

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

本製品用のソフトウェアは3.5インチ(1.44MB)FDまたはCDに圧縮された形で格納されており、同メディア内の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。

ハンドラL I B： D Aボード制御の基本機能を関数化、サンプリング条件設定から実行(16bit/DOSライブラリ)まで、ユーザ記述のメインプログラムから呼び出すだけです。
【専用マニュアル：第5章】

ハンドラD L L： D Aボード制御の基本機能を関数化、サンプリング条件設定から実行(WIN32ライブラリ)まで、ユーザ記述のメインプログラムから呼び出すだけです。
【専用マニュアル：第6章】

サンプルソフト： D Aボード制御プログラムを手造りするユーザ向けの操作例です。(16bit/DOSソース) C、およびQ u i c k - B a s i c で例示しています。【本章】ポーリング、割り込み等の使用方法を学習することができます。

その他： W I N D O W S 9 5 / 9 8 / M E / N T / 2 0 0 0 / X P で、I / O 読み書きを実行するための汎用ドライバ、およびD L L が添付されています。
(for W I N D O W S) 基本的には当D L L を使用して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。
(当部分は圧縮されておらず、生のまま格納されています。)

= インストール作業 = (添付のCDROMを御使用ください。)

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) WINDOWS付属のエクスプローラで、
D: ¥INSTALL¥PCI¥DA¥MDA781 を開く。
- (2) “Setup.EXE”を実行(ダブルクリック)する。

当操作以下によりMDA-781PCI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:¥WINDOWS>CD¥【ENTER】
C:¥>CD D: ¥INSTALL¥PCI¥DA¥MDA781【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、または設定しておいてく
ださい。

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

全ファイルをインストールした場合のディレクトリ構造は図4-1の
ようになります。

【注1】 本ボード専用のWINDOWS版ハンドラDLL/デバイスドライバは当作業の後、
6-2項に従って必要なファイルを適合フォルダにコピーする必要があります。

【注2】 ボード依存性のない汎用のWINDOWS版I/O実行DLL/デバイスドライバは
当作業ではインストールされません。 WINDOWS95・98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。

WINDOWS NT用はWinNTフォルダ中にあり、同フォルダ中の専用インス
トールで導入してください。 また、WINDOWS2000用ドライバ(WDM)は
ボード自体のインストール時に自動インストールされます。 使用法、サンプルなどは
Win2Kフォルダ内にあります。

(追伸) CDROMの場合、Win9x、WinNT、およびWin2Kフォルダは
INSTALLフォルダ下のDrivers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。後日、追加・変更も有り得ます。

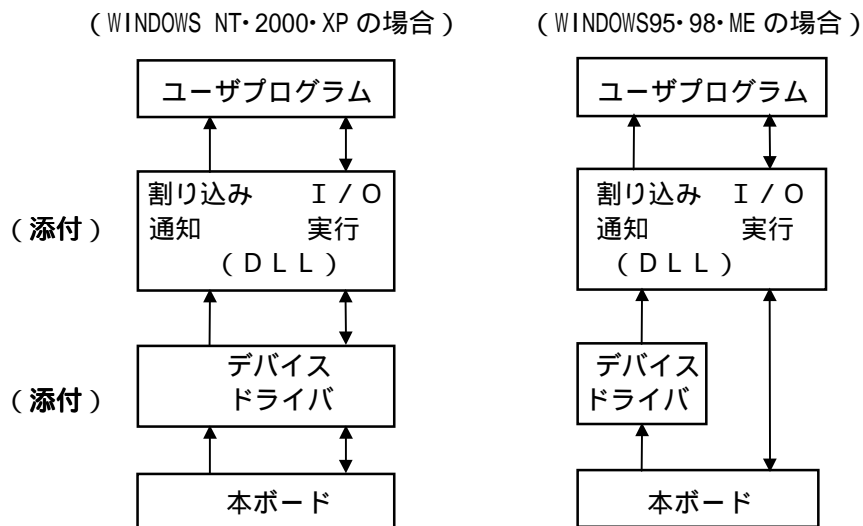
¥	凡例 MS - C : MICROSOFT - Cの略
	T - C : TURBO - Cの略
MSCIENCE	B - C : BORLAND - Cの略
	UTILITY WIN95 CF9050DP.COM : コンフィギュレーション (9x用)
	(本ボードの設定) - WINNT CF9050NT.EXE : コンフィギュレーション (NT用)
	- CF9050NT.sys : 上記NT用デバイスドライバ
	- REGSTDV.EXE : ドライバの登録ユーティリティ
	- BOARDTST - - - 781QB1.EXE : 本ボードの試運転・動作確認用プログラム
	- 781QB1.COM : 英語モードに切り替えた後、EXEを実行する
	- HND781C --- INCLUDE - SMPEXT.H : ハンドラ用・共通ヘッダファイル
	(DOS用ハンドラ)
	- LIB --- 781TS.LIB : T - C、B - C用スモールモデル
	781TL.LIB : T - C、B - C用ラージモデル
	781MS.LIB : MS - C用スモールモデル
	781ML.LIB : MS - C用ラージモデル
	SMP781 - SAMPLE .C
	- - SMP781C - - - MICROSOFT.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
	- BKINT781.C : クロック同期・非サイクル動作 (ブロック転送/割り込み)
	- BKPOL781.C : クロック同期・非サイクル動作 (同 上 /ポーリング)
	- INT781.C : クロック同期・非サイクル動作 (1データ毎/割り込み)
	- POL781.C : クロック同期・非サイクル動作 (同 上 /ポーリング)
	- CYC781.C : 波形出力/クロック同期・サイクル動作
	- MSCYC781.C : 波形出力/クロック同期・サイクル動作 (複数ボード)
	- PCI.C : PCI関連ルーチン
	(Q-Basic)
	- - SMP781B - - - 781QB1.BAS : Quick - Basic用サンプル・ソース
	Hnd_95 Mda781 Dll : ハンドラDLL
	(WINDOWS 9x・ME用ハンドラ) - 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 : Visual C++ (5.0)用サンプル
	- Delphi3 : Delphi (3.0)用サンプル
	Hnd_NT Mda781 D781reg : デバイスドライバ設定ユーティリティ
	(WINDOWS-NT用ハンドラ) - Dll : 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
	Hnd_2K Mda781 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)
	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
	(Win95・98・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

【注】 WINDOWS (NT/9x・ME/2000/XP)用の各汎用I/O制御DLL、デバイスドライバは別途。

4-2 . W I N D O W S ドライバについて

W I N D O W S 9 x / M E / N T / 2000 / X P 向に汎用 I / O 読み書き用 D L L が添付されています。

基本的には当 D L L (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 W I N D O W S** 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。



W I N D O W S 3.1 : W i n 3 1 フォルダ以下に格納されており、 V B (2.0) で利用できます。 C , C + + の場合は当 D L L 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、 D O S 同様に直接制御で対応できます。

W I N D O W S 9 x : W i n 9 x フォルダ以下に格納されており、 V B (4.0/5.0) で利用 (or M E) 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 D L L 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 D O S 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。

W I N D O W S N T : W i n N T フォルダ以下に格納されており、 V B (5.0) で利用できます。 (4.0) ブロック I / O 命令、割り込みもサポートされています。

N T 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 1 6 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 /

W I N D O W S 2000 : W i n 2 K フォルダ以下に格納されており、 V B (6.0) で利用できます。 (or X P) なおドライバ (W D M) 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 D L L 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】 W D M ドライバの性格から割り込みは使用できません。

詳細は ¥ M S C I E N C E ¥ W I N 2 K ¥ D O C 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 以降では不要)

汎用リソース情報取得関数 `MS_P C I . D L L` / WINDOWS9x・ME・NT 用 について

ユーザプログラムから本 P C I 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 D L L) 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 D L L & ドライバが必要です。使用手順は、

【1】P C I バス上のボードの検出

Int GetPciDevice (WORD VendelD, WORD DeviceID, WORD nNum, WORD Flag, WORD *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

P C I 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 = 1 3 F D H (マイクロサイエンス社 P C I ボード共通)、
デバイス ID = 1 0 B H (MDA-781PCI)。

【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

P C I 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 P C I インターフェース素子 9 0 5 0 (P L X 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 1 0 H : P C I インターフェース素子 9 0 5 0 で使用。
1 4 H : 未使用
1 8 H : I / O マップレジスタのベースアドレス B A S E 1 (3 - 4 項)。
1 C H : 未使用
2 0 H : 未使用
2 4 H : 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。



【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 VendorID, WORD DeviceID, WORD nNum, WORD Flag, WORD _far *magic)	
引数	VendorID :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。)

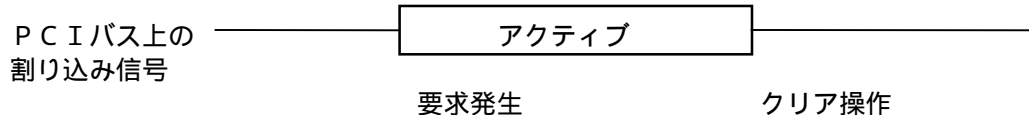
Int ReadPciByte (WORD magic , WORD reg , BYTE _far *data)	
引数	magic : 【1】GetPciDevice で得られた マジック番号、 *data : データ取得先ポインタ、 reg : PCI コンフィギュレーション空間ヘッダ領域のレジスタ番号。
戻り値	0:成功、 -1:失敗

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。

添付の汎用デバイスドライバ（WINDOWS 9x / NT用）では、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。

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。

同じく添付の本ボード専用ハンドラ/デバイスドライバ（WINDOWS 9x / NT用）では割り込みによるDAデータ転送、または外部イベント対応処理が標準でサポートされています。（割り込み不使用でもマルチスレッドでポーリング/DAデータ転送できます。）

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 - 4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。

4-5. Quick Basicサンプル(MS-DOS)

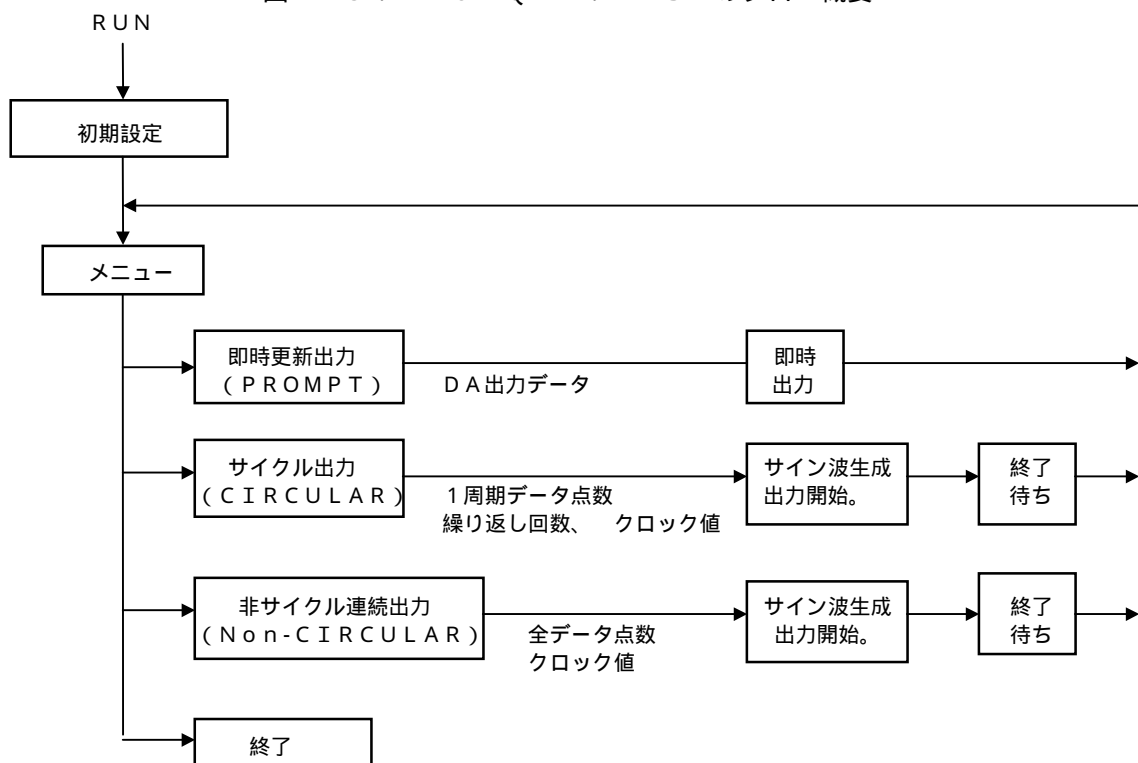
DOS上のBASIC文だけで記述したサンプルソフト781QB1.BASがあります。

781QB1.BAS

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“781QB1.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.“781QB1.BAS”のフロ-概要



【ESC】キイで動作中止、メニューに戻る

4-6. Cのサンプル (MS-DOS)

単純な2チャンネル即時更新DA動作、クロック同期・サイクルモードによる自動波形出力動作、同・複数ボードの同期運転、およびクロック同期・非サイクルモードによる大量データ連続出力、等のサンプルプログラム・ソースが用意されています。以下にアルゴリズムの概要を記します。具体的にはソースリスト【* . C】を御参照ください。

図4-6A. マニュアル (即時更新DA出力) 動作【GPP781.C】

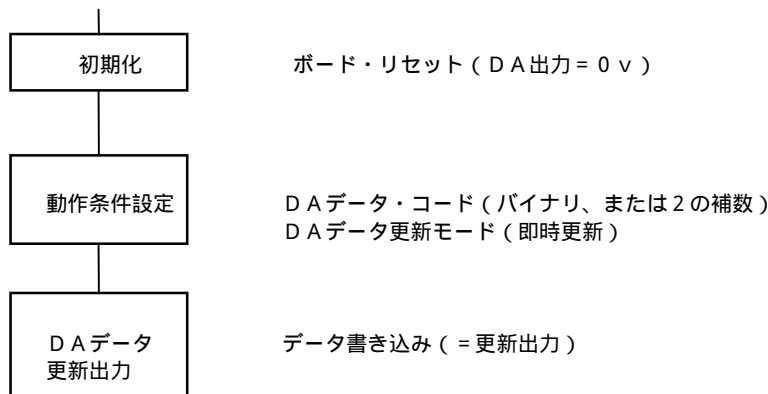
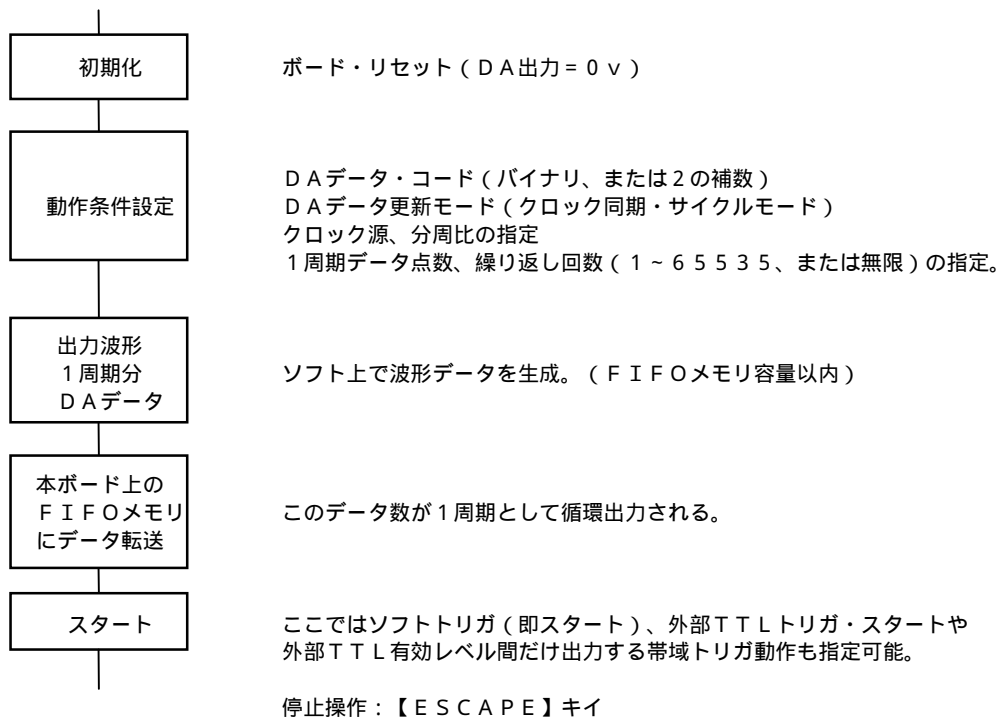


図4-6B. クロック同期・サイクルモード動作【CYC781.C】

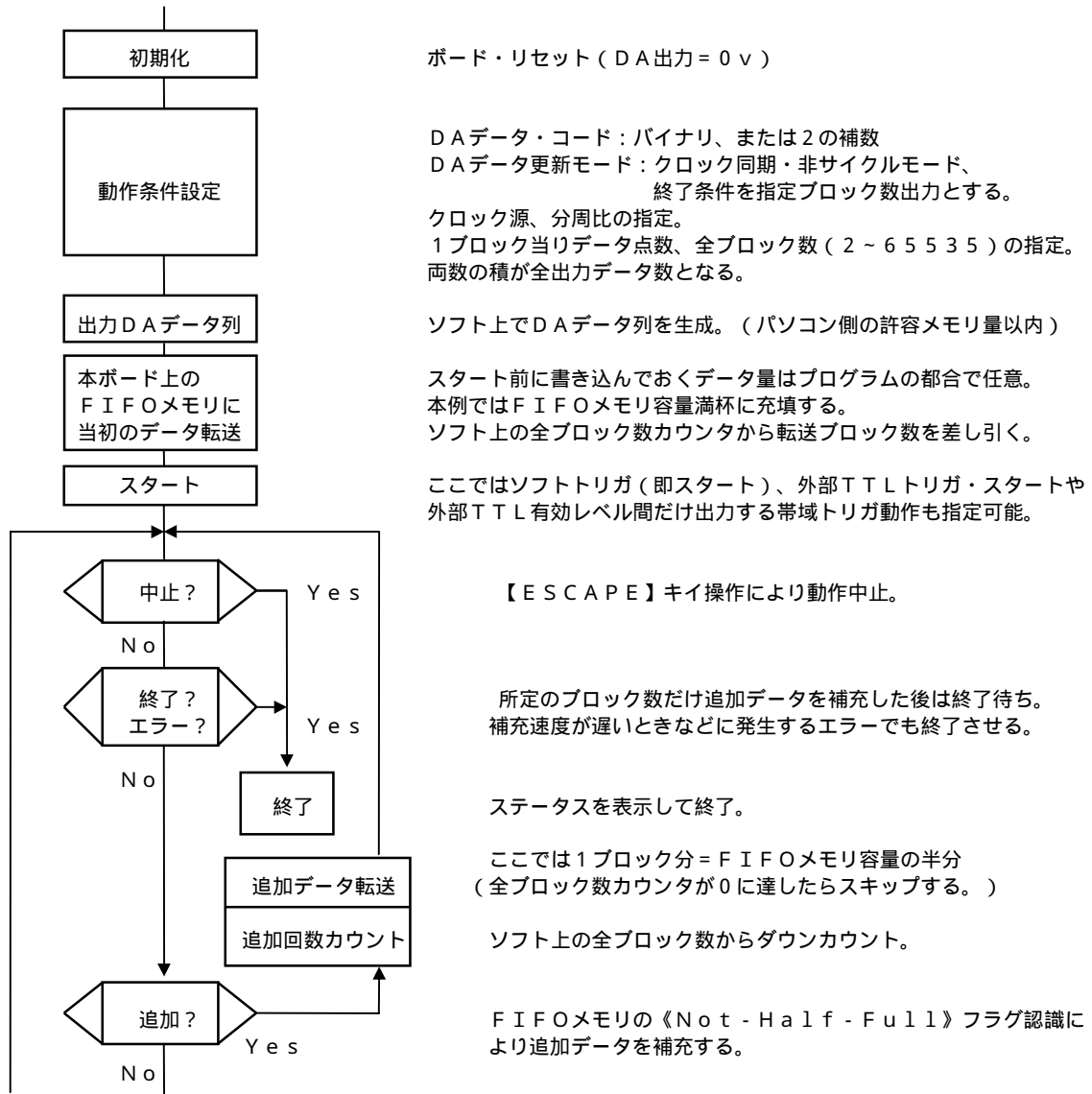


複数ボードの同期運転サンプルプログラムMSCYC781.Cもあります。(3-19項も参照)

連続出力すべきDAデータ列が本ボード上のFIFOメモリ容量を超えるとときは、最初にメモリ容量分だけ書き込んでおき、以後はメモリ充満状態を示すフラグによる割り込み、またはフラグを監視（ポーリング）することにより逐次追加データを転送する非サイクル・モードを利用します。

図4-6C. クロック同期・非サイクルモード動作

【ポーリングによるブロック転送例：BKPOL781.C】



ポイント1

非サイクル・モードではブロック転送 (OUTSB) 命令が最も速いデータ転送手段です。1ブロックのデータ点数が数百点以上あれば、プログラム内容にもよりますが、一般的にはDMAより速くなります。出力データ点数が固定的な場合は当方法が最適です。

ポイント2

当サンプルでは1ブロック = FIFOメモリ容量の半分 (標準なら512語) としています。これによりアルゴリズムが簡単になります。もし、出力データ点数に端数があるときは最終ブロックの余剰部分を最終データで埋めるような方法もあるでしょう。

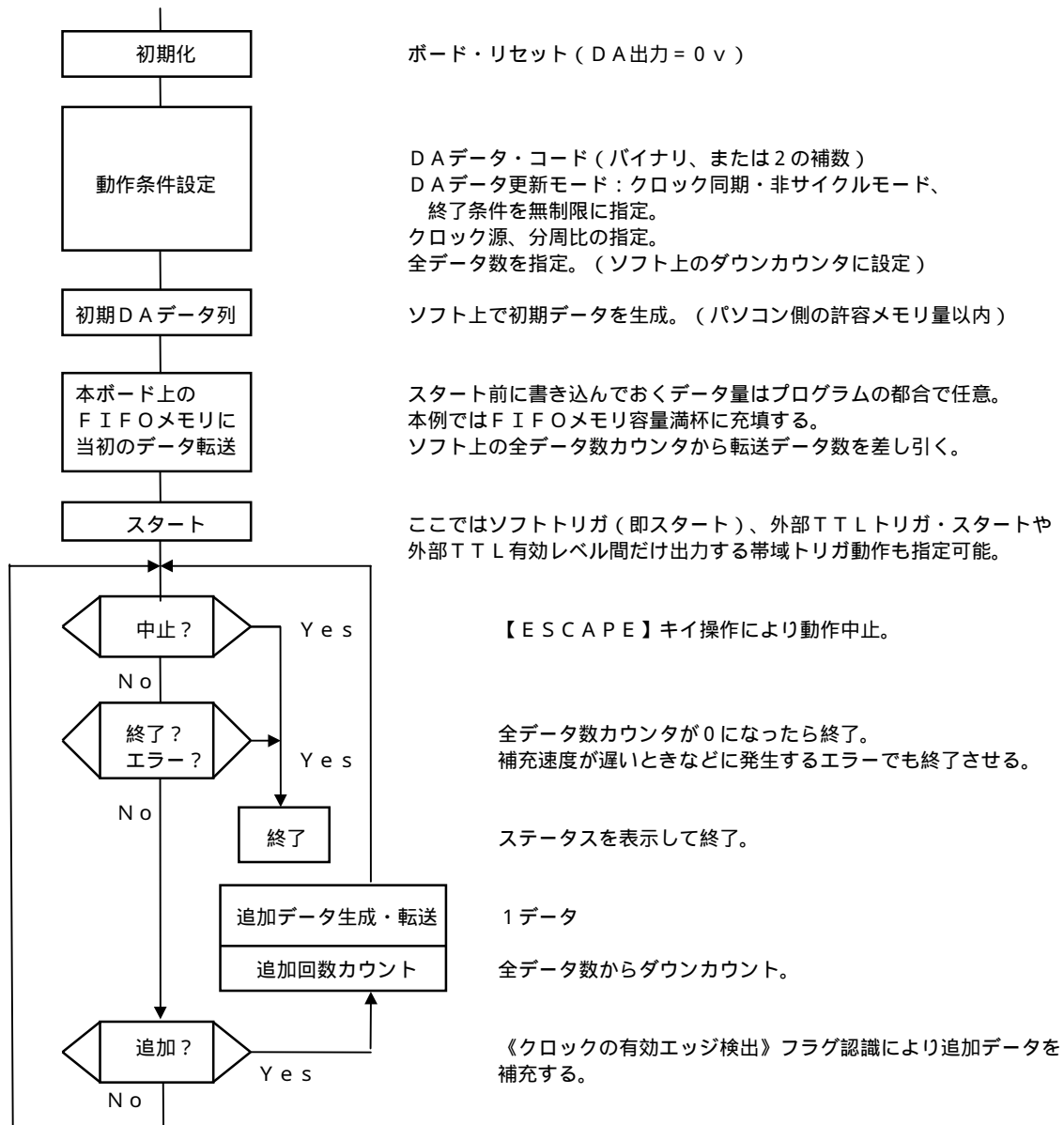
割り込み

追加データ転送を《Not-Half-Full》フラグ割り込みの処理ルーチンで行う例が【BKINT781.C】です。

制御ループ中で次のデータが（演算・処理等により）生成されながら出力される場合は1データずつ転送することになるでしょう。

図4-6D. クロック同期・非サイクルモード動作

【ポーリングによる1データずつ転送例：POL781.C】



ポイント1

当サンプルではFIFOメモリを使用したクロック同期・非サイクルモードを使用していますが、初期に複数のDAデータが確定していないとき（これが普通？）はFIFOメモリを使用しない即時更新モードも全と同じアルゴリズムが使えるでしょう。

割り込み

追加データ転送を《クロック》割り込みの処理ルーチンで行う例が【INT781.C】です。

第5章. MS - DOSハンドラ

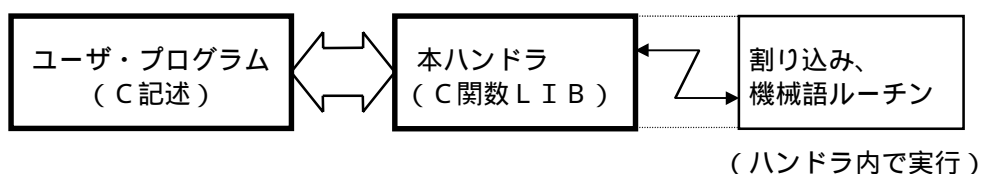
本機MDA-781PCIで簡単に波形出力することのできる汎用ハンドラ(LIB)です。
MS-C, TURBO-C, BORLAND-C等でリンクできます。本機の基本的な機能が
関数化されており、ユーザは御自身の記述するメインルーチンの中から自由に呼び出して使用する
ことができます。【最大8チャンネル=ボード数の同期運転も可能】

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98NX機)
 本体メモリ量 : 本体メモリ640KB以上(EMS、XMSメモリ使用可能)
- OS・言語 : MS-DOS(3.1以上)、および
 MS-C(7.0)、Visual-C++、
 TURBO-C++(4.0)、または
 BORLAND-C++(3.1以上)
- 供給メディア : 本ボード添付のサンプルディスク内。
- 対応DAボード : MDA-781PCI 【注】FIFOメモリ32K語以内
- チャンネル数 : 最大8(=ボード数/マスタスレーブ接続)【注】
- サンプリング速度 : クロック同期・サイクルモードでは2MHz、
 非サイクルモードではチャンネル数、データ数、CPU速度に依存。
 【例】1ch/32K語/Pen(400MHz):約800KHz
- DA出力データ数 : 標準メモリ、または拡張メモリ(EMS, XMS)上で確保可能な容量
 (バイト数÷2)

- 【注】動作の制約等 : 複数ボード使用時のFIFOメモリは全ボード同一容量とする。
 " " " の波形データ点数は全チャンネル同数とする。
 " " " のスタート/ストップは全チャンネル同時となる。
- 非サイクルモードでの追加データ転送はポーリングまたは割り込みに
 限る。なお、本ハンドラ自体に自動繰り返し動作機能はない。
- 複数ボード使用時、マスタのクロック出力を直結できるスレーブ数は
 使用する最高クロック値の制限を受ける。その原因は波形が鈍るため
 で、2MHz:5枚、1MHzのとき7枚。
 なおマスタからクロックを供給されたスレーブのクロック出力を他の
 スレーブに供給可能。(通過遅れ:約250ns)

図5-1. プログラム構造



5 - 2 . 使用準備

ハードウェアの準備

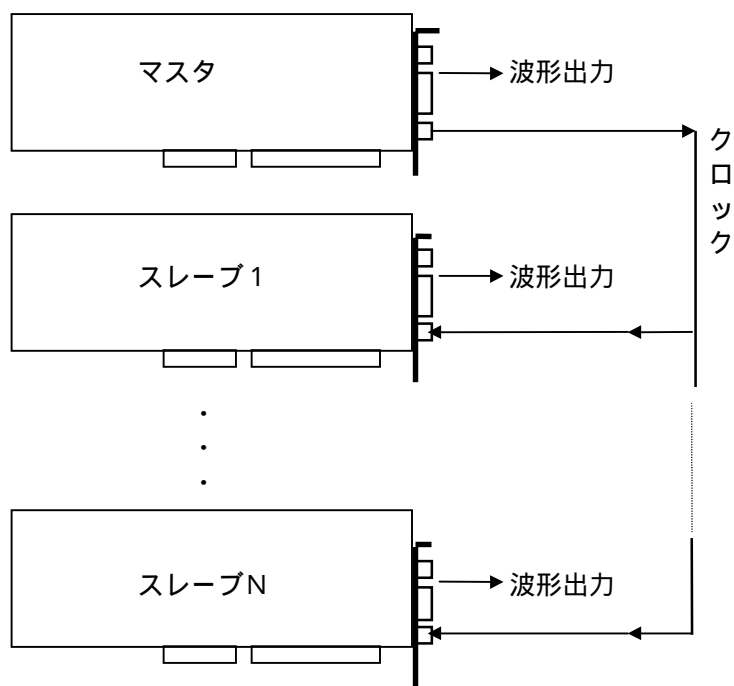
本ハンドラは最大 8 枚の MDA - 7 8 1 P C I を運転することができます。 各ボードの設定、接続は以下のとおりです。

1 枚だけ使用するときにはボード上のスイッチ S W - B N の設定を出荷時の値 (= 0) とする。

複数枚使用するときには、2 枚目以降 (スレーブ各機) ボード上のスイッチ S W - B N の設定を 1、2、3、…… 7 とし、1 枚目 (マスタボード) のクロック出力を各スレーブボードのクロック源入力に接続する。(3 - 1 9 項 . 参照)

マスタ機は外部クロック源入力、外部トリガを使用することもできる。
各ボードの出力範囲は任意で、一致している必要はない。

図 1 - 1 C . マスタスレーブ接続 (最大 7 スレーブ)



5-3. ユーザプログラム記述

以下の手順で記述します。 具体的にはサンプル・ソースに習ってください。

(1) 初期化を行う。【`da__open_dasys()`】

ここではボードの存在やPnPで設定されたリソース本ハンドラが認識すると同時にボードリセット、その他、ハンドラ内の参照テーブルやデータバッファを初期化する。

(2) パソコン本体の(標準, EMS, or XMS)メモリ上にDAデータバッファを確保し、出力すべきDAデータを書き込む。

(3) サンプリング(DA出力)条件を設定する。

サンプリング・モード関連 【`da__set_sampmode()`】
 トリガ関連 【`da__set_trigger()`】
 クロック関連 【`da__set_clock()`】
 内部基準電圧の設定 【`da__set_scale()`】

(4) サンプリング開始。【`da__start_samp()`】

(5) 動作状態(ステータス)取得。【`da__get_status()`】

(6) サンプリング停止。【`da__stop_samp()`】

表5-3. 関数一覧

関数名	機能・動作	主なパラメータ等
<code>da__open_dasys</code>	ボード、本ハンドラの初期化	使用ボード数
<code>da__reset_board</code>	全ボードのリセット	
<code>da__set_trigger</code>	トリガ(スタート)関連の設定	トリガ源、極性
<code>da__set_sampmode</code>	波形データ転送モードの設定	バッファ領域
<code>da__set_exclk</code>	オプション、外部クロックの設定	外部クロック周波数
<code>da__set_clock</code>	クロック源、クロック値の設定	クロック源/分周比
<code>da__set_scale</code>	スケール変数の設定	振幅制御データ
<code>da__start_samp</code>	波形出力スタート(32KB以内)	DAデータ数/更新モード
<code>da__start_samp_h</code>	波形出力スタート(32KB超過)	" " " "
<code>da__start_user</code>	波形出力スタート(ハンドラのバッファ不使用)	" " " "
<code>da__write_direct</code>	ボード上のFIFOにデータを直接書き込む	
<code>da__out_prompt</code>	DA即時更新出力	DAデータ
<code>da__get_status</code>	ボード・ステータスの取得	ボード・ステータス
<code>da__stop_samp</code>	波形出力の強制停止	
<code>da__close_dasys</code>	本ハンドラの終了	
<code>da__out_aux</code>	汎用デジタル(ラッチ)出力	更新データ
<code>da__inp_aux</code>	汎用デジタル(現在値)入力	
<code>da__write_exmem</code>	拡張メモリにDAデータを書き込む	データ数/ブロック数/バッファ
<code>da__write_exmem_h</code>	拡張メモリにDAデータを書き込む	" " "
<code>da__set_datacode</code>	DAデータコードの選択	バイナリ/2の補数
<code>da__sel_outsig</code>	出力信号(クロック/SYNC)切り替え	
<code>da__onkey_quit</code>	KEY操作によるトリガ待ち停止	(CNTL)+(PAUSE)
<code>da__onkey_trg</code>	KEY操作による強制トリガ(出力開始)	ESC/SPACE/ENTER
<code>da__set_timeout</code>	トリガ待ちタイムアウト時間の設定	秒単位
<code>da__clear_flags</code>	波形出力条件の再設定(前回と同一)	
<code>da__onintr_func</code>	割り込みで実行するユーザ関数の指定	割り込み要因
<code>da__get_libver</code>	本ハンドラのバージョン情報取得	

5 - 4 . 関数仕様、エラーコード

初期化、動作条件設定、スタート、ストップ、ステータス取得、汎用割り込み処理、等々、各関数は波形出力を実現する基本機能単位となっています。また、各関数は自身の性格から適切な実行手順があります。（前5 - 3項・参照）

【1】本ハンドラのオープン、および初期化

<code>int da_open_dasys(int num_board, long *fifo_size)</code>	
<code>num_board</code> <code>fifo_size</code>	MDA-781PCIボードの使用枚数。 検出した（マスタボード上の）FIFOメモリ容量 / Byte
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5 - 4参照）
機能・動作	<p>プラグアンドプレイで設定された（各ボードの）I/Oアドレス、および割り込みレベルを認識すると同時に、ボードリセット、その他、ハンドラ内の参照テーブルやデータバッファ等を初期化する。</p> <p>FIFOメモリ容量検出はマスタボードのみ</p> <p>当関数実行直後の各ボードは0V出力、また即時更新モード。 スケール変数を設定【7】すれば、いつでも1データ即時更新出力【9】が可能。</p> <p>波形出力するときは【3】～【7】の条件設定後、 【8】or【8】' or【8】' でスタート。</p>

【2】全ボード（MDA-781PCI）のリセット

<code>int da_reset_board(void)</code>	
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5 - 4参照）
機能・動作	<p>全ての制御レジスタは初期状態に戻る。（汎用デジタル出力は変化しない） アナログ出力は0Vに、スケール変数は0に戻る。 なお【1】初期化実行直後には必要ない。（本関数の動作は【1】初期化の一部）</p>

【3】トリガ（スタート方法）関連の設定

<code>int da_set_trigger(int trig_source, int trig_pol)</code>	
<code>trig_source</code>	トリガ源。 / 0：即トリガ、 1：外部トリガ（入力待ち）
<code>trig_pol</code>	外部トリガ極性。 / 0：負エッジ（ ）、 1：正エッジ（ ）、 2：負レベル、 3：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5 - 4参照）
機能・動作	<p>波形出力スタート条件を設定する。 外部トリガ極性でレベルを指定したときは指定されたレベル間だけ波形出力が行われる帯域動作となる。（3 - 13項参照）</p>

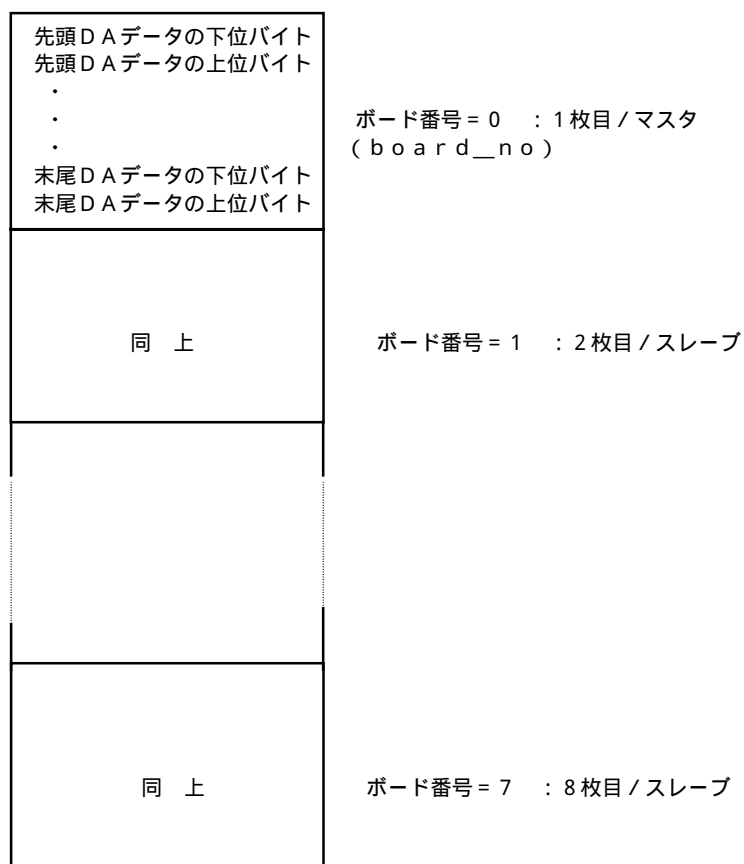
《注》当設定はスタート関数【8】【8】'【8】' 実行時の動作に反映される。

【4】波形データ転送モードの設定

<pre>int da__set__samppmode(int trs__mode[], int buf__area, int intr__sw)</pre>	
trs__mode[0]	波形データ追加転送方法。：不使用（本ボードではI/O命令に限る）
trs__mode[1]	波形データ追加転送の起動要因（割り込み使用時は要求発信要因）。 / 0 : Not-Full、 1 : Not-Half Full
buf__area	波形データ格納バッファ。 / 0 : 標準（コンベンショナル）メモリ、 1 : EMS、 2 : XMS
intr__sw	波形データ追加転送に割り込み使用の有無。 / 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5-4参照）
機能・動作	波形データ追加転送（非サイクルモード）、および波形データバッファに関する条件を設定する。ここで割り込みを不使用とすれば追加転送の起動はポーリングによって検出・実行される。なお、ここで割り込みを使用しない設定のときは、割り込みを【23】ユーザ関数の起動に使用することもできる。

《注》当設定はスタート関数【8】【8】'【8】" 実行時の動作に反映される。

図5-4. 波形データ格納バッファ内のデータ配置



【5】オプション、または外部クロック源周波数値の設定

<code>int da_set_exclk(ULONG exclk_freq)</code>	
<code>exclk_freq</code>	オプション、または外部クロック源の周波数 (Hz 単位)
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 5 - 4 参照)
機能・動作	オプションまたは外部クロック源を使用する場合、その周波数 (Hz 単位) を設定する。 後記【6】 <code>da_set_clock</code> で分周比によるクロック指定、またはボードに標準搭載の内部クロック源 (10MHz、8.192MHz) を使用するときは必要ない。

《注》当設定はスタート関数【8】【8】'【8】" 実行時の動作に反映される。

【6】サンプリング (波形出力同期) クロック値の設定

<code>int da_set_clock(int clk_source, int set_mode, int *time_unit, ULONG *clk_period)</code>	
<code>clk_source</code>	クロック源指定。 / 0: 内部クロック源 0 (10.000MHz) 1: 内部クロック源 1 (8.192MHz) 2: 内部クロック源 2 (オプション) 3: 内部クロック源 3 (未使用) 4: 外部クロック源 (有効極性:) 5: 外部クロック源 (有効極性:)
<code>set_mode</code>	サンプリング・クロックの指定方法。 / 0: クロック周期の値、 1: 分周比
<code>*time_unit</code>	クロック周期の単位。 / 0: s, 1: ms, 2: μ s, 3: ns
<code>*clk_period</code>	クロック周期の値、または分周比。
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 5 - 4 参照)
機能・動作	サンプリング・クロック値を設定する。 なお設定できない値が指定されたときは設定可能な長い方の近似値が設定される。 またクロック源を 2, 4, 5 から選び、クロックを周期の値で指定するときは前記【5】 <code>da_set_exclk</code> でクロック源の周波数値を定義しておく必要がある。

《注》当設定はスタート関数【8】【8】'【8】" 実行時の動作に反映される。

【7】スケール変数の設定

<code>int da_set_scale(int board_no, int scale)</code>	
<code>board_no</code>	更新設定する対象ボードの番号 / 0: 1 枚目 (マスタ)、 1 ~ 7: スレーブ
<code>scale</code>	スケール変数データ。 / 整数 0 ~ 4095 (digit)
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 5 - 4 参照)
機能・動作	即時更新モードのときは新データで即時更新されます。 本関数は【1】初期化または【2】リセット操作の後、いつでも単独で実行可能です。 クロック同期更新動作 (= 波形出力) 中に本関数を実行することは振幅制御を意味します。 但し、クロックに同期して更新されるので最大 1 クロック時間の遅れが生じる。

【8】サンプリング（波形出力）スタート 《波形データバッファが32K語以内のとき》

<pre>int da_start_samp(WORD num_data, WORD num_block, int upd_mode, WORD *bufptr)</pre>	
num_data	1ブロック当りのデータ数。(サイクルモードでは1周期分=FIFO容量-1以内)
num_block	出力ブロック数。(num_data)×(num_block)=総出力データ数。
upd_mode	更新モード。/0:サイクル(停止操作まで無限)、1:サイクル(xを出力) 2:非サイクル(xを出力)
*bufptr	波形データバッファのポインタ。
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5-4参照)
機能・動作	<p>サンプリングをスタート(外部トリガ指定のときはトリガ待ち)させます。 引数は複数チャンネル(複数ボード)使用時も全チャンネル共通です。</p> <p>データバッファ(パソコン側)は全チャンネル分で、図5-4の構成です。なお、 【4】da_set_sampmodeでbuf_areaを拡張(EMS,XMS)メモリに指定したとき ~ は全て無視されます。(【16】で設定される)</p>

【8】‘ サンプリング（波形出力）スタート 《波形データバッファが32K語を超えるととき》

<pre>int da_start_samp_h(WORD num_data, WORD num_block, int upd_mode, WORD __huge *bufptr)</pre>	
num_data	1ブロック当りのデータ数。(サイクルモードでは1周期分=FIFO容量-1以内)
num_block	出力ブロック数。(num_data)×(num_block)=総出力データ数。
upd_mode	更新モード。/0:サイクル(停止操作まで無限)、1:サイクル(xを出力) 2:非サイクル(xを出力)
*bufptr	波形データバッファのポインタ。
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5-4参照)
機能・動作	<p>サンプリングをスタート(外部トリガ指定のときはトリガ待ち)させます。 引数は複数チャンネル(複数ボード)使用時も全チャンネル共通です。</p> <p>データバッファ(パソコン側)は全チャンネル分で、図5-4の構成です。なお、 【4】da_set_sampmodeでbuf_areaを拡張(EMS,XMS)メモリに指定したとき ~ は全て無視されます。(【16】’で設定される)</p>

《注1》 サンプリング（波形出力）スタート関数【8】と【8】‘ は総出力データ量の大きさにより使い分けてください。機能は全く同一です。

《注2》 サイクルモード時の1ブロック当りデータ数()はFIFOメモリ容量-1以内で、指定回数()または停止操作まで繰り返し出力します。一方、非サイクルモード時は総出力データ(x)を1回だけ出力して動作終了となります。

【8】” サンプリング（波形出力）スタート 《ユーザプログラム側から直接データ転送するとき》

<code>int da_start_user(WORD num_data, WORD num_block, upd_mode)</code>	
<code>num_data</code>	1ブロック当りのデータ数。
<code>num_block</code>	出力ブロック数。 $(num_data) \times (num_block) = \text{総出力データ数}$ 。
<code>upd_mode</code>	更新モード。/ 0 : 非サイクル（ <code>x</code> を出力）、1 : 非サイクル（停止操作まで無限）
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5-4参照）
機能・動作	サンプリングをスタート（外部トリガ指定のときはトリガ待ち）させます。 引数は複数チャンネル（複数ボード）使用時も全チャンネル共通です。

《注1》 当スタート関数は本ハンドラのデータバッファを使用せず、ユーザプログラム側から直接、ボードのFIFOメモリにデータ転送する関数（次【9】）と合わせて使用する。特に注意することは当スタート関数を呼ぶ前に最初の出力データを（【9】で）転送しておく必要がある。スタート以後はステータス関数【11】によりFIFOメモリの充満状態を監視して適時（必要なら）追加データを転送する、また、【4】波形データ転送モード設定関数で`intr_sw = 0`（データ追加転送に割り込み不使用）としていたときは割り込み発生時に実行すべきユーザ関数【23】を利用することもできる。

《注2》 なお当スタート関数は動作の性格から、機能を非サイクルモードに限っている。

【9】対ボード（FIFOメモリ）直接データ転送 /// 【8】”と組み合わせ使用、上記《注1》参照。

<code>int da_write_directfifo(int board_no, long num_data, WORD *bufptr)</code>	
<code>board_no</code> <code>num_data</code> <code>bufptr</code>	波形データ書き込む対象ボードの番号 / 0 : 1枚目（マスタ）、1～7 : スレーブ ここで書き込むデータ数 / （前記【8】”の同名引き数とは意味が違う） 波形データバッファのポインタ
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5-4参照）
機能・動作	本ハンドラのデータバッファを使用せずユーザプログラム側から直接、ボード上のFIFOメモリにデータを転送します。高速のブロック転送命令を使用するので、1度書き込むデータ点数はFIFOメモリ容量の半分以下とするのが合理的です。

【10】マニュアル（即時更新）DA出力

<code>int da_out_prompt(int board_no, int upd_prompt)</code>	
<code>board_no</code> <code>upd_prompt</code>	更新出力する対象ボードの番号 / 0 : 1枚目（マスタ）、1～7 : スレーブ DA出力データ。/ 整数0～4095（digit）
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5-4参照）
機能・動作	新データで即時更新されます。 本関数は【1】初期化【7】スケール変数設定の後、単独で実行可能です。 波形出力中に本関数が実行されると、その時点で波形出力が停止し、続いて本関数の結果が得られます。乗算出力モードのときは、この過程で出力に（瞬間的な）乱れが生じることがありますから御注意ください。

【11】ボードステータスの取得

<pre>int da_get_status(int board_no, WORD *cnt0_data, WORD *cnt1_block, int *eos, int *intr_req, int *upd_err, int status[])</pre>	
board_no	ステータス取得する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ
cnt0_data cnt1_block eos intr_req upd_err status[0] status[1]	(1 ブロック中の) 出力データ数カウンタの現在値。 / カウンタ # 0。 出力ブロック数カウンタの現在値。 / カウンタ # 1。 D A 出力終了。 / 1 : 終了、 0 : 出力中または開始前。 割り込み要求発生。 / 1 : 発生、 0 : 未発生 / (3 - 1 7 項) 更新出力エラー。 / 1 : 発生、 0 : 未発生 / (3 - 1 4 項) ボードステータス 1。 / 1 バイト生データ (3 - 1 4 項) ボードステータス 2。 / 未使用
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 5 - 4 参照)
機能・動作	は動作進行状況、または終了の検出に使用します。 なお は一旦セットされると【1】初期化、【2】ボードリセット、または 【22】フラグクリア関数でリセットするまで保持されます。

【12】サンプリング (波形出力) 動作の強制停止

<pre>int da_stop_samp(void)</pre>	
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 5 - 4 参照)
機能・動作	全チャンネル (ボード) のサンプリング動作を強制的に停止します。 各チャンネルの出力は最後に更新された値を保持しています。 この後、スタート関数により再スタート (続きを実行) することができます。 F I F O メモリ内の残りデータはそのまま待機状態です。 この後、新たな波形出力を行うときは再度【3】 ~ 【7】の手順を踏むか、同一条件で再設定の関数【22】を実行してからスタート【8】 or 【8】' or 【8】' ' させます。

【13】本ハンドラの終了

<pre>int da_close_dasys(void)</pre>	
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 5 - 4 参照)
機能・動作	本ハンドラの終了。 確保したメモリ領域の開放等、開始前の状態に戻す。 【注】D A 出力の最終状態を維持するためにボードのリセットは行いません。

以上【1】 ~ 【13】が主要関数です。 以下の【14】 ~ は補助的な関数です。

【14】汎用デジタル（ラッチ）出力更新

<code>int da_out_aux(int board_no, int out_data)</code>	
<code>board_no</code>	更新設定する対象ボードの番号 / 0 : 1 枚目（マスタ）、 1 ~ 7 : スレーブ
<code>out_data</code>	1ビット汎用デジタル（ラッチ）出力データ。
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 5 - 4 参照）
機能・動作	TTLレベルの1ビット汎用デジタル（ラッチ）出力を更新する。 当出力ポートは電源投入（ハードウェア・リセット）時 = 0 Hとなるが、 【1】初期設定、【2】ボード・リセット、【13】ハンドラ終了などを実行しても 変化しない。

【15】汎用デジタル（現在値）入力

<code>int da_inp_aux(int board_no)</code>	
<code>board_no</code>	更新設定する対象ボードの番号 / 0 : 1 枚目（マスタ）、 1 ~ 7 : スレーブ
戻り値	正常終了時： = 1ビット汎用デジタル（現在値）入力データ エラー時： エラーコード（負の値 / エラーコード表 5 - 4 参照）
機能・動作	TTLレベルの1ビット汎用デジタル入力（現在状態）を読み込む。

【16】拡張メモリにDAデータを書き込む（hugeポインタを使用しない場合）

<code>int da_write_exmem(int board_no, WORD num_data, WORD num_block, int upd_mode, WORD *bufptr)</code>	
<code>board_no</code> <code>num_data</code> <code>num_block</code> <code>upd_mode</code>	書き込み対象チャンネル（ボード番号：0 ~ 7） 1ブロック当りのデータ数 全ブロック数 DA出力更新モード / 0 : サイクル（停止操作まで無限）、 1 : サイクル（ x を出力）、 2 : 非サイクル（ x を出力）
<code>bufptr</code>	バッファのポインタ
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 5 - 4 参照）
機能・動作	標準メモリに入りきれない大容量のデータをDA出力するときは、当関数を使用して 拡張メモリに書き込んで使用します。 総データ数を標準メモリ上のバッファに入る程度の複数ブロックに等分し、1ブロック ずつバッファに書き込んでから当関数を実行する（ブロック数だけ繰り返す）。 なお、別データを再書き込みするときは本ハンドラを一旦終了、再オープンして行う。

【16】‘拡張メモリにDAデータを書き込む(hugeポインタを使用する場合)’

<pre>int da_write_exmem(int board_no, WORD num_data, WORD num_block, int upd_mode, WORD huge *bufptr)</pre>	
board_no num_data num_block upd_mode	書き込み対象チャンネル(ボード番号: 0~7) 1ブロック当りのデータ数 全ブロック数 DA出力更新モード / 0: サイクル(停止操作まで無限)、 1: サイクル(x を出力)、 2: 非サイクル(x を出力)
bufptr	バッファのポインタ
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5-4参照)
機能・動作	標準メモリに入りきれない大容量のデータをDA出力するときは、当関数を使用して 拡張メモリに書き込んで使用します。 総データ数を標準メモリ上のバッファに入る程度の複数ブロックに等分し、1ブロック ずつバッファに書き込んでから当関数を実行する(ブロック数だけ繰り返す)。 なお、別データを再書き込みするときは本ハンドラを一旦終了、再オープンして行う。

【17】DAデータコードの指定

<pre>int da_set_datacode(int data_code)</pre>	
data_code	DAデータ・コードの選択。 / 0: バイナリ、 1: 2の補数
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5-4参照)
機能・動作	本ボードMDA-781PCIの入力コード(DAデータの型)を指定する。 / 全機共通

【18】クロック / SYNC・外部出力の切り替え

<pre>int da_sel_outsig(int out_sig[])</pre>	
out_sig[]	クロック / SYNC・出力選択。 / 0: クロック、 1: SYNC []内はボード番号0~7、複数ボード使用時の[0]はマスタなので必ず0とする。
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5-4参照)
機能・動作	外部へのクロック信号出力(コネクタCN2のCLK-OUT)を必要ならSYNC 信号(各1ブロック・データの開始タイミング)に切り替えることができる。 但し、 複数ボード使用時のマスタはクロックを出力しなくてはならないので、必ずクロック側 に設定すること。

【19】 《BREAKキイ押下によるトリガ待ち停止》の設定 / 解除

<code>int da_onkey_quit(int set_mode)</code>	
<code>set_mode</code>	動作の指定。 / 0 : 解除、 1 : 設定
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 5 - 4 参照)
機能・動作	BREAK (= CTRL + PAUSE) キイの押下により、トリガ待ちを停止させる。 サンプリング (波形出力) 中であればサンプリングを強制停止させる。 スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、人為的にトリガ待ちループから抜け出る手段となる。

【20】 《有効キイ押下による強制的・即トリガ動作》の設定 / 解除

<code>int da_onkey_trg(int act_key)</code>	
<code>act_key</code>	有効な動作key。 / 0 : 解除、 1 : ESC、 2 : SPACE、 3 : ENTER
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 5 - 4 参照)
機能・動作	有効な動作キイの押下により、強制的に即トリガ動作 (波形出力開始) させる。 スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、人為的にトリガ待ちループから抜け出る手段となる。

【21】 《(トリガ待ち)タイムアウト》の設定 / 解除

<code>int da_set_timeout(WORD set_time)</code>	
<code>set_time</code>	タイムアウトまでの時間 (単位 = 秒)。 / 1 ~ 65535、 0 : 解除、
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 5 - 4 参照)
機能・動作	スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、当時間が経過するとトリガ待ちを停止する。

【22】フラグクリア、およびサンプリング条件の再設定

<code>int da_clear_flags(set_mode)</code>	
<code>set_mode</code>	DA出力条件再設定の有無。 / 0 : フラグクリアのみ。 1 : フラグクリア後、DA出力条件を再設定する。
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 5 - 4 参照)
機能・動作	各フラグをクリアし (3 - 14 項)、サンプリング条件を前回と同一に再設定する。 DA出力条件の再設定 : 具体的には【3】～【7】の関数を実行する。 フラグクリアにはFIFOメモリのリセットが含まれており、停止時の残りデータを破棄して (停止時の出力値を起点に) 新データ群を定義・出力することができる。

【23】 割り込み発生時に実行するユーザ関数の設定

<pre>int da_onint_func(int intr_source, void interrupt far *func)</pre>	
intr_source	割り込み要求の発生要因。 / 0 : クロック (波形出力の各データ更新タイミング) 1 : 外部割り込み入力 () 2 : 外部割り込み入力 () 3 : トリガ発生 4 : 指定ブロック数の更新出力終了 5 : 各ブロックの先頭データ更新出力時 6 : (FIFO) Not Full 7 : (FIFO) Not Half - Full
*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 5 - 4 参照)
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。 但し、当指定は【4】波形データ転送モードで、割り込みを不使用に設定した場合のみ可能です。 なお、【1】初期化実行の直後は可能な状態です。

【24】 本ハンドラのバージョン取得

<pre>int da_get_libver(int ver)</pre>	
ver	0 : 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) 1 : 戻り値は (メジャー・バージョン番号) 2 : 戻り値は (マイナー・バージョン番号)
戻り値	正常終了時 : 本ハンドラのバージョン番号 エラー時 : エラーコード (負の値 / エラーコード表 5 - 4 参照)
機能・動作	本ハンドラのバージョン情報を得る 例えばバージョンが 1.01 の場合、本関数を ver = 0 として実行すると 戻り値は 0 x 1 0 1 となります。

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5-4A. 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。（未装着）	da_open_adsys
- 2	指定枚数分のボードを認識できない。	da_open_adsys
- 3	ID不一致。	da_open_adsys
- 4	ボード（FIFOメモリ）にアクセスできない。	da_open_adsys
- 5	当ハンドラ初期化時のメモリ不足。	da_open_adsys
- 6	初期化（da_open_dasys）が未実行。	
- 7	スタート関数【8】【8】'【8】" 実行前の条件設定不足。	da_start_samp
- 10	ボード数、またはボード番号指定エラー。	num_board, board_no
- 11	トリガ源の指定が不適当。	trig_source
- 12	トリガ極性の指定が不適当。	trig_pol
- 13	波形データ追加転送方法の指定が不適当。	trs_mode[0]
- 14	波形データ追加転送の起動要因指定が不適当。	trs_mode[1]
- 15	波形データ格納バッファの指定が不適当。	buf_area
- 16	割り込み使用の有無指定が不適当。	intr_sw
- 17	クロック源の指定が不適当。	clk_source
- 18	クロック値の指定が不適当。	set_mode
- 19	クロック周期単位の指定が不適当。	time_unit
- 20	クロック周期値の指定が不適当。	clk_period
- 21	1ブロック当りのデータ数、ブロック数値の指定が不適当。	num_data, num_block
- 22	更新モードの指定が不適当。	upd_mode
- 23	更新モード=サイクル時に割り込みは不可。	
- 24	FIFOメモリ容量を超えるデータ数の書き込み。	
- 25	データ数が64KBを超える。	da_start_samp()
- 26	スケール変数の指定値が不適当。	scale
- 30	データ格納バッファに拡張メモリが指定されていない。	da_write_exmem()
- 31	拡張メモリへのデータ書き込みは全て終了済み。	同上。
- 32	指定ボード用のデータは既に拡張メモリに書き込み済み。	同上。
- 33	前ボード用のデータ書き込み未完了。（若い番号ボードから順に！）	同上。
- 34	DAデータ・コード指定が不適当。	data_code
- 36	DA出力停止KEYは【CNTL+PAUSE】KEYに設定済み。	
- 37	トリガ待ち解除・強制実行（スタート）KEYは設定済み。	
- 38	DA出力（サンプリング）に割り込みを使用している。	da_onint_func
- 39	割り込み要求源の指定が不適当。	intr_source
- 40	ユーザ作成関数が存在しない。	func
- 41	外部クロック源周波数の指定なしでクロック周期を指定した。	clk_period
- 42	指定したクロック周期または分周比が大きすぎる。	clk_period
- 50	DA出力（サンプリング）終了。	
- 51	【CNTL+PAUSE】KEYで停止した。	
- 52	タイムアウト時間経過によりトリガ待ちを停止した。	
- 53	クロック同期更新エラー発生。（クロック値が速すぎるときなど）	

表5-4B. 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
- 6 0	E M S 関連：常駐していない。	
- 6 1	E M S 関連：物理ページのエン트리取得エラー。	
- 6 2	E M S 関連：物理ページ数が不足。（4 ページ必要）	
- 6 3	E M S 関連：ページ・フレームのアドレス取得エラー。	
- 6 4	E M S 関連：未アロケート・ページ数取得エラー。	
- 6 5	E M S 関連：メモリ不足エラー。	
- 6 6	E M S 関連：ページの割り当てとハンドルの取得エラー。	
- 6 7	E M S 関連：マップエラー、E M S データ転送エラー（割り込み時）。	
- 6 8	E M S 関連：既に開放されている。	
- 7 0	X M S 関連：常駐していない。	
- 7 1	X M S 関連：ドライバ・エン트리アドレス取得エラー。	
- 7 2	X M S 関連：フリーメモリ取得エラー。	
- 7 3	X M S 関連：メモリ不足エラー。	
- 7 4	X M S 関連：メモリ確保エラー。	
- 7 5	X M S 関連：転送バッファ確保エラー。	
- 7 6	X M S 関連：データ転送エラー。	
- 7 7	X M S 関連：マップエラー、X M S データ転送エラー（割り込み時）。	
- 7 8	X M S 関連：既に開放されている。	

第6章. WINDOWSハンドラ

本DAボード/MDA-781PCIをVC、VC++、VB等で簡単に使用することのできるWINDOWS95・98・ME・NT・2000・XP用ハンドラ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、ボード内蔵クロックによる自動出力：波形出力や外部イベントに同期した出力機能もサポートされています。ユーザは自身の記述するメインルーチン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 : WINDOWS95・98・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DAボード : MDA-781PCI 【注】FIFOメモリ32K語以内。

チャンネル数 : 最大8（=ボード数/マスタスレーブ接続）

サンプリング速度 : クロック同期・サイクルモードでは2MHz、
非サイクルモードではチャンネル数、データ数、CPU速度に依存。

【例】1ch / 32K語 / Pen（400MHz）：約800KHz

出力データ点数 : 拡張メモリ空容量（1語 = 2バイト）

割り込み : WINDOWS2000・XPでは使用不可。

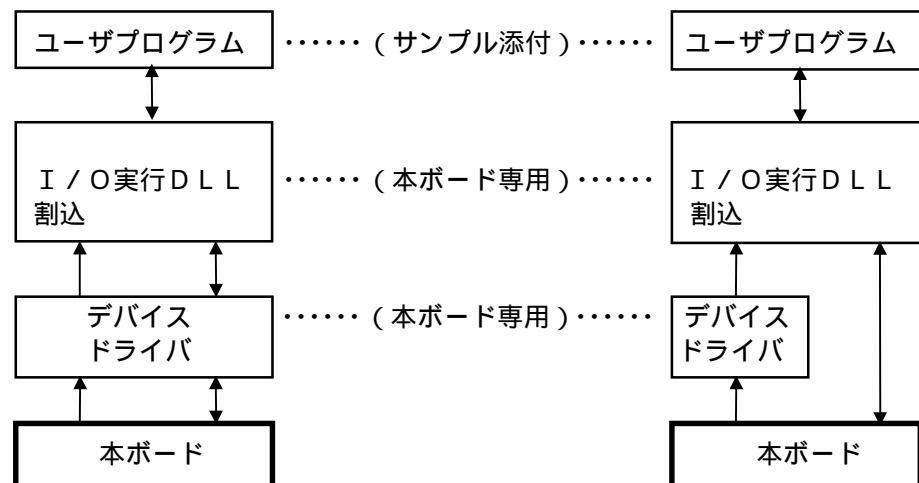
95・98・MEでの使用は任意。（要リソース取得 / 1-5項）

LabVIEW等で利用可能（サンプルv iソース参照）

図6-1. ソフトウェア構造

（WINDOWS NT/2000/XP の場合）

（WINDOWS95・98・ME の場合）



【注】動作の制約等： 複数ボード使用時のFIFOメモリは全ボード同一容量とする。

“ ” “ ” の波形データ点数は全チャンネル同数とする。

“ ” “ ” のスタート/ストップは全チャンネル同時。

非サイクルモードでの追加データ転送はポーリングまたは割り込みに限る。なお本ハンドラ自体に自動繰り返し動作機能はない。

複数ボード使用時、マスタのクロック出力を直結できるスレーブ数は使用する最高クロック値の制限を受ける。原因は波形が鈍るためで、2MHz：5枚、1MHzのとき7枚。

なおマスタからクロックを供給されたスレーブのクロック出力を他のスレーブに供給する結線も可能。（通過遅れ：約250ns）

6-2. 使用準備

ハードウェアの準備

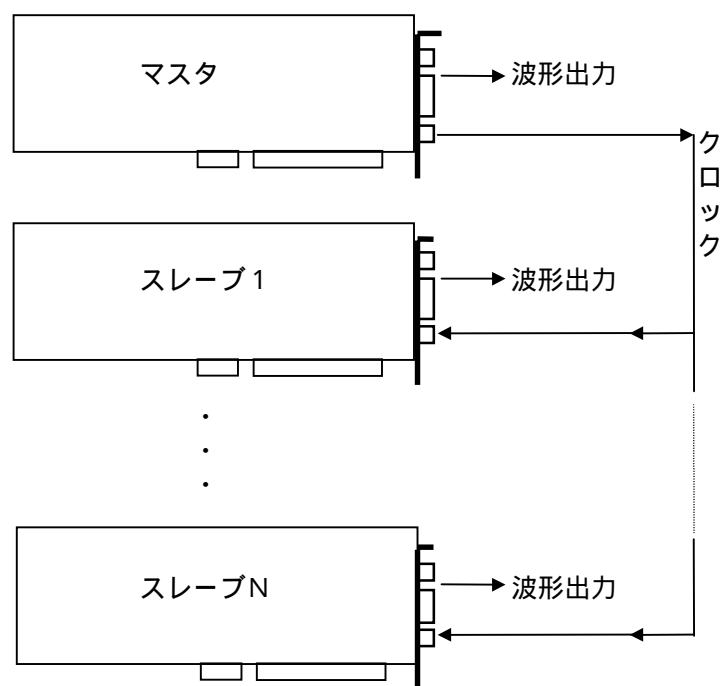
本ハンドラは最大8枚のMDA-781PCIを運転することができます。各ボードの設定、接続は以下のとおりです。

1枚だけ使用するときにはボード上のスイッチSW-BNの設定を出荷時の値（=0）とする。

複数枚使用するときには、2枚目以降（スレーブ各機）ボード上のスイッチSW-BNの設定を1、2、3、……7とし、1枚目（マスタボード）のクロック出力を各スレーブボードのクロック源入力に接続する。（3-19項、参照）

マスタ機は外部クロック源入力、外部トリガを使用することもできる。
各ボードの出力範囲は任意で、一致している必要はない。

図1-1C. マスタスレーブ接続（最大7スレーブ）



ソフトウェアの準備

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。
次に本関数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。

WINDOWS 95・98・MEの場合

- (1) ¥WINDOWS フォルダに“H781_95 .DLL”をコピーする。
- (2) ¥WINDOWS¥SYSTEM フォルダに“D781_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“H781_NT .DLL”をコピーする。
- (2) ¥WINDOWS¥SYSTEM32¥DRIVERS フォルダに “D781_NT0.SYS”
～ “D781_NT7.SYS” をコピーする。
- (3) 任意のフォルダにユーティリティ “D781Reg.exe” をコピーする。

【注】ファイル名中の末尾文字 はバージョン変更時に追加(初期は無し)されます。

《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(D781Reg)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
ストリに書き込み、(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¥MDA781 以下)を使用できます。

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となります。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) 初期化を行う。【DA_Open_DASys()】

ここではボードの存在やPnPで設定されたリソース本ハンドラが認識すると同時にボードリセット、その他、ハンドラ内の参照テーブルやデータバッファを初期化する。

(2) パソコン本体メモリ上にデータバッファを確保し、出力すべきDAデータを書き込む。
サンプリング・モード関連 【DA_Set_SampMode()】

(3) サンプリング(DA出力)条件を設定する。

トリガ関連	【DA_Set_Trigger()】
クロック関連	【DA_Set_Clock()】
スケール変数の設定	【DA_Set_Scale()】
波形データの設定	【DA_Write_DllBuf()】

(4) サンプリング開始。【DA_Start_Samp()】

(5) 動作状態(ステータス)取得。【DA_Get_Status()】

(6) サンプリング停止。【DA_Stop_Samp()】

表 6-3. 関数一覧

関数名	機能・動作	主なパラメータ等
DA_Open_DASys	ボード、本ハンドラの初期化	使用ボード数
DA_Reset_Board	全ボードのリセット	
DA_Set_Trigger	トリガ(スタート)関連の設定	トリガ源、極性
DA_Set_SampMode	波形データ転送モードの設定	バッファ領域
DA_Set_Exclk	オプション、外部クロックの設定	外部クロック周波数
DA_Set_Clock	クロック源、クロック値の設定	クロック源/分周比
DA_Set_Scale	スケール変数の設定	スケール変数データ
DA_Write_DLLBuf	対DLL管理バッファへのデータ転送	波形出力データ
DA_write_DirectFifo	対ボードFIFOへの直接データ転送	波形出力データ
DA_Start_Samp	波形出力スタート(32KB以内)	DAデータ数/更新モード
DA_Out_Prompt	DA即時更新出力	DAデータ
DA_Get_Status	ボード・ステータスの取得	ボード・ステータス
DA_Stop_Samp	波形出力の強制停止	
DA_Close_DASys	本ハンドラの終了	
DA_Out_Aux	汎用デジタル(ラッチ)出力	更新データ
DA_Inp_Aux	汎用デジタル(現在値)入力	
DA_Set_Datacode	DAデータコードの選択	バイナリ/2の補数
DA_Sel_Outsig	(クロック/SYNC)出力切り替え	
DA_Clear_Flags	波形出力条件の再設定(前回と同一)	
DA_Get_Libver	本ハンドラのバージョン情報取得	

【注】 WINDOWS 2000用に限り、関数名が異なります。

すなわち関数名中、文字“DA”の直後に対象ボード名(3桁番号)が含まれます。

例: DA781_Open_DASys

重要 1

本ハンドラを構成する関数は一定手順、または特定の組み合わせでのみ有効に動作します。基本手順については本項先頭に記しましたとおり、本ハンドラをオープン、関数【3】～【7】でサンプリング出力条件を設定したら後はスタート（トリガ待ち）関数【10】を呼び出すだけです。

- <例1> サイクルモード動作のときは必ず、
【4】で波形データ格納バッファを本ハンドラDLLの管理領域（buf_area = 0）とし、関数【8】で出力波形データを同バッファに事前転送してからスタート【10】してください。なおユーザプログラムの管理領域から各ボードに直接データ転送する形はとれません。

非サイクルモード動作のとき注意することは関数【4】で指定する波形データ格納バッファと関数【8】【9】による波形データ設定です。

- <例2> 【4】で波形データ格納バッファを本ハンドラDLLの管理領域（buf_area = 0）として完全自動出力させる場合は、“スタート操作”【10】実行前に【8】出力波形データを同バッファに事前転送しておきます。

このとき、“波形データ転送の起動要因”を《Not-HALF-FULL》に設定してある場合は“FIFOメモリ容量”の半分単位でブロック転送、また《Not-Full》に設定した場合は“1データ”単位で転送する。

以上はバックグラウンドで自動実行され、指定データ点数に達すると自動停止する。

- <例3> 【4】で波形データ格納バッファをユーザプログラムの管理領域（buf_area = 1）として半自動出力させる場合は、“スタート操作”【10】でクロック自動出力は開始されますが、当ハンドラによるボード上のFIFOメモリへの追加データ自動補給は行なわれません。ユーザプログラム側で【12】ステータスをポーリングするか、割り込みイベント（【4】intr_swで要因指定）によりFIFOメモリの充満状態を検出し、【9】逐次・追加データ補給を行う必要があります。

また前記<例2>と違い、“スタート操作”【10】前に出力データの先頭部分を書き込んでおく（【9】）必要もあるでしょう。

この手順では連続サンプリングが“ストップ操作”まで無限に実行できますから、制御ループを含むアプリケーションに適します。

重要 2

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各種メッセージを受け取ることができませんから、この条件を踏まえた利用に制限されます。

6-4 . 関数仕様、エラーコード

初期化、動作条件設定、スタート、ストップ、ステータス取得、汎用割り込みイベント等、各関数は波形出力を実現する基本機能単位となっています。また、各関数は自身の性格から適切な実行手順があります。（前6-3項参照）

【注】 WINDOWS 2000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：DA781_Open_DASys

【1】本ハンドラのオープン、および初期化

<pre>int DA_Open_DASys (HWND Owner , int num_board , int *fifo_size)</pre>	
num_board fifo_size	MDA-781PCIボードの使用枚数。 検出した（マスタボード上の）FIFOメモリ容量 / Byte
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6-4参照）
機能・動作	<p>プラグアンドプレイで設定された（各ボードの）I/Oアドレス、および割り込みレベルを認識すると同時に、ボードリセット、その他、ハンドラ内の参照テーブルやデータバッファ等を初期化する。</p> <p>FIFOメモリ容量検出はマスタボードのみ</p> <p>当関数実行直後の各ボードは0V出力、また即時更新モード。 スケール変数を設定【7】すれば、いつでも1データ即時更新出力【11】が可能。 波形出力するときは【3】～【8or9】の条件設定後、【10】でスタート。</p>

【2】全ボード（MDA-781PCI）のリセット

<pre>int DA_Reset_Board (void)</pre>	
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6-4参照）
機能・動作	<p>全ての制御レジスタは初期状態に戻る。（汎用デジタル出力は変化しない） アナログ出力は0V、スケール変数は0に戻る。 なお【1】初期化実行直後には必要ない。（本関数の動作は【1】初期化の一部）</p>

【3】トリガ（スタート方法）関連の条件設定

<pre>int DA_Set_Trigger (int trig_source , int trig_pol)</pre>	
trig_source	トリガ源。 / 0：即トリガ（即スタート）、 1：外部トリガ（入力待ち）
trig_pol	外部トリガ極性。 / 0：負エッジ（ ）、 1：正エッジ（ ）、 2：負レベル、 3：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6-4参照）
機能・動作	<p>波形出力スタート条件を設定する。 外部トリガ極性でレベルを指定したときは指定されたレベル間だけ波形出力が行われる帯域動作となる。（3-13項参照）</p>

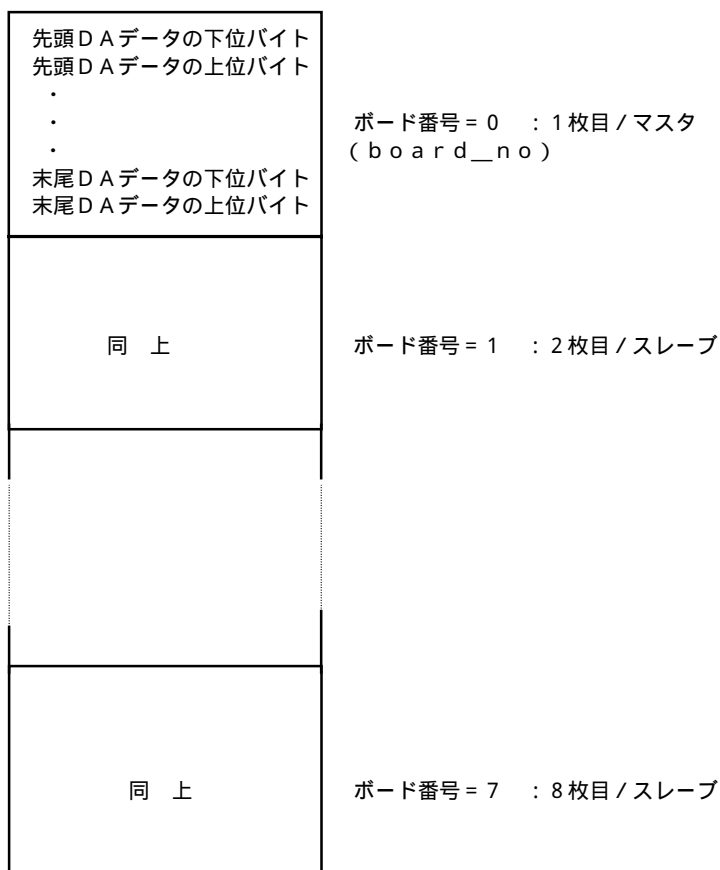
《注》当設定はスタート関数【10】実行時の動作に反映される。

【4】波形データ転送モード、および割り込みイベント発生要因の設定

<pre>int DA_Set_SampMode(int trs_mode, int buf_area, int intr_sw)</pre>	
trs_mode	波形データ追加転送の起動要因（割り込みは使用されない）。 / 0: Not-Full、 1: Not-Half Full
buf_area	波形データ格納バッファ指定。 / 0: 当ハンドラ内DLLの管理するバッファ、 1: ユーザプログラムの管理するバッファ
intr_sw	割り込みイベント発生要因指定。 / 0: 割り込み不使用、 1: 連続サンプリング・クロック 2: 外部割り込み入力信号（ ）、 3: 外部割り込み入力信号（ ）、 4: トリガ発生、 5: 指定ブロック数出力終了、 6: 1ブロック出力開始、 7: Not-Full状態 8: Not-Half Full状態
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6-4参照）
機能・動作	波形データ追加転送（非サイクルモード）、および波形データバッファに関する条件を設定する。 割り込みイベントは任意に使用できる。（要割り込みリソース取得）

《注》当設定はスタート関数【10】 実行時の動作に反映される。

図6-4. 波形データ格納バッファ内のデータ配置



【5】オプション、または外部クロック源周波数値の設定

<code>int DA_Set_Exclk(int exclk_freq)</code>	
<code>exclk_freq</code>	オプション、または外部クロック源の周波数 (Hz 単位)
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 6-4 参照)
機能・動作	オプションまたは外部クロック源を使用する場合、その周波数 (Hz 単位) を設定する。 後記【6】 <code>DA_Set_Clock</code> で分周比によるクロック指定、またはボードに標準搭載の内部クロック源 (10MHz、8.192MHz) を使用するときは必要ない。

《注》当設定はスタート関数【10】実行時の動作に反映される。

【6】サンプリング (波形出力同期) クロック値の設定

<code>int DA_Set_Clock(int clk_source, int set_mode, int *time_unit, int *clk_period)</code>	
<code>clk_source</code>	クロック源指定。 / 0: 内部クロック源0 (10.000MHz) 1: 内部クロック源1 (8.192MHz) 2: 内部クロック源2 (オプション) 3: 内部クロック源3 (未使用) 4: 外部クロック源 (有効極性:) 5: 外部クロック源 (有効極性:)
<code>set_mode</code>	サンプリング・クロックの指定方法。 / 0: クロック周期の値、 1: 分周比
<code>*time_unit</code>	クロック周期の単位。 / 0: s, 1: ms, 2: μ s, 3: ns
<code>*clk_period</code>	クロック周期の値、または分周比。
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 6-4 参照)
機能・動作	サンプリング・クロック値を設定する。 なお設定できない値が指定されたときは設定可能な長い方の近似値が設定される。 またクロック源を 2, 4, 5 から選び、クロックを周期の値で指定するときは前記【5】 <code>DA_Set_Exclk</code> でクロック源の周波数値を定義しておく必要がある。

《注》当設定はスタート関数【10】実行時の動作に反映される。

【7】スケール変数の設定

<code>int DA_Set_Scale(int board_no, int scale)</code>	
<code>board_no</code>	更新設定する対象ボードの番号 / 0: 1枚目 (マスタ)、 1~7: スレーブ
<code>scale</code>	スケール変数データ。 / 整数 0~4095 (digit)
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 6-4 参照)
機能・動作	即時更新モードのときは新データで即時更新されます。 本関数は【1】初期化または【2】リセット操作の後、いつでも単独で実行可能です。 クロック同期更新動作 (波形出力) 中に本関数を実行することは振幅制御を意味します。 但し、クロックに同期して更新されるので最大 1クロック時間の遅れが生じる。

【8】対DLL管理バッファへのデータ転送 /// 使用法は前6-3項《例1》《例2》参照 ///

<pre>int DA_Write_DLLBuf(int board_no, int num_samp, int num_data, WORD *bufptr)</pre>	
board_no num_samp num_data bufptr	波形データ書き込む対象ボードの番号 / 0 : 1枚目(マスタ)、 1 ~ 7 : スレーブ 対象ボード用の総データ数 (各ボード共通とする。) ここで書き込むデータ数 / (後記【10】の同名引き数と同一値にすると管理し易い。) 波形データバッファのポインタ
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6-4参照)
機能・動作	総データ数を同サイズ (2 ~ 65535) の複数ブロックに等分し、当関数で1ブロック ずつDLLの管理するバッファに書き込む (ブロック数だけ繰り返す) とよい。 なお、別データを再書き込みするときは本ハンドラを一旦終了、再オープンして行う。

【9】対ボード (FIFOメモリ) 直接データ転送 /// 使用法は前6-3項《例3》参照 ///

<pre>int DA_Write_DirectFifo(int board_no, int num_data, WORD *bufptr)</pre>	
board_no num_data bufptr	波形データ書き込む対象ボードの番号 / 0 : 1枚目(マスタ)、 1 ~ 7 : スレーブ ここで書き込むデータ数 / (後記【10】の同名引き数とは意味が違う) 波形データバッファのポインタ
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6-4参照)
機能・動作	本ハンドラのデータバッファを使用せずユーザプログラム側から直接、ボード上の FIFOメモリにデータを転送します。 高速のブロック転送命令を使用するので、 1度に書き込むデータ点数はFIFOメモリ容量の半分以下とするのが合理的です。

【10】サンプリング (波形出力) スタート

<pre>int DA_Start_Samp(WORD num_data, WORD num_block, int upd_mode)</pre>	
num_data num_block upd_mode	1ブロック当りのデータ数。(サイクルモードでは1周期分 = FIFO容量 - 1以内) 出力ブロック数。(num_data) × (num_block) = 総出力データ数。 更新モード / 0 : サイクル (を停止操作まで無限)、 1 : サイクル (x を出力) 2 : 非サイクル (x を出力)
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6-4参照)
機能・動作	サンプリングをスタート (外部トリガ指定のときはトリガ待ち) させます。 引数 は複数チャンネル (複数ボード) 使用時も全チャンネル共通です。 《注1》前記【4】で波形データバッファをユーザプログラムの管理領域に設定した場合、 引数 はボード上のカウンタに渡され、出力データ数を制御する。 引数 の範囲は2 ~ 65535、 引数 の範囲は1 ~ 65535 (3 - 10項参照)。 《注2》前記【4】で波形データバッファをユーザプログラムの管理領域に設定した場合、 upd_mode = 2に限り、停止操作【13】まで無限にサンプリング出力 する非サイクルモード。

【11】マニュアル（即時更新）DA 出力

<code>int DA_Out_Prompt(int board_no, int upd_prompt)</code>	
<code>board_no</code> <code>upd_prompt</code>	更新出力する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ DA 出力データ。 / 整数 0 ~ 4095 (digit)
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 6 - 4 参照)
機能・動作	新データで即時更新されます。 本関数は【1】初期化【7】スケール変数設定の後、単独で実行可能です。 波形出力中に本関数が実行されると、その時点で波形出力が停止し、続いて本関数の結果が得られます。乗算出力モードのときは、この過程で出力に (瞬間的な) 乱れが生じることがありますから御注意ください。

【12】ボードステータスの取得

<code>int DA_Get_Status(int board_no, WORD *cnt0_data, WORD *cnt1_block, int *eos, int *intr_req, int *upd_err, int status[])</code>	
<code>board_no</code>	ステータス取得する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ
<code>cnt0_data</code> <code>cnt1_block</code> <code>eos</code> <code>intr_req</code> <code>upd_err</code> <code>status[0]</code> <code>status[1]</code>	(1 ブロック中の) 出力データ数カウンタの現在値。 / カウンタ # 0。 出力ブロック数カウンタの現在値。 / カウンタ # 1。 DA 出力終了。 / 1 : 終了、 0 : 出力中または開始前。 割り込み要求発生。 / 1 : 発生、 0 : 未発生 / (3 - 17 項) 更新出力エラー。 / 1 : 発生、 0 : 未発生 / (3 - 14 項) ボードステータス 1。 / 1 バイト生データ (3 - 14 項) ボードステータス 2。 / 未使用
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 6 - 4 参照)
機能・動作	は動作進行状況、または終了の検出に使用します。 なお は一旦セットされると【1】初期化、【2】ボードリセット、または【19】フラグクリア関数でリセットするまで保持されます。

【13】サンプリング（波形出力）動作の強制停止

<code>int DA_Stop_Samp(void)</code>	
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 6 - 4 参照)
機能・動作	全チャンネル (ボード) のサンプリング動作を強制的に停止します。 各チャンネルの出力は最後に更新された値を保持しています。この後、スタート関数により再スタート (続きを実行) することができます。 FIFO メモリ内の残りデータはそのまま待機状態です。この後、新たな波形出力を行うときは再度【3】 ~ 【10】の手順を踏みます。

【14】本ハンドラの終了

<code>int DA_Close_DASys(void)</code>	
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6-4参照）
機能・動作	本ハンドラの終了。 確保したメモリ領域の開放等、開始前の状態に戻す。 【注】DA出力最終状態を維持するためにボードのリセットは行いません。

以上【1】～【14】が主要関数です。 以下の【15】～ は補助的な関数です。

【15】汎用デジタル（ラッチ）出力更新

<code>int DA_Out_Aux(int board_no, int out_data)</code>	
<code>board_no</code>	更新設定する対象ボードの番号/0:1枚目（マスタ）、 1～7:スレーブ
<code>out_data</code>	1ビット汎用デジタル（ラッチ）出力データ。
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6-4参照）
機能・動作	TTLレベルの1ビット汎用デジタル（ラッチ）出力を更新する。 当出力ポートは電源投入（ハードウェア・リセット）時=0Hとなるが、 【1】初期設定、【2】ボード・リセット、【14】ハンドラ終了などを実行しても 変化しない。

【16】汎用デジタル（現在値）入力

<code>int DA_Inp_Aux(int board_no)</code>	
<code>board_no</code>	更新設定する対象ボードの番号/0:1枚目（マスタ）、 1～7:スレーブ
戻り値	正常終了時： =1ビット汎用デジタル（現在値）入力データ エラー時： エラーコード（負の値/エラーコード表6-4参照）
機能・動作	TTLレベルの1ビット汎用デジタル入力（現在状態）を読み込む。

【17】DAデータコードの指定

<code>int DA_Set_Datacode(int data_code)</code>	
<code>data_code</code>	DAデータ・コードの選択。 / 0 : バイナリ、 1 : 2の補数
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 6 - 4 参照)
機能・動作	本ボードMDA - 781PCIの入力コード (DAデータの型) を指定する。 / 全機共通

【18】クロック / SYNC・外部出力の切り替え

<code>int DA_Sel_Outsig(int out_sig[])</code>	
<code>out_sig[]</code>	クロック / SYNC・出力選択。 / 0 : クロック、 1 : SYNC [] 内はボード番号 0 ~ 7、複数ボード使用時の [0] はマスタなので必ず 0 とする。
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 6 - 4 参照)
機能・動作	外部へのクロック信号出力 (コネクタ CN2 の CLK - OUT) を必要なら SYNC 信号 (各 1 ブロック・データの開始タイミング) に切り替えることができる。 但し、複数ボード使用時のマスタはクロックを出力しなくてはならないので、必ずクロック側に設定すること。

【19】フラグクリア、およびサンプリング条件の再設定

<code>int DA_Clear_Flags(set_mode)</code>	
<code>set_mode</code>	DA出力条件再設定の有無。 / 0 : フラグクリアのみ。 1 : フラグクリア後、DA出力条件を再設定する。
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 6 - 4 参照)
機能・動作	各フラグをクリアし (3 - 14 項)、サンプリング条件を前回と同一に再設定する。 DA出力条件の再設定 : 具体的には【3】～【7】の関数を実行する。 フラグクリアにはFIFOメモリのリセットが含まれており、停止時の残りデータを破棄して (停止時の出力値を起点に) 新データ群を定義・出力することができる。

【20】本ハンドラのバージョン取得

<code>int DA_Get_Libver(int ver)</code>	
<code>ver</code>	0 : 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) 1 : 戻り値は (メジャー・バージョン番号) 2 : 戻り値は (マイナー・バージョン番号)
戻り値	正常終了時 : 本ハンドラのバージョン番号 エラー時 : エラーコード (負の値 / エラーコード表 6 - 4 参照)
機能・動作	本ハンドラのバージョン情報を得る 例えばバージョンが 1.01 の場合、本関数を <code>ver = 0</code> として実行すると戻り値は <code>0 x 101</code> となります。

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 . 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。（未装着）	DA_Open_DASys
- 2	指定枚数分のボードを認識できない。	DA_Open_DASys
- 3	IDの不一致。	DA_Open_DASys
- 4	ボード（FIFOメモリ）にアクセスできない。	DA_Open_DASys
- 5	ドライバ・ファイルが見つからない。	DA_Open_DASys
- 6	ドライバ・ファイルのバージョンが違う。	DA_Open_DASys
- 7	初期化（DA_Open_DASys）が未実行。	
- 8	スタート関数【10】実行前の条件設定不足。	DA_Start_Samp
- 10	ボード数、またはボード番号指定エラー。	num_board, board_no
- 11	トリガ源の指定が不適当。	trig_source
- 12	トリガ極性の指定が不適当。	trig_pol
- 13	波形データ追加転送方法の指定が不適当。	trs_mode
- 14	波形データ格納バッファの指定が不適当。	buf_area
- 15	割り込み使用の有無指定が不適当。	intr_sw
- 16	クロック源の指定が不適当。	clk_source
- 17	クロック値の指定が不適当。	set_mode
- 18	クロック周期単位の指定が不適当。	time_unit
- 19	クロック周期値の指定が不適当。	clk_period
- 20	スケール変数の指定値が不適当。	scale
- 21	総データ点数の指定値が不適当。	num_samp
- 22	書き込みデータ数の指定が不適当。	num_data
- 23	データ格納バッファの指定が不適当。	bufptr
- 24	更新モードの指定が不適当。	upd_mode
- 25	マニュアル（即時更新）出力DAデータの指定が不適当。	upd_prompt
- 26	DAデータコードの指定が不適当。	data_code
- 27	クロック/SYNC出力選択指定が不適当。	out_sig
- 28	外部クロック源使用時、周期指定なのにクロック源周波数を未指定。	
- 29	クロック周期、または分周比の値が大きすぎる。	clk_period
- 30	FIFO容量を超えるデータ数の書き込み。	
- 31	DLL管理領域内バッファ容量を超えるデータ数の書き込み。	DA_Write_DllBuf
- 32	指定ボード用のデータは既に書き込み済み。	同上。
- 33	前ボード用のデータ書き込み未完了。（若い番号ボードから順に）	同上。
- 40	割り込みが割り当てられていない。	
- 41	割り込み要因設定が不適当。	
- 42	メモリ確保エラー	
- 43	ユーザプログラム領域のバッファを使用する設定になっている。	DA_Write_DLLBuf
- 44	DLL管理領域のバッファを使用する設定になっている。	DA_Write_DirectFifo
- 45	DA波形出力（サンプリング）中。	
- 46	サンプリングは停止している。	

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)
他のボードと重複していないか？ |
| (2) 割り込みレベル | リソースは取得できたか？(1-5項)/通常は割り込み不要。 |
| (3) アナログ出力 | 負荷が重過ぎないか？(2-1項)。
接続機器間のGND電位差にも注意。 |
| (4) トリガ関連 | トリガ待ち状態ではプログラムが止まって見える。 |
| (5) デジタル入出力 | 本ボードのTTL入力(外部割り込み、および汎用ビット)に
接続できる信号源はTTL(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本ボード内
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、

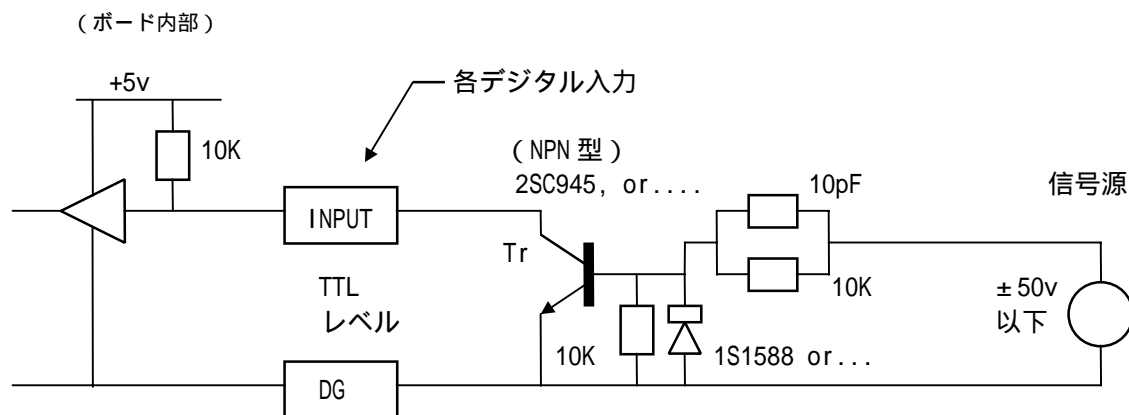
ファンクション・ジェネレータ等の交流信号出力を接続して破損させる例が多いようです。矩形波でも±に振れる信号は接続できません。特に、負側の許容レベル【-0.6V】が低いことに注意してください。

+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・有償にて修理をお願いします。 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. アナログ出力範囲の再調整

動作テスト・確認の方法は【1-6項】のとおりです。同テストから得られた値に出力範囲の変化やオフセットが認められるときは再調整が必要です。アナログ回路は経年・環境変化に対する保守を定期的に行うことが望ましく、夏冬の使用環境・周囲温度に差がある場合は季節単位、通年安定した使用環境の場合は1～2年に1度は校正することが理想的です。

再調整の方法・手順を以下に記しますが、極細のドライバ、デジタル電圧計を必要とし、手順もやや複雑ですから御希望により当社でも（実費で）お請けします。

== 準備 ==

本ボード上の設定は出荷時の状態に加えて、スイッチ“BUSEL”は必ず【IN】側、

スイッチ“BUSEL”：バイポーラ（ $\pm 10\text{V}$ ）で調整するときは【B】側、
ユニポーラ（ $0 \sim 10\text{V}$ ）で調整するときは【U】側に設定。

スイッチABSEL：Aモードで調整するときは【A】側、
Bモードで調整するときは【B】側に設定。

本ボードをパソコン本体または拡張I/Oボックスに装着・インストールします。

当作業の詳細については1-5項に記されています。

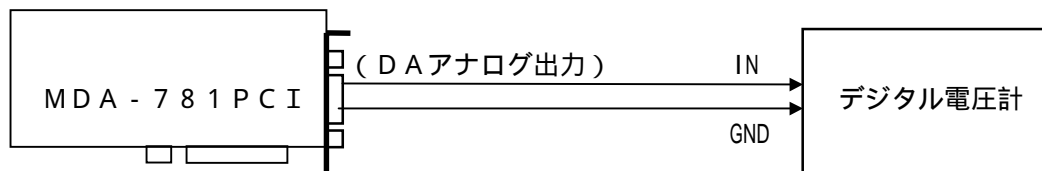
（以上、通常の使用状態）

図1-6のように、本ボードのアナログ出力をデジタル電圧計等に入力接続します。

デジタル入出力の接続は不要です。

以上で準備完了です。電源投入順序は全機器同時、または外部機器を先にパソコン本体を最後に行います。電源切断は逆順序です。

図1-6. 動作確認用の測定機接続



DAアナログ出力：1番ピン
アナロググランド：6番ピン ———— コネクタ CN1（1-3項・参照）

精度（正確度）

本機のアナログ出力回路は高精度の素子を使用しているので調整後は出力モード切り替えによる再調整の必要は全んどありません。

出荷時設定の内部基準電圧使用 $\pm 10\text{V}$ 出力モードでは $\pm 6.2\text{mV}$ 、すなわち内部基準電圧の設定値 $= 10\text{V}$ のとき $\pm 0.062\% \text{FS}$ となるが、基準電圧値を下げるとFSに対する精度は悪化する。

外部基準電圧使用を含むその他の出力モードに切り替えたときは $\pm 8.2\text{mV}$ （再調整で $\pm 6.2\text{mV}$ まで改善可能）、すなわち基準電圧の設定値 $= 10\text{V}$ のとき $\pm 0.082\% \text{FS}$ となるが、基準電圧値を下げるとFSに対する精度は悪化する。

【注】FS = フルスケール（出力範囲幅）

== 操作 ==

電源を投入、MS-DOSシステムを立ち上げます。(WINDOWS95/98のDOS窓でも可能)
再調整に使用するプログラムは試運転でも使用した“781QB1”です。
> 781QB1【ENTER】でプログラムが走り始めます。最初にI/Oベースアドレスの入力を要求されるので1-5項で調査した4桁のHex値を入力すると以下、操作メニューとなります。以後の各DAデータは10進数で設定・表示します。

振幅制御用DACのオフセット調整

メニューから【1. Set DA out Data】を選択し、
バイポーラ(±10V)のときは2048、ユニポーラ(0~10V)のときは0を設定。
次にメニューから【2. DA Set Scale Data】を選択し、0を設定する。
この状態で出力電圧が0VとなるようにトリマTM4を調整する。

(波形生成用)出力DACの調整

メニューから【2. DA Set Scale Data】を選択し、4095を設定する。
次にメニューから【1. Set DA out Data】を選択し、
バイポーラ(±10V)のときは2048、ユニポーラ(0~10V)のときは0を設定。
この状態で出力電圧が0Vとなるようにバイポーラ(±10V)のときはトリマTM1を、
ユニポーラ(0~10V)のときはトリマTM0を調整する。 // オフセット調整 //

メニューから【2. DA Set Scale Data】を選択し、
Aモードのときは4000、Bモードのときは4095を設定。
次にメニューから【1. Set DA out Data】を選択し、
Aモードでバイポーラ(±10V)のときは4048、
Aモードでユニポーラ(0~10V)のときは4000、
Bモードでバイポーラ(±10V)のときは4095、
Bモードでユニポーラ(0~10V)のときは4095を設定する。
この状態で出力電圧が目標値(表7-3/太枠内)となるようにトリマTM2を調整する。
// ゲイン調整 //

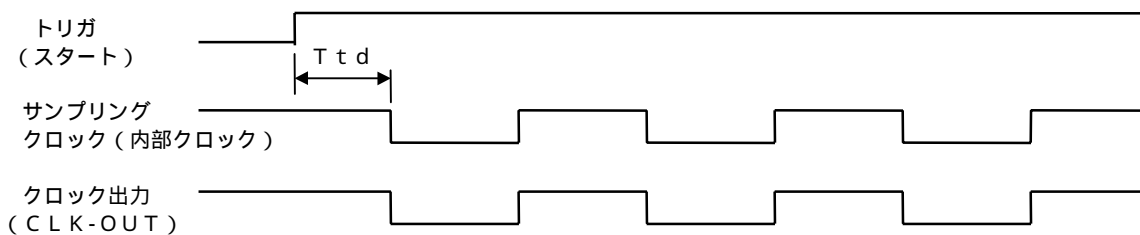
表7-3. 調整対象・目標(スイッチRESELは必ず【IN】側に設定)

出力範囲モード（スイッチA B S E L）			【A】側：Aモード		【B】側：Bモード	
出力極性モード（スイッチB U S E L）			【BI】側 ± 1 0 v	【UN】側 0 ~ 1 0 v	【BI】側 ± 1 0 v	【UN】側 0 ~ 1 0 v
振幅制御用 DACの 調整	オフセット 調整	設定値	2 0 4 8	0	2 0 4 8	0
		目標出力	0 v	0 v	0 v	0 v
		トリマ	T M 4	T M 4	T M 4	T M 4
波形生成用 出力 DACの 調整	オフセット 調整	設定値	2 0 4 8	0	2 0 4 8	0
		目標出力	0 v	0 v	0 v	0 v
		トリマ	T M 1	T M 0	T M 1	T M 0
	ゲイン調整 （スケール）	設定値	4 0 4 8	4 0 0 0	4 0 9 5	4 0 9 5
		目標出力	+ 1 0 v	+ 1 0 v	+9.99512 v	+9.99756 v
トリマ		T M 2	T M 2	T M 2	T M 2	

7-4. 制御信号・タイミング等

クロック入出力

図7-4A. 内部クロック源を使用する場合

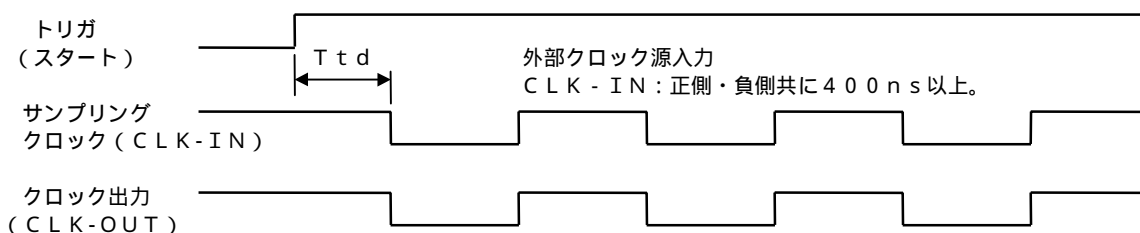


: クロック出力の有効エッジは常に立下り。

T_{td} : トリガ認識から最初のクロック有効エッジまでの最大遅れ時間 (最大 250 ns)

実際のサンプリング実行はクロックの各有効エッジから 125 ns 以内に開始。

図7-4B. 外部クロック源を非分周 (1/1) で使用する場合

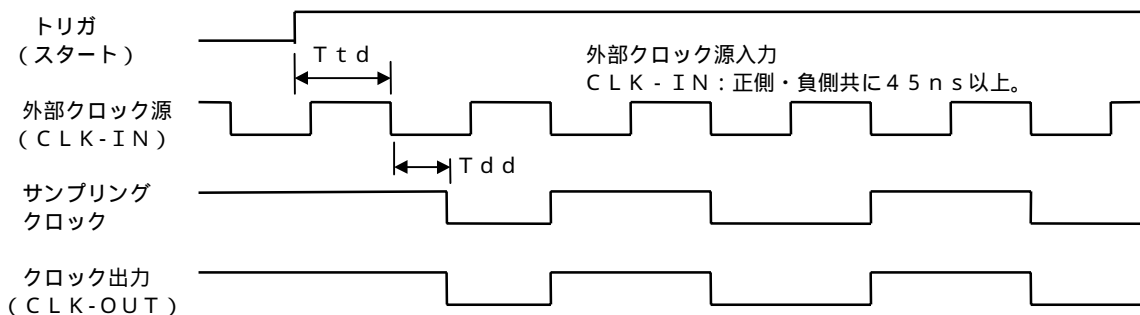


: クロック出力の有効エッジは常に立下り。

T_{td} : トリガ認識から最初のクロック有効エッジまでの最大遅れ時間 (外部クロック源 1 周期)

実際のサンプリング実行はクロックの各有効エッジから 125 ns 以内に開始。

図7-4C. 外部クロック源を (任意に) 分周して使用する場合



: クロック出力の有効エッジは常に立下り。

T_{td} : トリガ認識から最初のクロック有効エッジまでの最大遅れ時間 (外部クロック源 1 周期)

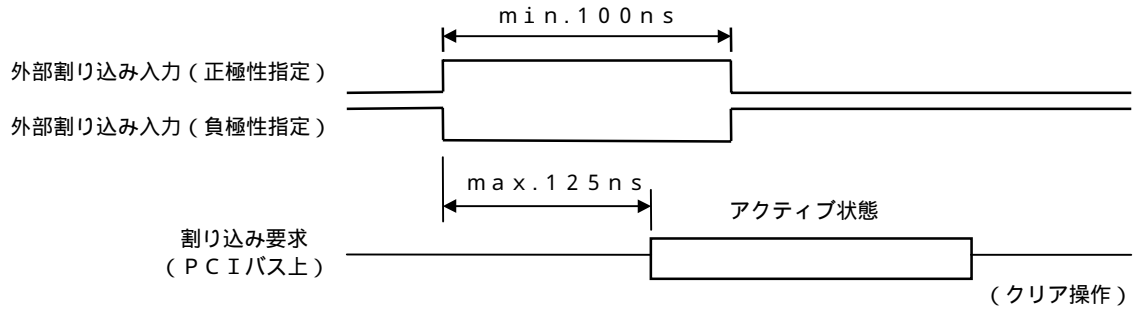
実際のサンプリング実行はクロックの各有効エッジから 125 ns 以内に開始。

T_{dd} : 分周開始の遅れ時間 (最大 250 ns)

外部割り込み入力

（許可されている場合 / 3-17項）

図7-4D. 外部割り込み入力 ~ 割り込み受け付け

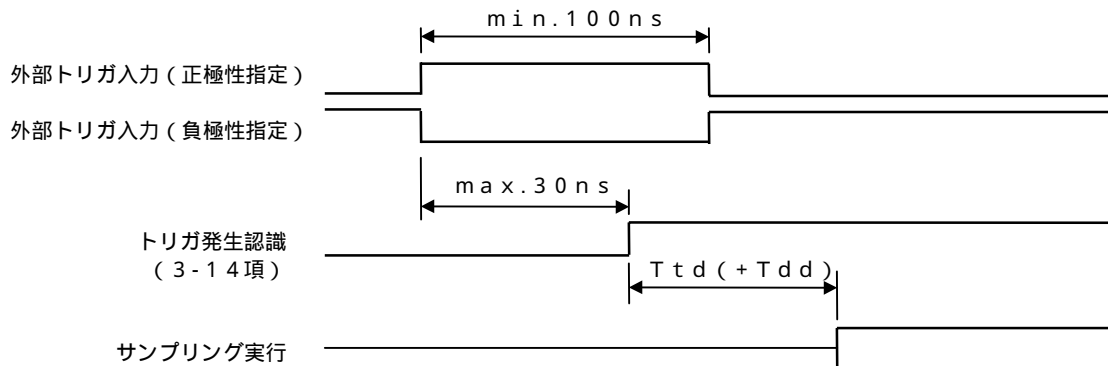


: 外部割り込み信号入力 (INT-IN) の有効エッジ。
 : 割り込みクリア操作タイミング。(通常、デバイスドライバ内で実行する)

外部トリガ入力

（許可されている場合 / 3-13項）

図7-4E. 外部トリガ ~ 連続サンプリング開始



: 外部トリガ信号入力 (TRG-IN) の有効エッジ。
 : 連続サンプリングの開始タイミング。
 Ttd: 図7-4A, B, C参照。
 Tdd: 図7-4C参照。

7-5. 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

(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

(所属部・課)

(所在地)