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

株式会社 ハイバーテック
東京都江東区新大橋 1－8－11
三井生命新大橋ビル
TEL 03－3846－3801
FAX 03－3846－3773
sales@hivertec.co.jp

第 1. 30 版 2016 年 11 月 7 日発行

目　次

図　表　目　次

1. はじめに

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，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.
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，本書は，本製品を使用する開発環境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.

「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 CD

1.1 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，2 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.

1.2 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.

1.3 安全にお使い頂くために

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．

1.4 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．

1.5 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（エアーキャップなど）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.　重いものを乗せますと，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下さい.
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.

1.6 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，装置が思わ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．

1.7 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りますと，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．

1.8 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．

1.9 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
コンパレーター致信号	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/ダウンパルス」, 「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・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

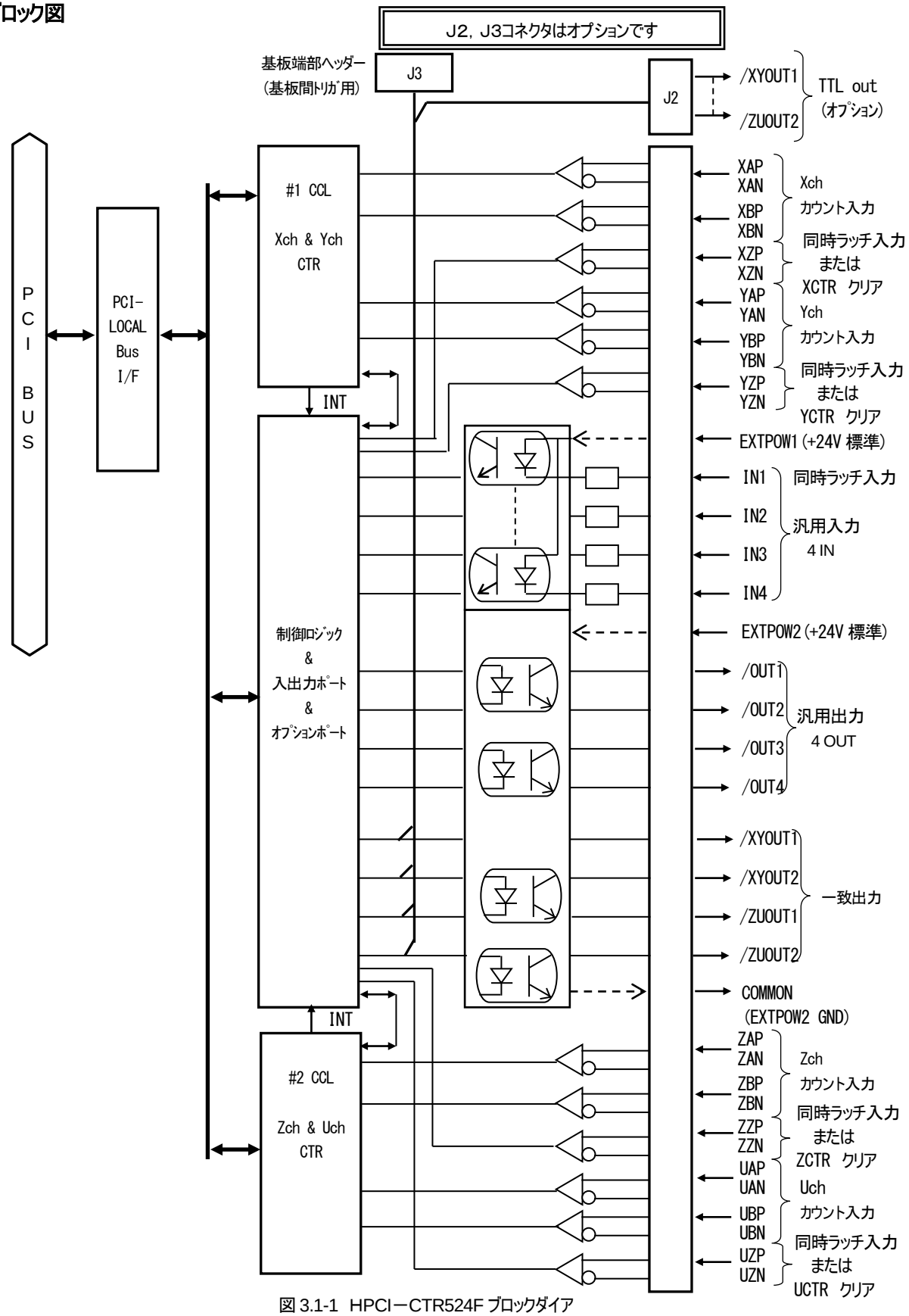


図 3.1-1 HPCIーCTR524F ブロックダイア

3.2 ポートアドレス

ポートはすべて I/O マップです．代表としてHPCI－CTR524F のポート表を下表に示します．
（このボードでは 128 ﾊﾞｲﾄ(80h)を占有しています．）
ボード型式によりポートアドレスが異なる部分がありますので，各ボードのポートアドレスは CTR ボードのユーザーズマニュアル ＜ 個別編 >を参照して下さい．

区 分		I/O アドレス (hex)		読込み(INP) 呼称	読込み(INP) 内 容		書込み（OUT） 呼称	書込み（OUT） 内 容		記事
XYch (#1CCL)		+0		STS	ステータス		CMD	コマンド		
		+2		-	不使用(予約)		-	不使用(予約)		
		+4		BUF0	入出力ﾊﾞｯﾌｧ IN (15～ 0)		BUF0	入出力ﾊﾞｯﾌｧ OUT (15～ 0)		
		+6		BUF1	入出力ﾊﾞｯﾌｧ IN (31～16)		BUF1	入出力ﾊﾞｯﾌｧ OUT (31～16)		
ZUch (#2CCL)		+8		STS	ステータス		CMD	コマンド		注 1 参照
		+a		-	不使用(予約)		-	不使用(予約)		
		+c		BUF0	入出力ﾊﾞｯﾌｧ IN (15～ 0)		BUF0	入出力ﾊﾞｯﾌｧ OUT (15～ 0)		
		+e		BUF1	入出力ﾊﾞｯﾌｧ IN (31～16)		BUF1	入出力ﾊﾞｯﾌｧ OUT (31～16)		
Z 相 及び タイマモニタ	OPTION PORT 注1	+20		ZIN TMR MONTR	Z 相入力状態読出し, ｲﾍﾞﾝﾄﾀｲﾏﾓﾆﾀ		-	不使用		個別編 3.2.4 (6)
汎用入力		+28	bit0	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	bit0	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		個別編 3.2.4 (1)
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		個別編 3.2.4 (3)
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)		個別編 3.2.4 (2)
Z 相カウンタクリア 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		個別編 3.2.4 (5)
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		個別編 3.2.4 (4)
割込ルート ポート (ポート+50,51 は バイトアクセス)		+50		INT ROUTE MSK	割込みルート MSK 状態読出し (#1CCL, #2CCL, and IN1)		INT ROUTE MSK	割込ルート MSK 設定 (#1CCL, #2CCL,IN1)		個別編 3.2.4 (7) (8)
		+51		INT ROUTE STS	#1CCL, #2CCL,IN1 割込ルート状態読出し			(+51 は読出し専用)		
PCI BUS 割込み ポート		+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		個別編 3.2.4 (9)
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

注 1. XYch, ZUch(CCL)以外のポートはオプションポートとして，ボード全体に 1 組おかれています．

3.3 ポートとレジスタ配置

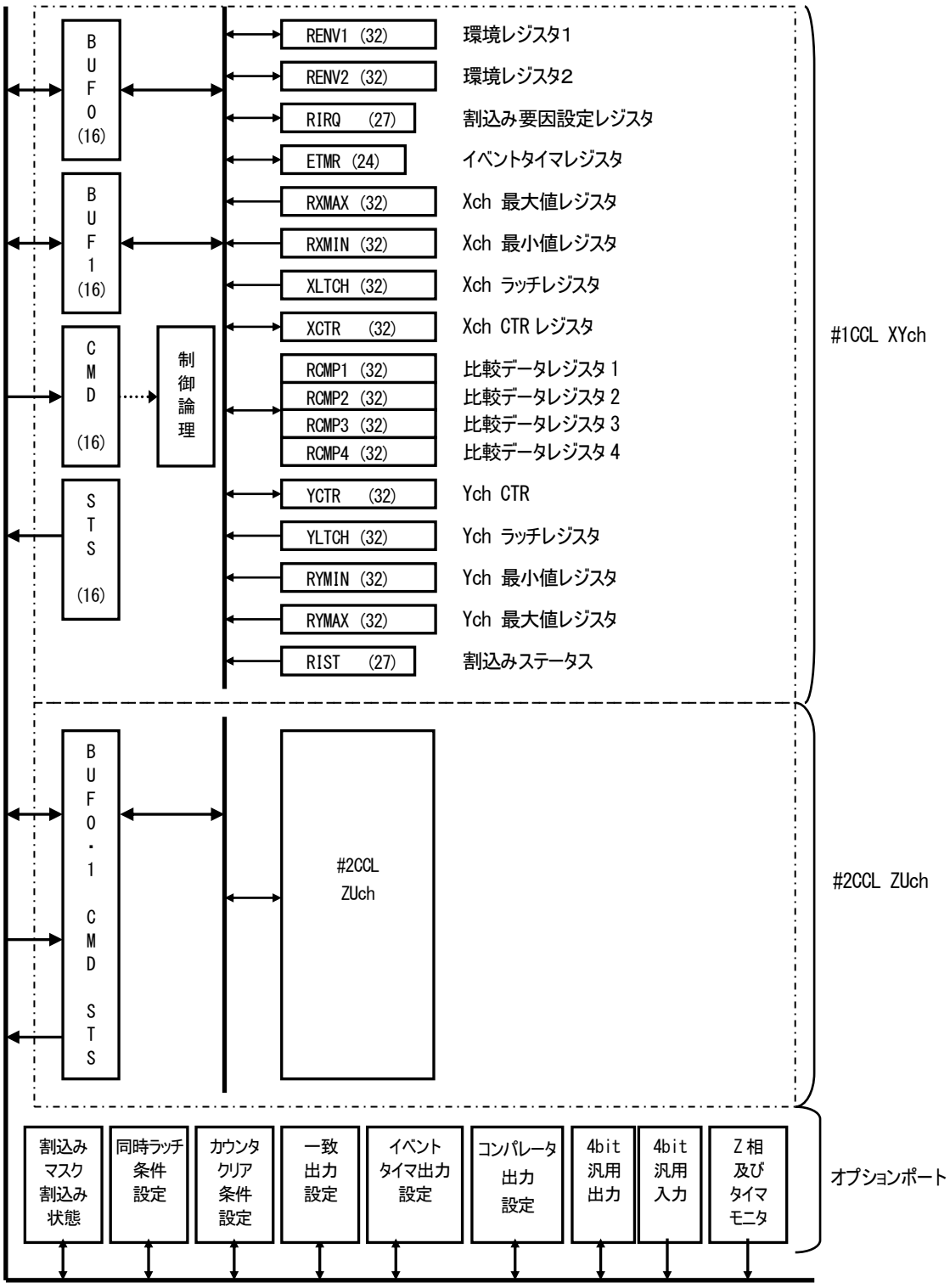


図 3.3-1 HPCI－CTR524F ポートとレジスタ配置

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 の各ポートがあります．これらの他に全体に共通に１組のオプションポートがあります．オプションポートは割込みおよび一致信号の外部出力ルート設定，比較出力など機能設定および 4Bit 汎用入出力ポートなどの機能があります．CCL に対する書込み読出しは，大別して 2 種類あります．

(1)レジスタ制御コマンド・・・#xCCL のレジスタ類およびカウンタに対してコマンドデータの読み書きをします．

書込みデータを伴う場合は CMD ポート，BUF0,1 データポート経由で指定するレジスタ，カウンタにセットします．

データの読出しは，CMD ポートに読出しコマンドを書きます．その結果，BUF0，1 に読み出されます．

(2)単独コマンド・・・・・・データを伴わないコマンド

CMD ポートに書くだけです．測定開始，CTR LTCH，CTR CLR などのコマンドが該当します．

ステータスは直接 STS ポートを読出します．

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

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 から LTCH1 にラッチする	0010h	カウント動作中の CTR 読出しは，LTCH コマンドを出し，ﾗｯﾁﾚｼﾞｽﾀ(LTCH)から読む．
6	LTCH CTR2	CTR2 から LTCH2 にラッチする	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	2 通りの測定機能があります．
10	MM1 STOP	CTR1 の 最大値最小値の測定終了	0049h	(1)カウント値の Min-Max 計測(c.f.5.8 節)
11	MM2 START	CTR2 の 最大値最小値の測定開始	004ah	(2)到来信号の edge～edge 間 時間幅計測
12	MM2 STOP	CTR2 の 最大値最小値の測定終了	004bh	(c.f.5.9 節

注:カウンタスタートは RENV1 b12，b13 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:CTR2 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	*	予約	
∫	∫		
14	*		
15	INTS	1:割込み要求中：割込み内容は「4.4.2 割込みステータス」による	RIST 読出し後，この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	CMP4II	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	CMP1 I 0	1:CMP1 が条件成立し，割込	
1	CMP1 I 1	1:CMP1 の条件成立が解けて，割込	
2	CMP2 I 0	1:CMP2 が条件成立し，割込	
3	CMP2 I 1	1:CMP2 の条件成立が解けて，割込	
4	CMP3 I 0	1:CMP3 が条件成立し，割込	
5	CMP3 I 1	1:CMP3 の条件成立が解けて，割込	
6	CMP4 I 0	1:CMP4 が条件成立し，割込	
7	CMP4 I 1	1:CMP4 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	XZ(ZZ)相端子からの CTR1 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	YZ(UZ)相端子からの CTR2 クリア
10	LTCH1I	1:CTR 1 XZ(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 XZ(ZZ)個別ラッチ
11	LTCH2I	1:CTR 2 YZ(UZ)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 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割込	外部信号による全 ch 同時ラッチ，又は同時ラッチ CMD(41h)
25	MENDI	1:幅計測終了エッジで割込（edge～edge time count 計測）	
26	ETMRINT	1:イベントタイマ周期割込み	注意：ﾀｲﾏｽﾀｰﾄ時にも割込む
27～31		不使用 all ‘0’	

表 4.4-2 割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 M0	1:CMP1 が条件成立 割込みイネーブル.	
1	CMP1 M1	1:CMP1 の条件成立が解けて 割込みイネーブル	
2	CMP2 M0	1:CMP2 が条件成立 割込みイネーブル	
3	CMP2 M1	1:CMP2 の条件成立が解けて 割込みイネーブル.	
4	CMP3 M0	1:CMP3 が条件成立 割込みイネーブル	
5	CMP3 M1	1:CMP3 の条件成立が解けて 割込みイネーブル	
6	CMP4 M0	1:CMP4 が条件成立 割込みイネーブル.	
7	CMP4 M1	1:CMP4 の条件成立が解けて 割込みイネーブル	
8	CLR1M	1:CTR1 に Z 相入力によるクリア 割込みイネーブル.	
9	CLR2M	1:CTR2 に Z 相入力によるクリア 割込みイネーブル	
10	LTCH1M	1:CTR1XZ(ZZ)信号入力による 割込みイネーブル	Z 相入力による個別ラッチ使用時の
11	LTCH2M	1:CTR2YZ(UZ)信号入力による 割込みイネーブル.	ラッチ ch 判別用
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	同時ラッチ
25	MENDM	1:幅計測終了エッジで割込みイネーブル	edge～edge time count 計測
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）

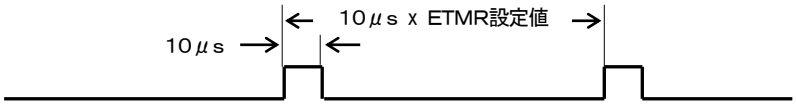
#1CCL, #2CCL 共に 4 種類の比較データレジスタ（RCMP1～RCMP4）があります．

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. フリーラニングタイマーです.
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			

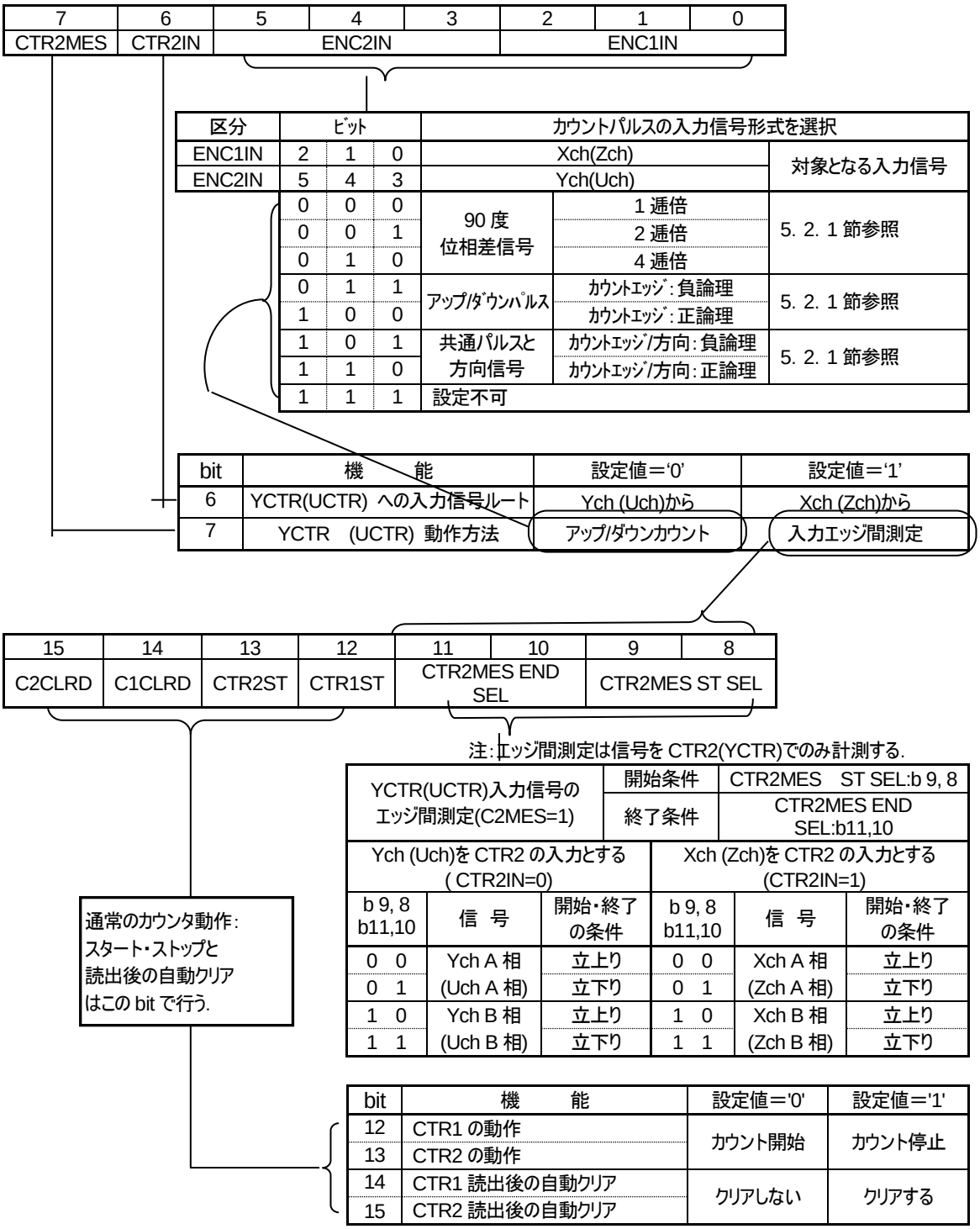
4.5.5 最大値レジスタ（MAX1, 2）, 最小値レジスタ（MIN1, 2）

#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）

#1CCL, #2CCL 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.



7	6	5	4	3	2	1	0
CTR2MES	CTR2IN	ENC2IN			ENC1IN		

区分	ﾋﾞｯﾄ			カウントパルスの入力信号形式を選択		
ENC1IN	2	1	0	Xch(Zch)		対象となる入力信号
ENC2IN	5	4	3	Ych(Uch)		
	0	0	0	90 度 位相差信号	1 逓倍	5. 2. 1 節参照
	0	0	1		2 逓倍	
	0	1	0		4 逓倍	
	0	1	1	ｱｯﾌﾟ/ﾀﾞｳﾝﾊﾟﾙｽ	ｶｳﾝﾄｴｯｼﾞ：負論理	5. 2. 1 節参照
	1	0	0		ｶｳﾝﾄｴｯｼﾞ：正論理	
	1	0	1	共通パルスと 方向信号	ｶｳﾝﾄｴｯｼﾞ/方向：負論理	5. 2. 1 節参照
	1	1	0		ｶｳﾝﾄｴｯｼﾞ/方向：正論理	
	1	1	1	設定不可		

bit	機　　能	設定値＝‘0’	設定値＝‘1’
6	YCTR(UCTR) への入力信号ルート	Ych (Uch)から	Xch (Zch)から
7	YCTR (UCTR) 動作方法	アップ/ダウンカウント	入力エッジ間測定

15	14	13	12	11	10	9	8
C2CLRD	C1CLRD	CTR2ST	CTR1ST	CTR2MES END SEL		CTR2MES ST SEL	

注：ｴｯｼﾞ間測定は信号を CTR2(YCTR)でのみ計測する.

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

Ych (Uch)を CTR2 の入力とする (CTR2IN=0)			Xch (Zch)を CTR2 の入力とする (CTR2IN=1)		
b 9, 8 b11,10	信　号	開始・終了の条件	b 9, 8 b11,10	信　号	開始・終了の条件
0　0	Ych A 相 (Uch A 相)	立上り	0　0	Xch A 相 (Zch A 相)	立上り
0　1		立下り	0　1		立下り
1　0	Ych B 相 (Uch B 相)	立上り	1　0	Xch B 相 (Zch B 相)	立上り
1　1		立下り	1　1		立下り

通常のカウンタ動作：
スタート・ストップと
読出後の自動ｸﾘｱ
はこの bit で行う.

bit	機　　能	設定値＝'0'	設定値＝'1'
12	CTR1 の動作	カウント開始	カウント停止
13	CTR2 の動作		
14	CTR1 読出後の自動ｸﾘｱ	ｸﾘｱしない	ｸﾘｱする
15	CTR2 読出後の自動ｸﾘｱ		

Z 相信号端子（XZ, YZ, ZZ, UZ）からカウンタクリアする，各 Ch毎に同時ラッチする，全 ch 同時ラッチの使い方が出来ます．
何れも一部設定を RENV1bit18～bit20 で行います．残部の設定をオプションポートの同時ラッチ設定ポートでラッチ条件を設定します．（個別編　オプションポートの「同時ラッチ設定ポート」を参照）

RENV1

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

＜Z 相によるカウンタラッチエッジ選択/カウンタクリア有効無効設定＞

bit	機　　能		設定値＝'0'	設定値＝'1'
18	Xch(Zch)Z 相信号入力によるカウンタクリア		有効	無効
19	Ych(Uch)Z 相信号入力によるカウンタクリア			
上記 2 ビットは"Z 相カウンタクリア設定オプションポート" カウンタクリア不使用時は"無効　'1"とする，使用時は下記手順で設定 ①RENV1：無効，②オプションポート設定，③RENV1：有効				
20	(Xch,～Uch)Z 相信号入力による ラッチのエッジ選択		設定値＝'0'	設定値＝'1'
	オプションポートで 選択	個別ラッチ	立上がり LTCH	立下がり LTCH
		同時ラッチ	任意(Don't Care)	

＜Z 相によるカウンタクリア信号条件＞

XZ,YZ(ZZ,UZ)クリア条件		
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）

RENV1

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	CMP1	比較カウンタ選択	XCTR (ZCTR)	YCTR (UCTR)
25	CMP2			
26	CMP3			
27	CMP4			

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	CTR1	タイマ周期でラッチ	0	1	0	CTR1	タイマ周期でラッチ＆クリア
0	1	1	CTR2		1	0	0	CTR2	
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（RENV1）

4.5.7 環境レジスタ 2（RENV2）

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

RENV2

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

＜CMPOUTx信号出力機能設定(図 5.6-1 参照)＞

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

b17,16	CMPOUT1,2	パルス幅
b19,18	CMPOUT3,4	
0　0	レベル出力	
0　1	50μs	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		

RENV2

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 4.5-3　環境レジスタ 2（RENV2）

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.

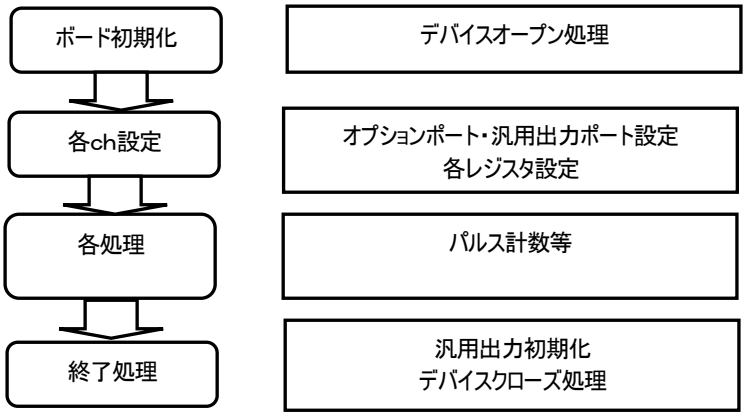


表 5.1-1 基本的な操作手順

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込み

5.2.1 カウントパルスの入力信号形式

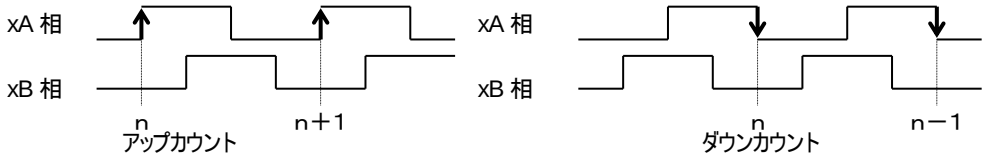
環境レジスタ 1（RENV1）のbit0～5 を設定します.

区分	ﾋﾞｯﾄ			カウントパルスの入力信号形式を選択	
ENC1IN	2	1	0	Xch (Zch)	
ENC2IN	5	4	3	Ych (Uch)	
	0	0	0	90 度 位相差信号	1 逓倍
	0	0	1		2 逓倍
	0	1	0		4 逓倍
	0	1	1	ｱｯﾌﾟ/ﾀﾞｳﾝ パルス	ｶｳﾝﾄｴｯｼﾞ：負論理
	1	0	0		ｶｳﾝﾄｴｯｼﾞ：正論理
	1	0	1	共通パルス と方向信号	ｶｳﾝﾄｴｯｼﾞ/方向：負論理
	1	1	0		ｶｳﾝﾄｴｯｼﾞ/方向：正論理
	1	1	1	設定不可	

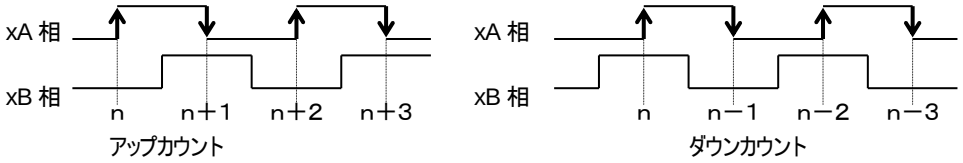
表 5.2-1 カウントパルスの入力信号形式

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.

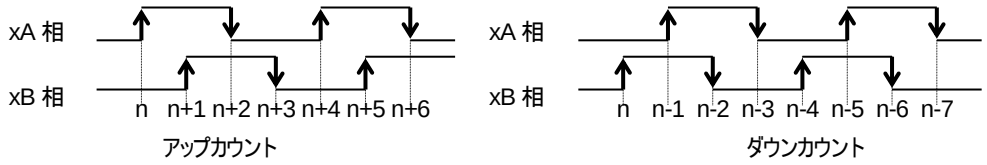
■ 90 度位相差信号 1 逓倍



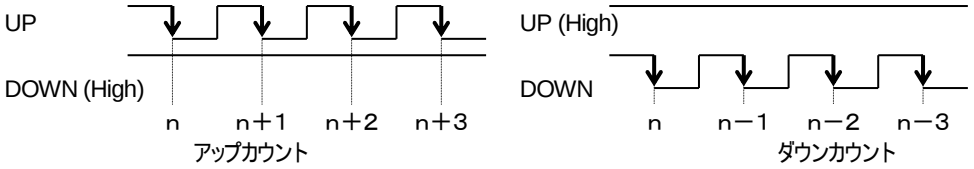
■ 90 度位相差信号 2 逓倍



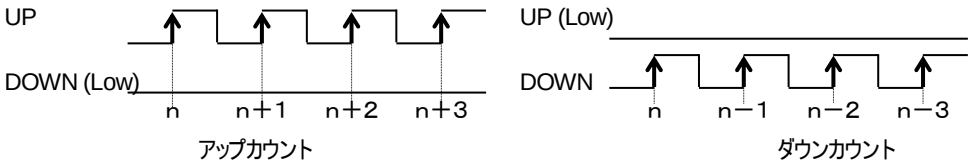
■ 90 度位相差信号 4 逓倍



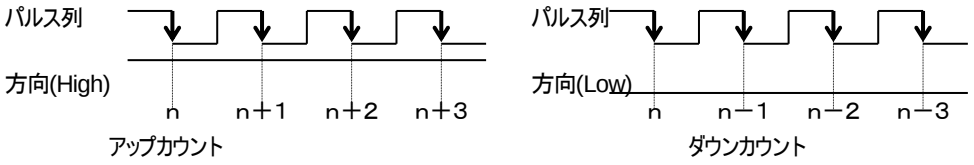
■ アップ/ダウンパルス負論理



■ 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

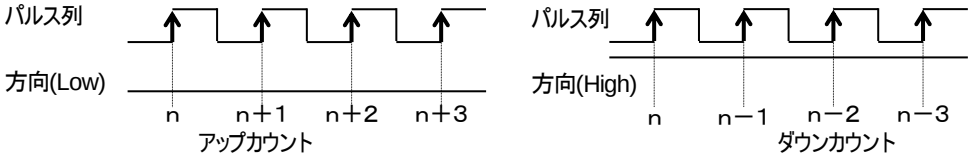


表 5.2-2 各入力信号形式におけるカウント動作のタイミング

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, CMP1 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, CMP2 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 ch の Z 相信号入力により各 ch カウンタクリアされます.
環境レジスタ(RENV1)及び,
HPCIーCTR524F, CTR522F はカウンタクリア設定ポート,
HPCーCTR224F, CTR222F, HPC104ー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

5.3 同時ラッチ

ボード内の全 ch を, 以下の方法で同時にラッチできます.

(1)各 ch 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(個別編を参照)により, 対応する Ch 毎の個別ラッチが可能となります.

(2)汎用入力 IN1 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(TMROUT)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, IN1 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

#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.

- 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)
- CMP 条件成立により CTR クリア. (イコール比較による)
- 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)
- 2 次元エリアのウィンド・コンパレート(例:XY エリアにあるとき, XYOUT1 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(STS)
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(RIST)」,「4.4.3 割込み要因設定レジスタ(RIRQ)」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, CMP1 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, CMP2 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.

- 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
- 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）
- 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

CMP1：bit24=0 でカウンタ 1(CTR1)，bit24=1 でカウンタ 2(CTR2)
CMP2：bit25=0 でカウンタ 1(CTR1)，bit25=1 でカウンタ 2(CTR2)
CMP3：bit26=0 でカウンタ 1(CTR1)，bit26=1 でカウンタ 2(CTR2)
CMP4：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				※CMP1 と CMP2 に設定する値は共通（下表）	
CMP1	3	2	1	0	CMP1 の比較方法	
CMP2	7	6	5	4	CMP2 の比較方法	
0	0	0	0	0	RCMP1 **<** [CTR1/CTR2]	
1	0	0	0	1	RCMP1 **>** [CTR1/CTR2]	
2	0	0	1	0	RCMP1 **=** [CTR1/CTR2]	カウント方向に無関係
3	0	0	1	1		カウントアップ時
4	0	1	0	0		カウントダウン時
5	0	1	0	1	RCMP2 **<** [CTR1/CTR2]	
6	0	1	1	0	RCMP2 **>** [CTR1/CTR2]	
7	0	1	1	1	RCMP2 **=** [CTR1/CTR2]	カウント方向に無関係
8	1	0	0	0		カウントアップ時
9	1	0	0	1		カウントダウン時
10	1	0	1	0	RCMP1 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP2	
11	1	0	1	1	RCMP1 **>** [CTR1/CTR2] OR [CTR1/CTR2] **>** RCMP2	
12	1	1	0	0	(RCMP1 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP2) AND (RCMP3 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP4)	
13	1	1	0	1	(RCMP1 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP2) OR (RCMP3 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP4)	
14	1	1	1	0	(RCMP1 **>** [CTR1/CTR2] OR [CTR1/CTR2] **>** RCMP2) AND (RCMP3 **>** [CTR1/CTR2] OR [CTR1/CTR2] **>** RCMP4)	
15	1	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(bit24)～CMP4(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	0	RCMP3 **<** [CTR1/CTR2]	
1	0	0	0	1	RCMP3 **>** [CTR1/CTR2]	
2	0	0	1	0	RCMP3 **=** [CTR1/CTR2]	カウント方向に無関係
3	0	0	1	1		カウントアップ時
4	0	1	0	0		カウントダウン時
5	0	1	0	1	RCMP4 **<** [CTR1/CTR2]	
6	0	1	1	0	RCMP4 **>** [CTR1/CTR2]	
7	0	1	1	1	RCMP4 **=** [CTR1/CTR2]	カウント方向に無関係
8	1	0	0	0		カウントアップ時
9	1	0	0	1		カウントダウン時
10	1	0	1	0	RCMP3 **<** [CTR1/CTR2] AND [CTR1/CTR2] **<** RCMP4	
11	1	0	1	1	RCMP3 **>** [CTR1/CTR2] OR [CTR1/CTR2] **>** RCMP4	
12	1	1	0	0	CTR2 **<** CTR1	
13	1	1	0	1	CTR2 **>** CTR1	
14	1	1	1	0	CTR2 **=** CTR1	
15	1	1	1	1	常に比較条件不成立	
※[CTR1/CTR2]の表記は RENV1 CMP1(bit24)～CMP4(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 CCL1 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 のコンパレーター致信号（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）J1 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 J1 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）J1 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 J1 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.

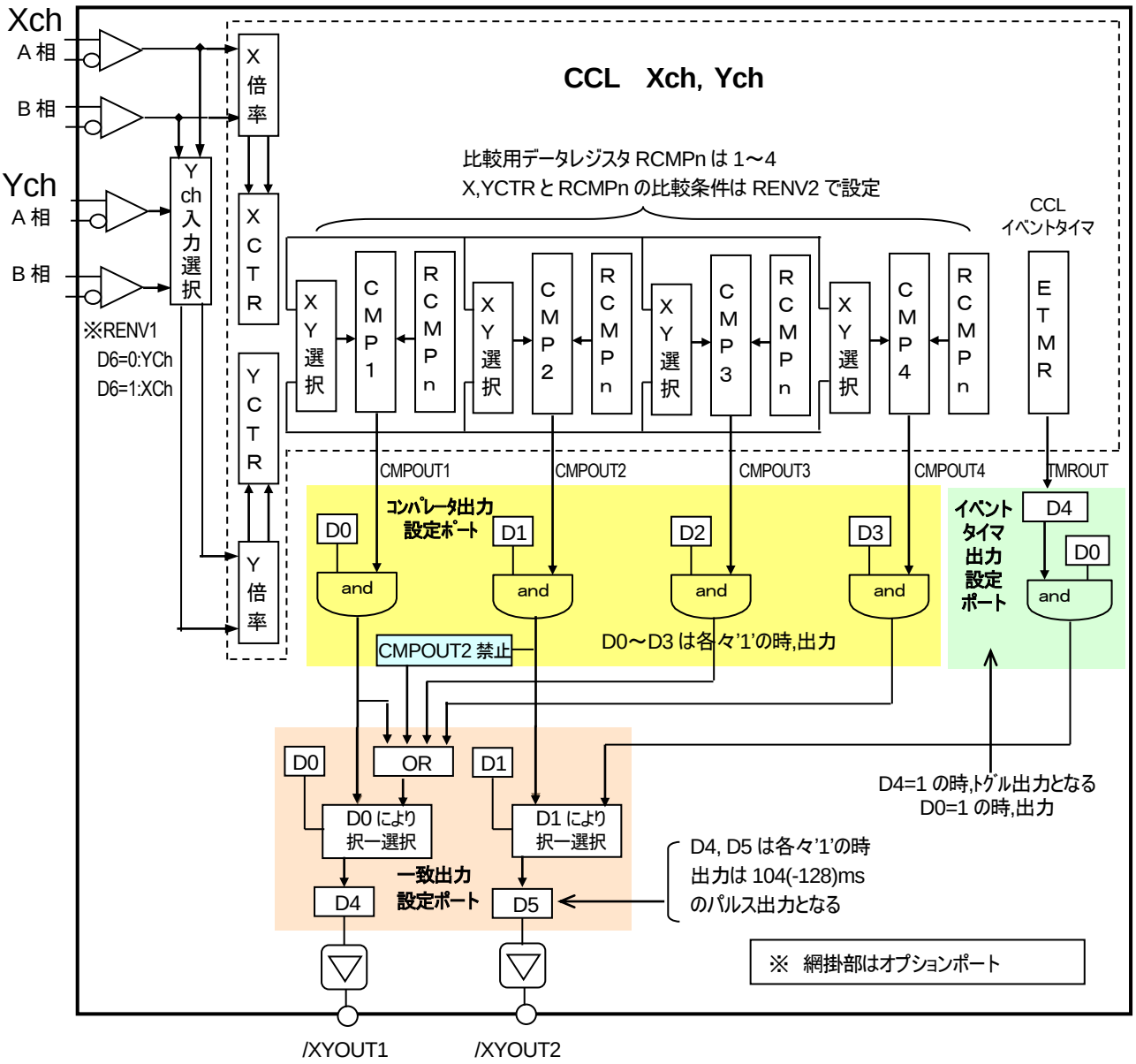


図 5.6-1 HPCI－CTR522F 一致出力ルート選択

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, CTR1, CTR2 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ト値にセットされ, 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ト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)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：CTR1 の最大値・最小値測定中, bit6=0：CTR1 の最大値・最小値測定停止中

bit7=1：CTR2 の最大値・最小値測定中, bit7=0：CTR2 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 CTR1 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 CTR2 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 CTR1 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 CTR2 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 CTR1 の最大値・最小値は現在のカウント値にセットされます.

単独コマンド INIT MM2 (0008h)書込みで CTR2 の最大値・最小値は現在のカウント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), CTR2 のラッチレジスタ(LTCH2)を 0 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 CTR2 動作停止, 最大値(MAX2), 最小値(MIN2), CTR2 のラッチレジスタ(LTCH2)は変化しません.

(1)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)CTR2 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 Ych（Uch）
環境レジスタ 1(RENV1)の bit6=1：CTR2 入力信号 Xch（Zch）

(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		
Ych (Uch)を CTR2 の入力とする (CTR2IN=0)			Xch (Zch)を CTR2 の入力とする (CTR2IN=1)		
b 9, 8 b11,10	信号	変化	b 9, 8 b11,10	信号	変化
0 0	YA 相 (UA 相)	立上り	0 0	XA 相 (ZA 相)	立上り
0 1		立下り	0 1		立下り
1 0	YB 相 (UB 相)	立上り	1 0	XB 相 (ZB 相)	立上り
1 1		立下り	1 1		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)書込みとカウントスタート処理で CTR2 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)書込みとカウントストップ処理で CTR2 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)書込みで CTR2 の最大値・最小値,ラッチデータを 0 にリセットされます.

【 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 割込み機構と割込み処理

割込み処理は 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 割込み機構

割込み要因によって割込みは次の要因で生起します.

(1)#1CCL → 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)#2CCL → 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

何れも割込み許可状態としてのことです. #1, #2CCL 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込みマスクによってボードに最終的なグループ割込みマスクが設けられています. 汎用ポートの割込みマスクはここでのみ行います.
以上の割込み機構を「 図 5.10-1　割込み機構」に示します.

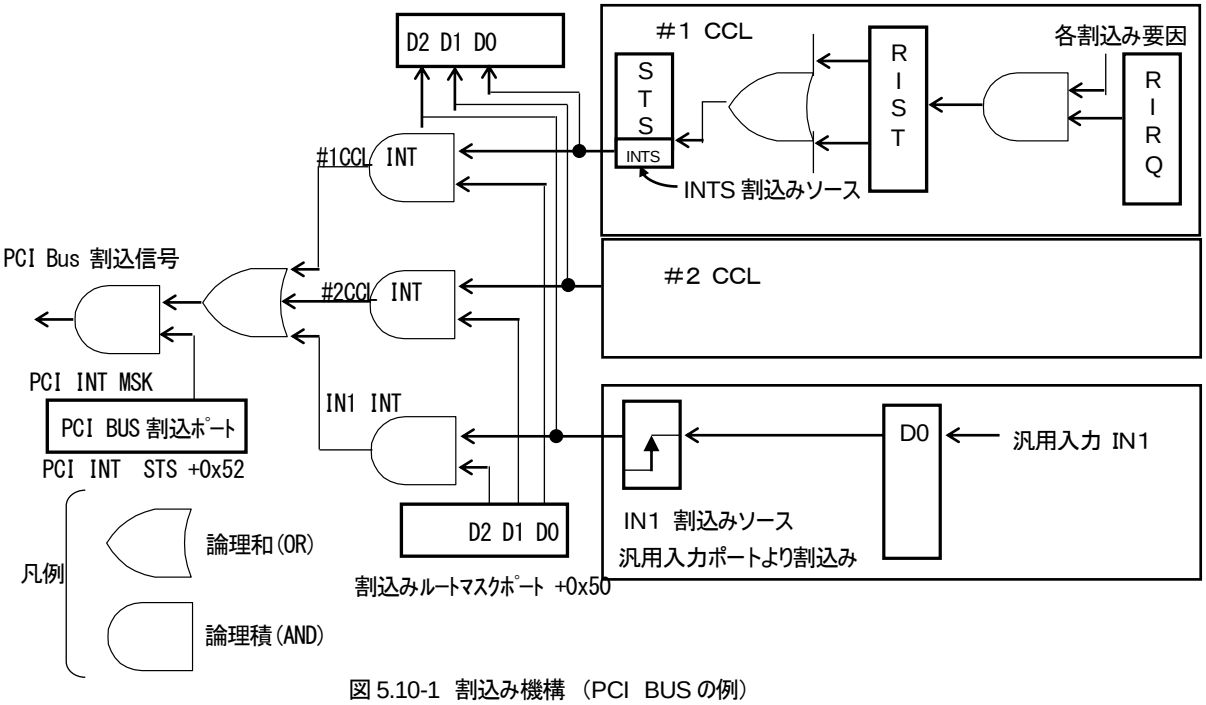


図 5.10-1 割込み機構 （PCI BUS の例）

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 割込み中のﾋﾞｯﾄが'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に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.

◇Windows 10(64 ビット)　以降 Win(x64) と記します.
◇Windows 8.1(64 ビット)　以降 Win(x64) と記します.
◇Windows 7(64 ビット)　以降 Win(x64) と記します.
◇Windows Vista (64 ビット)　以降 Win(x64) と記します.
◇Windows 10(32 ビット)　以降 Win(x86) と記します.
◇Windows 8.1(32 ビット)　以降 Win(x86) と記します.
◇Windows 7(32 ビット)　以降 Win(x86) と記します.
◇Windows Vista (32 ビット)　以降 Win(x86) と記します.
◇Windows XP　以降 WinXP と記します.
◇Windows 2000　以降 Win2K と記します.
◇Windows NT4.0　以降 WinNT と記します.
◇Windows 98SE　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					HPCI	HPC	HPC104
バス種別					PCI	ISA	PC/104
ボード					CTR524F/522F	CTR224F/222F	CTR122F
Windows 版	ファイル名	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			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
		VB.NET 用関数定義ファイル			hictr520. vb		
		VC#用関数定義ファイル			hictr520. cs		
DOS 版	ファイル名	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 hcpdtype.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

※ INTEL 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, HPCI－CTR524F/522F のファイル名, 関数名で説明します.

No	関 数 名	機　　能	HPCI-524F/522F	HPC-224F/222F	HPC104-122F
	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()	XYchステータスの読出し格納	○	○	○
	ctxxx_rZUSts()	ZUchステータスの読出し格納	○	○	○
6	ctxxx_wXYCmd()	XYch制御コマンド書込	○	○	○
	ctxxx_wZUCmd()	ZUch制御コマンド書込	○	○	○
7	ctxxx_rXYReg()	XYchレジスタの読込	○	○	○
	ctxxx_rZUReg()	ZUchレジスタの読込	○	○	○
	ctxxx_wXYReg()	XYchレジスタの書込	○	○	○
	ctxxx_wZUReg()	ZUchレジスタの書込	○	○	○
8	ctxxx_rPortB()	オプションポートのバイト読出し	○	○	○
	ctxxx_rPortW()	オプションポートのワード読出し	○	○	○
	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()	XYch入出力バッファの読出し	○	○	○
	ctxxx_rZUBuf()	ZUch入出力バッファの読出し	○	○	○
	ctxxx_wXYBuf()	XYch入出力バッファの書込	○	○	○
	ctxxx_wZUBuf()	ZUch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．

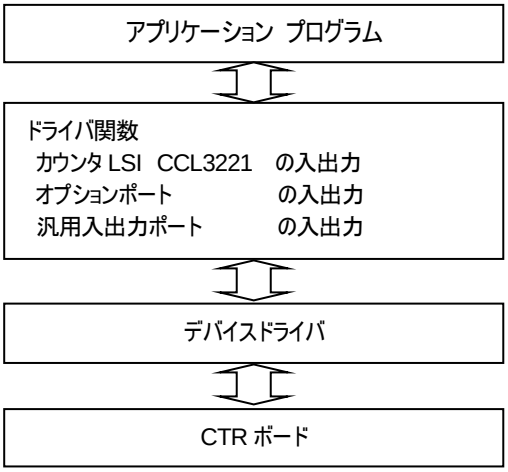


図 6.1-1 ソフトウェアの関連

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）

6.2.1 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 5.0/6.0 ］

```
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
```

［ Visual Basic .NET 2002/.NET 2003/2005/2008 ］

```
Public Structure HPCDEVICEINFO
    Dim nBusNumber As Integer          ' バス番号
    Dim nDeviceNumber As Integer       ' デバイス番号
    Dim dwIoPortAddress As Integer     ' I/Oポートアドレス
    Dim dwIrqNo As Integer             ' IRQ番号
    Dim dwNumber As Integer            ' 管理番号
    Dim dwBoardID As Integer           ' ボードID
End Structure
```

［ Visual C# ］

```
public struct HPCDEVICEINFO
{
    /// バス番号
    public uint nBusNumber;
    /// デバイス番号
    public uint nDeviceNumber;
    /// I/O ポートアドレス
    public uint dwIoPortAddress;
    /// IRQ 番号
    public uint dwIrqNo;
    /// 管理番号(Windows 9x では無視)
    public uint dwNumber;
    /// ボード ID
    public uint dwBoardID;
}
```

（注）1. 管理番号は Windows98 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;
```

6.2.2 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;
```

6.2.3 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.

```
//-------------------------------------------------
//        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
#ifdef ___cplusplus
}
#endif
```

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.

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.

```
(例)
    #define   CPLUS   1
              ・・・・・・・・・・
    #if         CPLUS
              ・・・・・・・・・・
    #endif
```

"1"で C++ 用(ファイル拡張子:cpp)
"0"で C 用(ファイル拡張子:c)

[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

・hictr520.bas ・・ドライバ I/F 用 DLL 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.

[Visual Basic .NET 2002/.NET 2003/2005/2008 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

・hictr520.vb ・・ドライバ I/F 用 DLL 関数定義ファイル
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.

[Visual C#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

・hictr520.cs ・・ドライバ I/F 用 DLL 関数定義ファイル
このファイルに外部関数宣言(DllImport 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 LIB 関数結合用ヘッダーファイル
・hcpdtype.h ・・HPCI 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 LIB 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, C#	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 度に多数のウインドウを開いた
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	Visual Basic.NET	Visual C#
データ型	Windows版	BYTE	Byte	byte	byte
		WORD	Integer	Short	Short
		DWORD	Long	Integer	Integer
	DOS 版 （16 ﾋﾞｯﾄ）	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, C#	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 CTR ボードの枚数を取得します.（HPCI-CTR524F/522F のみ）

《書 式》
[C 言語]
DWORD WINAPI ct520_GetDeviceCount (DWORD* *HpcDevNumber*);

[Visual Basic]
Declare Function ct520_GetDeviceCount Lib "hictr520.dll" (ByRef *HpcDevNumber* As Long) As Long

《引 数》
◆ DWORD* *HpcDevNumber* ・・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》
[C 言語]

```
DWORD  count;      //CTR ボードの枚数
DWORD  ret;        //関数の戻り値

  ret = ct520_GetDeviceCount( &count );
```

[Visual Basic]

```
Dim count  As Long      'CTR 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 "hictr520.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 "hictr520.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 "hictr520.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 )
```

(5)ct520_rXYSts( ) XYch ステータスを読み，格納する.
ct520_rZUSts( ) ZUch ステータスを読み，格納する.

《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(STS)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 "hictr520.dll"(ByVal hDevID As Long, ByRef sts As Integer) As Long
Declare Function ct520_ rZUSts Lib "hictr520.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
```

(6)ct520_wXYCmd( )　XYch 制御コマンドを CMD ポートに書く.
ct520_wZUCmd( )　ZUch 制御コマンドを CMD ポートに書く.

《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 "hictr520.dll" (ByVal *hDevID* As Long, ByVal *cmd* As Integer) As Long
Declare Function ct520_wZUCmd Lib "hictr520.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 );       //XYch 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 )        'XYch カウンタクリア
```

(7)ct520_rXYReg( ) XYch レジスタを読み，格納する.
ct520_rZUReg( ) ZUch レジスタを読み，格納する.
ct520_wXYReg( ) XYch レジスタに値を書込む.
ct520_wZUReg( ) ZUch レジスタに値を書込む.

《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 "hictr520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _
  ByRef reg As Long) As Long
Declare Function ct520_rZUReg Lib "hictr520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _
  ByRef reg As Long) As Long
Declare Function ct520_wXYReg Lib "hictr520.dll" (ByVal hDevID As Long, ByVal cmd As Integer, _
  ByVal reg As Long) As Long
Declare Function ct520_wZUReg Lib "hictr520.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,        //Xch 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,        //Xch 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, _          'Xch 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, _          'Xch 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出し格納.
ct520_rPortW( ) オプションポートのワード(2 ﾊﾞｲﾄ)読出し格納.
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 ﾊﾞｲﾄ)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込み，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 "hictr520.dll" _
        (ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByRef byData As Byte) As Long
Declare Function ct520_wPortB Lib "hictr520.dll" _
        (ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByVal byData As Byte) As Long
Declare Function ct520_rPortW Lib "hictr520.dll" _
        (ByVal hDevID As Long, ByVal OffsetAdrs As Byte, ByRef wData As Integer) As Long
Declare Function ct520_wPortW Lib "hictr520.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
                     0x2c,        //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
                     0x2c,        //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
                     &H2c, _           '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
                     &H2c, _           '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( )　XYch 入出力バッファの読出し格納.
ct520_rZUBuf( )　ZUch 入出力バッファの読出し格納.
ct520_wXYBuf( )　XYch 入出力バッファに書込.
ct520_wZUBuf( )　ZUch 入出力バッファに書込.

《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 ・・ XY(ZU)ch の入出力バッファを読込み，指定エリアに格納します.
入出力バッファへ書込 ・・ XY(ZU)ch 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 "hictr520.DLL" (ByVal hDevID As Long, ByRef data As Long) As Long
Declare Function ct520_rZUBuf Lib "hictr520.DLL" (ByVal hDevID As Long, ByRef data As Long) As Long
Declare Function ct520_wXYBuf Lib "hictr520.DLL" (ByVal hDevID As Long, ByRef data As Long) As Long
Declare Function ct520_wZUBuf Lib "hictr520.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 ステータスを読み，格納する.
ct520_rZUSts( )　ZUch ステータスを読み，格納する.

《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 制御コマンドを CMD ポートに書く.
ct520_wZUCmd( )　ZUch 制御コマンドを CMD ポートに書く.

《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( )　XYch レジスタを読み，格納する.
ct520_rZUReg( )　ZUch レジスタを読み，格納する.
ct520_wXYReg( )　XYch レジスタに値を書込む.
ct520_wZUReg( )　ZUch レジスタに値を書込む.

《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,          /* Xch 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,          /* Xch 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出し格納.
ct520_rPortW( ) オプションポートのワード(2 バイト)読出し格納.
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込み, 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 */
                       0x2c,          /* 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 */
                       0x2c,          /* 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( )　XYch 入出力バッファの読出し格納.
ct520_rZUBuf( )　ZUch 入出力バッファの読出し格納
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

《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 ・・ XY(ZU)ch の入出力バッファを読込み，指定エリアに格納します.
入出力バッファへ書込 ・・ XY(ZU)ch 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"アプリケーション側作成の割込処理関数"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・・アプリケーション側作成の割込処理関数のアドレス
　　削除時・・NUL（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 */
```

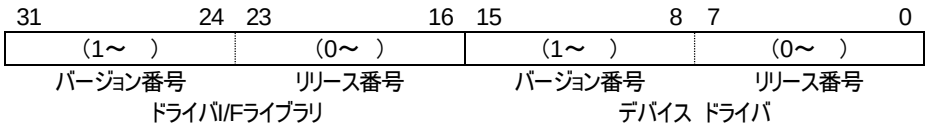
《 ご注意 》
　割込を使用する場合には、次の点に留意して下さい。
　(1) 初期化時の指令関数の順序
　　① 割込以外の全てのボード初期化を実行します（割込不使用）
　　② ct520_SetIntCall()　関数で割込処理モジュールを設定します.
　　③ 割込使用を許可します.
　(2) 割込処理モジュール
　　① ボード単位で処理を行います。
　　② モジュール内では CPU に対して"割込許可"としないで下さい.
　　③ 割込要因はステータス読込関数で読込ます.（一括読込）
　　④ 上記関数起動で割込要因がない場合もあります・
　　⑤ 作成方法はサンプルプログラムを参照して下さい・
　(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 );
```