

PCI Bus, ISA Bus, PC/104

CTRボードシリーズ

(CCL3221搭載)

ユーザーズマニュアル 〈共通編〉

高速・多機能 32ビット・カウンタ・ボード



株式会社ハイバーテック

<http://www.hivertec.co.jp/>

この説明書は

次のCCL3221搭載CTRボードシリーズのボードに適応しています。

CTR520シリーズ・・PCI Bus

HPCI -CTR524F

HPCI -CTR522F

CTR220シリーズ・・ISA Bus

HPC -CTR224F

HPC -CTR222F

CTR120シリーズ・・PC/104

HPC104-CTR122F

本書及びプログラムの全部又は一部の無断転載、コピーを禁止します。
本製品の内容に関しましては、改良等により将来予告なしに変更することがあります。
本製品の内容についてお気づきの点がございましたら、お手数ながら当社までご連絡下さい。

Windows95, Windows98, WindowsNT 4.0, Windows2000, WindowsXP Home Edition, WindowsXP Professional,
VisualC++, Visual Basic, Microsoft C/C++ は Microsoft Corporation の米国及びその他の国における登録商標です。
その他、記載されている会社名、製品名は、各社の商標又は登録商標です。

株式会社 ハイバーテック
東京都墨田区両国4-8-1
ダイユービル

TEL 03-3846-3801

FAX 03-3846-3773

sales@hivertec.co.jp

第1.12版 2003年 8月22日発行
不許複製・転載

保証範囲

1. 本製品の保証期間は、お買い上げ頂いた日より3年間です。保証期間中に弊社の判断により欠陥が判明した場合には、本製品を弊社に引き取り、修理または交換を行います。
2. 保証期間内外に関わらず、弊社製品の使用、供給（納期）または故障に起因する、お客様及び第三者が被った、直接、間接、二次的な損害あるいは、遺失利益の損害に付いて、弊社は本製品の販売価格以上の責任を負わないものとしますので、予めご了承下さい。

免責事項

1. 本マニュアルに記載された内容に沿わない、製品の取付、接続、設定、運用により生じた損害に対しましては、一切の責任を負いかねますので、予めご了承下さい。
2. 本製品は、一般電子機器用（工作機械・計測機器・FA/OA機器・通信機器等）に製造された半導体製品を使用していますので、その誤作動や故障が直接、生命を脅かしたり、身体・財産等に危害を及ぼしたりする恐れのある装置（医療機器・交通機器・燃焼機器・安全装置等）に適用できるような設計、意図、または、承認、保証もされていません。
ゆえに本製品の安全性、品質および性能に関しては、本マニュアル（またはカタログ）に記載してあること以外は明示的にも黙示的にも一切保証するものではありませんので、予めご了承下さい。
3. 保証期間内外に関わらず、お客様が行った弊社の承認しない製品の改造または、修理が原因で生じた損害に対しましては、一切の責任を負いかねますので、予めご了承下さい。
4. 本マニュアルに記載された内容について、弊社もしくは、第三者の特許権、著作権、商標権、その他の知的所有権の権利に対する保証または実施権の許諾を行うものではありません。
また本マニュアルに記載された情報を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社は、その責任を負いかねますので、予めご了承下さい。

安全にお使い頂くために

この度は、弊社NCボードシリーズをご採用頂きまして、誠に有り難う御座います。

本書は、本製品をご使用して頂く場合の取扱い、留意点に付いて記入してありますので、必ずご一読の上ご利用をお願い致します。

尚、本マニュアルは、本書が添付されたNCボード常設箇所付近の分かりやすい場所に常時保管し、必要に応じて適宜参照・確認頂きますよう、お願い致します。

安全上の注意

本製品のご使用前に、必ずこのユーザーズマニュアル及び付属書類を全て熟読し、内容を理解してから正しくご使用下さい。本製品の知識、安全の情報及び注意事項の全てに付いて習熟してからご使用下さい。

本ユーザーズマニュアルでは、安全注意事項のランクを「警告」、「注意」として区分してあります。



警告

この表示を無視して、誤った取扱いをすると、人が死亡または重傷を負う可能性が想定される内容を示しています。



注意

この表示を無視して、誤った取扱いをすると、人が傷害を負う可能性または物的損害が想定される内容を示しています。

1. 対象ユーザー



注意

本製品およびマニュアルは、以下の様な、ユーザーを対象としています。



- ・ 拡張用ボードの増設および配線に付いて基本的な知識を有している方。
- ・ 制御用電子機器およびパソコン等に付いて基本的な知識を有している方。

2. 運搬・取り付け



警告



本製品にふれる前に、金属に触り身体の静電気を取り除いて下さい。
静電気は、本ボードの故障の原因になります。



本製品を静電気の帯びやすい梱包材（エアークラップなど）でくるまないで下さい。
静電気は、本ボードの故障の原因になります。



本製品のエッジコネクタ部分に触らないで下さい。
エッジコネクタ部分が汚れますと、誤動作の原因になります。



本製品の上に重いものを載せないで下さい。重いものを乗せると、部品が損傷し
故障の原因になります。



本製品のジャンパ設定は、パソコン等に取り付ける前に行ってください。電源がONの
状態で設定しますと、設定を正しく認識しないで誤動作の原因になります。



本製品のジャンパ設定は、正しく行って下さい。
設定を間違えますと誤動作の原因になります。



本製品をパソコン等に取り付ける時は、必ずパソコン等の電源をOFFにし、電源
コードを抜いてから作業を行ってください。電源コードを抜かないで作業を行った場合、
故障の原因になります。また、装置が思わぬ動作をすることがあります。



注意












本製品を落としたり乱暴に扱ったりしないで下さい。
衝撃や振動が故障の原因となります。






本製品の半田面を手で直接触らないで下さい。
部品の突起などにより怪我をする恐れがあります。

3. 配線

 警告	
	外線用コネクタへの配線作業や外線用コネクタの着脱は、パソコン等の電源をOFFし、電源コードを抜いてから行って下さい。 電源コードを抜かないで作業を行った場合、故障の原因になります。 また、装置が思ぬ動作をすることがあります。
	外線用コネクタへの配線は、コネクタ信号表などをよく確認し、正しく配線して下さい。 間違った配線をしますと、故障・焼損の原因になります。
	外部から供給する電源は、必ず定格以内でご使用下さい。定格以外で使用されますと、故障・焼損・誤動作の原因となります。
	入出力回路に接続する回路は、必ず定格電流・電圧以内でご使用下さい。定格以外で使用されますと、故障・焼損・誤動作の原因となります。
	外部配線用コネクタは、推奨のコネクタをご使用下さい。推奨以外のコネクタを使用されますと、接触不良などにより誤動作の原因となります。
	外部配線用コネクタは、必ずロックしてご使用下さい。ロックしないで使用されますと、コネクタが外れる、または接触不良などにより誤動作の原因となります。
	外部配線用ケーブルは、引っ張る、または重い荷重を掛けしないで下さい。コネクタが外れる、または接触不良などにより誤動作の原因となります。
	外部配線用ケーブルは、モーターの配線やAC電源ケーブルなど、ノイズの多い配線とは出来るだけ離して下さい。配線が近いとノイズが誤動作の原因となります。

4. 試運転・調整

 警告	
	本製品を使用し装置を動作させる時は、プログラムのデバッグを充分行ってから動作させて下さい。プログラムに間違いがあると、思わぬ動きをすることがあります。
	本製品に添付してあるプログラムを使用し装置を動作させる時、機械系に合った設定を行って動作を確認して下さい。機械系に合わない設定で動作を行うと思わぬ動きをすることがあります。

5. 廃棄



警告



本製品を廃棄する時は、関連する法律・規則に従って処理して下さい。

目 次

1. はじめに	1
1. 1 このマニュアルの表記について	1
2. CTRボードの機能・概要	2
3. ボード構成とポートアドレス	3
3. 1 ブロック図	3
3. 2 ポートアドレス	4
3. 3 ポートとレジスタ配置	5
4. コマンド・ステータス・レジスタ	6
4. 1 ポート及びレジスタの読み、書き込み	6
4. 2 レジスタ制御コマンド	6
4. 3 単独コマンド	6
4. 4 ステータス	7
4. 4. 1 ステータス (STS)	7
4. 4. 2 割込みステータスレジスタ (RIST)	7
4. 4. 3 割込み要因設定レジスタ (RIRQ)	8
4. 5 レジスタ	9
4. 5. 1 カウンタ (CTR1, CTR2)	9
4. 5. 2 比較データレジスタ (RCMP1~RCMP4)	9
4. 5. 3 イベントタイマレジスタ (ETMR)	9
4. 5. 4 ラッチレジスタ (LTCH1, LTCH2)	9
4. 5. 5 最大値レジスタ (MAX1, 2), 最小値レジスタ (MIN1, 2)	9
4. 5. 6 環境レジスタ1 (RENV1)	10
4. 5. 7 環境レジスタ2 (RENV2)	12
5. 基本的な設定と運用	14
5. 1 操作手順	14
5. 2 エンコーダ等パルス計数 (カウント)	14
5. 2. 1 カウントパルスの入力信号形式	14
5. 2. 2 カウントスタート/ストップ	16
5. 2. 3 カウンタクリア機能	16
5. 2. 4 カウンタラッチ機能	17
5. 3 同時ラッチ	17
5. 4 コンパレータ	18
5. 4. 1 コンパレータの機能	18
5. 4. 2 コンパレータ比較結果の確認	18
5. 4. 3 コンパレータ比較条件成立時の処理	18
5. 4. 4 コンパレータ比較方法	19
5. 5 イベントタイマ	20
5. 6 一致出力設定	21
5. 7 汎用入出力	22
5. 8 アップ/ダウンカウント時のカウンタ最大値・最小値の測定	22
5. 9 信号幅の計測	23
5. 10 割込み機構と割込み処理	24
5. 10. 1 割込み機構	24
5. 10. 2 割込み処理	25

6.	ソフトウェア編	26	
6.1	ソフトウェアの概要	27	
6.2	準備	28	
6.3	ドライバ関数の戻り値	31	
6.4	ドライバ関数詳細	32	
6.4.1	Windows版ドライバ関数	33	
(1)	ct520_GetDeviceCount()	ボード枚数の取得	33
(2)	ct520_GetDeviceInfo()	デバイス情報の取得	34
(3)	ct520_OpenDevice()	デバイスのオープン	35
(4)	ct520_CloseDevice()	デバイスのクローズ	36
(5)	ct520_rXYSts()	XYchステータスの読込	37
	ct520_rZUSts()	ZUchステータスの読込	
(6)	ct520_wXYCmd()	XYch制御コマンドの書込	38
	ct520_wZUCmd()	ZUch制御コマンドの書込	
(7)	ct520_rXYReg()	XYchレジスタの読込	39
	ct520_rZUReg()	ZUchレジスタの読込	
	ct520_wXYReg()	XYchレジスタの書込	
	ct520_wZUReg()	ZUchレジスタの書込	
(8)	ct520_rPortB()	オプションポートのバイト読込	40
	ct520_rPortW()	オプションポートのワード読込	
	ct520_wPortB()	オプションポートへバイト書込	
	ct520_wPortW()	オプションポートへワード書込	
(9)	ct520_rXYBuf()	XYch入出力バッファの読込	41
	ct520_rZUBuf()	ZUch入出力バッファの読込	
	ct520_wXYBuf()	XYch入出力バッファの書込	
	ct520_wZUBuf()	ZUch入出力バッファの書込	
6.4.2	DOS版ドライバ関数	42	
(1)	ct520_GetDeviceCount()	ボード枚数の取得	42
(2)	ct520_GetDeviceInfo()	デバイス情報の取得	42
(3)	ct520_OpenDevice()	デバイスのオープン	43
(4)	ct520_CloseDevice()	デバイスのクローズ	43
(5)	ct520_rXYSts()	XYchステータスの読込	44
	ct520_rZUSts()	ZUchステータスの読込	
(6)	ct520_wXYCmd()	XYch制御コマンドの書込	44
	ct520_wZUCmd()	ZUch制御コマンドの書込	
(7)	ct520_rXYReg()	XYchレジスタの読込	45
	ct520_rZUReg()	ZUchレジスタの読込	
	ct520_wXYReg()	XYchレジスタの書込	
	ct520_wZUReg()	ZUchレジスタの書込	
(8)	ct520_rPortB()	オプションポートのバイト読込	46
	ct520_rPortW()	オプションポートのワード読込	
	ct520_wPortB()	オプションポートへバイト書込	
	ct520_wPortW()	オプションポートへワード書込	
(9)	ct520_rXYBuf()	XYch入出力バッファの読込	47
	ct520_rZUBuf()	ZUch入出力バッファの読込	
	ct520_wXYBuf()	XYch入出力バッファの書込	
	ct520_wZUBuf()	ZUch入出力バッファの書込	
(10)	ct520_SetIntCall()	割込処理関数の登録/削除	48
(11)	ct520_GetDevVerNo()	バージョン番号の取得	49

図 表 目 次

1. はじめに	
表1. 1-1 CCL3221搭載CTRボードシリーズ製品ラインアップ	1
表1. 1-2 略称呼称	1
2. CTRボードの機能・概要	
3. ボード構成とポートアドレス	
図3. 1-1 HPCI-CTR524Fブロックダイア	3
表3. 2-1 HPCI-CTR524Fポート表	4
図3. 3-1 HPCI-CTR524Fポートとレジスタ配置	5
4. コマンド・ステータス・レジスタ	
表4. 2-1 レジスタ制御コマンド	6
表4. 3-1 単独コマンド	6
表4. 4-1 ステータス内容	7
表4. 4-2 割込みステータスの内容	7
表4. 4-3 割込み要因設定レジスタの内容	8
表4. 5-1 環境レジスタ1 (RENV1)	10~ 11
表4. 6-1 環境レジスタ2 (RENV2)	12~ 13
5. 基本的な設定と運用	
図5. 1-1 基本的な操作手順	14
図5. 2-1 カウントパルスの入力信号形式	14
図5. 2-2 各入力信号形式におけるカウント動作のタイミング	15
図5. 4-1 コンパレータ比較条件の設定	19~ 20
図5. 6-1 HPCI-CTR522F一致出力ルート選択	21
図5. 9-1 信号幅・エッジ間測定の条件設定	23
図5. 10-1 割込み機構 (PCI Bus)	24
6. ソフトウェア編	
表6. 1-1 各ボード別ファイル名・関数名対応表	26
表6. 1-2 ドライバ関数とボード種別の対応表	27
図6. 1-1 ソフトウェアの関連	27
表6. 3-1 ドライバ関数の戻り値	31
表6. 4-2 各言語の数値表記	32
表6. 4-1 ドライバ関数のデータ型	32

1. はじめに

このマニュアルは、高速カウンタ LSI CCL3221 (2ch) を搭載した高速・多機能 32ビット・アップ/ダウン・カウンタボード「CTRボードシリーズ」の 共通取扱説明書です。

「CCL3221搭載CTRボードシリーズ」はPCI Bus, ISA Bus, PC/104 の各Busに対して次の製品ラインアップがあります。

ch数	PCI Bus	ISA Bus	PC/104		
2ch	HPCI-CTR522F	HPC-CTR222F	HPC104-CTR122F		
4ch	HPCI-CTR524F	HPC-CTR224F			

表 1. 1-1 CCL3221搭載CTRボードシリーズ製品ラインアップ

製品には、通常次の説明資料が付属します。

- ① CTRボードシリーズ ユーザーズマニュアル < 共通編 > (このマニュアル)
- ② (個別ボード名) ユーザーズマニュアル < 個別編 > ・ ・ 製品のハードウェアの説明
ソフトウェアのスタートアップ説明
- ③ 添付ソフトウェア (FD)

1. 1 このマニュアルの表記について

カウンタ LSI CCL3221 は「2チャンネルのアップ/ダウンカウンタ と4組のコンパレータ および 1組のイベントタイマ」で構成されています。

2chのボードは 1組のCCL3221を搭載し、ch1をXch, ch2をYchと呼称します。同様にカウンタは XCTR, YCTRと呼びます。コンパレータはCMPn (n=1~4)と呼びます。

4chのボードは XYchの機能のCCL3221を#1CCLと称し、もう一方のそれを#2CCLと呼び、このch3をZch, ch4をUchと称します。主な呼称を次に掲げます。

名 称	呼 称		代表呼称	備 考
1, 2ch/3, 4ch	#1 CCL	#2 CCL	CCL	
チャンネル	Xch Ych	Zch Uch	ch1 ch2	
カウンタ	XCTR YCTR	ZCTR UCTR	CTR1 CTR2	32ビット アップ/ダウンカウンタ
コンパレータ	XYCMP1 XYCMP2 XYCMP3 XYCMP4	ZUCMP1 ZUCMP2 ZUCMP3 ZUCMP4	CMP1 CMP2 CMP3 CMP4	1個のCCLに4組
コンパレータ一致信号	XYCMPOUT1 XYCMPOUT2 XYCMPOUT3 XYCMPOUT4	ZUCMPOUT1 ZUCMPOUT2 ZUCMPOUT3 ZUCMPOUT4	CMPOUT1 CMPOUT2 CMPOUT3 CMPOUT4	1個のCCLに4組
比較データレジスタ	XYRCMP1 XYRCMP2 XYRCMP3 XYRCMP4	ZURCMP1 ZURCMP2 ZURCMP3 ZURCMP4	RCMP1 RCMP2 RCMP3 RCMP4	比較データレジスタも 4組あるが、 CMP1~CMP4の いずれにも対応する
イベントタイマ	XYETMR	ZUETMR	ETMR	周期割込みタイマ
イベントタイマ信号	XYETMROUT	ZUETMROUT	ETMROUT	周期割込みタイマ信号

表 1. 1-2 略称呼称

なお、本書の説明文中ではCCL3221搭載CTRボードシリーズのボードの名称を総じてCTRボードと呼称します。

この取扱い説明は最も標準的な PCI Bus 4ch CTR ボード「HPCI-CTR524F」を基本に説明します。

2. CTRボードの機能・概要

CTRボードは概略次の機能項目があります。

(1) エンコーダ等パルス計数(カウント)

カウント開始から終了までの間に到来するパルスカウントを行います。計数パルスはエンコーダなど「位相差(2相)パルス」、単に「アップ/ダウパルス」、「カウント方向とパルス列」です。これらは信号形式と言います。

【応用】

●時間当たりカウント

- ① PC(パソコン)から時間単位でCTRラッチ。
- ② イベントタイマを利用して周期ごとにCTRラッチ。

●外部の信号によりカウント値をラッチ

- ① XZ, YZ, などZ相信号入力により全CTR同時ラッチ。
- ② 汎用入力 IN1 信号入力により全CTR同時ラッチ。

●最大値-最小値計測

- ① アップ/ダウンカウント計測期間中の最大値及び最小値を計測。
- ② 信号幅, パルス周期, 位相差計測期間中の最大値及び最小値を計測。

(2) コンパレータ

#1CCL (#2CCL)には4組のコンパレータ(以下CMP)があります。これらのCMPを利用して主として次の機能が実現できます。

- ① CMP条件成立によりPCへ割込み。(カウント周期的にPCに通知できる)
- ② CMP条件成立によりCTRクリア。(イコール比較による)
- ③ CMP条件成立により外部へパルス出力(A/Dコンバータサンプリング・トリガーなど)
- ④ 2次元エリアのウィンドウ・コンパレート(例:XYエリアにあるとき,XYOUT1信号 ON)

(3) イベントタイマ

#1CCL (#2CCL)には1組のTIMER(EVENT TIMER)があります。このTIMER機能は次の通りです。

- ① TIMER周期でCTRラッチ,ラッチ後CTRクリア(同一CCLのChのみ),全CTR同時ラッチ。
- ② TIMER周期で一致出力。
- ③ TIMER周期でPCへ割込み(DOS版ソフトのみ対応)

3. ボード構成とポートアドレス

3. 1 ブロック図

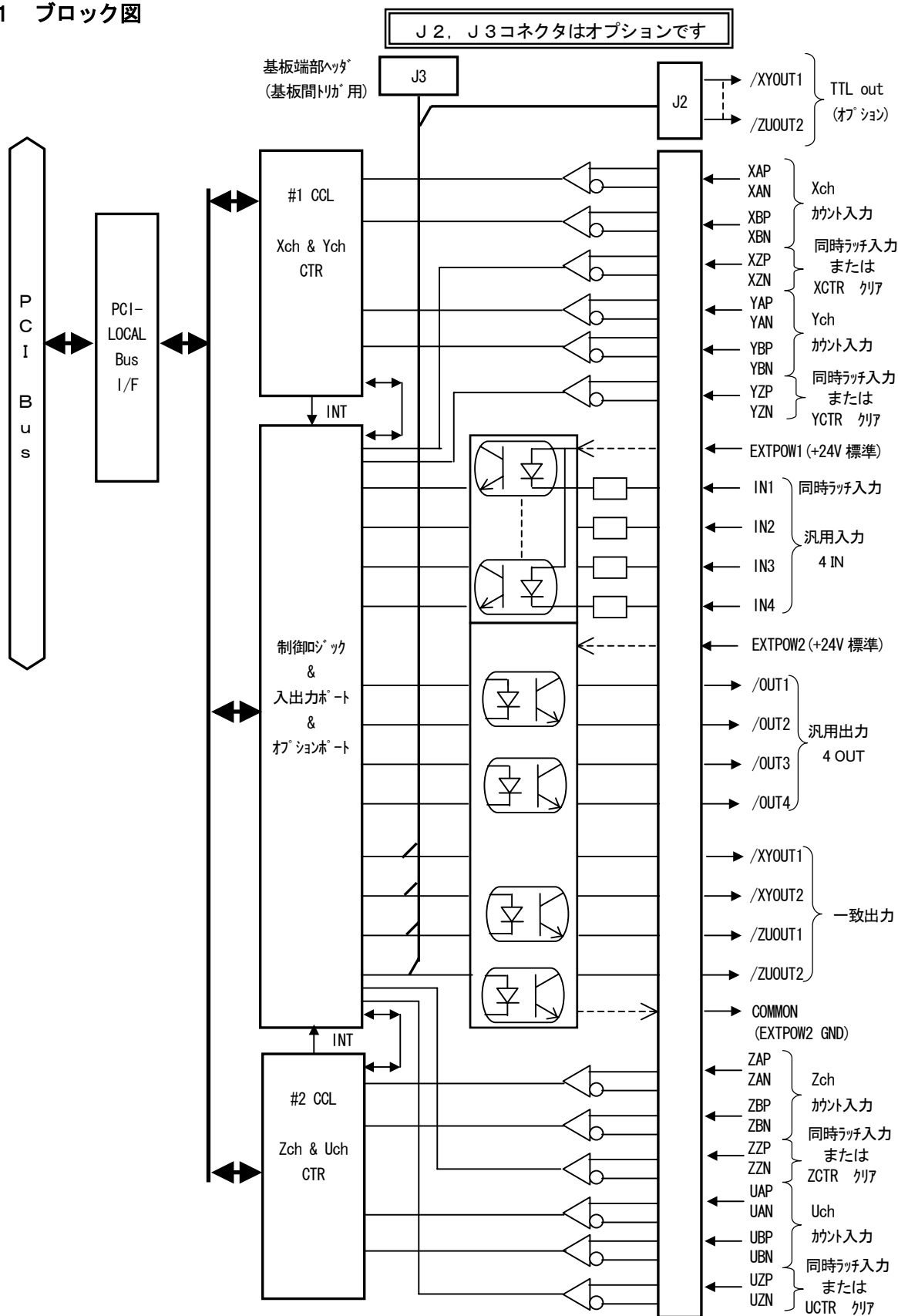


図3. 1-1 HPCI-CTR524Fブロックダイア

3. 2 ポートアドレス

ポートはすべてI/Oマップです。表3-2 にHPCI-CTR524Fポート表を示します。
各Busごとに違いがありますので、詳細はユーザーズマニュアル<個別ボード編>を参照してください。

表3. 2-1 にポート表を示します。(このボードでは、80h(128)バイトを占有しています。)

区分	I/Oアドレス (hex)	読み込み (INP)		書き込み (OUT)		記事				
		呼称	内容	呼称	内容					
XYch (#1CCL)	+0	CMD	ステータス	STS	コマンド					
	+2	-	不使用(予約)	-	不使用(予約)					
	+4	BUF0	入出力バッファ IN (15~ 0)	BUF0	入出力バッファ OUT (15~ 0)					
	+6	BUF1	入出力バッファ IN (31~16)	BUF1	入出力バッファ OUT (31~16)					
ZUch (#2CCL)	+8	CMD	ステータス	STS	コマンド					
	+a	-	不使用(予約)	-	不使用(予約)					
	+c	BUF0	入出力バッファ IN (15~ 0)	BUF0	入出力バッファ OUT (15~ 0)					
	+e	BUF1	入出力バッファ IN (31~16)	BUF1	入出力バッファ OUT (31~16)					
Z相 及び タイマモニタ	+20	ZIN TMR MONTR	Z相入力状態読出し, イベントタイマモタ	-	不使用					
汎用入力	+28	0	IN1	汎用入力1 兼外部同時ラッチ	'1' 入力中	-	不使用			
		1	IN2	汎用入力2						
		2	IN3	汎用入力3						
		3	IN4	汎用入力4						
		4-7	-	不使用						
汎用出力	+2c	0	OUT1M	汎用出力1状態	'1' 出力中 (出力ON)	OUT1	汎用出力1	'1' 出力ON '0' 出力OFF		
		1	OUT2M	汎用出力2状態					OUT2	汎用出力2
		2	OUT3M	汎用出力3状態					OUT3	汎用出力3
		3	OUT4M	汎用出力4状態					OUT4	汎用出力4
		4-7	-	不使用					-	不使用
コンパレータ 出力設定	+30	CMP MSK MONTR	CMPOUT1~4 出力設定 設定状態読出し	CMP MSK	CMPOUT1~4 出力設定					
イベントタイマ 出力設定	+32	TMR MSK MONTR	ETMR 出力設定状態読出し	TMR MSK	ETMR 出力設定					
一致出力 設定	+34	OUT SEL MONTR	一致出力設定状態読出し	OUT SEL	一致出力設定 (CMP出力OUT1~4, ETMR出力を XYOUT, ZUOUT 選択と出力幅選択)					
Z相カウンタ 条件設定	+36	CTRCL SEL MONTR	Z相入力でのカウンタクリア 条件設定状態読出し	CTRCL SEL	Z相入力での カウンタクリア 条件設定					
同時ラッチ 条件設定	+38	LTCH SEL MONTR	同時ラッチ条件 設定状態読出し	LTCH SEL	同時ラッチ 条件設定					
割込みマスク	+50	INT STS	#1CCL, #2CCL, IN1 外部入力 での割込み状態読出し	INT MSK	#1CCL, #2CCL, IN1 外部入力 での割込みマスク設定					
	+52	PCI INT STS	PCIバスへの割込みマスク 設定状態, 割込み状態読出し	PCI INT MSK	PCIバスへの割込みマスク設定					
ボードID	+60	BRDID	bit0-3: 0~15 bit4-7: 不使用	-	不使用					

表3. 2-1 HPCI-CTR524Fポート表

3. 3 ポートとレジスタ配置

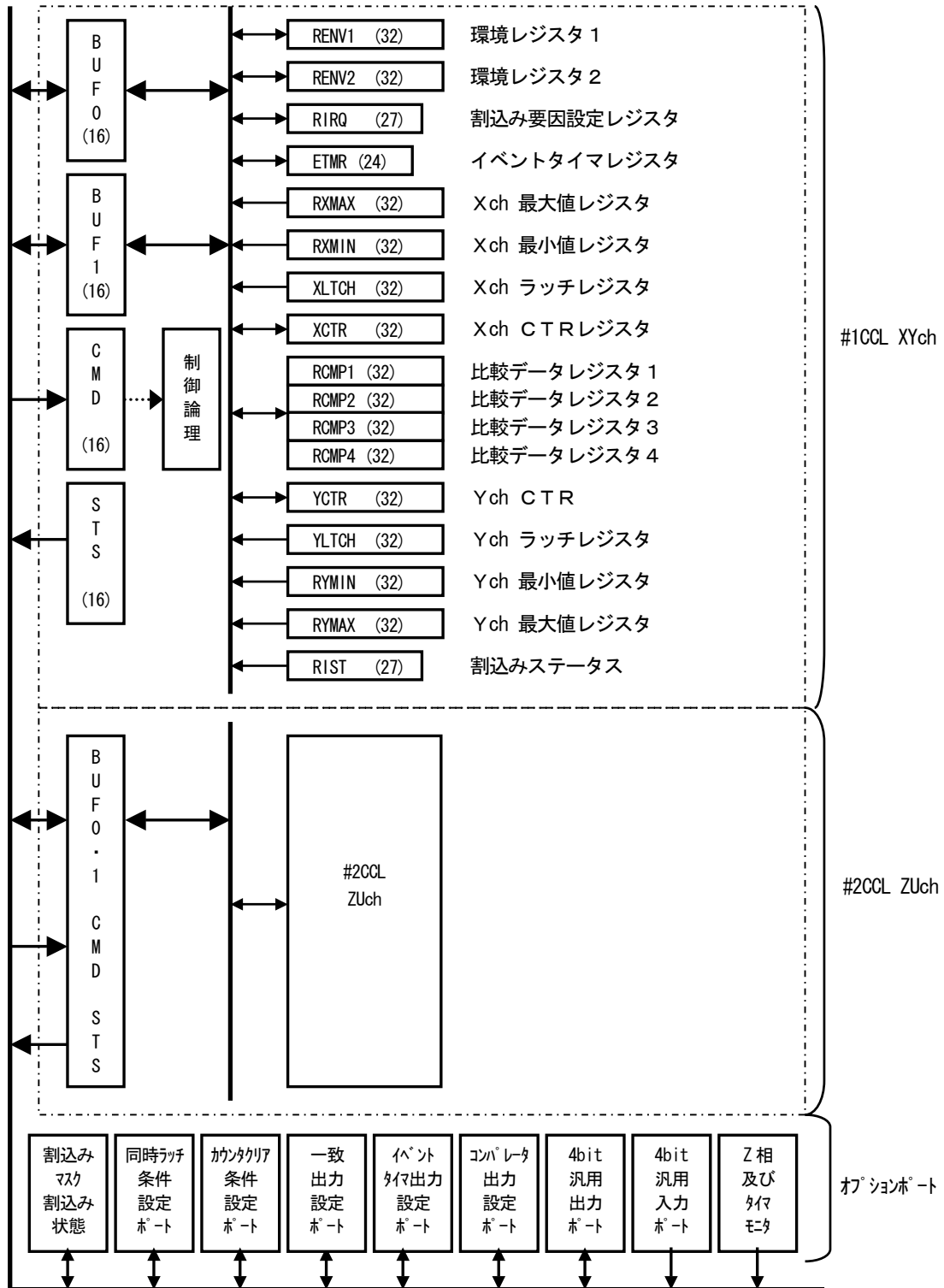


図3. 3-1 HPCI-CTR524Fポートとレジスタ配置

4. コマンド・ステータス・レジスタ

4. 1 ポートおよびレジスタの書込み, 読出し

「表3. 2-1 HPCI-CTR524Fポート表」に示すように#xCCLにCMD, BUF0, BUF1, STSポートがあります。

全体に共通に1組のオプションポート(割込みおよび外部出力設定, 4bit汎用ポート)があります。

CCLに対するコマンド, データは, CMD, BUF xポート経由で指定するレジスタ, カウンタに書くことによりセットします。

データを伴わないような測定開始, CTR LTCH, CTR CLRなどのコマンドのみの場合はCMDポートに書くだけで行います。

レジスタを読み出す場合はCMDポートに読出しコマンドを書きます。その結果, BUF0, BUF1に読み出されます。

4. 2 レジスタ制御コマンド

次表にレジスタ制御コマンド一覧を掲げます。

項	レジスタ名	内 容	書込みコマンド	読出しコマンド	ビット長	データ範囲	参照頁
1	CTR1 (XCTR, ZCTR)	カウンタ 1	0080h	00c0h	32	-2,147,483,648 2,147,483,647	9
2	CTR2 (YCTR, UCTR)	カウンタ 2	0081h	00c1h	32		9
3	RCMP1	比較データレジスタ 1	0082h	00c2h	32		9
4	RCMP2	比較データレジスタ 2	0083h	00c3h	32		9
5	RCMP3	比較データレジスタ 3	0084h	00c4h	32		9
6	RCMP4	比較データレジスタ 4	0085h	00c5h	32		9
7	ETMR	イベントタイマレジスタ	0086h	00c6h	24	2~16,777,215	9
8	RENV1	環境レジスタ 1 (入力仕様設定)	0087h	00c7h	32	logical data	10
9	RENV2	環境レジスタ 2 (コンパレータ条件設定)	0088h	00c8h	30	logical data	12
10	RIRQ	割込み要因設定レジスタ	0089h	00c9h	27	logical data	8
11	LTCH1 (XLTCH, ZLTCH)	カウンタ 1 のラッチレジスタ	/	00cah	32	-2,147,483,648 2,147,483,647	9
12	LTCH2 (YLTCH, ULTCH)	カウンタ 2 のラッチレジスタ		00cbh	32		9
13	MAX1 (XMAX, ZMAX)	カウンタ 1 の最大値レジスタ		00cch	32		9
14	MIN1 (XMIN, ZMIN)	カウンタ 1 の最小値レジスタ		00cdh	32		9
15	MAX2 (YMAX, UMAX)	カウンタ 2 の最大値レジスタ		00ceh	32		9
16	MIN2 (YMIN, UMIN)	カウンタ 2 の最小値レジスタ		00cfh	32		9
17	RIST	割込みステータス・レジスタ		00d0h	27	logical data	7

表4. 2-1 レジスタ制御コマンド

4. 3 単独コマンド

次表に単独コマンド一覧を掲げます。これらのコマンドはデータを伴わず、単独にCMDポートに書込み、動作します。

項	コマンド名	内 容	CMDコード	備 考
1	CLR CTR1	CTR1 を 0 クリア	0001h	
2	CLR CTR2	CTR2 を 0 クリア	0002h	
3	INIT MM1	CTR1 の MAX1, MIN2 を初期化	0004h	
4	INIT MM2	CTR1 の MAX2, MIN2 を初期化	0008h	
5	LTCH CTR1	CTR1 を LTCH1 にラッチ読出し	0010h	カント動作中のCTR読出しは, LTCHコマンドを出し, ラッチレジスタ (LTCH) から読む。
6	LTCH CTR2	CTR2 を LTCH2 にラッチ読出し	0020h	
7	SRST	ソフトリセット	0040h	全レジスタ, BUF0, 1 は 0 クリアされる。
8	SLTCH	全 ch 同時ラッチ	0041h	
9	MM1 START	CTR1 の 最大値最小値の測定開始	0048h	MMSTART で CTR, MAX, MIN は 0 クリアされる。 注意: CTR 1, CTR 2 の通常のカント レベルは RENV 1 の各々 b12, b13 で行う。
10	MM1 STOP	CTR1 の 最大値最小値の測定終了	0049h	
11	MM2 START	CTR2 の 最大値最小値の測定開始	004ah	
12	MM2 STOP	CTR2 の 最大値最小値の測定終了	004bh	

注: カウンタスタートは RENV 1 b 1 2, b 1 3 で行います。

表4. 3-1 単独コマンド

4. 4 ステータス

ステータスは2種類あり、直接ポートから読出す「STS」とレジスタとして読出す割り込みステータス「RIST」があります。

4. 4. 1 ステータス (STS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTS	*	*	*	*	*	*	*	CTR2MM	CTR1MM	CTR2S	CTR1S	CMP4	CMP3	CMP2	CMP1

bit	名称	説明	備考
0	CMP1	1:CMPOUT1 出力中	これらは（イコール比較など）カウンタが動作中の場合は正しい状態を反映しません。このような場合は INTS(bit15)を使用します。
1	CMP2	1:CMPOUT2 出力中	
2	CMP3	1:CMPOUT3 出力中	
3	CMP4	1:CMPOUT4 出力中	
4	CTR1S	1:CTR1 停止状態, 0:CTR1 動作状態	
5	CTR2S	1:CTR1 停止状態, 0:CTR2 動作状態	
6	CTR1MM	1:CTR1 MM 測定状態, 0:CTR1 MM 測定停止状態	
7	CTR2MM	1:CTR2 MM 測定状態, 0:CTR2 MM 測定停止状態	
8	*		
~	~	DON'T CARE (通常保守用途)	
14	*		
15	INTS	1:割り込み要求中: 割り込み内容は「4.4.2 割り込みステータス」による	RIST 読出し後、ビットはクリアされる。

表 4. 4-1 ステータス内容

4. 4. 2 割り込みステータスレジスタ (RIST)

RISTの各ビットは、RIST読出し後リセットされます。さらに、STS b15 (INTS) がリセットされます。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
← 不使用 →	LTCH2I	LTCH1I	CLR2I	CLR1I	CMP4I1	CMP4I0	CMP3I1	CMP3I0	CMP2I1	CMP2I0	CMP1I1	CMP1I0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	ETMRINT	MENDI	SLTCHI	C2OVFN	C2OVFP	C1OVFN	C1OVFP	ENC2ERI	ENC1ERI	CTR2=0	CTR1=0

bit	名称	説明	備考
0	CMP1I0	1: CMP 1が条件成立し, 割込	
1	CMP1I1	1: CMP 1の条件成立が解けて, 割込	
2	CMP2I0	1: CMP 2が条件成立し, 割込	
3	CMP2I1	1: CMP 2の条件成立が解けて, 割込	
4	CMP3I0	1: CMP 3が条件成立し, 割込	
5	CMP3I1	1: CMP 3の条件成立が解けて, 割込	
6	CMP4I0	1: CMP 4が条件成立し, 割込	
7	CMP4I1	1: CMP 4の条件成立が解けて, 割込	
8	CLR1I	1: CTR 1にZ相入力によるクリアがかかり, 割込	Z相端子からのCTRクリア
9	CLR2I	1: CTR 2にZ相入力によるクリアがかかり, 割込	Z相端子からのCTRクリア
10	LTCH1I	1: CTR 1XZ (ZZ) 信号入力によるラッチがかかり, 割込	Z相個別ラッチを 選択した場合
11	LTCH2I	1: CTR 2YZ (UZ) 信号入力によるラッチがかかり, 割込	XZ (ZZ) 個別ラッチ YZ (UZ) 個別ラッチ
12			
~	不使用		
15			
16	CTR1=0	1: CTR1 が0になり, 割込	
17	CTR2=0	1: CTR2 が0になり, 割込	
18	ENC1ERI	1: Xch (Zch) の+/-カウント信号が同時に変化したエラー割込	
19	ENC2ERI	1: Ych (Uch) の+/-カウント信号が同時に変化したエラー割込	
20	C1 OVFP	1: CTR1 の+側のオーバーフロー発生, 割込	
21	C1 OVFN	1: CTR1 の-側のオーバーフロー発生, 割込	
22	C2 OVFP	1: CTR2 の+側のオーバーフロー発生, 割込	
23	C2 OVFN	1: CTR2 の-側のオーバーフロー発生, 割込	
24	SLTCHI	1: 全 ch 同時ラッチ信号入力での割込	外部信号による全 ch 同時ラッチ, 同時ラッチ CMD (41h)
25	MENDI	1: 幅計測終了エッジでの割込	
26	ETMRINT	1: イベントタイマ周期割り込み	※タイマスタート時にも割り込み
27~31		不使用 a l l '0'	

表 4. 4-2 割り込みステータスの内容

4. 4. 3 割込み要因設定レジスタ (RIRQ)

RIRQの割込みイネーブルに設定したビットが割込み対象となります。電源投入直後 RIRQは全て割込みディセーブル状態です。単独コマンドの「ソフトウェアリセット」もRIRQを全て '0' にします。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	LTCH2M	LTCH1M	CLR2M	CLR1M	CMP4M1	CMP4M0	CMP3M1	CMP3M0	CMP2M1	CMP2M0	CMP1M1	CMP1M0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	ETMRM	MENDM	SLTCHM	0	C2OVFM	0	C1OVFM	ENC2M	ENC1M	C2ZRO	C1ZRO

bit	名称	説明	備考
0	CMP1 MO	1: CMP 1 が条件成立	割込みイネーブル.
1	CMP1 M1	1: CMP 1 の条件成立が解けて	割込みイネーブル
2	CMP2 MO	1: CMP 2 が条件成立	割込みイネーブル.
3	CMP2 M1	1: CMP 2 の条件成立が解けて	割込みイネーブル.
4	CMP3 MO	1: CMP 3 が条件成立	割込みイネーブル.
5	CMP3 M1	1: CMP 3 の条件成立が解けて	割込みイネーブル.
6	CMP4 MO	1: CMP 4 が条件成立	割込みイネーブル.
7	CMP4 M1	1: CMP 4 の条件成立が解けて	割込みイネーブル.
8	CLR1M	1: CTR 1 に Z 相入力によるクリア	割込みイネーブル.
9	CLR2M	1: CTR 2 に Z 相入力によるクリア	割込みイネーブル.
10	LTCH1M	1: CTR 1 XZ (ZZ) 信号入力による	割込みイネーブル
11	LTCH2M	1: CTR 2 YZ (UZ) 信号入力による	割込みイネーブル.
12	不使用	0:	
13		0:	
14		0:	
15		0:	
16	C1ZRO	1: CTR1 の内容が 0 になった時,	割込みイネーブル
17	C2ZRO	1: CTR2 の内容が 0 になった時,	割込みイネーブル.
18	ENC1M	1: Xch (Zch) の +/- カウント信号が同時に変化	エラー割込みイネーブル
19	ENC2M	1: Ych (Uch) の +/- カウント信号が同時に変化	エラー割込みイネーブル
20	C1 OVFM	1: CTR1 のオーバーフロー発生,	割込みイネーブル
21	不使用	0:	
22	C2 OVFM	1: CTR2 のオーバーフロー発生,	割込みイネーブル
23	不使用	0:	
24	SLTCHM	1: 全 ch 同時ラッチが発生,	割込みイネーブル
25	MENDM	1: 幅計測終了エッジで割込みイネーブル	
26	ETMRM	1: イベントタイマ周期割込みイネーブル	
27~31		不使用 all '0'	

表 4. 4-3 割込み要因設定レジスタの内容

4. 5 レジスタ

4. 5. 1 カウンタ (CTR1, CTR2)

#1CCL・・・XCTR: CTR1, YCTR: CTR2

#2CCL・・・ZCTR: CTR1, UCTR: CTR2

31	-----	24	23	-----	16	15	-----	8	7	-----	0
-2, 147, 483, 648 ~ 2, 147, 483, 647											

4. 5. 2 比較データレジスタ (RCMP1~RCMP4)

#1CCL, #2CCL共に4種類の比較データレジスタ (RCMP1~RCMP4) があります。

31	-----	24	23	-----	16	15	-----	8	7	-----	0
-2, 147, 483, 648 ~ 2, 147, 483, 647											

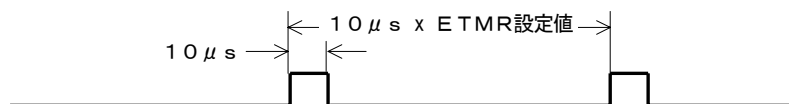
4. 5. 3 イベントタイマレジスタ (ETMR)

#1CCL, #2CCL共に1個のイベントタイマ・レジスタがあります。

イベントタイマの周期をこのレジスタで設定します。

設定単位は10μsであり, 設定値100で1msとなります。信号幅は10μsです。

31	-----	24	23	-----	16	15	-----	8	7	-----	0
0			2 ~ 16, 777, 215 (20μs ~ 167.77秒)								



4. 5. 4 ラッチレジスタ (LTCH1, LTCH2)

#1CCL, #2CCL共に1組のラッチレジスタがあり, 各カウンタと一対となります。

#1CCL・・・XCTR: XLTCH(LTCH1), YCTR: YLTCH(LTCH2)

#2CCL・・・ZCTR: ZLTCH(LTCH1), UCTR: ULTCH(LTCH2)

31	-----	24	23	-----	16	15	-----	8	7	-----	0
-2, 147, 483, 648 ~ 2, 147, 483, 647											

4. 5. 5 最大値レジスタ (MAX1, 2), 最小値レジスタ (MIN1, 2)

#1CCL, #2CCL共に2組の最大値・最小値レジスタがあり, 各カウンタと一対となります。

#1CCL・・・XCTR: RXMAX(MAX1), RXMIN(MIN1)

YCTR: RYMAX(MAX2), RYMIN(MIN2)

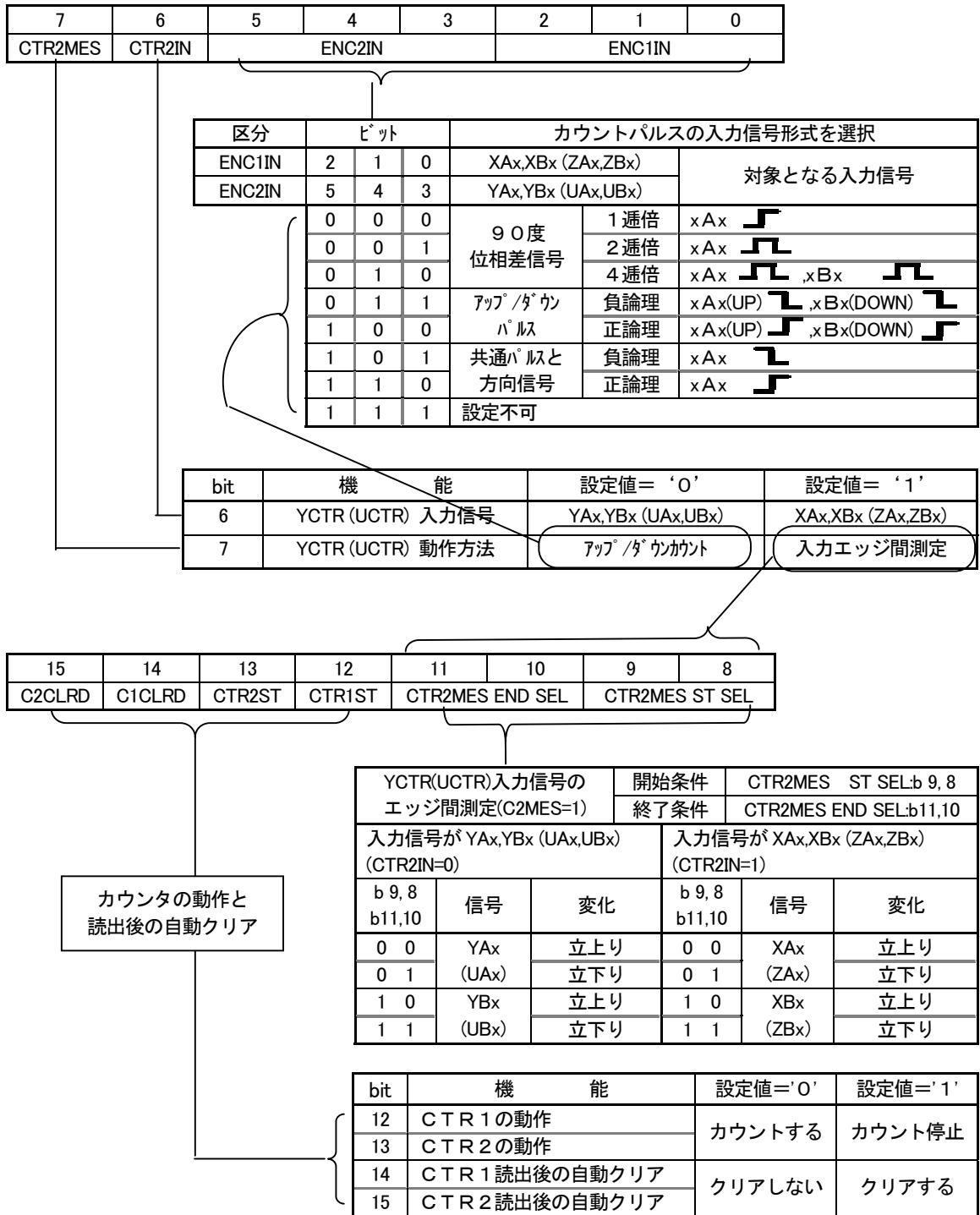
#2CCL・・・ZCTR: RZMAX(MAX1), RZMIN(MIN1)

UCTR: RUMAX(MAX2), RUMIN(MIN2)

31	-----	24	23	-----	16	15	-----	8	7	-----	0
-2, 147, 483, 648 ~ 2, 147, 483, 647											

4. 5. 6 環境レジスタ1 (RENV1)

#1 CCL, #2 CCL共に1組の環境レジスタがあり, 環境レジスタ1ではカウンタの動作条件を設定し, カウンタの起動・停止を行います.



REN V 1

23	22	21	20	19	18	17	16
1	1	0	ZLEDG	C2CLIN	C1CLIN	CLR-COND	

bit	機 能	設定値='0'	設定値='1'
18	XZ(ZZ)信号入力 カウンタクリア	有効	無効
19	YZ(UZ)信号入力 カウンタクリア		
上記2ビットはオプションポートと関連 不使用時は“無効(1)”とし、使用時は下記手順 ①REN V1:無効, ②オプションポート設定, ③REN V1:有効			
20	Z相(XZ, YZ, ZZ, UZ)信号入力 によるラッチのエッジ選択	設定値='0'	設定値='1'
	オプション ポート選択	同時ラッチ 個別ラッチ	任意(Don't Care) 立上がり 立下がり

XZ,YZ(ZZ,UZ)の7条件		
bit		内 容
17	16	
0	0	立上りエッジ
0	1	立下りエッジ
1	0	High レベル
1	1	Low レベル

(00:立上りエッジを推奨)

REN V 1

31	30	29	28	27	26	25	24
0	EVENT-ON CYCLE TIMER			CMP4·C	CMP3·C	CMP2·C	CMP1·C

比較カウンタ選択				
bit	機 能	設定値='0'	設定値='1'	
24	CMP 1	比較 カウンタ 選択	XCTR (ZCTR)	YCTR (UCTR)
25	CMP 2			
26	CMP 3			
27	CMP 4			

イベントタイマ出力時の処理									
bit			処理内容	bit			処理内容		
30	29	28		30	29	28			
0	0	1	CTR 1	ラッチ	0	1	0	CTR 1	ラッチ & クリア
0	1	1	CTR 2		1	0	0	CTR 2	
1	0	1	CTR1,CTR2		1	1	0	CTR1,CTR2	
0	0	0	処理なし		1	1	1		

表4. 5-1 環境レジスタ1 (REN V 1)

4. 5. 7 環境レジスタ 2 (RENV2)

#1 CCL, #2 CCL 共に 1 組の環境レジスタがあり、環境レジスタ 2 では各種コンパレータの比較条件の設定と、比較結果出力の設定等を行います。

7	6	5	4	3	2	1	0
CMPOUT2 COND SEL				CMPOUT1 COND SEL			

区分	bit				※CMP 1 と CMP 2 に設定する値は共通 (下表)	
CMP 1	3	2	1	0	CMP1 の比較方法	
CMP 2	7	6	5	4	CMP2 の比較方法	
	0	0	0	0	RCMP1 < [CTR1/CTR2]	
	1	0	0	0	RCMP1 > [CTR1/CTR2]	
	2	0	0	1	RCMP1 = [CTR1/CTR2]	カウント方向に無関係
	3	0	0	1		カウントアップ時
	4	0	1	0		カウントダウン時
	5	0	1	0	RCMP2 < [CTR1/CTR2]	
	6	0	1	1	RCMP2 > [CTR1/CTR2]	
	7	0	1	1	RCMP2 = [CTR1/CTR2]	カウント方向に無関係
	8	1	0	0		カウントアップ時
	9	1	0	0		カウントダウン時
	10	1	0	1	RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2	
	11	1	0	1	RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2	
	12	1	1	0	(RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2) AND (RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4)	
	13	1	1	0	(RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2) OR (RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4)	
	14	1	1	1	(RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2) AND (RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4)	
	15	1	1	1	(RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2) OR (RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4)	
※[CTR1/CTR2] の表記は RENV1 CMP1・C(bit24)~CMP4・C(bit27) で決定されるカウンタの意味						

15	14	13	12	11	10	9	8
CMPOUT4 COND SEL				CMPOUT3 COND SEL			

区分	bit				※CMP 3 と CMP 4 に設定する値は共通 (下表)	
CMP 3	11	10	9	8	CMP3 の比較方法	
CMP 4	15	14	13	12	CMP4 の比較方法	
	0	0	0	0	RCMP3 < [CTR1/CTR2]	
	1	0	0	0	RCMP3 > [CTR1/CTR2]	
	2	0	0	1	RCMP3 = [CTR1/CTR2]	カウント方向に無関係
	3	0	0	1		カウントアップ時
	4	0	1	0		カウントダウン時
	5	0	1	0	RCMP4 < [CTR1/CTR2]	
	6	0	1	1	RCMP4 > [CTR1/CTR2]	
	7	0	1	1	RCMP4 = [CTR1/CTR2]	カウント方向に無関係
	8	1	0	0		カウントアップ時
	9	1	0	0		カウントダウン時
	10	1	0	1	RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4	
	11	1	0	1	RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4	
	12	1	1	0	CTR2 < CTR1	
	13	1	1	0	CTR2 > CTR1	
	14	1	1	1	CTR2 = CTR1	
	15	1	1	1	常に比較条件不成立	
※[CTR1/CTR2] の表記は RENV1 CMP1・C(bit24)~CMP4・C(bit27) で決定されるカウンタの意味						

REN V 2

23	22	21	20	19	18	17	16
CMP3,4-P	CMP1,2 P	CMP2 EQ	CMP1 EQ	CMPOUT3,4-WIDTH		CMPOUT1,2-WIDTH	

bit	機 能	設定値=0	設定値=1
20	CMPOUT1	"="比較条件成立時	比較カウンタのクリア
21	CMPOUT2		
22	CMPOUT1,CMPOUT2	出力論理	正論理
23	CMPOUT3,CMPOUT4		

b17,16	CMPOUT1,2	パルス幅
b19,18	CMPOUT3,4	
0 0	レベル出力	
0 1	50 μs	ワンショット出力
1 0	1ms	
1 1	6.25ms	
※オプションポートも参照		

REN V 2

31	30	29	28	27	26	25	24
0	0	FILTR ON	CMP1,2 MSK	0	0	0	0

bit	機 能	設定値='0'	設定値='1'
28	CMPOUT1,2 出力マスク	マスクしない	マスクする
29	xAx,xBx 入力信号フィルタ	フィルタON	フィルタOFF

表 4. 5-2 環境レジスタ 2 (REN V 2)

5. 基本的な設定と運用

5. 1 操作手順

操作手順は以下の流れで行います。

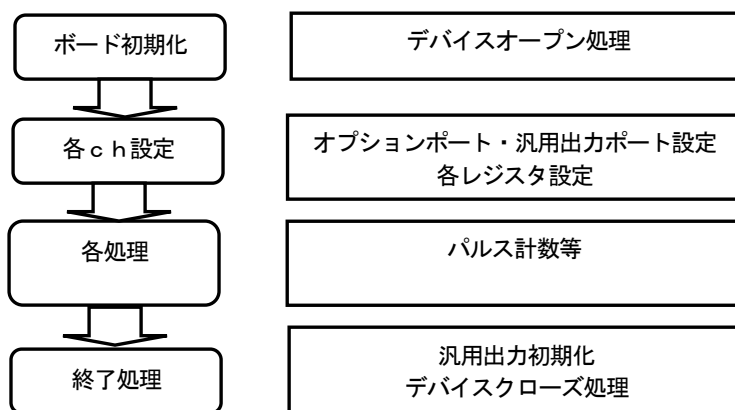


表5. 1-1 基本的な操作手順

5. 2 エンコーダ等パルス計数 (カウント)

以下に基本的なカウントの手順を記します。

- ①使用するカウンタのカウントを停止
環境レジスタ 1 (RENV1) の bit12=1 (CTR1), bit13=1 (CTR2) 設定.
- ②使用するカウンタのカウントが停止していることを確認
ステータスの bit4=1 (CTR1 停止中), bit5=1 (CTR2 停止中) を確認.
- ③使用するカウンタの入力信号形式を選択
環境レジスタ 1 (RENV1) の bit0~bit2 (CTR1), bit3~bit5 (CTR2) で設定.
- ④使用するカウンタをクリア
単独コマンド CLRCTR1 (0001h), CLRCTR2 (0002h) 書込みで, 使用するカウンタを 0 にする.
あらかじめオフセットするならば, 使用する CTR レジスタにオフセット値を書込む.
- ⑤使用するカウンタをスタート
環境レジスタ 1 (RENV1) の bit12=0 (CTR1), bit13=0 (CTR2) 設定.
- ⑥使用するカウンタレジスタ読み

5. 2. 1 カウントパルスの入力信号形式

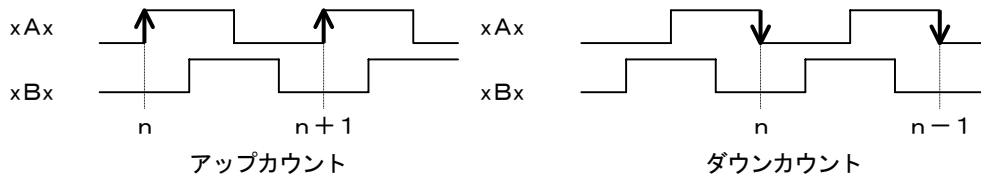
環境レジスタ 1 (RENV1) の bit 0~5 を設定します。

区分	ビット			カウントパルスの入力信号形式を選択	
ENC1IN	2	1	0	XAx, XBx (ZAx, ZBx)	
ENC2IN	5	4	3	YAx, YBx (UAx, UBx)	
	0	0	0	90度 位相差信号	1 通倍
	0	0	1		2 通倍
	0	1	0		4 通倍
	0	1	1	アップ/ダウン パルス	負論理
	1	0	0		正論理
	1	0	1	共通パルス と方向信号	負論理
	1	1	0		正論理
	1	1	1	設定不可	

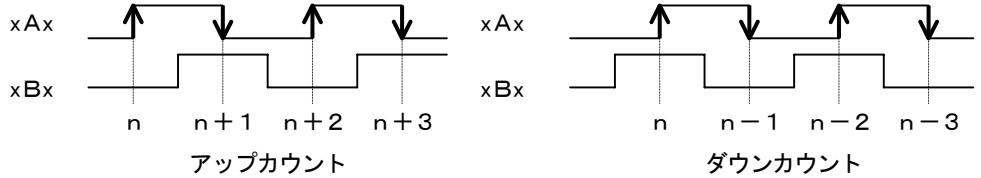
表5. 2-1 カウントパルスの入力信号形式

各入力信号形式におけるカウント動作時のタイミングは、次頁の通りです。

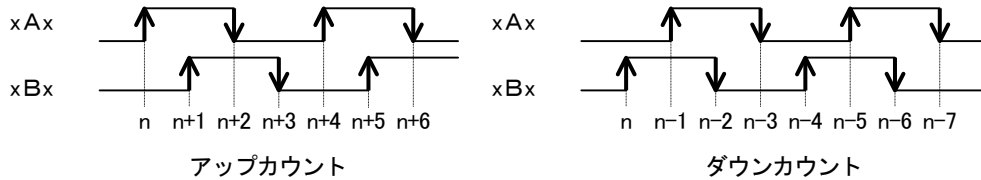
■ 90度位相差信号1通倍



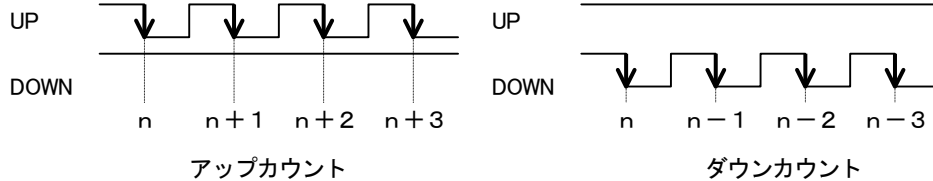
■ 90度位相差信号2通倍



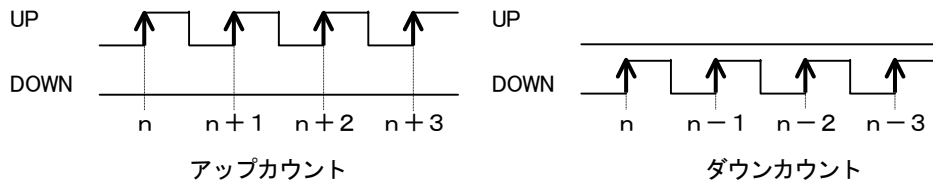
■ 90度位相差信号4通倍



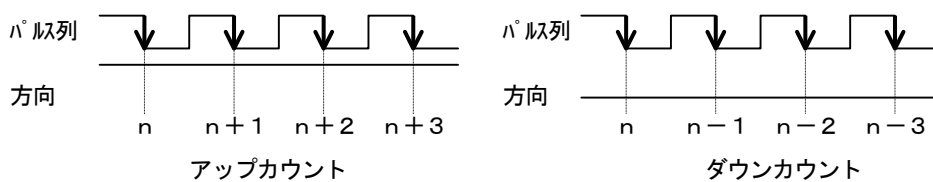
■ アップ/ダウンパルス負論理



■ アップ/ダウンパルス正論理



■ 共通パルスと方向信号負論理



■ 共通パルスと方向信号正論理

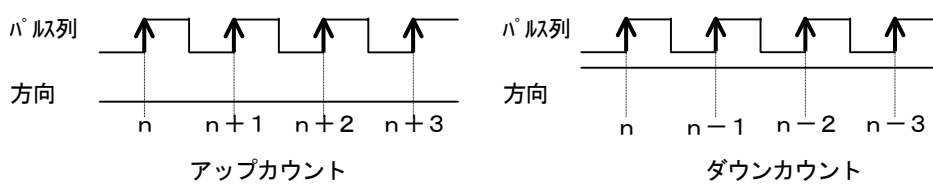


表5. 2-2 各入力信号形式におけるカウント動作のタイミング

5. 2. 2 カウントスタート/ストップ

環境レジスタ 1 (RENV1) の bit12 の設定でカウンタ 1 (CTR1), bit13 の設定でカウンタ 2 (CTR2) をカウントスタート/ストップします。

(1) カウントスタート

カウンタ 1 は環境レジスタ 1 (RENV1) の bit12=0 でカウントスタートします。
カウンタ 2 は環境レジスタ 1 (RENV1) の bit13=0 でカウントスタートします。

(2) カウントストップ

カウンタ 1 は環境レジスタ 1 (RENV1) の bit12=1 でカウントストップします。
カウンタ 2 は環境レジスタ 1 (RENV1) の bit13=1 でカウントストップします。

5. 2. 3 カウンタクリア機能

カウンタ 1, およびカウンタ 2 は, 以下の方法でカウント値を 0 クリアできます。

(1) コマンドによるカウンタクリア

単独コマンド書込みでカウント値を 0 クリアします。

カウンタ 1 は単独コマンド CLR_CTR1 (0001h) 書込みで 0 クリアされます。
カウンタ 2 は単独コマンド CLR_CTR2 (0002h) 書込みで 0 クリアされます。
尚, クリアビットは記憶されませんので, 解除コマンドは不要です。

(2) カウンタデータ読出し後のカウンタクリア

カウンタ 1, およびカウンタ 2 は環境レジスタ 1 (RENV1) の設定により, カウンタ値を読出した後に自動的にカウンタ値をクリアできます。

環境レジスタ 1 (RENV1) の bit14=1 の時, カウンタ 1 読出し後カウンタ 1 が 0 クリアされます。
環境レジスタ 1 (RENV1) の bit15=1 の時, カウンタ 2 読出し後カウンタ 2 が 0 クリアされます。

(3) コンパレータ条件一致でカウンタクリア

カウンタ 1, およびカウンタ 2 は環境レジスタ 2 (RENV2) の設定により, CMP1, または CMP2 の比較条件が成立した時, 自動的にカウンタ値をクリアできます。但し比較条件は“比較データ=比較カウンタ”のみです。

環境レジスタ 2 (RENV2) の bit28=0 かつ,
環境レジスタ 2 (RENV2) の bit20=1 の時, CMP 1 条件一致で比較カウンタが 0 クリアされます。
環境レジスタ 2 (RENV2) の bit21=1 の時, CMP 2 条件一致で比較カウンタが 0 クリアされます。

(4) イベントタイマでカウンタクリア

カウンタ 1, およびカウンタ 2 は環境レジスタ 1 (RENV1) の設定により, イベントタイマ信号出力で自動的にカウンタ値をクリアできます。

環境レジスタ 1 (RENV1) の bit30~28 の設定

- 000 : 処理なし
- 001 : カウンタ 1 の値をラッチ
- 010 : カウンタ 1 の値をラッチ後クリア
- 011 : カウンタ 2 の値をラッチ
- 100 : カウンタ 2 の値をラッチ後クリア
- 101 : カウンタ 1, 2 の値をラッチ
- 110 : カウンタ 1, 2 の値をラッチ後クリア

- (5) Z相信号入力によるカウンタクリア
各c hのZ相信号入力により各c hカウンタクリアされます。
環境レジスタ (RENV1) 及び、
HPCI-CTR524F, CTR522Fはカウンタクリア設定ポート、
HPC-CTR224F, CTR222F, HPC104-CTR122Fは同時ラッチ設定ポート
を設定します。

XZ (ZZ) 入力でカウンタ 1 をクリアする場合は次の手順で設定します。

- ①環境レジスタ (RENV1) の bit18=1 (カウンタクリア無効)
- ②オプションポート設定
- ③環境レジスタ (RENV1) の bit18=0 (カウンタクリア有効)

YZ (UZ) 入力でカウンタ 2 をクリアする場合は次の手順で設定します。

- ①環境レジスタ (RENV1) の bit19=1 (カウンタクリア無効)
- ②オプションポート設定
- ③環境レジスタ (RENV1) の bit19=0 (カウンタクリア有効)

オプションポートの設定は各々、ユーザーズマニュアル<個別ボード編> オプションポートの設定を参照してください。

5. 2. 4 カウンタラッチ機能

カウンタ 1, およびカウンタ 2 は、以下の方法でカウント値をラッチ (カウント値をラッチレジスタにコピー) できます。

- (1) コマンドによるカウンタラッチ

単独コマンド書込みでカウント値をラッチします。

カウンタ 1 は単独コマンド LTCH CTR1 (0010h) 書込みでカウンタ値がラッチされます。

カウンタ 2 は単独コマンド LTCH CTR2 (0020h) 書込みでカウンタ値がラッチされます。

尚、ラッチビットは記憶されませんので、解除コマンドは不要です。

- (2) イベントタイマでラッチ

カウンタ 1, およびカウンタ 2 は環境レジスタ 1 (RENV1) の設定により、イベントタイマ信号出力で自動的にカウンタ値をラッチできます。

環境レジスタ 1 (RENV1) の bit30~28 の設定

- 000 : 処理なし
- 001 : カウンタ 1 の値をラッチ
- 010 : カウンタ 1 の値をラッチ後クリア
- 011 : カウンタ 2 の値をラッチ
- 100 : カウンタ 2 の値をラッチ後クリア
- 101 : カウンタ 1, 2 の値をラッチ
- 110 : カウンタ 1, 2 の値をラッチ後クリア

- (3) 同時ラッチ信号入力によるカウンタラッチ

同時ラッチ信号入力で全C h同時ラッチされます。「5. 3 同時ラッチ」で説明します。

このラッチ入力信号がZ相である場合には、個別ラッチの方法もあります。

5. 3 同時ラッチ

ボード内の全c hを、以下の方法で同時にラッチできます。

- (1) 各c hのZ相信号入力 (XZ, YZ, ZZ, UZ信号) による同時ラッチ

同時ラッチ設定ポートを設定します。

オプションポートの設定 (個別編) により、対応するc h毎の個別ラッチが可能となります。

- (2) 汎用入力 I N 1 による同時ラッチ
同時ラッチ設定ポートを設定します。
- (3) 一致出力 (XY (ZU) OUT1 出力) による同時ラッチ
一致出力設定と同時ラッチ設定ポートを設定します。
一致出力設定は「5. 6 一致出力設定」を参照してください。
- (4) イベントタイマ出力 (TMR0UT) による同時ラッチ
一致出力設定と同時ラッチ設定ポートを設定します。
一致出力設定は「5. 6 一致出力設定」を参照してください。
- (5) コマンド書込み
単独コマンド SLTCH(0041h) 書込みで同時ラッチされます。
コマンド書込みの CCL は #1CCL, #2CCL のどちらでも構いません。

尚、Z相信号入力は差動レシーバ、I N 1 はカプラで構成されています。
同時ラッチ設定ポートの設定は各々、ユーザーズマニュアル<個別ボード編> オプションポートの設定を参照してください。

5. 4 コンパレータ

5. 4. 1 コンパレータの機能

#1 CCL (#2 CCL) には4組のコンパレータ (以下CMP) があります。
これらのCMPを利用して主として次の機能が実現できます。

- ① CMP条件成立によりP Cへ割込み。(カウント周期的にP Cに通知できる)
- ② CMP条件成立によりC T Rクリア。(イコール比較による)
- ③ CMP条件成立により外部へパルス出力(A/Dコンバータサンプリング・トリガーなど)
- ④ 2次元エリアのウィンド・コンパレート (例: X Yエリアにあるとき, X Y O U T 1 信号 ON)

5. 4. 2 コンパレータ比較結果の確認

CMP比較結果はステータス、一致出力で確認できます。

■ステータス (S T S)

CMPOUT1 出力: bit0=1 で出力中, bit0=0 で出力なし
CMPOUT2 出力: bit1=1 で出力中, bit1=0 で出力なし
CMPOUT3 出力: bit2=1 で出力中, bit2=0 で出力なし
CMPOUT4 出力: bit3=1 で出力中, bit3=0 で出力なし

これらは (イコール比較など) カウンタが動作中の場合は正しい状態を反映しません。
このような場合はINTSを使用します。

「4. 4. 2 割込みステータスレジスタ(R I S T)」, 「4. 4. 3 割込み要因設定レジスタ(R I R Q)」参照

5. 4. 3 コンパレータ比較条件成立時の処理

比較条件成立時には、以下のことができます。

- (1) 一致出力
一致出力設定は「5. 6 一致出力設定」で説明します。
- (2) 同時ラッチ
一致出力設定と同時ラッチ設定ポートを設定します。
一致出力設定は「5. 6 一致出力設定」を参照してください。
同時ラッチ設定ポートの設定は各々、ユーザーズマニュアル<個別ボード編> オプションポートの設定を参照してください。

(3) 比較カウンタのクリア (比較データ=比較カウンタのみ)

カウンタ 1, およびカウンタ 2 は環境レジスタ 2 (RENV2) の設定により, CMP1, または CMP2 の比較条件が成立した時, 自動的にカウンタ値をクリアできます。

環境レジスタ 2 (RENV2) の bit28=0 かつ,

環境レジスタ 2 (RENV2) の bit20=1 の時, CMP 1 条件一致で比較カウンタが 0 クリアされます。

環境レジスタ 2 (RENV2) の bit21=1 の時, CMP 2 条件一致で比較カウンタが 0 クリアされます。

5. 4. 4 コンパレータ比較方法

コンパレータの比較カウンタは, コンパレータ毎にカウンタ 1, またはカウンタ 2 を選択します。

比較方法は環境レジスタ 2 の設定により, 主に以下の 3 種類あります。

- ① 比較データと比較カウンタとの間で, <, = (カウント方向判別付), > の判断
- ② ウィンドウ・コンパレータ機能 (比較データと比較カウンタの間で, > の判断後の比較結果を AND, または OR 条件で結合して比較判断)
- ③ カウンタ同士を比較判断

(1) コンパレータ比較カウンタの設定

環境レジスタ 1 (RENV1) の bit27~24 の設定

CMP 1 : bit24=0 でカウンタ 1 (CTR1), bit24=1 でカウンタ 2 (CTR2)

CMP 2 : bit25=0 でカウンタ 1 (CTR1), bit25=1 でカウンタ 2 (CTR2)

CMP 3 : bit26=0 でカウンタ 1 (CTR1), bit26=1 でカウンタ 2 (CTR2)

CMP 4 : bit27=0 でカウンタ 1 (CTR1), bit27=1 でカウンタ 2 (CTR2)

(2) コンパレータ比較条件の設定

環境レジスタ 2 の bit15~bit0 の設定

7	6	5	4	3	2	1	0
CMPOUT2 COND SEL				CMPOUT1 COND SEL			

区分	bit				※CMP 1 と CMP 2 に設定する値は共通 (下表)	
CMP 1	3	2	1	0	CMP1 の比較方法	
CMP 2	7	6	5	4	CMP2 の比較方法	
	0	0	0	0	RCMP1 < [CTR1/CTR2]	
	1	0	0	0	RCMP1 > [CTR1/CTR2]	
	2	0	0	1	RCMP1 = [CTR1/CTR2]	カウント方向に無関係
	3	0	0	1		カウントアップ時
	4	0	1	0		カウントダウン時
	5	0	1	0	RCMP2 < [CTR1/CTR2]	
	6	0	1	1	RCMP2 > [CTR1/CTR2]	
	7	0	1	1	RCMP2 = [CTR1/CTR2]	カウント方向に無関係
	8	1	0	0		カウントアップ時
	9	1	0	0		カウントダウン時
	10	1	0	1	RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2	
	11	1	0	1	RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2	
	12	1	1	0	(RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2) AND (RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4)	
	13	1	1	0	(RCMP1 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP2) OR (RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4)	
	14	1	1	1	(RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2) AND (RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4)	
	15	1	1	1	(RCMP1 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP2) OR (RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4)	
※[CTR1/CTR2] の表記は RENV1 CMP1·C(bit24)~CMP4·C(bit27) で決定されるカウンタの意味						

15	14	13	12	11	10	9	8
CMPOUT4 COND SEL				CMPOUT3 COND SEL			

区分	bit				※CMP3とCMP4に設定する値は共通（下表）		
CMP3	11	10	9	8	CMP3の比較方法		
CMP4	15	14	13	12	CMP4の比較方法		
	0	0	0	0	RCMP3 < [CTR1/CTR2]		
	1	0	0	0	RCMP3 > [CTR1/CTR2]		
	2	0	0	1	RCMP3 = [CTR1/CTR2]	カウント方向に無関係	
	3	0	0	1		カウントアップ時	
	4	0	1	0		カウントダウン時	
	5	0	1	0	RCMP4 < [CTR1/CTR2]		
	6	0	1	1	RCMP4 > [CTR1/CTR2]		
	7	0	1	1	RCMP4 = [CTR1/CTR2]	カウント方向に無関係	
	8	1	0	0		カウントアップ時	
	9	1	0	1		カウントダウン時	
	10	1	0	1	RCMP3 < [CTR1/CTR2] AND [CTR1/CTR2] < RCMP4		
	11	1	0	1	RCMP3 > [CTR1/CTR2] OR [CTR1/CTR2] > RCMP4		
	12	1	1	0	CTR2 < CTR1		
	13	1	1	0	CTR2 > CTR1		
	14	1	1	1	CTR2 = CTR1		
	15	1	1	1	常に比較条件不成立		
※[CTR1/CTR2]の表記は RENV1 CMP1・C(bit24)~CMP4・C(bit27) で決定されるカウンタの意味							

表5. 4-1 コンパレータ比較条件の設定

5. 5 イベントタイマ

#1CCL (#2CCL) には1組のTIMER (EVENT TIMER) があります。
このTIMER 機能は次の通りです。

(1) TIMER 周期でCTRラッチ、ラッチ後CTRクリア (同一CCLのChのみ)、全CTR同時ラッチ。

■TIMER 周期でCTRラッチ、ラッチ後CTRクリア (同一CCLのChのみ)

環境レジスタ1 (RENV1) のbit30~28 の設定

- 000: 処理なし
- 001: カウンタ1の値をラッチ
- 010: カウンタ1の値をラッチ後クリア
- 011: カウンタ2の値をラッチ
- 100: カウンタ2の値をラッチ後クリア
- 101: カウンタ1, 2の値をラッチ
- 110: カウンタ1, 2の値をラッチ後クリア

■全CTR同時ラッチ

- 一致出力設定と同時ラッチ設定ポートを設定します。
- 一致出力設定は「5. 6 一致出力設定」を参照してください。

(2) TIMER 周期で一致出力

- 一致出力設定します。
- 一致出力設定は「5. 6 一致出力設定」を参照してください。

(3) TIMER 周期でPCへ割込み。(DOS版ソフトのみ対応)

5. 6 一致出力設定

一致出力はCCL 1個当たり2式の一致出力端子 (XYOUT1, XYOUT2, ZUOUT1, ZUOUT2) に出力されます。CCLのコンパレータ一致信号 (CMPOUT1~CMPOUT4) 及びイベントタイマ信号 (TMROUT) を選択して、外部へ出力します。

- (1) J 1コネクタ信号出力端子 XYOUT1 はXYCMPOUT1~XYCMPOUT4の何れか1つを選択できます。同様にJ 1コネクタ信号出力端子 ZUOUT1 はZUCMPOUT1~ZUCMPOUT4の何れか1つを選択できます。
- (2) J 1コネクタ信号出力端子 XYOUT2 はXYCMPOUT2かXYTMROUTの何れか1つを選択できます。同様にJ 1コネクタ信号出力端子 ZUOUT2 はZUCMPOUT2かZUTMROUTの何れか1つを選択できます。

次図に「HPCI-CTR522F一致出力ルート選択」の概念を示します。

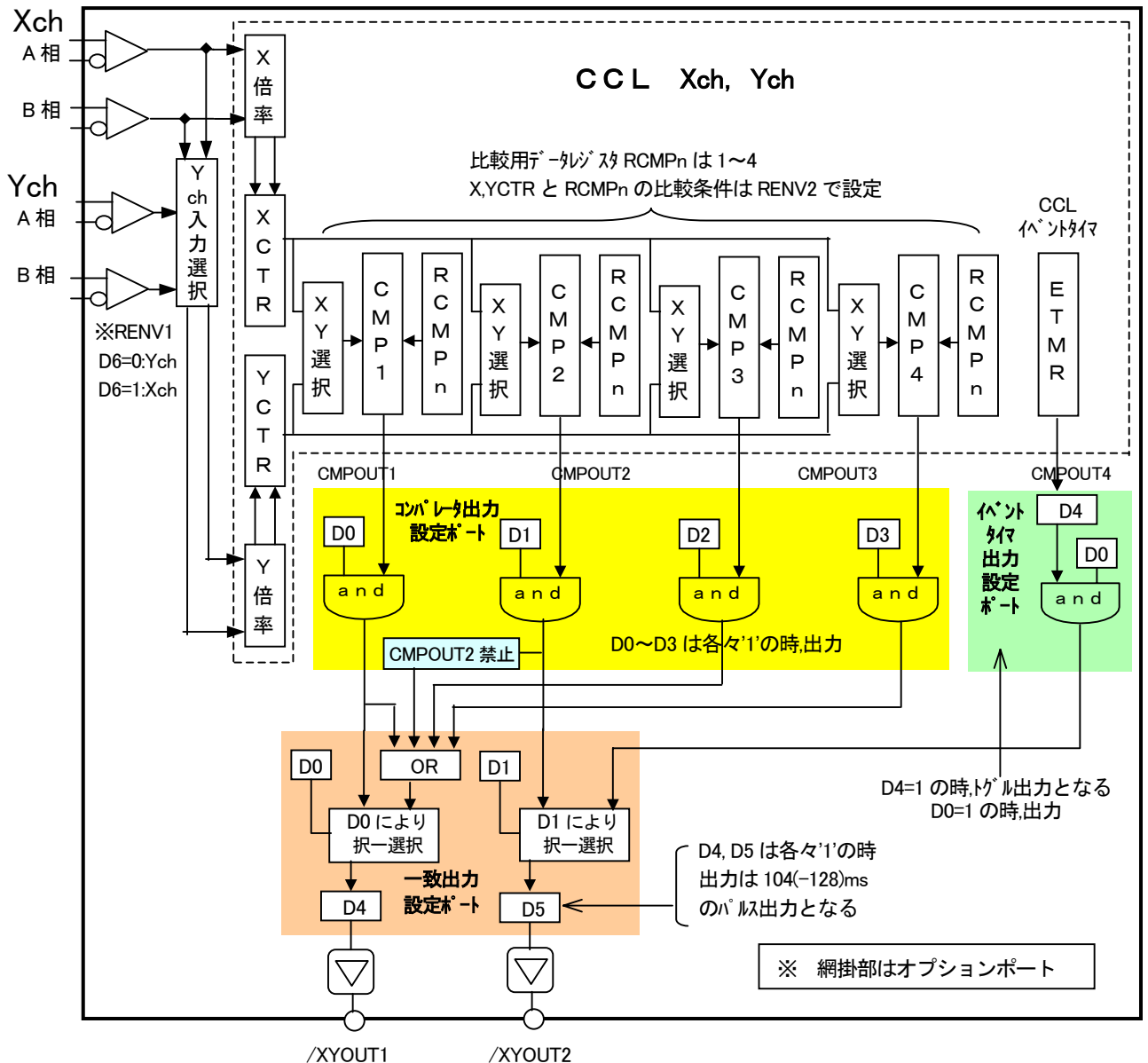


図 5. 6-1 HPCI-CTR522F一致出力ルート選択

尚、オプションポートの設定は各Busの種別により、異なる部分がありますので、設定は各ボードのユーザーズマニュアル<個別ボード編>を参照してください。

5. 7 汎用入出力

(1) 汎用入力ポート

IN1～IN4の汎用入力の状態です。(読出し専用)

‘1’で入力中, ‘0’で入力なし

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	IN4	IN3	IN2	IN1

(2) 汎用出力ポート

OUT1～OUT4の汎用出力状態または汎用出力します。

読出し時は‘1’で汎用出力中, ‘0’で汎用出力なし

書込み時は‘1’で汎用出力ON, ‘0’で汎用出力OFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	OUT4	OUT3	OUT2	OUT1

5. 8 アップ/ダウンカウント時のカウンタ最大値・最小値の測定

アップ/ダウンカウント動作時は, カウンタ1, カウンタ2ともにカウンタ最大値・最小値の測定が行えます。最大値・最小値の測定は, 単独コマンドMM1 START(MM2 START)の書込みにより, 最大値・最小値は現在のカウンタ値にセットされ, ステータス(STS)のCTR1MM(CTR2MM)が1となります。

測定中は, 常にカウンタ値と現在の最大値・最小値とが比較され, カウンタ値>最大値となると最大値が, カウンタ値<最小値となると最小値が更新されます。

単独コマンドMM1 STOP(MM2 STOP)の書込みにより, ステータス(STS)のCTR1MM(CTR2MM)が0となり, 測定は終了します。以後はカウンタ値が変化しても最大値・最小値は変化しません。

また, 測定中でも, 単独コマンドINIT MM1(INIT MM2)の書込みにより, 最大値・最小値は現在のカウンタ値にセットすることができます。

尚, INIT MM1(INIT MM2)の書込みでビットは記憶されませんので解除コマンドは不要です。

(1) カウンタ2(CTR2)の動作方法の設定

環境レジスタ1(RENV1)のbit7=0 (アップ/ダウンカウント)

(2) 最大値・最小値測定状態

ステータス(STS)のbit6, 7

bit6=1 : カウンタ1の最大値・最小値測定中, bit6=0 : カウンタ1の最大値・最小値測定停止中

bit7=1 : カウンタ2の最大値・最小値測定中, bit7=0 : カウンタ2の最大値・最小値測定停止中

(3) 最大値・最小値測定開始コマンド

単独コマンドMM1 START (0048h) 書込みでカウンタ1の最大値・最小値測定を開始します。

単独コマンドMM2 START (004ah) 書込みでカウンタ2の最大値・最小値測定を開始します。

その後, カウントスタートの処理をします。

(4) 最大値・最小値測定終了コマンド

単独コマンドMM1 STOP (0049h) 書込みでカウンタ1の最大値・最小値測定を終了します。

単独コマンドMM2 STOP (004bh) 書込みでカウンタ2の最大値・最小値測定を終了します。

その後, カウントストップの処理をします。

(5) 最大値・最小値初期化コマンド

単独コマンドINIT MM1 (0004h) 書込みでカウンタ1の最大値・最小値は現在のカウンタ値にセットされます。

単独コマンドINIT MM2 (0008h) 書込みでカウンタ2の最大値・最小値は現在のカウンタ値にセットされます。

5. 9 信号幅の計測

信号幅の計測はCTR2のみ可能です。

到来するパルスのA相、B相の[開始エッジ～終了エッジ]を[立下り～立上り]、または[立上り～立下り]、または[立下り～立下り]、または[立上り～立上り]間のパルス幅が計測出来ます。

計測は[エッジ～エッジ]間に40MHzすなわち25ns周期CLOCKをCTR2がカウントした数を読みます。
パルスの幅は=25ns x カウント値となります。

計測開始はCTR2に対するMM開始コマンド発行後の[エッジ～エッジ]条件でカウント開始、終了します。
MM開始コマンド発行すると、カウンタ2(CTR2)、最大値(MAX2)、最小値(MIN2)、カウンタ2のラッチレジスタ(LTCH2)をクリアします。

以降[エッジ～エッジ]の開始、終了ごとに最大値(MAX2)、最小値(MIN2)が更新されます。

(CTR2値>MAX2値→MAX2値更新, CTR2値<MIN2値→MIN2値更新)

MM終了コマンド発行以降はカウンタ2(CTR2)動作停止、最大値(MAX2)、最小値(MIN2)、カウンタ2のラッチレジスタ(LTCH2)は変化しません。

- (1) カウンタ2(CTR2)の動作方法の設定
環境レジスタ1(RENV1)のbit7=1(入力信号のエッジ間測定)
- (2) カウンタ2のパルス入力端子の設定
環境レジスタ1(RENV1)のbit6=0:CTR2入力信号YA_x, YB_x(UA_x, UB_x)
環境レジスタ1(RENV1)のbit6=1:CTR2入力信号XA_x, XB_x(ZA_x, ZB_x)
- (3) エッジ間測定の開始条件及び終了条件の設定
環境レジスタ1(RENV1)のbit8~bit11

YCTR(UCTR)入力信号のエッジ間測定(C2MES=1)			開始条件		CTR2MES ST SEL:b 9, 8	
			終了条件		CTR2MES END SEL:b11, 10	
入力信号がYA _x , YB _x (UA _x , UB _x) (CTR2IN=0)			入力信号がXA _x , XB _x (ZA _x , ZB _x) (CTR2IN=1)			
b 9, 8 b11, 10	信号	変化	b 9, 8 b11, 10	信号	変化	
0 0	YA _x	立上り	0 0	XA _x	立上り	
0 1	(UA _x)	立下り	0 1	(ZA _x)	立下り	
1 0	YB _x	立上り	1 0	XB _x	立上り	
1 1	(UB _x)	立下り	1 1	(ZB _x)	立下り	

表 5. 9-1 信号幅・エッジ間測定の条件設定

- (4) 最大値・最小値測定開始コマンド
単独コマンドMM2 START (004ah)書き込みとカウントスタート処理でカウンタ2の最大値・最小値測定を開始します。
- (5) 最大値・最小値測定終了コマンド
単独コマンドMM2 STOP (004bh)書き込みとカウントストップ処理でカウンタ2の最大値・最小値測定を終了します。
- (6) 最大値・最小値初期化コマンド
単独コマンドINIT MM2 (0008h)書き込みでカウンタ2の最大値・最小値は現在のカウント値にセットされます。

【応用】

- 周期測定・・・[エッジ～エッジ]の条件を[立下り～立下り]または[立上り～立上り]とすれば、パルス周期の測定が出来ます。
- 位相差測定・・・A相、B相としてそれぞれの[エッジ～エッジ]条件を[A相の立上り～B相の立上り]とすれば、位相差の測定が出来ます。

5. 10 割り込み機構と割り込み処理

割り込み処理はDOSあるいはRTOS等のリアルタイム性の得られるOSのもとで可能です。
Windowsにおいてはサポートしていません。

5. 10. 1 割り込み機構

割り込み要因によって割り込みは次のルートで発生します。

- (1) #1 CCL → CMP 成立, CTR ゼロ, 全ch同時ラッチ, 幅計測終了エッジ, タイマ周期等
- (2) #2 CCL → CMP 成立, CTR ゼロ, 全ch同時ラッチ, 幅計測終了エッジ, タイマ周期等
- (3) 汎用入力ポート → IN1 on

何れも割り込み許可状態としてのことです。#1, #2 CCLの場合はそれぞれのRIRQで設定。さらに、オプションポートの割り込みマスクによってボードに最終的なグループ割り込みマスクが設けられています。汎用ポートの割り込みマスクはここでのみ行います。

以上の割り込み機構を「図5. 10-1 割り込み機構」に示します。

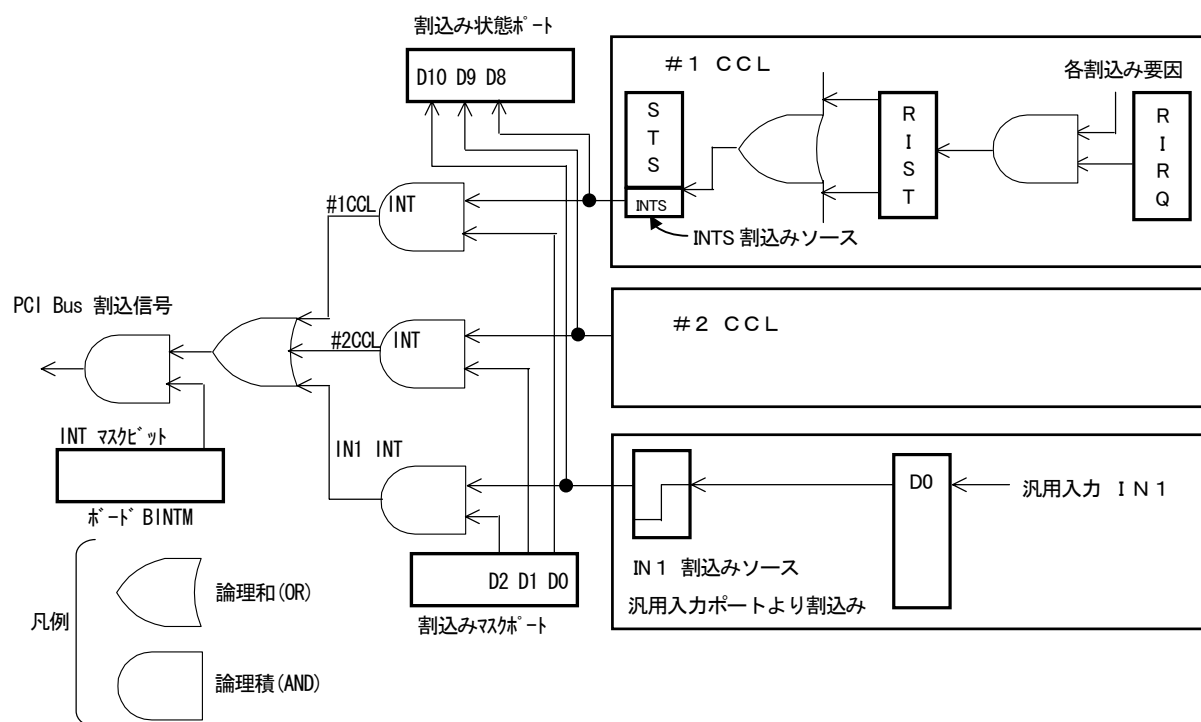


図5. 10-1 割り込み機構 (PCI Bus)

5. 10. 2 割込み処理

以下に処理手順を次に示します。詳細は各ボード ユーザーズマニュアル<個別編>を参照してください。

(1) 初期の設定

初期時に使用目的に応じてCCLのRIRQを設定しておきます。さらに割込みマスクポートで3ルートのマスク(#1,#2CCL,IN1)を決めておきます。割込み運用するルートはマスクをしません。PCIバスではPCI割込みポートでPCIバスへの割込み出力を許可します。ISA, PC/104ではジャンパ設定します。

(2) 割込み

動作を開始したあとは、割込みにより、ユーザーの書いた割込みエントリにプログラムコントロールが渡ります。ここですべきことは次の処理です。

① 「割込みマスクポート」を全てマスク (ALL '0') する

- ② 「割込み状態ポート」を読み、割込みの生起しているルートのCCLまたはIN1を取り上げて処理を行う。割込みがCCLの場合はSTSのINTS='1'です。RISTをリードし割込み原因を解析します。RISTをリードすると、INTSは '0'となります。割込みがIN1の場合は「割込み状態ポート」をリードするとIN1割込み中のビットが '0' となります。割込み原因を処理してから、次のCCL, またはIN1の処理を行います。

③ 割込みから抜け出すときに①で行ったマスクをはずす。

もし、この割込み処理の途中で以前のルートに割込みが生起していれば、抜け出す直前のマスクをはずした時点で、再び割込みがCPUに到来します。

毎回の割込みは 以上の繰り返しとなります。

ポーリング方式では①と③の処理をしません。適当な間隔で、②をポーリングによって実施します。

要点は、「割込み状態ポート」を読む。このポートの割込みがあるルートの#nCCLのRISTを読むことにより、STSのINTSをクリアすることにあります。

6. ソフトウェア編

この章ではCTRボードのソフトウェア共通部分について説明をします。

対応するOSとして次の種類があります。

- WindowsXP Professional/Home Edition ・以降WinXPと記します
- Windows2000以降Win2Kと記します
- WindowsNT4.0以降WinNTと記します
- Windows98以降Win98と記します
- PC DOS, MS-DOS以降DOS版 と記します

デバイスドライバのI/F用ライブラリとしてデバイスドライバI/F用DLL（DOS版ではLIB）が用意されています。このDLL（LIB）関数をドライバ関数と称します。

デバイスドライバは、個々のOS毎にあります。

デバイスドライバ関数はCTRボードの各ポートアドレスへの入出力を制御します。

（注）デバイスドライバのインストール、アンインストールは ユーザーズマニュアル<個別ボード編>に記載されています。

各ボードに対応するファイル名、関数名を以下の表に示します。

ボ ー ド 種 別		H P C I		H P C		H P C 1 0 4	
バ ス 種 別		P C I		I S A		P C / 1 0 4	
ボ ー ド		C T R 5 2 4 F / 5 2 2 F		C T R 2 2 4 F / 2 2 2 F		C T R 1 2 2 F	
Windows 版	フ ア イ ル 名	ドライバI/F用DLL		hictr520. dll	hctr220. dll	hctr120. dll	
		C言語用 インポートライブラリ		hictr520. lib	hctr220. lib	hctr120. lib	
		C言語用 関数結合用ヘッダー		hictr520. h	hctr220. h	hctr120. h	
		VB用 関数定義標準モジュール		hictr520. bas	hctr220. bas	hctr120. bas	
DOS版	フ ア イ ル 名	ドライバ I/F用 LIB	メ モ リ モ デ ル	ラージ	lctr520. lib	lctr220. lib	lctr120. lib
				ミディアム	mctr520. lib	mctr220. lib	mctr120. lib
				コンパクト	cctr520. lib	cctr220. lib	cctr120. lib
				スモール	sctr520. lib	sctr220. lib	sctr120. lib
		C言語用 関数結合用ヘッダー		hctr520. h hcopdtype. h	hctr220. h	hctr120. h	
関数名 (Windows版・DOS版共通)				ct520_xxxx ()	ct220_xxxx ()	ct120_xxxx ()	

表6. 1-1 各ボード別ファイル名・関数名対応表

※ I N T E L互換のCPUを搭載したマシン用です。その他のプラットフォームには対応していません。
ドライバ関数名の” xxxx” 部分については「表6. 1-2 ドライバ関数とボード種別の対応表」に記載されています。

以降、H P C I - C T R 5 2 4 F / 5 2 2 Fのファイル名、関数名で説明します。

No	関数名	機能	HPCI-	HPC-	HPC104-
			524F/ 522F	224F/ 222F	122F
関数名“xxx” (右欄の3桁数値)			520	220	120
1	ctxxx_GetDeviceCount()	ボード枚数の取得	○	×	×
2	ctxxx_GetDeviceInfo()	デバイス情報の取得	○	×	×
3	ctxxx_OpenDevice()	デバイスのオープン	○	○(※1)	○(※1)
4	ctxxx_CloseDevice()	デバイスのクローズ	○	○	○
5	ctxxx_rXYSts()	X Y c hステータスの読込	○	○	○
	ctxxx_rZUSts()	Z U c hステータスの読込	○	○	○
6	ctxxx_wXYCmd()	X Y c h制御コマンド書込	○	○	○
	ctxxx_wZUCmd()	Z U c h制御コマンド書込	○	○	○
7	ctxxx_rXYReg()	X Y c hレジスタの読込	○	○	○
	ctxxx_rZUReg()	Z U c hレジスタの読込	○	○	○
	ctxxx_wXYReg()	X Y c hレジスタの書込	○	○	○
	ctxxx_wZUReg()	Z U c hレジスタの書込	○	○	○
8	ctxxx_rPortB()	オプションポートのバイト読込	○	○	○
	ctxxx_rPortW()	オプションポートのワード読込	○	○	○
	ctxxx_wPortB()	オプションポートへバイト書込	○	○	○
	ctxxx_wPortW()	オプションポートへワード書込	○	○	○
9	ctxxx_rXYBuf()	X Y c h入出力バッファの読込	○	○	○
	ctxxx_rZUBuf()	Z U c h入出力バッファの読込	○	○	○
	ctxxx_wXYBuf()	X Y c h入出力バッファの書込	○	○	○
	ctxxx_wZUBuf()	Z U c h入出力バッファの書込	○	○	○
10	ctxxx_SetIntCall()	割込処理関数の登録/解除	○(※2)	×	×
11	ctxxx_GetDevVrtNo()	ドライバのバージョン番号取得	○(※3)	×	×

※1 HPC, HPC104ボード DOS版で割込使用時の割込処理関数の登録(削除)を行います。

※2 HPCIボード DOS版で割込使用時の割込処理関数の登録(削除)を行います。

※3 HPCIボード DOS版で“デバイスドライバとドライバ関数”バージョン番号の取得が可能です。

表6. 1-2 ドライバ関数とボード種別の対応表

6. 1 ソフトウェアの概要

弊社の提供するソフトウェアは、ドライバ関数、デバイスドライバです。アプリケーションプログラムと、これらのソフトウェアの関連は次図の通りです。

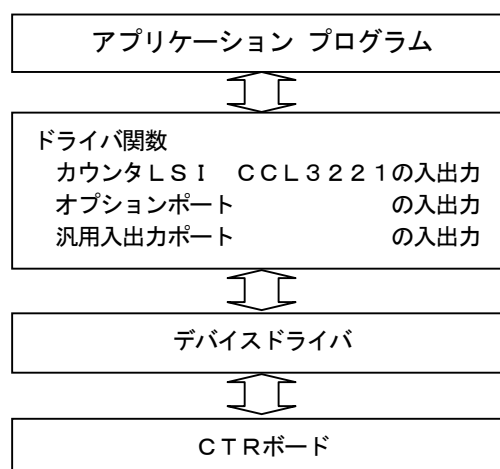


図6. 1-1 ソフトウェアの関連

6. 2 準備

ドライバ関数では複数のCTRボードを制御することができます。

あるCTRボードを制御するために、まずこのデバイスをオープンして、デバイスハンドル値を取得します。

デバイスをオープンするためには、オープンするデバイスのデバイス情報（ハードウェアリソース情報）が必要となります。（ハードウェアリソース → ボードアドレス、ボードID等）

(1) HPCI-CTR524F/522Fのボード認識用のデータ構造体

■Windows版ボード（デバイス）認識用のデータ構造体

ボード認識のために次に示す HCTRDEVINF 型構造体を、ボード枚数最大16枚として、使用枚数分用意します。

[C言語 : Visual C++]

```
typedef struct _HPCDEVICEINFO {
    DWORD   nBusNumber;           /* バス番号 */
    DWORD   nDeviceNumber;       /* デバイス番号*/
    DWORD   dwIoPortAddress;     /* I/O ポートアドレス */
    DWORD   dwIrqNo;             /* IRQ 番号 */
    DWORD   dwNumber;            /* 管理番号 */
    DWORD   dwBoardID;           /* ボードID (0~15) */
} HPCDEVICEINFO, *PHPCDEVICEINFO
```

[Visual Basic]

```
Public Type HPCDEVICEINFO
    nBusNumber As Long 'バス番号
    nDeviceNumber As Long 'デバイス番号
    dwIoPortAddress As Long 'I/O ポートアドレス
    dwIrqNo As Long 'IRQ 番号
    dwNumber As Long '管理番号
    dwBoardID As Long 'ボードID (0~15)
End Type
```

(注) 1. 管理番号はWindows 98 では使用されません。

常に「INVALID_HPC_NUMBER (-1/VB, 0xffffffff/C言語)」が格納されています。

■DOS版ボード（デバイス）認識用のデータ構造体

ボード認識のために次に示す HPCDEVINFO 型構造体を、ボード枚数最大16枚として、使用枚数分用意します。

```
typedef unsigned short WORD;
typedef struct {
    WORD   nBusNumber;           /* バス番号 */
    WORD   nDevNumber;          /* デバイス番号 */
    WORD   dwIoPortAdrs;       /* I/O ポートアドレス */
    WORD   dwIrqNo;            /* IRQ 番号 */
    WORD   dwNumber;           /* 管理番号 */
    WORD   dwBoardID;          /* ボードID */
} HPCDEVINFO, *PHPCDEVINFO, far *LPHPCDEVINFO;
```

(2) HPC-CTR224F/222F, HPC104-CTR122Fのボード認識用のデータ構造体

■Windows版ボード（デバイス）認識用のデータ構造体

ボード認識のために次に示す HCTRDEVINF 型構造体を、ボード枚数最大16枚として、使用枚数分用意します。

[C言語 : Visual C++]

```
typedef struct _HCTRDEVINF {
    DWORD   dwIoPortAddress;     /* I/O ポートアドレス */
    DWORD   dwCh;                /* 使用ボードのch数 (CTR224は4, CTR222, CTR122は2) */
    DWORD   dwReserved1;        /* 予約 */
    DWORD   dwReserved2;        /* 予約 */
} HCTRDEVINF, *PHCTRDEVINF
```

[Visual Basic]

```
Public Type HCTRDEVINF
    dwIoPortAddress As Long ' ボードアドレス
    dwCh As Long ' 使用ボードのch数 (CTR224 は 4, CTR222, CTR122 は 2)
    dwReserved1 As Long ' 予約
    dwReserved2 As Long ' 予約
End Type
```

■DOS版ボード（デバイス）認識用のデータ構造体

ボード認識のために次に示す HCTRDEVINF 型構造体を、ボード枚数最大16枚として、使用枚数分用意します。

```
typedef struct _HCTRDEVINF {
    short    badr;          /* board address */
    short    count;        /* channel count */
    short    intno;        /* interrupt no. (0/3/5/6/7/11/12) */
    PINTPROC module;      /* intrerrupt module(/NULL) */
} HCTRDEVINF;
```

(3) アプリケーションの構築

[Visual C++ (5.0 以上) によるアプリケーションの構築]

次のファイルをプロジェクトへ追加して下さい。

■プロジェクト追加ファイル

・hictr520.lib ・ ・ ドライバI/F用DLLインポートライブラリ

■インクルードファイル

・hictr520.h ・ ・ ドライバI/F用DLL関数結合用ヘッダーファイル

- (注) 1. ドライバI/F用DLL関数はC言語で作成されています。
 2. ドライバI/F用DLL関数のプロトタイプ宣言は次のように記述されています。

<pre>//----- // 関数プロトタイプ宣言 //----- #ifdef __cplusplus extern "C" { #endif #ifdef __cplusplus ----- 関数のプロトタイプ宣言 ----- #endif } #endif</pre>	<p>これは、アプリケーションをC++コーディング（ファイル拡張子=cpp）で作成する場合に備えての処理です。</p>
---	---

3. 「#ifdef __cplusplus」の定義は“Visual C++”用です。
 他言語で使用する場合には、明示的な宣言に変更できます。

<p>(例)</p> <pre>#define CPLUS 1 #if CPLUS #endif</pre>	<p>“1”でC++用（ファイル拡張子：cpp） “0”でC用（ファイル拡張子：c）</p>
--	---

[Visual Basic (5.0/6.0) によるアプリケーションの構築]

次のファイルをプロジェクトへ追加して下さい。

■ ドライバ用

- ・ hctr520. bas ・ ・ ドライバ I / F 用 D L L 関数定義標準モジュールファイル
このファイルに外部関数宣言 (Declare 宣言)、及びユーザー定義型宣言が記述されています。

[DOS版 : C言語 (MS-C, その他) によるアプリケーションの構築]

次のファイルをコンパイル・リンクへ追加して下さい。

■ コンパイル用追加ファイル・・・インクルードファイル

- ・ hctr520. h ・ ・ ドライバ I / F 用 L I B 関数結合用ヘッダーファイル
- ・ hcpdtype. h ・ ・ H P C I ボード用ヘッダーファイル (構造体定義)
「hctr520. h」ファイルからインクルードされます。

■ リンク用追加ファイル・・・スタティックリンク ライブラリ

- ・ lctr520. lib ・ ・ ・ ・ ラージモデル用
- ・ mctr520. lib ・ ・ ・ ・ メディアムモデル用
- ・ cctr520. lib ・ ・ ・ ・ コンパクトモデル用
- ・ sctr520. lib ・ ・ ・ ・ スモールモデル用

- (注) 1. ドライバ I / F 用 L I B 関数は C 言語で作成されています。
 2. アプリケーション作成メモリモデルと同一のドライバ関数メモリモデルを使用します。
 3. 割込処理を行う場合は、ラージモデルとして下さい。

6. 3 ドライバ関数の戻り値

ドライバの諸関数を使用する時、関数の戻り値が異常値（'0'以外）であった場合には、異常内容に対応した処理を行います。

No	戻り値			異常内容と確認項目
	記号表記	16進数表記		
		C言語	VB	
1	NO_ERROR	0x000	&H0	正 常 異常は発生していません
2	NOT_FOUND	0x001	&H1	デバイスが見つからない ◎デバイスドライバがインストールされていない ◎デバイスドライバが所定のフォルダに格納されていない ◎CTRボードがPCに挿入されていない
3	ALREADY_OPENED	0x002	&H2	既にオープン済のデバイスをオープン ◎オープン済みデバイスに更にオープン指令 ◇オープンしたデバイスはクローズするまで使用 （多重のオープンは禁止） ◎ボード2枚以上を使用する場合、オープンするデバイス情報を確認します。
4	NOT_MEMORY	0x004	&H4	デバイス情報格納メモリが不足 ◎アプリケーション用のメモリ不足 ◇パソコン主記憶メモリの不足 ◎システムリソース（OS用メモリ）の不足 ◇多数のアプリケーション起動 ◇1度に多数のウィンドウを開いた
5	INVALID_HANDLE	0x008	&H8	無効なデバイスハンドルを指定 ◎デバイスオープンで得られた“デバイスハンドル”の不使用 ◎このデバイスは既にクローズされている
6	NOT_READY	0x010	&H10	デバイスの入出力ポートが使用できない ◎システムが不安定になっている可能性がありますので、弊社サポートまでお問い合わせください
7	ILLEGAL_DEVICE	0x020	&H20	ボード固有情報が不正 ◎システムが不安定になっている可能性がありますので、弊社サポートまでお問い合わせください
8	ILLEGAL_ADDRESS	0x040	&H40	不正なベースアドレス（HPCボード） ◎指定したベースアドレスの確認
9	ILLEGAL_ACCESS	0x080	&H80	読込/書込み中の軸への読込/書込指令（DOS版） ◎多重入出力処理の指令 ボード上LSIへの読込/書込み中に同一ポートへの読込/書込指令が行われた。 この結果としてLSIの動作が保証されなくなり、読込/書込指令を禁止した。 ◇マルチタスク処理では先行タスクの処理終了まで待つ。 ◇割込処理モジュール内では、この処理を“待処理”とし、割込処理を終了させる。
10	ILLEGAL_PARAM	0x100	&H100	関数の引数の値が異常

表 6. 3-1 ドライバ関数の戻り値

6. 4 ドライバ関数詳細

関数説明文中で引数のデータ型、及び16進数の表記はC言語記述で行っています。
Visual Basicでご利用の場合には、データ型を次のようにして下さい。
また、C言語16進数表記を次のように読み替えて下さい。

言語区分	C言語		Visual Basic
データ型	Windows版 (32ビット)	BYTE	Byte
		WORD	Integer
		DWORD	Long
	DOS版 (16ビット)	char	(Byte)
		U_CHAR (BYTE) (unsigned char)	
		short	(Integer)
		U_SHORT (WORD) (unsigned short)	
		long	(Long)
		U_LONG (DWORD) (unsigned long)	

表 6. 4-1 ドライバ関数のデータ型

言語区分	C言語	Visual Basic
16進数表記	0x0000	&H0
	0x1000	&H1000
	0xffffffff	-1

表 6. 4-2 各言語の数値表記

6. 4. 1 Windows 版ドライバ関数

(1) ct520_GetDeviceCount () ボード枚数の取得

《機能》

現在パソコンに装着されている C T R ボードの枚数を取得します。 (H P C I - C T R 5 2 4 F / 5 2 2 F のみ)

《書式》

[C 言語]

```
DWORD WINAPI ct520_GetDeviceCount (DWORD* HpcDevNumber );
```

[Visual Basic]

```
Declare Function ct520_GetDeviceCount Lib "hict520.dll" (ByRef HpcDevNumber As Long) As Long
```

《引数》

◆ DWORD* *HpcDevNumber* ・ ・ C T R ボードの枚数

《戻り値》 処理結果

0 : 成功

0 以外 : 失敗 ・ ・ 「 6 . 3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C 言語]

```
DWORD count; // C T R ボードの枚数
```

```
DWORD ret; // 関数の戻り値
```

```
ret = ct520_GetDeviceCount ( &count );
```

[Visual Basic]

```
Dim count As Long ' C T R ボードの枚数
```

```
Dim ret As Long ' 関数の戻り値
```

```
ret = ct520_GetDeviceCount ( count )
```

(2) ct520_GetDeviceInfo() デバイス情報の取得

《機能》

現在パソコンに装着されているCTRボードのデバイス情報(※1)を取得します。この結果、HPCDEVICEINFO型(※2)の配列にデバイス情報が格納されます。この値は、デバイスオープン時に利用します。(HPCI-CTR524F/522Fのみ)

《書式》

[C言語]

```
DWORD WINAPI ct520_GetDeviceInfo( DWORD* HpcDevNumber, HPCDEVICEINFO* HpcDevInfo );
```

[Visual Basic]

```
Declare Function ct520_GetDeviceInfo Lib "hict520.dll" (ByRef HpcDevNumber As Long, _  
HpcDevInfo As HPCDEVICEINFO) As Long
```

《引数》

- ◆ DWORD* *HpcDevNumber* ・ ・ 情報を取得するボードの最大枚数が格納された DWORD型エリアのアドレスを渡します。
関数の呼び出し後、実際に情報を取得したボードの枚数が格納されます。
- ◆ HPCDEVICEINFO *HpcDevInfo* ・ ・ 各ボードのデバイス情報がセットされるべきエリアのアドレス、すなわちHPCDEVICEINFO型の配列の先頭アドレスを渡します。

《戻り値》 処理結果

- 0 : 成功
- 0以外 : 失敗 ・ ・ 「6.3 関数の戻り値」を参照して下さい。

《呼び出し例》 パソコンにCTRボードが2枚装着されていることを想定します。

[C言語]

```
DWORD ret; //関数の戻り値  
DWORD count = 2; //最大枚数は2  
HPCDEVICEINFO HpcDevInfo[2]; //2枚のCTRボードのデバイス情報がセットされるべきエリア
```

```
ret = ct520_GetDeviceInfo( &count, //countのアドレスを渡す。  
&HpcDevInfo[0] ); //配列の先頭アドレスを渡す。
```

[Visual Basic]

```
Dim ret As Long '関数の戻り値  
Dim count As Long '枚数  
Dim HpcDevInfo(2) As HPCDEVICEINFO 'デバイス情報のエリア
```

```
count = 2 '最大枚数は2枚
```

```
ret = ct520_GetDeviceInfo( count, _ 'countのアドレスを渡す。  
HpcDevInfo(0) ) '配列の先頭アドレスを渡す。
```

※1, ※2. デバイス情報格納の構造体名及び構造体メンバは、ドライバの種類により異なる場合があります。

(3) ct520_OpenDevice() デバイスのオープン

《機能》

渡したデバイス情報を持つCTRボードをオープンし、他のCTRボードと識別するためのデバイスハンドルを取得します。以降このデバイスハンドルは、このCTRボードにアクセスするためのハンドルとなります。

《書式》

[C言語]

```
DWORD WINAPI ct520_OpenDevice( DWORD * hDevID, HCTRDEVINF * HpcDevInfo );
```

[Visual Basic]

```
Declare Function ct520_OpenDevice Lib "hict520.dll" (Byref hDevID As Long, _  
HpcDevInfo As HCTRDEVINF) As Long
```

《引数》

- ◆ DWORD *hDevID* .. デバイスハンドル
- ◆ HCTRDEVINF* *HpcDevInfo* .. オープンするデバイスの情報がセットされたエリアのアドレス
(※ 構造体のメンバーはボード種別により異なります。)

《戻り値》 処理結果

- 0 : 成功
- 0以外 : 失敗 .. 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

パソコンにCTRボードが2枚装着されていることを想定します。
デバイス情報格納エリアとしてHCTRDEVINF型の配列HpcDevInfo[2]を準備します。

[C言語]

```
DWORD ret;            //関数の戻り値  
DWORD hDevID[2];     //デバイスハンドル取得エリア  
  
ret = ct520_OpenDevice(&hDevID[0], &HpcDevInfo[0]);    // 1番目のデバイス情報  
ret = ct520_OpenDevice(&hDevID[1], &HpcDevInfo[1]);    // 2番目のデバイス情報
```

[Visual Basic]

```
Dim ret            As Long    ' 関数の戻り値  
Dim hDevID(2)     As Long    ' デバイスハンドル取得エリア  
  
ret = ct520_OpenDevice(hDevID(0), HpcDevInfo(0))    ' 1番目のデバイス情報  
ret = ct520_OpenDevice(hDevID(1), HpcDevInfo(1))    ' 2番目のデバイス情報
```

《デバイスハンドル》

デバイスハンドルは次のデータ内容となっています。

31-28	27-24	23-20	19-16	15-12	11- 8	7- 4	3- 0
ボードID設定値				デバイスオープンNo			
0	0	0	0~15	0	0	1~16	

(4) ct520_CloseDevice() デバイスのクローズ

《機能》

デバイスハンドルで指定されたCTRボードをクローズします。
以降、このデバイスハンドルは無効となります。

《書式》

[C言語]

```
DWORD WINAPI ct520_CloseDevice( DWORD hDevID );
```

[Visual Basic]

```
Declare Function ct520_CloseDevice Lib "hict520.dll" ( ByVal hDevID As Long ) As Long
```

《引数》

◆ **DWORD hDevID** . . . クローズするボードのデバイスハンドル

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 . . . 「6.3 関数の戻り値」を参照して下さい。

《呼び出し例》

既にデバイスハンドルとして hDevID が取得されているものとします。

[C言語]

```
DWORD ret; //関数の戻り値
```

```
ret = ct520_CloseDevice( hDevID );
```

[Visual Basic]

```
Dim ret As Long ' 関数の戻り値
```

```
ret = ct520_CloseDevice( hDevID )
```

<p>(5) ct520_rXYSts() XYchステータスの読込 ct520_rZUSts() ZUchステータスの読込</p>
--

《機能》

デバイスハンドルで指定されたCTRボードの、XY(ZU)chのステータスを読込み、指定したエリアに格納します。

《書式》

[C言語]

```
DWORD WINAPI ct520_rXYSts1(DWORD hDevID, WORD * sts);
DWORD WINAPI ct520_rZUSts1(DWORD hDevID, WORD * sts);
```

[Visual Basic]

```
Declare Function ct520_rXYSts Lib "hict520.dll" (ByVal hDevID As Long, ByRef sts As Integer) As Long
Declare Function ct520_rZUSts Lib "hict520.dll" (ByVal hDevID As Long, ByRef sts As Integer) As Long
```

《引数》

- ◆ DWORD hDevID ・ ・ 対象デバイスのデバイスハンドル
- ◆ WORD * sts ・ ・ 読込んだデータが格納されるエリアのアドレス

《戻り値》 処理結果

0 : 成功
0以外 : 失敗 ・ ・ 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C言語]

```
DWORD ret;     //関数の戻り値
WORD sts;     //ステータス

ret = ct520_rXYSts ( hDevID,         //デバイスハンドル
                                      &sts );         //格納先のアドレス
```

[Visual Basic]

```
Dim ret        As Long         ' 関数の戻り値
Dim sts       As Integer      ' ステータス

ret = ct520_rXYSts ( hDevID, _     ' デバイスハンドル
                                  sts )     ' 格納先のアドレス
```

<p>(6) ct520_wXYCmd() X Y c h 制御コマンドの書込 ct520_wZUCmd() Z U c h 制御コマンドの書込</p>

《機能》

デバイスハンドルで指定されたCTRボードの、XY(ZU)chのコマンドバッファへコマンドデータを書込みます。

《書式》

[C言語]

```
DWORD WINAPI ct520_wXYCmd( DWORD hDevID, WORD cmd );
DWORD WINAPI ct520_wZUCmd( DWORD hDevID, WORD cmd );
```

[Visual Basic]

```
Declare Function ct520_wXYCmd Lib "hict520.dll" ( ByVal hDevID As Long, ByVal cmd As Integer ) As Long
Declare Function ct520_wZUCmd Lib "hict520.dll" ( ByVal hDevID As Long, ByVal cmd As Integer ) As Long
```

《引数》

- ◆ DWORD *hDevID* .. 対象デバイスのデバイスハンドル
- ◆ WORD *cmd* .. コマンドデータ

《戻り値》 処理結果

0 : 成功
0以外 : 失敗 .. 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C言語]

```
DWORD ret;    //関数の戻り値

ret = ct520_wXYCmd( hDevID,    //デバイスハンドル
                    0x03 );    //X Y c hカウンタクリア
```

[Visual Basic]

```
Dim ret As Long    ' 関数の戻り値

ret = ct520_wXYCmd( hDevID, _    ' デバイスハンドル
                    &H3 )        ' X Y c hカウンタクリア
```

(7) ct520_rXYReg()	X Y c h レジスタの読込
ct520_rZUReg()	Z U c h レジスタの読込
ct520_wXYReg()	X Y c h レジスタの書込
ct520_wZUReg()	Z U c h レジスタの書込

《機能》

デバイスハンドルで指定されたCTRボードの、

レジスタの読込・・・XY(ZU)chのレジスタ読込コマンドで指定したレジスタを読み込み、指定エリアに格納します。

レジスタの書込・・・XY(ZU)chのレジスタ書込コマンドで指定したレジスタにデータを書込みます。

《書式》

[C言語]

```
DWORD WINAPI ct520_rXYReg(DWORD hDevID, WORD cmd, DWORD * reg);
```

```
DWORD WINAPI ct520_rZUReg(DWORD hDevID, WORD cmd, DWORD * reg);
```

```
DWORD WINAPI ct520_wXYReg(DWORD hDevID, WORD cmd, DWORD reg);
```

```
DWORD WINAPI ct520_wZUReg(DWORD hDevID, WORD cmd, DWORD reg);
```

[Visual Basic]

```
Declare Function ct520_rXYReg Lib "hict520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _  
ByRef reg As Long) As Long
```

```
Declare Function ct520_rZUReg Lib "hict520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _  
ByRef reg As Long) As Long
```

```
Declare Function ct520_wXYReg Lib "hict520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _  
ByVal reg As Long) As Long
```

```
Declare Function ct520_wZUReg Lib "hict520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _  
ByVal reg As Long) As Long
```

《引数》

- ◆ DWORD *hDevID* ・・ 対象デバイスのデバイスハンドル
- ◆ WORD *cmd* ・・ レジスタ読込/書込コマンド
- ◆ DWORD * *reg* ・・ 読込んだデータが格納されるエリアのアドレス
- ◆ DWORD *reg* ・・ レジスタ書込データ

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 ・・ 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C言語]

```
DWORD ret; //関数の戻り値
```

```
DWORD reg; //レジスタのデータ
```

```
ret = ct520_rXYReg( hDevID, //デバイスハンドル  
0x00c0, //X c hのカウンタを読む  
&reg); //格納先のアドレス
```

```
ret = ct520_wXYReg( hDevID, //デバイスハンドル  
0x0080, //X c hのカウンタを指定  
10000 ); //書込データ
```

[Visual Basic]

```
Dim ret As Long ' 関数の戻り値
```

```
Dim reg As Long ' レジスタのデータ
```

```
ret = ct520_rReg( hDevID, _ ' デバイスハンドル  
&Hc0, _ ' X c hのカウンタを読む  
reg ) ' 格納先のアドレス
```

```
ret = ct520_wReg( hDevID, _ ' デバイスハンドル  
&H80, _ ' X c hのカウンタを指定  
10000 ) ' 書込データ
```

(8) ct520_rPortB()	オプションポートのバイト読込
ct520_rPortW()	オプションポートのワード (2バイト) 読込
ct520_wPortB()	オプションポートへバイト書込
ct520_wPortW()	オプションポートへワード (2バイト) 書込

《機能》

デバイスハンドルで指定されたCTRボードの、

オプションポートの読込・・・オプションポートを読み、指定エリアに格納します。

オプションポートへ書込・・・オプションポートに指定データを書込みます。

《書式》

[C言語]

```
DWORD WINAPI ct520_rPortB(DWORD hDevID, BYTE OffsetAdrs, BYTE * byData);
```

```
DWORD WINAPI ct520_wPortB(DWORD hDevID, BYTE OffsetAdrs, BYTE byData);
```

```
DWORD WINAPI ct520_rPortW(DWORD hDevID, BYTE OffsetAdrs, WORD * wData);
```

```
DWORD WINAPI ct520_wPortW(DWORD hDevID, BYTE OffsetAdrs, WORD wData);
```

[Visual Basic]

```
Declare Function ct520_rPortB Lib "hict520.dll" _
```

```
(ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByRef byData As Byte) As Long
```

```
Declare Function ct520_wPortB Lib "hict520.dll" _
```

```
(ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByVal byData As Byte) As Long
```

```
Declare Function ct520_rPortW Lib "hict520.dll" _
```

```
(ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByRef wData As Integer) As Long
```

```
Declare Function ct520_wPortW Lib "hict520.dll" _
```

```
(ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByVal wData As Integer) As Long
```

《引数》

- ◆ DWORD *hDevID*・・・対象デバイスのデバイスハンドル
- ◆ BYTE *OffsetAdrs*・・・オプションポートのオフセットアドレス
- ◆ BYTE * *byData*・・・読込んだデータが格納される1バイトエリアのアドレス
- ◆ BYTE *byData*・・・オプションポートへの書込1バイトデータ
- ◆ WORD * *wData*・・・読込んだデータが格納される2バイトエリアのアドレス
- ◆ WORD *wData*・・・オプションポートへの書込2バイトデータ

《戻り値》 処理結果

0 : 成功

0以外 : 失敗・・・「6.3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C言語]

```
DWORD ret; //関数の戻り値
```

```
BYTE byData;
```

```
ret = ct520_rPortB(hDevID, //デバイスハンドル
                  0x12, //オフセットアドレス(汎用出力ポート)
                  &byData); //格納先のアドレス
```

```
ret = ct520_wPortB(hDevID, //デバイスハンドル
                  0x12, //オフセットアドレス(汎用出力ポート)
                  0x01); //書込データ
```

[Visual Basic]

```
Dim ret As Long '関数の戻り値
```

```
Dim byData As Byte
```

```
ret = ct520_rPortB(hDevID, _ 'デバイスハンドル
                  &H12, _ 'オフセットアドレス(汎用出力ポート)
                  byData) '格納先のアドレス
```

```
ret = ct520_wPortB(hDevID, _ 'デバイスハンドル
                  &H12, _ 'オフセットアドレス(汎用出力ポート)
                  &H1) '書込データ
```

(9) ct520_rXYBuf ()	X Y c h 入出力バッファの読込
ct520_rZUBuf ()	Z U c h 入出力バッファの読込
ct520_wXYBuf ()	X Y c h 入出力バッファの書込
ct520_wZUBuf ()	Z U c h 入出力バッファの書込

《機能》

デバイスハンドルで指定されたCTRボードの、

入出力バッファの読込 . . . X Y (Z U) c h の入出力バッファを読み込み、指定エリアに格納します。

入出力バッファへ書込 . . . X Y (Z U) c h の入出力バッファにデータを書込みます。

《書式》

[C言語]

```
DWORD WINAPI ct520_rXYBuf (DWORD hDevID, DWORD * data);
```

```
DWORD WINAPI ct520_rZUBuf (DWORD hDevID, DWORD * data);
```

```
DWORD WINAPI ct520_wXYBuf (DWORD hDevID, DWORD data);
```

```
DWORD WINAPI ct520_wZUBuf (DWORD hDevID, DWORD data);
```

[Visual Basic]

```
Declare Function ct520_rXYBuf Lib "hict520.DLL" (ByVal hDevID As Long, ByVal data As Long) As Long
```

```
Declare Function ct520_rZUBuf Lib "hict520.DLL" (ByVal hDevID As Long, ByVal data As Long) As Long
```

```
Declare Function ct520_wXYBuf Lib "hict520.DLL" (ByVal hDevID As Long, ByVal data As Long) As Long
```

```
Declare Function ct520_wZUBuf Lib "hict520.DLL" (ByVal hDevID As Long, ByVal data As Long) As Long
```

《引数》

- ◆ DWORD *hDevID* . . . 対象デバイスのデバイスハンドル
- ◆ DWORD **data* . . . 読込んだデータが格納されるエリアのアドレス
- ◆ DWORD *data* . . . 入出力バッファへの書込データ

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

[C言語]

```
DWORD ret; //関数の戻り値
```

```
DWORD dwData; //入出力バッファデータ
```

```
ret = ct520_rXYBuf (hDevID, //デバイスハンドル
                  &dwData); //格納先のアドレス
```

```
ret = ct520_wXYBuf (hDevID, //デバイスハンドル
                  10000); //入出力バッファデータ
```

[Visual Basic]

```
Dim ret As Long '関数の戻り値
```

```
Dim dwData As Long '入出力バッファデータ
```

```
ret = ct520_rXYBuf (hDevID, _ 'デバイスハンドル
                  dwData) '格納先のアドレス
```

```
ret = ct520_wXYBuf (hDevID, 'デバイスハンドル
                  10000) '入出力バッファデータ
```

6. 4. 2 DOS版ドライバ関数

DOS版ではC言語対応となっています。

(1) ct520_GetDeviceCount() ボード枚数の取得

《機能》

現在パソコンに装着されているCTRボードの枚数を取得します。(HPCI-CTR524F/522Fのみ)

《書式》

```
short ct520_GetDeviceCount( short* count );
```

《引数》

◆ short* count . . . CTRボードの枚数

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short count; /* CTRボードの枚数 */
```

```
short ret; /* 関数の戻り値 */
```

```
ret = ct520_GetDeviceCount( &count );
```

(2) ct520_GetDeviceInfo() デバイス情報の取得

《機能》

現在パソコンに装着されているCTRボードのデバイス情報(※1)を取得します。この結果、HPCDEVINFO型(※2)の配列にデバイス情報が格納されます。この値は、デバイスオープン時に利用します。(HPCI-CTR524F/522Fのみ)

《書式》

```
short ct520_GetDeviceInfo( short* pcnDevNo, HPCDEVINFO* p );
```

《引数》

◆ short* pcnDevNo . . . 情報を取得するボードの最大枚数が格納されたshort型エリアのアドレスを渡します。関数の呼び出し後、実際に情報を取得したボードの枚数が格納されます。

◆ HPCDEVINFO* p . . . 各ボードのデバイス情報がセットされるべきエリアのアドレス、すなわちHPCDEVINFO型の配列の先頭アドレスを渡します。

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》 パソコンにCTRボードが2枚装着されていることを想定します。

```
short ret; /* 関数の戻り値 */
```

```
short count = 2; /* 最大枚数は2 */
```

```
HPCDEVINFO p[2]; /* 2枚のCTRボードのデバイス情報がセットされるべきエリア */
```

```
ret = ct520_GetDeviceInfo( &count, /* countのアドレス */
                          &p[0] ); /* 配列の先頭アドレス */
```

※1, ※2. デバイス情報格納の構造体名及び構造体メンバは、ドライバの種類により異なる場合があります。

(3) ct520_OpenDevice() デバイスのオープン

《機能》

渡したデバイス情報を持つCTRボードをオープンし、他のCTRボードと識別するためのデバイスハンドルを取得します。以降このデバイスハンドルは、このCTRボードにアクセスするためのハンドルとなります。

《書式》

```
short ct520_OpenDevice( short* hdev, HPCDEVINFO* p );
```

《引数》

- ◆ short hdev . . . デバイスハンドル
- ◆ HPCDEVINFO* p . . . オープンするデバイスの情報がセットされたエリアのアドレス
(※ 構造体のメンバーはボード種別により異なります。)

《戻り値》 処理結果

0 : 成功
0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

パソコンにCTRボードが2枚装着されていることを想定します。
デバイス情報格納エリアとしてHCTRDEVINFO型の配列p[2]を準備します。

```
short ret; /* 関数の戻り値 */
short hdev[2]; /* デバイスハンドル取得エリア */

ret = ct520_OpenDevice(&hdev[0], &p[0]); /* 1番目のデバイス情報 */
ret = ct520_OpenDevice(&hdev[1], &p[1]); /* 2番目のデバイス情報 */
```

《デバイスハンドル》

デバイスハンドルは次のデータ内容となっています。

15-12	11- 8	7- 4	3- 0
ボードID設定値		デバイスオープンNo	
0	0~15	1~16	

(4) ct520_CloseDevice() デバイスのクローズ

《機能》

デバイスハンドルで指定されたCTRボードをクローズします。以降、このデバイスハンドルは無効となります。

《書式》

```
short ct520_CloseDevice( short hdev );
```

《引数》

- ◆ short hdev . . . クローズするボードのデバイスハンドル

《戻り値》 処理結果

0 : 成功
0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

既にデバイスハンドルとしてhdevが取得されているものとします。

```
short ret; /* 関数の戻り値 */

ret = ct520_CloseDevice( hdev );
```

(5) ct520_rXYSts() XYchステータスの読込
 ct520_rZUSts() ZUchステータスの読込

《機能》

デバイスハンドルで指定されたCTRボードの、XY(ZU)chのステータスを読込み、指定したエリアに格納します。

《書式》

```
short ct520_rXYSts( short hdev, short* sts );
short ct520_rZUSts( short hdev, short* sts );
```

《引数》

- ◆ short hdev . . . 対象デバイスのデバイスハンドル
- ◆ short* sts . . . 読込んだデータが格納されるエリアのアドレス

《戻り値》 処理結果

0 : 成功
 0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short ret;     //関数の戻り値
short sts;     //ステータス

ret = ct520_rXYSts ( hdev,         /* デバイスハンドル */
                  &sts );       /* 格納先のアドレス */
```

(6) ct520_wXYCmd() XYch制御コマンドの書込
 ct520_wZUCmd() ZUch制御コマンドの書込

《機能》

デバイスハンドルで指定されたCTRボードの、XY(ZU)chのコマンドバッファへコマンドデータを書込みます。

《書式》

```
short ct520_wXYCmd( short hdev, short cmd );
short ct520_wZUCmd( short hdev, short cmd );
```

《引数》

- ◆ short hdev . . . 対象デバイスのデバイスハンドル
- ◆ short cmd . . . コマンドデータ

《戻り値》 処理結果

0 : 成功
 0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short ret;     //関数の戻り値

ret = ct520_wXYCmd( hdev,         /* デバイスハンドル */
                  0x03 );       /* XYchカウンタクリア */
```

(7) ct520_rXYReg()	X Y c h レジスタの読込
ct520_rZUReg()	Z U c h レジスタの読込
ct520_wXYReg()	X Y c h レジスタの書込
ct520_wZUReg()	Z U c h レジスタの書込

《機能》

デバイスハンドルで指定されたCTRボードの、

レジスタの読込・・・XY(ZU)chのレジスタ読込コマンドで指定したレジスタを読み込み、指定エリアに格納します。

レジスタの書込・・・XY(ZU)chのレジスタ書込コマンドで指定したレジスタにデータを書込みます。

《書式》

```
short ct520_rXYReg( short hdev, short cmd, U_LONG* reg );
short ct520_rZUReg( short hdev, short cmd, U_LONG* reg );
short ct520_wXYReg( short hdev, short cmd, U_LONG reg );
short ct520_wZUReg( short hdev, short cmd, U_LONG reg );
```

《引数》

- ◆ short hdev ・・・ 対象デバイスのデバイスハンドル
- ◆ short cmd ・・・ レジスタ読込／書込コマンド
- ◆ U_LONG* reg ・・・ 読込んだデータが格納されるエリアのアドレス
- ◆ U_LONG reg ・・・ レジスタ書込データ

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 ・・・ 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short ret;     /* 関数の戻り値 */
U_LONG reg;    /* レジスタのデータ */

ret = ct520_rXYReg( hdev,           /* デバイスハンドル */
                  0xc0,           /* X c hのカウンタを読む */
                  &reg);         /* 格納先のアドレス */
ret = ct520_wXYReg( hdev,           /* デバイスハンドル */
                  0x80,           /* X c hのカウンタを指定 */
                  10000 );         /* 書込データ */
```

(8) ct520_rPortB()	オプションポートのバイト読込
ct520_rPortW()	オプションポートのワード (2バイト) 読込
ct520_wPortB()	オプションポートへバイト書込
ct520_wPortW()	オプションポートへワード (2バイト) 書込

《機能》

デバイスハンドルで指定されたCTRボードの、

- オプションポートの読込・・・オプションポートを読み、指定エリアに格納します。
- オプションポートへ書込・・・オプションポートに指定データを書込みます。

《書式》

```
short ct520_rPortB( short hdev, short offset, short* data );
short ct520_wPortB( short hdev, short offset, short data );
short ct520_rPortW( short hdev, short offset, short* data );
short ct520_wPortW( short hdev, short offset, short data );
```

《引数》

- ◆ short hdev・・・対象デバイスのデバイスハンドル
- ◆ short offset・・・オプションポートのオフセットアドレス
- ◆ short* data・・・読込んだデータが格納される1バイト/2バイトエリアのアドレス
- ◆ short data・・・オプションポートへの書込1バイト/2バイトデータ

《戻り値》 処理結果

- 0 : 成功
- 0以外 : 失敗・・・「6.3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short ret; /* 関数の戻り値 */
short data;

ret = ct520_rPortB(hdev, /* デバイスハンドル */
                  0x12, /* オフセットアドレス(汎用出力ポート) */
                  &data ); /* 格納先のアドレス */
ret = ct520_wPortB(hdev, /* デバイスハンドル */
                  0x12, /* オフセットアドレス(汎用出力ポート) */
                  0x01 ); /* 書込データ */
```

(9) ct520_rXYBuf ()	X Y c h 入出力バッファの読込
ct520_rZUBuf ()	Z U c h 入出力バッファの読込
ct520_wXYBuf ()	X Y c h 入出力バッファの書込
ct520_wZUBuf ()	Z U c h 入出力バッファの書込

《機能》

デバイスハンドルで指定されたCTRボードの、

入出力バッファの読込 ・ ・ X Y (Z U) c h の入出力バッファを読み、指定エリアに格納します。

入出力バッファへ書込 ・ ・ X Y (Z U) c h の入出力バッファにデータを書込みます。

《書式》

```
short ct520_rXYBuf( short hdev, U_LONG* data );
```

```
short ct520_rZUBuf( short hdev, U_LONG* data );
```

```
short ct520_wXYBuf( short hdev, U_LONG data );
```

```
short ct520_wZUBuf( short hdev, U_LONG data );
```

《引数》

- ◆ short hdev ・ ・ 対象デバイスのデバイスハンドル
- ◆ U_LONG* data ・ ・ 読込んだデータが格納されるエリアのアドレス
- ◆ U_LONG data ・ ・ 入出力バッファへの書込データ

《戻り値》 処理結果

0 : 成功

0以外 : 失敗 ・ ・ 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
short ret; /* 関数の戻り値 */
```

```
U_LONG data; /* 入出力バッファデータ */
```

```
ret = ct520_rXYBuf( hdev, /* デバイスハンドル */
                  &data ); /* 格納先のアドレス */
```

```
ret = ct520_wXYBuf( hdev, /* デバイスハンドル */
                  10000 ); /* 入出力バッファデータ */
```

(10) ct520_SetIntCall() 割込処理関数の登録/削除

《機能》

デバイスハンドルで指定されたCTRボードの“アプリケーション作成割込処理関数”を割込ベクタテーブルへの登録または割込ベクタテーブルからの削除を行います。(HPCI-CTR524F/522Fのみ)

《書式》

```
short ct520_SetIntCall( short hdev, PINTPROC module );
```

《引数》

- ◆ short hdev . . . 対象デバイスのデバイスハンドル
- ◆ PINTPROC module . . . 登録時・アプリケーション作成割込処理関数のアドレス
削除時・NULL (0)

《戻り値》 処理結果

0 : 成功
0以外 : 失敗 . . . 「6. 3 関数の戻り値」を参照して下さい。

《呼び出し例》

```
typedef void (interrupt far * PINTPROC) ();     /* HPCI 割込み処理関数 */
#define HINTPROC     interrupt far     /* HPCI 割込み処理関数 */
void HINTPROC cal_int(void);     /* board interrupt_subroutine */

/*     Interrupt Subroutine     */
void HINTPROC cal_int(void)
{
    /* 割込処理 */
}

short     ret;     /* 関数の戻り値 */

ret = ct520_SetIntCall( hdev,     /* デバイスハンドル */
                       &cal_int );     /* set int_sub */
```

《 ご注意 》

割込を使用する場合には、次の点に留意して下さい。

- (1) 初期化時の指令関数の順序
 - ① 割込以外の全てのボード初期化を実行します (割込不使用)
 - ② ct520_SetIntCall() 関数で割込処理モジュールを設定します。
 - ③ 割込使用を許可します。
- (2) 割込処理モジュール
 - ① ボード単位で処理を行います。
 - ② モジュール内ではCPUに対して”割込許可”としないで下さい。
 - ③ 割込要因はステータス読込関数で読込ます。(一括読込)
 - ④ 上記関数起動で割込要因がない場合もあります。
 - ⑤ 作成方法はサンプルプログラムを参照して下さい。
- (3) 終了時に忘れてはならないこと。
 - ① 割込不使用とします。
 - ② ct520_SetIntCall() 関数で割込処理モジュールを削除します。

(11) ct520_GetDevVerNo() バージョン番号の取得
--

《 機能 》

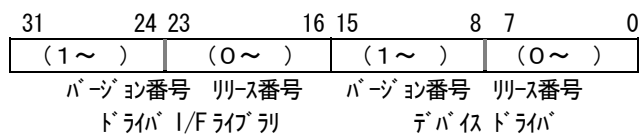
現在パソコンにインストールされているデバイスドライバと、アプリケーションにリンクしたドライバ I/F ライブラリのバージョン番号を取得します。

《 書 式 》

```
short ct520_GetDevVerNo( U_LONG* verno );
```

《 引 数 》

U_LONG* verno: ..バージョン番号が格納されます。



《 戻り値 》 .. 処理結果

0 : 成功

0以外 : 失敗 ..「6. 3 関数の戻り値」を参照して下さい。

《 呼び出し例 》

```
short    ans;
```

```
U_LONG   verno;
```

```
ans = ct520_GetDevVerNo( &verno );
```