第6章
割り込み

6-1 割り込み処理の概要

割り込みは，I／Oからの処理要求が発生したら，その時実行していたプログラムを中断(インターラプト)し，必要な処理に移る機能です．割り込みを上手に活用することによって，非常に効率のよいプログラムを作成することができます．

割り込み処理はBASICでも一部サポートされていることはご存じでしょう．

```
1000  'サンプル
1010  ON KEY GOSUB 2000
1020  KEY 1 ON
1030  'メインルーチン
1040  ....
1050  ....
1060  ....
.
.
1500  GOTO 1030
2000  'KEY INT ROUTINE
2010  XX=POS(0):YY=CSRLIN
2020  LOCATE0,0:PRINT"STATUS:";STA
2030  LOCATE XX,YY
2040  RETURN
```

上のプログラムではメインルーチンの処理中に[F1]キーが押されると画面の上隅に変数STAの内容を表示します．この場合，キーボードが「割り込み(をかける)デバイス」，[F1]キーの押下が「割り込み要因」，行番号2000からが「割り込み処理ルーチン」となります．プログラムの最初で[F1]割り込み処理ルーチンの行番号「割り込みベクタ」を宣言し(ON KEY GOSUB)，続いて割り込みを許可します(KEY 1 ON)．Z80の割り込み処理についてこのあと説明しますが，概念としてはこのBASICの処理にたいへん似ています．

X1シリーズには，キーボードやディスクを始めとして数多くのI／Oがありますが，そのほとんどは割り込みが使用できるように設計されています．しかもZ80の割り込み処理の方法のうち，最も強力なモード2割り込みが使用できます．そこで，最初にZ80の割り込み処理について説明しましょう．

6-2-1 Z80の割り込み処理

Z80CPUは，INTと名付けられた端子があり，この端子をLレベルにすることで割り込み処理が始まります．もう少し正確に言うと，1つの命令を実行し終った時点でINT端子がLであり，かつ割り込み許可の状態であればCPUは「ある方法で」割り込み処理ルーチンをcallするのです．この時のcallの方法にはモード0～モード2の3種類があります．どのモードを使うかは予めプログラムで設定しておきます(IM0～IM2命令)．

①モード0

このモードは，Z80の前身とも言える，インテル社の8080A CPUとほぼ同じ動作をします．

CPUは，通常はプログラムカウンターPCの指す番地のメモリの内容を読み込み，解釈し，実行する，ということを繰り返しています．これに対しモード0で割り込みが起こるとCPUは，メモリへの読み出し命令を出さずに命令を読み込み，それを実行します．そこでハードウェアを工夫し割り込みが発生した瞬間，適当な命令(通常はRST0～7のcall命令)をCPUに読み込ませるようにします．call命令の飛び先に割り込み処理ルーチンを書いておきます．

②モード1

このモードの動作は簡単です．割り込みが発生すると，CPUは自動的にRST7を実行します．すなわち現在のPCの値をスタックにPUSHして0038H番地にジャンプします．

モード1では割り込み処理ルーチンの入口が1つしかないため，複数の割り込みデバイスがある場合，どこから割り込みがかかったかの判定はソフトウェアで行います．例えばキーボード，プリンター，タイマーの3つの割り込みデバイスがあるときは図6-1のように処理します．この場合，複数のデバイスが同時に割り込みを起こした時，どれを最初に実行するか(割り込みの優先順位)は，割り込み要因を調べる順番で決まります．先に調べられるデバイスの方が優先的に処理されます．

③モード2

割り込み優先順位の決定から，各装置ごとの処理ルーチンのcallまでを自動的に行うようにしたのがモード2です．3つのモードの中では最も強力で，ソフトウェアでさまざまな判断をする必要がないので割り込みに対する応答が早くなります．X1シリーズでは，もっぱらこのモード2が使用されています．

モード2で割り込みが発生すると，CPUは「ベクタ」と呼ばれる1バイトの情報を読み込みます．そして，このベクタを下位，CPU内部のIレジスタの内容を上位とするアドレスから16ビットのアドレスを読み込み，そこをcallします．この様子を図6-2に示します．ここでは，Iレジスタの値は0ABH，割り込みデバイスであるPIOの出した(あらかじめPIOにセットしておいた)ベクタの値が0CDHだったのでCPUは0ABCDHの内容を読み込みます．それが1234Hであったので，CPUはcall 1234Hを実行します．

結局，Iレジスタが上位，00Hを下位とする番地から256バイトが割り込み処理ルーチンのテーブルとなり，受け付けられる割り込みの種類は最大128種類，ということになります．

割り込み処理ルーチンは，各レジスタのPUSHから始めます．割り込み処理によってレジスタが破壊されると，もとの処理に戻った時に暴走する原因となるからです．処理が終了したら，PUSHしたレジスタをもとに戻し，EI(割り込み許可)，RETI(割り込みからのリターン)の順番に実行します．EIを実行するのは，割り込みがかかると自動的にDI(割り込み禁止)状態になるため，EIを実行しないと2回目の割り込みがかからなくなるからです．また，RETIを実行すると割り込みをかけたデバイスは，処理終了とみなして割り込み要求を解除するようになっています．

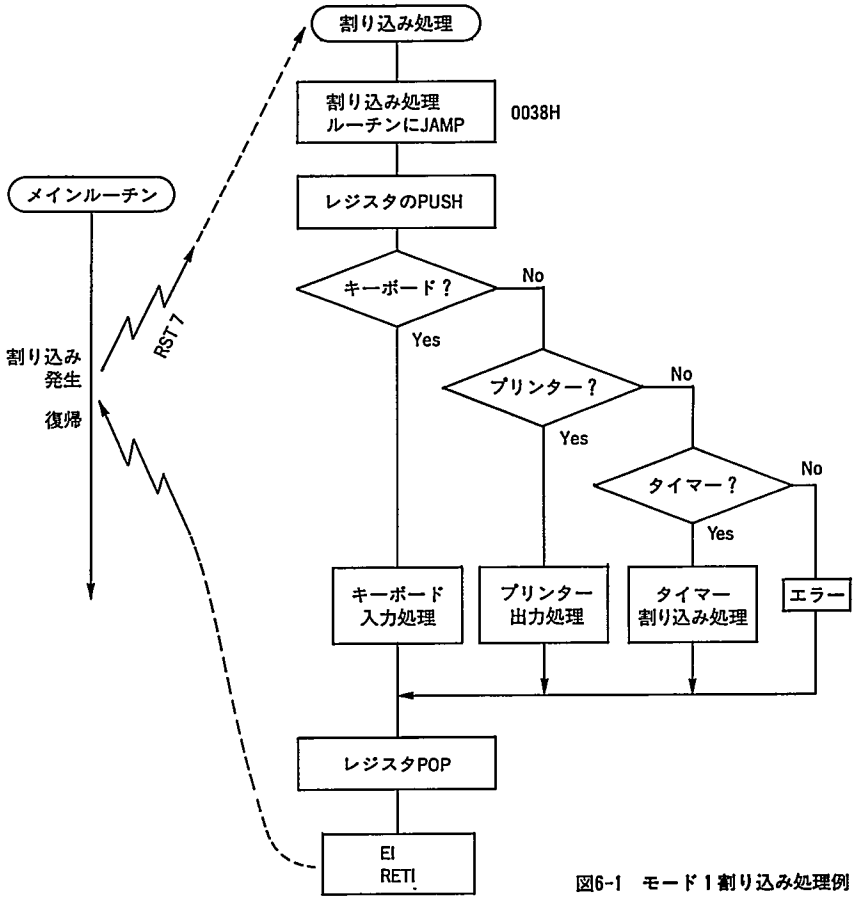


図6-1 モード1割り込み処理例

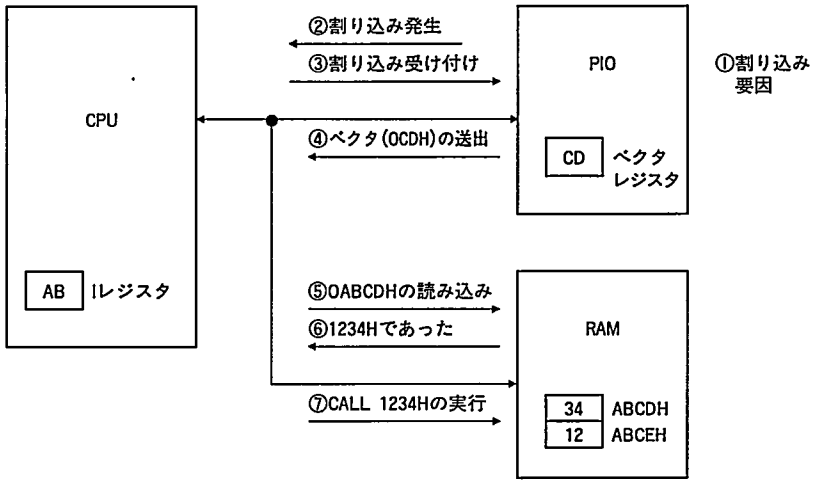


図6-2 モード2割り込み処理

6-1-2　メインCPUの割り込みデバイス

先に述べたように，X1シリーズでは，Z80CPUの割り込みモードのうち，最も強力なモード2割り込みが使用できるように設計されています．割り込みが使用できるデバイスには，以下のものがあります．

1．キー入力(サブCPU)
2．Z80-SIO(RS-232C，マウス)
3．Z80-CTC
4．Z80-DMA
5．拡張I/Oスロット(外部割り込み)

割り込みの優先順位はハードウェアで決められており，以下のようになっています．

I/Oスロット1＞I/Oスロット2＞SIO＞DMA＞CTC＞キー入力

左側のデバイスほど優先的に処理されます．なお，X1では，内部割り込みデバイスはキー入力だけであり，拡張スロットは1～4の4つになります．優先順位は，次のようになります．

I/Oスロット1＞I/Oスロット2＞I/Oスロット3＞I/Oスロット4＞キー入力

割り込み時のベクタの値は，予め各デバイスにセットしておかなければなりません．ベクタの一覧を表6-1-1に示します．個々のデバイスの使い方については，次節以降で説明します．

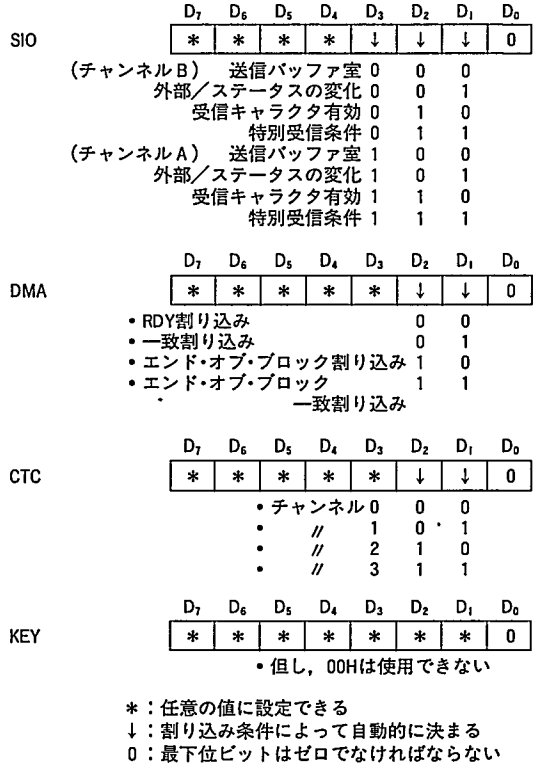


表6-1　各種割り込みデバイスのベクタアドレスの下位バイトの構成

6-2 シリアルI／O

X1シリーズにおけるシリアルI／Oは，RS-232Cインターフェイスとマウスインターフェイスです．これらシリアルI／Oには，Z80-SIOという非常に高機能なインターフェイスLSIが使用されています．

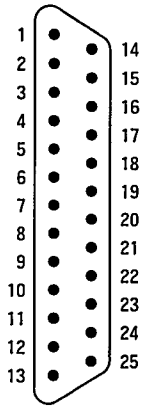
6-2-1 シリアルI／Oの概要

パソコン用としてシリアルI／Oが活用されはじめたのは最近のことです．いわゆる「パソコン通信」の発展とともに，これからはますますその重要性を増していくことと思われます．最初にシリアルI／Oで出てくるいくつかの用語について説明しましょう．

①RS-232Cインターフェイス

本来はEIA(アメリカ電子工業会)が，DCE(回線接続装置)と端末間のインターフェイス条件について定めた規格です．RS-232Cでは，コネクタのピン配置と信号の電気的な仕様が定められています．但し，信号の形式については何も定められていません．232Cを使用するときに非常に多くの設定をしなければならないのは，このためでもあります．

RS-232Cのピン配置を表6-2に示します．この中でパソコンレベルで実際に使われているのは，GND(グラウンド)と入出力線，及びRTS／CTSのハンドシェイク線くらいです．RTS／CTSすら使わず，3本の線だけで接続することも多いようです．232Cの信号は，±10V以上の電圧を持っており，そのままではパソコン内部のTTLレベルのLSIに接続することができません．このため専用のインターフェイス用ICが使われます．



ピン番号	内　　容
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground (Common Return)
8	Received Line Signal Detector
9	(Reserved)
10	(Reserved)
11	Unassigned
12	Sec. Rec'd. Line Sig. Detector
13	Sec. Clear to Send
14	Secondary Transmitted Data
15	Transmission Signal Element Timing (DCE Source)
16	Secondary Received Data
17	Receiver Signal Element Timing (DEC Source)
18	Unassigned
19	Secondary Request to Send
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicator
23	Data Signal Rate Selector (DTE/DCE Source)
24	Transmit Signal Element Timing (DTE Source)
25	Unassigned

表6-2 RS-232Cのピン配置

②調歩同期式

パソコンでRS-232Cインターフェイスの使われ方としては，パソコン同士を接続してのデータ転送，ネットワークにつなぐためのモデムとの接続，ミニコン等の端末として，といった使い方が多いと思います．これらの場合，300～9600ボー程度の調歩同期式通信が使われます．

この方法では，1本の信号線で直列に1ビットづつ信号を送っていきます．何も信号が無いとき，ラインはマーク(H)であり，そこに図6－3のように1文字分の信号をのせます．各信号の意味は，次の通りです．

・スタートビット：信号の始まりを示す．1ビット分の長さの「0」．
・文字ビット　　：5～8ビットのデータ．
・パリティ　　：1ビットの誤りを発見することができる．
　　偶数パリティ，奇数パリティ，パリティ無しのいずれか．
・ストップビット：1，1.5，2のいずれかの長さの「1」．

ここで，「1ビットの長さ」を示すのにボーレート(Baud Rate)という言葉が使われます．ボーレートが9600ボーであるとは，信号をぎっしり詰めて送ると，1秒間に9600ビット分ある，という意味です．実際にはスタートビットなどの無駄な信号がありますから，データの転送速度はこれより小さくなります．

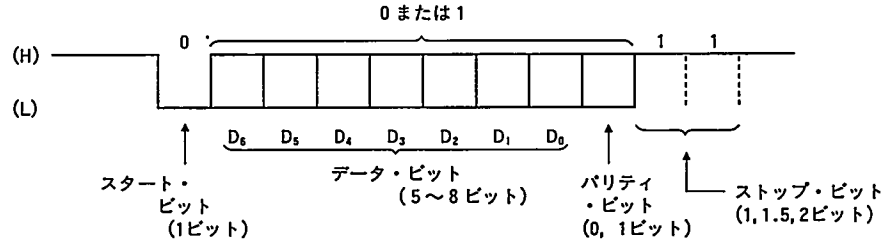


図6-3　調歩同期式通信のデータフォーマット

ある，パソコンネットワークの規格は次のようになっていました．

2400ボー，　8ビットデータ，　パリティ無し，
1ストップビット，XON／XOFF制御有り，

今までの説明で，最初の4項目の意味はわかるでしょう．最後のXON／XOFFとは，データの転送停止，再開をコントロールする方式の1つです．データを受け取る方の処理が間に合わなくなりそうになったら，相手方にXOFFキャラクタ(^S)を送り，転送を一時停止してもらい，処理が終了したらXONキャラクタ(^Q)を送って再開します．

③SDLC, HDLC

データを1文字ずつ送る従来の方法が持っていたいくつかの欠点を解決し，高速で，使いやすく，信頼性の高い方式として制定されたのがHDLC(High-level Data Link Control)方式です．この規格の基となったIBMの開発した方式がSDLC方式で，X1シリーズに使われている

Z80-SIO はこの SDLC をサポートしています.

SDLC は，任意の長さのデータと制御記号から成る「フレーム」を単位として行われます．細かい解説は省略しますが，この方式は以下のような特徴を持っています.

・バイナリデータを送ることが容易.
・任意の長さのデータをまとめて送るので転送速度が速い.
・CRC を使った誤り訂正制御が行われるので，信頼性が高い.
・訂正不能の誤りが発生した時の再送シーケンスについても厳密に定められており，それらの制御も連続的に行われるので伝送効率が高い.

6-6-2 SIO まわりの構成

X1 シリーズでは，SIO のチャンネルAは RS-232C，チャンネルBはマウスの制御に使われています．SIO まわりの構成を図 6 - 4 に示します.

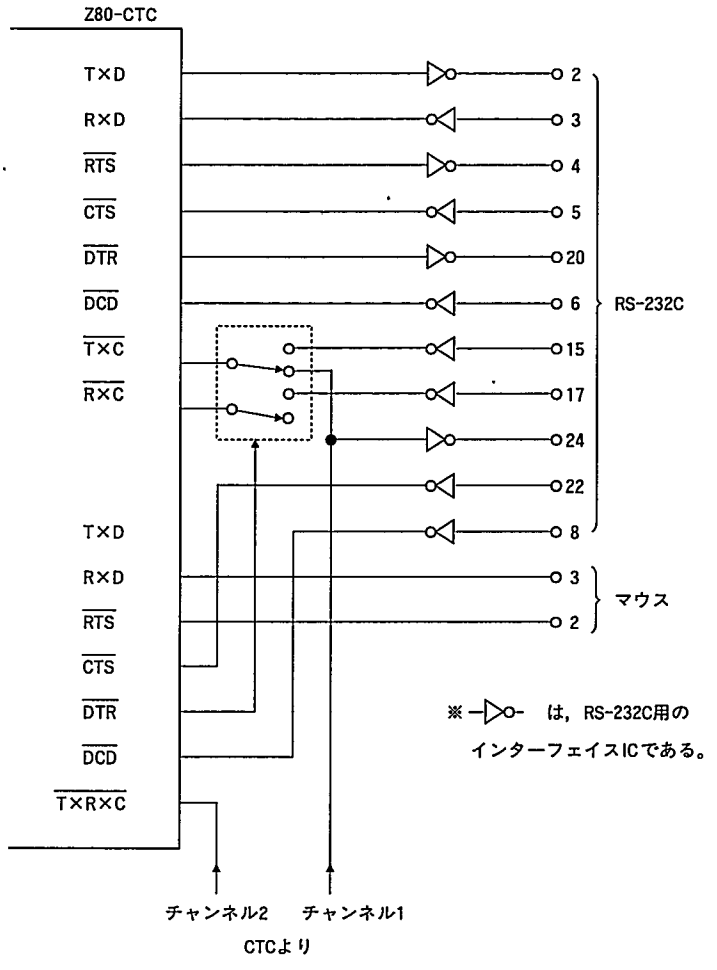


図6-4 SIOまわりの構成

RS-232C には多くの制御線がありますが，X1 シリーズでサポートされているのはその一部です．サポートされる制御線と，SIO の端子との関係は，以下の通りです．

SIO端子	チャンネル	IN/OUT	RS-232C端子（ピン番号）
TxD	A	OUT	Transmitted Data (2)
RxD	//	IN	Received Data (3)
RTS	//	OUT	Request to Send (4)
CTS	//	IN	Clear to Send (5)
DTR	//	OUT	Data Terminal Ready (20)
DCD	//	IN	Data Set Ready (6)
TxC	//	IN	Transmission Signal Element (15)/CTC ch-1 output. (*)
RxC	//	IN	Received Signal Element (17)/CTC ch-1 output. (*)
DCD	B	IN	Received Line Signal (8)
CTS	//	IN	Ring Indicator (22)
GND			Frame GND (1)
GND			Signal GND (7)

（*）チャンネルBのDTRが「H」の時CTCに接続される

表6-3 SIOとRS-232Cの関係

パソコン同士を接続して通信するときは，

RD(3) ⟷ TD(2)
RTS(4) ⟷ CTS(5)
DTR(20) ⟷ DCD(6)

の各線をひっくり返しにして接続すれば，通信できます．但し，この場合上記6つの信号線以外の入力データは意味がありません．

SIO のポートアドレスを表6-4 に示します．このようにチャンネル A とチャンネル B のそれぞれについてデータポートと制御語(コントロールポート)があります．SIO には制御のための書き込みレジスタが各チャンネルに8個ずつあります．レジスタに値を書き込むときには，まずレジスタ番号をコントロールポート(書き込みレジスタ0)にセットし，続いて目的のレジスタの値を書き込みます．例えばレジスタ2に10H を書き込むときはコントロールポートに02H，10H と書き込みます．但し，書き込みレジスタ0については1回で書き込めます．

また，内部状態を知るための読み出しレジスタの内容を読むときも，最初にレジスタ番号を書き込み，続いて読み出しを行います．読み出しレジスタ0だけは直接読み出せます．

書き込み，読み出しの各レジスタの内容を次頁から示します（表6-5）．SIO はSDLC をサポートしているため，レジスタの内容はたいへん複雑になっています．

アドレス	ポート内容
1F90H	チャンネルAデータポート
1F91H	チャンネルA制御語
1F92H	チャンネルBデータポート
1F93H	チャンネルB制御語

表6-4 SIOのI／Oポート

書き込みレジスタ0：WR0

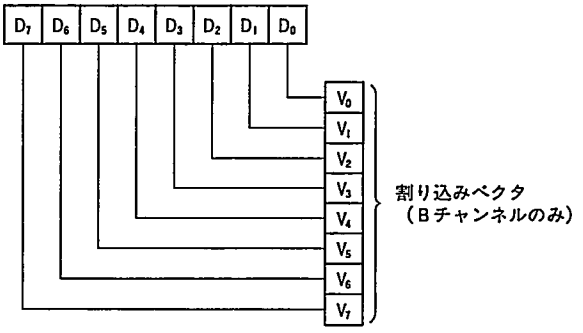
注）＊印の付いたビットは，一般に使われる「非同期通信」では，意味がない．

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
0	0				0	0	0	レジスタ0	書き込みレジスタ選択
					0	0	1	レジスタ1	
					0	1	0	レジスタ2	
					0	1	1	レジスタ3	
					1	0	0	レジスタ4	
					1	0	1	レジスタ5	
					1	1	0	レジスタ6	
					1	1	1	レジスタ7	
		0	0	0				動作に何の影響も与えない	制御コマンド ＊
				0				アポート送出	
				1				外部ステータス割り込みリセット	
				1				チャンネル・リセット	
				0				つぎの受信キャラクタで割り込みイネーブル	
				1				送信割り込みの保留リセット	
				1				エラー・リセット	
				1				割り込みからの復帰(Aチャンネルのみ)	
0	0							動作に何の影響も与えない	CRCリセット・コード
0	1							受信CRCチェッカ・リセット	＊
1	0							送信CRCジェネレータ・リセット	＊
1	1							送信アンダーラン/EOMリセット	＊

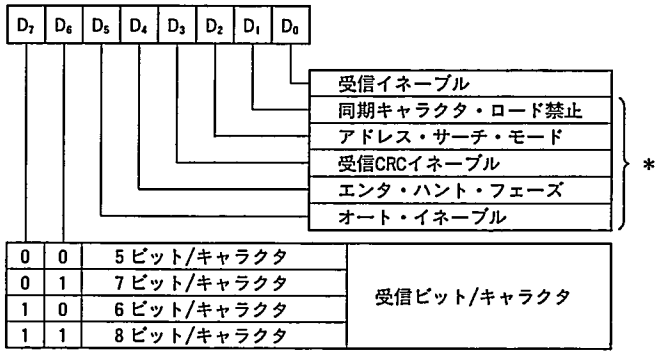
書き込みレジスタ1：WR1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
							1	外部/ステータス割り込みイネーブル	
						1		送信割り込みイネーブル	
					1			ステータス・アフェクト・ベクタ・イネーブル(Bチャンネルのみ)	
			0	0				受信割り込みディセーブル	割り込みモード
			0	1				最初のキャラクタのみで受信割り込み	
			1	0				すべての受信キャラクタで割り込み（パリティ・エラーは特別受信条件となる）	
			1	1				すべての受信キャラクタで割り込み（パリティ・エラーは特別受信条件とならない）	
0	0	×						$\overline{WAIT}$ ：フローティング	ウエイト・レディ機能選択
0	1	×						$\overline{READY}$：“H” レベル	
1	0	0						$\overline{WAIT}$ ：送信バッファがフルかつSIOのでータ・ポートが選択されているとき“L”レベル ：送信バッファが空のときフローティング	
1	1	0						$\overline{READY}$：送信バッファがフルのとき“H”レベル ：送信バッファが空のとき“L”レベル	
1	0	1						$\overline{WAIT}$ ：受信バッファがフルのときフローティング ：受信バッファが空かつSIOのデータ・ポートが選択されているとき“L”レベル	
1	1	1						$\overline{READY}$：受信バッファがフルのとき“L”レベル ：受信バッファが空のとき“H”レベル	

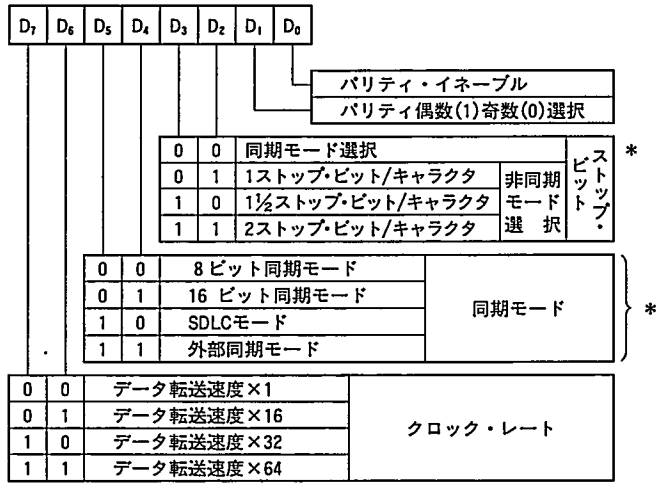
書き込みレジスタ 2 ：WR2



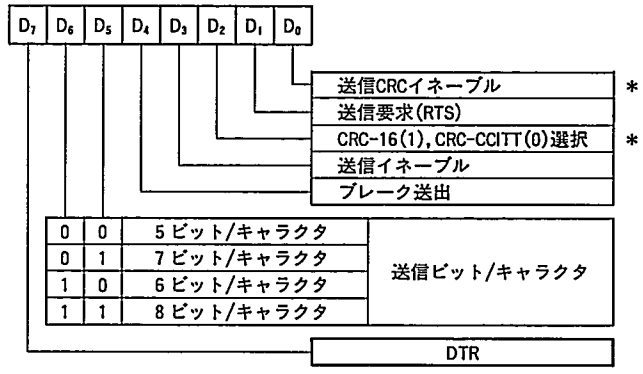
書き込みレジスタ 3 ：WR3



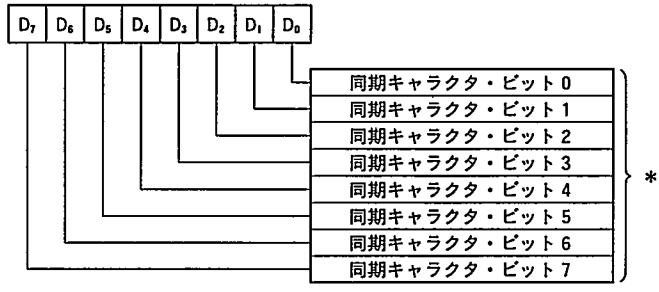
書き込みレジスタ 4 ：WR4



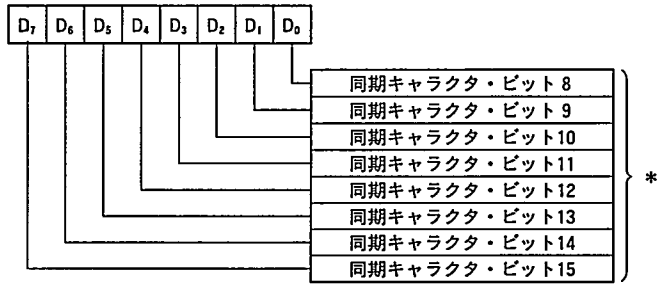
書き込みレジスタ 5：WR 5



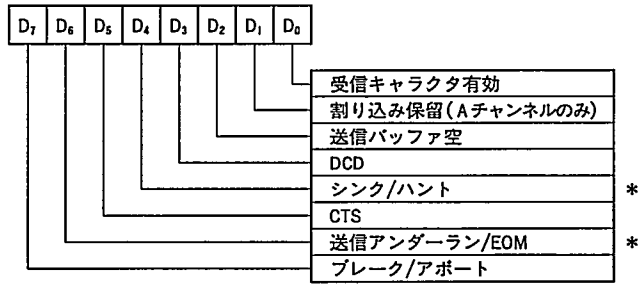
書き込みレジスタ 6：WR 6

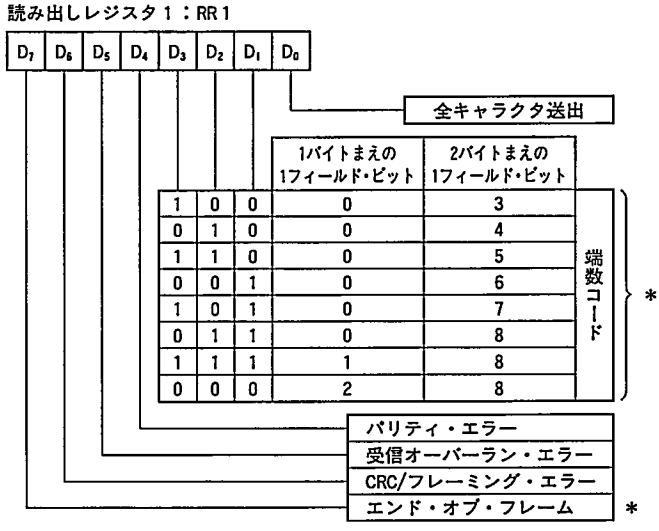


書き込みレジスタ 7：WR 7



読み出しレジスタ 0：RR0





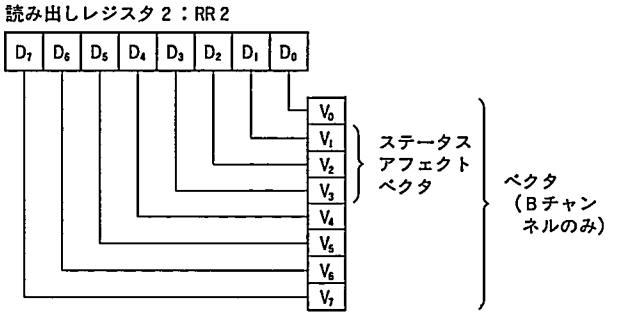


表6-5 Z80-SIOのレジスタ構成

6-2-3 チャンネルA(RS-232C)

チャンネルAは，RS-232Cの入出力として使用されています．

通信のためのクロックは，内部同期か外部同期かを選択することができます．この選択にはチャンネルBのDTR端子を使用します．外部同期にするときはここを0とし，内部同期の時は1にします．

内部同期の時のボーレートは，CTCのチャンネル1の分周比設定とSIOの設定の両方で決まります．ボーレートとこれらの設定の関係を，下に示します．

	SIO設定	
CTC分周比	1/16	1/64
13	9600	2400
26	4800	1200
52	2400	600
104	1200	300
208	600	150

＊CTC分周比＝52，SIO設定＝1/64ならば600ボーとなる．

表6-6 ボーレートの設定

BIOS ROMには，RS-232C関係の処理ルーチンが用意されていますから，これらを利用すると入出力が簡単にできます．しかし，ハードウェアによるハンドシェイクなどを行うときは，その処理はユーザープログラム側で行わなければなりません．もっとも，先に述べた通り，ハードウェアハンドシェイクが使われることはパソコンレベルではあまりありません．

6-2-4 チャンネルB(マウス)

SIOのチャンネルBはマウスの制御に割り当てられています．使われている端子は2つだけです．マウスへのコントロール出力($\overline{CTRL}$)としての$\overline{RTS}$出力と，マウスからのデータを受け取るRxD入力です．

マウスはRTS端子がHからLになると，3バイトのデータを送ってきます．送られてくるデータは次のようなフォーマットになっています．

4800ボー，8ビットデータ，パリティなし，1ストップビット

従って，SIOもこの通り設定します．ボーレートはCTCのチャンネルBを使ってRS-232Cと同様に設定します．4800ボーと決まっていますから，CTCの分周比は26，SIOの設定は1／16とします．

マウスからのデータは3バイトで，1バイト目がステータス，2バイト目と3バイト目が各々X，Y方向の前回からの移動量(−128〜127)です．ステータスの内容は次の通りです．

データ	内　　容
D0	スイッチ1の状態を示します．0＝OFF，1＝ON
D1	スイッチ2の状態を示します．0＝OFF，1＝ON
D2	―――
D3	―――
D4	オーバーフロービット，Xが　128以上の時　1
D5	アンダーフロービット，Xが−129以下の時　1
D6	オーバーフロービット，Yが　128以上の時　1
D7	アンダーフロービット，Yが−129以下の時　1

表6-7

リスト6-1　マウスからのデータの読み込み

```
SIOBAD  EQU     1F94H
RDMSE:  LD      BC,SIOBAD
        LD      HL,RDMDT
        LD      A,05H          ┐
        OUT     (C),A          │
        INC     HL             │ SIOのチンネルBのRTS端子を'L'にする
        LD      A,0E0H         │
        OUT     (C),A          ┘
        LD      D,03H
RDMS1:  IN      A,(0E0H)
        RRC     A
        JR      NC,RDMS1
        LD      A,01H          ┐
        OUT     (C),A          │
        IN      A,(C)          │
        LD      E,A            │
        DEC     C              │ SIOからデータを受信する
        INC     HL             │
        IN      A,(C)          │
        INC     C              │
        LD      (HL),A         ┘
```

```
        LD      A,E
        AND     70H
        JR      NZ,RDMS3
        DEC     D
        JR      NZ,RDMS1
        XOR     A
RDMS2:  RET
RDMS3:  LD      A,0FFH          ] マウスの移動量が－129～＋128をこえた
        JR      RDMS2           ]
        ;
RDMDT:  DS      3
        ;
        END
```

6-3 DMA

6-3-1 DMAの概要

DMA(Direct Memory Access)とは，CPUを介さずにハードウェアで直接データを転送することです．メモリやI/Oの間でデータの転送を行うとき，DMAを使用すると非常に高速に転送を行うことができます．X1turbo以降にはZ80-DMAと呼ばれるLSIが実装されており，フロッピーディスクの読み書きやV-RAMのスクロールなどに使われています．

DMAに指令を与えるには，SIOなどと同じようにOUT命令を使用します．実行命令が与えられるとDMAは，CPUを停止させ，CPUの代わりに各種の制御信号を出力しながらデータの転送を行います．従ってその動作は一般のI/Oとはかなり異なります．

Z80-DMAの特徴は次の通りです．

1．メモリ⟷メモリ，I/O⟷I/O，I/O⟷メモリのいずれの転送も可能．
2．データ転送のほか，データサーチやサーチしながらの転送が可能．
3．転送データの長さは，2バイト～64Kバイト．
4．転送が終了するまでCPUを止める，CPUと並列に転送するなど，4つのモードを持つ．
5．転送アドレスは，固定，インクリメント，デクリメントのいずれか．
6．転送速度は最高500Kバイト／秒以上，サーチ速度は最高1000Kバイト／秒以上(但し条件によって大きく異なる)．

CPUのプログラムの実行と，DMAのデータ転送のタイミングはDMAの転送モードで決まります．転送モードには次の4種類があります．

1．**コンティニュアスモード**：転送がすべて終了するまでCPUは停止する．
2．**バーストモード**：RDY信号(データ転送可能信号)が入力されている間CPUは停止する．転送可能でないときはCPUが動作する．
3．**バイトモード**：1バイト転送すると，CPUが動作する．データが連続的にくる場合は，DMAとCPUは交互に動作することになる．
4．**トランスペアレントモード**：CPUのメモリリフレッシュサイクルを使って，CPUの動作とDMAが同時に実行される．

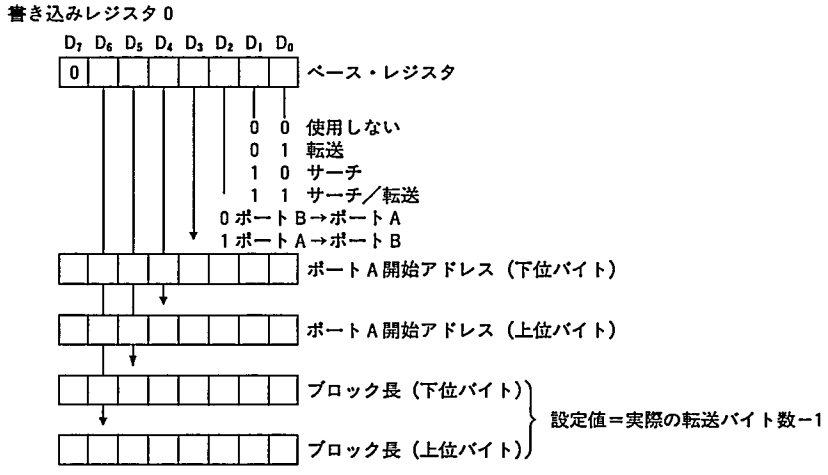
X1turboでは，フロッピーディスクの読み書きにバイトモードが，V-RAMのスクロールやその他のメモリ転送にバーストモードが，それぞれ使用されています．なお，トランスペアレントモードは，X1シリーズでは使用できません．

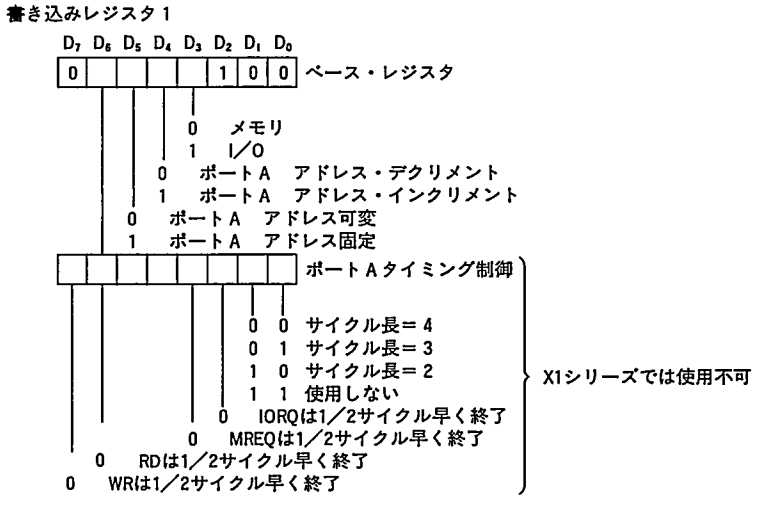
6-3-2 DMAの使い方

DMAには，書き込みレジスタが21個，読み出しレジスタが7個あります．

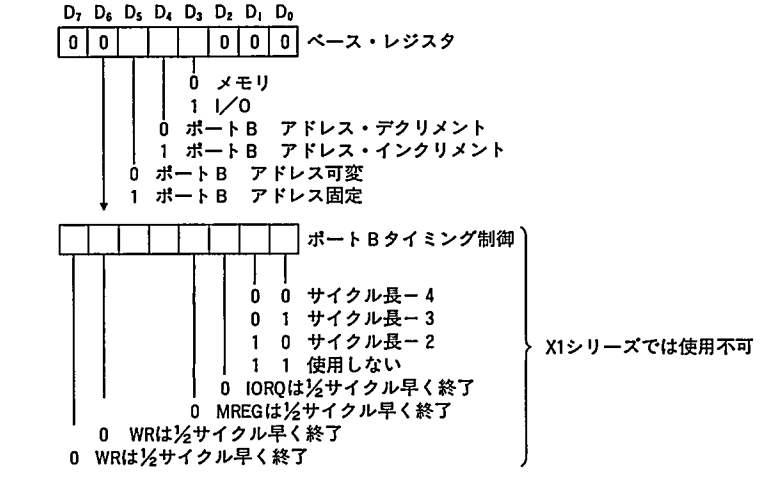
●書き込みレジスタ

21個の書き込みレジスタは，さらに7個のベースレジスタと21個の関連レジスタに分けられます．ベースレジスタはリセット時には内容は不定ですので，必ず7個とも初期化しなければなりません．関連レジスタは，必要なもののみ初期化します．これら書き込みレジスタの内容一覧を，図6-3-1に示します．

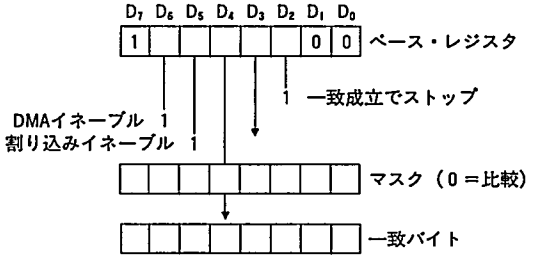




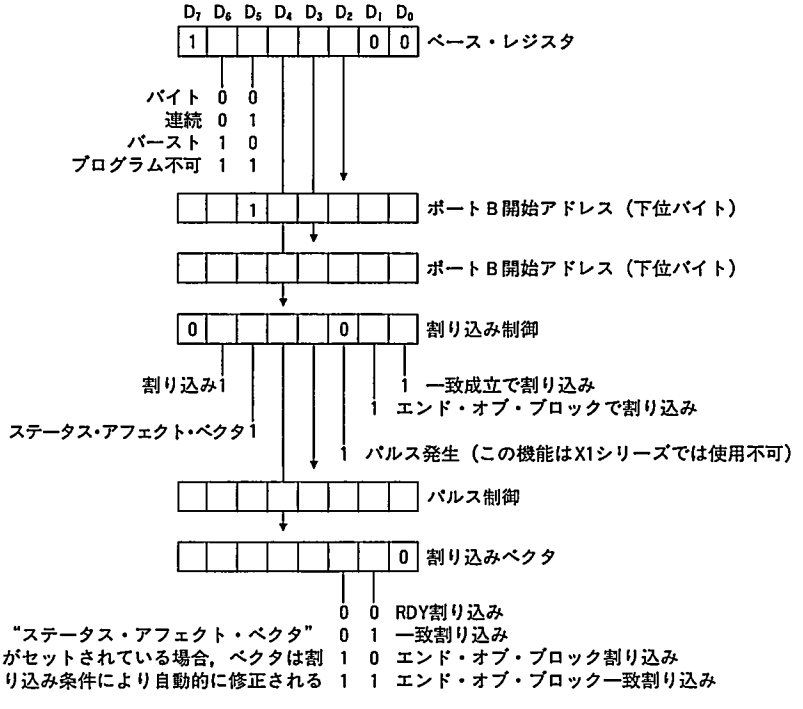
書き込みレジスタ 2



書き込みレジスタ 3

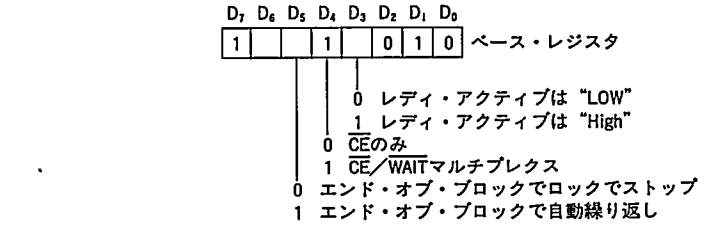


書き込みレジスタ 4

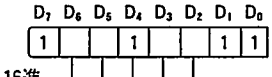


“ステータス・アフェクト・ベクタ”がセットされている場合，ベクタは割り込み条件により自動的に修正される

書き込みレジスタ 5



書き込みレジスタ 6



16進						
C3	1	0	0	0	0	割り込み回路リセット，割り込み要求とバス要求回路ディスエーブル，内部レディ状態の解除，CEマルチプレクス不可，自動繰り越しストップ
C7	1	0	0	0	1	ポートAタイミングをZ−80標準タイミングに設定
CB	1	0	0	1	0	ポートBタイミングをZ−80標準タイミングに設定
CF	1	0	0	1	1	両ポートの開始アドレスロード，バイト・カウンターのクリア
D3	1	0	1	0	0	現在値からアドレス続行，バイト・カウンターのクリア
AB	0	1	0	1	0	割り込みイネーブル
AF	0	1	0	1	1	割り込みディセーブル
A3	0	1	0	0	0	割り込み回路のリセットとディセーブル(RETIと同じ)，内部レディ状態の解除
87	0	0	0	0	1	DMAイネーブル } 割り込み以外の動作にすべて有効，ただし，すべての機能をリセットするわけではない。
83	0	0	0	0	0	DMAディセーブル }
A7	0	1	0	0	1	読み出しマスク・レジスタによって指定された第1レジスタに対し読み出しシーケンスを起動
BF	0	1	1	1	1	ステータス・レジスタに読み出し設定。次の読み出しはステータス・レジスタから
B3	0	1	1	0	0	強制的に内部レディ状態を "RDY" 端子に無関係とする（RDY信号を必要としないメモリー間DMAに使用。このコマンドは "バイト・モード" では動作しない）
8B	0	0	0	1	0	一致成立，エンド・オブ・ブロックのビットをクリア
B7	0	1	1	0	1	RETI後イネーブル。RETI実行後のみバス要求
BB	0	1	1	1	0	読み出しマスクがこの後に続く

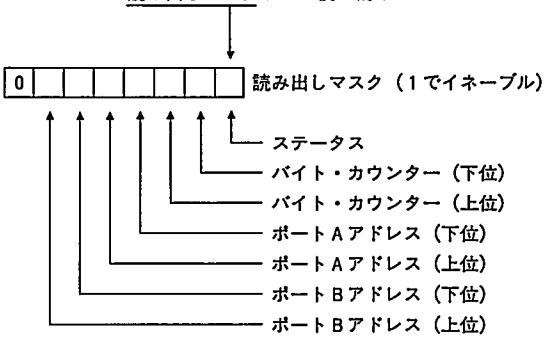


図6-5 Z80-DMA書き込みレジスタ

ベースレジスタに値を書き込むには，I/Oアドレス1080Hに対してOUT命令を実行します．I/Oアドレスが1つしかないのに，どうやって7つのレジスタを区別するのかと思われる人もいるでしょう．もう一度図6-5を見て下さい．各レジスタの中に，0か1かが決められているビットや，「使用しない」となっているデータパターンがあることに気が付くと思います．そして，さらに良く観察すると，データのビットパターンから，どのレジスタへの書き込みかが決定でき

ることがわかります．実はDMAは，書き込まれたデータのパターンからどのレジスタへの書き込みかを判断しているのです．

関連レジスタへデータを書き込むときは，ベースレジスタの所定のビットを1にして書き込み，それに続いて関連レジスタを書き込みます．複数の関連レジスタを指定した時は，図の順番に従って，続いて書き込みます．

DMAは，最終的に87H(DMAイネーブル)を書き込むと動作を開始し，転送が終了すると動作終了となります．また，バイトモードやバーストモードでは，CPUがDMAに対してアクセスしても動作を停止します．

DMAの使用例として，DMAの初期化プログラムと，V-RAMの内容を0FFHが来るまで転送するプログラムを示します．

リスト6-2 DMA初期化サブルーチン

```
DAMADD  EQU     1F80H
DMAIN:  LD      BC,DMAADD
        LD      HL,DMIDT
DMAI1:  LD      A,(HL)          ┐
        CP      0FFH            │
        JR      Z,DMAI2         │ DMAへコマンドとデータの送信
        OUT     (C),A           │
        INC     HL              │
        JR      DAMI1           ┘
DMAI2:  RET
        ;
DMIDT:  DB      83H,00H,14H,14H,80H
        DB      81H,80H,0C7H,0CBH,87H
        DB      0FFH
        ;
        END
```

リスト6-3 V-RAMの内容サーチ・アンド・転送

```
DMAADD  EQU     1F80H
VRSRC:  LD      BC,DMAADD
        LD      HL,VRSDT
VRSR1:  LD      A,(HL)          ┐
        CP      0FFH            │
        JR      Z,VRSR2         │ DMAへコマンド，データの送信
        OUT     (C),A           │
        INC     HL              │
        JR      VRSR1           ┘
VRSR2:  RET
        ;
VRSDT:  DB      83H,7FH,00H,40H,00H,20H
        DB      5CH,00H,58H,00H,94H,0FFH
        DB      0ADH,00H,80H,92H,0CFH,87H
        DB      0FEH
        ;
        END
```

●読み出しレジスタ

DMAには，動作状態を知るために7個の読み出しレジスタがあります．読み出しレジスタの一覧を図6-6に示します．

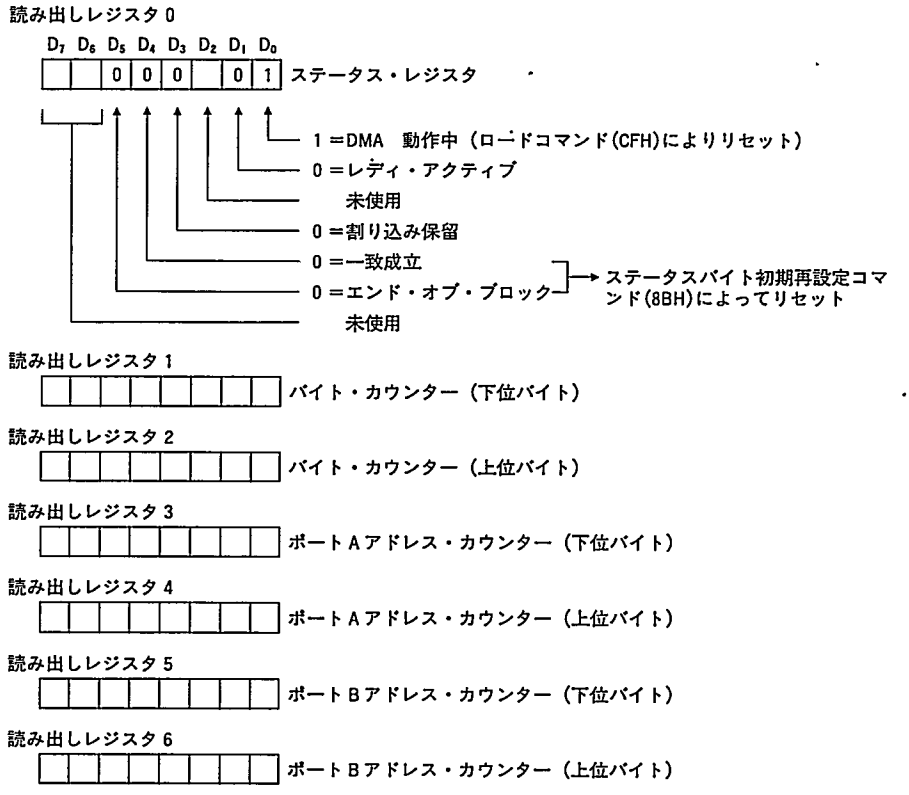


図6-6 Z80-DMA読み出しレジスタ

読み出しレジスタの内容を読む方法は2つあります.

(1) 読み出しステータスバイトコマンド

DMA に 0BFH(読み出しステータスバイトコマンド)を書き込みます(書き込みレジスタ5に書き込むことになる). 続いて DMA を IN 命令で読み出します. 読み出せるのは, ステータスレジスタ(読み出しレジスタ0)だけです.

(2) 読み出しシーケンスコマンド

このコマンドを使うと, 希望のレジスタだけを読み出すことができます.

DMA に 0BBH(読み出しマスク指定コマンド)を書き込みます. 続いて, 読み出したいレジスタの該当のビットを1にした, マスクバイトを書き込みます (図6-5参照). 次に 0A7H (読み出しシーケンスコマンド)を書き込み, 続いて IN 命令でレジスタを読み込みます. 読み込まれるレジスタはマスクバイトで指定したレジスタで, 順番はレジスタ0から6の方向です.

DMA が終了した後の読み出しレジスタの内容は, 表6-8 のようになっています. 実際に転送されるバイト数や, DMA 終了後のカウンターの値は, 動作クラス(転送, サーチ, 転送&サーチ)や動作モードによって少しずつ違うことに注意して下さい.

動作クラス	動作モード	実際に転送／サーチされるバイト数	バイトカウンターの内容	アドレスカウンターの内容(ソース)	アドレスカウンターの内容(デスティネーション)
転 送		x＋1	x	a±（x＋1）	a±x
転送／サーチ		x	x－1	a±x	a±（x－1）
サーチ	バイト	x	x	a±x	
	バースト	x＋1	x＋1	a±（x＋1）	
	連続	x＋1	x＋1	a±（x＋1）	

※xは，WR0に設定した転送ブロック長，またはサーチで一致した時のバイト数，aは，アドレスカウンターの設定値である．

表6-8 DMAの読み出しレジスタの内容

リスト6-4 DMAのレジスタをすべて読み出して格納

```
DMAADD  EQU     1F80H
DMARD:  LD      BC,DMAADD
        LD      HL,DMRDT
DMAR1:  LD      A,(HL)          ┐
        INC     HL              │
        CP      0FFH            │
        JR      Z,DMAR2         │ DMAへコマンドの送信
        INC     HL              │
        OUT     (C),A           │
        JR      DMAR1           ┘
DMAR2:  LD      D,07H           ┐
        IN      A,(C)           │
        LD      (HL),A          │ DMAからデータの受信
        DEC     D               │
        INC     HL              │
        JR      NZ,DMAR2        ┘
        RET
        ;
DMRDT:  DB      83H,0BBH,00H,0A7H,0FFH

        DS      7
        ;
        END
```

6-4 CTC

X1turboには，Z80-CTC(Counter／Timer Circuit)が内蔵されています．CTCはその名の通り，プログラムによってカウンターやタイマーとして使えるLSIです．X1turboでは，主にRS-232Cとマウスのクロック発生用として使用しています．余ったチャンネルを使用して，プログラムに，定期的に割り込みをかけたりすることもできます．

6-4-1 CTCの概要

CTCは4つのチャンネルからなっています．各チャンネルは，プログラム設定により，外部のクロック入力をカウントするカウンターモードか，内部クロック(4Mz)をカウントするタイマーモードのどちらかで動作します．どちらのモードでも，カウント終了時に割り込みをかけることができます．X1turboでは，CTCは，図6-7のように配線されており，従って各チャンネルは次のように使用します．

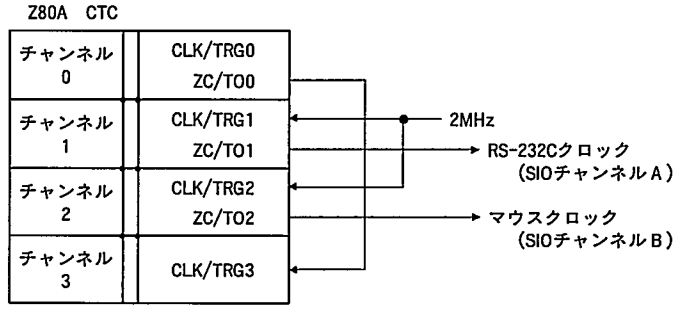


図6-7 CTCの配線

(1) チャンネル0

このチャンネルの入力には何もつながっていないので，4Mzの内部クロックを使ったタイマーモードで使用します．チャンネル0の出力パルスは，チャンネル3に接続されています．

(2) チャンネル1

2Mzのクロックが入力されており，カウンターモードで使用します．出力はZ80-SIOのチャンネルAに接続されており，RS-232Cのボーレートが，CTCの設定によって決まります．

(3) チャンネル2

チャンネル1と同様2Mzのクロックが入力されています．出力はZ80-SIOのチャンネルBに接続されており，マウスのボーレートの作成に使用します．

(4) チャンネル3

チャンネル0の出力パルスが入力されており，チャンネル0だけでは作れない，長い周期を設定するときに使います．

6-4-2 CTCの使い方

X1turboにおけるCTCのI／Oアドレスを次に示します．

チャンネル	I/Oアドレス
0	1FA0H
1	1FA1H
2	1FA2H
3	1FA3H

表6-9 CTCのI／Oアドレス

CTCには，チャンネル制御レジスタと時間定数レジスタが，各チャンネルごとに1つずつあります．また，割り込みベクタレジスタが1つあります．チャンネル制御レジスタと割り込みベクタレジスタは，書き込まれたデータの最下位ビットが0か1かで判断されます．また，時間定数レジスタは，チャンネル制御レジスタのビット2を1にした後に書き込みます．割り込み制御レジスタは1つしかなく，どのチャンネルに書いても同じです．各レジスタの内容を図6-8に示します．

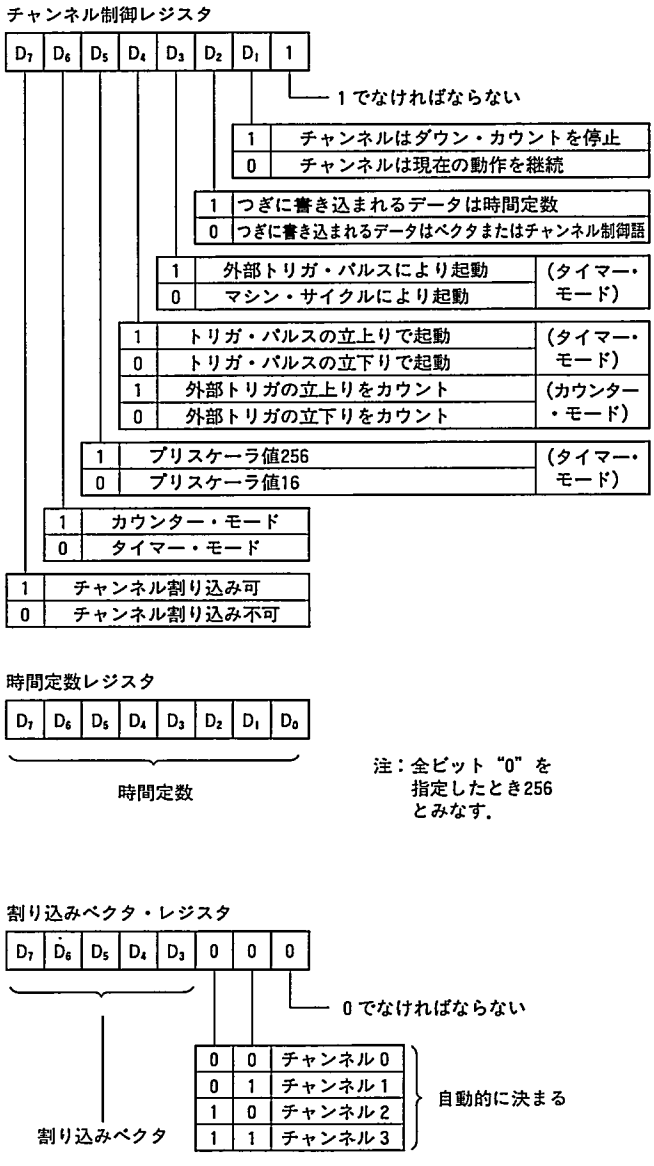


図6-8　Z80-CTCのレジスタ

6-5　キー入力

X1シリーズでは，キー入力を割り込み処理によって行うことができます.

キー入力処理はサブCPUが行っています.割り込みベクタの設定，キーデータの受け取り方などは，サブCPUの章を参照して下さい.