



REX-5059

Universal Pulse Processor PC CARD

ユーザーズマニュアル

2002年11月

第2.0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) 概要	1-1
(1-2) 添付品	1-2
第2章 ハードウェア仕様	
(2-1) 添付ケーブルBOX入出力コネクタ	2-1
(2-2) レジスタマップ	2-2
(2-3) UPPレジスタマップ	2-3
第3章 UPP のレジスタマップとプログラミング	
(3-1) パルス入出力	3-1
(3-1-1) 外部入出力端子の割り付け	3-1
(3-1-2) 入力、出力の設定	3-3
(3-1-3) UPPシステムコントロール	3-3
(3-1-4) 最大動作ファンクション数の設定	3-4
(3-1-5) ファンクションテーブルの選択	3-6
(3-1-6) ファンクションテーブルの設定	3-7
(3-1-7) UPPデータレジスタ	3-12
(3-2) UPPタイマコマンド一覧	3-13
(3-3) A/D コンバータの使用	3-18
(3-4) デジタル入出力	3-23
(3-5) 割り込みの使用	3-25
(3-5-1) UPP の割り込みの発生	3-25
(3-5-2) 割り込み制御用レジスタの構成	3-26
(3-6) その他の機能	3-27
(3-6-1) ウォッチドッグ部	3-27
(3-6-2) その他特殊使用について	3-27
第4章 Windows95/98/Me 解説	
(4-1) Windows95 インストール	4-1
(4-2) Windows98 インストール	4-4
(4-3) WindowsMe インストール	4-7
(4-4) インストール内容の確認	4-8
(4-5) アンインストール	4-10
(4-6) DLL ライブラリ解説	4-12
(4-6-1) DLL ライブラリについて	4-12
(4-6-2) Visual C/C++アプリケーション作成	4-18
(4-6-3) Visual BASIC アプリケーション作成	4-30

第5章 Windows2000/XP 解説

(5-1) セットアップ	5-1
(5-1-1) Windows2000 インストール	5-1
(5-1-2) WindowsXP インストール	5-3
(5-1-3) インストール内容の確認	5-4
(5-1-4) アンインストール	5-5
(5-2) Visual C 言語インターフェース	5-7
(5-2-1) DLL ライブラリ解説	5-7
(5-2-2) Visual C サンプルプログラム	5-11
(5-3) Visual BASIC 言語インターフェース	5-20
(5-3-1) DLL ライブラリの Declare 宣言	5-20
(5-3-2) カスタムコントロール	5-21
(5-3-3) Visual BASIC サンプルプログラム	5-24

第6章 MS-DOS/Windows3.1 解説

(6-1) MS-DOS/Windows3.1 でのインストール	6-1
(6-1-1) イネーブラのインストール	6-1
(6-1-2) DOS/V 版カードサービス版イネーブラを使用する場合	6-3
(6-1-3) DOS/V 版ポイントイネーブラを使用する場合	6-6
(6-1-4) PC-98 版カードサービス版イネーブラを使用する場合	6-7
(6-2) MS-DOS ライブラリ	6-10
(6-2-1) MS-DOS ライブラリ関数	6-10
(6-2-2) MS-DOS サンプルプログラム	6-12
(6-3) Windows3.1 ライブラリ	6-15
(6-3-1) Windows3.1DLL 関数	6-16

第1章 はじめに

(1-1) 概要

REX5059 は DOS/V,PC-9800 シリーズに対応した PC カード TYPE II のパルス入出力カードです。コアプロセッサに日立製高機能パルス制御プロセッサ HD63140(UPP : Universal Pulse Processor)を搭載することにより、複雑なパルス制御を自動的に行わせることができ、CPU の処理を軽減できます。ノート PC を使った Plug & Play 対応の FA/LA システムを提供します。

ユニバーサルパルスプロセッサコア(UPC)

- パルスの入出力機能として、15 種類の専用コマンドがあります。
- パルスの入出力端子を 16 本、入出力用内部レジスタを 8 本内蔵しています。パラルの入出力としても使用可能です。
- 同時に最大 16 ファンクションのコマンドを実行可能です。
- パルス分解能は、実行するファンクション数によって決まります。
 - => 1 ファンクション実行時 :0.5 μ sec
 - => 16 ファンクション実行時 :5.0 μ sec
- 注)パルス分解能より短かいパルスの検出や出力はできません。
- 全パルス信号の立ち上がり、または立ち下りエッジの検出で、割り込み発生が可能です。
- 16 ビット×24 の汎用レジスタがあり、カウンタ・シフタ・コンペア・キャプチャレジスタとして使用できます。

A/D コンバータ

- 10 チャンネルの 10 ビット逐次比較型 A/D コンバータを内蔵しています。
- 最大 4 チャンネルまでスキャン動作可能です。
- 変換終了で割り込み発生が可能です。
- 変換時間は、1 チャンネルあたり 42 μ sec です。
- 電圧レンジは 0~5V 固定です。

消費電力

- Typical 5V 0.3A

パルス入出力コネクタ

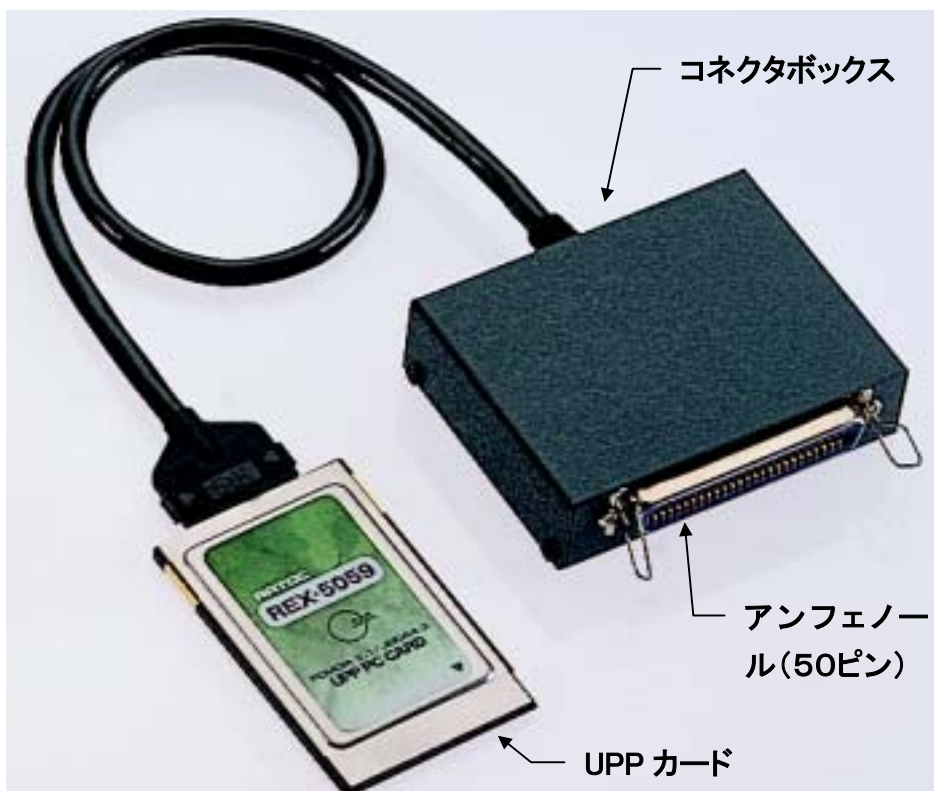
- ボックス側 アンフェノール 50 ピン(DDK 57LE40500-77C0)

(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。

- コネクタボックス(アンフェノール 50 ピンメス、ケーブル長 50cm)
- アンフェノール 50 ピンオス側コネクタ
- ドライバー・ライブラリディスク 1.44MB 3.5" (2 枚)
- ユーザーズマニュアル
- 日立 HD63140 UPP データシート
- ご愛用者登録はがき・保証書

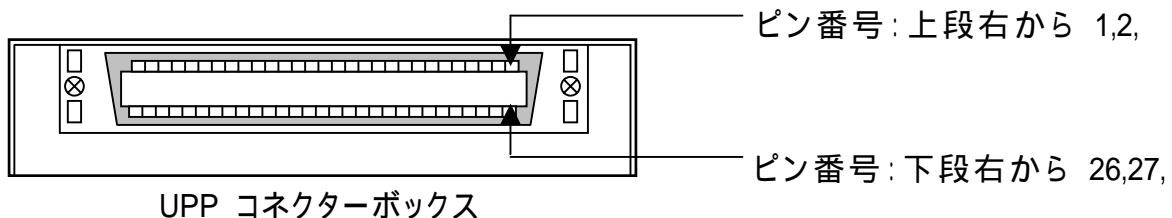
ご愛用者カードは保証書を切り離した後、必要事項を記入の上、必ずご返送ください。
ご返送頂けない場合、バージョンアップ等のサポートサービスは受けられませんのでご注意ください。



第2章 ハードウェア仕様

(2-1) 添付ケーブルBOX入出力コネクタ

ピン番	説明	I/O	信号名	ピン番	説明	I/O	信号名
1	+5V	O	+5V	26	+5V	O	+5V
2	アナログ入力	I	AN0	27	信号グランド	-	GND
3		I	AN1	28		-	GND
4		I	AN2	29		-	GND
5		I	AN3	30		-	GND
6		I	AN4	31		-	GND
7		I	AN5	32		-	GND
8		I	AN6	33		-	GND
9		I	AN7	34		-	GND
10		I	AN8	35		-	GND
11		I	AN9	36		-	GND
12	信号グランド	-	GND	37		-	GND
13	パルス入出力	I/O	U0	38	パルス入出力	I/O	U1
14		I/O	U2	39		I/O	U3
15		I/O	U4	40		I/O	U5
16		I/O	U6	41		I/O	U7
17		I/O	U8	42		I/O	U9
18		I/O	U10	43		I/O	U11
19		I/O	U12	44		I/O	U13
20		I/O	U14	45		I/O	U15
21	信号グランド	-	GND	46	信号グランド	-	GND
22	NC	-	NC	47	NC	-	NC
23	NC	-	NC	48	NC	-	NC
24	ウォッチドッグ出力	O	WTD0	49	基準クロック	O	8MHz
25	信号グランド	-	GND	50	信号グランド	-	GND



I/Oは信号の入出力の方向を示します。Iは入力、Oは出力、I/Oはプログラムによって入出力どちらにも指定可を示します。

WTD0信号(A-2)はUPPのウォッチドッグタイマのオーバーフロー出力ですが、出力はオープンドレインで内部でプルアップしていません

U0～U7の入力レベルはTTLレベル入力、U8～U15はシュミットトリガ入力です。

基準クロックは8MHzの方形波を常時出力しています。外部で基準クロックとして用いることができます。

(2-2) レジスタマップ

REX-5059 を制御するためには、(2-3)で示す UPP の各レジスタをプログラムする必要があります。PC カードでは、UPP の持つ全てのレジスタを一度に I/O マッピングすることはできません。

UPP の各レジスタにアクセスするためには、下記の手順で行います。

=>下記インデックスレジスタ0に、(2-3)で示す UPP レジスタインデックス番号の下位バイトをセットし、インデックスレジスタ1に上位バイトをセットする。

=>上記インデックス設定により、データレジスタに設定した UPP レジスタがマッピングされる。

=>データレジスタを通して、UPP のレジスタへの入出力を行う。

REX-5059 レジスタマッピング

オフセット	レジスタ名																
Base + 0	[データレジスタ] インデックスレジスタにより UPP のレジスタがマッピングされます																
Base + 1	Reserved																
Base + 2	[インデックスレジスタ0] UA7 ~ UA0 に UPP レジスタインデックス番号の下位 8 ビットをセットします。 <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="4">MSB</th> <th colspan="4">LSB</th> </tr> </thead> <tbody> <tr> <td>UA7</td> <td>UA6</td> <td>UA5</td> <td>UA4</td> <td>UA3</td> <td>UA2</td> <td>UA1</td> <td>UA0</td> </tr> </tbody> </table>	MSB				LSB				UA7	UA6	UA5	UA4	UA3	UA2	UA1	UA0
MSB				LSB													
UA7	UA6	UA5	UA4	UA3	UA2	UA1	UA0										
Base + 3	[インデックスレジスタ1] UA8 ~ UA10 に UPP レジスタインデックス番号の上位 3 ビットをセットします。 INTF READ => 1: UPP から割込み要求があることを示します。 UPP の割込み要因をクリアすることで 0 になります。 WRITE => 無効 <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="5">MSB</th> <th colspan="3">LSB</th> </tr> </thead> <tbody> <tr> <td>INTF</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>UA10</td> <td>UA9</td> <td>UA8</td> </tr> </tbody> </table>	MSB					LSB			INTF	-	-	-	-	UA10	UA9	UA8
MSB					LSB												
INTF	-	-	-	-	UA10	UA9	UA8										

(2-3) UPPレジスタマップ

レジスタマップ中で示されていないインデックスは未使用です。各レジスタへの入出力は必ずバイトアクセスしてください。詳細はHD63140 データシートをご参考ください。

インデックス	インデックス番号 レジスタ名称	記号	R/W	レジスタ名 ビット							
				7	6	5	4	3	2	1	0
00	Data Direction Register 2	DDR2	W	U15	U14	U13	U12	U11	U10	U9	U8
01	Data Direction Register 1	DDR1	W	U7	U6	U5	U4	U3	U2	U1	U0
02	Port2 Data Register 2	PORT2	R/W	P15	P14	P13	P12	P11	P10	P9	P8
03	Port1 Data Register 1	PORT1	R/W	P7	P6	P5	P4	P3	P2	P1	P0
05	Watchdog Timer Register	WDTR	R/W	OVF	WOE	-	MOD	-	OVC2	OVC1	OVC0
06	A/D Control and Status Reg.	ADCSR	R/W	ADEND	ADIE	ADST	SCAN	CH3	CH2	CH1	CH0
07	A/D Data Register 0 (H)	ADDR0	R/W	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
08	A/D Data Register 0 (L)	ADDR0	R/W	AD1	AD0	-	-	-	-	-	-
09	A/D Data Register 1 (H)	ADDR1	R/W	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0A	A/D Data Register 1 (L)	ADDR1	R/W	AD1	AD0	-	-	-	-	-	-
0B	A/D Data Register 2 (H)	ADDR2	R/W	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0C	A/D Data Register 2 (L)	ADDR2	R/W	AD1	AD0	-	-	-	-	-	-
0D	A/D Data Register 3 (H)	ADDR3	R/W	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0E	A/D Data Register 3 (L)	ADDR3	R/W	AD1	AD0	-	-	-	-	-	-
10	UPP Contact Enable Reg. 2	UCER2	W	UCE15	UCE14	UCE13	UCE12	UCE11	UCE10	UCE9	UCE8
11	UPP Contact Enable Reg. 1	UCER1	W	UCE7	UCE6	UCE5	UCE4	UCE3	UCE2	UCE1	UCE0
12	Output Register 2	UOR2	W	U15	U14	U13	U12	U11	U10	U9	U8
13	Output Register 1	UOR1	W	U7	U6	U5	U4	U3	U2	U1	U0
14	Next Data Enable Register	NDER	W	U15	U14	U13	U12	U11	U10	U9	U8
16	Next Data Register	NDR	W	U15	U14	U13	U12	U11	U10	U9	U8
20	UPP System Control Register	USCR	R/W	TST	TST	TST	TST	-	TST	GFE	UROME
21	Maximum Function Number Reg.	MFNR	R/W	-	-	-	MFN4	MFN3	MFN2	MFN1	MFN0
22	Function Number Register	FNR	R/W	-	-	-	FN4	FN3	FN2	FN1	FN0
23	Command Register	CMR	R/W	CMD3	CMD2	CMD1	CMD0	OM3	OM2	OM1	OM0
24	Register Assignment Reg. A	RASRA	R/W	-	-	-	CTN4	CTN3	CTN2	CTN1	CTN0
25	Register Assignment Reg. B	RASRB	R/W	-	-	-	CCL4	CCL3	CCL2	CCL1	CCL0
26	I/O Assignment Reg. A	IOARA	R/W	-	FEDGA	REDGA	CPN4	CPN3	CPN2	CPN1	CPN0
27	I/O Assignment Reg. B	IOARB	R/W	-	FEDGB	REDGB	SPN4	SPN3	SPN2	SPN1	SPN0
28	I/O Assignment Reg. C	IOARC	R/W	-	-	-	LPNA4	LPNA3	LPNA2	LPNA1	LPNA0
29	I/O Assignment Reg. D	IOARD	R/W	-	-	-	LPNB4	LPNB3	LPNB2	LPNB1	LPNB0
2A	INT Enable Register 3	IER3	R/W	IRE23	IRE22	IRE21	IRE20	IRE19	IRE18	IRE17	IRE16
2B	INT Enable Register 2	IER2	R/W	IRE15	IRE14	IRE13	IRE12	IRE11	IRE10	IRE9	IRE8
2C	INT Enable Register 1	IER1	R/W	IRE7	IRE6	IRE5	IRE4	IRE3	IRE2	IRE1	IRE0
2D	INT Request Register 3	IRR3	R	IRR23	IRR22	IRR21	IRR20	IRR19	IRR18	IRR17	IRR16
2E	INT Request Register 2	IRR2	R	IRR15	IRR14	IRR13	IRR12	IRR11	IRR10	IRR9	IRR8
2F	INT Request Register 1	IRR1	R	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0
30	INT Status Register 3	ISR3	R	IRS23	IRS22	IRS21	IRS20	IRS19	IRS18	IRS17	IRS16
31	INT Status Register 2	ISR2	R	IRS15	IRS14	IRS13	IRS12	IRS11	IRS10	IRS9	IRS8
32	INT Status Register 1	ISR1	R	IRS7	IRS6	IRS5	IRS4	IRS3	IRS2	IRS1	IRS0
33	INT Status Clear Register 3	ISCR3	W	ISC23	ISC22	ISC21	ISC20	ISC19	ISC18	ISC17	ISC16
34	INT Status Clear Register 2	ISCR2	W	ISC15	ISC14	ISC13	ISC12	ISC11	ISC10	ISC9	ISC8
35	INT Status Clear Register 1	ISCR1	W	ISC7	ISC6	ISC5	ISC4	ISC3	ISC2	ISC1	ISC0
36	UPP I/O Register	UIOR	R/W	U23	U22	U21	U20	U19	U18	U17	U16

インデックス番号		レジスタ名									
インデックス	レジスタ名称	記号	R/W	ビット							
				7	6	5	4	3	2	1	0
40	UPP Data Register 0 (H)	UDR0	R/W	D15	D14	D13	D12	D11	D10	D9	D8
41	UPP Data Register 0 (L)	UDR0	R/W	D7	D6	D5	D4	D3	D2	D1	D0
42	UPP Data Register 1 (H)	UDR1	R/W	D15	D14	D13	D12	D11	D10	D9	D8
43	UPP Data Register 1 (L)	UDR1	R/W	D7	D6	D5	D4	D3	D2	D1	D0
44	UPP Data Register 2 (H)	UDR2	R/W	D15	D14	D13	D12	D11	D10	D9	D8
45	UPP Data Register 2 (L)	UDR2	R/W	D7	D6	D5	D4	D3	D2	D1	D0
46	UPP Data Register 3 (H)	UDR3	R/W	D15	D14	D13	D12	D11	D10	D9	D8
47	UPP Data Register 3 (L)	UDR3	R/W	D7	D6	D5	D4	D3	D2	D1	D0
48	UPP Data Register 4 (H)	UDR4	R/W	D15	D14	D13	D12	D11	D10	D9	D8
49	UPP Data Register 4 (L)	UDR4	R/W	D7	D6	D5	D4	D3	D2	D1	D0
4A	UPP Data Register 5 (H)	UDR5	R/W	D15	D14	D13	D12	D11	D10	D9	D8
4B	UPP Data Register 5 (L)	UDR5	R/W	D7	D6	D5	D4	D3	D2	D1	D0
4C	UPP Data Register 6 (H)	UDR6	R/W	D15	D14	D13	D12	D11	D10	D9	D8
4D	UPP Data Register 6 (L)	UDR6	R/W	D7	D6	D5	D4	D3	D2	D1	D0
4E	UPP Data Register 7 (H)	UDR7	R/W	D15	D14	D13	D12	D11	D10	D9	D8
4F	UPP Data Register 7 (L)	UDR7	R/W	D7	D6	D5	D4	D3	D2	D1	D0
50	UPP Data Register 8 (H)	UDR8	R/W	D15	D14	D13	D12	D11	D10	D9	D8
51	UPP Data Register 8 (L)	UDR8	R/W	D7	D6	D5	D4	D3	D2	D1	D0
52	UPP Data Register 9 (H)	UDR9	R/W	D15	D14	D13	D12	D11	D10	D9	D8
53	UPP Data Register 9 (L)	UDR9	R/W	D7	D6	D5	D4	D3	D2	D1	D0
54	UPP Data Register 10 (H)	UDR10	R/W	D15	D14	D13	D12	D11	D10	D9	D8
55	UPP Data Register 10 (L)	UDR10	R/W	D7	D6	D5	D4	D3	D2	D1	D0
56	UPP Data Register 11 (H)	UDR11	R/W	D15	D14	D13	D12	D11	D10	D9	D8
57	UPP Data Register 11 (L)	UDR11	R/W	D7	D6	D5	D4	D3	D2	D1	D0
58	UPP Data Register 12 (H)	UDR12	R/W	D15	D14	D13	D12	D11	D10	D9	D8
59	UPP Data Register 12 (L)	UDR12	R/W	D7	D6	D5	D4	D3	D2	D1	D0
5A	UPP Data Register 13 (H)	UDR13	R/W	D15	D14	D13	D12	D11	D10	D9	D8
5B	UPP Data Register 13 (L)	UDR13	R/W	D7	D6	D5	D4	D3	D2	D1	D0
5C	UPP Data Register 14 (H)	UDR14	R/W	D15	D14	D13	D12	D11	D10	D9	D8
5D	UPP Data Register 14 (L)	UDR14	R/W	D7	D6	D5	D4	D3	D2	D1	D0
5E	UPP Data Register 15 (H)	UDR15	R/W	D15	D14	D13	D12	D11	D10	D9	D8
5F	UPP Data Register 15 (L)	UDR15	R/W	D7	D6	D5	D4	D3	D2	D1	D0
60	UPP Data Register 16 (H)	UDR16	R/W	D15	D14	D13	D12	D11	D10	D9	D8
61	UPP Data Register 16 (L)	UDR16	R/W	D7	D6	D5	D4	D3	D2	D1	D0
62	UPP Data Register 17 (H)	UDR17	R/W	D15	D14	D13	D12	D11	D10	D9	D8
63	UPP Data Register 17 (L)	UDR17	R/W	D7	D6	D5	D4	D3	D2	D1	D0
64	UPP Data Register 18 (H)	UDR18	R/W	D15	D14	D13	D12	D11	D10	D9	D8
65	UPP Data Register 18 (L)	UDR18	R/W	D7	D6	D5	D4	D3	D2	D1	D0
66	UPP Data Register 19 (H)	UDR19	R/W	D15	D14	D13	D12	D11	D10	D9	D8
67	UPP Data Register 19 (L)	UDR19	R/W	D7	D6	D5	D4	D3	D2	D1	D0
68	UPP Data Register 20 (H)	UDR20	R/W	D15	D14	D13	D12	D11	D10	D9	D8
69	UPP Data Register 20 (L)	UDR20	R/W	D7	D6	D5	D4	D3	D2	D1	D0
6A	UPP Data Register 21 (H)	UDR21	R/W	D15	D14	D13	D12	D11	D10	D9	D8
6B	UPP Data Register 21 (L)	UDR21	R/W	D7	D6	D5	D4	D3	D2	D1	D0
6C	UPP Data Register 22 (H)	UDR22	R/W	D15	D14	D13	D12	D11	D10	D9	D8
6D	UPP Data Register 22 (L)	UDR22	R/W	D7	D6	D5	D4	D3	D2	D1	D0
6E	UPP Data Register 23 (H)	UDR23	R/W	D15	D14	D13	D12	D11	D10	D9	D8
6F	UPP Data Register 23 (L)	UDR23	R/W	D7	D6	D5	D4	D3	D2	D1	D0

第3章 UPPのレジスタマップとプログラミング

HD63140(UPP)は、パルス入出力及び演算処理を行うユニバーサルパルスプロセッサコア(UPC)・10ビットA/Dコンバータ・ウォッチドッグタイマ及び1,024バイトRAMの4つのモジュールをワンチップ化したCMOS周辺LSIです。

UPCは、16ビットALUを内蔵したプログラマブルなパルス入出力モジュールで、カウンタ・シフタ・コンペアレジスタまたはキャプチャーレジスタとなる16ビット×24本の汎用レジスタと、16本のパルス入出力端子を持ち、応用機器に合った効率的なパルス制御システムを実現します。また、15種類のコマンドを組み合わせることにより、複雑なパルス制御を自動的に行わせることができるため、MPUの負荷を大幅に軽減できます。

本章では、UPPの数多くのパルス処理機能の中から、パルス入力・デジタル入出力・A/Dコンバータを中心に、UPPの動作させるために必要となる各レジスタの仕様とプログラミング方法について説明を行ないます。本製品添付のUPPデータシートと合わせて参照してください。

(3-1) パルス入出力

(3-1-1) 外部入出力端子の割り付け

インデックス[10h] W

UPP コンタクトイネーブルレジスタ 2(UCER2)

b7	b6	b5	b4	b3	b2	b1	b0
UCE15	UCE14	UCE13	UCE12	UCE11	UCE10	UCE9	UCE8

インデックス[11h] W

UPP コンタクトイネーブルレジスタ 1(UCER1)

b7	b6	b5	b4	b3	b2	b1	b0
UCE7	UCE6	UCE5	UCE4	UCE3	UCE2	UCE1	UCE0

UPPは外部入出力用の端子を16本持ち、入出力ボックスコネクタの13～20/38～45に割り当てられています。この端子は、1ビット単位でパルス入出力端子として使用するか、デジタル入出力端子として使用するかを選択できます。

インデックス10h,11hのUPPコンタクトイネーブルレジスタがこの選択用レジスタです。対応するビットに”1”を書き込むとパルス入出力、”0”を書き込むとデジタル入出力となります。

【動作モード】

パルス入出力モード =>UPCのパルス入出力端子として使用するモード

デジタル入出力モード =>MPUのデジタル入出力端子として使用するモード

リセット時は”0”となり、デジタル入出力モードとなっています。UPPの外部入出力数はU0～U23までの24本です。このうちのU0～U15は、UPPコンタクトイネーブルレジスタに”1”を書き込むことにより、入出力ボックスコネクタ13～20/38～45に割当てることができます。またU16～U23は、外部に対する入出力を行うことはできませんが、内部での中継用として、UPP I/Oレジスタ(インデックス36h UPP I/Oレジスタ)へのパルス入出力が可能です。

UCE=1のとき、対応する端子はUPCのパルス入出力端子となります。DDR=1のとき、UORの内容が端子に出力されます。また、この端子をUPCの入力ピンとして指定するとDDRにかかわらず端子の入出力値が入力値となります。このときの、UCERとUOR及び外部コネクタピン番号の関係は下記のようになります。

UCER2	UCE15	UCE14	UCE13	UCE12	UCE11	UCE10	UCE9	UCE8
	↓	↓	↓	↓	↓	↓	↓	↓
UOR2	U15	U14	U13	U12	U11	U10	U9	U8
	↓	↓	↓	↓	↓	↓	↓	↓
ピン番号	45	20	44	19	43	18	42	17

UCER1	UCE7	UCE6	UCE5	UCE4	UCE3	UCE2	UCE1	UCE0
	↓	↓	↓	↓	↓	↓	↓	↓
UOR1	U7	U6	U5	U4	U3	U2	U1	U0
	↓	↓	↓	↓	↓	↓	↓	↓
ピン番号	41	16	40	15	39	14	38	13

UCE=0のとき、対応する端子はMPUポートのデジタル入出力端子になります。DDR=1のとき、ポートデータレジスタの内容が端子に出力されます。UORは端子から切り離されますが、UPCの入出力レジスタとして使用することができ、UPCの入力ピンとして指定するとUOR1・UOR2の値が入力値となります。

インデックス36h UPP I/OレジスタとUPP入出力U16～U23との関係は下記の通りです。

インデックス[36h] R/W

UPP I/Oレジスタ (UIOR)

b7	b6	b5	b4	b3	b2	b1	b0
U23	U22	U21	U20	U19	U18	U17	U16

外部入出力端子をパルス入出力にした場合、実際にどのレジスタをその端子に割り当てるか、また入力にするか出力にするかという選択は別途行なう必要があります。

(3-1-2) 入力、出力の設定

インデックス [00h,01h] W

データディレクションレジスタ(DDR2,DDR1)

(3-1-1)で割り付けた端子の入力、出力の方向を指定します。“1”で出力、“0”で入力となります。リセット時は”00”となり、総て入力となっています。00hに書き込んだデータは、10h、01hは11hの割り付けと対応します。

また、REX-5059 カードでは、UPP からの入出力が直接コネクタにアサインされていますので、データ方向には充分注意してください。もし外部からのデータ入力と、REX-5059 側からのデータ出力が同一端子に割り当てられますと、UPP が破損する場合があります。

(3-1-3) UPPシステムコントロール

インデックス [20h] R/W

システムコントロールレジスタ(USCR)

UPP の状態をコントローラするレコードです。b1 の GFE=1 の時、UPP は動作状態、“0”の時停止状態となります。

b7	b6	b5	b4	b3	b2	b1	b0
TST	TST	TST	TST	-	TST	GFE	UROME
w	w	w	w	-	w	R/W	R/W

ビット	記号	説明
b7 ~ b4, b2	TST	メーカー側(日立)のテスト用であり、必ず”0”を書き込んでください
b3	Reserved	-
b1	GFE	General Function Enable “GFE=1”の場合、UPP は動作状態となります。UPP は動作中の場合、データの書き込みは行なえませんので、イニシャライズの際は必ず “GFE=0” として UPP を停止させてください。リセット時は “0”(停止)となっています。
b0	UROME	UROMEUPP ROM Enable UPP はマスク ROM を内蔵しているタイプがありますが、REX-5059 ではマスク ROM は持っていません。そのため、必ずこのビット=”0”としてください。”1”としても、無効です。

(3-1-4) 最大動作ファンクション数の設定

インデックス [21h] R/W

マキシマムファンクションナンバレジスタ(MFNR)

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	MNF4	MNF3	MNF2	MNF1	MNF0
-	-	-	R/W	R/W	R/W	R/W	R/W

UPP の動作ファンクションの最大数を設定します。パルス入出力動作時の実際の動作については、後述の「(3-1-5)ファンクションテーブルの選択」及び「(3-1-6)ファンクションテーブルの設定」により決定されます。ただし、設定された動作はここで指定する数しか実行されません。

また、UPP のパルス分解能はここで設定された値により決定されます。REX-5059 では、動作周波数は 4MHz のため、下記の式により算出されます。

$$\Rightarrow \text{パルス分解能} = (\text{MFNR}) \times 0.25 \mu\text{sec} \quad \text{注) MFNRはここで設定される値}$$

1ファンクションの場合は、

$$\Rightarrow \text{パルス分解能} = 2 \times 0.25 \mu\text{sec} = 0.5 \mu\text{sec}$$

16ファンクション(最大)の場合は、

$$\Rightarrow \text{パルス分解能} = 20 \times 0.25 \mu\text{sec} = 5 \mu\text{sec}$$

となります。

ただし、MFNR = 0 の場合、16ファンクション実行されますが、

$$\Rightarrow \text{パルス分解能} = 8 \mu\text{sec}$$

となります。

また、UPPの初期設定をする場合は、このレジスタに最大動作数を設定してからファンクションテーブルの選択、ファンクションテーブルの設定を行なってください。

MFNRの値と、ファンクションの数は一致していない場合がありますので、表 3-1-4 を参照の上、設定する値(MFNR)を決定してください。

表 3-1-4.MFNR の値と実行されるファクションの数の対応

MFNR の 値 10 進	16 進	MFN4	MFN3	MFN2	MFN1	MFN0	実行ファンク ション数
0	0	0	0	0	0	0	16
1	1	0	0	0	0	1	0
2	2	0	0	0	1	0	1
3	3	0	0	0	1	1	2
4	4	0	0	1	0	0	3
5	5	0	0	1	0	1	4
6	6	0	0	1	1	0	4
7	7	0	0	1	1	1	5
8	8	0	1	0	0	0	6
9	9	0	1	0	0	1	7
10	A	0	1	0	1	0	8
11	B	0	1	0	1	1	8
12	C	0	1	1	0	0	9
13	D	0	1	1	0	1	10
14	E	0	1	1	1	0	11
15	F	0	1	1	1	1	12
16	10	1	0	0	0	0	12
17	11	1	0	0	0	1	13
18	12	1	0	0	1	0	14
19	13	1	0	0	1	1	15
20	14	1	0	1	0	0	16
21	15	1	0	1	0	1	16
22	16	1	0	1	1	0	16
23	17	1	0	1	1	1	16
24	18	1	1	0	0	0	16
25	19	1	1	0	0	1	16
26	1A	1	1	0	1	0	16
27	1B	1	1	0	1	1	16
28	1C	1	1	1	0	0	16
29	1D	1	1	1	0	1	16
30	1E	1	1	1	1	0	16
31	1F	1	1	1	1	1	16

(注記)

印がついている設定は、ファクションの数と MFNR の値が対応していない部分ですので、ご注意ください。また、MFNR=14 から 1F(16 進数)までは実行されるファクションの数は 16 です。

(3-1-5) ファンクションテーブルの選択

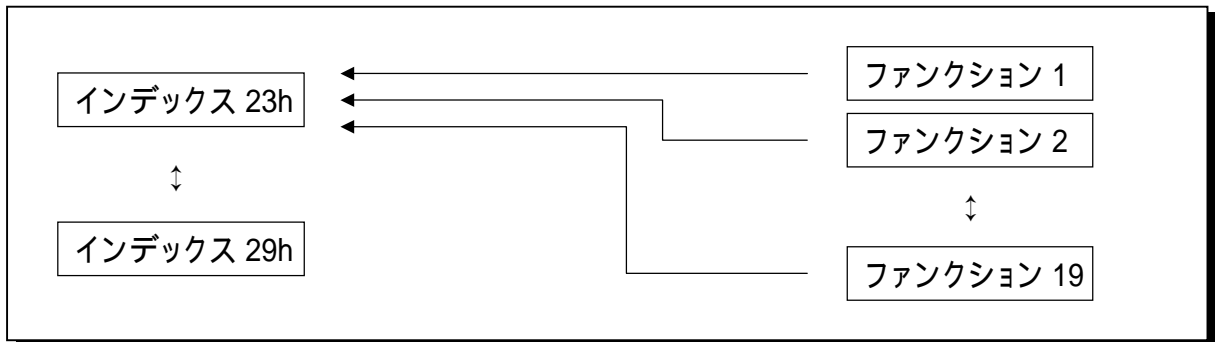
インデックス [22h] R/W

ファンクションナンバレジスタ (FNR)

B7	B6	B5	B4	B3	B2	B1	B0
未	使	用	FNR4	FNR3	FNR2	FNR1	FNR0
-	-	-	R/W	R/W	R/W	R/W	R/W

UPP は、インデックス 23h~29h の UPP は、インデックス 23h~29h の動作モード設定用テーブルに、値を書き込み動作モードを決定します。設定用テーブルには、動作テーブルの選択により、選択されたファンクションの値が表われています。

(注記)インデックス 22h で実行される値を”FNR”とします。



FNR の値を1とすると、上図矢印で示すように、インデックス 23h~29h に、ファンクション 1 の設定内容が表われます。そこで、動作モードの書き込みを行ないます。

UPP 動作中 (UPP システムコントロールビット GFE の値=1 の場合) には、このレジスタにアクセスすることはできません。また、リセット時には FNR の値は不定です。

UPP の動作中は、このレジスタは、1 ファンクション実行により、インクリメントされ、次のファンクションの番号となります。そして(3-1-4)で設定された MFNR の値と一致するとリセットされます。つまり、動作は MFNR で設定された値しか行なわれないということになります。

ファンクション番号として設定できる値は、1~4、6~9、11~14、16~19 の計 16 です。0、5、10、15 は設定しても何も実行されませんので注意してください。

記号	解説
C/T (CTR/TMR)	0のときレジスタ i はタイマとして動作し、内部クロックをカウントします。 1のときレジスタ i はカウンタとして動作し、信号 p の指定された方向のエッジをカウントします。
E/I (EXT/INT)	0のときレジスタ i は内部クロックでシフトします。 1のときレジスタ i は外部クロック信号 p の指定された方向のエッジでシフトします。
D/U (DWN/UP)	0のときレジスタ i はアップカウントします。 1のときレジスタ i はダウンカウントします。
I/N (INV/NINV)	アップダウンカウンタのカウントの方向を制御します。
H/L (HIGH/LOW)	出力パルスの極性、ゲート信号の極性を指定します。なお、カウンタのオーバーフロー信号の極性はこのビットの影響を受けません。
L/R (LEFT/RIGHT)	0のときレジスタ i はライトシフトになります。 1のときレジスタ i はレフトシフトになります。
i	カウンタ・タイマまたはシフトとなる UDR 番号
j	キャプチャレジスタ・コンペアレジスタまたはシフトコマンドで用いるデータレジスタとなる UDR 番号
p	クロック入力ピン番号
q	パルス信号入力ピン番号
r	パルス出力ピン番号
s	カウント方向制御信号・ゲート信号・トリガイネーブル信号またはシフト方向制御信号の入力ピン番号
FA (FEDGA)	1のとき信号 p の立ち下がりエッジを検出します。
RA (REDGA)	1のとき信号 p の立ち上がりエッジを検出します。
FB (FEDGB)	1のとき信号 q または信号 s の立ち下がりエッジを検出します。
RB (REDGB)	1のとき信号 q または信号 s の立ち上がりエッジを検出します。
S0-S2,O0-O4	シフトコマンドのシフトモードと出力方向を指定します。

インデックス [23h] R/W コマンドレジスタ (CMR)

B7	B6	B5	B4	B3	B2	B1	B0
CMD3	CMD2	CMD1	CMD0	OM3	OM2	OM1	OM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ビット	記号	説明
b7 ~ b4	CMD3-0	コマンドコードをセットします。
b3 ~ b0	OM3-0	設定されたコマンドの内容をさらに細かく定義します。コマンドにより同一ビットでも意味が変わります。

インデックス [24h] R/W レジスタアサイメントレジスタ A (RASRA)

b7	b6	b5	b4	b3	b2	b1	b0
未	使	用	CTN4	CTN3	CTN2	CTN1	CTN0
			R/W	R/W	R/W	R/W	R/W

各ファンクションでのカウンタ・タイマ・シフトとなる UPP データレジスタの番号の指定を行ないます。指定する値は 0 ~ 23 です。

データレジスタ=16 を指定する場合は、CTN4 = 1, CTN3 ~ CTN0 = 0 となりますので、

```

outp ( ADRS + 0x03, 0 );           (インデックス上位の指定)
outp ( ADRS + 0x02, 0x24 );       (インデックス下位の指定)
outp ( ADRS, 0x10 );              (データレジスタに 16 を指定)
    
```

となります。

インデックス [25h] R/W レジスタアサイメントレジスタ B (RASRB)

b7	b6	b5	b4	b3	b2	b1	b0
未	使	用	CCL4	CCL3	CCL2	CCL1	CCL0
			R/W	R/W	R/W	R/W	R/W

各ファンクションで、一時記憶 / 比較用レジスタまたはシフトコマンドの場合のリロードレジスタとして使用する UPP データレジスタの番号を指定します。レジスタの番号は 0 ~ 23 までしかありませんが、一時記憶用レジスタ等を必要としない場合は、24 ~ 31 をセットします。例としては、FRS コマンドを実行し、単純にパルスのカウントのみを行なうときには、RASRB の値を 24 とし、記憶用 (キャプチャ) レジスタを使用しないことができます。また、他のコマンドと重複して同一レジスタを指定する使用も可能です。


例として、他のコマンドでカウンタレジスタとして指定したレジスタを比較レジスタとして指定することにより、ある入力を基準とした大小比較等を行なうことができます。

インデックス [26h] R/W

I/O アサイメントレジスタ A (IOARA)

b7	b6	b5	b4	b3	b2	b1	b0
-	FEDGA	REDGA	CPN4	CPN3	CPN2	CPN1	CPN0
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

RASRA (24h) で指定したレジスタに対する外部クロックを指定するためのレジスタです。クロックの入力ピン番号とエッジの方向指定を行ないます。内部クロックモードを指定した場合 (=タイマモード内部クロックによるシフトモード) にはこのレジスタの内容は無視されます。

ビット	記号	説明
ビット6	FEDGA	このビットが“1”の場合、CPN4～CPN0 で指定された信号に立ち下りエッジがあると、カウントまたはシフトします。
ビット5	REDGA	このビットが“1”の場合、指定された信号の立ち上がりエッジにより、カウントまたはシフトします。 <div style="text-align: center;">  </div> <p>FEDGA=REDGA=1 の場合、 でカウントされます。 FEDGA=1,REDGA=0 の場合、 でカウントされます。 FEDGA=0,REDGA=1 の場合、 でカウントされます。 FEDGA=REDGA=0 の場合、カウントされません。(但し、内部クロックを指定した場合は除く。)</p>
ビット4～0	CPN4～CPN0	カウンタのクロック信号の入力ピン番号を0～23で指定します。ただし、0～15は入出力BOXのコネクタ 13～20 / 38～45に対応し、16～23はUPP内部のI/Oレジスタの各ビットを示します。

インデックス [27h] R/W

I/O アサイメントレジスタ B (IOARB)

b7	b6	b5	b4	b3	b2	b1	b0
-	FEDGB	REDGB	SPN4	SPN3	SPN2	SPN1	SPN0
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

サンプリングパルス、トリガパルス、リセットパルス等のパルスを指定するためのレジスタです。b6～b5 は、パルスのエッジ指定、b4～b0 は、パルス入力の入力ピン番号の指定に使用します。b7 は未使用です。

ビット	記号	説明
ビット6	FEDGB	このビットが"1"の時、指定された入力ピンの信号に立ち下がりがあると、サンプリング・トリガ・リセットを行ないます。
ビット5	REDGB	このビットが"1"の時、指定された入力ピンの信号に立ち上がりがあると、サンプリング・トリガ・リセットを行ないます。
ビット4～0	SPN4～SPN0	記憶レジスタへのサンプリングパルス、ワンショットパルス出力のトリガパルス、カウンタのリセットパルス、2相パルス計測時の一方のパルス及びゲート信号の入力ピン番号を指定します。

インデックス [28h] R/W

I/O アサイメントレジスタ C (IOARC)

b7	b6	b5	b4	b3	b2	b1	b0
未	使	用	LPNA4	LPNA3	LPNA2	LPNA1	LPNA0
			R/W	R/W	R/W	R/W	R/W

通常パルスの出力ピン番号を0～23で指定します。出力しない場合は24～31をセットします。また、2相パルスの計測時(TPC コマンド)の場合は、IOARA の CPN4～CPN0 と同じ値をセットしてください。

インデックス [29h] R/W

I/O アサイメントレジスタ D (IOARD)

b7	b6	b5	b4	b3	b2	b1	b0
未	使	用	LPNB4	LPNB3	LPNB2	LPNB1	LPNB0
			R/W	R/W	R/W	R/W	R/W

カウンタ、シフトの方向指定信号、ゲート及びトリガイネーブル信号のピン番号を指定します。また、2相パルスの計測時(TPC コマンド)の場合は、IOARB の SPN4～SPN0 と同じ値をセットしてください。指定する値は0～23です。

(3-1-7) UPPデータレジスタ

UPP のデータレジスタ部です。インデックス 40h~6Fh まで 2 バイトずつを、それぞれ 1 個のデータレジスタが占有します。計 24 本の 16 ビットレジスタがあり、それぞれファンクションの指定によりカウンタ/タイマ、シフト、一時記憶レジスタ、または比較用レジスタとなります。アドレスインデックス 40h、41h は UDR0 の H、L レジスタ、42h、43h は UDR1 となり、偶数バイトにアドレスされている方が上位 8 ビット、奇数バイトが下位 8 ビットとなります。

また、書き込み、読み出しを行なう場合には、必ず上位 8 ビット、下位 8 ビットの順で動作を行なってください。

=>UDR23 に 16 ビットデータ 50000 (C350h) を書き込む場合

```

outp( ADRS+0x03,00 );      /* インデックス上位の指定 */
outp( ADRS+0x02,0x6e );   /* インデックス下位の指定 */
outp( ADRS,0xC3 );        /* 値を UDR23H に書き込む */

outp( ADRS+0x03,00 );      /* インデックス上位の指定 */
outp( ADRS+0x02,0x6f );   /* インデックス下位の指定 */
outp( ADRS,0x50 );        /* 値を UDR23L に書き込む */

```

本製品添付のライブラリ関数 OutUpp()を使うと、次のように簡単に記述できます。

```

OutUpp ( ADRS, 0x6e, 0xC3 );
OutUpp ( ADRS, 0x6f, 0x50 );

```

=>UDR0 から 16 ビットデータを読み出す場合

```

outp( ADRS+0x03,00 );      /* インデックス上位の指定 */
outp( ADRS+0x02,0x40 );   /* インデックス下位の指定 */
Val = inp( ADRS );         /* 値を UDR0H から読込む */

outp( ADRS+0x03,00 );      /* インデックス上位の指定 */
outp( ADRS+0x02,0x41 );   /* インデックス下位の指定 */
Val = inp( ADRS );         /* 値を UDR0L から読込む */

```

本製品添付のライブラリ関数 InUpp()を使うと、次のように簡単に記述できます。

```

Val = InUpp ( ADRS, 0x40 );
Val = InUpp ( ADRS, 0x41 );

```

(3-2) UPPタイマコマンド一覧

詳しい動作については添付 HD63140 データシートの 頁 26 ~ 36 を参照してください。

コマンド	機能
カウンタ・タイマ とパルス入力機能	FRS =>フリーランカウンタ クロック信号の立ち下がりまたは立ち上がりにより、レジスタの値を+1 または-1 します。信号の立ち下がり / 立ち上がり、+1/-1 の選択は、別に指定します。 また、クロック信号以外にサンプリング信号を入力することができ、そのサンプリング信号の立ち下がり、または立ち上がりにより、現在のレジスタの値を記憶用レジスタにコピーします。
	INS =>インターバルカウンタ クロック信号の立ち上がり、または立ち下がりにより、カウンタをインクリメント / デクリメントします。 サンプリング信号の立ち下りまたは立ち上がりにより、現在のカウンタレジスタの値を記憶用レジスタにコピーし、カウンタレジスタをリセットします。
	UDS =>アップダウンカウンタ 1 のフリーランカウンタ機能に、入力のカウンタ方向指定信号により、アップダウンカウントを行ないます。
	GTS =>ゲートカウンタ ゲート信号が指定したレベルの期間に、クロック信号をカウントします。ゲート信号の立ち下がり、又は立ち上がりにより、カウンタレジスタの値を記憶用レジスタへ転送します。ゲート幅の測定を行ないません。
カウンタ・タイマ とパルス出力機能	FRC =>フリーランカウンタとレジスタとの比較 1 のフリーランカウンタ機能と同様にクロック信号の立ち下がり、立ち上がりによってカウントします。但し、比較レジスタを指定し、その比較レジスタとの比較効果を出力します。
	INC =>インターバルカウンタとレジスタとの比較 クロック信号の立ち下がり、又は立ち上がりによってカウントします。但し、比較レジスタを指定し、その比較レジスタとカウントレジスタが同じになった場合、出力信号を出し、カウントレジスタをリセットします。
	PWC =>パルスカウンタとレジスタの比較 リセット信号により、カウントレジスタ及び出力信号をリセットします。クロック信号の立ち下がり、又は立ち上がりによりカウントし、比較用レジスタとの比較結果を出力します。
	OSC =>ワンショットカウンタとレジスタの比較 トリガ信号に指定されたエッジが入力されると、カウンタが起動し、クロック信号の立ち下がり、又は立ち上がりによってカウントします。比較用レジスタとカウントレジスタを比較し、カウントレジスタ = 比較レジスタとなった場合に出力信号を反転し、カウントレジスタをリセットし、次のトリガ信号までその状態を保持します。

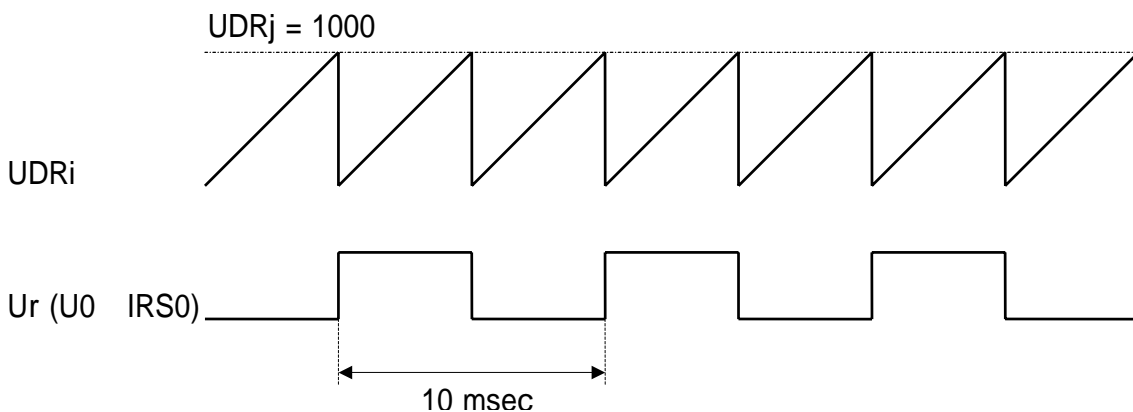
コマンド	機能
特殊カウンタ・ タイマ機能	FFC =>50%デューティパルス出力 クロック信号の立ち下がり又は立ち上がりにより、カウントし、比較レジスタとカウントレジスタが等しくなれば出力信号を反転し、カウントレジスタをリセットします。 つまり比較レジスタに設定した値×クロック信号の間、出力信号は”H”または”L”になり、50%デューティのパルスが得られます。
	TPC =>2相パルスの位相差によるアップ/ダウンカウント 2相パルス信号の位相関係により、カウンタをインクリメント/デクリメントします。
	GTC =>ゲートカウンタとレジスタの比較 ゲート信号が指定したレベルの期間に、クロック信号の立ち下がり、または立ち上りをカウントします。カウントレジスタと比較レジスタの値を比較して出力信号を出します。カウントレジスタはゲート信号の指定されたエッジによりリセットされます。 この機能は、ゲート信号の指定されたレベルの期間と、比較レジスタとの大小判定として利用できます。
	CTO =>トリガイネーブル機能付ワンショット出力 OSC コマンド(ワンショットカウンタ機能)に、トリガイネーブル機能を付加したモードです。トリガ信号は、トリガイネーブル端子が指定されたレベルの間のみ有効です。
シフトとパルス 入出力機能	SIT =>シフトインプット 入力信号の状態(0又は1)を、クロック信号に指定されたエッジがあると、シフト用のレジスタのLSBに取りこみ、レジスタの値を1ビットシフトします。サンプリング信号に指定された方向のエッジがあると、シフト用レジスタにより、記憶用レジスタへデータを転送します。シリアル同期通信の入力に応用可能です。
	SOT =>シフトアウトプット セット信号に指定されたエッジがあると、データセット用レジスタよりシフト又はローテートし、出力信号に1ビット出力します。シリアル同期通信の出力に応用可能です。
	SPO =>シフトパラレルアウトプット セット信号に指定されたエッジがあると、データセット用レジスタよりシフト用レジスタにデータが転送されます。クロック信号に指定されたエッジがあると、シフト用レジスタの内容が1ビットシフトされ、またその下位8ビットの内容がパラレル出力ポートに転送されます。

FFC・INC コマンドを使用して、1秒間隔でパルス出力を行う例

(1) FFC コマンドを使用して、デューティ 50% の 10msec 周期パルスを作成

クロック信号は内部信号を使用します。1クロックは、16 ファンクション実行時には 5 μ sec となります。

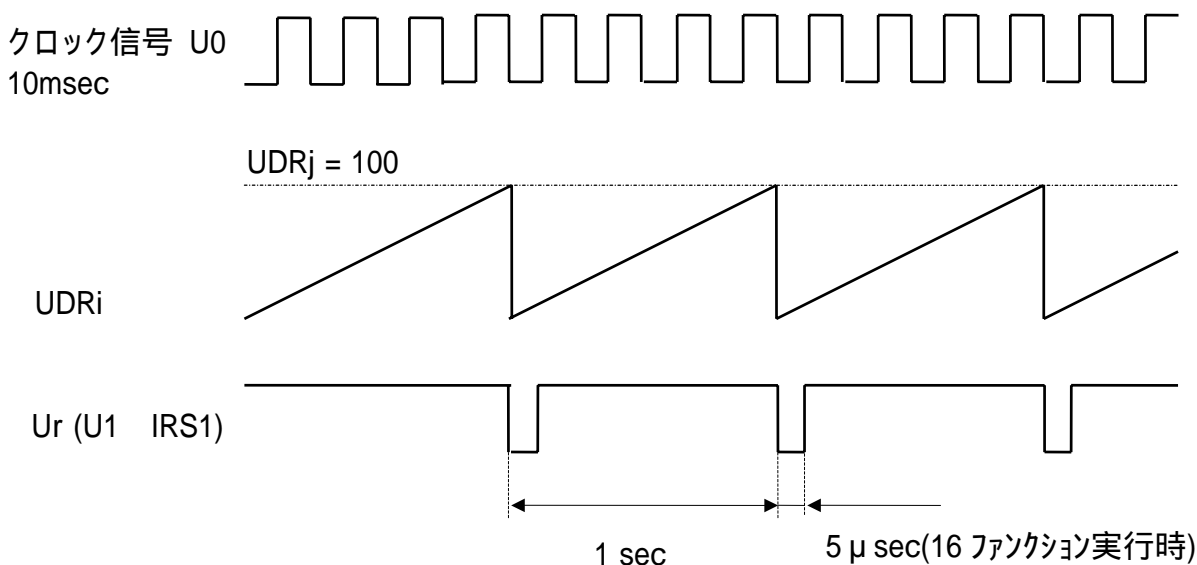
比較(コンペア)レジスタ UDRj に 1000 を設定します。カウンタ UDRi は 0 から UDRj の値まで内部クロックをカウントし、UDRj と同じ値になると出力を反転します。以上により、出力信号 Ur は 5 μ sec × 1000 × 2 = 10msec の周期となります。



(2) U0 をクロックとして使用し、INC コマンドから 1 秒周期のパルスを作成

10msec 周期のパルス U0 を INC モードの入力クロックとし、1 秒毎にパルス幅分解能に等しい幅のパルスを出力します。

INC モードでは、入力クロックが 10msec 周期のため、コンペアレジスタ UDRj に 100 を設定し、カウンタ UDRi の値がコンペアレジスタの値と同じになった時にパルスを出力します。



INC コマンドでは、パルス幅分解能に等しいパルス幅のパルスが出力されます。

☛ サンプルプログラム

```

#include < stdio.h >
#include < conio.h >
#include < time.h >
#include "UPPLIB.H"
BYTE fdata[16][8] = {
    /*FNR, CMR,RASRA,RASRB,IOARA,IOARB,IOARC,IOARD*/
    { 1, 0x80, 1, 0, 0xff, 255, 0, 255 }, /*FFC コマンド*/
    { 2, 0x58, 3, 2, 0x20, 255, 1, 255 }, /*INC コマンド*/
    { 3, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 4, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 6, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 7, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 8, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 9, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 11, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 12, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 13, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 14, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 16, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 17, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 18, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 19, 0xff, 255, 255, 0xff, 255, 255, 255 }
};

void main( void )
{
    time_t start,finish;
    WORD IOBase, count1, count2, fun, reg, udr;

    do{
        printf( "%x0cR E X 5 0 5 9レジスタベースアドレス $" );
        scanf( "%x", &IOBase );
    }while( ( IOBase<0x120) || ( IOBase > 0x350 ) );

    OutUpp( IOBase, USCR, 0x00 ); /*U P C 停止*/
    OutUpp( IOBase, IER3, 0x00 ); /*割り込みマスク(UPP IER3)*/
    OutUpp( IOBase, IER2, 0x00 ); /*割り込みマスク(UPP IER2)*/
    OutUpp( IOBase, IER1, 0x00 ); /*割り込みマスク(UPP IER1)*/
    OutUpp( IOBase, UOR2, 0x00 ); /*UPP Output Reg.2 Output Clear*/
    OutUpp( IOBase, UOR1, 0x00 ); /*UPP Output Reg.1 Output Clear*/
    OutUpp( IOBase, MFNR, 1 ); /*Maximum Function Number Reg. (0 Function)*/
    OutUpp( IOBase, FNR, 0x00 ); /*Function Number Reg. CLEAR*/
    OutUpp( IOBase, USCR, 0x02 ); /*U P C 動作(サンプリングフリップフロップクリア)*/
    OutUpp( IOBase, USCR, 0x00 ); /*U P C 停止*/
    OutUpp( IOBase, DDR2, 0xff ); /*Data Direction Reg.2 ALL OUTPUT*/
    OutUpp( IOBase, DDR1, 0xff ); /*Data Direction Reg.1 ALL OUTPUT*/
    OutUpp( IOBase, UCER2,0xff ); /*UPP Contact Enable Reg.2 Pulse I/O Enable*/
    OutUpp( IOBase, UCER1,0xff ); /*UPP Contact Enable Reg.1 Pulse I/O Enable*/
    OutUpp( IOBase, MFNR, 20 ); /*Maximum Function Number Reg. (16 Function 5us)*/
    for( fun=0;fun<=15;fun++ )
        for( reg=0;reg<=7;reg++ )
            OutUpp( IOBase, FNR+reg, fdata[fun][reg] ); /*ファンクションテーブルにデータセット*/
}

```

次頁に続く

```
count1=1000; /*分周カウント(コンペアデータ)10msec に1回のパルス出力*/
OutUpp( IOBase,UDR0H, (BYTE)(count1 >> 8) ); /*UPP Data Reg. 0 上位*/
OutUpp( IOBase,UDR0L, (BYTE)(count1 & 0xff) ); /*UPP Data Reg. 0 下位*/

count2=100; /*分周カウント(コンペアデータ)1msec に1回のパルス出力*/
OutUpp( IOBase,UDR2H, (BYTE)(count2 >> 8) ); /*UPP Data Reg.2 上位*/
OutUpp( IOBase,UDR2L, (BYTE)(count2 & 0xff) ); /*UPP Data Reg.2 下位*/

for( udr=4;udr<=7;udr++){
OutUpp( IOBase,UDR0H+udr*2, 0 ); /*UPP Data Reg. 1-23 CLEAR*/
OutUpp( IOBase,UDR0L+udr*2+1, 0 );
}
for( udr = 8; udr <= 15; udr++ ){
OutUpp( IOBase, UDR8H+(udr-8)*2, 0 ); /*UPP Data Reg. 1-23 CLEAR*/
OutUpp( IOBase, UDR8L+(udr-8)*2+1, 0 );
}
for( udr = 16; udr <= 23; udr++ ){
OutUpp( IOBase, UDR16H+(udr-16)*2, 0 ); /*UPP Data Reg. 1-23 CLEAR*/
OutUpp( IOBase, UDR16L+(udr-16)*2+1, 0 );
}

time(&start); /*タイムカウント開始*/
OutUpp( IOBase, FNR, 0x01 ); /*ファンクション1から実行*/
OutUpp( IOBase, USCR, 0x02 ); /*UPC動作(ファンクション実行)*/

while( kbhit() ) /* 先行入力引き取り */
    getch();

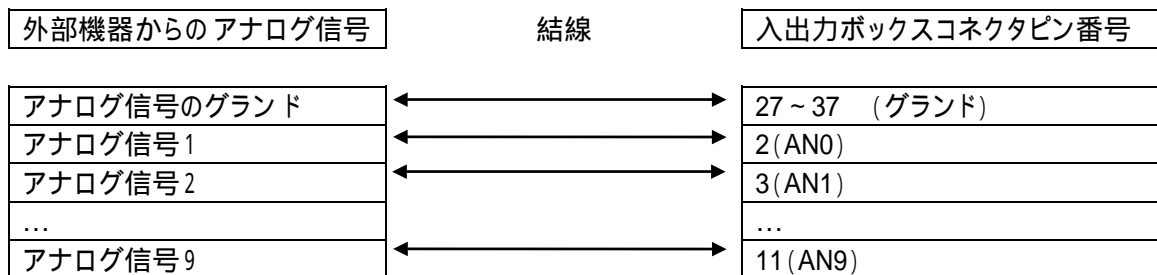
while( !kbhit() ){
    while( ( InUpp( IOBase, ISR1 ) & 0x02 ) == 0 ); /*インタラプトステータスリード*/
    OutUpp( IOBase, ISCR1, 0x00 ); /*インタラプトステータスクリア*/
    time(&finish);
    printf( " time = %6.2f sec\n",difftime(finish,start));
}
}
```

(3-3) A/D コンバータの使用

UPP は 10 ビット分解能の A/D コンバータを、10ch 内蔵しています。この A/D コンバータの入力は、入出力 BOX コネクタの 2～11 に割り付けられています。また、アナログ入力用グラウンドは、27～37 に割り付けられています。

入力電圧は、0～5V までのアナログ入力です。また、REX-5059 では、アナログ入力用の電源部は、デジタル部と分離されておりません。

接続配線図



インデックス [06h] R/W

A / D コントロールステータスレジスタ (ADCSR)

B7	B6	B5	B4	B3	B2	B1	B0
ADEND	ADIE	ADST	SCAN	CH3	CH2	CH1	CH0

ビット	記号	説明
b7	ADEND	<p>変換終了</p> <p>A/D 変換の終了時に"1"にセットされます。スキャンモードの場合は、選択された総てのチャンネルの 1 回分の A/D 変換が終了した場合に"1"がセットされます。このビットのクリアは、このレジスタ (ADCSR) をリード後、A/D データレジスタをリードするか、このビットに直接"0"を書き込むことにより行なわれます。</p> <p>A/D 変換データを読む場合は必ず、このビットが"1"であることを確認してからデータを読んでください。</p>
b6	ADIE	<p>A/D 変換終了割り込み許可</p> <p>A/D 変換終了時に、CPU に対し割り込み (INT) を発生するかどうかを選択します。"1"を書き込むと割り込み発生、"0"を書き込むと、割り込みを発生しません。</p>

ビット	記号	説明
b5	ADST	<p>A/D 変換スタート</p> <p>このビットに"1"を書き込むことによりA/D変換を開始します。単一モードの場合、"1"の書き込みにより変換をスタートし、変換中は"1"の状態が保持され、変換終了により"0"にクリアされます。</p> <p>スキャンモードの場合、このビットのセット("1")により、変換を開始し、"0"を書き込むまで変換を続行します。また変換中に"1"をライトすると、スキャンの最初のチャンネルから変換を行ないます。</p> <p>スキャンモードの場合、1回分の変換終了により、ビット7のADEND="1"になりますが、その間もA/D変換は続行しています。そのため、スキャンモードの場合は、A/D変換終了ビットが"1"にセットされてから、1チャンネル当りの変換時間(42 μsec) × スキャンチャンネルの時間以内にデータを読み終わらなければなりません。</p>
b4	SCAN	<p>スキャンモードの選択</p> <p>単一モード/スキャンモードの選択フラグです。"0"の時単一モード、"1"の時スキャンモードになります。リセット時は"0"となっています。</p>
b3 ~ b0	CH3 ~ CH0	<p>チャンネル選択</p> <p>アナログ入力チャンネル選択用のビットです。単一モードの場合、変換を行なうチャンネルをセットし、変換スタートビットを"1"にし、変換を開始します。変換終了時にデータがセットされるレジスタは、表 4-3 を参照してください。</p> <p>スキャンモードの場合、スキャンを行なうチャンネルを指定することになります。"0011"を書き込むと、ch0 ~ ch3 までのスキャンを選択したことになり、CH0 のデータはインデックス 07h、08h、CH1 は 09h、0Ah、CH2 は 0Bh、0Ch、CH3 は 0Dh、0Eh にデータがセットされます。詳しくは表 4-3 を参照してください。</p>

UPP の内蔵 A/D コンバータは、10 チャンネルのうち最大 4 チャンネルをスキャンモードとして使用することができます。但し、使用するチャンネルは、表 3-3 のようになります。

表 3-3.A/D 変換モードとデータレジスタ

モード	A/D コントロールレジスタ					A/D データレジスタ			
	SCAN	CH3	CH2	CH1	CH0	ADDR0	ADDR1	ADDR2	ADDR3
単一 モード	0	0	0	0	0	AN0	-	-	-
		0	0	0	1	-	AN1	-	-
		0	0	1	0	-	-	AN2	-
		0	0	1	1	-	-	-	AN3
		0	1	0	0	AN4	-	-	-
		0	1	0	1	-	AN5	-	-
		0	1	1	0	-	-	AN6	-
		0	1	1	1	-	-	-	AN7
		1	0	0	0	AN8	-	-	-
		1	0	0	1	-	AN9	-	-

モード	A/D コントロールレジスタ					A/D データレジスタ			
	SCAN	CH3	CH2	CH1	CH0	ADDR0	ADDR1	ADDR2	ADDR3
スキャン モード	1	0	0	0	0	AN0	-	-	-
		0	0	0	1	AN0	AN1	-	-
		0	0	1	0	AN0	AN1	AN2	-
		0	0	1	1	AN0	AN1	AN2	AN3
		0	1	0	0	AN4	-	-	-
		0	1	0	1	AN4	AN5	-	-
		0	1	1	0	AN4	AN5	AN6	-
		0	1	1	1	AN4	AN5	AN6	AN7
		1	0	0	0	AN8	-	-	-
		1	0	0	1	AN8	AN9	-	-

インデックス [07h-0Eh] R/W

A/D データレジスタ (ADDR0-ADDR3)

A/D 変換の結果がセットされます。A/D データレジスタは2バイト構成で分解能8ビットで使用することを考慮して、10ビットの A/D 変換データは上位8ビットが ADDR の上位バイトに、下位2ビットが ADDR の下位バイトの上位2ビットにセットされます。

インデックス	レジスタ名 と 記号	B7	B6	B5	B4	B3	B2	B1	B0	
07h	ADDR0	AD Data Register 0 (H)	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
08h		AD Data Register 0 (L)	AD1	AD0	-	-	-	-	-	-
09h	ADDR1	AD Data Register 1 (H)	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0Ah		AD Data Register 1 (L)	AD1	AD0	-	-	-	-	-	-
0Bh	ADDR2	AD Data Register 2 (H)	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0Ch		AD Data Register 2 (L)	AD1	AD0	-	-	-	-	-	-
0Dh	ADDR3	AD Data Register 3 (H)	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
0Eh		AD Data Register 3 (L)	AD1	AD0	-	-	-	-	-	-

単一モードでは、A/D 変換が終了すると所定の ADDR に変換データがセットされます。スキャンモードでは上位バイトに続いて下位バイトをリードするまで、データは更新されません。各モードで 10 ビットの変換データを得るためのプログラムを下記に示します。

=>単一モードの場合の実行例

```

WORD DataLow, DataHigh, AdData;
double Volt;

ADRS = 0x300; /* ベースアドレス */
ADSTS = 0x06; /* ステータスレジスタインデックス */
CH = 1; /* チャンネル番号 */
outp(ADRS+3,0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2,ADSTS); /* アドレスレジスタ下位指定 */
outp(ADRS,0); /* 変換ストップ */
outp(ADRS,0x20+CH); /* CH=1 変換スタート 単一モード */
while((inp(ADRS)&0x80) == 0) /* 変換終了チェック */
;
outp(ADRS+3,0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2,0x07+CH*2); /* アドレスレジスタ下位指定 */
DataHigh = inp(ADRS) /* 上位 8 ビットリード */
outp(ADRS+3,0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2,0x08+CH*2); /* アドレスレジスタ下位指定 */
DataLow = inp(ADRS) /* 下位 2 ビットリード */
AdData = (DataHigh << 2) | (DataLow >> 6) & 0x3FF; /* 10 ビットデータを得る */
Volt = ((double)AdData)*5.0/1024.0; /* 電圧に変換 */
printf ("volt : %3.5f¥n", Volt); /* 電圧を表示 */
    
```

=>スキャンモードの場合の実行例

```
WORD DataLow, DataHigh, AdData;
double Volt;

ADRS = 0x300; /* ベースアドレス */
outp(ADRS+0x03,00); /* アドレスレジスタ上位指定 */
outp(ADRS+0x02,0x06); /* アドレスレジスタ下位指定 */
outp(ADRS,0); /* 変換ストップ */
outp(ADRS,0x33); /* スキャンモード変換スタート */
while(1){
    while((inp(ADRS)&0x80)==0) /* 終了チェック */
        ;
    for ( ch = 0;ch<=3;ch++){
        outp(ADRS+0x03,00); /* アドレスレジスタ上位指定 */
        outp(ADRS+0x02,0x07+ch*2); /* アドレスレジスタ下位指定 */
        DataHigh = inp(ADRS); /* 上位 8 ビット入力 */
        outp(ADRS+0x03,00); /* アドレスレジスタ上位指定 */
        outp(ADRS+0x02,0x08+ch*2); /* アドレスレジスタ下位指定 */
        DataLow = inp(ADRS); /* 下位 2 ビット入力 */
        AdData = (DataHigh <<2)+( DataLow >>6); /* 10 ビット入力 */
        Volt = ((double) AdData)*5.0/4096.0; /* 電圧に変換 */
        printf ("volt %3.5f¥n", Volt); /* 電圧を表示 */
    }
}
```


=>デジタル入出力の設定例

16ビットの外部入出力ラインをすべてデジタル入出力として使用し、ポート2を入力、ポート1を出力として設定します。

```

ADRS = 0x300; /* REX-5059 データベースアドレスの設定 */
DDR2 = 0; /* データ方向レジスタ 2 */
DDR1 = 1; /* データ方向レジスタ 1 */
PORT2 = 2; /* デジタル入出力ポート 2 */
PORT1 = 3; /* デジタル入出力ポート 1 */
UPP_CE2 = 0x10; /* UPP コンタクトイネーブルレジスタ 2 */
UPP_CE1 = 0x11; /* UPP コンタクトイネーブルレジスタ 1 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, UPP_CE2); /* アドレスレジスタ下位指定 */
outp(ADRS,0x00); /* デジタル入出力ポート 2 を使用 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, UPP_CE1); /* アドレスレジスタ下位指定 */
outp(ADRS,0x00); /* デジタル入出力ポート 1 を使用 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, DDR2); /* アドレスレジスタ下位指定 */
outp(ADRS,0x00); /* ポート 2 を入力として使用 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, DDR1); /* アドレスレジスタ下位指定 */
outp(ADRS,0xff); /* ポート 1 を出力として使用 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, PORT2); /* アドレスレジスタ下位指定 */
Val = inp(ADRS) /* ポート 2 より 8 ビットデータ入力 */

outp(ADRS+3, 0x00); /* アドレスレジスタ上位指定 */
outp(ADRS+2, PORT1); /* アドレスレジスタ下位指定 */
outp(ADRS,Val); /* ポート 1 へ入力したデータを出力 */

```


(3-5-2) 割り込み制御用レジスタの構成

インデックス [2Ah-2Ch] R/W

割り込みイネーブルレジスタ (IER3-IER1)

割り込み要求を許可するレジスタです。ISR3～1の対応するビットが"1"の場合、ISRが"1"になった時に割り込みが発生します。0を書き込むと、割り込み要求は禁止されます。

インデックス	レジスタ名と記号	B7	B6	B5	B4	B3	B2	B1	B0
2Ah	IER3 Interrupt Enable Register 3	IRE23	IRE22	IRE21	IRE20	IRE19	IRE18	IRE17	IRE16
2Bh	IER2 Interrupt Enable Register 2	IRE15	IRE14	IRE13	IRE12	IRE11	IRE10	IRE9	IRE8
2Ch	IER1 Interrupt Enable Register 1	IRE7	IRE6	IRE5	IRE4	IRE3	IRE2	IRE1	IRE0

インデックス [2Dh-2Fh] R/W

割り込みリクエストレジスタ (IRQR3-IRQR1)

IERとISRのANDを取った値が入っています。実際に割り込みを使用する場合、サービスルーチンではこの値を調べ、どのビットが立っているかにより、サービスを行いません。

インデックス	レジスタ名と記号	B7	B6	B5	B4	B3	B2	B1	B0
2Dh	IRQR3 Interrupt Request Register3	IRR23	IRR22	IRR21	IRR20	IRR19	IRR18	IRR17	IRR16
2Eh	IRQR2 Interrupt Request Register2	IRR15	IRR14	IRR13	IRR12	IRR11	IRR10	IRR9	IRR8
2Fh	IRQR1 Interrupt Request Register1	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0

インデックス [33h-35h] W

割り込みステータスクリアレジスタ (ISCR3-ISCR1)

ISR3～1をクリアするためのレジスタです。サービスルーチンでは所定の動作を行なった後、割り込み要求元となったISRに対応するISCRに"0"を書き込みます。

インデックス	レジスタ名と記号	B7	B6	B5	B4	B3	B2	B1	B0
33h	ISCR3 Interrupt Clear Register 3	ISC23	ISC22	ISC21	ISC20	ISC19	ISC18	ISC17	ISC16
34h	ISCR2 Interrupt Clear Register 2	ISC15	ISC14	ISC13	ISC12	ISC11	ISC10	ISC9	ISC8
35h	ISCR1 Interrupt Clear Register 1	ISC7	ISC6	ISC5	ISC4	ISC3	ISC2	ISC1	ISC0

(3-6) その他の機能

(3-6-1) ウォッチドッグ部

REX-5059 では、UPP の有するウォッチドッグ機能のタイマ出力端子を入出力ボックスコネクタの 24 に出力しています。外部に対し、ある一定時間以上プログラムが停止していることを通知する場合などに使用することができます。

通常では、使用することはありません。詳しくは、添付の UPP データシート頁 24 を参照してください。

(3-6-2) その他特殊使用について

ネクストデータイネーブルレジスタ及びネクストデータレジスタにより、ある特定タイミングでポートにデータを出力することができます。

インデックス [14h] W

ネクストデータイネーブルレジスタ (NDR)

NDR の値をポート 2 に出力するか否かを 1 ビット単位で指定します。

“0” の場合は出力しない、“1” の場合は出力可能となります。もちろんコネクタの入出力コネクタにポート 2 の値を出力するには、UPP コンタクトイネーブルレジスタ 2 の値が“0”で、データ方向には“1”が書き込まれていなくてはなりません。

インデックス [16h] W

ネクストデータレジスタ (NDR)

NDR の対応するビットが“1”の場合に、ポート 2 にデータが出力されます。出力のタイミングは、UOR2 のビット 0 が 1 0 に変化した時です。つまり、U8 がパルス入出力動作により、1 0 になった時です。

(空白ページ)

第4章 Windows95/98/Me解説

(4-1) Windows95 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

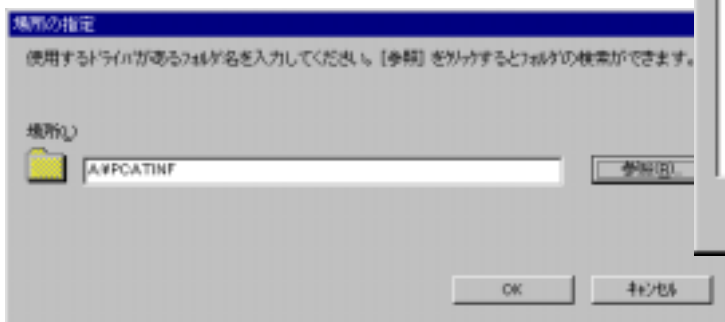
次にドライバーファイル (INF ファイル) の場所を指定します。PC/AT にインストールされる場合は、

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完了した場合は、「このデバイス用に更新されたドライバーが見つかりました。」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

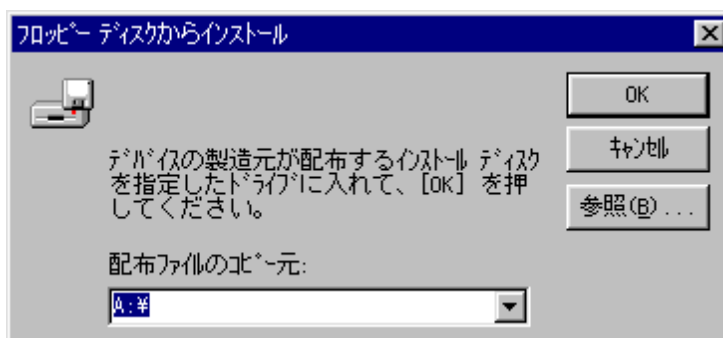
【1】PC カードをカードスロットへ挿入

カードを挿入すると、ハードウェアウィザードが自動的に起動し「新しいハードウェア」のダイアログが表示されます。「ハードウェアの製造元が提供するドライバ」を選択し「OK」をクリックします。



【2】ドライバディスクのセットアップ

UPP PC カードの添付ディスクを FD ドライブに挿入し、「配布ファイルのコピー元」ドライブ名を指定します。



【3】カードモデル名の選択

デバイスの選択から、DOS/V 機で使用するのであれば

☑ 「REX5059 UPP PC CARD For DOS/V」

PC-9800 シリーズで使用するのであれば

☑ 「REX5059 UPP PC CARD For PC-98」を指定します。

カードの設定が正常に終了すると、「ピッポッ」というビープ音とともに、ハードウェアウィザードによる自動設定が終了します。



(4-2) Windows98 インストール

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が表示されます。ここでは、次へを押します。



ドライバの検索方法は「特定の場所にあるすべてのドライバの一覧を作成し、インストールするドライバを選択する。」を選択し、次へを押します。

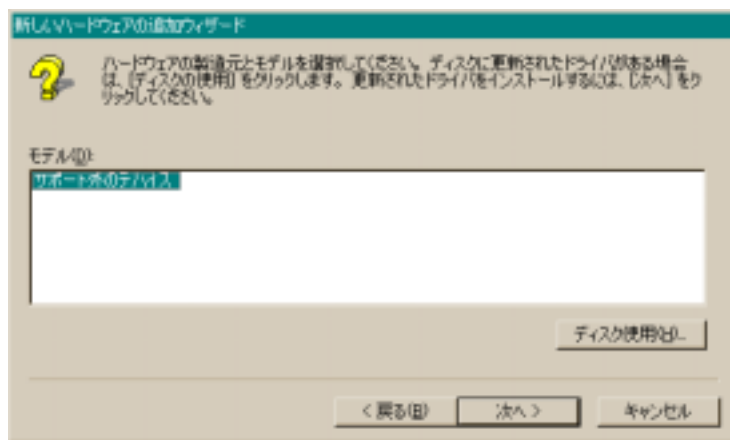


デバイスの種類は「その他のデバイス」または「Otherdevices」を選択し、次へを押します。



【2】ドライバーファイル場所の指定

モデルの選択では「ディスク使用」を押します。



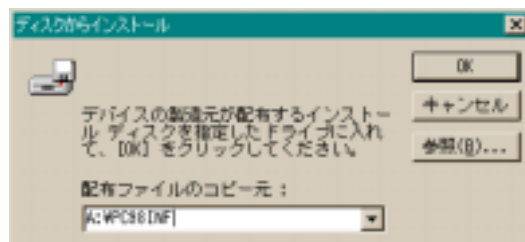
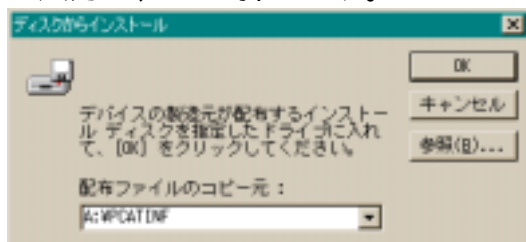
製品添付の Windows95/98/Me セットアップディスクをフロッピーディスクドライブに挿入し、次にドライバーファイル(INF ファイル)の場所を指定します。PC-AT にインストールされる場合は、

A:¥PCATINF

PC-98 にインストールされる場合は、

C:¥PC98INF

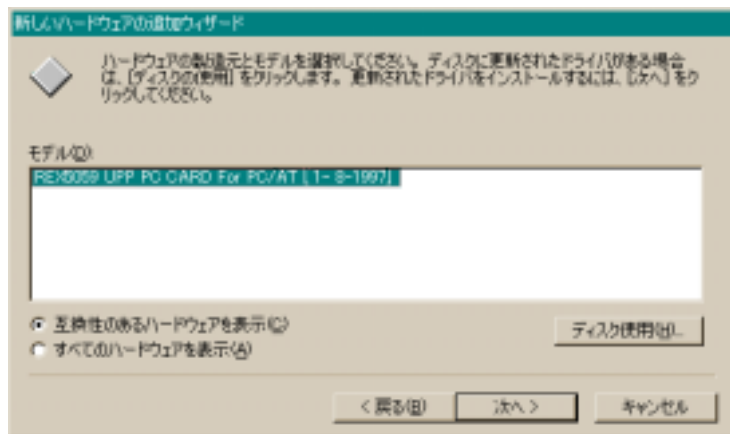
と入力し、OK を押します。



正しいモデル名「REX5059 UPP PC CARD for PC/AT」が表示されたら、次へを押します。

(注意)

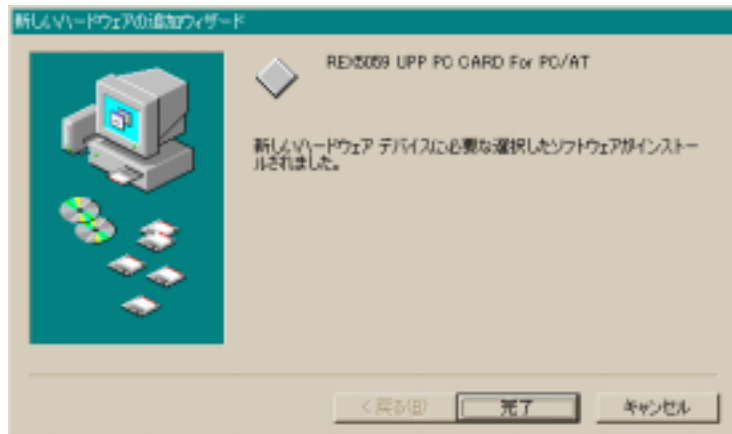
NEC PC9821 シリーズをご利用の場合は、for PC/AT の表示が for PC-98 になっていることを確認します。



インストール準備が完了したら、次へを押します。



インストール完了が表示されたら、完了を押してハードウェアウィザードを終了します。



(4-3) WindowsMe インストール

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。

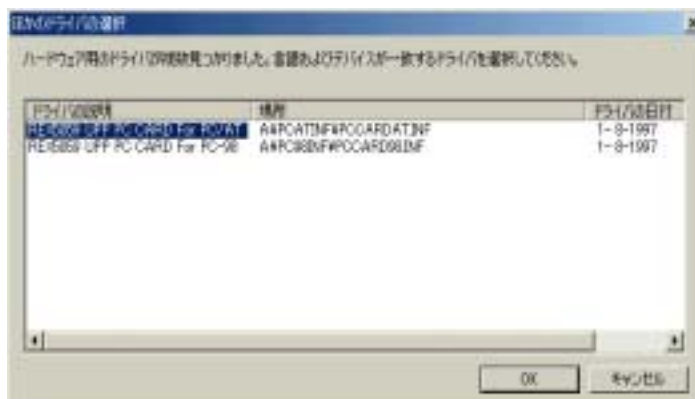
次に、右の画面で「適切なドライバを自動的に検索する（推奨）(A)」を選択し「次へ」ボタンを押します。



【2】ドライバーファイル場所の指定

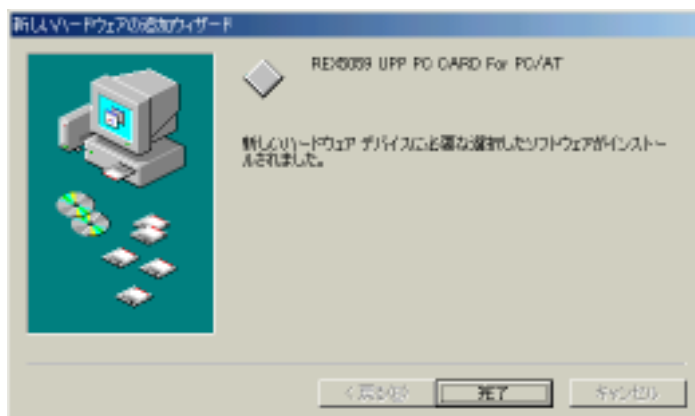
右のようにセットアップ情報ファイル(.inf ファイル)が自動的に検索されますので、

「REX-5059 UPP PC CARD 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-5059 インストールは終了です。



(4-4) インストール内容の確認

=> システムプロパティの起動

次にカードのセットアップが正常に完了しているか調べます。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「REX5059 UPP PC CARD For PC/AT(または PC-98)」が登録されます。



=> リソース情報の取得

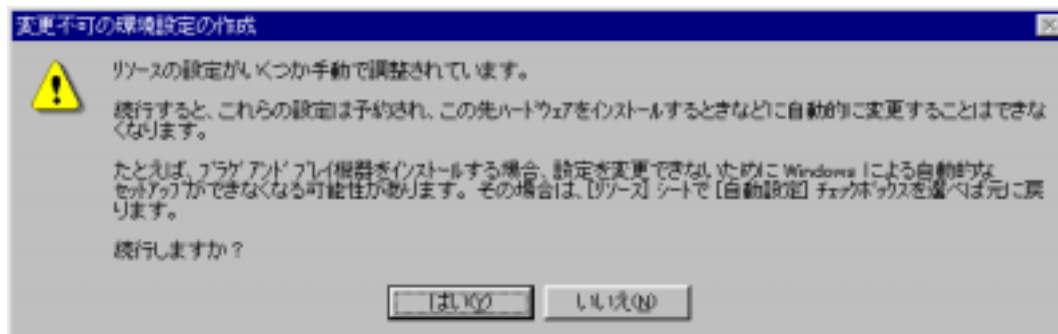
レジストリ「REX5059 UPP PC CARD For PC/AT(または PC-98)」をダブルクリックし、プロパティを表示します。リソースタブを選択し自動設定でカードに割り当てられた I/O アドレスと割り込み番号を調べます。同時に、競合するデバイスが「競合なし」になっていることを確認してください。他のデバイスと競合している場合は、自動設定のチェックをはずし、リソースを変更してください。



=> リソースの変更

リソースの変更は、自動設定を解除して、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」というピ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「REX5059 UPP PC CARD For PC/AT(または PC-98)」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。



=> I/O アドレスの確認

カードに割り当てた I/O アドレスにアクセスできるかどうか MS-DOS のユーティリティ”DEBUG.EXE”または”SYMDEB.EXE”を使って確認します。カードが使用する I/O アドレスは、ベースアドレスから連続した 4 バイトの範囲です。下記のように指定した値(4, 8)が返ってくれば、カードの設定は正常に行われています。(下記はカードに割り当てられた I/O ベースアドレスが 0120 - 0123 のときの例です。)なお、確認の際は必ずコネクタボックスを接続して確認作業を行ってください。

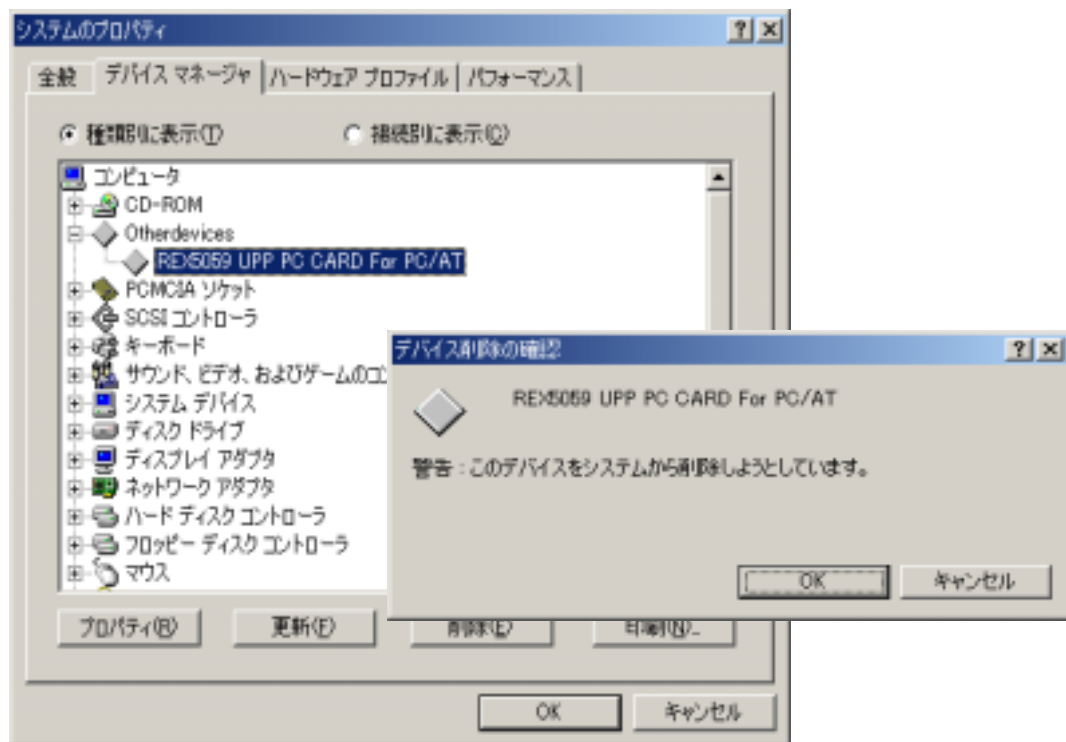
```
C:¥WINDOWS > DEBUG
-o 123 0
-o 122 40
-o 120 4
-o 123 0
-o 122 41
-o 120 8
-o 122 40
-i 120
04
-o 122 41
-i 120
08
-q
C:¥WINDOWS >
```

(4-5) 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等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、Page4-1 よりインストールを行って下さい。

【1】カード情報の削除

「コントロールパネル」から「システム」を起動し、「デバイスマネージャ」のタブを選択します。Otherdevices にある「REX5059 UPP PC CARD For PC/AT(または PC-98)」を選択し、「削除」ボタンを押します。確認のダイアログが表示されますので「OK」ボタンを押してください。



[2] INF ファイルの削除

「エクスプローラ」を開いて¥Windows¥Inf¥Other フォルダにある「RATOC System,Inc.PCCARDAT.INF」を削除して下さい。



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 -5059 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「表示」から「フォルダオプション」を選択し、表示タブ内の詳細設定で、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。

(4-6) DLL ライブラリ解説

(4-6-1) DLL ライブラリについて

Microsoft Visual C++ 5.0 で作成したダイナミックリンクライブラリ(DLL)が添付されています。UPP カードに入出力を行うための関数が提供されています。

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ライブラリ UppLib32.lib をプロジェクトに組み込み、各関数のインポート宣言を行います。

Visual BASIC 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モジュール定義ファイル(.BAS)で Declare 宣言します。

ライブラリ名	UPPLIB32.DLL (32Bit Version) UPPLIB32.LIB (プロジェクトファイルへ追加)
仮想デバイスドライバ	VR5059D.VXD
インクルードファイル	UPPLIB32.H

(注記)

- 1.アプリケーション実行時、DLL ファイルと VXD ファイルを 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。

=> インポート宣言 (Visual C/C++)

```
DllImport BOOL APIENTRY UppStartEventSynclnt( HWND, WORD, WORD, WORD, DWORD );
DllImport BOOL APIENTRY UppEndEventSynclnt( void );
DllImport BOOL APIENTRY UppGetCardResource( HWND, WORD, LPWORD, LPWORD );
DllImport void APIENTRY UppGetVersion( HWND );
DllImport VOID APIENTRY WaitMilliseconds( DWORD );
DllImport VOID APIENTRY ShowCardUtil( HWND );
DllImport void APIENTRY OutUpp( WORD, WORD, BYTE );
DllImport WORD APIENTRY InUpp( WORD, WORD );
DllImport void APIENTRY OutPort( WORD, WORD );
DllImport void APIENTRY wOutPort( WORD, WORD );
DllImport WORD APIENTRY InPort( WORD );
DllImport WORD APIENTRY wInPort( WORD );
DllImport VOID APIENTRY UppCardReset( WORD );
```

尚、インポート宣言 `DllImport` は下記のように定義されています。

```
#define DllImport _declspec( dllimport )
```

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

```
Declare Sub OutUpp Lib "UPPLIB32.DLL" (ByVal wBase As Integer, ByVal Reg As Integer, ByVal OutVal As Integer)
Declare Function InUpp Lib "UPPLIB32.DLL" (ByVal wBase As Integer, ByVal RegAddr As Integer) As Long
Declare Sub OutPort Lib "UPPLIB32.DLL" (ByVal IOAddr As Integer, ByVal OutVal As Integer)
Declare Sub wOutPort Lib "UPPLIB32.DLL" (ByVal IOAddr As Integer, ByVal OutVal As Integer)
Declare Function InPort Lib "UPPLIB32.DLL" (ByVal IOAddr As Integer) As Long
Declare Function wInPort Lib "UPPLIB32.DLL" (ByVal IOAddr As Integer) As Long
Declare Sub UppGetVersion Lib "UPPLIB32.DLL" (ByVal hWnd As Integer)
Declare Sub WaitMilliseconds Lib "UPPLIB32.DLL" (ByVal DWORD As Long)
Declare Function UppGetCardResource Lib "UPPLIB32.DLL" (ByVal hDlg As Long, ByVal SlotNo As Integer, IOAdrs As Integer, IrqNo As Integer) As Long
Declare Sub ShowCardUtil Lib "UPPLIB32.DLL" (ByVal hWnd As Long)
```

=> DLL 関数仕様

UppGetCardResource

リソース情報の取得

書式	BOOL APIENTRY UppGetCardResource (HWND hWnd, WORD SlotNo, LPWORD pIOBase, LPWORD plrqNo)	
機能	指定スロットに挿入されているカードが REX-5059 UPP PC カードなら、現在割り当てられている I/O ベースアドレスおよび IRQ 番号情報を返します。	
引数	HWND	hWnd : 呼び出し元ウィンドウハンドル
	WORD	SlotNo : サーチするカード挿入スロット番号
	LPWORD	pIOBase : (出力)I/O リソース情報を格納する変数のアドレス
	LPWORD	plrqNo : (出力)IRQ リソース情報を格納する変数のアドレス
戻値	0 : 正常終了 -1 : DEVICE I/O コントローラ -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

カード情報ユーティリティを表示

書式	void APIENTRY ShowCardUtil(HWND hWnd)
機能	PCカード情報ユーティリティダイアログを表示します。
引数	HWND hWnd : 呼び出し元ウィンドウハンドル
戻値	なし

UppGetVersion

DLL のバージョンダイアログ表示

書式	void APIENTRY UppGetVersion(HWND hWnd)
機能	DLL のバージョンダイアログを呼び出します
引数	HWND hWnd : 呼び出し側のウィンドウハンドル
戻値	なし

OutUpp

UPP ポートに1バイトを出力

書式 void APIENTRY OutUpp(WORD wBase, WORD RegIndex, BYTE Val)

機能 1バイトをポートに出力

引数 WORD wIOBase : カードのベースアドレス
WORD UppIndex : UPP レジスタインデックス
BYTE Val : バイト出力値

戻値 なし

解説 UPP のレジスタへ出力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し出力を行います。DLL ライブラリにあるポートへの出力命令を使うと下記コーディングになります。

OutPort (ベースアドレス + 3, インデックス上位バイト);

OutPort (ベースアドレス + 2, インデックス下位バイト);

OutPort (ベースアドレス + 0, 出力データ);

関数 OutUpp() を使うことにより下記のように1行で記述することができます。

OutUpp (ベースアドレス, インデックス, 出力データ);

InUpp

UPP ポートから1バイト入力

書式 WORD APIENTRY InUpp(WORD wBase, WORD RegIndex)

機能 ポートから1バイト読み込む

引数 WORD IOBase : カードのベースアドレス
WORD RegIndex : UPP レジスタインデックス

戻値 バイト入力値

解説 UPP のレジスタへ入力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し入力を行います。DLL ライブラリにあるポートへの入力命令を使うと下記のコーディングを行う必要があります。

OutPort (ベースアドレス + 3, インデックス上位バイト);

OutPort (ベースアドレス + 2, インデックス下位バイト);

入力データ = InPort (ベースアドレス);

関数 InUpp() を使うことにより下記のように1行で記述することができます。

入力データ = InUpp (ベースアドレス, インデックス);

OutPort **ポートに1バイトを出力**

- 書式 void APIENTRY **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 なし

wOutPort **ポートに1ワードを出力**

- 書式 void APIENTRY **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 なし

InPort **ポートから1バイト入力**

- 書式 WORD APIENTRY **InPort** (WORD IOAddr)
- 機能 ポートから1バイト読み取ります
- 引数 WORD IOAddr : 出力する I/O ポートアドレス
- 戻値 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 **ポートから1ワード入力**

- 書式 WORD APIENTRY **wInPort** (WORD IOAddr)
- 機能 ポートから1ワード読み取ります
- 引数 WORD IOAddr : 出力する I/O ポートアドレス
- 戻値 読み込んだワードデータを返します

UppCardReset

REX-5059 をリセット

- 書式 VOID UppCardReset(WORD SlotNo)
- 機能 CCOR にアクセスして UPP カードをリセットします。
- 引数 WORD SlotNo : ソケット番号
- 戻値 なし

UppStartEventSyncInt

ユーザ定義メッセージ割り込みの開始

- 書式 BOOL UppStartEventSyncInt(HWND hWnd, WORD MyIOBase, WORD MyIrqNo, WORD ISCReg, DWORD StopCount)
- 機能 VPICD に割り込み登録を行い、割り込みに同期して VxD ハンドラからユーザアプリケーションにユーザ定義メッセージ WM_VXDEVENT 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
WORD UseIOBase : 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 ISCReg : 割り込みステータスクリアレジスタ指定
DWORD StopCount : 割り込み終了回数(0 指定で無限に繰り返す)
- 戻値
0 : 正常終了
-1 : 割り込み登録エラー
-2 : I/O コントロールのエラー
-3 : 割り込みクリアレジスタ指定エラー

UppEndEventSyncInt

割り込みを解除

- 書式 BOOL UppEndEventSyncInt(void)
- 機能 VPICD に登録されている割り込みを解除します。
- 引数 なし
- 戻値 常に 0 を返す

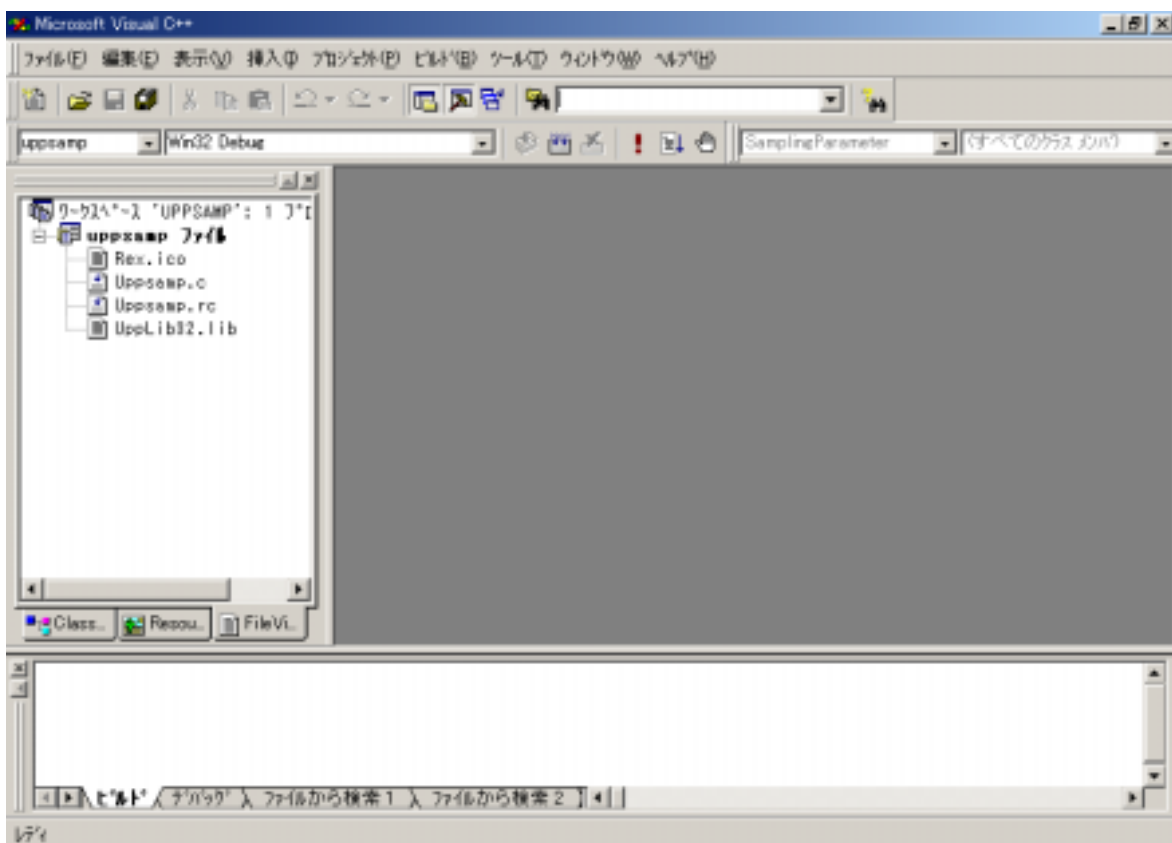
(4-6-2) Visual C/C++アプリケーション作成

=>ファイルのコピー

本製品添付のディスクの内容をサブディレクトリーを含め、全てハードディスクにコピーします。

=>プロジェクトの作成

Visual C++ 5.0 または 6.0 を起動後、ファイルの新規作成からプロジェクトタイプを "Win32 Application" にして [空のプロジェクトワークスペース] を作成します。次に、[プロジェクト] の [プロジェクトへ追加] [ファイル] を選択し、"Uppsamp.c"・"Uppsamp.rc"・"UppLib32.lib" をプロジェクトに追加します。プロジェクトのファイル構成が下図のようになっていることを確認してください。最後に、プロジェクトのビルドを実行します。



=>サンプルプログラムの実行

サンプルプログラムには、

実行例 1 : ロータリーエンコーダからの2相パルス入力アップダウンカウント

実行例 2 : A/D 変換

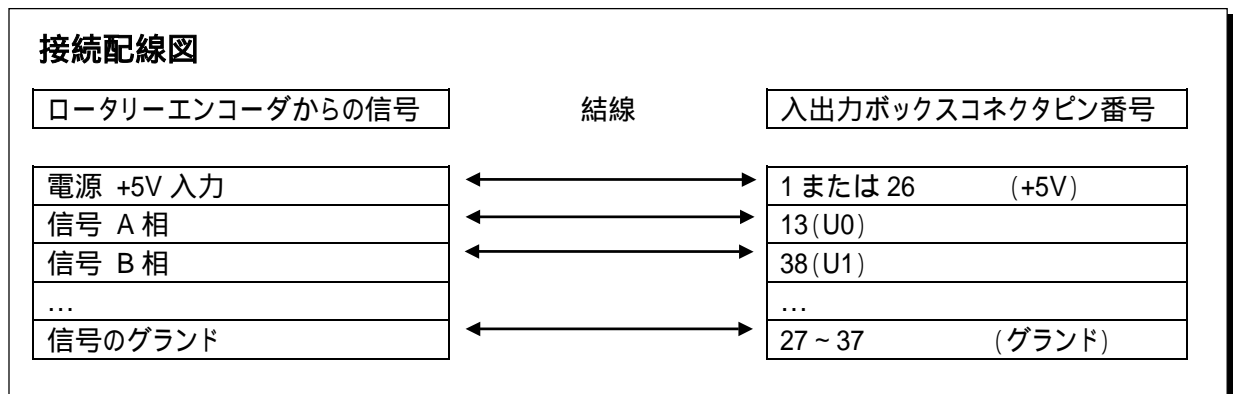
実行例 3 : FFC インターバル割り込み

があります。ソースコードは添付のディスクを参照してください。

以下に**実行例 1、2**の解説をいたします。

実行例 1

ロータリーエンコーダからの入力例を実行する場合の配線図及び実行画面を下図に示します。



実行画面



田 サンプルプログラム抜粋

```
BOOL APIENTRY DlgProcEncoder(HWND hDlg, UINT message, UINT wParam, LONG lParam)
{
    WORD SlotNo;

    switch( message )
    {
    case WM_INITDIALOG:
        /* スロ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 */
        for( SlotNo = 0; SlotNo < 2; SlotNo++ )
            if ( UppGetCardResource( hDlg, SlotNo, &Param.IOAdrs, &Param.IrqNo ) == 0 )
                break;
        // インターバルタイマーを 100msec に設定
        Param.IntervalTime = 100U;
        return TRUE ;
    case WM_TIMER:
        DispEncoderAngle( hDlg, Param.IOAdrs );
        return TRUE ;
    case WM_COMMAND:
        switch( wParam )
        {
        case IDOK:
            if ( GetEncoderParam( hDlg ) != FALSE )
            {
                /* UPP 設定 */
                UPPInit( Param.IOAdrs );
                SetFuncTbIForEN( Param.IOAdrs );
                SetUPPDataRegForEN( Param.IOAdrs );
                /* インターバルタイマーを起動 */
                SetTimer(hDlg, ID_MYTIMER, Param.IntervalTime, NULL);
            }
            return TRUE;
        case IDCANCEL:
            /* UPC 停止(GFE=0) */
            OutUpp( Param.IOAdrs, USCR, 0 );
            /* タイマー停止 */
            KillTimer(hDlg, ID_MYTIMER);
            /* リソース解放 */
            DeleteObject( hBr );
            DeleteObject( hBl );
            EndDialog( hDlg, FALSE );
            return TRUE ;
        default:
            return TRUE ;
        }
        break ;
    }
    return FALSE ;
}
```

➤ 入力パラメータチェック

```
BOOL GetEncoderParam( HWND hDlg )
{
    // I/Oアドレスレジスタから入力文字列を取得
    GetDlgItemText( hDlg, IDC_EDIT_IOADRS, szTmp, sizeof(szTmp) );
    Param.IOAdrs = AscHexToInt( szTmp );

    // 実行時間レジスタから入力文字列を取得
    GetDlgItemText( hDlg, IDC_EDIT_TIME, szTmp, sizeof(szTmp) );
    Param.IntervalTime = (WORD)atoi( szTmp );

    return TRUE;
}
```

➤ TPC ファンクションテーブルの内容をレジスタに設定

```
void SetFuncTblForEN( WORD IOAdrs )
{
    short      FuncNo, RegNo;

    OutUpp( IOAdrs, DDR2, 0x00 );      // Data Direction Reg.2 ALL INPUT
    OutUpp( IOAdrs, DDR1, 0x00 );      // Data Direction Reg.1 ALL INPUT
    OutUpp( IOAdrs, UCER2, 0xFF );     // UPP Contact Enable Reg.2 Pulse I/O Enable
    OutUpp( IOAdrs, UCER1, 0xFF );     // UPP Contact Enable Reg.1 Pulse I/O Enable
    OutUpp( IOAdrs, MFNR, 20 );        // Maximum Function Number Reg. (16 Function 5us)

    for( FuncNo = 0; FuncNo <= 15; FuncNo++ )
    {
        for( RegNo = 0; RegNo <= 7; RegNo++ )
        {
            OutUpp( IOAdrs, (WORD)(FNR + RegNo), (BYTE)TPCtbl[ FuncNo ][ RegNo ] );
        }
    }
}
```

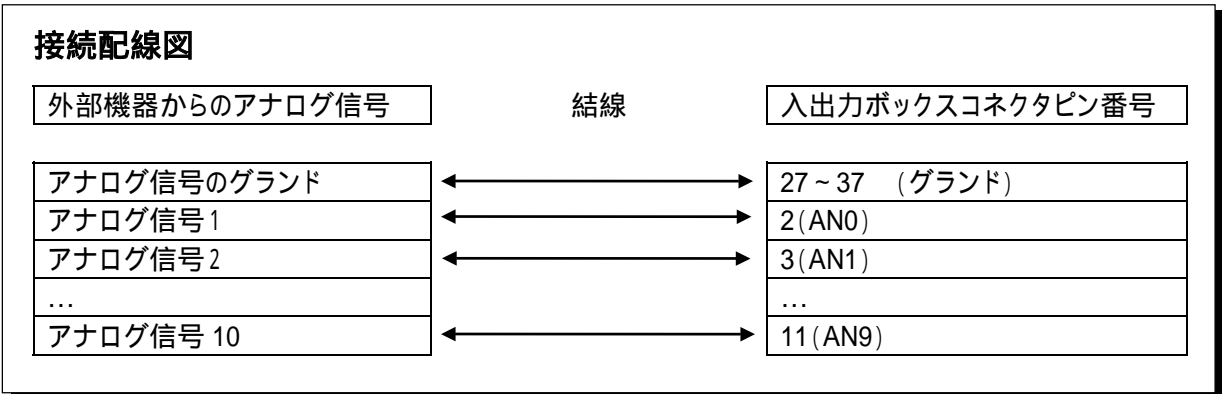
➤ UPP データレジスタクリア

```
void SetUPPDataRegForEN( WORD IOAdrs )
{
    WORD UdrNo;

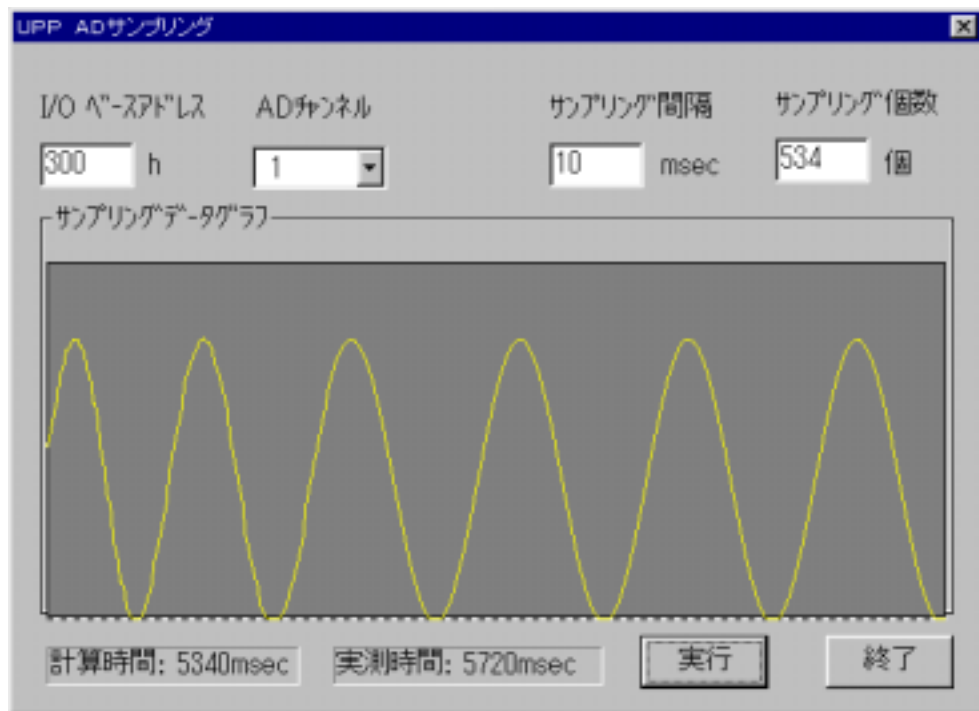
    // UPP Data Reg.0 クリア
    OutUpp( IOAdrs, UDROH, 0 );
    OutUpp( IOAdrs, UDR0L, 0 );
    // UPP Data Reg.1 から 23 をクリア
    for( UdrNo = 1; UdrNo <= 7; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDROH + UdrNo * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDROL + UdrNo * 2 + 1), 0 );
    }
    for( UdrNo = 8; UdrNo <= 15; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDR8H + (UdrNo - 8) * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDR8L + (UdrNo - 8) * 2 + 1), 0 );
    }
    for( UdrNo = 16; UdrNo <= 23; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDR16H + (UdrNo - 16) * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDR16L + (UdrNo - 16) * 2 + 1), 0 );
    }
}
```

実行例 2

A/D 変換入力の実行する場合の配線図及び実行画面を下図に示します。



実行画面



田 サンプルプログラム抜粋

```

BOOL APIENTRY DlgProcAD( HWND hDlg, UINT message, UINT wParam, LONG lParam )
{
    WORD uiMemSize;                // アプリケーションに必要なメモリーサイズ

    switch( message )
    {
    case WM_INITDIALOG:
        /* ダイアログコントロールの初期化 */
        InitADSampParam( hDlg );
        return TRUE ;
    case WM_COMMAND:
        switch( wParam )
        {
        case IDOK:
            if ( GetADSampParam( hDlg ) != FALSE )
            {
                // サンプリングデータ格納領域確保
                // 必要なメモリーサイズ = サンプリング回数 × データサイズ
                // × サンプリングチャンネル数
                uiMemSize = Param.SampCnt * (WORD) sizeof(WORD) * 1U;
                if ( AllocLocalHeapMemory( hDlg, uiMemSize ) != TRUE )
                    return TRUE;

                /* UPP 設定 */
                UPPInit( Param.IOAdrs );

                /* AD 変換実行 */
                SetFuncTblForAD( Param.IOAdrs );
                SetUPPDataRegForAD( Param.IOAdrs, Param.Freq );
                StartADSamp( hDlg, Param.IOAdrs, Param.SampCnt,
                            Param.Freq, Param.Chan, fpLocalMemory );
                LocalUnlock( hLocalMemory );
                LocalFree( hLocalMemory );
                SetCursor( hSaveCursor );           // カーソル復帰
            }
            return TRUE;
        case IDCANCEL:
            DeleteObject( hBI );
            EndDialog( hDlg, FALSE );
            return TRUE ;
        default:
            return TRUE ;
        }
        break ;
    }
    return FALSE ;
}

```

➤ 各コントロールの初期化

```
void InitADSampParam( HWND hDlg )
{
    SHORT      GraphLength;    // グラフにプロットできる点の最大数
    SHORT      i;
    WORD       SlotNo;

    /* 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する */
    for( SlotNo = 0; SlotNo < 2; SlotNo++ )
        if ( UppGetCardResource( hDlg, SlotNo, &Param.IOAdrs, &Param.IrqNo ) == 0 )
            break;

    // 自動取得した I/O アドレスの表示
    sprintf( szTmp, "%x", Param.IOAdrs );
    SetDlgItemText( hDlg, IDC_EDIT_IOADRS, szTmp );

    // AD 変換チャンネルのコンボボックスへのアイテムの送り込み
    for ( i = 0; ChanRange[i] != 0; i++ )
        SendDlgItemMessage( hDlg, IDC_COMB_CHANNEL, CB_ADDSTRING,
                            0, (LPARAM)((LPCSTR)ChanRange[i]) );

    // デフォルト値項目を表示
    Param.Chan = 0;
    SendDlgItemMessage( hDlg, IDC_COMB_CHANNEL, CB_SETCURSEL, Param.Chan, 0L );

    // サンプリング周期のエディットボックスを 10 msec で初期化
    Param.Freq = 10;
    sprintf( szTmp, "%u", Param.Freq );
    SetDlgItemText( hDlg, IDC_EDIT_FREQ, szTmp );

    // サンプリング個数はデフォルト値:GraphLength を設定する
    GraphLength = (short)(pmax.x - pmin.x);
    Param.SampCnt = (WORD)GraphLength;
    sprintf( szTmp, "%u", Param.SampCnt );
    SetDlgItemText( hDlg, IDC_EDIT_COUNT, szTmp );
}
```

➤ 設定された I/O アドレスと電圧範囲を取得

```

BOOL GetADSampParam( HWND hDlg )
{
    short          GraphLength;    // グラフにプロットできる点の最大数

    // I・O アドレスエディットボックスから入力文字列を取得
    GetDlgItemText( hDlg, IDC_EDIT_IOADRS, szTmp, sizeof(szTmp) );
    Param.IOAdrs = AscHexToInt( szTmp );

    // UPP PC カード A/D 信号入力チャンネル番号の取得
    Param.Chan = (short)SendDlgItemMessage( hDlg, IDC_COMB_CHANNEL, CB_GETCURSEL, 0, 0L ) + 1;

    // サンプリング周期エディットテキストから入力文字列を取得
    GetDlgItemText( hDlg, IDC_EDIT_FREQ, szTmp, sizeof(szTmp) );
    Param.Freq = (WORD)atoi( szTmp );

    // サンプリング個数エディットテキストから入力文字列を取得
    GetDlgItemText( hDlg, IDC_EDIT_COUNT, szTmp, sizeof(szTmp) );
    Param.SampCnt = (WORD)atoi( szTmp );
    GraphLength = (short)(pmax.x - pmin.x);

    return TRUE;
}

```

➤ UPP 初期化処理

```

void UPPInit( WORD IOAdrs )
{
    OutUpp( IOAdrs, USCR, 0x00 );    // U P C 停止
    OutUpp( IOAdrs, IER3, 0x00 );    // 割り込みマスク(UPP IER3)
    OutUpp( IOAdrs, IER2, 0x00 );    // 割り込みマスク(UPP IER2)
    OutUpp( IOAdrs, IER1, 0x00 );    // 割り込みマスク(UPP IER1)
    OutUpp( IOAdrs, UOR2, 0x00 );    // UPP Output Reg.2 Output Clear
    OutUpp( IOAdrs, UOR1, 0x00 );    // UPP Output Reg.1 Output Clear
    OutUpp( IOAdrs, MFNR, 1 );        // Maximum Function Number Reg. (0 Function)
    OutUpp( IOAdrs, FNR, 0x00 );     // Function Number Reg. CLEAR
    OutUpp( IOAdrs, USCR, 0x02 );    // U P C 動作(サンプリングフリップフロップクリア)
    OutUpp( IOAdrs, USCR, 0x00 );    // U P C 停止
}

```

➤ FFC ファンクションテーブルの内容をレジスタに設定

```

void SetFuncTblForAD( WORD IOAdrs )
{
    short  FuncNo, RegNo;

    OutUpp( IOAdrs, DDR2, 0xff ); // Data Direction Reg.2 ALL OUTPUT
    OutUpp( IOAdrs, DDR1, 0xff ); // Data Direction Reg.1 ALL OUTPUT
    OutUpp( IOAdrs, UCER2,0xff ); // UPP Contact Enable Reg.2 Pulse I/O Enable
    OutUpp( IOAdrs, UCER1,0xff ); // UPP Contact Enable Reg.1 Pulse I/O Enable
    OutUpp( IOAdrs, MFNR, 20 ); // Maximum Function Number Reg. (16 Function 5us)

    for( FuncNo = 0; FuncNo <= 15; FuncNo++ )
    {
        for( RegNo = 0; RegNo <= 7; RegNo++ )
        {
            OutUpp( IOAdrs, (WORD)(FNR + RegNo), (BYTE)FFCtbl[ FuncNo ][ RegNo ] );
        }
    }
}

```

➤ AD 変換及びエンコーダパルス変換のタイマー設定

```

void SetUPPDataRegForAD( WORD IOAdrs, WORD Freq )
{
    short      UdrNo;
    WORD       CompReg;

    // FFC コマンド実行時のコンペアレジスタに値をセット...
    // サンプリング周期 : Freq(msec)->TimerVal(usec)
    CompReg = Freq * 100;
    OutUpp( IOAdrs, UDROH, (BYTE)(CompReg >> 8) ); // UPP Data Reg. 0 上位
    OutUpp( IOAdrs, UDROL, (BYTE)(CompReg & 0xff) ); // UPP Data Reg. 0 下位

    // UPP Data Reg.1 から 23 をクリア
    for( UdrNo = 1; UdrNo <= 7; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDROH + UdrNo * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDROL + UdrNo * 2 + 1), 0 );
    }
    for( UdrNo = 8; UdrNo <= 15; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDR8H + (UdrNo - 8) * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDR8L + (UdrNo - 8) * 2 + 1), 0 );
    }
    for( UdrNo = 16; UdrNo <= 23; UdrNo++ )
    {
        OutUpp( IOAdrs, (WORD)(UDR16H + (UdrNo - 16) * 2), 0 );
        OutUpp( IOAdrs, (WORD)(UDR16L + (UdrNo - 16) * 2 + 1), 0 );
    }
}

```


➤ サンプリング実行

```

BOOL StartADSamp( HWND hDlg, WORD IOAdrs, WORD SampCnt,
                  WORD uiSTime, WORD Channel, WORD *HexSampData )
{
    WORD          DataRegHi, DataRegLo;    // A D変換用データレジスタ
    WORD          dh, dl, HexVal;         // A D変換値
    WORD          SetVal;                 // 一時変数
    WORD          i;
    unsigned long TimeSpan;               // サンプリング所要時間 : msec
    clock_t       tStart, tEnd;          // 所要時間計測用変数
    unsigned long tDiff;                 // 所要時間計測用変数
#ifdef DISpdata
    double        ValSampData;           // サンプリングデータの電圧換算値
#endif

    // 単一モードによるA D変換用データレジスタのオフセットアドレス設定
    switch( Channel )
    {
        case 1: case 5: case 9:
            DataRegHi = ADDR0H;          DataRegLo = ADDR0L;    break;
        case 2: case 6: case 10:
            DataRegHi = ADDR1H;          DataRegLo = ADDR1L;    break;
        case 3: case 7:
            DataRegHi = ADDR2H;          DataRegLo = ADDR2L;    break;
        case 4: case 8:
            DataRegHi = ADDR3H;          DataRegLo = ADDR3L;    break;
        default:
            DataRegHi = ADDR0H;          DataRegLo = ADDR0L;    break;
    }

    // 計算時間表示
    TimeSpan = (unsigned long)uiSTime * (unsigned long)SampCnt;
    sprintf( szTmp, "計算時間: %lumsec", TimeSpan );
    SetDlgItemText( hDlg, IDC_EDIT_START, (LPSTR)szTmp );

    memset( szTmp, 0x00, sizeof(szTmp) );
    SetDlgItemText( hDlg, IDC_EDIT_END, (LPSTR)szTmp );

    // ファンクション 1 から実行
    OutUp( IOAdrs, FNR, 0x01);

    // U P C 動作(GFE=1:ファンクション実行)
    OutUp( IOAdrs, USCR, 0x02);

    tStart = clock();

```

(次頁に続く)

```
/* サンプリング実行 */
for ( i = 0; i < SampCnt; i++ )
{
    while( ( InUpp( I0Adrs, ISR1) & 0x01 ) == 0 )
    {
        ; // インタラプトステータスリードし、U0 立ち下がリエッジ検出
    }
    OutUpp( I0Adrs, ISCR1, 0x00); // インターラプトステータスクリア

    // 単一モード AD 変換実行
    SetVal = 0x20 | (0x0F & (Channel - 1));
    OutUpp( I0Adrs, ADCSR, (BYTE)SetVal);

    // AD 変換終了待ち
    while( ( InUpp( I0Adrs, ADCSR) & 0x80 ) == 0 )
        ;

    // AD 変換データ上位 8 ビット読み込み
    dh = InUpp( I0Adrs, (BYTE)DataRegHi );
    // AD 変換データ下位 8 ビット読み込み
    dl = InUpp( I0Adrs, (BYTE)DataRegLo );
    // dh:b9-b2 dl:b1-b0 から 10 ビットデータに編成
    HexVal = (dh << 2) + (dl >> 6);
    HexSampData[i] = HexVal;

#ifdef DISPDATA
    ValSampData = HextoVal( HexVal );
    sprintf( szTmp, "%1.3lf", ValSampData );
    SetDlgItemText( hDlg, IDC_EDIT_VOLT, (LPSTR)szTmp );

    sprintf( szTmp, "%04x", HexSampData[i] & 0xffff );
    SetDlgItemText( hDlg, IDC_EDIT_HEX, (LPSTR)szTmp );
#endif
}

// UPC 停止(GFE=0)
OutUpp( I0Adrs, USCR, 0x00 );

// サンプリング終了
tEnd = clock();
tDiff = (unsigned long)(tEnd - tStart);

// 実測時間表示
sprintf( szTmp, "実測時間: %lumsec", tDiff );
SetDlgItemText( hDlg, IDC_EDIT_END, (LPSTR)szTmp );

// サンプリング波形をグラフにプロット
PlotSampData( hDlg, HexSampData );

return TRUE;
}
```

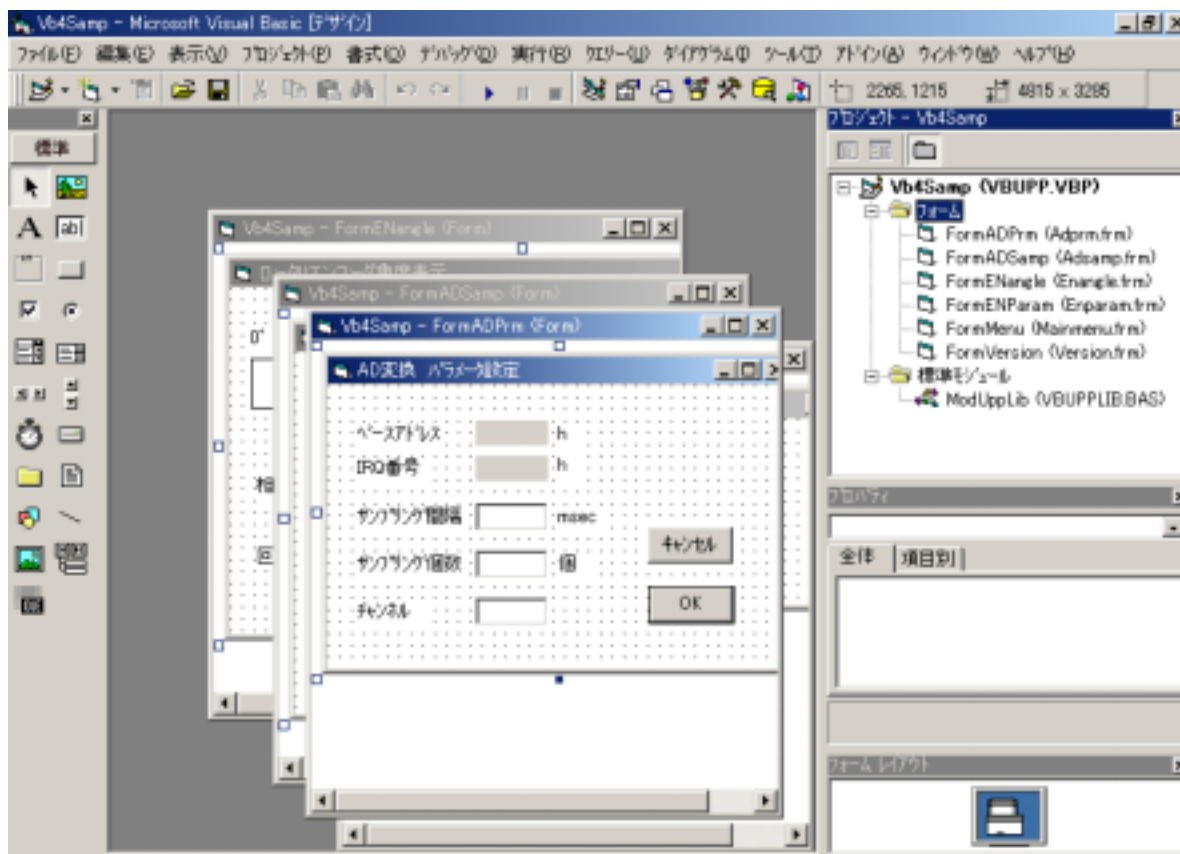
(4-6-3) Visual BASIC アプリケーション作成

=>ファイルのコピー

本製品添付のディスクの内容をサブディレクトリーを含め、全てハードディスクにコピーします。

=>プロジェクトへのファイルの追加

新規プロジェクトを作成し、下図プロジェクトウィンドウに示す各フォーム及びモジュールファイルをプロジェクトへ追加します。サンプルプログラムの実行内容は、Visual C のサンプルと同じです。配線図につきましては、前節を参照してください。



=> サンプルプログラムの実行

サンプルプログラムには、

実行例 1 : ロータリーエンコーダからの2相パルス入力アップダウンカウント

実行例 2 : A/D 変換

があります。ソースコードは添付のディスクを参照してください。

以下に**実行例 1、2**の解説をいたします。

実行例 1 : ロータリーエンコーダからの2相パルス入力アップダウンカウント

田 サンプルプログラム抜粋

➤ リソースの自動取得

```
Private Sub Form_Load()  
    Dim Status As Integer  
    Dim SlotNo As Integer  
    Dim hDlg As Long  
  
    'リソース情報の自動取得  
    For SlotNo = 0 To 1 Step 1  
        Status = UppGetCardResource(hDlg, SlotNo, UseIOBaseAdrs, MyIrqNo)  
        If (Status = 0) Then  
            'リソース情報を表示する  
            LBLIOAdrs.Caption = Hex(UseIOBaseAdrs)  
            LBLIrqNo.Caption = Str(MyIrqNo)  
            Exit For  
        End If  
    Next SlotNo  
    If (Status <> 0) Then  
        LBLIOAdrs.Caption = "ERROR"  
        LBLIrqNo.Caption = "ERROR"  
        Msg = "レジスタ登録情報取得エラー"  
        MsgBox Msg, vbOKOnly + vbCritical, "エラー"  
        Exit Sub  
    End If  
End Sub
```

➤ファンクションテーブルの設定

```

Sub SetFuncTblForEN()
  Dim FuncNo, TblNo, Dummy As Integer

  Call OutUpp(UseIOBaseAdrs, DDR2, &H0) 'Data Direction Reg.2 ALL INPUT
  Call OutUpp(UseIOBaseAdrs, DDR1, &H0) 'Data Direction Reg.1 ALL INPUT
  Call OutUpp(UseIOBaseAdrs, UCER2, &HFF) 'UPP Contact Enable Reg.2 Pulse I/O Enable
  Call OutUpp(UseIOBaseAdrs, UCER1, &HFF) 'UPP Contact Enable Reg.1 Pulse I/O Enable
  Call OutUpp(UseIOBaseAdrs, MFNR, 20) 'Maximum Function Number Reg. (16 Function 5us)

  'ファンクションテーブル設定
  UPPFuncTbl(0).FNReg = 1 'TPCコマンド設定
  UPPFuncTbl(0).CMReg = &H98 'I/N=0 RA=1 : q=low で p の立ち上がりでカウントアップ
  ' q=high で p の立ち上がりでカウントダウン
  UPPFuncTbl(0).RASRegA = 0 'UDRO にカウンタ値を出力
  UPPFuncTbl(0).RASRegB = 255 'Don't care
  UPPFuncTbl(0).IOARegA = &H40 'FA=0,RA=1
  UPPFuncTbl(0).IOARegB = &H1 'FB=0,RB=0
  UPPFuncTbl(0).IOARegC = 0 'クロック入力ピン : p = 0
  UPPFuncTbl(0).IOARegD = 1 'パルス入力ピン : q = 1

  FuncNo = 1
  For TblNo = 1 To 15
    FuncNo = FuncNo + 1
    If ((FuncNo Mod 5) = 0) Then
      FuncNo = FuncNo + 1
    End If
    UPPFuncTbl(TblNo).FNReg = FuncNo
    UPPFuncTbl(TblNo).CMReg = &HFF
    UPPFuncTbl(TblNo).RASRegA = 255
    UPPFuncTbl(TblNo).RASRegB = 255
    UPPFuncTbl(TblNo).IOARegA = &HFF
    UPPFuncTbl(TblNo).IOARegB = 255
    UPPFuncTbl(TblNo).IOARegC = 255
    UPPFuncTbl(TblNo).IOARegD = 255
  Next TblNo

  'ファンクションテーブルの内容をレジスタに設定
  For TblNo = 0 To 15
    Call OutUpp(UseIOBaseAdrs, FNR + 0, UPPFuncTbl(TblNo).FNReg)
    Call OutUpp(UseIOBaseAdrs, FNR + 1, UPPFuncTbl(TblNo).CMReg)
    Call OutUpp(UseIOBaseAdrs, FNR + 2, UPPFuncTbl(TblNo).RASRegA)
    Call OutUpp(UseIOBaseAdrs, FNR + 3, UPPFuncTbl(TblNo).RASRegB)
    Call OutUpp(UseIOBaseAdrs, FNR + 4, UPPFuncTbl(TblNo).IOARegA)
    Call OutUpp(UseIOBaseAdrs, FNR + 5, UPPFuncTbl(TblNo).IOARegB)
    Call OutUpp(UseIOBaseAdrs, FNR + 6, UPPFuncTbl(TblNo).IOARegC)
    Call OutUpp(UseIOBaseAdrs, FNR + 7, UPPFuncTbl(TblNo).IOARegD)
  Next TblNo

End Sub

```

実行例 2 : A/D 変換

■ サンプルプログラム抜粋

▶ ファンクションテーブルの設定

```

Sub SetFuncTblForAD()
  Dim FuncNo, TblNo, Dummy As Integer

  Call OutUpp(UseIOBaseAdrs, DDR2, &HFF) 'Data Direction Reg.2 ALL OUTPUT
  Call OutUpp(UseIOBaseAdrs, DDR1, &HFF) 'Data Direction Reg.1 ALL OUTPUT
  Call OutUpp(UseIOBaseAdrs, UCER2, &HFF) 'UPP Contact Enable Reg.2 Pulse I/O Enable*/
  Call OutUpp(UseIOBaseAdrs, UCER1, &HFF) 'UPP Contact Enable Reg.1 Pulse I/O Enable*/
  Call OutUpp(UseIOBaseAdrs, MFNR, 20) 'Maximum Function Number Reg. (16 Function 5us)*/

  'ファンクションテーブル設定
  UPPFuncTbl(0).FNReg = 1 'FFC コマンド設定
  UPPFuncTbl(0).CMReg = &H80 'C/T=0 : レジスタ i はタイマーとして動作し内部クロックをカウント
  UPPFuncTbl(0).RASRegA = 1 'カウンタレジスタにデータレジスタ 1 (UDR1) を使用
  UPPFuncTbl(0).RASRegB = 0 'コンペアレジスタにデータレジスタ 0 (UDR0) を使用
  UPPFuncTbl(0).IOARegA = &HFF 'FA=1,RA=1 : 両側エッジカウント
  UPPFuncTbl(0).IOARegB = 255
  UPPFuncTbl(0).IOARegC = 0 'パルス出力ピンを U0 に設定 . これにより、U0 のパルス信号エッジで
  UPPFuncTbl(0).IOARegD = 255 'インターラプトステータスレジスタ (ISR1:Bit0) のビットがセット。

  FuncNo = 1
  For TblNo = 1 To 15
    FuncNo = FuncNo + 1
    If ((FuncNo Mod 5) = 0) Then
      FuncNo = FuncNo + 1
    End If
    UPPFuncTbl(TblNo).FNReg = FuncNo
    UPPFuncTbl(TblNo).CMReg = &HFF
    UPPFuncTbl(TblNo).RASRegA = 255
    UPPFuncTbl(TblNo).RASRegB = 255
    UPPFuncTbl(TblNo).IOARegA = &HFF
    UPPFuncTbl(TblNo).IOARegB = 255
    UPPFuncTbl(TblNo).IOARegC = 255
    UPPFuncTbl(TblNo).IOARegD = 255
  Next TblNo

  'ファンクションテーブルの内容をレジスタに設定
  For TblNo = 0 To 15
    Call OutUpp(UseIOBaseAdrs, FNR + 0, UPPFuncTbl(TblNo).FNReg)
    Call OutUpp(UseIOBaseAdrs, FNR + 1, UPPFuncTbl(TblNo).CMReg)
    Call OutUpp(UseIOBaseAdrs, FNR + 2, UPPFuncTbl(TblNo).RASRegA)
    Call OutUpp(UseIOBaseAdrs, FNR + 3, UPPFuncTbl(TblNo).RASRegB)
    Call OutUpp(UseIOBaseAdrs, FNR + 4, UPPFuncTbl(TblNo).IOARegA)
    Call OutUpp(UseIOBaseAdrs, FNR + 5, UPPFuncTbl(TblNo).IOARegB)
    Call OutUpp(UseIOBaseAdrs, FNR + 6, UPPFuncTbl(TblNo).IOARegC)
    Call OutUpp(UseIOBaseAdrs, FNR + 7, UPPFuncTbl(TblNo).IOARegD)
  Next TblNo

End Sub

```

(空白ページ)

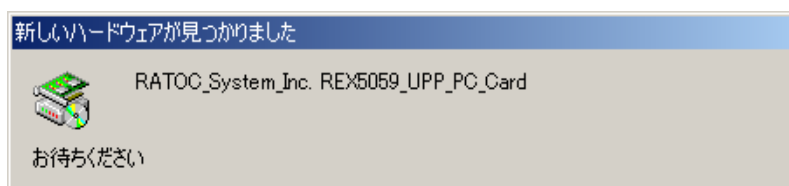
第5章 Windows2000/XP解説

(5-1) セットアップ

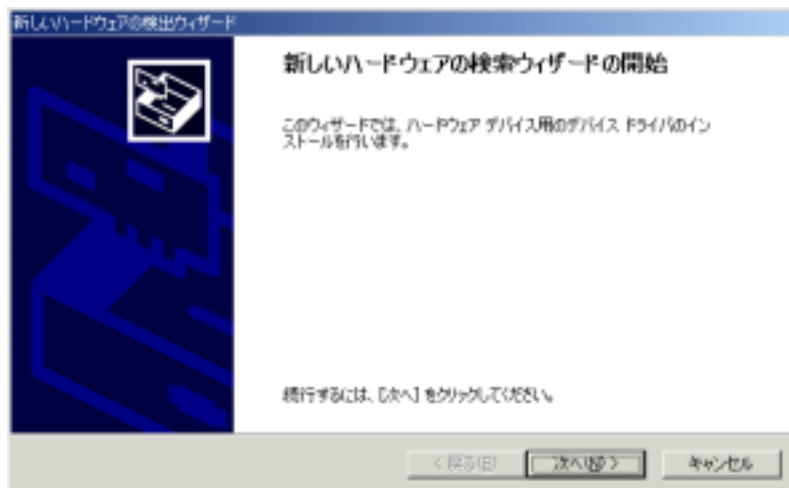
(5-1-1) Windows2000 インストール

[1] PC カードの挿入

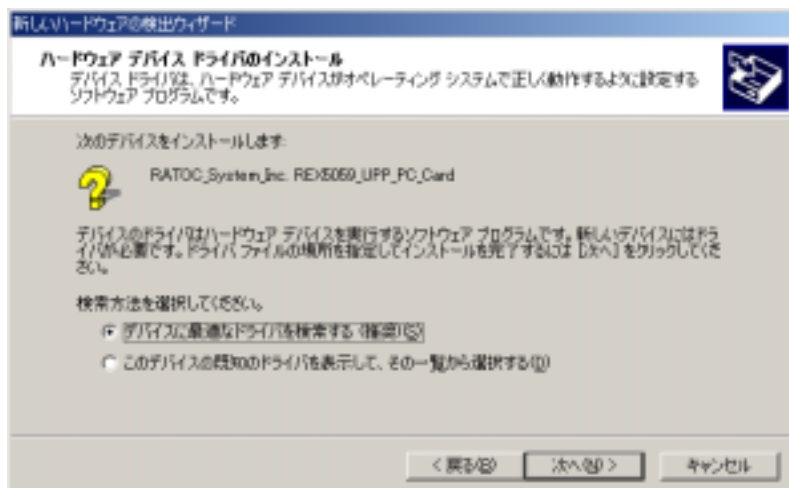
PC カードを挿入すると「ハードウェアウィザード」が起動し(右下画面)、インストールが開始されます。「RATOC_System_Inc. REX5059_UPP_PC_Card」と表示されているかを確認し、以下の手順でインストールを行ってください。



「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。



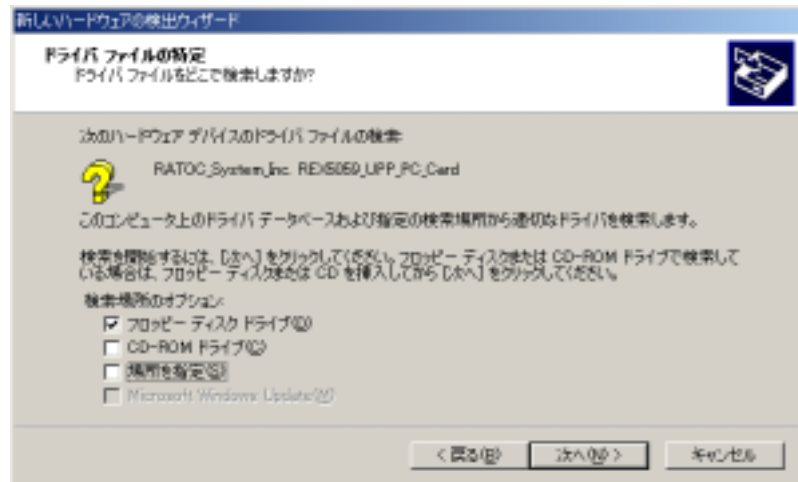
「ハードウェアデバイスドライバのインストール」では「デバイスに最適なドライバを検索する(推奨)(S)」にチェックを入れて「次へ(N)>」ボタンを押します。



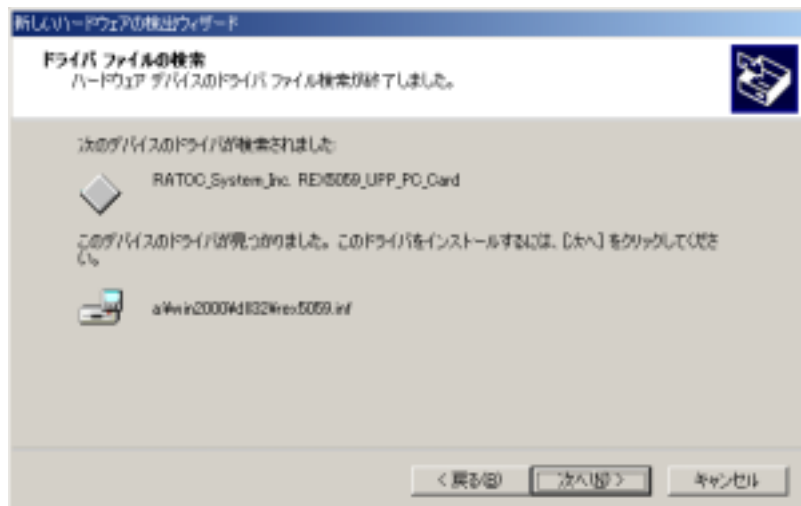
[2] inf ファイルの自動検索

製品添付の「REX-5059 UPP PC Card セットアップ・ライブラリディスク」Windows2000/XP 用 FD をフロッピーディスクドライブに挿入します。

「ドライバファイルの特定」の「検索場所のオプション:」で「フロッピーディスクドライブ(D)」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検索」で、右の inf ファイルが自動的に検索されますので、「次へ(N)>」ボタンを押します。



「新しいハードウェアの検出ウィザードの完了」で「REX-5059.SYS for REX-5059 UPP PC CARD」が表示されます。「完了」ボタンを押してください。

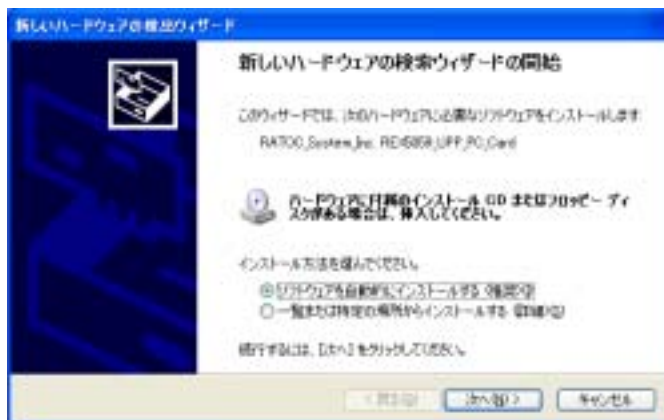


以上で、REX-5059 のインストールは終了です。

(5-1-2) WindowsXP インストール

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、インストールが開始します。以下の手順でインストールを行って下さい。

製品添付の Windows2000/XP 用ディスクをフロッピーディスクドライブに挿入し、「新しいハードウェアの検索ウィザードの開始」で「ソフトウェアを自動的にインストールする(推奨)(I)」にチェックを入れて「次へ(N)>」ボタンを押して先へ進みます。



セットアップ情報ファイル(.inf ファイル)が、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。



「新しいハードウェアの検索ウィザードの完了」で「RE X5059.SYS for REX5059 UPP PC CARD」が表示されます。

「完了」ボタンを押してください。

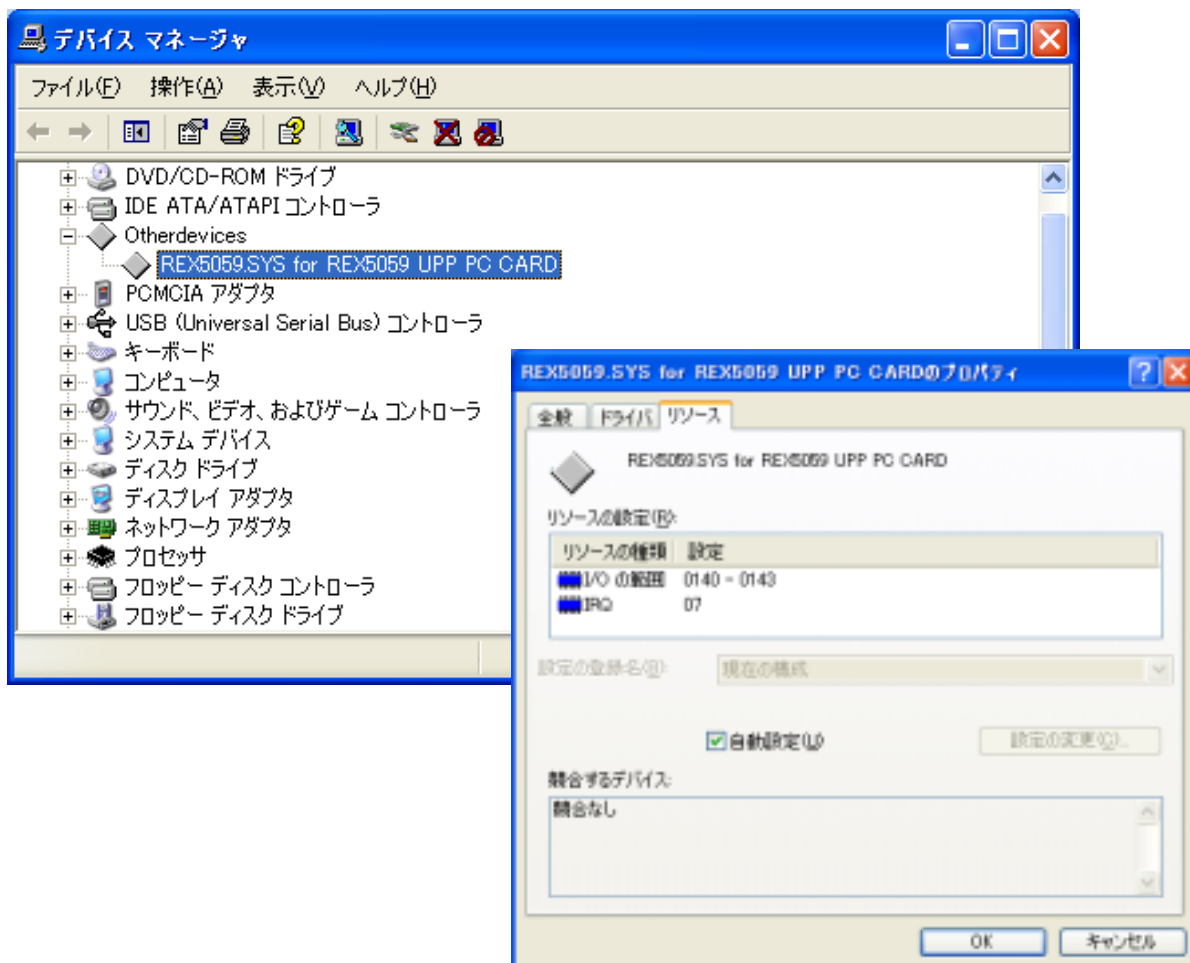
以上で、REX-5059 のインストールは終了です。



(5-1-3) インストール内容の確認

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 REX5059.SYS for REX-5059 UPP 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」から「リソース」タブを開き、I/O に連続4バイトが割当てられ、I/O、IRQ それぞれに競合が無いことを確認してください。



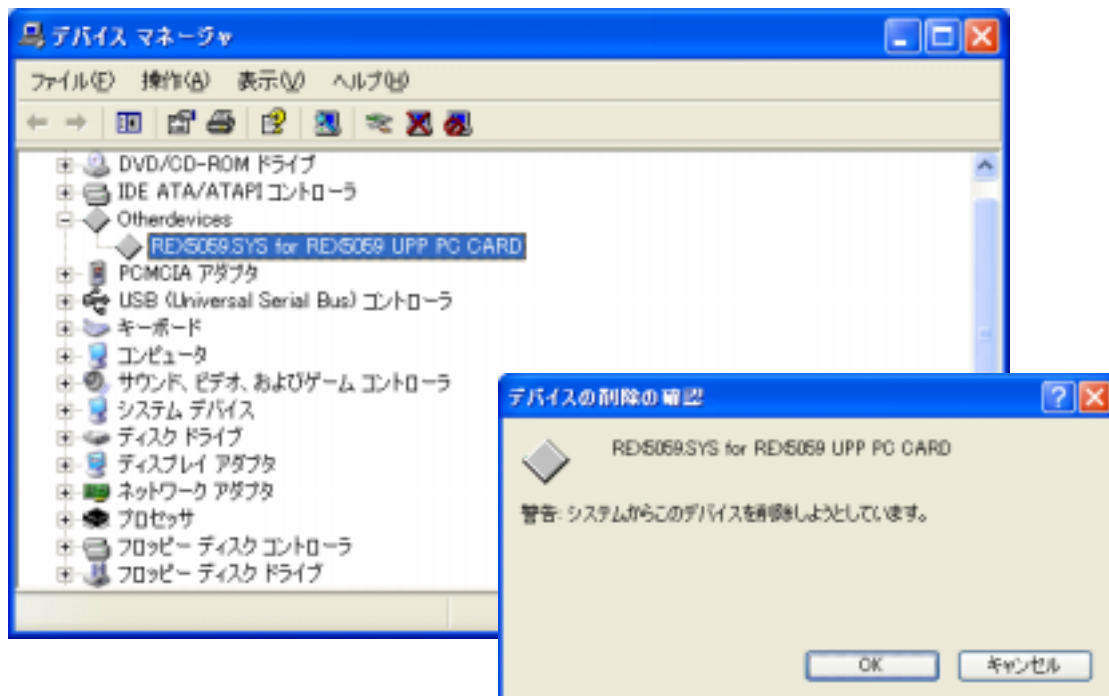
(5-1-4) アンインストール

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

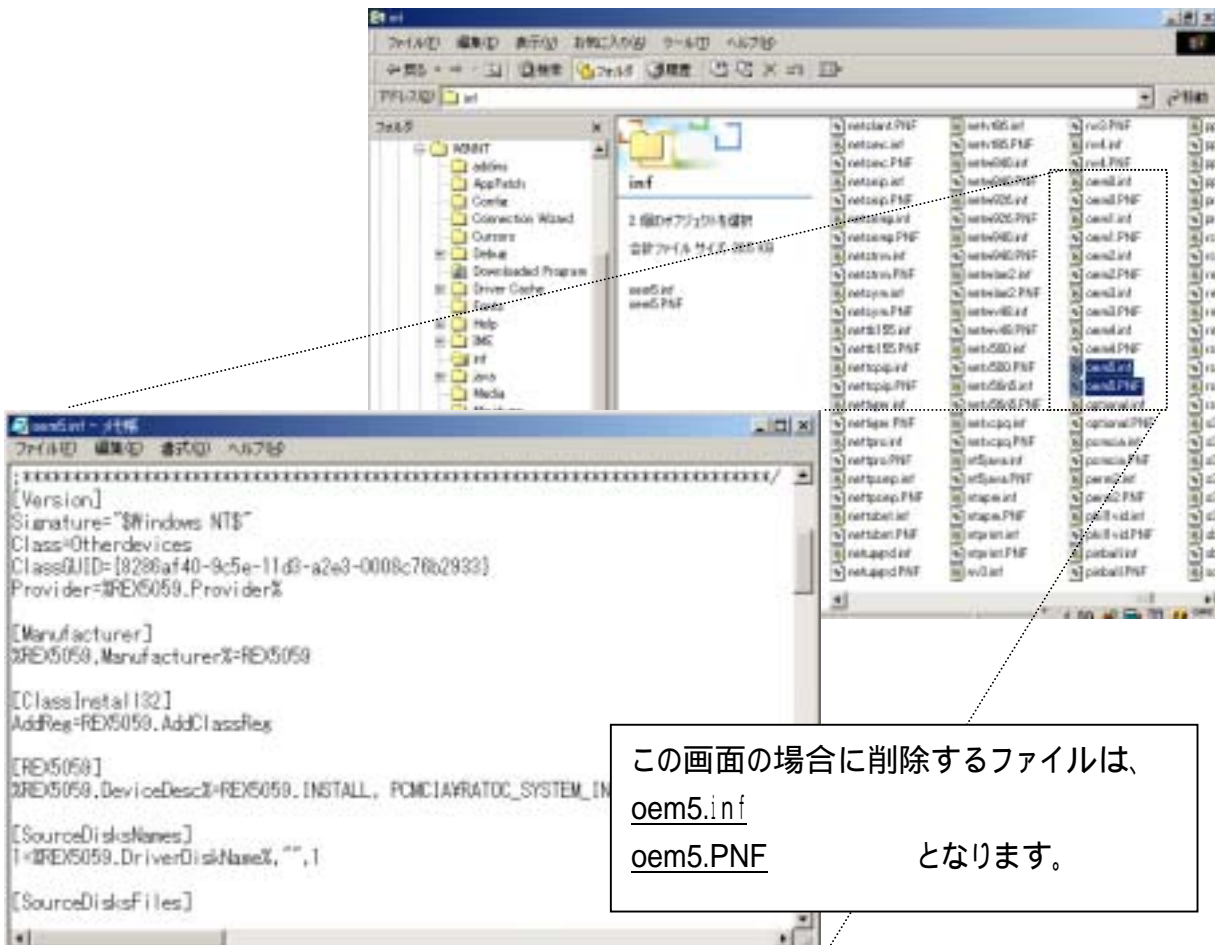
PC カードを挿入した状態で、コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D_)」ボタンを押します。「Otherdevices」をクリックして REX5059.SYS for REX5059 UPP PC CARD を表示させをクリックします。

メニューバーより「操作(A)」 - 「削除(U)」を選択します。デバイスの削除の確認で「OK」ボタンを押して削除してください。



[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 %REX5059, Manufacturer%=REX5059 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-5059 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、表示タブ内の詳細設定で、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。

(5-2) Visual C 言語インターフェース

(5-2-1) DLL ライブラリ解説

本製品には、Microsoft Visual C++ 5.0 で作成したダイナミックリンクライブラリ“UPPLIB2K.DLL”が添付されており、UPP カードに入出力を行うための API 関数が提供されています。

Visual C/C++で作成したアプリケーションプログラムから Windows2000 用 32Bit 版 DLL “UPPLIB2K.DLL”を呼び出すためには、次の行程が必要です。

アプリケーションプログラムに UPPLIB2K.H ファイルをインクルードする。

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 UPPLIB2K.LIB ファイルを追加する。

=> DLL 関数のインポート宣言 (UPPLIB2K.H より抜粋)

```
DllImport BOOL APIENTRY UppStartEventSyncInt( HWND, WORD, WORD, WORD, DWORD );
DllImport BOOL APIENTRY UppEndEventSyncInt( void );
DllImport BOOL APIENTRY UppGetCardResource( HWND, WORD, LPWORD, LPWORD );
DllImport WORD APIENTRY WaitMilliseconds( DWORD );
DllImport WORD APIENTRY OutUpp( WORD, WORD, BYTE );
DllImport WORD APIENTRY InUpp( WORD, WORD );
DllImport WORD APIENTRY OutPort( WORD, WORD );
DllImport WORD APIENTRY wOutPort( WORD, WORD );
DllImport WORD APIENTRY InPort( WORD );
DllImport WORD APIENTRY wInPort( WORD );
```

尚、インポート宣言 `DllImport` は下記のように定義されています。

```
#define DllImport _declspec( dllimport )
```

=> DLL 関数仕様 (Visual C/C++)

UppGetCardResource

リソース情報の取得

書式	BOOL UppGetCardResource(HWND hwnd, WORD SlotNo, LPWORD pIOBase, LPWORD plrqNo)												
機能	スロットに挿入されているカードが REX-5059 UPP PC カードなら、現在割り当てられている I/O ベースアドレスおよび IRQ 番号情報を返します。												
引数	<table border="0"> <tr> <td>HWND</td> <td>hwnd</td> <td>: 呼び出し元ウィンドウハンドル</td> </tr> <tr> <td>WORD</td> <td>SlotNo</td> <td>: サーチするカード挿入スロット番号 (0 を指定します)</td> </tr> <tr> <td>LPWORD</td> <td>pIOBase</td> <td>: (出力)I/O リソース情報を格納する変数のアドレス</td> </tr> <tr> <td>LPWORD</td> <td>plrqNo</td> <td>: (出力)IRQ リソース情報を格納する変数のアドレス</td> </tr> </table>	HWND	hwnd	: 呼び出し元ウィンドウハンドル	WORD	SlotNo	: サーチするカード挿入スロット番号 (0 を指定します)	LPWORD	pIOBase	: (出力)I/O リソース情報を格納する変数のアドレス	LPWORD	plrqNo	: (出力)IRQ リソース情報を格納する変数のアドレス
HWND	hwnd	: 呼び出し元ウィンドウハンドル											
WORD	SlotNo	: サーチするカード挿入スロット番号 (0 を指定します)											
LPWORD	pIOBase	: (出力)I/O リソース情報を格納する変数のアドレス											
LPWORD	plrqNo	: (出力)IRQ リソース情報を格納する変数のアドレス											
戻値	正常終了時は 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。												
解説	第2引数 SlotNo は、Windows95/98/Me 用ライブラリとの互換性のため、存在します。Windows2000/XP 用ライブラリでは必要としないので、0 を指定してください。												

OutUpp**UPP レジスタに1バイトを出力**

書 式 WORD **OutUpp**(WORD **IOAdrs**, WORD **RegAddr**, BYTE **Val**)

機 能 1バイトを UPP レジスタに出力

引 数 WORD **IOAdrs** : カードのベースアドレス
 WORD **RegAddr** : UPP レジスタインデックス
 BYTE **Val** : バイト出力値

戻 値 正常終了時は 0 を返し、エラーの場合は 0xFFFF を返します。

解 説 UPP のレジスタへ出力を行う場合、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットし、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットした後、データレジスタに対し出力を行います。DLL ライブラリにあるポートへの出力命令を使うと下記コーディングになります。

OutPort (ベースアドレス + 3, インデックス上位バイト);

OutPort (ベースアドレス + 2, インデックス下位バイト);

OutPort (ベースアドレス, 出力データ);

関数 **OutUpp**() を使うことにより下記のように1行で記述することができます。

OutUpp (ベースアドレス, インデックス, 出力データ);

InUpp**UPP レジスタから1バイト入力**

書 式 WORD **InUpp**(WORD **IOAdrs**, WORD **RegAddr**)

機 能 ポートから1バイト読み込む

引 数 WORD **IOAdrs** : カードのベースアドレス
 WORD **RegAddr** : UPP レジスタインデックス

戻 値 バイト入力値を返します(上位バイトは無視してください)。

解 説 UPP レジスタからの入力を行う場合、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットし、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットした後、データレジスタからの入力を行います。DLL ライブラリにあるポートからの入力命令を使うと下記のコーディングを行う必要があります。

OutPort (ベースアドレス + 3, インデックス上位バイト);

OutPort (ベースアドレス + 2, インデックス下位バイト);

 入力データ = **InPort** (ベースアドレス);

関数 **InUpp**() を使うことにより下記のように1行で記述することができます。

 入力データ = **InUpp** (ベースアドレス, インデックス);

OutPort ポートに1バイトを出力

- 書式 WORD OutPort(WORD IOAdrs, WORD OutVal)
- 機能 1バイトをポートに出力
- 引数 WORD IOAdrs : 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 正常終了時は 0 を返し、エラーの場合は 0xFFFF を返します。

wOutPort ポートに1ワードを出力

- 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 : 出力する I/O ポートアドレス
WORD wOutVal : ワード出力値
- 戻値 正常終了時は 0 を返し、エラーの場合は 0xFFFF を返します。

InPort ポートから1バイト入力

- 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 : 入力する I/O ポートアドレス
- 戻値 バイト入力値を返します(上位バイトは無視してください)。

wInPort ポートから1ワード入力

- 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 : 入力する I/O ポートアドレス
- 戻値 ワード入力値を返します。

UppStartEventSyncInt

ユーザ定義メッセージ割り込みの開始

書式	BOOL UppStartEventSyncInt(HWND hwnd, WORD IOAdrs, WORD IrqNo, WORD ISCRreg, DWORD StopCount)	
機能	割り込みに同期して DLL からユーザアプリケーションにユーザ定義メッセージ WM_VXDEVENT 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
	WORD IOAdrs	: カードに割り当てられている I/O ベースアドレス
	WORD IrqNo	: カードに割り当てられている IRQ 番号
	WORD ISCRreg	: 割り込みステータスクリアレジスタ指定
	DWORD StopCount	: 割り込み終了回数(0 指定で無限に繰り返す)
戻値	0: 正常終了 -1: 二重起動防止 -2: パラメータ設定エラー -3: スレッド作成エラー	

UppEndEventSyncInt

割り込みを解除

書式	BOOL UppEndEventSyncInt(void)	
機能	DLL 内の割り込み待機ルーチンを終了します。	
引数	なし	
戻値	正常終了時は 0 を返し、エラーの場合は -1 を返します。	

WaitMilliSeconds

プログラムの中断

書式	WORD WaitMilliSeconds(DWORD MsecCount)	
機能	指定した時間プログラムの実行を停止する	
引数	DWORD MsecCount	: ミリ秒単位のウェイト時間
戻値	正常終了時は 0 を返し、エラーの場合は 0xFFFF を返します。	

(5-2-2) Visual C サンプルプログラム

REX-5059 UPP PC カードを制御するアプリケーションを Visual C++で開発する場合は、本製品添付のソースプログラム(*.c)を参考にして下さい。

添付のサンプルプログラムを使用してプログラム作成・修正する場合は、各サンプルのフォルダ内にあるファイル(*.C、*.RC、RESOURCE.H、REX.ICO、UPPLIB2K.H、UPPLIB2K.LIB)を新規プロジェクトに追加してコンパイルを行ってください。

=>サンプルプログラムの実行

サンプルプログラムには、

実行例 1 : ロータリーエンコーダからの2相パルス入力アップダウンカウント

実行例 2 : A/D 変換

実行例 3 : FFC インターバル割り込み

があります。ソースコードは添付のディスクを参照してください。

次頁より**実行例**の解説をいたします。

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 **UPPLIB2K.H** とライブラリファイル **UPPLIB2K.LIB** を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 **UPPLIB2K.DLL** ではサポートしておりませんのでご注意ください。

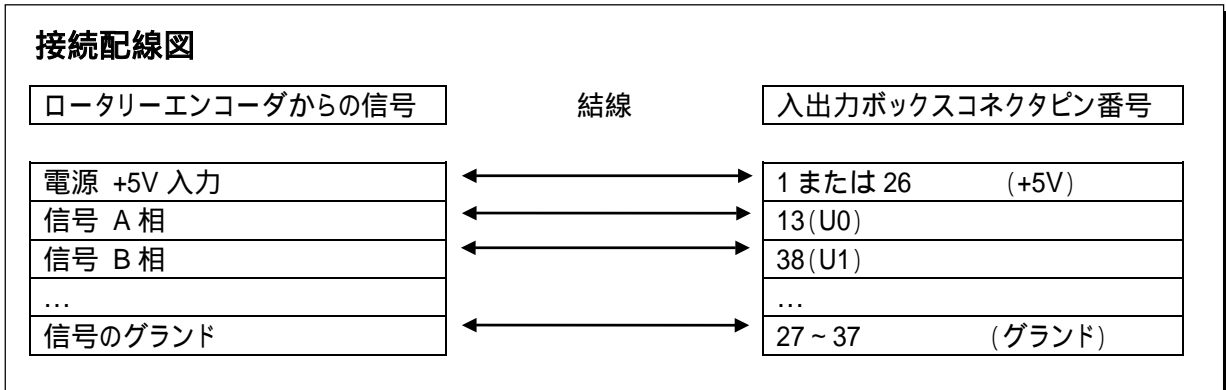
ShowCardUtil()
UppGetVersion()
UppCardReset()

実行例 1 ロータリーエンコーダからの2相パルス入力アップダウンカウント

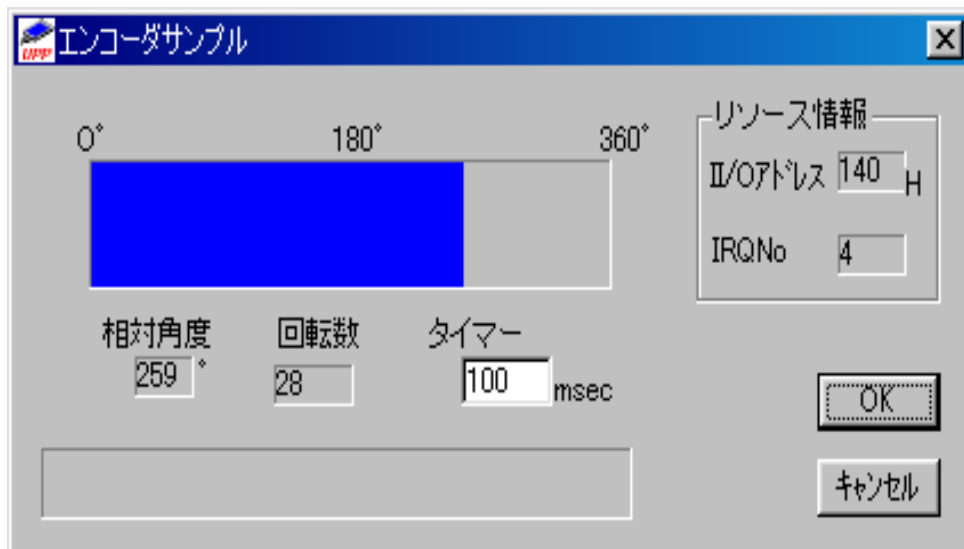
TPC コマンドを使用して、UPP に入力された2相パルスのカウントを行います。

プログラムでは、タイマイベントを使用し、タイマイベント内でカウント値を読み込み、その値からロタリーエンコーダの回転角、回転数を算出・表示してします。

ロタリーエンコーダからの入力例を実行する場合の配線図及び実行画面を下図に示します。



実行画面



■ サンプルプログラム抜粋

➤ ファンクションテーブルの選択及び設定

```
BYTE fdata[16][8] = {
    /* FNR, CMR, RASRA, RASRB, IOARA, IOARB, IOARC, IOARD */

    { 1, 0x98, 0, 255, 0x40, 0x01, 0, 1 }, // TPC コマンド
    { 2, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 3, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 4, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 6, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 7, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 8, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 9, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 11, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 12, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 13, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 14, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 16, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 17, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 18, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 19, 0xff, 255, 255, 0xff, 255, 255, 255 },
};
```

➤ TPC ファンクションテーブルの内容をレジスタに設定

```
// ファンクションテーブルを設定
WORD Func, Reg; // fdata[Func][Reg]

for( Func = 0; Func <= 15; Func++ ){
    for( Reg = 0; Reg <= 7; Reg++ ){
        // ファンクションテーブルのセット
        OutUpd( IOAdrs, (WORD)(FNR+Reg), fdata[Func][Reg] );
    }
}
```

➤ パルス入力アップダウンカウントチェック

```
void CALLBACK TimerProc( UINT TimerID, UINT message, HWND hwnd, DWORD dwRsv1, DWORD dwRsv2 )
{
    WORD Count;           // UDR0(カウント値)
    WORD Phase;          // 位相角
    WORD Rotation;       // 回転数

    // カウント数取得
    Count = 256 * (BYTE)InUpp( IOAdrs, UDR0H ) + (BYTE)InUpp( IOAdrs, UDR0L );

    // 7FFFH < Count 数 <= FFFFH のとき( BasicCount=7FFFH )
    if( BasicCount < Count )
    {
        Rotation = ( Count - BasicCount ) / 360; // 360P/R
        Phase = ( Count - BasicCount ) % 360;
        SetDlgItemInt( hwnd, IDS_PHASE, Phase, FALSE );
        SetDlgItemInt( hwnd, IDS_ROTATION, Rotation, FALSE );

        // 背景を灰色にする
        FillRect( hdc, &rcGrey, hbrGrey);
        rcBlue.right = Phase * rcGrey.right / 360;
        // 長方形の描画
        FillRect( hdc, &rcBlue, hbrBlue);
    }

    // 0000H <= Count 数 < 7FFFH のとき
    else if( BasicCount > Count )
    {
        Rotation = ( BasicCount - Count ) / 360; // 360P/R
        Phase = ( BasicCount - Count ) % 360;
        SetDlgItemInt( hwnd, IDS_PHASE, (360-Phase), FALSE );
        sprintf( szBuf, "%d", Rotation );
        SetDlgItemText( hwnd, IDS_ROTATION, szBuf );

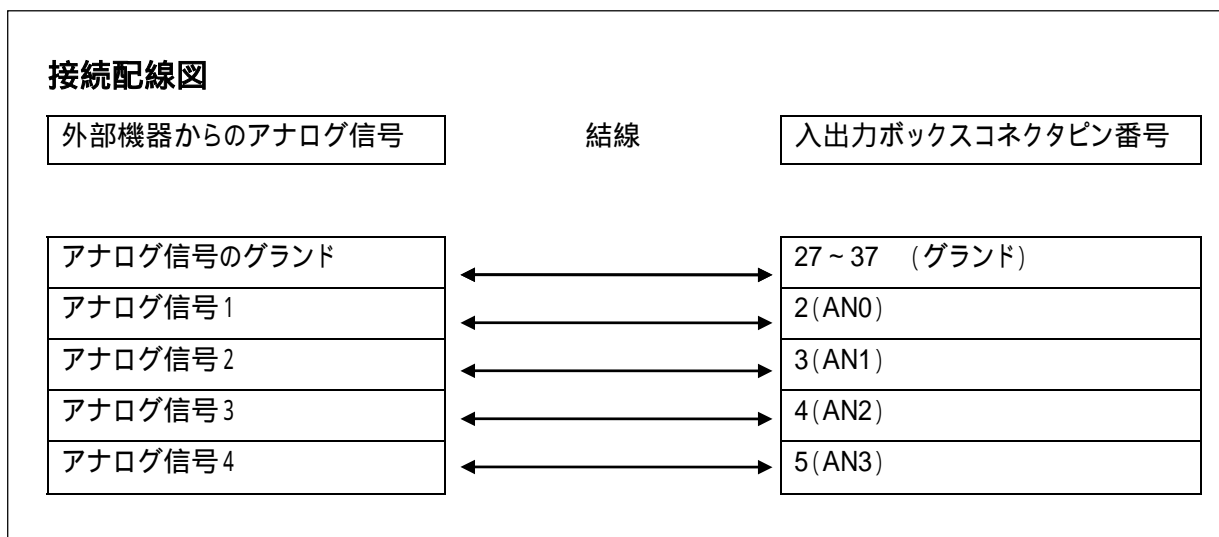
        // 背景を灰色にする
        FillRect( hdc, &rcGrey, hbrGrey);
        rcBlue.right = rcGrey.right - ( Phase * rcGrey.right / 360 );
        // 長方形の描画
        FillRect( hdc, &rcBlue, hbrBlue);
    }

    // カウント値が以下の条件となった時、カウント値を BasicCount にリセット
    if( Count >= 65527 || Count <= 7 )
    {
        OutUpp( IOAdrs, UDR0H, 0x7f );
        OutUpp( IOAdrs, UDR0L, 0xff );
    }
}
```

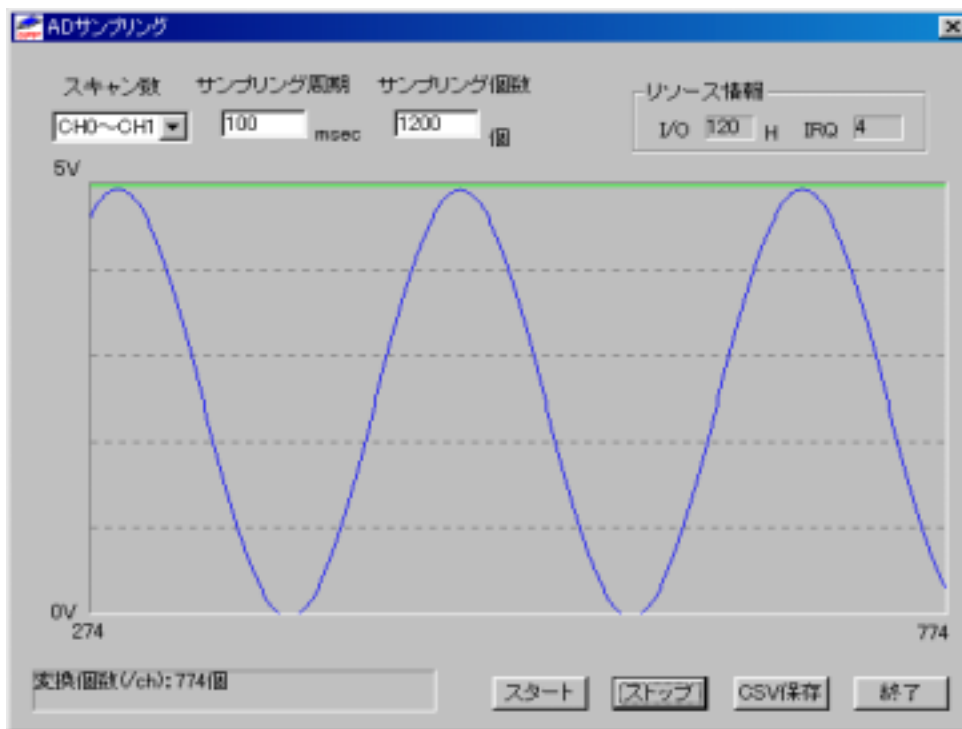
実行例 2 スキャンモードでの A/D 変換

プログラムでは、指定周期の割り込み信号に同期したポストメッセージを受け取り、メッセージ処理内で、AD 変換データのサンプリングを行っています。

A/D 変換入力の例を実行する場合の配線図及び実行画面を下図に示します。



実行画面



■ サンプルプログラム抜粋

➤ 変換データの取得

```
void DIg_OnUserDefineMessage( HWND hwnd, WPARAM wParam, LPARAM lParam )
{
    char  ch; // ループカウンタ
    BYTE  dh; // A/D データレジスタ(H)
    BYTE  dl; // A/D データレジスタ(L)

    // AD 変換終了待ち--->1 スキャン終了で bit7 に 1 がたつ
    while( (InUpp( IOAdrs, ADCSR ) & 0x80) == 0 )
    {
        if( StopFlag == TRUE )
            return;
    }
    // 変換データの読込
    for( ch = 0; ch <= ScanCH; ch++ )
    {
        dh = (BYTE)InUpp( IOAdrs, (WORD)(ADDR0H + ch * 2) );
        dl = (BYTE)InUpp( IOAdrs, (WORD)(ADDR0L + ch * 2) );
        AdBuf[(ScanCH+1)*(lParam-1)+ch] = (dh << 2) + (dl >> 6);
    }

    // 現在のサンプリング個数(1チャンネル)を取得
    GetCount = lParam;
    sprintf( szBuf, "変換個数(/ch) : %d 個", GetCount );
    SetDlgItemText( hwnd, IDS_STATUS, szBuf);

    // サンプリング個数まで到達した時
    if( GetCount == SampCount )
    {
        UppEndEventSyncInt(); // 割り込み解除
        OutUpp( IOAdrs, ADCSR, 0x00 ); // AD 変換停止
        OutUpp( IOAdrs, IER1, 0x00 );
        OutUpp( IOAdrs, USCR, 0x00 ); // UPP 停止
    }
}
```

➤ AD 変換開始

```
BOOL Cmd_OnStart( HWND hwnd )
{
(省略) . . . . .
// 割り込み信号に同期して WM_VXDEVENT をポストする
Status = UppStartEventSyncInt( hwnd, IOAdrs, IrqNo, ISCR1, SampCount );
// エラー処理
if ( Status != 0 )
{
    sprintf( szBuf, "割り込み開始エラー-%d", Status );
    SetDlgItemText( hwnd, IDS_STATUS, szBuf );
    return FALSE;
}

// 割り込みルーチンの登録・割り込みイネーブル => U0 の信号の立下りエッジにより割り込み発生
OutUpp( IOAdrs, IER1, 0x01 );
// ファンクション1から実行
OutUpp( IOAdrs, FNR, 0x01 );

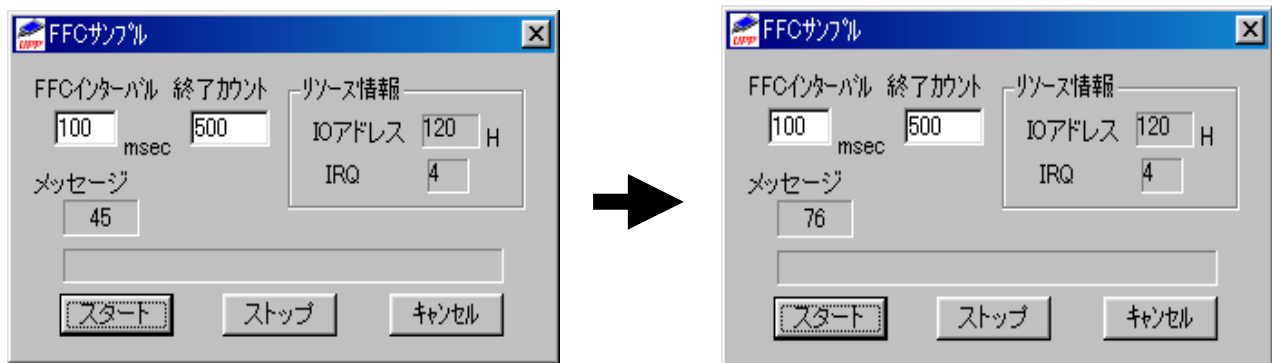
// AdBuf がアロケートされている場合、一旦開放する
if( AdBuf != NULL )
    LocalFree( AdBuf );
// AD 変換データを格納するバッファを ( サンプル数 × ch 数分 × 2 バイト ) 分確保
AdBuf = LocalAlloc( LPTR, (SampCount * (ScanCH+1)) * 2 );
if( AdBuf == NULL )
{
    sprintf( szBuf, "AdBuf LocalAlloc ERROR" );
    MessageBox( hwnd, szBuf, "AppliMsg", MB_OK );
    return FALSE;
}
// AD 変換スタート
OutUpp( IOAdrs, ADCSR, (BYTE)(0x30 | ScanCH) );
// UPP 動作(ファンクション実行)
OutUpp( IOAdrs, USCR, 0x02 );
. . . . .
}
```


実行例 3 FFC インターバル割り込み

FFC コマンドでパルス出力を行い、そのパルスを割り込み信号として、割り込みに同期したメッセージをアプリケーションに送ります。そのとき、追加情報として、割り込み累計回数が IParam に格納されています。

プログラムでは、受け取ったメッセージの処理内で IParam の値を表示しています。

FFC インターバル割り込みサンプルの実行画面を下図に示します。



田 サンプルプログラム抜粋

➤ ファンクションテーブルの選択及び設定

```
BYTE fdata[16][8] = {
    /* FNR, CMR, RASRA, RASRB, IOARA, IOARB, IOARC, IOARD */

    { 1, 0x80, 1, 0, 0xff, 255, 0, 255 }, // FFC(1)コマンド
    { 2, 0x88, 3, 2, 0x20, 255, 1, 255 }, // FFC(2)コマンド
    { 3, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 4, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 6, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 7, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 8, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 9, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 11, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 12, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 13, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 14, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 16, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 17, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 18, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 19, 0xff, 255, 255, 0xff, 255, 255, 255 },
};
```

➤ 割込みポストメッセージ処理

```
void Dlg_OnUserDefineMessage( HWND hwnd, WPARAM wParam, LPARAM lParam )
{
    SetDlgItemInt( hwnd, IDS_POSTMSG, lParam, FALSE );
    if( (DWORD)lParam == StopCount )
    {
        UppEndEventSyncInt();
        OutUpp( IOAdrs, IER1, 0x00 );
        OutUpp( IOAdrs, USCR, 0x00 );
        sprintf( szBuf, "メッセージ指定回数終了" );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
    }
}
```

(5-3) Visual BASIC 言語インターフェース

(5-3-1) DLL ライブラリの Declare 宣言

本製品には、Microsoft Visual C++ 5.0 で作成したダイナミックリンクライブラリ“UPPLIB2K.DLL”が添付されており、UPPカードに入出力を行うためのAPI関数が提供されています。

Visual BASIC 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“UPPLIB2K.DLL”を呼び出すためには、モジュール定義ファイル(.BAS)で各API関数を Declare 宣言する必要があります。製品には、モジュール定義ファイル“VBUPPLIB.BAS”が添付してありますので、ご利用下さい。

また割込み制御を行う場合は、割込み要求に同期した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、OLE カスタムコントロール“MBOX5059.OCX”を使用します。MBOX5059.OCX の使用方法については(5-3-2) カスタムコントロールを参照してください。

=> モジュール定義ファイル Declare 宣言例 (VBUPPLIB.BAS より抜粋)

```
Declare Function UppGetCardResource Lib "UPPLIB2k.DLL" (ByVal hwnd As Long, ByVal SlotNo As Integer, IOAdrs As Integer, IrqNo As Integer) As Long
Declare Function OutPort Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer, ByVal OutVal As Integer) As Integer
Declare Function wOutPort Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer, ByVal OutVal As Long)
Declare Function InPort Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer) As Long
Declare Function wInPort Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer) As Long

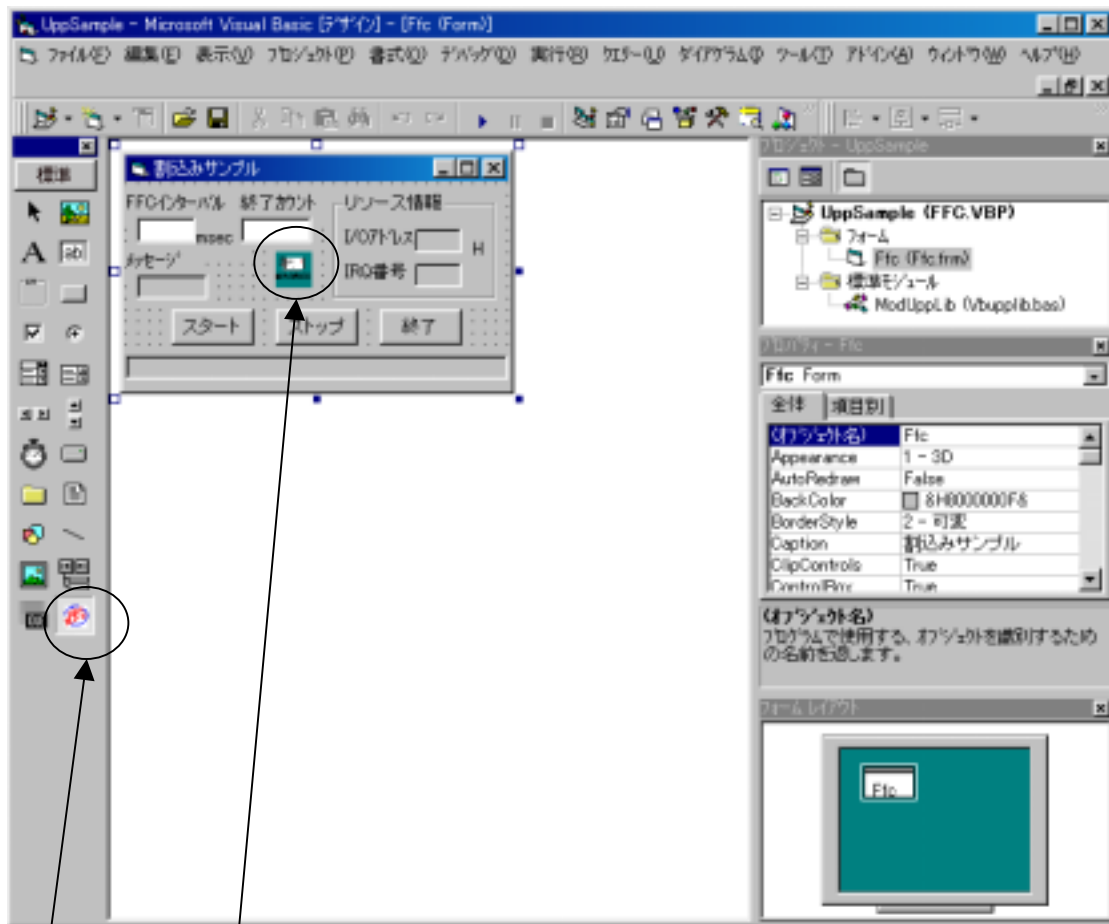
Declare Function OutUpp Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer, ByVal RegAddr As Integer, ByVal OutVal As Byte) As Integer
Declare Function InUpp Lib "UPPLIB2k.DLL" (ByVal IOAdrs As Integer, ByVal RegAddr As Integer) As Long
Declare Function WaitMilliseconds Lib "UPPLIB2k.DLL" (ByVal MsecCount As Long) As Integer
'
Declare Function UppStartEventSynclnt Lib "UPPLIB2k.DLL" (ByVal hwnd As Long, ByVal IOAdrs As Integer, ByVal IrqNo As Integer, ByVal ISCReg As Integer, ByVal StopCount As Long) As Long
Declare Function UppEndEventSynclnt Lib "UPPLIB2k.DLL" () As Long
```

関数の仕様については、「(5-2-1) DLL ライブラリ解説」を参照してください。

(5-3-2) カスタムコントロール

下記画面は、Visual BASIC 6.0 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 MBO X5059.OCX を使用します。

次頁より MBOX5059.OCX の使用方法について説明します。



本製品添付の OLE カスタムコントロール(MBO X5059.OCX)

使用するコンポーネントの中から、「MBO X OLE Control module」を追加する。

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 “MBOX5059.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¥Mbox5059.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¥Mbox5059.ocx
```



登録成功メッセージ



登録削除成功メッセージ

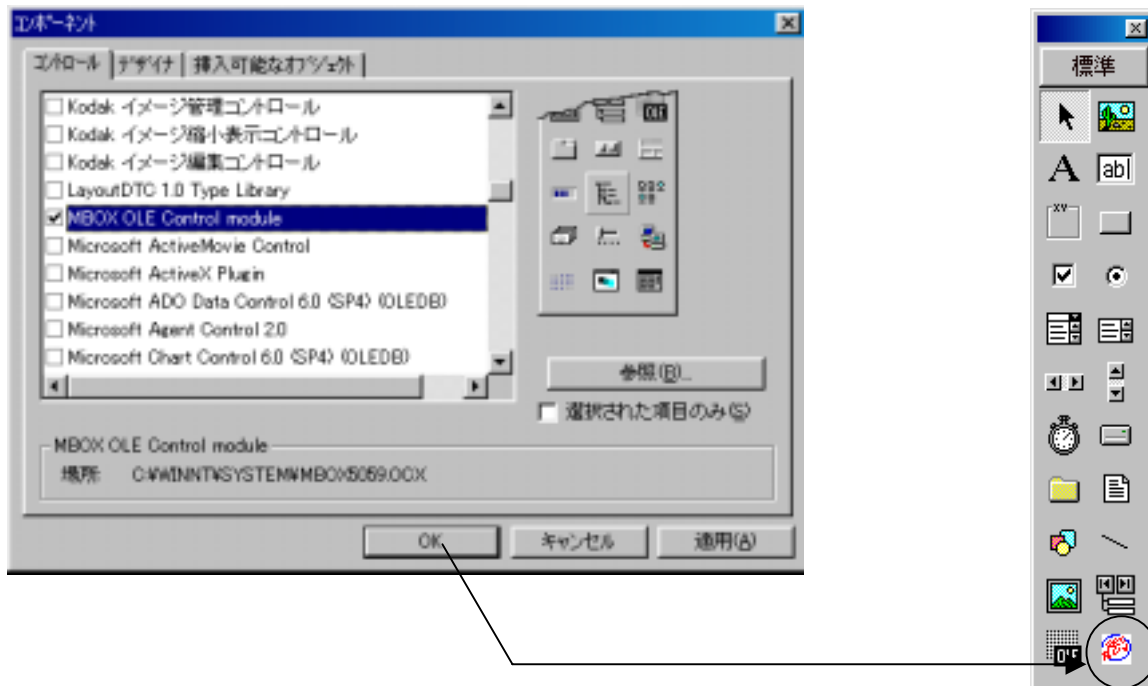
Step.2 => MBOX OLE Control Module の追加

(割り込みサービス使用時必須)

VB 5.0/6.0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。この時、左下図の「場所:」に、Step.1 で登録したパス名が表示されているか確認してください。

「OK」を押すと、VB ツールボックスに下記のツールボタンが追加されます。

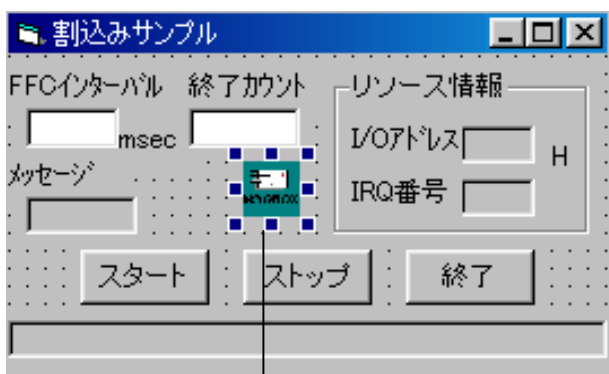
▶ VB 5.0/6.0 の場合



Step.3 => 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)

フォームに、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

MBOX5059_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。



```

MBOX5059 OnMsgPost
Private Sub MBOX5059_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)
    ' IParamをラベルに表示する
    MsgLabel.Caption = Str(lParam)
    ' 割り込み回数が指定回となった時
    If lParam = StopCount Then
        UppEndEventSyncInt ' 割り込み解除
        OutUpp MyIOAdrs, IER1, &H0
        OutUpp MyIOAdrs, USCR, &H0
        StatusLabel.Caption = "メッセージ指定回数終了"
    End If
End Sub

```

(5-3-3) Visual BASIC サンプルプログラム

REX-5059 UPP PC カードを制御するアプリケーションを Visual BASIC で開発する場合は、本製品添付のフォームモジュール(*.frm)を参考にして下さい。

=>サンプルプログラムの実行

サンプルプログラムには、

実行例 1 : ロータリーエンコーダからの2相パルス入力アップダウンカウント

実行例 2 : A/D 変換

実行例 3 : FFC インターバル割り込み

があります。ソースコードは添付のディスクを参照してください。

次頁より**実行例**の解説をいたします。

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、モジュール定義ファイルの Declare 宣言を VBUPPLIB.BAS に示すように変更し、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 UPPLIB2K.DLL ではサポートしておりませんのでご注意ください。

ShowCardUtil()
UppGetVersion()
UppCardReset()

実行例 1 ロータリーエンコーダからの2相パルス入力アップダウンカウント
TPC コマンドを使用して、UPP に入力された2相パルスのカウントを行います。
配線図及び実行画面につきましては、(5-2-2)節を参照してください。

田 サンプルプログラム抜粋

➤ ファンクションテーブルの選択及びレジスタ設定

```
'TPC コマンド用ファンクションテーブルを設定
Sub SetFuncTblForSyncInt()
  Dim FuncNo, TblNo As Integer
  Dim UPPFuncTbl(16) As UPPFuncTableType
  'ファンクションテーブル設定
  UPPFuncTbl(0).FNReg = 1
  UPPFuncTbl(0).CMReg = &H98      'TPC コマンド設定
  UPPFuncTbl(0).RASRegA = 0      'UDR0 にカウント値を出力
  UPPFuncTbl(0).RASRegB = 255   'Don't Care
  UPPFuncTbl(0).IOARegA = &H40  'U0 の立下りを検出
  UPPFuncTbl(0).IOARegB = &H1   'U1 のエッジを検出ししない
  UPPFuncTbl(0).IOARegC = 0     'クロック入力ピン
  UPPFuncTbl(0).IOARegD = 1     'パルス入力ピン

  FuncNo = 1
  For TblNo = 1 To 15
    FuncNo = FuncNo + 1
    If ((FuncNo Mod 5) = 0) Then
      FuncNo = FuncNo + 1
    End If
    UPPFuncTbl(TblNo).FNReg = FuncNo
    UPPFuncTbl(TblNo).CMReg = &HFF
    UPPFuncTbl(TblNo).RASRegA = 255
    UPPFuncTbl(TblNo).RASRegB = 255
    UPPFuncTbl(TblNo).IOARegA = &HFF
    UPPFuncTbl(TblNo).IOARegB = 255
    UPPFuncTbl(TblNo).IOARegC = 255
    UPPFuncTbl(TblNo).IOARegD = 255
  Next TblNo
  'ファンクションテーブルの内容をレジスタに設定
  For TblNo = 0 To 15
    OutUpp MyIOAdrs, FNR + 0, UPPFuncTbl(TblNo).FNReg
    OutUpp MyIOAdrs, FNR + 1, UPPFuncTbl(TblNo).CMReg
    OutUpp MyIOAdrs, FNR + 2, UPPFuncTbl(TblNo).RASRegA
    OutUpp MyIOAdrs, FNR + 3, UPPFuncTbl(TblNo).RASRegB
    OutUpp MyIOAdrs, FNR + 4, UPPFuncTbl(TblNo).IOARegA
    OutUpp MyIOAdrs, FNR + 5, UPPFuncTbl(TblNo).IOARegB
    OutUpp MyIOAdrs, FNR + 6, UPPFuncTbl(TblNo).IOARegC
    OutUpp MyIOAdrs, FNR + 7, UPPFuncTbl(TblNo).IOARegD
  Next TblNo
End Sub
```


➤ パルス入力アップダウンカウントチェック

```
Private Sub PaintTimer_Timer()  
    Dim Count As Long 'UDRO(カウント値)  
    Dim Phase As Integer '位相角  
    Dim Rotation As Integer '回転数  
  
    'カウント数取得  
    Count = 256 * (InUpp(MyIOAdrs, UDROH) And &HFF) + (InUpp(MyIOAdrs, UDROL) And &HFF)  
  
    '7FFFH < Count 数 <= FFFFH のとき  
    If BasicCount < Count Then  
        Rotation = (Count - BasicCount) ¥ 360 '360P/R  
        Phase = (Count - BasicCount) Mod 360  
  
        RotNumLabel.Caption = Str(Rotation)  
        PhaseLabel.Caption = Str(Phase)  
  
        'ピクチャーボックスに描画  
        PhasePictureBox.Line (0, 0)-(Phase * PhasePictureBox.ScaleWidth / 360,  
                               PhasePictureBox.ScaleHeight), RGB(0, 0, 255), BF  
        PhasePictureBox.Line (Phase * PhasePictureBox.ScaleWidth / 360, 0)  
                               -(PhasePictureBox.ScaleWidth, PhasePictureBox.ScaleHeight), RGB(180, 180, 180), BF  
  
    '0000H <= Count 数 < 7FFFH のとき  
    ElseIf BasicCount > Count Then  
        Rotation = (BasicCount - Count) ¥ 360 '360P/R  
        Phase = (BasicCount - Count) Mod 360  
  
        RotNumLabel.Caption = " - " + Str(Rotation)  
        PhaseLabel.Caption = Str(360 - Phase)  
  
        'ピクチャーボックスに描画  
        PhasePictureBox.Line (0, 0)-((PhasePictureBox.ScaleWidth - Phase * PhasePictureBox.ScaleWidth  
                                       / 360), PhasePictureBox.ScaleHeight), RGB(0, 0, 255), BF  
        PhasePictureBox.Line ((PhasePictureBox.ScaleWidth - Phase * PhasePictureBox.ScaleWidth / 360),  
                               0)-(PhasePictureBox.ScaleWidth, PhasePictureBox.ScaleHeight), RGB(180, 180, 180), BF  
    End If  
  
    If Count >= 65527 Or Count <= 7 Then  
        OutUpp MyIOAdrs, UDROH, &H7F  
        OutUpp MyIOAdrs, UDROL, &HFF  
    End If  
End Sub
```

実行例 2 スキャンモードでの A/D 変換

プログラムでは、指定周期の割り込み信号に同期したポストメッセージを受け取り、メッセージ処理内で、AD 変換データのサンプリングを行っています。

配線図及び実行画面につきましては、(5-2-2)節を参照してください。

■ サンプルプログラム抜粋**➤ 変換データの取得**

```
Private Sub MBOX5059_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)
    Dim Status As Long '関数戻り値
    Dim ch As Byte 'ループカウンタ
    Dim dh As Long 'A/D データレジスタ(H)格納
    Dim dl As Long 'A/D データレジスタ(L)格納

    'AD 変換終了待ち--->1 スキャン終了で bit7 に 1 がたつ
    While (InUpp(MyIOAdrs, ADCSR) And &H80) = 0
        If StopFlag = True Then
            Exit Sub
        End If
    Wend

    '変換データの読込
    For ch = 0 To ScanCH
        dh = InUpp(MyIOAdrs, (&H7 + ch * 2))
        dl = InUpp(MyIOAdrs, (&H8 + ch * 2))
        dh = dh And &HFF
        dl = dl And &HFF

        '配列 AdBuf() にデータを格納
        AdBuf((ScanCH + 1) * (lParam - 1) + ch) = (dh * 4) + (dl ¥ 64)
    Next ch

    '現在のサンプリング回数(1チャンネル)を取得
    GetCount = lParam
    MessageLabel.Caption = "変換回数 (/1ch): " + Str(GetCount) + "個"

    ' サンプリング回数まで到達した時
    If GetCount = SampCount Then
        UppEndEventSyncInt '割り込み解除
        OutUpp MyIOAdrs, ADCSR, &H0 'AD 変換停止
        OutUpp MyIOAdrs, IER1, &H0 '割り込みマスク
        OutUpp MyIOAdrs, USCR, &H0 'UPP 停止
        Exit Sub
    End If
End Sub
```

➤ AD 変換開始

```
Private Sub StartCB_Click()  
(省略) . . . . .  
    '配列変数の割当て  
    ReDim AdBuf((ScanCH + 1) * SampCount)  
  
    'MBOX5059(OCX)のウインドウハンドル取得  
    OleHandle = MBOX5059.GetMboxWnd  
  
    '初期化  
    GetCount = 0  
    StopFlag = False  
    SaveCB.Enabled = True  
  
    '割り込み信号に同期してメッセージをポストする  
    Status = UppStartEventSyncInt(OleHandle, MyIOAdrs, MyIrqNo, ISCR1, SampCount)  
    'エラー処理  
    If Status <> 0 Then  
        MessageLabel.Caption = "割り込み開始エラー-[%d]" + Str(Status)  
        Exit Sub  
    End If  
  
    '割り込みルチの登録・割り込みイェブル => U0 の信号の立下りエッジにより割り込み発生  
    OutUpp MyIOAdrs, IER1, &H1  
    'ファンクション 1 から実行( 1 ファンクション実行後、インクリメントされ、次の  
    'ファンクションを実行 => FNR=MFNR でリセット)  
    OutUpp MyIOAdrs, FNR, &H1  
    'AD 変換スタート  
    OutUpp MyIOAdrs, ADCSR, &H30 Or ScanCH  
    'UPP 動作(ファンクション実行)  
    OutUpp MyIOAdrs, USCR, &H2  
    . . . . .  
End Sub
```

実行例3 FFC インターバル割り込み

FFC コマンドでパルス出力を行い、そのパルスを割り込み信号として、割り込みに同期したメッセージをアプリケーションに送ります。

サンプルの実行画面につきましては、(5-2-2)節を参照してください。

田 サンプルプログラム抜粋

➤ ファンクションテーブルの選択及び設定

```
'FFC コマンド用ファンクションテーブルを設定
Sub SetFuncTblForSyncInt()
    Dim FuncNo, TblNo As Integer

    'ファンクションテーブル設定
    UPPFuncTbl(0).FNReg = 1           'FFC コマンド設定
    UPPFuncTbl(0).CMReg = &H80       'UDR1 は内部クロックをカウント
    UPPFuncTbl(0).RASRegA = 1        'UDR1 にカウンタ値を出力
    UPPFuncTbl(0).RASRegB = 0        'コンペアレジスタ UDR0
    UPPFuncTbl(0).IOARegA = &HFF     'クロック入力なし
    UPPFuncTbl(0).IOARegB = 255      'Don't Care
    UPPFuncTbl(0).IOARegC = 0        'パルス出力ピン U0
    UPPFuncTbl(0).IOARegD = 255     'Don't Care

    UPPFuncTbl(1).FNReg = 2           'FFC コマンド設定
    UPPFuncTbl(1).CMReg = &H88       'UDR3 は U0 のエッジをカウント
    UPPFuncTbl(1).RASRegA = 3        'UDR3 にカウンタ値を出力
    UPPFuncTbl(1).RASRegB = 2        'コンペアレジスタ UDR2
    UPPFuncTbl(1).IOARegA = &H20     'U0 の立ち下がりエッジを検出
    UPPFuncTbl(1).IOARegB = 255      'Don't Care
    UPPFuncTbl(1).IOARegC = 1        'パルス出力ピン U1
    UPPFuncTbl(1).IOARegD = 255     'Don't Care
```

(次頁へ)

(前頁から)

```

FuncNo = 2
For TbIno = 2 To 15
    FuncNo = FuncNo + 1
    If ((FuncNo Mod 5) = 0) Then
        FuncNo = FuncNo + 1
    End If
    UPPFuncTbl(TbIno).FNReg = FuncNo
    UPPFuncTbl(TbIno).CMReg = &HFF
    UPPFuncTbl(TbIno).RASRegA = 255
    UPPFuncTbl(TbIno).RASRegB = 255
    UPPFuncTbl(TbIno).IOARegA = &HFF
    UPPFuncTbl(TbIno).IOARegB = 255
    UPPFuncTbl(TbIno).IOARegC = 255
    UPPFuncTbl(TbIno).IOARegD = 255
Next TbIno

```

'ファンクションテーブルの内容をレジスタに設定

```

For TbIno = 0 To 15
    OutUpp MyIOAdrs, FNR + 0, UPPFuncTbl(TbIno).FNReg
    OutUpp MyIOAdrs, FNR + 1, UPPFuncTbl(TbIno).CMReg
    OutUpp MyIOAdrs, FNR + 2, UPPFuncTbl(TbIno).RASRegA
    OutUpp MyIOAdrs, FNR + 3, UPPFuncTbl(TbIno).RASRegB
    OutUpp MyIOAdrs, FNR + 4, UPPFuncTbl(TbIno).IOARegA
    OutUpp MyIOAdrs, FNR + 5, UPPFuncTbl(TbIno).IOARegB
    OutUpp MyIOAdrs, FNR + 6, UPPFuncTbl(TbIno).IOARegC
    OutUpp MyIOAdrs, FNR + 7, UPPFuncTbl(TbIno).IOARegD
Next TbIno
End Sub

```

➤ 割り込みポストメッセージ処理

```
Private Sub MBOX5059_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)
```

```

    'lParam をラベルに表示する
    MsgLabel.Caption = Str(lParam)
    '割り込み回数が指定回となった時
    If lParam = StopCount Then
        UppEndEventSyncInt '割り込み解除
        OutUpp MyIOAdrs, IER1, &H0
        OutUpp MyIOAdrs, USCR, &H0
        StatusLabel.Caption = "メッセージ指定回数終了"
    End If
End Sub

```

第6章 MS-DOS/Windows3.1 解説

(6-1) MS-DOS/Windows3.1 でのインストール

(6-1-1) イネーブラのインストール

MS-DOS/Windows3.1 で PC カードのイネーブルを行うために、イネーブラのインストールを行う必要があります。DOS/V をお使いの場合は、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ので、最初にどちらを使用するか選択してください。

DOS/V 版カードサービス対応イネーブラのインストール

添付フロッピーディスクからハードディスクに DOS/V 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。

```
C:>COPY A:%PCMCIA%DOSV%CSV%UPPCARDV.EXE C:%CARD
```

UPPCARDV.EXE はデバイスドライバ形式ですので、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

添付フロッピーディスク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:%PCMCIA%DOSV%I365%UPP365.EXE C:%CARD
```

UPP365.EXE は、DOS プロンプトから実行します。

PC-98 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。

```
C:>COPY B:%PCMCIA%PC98%UPPCRD98.EXE A:%CARD
```

UPPCRD98.EXE はデバイスドライバ形式ですので、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 カードにコンフィギュ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、コンフィギュレーションソフト「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 カードにコンフィギュレーションします。イネーブラによるコンフィギュ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。

☐ DOS/V 版対応カードサービスについて...

カードサービスはパソコン本体に添付しているソフトウェアでソケットサービス(SS)・カードサービス(CS)・リソースマネージャ・コモンイネーブラ等のドライバがセットになっています。本製品は PCMCIA Release 2.0 以降の下記カードサービスに対応しています。

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
IBM 版 PlayAtWill 2.xx / 3.xx	IBMDSS01.SYS, IBMDOSCS.SYS	IBM ThinkPad
IBM 版 PCMCIA 2.00 相当	IBMDSS01.SYS, IBMDOSCS.SYS	IBM ThinkPad Panacom PRONOTE jet
IBM PCMCIA 1.07 相当	IBMDSS02.SYS, IBMDOSCS.SYS	IBM ThinkPad
SystemSoft 版 CardSoft PCMCIA2.01 相当 v4.1x PCMCIA2.10 相当 v2.0x	SS365SL.EXE,SSCIRRUS.EXE, SSDBOOK.EXE, SSVADEM.EXE, CS.EXE,CSALLOC.EXE	SOTEC WiNBooK, IDEXON NT66CL2, DELL Latitude
SystemSoft 版 CardSoft PCMCIA2.0 相当 v2.0x	SSVLSI.EXE、CS.EXE,CSALL OC.EXE	COMPAQ CONTURA AERO 4/25,4/33C
Phoenix Technologies 版 CARD Manager Plus PCMCIA2.00 相当 v1.0 PCMCIA2.1 相当 v2.2x	PCCMSS.EXE, PCMCS.EXE	FUJITSU FMV Note, TOSHIBA DynaBook
DATABOOK 版 CardTalk	SOCKET.SYS, CTALKCCS.EXE, CARDTALK.SYS	MDT Arowana

注 1)PCMCIA ドライバとして、Phoenix Technologies の PCMCUU が提供されている機種 (Olivetti QUADERUNO 33/J)では動作しません。IBM PC-DOSS J6.1/V,6.3/V または、PlayAtWill 等のカードサービスを別途お買い求めになるか、PCMCUU を登録しないで本製品添付のポイントイネーブラを使ってイネーブルしてください。

注 2)DATABOOK CardTalk v2.20.12,v2.20.12 はソケットサービスしかサポートしてませんので動作しません (PCiN P-NOTE,AT&T WaveNote,MDT, Arowana の発売初期の機種)。カードサービス版の CardTalk を入手してください。

(6-1-2) DOS/V 版カードサービス版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。カードサービスのインストール方法については、パソコン側のマニュアル記載内容に従ってください。カードサービスのインストールが完了していれば、本製品添付のカードサービス版イネーブラをカードサービスの後に追加するだけです。次頁以降に CONFIG.SYS の登録例を示します。CONFIG.SYS の内容はお使いの機種によってまちまちですので、登録例の通りに修正する必要はありません。

⇒ オプション仕様

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

オプション : /P = x : I/O ベースアドレス x を 16 進表記で指定

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てます。

/I = x : 割り込み番号 x を 10 進表記で指定

何も指定しない場合は、割り込みは使用しません。
指定可能な割り込み番号は、5,7,10,11,12,15 になります。

/S = n : スロット番号 n を指定

/S オプションを省略した場合、または /S=0 が指定された場合はスロットを順に調べてイネーブルします。スロットを指定する場合は、/S=1 から /S=4 を追加します。

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\UPPCARDV.EXE /P=300 /I=5 (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。
I/O ベースアドレス 300h・IRQ5 を割り当ててくれるように指定しています。

CONFIG.SYS 記述例2 => COMPAQ カードサービスの場合

```
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\UPPCARDV.EXE /P=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。
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 カードサービスの場合

```
DEVICE=C:\DOS\EMM386.EXE RAM P0=D000 P1=D400 P2=D800 P3=DC00 I=B000-B7FF
X=CC800-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\UPPCARDV.EXE /S=300 /I=5 (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。(行に記述してください)
カードに I/O ベースアドレス 300h・IRQ5 を割り当てます。

(6-1-3) DOS/V 版ポイントインーブラを使用する場合

カードサービスが提供されていない機種でUPP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、ポイントインーブラを使ってカードのイネーブルを行います(注1)。

ポイントインーブラは、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、<X=>オプションで [DF000h ~ DFFFFh] の 4K バイトのメモリーウィンドウを確保してください。

```
C:\CARD>UPP365.EXE [/<オプション>] [ ] … [ ]
```

オプション : /P = x : I/O ベースアドレス x を 16 進表記で指定

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てます。

/I = x : 割り込み番号 x を 10 進表記で指定

何も指定しない場合は、割り込みは使用しません。
指定可能な割り込み番号は、5,7,10,11,12,15 になります。

/S = n : スロット番号 n を指定

/S オプションを省略した場合、または /S=0 が指定された場合はスロットを順に調べてイネーブルします。スロットを指定する場合は、/S=1 から /S=4 を追加します。

/MEM = x : 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

指定しないときは DF00 から 4K バイトを使います。
EMM386.EXE の <X=> オプションでイクスクルードしたメモリーウィンドウの範囲と一致するようにしてください。

(注1) 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。

(6-1-4) PC-98 版カードサービス版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。カードサービスのインストール方法については、パソコン側のマニュアル記載内容に従ってください。カードサービスのインストールが完了していれば、本製品添付のカードサービス版イネーブラをカードサービスの後に追加するだけです。次頁以降に CONFIG.SYS の登録例を示します。

```
DEVICE=A:¥CARD¥UPPCRD98.EXE [/<オプション>] [] … []
```

オプション : /P = x : I/O ベースアドレス x を 16 進表記で指定


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てます。

/I = x : 割り込み番号 x を 10 進表記で指定

何も指定しない場合は、割り込みは使用しません。
指定可能な割り込み番号は、3,5,6,10,12,13 になります。

/S = n : スロット番号 n を指定

/S オプションを省略した場合、または /S=0 が指定された場合はスロットを順に調べてイネーブルします。スロットを指定する場合は、/S=1 から /S=4 を追加します。

 PC-98 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購入してください。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
標準カードサービス SystemSoftCardSSoft2.10 Version2.06	SSDRV.EXE, CS.EXE	PC-9821 Np,Ns,Ne2,Nd,Ld Nf,Nm,Lt,Ne3,Nd2 PC-9801 NL/A

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

CONFIG.SYS 記述例4 => NEC 添付のカードサービス

PC-9821 Np,Ns,Ne22,Nd,Ld,Nf,Nm,Lt,Ne3,Nd2
PC-9801 NL/A

```

DEVICE=A:¥DOS¥HIMEM.SYS
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¥UPPCRD98.EXE /P=D0 /I=3             (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) 本製品添付のカードサービス版イネーブラを起動しています。
カードに I/O ベースアドレス D0h・IRQ3 を割り当てます。

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

```

DEVICE=A:¥DOS¥HIMEM.SYS
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¥UPPCRD98.EXE /P=D0 /I=3             (6)

```

【解説】

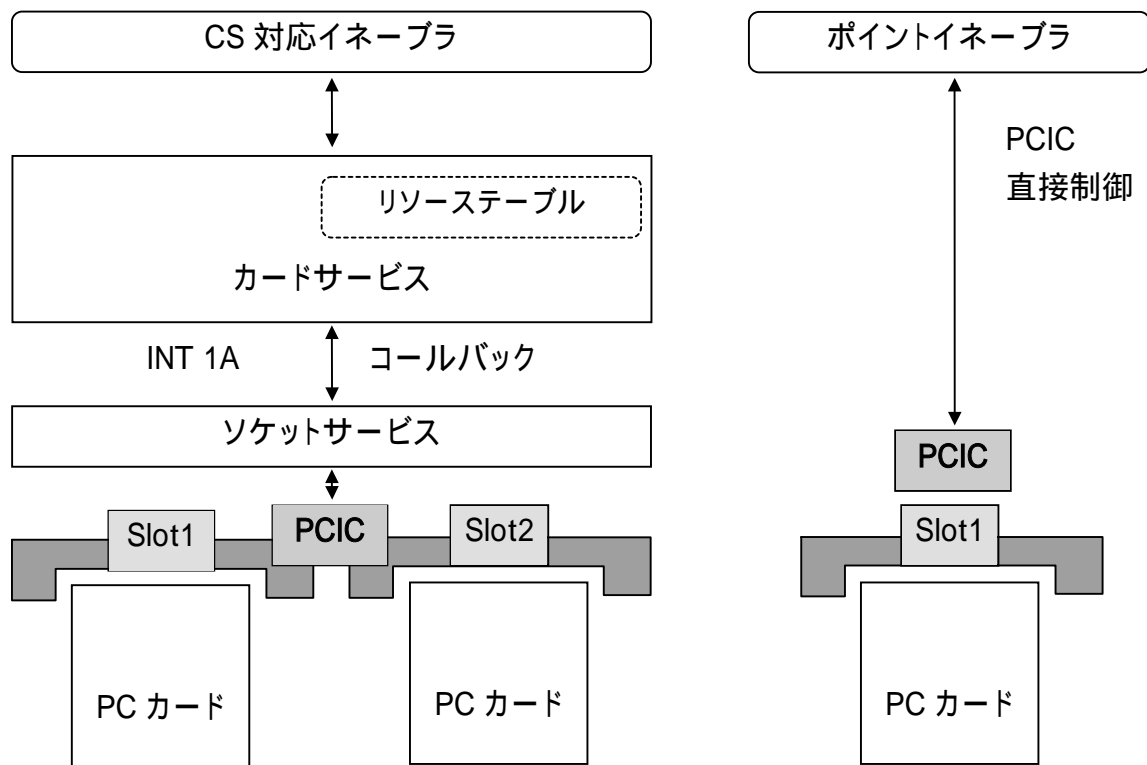
上記解説参照

- (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。
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。

☐ 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。



(6-2) MS-DOS ライブラリ

Microsoft Visual C++ Version 1.0 で作成したスモールモデル及びラージモデルのライブラリが添付されています。UPP カードに入出力を行うための関数とカードサービスから UPP カードに割り当てられた I/O アドレスおよび IRQ リソース情報を問い合わせための関数が提供されています。

=>ライブラリモデル	スモールモデル	SLIBUPP.LIB
	ラージモデル	LLIBUPP.LIB

=>インクルードファイル UPPLIB.H

(6-2-1) MS-DOS ライブラリ関数

OutUpp	UPP ポートに1バイトを出力
---------------	------------------------

書 式 void **OutUpp** (WORD wBase, WORD UppIndex, BYTE Val)

機 能 1バイトをポートに出力

引 数 WORD wIOBase : カードのベースアドレス
 WORD UppIndex : UPP レジスタインデックス
 BYTE Val : バイト出力値

戻 値 なし

解 説 UPP のレジスタへ出力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し出力を行います。Microsoft C のランタイムライブラリにあるポートへの出力命令を使うと下記のコーディングを行う必要があります。

outp (ベースアドレス + 3, インデックス上位バイト);

outp (ベースアドレス + 2, インデックス下位バイト);

outp (ベースアドレス + 0, 出力データ);

関数 **OutUpp()**を使うことにより下記のように1行で記述することができます。

OutUpp (ベースアドレス, インデックス, 出力データ);

InUpp**UPP ポートから1バイト入力**

書式	BYTE InUpp (WORD IOBase, WORD RegAddr)
機能	ポートから1バイト読み込む
引数	WORD IOBase : カードのベースアドレス WORD RegAddr : UPP レジスタアドレス
戻値	バイト入力値
解説	<p>UPP のレジスタへ入力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し入力を行います。Microsoft C のランタイムライブラリにあるポートへの入力命令を使うと下記コーディングになります。</p> <pre>outp (ベースアドレス + 3, インデックス上位バイト); outp (ベースアドレス + 2, インデックス下位バイト); 入力データ = inp (ベースアドレス);</pre> <p>関数 InUpp() を使うことにより下記のように1行で記述することができます。</p> <pre>入力データ = InUpp (ベースアドレス, インデックス);</pre>

CheckCSRegistration**カードサービス常駐チェック**

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 *pIOAdrs : I/O アドレス格納先を示すポインター WORD *pIRQNo : 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。

(6-2-2) MS-DOS サンプルプログラム

下記のサンプルプログラムが添付されています。詳細は添付ディスクを参照してください。

UppAdCon.c	A/D 変換実行例
UppCheck.c	カウンタタイマーとしての実行例
UppDio.c	DIO 出力の例
UppChOut.c	一定時間間隔で文字列を表示する例
UppEcode.c	オムロン製エンコーダを使った角度表示例
UppMotor.c	ステッピングモータ制御例

=>A/D 変換実行例について

UPPのパルス入出力機能を使用し、一定時間ごとのインターバルを作成します。プログラムでは、このインターバルを使用して、A/D 変換をスタートさせ、一定時間ごとに A/D データを入力表示します。A/D コンバータはスキャンモードを使用し、チャンネル0~3のデータを入力します。次頁以降にソースリストを参照してください。

```

#include <stdio.h >
#include <conio.h >

#include "UPPLIB.H"

BYTE fdata[16][8] = {
    /* FNR, CMR,RASRA,RASRB,IOARA,IOARB,IOARC,IOARD*/
    { 1, 0x80, 1, 0, 0xff, 255, 0, 255 }, /* FFC コマンド*/
    { 2, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 3, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 4, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 6, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 7, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 8, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 9, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 11, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 12, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 13, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 14, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 16, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 17, 0xff, 255, 255, 0xff, 255, 255, 255 }, { 18, 0xff, 255, 255, 0xff, 255, 255, 255 },
    { 19, 0xff, 255, 255, 0xff, 255, 255, 255 }
};

void main( void )
{
    WORD IOBase, count, fun, reg, udr, ch, dt;
    BYTE dh, dl;
    double v;

    do{
        printf( "¥x0cR E X 5 0 5 9レジスタベースアドレス $" );
        scanf( "%x", &IOBase );
    }while( ( IOBase<0x120) || ( IOBase > 0x350 ) );

    OutUpp( IOBase, USCR, 0x00 ); /* U P C 停止 */
    OutUpp( IOBase, IER3, 0x00 ); /* 割り込みマスク(UPP IER3) */
    OutUpp( IOBase, IER2, 0x00 ); /* 割り込みマスク(UPP IER2) */
    OutUpp( IOBase, IER1, 0x00 ); /* 割り込みマスク(UPP IER1) */
    OutUpp( IOBase, UOR2, 0x00 ); /* UPP Output Reg.2 Output Clear */
    OutUpp( IOBase, UOR1, 0x00 ); /* UPP Output Reg.1 Output Clear */
    OutUpp( IOBase, MFNR, 1 ); /* Maximum Function Number Reg. (0 Function) */
    OutUpp( IOBase, FNR, 0x00 ); /* Function Number Reg. CLEAR */
    OutUpp( IOBase, USCR, 0x02 ); /* U P C 動作(サンプリングフリップフロップクリア) */
    OutUpp( IOBase, USCR, 0x00 ); /* U P C 停止 */
    OutUpp( IOBase, DDR2, 0xff ); /* Data Direction Reg.2 ALL OUTPUT */
    OutUpp( IOBase, DDR1, 0xff ); /* Data Direction Reg.1 ALL OUTPUT */
    OutUpp( IOBase, UCER2, 0xff ); /* UPP Contact Enable Reg.2 Pulse I/O Enable */
    OutUpp( IOBase, UCER1, 0xff ); /* UPP Contact Enable Reg.1 Pulse I/O Enable */
    OutUpp( IOBase, MFNR, 20 ); /* Maximum Function Number Reg. (16 Function 5us) */

    /* ファンクションテーブルにデータセット */
    for( fun=0;fun<=15;fun++ )
        for( reg=0;reg<=7;reg++ ){
            OutUpp( IOBase, FNR+reg, fdata[fun][reg] );
        }

    /* A/D サンプリング周期 0.5s */
    count=50000;
    OutUpp( IOBase,UDR0H, (BYTE)(count >> 8) ); /* UPP Data Reg. 0 上位 */
    OutUpp( IOBase,UDR0L, (BYTE)(count & 0xff) ); /* UPP Data Reg. 0 下位 */
}

```

(次頁に続く)

```

/* データレジスタを全てクリア */
for( udr=1;udr<=7;udr++){
    OutUpp( IOBase,UDR0H+udr*2, 0 );
    OutUpp( IOBase,UDR0L+udr*2+1, 0 );
}
for( udr = 8; udr <= 15; udr++){
    OutUpp( IOBase, UDR8H+(udr-8)*2, 0 );
    OutUpp( IOBase, UDR8L+(udr-8)*2+1, 0 );
}
for( udr = 16; udr <= 23; udr++){
    OutUpp( IOBase, UDR16H+(udr-16)*2, 0 );
    OutUpp( IOBase, UDR16L+(udr-16)*2+1, 0 );
}
/* ファンクション1から実行 */
OutUpp( IOBase, FNR, 0x01 );

/* A/D スキャンモード 0-3 */
OutUpp( IOBase, ADCSR, 0x33 );

/* 1回目 A/D 変換終了 */
while( ( InUpp( IOBase, ADCSR ) & 0x80 ) == 0 ){
    kbhit();
}

/* U P C 動作(ファンクション実行) */
OutUpp( IOBase, USCR, 0x02 );

while( kbhit() )
    getch();          /* 先行入力引き取り */

while( !kbhit() )
{
    while( ( InUpp( IOBase, ISR1 ) & 0x01 ) == 0 )           /* インタラプトステータスリード */
        printf( " " );                                     /* U0 立ち下がりエッジ検出 */

    OutUpp( IOBase, ISCR1, 0x00 );                           /* インタラプトステータスクリア */
    for( ch = 0; ch <= 3; ch++ )
    {
        dh = InUpp( IOBase, ADDR0H+(ch & 0x03)*2 );         /* A/D 上位8bit 読み込み */
        dl = InUpp( IOBase, ADDR0L+(ch & 0x03)*2 );         /* A/D 下位2bit 読み込み */
        dt = (WORD)((dh << 2)+(dl >> 6));
        v = (double)dt * 5.0 / 1024.0;
        printf( "CH%1d $%03X %5.3f[V] ",ch,dt,v );
    }
    printf( "\n" );
}
}

```

(6-3) Windows3.1 ライブラリ

Microsoft Visual C++ Version 1.0 で作成したダイナミックリンクライブラリ (DLL) が添付されています。UPP カードに入出力を行うための関数とカードサービスから UPP カードに割り当てられた I/O アドレスおよび IRQ リソース情報を問い合わせための関数が提供されています。

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します。Visual BASIC 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モジュール定義ファイル (.BAS) で Declare 宣言します。

ライブラリ名 WUPPLIB.DLL (16Bit Version)
 インクルードファイル WUPPLIB.H

(注記)

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 をサポートしていません (実際はインプリメントされている)。しかしながら、WindowsOS 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。

=>DEF ファイルのインポート宣言例 (Visual C)

```

NAME                    wUppSamp
DESCRIPTION            'UPP PC Card Sample Program (c)RATOC SYSTEM,Inc. 1996'
EXETYPE                WINDOWS
STUB                    'WINSTUB.EXE'
CODE                    PRELOAD MOVEABLE DISCARDABLE
DATA                    PRELOAD MOVEABLE MULTIPLE
HEAPSIZE               1024

IMPORTS                WUPPLIB.OutUpp
                        WUPPLIB.InUpp
                        WUPPLIB.OutPort
                        WUPPLIB.InPort
                        WUPPLIB.wOutPort
                        WUPPLIB.wInPort
                        WUPPLIB.CheckCSRegistration
                        WUPPLIB.GetCSConfigInfo
                        WUPPLIB.UppGetVersion
  
```

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

```

Declare Sub OutUpp Lib "WUPPLIB.DLL"
    (ByVal wBase As Integer, ByVal Reg As Integer, ByVal OutVal As Integer)
Declare Function InUpp Lib "WUPPLIB.DLL"
    (ByVal wBase As Integer, ByVal RegAddr As Integer) As Integer
Declare Sub OutPort Lib "WUPPLIB.DLL" (ByVal IOAddr As Integer, ByVal OutVal As Integer)
Declare Sub wOutPort Lib "WUPPLIB.DLL" (ByVal IOAddr As Integer, ByVal OutVal As Integer)
Declare Function InPort Lib "WUPPLIB.DLL" (ByVal IOAddr As Integer) As Integer
Declare Function wInPort Lib "WUPPLIB.DLL" (ByVal IOAddr As Integer) As Integer
Declare Function CheckCSRegistration Lib "WUPPLIB.DLL" () As Integer
Declare Function GetCSConfigInfo Lib "WUPPLIB.DLL"
    (ByVal Slot As Integer, lpIOAddr As Any, lpIOAddr As Any) As Integer
Declare Sub UppGetVersion Lib "WUPPLIB.DLL" (ByVal hWnd As Integer)

```

(6-3-1) Windows3.1 DLL 関数

OutUpp

UPP ポートに1バイトを出力

書式 void _export WINAPI **OutUpp**(WORD wBase, WORD RegAddr, BYTE Val)

機能 1バイトをポートに出力

引数 WORD wIOBase : カードのベースアドレス
 WORD UppIndex : UPP レジスタインデックス
 BYTE Val : バイト出力値

戻値 なし

解説 UPP のレジスタへ出力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し出力を行います。DLL ライブラリにあるポートへの出力命令を使うと下記コーディングになります。

OutPort (ベースアドレス + 3, インデックス上位バイト);

OutPort (ベースアドレス + 2, インデックス下位バイト);

OutPort (ベースアドレス + 0, 出力データ);

関数 **OutUpp**() を使うことにより下記のように1行で記述することができます。

OutUpp (ベースアドレス, インデックス, 出力データ);

InUpp**UPP ポートから1バイト入力**

- 書式 WORD _export WINAPI InUpp(WORD wBase, WORD RegIndex)
- 機能 ポートから1バイト読み込む
- 引数 WORD IOBase : カードのベースアドレス
WORD RegIndex : UPP レジスタアドレス
- 戻値 バイト入力値(上位バイトは無視してください)
- 解説 UPP のレジスタへ入力を行う場合、インデックスレジスタ0に UPP レジスタ番号の下位8ビットをセットし、インデックスレジスタ1に UPP レジスタ番号の上位8ビットをセットした後、データレジスタに対し入力を行います。DLL ライブラリにあるポートへの入力命令を使うと下記のコーディングを行う必要があります。
- ```
OutPort (ベースアドレス + 3, インデックス上位バイト);
OutPort (ベースアドレス + 2, インデックス下位バイト);
入力データ = InPort (ベースアドレス);
```
- 関数 InUpp() を使うことにより下記のように1行で記述することができます。
- ```
入力データ = InUpp ( ベースアドレス, インデックス );
```

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 *pIRQN : 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。

OutPort**ポートに1バイトを出力**

書式 void _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 なし

wOutPort**ポートに1ワードを出力**

書式 void _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 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バイト読み取ります

引数 WORD IOAddr : 出力する I/O ポートアドレス

戻値 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**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ワード読み取ります

引数 WORD IOAddr : 出力する I/O ポートアドレス

戻値 読み込んだワードデータを返します

UppGetVersion

DLL のバージョンダイアログ表示

書 式 void _export WINAPI UppGetVersion(HWND hWnd)

機 能 DLL のバージョンダイアログを呼び出します

引 数 HWND hWnd : 呼び出し側のウィンドウハンドル

戻 値 なし

(空白ページ)

発行 ラトックシステム株式会社
2002年11月19日 第2.0版 第1刷発行

製品に対するお問い合わせ

REX-5059 の技術的なご質問やご相談の窓口を用意しており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、インターネットのホームページの以下のアドレス
でも受け付けています。

HomePage ⇨ <http://www.ratocsystems.com>

◆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。
- 運用の結果につきましては、責任を負いかねますので、予めご了承ください。
- 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。