0</td> <td>ライトした値がそのままリードバックされます。</td> </tr> <tr> <td>B7</td> <td>IntFlag: 割込要求の有無を示します。 0 割込要求はありません。 1 割込要求が発生しています。</td> </tr> </tbody> </table>	B7	B6	B5	B4	B3	B2	B1	B0	IntFlag	-	-	-	ICLRW3	ICLRW2	ICLRR3	ICLRR2	ビット	内 容	B3-B0	ライトした値がそのままリードバックされます。	B7	IntFlag: 割込要求の有無を示します。 0 割込要求はありません。 1 割込要求が発生しています。						
B7	B6	B5	B4	B3	B2	B1	B0																						
IntFlag	-	-	-	ICLRW3	ICLRW2	ICLRR3	ICLRR2																						
ビット	内 容																												
B3-B0	ライトした値がそのままリードバックされます。																												
B7	IntFlag: 割込要求の有無を示します。 0 割込要求はありません。 1 割込要求が発生しています。																												

OFFSET	REX-5055 レジスタ仕様							
Base+2	データレジスタ下位バイト[READ/WRITE]							
	B7	B6	B5	B4	B3	B2	B1	B0
	PIO 7	PIO 6	PIO 5	PIO 4	PIO 3	PIO 2	PIO 1	PIO 0
	ビット	内 容						
B7-B0	PIO7-PIO0: 外部とのデータの入出力を行います。 コントロールレジスタの DIRL ビットを 1 にセットした場合、出力モードになります。 0 ⇨外部ラインは High(5V) にドライブされます。 1 ⇨外部ラインは Low(0V) にドライブされます。 コントロールレジスタの DIRL ビットを 0 にセットした場合、入力モードになります。 0 ⇨外部ラインから High(5V) が入力されています 1 ⇨外部ラインから Low(0V) が入力されています							
Base+3	データレジスタ上位バイト[READ/WRITE]							
	B7	B6	B5	B4	B3	B2	B1	B0
	PIO 15	PIO 14	PIO 13	PIO 12	PIO 11	PIO 10	PIO 9	PIO 8
	ビット	内 容						
B15-B8	PIO15-PIO8: 外部とのデータの入出力を行います。 コントロールレジスタの DIRH ビットを 1 にセットした場合、出力モードになります。 0 ⇨外部ラインは High(5V) にドライブされます。 1 ⇨外部ラインは Low(0V) にドライブされます。 コントロールレジスタの DIRH ビットを 0 にセットした場合、入力モードになります。 0 ⇨外部ラインから High(5V) が入力されています 1 ⇨外部ラインから Low(0V) が入力されています							

(空白ページ)

第2章 Windows95/98/Me解説

本章では Windows95、Windows98(SecondEdition を含む)および WindowsMe での REX-5055 の使用方法について解説しております。

(2-1) インストール

Windows95 OSR-2^(注1)のリリースにより現在 Windows95 のバージョンには、Windows95 OSR-2 と OSR-2 以前のバージョンがあります。「マイコンピュータ」を右クリックし「プロパティ」情報を表示することによりどちらのバージョンがインストールされているか調べることができます。システム情報が「Microsoft Windows95 4.00.950 a」の場合は OSR-2 以前のバージョンになり、OSR-2 の場合は「Microsoft Windows95 4.00.950 B」となります。ご利用の Windows95 が OSR-2 かそれ以前のバージョンかによりインストールの方法が異なりますので注意してください。

(注1) OSR-2 (OEM Service Release 2) では FAT32、CardBus 等の新しい機能がサポートされています。

Windows95 OSR-2 でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、ハードウェアウィザードが起動し右のデバイスドライバーウィザードのインストールが表示されます。ここでは、次へを選択します。



【2】ドライバーファイル場所の指定

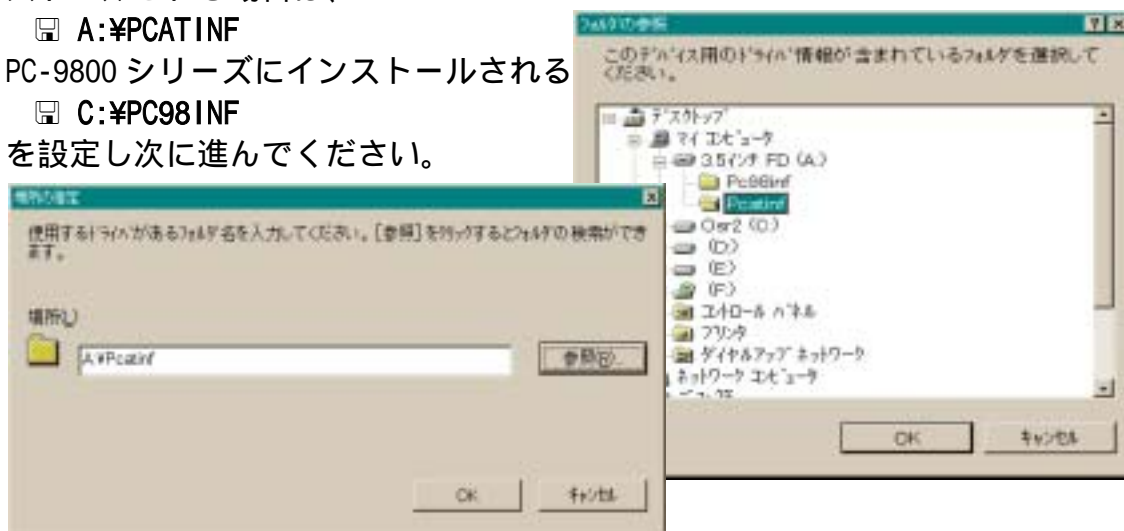
次にドライバーファイル (INF ファイル) の場所を指定します。PC/AT にインストールされる場合は、

A:¥PCATINF

PC-9800 シリーズにインストールされる

C:¥PC98INF

を設定し次に進んでください。



【3】インストールの完了

インストールが正常に完了した場合は、「このデバイス用に更新されたドライバが見つかりました。」のメッセージが表示されますので確認してください。

以上でインストールは完了です。



Windows95 (OSR-2 以前のバージョン) でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、右のハードウェアウィザードが起動します。ここでは「ハードウェアの製造元が提供する「ドライバ」」選択し次に進みます。



【2】配布ファイルコピー元の指定

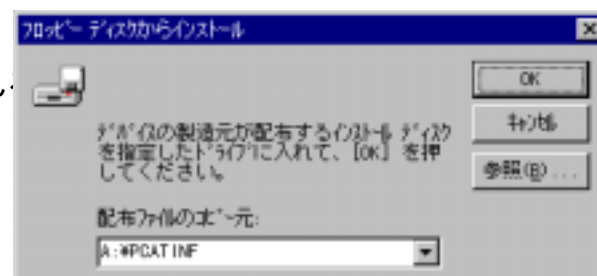
次にドライバーファイル (INF ファイル) の場所を指定します。PC/AT 互換機にインストールされる場合は、

☐ A:¥PCATINF

PC-9800 シリーズにインストールされる場合は、

☐ C:¥PC98INF

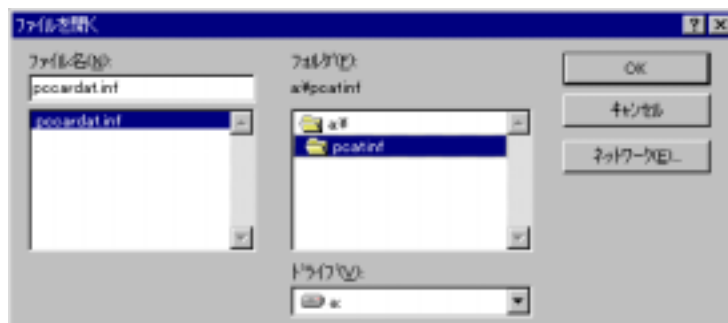
を設定し次に進んでください。



【3】インストールの完了

インストールが正常に行われるとビープ音で完了が通知され、ハードウェアウィザードは自動的に終了します

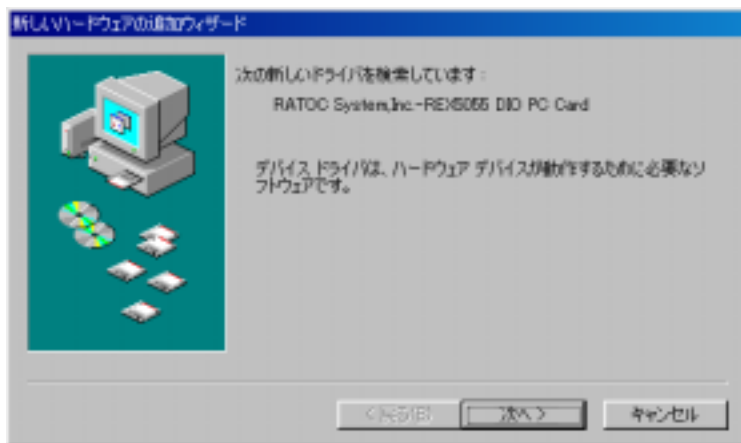
以上でインストールは完了です。



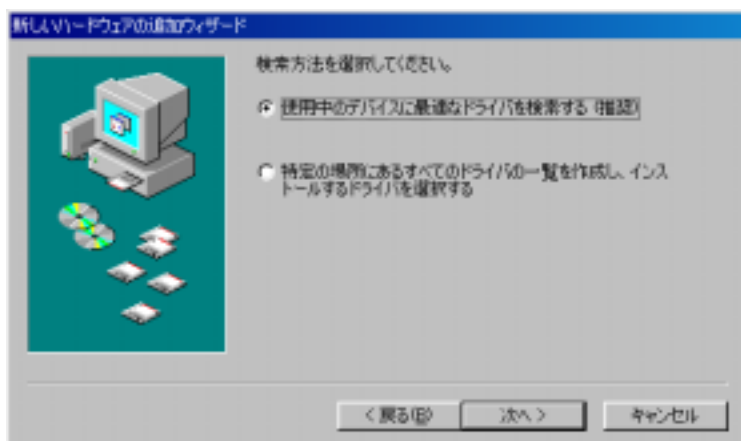
Windows98 でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、ハードウェアウィザードが起動し右のデバイスドライバーウィザードのインストール画面が表示されますので、「次へ」ボタンを押します。



ドライバの検索方法を選択するため「使用中のデバイスに最適なドライバを検索する（推奨）」をチェックし、「次へ」ボタンを押します。



【2】配布ファイルコピー元の指定

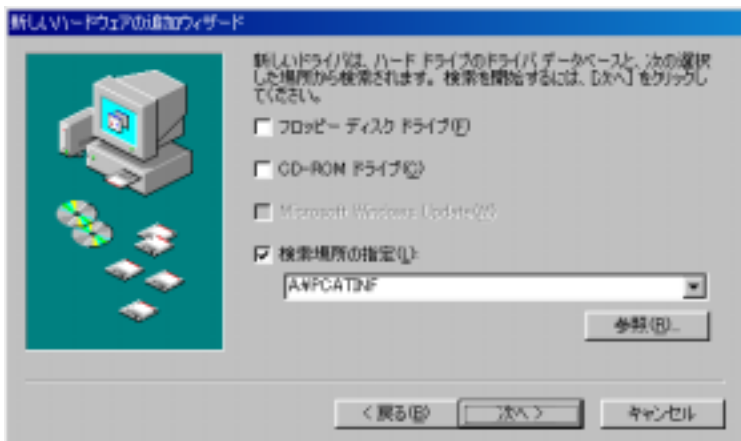
製品添付FDをドライブに挿入し、「検索場所の指定」にチェックします。PC/AT 互換機にインストールされる場合は、

A:¥PCATINF

PC-9800 シリーズにインストールされる場合は、

C:¥PC98INF

を設定し「次へ」ボタンを押します。



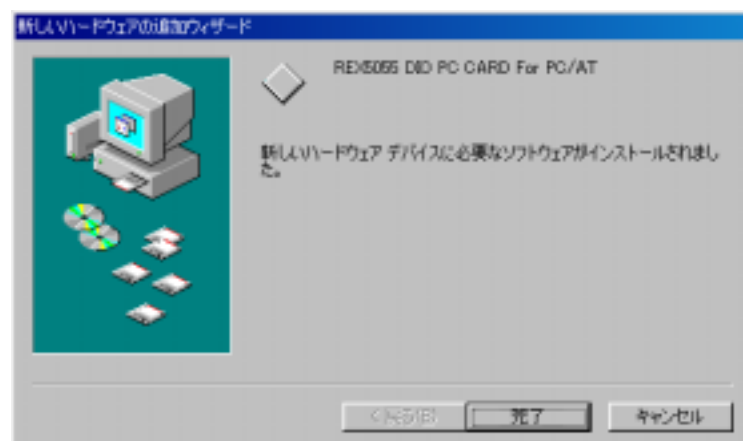
ドライバファイルが検索され、右画面のようにドライバのある場所が表示されますので「次へ」ボタンを押します。



【3】インストールの完了

インストールが正常に行われるとビープ音で完了が通知され、右画面が表示されますので「完了」ボタンを押します。

以上でインストールは完了です。



WindowsMe でのインストール方法


【1】PC カードの挿入

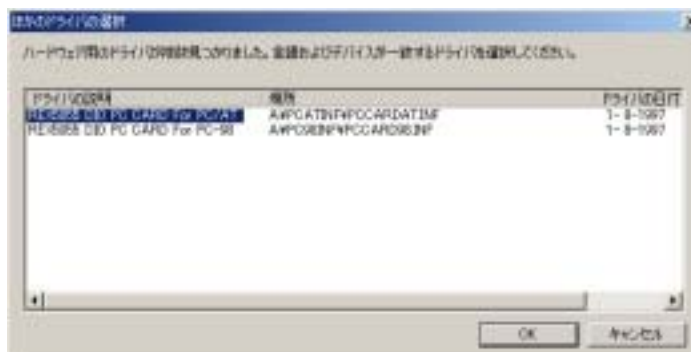
PC カードをスロットに挿入すると、右の新しいハードウェアの追加ウィザードが表示されますので、製品添付の Win95/98/Me 用フロッピーディスクを FD ドライブへ挿入してください。

次に、「適切なドライバを自動的に検索する（推奨）(A)」を選択し「次へ」ボタンを押します。

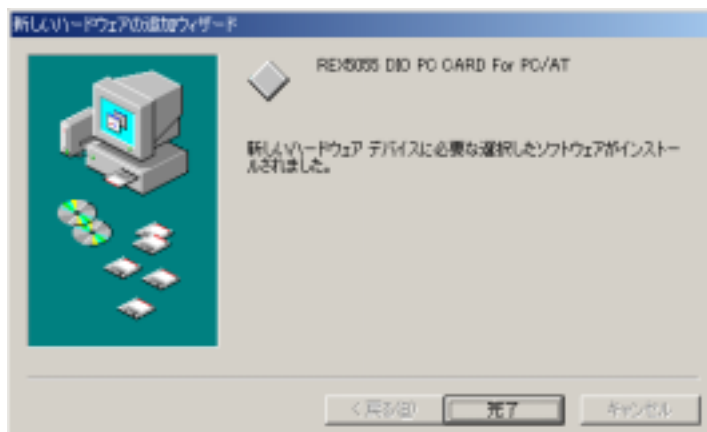


【2】ドライバーファイル場所の指定

右のようにセットアップ情報ファイル (inf ファイル) が、フロッピーディスク上から自動的に検索されますので、 「REX-5055 DIO PC CARD For PC/AT」を選択し、「OK」ボタンを押します。



右の画面が表示されましたら、「完了」ボタンを押します。



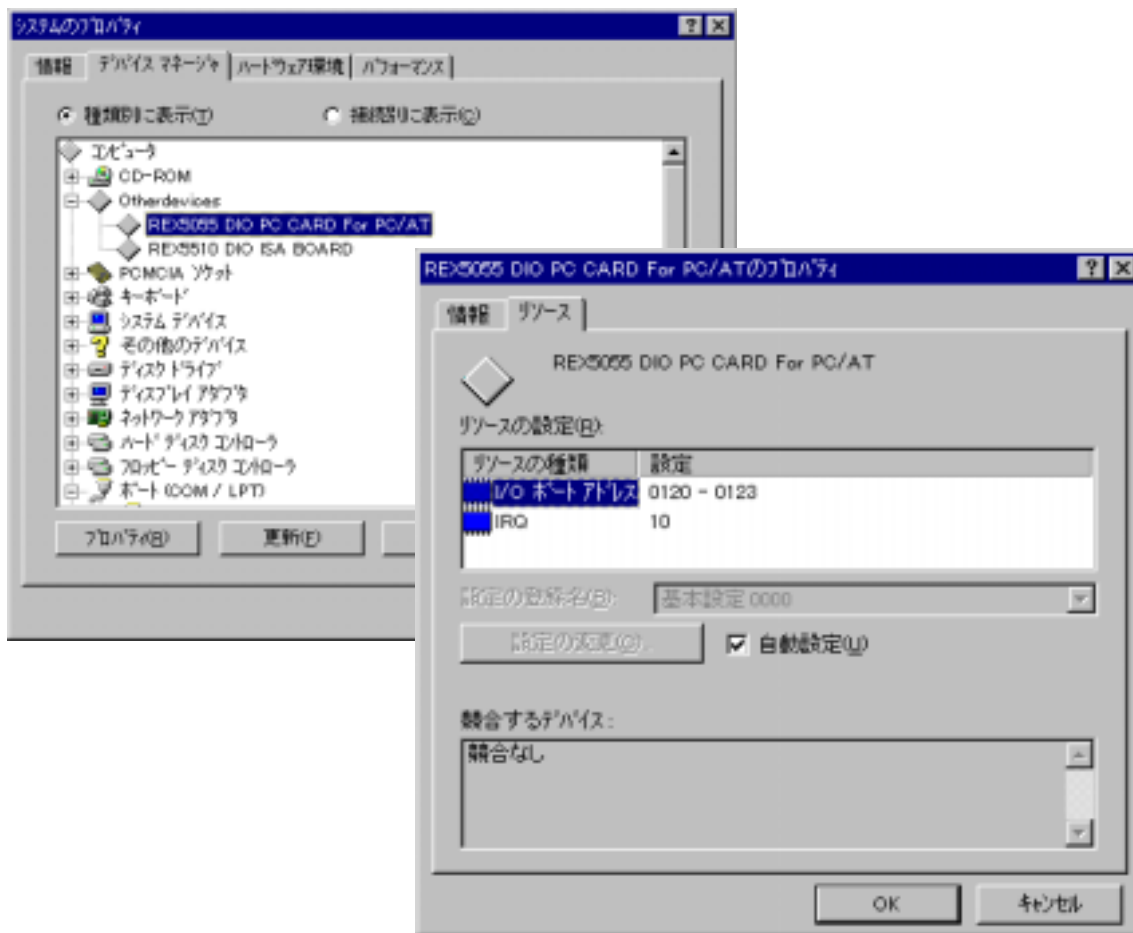
以上で、REX-5055 インストールは終了です。

(2-2) PC カード設定内容の確認

システムプロパティの起動

コントロールパネルのシステムを起動し、デバイスマネージャを選択します。カードの設定が正常に行われていれば、コンピュータのレジストリツリー「Otherdevices」の下に「REX5055 DIO PC CARD For PC/AT(またはPC98)」が登録されます。

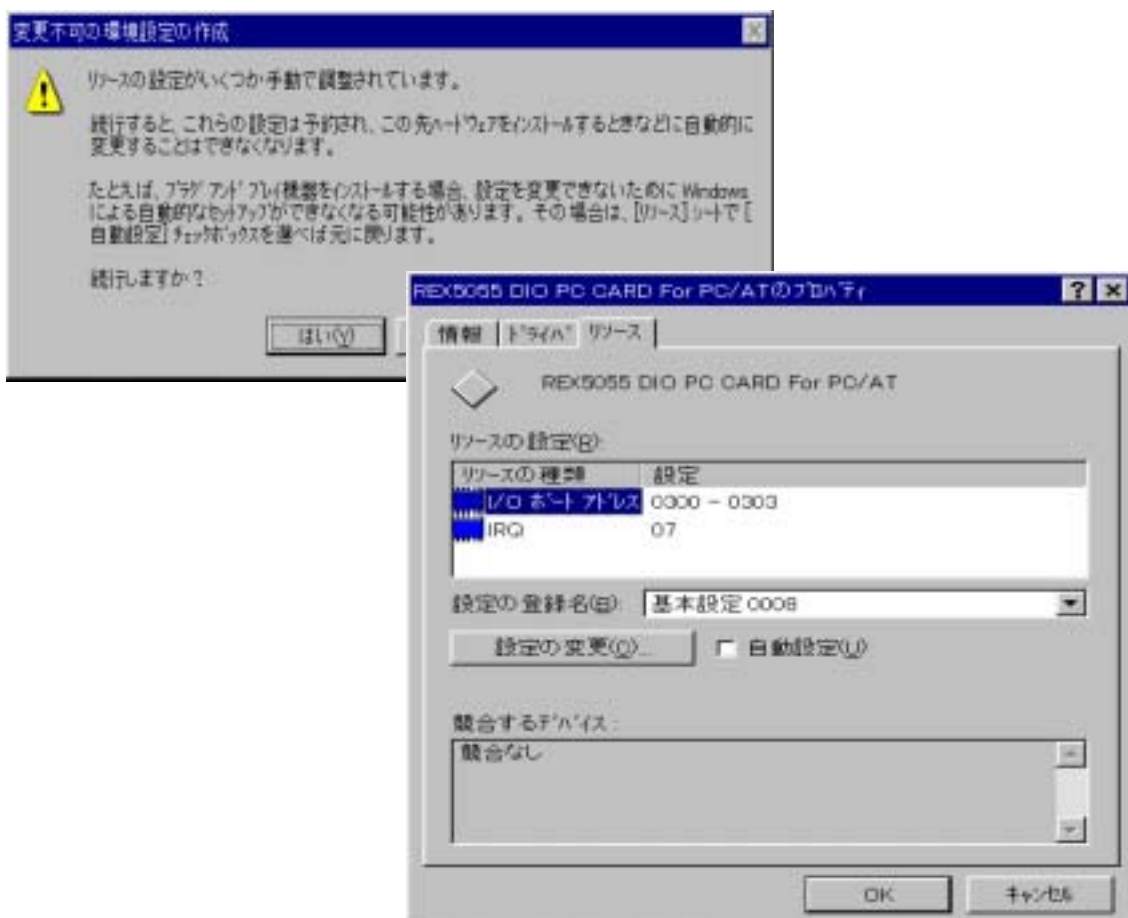
プロパティのリソースタブを選択してI/OポートアドレスおよびIRQの割り当てで競合していないことを確認してください。



リソースの変更

リソースの変更は、自動設定を選択しないようにして、設定の登録名を別の設定に変更します。手動設定を行うと、「変更不可の環境設定の作成」ダイアログが表示されますが、続行「はい」を選択してください。

手動設定したリソースが他のデバイスと競合していなければ、「ピッポッ」というビープ音とともにシステムプロパティ画面に戻ります。もう一度、レジストリ「REX5055 DIO PC CARD For PC/AT」をダブルクリックし、手動設定したリソースが他のデバイスと競合していないことを確認してください。



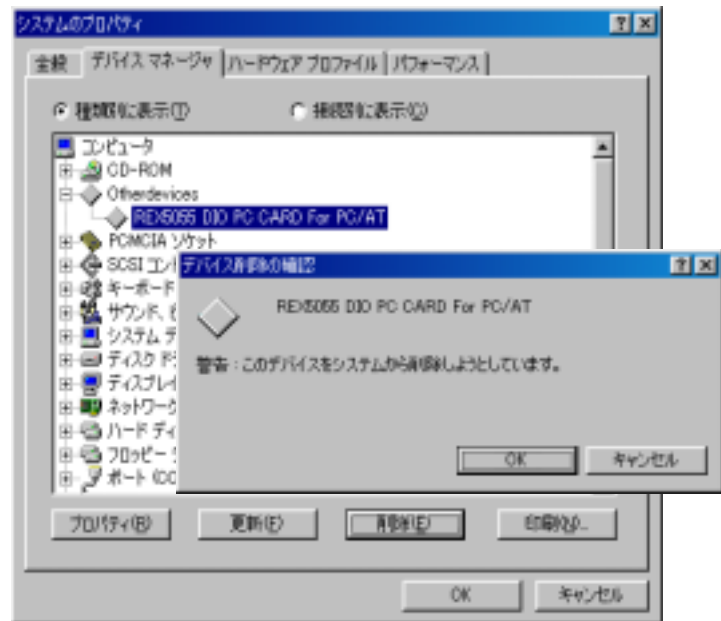
(2-3) アンインストール

カードが正しくインストールされなかった場合は以下の手順でカード情報の削除と INF ファイルの削除を行い、再度、(2-1)の方法でインストールを行ってください。

【1】カード情報の削除

PC カードを挿入した状態で、コントロールパネルからシステムを起動します。

「デバイスマネージャ」のタブを選択し、Otherdevices の下にある「REX-5055 DIO PC CARD For PC/AT (または PC-98)」を選択して「削除」ボタンを押します。デバイス削除の確認が表示されますので、「OK」を押してください。



【2】INF ファイルの削除

「エクスプローラ」を起動し、¥Windows¥Inf¥Other フォルダにある「RATOC System,Inc.PCCARDAT.INF」ファイルを削除してください。



(2-4) C 言語 API ライブラリ解説

(2-4-1) Visual C によるアプリケーション開発

Visual C 4.0 以上を使って REX-5055 DIO PC カードを制御するアプリケーションを開発する場合は、本製品に添付されている 32Bit 版 DLL の関数をコールする必要があります。

Visual C で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、

1. アプリケーションプログラムに DIOLIB32.H ファイルをインクルードする。
2. アプリケーションプログラムのプロジェクトファイルに DIOLIB32.LIB を追加する。
3. アプリケーションの実行ディレクトリまたは WINDOWS¥SYSTEM に、DIOLIB32.DLL ファイルと VR5055D.VXD ドライバーをコピーする。

必要があります。

アプリケーション作成上のアドバイス

PC カードに割り当てられてるリソースを取得する

Windows95/98/Me では、PC カードの Plug&Play がサポートされています。これにより、PC カードが使用する I/O ベースアドレス・IRQ 番号は、カードが挿入された時点でシステムがダイナミックに割り当てを行います。従って、常に同じ I/O ベースアドレス・IRQ 番号がカードに割り当てられる訳ではなく、その時点で使用可能なリソースが割り当てられます。

Windows95/98/Me のダイナミックコンフィギュレーションに対応するために、DLL では GetMyCardResource() をサポートしています。アプリケーションの初期化手続き部分で GetMyCardResource() により PC カードに割り当てられている I/O ベースアドレス・IRQ 番号を取得してください。

PC カードへの入出力

Visual C では、I/O 入出力関数がサポートされていません。これは Windows95 の保護機能に基づいたものと考えられます。従って、PC カードへの I/O 入出力は DLL でサポートされている OutPort()・InPort()・wOutPort()・wInPort() を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、(1)割り込みハンドラからユーザ定義メッセージにより割り込み通知を受け取る方法

(2)非同期プロシージャコールにより割り込み通知を受け取る方法

(3)割り込みハンドラ内で全ての処理を行う方法

があります。高速な処理が要求されるような場合は、(3)の方法で行う必要があります。詳細は、各サンプルプログラムの説明を参照してください。

DLL 関数仕様

GetMyCardResource

リソース情報の取得

書式 BOOL GetMyCardResource(HWND hWnd, LPSTR MyCardName, WORD NameLen, WORD SlotNo, LPWORD IOBase, LPWORD IrqNo)

機能 指定スロットに挿入されているカードが指定のカード名と一致するか否か調べます。一致した場合は、割り当てられている I/O ベースアドレス・IRQ リソース情報を返します。

引数 HWND hWnd ➤ ウィンドウハンドル
 LPSTR MyCardName ➤ カード識別名を示す文字列へのポインタ
 WORD NameLen ➤ 文字列バッファの長さ
 WORD SlotNo ➤ カード挿入スロット番号
 LPWORD IOBase ➤ (出力) I/O リソース情報を格納する変数アドレス
 LPWORD IrqNo ➤ (出力) IRQ リソース情報を格納する変数アドレス

戻値 0 : 正常終了
 -1 : ドライバ呼び出しエラー
 -2 : カードサービスドライババージョンエラー
 -3 : GET_CARD_SERVICES_INFO ファンクションコールサービスエラー
 -4 : GET_FIRST_TUPLE ファンクションコールサービスエラー
 -5 : GET_TUPLE_DATA ファンクションコールサービスエラー
 -6 : GET_CONFIG_INFO ファンクションコールサービスエラー
 -7 : メモリアロケーションエラー
 -8 : スロットにカードが挿入されていない
 -9 : スロットに挿入されているカードは自分のカードと一致しない

サンプル

```
#define MyCardName "REX5055 D10 PC Card" // DIOLIB32.H で定義されています
WORD SlotNo // サーチするスロット番号
WORD MyIOBase; // 取得したベースアドレス
WORD MyIrqNo; // 取得した割り込み番号

BOOL Function( void )
{
  /*スロット0またはスロット1に挿入されている自分のカードのリソース情報を取得 */
  for ( SlotNo = 0; SlotNo < 2; SlotNo++ ){
    Status = GetMyCardResource( hDlg, MyCardName, sizeof(MyCardName), SlotNo,
    &MyIOBase, &MyIrqNo );
    if ( Status == 0 ){
      /* リソース取得正常終了時の処理 */
      return TRUE ;
    }
  }
  /* リソース取得失敗時の処理 */
  return FALSE
}
```

ShowCardUtil

リソース表示ユーティリティ起動

書式 void ShowCardUtil(HWND hWnd)

機能 カードリソース情報表示ユーティリティを起動します。

引数 HWND hWnd ➤ ウィンドウハンドル

戻値 なし

画面



StartHWIntPostMessage

ユーザ定義メッセージ割り込みの開始

書式 BOOL StartHWIntPostMessage(HWND hWnd, WORD MyIOBase, WORD MyIrqNo, BYTE DirDataRegLo, BYTE DirDataRegHi, BYTE IrqPin, BYTE IrqMode, BYTE IntClrMode, WORD StopCount)

機能 ユーザ定義メッセージ版の割り込み処理を開始します。割り込みが発生すると割り込みハンドラからアプリケーションにメッセージを通知します。wParam の上位バイトに PI07-PI00 の読み込み値が、下位バイトに PI015-PI08 の読み込み値がセットされています。また、lParam には割り込み発生時の累計カウント数がセットされています。

引数

HWND hWnd	➤ ユーザアプリケーションのウィンドウハンドル
WORD MyIOBase	➤ カードに割り当てられている I/O ベースアドレス
WORD MyIrqNo	➤ カードに割り当てられている IRQ 番号
BYTE DirDataRegLo	➤ データレジスタ下位バイトの入出力方向 0:入力方向 1:出力方向
BYTE DirDataRegHi	➤ データレジスタ上位バイトの入出力方向 0:入力方向 1:出力方向
BYTE IrqPin	➤ 割り込みソース入力ポート番号 0:PI00 1:PI01 2:PI02 …… 15:PI015
BYTE IrqMode	➤ 割り込みトリガーモード 0:立下りエッジ 1:立上りエッジ
BYTE IntClrMode	➤ 割り込みクリア条件 0:下位バイトリード 1:上位バイトリード 2:下位バイトライト 3:上位バイトライト
WORD StopCount	➤ 割り込み終了回数

戻値 0 : 正常終了
-1 : 割り込み登録エラー
-2 : DeviceIoControl ステータスエラー

サンプル Visual C サンプルプログラム解説参照

EndHWIntPostMessage

ユーザ定義メッセージ割り込みの終了

書式 BOOL EndHWIntPostMessage(void)

機能 ユーザ定義メッセージ版の割り込み処理を終了します。ユーザ定義メッセージによる割り込み処理終了時は、必ず EndHWIntPostMessage() を呼び出して下さい。

引数 なし

戻値 常に 0 を返す

サンプル Visual C サンプルプログラム解説参照

RegistAsyncProcCall**非同期プロシジャーコール割り込みの開始**

書式 `BOOL RegistAsyncProcCall(HWND hWnd, PVOID AsyncCallAdrs, WORD MyIOBase, WORD MyIrqNo, BYTE DirDataRegLo, BYTE DirDataRegHi, BYTE IrqPin, BYTE IrqMode, BYTE IntClrMode, WORD StopCount)`

機能 非同期プロシジャーコール(APC:Asynchronous Procedure Call)版の割り込み処理を開始します。割り込みが発生すると割り込みハンドラが登録されたプロシジャーを呼び出します。

引数

HWND hWnd	
PVOID AsyncCallAdrs	➤ ユーザアプリケーションのウィンドハンドル
WORD MyIOBase	➤ 呼び出すプロシジャーのアドレス
WORD MyIrqNo	➤ カードに割り当てられている I/O ベースアドレス
BYTE DirDataRegLo	➤ カードに割り当てられている IRQ 番号
BYTE DirDataRegHi	➤ データレジスタ下位バイトの入出力方向 0:入力方向 1:出力方向
BYTE IrqPin	➤ データレジスタ上位バイトの入出力方向 0:入力方向 1:出力方向
BYTE IrqMode	➤ 割り込みソース入力ポート番号 0:PI00 1:PI01 2:PI02 … 15:PI015
BYTE IntClrMode	➤ 割り込みトリガーモード 0:立下りエッジ 1:立上りエッジ
WORD StopCount	➤ 割り込みクリア条件 0:下位バイトリード 1:上位バイトリード 2:下位バイトライト 3:上位バイトライト
	➤ 割り込み終了回数

戻値

- 0 : 正常終了
- 1 : 割り込み登録エラー
- 2 : DeviceIoControl ステータスエラー

サンプル Visual C サンプルプログラム解説参照

ReleaseAsyncProcCall**非同期プロシジャーコール割り込みの終了**

書式 `BOOL ReleaseAsyncProcCall(void)`

機能 非同期プロシジャーコール版の割り込み処理を終了します。
RegistAsyncProcCall()による割り込み処理終了時は、必ず
ReleaseAsyncProcCall()を呼び出してください。

引数 なし

戻値 常に 0 を返す

サンプル Visual C サンプルプログラム解説参照

StartHWIntMyVxD

高速割り込み処理の開始

書式 BOOL StartHWIntMyVxD(HWND hWnd, WORD MyIOBase, WORD MyIrqNo, BYTE InputDataReg, BYTE OutputDataReg, BYTE IrqPin, BYTE IrqMode, BYTE IntClrMode, WORD StopCount)

機能 割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。割り込みハンドラ内部では、入力方向に設定されたデータレジスタの値をリードし、出力方向に設定されたデータレジスタにリードした値を出力します。指定回数の割り込み処理が終了すると、呼び出し側プログラムに終了メッセージをポストします。IParam には割り込み発生の累計カウンタ数がセットされています。

引数

HWND hWnd	➤ ユーザアプリケーションのウィンドウハンドル
WORD MyIOBase	➤ カードに割り当てられている I/O ベースアドレス
WORD MyIrqNo	➤ カードに割り当てられている IRQ 番号
BYTE InputDataReg	➤ 入力データレジスタ 0:下位バイト(PI07-PI00) 1:上位バイト(PI015-PI08)
BYTE OutputDataReg	➤ 出力データレジスタ 0:下位(PI07-PI00) 1:上位(PI015-PI08)
BYTE IrqPin	➤ 割り込みソース入力ポート番号 0:PI00 1:PI01 2:PI02 ... 15:PI015
BYTE IrqMode	➤ 割り込みトリガーモード 0:立下りエッジ 1:立上りエッジ
BYTE IntClrMode	➤ 割り込みクリア条件 0:下位バイトリード 1:上位バイトリード 2:下位バイトライト 3:上位バイトライト
WORD StopCount	➤ 割り込み終了回数

戻値

- 0 : 正常終了
- 1 : 割り込み登録エラー
- 2 : DeviceIoControl ステータスエラー

EndHWIntMyVxD

高速割り込み処理の終了

書式 BOOL EndHWIntMyVxD(void)

機能 高速版の割り込み処理を終了します。StartHWIntMyVxD ()による割り込み処理終了時は、必ず EndHWIntMyVxD ()を呼び出してください。

引数 なし

戻値 常に 0 を返す

OutPort **1バイトをポート出力**

- 書式 WORD **OutPort**(WORD **IOAddr**, WORD **OutVal**)
- 機能 1バイトをポートに出力
- 引数 WORD **IOAddr** ➤ ポート番号
 WORD **OutVal** ➤ バイト出力値 (上位バイトは無視)
- 戻値 バイト出力値をそのまま返し、エラー値はなし

wOutPort **1ワードをポート出力**

- 書式 WORD **wOutPort**(WORD **IOAddr**, WORD **OutVal**)
- 機能 1ワードをポートに出力
- 引数 WORD **IOAddr** ➤ ポート番号
 WORD **OutVal** ➤ ワード出力値
- 戻値 ワード出力値をそのまま返し、エラー値はなし

InPort **1バイトをポート入力**

- 書式 WORD **InPort**(WORD **IOAddr**)
- 機能 ポートから1バイト読み込む
- 引数 WORD **IOAddr** ➤ ポート番号
- 戻値 ポートから読み込んだバイトデータを返します (上位バイトは無視してください)

wInPort **1ワードをポート入力**

- 書式 WORD **wInPort**(WORD **IOAddr**)
- 機能 ポートから1ワード読み込む
- 引数 WORD **IOAddr** ➤ ポート番号
- 戻値 ポートから読み込んだワードデータを返します

GetDllVersion**DLL バージョンダイアログの表示**

書式 void GetDllVersion(HWND hWnd)

機能 DIOライブラリのバージョン表示ダイアログを呼び出す

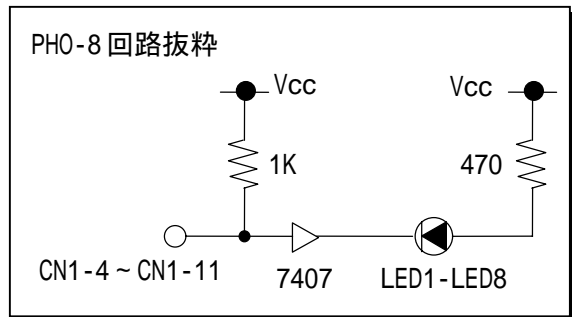
引数 HWND hWnd > ウィンドウのハンドル

戻値 なし

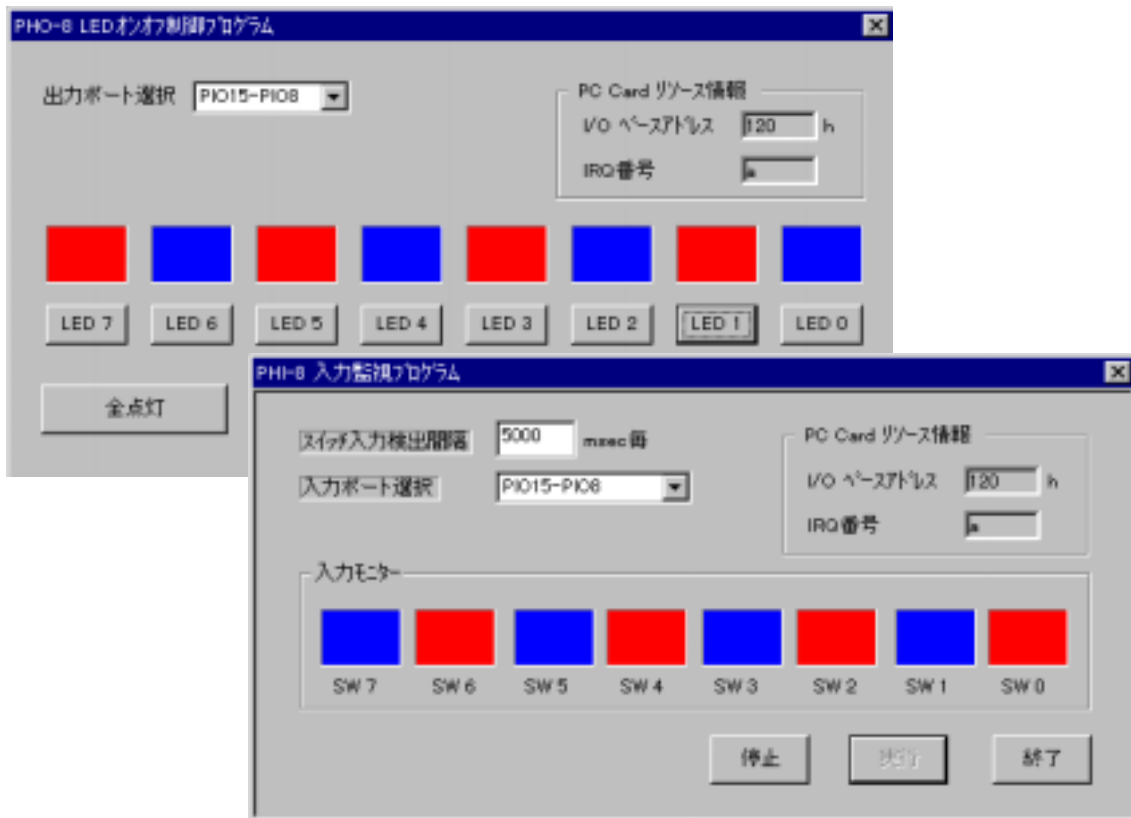
(2-4-2) Visual C サンプルプログラム

アイソレーションユニット PH0-8 または PHI-8 を使った、サンプルプログラムが添付しています。

PH0-8 は右図に示す LED 点灯回路とアイソレーションを行って外部へ出力する回路を 8 ブロック実装しており、CN1-4 ~ CN1-11 に本カードの PIO0 から PIO15 を接続することによりオンオフ制御のシミュレーションを行うことができます。PHI-8 は PH0-8 とは反対に外部オンオフ信号をアイソレーションして入力するためのボードです。



サンプルプログラムは、6 つのモジュールで構成されています。PH0-8 LED オンオフ制御プログラムと PHI-8 入力監視プログラムは、割り込みを使わないで単純に入出力を行うサンプルプログラムです。割り込みを使ったサンプルプログラムとして、ユーザ定義メッセージによる割り込み制御と、非同期プロシジャコールを使った割り込み制御、及び割り込みハンドラ内部で全ての入出力制御を行う高速割り込み制御プログラムがあります。その他にカードリソース表示ユーティリティがあります。、は下図に示す画面になります。次ページ以降では、、のサンプルプログラムについて解説を行います。



ユーザ定義メッセージによる割り込みプログラム

割り込みソース入力ポートを指定して StartHWIntPostMessage() を実行することにより、割り込み発生に同期したユーザ定義メッセージ WM_VXDEVENT が割り込みハンドラから送られてきます。このとき、追加情報 wParam の上位バイトには PI07-PI00 の読み込み値が、下位バイトには PI015-PI08 の読み込み値がセットされています。また、lParam には割り込み発生の累計カウント数がセットされています。

```
#include "diolib32.h"

/*
 * 機能 ユーザ定義メッセージによる通知を使った割り込み入出力制御
 */
BOOL CALLBACK DlgProcPostMessage( HWND hDlg,          // ウィンドウハンドル
                                  UINT message,       // メッセージのタイプ
                                  WPARAM wParam,      // 追加情報
                                  LPARAM lParam       // 追加情報
                                  )
{
    switch( message )
    {
        /* 割り込みハンドラからユーザ定義メッセージ受信 */
        case WM_VXDEVENT:
            /*
             * wParam -> 上位バイト:PI07-PI00 下位バイト:PI015-PI08
             */
            // PI07-PI00 の値 = (BYTE)wParam;
            // PI015-PI08 の値 = (BYTE)(wParam>>8);
            /*
             * lParam -> 割り込みカウント数
             */
            // 割り込みカウント数 = (WORD)lParam;
            /* 割り込み終了回数に達した時の処理 */
            if( 割り込み終了回数に達した時 )
                EndHWIntPostMessage();
            return TRUE;
        case WM_COMMAND:
            switch( wParam ){
                case IDOK:
                    StartHWIntPostMessage( hDlg, MyIOBase, MyIrqNo, DataRegLoDir, DataRegHiDir,
                                             IntPIONo, TrgMode, IntClrMode, MaxCount );
                    return TRUE;
                case IDCANCEL:
                    EndHWIntPostMessage();
                    EndDialog( hDlg, TRUE );          // ダイアログボックスの終了
                    return TRUE;
            }
            break ;
    }
    return FALSE ;
}
```

非同期プロシジャーコールを使った割り込み制御

割り込みハンドラから呼び出すプロシジャーのアドレスと割り込みソース入力ポートを指定して非同期プロシジャーコール(APC:Asynchronous Procedure Call) 版の割り込み処理 `RegistAsyncProcCall()` を実行します。この後、`SleepEx()` を使って呼び出し側のプログラムを呼び出し可能な待ち状態に設定します。割り込みが発生し割り込みハンドラがプロシジャーを呼び出しを行うと、呼び出し側のプログラムは呼び出し可能な待ち状態から復帰します。割り込みハンドラから呼び出されるプロシジャーは、ASYNCPROC 型構造体へのポインターが渡されます。構造体のメンバー `CallbackProc` と `StopCount` は `RegistAsyncProcCall()` で設定した値ですので、無視します。 `ReadVal` の上位バイトが `PI07` から `PI00` を読み出し値で、下位バイトが `PI015` から `PI08` を読み出した値になります。 `IntCount` は、現在までの割り込み発生累積カウント値になります。

```
#include "diolib32.h"

BYTE    ExitSleepFlag = 0; // 非同期プロシジャーコール SleepEx() 脱出用フラグ

BOOL CALLBACK DlgProcAsyncProcCall( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch( message )
    {
        case WM_COMMAND:
            switch( wParam )
            {
                case IDOK:
                    /* コールバックルーチン登録等のパラメータ設定と実行 */
                    RegistAsyncProcCall(hDlg, ApcCallback, MyIOBase, MyIrqNo,
                    DataRegLoDir, DataRegHiDir, IntPI0No, TrgMode, IntClrMode, MaxCount);
                    ExitSleepFlag = 0;
                    while( ExitSleepFlag == 0 )
                        SleepEx( (DWORD)(-1), TRUE );           // 呼び出し可能な状態にする
                    ReleaseAsyncProcCall();
                    return TRUE;
                case IDCANCEL:
                    EndDialog( hDlg, TRUE );
                    return TRUE;
            }
            break ;
    }
    return FALSE;
}

/* 次ページに続く */
```

```

/*
* 書式 DWORD WINAPI ApcCallback( PVOID pApcPrm )
* 機能 非同期コールバックプロシージャ
* 引数 PVOID pApcPrm : ASYNCPROC 型構造体へのポインタ
*      typedef struct AsyncProcCall_t
*      {
*          DWORD CallbackProc;
*          WORD StopCount;
*          WORD ReadVal; 上位バイト (PI07-0)下位バイト(PI015-8) リード値
*          WORD IntCount; 割り込み回数積算値
*      } ASYNCPROC, *pASYNCPROC;
* 戻値 常に 0 を返す
*/
DWORD WINAPI ApcCallback( PVOID pApcPrm )
{
    WORD          ApcCount;  // VxD から渡される現在までの割り込み回数
    WORD          ApcRead;   // VxD から渡されるポート 0-15 のリードデータ

    ApcCount = ((pASYNCPROC)pApcPrm)->IntCount;
    ApcRead = ((pASYNCPROC)pApcPrm)->ReadVal;
    /*
    *   呼び出された時の処理を記述します
    */
    ExitSleepFlag = 1;      // この例では 1 回の呼び出しで終了します
    return 0;
}

```

実行画面



高速割り込み制御プログラム

割り込みハンドラ内で全ての入出力制御を行うことにより高速割り込み処理を行うことができます。但し、割り込みハンドラ内で全ての入出力制御を行う場合は、ユーザ毎の仕様に基づいて仮想デバイスドライバ内の割り込みハンドラをカスタマイズする必要があります。弊社では、有償によるカスタマイズ作業のご相談も承っております。

サンプルプログラムの割り込みハンドラは、入力方向に設定されたデータレジスタの値をリードし、出力方向に設定されたデータレジスタにリードした値を出力します。指定回数の割り込み処理が終了すると、呼び出し側プログラムにユーザ定義メッセージをポストし、追加情報 IParam には割り込み発生累計カウンタ数がセットされています。

入出力データレジスタと割り込み終了回数を指定して、StartHWIntMyVxD() を実行することにより、割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。

実行画面

高速 VxD 内部割り込み処理バージョン	
割り込み終了回数	10 回で終了
データ入力ポート選択	PIO7-PIO0
データ出力ポート選択	PIO7-PIO0
割り込みソースビット	PIO0
割り込みトリガモード	立下りエッジ
割り込みクリア条件	下位バイトリード
PC Card リソース情報	
I/O ベースアドレス	120 h
IRQ番号	9
実行	
終了	

(2-5) BASIC 言語 API ライブラリ解説

(2-5-1) Visual BASIC によるアプリケーション開発

Visual BASIC 4.0 を使って REX-5055 DIO PC カードを制御するアプリケーションを開発する場合は、本製品に添付されている 32Bit 版 DLL の関数をコールする必要があります。また、割り込み制御を行う場合は OLE カスタムコントロール(OCX)を使用します。

Visual BASIC で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、

1. モジュールファイルで DLL 関数の参照宣言を行う。
2. アプリケーションの実行ディレクトリまたは WINDOWS¥SYSTEM に、DIOLIB32.DLL ファイルと VR5055D.VXD ドライバーをコピーする。

必要があります。

また、割り込み制御を行う場合は割り込みハンドラから送られてくるユーザ定義メッセージを Visual BASIC 側のアプリケーションで受け取るために、本製品に添付されている OLE カスタムコントロール MBOX を使用します。MBOX の使用方法については、サンプルプログラム解説を参照してください。

アプリケーション作成上のアドバイス

PC カードに割り当てられてるリソースを取得する

Windows95/98/Me では、PC カードの Plug&Play がサポートされています。これにより、PC カードが使用する I/O ベースアドレス・IRQ 番号は、カードが挿入された時点でシステムがダイナミックに割り当てを行います。従って、常に同じ I/O ベースアドレス・IRQ 番号がカードに割り当てられる訳ではなく、その時点で使用可能なリソースが割り当てられます。

Windows95/98/Me のダイナミックコンフィグレーションに対応するために、DLL では **GetMyCardResource()** をサポートしています。アプリケーションの初期化手続き部分で **GetMyCardResource()** により PC カードに割り当てられている I/O ベースアドレス・IRQ 番号を取得してください。

PC カードへの入出力

Visual C では、I/O 入出力関数がサポートされていません。これは Windows95 の保護機能に基づいたものと考えられます。従って、PC カードへの I/O 入出力は DLL でサポートされている **OutPort()**・**InPort()**・**wOutPort()**・**wInPort()** を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、

- (1) 割り込みハンドラからポストメッセージにより割り込み通知を受け取る方法
- (2) 割り込みハンドラ内で全ての処理を行う方法

があります。高速な処理が要求されるような場合は、(2)の方法で行う必要があります。詳細は、各サンプルプログラムの説明を参照してください。

DLL 関数仕様

GetMyCardResource

リソース情報の取得

- 書式 Declare Function **GetMyCardResource** Lib "diolib32.dll" (ByVal **hWnd** As Long, ByVal **MyCardName** As String, ByVal **NameLen** As Integer, ByVal **SlotNo** As Integer, **IOBase** As Integer, **IrqNo** As Integer) As Long
- 機能 指定スロットに挿入されているカードが指定のカード名と一致するか否か調べます。一致した場合は、割り当てられている I/O ベースアドレス・IRQ リソース情報を返します。
- 引数 **hWnd** ➤ ウィンドウハンドル
 MyCardName ➤ カード識別名を示す文字列へのメモリアドレス
 NameLen ➤ 文字列バッファの長さ
 SlotNo ➤ カード挿入スロット番号
 IOBase ➤ I/O リソース情報を格納する変数アドレス
 IrqNo ➤ IRQ リソース情報を格納する変数アドレス
- 戻値 0 : 正常終了
 -1 : ドライバ呼び出しエラー
 -2 : カードサービスドライババージョンエラー
 -3 : GET_CARD_SERVICES_INFO ファンクションコールサービスエラー
 -4 : GET_FIRST_TUPLE ファンクションコールサービスエラー
 -5 : GET_TUPLE_DATA ファンクションコールサービスエラー
 -6 : GET_CONFIG_INFO ファンクションコールサービスエラー
 -7 : メモリアロケーションエラー
 -8 : スロットにカードが挿入されていない
 -9 : スロットに挿入されているカードは自分のカードと一致しない

ShowCardUtil

リソース表示ユーティリティ起動

- 書式 Declare Function **ShowCardUtil** Lib "diolib32.dll" (ByVal **hWnd** As Long) As Long
- 機能 カードリソース情報表示ユーティリティを起動します。
- 引数 **hWnd** ➤ ウィンドウハンドル
- 戻値 なし

StartHWIntPostMessage**ユーザ定義メッセージ割り込みの開始**

書式 Declare Function **StartHWIntPostMessage** Lib "diolib32.dll" (ByVal **hWnd** As Long, ByVal **MyIOBase** As Integer, ByVal **MyIrqNo** As Integer, ByVal **DirDataRegLo** As Integer, ByVal **DirDataRegHi** As Integer, ByVal **IrqPin** As Integer, ByVal **IrqMode** As Integer, ByVal **IntClrMode** As Integer, ByVal **StopCount** As Integer) As Long

機能 ユーザ定義メッセージ版の割り込み処理を開始します。割り込みが発生すると割り込みハンドラからアプリケーションにメッセージを通知します。wParam の上位バイトに PI07-PI00 の読み込み値が、下位バイトに PI015-PI08 の読み込み値がセットされています。また、lParam には割り込み発生時の累計カウンタ数がセットされています。

引数

hWnd	➤ ユーザアプリケーションのウィンドウハンドル
MyIOBase	➤ カードに割り当てられている I/O ベースアドレス
MyIrqNo	➤ カードに割り当てられている IRQ 番号
DirDataRegLo	➤ データレジスタ下位バイトの入出力方向 0:入力方向 1:出力方向
DirDataRegHi	➤ データレジスタ上位バイトの入出力方向 0:入力方向 1:出力方向
IrqPin	➤ 割り込みソース入力ポート番号 0:PI00 1:PI01 2:PI02 … 15:PI015
IrqMode	➤ 割り込みトリガモード 0:立下りエッジ 1:立上りエッジ
IntClrMode	➤ 割り込みクリア条件 0:下位バイトリード 1:上位バイトリード 2:下位バイトライト 3:上位バイトライト
StopCount	➤ 割り込み終了回数

戻値 0 : 正常終了
-1 : 割り込み登録エラー
-2 : DeviceIoControl ステータスエラー

EndHWIntPostMessage**ユーザ定義メッセージ割り込みの終了**

書式 Declare Function **EndHWIntPostMessage** Lib "diolib32.dll" () As Long

機能 ユーザ定義メッセージ版の割り込み処理を終了します。ユーザ定義メッセージ版による割り込み処理終了時は、必ず EndHWIntPostMessage() を呼び出してください。

引数 なし

戻値 常に 0 を返す

StartHWIntMyVxD

高速割り込み処理の開始

書式 Declare Function **StartHWIntMyVxD** Lib "diolib32.dll" (ByVal **hWnd** As Long, ByVal **MyIOBase** As Integer, ByVal **MyIrqNo** As Integer, ByVal **InputDataReg** As Integer, ByVal **OutputDataReg** As Integer, ByVal **IrqPin** As Integer, ByVal **IrqMode** As Integer, ByVal **IntClrMode** As Integer, ByVal **StopCount** As Integer) As Long

機能 割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。割り込みハンドラ内部では、入力方向に設定されたデータレジスタの値をリードし、出力方向に設定されたデータレジスタにリードした値を出力します。指定回数の割り込み処理が終了すると、呼び出し側プログラムに終了メッセージをポストします。IParam には割り込み発生のカウント数がセットされています。

引数

hWnd	➤ ユーザアプリケーションのウィンドウハンドル
MyIOBase	➤ カードに割り当てられている I/O ベースアドレス
MyIrqNo	➤ カードに割り当てられている IRQ 番号
InputDataReg	➤ 入力データレジスタ 0: 下位バイト (PI07-PI00) 1: 上位バイト (PI015-PI08)
OutputDataReg	➤ 出力データレジスタ 0: 下位 (PI07-PI00) 1: 上位 (PI015-PI08)
IrqPin	➤ 割り込みソース入力ポート番号 0: PI00 1: PI01 2: PI02 ... 15: PI015
IrqMode	➤ 割り込みトリガーモード 0: 立下りエッジ 1: 立上りエッジ
IntClrMode	➤ 割り込みクリア条件 0: 下位バイトリード 1: 上位バイトリード 2: 下位バイトライト 3: 上位バイトライト
StopCount	➤ 割り込み終了回数

戻値

- 0 : 正常終了
- 1 : VPICD 割り込み登録エラー
- 2 : DIO コントロールステータスエラー
- 3 : パラメータ設定エラー
- 4 : ポストメッセージ用メモリアロケーションエラー
- 5 : 割り込みスタートエラー

EndHWIntMyVxD

高速割り込み処理の終了

書式 Declare Function **EndHWIntMyVxD** Lib "diolib32.dll" () As Long

機能 高速版の割り込み処理を終了します。StartHWIntMyVxD () による割り込み処理終了時は、必ず EndHWIntMyVxD () を呼び出してください。

引数 なし

戻値 常に 0 を返す

OutPort**1バイトをポート出力**

書式 `Declare Function OutPort Lib "diolib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer`

機能 1バイトをポートに出力

引数 **IOAddr** ➤ ポート番号
Val ➤ バイト出力値（上位バイトは無視）

戻値 バイト出力値をそのまま返し、エラー値はなし

wOutPort**1ワードをポート出力**

書式 `Declare Function wOutPort Lib "diolib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer`

機能 1ワードをポートに出力

引数 **IOAddr** ➤ ポート番号
Val ➤ ワード出力値

戻値 ワード出力値をそのまま返し、エラー値はなし

InPort**1バイトをポート入力**

書式 `Declare Function InPort Lib "diolib32.dll" (ByVal IOAddr As Integer) As Integer`

機能 ポートから1バイト読み込む

引数 **IOAddr** ➤ ポート番号

戻値 ポートから読み込んだバイトデータを返します(上位バイトは無視してください)

wInPort**1ワードをポート入力**

書式 `Declare Function wInPort Lib "diolib32.dll" (ByVal IOAddr As Integer) As Integer`

機能 ポートから1ワード読み込む

引数 **IOAddr** ➤ ポート番号

戻値 ポートから読み込んだワードデータを返します

GetDllVersion**DLL バージョンダイアログの表示**

書式 Declare Sub **GetDllVersion** Lib "diolib32.dll" (ByVal **hWnd** As Long)

機能 DI0ライブラリのバージョン表示ダイアログを呼び出す

引数 **hWnd** ➤ ウィンドウのハンドル

戻値 なし

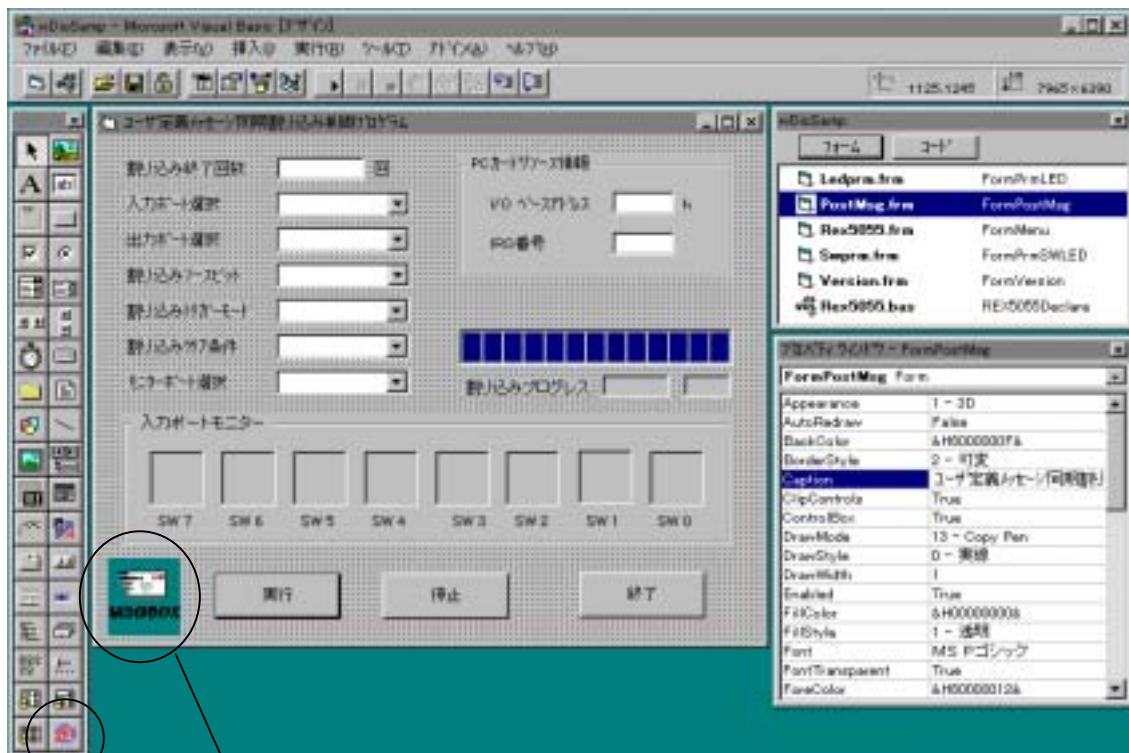
(2-5-2) Visual BASIC サンプルプログラム

割り込みを使わないで単純に入出力を行うプログラム例	PH0-8 LED オンオフ制御プログラム PHI-8 入力プログラム
割り込みを使ったプログラム例	ユーザ定義メッセージによる割り込み制御 割り込みハンドラ内部で全ての入出力制御を行う高速割り込み制御

動作環境等については、Visual C のサンプルプログラム解説を参照してください。、実行画面は下記のようになります。ここでは、ユーザ定義メッセージによる割り込み制御について解説し、、については省略致します。

ユーザ定義メッセージによる割り込み制御サンプルプログラムの解説

下記画面は、VB4.0 でのデザイン完成時の画面です。割り込み発生に同期したユーザ定義メッセージを VB で作成したプログラムで受け取るためには、本製品添付の OLE カスタムコントロール(OCX)MBOX を使用します。以下、作成の手順について説明します。



本製品に添付されている OLE カスタムコントロール”MBOX”

カスタムコントロール「MBOX OLE Control module」を追加

Step.1 本製品添付ソフトのコピー

OLE カスタムコントロール : MBOX.OCX と 32 ビット版 DLL : DIOLIB32.DLL 及び 仮想デバイスドライバー : VR5055D.VXD を添付のフロッピーディスクからコピーします。VB プログラム作成ディレクトリ名を“MyProg”とします。

```
>COPY “コピー元ドライブ名”:%Win95%DI132%Diolib32.dll “コピー先ドライブ名”:%Windows%System
>COPY “コピー元ドライブ名”:%Win95%DI132%Vr5055d.vxd “コピー先ドライブ名”:%Windows%System
>COPY “コピー元ドライブ名”:%Win95%DI132%Mbox.ocx “コピー先ドライブ名”:%MyProg
```

Step.2 OCX のレジストリー登録 (割り込みサービス使用時必須)

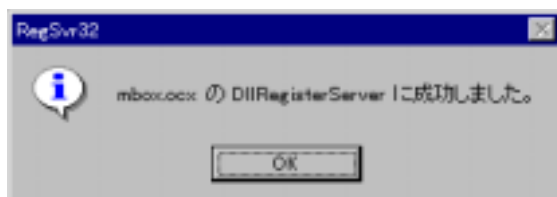
本製品添付の OCX “MBOX.OCX”を VB で使用するためには、VB の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリー登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、Windows の DOS BOX から実行します。尚、“REGSVR32.EXE”は VB の CD-ROM の“TOOLS¥PSS”に添付されています。

OCX をレジストリー登録するときは、下記構文で実行します。

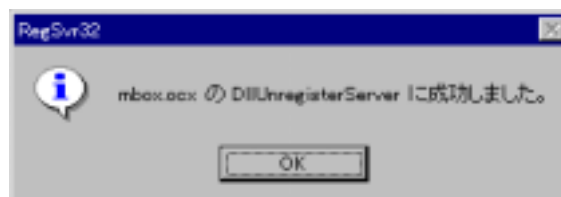
```
>REGSVR32 “ドライブ名”:%MyProg%Mbox.ocx
```

OCX をレジストリー登録から削除するときは、“/U”を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:%MyProg%Mbox.ocx
```



登録成功メッセージ



登録削除成功メッセージ

Step.3 DIOLIB32 関数の Declare 宣言

次に、VB プログラムの作成に入ります。VB デザインメニューから新規プロジェクトを作成し、「ファイル」の「挿入」から標準モジュールを追加します。追加した標準モジュールファイルで DLL 関数の参照宣言を行います。宣言部分は、サンプルプログラム“REX5055.BAS”の下記部分をコピーしてください。

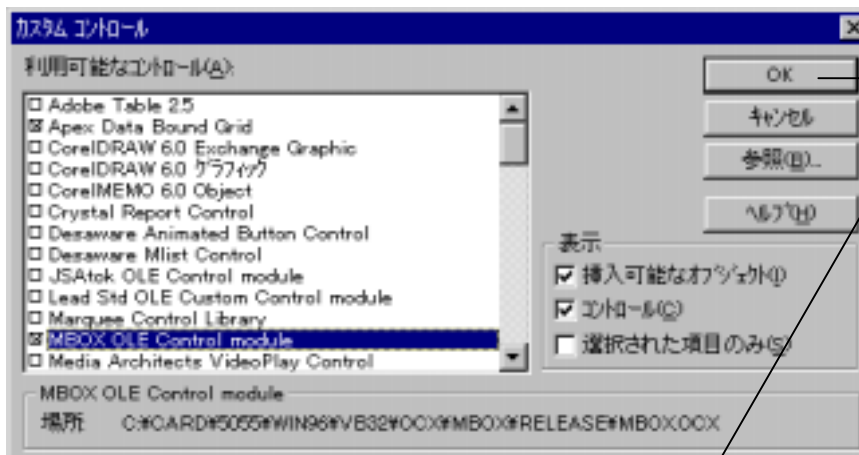
```
Declare Function OutPort Lib "diolib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer
Declare Function wOutPort Lib "diolib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer
Declare Function InPort Lib "diolib32.dll" (ByVal IOAddr As Integer) As Integer
Declare Function wInPort Lib "diolib32.dll" (ByVal IOAddr As Integer) As Integer
Declare Function StartHWIntPostMessage Lib "diolib32.dll" (ByVal hWnd As Long, ByVal MyIOBase As Integer,
ByVal MyIrqNo As Integer, ByVal DirDataRegLo As Integer, ByVal DirDataRegHi As Integer, ByVal IrqPin As
Integer, ByVal IrqMode As Integer, ByVal IntClrMode As Integer, ByVal StopCount As Integer) As Long
Declare Function EndHWIntPostMessage Lib "diolib32.dll" () As Long
Declare Function StartHWIntMyVxD Lib "diolib32.dll" (ByVal hWnd As Long, ByVal MyIOBase As Integer, ByVal
MyIrqNo As Integer, ByVal InputDataReg As Integer, ByVal OutputDataReg As Integer, ByVal IrqPin As Integer,
ByVal IrqMode As Integer, ByVal IntClrMode As Integer, ByVal StopCount As Integer) As Long
Declare Function EndHWIntMyVxD Lib "diolib32.dll" () As Long
Declare Function GetMyCardResource Lib "diolib32.dll" (ByVal hWnd As Long, ByVal MyCardName As String, ByVal
NameLen As Integer, ByVal SlotNo As Integer, IOBase As Integer, IrqNo As Integer) As Long
Declare Function ShowCardUtil Lib "diolib32.dll" (ByVal hWnd As Long) As Long
Declare Sub GetDIIVersion Lib "diolib32.dll" (ByVal hWnd As Long)
```

Step.4 MBOX OLE Control Module の追加 (割り込みサービス使用時必須)

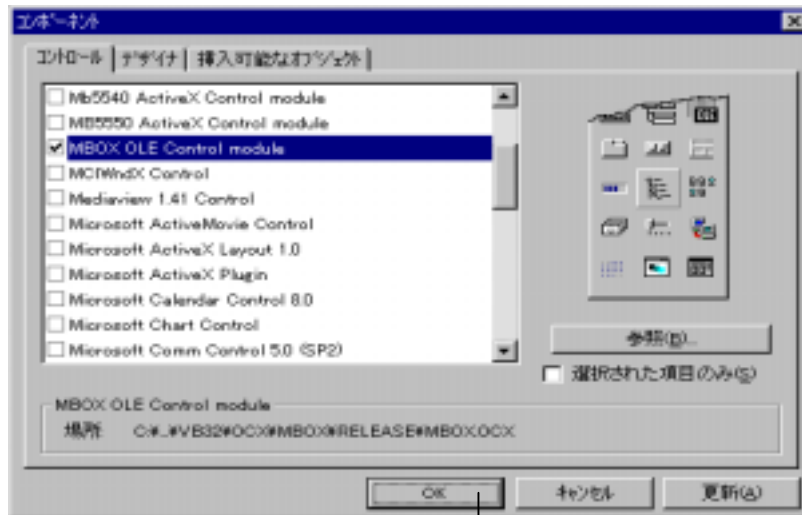
VB のカスタムコントロールに MBOX(OCX)を追加します。

VB4.0 の場合、VB デザインメニューの「ツール」の「カスタムコントロール」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。

➤ VB4.0 の場合



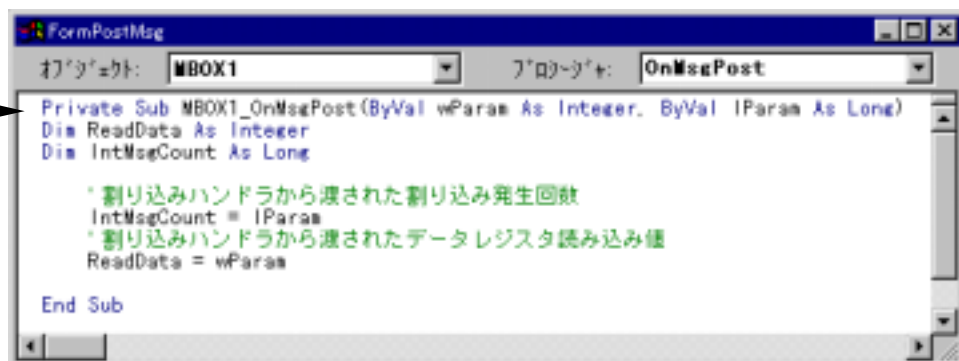
VB5.0/6.0の場合、VB デザインメニューの「プロジェクト」から「コンポーネント」を開き、「コントロール」タブを選びます。利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。



Step.5 フォームに MBOX(OCX)を貼り付ける (割り込みサービス使用時必須)

フォームを作成し、割り込みハンドラが割り込み起動元プログラムに送るユーザ定義メッセージを受け取るための MBOX(OCX)を貼り付けます。これにより、割り込みが発生すると MBOX がサービスするプロシージャ

MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)が呼び出されます。この中で、割り込み通知に同期した処理を記述します。



Step.6 カードに割り当てられているリソース情報取得

DIO カードを制御するためには、プログラム実行時に自分のカードに割り当てられている I/O ベースアドレスと割り込み番号のリソース情報を取得する必要があります。通常は、割り込みを実行するフォームがロードされた時にリソースを取得します。

```
Global Const MyName = "REX5055 DIO PC Card" ' 製品情報フォルダの製品名の定義
Global MyIOBase As Integer ' カード I/O ベースアドレス
Global MyIrqNo As Integer ' カード IRQ 番号

Private Sub Form_Load()
Dim SlotNo, Stat As Integer
' スロット 0 または 1 に挿入されている自分のカードのリソース情報取得
For SlotNo = 0 To 1
Stat = GetMyCardResource(hWnd, MyName, Len(MyName), SlotNo, MyIOBase, MyIrqNo)
If Stat = 0 Then
TextAdrs.Text = Hex(MyIOBase)
TextInt.Text = Str(MyIrqNo)
Exit For
End If
Next SlotNo
End Sub
```

Step.7 ユーザ定義メッセージによる割り込み処理の起動

フォームの OK ボタンが押されたらユーザ定義メッセージによる割り込み制御 StartHWIntPostMessage() を実行します。StartHWIntPostMessage() へは MBOX1.GetMboxWnd() により取得した OCX のウィンドウハンドルを引き渡してください。尚、StartHWIntPostMessage() を実行した際は、終了時には必ず EndHWIntPostMessage() を実行し割り込み登録の解除を行ってください。また、VB 上の全てのバグをフィックスした後に記述してください。

```
Private Sub CommandOK_Click()
Global MyIOBase, MyIrqNo As Integer
Dim OleHandle As Long
Dim RegVal, dummy As Integer
Dim DataRegLoDir, DataRegHiDir, IntSrcBit, IntTrgMode, IntClrMode As Integer
' OLE のウィンドウハンドル取得
OleHandle = MBOX1.GetMboxWnd()
If (OleHandle = 0) Then
MsgBox "OLE のハンドルが取得できません。", vbOKOnly + vbCritical, "エラー"
Exit Sub
End If
' ユーザ定義メッセージを受け取る割り込み処理の起動
dummy = StartHWIntPostMessage(OleHandle, MyIOBase, MyIrqNo, DataRegLoDir, DataRegHiDir,
IntSrcBit, IntTrgMode, IntClrMode, IsrStopCount)
End Sub
```

(空白ページ)

第3章 Windows2000/XP解説

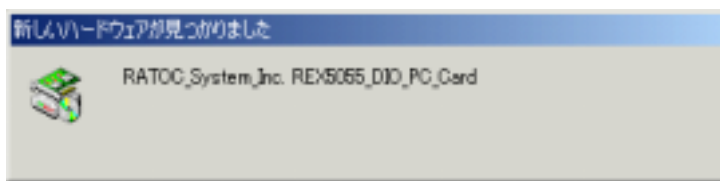
本章では Windows2000 および WindowsXP での REX-5055 の使用方法について解説しております。

(3-1) インストール

田 Windows2000 でのインストール方法

【1】PC カードの挿入

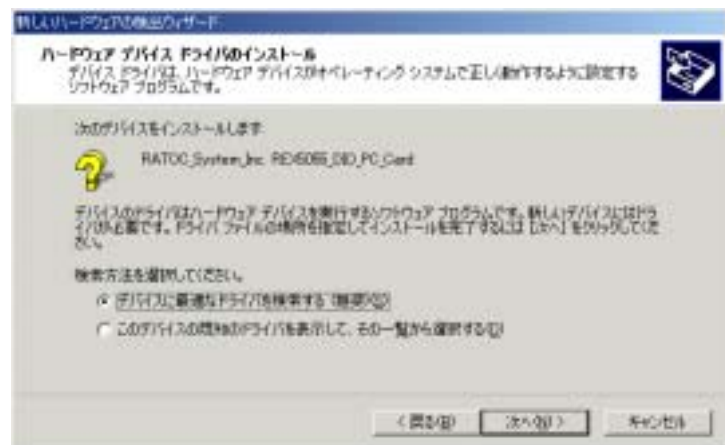
PC カードを挿入すると「ハードウェアウィザード」が起動し(右下画面)、インストールが開始されます。「RATOC_System_Inc. REX5055_DIO_PC_Card」と表示されているかを確認し、以下の手順でインストールを行ってください。



「新しいハードウェアの検索ウィザードの開始」で「次へ(N)>」ボタンを押します。



「ハードウェアデバイスドライバのインストール」では「デバイスに最適なドライバを検索する(推奨)(S)」にチェックを入れて「次へ(N)>」ボタンを押します。

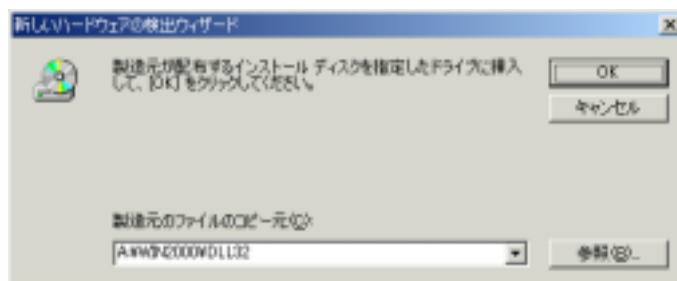


「ドライバファイルの特定」で「場所を指定(S)」にチェックを入れて「次へ(N)>」ボタンを押します。



[2] inf ファイル場所の指定

製品添付の FD をフロッピーディスクドライブに挿入します。製造元のファイルのコピー元(C)で inf ファイルの場所を指定し、「OK」ボタンを押します。

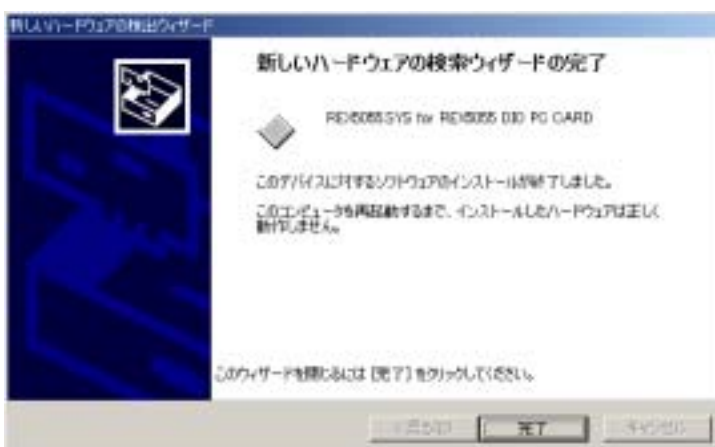


「ドライバファイルの検索」では、製品添付のFDより右画面のように inf ファイルが検索されますので「次へ(N)>」ボタンを押します。



[3] インストールの完了

「新しいハードウェアの検出ウィザードの完了」で「REX-5055.SYS for REX-5055 DIO PC CARD」が表示されます。「完了」ボタンを押して、パソコンの再起動を行ってください。



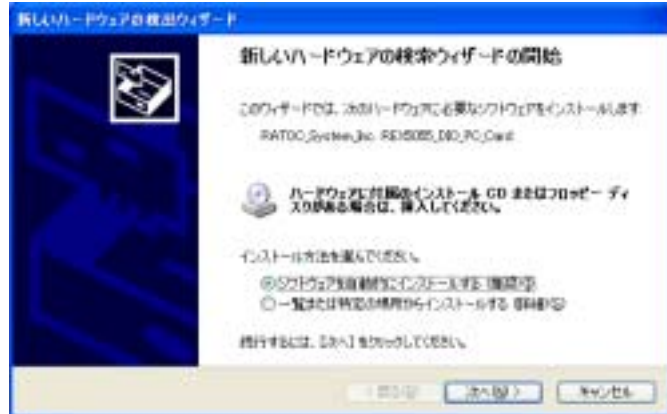
以上で、REX-5055 のインストールは終了です。

WindowsXP でのインストール方法

【1】 PC カードの挿入

PC カードをスロットに挿入すると、右図の「新しいハードウェアの検出ウィザード」が表示されますので、製品添付の Win2000/XP 用フロッピーディスクを FD ドライブへ挿入してください。

次に、「ソフトウェアを自動的にインストールする（推奨）(I)」を選択し「次へ」ボタンを押します。



セットアップ情報ファイル (inf ファイル) が、フロッピーディスク上から検索され、自動的にインストールが行われます。

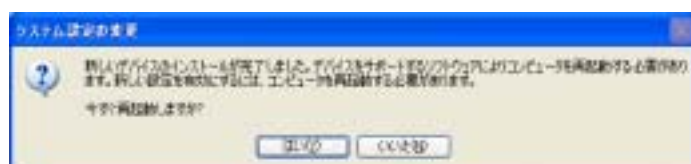


右の画面が表示されましたら、「完了」ボタンを押します。

「完了」ボタンを押した後、右下の画面が出ますので、「はい(Y)」を押してパソコンを再起動してください。



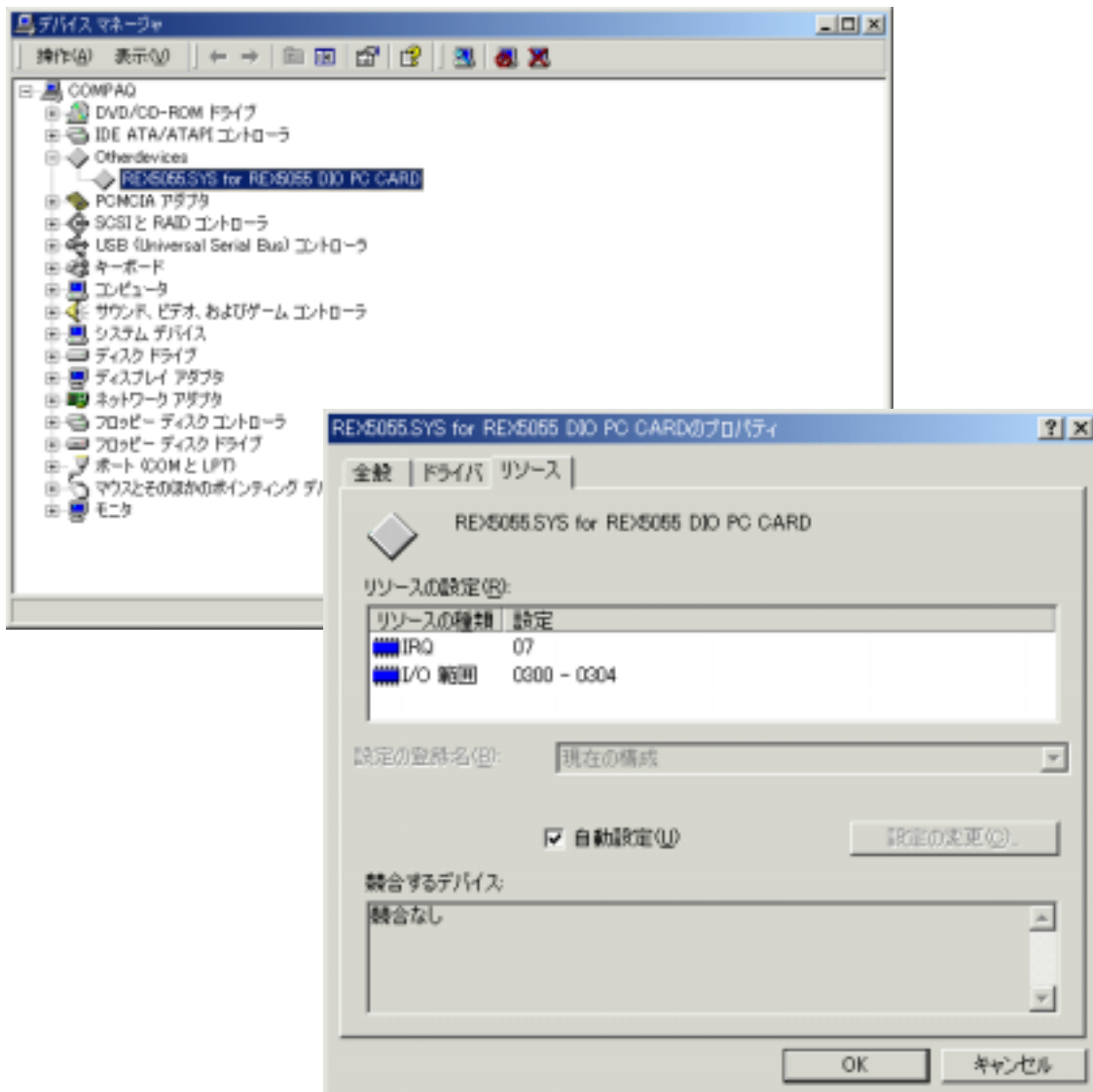
以上で、REX-5055 インストールは終了です。



(3-2) PC カード設定内容の確認

コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D)」ボタンを押します。「OtherDevices」をクリックして新しく REX5055.SYS for REX-5055 DIO PC CARD が追加されているのを確認してください。

また、「プロパティ」でリソースが正しく割当てられているかを確認してください。デバイスの競合が発生した場合は「自動設定(U)」のチェックを外し、競合が起こらない値に設定を変更してください。



(3-3) アンインストール

Windows2000 でのアンインストール方法

インストールした内容を削除する方法について説明します。

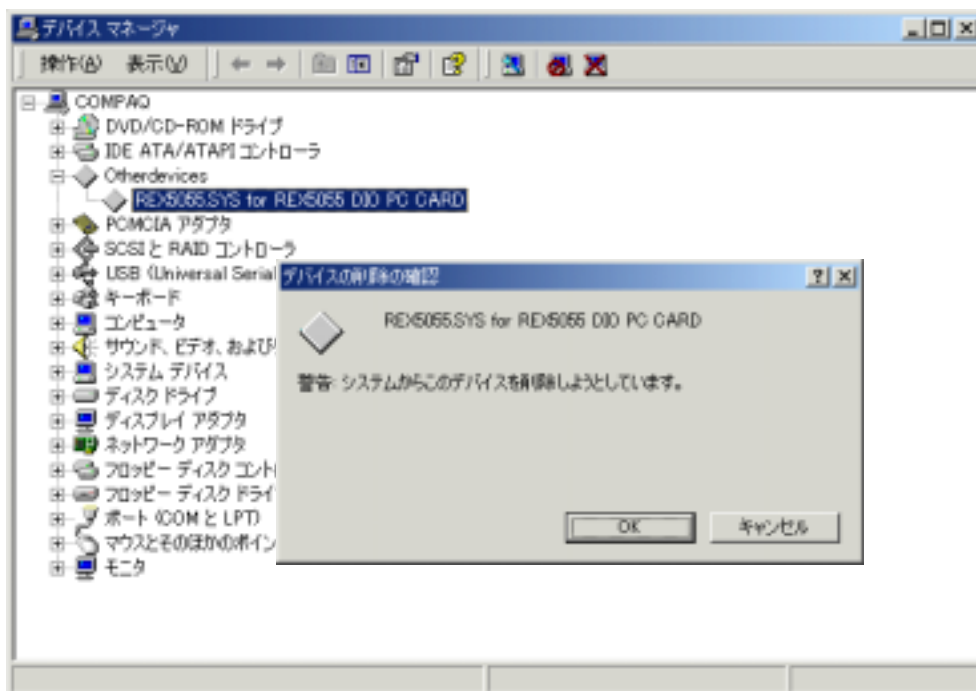
削除は、

- (1)デバイスの削除
- (2)INF ファイルの削除 の手順で行います。

【1】デバイスの削除

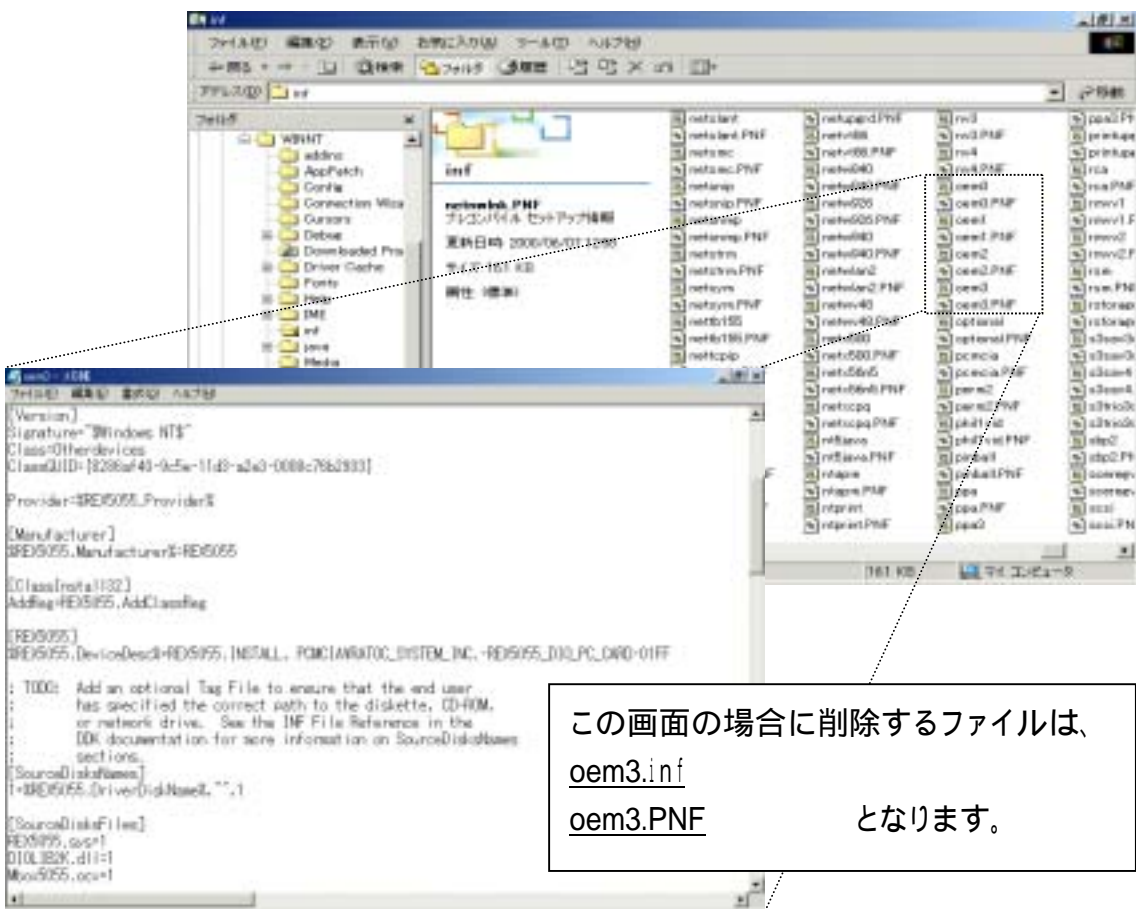
PC カードを挿入した状態で、コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D)」ボタンを押します。「Otherdevices」をクリックして REX-5055.SYS for DIO PC CARD を表示させクリックします。

メニューバーの「操作(A)」 - 「削除(U)」をクリックするか、キーボード上の「Delete」キーを押します。デバイスの削除の確認で「OK」ボタンを押し削除してください。



[2] INF ファイルの削除

エクスプローラからフォルダ「C:\WINNT\inf」を開き、oemX.inf ファイル(X=数字)を検索し、例えば oem0.inf が1つだけの場合は、oem0.infと拡張子のみ異なる oem0.PNF を削除してください。oemX.infが複数ある場合 (oem0.inf , oem1.inf・・・)は、メモ帳などでそれぞれの inf ファイルを開いて、その内容の[Manufacturer]セクションが %REX5055 , Manufacturer%=REX5055 となっているファイルと拡張子のみ異なる PNF ファイルを削除してください。



以上の操作でアンインストール完了です。
カードスロットより、REX-5055 を抜きパソコンを再起動してください。

◆注意...◆

エクスプローラの設定が「全てのファイルを表示」になっていないとフォルダ「C:\WINNT\INF」は表示されません。設定の変更は、エクスプローラメニューの「ツール」から「フォルダオプション」を選択し、表示タグ内の詳細設定で、すべてのファイルとフォルダを表示するに設定してください。

(3-4) C 言語 API ライブラリ解説

(3-4-1) Visual C によるアプリケーション開発

Visual C 5.0 以上を使って REX-5055 DIO PC カードを制御するアプリケーションを開発する場合は、本製品添付フロッピーディスクの¥Win2000¥DII32 フォルダに格納されている 32Bit 版 DLL の関数をコールする必要があります。

Visual C で作成したアプリケーションプログラムから Windows2000/XP 用 32Bit 版 DLL(DIOLIB2K.DLL)を呼び出すためには、

1. アプリケーションプログラムに DIOLIB2K.H ファイルをインクルードする。
2. アプリケーションプログラムのプロジェクトファイルに DIOLIB2K.LIB を追加する。

必要があります。

アプリケーション作成上のアドバイス

PC カードに割り当てられてるリソースを取得する

Windows2000/XP では、Windows95/98/Me と同様に、**GetMyCardResource()** をサポートしています。アプリケーションの初期化手続き部分で **GetMyCardResource()**により PC カードに割り当てられている I/O ベースアドレス・IRQ 番号を取得してください。

PC カードへの入出力

Windows95 と同様、Visual C では、I/O 入出力関数がサポートされていません。従って、PC カードへの I/O 入出力は DLL でサポートされている **OutPort()**・**InPort()**・**wOutPort()**・**wInPort()**を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、
(1)割り込みハンドラからユーザ定義メッセージにより割り込み通知を受け取る方法
(2)割り込みハンドラ内で全ての処理を行う方法
があります。高速な処理が要求されるような場合は、(2)の方法で行う必要があります。詳細は、各サンプルプログラムの説明を参照してください。

DLL 関数仕様

GetMyCardResource

リソース情報の取得

- 書式 `BOOL GetMyCardResource(HWND hwnd, LPSTR MyCardName, WORD NameLen, WORD SlotNo, LPWORD IOBase, LPWORD IrqNo)`
- 機能 指定スロットに挿入されているカードが指定のカード名と一致するか否か調べます。一致した場合は、割り当てられている I/O ベースアドレス・IRQ リソース情報を返します。
- 引数 `HWND hwnd` ➤ ウィンドウハンドル
 `LPSTR MyCardName` ➤ カード識別名を示す文字列へのポインタ
 `WORD NameLen` ➤ 文字列バッファの長さ
 `WORD SlotNo` (Windows95/98 互換用) (0 を指定してください)
 `LPWORD IOBase` ➤ (出力) I/O リソース情報を格納する変数アドレス
 `LPWORD IrqNo` ➤ (出力) IRQ リソース情報を格納する変数アドレス
- 戻値 正常終了時 0 を返し、リソースが正常に取得できなかった場合は -1 を返します。

StartHWIntPostMessage**ユーザ定義メッセージ割り込みの開始**

- 書式** **BOOL StartHWIntPostMessage(HWND hWnd, WORD MyIOBase, WORD MyIrqNo, WORD DirDataRegLo, WORD DirDataRegHi, WORD IrqPin, WORD IrqMode, WORD IntClrMode, WORD StopCount)**
- 機能** ユーザ定義メッセージ版の割り込み処理を開始します。割り込みが発生すると割り込みハンドラからアプリケーションにメッセージを通知します。wParam の上位バイトに PI07-PI00 の読み込み値が、下位バイトに PI015-PI08 の読み込み値がセットされています。また、lParam には割り込み発生 of 累計カウント数がセットされています。
- 引数**
- | | |
|-------------------|---|
| HWND hWnd | ➤ ユーザアプリケーションのウィンドウハンドル |
| WORD MyIOBase | ➤ カードに割り当てられている I/O ベースアドレス |
| WORD MyIrqNo | ➤ カードに割り当てられている IRQ 番号 |
| WORD DirDataRegLo | ➤ データレジスタ下位バイトの入出力方向
0:入力方向 1:出力方向 |
| WORD DirDataRegHi | ➤ データレジスタ上位バイトの入出力方向
0:入力方向 1:出力方向 |
| WORD IrqPin | ➤ 割り込みソース入力ポート番号
0:PI00 1:PI01 2:PI02 …… 15:PI015 |
| WORD IrqMode | ➤ 割り込みトリガーモード
0:立下りエッジ 1:立上りエッジ |
| WORD IntClrMode | ➤ 割り込みクリア条件
0:下位バイトリード 1:上位バイトリード
2:下位バイトライト 3:上位バイトライト |
| WORD StopCount | ➤ 割り込み終了回数 |
- 戻値** 正常終了 0 を返します。I/O ベースアドレス設定エラー等は-1 を返し、その他のマイナスエラーはDirDataRegLo・DirDataRegHi・IrqMode・IntClrMode の設定エラーです。

EndHWIntPostMessage**ユーザ定義メッセージ割り込みの終了**

- 書式** **BOOL EndHWIntPostMessage(void)**
- 機能** ユーザ定義メッセージ版の割り込み処理を終了します。ユーザ定義メッセージによる割り込み処理終了時は、必ず EndHWIntPostMessage() を呼び出して下さい。
- 引数** なし
- 戻値** 常に 0 を返す

StartHWIntMyVxdBuf

高速割り込み処理の開始

- 書式 `BOOL StartHWIntMyVxdBuf(HWND hWnd, WORD MyIOBase, WORD MyIrqNo, WORD InputDataReg, WORD IrqPin, WORD IrqMode, WORD IntClrMode, WORD StopCount, char *buf)`
- 機能 IrqPin で指定した上位ポート（下位ポート）に割り込み信号が発生すると下位ポート（上位ポート）のデータレジスタを読み込み、指定したバッファ `char*buf` に格納します。指定した Count 数だけ割り込みが発生すると本関数は処理を呼び出し側に返します。
- 引数
- | | |
|-------------------|---|
| HWND hWnd | ➤ ユーザアプリケーションのウィンドウハンドル |
| WORD MyIOBase | ➤ カードに割り当てられている I/O ベースアドレス |
| WORD MyIrqNo | ➤ カードに割り当てられている IRQ 番号 |
| WORD InputDataReg | ➤ 入力データレジスタ
0:下位バイト(PI07-PI00) 1:上位バイト(PI015-PI08) |
| WORD IrqPin | ➤ 割り込みソース入力ポート番号
0:PI00 1:PI01 2:PI02 ... 15:PI015 |
| WORD IrqMode | ➤ 割り込みトリガーモード
0:立下りエッジ 1:立上りエッジ |
| WORD IntClrMode | ➤ 割り込みクリア条件
0:下位バイトリード 1:上位バイトリード
2:下位バイトライト 3:上位バイトライト |
| WORD StopCount | ➤ 割り込み終了回数 |
| char *buf | ➤ データを格納するバッファアドレス |
- 戻値 正常終了時 0 を返します。引数設定エラー等は-1、メモリアロケーションエラーの場合は-2 を返します。

EndHWIntMyVxdBuf

高速割り込み処理の終了

- 書式 `BOOL EndHWIntMyVxdBuf(void)`
- 機能 高速版の割り込み処理を終了します。StartHWIntMyVxdBuf()による割り込み処理終了時は、必ず EndHWIntMyVxdBuf()を呼び出してください。
- 引数 なし
- 戻値 常に 0 を返す

OutPort **1バイトをポート出力**

- 書式 WORD **OutPort**(WORD **IOAdrs**, WORD **bOutVal**)
- 機能 1バイトをポートに出力
- 引数 WORD **IOAdrs** ➤ ポート番号
WORD **bOutVal** ➤ バイト出力値(上位バイトは無視)
- 戻値 バイト出力値をそのまま返し、ドライバ呼び出しエラーは-1を返します。

wOutPort **1ワードをポート出力**

- 書式 WORD **wOutPort**(WORD **IOAdrs**, WORD **wOutVal**)
- 機能 1ワードをポートに出力
- 引数 WORD **IOAdrs** ➤ ポート番号
WORD **wOutVal** ➤ ワード出力値
- 戻値 ワード出力値をそのまま返し、ドライバ呼び出しエラーは-1を返します。

InPort **1バイトをポート入力**

- 書式 WORD **InPort**(WORD **IOAdrs**)
- 機能 ポートから1バイト読み込む
- 引数 WORD **IOAdrs** ➤ ポート番号
- 戻値 ポートから読み込んだバイトデータを返します。ドライバ呼び出しエラーは-1を返します。(上位バイトは無視してください)

wInPort **1ワードをポート入力**

- 書式 WORD **wInPort**(WORD **IOAdrs**)
- 機能 ポートから1ワード読み込む
- 引数 WORD **IOAdrs** ➤ ポート番号
- 戻値 ポートから読み込んだワードデータを返します。ドライバ呼び出しエラーは-1を返します。

(3-4-2) Visual C サンプルプログラム

本製品には、以下3つのサンプルプログラムを添付しております。

<p>基本デジタル入出力プログラム (割り込みを使わないで単純に入出力を行うサンプルプログラム)</p> <p>外部イベント監視プログラム (割り込みを使ったサンプルプログラム)</p> <p>データ受信プログラム (高速割り込みを使ったサンプルプログラム)</p>

次頁より、各サンプルプログラムについて解説いたします。
(詳細については各サンプルプログラムのソースをご参照ください。)

Windows95/98/Me で作成したアプリケーションを Windows2000/XP で使用する場合

基本的には Windows2000/XP 用ヘッダファイル **DIOLIB2K.H** とライブラリファイル **DIOLIB2K.LIB** を新規プロジェクトに追加し、Windows95/98/Me で作成したソースファイルにインクルード後、コンパイルすることによって使用可能になります。

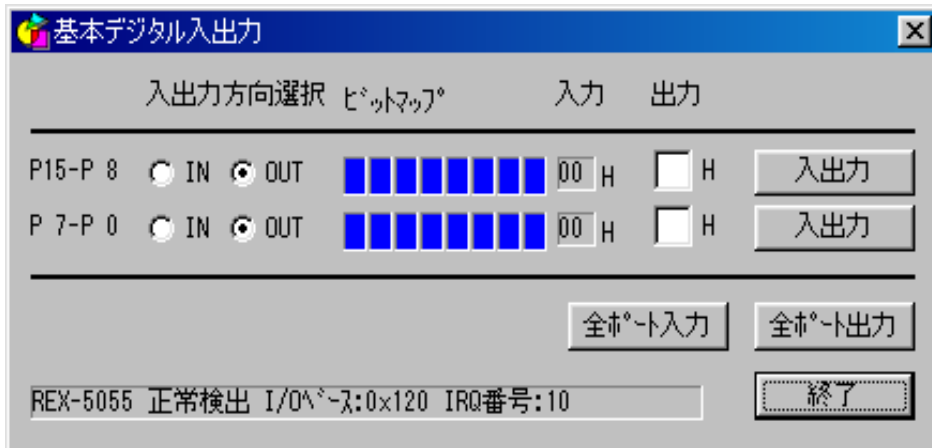
但し、以下の関数を使用されている場合は、Windows2000/XP の **DIOLIB2K.DLL** ではサポートしておりませんのでご注意ください。

ShowCardUtil()
ResistAsyncProcCall()
ReleaseAsyncProcCall()
StartHWIntMyVxD() ¹
EndHWIntMyVxD() ²
GetDllversion()

- 1 . Windows2000/XP では、StartHWIntMyVxdBuf()をご使用ください。
- 2 . Windows2000/XP では、EndHWIntMyVxdBuf()をご使用ください。

基本デジタル入出力プログラム

本サンプルプログラムは 16 ポート(8 ポート単位)を使って外部機器とオンオフ入出力を行う場合のサンプルプログラムです。



DIO カードを制御するためには、プログラム実行時にカードに割り当てられている I/O ベースアドレスと割り込み番号のリソース情報を取得する必要があります。また、Output() , Inport()により方向設定を行ってください。

リソースの取得、方向設定の記述例

```

BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
    WORD        SlotNo;           //Windows95/98 互換用
    WORD        Status;
    /*
     *   シットに挿入されている自分のカードのリソース情報を取得する
     */
    Status = GetMyCardResource( hwnd, MyCardName, sizeof(MyCardName), 0, &MyIOBase, &MyIrqNo );
    if ( Status == 0 )
    {
        /*
         *           リソース情報を表示する
         */
        sprintf( MsgBuf, "REX-5055 正常検出 I/O ^-λ:0x%x IRQ 番号:%d", MyIOBase, MyIrqNo);
        SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );

        // 出力方向設定 (PI00-PI07)
        Status = Output( 0x120, 0x1 );
        // 下位バイト (PI00-PI07) すべて出力
        Status = Output( 0x122, 0xFF );
        Return = TRUE ;
    }
    return TRUE ;
}

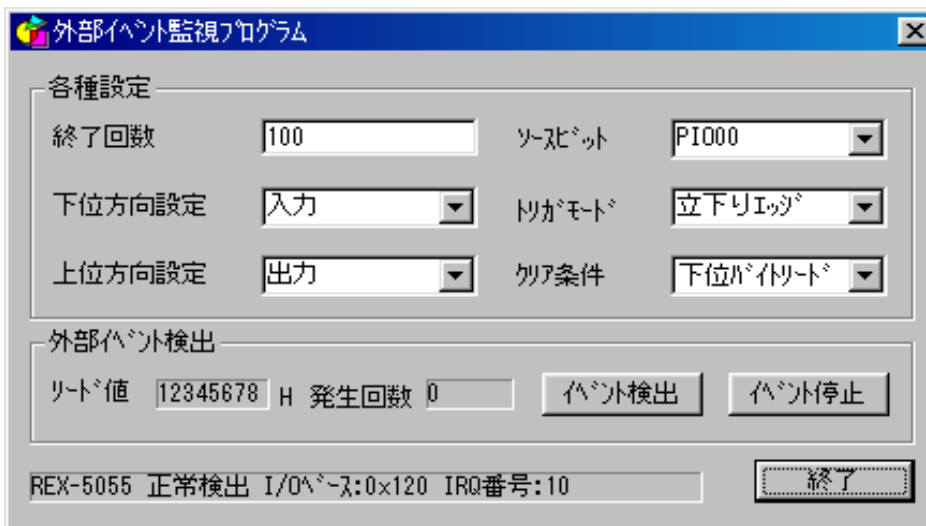
```

外部イベント監視プログラム

本サンプルプログラムは割り込みソースピットで指定したポートに外部機器からの割り込み信号を入力し、オンオフの状態変化を検出するサンプルプログラムです。

割り込みソースピットを指定して StartHWIntPostMessage() を実行することにより、割り込み発生に同期したユーザ定義メッセージ WM_VXDEVENT が割り込みハンドラから送られてきます。このとき、追加情報 wParam の上位バイトには PIO7-PIO0 の読み込み値が、下位バイトには PIO15-PIO8 の読み込み値がセットされています。また、lParam には割り込み発生の累計カウンタ数がセットされています。

リソース取得、方向設定方法については、基本デジタル入出力プログラムをご参照ください。



```
void Cmd_OnStartEvent ( HWND hwnd )
{
    WORD    Status;
    WORD    DataRegLo;    // データレジスタ下位バイトの入出力方向
    WORD    DataRegHi;    // データレジスタ上位バイトの入出力方向
    WORD    IntPIONo;     // 割り込みソース入力ポート番号
    WORD    TrgMode;      // 割り込みトリガーモード
    WORD    ClrMode;      // 割り込みクリア条件
    WORD    MaxCount;     // 割り込み終了回数

    Status = StartHWIntPostMessage( hwnd, MyIOBase, MyIrqNo, DataRegLo, DataRegHi, IntPIONo,
    TrgMode, ClrMode, MaxCount );
    if ( Status != 0 )
    {
        sprintf( MsgBuf, "StartHWIntPostMessage() 戻り値:%d", Status );
        SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );
    }
}
```

```
BOOL CALLBACK Dlg_Proc (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch ( uMsg )
    {
        case WM_INITDIALOG:
            Dlg_OnInitDialog( hwnd, (HWND)wParam, lParam );
            return TRUE;
        case WM_COMMAND:
            Dlg_OnCommand( hwnd, (int)(LOWORD(wParam)), (HWND)lParam, (UINT)HIWORD(wParam));
            return TRUE;
        case WM_VXDEVENT:
            Dlg_OnUserDefineMessage( hwnd, uMsg, wParam, lParam );
            return TRUE;
        case WM_DESTROY:
            Dlg_OnDestroy( hwnd );
            return TRUE;
    }
    return FALSE;
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    WORD MaxCount;          // 割り込み終了回数

    EventCount++;
    sprintf( MsgBuf, "%d", EventCount );
    SetDlgItemText( hwnd, IDS_COUNTER, MsgBuf );
    sprintf( MsgBuf, "0x%x", wParam );
    SetDlgItemText( hwnd, IDS_EVENTVAL, MsgBuf );
    /* 終了回数 */
    MaxCount = (WORD)GetDlgItemInt( hwnd, IDE_COUNT, NULL, FALSE );
    if( MaxCount == EventCount )
    {
        sprintf( MsgBuf, "指定個数終了" );
        SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );
    }
}
```

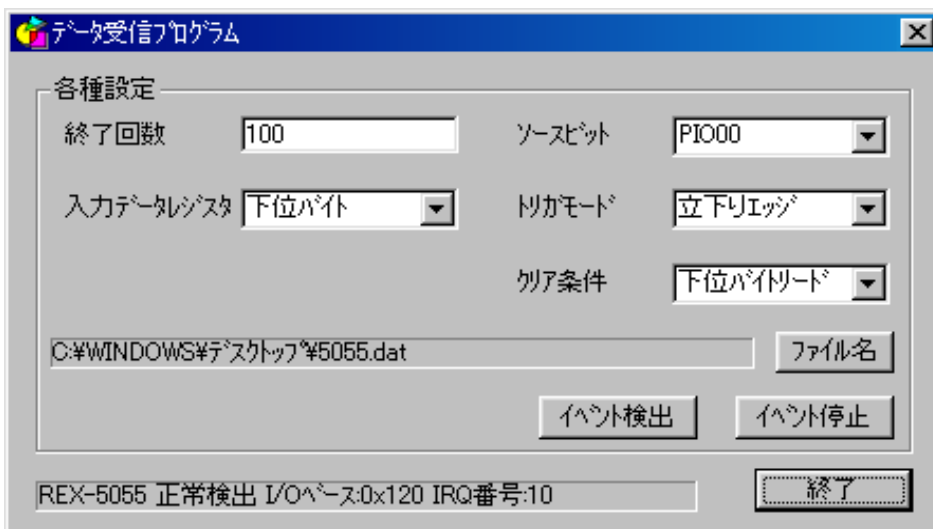
データ受信プログラム

本サンプルプログラムは割り込みハンドラ内ですべての入出力を行うことにより高速割り込み処理を行うことが可能なサンプルプログラムです。

サンプルプログラムの割り込みハンドラは、入力方向に設定されたデータレジスタの値をリードし、指定バッファに格納します。指定回数の割り込み処理が終了すると、呼び出し側プログラムにユーザ定義メッセージをポストし、追加情報 IParam には割り込み発生 of 累計カウンタ数がセットされています。

入出力データレジスタと割り込み終了回数を指定して、StartHWIntMyVxdBuf() を実行することにより、割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。

リソース取得、方向設定方法については、基本デジタル入出力プログラムをご参照ください。



```

void Cmd_OnStartEvent ( HWND hwnd )
{
    WORD    Status;
    WORD    DataReg;        // 入力データレジスタ
    WORD    IntPIONo;      // 割り込みソース入力ポート番号
    WORD    TrgMode;       // 割り込みトリガーモード
    WORD    ClrMode;       // 割り込みクリア条件
    WORD    MaxCount;      // 割り込み終了回数

    Status = StartHWIntMyVxdBuf( hwnd, MyIOBase, MyIrqNo, DataReg, IntPIONo, TrgMode, ClrMode,
    MaxCount, pRcvData );
    if ( Status != 0 )
    {
        sprintf( MsgBuf, "StartHWIntMyVxdBuf()エラー:%d", Status );
        SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );
        return;
    }
}

```

```
voidDlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    WORD    Status;
    WORD    EventCount;
    HANDLE  hFile;           // ファイルハンドル
    DWORD   nBytesRead;     // リードしたバイト数
    char    szFile[MAX_PATH]; // ファイル名

    MessageBeep(0xFFFFFFFF);
    /* サイス取得 */
    EventCount = GetDlgItemInt( hwnd, IDE_COUNT, NULL, FALSE );
    /* 選択された送信ファイル名取得 */
    GetDlgItemText( hwnd, IDS_RCVFILE, szFile, sizeof(szFile) );

    /* ファイルオープン */
    hFile = CreateFile( szFile,
                       GENERIC_WRITE,           /* アクセスモード */
                       0,                       /* 共有モード */
                       ( LPSECURITY_ATTRIBUTES )NULL, /* ホイタ */
                       CREATE_ALWAYS,         /* 創作方法 */
                       FILE_ATTRIBUTE_NORMAL,  /* ファイルの限定 */
                       ( HANDLE )NULL);       /* テンプレートファイルのハンドル */

    Status = WriteFile( hFile, pRcvData, EventCount, &nBytesRead, NULL );
    if ( (Status != TRUE) || (nBytesRead != EventCount) )
    {
        CloseHandle( hFile );
        LocalFree( pRcvData );
        sprintf( MsgBuf, "WriteFile error" );
        SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );
        return ;
    }
    CloseHandle( hFile );
    LocalFree( pRcvData );
    sprintf( MsgBuf, "指定個数終了" );
    SetDlgItemText( hwnd, IDS_STATUS, MsgBuf );
}
```

(3-5) BASIC 言語 API ライブラリ解説

(3-5-1) Visual BASIC によるアプリケーション開発

Visual BASIC 5.0 以上を使って REX-5055 DIO PC カードを制御するアプリケーションを開発する場合は、本製品に添付されている 32Bit 版 DLL の関数をコールする必要があります。また、割り込み制御を行う場合は OLE カスタムコントロール(OCX)を使用します。

Visual BASIC で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、

モジュールファイルで DLL 関数の参照宣言を行う。

必要があります。

また、割り込み制御を行う場合は割り込みハンドラから送られてくるユーザ定義メッセージを Visual BASIC 側のアプリケーションで受け取るために、本製品に添付されている OLE カスタムコントロール MBOX を使用します。MBOX の使用方法については、(3-5-2)カスタムコントロールを参照してください。

アプリケーション作成上のアドバイス

PC カードに割り当てられてるリソースを取得する

Windows2000/XP では、Windows95/98/Me と同様に、**GetMyCardResource()**をサポートしています。アプリケーションの初期化手続き部分で **GetMyCardResource()**により PC カードに割り当てられている I/O ベースアドレス・IRQ 番号を取得してください。

PC カードへの入出力

Windows95 と同様、Visual C では、I/O 入出力関数がサポートされていません。従って、PC カードへの I/O 入出力は DLL でサポートされている **OutPort()**・**InPort()**・**wOutPort()**・**wInPort()**を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、

(1) 割り込みハンドラからポストメッセージにより割り込み通知を受け取る方法

(2) 割り込みハンドラ内で全ての処理を行う方法

があります。高速な処理が要求されるような場合は、(2)の方法で行う必要があります。詳細は、各サンプルプログラムの説明を参照してください。

DLL 関数仕様

GetMyCardResource

リソース情報の取得

書式	Declare Function GetMyCardResource Lib "diolib2K.dll" (ByVal hWnd As Long, ByVal MyCardName As String, ByVal NameLen As Integer, ByVal SlotNo As Integer, IOBase As Integer, IrqNo As Integer) As Long	
機能	指定スロットに挿入されているカードが指定のカード名と一致するか否か調べます。一致した場合は、割り当てられている I/O ベースアドレス・IRQ リソース情報を返します。	
引数	hWnd	➤ ウィンドウハンドル
	MyCardName	➤ カード識別名を示す文字列へのメモリアドレス
	NameLen	➤ 文字列バッファの長さ
	SlotNo	(Windows95/98 互換用)
	IOBase	➤ I/O リソース情報を格納する変数アドレス
	IrqNo	➤ IRQ リソース情報を格納する変数アドレス
戻値	正常終了時 0 を返し、リソースが正常に取得できなかった場合は -1 を返します。	

StartHWIntPostMessage**ユーザ定義メッセージ割り込みの開始**

書式 Declare Function **StartHWIntPostMessage** Lib "diolib2K.dll" (ByVal **hWnd** As Long, ByVal **MyIOBase** As Integer, ByVal **MyIrqNo** As Integer, ByVal **DirDataRegLo** As Integer, ByVal **DirDataRegHi** As Integer, ByVal **IrqPin** As Integer, ByVal **IrqMode** As Integer, ByVal **IntClrMode** As Integer, ByVal **StopCount** As Integer) As Long

機能 ユーザ定義メッセージ版の割り込み処理を開始します。割り込みが発生すると割り込みハンドラからアプリケーションにメッセージを通知します。wParam の上位バイトに PI07-PI00 の読み込み値が、下位バイトに PI015-PI08 の読み込み値がセットされています。また、lParam には割り込み発生時の累計カウンタ数がセットされています。

引数

hWnd	➤ ユーザアプリケーションのウィンドウハンドル
MyIOBase	➤ カードに割り当てられている I/O ベースアドレス
MyIrqNo	➤ カードに割り当てられている IRQ 番号
DirDataRegLo	➤ データレジスタ下位バイトの入出力方向 0:入力方向 1:出力方向
DirDataRegHi	➤ データレジスタ上位バイトの入出力方向 0:入力方向 1:出力方向
IrqPin	➤ 割り込みソース入力ポート番号 0:PI00 1:PI01 2:PI02 … 15:PI015
IrqMode	➤ 割り込みトリガーモード 0:立下りエッジ 1:立上りエッジ
IntClrMode	➤ 割り込みクリア条件 0:下位バイトリード 1:上位バイトリード 2:下位バイトライト 3:上位バイトライト
StopCount	➤ 割り込み終了回数

戻値 正常終了 0 を返します。I/O ベースアドレス設定エラー等は -1 を返し、その他のマイナスエラーは DirDataRegLo・DirDataRegHi・IrqMode・IntClrMode の設定エラーです。

EndHWIntPostMessage**ユーザ定義メッセージ割り込みの終了**

書式 Declare Function **EndHWIntPostMessage** Lib "diolib2K.dll" () As Long

機能 ユーザ定義メッセージ版の割り込み処理を終了します。ユーザ定義メッセージ版による割り込み処理終了時は、必ず EndHWIntPostMessage() を呼び出してください。

引数 なし

戻値 常に 0 を返す

StartHWIntMyVxdBuf

高速割り込み処理の開始

- 書式** Declare Function **StartHWIntMyVxdBuf** Lib "diolib2K.dll" (ByVal **hWnd** As Long, ByVal **MyIOBase** As Integer, ByVal **MyIrqNo** As Integer, ByVal **InputDataReg** As Integer, ByVal **IrqPin** As Integer, ByVal **IrqMode** As Integer, ByVal **IntClrMode** As Integer, ByVal **StopCount** As Integer, **Buf** As Any) As Long
- 機能** IrqPin で指定した上位ポート(下位ポート)に割り込み信号が発生すると下位ポート(上位ポート)のデータレジスタを読み込み、指定したバッファ **Buf** に格納します。指定した **StopCount** 数だけ割り込みが発生すると本関数は処理を呼び出し側に返します。
- 引数**
- | | |
|---------------------|---|
| hWnd | ➤ ユーザアプリケーションのウィンドウハンドル |
| MyIOBase | ➤ カードに割り当てられている I/O ベースアドレス |
| MyIrqNo | ➤ カードに割り当てられている IRQ 番号 |
| InputDataReg | ➤ 入力データレジスタ
0:下位バイト(PI07-PI00) 1:上位バイト(PI015-PI08) |
| IrqPin | ➤ 割り込みソース入力ポート番号
0:PI00 1:PI01 2:PI02 … 15:PI015 |
| IrqMode | ➤ 割り込みトリガーモード
0:立下りエッジ 1:立上りエッジ |
| IntClrMode | ➤ 割り込みクリア条件
0:下位バイトリード 1:上位バイトリード
2:下位バイトライト 3:上位バイトライト |
| StopCount | ➤ 割り込み終了回数 |
| Buf | ➤ データを格納するバッファアドレス |
- 戻値** 正常終了時 0 を返します。引数設定エラー等は-1、メモリアロケーションエラーの場合は-2 を返します。

EndHWIntMyVxdBuf

高速割り込み処理の終了

- 書式** Declare Function **EndHWIntMyVxdBuf** Lib "diolib2K.dll" () As Long
- 機能** 高速版の割り込み処理を終了します。StartHWIntMyVxdBuf()による割り込み処理終了時は、必ず EndHWIntMyVxdBuf()を呼び出してください。
- 引数** なし
- 戻値** 常に 0 を返す

OutPort **1バイトをポート出力**

書式 `Declare Function OutPort Lib "diolib2K.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer`

機能 1バイトをポートに出力

引数 **IOAddr** ➤ ポート番号
OutVal ➤ バイト出力値（上位バイトは無視）

戻値 バイト出力値をそのまま返し、ドライバ呼び出しエラーは-1を返します。

wOutPort **1ワードをポート出力**

書式 `Declare Function wOutPort Lib "diolib2K.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer`

機能 1ワードをポートに出力

引数 **IOAddr** ➤ ポート番号
OutVal ➤ ワード出力値

戻値 ワード出力値をそのまま返し、ドライバ呼び出しエラーは-1を返します。

InPort **1バイトをポート入力**

書式 `Declare Function InPort Lib "diolib2K.dll" (ByVal IOAddr As Integer) As Integer`

機能 ポートから1バイト読み込む

引数 **IOAddr** ➤ ポート番号

戻値 ポートから読み込んだバイトデータを返します。ドライバ呼び出しエラーは-1を返します。（上位バイトは無視してください）

wInPort **1ワードをポート入力**

書式 `Declare Function wInPort Lib "diolib2K.dll" (ByVal IOAddr As Integer) As Integer`

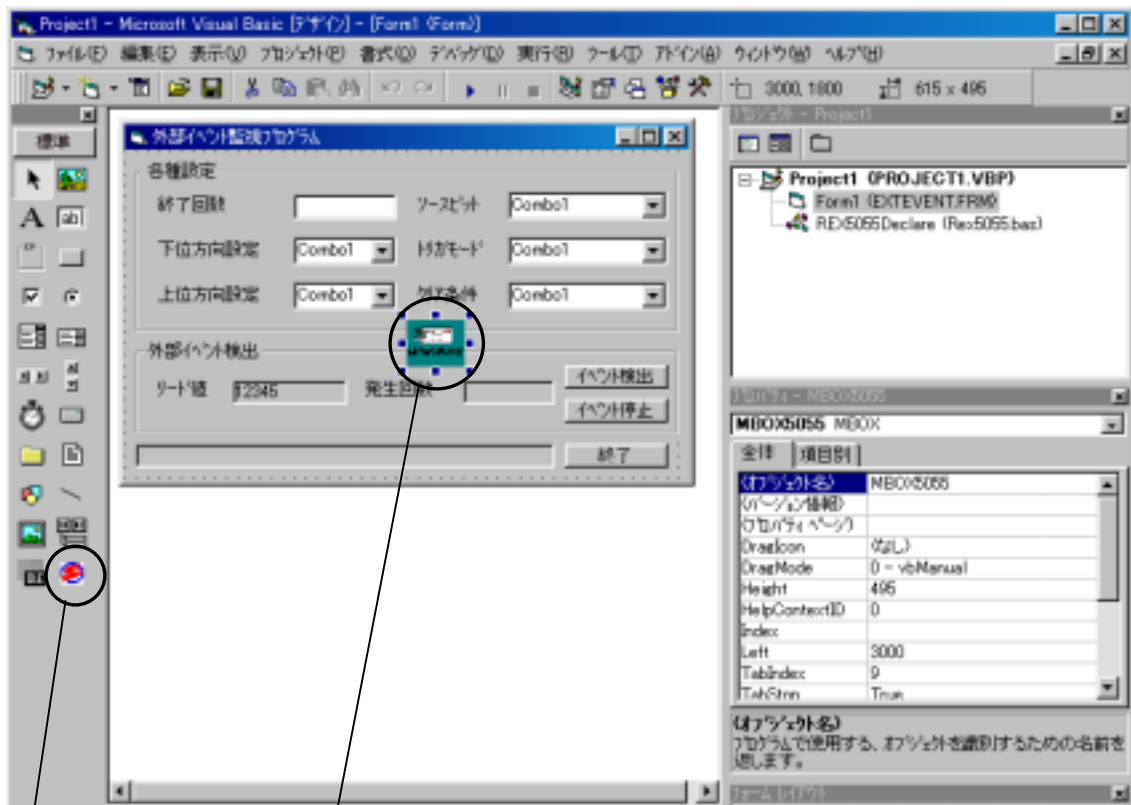
機能 ポートから1ワード読み込む

引数 **IOAddr** ➤ ポート番号

戻値 ポートから読み込んだワードデータを返します。ドライバ呼び出しエラーは-1を返します。

(3-5-2) カスタムコントロール

下記画面は、VB5.0でのデザイン完成時の画面です。割り込み発生に同期したユーザ定義メッセージをVBで作成したプログラムで受け取るために、本製品添付のOLEカスタムコントロール(OCX)MBOXを使用します。次頁より作成の手順について説明します。



本製品に添付されているOLEカスタムコントロール”MBOX”

カスタムコントロール「MBOX OLE Control module」を追加

Step.1 OCX のレジストリ登録 (割り込みサービス使用時必須)

本製品添付の OCX “MBOX5055.OCX” を VB で使用するためには、VB の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリ登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、Windows の DOS BOX から実行します。尚、“REGSVR32.EXE”は VB の CD-ROM に添付されています。

OCX をレジストリ登録するときは、下記構文で実行します。

```
>REGSVR32 “ドライブ名”:¥WINNT¥SYSTEM¥Mbox5055.ocx
```

OCX をレジストリ登録から削除するときは、“/U”を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:¥WINNT¥SYSTEM¥Mbox5055.ocx
```



登録成功メッセージ



登録削除成功メッセージ

Step.2 DIOLIB2K 関数の Declare 宣言

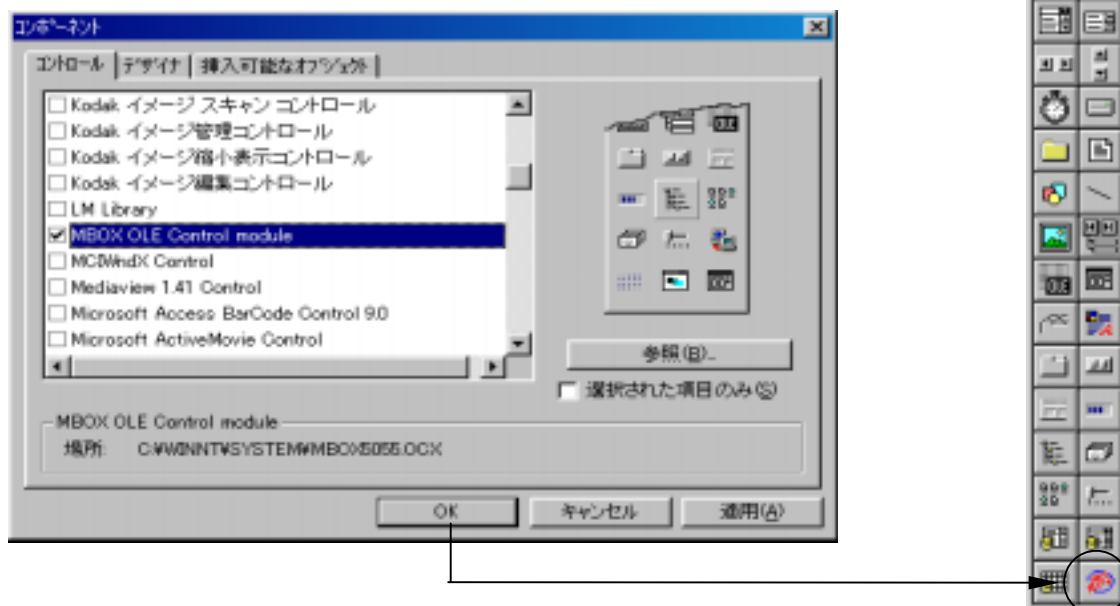
次に、VB プログラムの作成に入ります。VB デザインメニューから新規プロジェクトを作成し、「プロジェクト」の「標準モジュールの追加」から標準モジュールを追加します。追加した標準モジュールファイルで DLL 関数の参照宣言を行います。宣言部分は、サンプルプログラム“REX5055.BAS”の下記部分をコピーしてください。

```
'DLL 外部関数の参照宣言
Declare Function OutPort Lib "diolib2K.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer
Declare Function wOutPort Lib "diolib2K.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer
Declare Function InPort Lib "diolib2K.dll" (ByVal IOAddr As Integer) As Integer
Declare Function wInPort Lib "diolib2K.dll" (ByVal IOAddr As Integer) As Integer
Declare Function StartHWIntPostMessage Lib "diolib2K.dll" (ByVal hWnd As Long, ByVal MyIOBase As Integer,
ByVal MyIrqNo As Integer, ByVal DirDataRegLo As Integer, ByVal DirDataRegHi As Integer, ByVal IrqPin As
Integer, ByVal IrqMode As Integer, ByVal IntClrMode As Integer, ByVal StopCount As Integer) As Long
Declare Function EndHWIntPostMessage Lib "diolib2K.dll" () As Long
Declare Function GetMyCardResource Lib "diolib2K.dll" (ByVal hWnd As Long, ByVal MyCardName As String, ByVal
NameLen As Integer, ByVal SlotNo As Integer, IOBase As Integer, IrqNo As Integer) As Long
Declare Function StartHWIntMyVxdBuf Lib "diolib2K.dll" (ByVal hWnd As Long, ByVal MyIOBase As Integer, ByVal
MyIrqNo As Integer, ByVal InputDataReg As Integer, ByVal IrqPin As Integer, ByVal IrqMode As Integer, ByVal
IntClrMode As Integer, ByVal StopCount As Integer, Buf As Any) As Long
Declare Function EndHWIntMyVxdBuf Lib "diolib2K.dll" () As Long
```

Step.3 MBOX OLE Control Module の追加 (割り込みサービス使用時必須)

VB5.0/6.0 の場合、VB デザインメニューの「プロジェクト」の「コンポーネント」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。

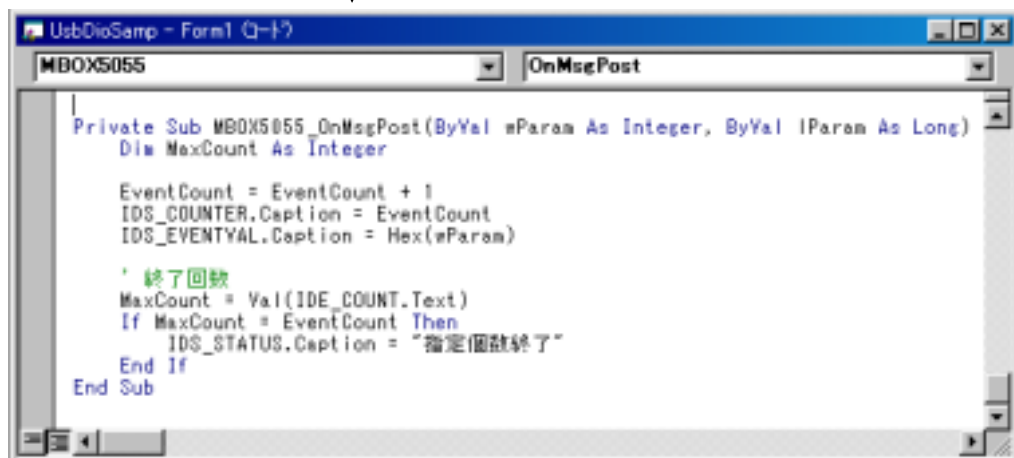
➤ VB5.0/6.0 の場合



Step.4 フォームに MBOX(OCX)を貼り付ける (割り込みサービス使用時必須)

フォームを作成し、割り込みハンドラが割り込み起動元プログラムに送るユーザ定義メッセージを受け取るための MBOX(OCX)を貼り付けます。これにより、割り込みが発生すると MBOX がサービスするプロシージャ

MBOX5055_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long) が呼び出されます。この中で、割り込み通知に同期した処理を記述します。



(3-5-3) Visual BASIC サンプルプログラム

本製品には、以下の3つのサンプルプログラムを添付しております。

基本デジタル入出力プログラム

(割り込みを使わないで単純に入出力を行うサンプルプログラム)

外部イベント監視プログラム

(割り込みを使ったサンプルプログラム)

データ受信プログラム

(高速割り込みを使ったサンプルプログラム)

次頁より、各サンプルプログラムについて解説いたします。

(詳細については各サンプルプログラムのソースをご参照ください。)

Windows95/98/Me で作成したアプリケーションを Windows2000/XP で使用する場合

基本的にはモジュールファイルで DLL 関数の参照宣言を行い、コンパイルすることによって使用可能になります。

但し、以下の関数を使用されている場合は、Windows2000/XP の DIOLIB2K.DLL ではサポートしておりませんのでご注意ください。

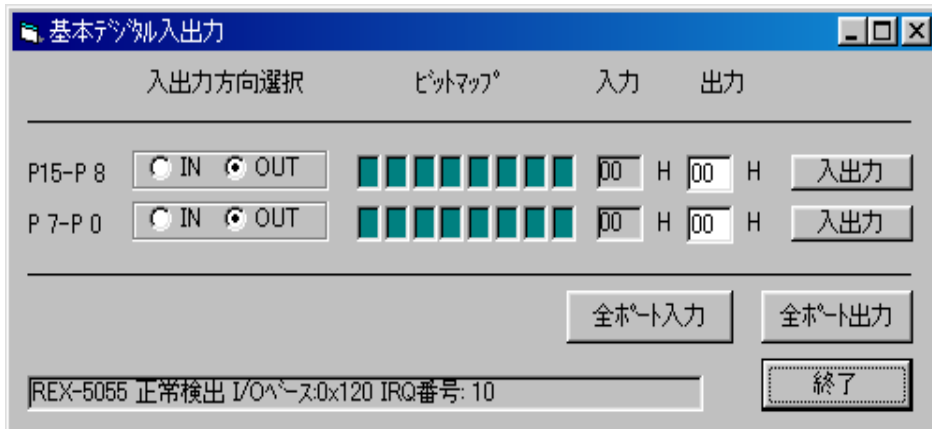
ShowCardUtil()
ResistAsyncProcCall()
ReleaseAsyncProcCall()
StartHWIntMyVxD() ¹
EndHWIntMyVxD() ²
GetDllversion()

1 . Windows2000/XP では、StartHWIntMyVxdBuf()をご使用ください。

2 . Windows2000/XP では、EndHWIntMyVxD()をご使用ください。

基本デジタル入出力プログラム

本サンプルプログラムは 16 ポート(8 ポート単位)を使って外部機器とオンオフ入出力を行う場合のサンプルプログラムです。



DIO カードを制御するためには、プログラム実行時にカードに割り当てられている I/O ベースアドレスと割り込み番号のリソース情報を取得する必要があります。また、Output()、Inport()により方向設定を行ってください。

リソースの取得、方向設定の記述例

```
Global Const MyName = "REX5055 DIO PC Card" ' 製品情報ファイルの製品名の定義
Global MyIOBase As Integer ' カード I/O ベースアドレス
Global MyIrqNo As Integer ' カード IRQ 番号

Private Sub Form_Load()
    Dim RetCode As Integer
    Dim STATUS As Integer

    RetCode = GetMyCardResource(hWnd, MyCardName, Len(MyCardName), 0, MyIOBase, MyIrqNo)
    If RetCode = 0 Then
        IDS_STATUS.Caption = "REX-5055 正常検出 I/Oベース:0x" + Hex(MyIOBase) + " IRQ番号:" +
            Str(MyIrqNo)

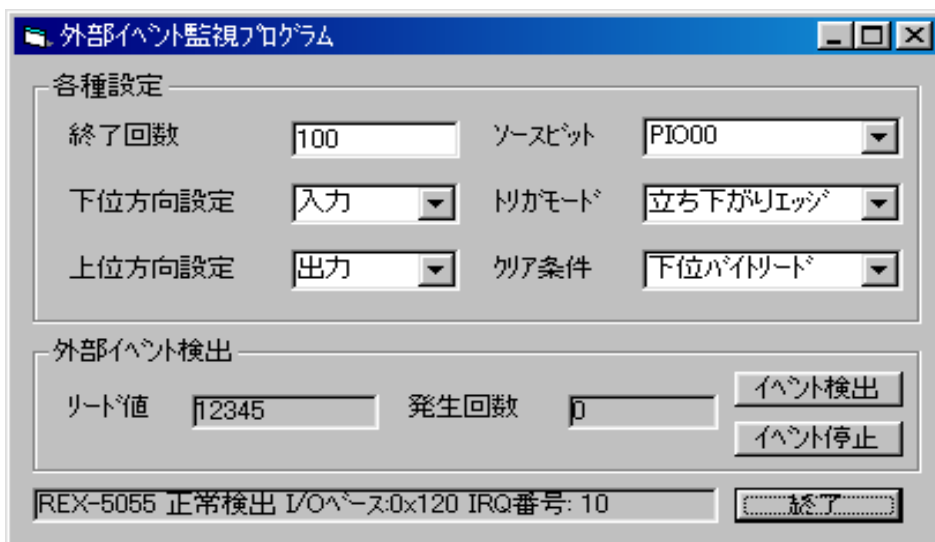
        ' 出力方向設定 (PI00-PI07)
        STATUS = Output(&H120, &H1)
        ' 下位バイト (PI00-PI07) すべて出力
        STATUS = Output(&H122, &HFF)
        Exit Sub
    End If
    IDS_STATUS.Caption = "DIOカード検出エラー"
End Sub
```

外部イベント監視プログラム

本サンプルプログラムは割り込みソースビットで指定したポートに外部機器からの割り込み信号を入力し、オンオフの状態変化を検出するサンプルプログラムです。

割り込みソースビットを指定して StartHWIntPostMessage()を実行することにより、割り込み発生に同期したユーザ定義メッセージが割り込みハンドラから送られてきます。このとき、追加情報 wParam の上位バイトには PIO7-PIO0 の読み込み値が、下位バイトには PIO15-PIO8 の読み込み値がセットされています。また、lParam には割り込み発生の累計カウンタ数がセットされています。

リソース取得、方向設定方法については、基本デジタル入出力プログラムをご参照ください。



```
Private Sub IDB_START_EVENT_Click()  
    Dim STATUS As Integer  
    Dim DataRegLo As Integer      ' データレジスタ下位バイトの入出力方向  
    Dim DataRegHi As Integer     ' データレジスタ上位バイトの入出力方向  
    Dim IntPIONo As Integer      ' 割り込みソース入力ポート番号  
    Dim TrgMode As Integer       ' 割り込みトリガーマード  
    Dim ClrMode As Integer       ' 割り込みクリア条件  
    Dim MaxCount As Integer      ' 割り込み終了回数  
    Dim OleHandle As Long        ' MBOX5055.OCX ハンドル  
  
    'OLE のウィンドウハンドル取得  
    OleHandle = MBOX5055.GetMboxWnd()  
    If (OleHandle = 0) Then  
        MsgBox "OLE のハンドルが取得できません。", vbOKOnly + vbCritical, "エラー"  
        Exit Sub  
    End If  
    STATUS = StartHWIntPostMessage(OleHandle, MyIOBase, MyIrqNo, DataRegLo, DataRegHi,  
    IntPIONo, TrgMode, ClrMode, MaxCount)  
    If STATUS <> 0 Then  
        IDS_STATUS.Caption = "StartHWIntPostMessage()エラー : " + Str(STATUS)  
        Exit Sub  
    End If  
End Sub
```

```
Private Sub MBOX5055_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)  
    Dim MaxCount As Integer      ' 割り込み終了回数  
  
    EventCount = EventCount + 1  
    IDS_COUNTER.Caption = EventCount  
    IDS_EVENTVAL.Caption = Hex(wParam)  
  
    ' 終了回数  
    MaxCount = Val(IDE_COUNT.Text)  
    If MaxCount = EventCount Then  
        IDS_STATUS.Caption = "指定個数終了"  
    End If  
End Sub
```

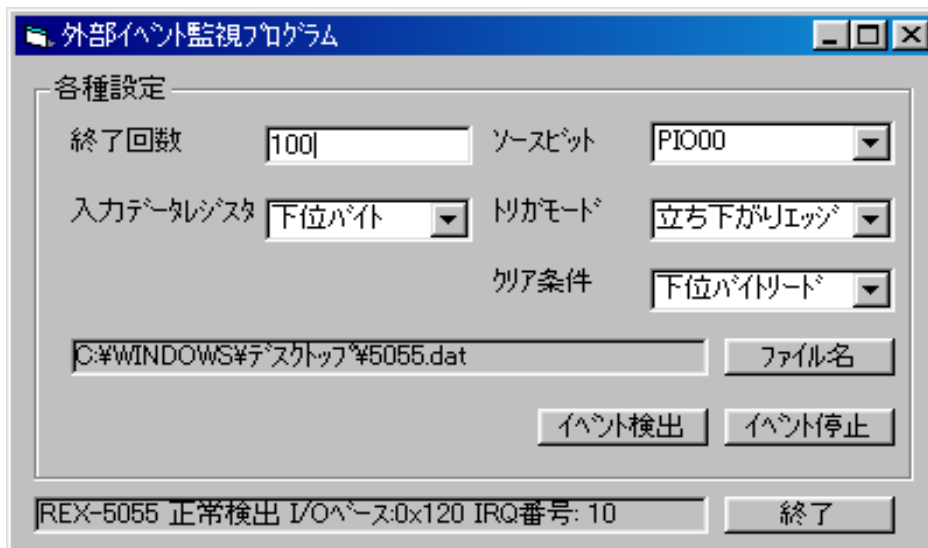
データ受信プログラム

本サンプルプログラムは割り込みハンドラ内ですべての入出力を行うことにより高速割り込み処理を行うことが可能なサンプルプログラムです。

サンプルプログラムの割り込みハンドラは、入力方向に設定されたデータレジスタの値をリードし、指定バッファに格納します。指定回数の割り込み処理が終了すると、呼び出し側プログラムにユーザ定義メッセージをポストし、追加情報 IParam には割り込み発生時の累計カウンタ数がセットされています。

入出力データレジスタと割り込み終了回数を指定して、StartHWIntMyVxdBuf()を実行することにより、割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。

リソース取得、方向設定方法については、基本デジタル入出力プログラムをご参照ください。



```
Private Sub IDB_START_EVENT_Click()  
    Dim STATUS As Integer  
    Dim DataReg As Integer          ' 入力データレジスタ  
    Dim IntPIONo As Integer        ' 割り込みソース入力ポート番号  
    Dim TrgMode As Integer        ' 割り込みトリガーモード  
    Dim ClrMode As Integer        ' 割り込みクリア条件  
    Dim MaxCount As Integer       ' 割り込み終了回数  
    Dim OleHandle As Long        ' MBOX5055.OCX ハンドル  
  
    ReDim pRcvData(MaxCount - 1) As Byte  
    'OLE のウィンドウハンドル取得  
    OleHandle = MBOX5055.GetMboxWnd()  
    If (OleHandle = 0) Then  
        MsgBox "OLE のハンドルが取得できません。", vbOKOnly + vbCritical, "エラー"  
        Exit Sub  
    End If  
    STATUS = StartHWIntMyVxdBuf(OleHandle, MyIOBase, MyIrqNo, DataReg, IntPIONo, TrgMode,  
    ClrMode, MaxCount, pRcvData(0))  
    If STATUS <> 0 Then  
        IDS_STATUS.Caption = "StartHWIntMyVxdBuf()エラー:" + Str(STATUS)  
        Exit Sub  
    End If  
  
End Sub
```

```
Private Sub MBOX5055_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)  
    Dim EventCount As Integer      ' 割り込み終了回数  
  
    Beep  
    ' サイズ取得  
    EventCount = IDE_COUNT.Text  
    ' 選択された送信ファイル名取得  
    RcvFileName = IDS_RCVFILE.Caption  
    ' ファイルオープン  
    Open RcvFileName For Binary As #1  
  
    Put #1, , pRcvData  
  
    ' ファイルを閉じる  
    Close #1  
    IDS_STATUS.Caption = "指定個数終了"  
  
End Sub
```

(空白ページ)

第4章 MS-DOS/Windows3.1 解説

(4-1) MS-DOS/Windows3.1 でのインストール

(4-1-1) イネーブラのインストール

PC カードのイネーブルを行うために、本製品添付のイネーブラのインストールを行う必要があります。PC-AT または互換機でお使いの場合は、DOS/V 版カードサービス対応イネーブラとポイントイネーブラを用意しています。どちらを使用するか選択してください。PC-9800 シリーズの場合は、PC-9800 シリーズ版カードサービス対応イネーブラを使用します。

☐ DOS/V 版カードサービス対応イネーブラのインストール

添付のフロッピーからハードディスクにカードサービス対応イネーブラをコピーしてください。

```
C:¥>COPY A:¥ENABLER¥DOSV¥REXDIO.EXE C:¥CARD
```

REXDIO.EXE はデバイスドライバですので、CONFIG.SYS に登録して使います。

☐ DOS/V 版ポイントイネーブラのインストール

添付のフロッピーからハードディスクにポイントイネーブラをコピーしてください。

```
C:¥>COPY A:¥ENABLER¥DOSV¥DIO365.EXE C:¥CARD
```

DIO365.EXE は、カード挿入状態で DOS プロンプトから実行します。

☐ PC-9800 シリーズ版カードサービス対応イネーブラのインストール

添付フロッピーディスクからハードディスクに PC-9800 シリーズ用カードサービス対応イネーブラをコピーしてください。

```
A:¥>COPY B:¥ENABLER¥PC98¥REXDIO98.EXE A:¥CARD
```

REXDIO98.EXE はデバイスドライバですので、CONFIG.SYS に登録して使います。

☐ カードイネーブラとは...

パソコンのスロットに挿入した直後はメモリーカードとして認識されており、I/O カードとしての動作はしていません。このメモリーカードの中には、PC カードを I/O カードにコンフィグレーションするために必要な情報(カード属性情報)が書き込まれています。

PC カードを I/O カードとして機能させるためには、コンフィグレーションソフト「イネーブラ」が必要となります。イネーブラは、PC カードのカード属性情報を読み込んだ後、その情報に基づいて PC カードを所定の I/O カードにコンフィグレーションします。イネーブラによるコンフィグレーションが正常に行なわれて、はじめて PC カードは I/O カードとして使える状態になります。

(4-1-2) DOS/V 版カードサービス対応イネーブラを使用する場合

最初に、カードサービスのインストールが完了しているか確認してください。次に、本製品添付のカードサービス対応イネーブラをカードサービスドライバの後に追加します。カードサービス対応イネーブラには、下記オプション仕様に従って必要なオプション情報を記載します。

次頁以降に CONFIG.SYS の登録例を示します。CONFIG.SYS の内容はお使いのパソコンにより多少異なることがあります。登録内容については、ご利用されているパソコン添付のカードサービスマニュアル記載内容に従ってください。

オプション仕様

```
DEVICE=C:\%CARD%\REXDIO.EXE [/<オプション>] [ ] … [ ]
```

オプション	解 説
/SLOTnBASE=x	<p>スロット番号<n>と I/O ベースアドレス<x>を指定します。</p> <p>(1)スロット番号は1から4を指定します。パソコン側の PCMCIA スロットの順番に対応していますので、予めスロット番号を調べておく必要があります。</p> <p>1枚のカードしか使用しないときは、0を指定することもできます。0が指定された場合はスロットを順に調べて最初に見つかった DIO カードのみをイネーブルします。</p> <p>(2)スロット n に挿入するカードに割り当てる I/O ベースアドレスを16進表記で指定します。何も指定しない場合は、300h にアドレスを割り当ります。</p>
/SLOTnIRQ=x	<p>スロット番号<n>と割り込み番号<x>を10進表記で指定します。</p> <p>(1)何も指定しない場合、割り込みは使用しません。</p> <p>(2)指定可能な割り込み番号は、5,7,10,11,12,15 です。</p>

■ 複数枚のカードを使用する場合の留意点

パソコンの PC カードスロットには、スロット 1・スロット 2 というようにスロット番号が付いています。上記イネーブラオプションで指定する"/SLOTn"は、パソコンで決められているスロット番号に対応しています。複数枚のカードの抜き差しを行った場合にカードに割り当てられる I/O アドレス・IRQ 番号は、オプションで指定されたスロットに対応する I/O アドレス・IRQ 番号になります。

CONFIG.SYS 記述例1 IBM カードサービス PlayAtWill の場合

```
DEVICE=C:\WINDOWS\EMM386.EXE RAM X=C800-CFFF (1)
.....
DEVICEHIGH=C:\EZPLAY\SSDPCIC1.SYS (2)
DEVICEHIGH=C:\EZPLAY\IBMDOSCS.SYS (3)
DEVICEHIGH=C:\EZPLAY\RMUDOSAT.SYS /SH=1 /NS=1 /MA=C800-CFFF (4)
.....
DEVICEHIGH=C:\EZPLAY\AUTODRV.SYS (5)
.....
DEVICE=C:\CARD\REXDIO.EXE /Slot1Base=300 /Slot1Irq=5
                               /Slot2Base=304 /Slot2Irq=7 (6)
```

【解説】

- (1) 拡張メモリマネージャが[C800～CFFF]のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。ソケットサービスファイル名はインストール時に選択したマシーンにより異なります。
- (3) カードサービスを起動しています。
- (4) リソースマップユーティリティに対しカードサービスが[C800～CFFF]のメモリウィンドウセグメントを使用するように指定しています。
- (5) カードサービス標準イネーブラを起動しています。
- (6) 本製品添付のカードサービス版イネーブラを起動しています(1行で記述してください)。スロット1に挿入されるカードにI/Oベースアドレス300hおよびIRQ5を、スロット2に挿入されるカードにI/Oベースアドレス304hおよびIRQ7を割り当てるように指定しています。

■ アプリケーションからカード情報を取得する

実際にカードに割り当てられたI/Oアドレスと割り込み番号等のコンフィグレーション情報は、カードサービスコール GetConfigurationInfo(ファンクション: 04h)を使うことにより取得することができます。DIOライブラリの関数 GetConfigurationInfo()を使うことにより、アプリケーションの中から、このI/Oアドレス及び割り込み番号を取得することができます。

CONFIG.SYS 記述例 2 COMPAQ SystemSoft CardSoft の場合

```

DEVICE=C:\DOS\EMM386.EXE 1024 X=D000-DFFF (1)
DEVICE=C:\CPQDOS\SSVLSI.EXE (2)
DEVICE=C:\CPQDOS\CS.EXE (3)
DEVICE=C:\CPQDOS\CSALLOC.EXE (4)
INSTALL=C:\CPQDOS\CARDID.EXE C:\CPQDOS\CARDID.INI (5)
.....
DEVICE=C:\CARD\REXDIO.EXE /Slot0Base=300 (6)

```

【解説】

- (1) 拡張メモリマネージャが[D000 ~ DFFF]のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) リソースマネージャを起動しています。
- (5) カードサービス添付の標準イネーブラを起動しています。
- (6) 本製品添付のカードサービス版イネーブラを起動しています。
1枚のカードしか使用しない時の例になります。挿入スロットを指定しないで、挿入されたカードにI/O ベースアドレスを300hに割り当て、割り込みは使用しません。

CONFIG.SYS 記述例 3 TOSHIBA Phoenix PCM Plus の場合

```

DEVICE=C:\DOS\EMM386.EXE RAM P0=D000 P1=D400 P2=D800 P3=DC00 I=B000-B7FF X=C800-C8FF (1)
.....
DEVICE=C:\PCPLUS3\CNFIGMAN.EXE /DEFAULT
DEVICE=C:\PCPLUS3\PCMSS.EXE (2)
DEVICE=C:\PCPLUS3\PCMCS.EXE (3)
DEVICE=C:\PCPLUS3\PCMRMAN.SYS
DEVICE=C:\PCPLUS3\PCMSCD.EXE (4)
.....
DEVICE=C:\CARD\REXDIO.EXE /Slot1Base=300 /Slot1Irq=5
                               /Slot2Base=304 /Slot2Irq=7
                               /Slot3Base=308 /Slot3Irq=10
                               /Slot4Base=30C /Slot4Irq=11 (5)

```

- (1) 拡張メモリマネージャが[C800 ~ C8FF]のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) カードサービス添付の標準イネーブラを起動しています。
- (5) 本製品添付のカードサービス版イネーブラを起動しています(1行で記述してください)。スロット1から4に挿入されるカードに、それぞれI/O ベースアドレス300h・304h・308h・30Chを、割り込みIRQ5・IRQ7・IRQ10・IRQ11を割り当てます。

(4-1-3) PC-9800 シリーズ版カードサービス対応イネーブラを使用する場合

最初に、カードサービスのインストールが完了しているか確認してください。次に、本製品添付のカードサービス対応イネーブラをカードサービスドライバの後に追加します。カードサービス対応イネーブラには、下記オプション仕様に従って必要なオプション情報を記載します。

次頁に CONFIG.SYS の登録例を示します。CONFIG.SYS の内容はお使いのパソコンにより多少異なることがあります。登録内容については、ご利用されているパソコン添付のマニュアル記載内容に従ってください。

オプション仕様

```
DEVICE=A:¥CARD¥REXDI098.EXE [/<オプション>] [ ] … [ ]
```

オプション	解 説
/SLOTnBASE=x	<p>スロット番号<n>と I/O ベースアドレス<x>を指定します。</p> <p>(1)スロット番号は1から4を指定します。パソコン側の PCMCIA スロットの順番に対応していますので、予めスロット番号を調べておく必要があります。</p> <p>1枚のカードしか使用しないときは、0を指定することもできます。0が指定された場合はスロットを順に調べて最初に見つかった DIO カードのみをイネーブルします。</p> <p>(2)スロット n に挿入するカードに割り当てる I/O ベースアドレスを 16 進表記で指定します。何も指定しない場合は、0D0h にアドレスを割り当めます。</p>
/SLOTnIRQ=x	<p>スロット番号<n>と割り込み番号<x>を 10 進表記で指定します。</p> <p>(1)何も指定しない場合、割り込みは使用しません。</p> <p>(2)指定可能な割り込み番号は、3,5,6,10,12,13 です。</p>

■ PC-9800 シリーズ版対応カードサービスについて...

PC-9800 シリーズで初期の機種では注 1)のソケットサービスしか提供されておらず、本製品添付のイネーブラを使ってカードをイネーブルすることはできません。別売版カードサービスを入手してください。PC-9800 シリーズ対応カードサービスと搭載機種は下表の通りです。

CS バージョン識別名	SS,CS ドライバー名	搭載パソコン機種
別売版カードサービス SystemSoftCardSSoft2.10 Version2.06	SSMECIA.SYS, CS.EXE	PC-9821 Ne PC-9801 NX/C,P,NS/A,NL/R
バンドル版カードサービス SystemSoftCardSSoft2.10 Version2.06	SSDRV.EXE, CS.EXE	PC-9821 Np,Ns,Ne2,Nd,Ld Nf,Nm,Lt,Ne3,Nd2 PC-9801 NL/A その他新機種

注 1) ソケットサービス NEC SocketService 2.00 Version 1.00

CONFIG.SYS 記述例 4 NEC バンドル版カードサービス SystemSoft CardSoft

PC-9821 Np,Ns,Ne22,Nd,Ld,Nf,Nm,Lt,Ne3,Nd2,La10/7,Na13/12/9,Nb7 等新機種
PC-9801 NL/A

```

DEVICE=A:¥DOS¥EMM386.EXE /UMB /E=DC00-DFFF                (1)
.....
DEVICE=A:¥DOS¥SSDRV.SYS                                    (2)
DEVICE=A:¥DOS¥CS.EXE                                      (3)
DEVICE=A:¥DOS¥CSALLOC.EXE A:¥DOS¥CSALLOC.INI             (4)
INSTALL=A:¥DOS¥CARDID.EXE A:¥DOS¥CARDID.INI             (5)
.....
DEVICE=A:¥CARD¥REXD1098.EXE /Slot1Base=D0 /Slot1Irq=3
                               /Slot2Base=D4 /Slot2Irq=5   (6)

```

【解説】

- (1) 拡張メモリマネージャが [DC00 ~ DFFF] のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) リソースマネージャを CSALLOC.INI を参照するようにして起動しています。
- (5) カードサービス添付の標準イネーブラを起動しています。
- (6) 本製品添付の PC-9800 シリーズ版カードサービス対応イネーブラを起動しています (1 行で記述してください)。スロット 1 に挿入されるカードに I/O ベースアドレス D0h および IRQ3 を、スロット 2 に挿入されるカードに I/O ベースアドレス D4h および IRQ5 を割り当てるように指定しています。

NEC 別売り版カードサービス SystemSoft CardSoft

PC-9821 Ne
PC-9801 NX/C,P,NS/A,NL/R

```

DEVICE=A:¥DOS¥EMM386.EXE /UMB /E=DA00-DBFF                (1)
.....
DEVICE=A:¥DOS¥SSMECIA.SYS                                  (2)
DEVICE=A:¥DOS¥CS.EXE                                      (3)
DEVICE=A:¥DOS¥CSALLOC.EXE A:¥DOS¥CSALLOC.INI             (4)
INSATLL=A:¥DOS¥CARDID.EXE A:¥DOS¥CARDID.INI             (5)
.....
DEVICE=A:¥CARD¥REXD1098.EXE /Slot0Base=D0                (6)

```

【解説】

- (1) ~ (5) 上記解説参照
- (6) 本製品添付のカードサービス版イネーブラを起動しています。1 枚のカードしか使用しない時の例になります。挿入スロットを指定しないで、挿入されたカードに I/O ベースアドレスを D0h に割り当て、割り込みは使用しません。

(4-1-4) DOS/V 版ポイントイネーブラを使用する場合

DOS/V でカードサービスが提供されていない機種をご利用されている場合、本製品に添付されているポイントイネーブラにより DIO カードをイネーブルすることができます。また、カードサービス等のドライバをメモリーに常駐させるとコンベンショナルメモリーの空き領域が不足して不都合が生じることがあります。このような場合、カードサービスを CONFIG.SYS に登録しないでポイントイネーブラを使ってカードのイネーブルを行うことができます。

ポイントイネーブラは、パソコン本体のメモリーウィンドウを通してカードの情報を読み出します。EMM386.EXE が CONFIG.SYS に組み込まれている場合には、4K バイトのメモリーウィンドウを EMM386.EXE が使用しないように <X=> オプションを追加してください。下記例は、ポイントイネーブラが使用するメモリーウィンドウのセグメントアドレス [DF00-DFFF] を設定する場合の例になります。DOS プロンプトから起動します。

```
DEVICE=C:\EMM386.EXE 512 X=DF00-DFFF
```

オプション仕様

```
C:\>DI0365.EXE [/<オプション>] [ ] … [ ]
```

オプション	解 説
/SLOTnBASE=x	<p>スロット番号<n>と I/O ベースアドレス<x>を指定します。</p> <p>(1)スロット番号は1から4を指定します。パソコン側の PCMCIA スロットの順番に対応していますので、予めスロット番号を調べておく必要があります。</p> <p>1枚のカードしか使用しないときは、0 を指定することもできます。0 が指定された場合はスロットを順に調べて最初に見つかった DIO カードのみをイネーブルします。</p> <p>(2)スロット n に挿入するカードに割り当てる I/O ベースアドレスを 16 進表記で指定します。何も指定しない場合は 300h にアドレスを割り当めます。</p>
/SLOTnIRQ=x	<p>スロット番号<n>と割り込み番号<x>を 10 進表記で指定します。</p> <p>(1)何も指定しない場合、割り込みは使用しません。</p> <p>(2)指定可能な割り込み番号は 5,7,10,11,12,15 です。</p>
/MEM=x	<p>イネーブラが使用するメモリーウィンドウセグメントアドレスを指定します。指定しないときはセグメント DF00h から 4K バイトを使います。</p>

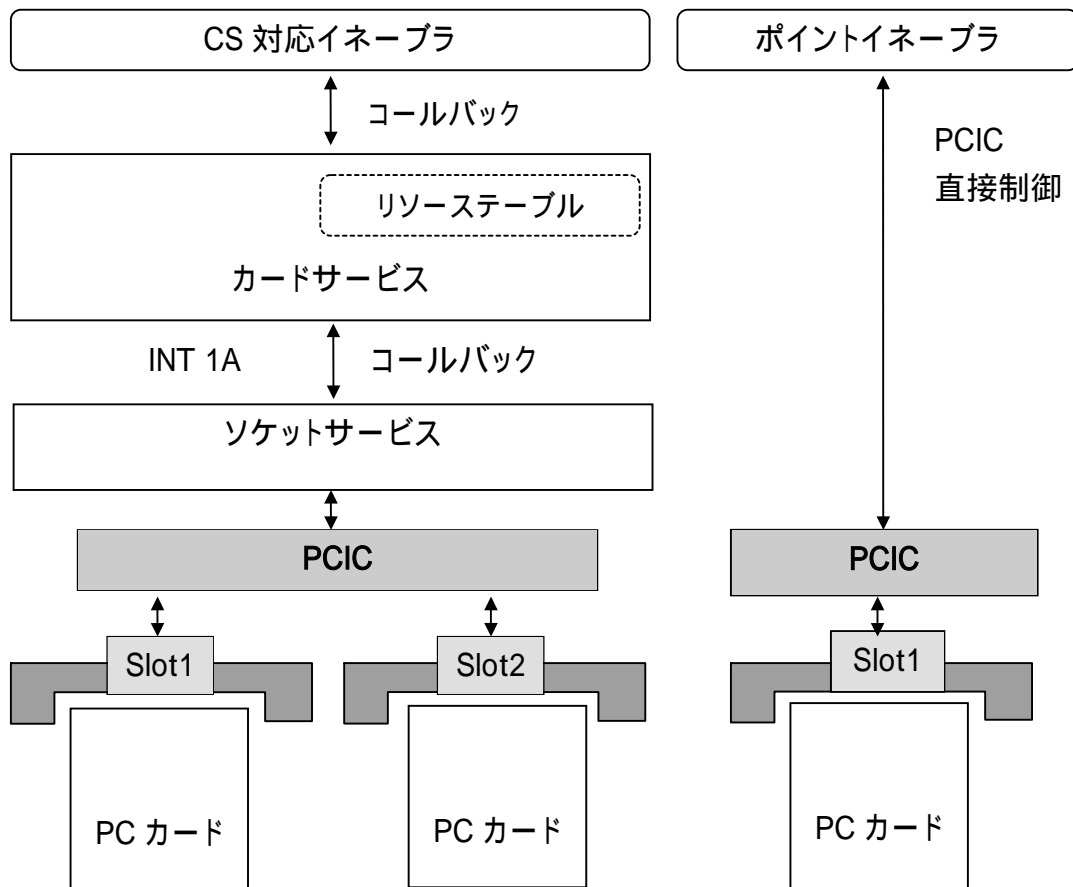
●※注意...※●

PCMCIA コントローラがインテル 82365L または互換チップ以外は動作しません。

❏ カードサービス対応イネーブラとポイントイネーブラ

カードサービス(CS)対応イネーブラは起動された時点で、CS のファンクションセットである GetCardServiceInfo により、CS が常駐しているかチェックします。CS が常駐していれば、イネーブラは CS のファンクションセット RegisterClient により、カードが抜き差しされた時 CS がイネーブラを呼び出すために必要なコールバック情報を登録しメモリに常駐します。PC カードが挿入または抜き取られると、CS は登録されたコールバック情報をもとに全てのイネーブラに抜き差しの通知を行います。CS は、複数の PC カードが使用する I/O アドレス・IRQ のリソースをリソース管理テーブルで管理します。同時に、上記のカード抜き差しの監視を行います。図で示すようにカードが挿入されるとそれを検出してイネーブラに通知します。イネーブラは CS からの通知を受けて自分のカードかどうか調べます。自分のカードの時は、CS に対し必要な I/O アドレスおよび IRQ を割り当ててくれるようにリソースの要求とイネーブルの要求を発行します。この要求を受けて CS は要求されたリソースが他で使われていなければ、ソケットサービス(SS)と呼ばれる低レベルのファンクションセットを呼び出してリソースを確保しカードのイネーブルを行います。

ポイントイネーブラは、PC Card Interface Controller(PCIC)を直接制御してカードをイネーブルします。カードの抜き差しの管理は行いません。



(4-2) MS-DOS ライブラリ解説

(4-2-1) MS-DOS ライブラリ

カードサービスを使って REX-5055 DIO PC カードをイネーブルした場合、実際にカードに割り当てられている I/O ベースアドレスおよび IRQ 情報をカードサービスに問い合わせるために関数が提供されています。I/O ベースアドレスおよび IRQ 情報を作成したアプリケーションで取得する必要がある場合は本ライブラリをリンクしてください。

I/O アドレスおよび IRQ 等の資源情報はイネーブル起動時のオプションで指示している内容になります。従って、特にアプリケーション内で自動取得する必要がない場合は、MS-DOS ライブラリをリンクする必要はありません。

ライブラリモデル	モデルサイズ	ライブラリ名
	スモールモデル	SLIBDIO.LIB
	ラージモデル	LLIBDIO.LIB
	ヒュージモデル	HLIBDIO.LIB

インクルードファイル DIOLIB.H
作成したアプリケーションにインクルードします。

CheckCSRegistration

カードサービス常駐確認

書式 `BOOL CheckCSRegistration (void)`

機能 カードサービスが常駐しているか調べます

引数 なし

戻値 カードサービス常駐していれば 0 以外の値を返します。

GetCSConfigInfo

リソース情報の取得

書式 `BOOL GetCSConfigInfo (WORD Slot, WORD *pIOAdrs, WORD *pIRQNo)`

機能 カードサービスをコールしてカードに割り当てられている I/O アドレス・割り込み番号を取得します

引数 `WORD Slot` : カードが挿入されているスロットの番号
`WORD *pIOAdrs` : I/O アドレス格納先を示すポインタ
`WORD *pIRQNo` : 割り込み番号格納先を示すポインタ

戻値 正常に取得できた場合 0 を返します。その他はエラー。

(4-2-2) MS-DOS サンプルプログラム

RYO-8 との接続

リードリレー出力ユニット(RYO-8)を REX-5055 と接続し、順次1つずつリレーを動作させます。

REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	RYO-8 外部入出力コネクタ信号名
+5V (1)	←→	+5V (1)
GND (19)	←→	GND (12)
PI07 (9)	→	DI-8 (11)
PI06 (21)	→	DI-7 (10)
PI05 (10)	→	DI-6 (9)
PI04 (22)	→	DI-5 (8)
PI03 (11)	→	DI-4 (7)
PI02 (23)	→	DI-3 (6)
PI01 (12)	→	DI-2 (5)
PI00 (24)	→	DI-1 (4)

C 言語プログラム例

添付ディスク中の [RY08.C] がソースプログラムです。

このプログラムは、各種出力ユニット (PH0-8, SSR-81, SSR-83) 等でも同様です。

```

void main( void )
{
    CardInit( IOBase );
    while( !kbhit() ){
        for( BitData = 0; BitData <= 15; BitData++ ){
            printf( "%3d", BitData );
            Relay( IOBase, BitData, 1 );
            delay(40000);
            Relay( IOBase, BitData, 0 );
        }
    }
}

void CardInit( WORD adrs )
{
    outp ( adrs + CTRG, 0x03 );      /* mode = low & high output mode */
    outp ( adrs + INTEBL, 0 );     /* INT Disable */
}

void Relay( WORD adrs, WORD BitData, WORD flg )
{
    BOOL    d;
    d = inpw( adrs + DRL );
    outpw( adrs + DRL, ( d & ( 1 << BitData ) ) | ( flg << BitData ) );
}

```


📁 N88BASIC 言語プログラム例

添付ディスク中の"RYO-8.BAS" がソースプログラムです。

```
10 'SAVE" ryo-8.bas",A
20 'REX-5055 DIO PC Card & RYO-8
30 '-----
120 REXPORT=&HDO          ' Default Port
130 CTRG = 0              ' Controll register ( Write Only )
140 INTEBL = 1           ' INT Enable register ( Write Only )
150 DRL = 2              ' Data register low ( Read / Write )
160 DRH = 3              ' Data register high ( Read / Write )
170 ADRS$ = ""
180 IOADRS = 0
190 SPEED=2000
200 CLS
210 PRINT" I / Oアドレス値の入力",HEX$( REXPORT )
220 INPUT"アドレス値 ( xxxxh ) : ";ADRS$
230 IF LEN( ADRS$ ) > 4 THEN 200
240 IOADRS=VAL( "&h"+ADRS$ )
250 IF IOADRS <> 0 THEN REX5055 = IOADRS ELSE REX5055 = REXPORT
260 OUT REX5055 + CTRG ,&H3          ' mode = low & high output mode
270 OUT REX5055 + INTEBL , 0        ' INT Disable
280 WHILE(INKEY$="")
290   RANP=1
300   FOR BIT = 0 TO 7
310     PRINT USING"###";BIT;
320     OUT REX5055+DRL,RANP
330     FOR I=1 TO SPEED:NEXT
340     RANP=RANP+RANP
350   NEXT
360   OUT REX5055+DRL,0
370   RANP=1
380   FOR BIT = 8 TO 15
390     PRINT USING"###";BIT;
400     OUT REX5055+DRH,RANP
410     FOR I=1 TO SPEED:NEXT
420     RANP=RANP+RANP
430   NEXT
440   OUT REX5055+DRH,0
450 WEND
460 END
```

PHI-8 との接続

フォトカプラ入力ユニット(PHI-8)を REX-5055 と接続し、ON-OFF の状態を読み込みます。

REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	PHI-8 外部入出力コネクタ信号名
+5V(1)	↔	+5V
GND (19)	↔	GND
PIO7(9)	←	DO-1
PIO6(21)	←	DO-2
PIO5(10)	←	DO-3
PIO4(22)	←	DO-4
PIO3(11)	←	DO-5
PIO2(23)	←	DO-6
PIO1(12)	←	DO-7
PIO0(24)	←	DO-8

田 C 言語プログラム例

添付ディスク中の [PHI8.C] がソースプログラムです。
このプログラムは、各種入力ユニット (PHI-8, SSR-82, SSR-84) 等でも同様です。

```

void main( void )
{
/* --- 途中省略 --- */

MAIN_PROG:
  CardInit( IOBase );
  while( !kbhit() ) {
    for( port = 0; port <= 1; port++ ) {
      data = (WORD)~_inp( IOBase + port + 2 );          /* Data read */
      printf( " Address %04X -- Data %02X ", IOBase + port + 2, data );
      delay( 40000 );                                  /* delay */
    }
    printf( "%n" );
  }
}

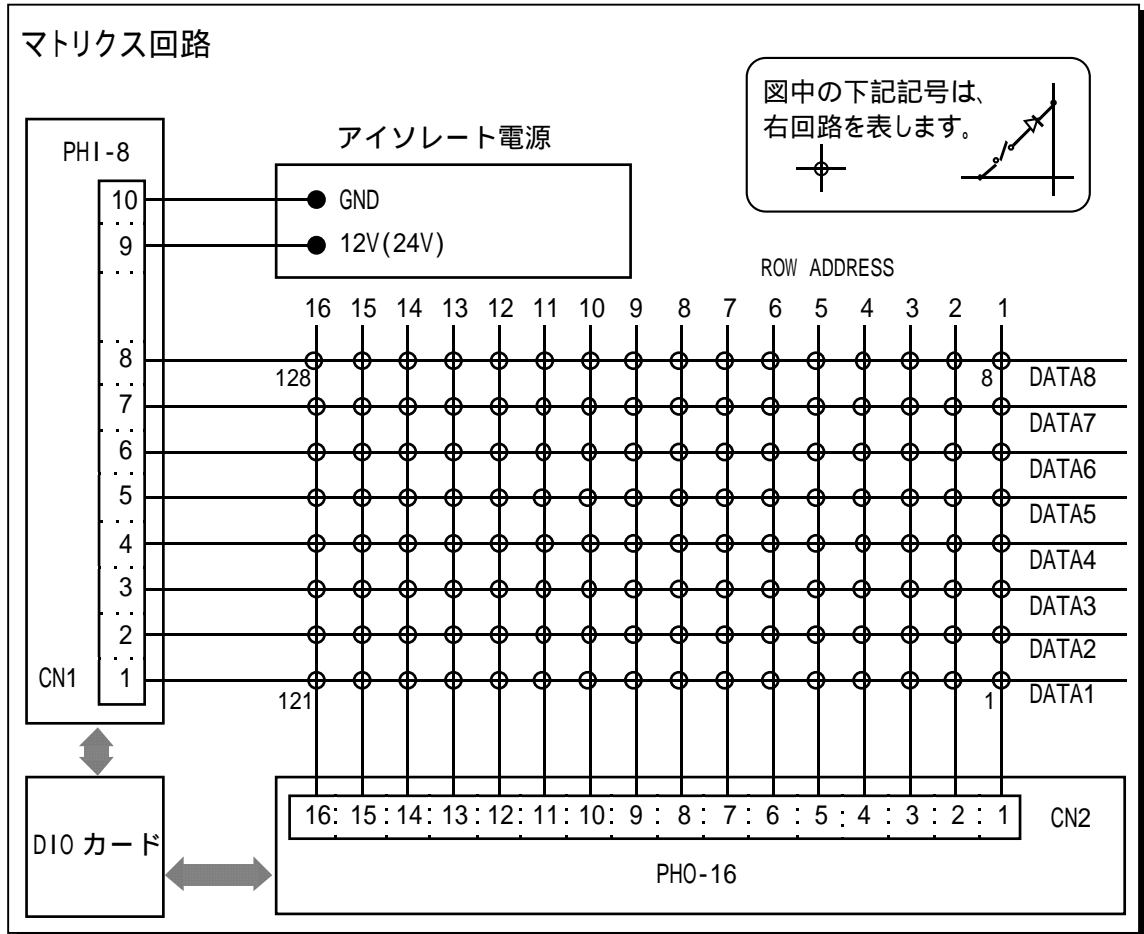
void CardInit( WORD Adrs )
{
  outp( Adrs + CTRG, 0 );          /* コントロールレジスタ Base + 0 */
                                  /*  IMD:0  BSEL3:0 BSEL2:0 BSEL1:0 */
                                  /*  BSEL0:0  DIRH :0 DIRL :0 */
  outp( Adrs + INTEBL , 0 );      /* 割り込みディセーブル */
}

```

マトリクス入力回路

フォトカプラ入力ユニット(PHI-8)とフォトカプラ出力ユニット(PHO-16)で、マトリクスを構成します。マトリクス上の、合計 128 点の接点入力の状態をスキャンして調べます。

機器の稼働状態のモニタや、ドアの開閉状態のモニタ等に、幅広く用いることができます。



REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	PHO-16 外部入出力コネクタ信号名
+5V (7)	↔	+5V (1)
GND (19)	↔	GND (12)
PI07 (9)	→	NC (4)
PI06 (21)	→	NC (5)
PI05 (10)	→	NC (6)
PI04 (22)	→	NC (7)
PI03 (11)	→	D (8)
PI02 (23)	→	C (9)
PI01 (12)	→	B (10)
PI00 (24)	→	A (11)

REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	PHI-8 外部入出力コネクタ信号名
+ 5V (1)	↔	+ 5V (1)
GND (13)	↔	GND (12)
PIO15(3)	←	DO - 8 (11)
PIO14(15)	←	DO - 7 (10)
PIO13(4)	←	DO - 6 (9)
PIO12(16)	←	DO - 5 (8)
PIO11(5)	←	DO - 4 (7)
PIO10(17)	←	DO - 3 (6)
PIO9 (6)	←	DO - 2 (5)
PIO8 (18)	←	DO - 1 (4)

(注)PHI-8/PHO-16 のネジターミナルから、接点までのケーブルは約 1000m までの延長が可能です。アイソレート電源の GND は、パソコン本体の GND と接続しないでください。

田 C 言語プログラム例

添付ディスク中の[MATRIX.C]がソースプログラムです。

```

void main( void )
{
    CardInit( IOBase );
    printf( "%x0c" );
    while( !kbhit() ) {
        printf( "%x0b" );
        for( rowad = 0; rowad <= 15; rowad++ ){
            outp( IOBase + DRL, rowad );          /* row data output */
            swdt = (WORD)_inp( IOBase + DRH );    /* swich data read */
            for( no = 0; no <= 7; no++ ) {
                printf( "%3d-", rowad * 8 + no + 1 );
                if( swdt & ( 1 << no ) )
                    printf( "OFF  " );
                else
                    printf( "ON   " );
            }
        }
    }
}

void CardInit( WORD Adrs )
{
    outp ( Adrs + CTRG, 1 );          /* low byte = output high byte = input mode */
    outp ( Adrs + INTEBL, 0 );       /* INT Disable */
}

```

アスキーキャラクタの読み込み

外部からのデータストローブ入力を PIO7 で受け、PIO8 からアクノリッジを出力し、ポート下位7ビットをデータ入力として使用します。

REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	外部機器
GND (19)	←→	GND
PIO8 (18)	→	アクノリッジ
PIO7 (9)	←	ストローブ
PIO6 (21)	←	b6 データ
PIO5 (10)	←	b5 データ
PIO4 (22)	←	b4 データ
PIO3 (11)	←	b3 データ
PIO2 (23)	←	b2 データ
PIO1 (12)	←	b1 データ
PIO0 (24)	←	b0 データ

田 C 言語プログラム例

添付ディスク中の[ASCIN.C]がソースプログラムです。

```

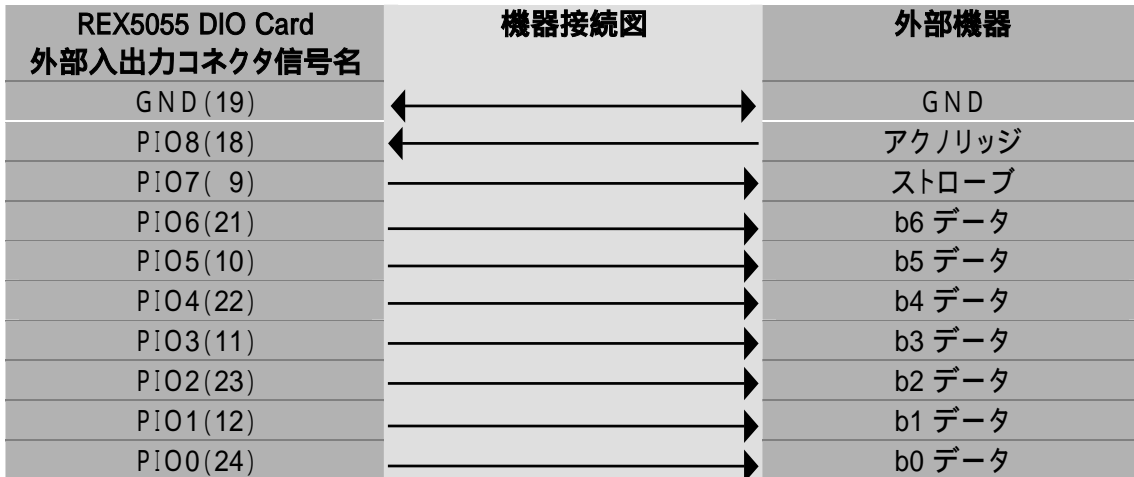
void main( void )
{
    CardInit( IOBase );
    while( !kbhit() ){
        if( inp( IOBase + DRL ) >> 7 == 1 )      /* strobe in ? */
            continue;
        c = ~(inp( IOBase + DRL ));              /* data read */
        outp ( IOBase + DRH , 0x01 );            /* ack out */
        putchar( c & 0x7f );
        for( i=0; i<1000000; i++ )                /* すぐに続けるとデータを */
            ;                                       /* とり続けてしまうため */
                                                /* Strobe off を待つ時間*/
    }
}

void CardInit( WORD Adrs)
{
    outp( Adrs + CTRG , 0x02 ); /* Low Byte input mode / High Byte output mode */
    outp( Adrs + INTEBL ,0 ); /* INT Disable */
}

```

アスキーキャラクタの出力

PIO7 を外部へのストロープ出力、PIO8 をアクノリッジ入力、ポート B の下位7ビットをデータ出力として使用します。



田 C 言語プログラム例

添付ディスク中の[ASCOUT.C]がソースプログラムです。

```

void main( void )
{
    CardInit( IOBase );
    while( !kbhit() ){
        s = string;
        while( *s ){
            outp( IOBase + DRL, *s );           /* データ出力 */
            outp( IOBase + DRL, *s | 0x80 );    /* ストロープ出力 */
            for( i=0L; i<100000L; i++ )
                ;
            while( !( (~inp( IOBase+DRH )) & 1 ) ); /* Ack in ? */
            putchar( *s++ );
            outp( IOBase + DRL, 0 );           /* ストロープを Off */
        }
    }
}

void CardInit( WORD Adrs)
{
    outp ( Adrs + CTRG , 0x01 ); /* Low Byte output mode / High Byte input mode */
    outp ( Adrs + INTEBL , 0 ); /* INT Disable */
}

```



```
/* 割り込みによるデータ入力を開始する */
StartIrq();
IrqStatus = IRQMODE_RUN;
printf("Start Interrupt Test\n");

/* 入力完了まで待つ */
switch( IrqLoopProc() )
{
case 1:      printf("ESC Key Stop\n");      break;
case 0:      printf("Normally Stop\n");     break;
}
}

void StartIrq( void )
{
/* コントロールレジスタの設定 */
/* PIO 0-7 ->入力; PIO 8-15 ->出力; 割り込みソース->PI07 ;立上りエッジで割り込み */
SetControlReg( 0, 1, 7, 1 );
/* 割り込みハンドラの登録 */
IrqSetup();
/* 割り込み許可ビットをONにする */
IrqEnable( CLEAR_R_LOW );
}

BOOL IrqSetup( void )
{
WORD ValIMR,MaskBit;

/* 旧割り込み処理ハンドラのアドレスを待避 */
OldFunc = _dos_getvect( drv_irq_vect_table[tbl_int_no] );
/* 新割り込み処理ハンドラのアドレスを設定 */
_dos_setvect( drv_irq_vect_table[tbl_int_no], HandlerFunc );

/* PICにコマンドを書き込む */
if ( tbl_int_no > 1 ) {
/* INT9以上ならスレーブPIC */
ValIMR = _inp( SlvIMRAddr ); /* Read IMR of slave PIC */
MaskBit = ( WORD )(0xff^drv_irq_mask_table[ tbl_int_no ]);
ValIMR = ValIMR & MaskBit;
_outp( SlvIMRAddr, (BYTE)ValIMR ); /* Write IMR of slave PIC */
} else {
/* マスタPIC */
ValIMR = _inp( MstIMRAddr ); /* Read IMR of master PIC */
MaskBit = (WORD)(0xff^drv_irq_mask_table[ tbl_int_no ]);
ValIMR = ValIMR & MaskBit;
_outp( MstIMRAddr, (BYTE)ValIMR ); /* Write IMR of master PIC */
}
return 0;
}

/* --- 次ページに続く --- */
```



```
void _interrupt _far HandlerFunc( void )
{
    _disable();
    /* データの転送 */
    ReadVal = _inp( IOBase + 0x02 );
    IrqStat = _inp( IOBase + 0x01 );
    wCount++;

    /* EOIを発行 */
    SendEOI();

    /* 割り込み処理中に次の割り込みがある場合、ここでは故意に中止します。 */
    if ( ( _inp( IOBase + 0x01 ) & 0x80 ) == 0x80 ){
        IrqStatus = IRQMODE_STOP;
    }
    _enable();
}

BOOL IrqLoopProc( void )
{
    /* 割込処理ループ */
    for( ;; ){
        if( KeyIn() == KY_ESC ) {
            StopIrq();
            return 1;
        }

        /* ステータスチェック */
        switch( IrqStatus )
        {
            case IRQMODE_STOP:
                /* 割り込み処理中止 */
                StopIrq();
                return 0;
            case IRQMODE_RUN:
                if( wCount != wCountOld ){
                    wCountOld = wCount;
                    printf("%lu[%x:%x] %t",wCount, ReadVal, IrqStat);
                }
                break;
        }
    }
}
```

(4-3) Windows3.1 16Bit ライブラリ解説

(4-3-1) Windows3.1 アプリケーション

Visual C による作成

Windows3.1 環境で Microsoft Visual C でアプリケーションを作成する場合、Visual C では、DIO カードへの入出力を行うために必要な_inp()/_outp()等のポート入出力関数はサポートされていません。本製品添付の DLL は FD:¥LIBRARY¥WIN31¥DLL16 に収録されており、DIO カードに入出力を行うための関数とカードサービスに関する関数を提供しています。

Visual C/C++でアプリケーションを作成する場合は、DLL からイクスポートされた関数を DEF ファイルでインポート宣言すると同時に、下記のように作成したプログラムのヘッダー部でインポート関数の宣言を行ってください。

DLL は、プログラム実行時のカレントディレクトリにコピーしてください。カレントディレクトリから DLL のロードができない場合は、WINDOWS¥SYSTEM ディレクトリにコピーしてください。

DLL ライブラリ名 : DIOLIB.DLL (16Bit Version)

DEF ファイルの作成例 (Visual C)

```

NAME                REX5055
DESCRIPTION          'REX5055 DIO PC Card Sample Program (C)RATOC System,Inc.'
EXETYPE              WINDOWS
STUB                 'WINSTUB.EXE'
CODE                 PRELOAD MOVEABLE DISCARDABLE
DATA                 PRELOAD MOVEABLE MULTIPLE
HEAPSIZE             1024
IMPORTS
    DIOLIB.OutPort
    DIOLIB.wOutPort
    DIOLIB.InPort
    DIOLIB.wInPort
    DIOLIB.CheckCSRegistration
    DIOLIB.GetCSConfigInf

```

アプリケーションでのインポート宣言 (Visual C)

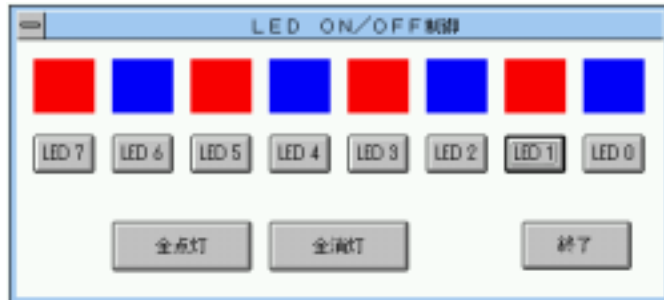
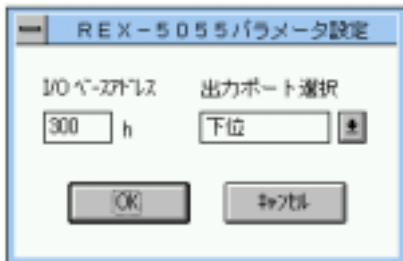
```

WORD _export WINAPI OutPort( WORD, WORD );
WORD _export WINAPI wOutPort( WORD, WORD );
WORD _export WINAPI InPort( WORD );
WORD _export WINAPI wInPort( WORD );
BOOL _export WINAPI CheckCSRegistration( void );
BOOL _export WINAPI GetCSConfigInfo( WORD, WORD *, WORD * );

```

■ サンプルプログラム実行例

フォトブラ出力ユニットPHO-8と接続し、LEDをオンオフ制御を行うサンプルプログラムになります。実行時の画面は下記のようになります。



■ Windows3.1 での割り込み処理

添付ディスクのFD:¥LIBRARY¥WIN31¥HANDLERの中に、MS-DOSの割り込みサンプルプログラムをWindows3.1上に移植したサンプルプログラムWINIRQ.Cが収録されています。WINIRQ.EXEは、FD:¥LIBRARY¥WIN31¥HANDLER¥DLLの中のIRQLIB.DLLをロードします。割り込みハンドラ内の処理は、DLLライブラリの中に記述してあります。必要な割り込み処理を記述して再構築してください。

DIOLIB.DLLとWINIRQ.DLLとは、下記の内容が相違しています。

DEFファイル

<pre> CODE PRELOAD FIXED DATA PRELOAD SINGLE HEAPSIZE 512 EXPORTS WEP SetCardResource StopIrq StartIrq SetISRWindow </pre>	<p>← コードの移動・破棄は不可を指定</p> <p>← 割り込み用関数をイクスポート</p>
---	--

WinGetVect/WinSetVect 関数をサポート

WinGetVect(_dos_getvect 相当の関数)

書式 void (_cdecl _interrupt _far * _cdecl WinGetVect(BYTE VectNo))()

機能 プロテクトモード割り込みベクタの取得

引数 VectNo: 割り込みベクタ番号

戻値 割り込みベクタ VectNo の far ポインタ

WinSetVect(_dos_setvect 相当の関数)

書式 BOOL _cdecl WinSetVect(BYTE VectNo, void (_cdecl _interrupt _far *IsrAdrs)())

機能 プロテクトモードの割込ベクタ設定

引数 BYTE VectNo : ベクター番号

void (_cdecl _interrupt _far *IsrAdrs)(): 割込ハンドラのエントリポイント

戻値 0:成功 -1:失敗

Visual BASIC による作成

Visual BASIC では、DIO カードへの入出力を行うためのポート入出力命令 IN/OUT はサポートされていません。Visual BASIC でプログラムを作成する場合も Visual C と同じように DLL ライブラリでサポートしているポート入出力命令を呼び出します。

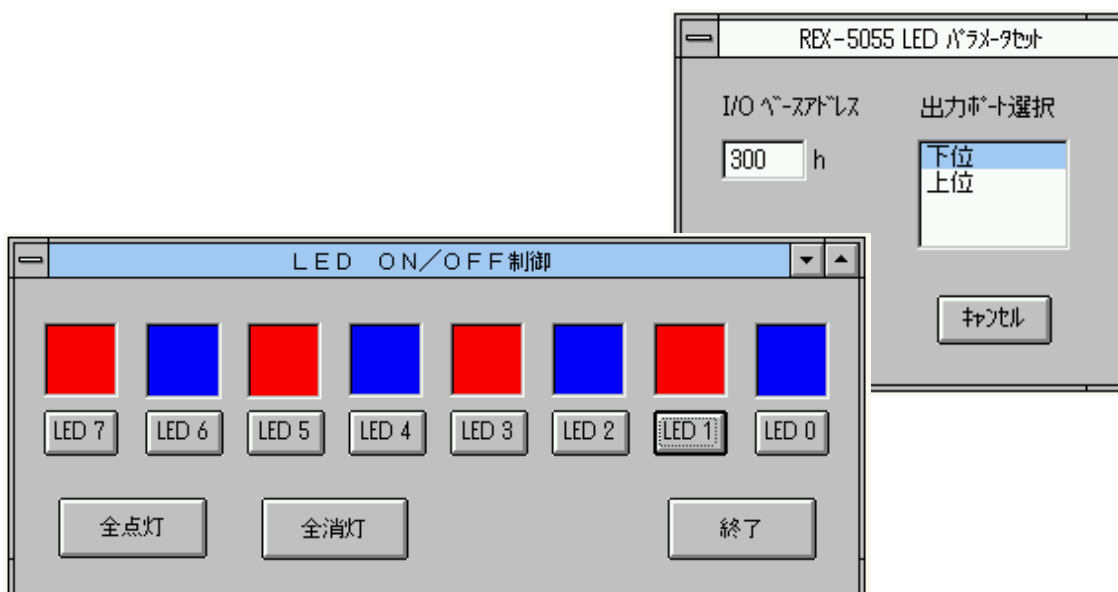
Visual BASIC から DLL ライブラリの API を呼び出すためには、コードモジュールファイルの中で DLL 外部関数の参照宣言を行う必要があります。DLL 関数の参照宣言例を下記に示します。詳細は、添付ディスクの FD:¥LIBRARY¥WIN31¥VB4 に収録されています。尚、Visual BASIC は Version4.0 16Bit Version を使用しています。

■モジュール定義ファイル Declare 宣言例 (Visual BASIC)

```
'DLL 外部関数の参照宣言
Declare Function OutPort Lib "diolib.dll" (ByVal IOAddr As Integer,
                                           ByVal OutVal As Integer) As Integer
Declare Function wOutPort Lib "diolib.dll" (ByVal IOAddr As Integer,
                                           ByVal OutVal As Integer) As Integer
Declare Function InPort Lib "diolib.dll" (ByVal IOAddr As Integer) As Integer
Declare Function wInPort Lib "diolib.dll" (ByVal IOAddr As Integer) As Integer
Declare Function GetCSConfigInfo Lib "diolib.dll" (ByVal Socket As Integer,
                                                  IpCsvIOAddr As Any, IpCsvIRQNo As Any) As Integer
Declare Function CheckCSRegistration Lib "diolib.dll" () As Integer
```

■ サンプルプログラム実行例

フォトプラ出力ユニット PHO-8 と接続し、LED をオンオフ制御を行うサンプルプログラムになります。実行時の画面は下記のようになります。



(4-3-2) 16Bit DLL 関数仕様

OutPort **ポートに1バイトを出力**

書 式 WORD_export WINAPI **OutPort**(WORD IOAddr, WORD OutVal)

機 能 1バイトをポートに出力

引 数 WORD IOAddr : 出力する I/O ポートアドレス
WORD OutVal : バイト出力値(上位バイトは無視されます)

戻 値 バイト出力値をそのまま返し、エラー値はなし

wOutPort **ポートに1ワードを出力**

書 式 WORD_export WINAPI **wOutPort**(WORD IOAddr, WORD OutVal)

機 能 1ワードをポートに出力

引 数 WORD IOAddr : 出力する I/O ポートアドレス
WORD OutVal : ワード出力値

戻 値 ワード出力値をそのまま返し、エラー値はなし

InPort **ポートから1バイト入力**

書 式 WORD_export WINAPI **InPort** (WORD IOAddr)

機 能 ポートから1バイト読み込む

引 数 WORD IOAddr : 出力する I/O ポートアドレス

戻 値 ポートから読み込んだバイトデータ

wInPort **ポートから1ワード入力**

書 式 WORD_export WINAPI **wInPort** (WORD IOAddr)

機 能 ポートから1ワード読み込む

引 数 WORD IOAddr : 出力する I/O ポートアドレス

戻 値 ポートから読み込んだワードデータ

CheckCSRegistration**カードサービス常駐チェック**

書式 `BOOL _export WINAPI CheckCSRegistration (void)`

機能 カードサービスが常駐しているか調べます

引数 なし

戻値 カードサービス常駐していれば 0 以外の値を返します。

GetCSConfigInfo**リソース情報の取得**

書式 `BOOL _export WINAPI GetCSConfigInfo (WORD Slot, WORD *pAdrs, WORD *pIRQ)`

機能 カードサービスをコールしてカードに割り当てられている I/O アドレス・割り込み番号を取得する

引数 `WORD Slot` : カードが挿入されているスロットの番号
`WORD *pAdrs` : I/O アドレス格納先を示すポインター
`WORD *pIRQ` : 割り込み番号格納先を示すポインター

戻値 正常に取得できた場合 0 を返します。その他はエラー。

◆注意◆

- 1.アプリケーション実行時、DLL ファイルは `WINDOWS¥SYSTEM` ディレクトリに置いてください。通常アプリケーション実行ディレクトリに置いてもロードされますが、ディレクトリの階層が深いとロードされないことがあります。
- 2.Windows3.1 ではハードウェアを直接操作する I/O は基本的には禁止されています。従って、Microsoft Visual C/Visual BASIC についても Windows 上では、`_inp()/_outp()` 等の API をサポートされていません(実際はインプリメントされている)。しかしながら、Windows システム動作に関係しない外部機器との通信機器との通信制御の用途に限ってはハードウェアを直接操作しても問題無いと考え、DLL ライブラリの中で `_inp()/_outp()` 相当の API をサポートしました。

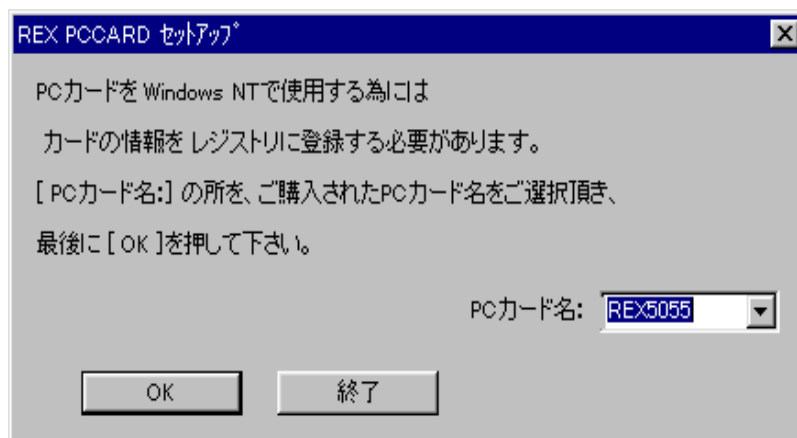
第5章 WindowsNT 解説

(5-1) インストール

WindowsNT 上で REX5055 を使用した計測システムを接続する場合は、本製品添付のセットアッププログラム (INISSETUP.EXE) を実行します。

セットアッププログラムの実行

本製品に含まれるセットアッププログラム (INISSETUP.EXE) を実行すると、下記のような画面が表示されます。使用されるカードを選択し [OK] を押してください。



実行画面

このセットアッププログラムにより指定されたカードのドライバの登録と、ドライバ、DLL (ダイナミックライブラリ) のコピーが行われます。

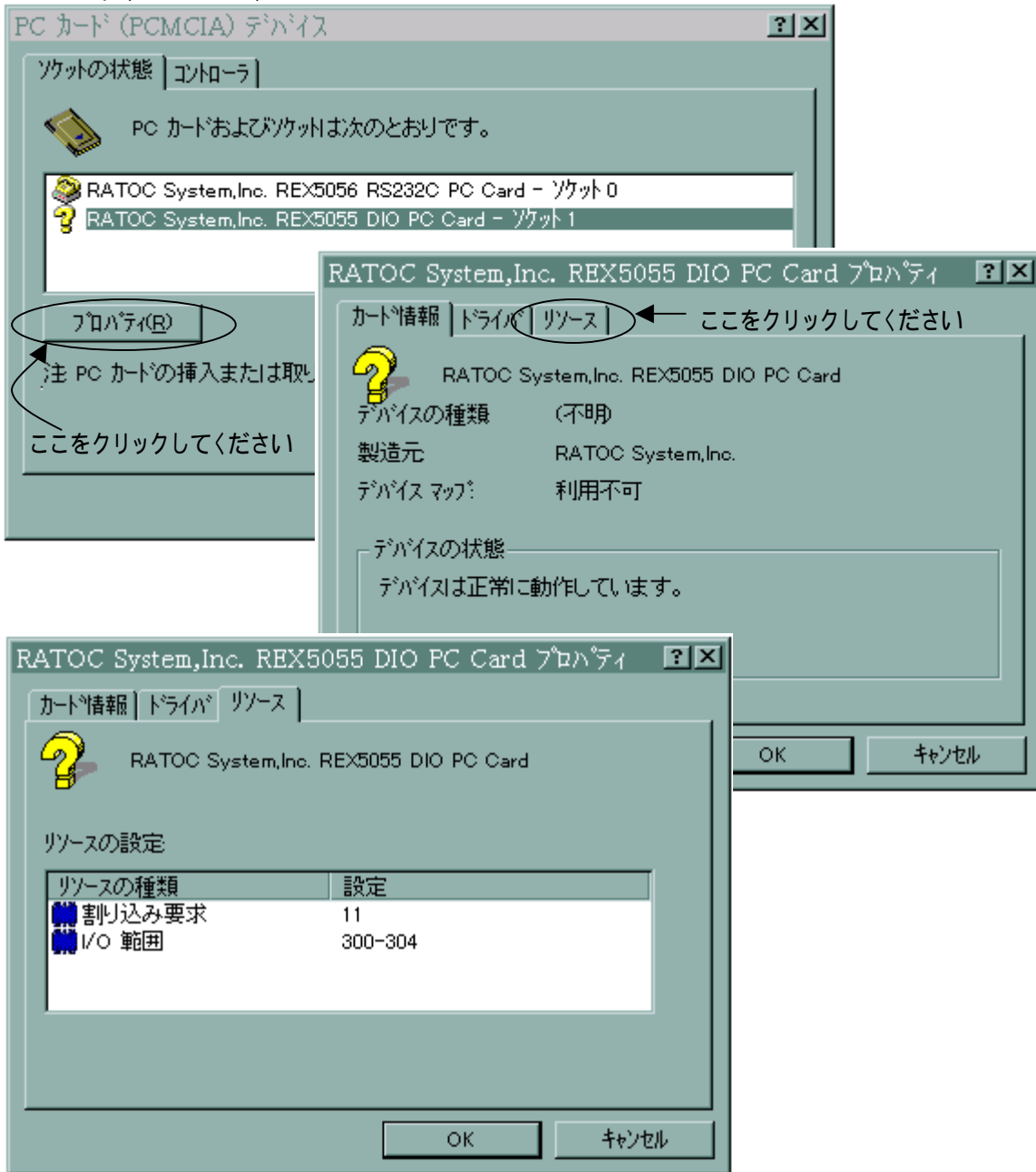
ドライバは ¥WinNT¥System32¥Drivers

DLL は¥WinNT¥System32 へコピーされます。

以上で設定完了です。WindowsNT を再起動し、REX5055 が使用可能な状態になったかどうかを確認します。

再起動後の確認

再起動後リソースの取得が正常に行われていればセットアップは正常に行われています。(下図参照)



これらの画面はコントロールパネルの[PC カード]を選択することにより画面に表示されます。

リソースが取得できていれば、セットアップ完了です。また、コントロールパネルの[デバイス]を起動することにより、Windows NT で現在使用できるドライバの一覧が表示されます。そこに REX-5055 のドライバも再起動により表示されます。ドライバのロード、アンロード等の設定を行う場合はここで設定を行います。(特に指定のない場合は何も行う必要はありません)

(5-2) Visual C によるアプリケーション開発

Visual C 4.0 以上を使って REX-5055 カードを制御するアプリケーションを開発する場合は、本製品に添付されている 32Bit 版 DLL の関数をコールする必要があります。

Visual C で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、下記 3 点の作業を行ってください。

1. セットアッププログラムを実行しリソースのレジストリ登録及びドライバのインストールを行う(「(5-1)インストール参照」)
2. アプリケーションプログラムに REX5055.H ファイルをインクルードする。
3. アプリケーションプログラムのプロジェクトファイルに 5055LIB.LIB を追加する。

アプリケーション作成上のアドバイス

I/O ポートへの入出力

WindowsNT の保護機能によりアプリケーションレベルでは I/O への入出力は行えません。従って、PC カードへの I/O 入出力は DLL でサポートされている `OutPort()`・`wOutPort()`・`InPort()`・`wInPort()`を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、
(1) 割り込みハンドラからユーザ定義メッセージにより割り込み通知を受け取る方法
(2) 割り込みハンドラ内で全ての処理を行う方法
があります。高速な処理が要求されるような場合は、(2)の方法で行う必要があります。
サンプルプログラムは高速割り込み処理の例になりますが、割り込み処理内で要求される内容はユーザー毎で変わります。サポートセンターでは高速割り込み処理のカスタマイズに関するご相談も承っています。

(5-2-1) DLL 関数仕様

FastIntStart**ドライバ内部処理による高速割り込み開始**

書式 void *FastIntStart*(HWND hWnd, UCHAR InPortReg, UCHAR OutPortReg, ULONG Count, USHORT Flg)

機能 割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。割り込みハンドラ内部では、入力方向に設定されたデータレジスタの値をリードし、出力方向に設定されたデータレジスタにリードした値を出力します。割り込み終了回数Countで指定した回数の割り込み処理が終了すると、呼び出し側プログラムに終了メッセージをポストします。

引数	HWND	hWnd	➤ ユーザアプリケーションのウィンドウハンドル
	UCHAR	InPortReg	➤ 入力データレジスタ(オフセット) 2:PA7-PA0 3:PA15-PA8
	UCHAR	OutPortReg	➤ 出力データレジスタ(オフセット) 2:PA7-PA0 3:PA15-PA8
	ULONG	Count	➤ 割り込み終了回数
	USHORT	Flg	➤ 割り込み設定 0: InPortReg読み込みのみ (割り込みフラグクリアのみ) 1: InPortRegをOutPortRegに出力

戻値 なし

MsgIntStart**ユーザ定義メッセージ同期割り込み開始**

書式 void *MsgIntStart* (HWND hWnd, WORD Count, WORD Flg)

機能 ユーザー定義メッセージ版の割り込み処理を開始します。割り込み発生に同期したユーザー定義メッセージをユーザー側アプリケーションに毎回ポストします。これによりユーザーアプリケーション側で割り込み発生に同期した処理を行うことが出来ます。

引数	HWND	hWnd	➤ ユーザアプリケーションのウィンドウハンドル
	WORD	Count	➤ 割り込み終了回数
	WORD	Flg	➤ 割り込み設定 1以外: 割り込みハンドラ内でメッセージ通知のみを行う 1: 割り込みハンドラ内でデータポート(16ビット)をリードしメッセージの第一引数(wParam)に出力

戻値 Flg が 1 の時のみ wParam にデータレジスタの内容を返します。

アプリケーション作成上のアドバイス

WindowsNT ではシステムのオーバーヘッド等により *MsgIntStart* を使用した場合、50Hz、*FastIntStart* を使用した場合、100Hz よりも低速の割り込み信号を想定しております。(PentiumPro200MHz 32MB の場合)

但し、割り込み処理は、処理内容、機種等に依存します。

IntSet**コントロールレジスタ、割込みマスクレジスタの設定**

書式 void *IntSet* (UCHAR CtrlReg, UCHAR IntMask)

機能 コントロールレジスタ、IrEblビットを除く割込みマスクレジスタの設定を行います。
ここで設定される内容はドライバ内で保持され割り込み処理ルーチン内で使用されます。
(コントロールレジスタの制御でOutPort関数を使用しないでください)

引数 UCHAR CtrlReg ➤ コントロールレジスタ値
 UCHAR IntMask ➤ 割込みマスクレジスタ値

戻値 なし

IODrvResource**I/O ベースアドレスの取得**

書式 long *IODrvResource* (void)

機能 レジストリ内に登録されているI/Oベースアドレスの取得を行います。
(但し、DLLでドライバーのオープンに失敗した場合正しい取得を行えません。)

引数 なし

戻値 I/O ベースアドレスを返します。

IRQDrvResource**IRQ 番号の取得**

書式 long *IRQDrvResource* (void)

機能 レジストリ内に登録されているIRQ番号の取得を行います。
(但し、DLLでドライバーのオープンに失敗した場合正しい取得を行えません)

引数 なし

戻値 IRQ 番号を返します。

OutPort**1バイトをポート出力**

書式 WORD *OutPort*(WORD IOAddr, WORD OutVal)

機能 1バイトをポートに出力

引数 WORD IOAddr ➤ ポート番号
 WORD OutVal ➤ バイト出力値(上位バイトは無視)

戻値 バイト出力値をそのまま返し、エラー値はなし。

wOutPort**1ワードをポート出力**

書式 WORD *wOutPort*(WORD IOAddr, WORD OutVal)

機能 1ワードをポートに出力

引数 WORD IOAddr ➤ ポート番号
 WORD OutVal ➤ ワード出力値

戻値 ワード出力値をそのまま返し、エラー値はなし。

InPort**1バイトをポート入力**

書式 WORD *InPort*(WORD IOAddr)

機能 ポートから1バイト読み込む

引数 WORD IOAddr ➤ ポート番号

戻値 ポートから読み込んだバイトデータを返します。

wInPort

1ワードをポート入力

書式 WORD *wInPort*(WORD IOAddr)

機能 ポートから1ワード読み込む

引数 WORD IOAddr > ポート番号

戻値 ポートから読み込んだワードデータを返します。

TerminateInterrupt

メッセージ割り込み処理の中断

書式 void *TerminateInterrupt* (void)

機能 割り込み処理の中断

引数 なし

戻値 なし

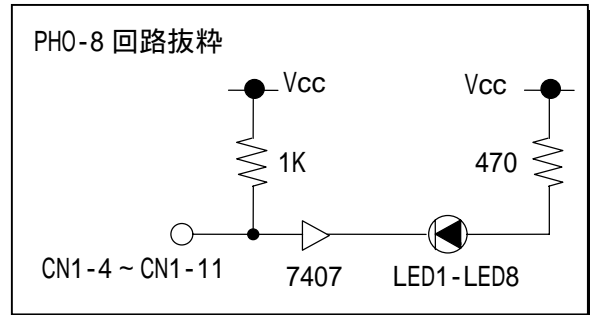
アプリケーション作成上のアドバイス

MsgInterrupt は中断処理を実行することができますが、*FastIntStart* は入力した回数の割り込みが起るまで処理をユーザー側には返すことができません。
FastIntStart を使用される際にはご注意ください。

(5-2-2) Visual C サンプルプログラム

アイソレーションユニット PH0-8 または PHI-8 を使った、サンプルプログラムについて説明します。

PH0-8 は右図に示す LED 点灯回路とアイソレーションを行って外部へ出力する回路を 8 ブロック実装しており、下表のように接続することによりオンオフ制御のシミュレーションを行うことができます。PHI-8 は PH0-8 とは反対に外部オンオフ信号をアイソレーションして入力するためのボードです。



REX5055 DIO Card 外部入出力コネクタ信号名	機器接続図	PHI-8 外部入出力コネクタ信号名
+5V	↔	+5V
GND	↔	GND
PA7	→	DO-1
PA6	→	DO-2
PA5	→	DO-3
PA4	→	DO-4
PA3	→	DO-5
PA2	→	DO-6
PA1	→	DO-7
PA0	→	DO-8

サンプルプログラムは、下記のモジュールで構成されています。

PH0-8 LED オンオフ制御プログラム

PHI-8 入力プログラム (割り込みを使用しない単純に入力を行うサンプルプログラム)

ユーザ定義メッセージによる割り込み制御

割り込みハンドラ内部で全ての入出力制御を行う高速割り込み制御プログラム

次ページ以降では、 ， のサンプルプログラムについて解説を行います。

ユーザ定義メッセージによる割り込みプログラム

割り込みソース入力ポートを指定して MsgIntStart() を実行することにより、割り込み発生に同期したユーザ定義メッセージ "WM_USER + 5055" が割り込みハンドラから送られてきます。このとき、MsgIntStart の引数である Flg に 1 を指定していれば、wParam には両方のデータポートの内容がセットされています。

```
LRESULT CALLBACK IntProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam ) {
    switch( message ) {
        case WM_USER + 5510:           // 割り込み通知メッセージ
            data = (BYTE)InPort(2);   // 割り込みフラグクリア
            break;
        case WM_COMMAND:
            switch( wParam ) {
                case ID_INT_START:     // 割り込み開始要因
                    IntSet( 0x00, 0x01 ); // 割り込みソース入力ポート設定
                    MsgIntStart( hDlg, 100, 0); // 割り込み開始
            }
            break;
        case IDEND:
        case IDCANCEL:
            return TRUE;
        default:
            break;
    }
}
```

実行画面

割り込み処理テスト

割り込み設定

割り込み終了回数:

割り込みソースビット:

割り込みフラグ:

割り込みトリガ:

リソース状態

ベースアドレス:

IRQ番号:

メッセージ受信回数:

ポート入出力設定

ポート上位:

ポート下位:

モニタリングポート

R8 R7 R6 R5 R4 R3 R2 R1

高速割り込み制御プログラム

割り込みハンドラ内で全ての入出力制御を行うことにより高速割り込み処理を行うことができます。但し、割り込みハンドラ内で全ての入出力制御を行う場合は、ユーザ毎の仕様に基づいてデバイスドライバ内の割り込みハンドラをカスタマイズする必要があります。弊社では、有償によるカスタマイズ作業のご相談も承っております。

サンプルプログラムの割り込みハンドラは、入力方向に設定されたデータレジスタの値をリードし、出力方向に設定されたデータレジスタにリードした値を出力します。指定回数の割り込み処理が終了すると、呼び出し側プログラムにユーザ定義メッセージをポストします。

入出力データレジスタと割り込み終了回数を指定して、FastIntStart ()を実行することにより、割り込みハンドラ内部で全ての処理を行う高速版の割り込み処理を開始します。

実行画面

高速割り込みテスト(PHI-8 PHO-8)

割り込み設定

割り込み終了回数:

割り込みソースビット:

割り込みフラグ:

割り込みトリガ:

リソース情報

I/Oアドレス:

IRQ番号:

ポート入出力設定

ポート上位:

ポート下位:

出力ポート:

入力ポート:

開始

(5-3) Visual BASIC によるアプリケーション開発

Visual BASIC 4.0 以上を使って REX-5055 を制御するアプリケーションを開発する場合は、本製品に添付されている 32Bit 版 DLL の関数をコールする必要があります。Visual Basic で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、

1. セットアッププログラムを実行しリソース設定、ドライバのインストールを行う。
2. モジュールファイルで DLL 関数の参照宣言を行う。

(DLL の関数仕様については「(5-2-1)DLL 関数仕様」を参照してください)を行う必要があります。

また、割り込み制御を行う場合は割り込みハンドラから送られてくるユーザ定義メッセージを Visual BASIC 側のアプリケーションで受け取るために、本製品に添付されている OLE カスタムコントロール MBOX を使用します。MBOX の使用方法については、サンプルプログラム解説を参照してください。

アプリケーション作成上のアドバイス

PC カードへの入出力

WindowsNT の保護機能によりアプリケーションレベルでは I/O への入出力は行えません。従って、PC カードへの I/O 入出力は DLL でサポートされている `OutPort()`・`wOutPort()`・`InPort()`・`wInPort()`を使って行います。

割り込み制御

DLL では割り込みサービスを提供しています。割り込みサービスには、
(1)割り込みハンドラからユーザ定義メッセージにより割り込み通知を受け取る方法
(2)割り込みハンドラ内で全ての処理を行う方法
があります。高速な処理が要求されるような場合は、(2)の方法で行う必要があります。詳細は、各サンプルプログラムを参照してください。

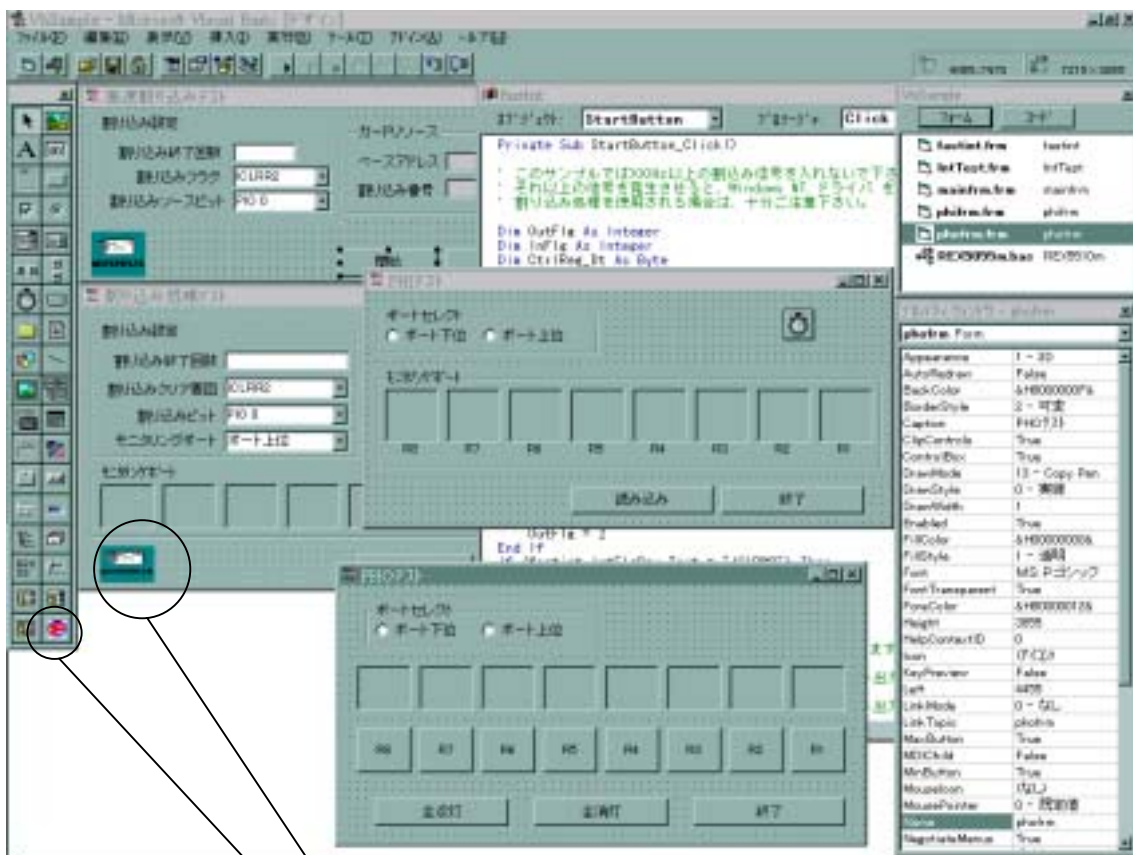
(5-3-1) Visual BASIC サンプルプログラム

割り込みを使わないで単純に入出力を行うプログラム例	PH0-8 LED オンオフ制御プログラム PHI-8 入力プログラム
割り込みを使ったプログラム例	ユーザ定義メッセージによる割り込み制御 割り込みハンドラ内部で全ての入出力制御を行う高速割り込み制御

動作環境等については、Visual Cのサンプルプログラム解説を参照してください。ここでは、ユーザ定義メッセージによる割り込み制御について解説し、
、
、
については省略致します。

ユーザ定義メッセージによる割り込み制御サンプルプログラムの解説

下記画面は、VB32でのデザイン作成時の画面です。割り込み発生に同期させユーザ定義メッセージをVBで作成したプログラムで受け取るためには、本製品添付のOLEカスタムコントロール(OCX)MBOXを使用しなければVBではユーザ定義メッセージを受け取ることが出来ません。以下、カスタムコントロールのインストール手順とMBOXサンプルプログラムの内容について説明します。



本製品に添付されているOLEカスタムコントロール「MBOX」
カスタムコントロール「MBOX OLE Control module」を追加

Step.1 本製品添付ソフトのコピー

OLE カスタムコントロール:MBOX.OCX を添付のフロッピーディスクからコピーします。WindowsNT システムディレクトリ名を“WinNT¥System32”とします。

```
>COPY “コピー元ドライブ名” MBOX.OCX “コピー先ドライブ名” :¥WinNT¥System32
```

Step.2 OCX のレジストリー登録 (割り込みサービス使用時必須)

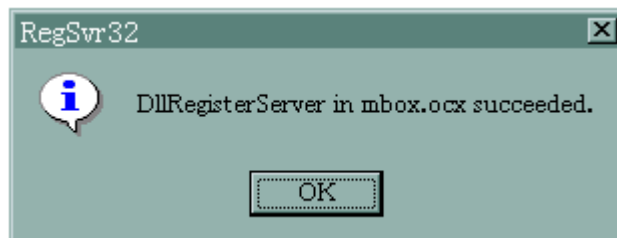
本製品添付の OCX “MBOX.OCX”を VB で使用するためには、VB の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリ登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、WindowsNT の DOS BOX から実行します。尚、“REGSVR32.EXE”は VB の CD-ROM の “TOOLS¥PSS”に添付されています。

OCX をレジストリー登録するときは、下記構文で実行します。

```
>REGSVR32 “ドライブ名”:¥WinNT¥System32¥Mbox.ocx
```

OCX をレジストリー登録から削除するときは、“/U”を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:¥ WinNT¥System32¥Mbox.ocx
```



登録成功メッセージ



登録削除成功メッセージ

Step.3 5055lib 関数の Declare 宣言

次に、VBプログラムの作成に入ります。VBデザインメニューから新規プロジェクトを作成し、「ファイル」の「挿入」から標準モジュールを追加します。追加した標準モジュールファイルで DLL 関数の参照宣言を行います。宣言部分は、サンプルプログラム“REX5055M.BAS”の下記部分をコピーしてください。

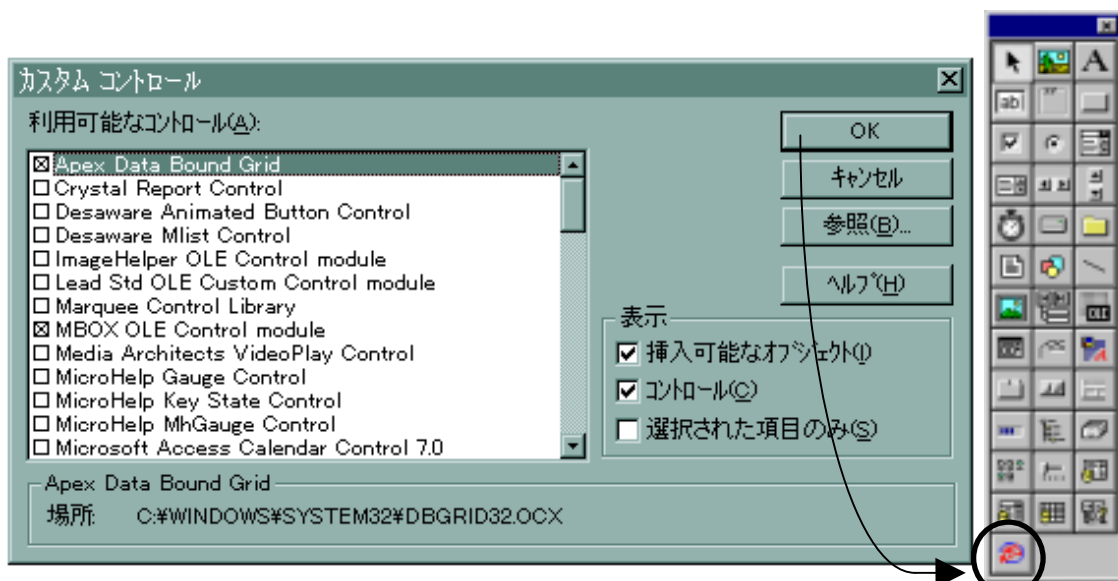
```
Declare Function IODrvResource Lib "5055lib" () As Long
Declare Function IRQDrvResource Lib "5055lib" () As Long
Declare Function OutPort Lib "5055lib" (ByVal Adrs As Integer, ByVal WriteData As Integer) As Integer
Declare Function wOutPort Lib "5055lib" (ByVal Adrs As Integer, ByVal WriteData As Integer) As Integer

Declare Function InPort Lib "5055lib" (ByVal Adrs As Integer) As Integer
Declare Function wInPort Lib "5055lib" (ByVal Adrs As Integer) As Integer

Declare Function IntSet Lib "5055lib" (ByVal CtrIReg As Byte, ByVal IntMask As Byte) As Integer
Declare Sub MsgIntStart Lib "5055lib" (ByVal hWnd As Long, ByVal Count As Integer, ByVal Flg As Integer)
Declare Sub FastIntStart Lib "5055lib" (ByVal hWnd As Long, ByVal InPortReg As Byte, ByVal OutPortReg As Byte, ByVal Count As Long, ByVal Flg As Integer)
Declare Sub TerminateInterrupt Lib "5055lib" ()
```

Step.4 MBOX OLE Control Module の追加 (割り込みサービス使用時必須)

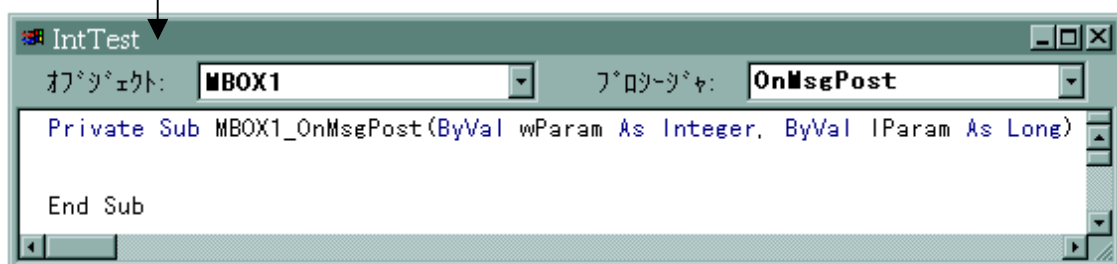
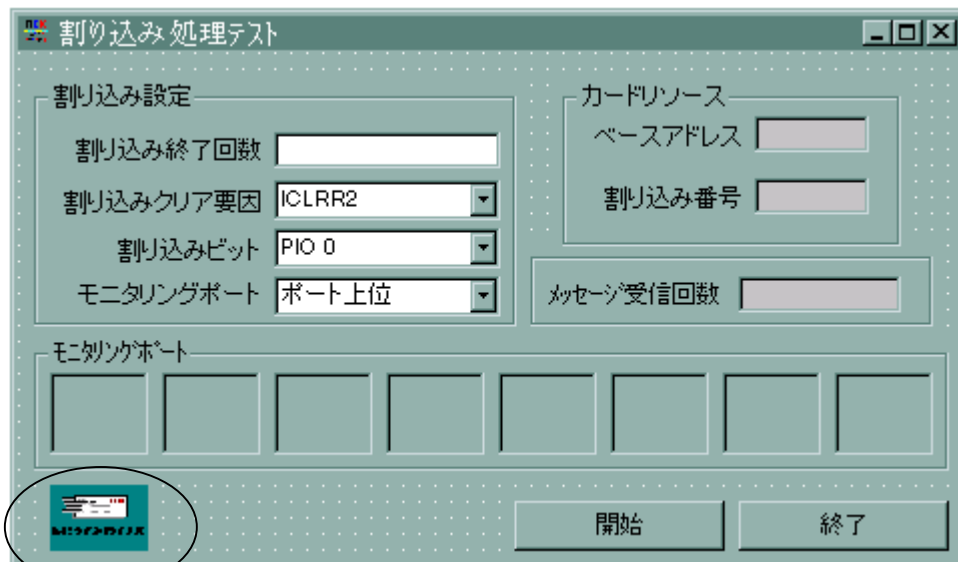
VB のカスタムコントロールに MBOX(OCX)を追加します。VB デザインメニューの「ツール」の「カスタムコントロール」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。



Step.5 フォームに MBOX(OCX)を貼り付ける (割り込みサービス使用時必須)

フォームを作成し、割り込みハンドラが割り込み起動元プログラムに送るユーザ定義メッセージを受け取るための MBOX(OCX)を貼り付けます。これにより、割り込みが発生すると MBOX がサービスするプロシージャ

MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)
が呼び出されます。この中で、割り込み通知に同期した処理を記述します。



第6章 オプションアイソレーションユニット

この章では、REX-5055 のオプションについて説明します。

(6-1) REX-IPI16 製品概要

REX-IPI16 は、PHI-8 の後継型であり外部機器からの無電圧接点入力をフォトプラによって電氣的にアイソレートして TTL レベル信号に変換するアイソレーション入力ユニットです。リードスイッチや近接センサーからの信号入力インターフェイスとして使用できます。各ポートの信号状態を LED で確認できます。

入力電圧を 5V, 12V, 24V に変更可能です。パソコンの電源電圧 (5V) 以上の外部機器と接続される場合には、使用する必要があります。

PHI-8 との違いは、入力ポート数 16、ワンタッチスクリューの螺子端子を使用し、また外形も BOX タイプとなっており非常に使いやすくなっています。



REX-5055 と接続するには、別売オプションケーブル RCL-5055T が必要です。

サンプルプログラムでのご使用について

MS-DOS/Windows3.1/Windows95/98/WindowsNT に添付している PHI-8 を使用したサンプルプログラムで、そのままご使用いただけます。

(6-2) REX-IPO16 製品概要

REX-IPO16 は、PH0-8 の後継型でありパソコン側からの 16 ポートの TTL レベル信号をフォトプラによって電氣的にアイソレートして外部出力を行うアイソレーション出力ユニットです。

出力は、最大 300V、165mA の駆動能力があり、電磁弁やパワーリレーの駆動制御、スイッチの ON/OFF 制御のインターフェイスとして使用できます。各ポートの信号状態を LED で確認できます。L レベル信号出力時に LED が点灯します。

PH0-8 との違いは、出力ポート数 16、ワンタッチスクリューの螺子端子を使用し、また外形も BOX タイプとなっており非常に使いやすくなっています。



REX-5055 と接続するには、別売オプションケーブル RCL-5055T が必要です。

サンプルプログラムでのご使用について

MS-DOS/Windows3.1/Windows95/98/WindowsNT に添付している PH0-8 を使用したサンプルプログラムで、そのままご使用いただけます。

(6-3) RCL-TRM48 製品概要

製品添付のオス側コネクタと外部機器を結線し、そのコネクタを本体メス側コネクタに挿入して信号の入出力を行うワンタッチスクリーターミナルユニットです。REX-5055 の信号を全てワンタッチスクリーターの螺子端子に出すためのユニットです。

面倒な外部機器からの配線接続作業をドライバ1本で行うことが可能です。また、外形もBOXタイプとなっており非常に使いやすくなっています。



REX-5055 と接続するには、別売オプションケーブル RCL-5055T が必要です。

発行 ラトックシステム株式会社
2002年03月22日 第7.0版 第1刷発行

📧 製品に対するお問い合わせ

REX-5055 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
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.06-6633-3553
<サポート受付時間>
月曜 - 金曜 (祝祭日は除く) 10:00 ~ 13:00
14:00 ~ 17:00

また、以下の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>

🔔 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。