

第4章．ソフトウェア

《注1》 W I N D O W S 9 8 / M E 対応： 特に断りのない限りW I N D O W S 9 5 用ソフトがそのまま使用できます。

《注2》 W I N D O W S X P 対応： 特に断りのない限りW I N D O W S 2 0 0 0 用のソフトがそのまま使用できます。

4-1. (ソフトウェアの)インストール

製品添付のソフトウェアは3.5インチ(1.44MB)フロッピーまたはCDに圧縮された形で格納されており、インストーラの実行により展開されます。

なお内容については充実・改良の目的で後日、追加・変更も有り得ます。

重要な変更については同メディア内のドキュメントファイルに記すこととします。

操作手順

インストール元：Dドライブ(CDROM)
インストール先：Cドライブ(HDD) の場合で例示。

(1) W I N D O W S 付属のエクスプローラで、
D:¥INSTALL¥PCI¥DIO¥DIO270 を開く。

(2) “ S e t u p . E X E ” を実行(ダブルクリック)する。

当操作以下によりDIO-270PCI関連プログラムが図4-1に示すロケーションに展開・インストールされます。

(2005年4月より以前のCDROMを使用する場合：DOS窓利用)

操作手順

インストール元：Dドライブ(CDROM)
インストール先：Cドライブ(HDD) の場合で例示。

(はスペース)

```
C:¥WINDOVS>CD¥:【ENTER】
C:¥>CD D:¥INSTALL¥PCI¥DIO¥DIO270【ENTER】
C:¥>D:INSTALL D: C:【ENTER】
```

各プログラムグループ(C, BASIC等)ごとにインストール実行の有無を問うてきますから、【Y】=yes, 【N】=no, で答えるだけで作業が進みます。

《注》 MS-DOSの環境変数“COMSPEC”が設定されていないか、または正常に設定されていないと本インストール・プログラムの作業が途中で停止してしまいます。 実行前に確認または設定しておきます。

= 設定例 = COMMAND.COMがCドライブの¥にある場合、
>SET COMSPEC=C:¥COMMAND.COM【ENTER】

【注】 W I N D O W S 9 x ・ M E ・ N T 用の汎用I/O実行DLLとデバイスドライバ(本機専用ではない!)は当作業ではインストールされません。 W I N D O W S 9 x ・ M E 用はWin9xフォルダにありますので、各ファイルを適合フォルダにコピーする必要があります。 またNT用はWinNTフォルダ中にあるので、同フォルダ中の専用インストーラで導入してください。(1-5項参照)

(追伸) CDROMの場合、Win9xおよびWinNTフォルダはINSTALLフォルダ下のDriverフォルダ下にあります。

図4-1. インストール後のディレクトリ

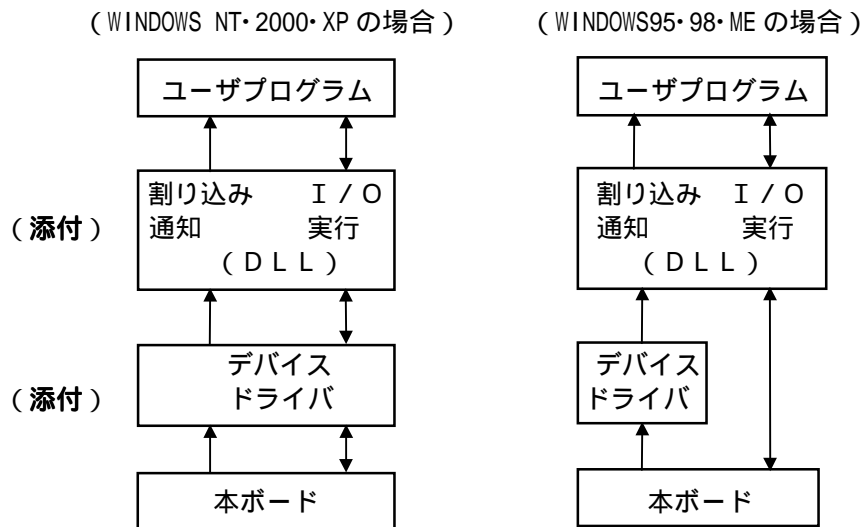
本図は原形です。 充実・改良の目的で後日、追加・変更も有り得ます。
T-C: TURBO-C、B-C: BORLAND-C

¥			
MSCIENCE			
	UTILITY (本ボードの設定)	WIN95	CF9050DP.COM: コンフィギュレーション(95用)
		-WINNT	CF9050NT.EXE: コンフィギュレーション(NT用)
			-CF9050NT.sys: 上記NT用デバイスドライバ
			-REGSTDV.EXE: デバイスドライバ登録・削除用
- - BOARDTST - -			270QB1.EXE: 本ボードの試運転・動作確認用プログラム
			270QB1.COM: 英語モードに切り替えた後、EXEを実行する
- - SMP270C - -			MICROSFT.H: MS-C用ヘッダファイル
(各種Cサンプル)			BORLAND.H: TURBO-C, BORLAND用ヘッダファイル
			INT270.C: クロック割り込み時にデジタル入出力動作
			DIV270.C: クロックをポーリングしてデジタル入出力動作
			PST270.C: 単発プリセット動作
	SMP270B		270QB1.BAS: Quick-Basic(4.5)用サンプル
	SMP270VB (Visual)		SMP270.VBP: プロジェクト
	(BASIC用)		DRIVER.FRM: ドライバ関連フォーム
	(サンプル)		SMP270.FRM: メインフォーム
			DRVDLL.BAS: 汎用IO制御ドライバ定義
			MS_PCI.BAS: PCIリソース取得定義
			DIO270.BAS: ハードウェア定義
- HND270C (DOS用ハンドラ)		INCLUDE	SMPEXT.H: ハンドラ用・共通ヘッダファイル
		- LIB	270TS.LIB: T-C、B-C用スモールモデル
			270TL.LIB: T-C、B-C用ラージモデル
			270MS.LIB: MS-C用スモールモデル
			270ML.LIB: MS-C用ラージモデル
			SMP270 - SAMPLE.C
	Hnd_95 (WINDOWS 9x・ME用ハンドラ)	Dio270	D11: ハンドラDLL
			- Vxd: デバイスドライバ
			- Vb5: Visual-Basic(5.0)用サンプル
			- Vc5: Visual-C(5.0)用サンプル
			- Bc5: Borland-C(5.0)用サンプル
			- Vc5_cpp: VC++(5.0)用サンプル
			- Delphi3: Delphi(3.0)用サンプル等
	Hnd_NT (WINDOWS NT4用ハンドラ)	Dio270	D270reg: デバイスドライバ設定ユーティリティ
			- D11: ハンドラDLL
			- Sys: デバイスドライバ
			各サンプルは上記WINDOWS 9x・MEハンドラ同様
	(WINDOWS 2000/XP用ハンドラ)		各サンプルは上記WINDOWS 9x・MEハンドラ同様
	Hnd_2K (WINDOWS 2000/XP用ハンドラ)	Dio270	
	GL_DOS	MSPCID.H	: (DOS版) リソース取得ライブラリ・ヘッダ
		- MSPCIDM.LIB	: (MS-C用) ライブラリ/全モデル対応
		- MSPCIDT.LIB	: (T-C, B-C用) ライブラリ/全モデル対応
	GL_W32	MS_PCI.H	: (Win32版) リソース取得ライブラリ・ヘッダ
		- MS_PCI.DLL	: リソース取得ライブラリDLL
			(Win9x・ME・NT兼用、要デバイスドライバ)
		- MS_PCI.LIB	: DLLインポートライブラリ
		- MS_PCI.BAS	: (VB/32bit版用) DLL関数定義モジュール

4-2. WINDOWSドライバについて

WINDOWS 9x / ME / NT / 2000 / XP 向に汎用 I / O 読み書き用 DLL が添付されています。

基本的には当 DLL (およびデバイスドライバ) を使用して本ボード上の各レジスタを読み書きすることでプログラミングが可能です。これらは御自身で (ボードにアクセスする部分の) ライブラリ等を製作する場合への便宜です。添付の**本ボード専用WINDOWSドライバ/関数ライブラリ** (第6章) を利用する場合は不要です。



WINDOWS 3.1 : Win31フォルダ以下に格納されており、VB (2.0) で利用できます。C, C++ の場合は当 DLL を使用せずともインラインアセンブラで直接 I / O 命令を記述できます。割り込みを使用するときは、DOS 同様に直接制御で対応できます。

WINDOWS 9x : Win9xフォルダ以下に格納されており、VB (4.0/5.0) で利用 (or ME) できます。ブロック I / O 命令もサポートされています。C++、C の場合は当 DLL を使用せずともインラインアセンブラで直接 I / O を記述できます。割り込みは DOS 同様に直接制御、またはデバイスドライバ (Pta95_0.vxd) で対応します。

WINDOWS NT : WinNTフォルダ以下に格納されており、VB (5.0) で利用 (4.0) できます。ブロック I / O 命令、割り込みもサポートされています。

NTではI / O制御、割り込み、共に必ずデバイスドライバが必要です。

本デバイスドライバは最大16枚のボードを (各単独に) 制御することができます。 / 当社製品でなくても可能 /

WINDOWS 2000 : Win2Kフォルダ以下に格納されており、VB (6.0) で利用 (or XP) できます。なおドライバ (WDM) は複数種類のボードで共用利用できるもので、第6章で説明する本ボード専用ハンドラ関数 DLL から利用します。 (ボードインストール作業時にインストールされます。)

【注】WDMドライバの性格から割り込みは使用できません。
詳細は¥MSCIENCE¥WIN2K¥DOC フォルダ内のテキスト参照。

4-3. ボードアクセス関連ライブラリ (WINDOWS2000/XP 以降のWDMでは不要)

汎用リソース情報取得関数 `MS_PCI.DLL / WINDOWS9x・ME・NT 用` について

ユーザプログラムから本PCIボードにアクセス・制御するときはボードを検出し、リソース情報（ベースアドレス値、割り込み番号等）を取得する必要があります。第6章に記す本機の専用ハンドラ（専用ドライバ&関数DLL）を使用する場合は内部で処理されているため、必要ありませんが、ユーザプログラムから本ボードを直接制御するときは本DLL&ドライバが必要です。使用手順は、

【1】PCIバス上のボードの検出

Int GetPciDevice (WORD VendeID, WORD DeviceID, WORD nNum, WORD Flag, WORD *magic)	
引数	VendeID: ベンダID、nNum: 検出対象ボード(0~) / 同ボード複数に対処・特定、DeviceID: デバイスID、Flag: 0 (固定)、*magic:マジック番号取得先ポインタ
戻り値	0:成功、-1:失敗

PCIボードの特定はロケーション（バス・デバイス・ファンクション）で行います。
本ボードのベンダID、デバイスID、検出対象ボード番号（1枚目=0）を指定して実行すると同ボードのロケーションがmagicに得られます。
同一ボードを複数インストールしているときは続いて検出対象ボード番号=1, 2, として実行すれば各ボードごとのロケーションが得られます。

ベンダID = 13FDH (マイクロサイエンス社PCIボード共通)、
デバイスID = 106H (DIO-270PCI)。

【2】指定ボード・指定レジスタのダブルワード読み込み（ベースアドレス値取得に使用できる。）

Int ReadPciDword (WORD magic, WORD reg, DWORD *data)	
引数	magic: 【1】GetPciDeviceで得られたマジック番号、*data:データ取得先ポインタ、reg: PCIコンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、-1:失敗

【3】指定ボード・指定レジスタのワード読み込み（通常は不使用。）

Int ReadPciWord (WORD magic, WORD reg, WORD *data)	
引数	magic: 【1】GetPciDeviceで得られたマジック番号、*data:データ取得先ポインタ、reg: PCIコンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、-1:失敗

PCIリソース情報の取得は【1】で検出したロケーション（magic）とレジスタ番号を指定して行います。物理ベースアドレス値はダブルワードなので【2】の関数を使用します。
各レジスタは本ボード上のPCIインターフェース素子9050（PLX社製）内にあり、次のように割り付けられています。

レジスタ番号10H : PCIインターフェース素子9050で使用。
14H : 未使用
18H : I/OマップレジスタのベースアドレスBASE1(3-1項)。
1CH : 未使用
20H : 未使用
24H : 未使用

なお、

I/Oマップでは得られた data の下位 2 bit をマスクした値がベースアドレス値です。

bit31	bit 2 1 0
物理ベースアドレス	× 1

【4】指定ボード・指定レジスタのバイト読み込み（割り込み番号値取得に使用できる。）

Int ReadPciByte (WORD magic, WORD reg, BYTE *data)	
引数	magic: 【1】GetPciDevice で得られたマジック番号、 *data: データ取得先ポインタ、 reg: PCI コンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0: 成功、 -1: 失敗

割り込みリソース情報取得も【1】で検出したロケーション (magic) とレジスタ番号を指定して行います。 data 値はバイトなので【4】の関数を使用します。

本レジスタも本ボード上のPCIインターフェース素子9050 (PLX社製)内にあり、レジスタ番号 = 0EHです。

得られる data 値 (1 ~ 15) は割り込み番号です。 / 不使用時 = 0、または255 / プログラム内ではベクタに変換して御使用ください。

なお、

本ボードの標準設定では (プラグアンドプレイ実行時に) 割り込みリソースを要求しません。割り込みを使用するときは1 - 5項に記す手順で設定を変更してください。

汎用リソース情報取得関数ライブラリ / MS-DOS 版 について

関数一覧

【1】PCIバス上のボードの検出

Int_far GetPciDevice(WORD VendelD,WORD DeviceID,WORD nNum,WORD Flag,WORD _far *magic)	
引数	VendelD : ベンダ ID、 nNum : 検出対象ボード (0 ~) / 同ボード複数に対処・特定、 DeviceID : デバイス ID、 Flag : 0 (固定)、 *magic :マジック番号取得先ポインタ
戻り値	0:成功、 -1 : 失敗

【2】指定ボード・指定レジスタのダブルワード読み込み (I / O アドレス値取得に使用できる。)

Int ReadPciDword (WORD magic , WORD reg , DWORD _far *data)	
引数	magic : 【1】GetPciDevice で得られた マジック番号、 *data : データ取得先ポインタ、 reg : PCI コンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、 -1 : 失敗

【3】指定ボード・指定レジスタのワード読み込み (通常は不使用。)

Int ReadPciWord (WORD magic , WORD reg , WORD _far *data)	
引数	magic : 【1】GetPciDevice で得られた マジック番号、 *data : データ取得先ポインタ、 reg : PCI コンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、 -1 : 失敗

【4】指定ボード・指定レジスタのバイト読み込み (割り込みレベル値取得に使用できる。)

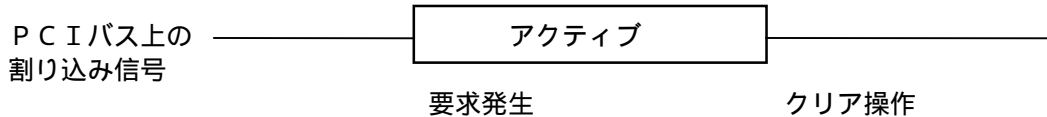
Int ReadPciByte (WORD magic , WORD reg , BYTE _far *data)	
引数	magic : 【1】GetPciDevice で得られた マジック番号、 *data : データ取得先ポインタ、 reg : PCI コンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、 -1 : 失敗

使用方法 : 前記W I N D O W S 版と同様です。

4-4. 割り込みについて

PCIバス上の割り込み信号は、これを検知したソフトウェアからクリア操作を行うまでアクティブ状態を（要求元側が）維持する“**レベル動作**”です。この仕組みでは複数のデバイスが1本の割り込みリソースを共有することもできます。

図4-4.



要注意

当社製を始め、多くのISAバスボードの割り込みは要求元がパルス状の単発信号を発信する“**エッジ動作**”ですから割り込み要求のアクティブ状態は自動解消されるのですが、PCIバス上の“**レベル動作**”ではプログラム開発中などの事情で適切なクリア操作が行われなかった場合のハングアップ等、非常事態解消のためのハードウェアリセット（電源OFF）を余儀なくされることも考えられます。このような場合はハードディスクのクラッシュ等の大きな損害が発生する恐れがあります。

添付のWINDOWS 95・98・NT用デバイスドライバは汎用のため、アプリケーション側からクリア操作に必要なパラメータを（あらかじめ）受け取っておき、割り込みが発生したらクリア操作を実行します。またアプリケーション側からは割り込み発生（回数：Read Clear）を読み取る関数DLLをポーリングする形をサポートしていますが、このようなアルゴリズムは（割り込みを使用せず）ボードのステータスをポーリングする方法と等価ですから、無用なトラブルを回避するためにも後者をお勧めします。

なお、本ボード上のROMに書き込まれているデフォルト（初期）のコンフィギュレーション情報では（プラグアンドプレイの動作時に）割り込みリソースを要求しません。もし要求したときに空きが無く拒否されるとI/Oアドレスの割り当ても受けられず認識不能状態になる恐れがあるからです。割り込みを利用するときはリソースに空きがあることを確認してから添付のコンフィギュレーション・ユーティリティで（割り込みリソースを要求するように）修正してください。【1-5項.参照】

（追伸） 一部のパソコンは標準状態で割り込みリソースに空きが無いものがあります。

割り込み制御の手順

割り込み制御ポート【3-7項】でクロック出力またはトリガ発生による割り込み許可ビット（B7 or B6）をセットします。これだけの場合、有効な信号エッジが発生しても《PCIバス上への割り込み要求信号出力》はアクティブになりません。

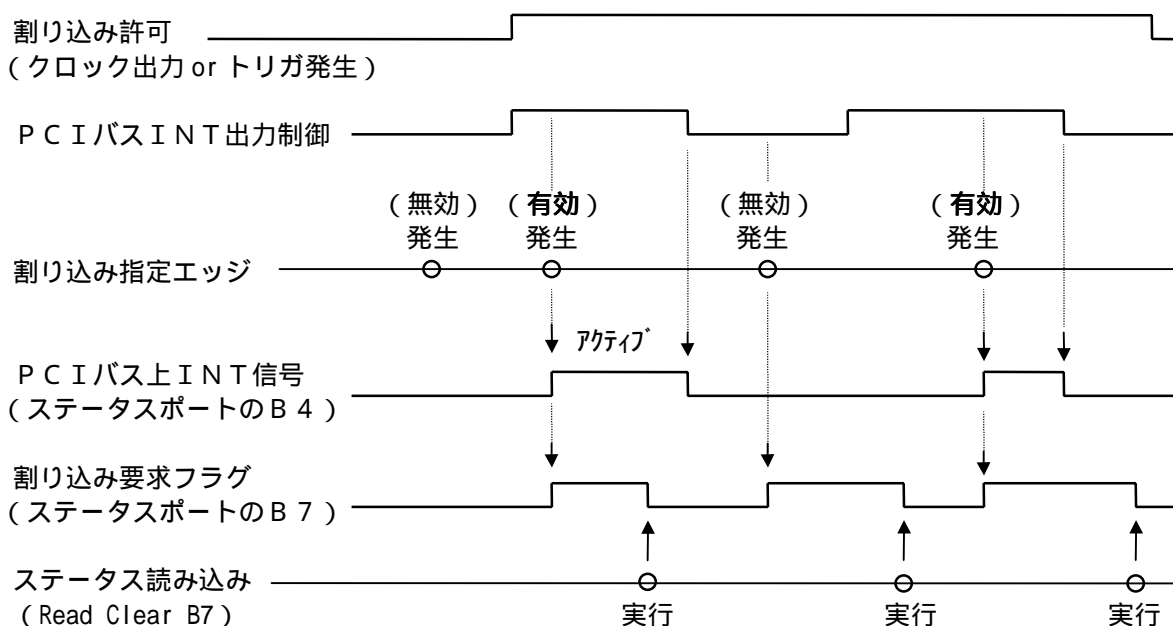
ステータスポート【3-8項】の割り込み要求フラグ（B7）がセットされるだけです。さらにPCIバスINT出力制御ビットB0をセットすることによって実際の割り込み要求信号が出力（アクティブ）可能になります。

指定エッジが発生し、割り込み要求が発生しました。

割り込み要求が受け付けられた場合、通常はデバイスドライバ内でこれをクリアします。操作は割り込み制御ポート【3-7項】のPCIバスINT信号制御ビット（B0）をクリアします。但し同ビットを再びセットするまではクリア状態で、次の割り込み要求が発生できない状態なので通常は続けて再セットします。この様子はステータスポートのPCIバス上INT信号ビット（B4）に反映されますが、通常はデバイスドライバ内ですぐクリアされるためユーザプログラムからの検出は困難です。（必要もないでしょう。）

PCIバス上のINT信号出力はボード・リセット【3-2項】でもクリアできます。

図4-4B. 割り込み制御・ステータス変化タイミング



上記タイミング図では各制御ビットの効果を示すため、割り込み要求信号の有効エッジが必要以上に発生した場合で記してあります。

ポーリングによるイベント制御 (割り込み不使用)

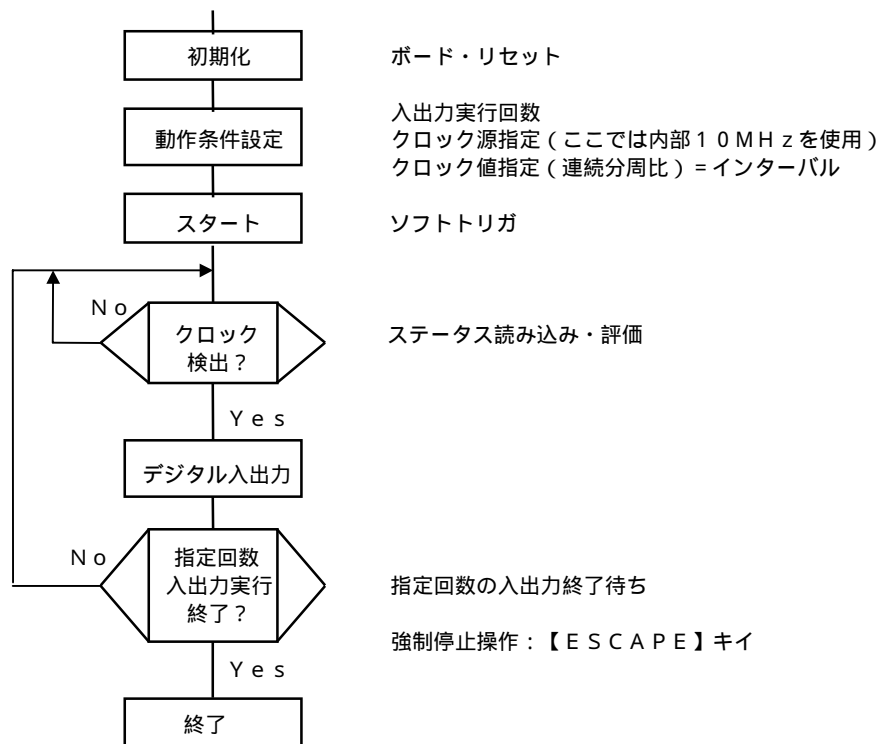
前で説明したように割り込み制御ポートのクロック割り込み許可ビットB7をセットしておき、PCIバス上のINT出力制御ビットB0がクリア状態ならクロック出力の有効エッジが発生する毎に（割り込みは発生しませんが、）ステータスポートの割り込み要求フラグがセットされます。これをポーリングしてイベントの発生を検出する方法があります。

(推奨します。)

4-5. Quick - Basicのサンプル

Quick - Basic (4.5)用のサンプルプログラム“270QB1.BAS”は基本的なBASIC文のみによる使用例です。なお本プログラムの実行形式“270QB1.EXE”は試運転・動作確認用にもなります。コーディングの詳細は同ソースのリストを御覧ください。

図4-5. “270QB1.BAS”のフロ-概要



4-6 . Cのサンプル (1 6 b i tコード専用)

指定クロック (時間々隔) のポーリングによるデジタル入出力、 同割り込みによる入出力例です。 以下にアルゴリズムの概要を記します。 具体的にはソースを御参照ください。

図 4 - 6 A . クロックのポーリングによる入出力【D I V 2 7 0 . C】

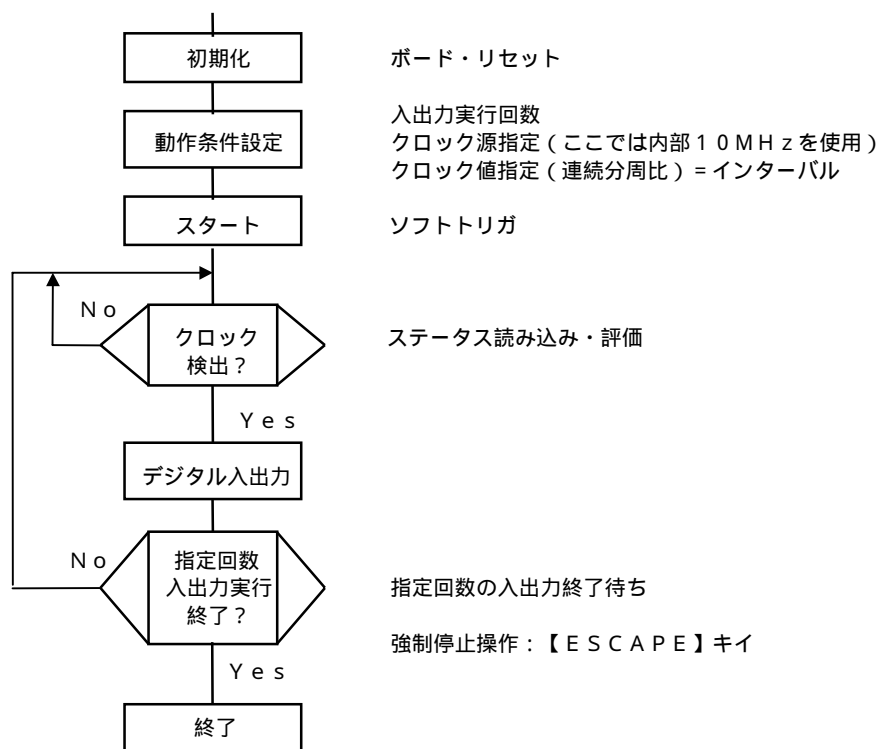
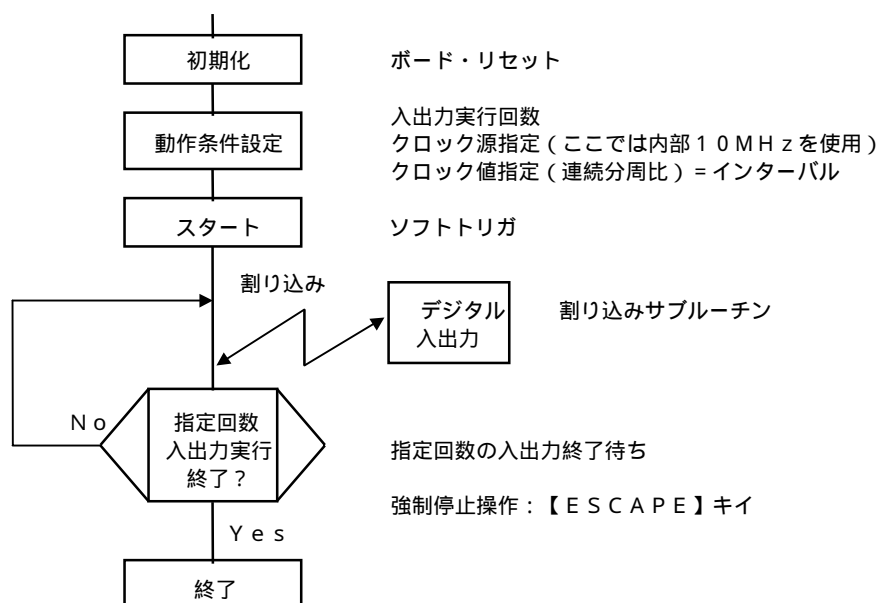


図 4 - 6 B . クロック割り込みによる入出力【I N T 2 7 0 . C】



4-7. Visual Basic (32BIT 版) のサンプル

WINDOWS 9x・MEまたはNT上で本ボードの機能を一通り動作させてみるものです。

Visual Basic (4.0) で作成されており、VB (5.0/6.0) でも動作します。

当サンプルは当社提供のWINDOWS 9x・ME・NTの汎用I/Oドライバ/DLLを使用したものです。【1-5項参照】 第6章に記す専用ドライバ/関数DLL (推奨) にもVBサンプルがありますが、両者は無関係です。なお本サンプルプログラム中の制御には割り込みを使用していません。割り込みリソースがあり、ドライバに登録した場合は外部割り込み入力要因を選択して (単に動作確認のため) 発生回数を表示するだけです。

またWINDOWS 9x・MEの場合で割り込みを使用しないときはドライバのインストールが不要です。(9x・MEでのI/O操作はDLLが直接ハードウェアにアクセスするため)

【注1】 本ボードの (プラグアンドプレイに対する) コンフィギュレーション初期設定は割り込みリソースを要求しません。割り込みを使用するためには添付のユーティリティを利用して設定変更を行う必要があります。(1-5項参照)

【注2】 当サンプルで使用するWINDOWS版I/O実行DLL/デバイスドライバは4-1項の作業だけではインストールされません。WINDOWS 9x・ME用はWin9xフォルダにありますので各ファイルを適合フォルダにコピーする必要があります。NT4.0用はWinNTフォルダ中にあり、同フォルダ中の専用インストーラで導入してください。(1-5項参照)

(追伸) CDROMの場合、Win9xおよびWinNTフォルダはINSTALLフォルダ下のDriversフォルダ下にあります。

表 4 - 7 .

ソフトウェア要素	OS	使用するモジュール/ファイル
デバイスドライバ、および インタフェースDLL	9x (ME)	pta95_0.vxd accs_95.dll
	NT (4.0)	NtPta_?sys (? : 0 ~ 15 任意の整数) port_nt.dll
Visual Basic サンプルプログラム モジュール構成	9x ME NT 共通	smp270.vbp (プロジェクト) smp270.frm (メインフォーム) driver.frm (ドライバ関連フォーム) drvdll.bas (DLL定義) ms_pci.bas (PCIリソース取得定義) dio270.bas (ハードウェア定義)

本サンプルプログラムソースはWINDOWS 9x・ME・NT共通ですが、使用するDLLが異なるのでDLL関数定義用の標準モジュール (drvdll.bas) 先頭で、

```
#Const DRIVER = "Accs_95"      ' 9x・MEの場合 (デフォルト)
' #Const DRIVER = "Port_Nt"    ' NTの場合
```

の定義により条件コンパイルで対応し、WINDOWS 9x・NTでは関数名が異なるOPEN/CLOSEをAlias機能を使用してプログラム本文中では同一名で扱えるようにしています。

さらにOPEN操作はWINDOWS 9x・ME・NTの各DLLではパラメータが異なりますから、frmMainのLOADイベント中で対応したパラメータをcboDrvParamにセット、その値を元にOPEN関数のパラメータとしています。

《操作方法》 操作手順は ドライバのオープン、 以後はボードの各機能実行の順です。
 終了時は必ずドライバのクローズ操作を行います。
 テキストボックスに記入する値は全てH E X表記です。
 (例) 8 B I T データ : F 8 H なら 2 文字 “ F 8 ” と入力します。

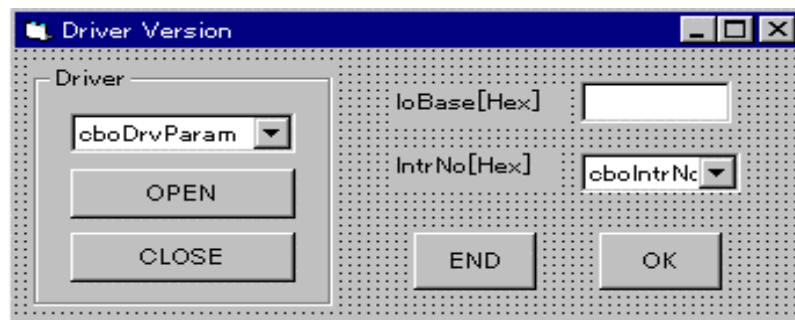
スタートアップ・フォーム (frmDriver)

本サンプルプログラムは当フォームから実行を開始します。

cmdOk_Click【OK ｷﾀﾞ】の処理で当 frmDriver を非表示し、frmMain をモーダルウインドウで表示します。これがアプリケーションの本体となります。

frmMain の cmdEnd_Click【END ｷﾀﾞ】の処理は frmMain をアンロードします。これにより frmDriver に制御が戻ってドライバのクローズ処理が行われ、プログラムを終了します。

frmMain の cmdEnd の処理が “ E N D ” ではなく “ Unload frmMain ” であることに注意してください。また frmMain で使用する “ iobase ” “ inter_no ” “ devno ” の各変数は dio270.bas でグローバル宣言してあります。



OPEN

(cmdOpen) : デバイスドライバをオープンします。
 W I N D O W S 9 x ・ M E の場合は当ボタンをクリックするだけ、N T の場合は使用するデバイスドライバの枝番号 (0 ~ 15) を指定してから当ボタンをクリックします。

CLOSE

(cmdClose) : デバイスドライバをクローズします。

OK

(cmdOk) : プラグアンドプレイで割り当てられている I / O ベースアドレスと割り込み番号が (H E X で) テキスト B O X に表示されます。クリックするとボードの存在を確認し、存在していれば “ frmMain ” に制御を渡します。
 また (“ frmMain ” のアンロードで) 制御が戻って来たらドライバのクローズを行い、終了します。

END

(cmdEnd) : 終了します。

サンプル実行フォーム (frmMain of smp270.frm)

R E S E T

(cmdReset) : 本ボードをリセットします。

E N D

(cmdEnd) : “frmMain” をアンロードして “frmDriver” に制御を戻します。

タイマ

(Timer1) : タイマのプロパティで設定した時間々隔で STATUS フレームの表示を更新し、デバイスドライバの GetIntrCount () 関数から得た割り込み発生回数値をテキストボックス txtIntrCount に表示します。

[DIOフレーム]

OUT	(cmdDioOut)	: テキストボックス txtDo に入力されたデータを各出力ポートに更新出力します。
IN	(cmdDioIn)	: 各入力ポートから現在値を読み込み、テキストボックス txtDi に表示します。

[DATAフレーム] of [COUNTER フレーム]

# 0 , # 1	(txtCntData)	: カウンタ # 0 , # 1 に書き込む値を入力、または読み出された値を表示します。
WRITE	(cmdDioOut)	: テキストボックス # 0 , # 1 (txtCntData) に入力されたデータを各カウンタに書き込みます。
READ	(cmdDioOut)	: 各カウンタから読み出したデータをテキストボックス # 0 , # 1 (txtCntData) に表示します。

[SOURCEフレーム] of [COUNTER フレーム]

10MHz / +EXT / -EXT	(cmdCntSrc) (optCntSrcInt) (optCntSrcExtRise) (optCntSrcExtFall)	: カウンタ入力源を3者択一で選択します。 選択肢は、内部 10MHz / 外部立上り (+EXT) / 外部立下り (-EXT) 。
---------------------------	---	--

[MODEフレーム] of [COUNTER フレーム]

Divider / Preset	(fraCntMode) (optCntModeDiv) (optCntModePre)	: カウンタ動作モードを選択します。 選択肢は、連続分周動作 (Divider) / プリセット動作 (Preset) 。
---------------------	--	---

[LENGTHフレーム] of [COUNTER フレーム]

16bit x1 / 16bit x2 / DIRECT	(fraCntLen) (optCntLen(0)) (optCntLen(1)) (optCntLen(2))	: カウンタ長を選択します。 DIRECT はカウンタを使用せず、カウンタ入力源が直接ステータス・割り込み・クロック出力に反映されます。
------------------------------------	---	---

[SOURCEフレーム] of [Trigger フレーム] of [COUNTER フレーム]

SOFT / EXT	(fraTrgSrc) (chkTrgSrcSoft) (chkTrgSrcExt)	: カウンタの計数開始タイミングを選択します。 即開始 (SOFT) / 外部トリガ入力で開始 (EXT) 。
START	(CmdTrgStart)	: 計数を即開始、またはトリガを待つて開始します。
STOP	(CmdTrgStop)	: 計数を即停止します。

[**MODE** フレーム] of [Trigger フレーム] of [COUNTER フレーム]

EDGE / LEVEL	(fraTrgMode)	: 外部トリガの動作モードを選択します。 選択肢は、
	(optTrgModeEdge)	外部トリガ入力の指定エッジで計数開始 (EDGE) /
	(optTrgModeLevel)	外部トリガ入力の指定レベル期間中計数 (LEVEL)。

[**POLARITY** フレーム] of [Trigger フレーム] of [COUNTER フレーム]

PLUS / MINUS	(fraTrgPol)	: 外部トリガの有効極性 (エッジ、またはレベル) を
	(optTrgPolPlus)	選択します。
	(optTrgPolMinus)	選択肢は、立上りエッジ、または正レベル (PLUS) / 立下りエッジ、または負レベル (MINUS)。

[**INTERRUPT** フレーム]

ENABLE	(cmdIntrEn)	: 割り込み発生回数を示す InterruptCount を クリアし、本ボードから P C I バスへの割り 込み信号出力を許可します。
DISABLE	(cmdIntrDis)	: 本ボードから P C I バスへの割り込み信号 出力を禁止します。
INTR COUNT	(InterruptCount)	: 割り込み発生回数を表示します。

[**SOURCE** フレーム] of [INTERRUPT フレーム]

CNT / TRG	(fraIntrSrc)	: 割り込み発生要因を指定します。
	(optIntrSrcCnt)	選択肢は、タイマ設定値 (TIMER) /
	(optIntrSrcTrg)	トリガ発生 (TRG)。

[**STATUS** フレーム]

SW - BN	(txtStatusBn)	: 本ボード上のスイッチ SW - BN に設定 された番号を表示します。
INTERRUPT	(shpStatusIntr)	: 割り込み要求フラグの状態を表示します。

【注】 [**SOURCE** フレーム] of [INTERRUPT フレーム] で選択された割り込み
要因は **ENABLE** されていなくても当ステータス (shpStatusIntr)
に反映されますが、実際の割り込みは **ENABLE** されなければ起り
ません。

第5章．DOSハンドラ

DIO-270PCIをMS-C、TURBO-C、およびBORLAND-Cで簡単に使用することのできるMS-DOS用ハンドラ(LIB)です。

本ボードの基本機能が関数化されており、ユーザは御自身の記述するメインルーチンの中から呼び出して使用することができます。

5-1. システム構成・ソフトウェア構造

パソコン本体 : IBM PC / AT互換機 (含む98NX機)
 本体メモリ量 : 640KB以上

OS / ライブラリ : MS-DOS (3.0以上) 上で、
 MS-C (7.0) / TURBO-C (4.0) / BORLAND-C (3.1) 以上。

【注1】 本ハンドラ・ライブラリは16bitコード専用です。

【注2】 本ハンドラ・ライブラリはWINDOWSのDOS窓では使用できません。

供給メディア : 本ボード添付のサンプルディスク内。

対応ボード : DIO-270PCI

【実現機能】

合計24ビット汎用デジタル入出力

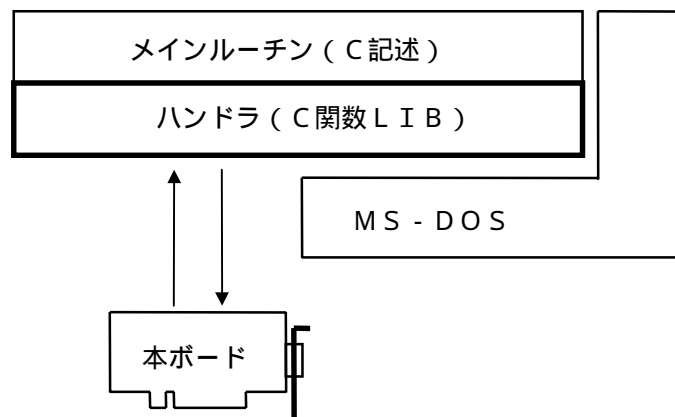
プログラマブルタイマ (連続分周) : ポーリング、または割り込みで検出。

プリセットカウンタ (単発動作) : 同上、外部イベント計数可能。

割り込み : 使用は任意。(使用する場合は要リソース取得 / 1-5項参照。)

その他のボード : I/Oアドレス、割り込み番号が重複しない限り同時に使用可能。

図5-1. ソフトウェア構造



5-2. 使用準備（ボード上の設定、入力接続）： 標準設定【1-2項】とします。

5-3. ユーザプログラム記述

御自身の記述したメインプログラムと本ハンドラ（LIB）をコンパイル/リンクして使用します。テストには付属のサンプルプログラムを御利用ください。なおライブラリは4-1項に従ってインストールしておきます。通常以下の手順です。（【4】～【11】はカウンタ利用）

具体的なコーディングについてはサンプル・ソースを御覧ください。

- (1)初期化 : 【1】
- (2)汎用デジタル入力 : 【2】
- (3)汎用デジタル出力 : 【3】
- (4)割り込み設定 : 【4】 使用の有無。
- (5)クロック源、動作モード指定 : 【5】 ☐ ペアで使用する。
- (6)カウンタ書き込み : 【6】 ☐
- (7)タイマ設定 : 【7】 時間値で指定・設定（この場合は前【5】【6】不要）
- (8)カウンタ・スタート : 【8】 計数開始（or トリガ待ち）
- (9)ステータス評価 : 【9】
- ()以下、任意
- (11)カウンタ・ストップ : 【11】 計数停止
- (12)ハンドラ終了 : 【12】 後処理

表 5-3 . 制御関数一覧

関数名	機能・内容	引数（パラメータ）等
【1】dio_open_diosys	ボード、および本ハンドラの初期化	
【2】dio_inp_24bit	汎用デジタル（現在値）入力	入力データ
【3】dio_out_24bit	汎用デジタル（ラッチ）出力	出力データ
【4】dio_set_irq	割り込み設定	
【5】dio_set_cntmode	クロック源選択&カウンタ動作モード	
【6】dio_write_cnt	カウンタ（入力レジスタ）書き込み	
【7】dio_set_timer	タイマ値&カウンタ動作モード	
【8】dio_start_cnt	カウンタ・スタート（トリガ許可）	
【9】dio_get_status	ステータス取得	
【10】dio_read_cnt	カウンタ読み出し	
【11】dio_stop_cnt	カウンタ・ストップ（トリガ禁止）	
【12】dio_close_diosys	本ハンドラの終了	
	《以上が基本、以下は補助的》	
【13】dio_onint_func	割り込み発生時に実行する関数の設定	ユーザ関数のポインタ
【14】dio_get_libver	本ハンドラのバージョン番号を得る	

5-4. 関数仕様・エラーコード

以下に各関数の仕様・詳細を記します。

【1】初期化

書式	int dio_open_diosys (void)
引数	なし
戻り値	正常終了時 : ボードのID / DIO-270PCI:DH エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	プラグアンドプレイで設定したI/Oアドレス、割り込み番号を本ハンドラが自動認識すると共に、本ボードのリセット、ハンドラ内部の参照テーブルやデータバッファを初期化する。

【2】汎用デジタル入力（現在値）の読み込み

書式	int dio_inp_24bit(int &in_2x, int &in_1x, int &in_0x)
引数	in_2x : 2xポート入力データ (8ビット現在値) in_1x : 1xポート入力データ (8ビット現在値) in_0x : 0xポート入力データ (8ビット現在値)
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	汎用デジタル入力ポートから24ビット (8ビット×3) の現在値を読み込む。 出力に割付けられているビットは (その出力値を) 読み返すことになる。

【3】汎用デジタル（ラッチ）出力の更新書き込み

書式	int dio_out_24bit(int out_2x, int out_1x, int out_0x)
引数	out_2x : 2xポート (ラッチ) 出力更新データ out_1x : 1xポート (ラッチ) 出力更新データ out_0x : 0xポート (ラッチ) 出力更新データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	汎用デジタル出力ポート24ビット (8ビット×3) に更新データを書き込む。 更新された出力データは次回の更新、またはパワーオンリセットまで保持される。 なお、ソフト的なリセット (制御部リセット操作) ではクリアされない。 また、入力に割付けられているビットは無効データとなる。

【4】割り込み設定

書式	int dio_set_irq (int irq_enb, int irq_pol)
引数	irq__enb : クロック割り込み制御 / 0 : 禁止、 1 : 許可
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	クロックによる割り込みを使用するか否か、および使用する場合の信号極性を指定する。 【注】実際に割り込みを使用するにはリソース取得 (1-5 項) が必要です。

【5】クロック源選択&カウンタ動作モード指定 /* 当関数は後【6】とペアで使用する */

書式	int dio_set_cntmode (int cnt_source, exclk_edge, cnt_length, int cnt_mode)
引数	cnt__source : 計数対象 / 0 : 内部クロック源、 1 : 外部クロック源入力 excl__k__edge : 外部クロック源入力の有効極性 / 0 : 立下り、 1 : 立上り cnt__length : カウンタ長 / 0 : 16ビット (1本)、 1 : 32ビット (16ビット×2本) cnt__mode : 動作モード / 0 : 連続分周、 1 : 単発プリセット、 3 : 外部クロック源を非分周
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	使用するカウンタ長、計数対象、極性、および動作モードを指定する。 動作モードについては3 - 5 項参照。 なお<連続分周>はタイマ専用で、イベント計数には利用できません。 / 読み出し計数値が無効 / また、<外部クロック源を非分周>は外部入力をそのままクロック出力、連続計数とするもの。

【6】カウンタ (入力レジスタ) 書き込み /* 当関数は前【5】とペアで使用する */

書式	int dio_write_cnt(WORD cnt_data_w0, WORD cnt_data_w1)
引数	cnt__data__w0 : カウンタ# 0 に書き込む 16 ビット・データ cnt__data__w1 : カウンタ# 1 に書き込む 16 ビット・データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	両カウンタに各 16 ビット・データを書き込む。 当関数実行時に前【5】で指定したカウンタ動作モードも併せて設定される。 指定カウンタ長が 16 ビット (カウンタ# 0 のみ使用) のときの cnt__data__1 はダミー。

【7】タイマ・データ算出・設定&動作モード指定

書式	int dio_set_timer(long &clk_period, int time_unit, int next_value, int cnt_mode)
引数	clk__period : 指定タイマ値。 time__unit : 指定タイマ値の単位 / 0 : sec、 1 : ms、 2 : μ s、 3 : ns next__value : 近似値指定 / 0 : 小さい方の値、 1 : 大きい方の値 cnt__mode : 動作モード / 0 : 連続分周、 1 : 単発プリセット
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能動作	内部クロック源 10 MHz を使用したタイマデータを算出してカウンタ (入力レジスタ) に書き込む。 ぴったりの値が設定できない場合は近似値を設定する。 当関数を使用する場合は前記関数【5】【6】は不要。

【8】カウンタ・スタート（トリガ許可動作）

書式	int dio_start_cnt(int trg_mode, int trg_pol)
引数	trg_mode : トリガ動作モード / 0 : ソフト (即)、1 : 外部トリガ (トリガ待ちスタート) trg_pol : 外部トリガ極性 / 0 : 負エッジ、1 : 正エッジ、2 : 負レベル、1 : 正レベル
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	計数動作を即開始 (またはトリガ待ち) する。 当関数実行時に前【5】または【7】で指定したクロック源選択、カウンタ関連パラメータも併せて設定される。

【9】ステータス取得

書式	int dio_get_status (int &clk_irq, int status)
引数	clk_irq : 割り込みフラグ / 0 : リセット、1 : セット (Read clear) status : 本ボードのステータス生データ (3 - 8 項)
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	当該時点でのクロック割り込みフラグ、および本ボードのステータス生データを得る。 クロック割り込みフラグは当関数でセット (= 1) を検出されると、直後に自動クリアされる。 clk_irq は実際の割り込み発生の有無にかかわらずクロックの有効エッジによりセットされる。 したがって、前【4】の割り込みを禁止した状態で当フラグをポーリングする利用法もある。

【10】カウンタ（ラッチ）読み出し

書式	int dio_read_cnt(WORD &cnt_data_r0, WORD &cnt_data_r1)
引数	cnt_data_r0 : カウンタ # 0 から読み出した 16 ビット・データ cnt_data_r1 : カウンタ # 1 から読み出した 16 ビット・データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	両カウンタから各 16 ビット・データを読み出す。 (マルチプルラッチ・コマンドで同時刻データをラッチして読み出す。) 指定カウンタ長が 16 ビット (カウンタ # 0 のみ使用) のときの cnt_data_r1 は無効値。

【11】カウンタ・ストップ（トリガ禁止動作）

書式	int dio_stop_cnt (int dis_trg_0, int dis_trg_1)
引数	なし
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	計数動作をストップ (強制停止) させる。

【12】本ハンドラの終了

書式	int dio_close_diosys (void)
引数	なし
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	本ハンドラの終了。 本ボードのリセットを行う。 但し、汎用デジタル出力とカウンタ内部はクリアされず、現状を維持する。

【13】割り込み発生時に実行するユーザ関数の設定

書式	int dio_onint_func (void interrupt_far *func)
引数	func : ユーザ作成関数のポインタ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	ユーザが任意に作成した割り込み処理関数の指定。 当指定は前【4】で割り込みを許可したときに有効となる。

【14】P C Iバス上の割り込み信号リセット & 再許可 (前【13】内で実行する)

書式	int dio_clear_irq (void)
引数	なし
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	P C Iバス上の割り込み信号出力を一旦リセットし、また次の割り込み信号出力を可能にする。 本ボードが出力するP C Iバス上の割り込み要求信号は割り込み処理関数 (前【13】) 内で一旦クリアする必要がある。

【15】本ハンドラのバージョン取得

書式	int dio_get_libver (int ver)
引数	ver : = 0 のとき、戻り値は メジャーバージョン番号 + マイナーバージョン番号 (上位バイト) (下位バイト)
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	本ハンドラ (ライブラリ) のバージョン番号を得る。

エラー 本ハンドラの各関数は実行前後（または実行中）に不適当なパラメータや動作状態を検出するとエラーコードを返してきます。

表5 - 4 B . エラーコード一覧

戻り値	不具合の内容、または因果情報	適用関数、引数、等
- 1	ボードを検出できない。（ボード不在、I/Oアドレス不一致、等）	dio_open_diosys ()
- 2	ハンドラ初期化の未実行。	
- 3	カウンタ関連設定の未実行（不足）	dio_start_cnt ()
- 4	割り込み設定 パラメータの不適当	irq_enb
- 5	計数対象 パラメータの不適当	cnt_source
- 6	外部クロック源入力の有効極性 パラメータの不適当	exclk_edge
- 7	カウンタ長 パラメータの不適当	cnt_length
- 8	動作モード パラメータの不適当	cnt_mode
- 9	カウンタ書き込みデータ パラメータの不適当	cnt_data_w0, W1
- 10	指定タイマ値 パラメータの不適当	clk_period
- 11	指定タイマ値の単位 パラメータの不適当	time_unit
- 12	近似値指定 パラメータの不適当	next_value
13	動作モード パラメータの不適当	cnt_mode
14	トリガ動作モード パラメータの不適当	trg_mode
- 15	外部トリガ動作極性 パラメータの不適当	trg_pol
- 16	ユーザ作成関数 パラメータの不適当	func
- 20	動作中	

第6章 . WINDOWSハンドラ

DIO-270PCIをVC、VB等で簡単に使用することのできるWINDOWS 9x/NT/2000用のハンドラDLL(+ドライバ)です。本ボードの基本機能が関数化されており、ユーザは御自身の記述するメインルーチンの中から呼び出して使用することができます。

6-1. システム構成・ソフトウェア構造

パソコン本体 : IBM PC / AT互換機 (含む98NX機)

拡張メモリ量 : 16MB以上

OS / コパイ : WINDOWS 9x・ME・NT・2000・XP / 32ビット専用。

添付サンプル : Visual-C, C++ (5.0)

Visual-Basic (5.0)

Borland-C (5.0), Delphi (3.0)

供給メディア : 本ボード添付のサンプルディスク内。

対応ボード : DIO-270PCI (1枚に限る。)

【実現機能】

合計24ビット汎用デジタル入出力

プログラマブルタイマ (連続分周) : ポーリングで検出。

プリセットカウンタ (単発動作) : 同上、外部イベント計数可能。

割り込み : 不使用

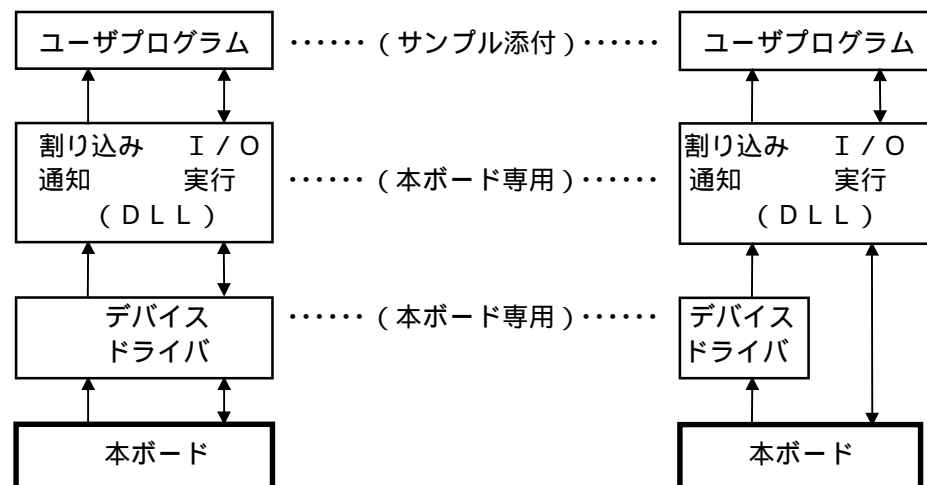
その他のボード : I/Oアドレスが重複しない限り同時に使用可能。

市販ソフト対応 : LabVIEW等から利用可能。

図6-1A . ソフトウェア構造

(WINDOWS NT/2000/XP の場合)

(WINDOWS 9x・ME の場合)



6-2. 使用準備

ボード上の設定・入力接続は標準設定【1-2項】とします。

まず第4章（4-1項）にしたがって本ボード関連ファイルをインストールします。
次に本ハンドラDLL / 専用ドライバを所定のフォルダに移すインストール作業は以下のとおりです。

WINDOWS 95・98・MEの場合 （ :バージョン番号）

- (1) %WINDOWS フォルダに“H270__95 .DLL”をコピーする。
- (2) %WINDOWS%SYSTEM フォルダに“D270__95 .VXD”をコピーする。

WINDOWS NT (4.0) の場合 / Administrator レベルで行う /

- (1) %WINDOWS%SYSTEM32 フォルダに“H270__NT .DLL”をコピーする。
- (2) %WINDOWS%SYSTEM32%DRIVERS フォルダに“D270__NT .SYS”をコピーする。
- (3) 任意のフォルダにユーティリティ“D270Reg.exe”をコピーする。

《デバイスドライバの設定 / リソースの確認》

デバイスドライバの設定 / リソースの確認ユーティリティ（D270Reg）を起動すると、本ボードの（プラグアンドプレイで設定された）I/Oアドレス・割り込みレベル情報をレジスタに書き込み、（確認のため）表示し、さらに本専用ドライバに同情報を設定します。

このとき同時に、以後の本専用ドライバ起動方法（自動 / 手動 / 禁止）を選択設定します。

WINDOWS 2000 / XP の場合 / Administrator レベルで行う /

- (1) 当社製PCIボード（複数可能）に共通使用できるWINDOWS 2000用のWDMドライバ“DMS_PCI.SYS”はボードインストール時に（ボードインストールディスクから）自動的にインストールされます。

インストール先：%WINDOWS%SYSTEM32%DRIVERS フォルダ

このWDMドライバは当社製の全PCIボード（複数可）から共通に使用できる汎用品です。すなわち各個別PCIボード専用の関数DLLを用意すれば、当WDMドライバ1本で当社製の全PCIボードを動作させることができます。

- (2) 本関数DLLもボードインストール時に所定の %WINDOWS%SYSTEM32 にコピーされているので、即サンプル（%MSCIENCE%HND_2K%Dio270 以下）を使用できます。

LabVIEW等の完成アプリケーションから利用する場合は

WINDOWSハンドルの引き渡し、すなわちメッセージングの利用ができない場合は6-4項【1】初期化関数で引数Owner = 0とすればメッセージング機能が無効となります。この場合の使用方法はアプリケーション側からステータス関数をポーリングして各関数実行タイミングを得ることを前提にしています。

6-3. ユーザプログラム記述

御自身の記述したメインプログラムから本ハンドラDLL（+ドライバ）を使用します。
 テストには付属のサンプルプログラムを御利用ください。前6-2項に従ってインストールしておきます。本ボードの操作は通常以下の手順です。（【4】～【10】はカウンタの利用）
 具体的なコーディングについてはサンプル・ソースを御覧ください。

- | | | |
|------------------|--------|---------------------------|
| (1)初期化 | : 【1】 | |
| (2)汎用デジタル入力 | : 【2】 | |
| (3)汎用デジタル出力 | : 【3】 | |
| (4)クロック源、動作モード指定 | : 【4】 | └──────────┘ ペアで使用する。 |
| (5)カウンタ書き込み | : 【5】 | |
| (6)タイマ設定 | : 【6】 | 時間値で指定・設定（この場合は前【4】【5】不要） |
| (7)カウンタ・スタート | : 【7】 | 計数開始（or トリガ待ち） |
| (8)ステータス評価 | : 【8】 | |
| ()以下、任意 | | |
| (10)カウンタ・ストップ | : 【10】 | 計数停止 |
| (11)ハンドラ終了 | : 【11】 | 後処理 |

表 6-3 . 制御関数一覧

関数名	機能・内容	引数（パラメータ）等
【1】Dio_Open_Diosys	ボード、および本ハンドラの初期化	
【2】Dio_Inp_24bit	汎用デジタル（現在値）入力	入力データ
【3】Dio_Out_24bit	汎用デジタル（ラッチ）出力	出力データ
【4】Dio_Set_CntMode	クロック源選択&カウンタ動作モード	
【5】Dio_Write_Cnt	カウンタ（入力レジスタ）書き込み	
【6】Dio_Set_Timer	タイマ値&カウンタ動作モード	
【7】Dio_Start_Cnt	カウンタ・スタート（トリガ許可）	
【8】Dio_Get_Status	ステータス取得	
【9】Dio_Read_Cnt	カウンタ読み出し	
【10】Dio_Stop_Cnt	カウンタ・ストップ（トリガ禁止）	
【11】Dio_Close_Diosys	本ハンドラの終了	

【注】 WINDOWS 2000 / XP用に限り、関数名が異なります。
 すなわち関数名中、文字“Dio”の直後に対象ボード名（3桁番号）が含まれます。
 例：Dio270_Open_DASys

重 要

本ハンドラDLLを（LabVIEWなどの）市販・完成アプリケーションから呼出し実行するときはオープン&初期化関数【1】でウインドウハンドル **HWND Owner** に **NULL** を渡します。この場合、アプリケーション側は本ハンドラからの波形出力終了メッセージを受け取ることができません。

6-4. 関数仕様・エラーコード

以下に各関数の仕様・詳細を記します。

【注】 WINDOWS 2000 / XP用に限り、
各関数名中“DA”の直後に3桁のボード番号が
入ります。

例 : Dio270_Open_DASys

【1】初期化

書式	int Dio_Open_DioSys (HWND Owner)
引数	Owner : ウインドウハンドル (メッセージングに使用。)
戻り値	正常終了時 : ボードのID / DIO-270PCI : DH エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	プラグアンドプレイで設定したI/Oアドレスを本ハンドラが自動認識すると共に、 本ボードのリセット、ハンドラ内部の参照テーブルやデータバッファを初期化する。 なお、Owner = 0とすればメッセージング機能が無効となり、 LabVIEW等からも利用可能になる。

【2】汎用デジタル入力 (現在値) の読み込み

書式	int Dio_Inp_24bit(int &in_2x, int &in_1x, int &in_0x)
引数	in_2x : 2xポート入力データ (8ビット現在値) in_1x : 1xポート入力データ (8ビット現在値) in_0x : 0xポート入力データ (8ビット現在値)
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	汎用デジタル入力ポートから24ビット (8ビット×3) の現在値を読み込む。 出力に割付けられているビットは (その出力値を) 読み返すことになる。

【3】汎用デジタル (ラッチ) 出力の更新書き込み

書式	int Dio_Out_24bit(int out_2x, int out_1x, int out_0x)
引数	out_2x : 2xポート (ラッチ) 出力更新データ out_1x : 1xポート (ラッチ) 出力更新データ out_0x : 0xポート (ラッチ) 出力更新データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	汎用デジタル出力ポート24ビット (8ビット×3) に更新データを書き込む。 更新された出力データは次回の更新、またはパワーオンリセットまで保持される。 なお、ソフト的なリセット (制御部リセット操作) ではクリアされない。 また、入力に割付けられているビットは無効データとなる。

【4】クロック源選択&カウンタ動作モード指定 /* 当関数は後【5】とペアで使用する */

書式	int Dio_Set_CntMode (int cnt_source , exclk_edge , cnt_length, int cnt_mode)
引数	cnt_source : 計数対象 / 0 : 内部クロック源、 1 : 外部クロック源入力 exclk_edge : 外部クロック源入力の有効極性 / 0 : 立下り、 1 : 立上り cnt_length : カウンタ長 / 0 : 16ビット(1本)、 1 : 32ビット(16ビット×2本) cnt_mode : 動作モード / 0 : 連続分周、 1 : 単発プリセット、 3 : 外部クロック源を非分周
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値(エラーコード表)
機能・動作	使用するカウンタ長、計数対象、極性、および動作モードを指定する。 動作モードについては3 - 5項参照。 なお、<連続分周>はタイマ専用で、イベント計数には利用できません。 / 読み出し計数値が無効 / また、<外部クロック源を非分周>は外部入力をそのままクロック出力、連続計数とするものです。

【5】カウンタ(入力レジスタ)書き込み /* 当関数は前【4】とペアで使用する */

書式	int Dio_Write_Cnt(int cnt_data_0, int cnt_data_1)
引数	cnt_data_w0 : カウンタ#0に書き込む16ビット・データ cnt_data_w1 : カウンタ#1に書き込む16ビット・データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値(エラーコード表)
機能・動作	両カウンタに各16ビット・データを書き込む。 当関数実行時に前【4】で指定したカウンタ動作モードも併せて設定される。 指定カウンタ長が16ビット(カウンタ#0のみ使用)のときの cnt_data__1はダミー。

【6】タイマ・データ算出・設定&動作モード指定

書式	int Dio_Set_Timer(long &clk_period, int time_unit , int next_value , int cnt_mode)
引数	clk_period : 指定タイマ値。 time_unit : 指定タイマ値の単位 / 0 : sec、 1 : ms、 2 : μs、 3 : ns next_value : 近似値指定 / 0 : 小さい方の値、 1 : 大きい方の値 cnt_mode : 動作モード / 0 : 連続分周、 1 : 単発プリセット
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値(エラーコード表)
機能・動作	内部クロック源10MHzを使用したタイマデータを算出してカウンタ(入力レジスタ)に書き込む。 ぴったりの値が設定できない場合は近似値を設定する。 当関数を使用する場合は前記関数【4】【5】は不要。

【7】カウンタ・スタート（トリガ許可動作）

書式	int Dio_Start_Cnt(int trg_mode, int trg_pol, int messaging)
引数	trg_mode : トリガ動作モード / 0 : ソフト（即）、1 : 外部トリガ（トリガ待ちスタート） trg_pol : 外部トリガ極性 / 0 : 負エッジ、1 : 正エッジ、2 : 負レベル、1 : 正レベル messaging : メッセージング使用の有無 / 0 : 不使用、1 : 使用
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	計数動作を即開始（またはトリガ待ち）する。 当関数実行時に前【4】または【6】で指定したクロック源選択、カウンタ関連パラメータも併せて設定される。 メッセージングは有効なクロック出力によるもの。（後記【8】の割り込みフラグ相当）

【8】ステータス取得

書式	int Dio_Get_Status (int clk_irq, int status)
引数	clk_irq : 割り込みフラグ / 0 : リセット、1 : セット（Read clear） status : 本ボードのステータス生データ（3 - 8 項）
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	当時点での割り込みフラグ、および本ボードのステータス生データを得る。 割り込みフラグは当関数でセット（= 1）を検出されると、直後に自動クリアされる。 clk_irq は実際の割り込み発生の有無にかかわらずクロックの有効エッジによりセットされる。（当ハンドラでは割り込みを使用していない。）

【9】カウンタ（ラッチ）読み出し

書式	int Dio_Read_Cnt(int &cnt_data_r0, int &cnt_data_r1)
引数	cnt_data_r0 : カウンタ # 0 から読み出した 16 ビット・データ cnt_data_r1 : カウンタ # 1 から読み出した 16 ビット・データ
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	両カウンタから各 16 ビット・データを読み出す。 （マルチプルラッチ・コマンドで同時刻データをラッチして読み出す。） 指定カウンタ長が 16 ビット（カウンタ # 0 のみ使用）のときの cnt_data_r1 は無効値。

【10】カウンタ・ストップ（トリガ禁止動作）

書式	int Dio_Stop_Cnt (int dis_trg_0, int dis_trg_1)
引数	なし
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	計数動作をストップ（強制停止）させる。

【11】本ハンドラの終了

書式	int Dio_Close_DioSys (void)
引数	なし
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	本ハンドラの終了。 本ボードのリセットを行う。 但し、汎用デジタル出力とカウンタ内部はクリアされず、現状を維持する。

エラー 本ハンドラの各関数は実行前後（または実行中）に不適当なパラメータや動作状態を検出するとエラーコードを返してきます。

表 6 - 4 B . エラーコード一覧

戻り値	不具合の内容、または因果情報	適用関数、引数、等
- 1	ボードを検出できない。	Dio_Open_DioSys ()
- 2	ドライバファイルを検出できない。	Dio_Open_DioSys ()
- 3	ドライバファイルのバージョンが違う。	Dio_Open_DioSys ()
- 4	ハンドラ初期化の未実行。	
- 5	カウンタ関連設定の未実行 (不足)	Dio_Start_Cnt ()
- 6	Owner パラメータの不適当	Dio_Open_DioSys ()
- 7	計数対象 パラメータの不適当	cnt_source
- 8	外部クロック源入力の有効極性 パラメータの不適当	extclk_edge
- 9	カウンタ長 パラメータの不適当	cnt_length
- 10	動作モード パラメータの不適当	cnt_mode
- 11	カウンタ書き込みデータ パラメータの不適当	cnt_data_w0, w1
- 12	指定タイマ値 パラメータの不適当	clk_period
- 13	指定タイマ値の単位 パラメータの不適当	time_unit
- 14	近似値指定 パラメータの不適当	next_value
- 15	動作モード パラメータの不適当	cnt_mode
- 16	トリガ動作モード パラメータの不適当	trg_mode
- 17	外部トリガ動作極性 パラメータの不適当	trg_pol
- 18	メッセージ要求 パラメータの不適当	messaging
- 20	動作中	

第7章．保守・その他

7-1. 故障・トラブル等の原因と対処

本機は【DOS/V系パソコン】+【拡張ボックス】のシステム構成で全数検査のうえ出荷されています。お手元での動作確認方法は1-6項に記されています。動作に不具合があるときは以下の諸点を再点検してください。それでも不明なときは巻末の【Q&Aフォーム】にシステム構成（特に外部機器の接続回路）等の動作条件を御記入のうえ、技術部宛FAXしてください。

迅速に応答する体制となっています。なおTELいただく場合も、客観情報の整理・評価は問題解決のスピードアップにつながりますから、事前に【Q&Aフォーム】をFAXしてください。

再点検・確認ポイント

- | | |
|-------------|---|
| (1) I/Oアドレス | ボードのインストール/認識は成功したか？(1-5項) |
| (2) 割り込みレベル | リソースは取得できたか？(1-5項) |
| (3) デジタル入出力 | 本ボードのTTL入力端子に接続できる信号源はTTLレベル（LS、CMOS等の5V電源動作素子）に限ります。
現場で不適切な信号源を接続したためにボード内のTTL入力素子を破損する事故が頻発していますので御注意ください。
(次ページ/図7-1参照) |

動作確認方法

当社では原則として、ユーザ作成のソフトウェアについては評価しません。動作確認は本製品添付の当社製プログラム(1-6項)の実行結果について推測・適否・判定を行います。

QAリクエスト時には当プログラムの実行結果をレポートしてください。

ボード内TTL入力素子破損の主な原因

TTL入力素子の絶対最大定格は【負側：-0.6V】【正側：+7V】です。このレベルを一瞬でも超えると入力素子破壊の原因になります。主な危険要素は、

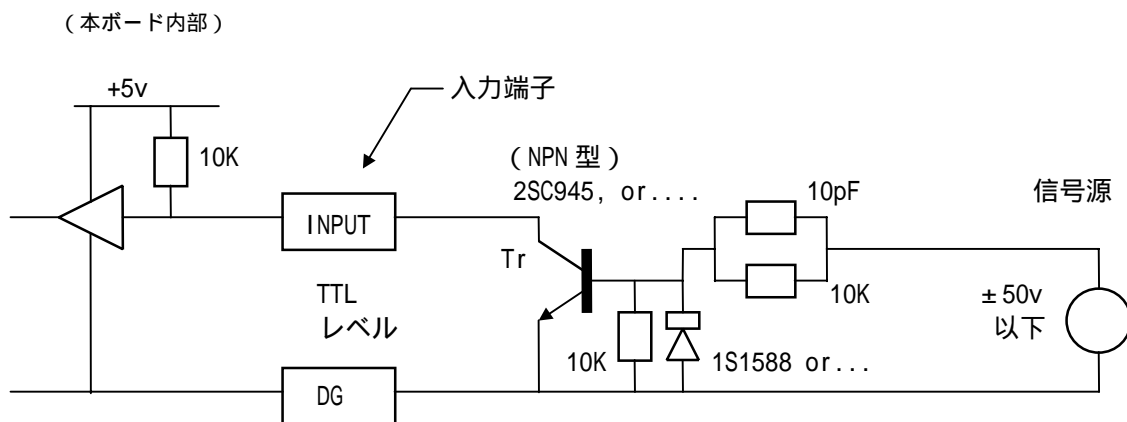
ファンクション・ジェネレータ等の交流信号出力を接続して破損させる例が多いようです。矩形波でも±に振れる信号は接続できません。特に、負側の許容レベル【-0.6V】が低いことに注意してください。

+5V以上に振れるロジック信号も接続できません。12V～24V電源を使用する機器からのデジタル信号は不可、信号レベルが不明なときは信号源の電源電圧が目安になります。

アナログ信号源は±15V電源によるオペアンプ出力が多く危険です。なお、TTL入力にアナログ信号を接続しても立上り/立下り特性等が仕様を満足せず、正常な動作は期待できないでしょう。

信号源と本ボードのグランド・レベルに差があるときも危険です。（テストで測定可能）

図7-1. 【高レベル信号】 【TTLレベル】変換回路例



《注》本回路はインバータ（極性反転）です。

7-2. 修理のときは

入手経路の如何にかかわらず当社宛に直接お申しつけください。 商社等を経由されますと時間がかかるだけでなく、情報交換の不便、費用の面でも不利になります。 なお当社では修理依頼を受けた製品が検査の結果、良品と判定された場合は（保証期間内でも）手数料を申し受けます。

特に最初からの不具合には誤解や情報不足によることが多いので、事前に御相談ください。

【Q & A フォーム】が便利です。

無償修理

納入後 1 年以内の自然故障、および当社製造上の問題に起因した故障に対しては無償修理を行います。 但し、故障・不具合の原因や無償修理の対象となるか否かは（過去の経験等に照らして）当社側で判定させていただきます。

なお当社では保証書を発行していませんが、社内では製造番号と出荷年月日の記録を基に判定しています。

有償修理

落雷等の自然現象、漏電・過電圧印加・機械的破損・その他、ユーザ側の責に帰する故障品、または納入後 1 年間を経過した製品の自然故障に対しては実費・有償にて修理をお請けします。 性格上、事前見積もりは不可能ですが、制限額を事前通知いただければ、作業過程で制限を超えそうな見通しがたった時点で連絡・相談させていただきます。

受け渡し : 宅配便によるセンドバックで行います。

修理期間 : 全んどの場合、当社内で 24 時間以内に完了・返送しています。時間を要する場合は御連絡いたします。

費用の目安 : 修理費用は事務管理手数料、技術者の所要時間（1 時間単位）手数料、および交換部品代の合計です。 2002 年 9 月現在（時勢により変動します）では、

事務管理手数料（1 件当り、返送運賃含）：＝ ¥ 4,000

修理時間手数料：＝（時間単価 ¥ 6,000）× 所要時間

交換部品代 : ＝ ¥ 実費

故障経緯、システム客観情報の添付は時間の節約・コストダウンに有効です。典型的な事例では費用合計が ¥ 20,000 を超えることは希れです。

【注】 当社製品に対してユーザが改造を行った場合は、当社サポートの対象外になります。 改造とは製品に新たな部品を追加実装、または実装部品を削除したり、回路パターン・接続に変更を加えることです。 なお、当社がオプションとして供給、または指定した部品の追加実装・交換はこの限りではありません。

7-3. 付録 (WINDOWS 2000・XPについて)

WINDOWS 2000

ボードのインストール： WINDOWS 2000はNT4.0の上位バージョンですが、プラグアンドプレイ機能を持つため、本ボード装着直後のインストール作業時にWINDOWS 2000 対応のインストールディスク（当社製 /vr 2.00 以上）が必要です。作業手順は本書 1 - 5 項、または本ボードに同梱の作業説明書に従ってください。

ソフトウェアサポート： 汎用のI/Oドライバ、および本ボード専用の関数DLLが追加されています。前者については4 - 2 項、後者については第6章をごらんください。

WINDOWS-XP

ボードのインストールからドライバ、ハンドラ関数DLLまで、添付のWINDOWS 2000用ソフトウェアがそのまま御利用いただけます。

FAX : 03 (3301) 5593

Q & A フォーム

発信： 年 月 日 / 時 分

製品名	D I O - 2 7 0 P C I		購入時期	年	月	
ボード上の 設定、 使用状況						
その他						
I / O、 周辺状況	同時使用の 他ボード			I / Oアドレス 割り込み、等		
本体 システム	パソコン本体			拡張 B O X		
	本体メモリ					
	O S	D O S () W I N ()				
ソフト	言語			コンパイラ	(v r)	
	プログラム名					
(動作状況)						

《60分以内に応答のないときはお叱りください。》TEL: 03(3396)8377

御使用者	(所属部・課)
団体名	
T E L	(所在地)
F A X	