マイクロコンピュータマニュアル No.451

SHARP

Z-80 テクニカル マニュアル

Z-80 CPU/Z-80A CPU/Z-80B CPU
Z-80 PIO/Z-80A PIO/Z-80B PIO
Z-80 CTC/Z-80A CTC/Z-80B CTC

SHARP

Z-80 テクニカル マニュアル

Z-80 CPU/Z-80A CPU/Z-80B CPU
Z-80 PIO/Z-80A PIO/Z-80B PIO
Z-80 CTC/Z-80A CTC/Z-80B CTC

SHARP

第1部　Z-80 CPU / Z-80A CPU / Z-80B CPU
テクニカルマニュアル

第2部　Z-80 PIO / Z-80A PIO / Z-80B PIO
テクニカルマニュアル

第3部　Z-80 CTC / Z-80A CTC / Z-80B CTC
テクニカルマニュアル

第1部

Z-80 CPU / Z-80A CPU / Z-80B CPU

テクニカルマニュアル

目　　次

第1章　緒　論

〝マイクロコンピュータ〞という言葉は、ここ２・３年来、小型の電子計算機のあらゆるタイプのものに使われてきた。この言葉は、小はTTL-MSIによって組まれたマイクロプログラム方式のコントローラから、大はTTL-LSIのビットスライス方式のCPUを持つミニコンの低位の機種まで、いろいろなものに用いられている。しかし近年MOS-LSIの技術は長足の進歩を遂げ、この技術を利用することにより、コンピュータとして満足しうる強力なシステムが数個のLSIで構成できるようになった。

シャープ Z－80 ファミリはこのようなマイクロコンピュータの基盤に立って開発された、他の追随を許さないコンポーネント群である。これらのコンポーネントを利用すれば、標準タイプの半導体メモリと組み合わせて、かなり幅の広い能力をもったコンピュータ・システムを作り上げることができる。たとえば、２個のLSIと３個の標準TTL-MSIを組み合わせるだけで簡単なコントローラを得ることができるし、メモリと入出力デバイスを追加すると従来のミニコンに匹敵する能力をもったコンピュータにすることができる。このようにこれらのコンポーネント群は、広範囲のコンピュータ機能を具現化する能力があるので、多方面にわたる応用分野について、要求に応じた標準モジュールを得ることができるようになっている。

MOS-LSIがマイクロコンピュータ市場で優位である主な理由は、少数のコンポーネントでシステムを構成することができ、低コストになるということである。

たとえばMOS-LSIのマイクロコンピュータはもうすでにいろいろな応用分野でTTLロジックに代って利用されている。すなわち、ターミナル・コントローラ、周辺装置のコントローラ、交通信号制御装置、POS端末機、インテリジェント端末機、検査システムなどがそれである。事実MOS-LSIのマイクロコンピュータはエレクトロニクスのあらゆる製品にとり入れられつつあるし、はかりや自動車の制御のような数多くの機械系のシステムまでもこれに置き換えられようとしている。

MOS-LSIのマイクロコンピュータはもうすでに市場に定着し、これを応用した新製品は破竹の勢いでその開発に拍車がかけられている。シャープ Z－80 ファミリはこの時代に即応して、以下の点で上記の市場に適した設計になっている。

1）Z－80は数社から市場に提供され多用されている8080Aと全くソフトウェアに互換性があり、現存のシステムをZ－80で置きかえて、レベルアップすることは容易である。

2）Z－80　ファミリは現在市場に出されているどのマイクロコンピュータ・システムよりも、ソフトウェア、ハードウェア両面で優れた能力をもっており、ソフトウェア、ハードウェアの開発にあたって低コストで済むことが明らかであるばかりでなく、新しい特長を盛り込んだシステムを創出し得るだけの能力を秘めている。

3）新製品開発にあたって開発者に負担を与えないようにするため、高水準の言語体系に主体をおいた完璧なソフトサポートと、卓越したリアルタイム・デバッグ能力のあるディスク・ベースの開発システムをZ－80製品群の中に加え、完全を期した。

マイクロコンピュータ・システムを組むのにZ－80コンポーネントを利用すれば、極めて単純な構成となり、どんなシステムでも以下に示す3部分とすることができる。

1）CPU（中央演算処理ユニット）

2）メモリ

3）周辺装置とのインターフェース回路

CPUはシステムの心臓部で、メモリから命令を受けとり、望み通りの仕事を実行するのがその与えられた機能である。命令を記憶させるのにメモリを用いるが、それだけではなくこのメモリは処理すべきデータを記憶するのにもよく使用される。よく利用される例として、特定の周辺装置からデータを読み出し、それをメモリのある位置に貯え、パリティ・チェックした後、もう一つの周辺装置に書き込むという手順があるが、Z－80ファミリにはCPUと種々の目的に使用できる汎用の入出力コントローラがあり、いろいろな形態の外部記憶装置が利用できるようになっている。

さらに、すべて必要なコンポーネント類はとくに外部ロジックを使用せずに簡単に接続することができ、ユーザはただソフトウェアの開発に意を注げばよいようになっている。すなわち、ユーザは、自分のコンピュータにやらせたいことを記述し、マイクロコンピュータのメモリに書き込む一連の命令群にこれを書き直すことに集中すればよい。また、このソフト開発もできるだけやさしくできるように配慮してあり、その好例として、アセンブリ言語がある。この言語体系は、CPUの実行し得る命令をすべて簡単なニーモニックで表現し、そのニーモニックからそれがどんな内容の命令であるか、あの複雑な対応表を見なくてもただちに理解できるようになっているセルフ・ドキュメンテーティング方式になっている。

なお、Z—80A CPUは周波数の上限を4 MHzまでZ–80B CPUは周波数の上限を6 MHzまで保証した高速タイプであり、第2章から第10章まではとくにZ—80A CPUあるいはZ–80B CPUと記述していないがZ—80 CPUと同様に適用される。電気的仕様、パッケージ仕様については個々に定めている。第11章を参照されたい。

第2章　Z-80 CPU アーキテクチャ

Z－80　CPU の内部構成のブロック図を図２－１に示す。

この図は CPU 内部の主要な部分を示している。

それぞれについては以下の説明を参照されたい。

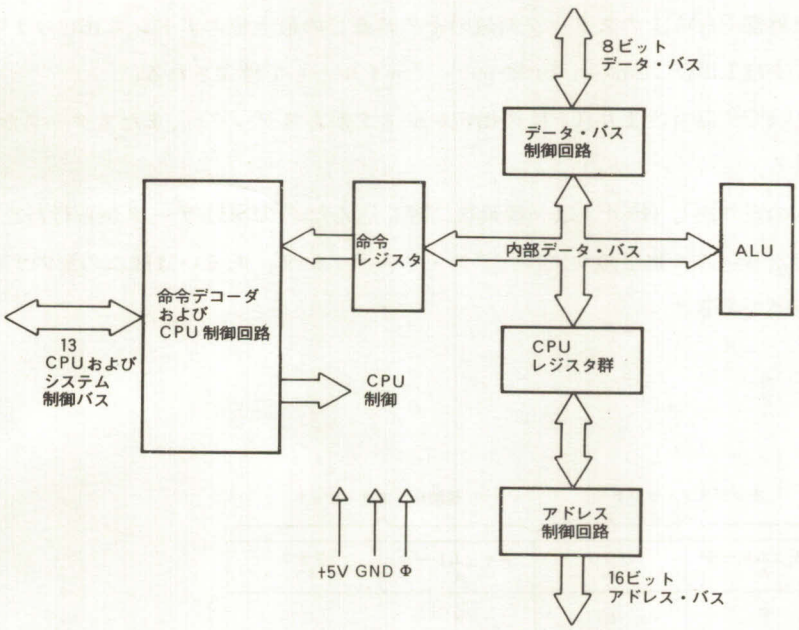


図２－１　Z－80　ＣＰＵの内部構成ブロック図

２．１　ＣＰＵレジスタ群

Z－80　CPU には208ビットの読み出し/書き込みメモリがあり、プログラマによって任意にアクセスできるようになっている。

図２－２にこのメモリの配列を示す。これらは16個の８ビット・レジスタ、４個の16ビット・レジスタから成っている。

Z－80のレジスタはすべてスタティック RAM で構成されている。

６個の汎用レジスタがそれぞれ２セット用意されていて、これを単独の８ビット・レジスタとして使用してもよいし、対にして16ビット・レジスタとして用いることもできる。

２セットあるアキュムレータ、フラグ・レジスタについても同様である。

専用レジスタ

1．プログラム・カウンタ（ＰＣ）

プログラム・カウンタは現在実行中の命令のメモリ・アドレス16ビットを保持しており、CPU はＰＣで示されるメモリ・アドレスから命令をフェッチする。

このＰＣはその内容をアドレス線に送り出すと、インクリメンタによりＰＣの値を自動的に＋１する。プログラム・ジャンプの場合、インクリメンタは動作せず、新しい値が直接ＰＣにセットされる。

2．スタック・ポインタ（ＳＰ）

スタック・ポインタは外部 RAM 上のスタック領域のその時点での最上位のアドレス16ビットを保持するものである。外部スタックは LIFO（Last-in, First-out）ファイルとして構成される。

データは PUSH および POP 命令により、CPU の指定レジスタからスタックへ、またスタックから CPU の指定レジスタへ転送される。

データのスタックからの取り出し（POP）は一番最後に押し込んだ（PUSH）データから行われる。このスタックは、多重レベルの割り込み、無限のサブルーチン・ネスティング、あるいは種々の形のデータ操作などを簡単にする場合に有効である。

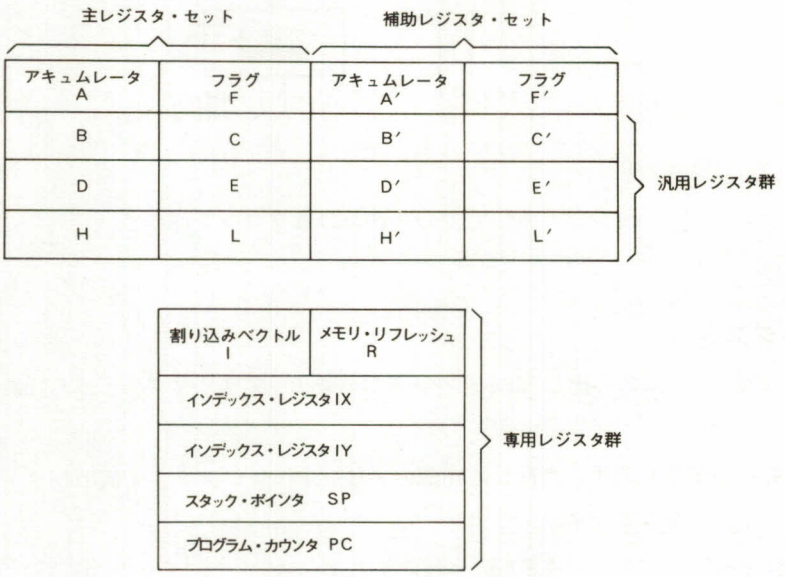


図２－２　Ｚ－80　ＣＰＵのレジスタ構成

3．**インデックス・レジスタ（ＩＸ&ＩＹ）**

インデックス・モードのアドレシング用として16ビットの基準アドレスを保持する２個の独立したインデックス・レジスタＩＸ、ＩＹがある。

このモードでは、インデックス・レジスタはデータを出し入れするメモリ領域を指定するための基準アドレスとして使用される。

インデックス・アドレシング命令では、このレジスタの内容に１バイトのディスプレイスメント（偏位値）を加算した値が実効アドレスとなる。この偏位値は２の補数の符号付整数で与えられる。

このアドレシング・モードは種々のプログラム、とくに、データ・テーブルを参照するといったプログラムなどによく使用される。

4．**割り込みページ・アドレス・レジスタ（Ｉ）**

Ｚ－80　CPUには、割り込みに応じてどのメモリの位置へでも、イン・ダイレクト・コール（間接サブルーチン・ジャンプ）できるモードがある。

この目的のために、Ｉレジスタが用意されていて、間接アドレスの上位８ビットにこのレジスタの内容が相当する。下位８ビットには、割り込みをかけてきたデバイスに対応するアドレスが当てられる。この方式によれば、割り込み処理ルーチンを動的にメモリのどこへでも配置でき、極めて短いアクセス時間で処理ルーチンへ飛ばせ得るという特長をもっている。

5．**メモリ・リフレッシュ・レジスタ（Ｒ）**

Ｚ－80　CPUはメモリ・リフレッシュ・カウンタを内蔵しているので、スタティック・メモリと同じ手軽さでダイナミック・メモリを使用できる。

この７ビットのＲレジスタは各命令のフェッチごとに自動的にインクリメントされる。

ＲカウンタのデータはCPUがフェッチした命令をデコードし、実行している間に、リフレッシュ 制御信号と同期してアドレス・バスの下位に乗せられる。

このリフレッシュのモードは、プログラマがとくに気をつかう必要もなく、またCPUの動作を遅らせるものでもないという特長がある。Ｒレジスタにテストの目的でプログラムによってデータをロードすることもできるが、普通は使用しない方がよい。

リフレッシュの期間、アドレス・バスの上位８ビットにはＩレジスタの内容が出力する。

アキュムレータとフラグ・レジスタ

CPUは２組の独立した８ビットのアキュムレータと、それと組み合わさった２組の８ビットのフラグ・レジスタをもっている。

アキュムレータは８ビットの算術、論理演算の結果を保持する。一方、フラグ・レジスタは８ビットあるいは16ビットの演算結果の状態、たとえば結果が零に等しいか否かを示す。

プログラマによって１つの交換命令を用いて、アキュムレータとフラグの対（ペア）のうちどちらか使いやすい方を選べばよい。

汎用レジスタ

2組の対になった汎用レジスタ群があり、それぞれ単独で8ビットのレジスタとして使用でき、また16ビットのレジスタ・ペアとしても使用できる。一方のセットとしてBC、DE、HLがあり、他方、BC′、DE′、HL′のセットがある。

プログラムによって、どの時点においても交換命令を用いてどちらかの側のレジスタ群を作業用として使用することができる。システム内で高速の割り込み応答が要求される場合、この手法を使ってアキュムレータ、フラグ類、汎用レジスタ群の内容を他方へすばやく退避させてもよいだろう。

1つの簡単な交換命令を用いるだけでルーチン間の移行が行える。

これにより割り込み、あるいはサブルーチン処理の期間、レジスタの内容を外部スタックへ移したり、戻したりする必要がなくなるので、割り込みサービス時間を大幅に短縮するのに役立つ。

これらの汎用レジスタ群は、どのような幅広い用途に対しても利用し得るものである。

単純なプログラム、とくにROMベースのシステムなどで、簡単な読み出し/書き込みメモリが必要な場合に、汎用レジスタで代用すればよい。

2．2　算術、論理演算ユニット（ALU）

8ビットの算術、論理演算命令はCPU内のALUで実行される。

ALUは各レジスタと内部バスとを接続しており、それらの間でデータの送受を行う。

ALUに関連した機能には次のようなものがある。

Add	（加　　算）
Subtract	（減　　算）
AND	（論 理 積）
OR	（論 理 和）
XOR	（排他的論理和）
Compare	（比　　較）
Shift Left, Right	（左、右シフト）
Rotates arithmetic, logical	（算術的、論理的ローテイト）
Increment	（インクリメント；＋1）
Decrement	（デクリメント；－1）
Set bit	（ビット・セット）
Reset bit	（ビット・リセット）
Test bit	（ビット・テスト）

2．3　命令レジスタ、ＣＰＵ制御

各命令はメモリから読み出されてきて、命令レジスタに保持され、デコードされる。制御部はこの制御を行うとともに、レジスタ群からのまたはそれらへの、データの読み込み、書き出しに必要な制御信号を発生している。

さらに、ALU の制御信号や必要な外部制御信号を作り出す。

第3章　Z-80 CPU端子説明

Z－80　CPUは標準型40ピンDIPパッケージに納められている。入出力端子配置図を図3－1に示す。以下にそれぞれの機能を説明する。

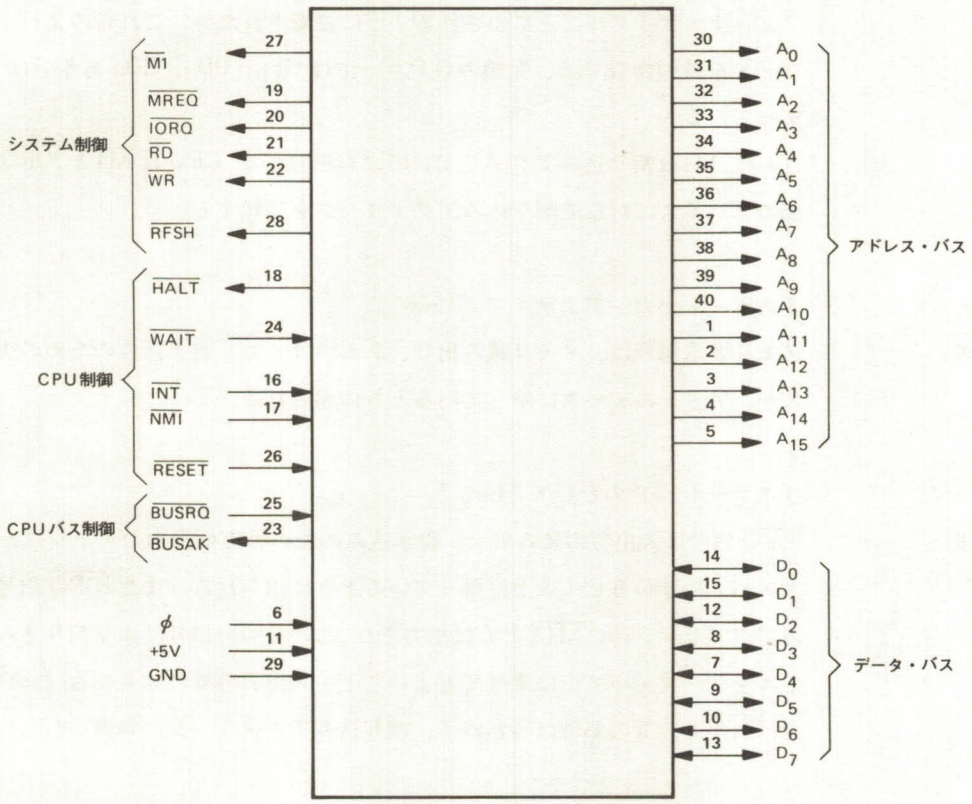


図3－1　Z－80　CPUの端子配置図

A_0－A_{15}
（アドレス・バス）

3ステート、アクティブ〝High〟。

A_0－A_{15}は16ビットのアドレス・バスである。このアドレス・バスにより、メモリ（最大64Kバイト）内のデータや入出力デバイスのデータの送受のためのアドレス指定を行う。入出力アドレシング用として下位8ビットを用いる。ユーザは直接256の入力ポートまたは256の出力ポートを選択できるようになっている。A_0がLSB（最下位ビット）となる。リフレッシュ期間には、アドレス・バスの下位7ビットにリフレッシュ用の実効アドレスが乗せられる。

D_0-D_7
（データ・バス）

3ステート入出力、アクティブ〝High〟。
D_0-D_7は8ビットの双方向性データ・バスである。データ・バスはメモリ、入出力間、あるいはそれらの相互間で、データ交換を行うために用いる。

$\overline{M1}$
（マシン・サイクル1）

出力、アクティブ〝Low〟。
$\overline{M1}$は、現在のマシン・サイクルが、命令実行中のOPコードのフェッチ・サイクルであるときに出力する。2バイトのOPコードの実行時には、$\overline{M1}$はOPコードのフェッチ・サイクルごとに出されることに注意されたい。これらの2バイトのOPコードを持つ命令では、先頭のOPコードはCB_H、DD_H、ED_HあるいはFD_Hで始まっている。
さらに、$\overline{M1}$は割り込みアクノリッジ時にも出力する。CPUは$\overline{M1}$と$\overline{IORQ}$により外部のデバイスに対して割り込みアクノリッジを通知する。

$\overline{MREQ}$
（メモリ要求）

3ステート出力、アクティブ〝Low〟。
メモリ要求信号は、メモリ読み出し、あるいはメモリ書き込みのための実効アドレスが、アドレス・バスに乗っているときに出される。

$\overline{IORQ}$
（入出力要求）

3ステート、アクティブ〝Low〟。
$\overline{IORQ}$信号は入出力の読み出し、書き込みのための実効入出力アドレスが、アドレス・バスの下位8ビット上に乗っているときに出される。また$\overline{IORQ}$信号は割り込みアクノリッジ時に$\overline{M1}$とともに出力され、この2つの信号により割り込み応答ベクトルをデータ・バス上に乗せてもよいことを入出力装置に知らせる。この$\overline{M1}$の期間では入出力装置の処理は行われず、割り込みアクノリッジの処理が行われる。

$\overline{RD}$
（メモリ読み出し）

3ステート出力、アクティブ〝Low〟。
$\overline{RD}$はCPUがメモリ、あるいは入出力デバイスからデータを受け入れられる期間で出される。指定された入出力デバイス、あるいはメモリのデータは、この信号でゲートして、CPUデータ・バスに乗せるとよい。

$\overline{WR}$
（メモリ書き込み）

3ステート出力、アクティブ〝Low〟。
$\overline{WR}$は、CPUデータ・バスに、指定したメモリあるいは入出力デバイスにストアするべきデータが乗っているときに出される。

$\overline{\text{RFSH}}$
（リフレッシュ）

出力、アクティブ〝Low〟。
$\overline{\text{RFSH}}$は、ダイナミック・メモリのためのリフレッシュ・アドレスがアドレス・バスの下位7ビットに乗っているときに出される。なお、ダイナミック・メモリをリフレッシュ（リフレッシュ読み出し）する場合は$\overline{\text{MREQ}}$信号も必要である。

$\overline{\text{HALT}}$
（ホールト）

出力、アクティブ〝Low〟。
$\overline{\text{HALT}}$は、CPUがHALT命令を実行し、ノン・マスカブルあるいは、マスカブルな割り込み待ちとなったときに出される。
ホールト時には、CPUはNOP命令を実行することによりリフレッシュ信号を出し続けている。

$\overline{\text{WAIT}}$
（ウエイト）

入力、アクティブ〝Low〟。
この入力信号を用いて、メモリあるいは入出力デバイスがデータの送出の用意ができていない旨をZ－80　CPUに伝える。この信号がアクティブである限り、CPUはウェイト状態を続ける。この期間リフレッシュ信号は出力しないので注意されたい。
この信号を用いることによりどのような動作速度のメモリや入出力デバイスに対してもCPUを同期させることができる。

$\overline{\text{INT}}$
（割り込み要求）

入力、アクティブ〝Low〟。
割り込み要求信号は入出力デバイスから発せられる。ソフト的に（プログラムで）、割り込みの許可フラグ（IFF）がセットしてあり、$\overline{\text{BUSRQ}}$信号が非アクティブならば、割り込み要求は実行中の命令が終わり次第受け付けられる。
CPUは割り込みを受け付ければ、アクノリッジ信号（M1期間の$\overline{\text{IORQ}}$）を次の命令サイクルの始めで送出する。第5章に詳細を示すがマスク可能な割り込みは3種ある。

$\overline{\text{NMI}}$
（マスク不能割り込み）

入力、立ち下がりエッジ検知。
ノン・マスカブル割り込み要求は$\overline{\text{INT}}$より高位の優先順位を持っていて、現在実行中の命令の最後のTサイクルの立ち上がりまでに入力していると、その命令完了後受け付けられる。これは割り込みの許可フラグの状態には関係しない。$\overline{\text{NMI}}$入力でCPUは自動的に、0066_Hの番地からリスタートする。プログラム・カウンタの内容は、割り込みがかけられた元のプログラムへ戻れるように、外部スタックへ自動的に退避される。
連続してWAITサイクルがあれば、NMIは待たされ、また$\overline{\text{BUSRQ}}$は$\overline{\text{NMI}}$より優

先順位が高い。

$\overline{\text{RESET}}$　　入力、アクティブ〝Low〟。

$\overline{\text{RESET}}$ 入力によりプログラム・カウンタは零となり、CPU は初期化される。このとき次の状態となる。

1）割り込みの許可フラグがリセットされる。

2）レジスタ I ＝ 00_H にセットされる。

3）レジスタ R ＝ 00_H にセットされる。

4）割り込みはモード 0 にセットされる。

リセット期間中、アドレス・バスとデータ・バスは高インピーダンスとなり、すべての制御出力は非アクティブ状態となる。

$\overline{\text{BUSRQ}}$　　入力、アクティブ〝Low〟。

（バス要求）　　バス要求信号により、CPU のアドレス・バス、データ・バスおよび 3 ステート出力の制御線は、他のデバイスがバスを使用できるようにするため高インピーダンスとなる。

$\overline{\text{BUSRQ}}$ がアクティブになったとき、その時点での CPU のマシン・サイクルが終わった瞬間に、バス線は高インピーダンスとなる。

$\overline{\text{BUSAK}}$　　出力、アクティブ〝Low〟。

（バス・アクノリッジ）　　バス・アクノリッジは、CPU のアドレス・バス、データ・バスおよび 3 ステート制御バスが高インピーダンスとなり、外部デバイスがこれらのバスを使用できるようになった時点で、要求のあったデバイスに対して出力する。

Φ　　単相の クロック入力。

第4章　Z-80 CPU タイミング

Z－80　CPUは基本的な操作を組み合わせ、1ステップずつ命令を実行していく。その操作は次のものから構成される。

メモリ・読み出し、書き込み、
入出力デバイス・読み出し、書き込み
割り込み、アクノリッジ

すべての命令はたんにこれらの基本的操作を組み合わせたものである。この基本操作は3～6クロックで行われる。これらの操作はCPUを外部デバイスの速度に同期させるために延長することもできる。基本のクロック期間をTサイクルとし、基本操作をMサイクル（マシン・サイクル）とする。図4－1に、ある命令に対して特定のMとTサイクルの配列がどのようになっているかを例として示す。ただし、この例では、3マシン・サイクル（M1、M2、M3）の命令について示してある。

最初のMサイクルはどの命令でも命令のフェッチ・サイクルである。このフェッチ・サイクルには、4、5、6Tサイクルの3種があるが、ウェイト信号（次節で詳述する。）を用いると、このサイクル数を変えることもできる。このフェッチ・サイクル（M1）は、次に実行すべき命令のOPコードをフェッチするための期間である。

後続するMサイクルで、CPUとメモリ、入出力デバイス間のデータ転送が行われる。これらも基本的には3～5Tサイクルであるが、外部デバイスと同期させるために入力する $\overline{\text{WAIT}}$ 信号によりサイクル数は変化する。

次項から基本マシン・サイクル内でのタイミングについて説明する。

各命令の詳細なタイミングを第11章に示す。

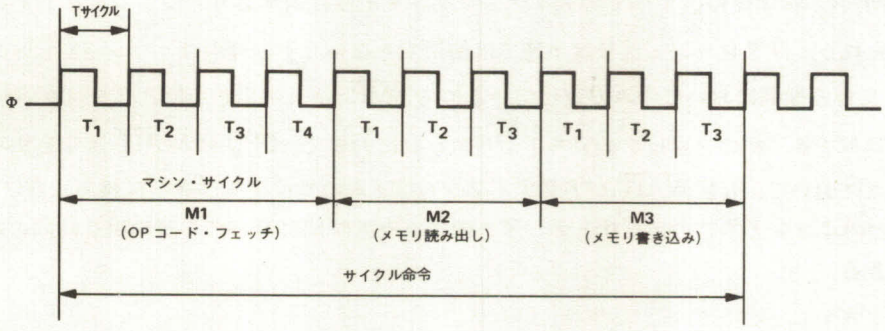


図4－1　CPU基本タイミング例

すべてのCPUタイミングは、図4－2に示すような、いくつかの非常に簡単なタイミング図に分解できる。ウェイトを含む場合と含まない場合の基本動作タイミングを以下において示す。(ウェイト状態は遅いメモリや入出力デバイスにCPUを同期させるために付加される。)

4．1　命令フェッチ

図4－2にM1サイクル（OPコード・フェッチ）のタイミングを示す。プログラム・カウンタの内容はM1サイクルの先頭でアドレス・バス上に乗せられ、クロックの半サイクル後$\overline{\text{MREQ}}$がアクティブとなる。このときすでに、メモリに対するアドレス信号は安定しているので、$\overline{\text{MREQ}}$の立ち下がりエッジをダイナミック・メモリへのチップ・イネーブル・クロックとして直接使用することができる。

$\overline{\text{RD}}$もメモリから読み出すデータをCPUのデータ・バスに乗せてもよいことを指示するためにアクティブとなる。

CPUはデータ・バス上にあるメモリからのデータを、T_3のクロックの立ち上がりエッジでサンプルする。その立ち上がりエッジで、$\overline{\text{RD}}$、$\overline{\text{MREQ}}$信号が切り替わるが、CPUはデータのサンプリングを、$\overline{\text{RD}}$が非アクティブになる前に行っている。

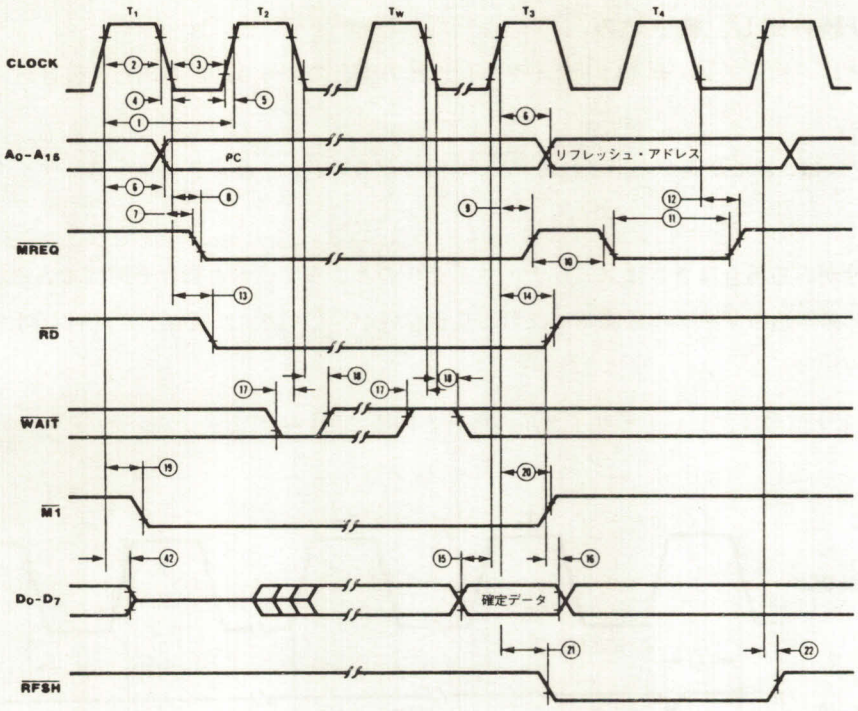
フェッチ・サイクルのクロックT_3、T_4のステートはダイナミック・メモリをリフレッシュするために使用する。(この期間においては、CPU内ではフェッチされた命令のデコードとその実行を行っているので、外部に対する他の動作は行われない。)

T_3、T_4の期間で、アドレス・バスの下位7ビットにメモリのリフレッシュ・アドレスが乗せられ、$\overline{\text{RFSH}}$信号がアクティブとなる。この信号により、すべてのダイナミック・メモリのリフレッシュがこの期間内に行われるように指示する。

注意する点として、データ・バスに接続されているはずの他のメモリ・セグメントからデータが乗らないように、リフレッシュの期間には$\overline{\text{RD}}$信号は出されていない、ということがあげられる。

リフレッシュ期間の$\overline{\text{MREQ}}$は、すべてのダイナミック・メモリに対するリフレッシュのための読み出しを行うために用いられる。リフレッシュ信号は単独では使用できない。その理由はリフレッシュ・アドレスは、$\overline{\text{MREQ}}$が出力している期間においてのみ安定しているので、$\overline{\text{MREQ}}$と併用しなければならないからである。

T_2およびそれに続く各Twのクロックの立ち下がりエッジにおいて、CPUは$\overline{\text{WAIT}}$信号をサンプルする。このサンプル時点において、もし$\overline{\text{WAIT}}$がアクティブならば、次のサイクルはさらに待ち状態になる。この方法を用いると、どのようなタイプのメモリ・デバイスのアクセス時間に対しても読み出し時間を延長することによって対処できる。



(注) Tw(ウエイト・サイクル)は、低速のメモリを使用する場合、必要に応じて発生させます。

図4－2　命令OPコードフェッチ

4．2　メモリ読み出し、書き込み

図4－3はOPコード・フェッチ（M1サイクル）以外の場合のメモリの読み出し、書き込みサイクルのタイミング図である。

これらのサイクルは、メモリ側から出される $\overline{WAIT}$ 信号がなければ、通常3クロック・サイクルの長さである。

$\overline{WAIT}$ 要求信号がある場合はさきにフェッチ・サイクルのところで述べた動作と同様である。実際には、メモリの読み出しと書き込みサイクルは同時にはおこり得ないが、この図では便宜上これらを同一図上に書いてある。

$\overline{MREQ}$ 信号と $\overline{RD}$ 信号はフェッチ・サイクルの場合と同様に用いられる。

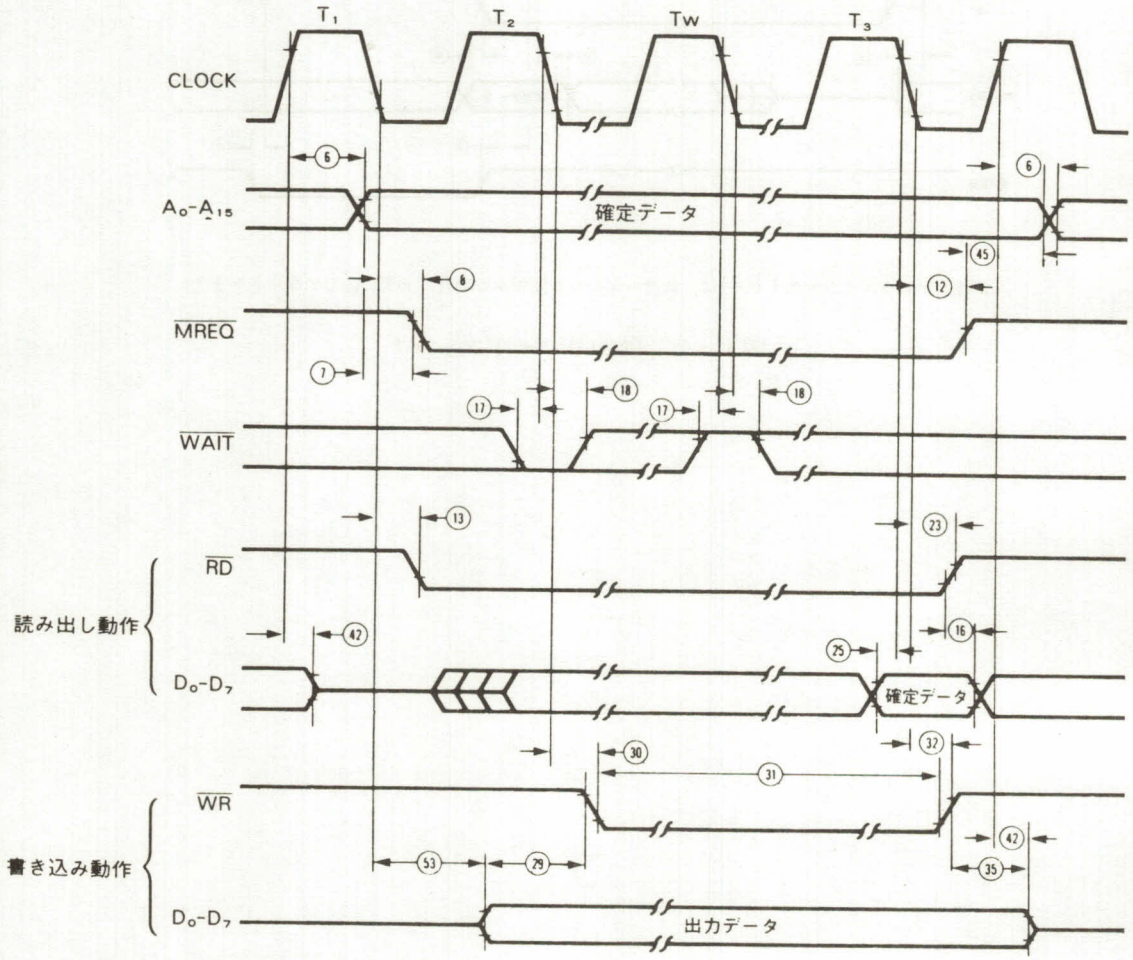


図4－3　メモリ読み出し、メモリ書き込みサイクル

メモリ書き込みサイクルにおいても、$\overline{MREQ}$はアドレス・バスが安定したときにアクティブになるので、これをダイナミック・メモリのチップ・イネーブル用に直接使用することができる。

$\overline{WR}$出力はデータ・バスが安定になったときアクティブとなるので、これを直接メモリの読み出し、書き込みパルスとして使用してもよい。実際上、この$\overline{WR}$出力はほとんどのタイプの半導体メモリの読み出し、書き込み信号として使用できる。

さらに、$\overline{WR}$信号はアドレス・バス、データ・バスの内容が変化するサイクルの前のTサイクルの中央で非アクティブとなるので、実際上どんなタイプの半導体メモリのオーバ・ラップ条件に対しても、問題なく使用できる。

4．3　入力、出力サイクル

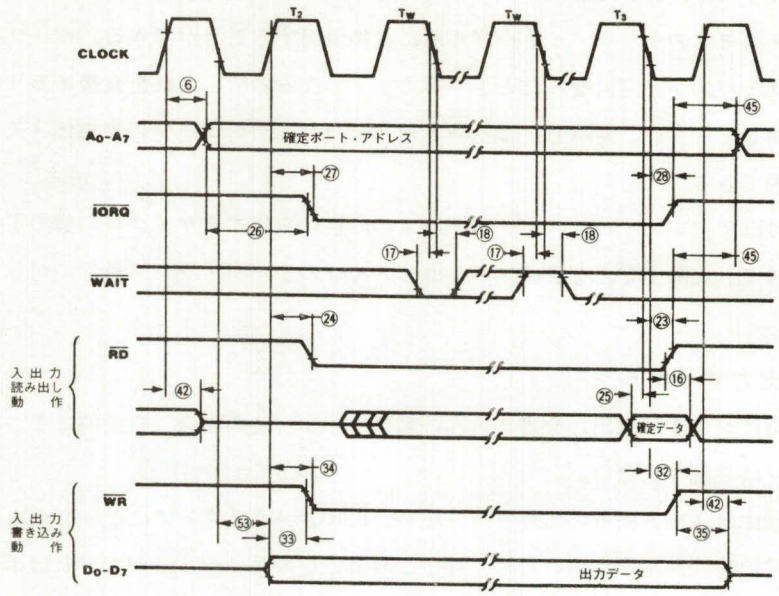
図4－4に入出力に対する読み出しや書き込みの動作を示す。この場合、自動的に1つの待ち状態Twサイクルが挿入される点に注意されたい。

このようにした理由は入出力装置の処理を行う場合、$\overline{IORQ}$がアクティブになってからCPUが$\overline{WAIT}$信号をサンプルするまでの時間が非常に短いため、もしこの余分なステートT_Wがなければポート・アドレスをデコードしてから$\overline{WAIT}$信号を起動していたのでは時間が足りないからである。

さらに、この待ち状態を設けなければ、MOS入出力デバイスを組み入れて、CPUを最高速で動作させるということもむずかしくなる。

このT_Wの期間に$\overline{WAIT}$要求信号をサンプリングする。メモリの読み出しの場合と同様に、入出力読み出し動作においても指定したポートをデータ・バスに接続するためには$\overline{RD}$信号を使用すればよい。入出力書き込み動作において、入出力ポートのクロックとして$\overline{WR}$信号を用いるが、充分なオーバ・ラップ時間が自動的に作られるのでこの立ち上がりエッジをデータ・クロックとして使用することができる。

$\overline{WAIT}$信号が入った場合に、余分の待ち状態サイクルが追加される動作はさきに述べた例（図4－2）と同様である。



（注）Tw* ＝CPUにより自動的に挿入される1ウエイト・サイクル

図４－４　入出力サイクル

４．４　バス要求/アクノリッジ・サイクル

図４－５にバス要求/アクノリッジ・サイクルのタイミングを示す。

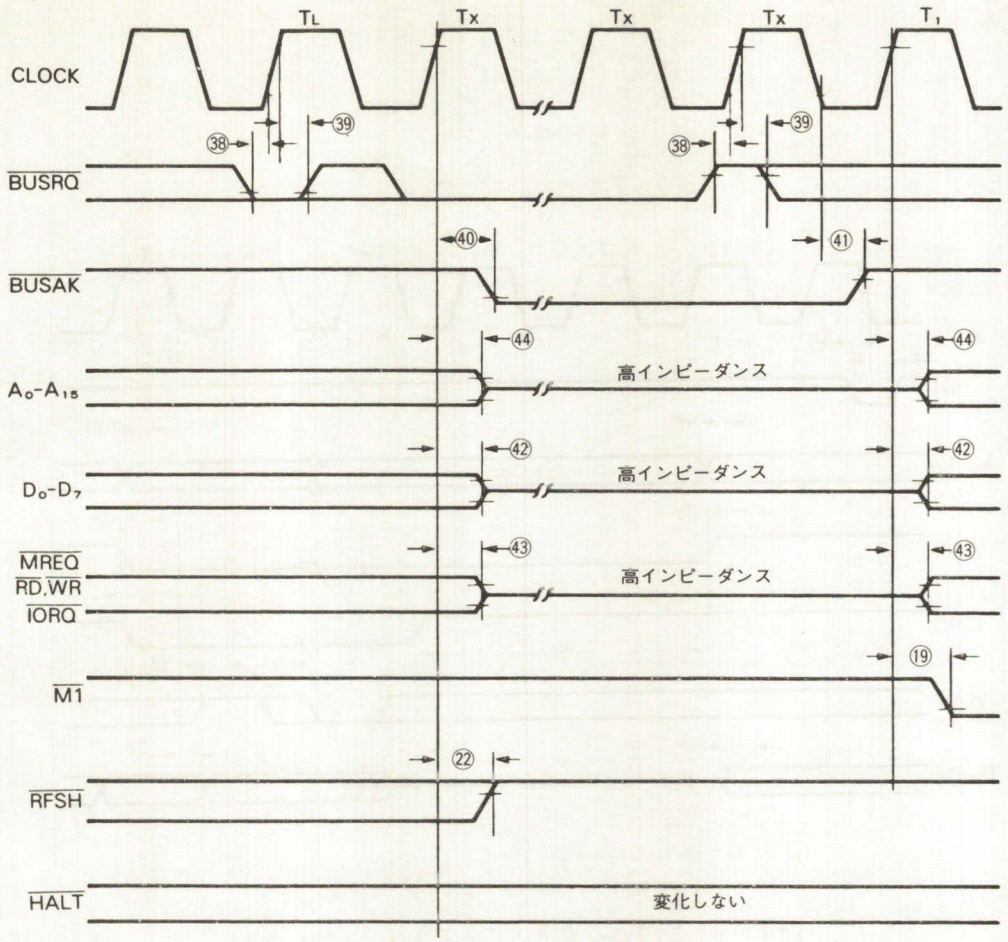
CPU は、各マシン・サイクルの最終クロックの立ち上がりエッジで $\overline{\text{BUSRQ}}$ 信号を調べ、もし $\overline{\text{BUSRQ}}$ 信号がアクティブならば、次のクロック・パルスの立ち上がりエッジでアドレス、データ、３ステート制御線を高インピーダンスにする。この時点以降、外部のデバイスによりこれらのバスを制御でき、メモリと入出力間のデータ転送に使用することが可能になる。（これは一般にサイクル・スティール方式のダイレクト・メモリ・アクセス　DMA として知られている。）

バス要求に対する CPU の応答遅れは大きくても１マシン・サイクル長であり、外部のコントローラは必要な時間だけバスを制御できる。

しかしダイナミック・メモリを使用していて、長時間 DMA サイクルを続けたい場合には、外部の制御回路によりダイナミック・メモリのリフレッシュが行えるようにしておかなければならない。

このような状態は、多数のデータ・ブロックを DMA で転送する場合にのみ生じる。

さらに、バス利用可の状態では、$\overline{\text{NMI}}$ 信号あるいは $\overline{\text{INT}}$ 信号による割り込みは無視される点に注意する必要がある。



（注）T_L＝マシン・サイクルの最終ステートを示します。

T_X＝バス要求を出力したデバイスに使用されるクロック・サイクル。

図４－５　バス要求／アクノリッジ・サイクル

４．５　割り込み要求/アクノリッジ・サイクル

図４－６に、割り込みサイクルに関連したタイミング例を示す。

CPU は各命令実行の最終クロックの立ち上がりエッジで割り込み信号 $\overline{INT}$ をサンプルする。

CPU 内部の割り込みイネーブル・フリップ・フロップがセットされていなければこの信号は受け付けられない。また、$\overline{BUSRQ}$ 信号がアクティブであるときも同様である。

割り込み信号を受け付けると、CPU は特別なＭ１サイクルを発生する。

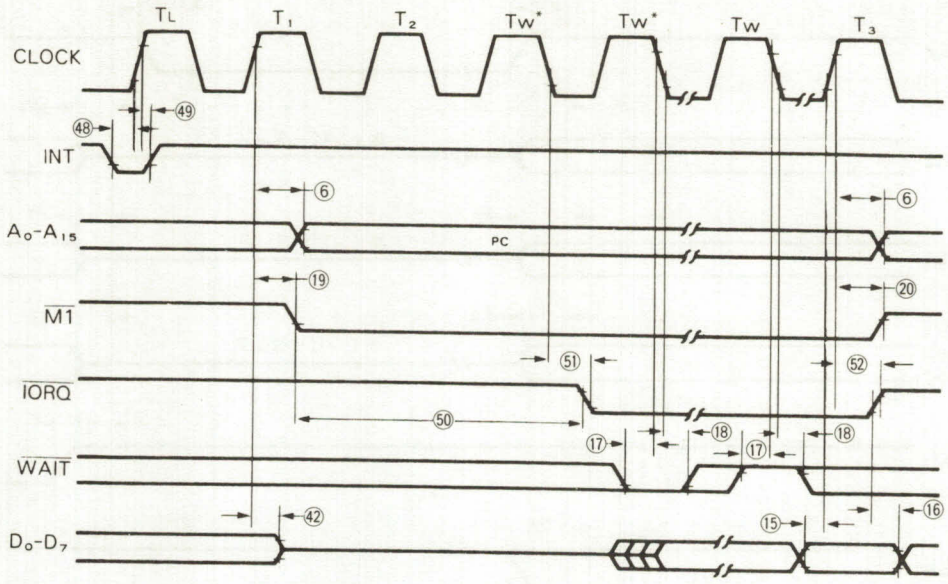
このＭ１サイクルの期間中に、$\overline{IORQ}$ 信号が通常の $\overline{MREQ}$ 信号に代ってアクティブとなる。これによって、割り込みの要求を出したデバイスからデータ・バス上に、８ビットのベクトルを乗せてもよい状態になる。これを割り込みアクノリッジという。

このサイクルには、自動的に待ち状態２サイクルが追加されるので注意を要する。

これらのサイクルの追加によって、リップル・プライオリティの割り込み方式が使用できるようになる。待ち状態として２サイクルあれば、リップル信号が安定に伝わり、入出力デバイスが出した応答ベクトルを CPU が確実にとり込むための時間としては、これで充分である。

第８章で、CPU 内での割り込み応答ベクトルの処理形式について詳述してあるので参照されたい。

図４－６Ａにプログラマブル・カウンタを使用してアクノリッジ時間を延長する方法を示す。



（注１）T_L＝前の命令の最後のステート

（注２）CPUより自動的に２つのウエイト・サイクルが挿入されます。

図４－６　割り込み要求／アクノリッジ・サイクル

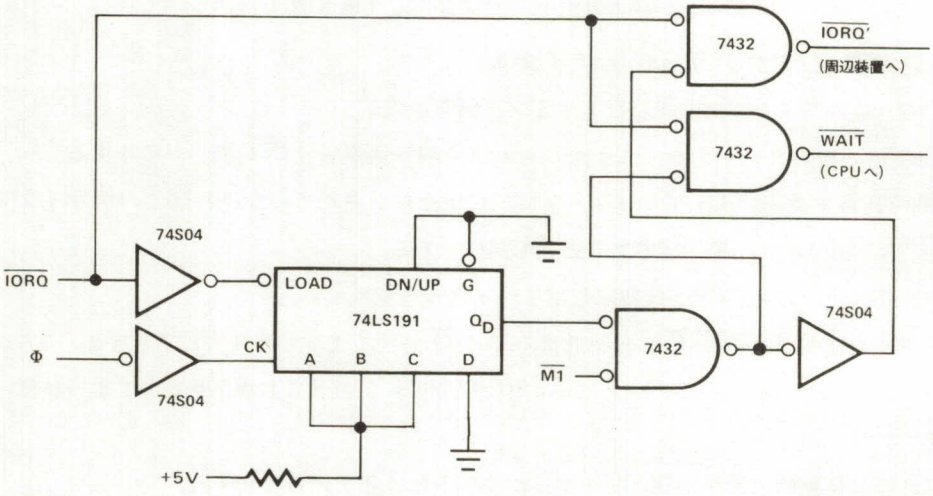


図４－６Ａ　ウェイト・ステートを含む延長された割り込みアクノリッジ

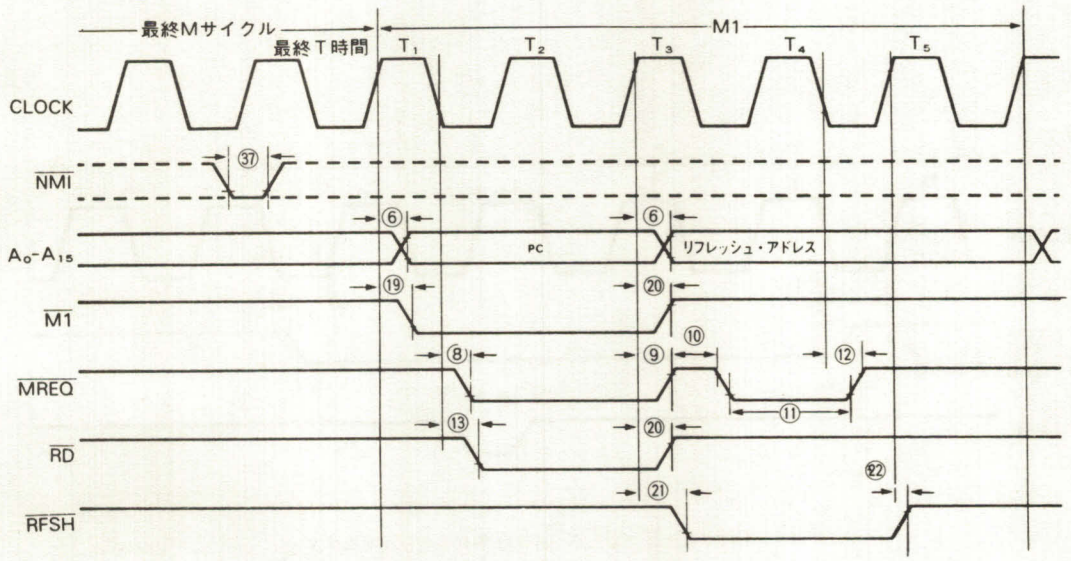
4．6　ノン・マスカブル割り込みの応答

図4－7にノン・マスカブル割り込みのある場合の要求/アクノリッジ・サイクルの例を示す。

この信号は$\overline{\text{INT}}$と異なり立ち下がりエッジが有効信号となっており、さらに各命令のどのタイミングで入力してもよい。ただし、現命令の実行完了後に割り込みアクノリッジ・サイクルに入るためには、少なくとも命令の最後のTサイクルまでに、$\overline{\text{NMI}}$が立ち下がっていなければならない。この信号線は$\overline{\text{INT}}$信号よりも優先順位が高く、またソフトウエアによってその機能を無効にすることもできない。

CPUはこのノン・マスカブル割り込みに対しては通常のメモリ読み出し動作と同じ応答をする。異なる点は、CPUがその時点でのデータ・バス上の内容を無視して自動的にＰＣ（プログラム・カウンタ）の内容を外部スタックに退避した後、0066H番地へジャンプする点である。

この割り込み方式を使用する場合、ノン・マスカブル割り込みのためのサービス・ルーチンは0066H番地から書いておかねばならない。



（注）$\overline{\text{NMI}}$は非同期入力なので、次のマシン・サイクルで確実にマスク不能割り込み応答サイクルに入るためには、$\overline{\text{NMI}}$の立ち下りエッジが最終Tサイクルの始まりのクロックの立ち上がりエッジより先にアクティブにならなければなりません。

図4－7　ノン・マスカブル割り込み要求動作

4．7　ホールト状態解除

プログラム内のHALT 命令ではCPU はNOP 命令を実行し続けるが、割り込み入力があれば、この状態から抜け出る。(この場合の割り込みはノン・マスカブル割り込みか、割り込みフリップ・フロップがセットされている場合のマスカブル割り込みのいずれでもよい。)

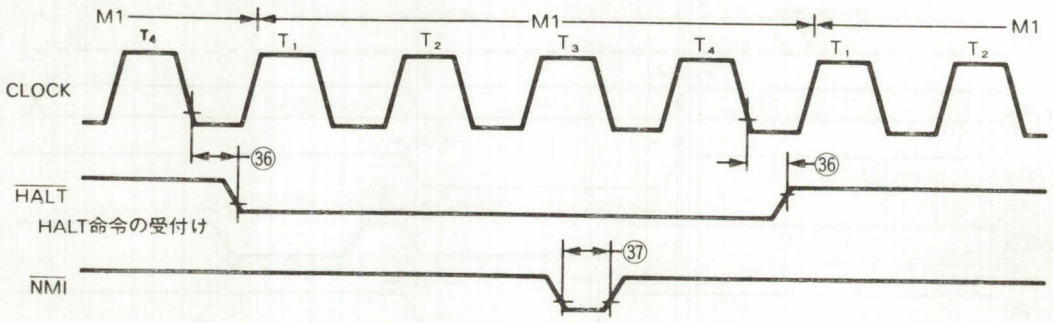
これら2種の割り込み線を図4－8に示すように、各T_4サイクルのクロックの立ち上がりエッジで調べて上述のいずれかであれば、CPU は次のクロックの立ち上がりエッジで、ホールト状態から解除される。

次のサイクルで、受け付けた割り込みの種類に応じてそれぞれの割り込みのアクノリッジ・サイクルに入ることになる。

もし、同時にノン・マスカブルとマスカブルの割り込み信号が入った場合には、優先順位の高いノン・マスカブル割り込みが受け付けられる。

ホールト状態でNOP 命令を実行させている理由は、メモリ用のリフレッシュ信号を出し続けるためである。

この状態での各サイクルは、メモリからのデータを無視する以外は通常のM1 (フェッチ) サイクルと同様であり、CPU 内部で自動的にNOP 命令を実行している。HALT 命令を実行すると、CPU はホールト状態に入ったことを外部に対して知らせるために$\overline{\text{HALT}}$信号がアクティブとなる。



(注) $\overline{\text{INT}}$アクティブでもホールト状態から抜け出せます。

図4－10　ホールト・サイクル

4．8　リセット・サイクル

CPUが正しくリセットされるためには、$\overline{\text{RESET}}$が3クロック・サイクル以上の間アクティブでなければならない。$\overline{\text{RESET}}$がアクティブになっている間、アドレスとデータ・バスはフロート状態に、制御出力は非アクティブ状態になる。$\overline{\text{RESET}}$が非アクティブになると、2Tサイクル後にCPUは通常の動作を開始する。$\overline{\text{RESET}}$は、PCレジスタをクリアするので、最初のOPコード・フェッチは0000番地から行われる。（図4－9参照）

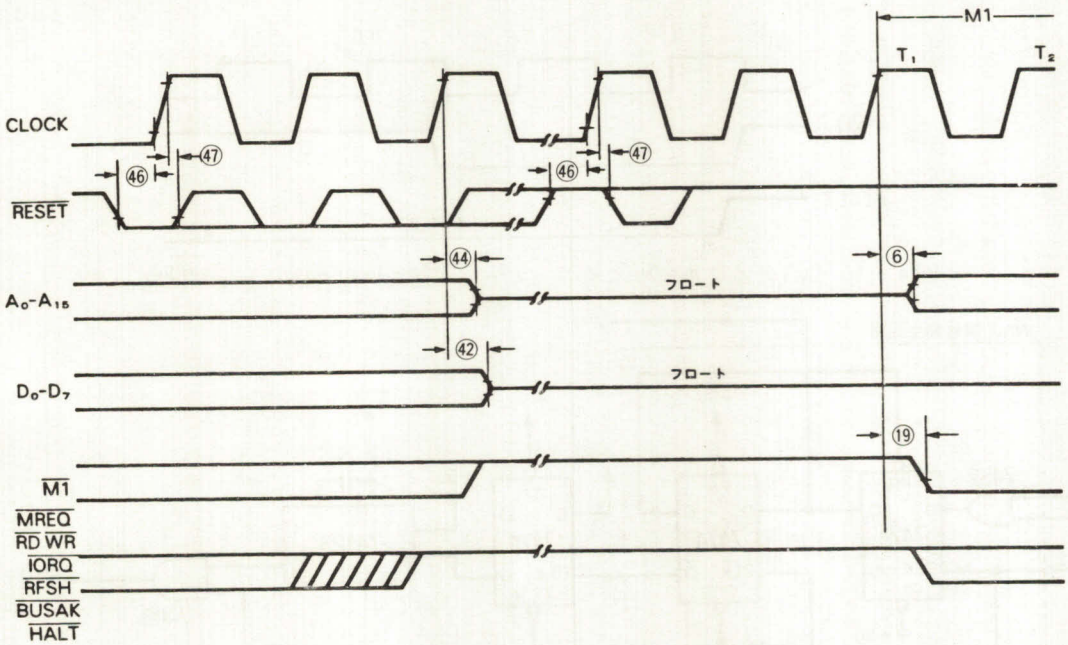


図4－9　リセット・サイクル

動作中のCPUをリセットするとき、CPU動作が中断されると、RAMの内容を破壊することがある。これを防ぐために、次のタイミングにて$\overline{\text{RESET}}$信号を入力する必要がある。

(1)　M1サイクルにウエイト・ステートがない場合

M1サイクルのT_2ステートのクロック立ち上がりにて、$\overline{\text{RESET}}$信号のサンプリングを開始するように、$\overline{\text{RESET}}$信号を入力する。

このようなタイミングをもつ$\overline{\text{RESET}}$信号を、発生する回路例を図4—10に示す。

(2)　M1サイクルにウエイト・ステートがある場合

M1サイクルのT_3ステートのクロック立ち上がりにて、$\overline{\text{RESET}}$信号のサンプリングを開始するように、$\overline{\text{RESET}}$信号を入力する。

1ウエイト・ステートが挿入される場合、図4—11に示す回路にて、このタイミングをもつ$\overline{\text{RESET}}$信号を発生することができる。図4—11の回路では、インタラプト・アクノリッジ・サイクルにて、$\overline{\text{RESET}}$信号を発生しないようになっている。

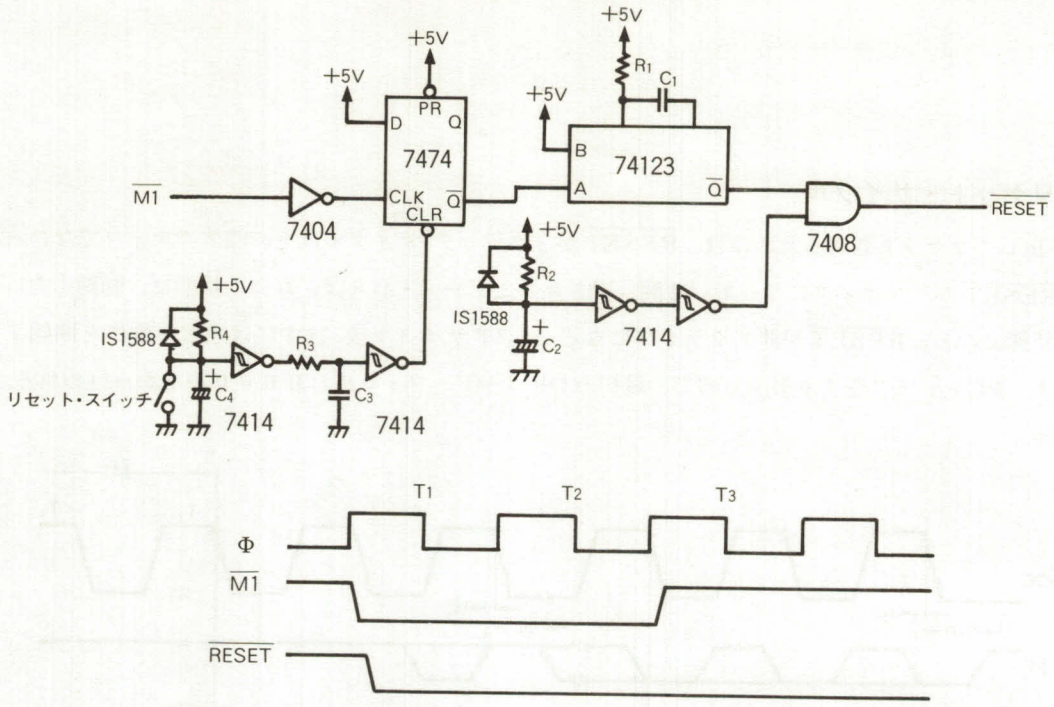


図4－10　M1サイクルにウエイト・ステートがない場合のリセット回路

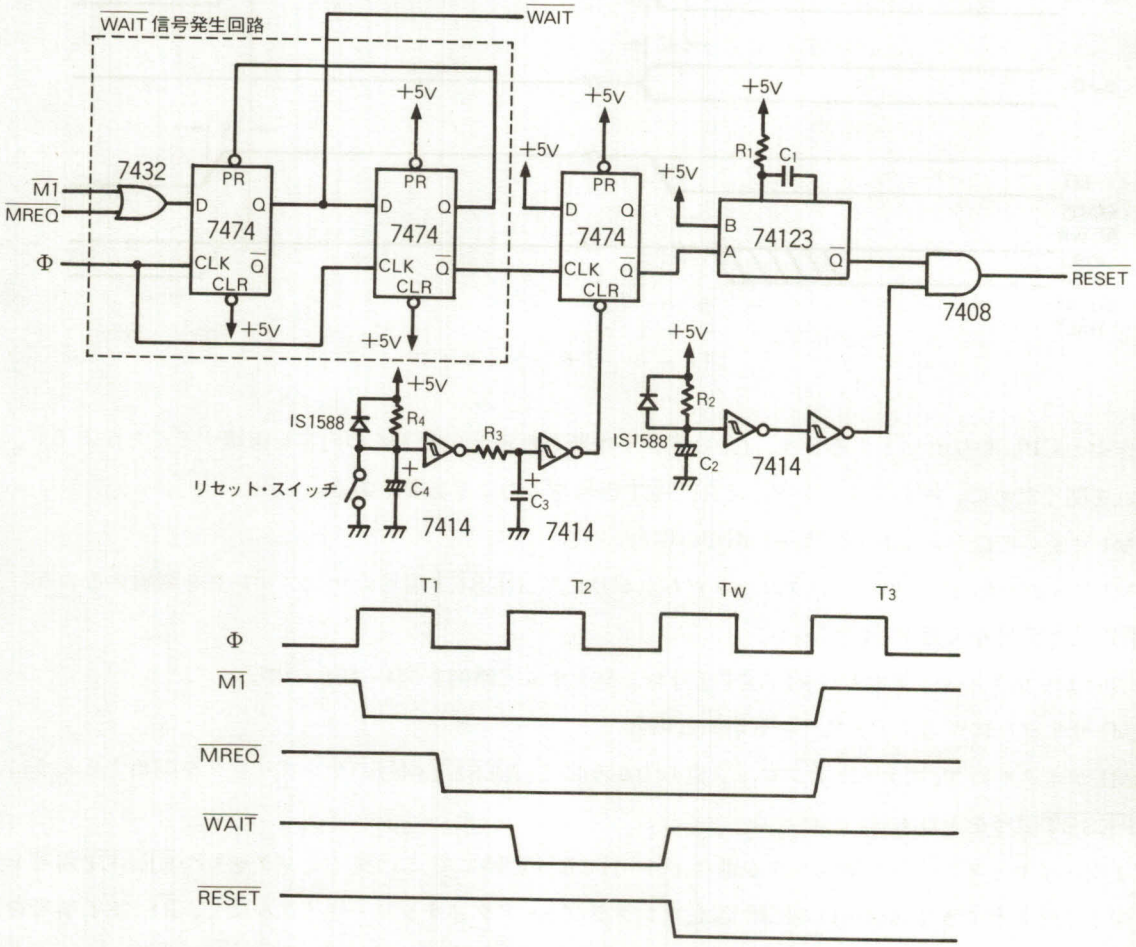


図4－11　M1サイクルに1ウエイト・ステートがある場合のリセット回路

第5章　Z-80 CPU 命令セット

Z－80 CPU には158種の命令があり、これらの命令には 8080A CPU の78種の命令がすべて含まれている。命令群は大別して次のようなグループに分けられる。

- ロード、交換（Exchange）
- ブロック転送、ブロック・サーチ
- 算術、論理処理
- ローテイト、シフト
- ビット操作（セット、リセット、テスト）
- ジャンプ、コール、リターン
- 入力、出力
- CPU 制御

5．1　命令の型について

ロード命令は、データと CPU 内部のレジスタ間で、あるいは CPU レジスタと外部メモリ間で、データを転送する機能を持っている。

この命令では、データを取り出すソースの番地と、それを格納するデスティネーションの番地を指定しなければならない。

ロード命令ではその実行後においてソースの内容は変化しない。

ロード命令の例として、レジスタＣからレジスタＢへデータを移すというような汎用レジスタ間のデータ転送命令がある。

また各 CPU レジスタあるいは外部メモリに直接データをロードする命令や、CPU レジスタとメモリ間のデータの転送を行う命令がある。

交換命令は２つのレジスタ内のデータを入れ替える機能がある。

Z－80にはブロック転送というユニークな命令群がある。１命令で、どのようなサイズのメモリ・ブロックでも、メモリの他の領域に移し替えられる。このブロック転送の命令群は、大量のデータを処理する必要のあるときに非常に有効なものである。

Z－80のブロック・サーチ命令群も上記のような処理に有効である。１命令で必要な範囲だけ外部メモリのブロックをサーチし、特定の８ビット・キャラクタを探し出せる。目的のキャラクタが見つかれば命令は自動的に終了する。

これらの命令の実行中に割り込みをかけることは可能であり、長時間CPUが独占されてしまうという弊害はない。

算術および論理演算命令は、アキュムレータ内のデータとCPU汎用レジスタあるいは外部メモリ内のデータの操作を行うものである。

この操作の結果はアキュムレータに入り、その結果に応じて関係するフラグがセットされる。

算術演算命令の例として、アキュムレータと外部メモリの内容の加算命令がある。加算の結果はアキュムレータに入る。

これらの命令群のうちには、16ビットCPUレジスタ間の加減算命令などがある。

ローテイト、シフトの命令群によりどのようなレジスタまたはメモリであれ、キャリまたはキャリなしに、算術的または論理的に左右にローテイトできる。またアキュムレータ内の1ディジットとメモリ内の2ディジットで左右にローテイトすることもできる。

ビット操作命令は1命令でアキュムレータ、各汎用レジスタ、あるいは外部メモリ内のデータのどのビットでも、セット、リセットあるいはテストができるものである。たとえば、レジスタH内のデータの最上位ビットをリセットする、といったことが可能である。

この命令群は、制御への応用や一般のプログラミングにおいてフラグを多用する場合にとくに有効なものである。

ジャンプ、コール、リターン命令はユーザ・プログラムの流れを変えるためのものである。

この命令群にはいくつかの異なった形式があり、これらの形式により外部メモリの特定アドレスからプログラム・カウンタに更新データ（新しいアドレス値）を取ってくる形式も異なる。

ジャンプのうち、RST命令はユニークなもので、8ビットOPコードの一部に新アドレスを含んでいる。この命令では外部メモリのページ・ゼロ内の8箇所のアドレスを指定できる。

プログラム・ジャンプには、レジスタHL、IX、IYの内容を直接PCに移して行う方法もあり、ジャンプ先のアドレスをその時点で実行中のルーチンに関連させて決めることもできる。

Z－80の入出力命令群は、外部メモリあるいはCPU汎用レジスタ群と多数の外部入出力デバイス群とのデータ転送に適している。

入出力処理を行う際のポート番号はアドレス・バスの下位8ビットにより256ポートを指定できる。ある種の命令では、このポート番号を命令の2バイト目に持ち、また他の命令では、Cレジスタの内容で指定できるようになっている。

Cレジスタを入出力デバイスのポインタとして用いる方式の利点は、異なった入出力ポートでも共通の駆動

ルーチンが使用できる点にある。これは、ルーチンがROMに入っていて、アドレスがOPコードの一部になっている場合には不可能である。

他の特長として、入力命令でフラグ・レジスタが自動的にセットされるので、入力データの状態（たとえば、入力のパリティ）を決めるのに余分の命令を使用する必要がない点があげられる。

Z－80 CPUには、自動的に256バイト以上のブロック・データをメモリのある領域から入出力ポートへ、またその逆に転送する命令もある。

対になっている汎用レジスタ群を両方使用して、これらの命令により高速の入出力ブロック転送ができる。

この入出力命令群の真価は次の例でよくわかるだろう。割り込み駆動形式を用いて倍密度フロッピー・ディスクを動作させる場合、Z－80 CPUは必要なフロッピー・ディスク・フォーマット（すなわち、プレ・アンブル、アドレス、データ、CRCコード使用・未使用など）をプログラムによってたやすく用意できる。

最後のCPU制御命令は種々の選択、モード設定のためのものである。この命令群には割り込みイネーブル・フリップ・フロップをセット、リセットしたり、割り込み応答のモードを設定したりする命令などがある。

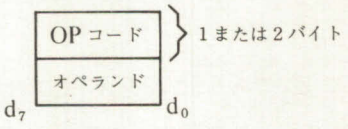
5．2　アドレシング・モード

Z－80命令群には、CPU内部レジスタ、外部メモリあるいは入出力ポート内にストアされたデータの操作命令が多く設けられている。

これらのデータのある番地の指定を行うことをアドレシングという。

この節では、Z－80で使用されるアドレシングのタイプについて簡単な説明をし、あとで各命令群の説明を行う際にさらに詳しく述べる。

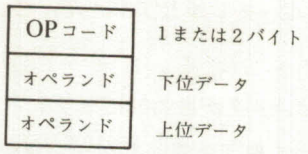
イミディエット・アドレシング　このアドレシング・モードでは、OPコードに続く1バイトを実効オペランドとする。



例として、アキュムレータに定数をロードするような命令がある。

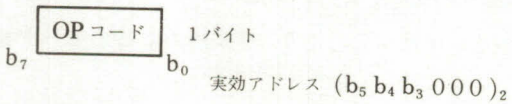
OPコードに続く1バイトが直接その定数値となる。

拡張イミディエット・アドレシング　このモードは上述のモードを拡張したものであり、オペランドが2バイトになっているものである。

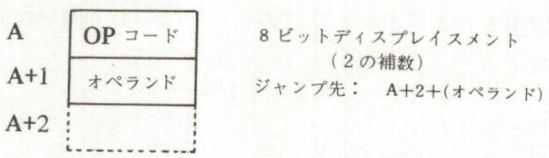


たとえば、16ビット（2バイト）データをHLレジスタ対にロードする命令がある。

ゼロ・ページ修飾アドレシング　Z－80には、メモリのゼロ・ページの8箇所を1バイトで指定する特別なコール命令がある。これはRST命令と呼ばれ、PCをゼロ・ページの実効アドレスにセットするものである。この命令の値は、共通に使用するサブルーチンのアドレス（16ビット）を1バイトで指定できるようになっていて、ステップ数を節約できる。



相対アドレシング　このモードでは、OPコードに続く1バイトにより、ジャンプ命令のある場所からのディスプレイスメントを指定する。このディスプレイスメントは符号付きの2の補数であり、次の命令の先頭番地の値にこのディスプレイスメントを加えた値がジャンプ先の番地となる。

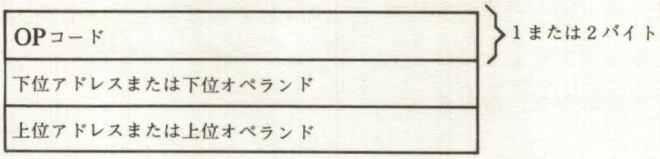


このモードは近隣のアドレスへ2バイトでジャンプでき、3バイトのジャンプ命令を使用する必要がないときに役に立つ。多くの場合、関係のあるプログラムのセグメントは近接して置かれるのが普通であり、このモードを利用してメモリの節約が図れる。

符号付き2の補数の範囲は－128から＋127であり、これをA＋2に対して加えるのでOPコードのアドレスから計算すると、－126から＋129の範囲へのジャンプ指定となる。

このモードの最大の特長はリロケータブル・コードとなる点である。

拡張アドレシング　このアドレシング・モードは命令に2バイトのアドレス・データを含む。64Kバイト内のどのアドレスに対しても、プログラムをジャンプさせられる。

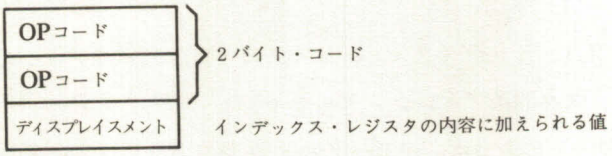


拡張アドレ シングはメモリのある位置から他の位置へジャンプしたり、ある位置でのデータの出し入れの命令に必要なものである。

このモードにおいて、ソースあるいはデスティネーション・アドレスを指定するオペランドとしてこれを(nn)と書いたとき、nn番地のメモリの内容を意味する。ここで、nnは16ビットのアドレス指定値であり、括孤の中のnnは2バイトのメモリ番地ポインタである。

たとえば、(1200)は1200番地のメモリの内容である。

インデックス・アドレシング　このアドレシングでは、OPコードのあとにはメモリ・アドレスのポインタである2つのインデックス・レジスタのうち、いずれか一方（OPコードで指定する）の内容に加えるディスプレイスメントを指定するデータ・バイトが置かれる。この操作ではインデックス・レジスタの内容は変らない。



インデックス・アドレシング命令の例として、アキュムレータにメモリ（インデックス＋ディスプレイスメント）の内容をロードする命令がある。ディスプレイスメントは符号付きの2の補数である。

インデックス・モードの特長として、データ・テーブルを使用する場合、インデックス・レジスタにはどのテーブルのスタート・アドレスでも入れておけるので、非常に簡単に使用できるという点がある。2つのインデックス・レジスタがあるので、2つ以上のテーブルをよく使用する必要があるときには都合がよい。インデックス・モードも相対アドレシングと同様にリロケータブル・コードである。

インデックス・レジスタとしてIX、IYがある。インデックス・アドレシングでは次のような書き方が使用される。

(IX + d) または (IY + d)

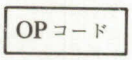
ここで、dはOPコードのあとに置かれるディスプレイスメントである。括孤はこの中の値が外部メモリのポインタとして使用されていることを意味している。

レジスタ・アドレシング　Z－80の多くのOPコードにはレジスタ指定ビットがあり、どのCPUレジスタを操作用として使用するかを指定するようになっている。レジスタ・アドレシングの例として、Bレジスタの内容をCレジスタにロードする命令などがある。

インプライド・アドレシング　このモードでは、OPコード自体がレジスタ指定を含んでいて、1つまたは2つのレジスタが自動的に指定される。

たとえば、算術演算命令では、つねにアキュムレータが結果をセットするレジスタとなる。

レジスタ間接アドレシング　このタイプでは、メモリ内の位置指定のポインタとして16ビットのレジスタ・ペア（例：HL）が使用される。これは大変強力で役に立つ命令である。



1または2バイト

例として、HLレジスタの内容をメモリの位置ポインタとし、その位置の内容をアキュムレータにロードするといった命令がある。

インデックス・アドレシングは、インデックス・レジスタの内容にディスプレイスメントを加えるといった操作を含んでいる以外は、レジスタ間接アドレシング・モードの一形式である。

ブロック転送、ブロック・サーチの命令はこのモードの拡張形であり、レジスタの内容の自動インクリメント、デクリメントおよび比較機能が付加されている。間接モードでの記法はレジスタ名を括弧で囲む。そしてそのレジスタをポインタとする。メモリ位置ポインタがHLレジスタの内容である場合、

(HL)

と書く。間接モードでは16ビット・オペランドをよく使用するが、この場合レジスタの内容はまず、オペランドの下位バイトを指し、次にレジスタの内容が自動的にインクリメントされ、上位バイトを指す。

ビット・アドレシング　Z－80には、多くのビット・セット、リセットおよびテスト命令がある。これらの命令は、どのメモリ位置、CPUレジスタに対しても有効で、前述した3モード（レジスタ、レジスタ間接、およびインデックス）のいずれかのアドレシング・モードでビット操作ができる。

8ビットのうちのどのビットを操作するかはOPコード内の3ビットにより指定する。

アドレシング・モードの組み合わせ

これらの命令には、算術演算やロード命令のように、1つ以上のオペランドが含まれている命令が多い。これらの命令では、2つのアドレシング・モードを組み合わせて用いる。たとえば、ロード命令で、ソースの指定はイミディエット、デスティネーションの指定はレジスタ間接あるいはインデックス・アドレシングで行う場合などである。

5．3　命令OPコード

この節では、Z－80の各命令について説明し、それらのOPコード表を示す。これら各表の陰影部で示したOPコードは8080Aのものと同一である。

各命令に対応するアセンブリ語ニーモニックを同時に示す。OPコードは16進数表現にしてあり、1バイトのコードは2桁の16進数、2バイトでは4桁の16進数で書く。16進から2進への変換は次の表を利用するとよい。

16進数	2進数	10進数	16進数	2進数	10進数
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	B	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

Z－80の命令ニーモニックとしてOPコード単独のものと、OPコードと1つまたは2つのオペランドより構成されているものがある。オペランドのない命令はオペランドがOPコードに内蔵されていると考える。

1つの論理オペランドのみの場合、あるいはロジカルORのような片方のオペランドが不定の場合には、1オペランド・ニーモニックになっている。

2変数を扱う命令は2オペランド・ニーモニックである。

ロードと交換命令（LD, PUSH, POP）

表5－1にCPUの8ビット・ロード命令群を示す。各命令で使用されるアドレシングの型も併記しておく。データのソースは上欄に、デスティネーションは左側端に示してある。例として、BレジスタからCレジスタへのロードはOPコード48Hとなる。

表中のコードはすべて16進数表現であるが、この48H（2進数0100 1000）コードはM1サイクル時に外部メモリからCPUにフェッチされ、CPUはこれをデコードしてレジスタ間のデータの転送を実行する。

この命令群のアセンブリ語ニーモニックはLDであり、これのあとにデスティネーション、ソースの順で書く。(LD dst , src)

これらの命令ではいくつかのアドレシング・モードを組み合わせて使用できる。たとえば、ソースをレジスタ・モード、デスティネーションをレジスタ間接モードとした命令では次のように記述できる。

LD (HL), D

これはHLレジスタで指定されたメモリへDレジスタの内容をロード（移す）するニーモニック命令で、OPコードは72Hである。

ここでHLを囲んだ括孤は、HLの内容をメモリの位置ポインタとして使う、ということを意味している。

Z－80のアセンブリ言語はプログラミングが容易なようにできている。

各命令は自己説明的（セルフ・ドキュメンティング）で意味がわかりやすく、プログラムの保守も楽である。

Z－80で使用されるOPコードには、2バイトのものがいくつかあるので注意されたい。

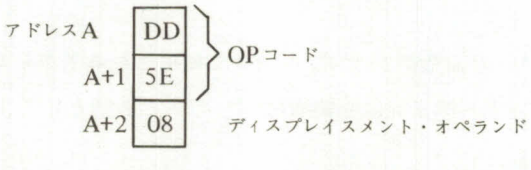
命令には8ビット、16ビット、24ビット、32ビットの命令があるが、これはメモリの有効利用に役立つ。つまり、算術、論理演算のような命令で、8ビットで済む場合にでも、16ビット固定の命令を使用しなければならないとしたら、メモリのむだになるからである。

LD命令で、ソースあるいはデスティネーションの位置のアドレシングにインデックス・モードを使用した場合、メモリは3バイト必要で、3バイト目はディスプレイスメントdとなる。

たとえば、次のような命令ニーモニックは、IXの内容から8バイト目の場所の内容をレジスタEにロードするもので、数字8がオフセットでありディスプレイスメントとなる。

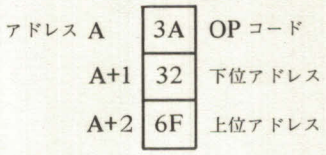
LD E, (IX + 8)

命令のメモリ上での順序は次のようになる。



拡張アドレシング命令も3バイトである。メモリ6F32H番地の内容をアキュムレータにロードする次のような命令では、メモリ上の順序は次のようになる。

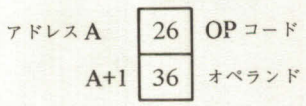
LD　A, (6F32H)



アドレスの下位バイトはつねに第1オペランドである点に注意しなければならない。

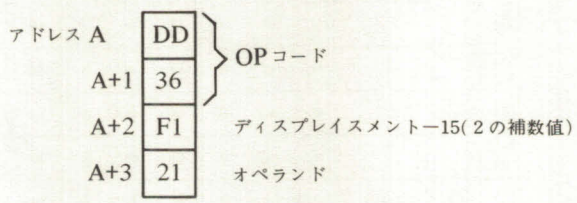
汎用8ビット・レジスタへ直接ロードする命令は2バイト命令である。値36Hをレジスタ Hへロードする命令は次のように書ける。

LD　H, 36H



デスティネーション側のメモリをインデックス・アドレシングで指定し、ソース側をイミディエットで指定してロードする場合、4バイトを必要とする。

LD　(IX − 15), 21H



インデックス・アドレシングでは、いつもディスプレイスメントはOPコードの直後にあるので注意を要する。

表5－2に16ビット・ロード命令群のOPコードを示す。これは8ビットのものとよく似ている。注意すべき点は、拡張アドレシングは全部のレジスタ・ペアに対して使用できることである。

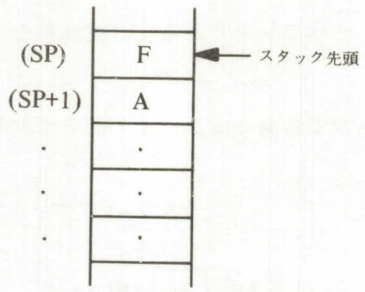
さらに、スタック・ポインタを用いたレジスタ間接アドレシング命令として、PUSH と POP 命令がある。これらのニーモニックはそれぞれ〝PUSH〟、〝POP〟である。この命令ではスタックへあるいはスタックからプッシュやポップする際、スタック・ポインタが自動的にインクリメントあるいはデクリメントされる点が、他の16ビット命令と異なっている。

PUSH　AF

これはOPコードF5Hの1バイト命令であるが、次のような操作が自動的に行われる。

- ○ (SP) を－1する。
- ○ LD (SP), A
- ○ (SP) を－1する。
- ○ LD (SP), F

この結果、外部スタックは次のようになる。



デスティネーション			ソース: インプライド I	インプライド R	レジスタ A	レジスタ B	レジスタ C	レジスタ D	レジスタ E	レジスタ H	レジスタ L	(HL)	(BC)	(DE)	インデックス (IX + d)	インデックス (IY + d)	(nn)	イミディエット n
レジスタ	A		ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B				47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C				4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D				57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E				5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H				67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L				6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
レジスタ間接	(HL)				77	70	71	72	73	74	75							36 n
	(BC)				02													
	(DE)				12													
インデックス	(IX+d)				DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)				FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
拡張アドレシング	(nn)				32 n n													
インプライド	I				ED 47													
	R				ED 4F													

表5－1　8ビットロード　"LD"

ポップ命令はプッシュ命令とちょうど逆の操作になる。

これらの命令では、オペランドは16ビットであり、上位バイトが最初にプッシュされ、また最後にポップされる。

PUSH BC は PUSH B そして C
PUSH DE は PUSH D そして E
PUSH HL は PUSH H そして L
POP HL は POP L そして H

となる。

ソースに対する拡張イミディエット・アドレシングでは、OP コードのあとに2バイトのデータが付く。

LD DE, 0659H

アドレス A	11	OP コード
A+1	59	下位バイト(レジスタE へ入る)
A+2	06	上位バイト(レジスタD へ入る)

すべての拡張イミディエットあるいは拡張モードでは、下位バイトがOP コードのすぐあとに続く。

表5－3にZ－80に設けられている16ビット・データの交換命令を示す。

OP コード 08_H は、2組のアキュムレータとフラグ類の切り替えをプログラムで実行させるためのもので、また $D9_H$ で、6個の汎用レジスタ群をもう一方のレジスタ群に切り替えることができる。

これらのOP コードは、これ以下に短くできないという最短の1バイト命令であり、2セットのレジスタ群を即時切り替えて使用できるので、高速の割り込み応答が必要な場合に利用できる。

ブロック転送およびサーチ

表5－4に、非常に特長があり、しかも強力なブロック転送命令を示す。これらの操作は次の3つのレジスタを使用して行う。

HL ソースの位置のポインタ
DE デスティネーションのポインタ
BC バイト・カウンタ

プログラムでは、これらのいずれかのブロック転送命令を使用するまえに、上記の3つのレジスタに値をセットしておく必要がある。

LDI（ロード&インクリメント）命令は、(HL) で指定される位置から (DE) で指定される位置へ、1バイトだけ転送する。その後、2つのレジスタの内容はインクリメントされて次の位置を指している。

バイト・カウンタ（BC）はデクリメントされる。この命令は、データ転送の途中で他の処理を行いたい場合に便利である。

LDIR（ロード&インクリメント&リピート）命令はLDIの拡張形で、自動的に全部（バイト・カウンタが零になるまで）データを転送してしまいたい場合に便利である。

16ビット・レジスタを使用するので、ブロックの大きさは最大64Kバイト（1Kバイト＝1024バイト）まで指定でき、ソース、デスティネーションのアドレスはメモリ空間のどこにでも配置できる。

さらに、ブロック相互のオーバラップも可能で、3つのレジスタ・ペアを使用して扱えるデータに何ら制限のないことに注目されたい。

LDDとLDDRはさきの2つとよく似ている。ただ、データ転送後、レジスタ・ペアHL、DEがデクリメントされる点が異なっていて、位置の高い方から低い方へ転送する場合に使用するのに適している。

ソース

デスティネーション			レジスタ							拡張イミディエット	拡張アドレシング	レジスタ間接
			AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
	レジスタ	AF										F1
		BC								01 n n	ED 4B n n	C1
		DE								11 n n	ED 5B n n	D1
		HL								21 n n	2A n n	E1
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	拡張アドレシング	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH命令 →	レジスタ間接	(SP)	F5	C5	D5	E5		DD E5	FD E5			

POP命令 ↑

（注）PUSHおよびPOP命令の全実行が終わったあとにSPが修正される。

表5－2　16ビットロード〝LD〟

		インプライド・アドレシング				
		AF′	BC′, DE′ & HL′	HL	IX	IY
インプライド	AF	08				
	BC, DE & HL		D9			
	DE			EB		
レジスタ間接	(SP)			E3	DD E3	FD E3

表5－3　交換

ソース

		レジスタ間接 (HL)	HLはソースのポインタ。 DEはデスティネーションのポインタ。 BCはバイト・カウンタ。
デスティネーション	レジスタ間接 (DE)	ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC
		ED B0	'LDIR' – Load (DE)←(HL) Inc HL & DE, Dec BC, Repeat until BC = 0
		ED A8	'LDD' – Load (DE)←(HL) Dec HL & DE, Dec BC
		ED B8	'LDDR' – Load (DE)←(HL) Dec HL & DE, Dec BC, Repeat until BC = 0

表5－4　ブロック転送グループ

サーチ
ロケーション

レジスタ間接 (HL)	HLはアキュムレータの内容と照合する メモリの位置のポインタ。 BCはバイト・カウンタ。
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR' Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

表5－5　ブロック・サーチ

表５－５に４種のブロック・サーチ命令のOPコードを示す。

CPI（コンペア＆インクリメント）は、HLレジスタで指定されるメモリの内容とアキュムレータ内のデータを比較し、その結果をフラグ・ビット（第６章を参照）の１つにセットする。その後、HLレジスタ・ペアをインクリメントし、バイト・カウンタ（BCレジスタ・ペア）をデクリメントする。

CPIRはCPI命令の拡張形であり、バイト・カウンタが零になるまで比較を繰り返す。この１命令で、ある８ビット・キャラクタを全メモリ内で探すこともできる。

CPD（コンペア＆デクリメント）とCPDR（コンペア＆デクリメント＆リピート）は同様な命令であるが、各比較ののちHLレジスタをデクリメントする点のみが異なっている。したがって、逆方向へのメモリのサーチが行える。（このサーチはメモリ・ブロックの最高アドレス近くからスタートし、下位アドレスへすすむ。）

このブロック転送、サーチ命令は、文字列の操作などの応用に非常に強力な道具となる。

算術と論理演算

表５－６にアキュムレータを中心にした８ビット算術演算命令群を示す。インクリメント（INC）、デクリメント（DEC）命令もともに示す。

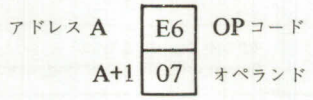
これらのうち、INC、DECを除いた８ビット操作命令で、アキュムレータと表に示したソース・データとの間の操作が行われ、その結果は、コンペア（CP）命令を除き、アキュムレータに置かれる。CPでは命令実行の前後でアキュムレータの内容は変わらない。これらの操作の結果で、フラグ・レジスタがセットされる。（各命令でセットされるフラグの詳細については、第６章に述べる。）

INC、DEC命令は、レジスタはどのメモリの位置に対しても実行できるが、ソースとデスティネーションを別々にとることはできない。

ソース・オペランド（ソースの指定）にインデックス・レジスタを使用した場合、ディスプレイスメントをすぐあとに続けて指定しておく。

イミディエット・アドレシングを使用した場合には、実際のオペランドで直接指定する。例として、次のような命令では、あとのようなメモリ配置となる。

AND　07H



アキュムレータの内容がF3Hであれば、結果として03Hが得られ、最終的にアキュムレータの内容は03Hとなる。

ソース

	レジスタ・アドレシング							レジスタ間接	インデックス		イミディエット
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

表５－６　８ビット算術、論理演算

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

表５－７　その他の操作

AND　07_H　の操作と結果

操作前の Acc の内容	1111	0011	＝	$F3_H$
オペランド	0000	0111	＝	07_H
操作後の Acc の内容	0000	0011	＝	03_H

ADD 命令（ADD）は、ソースの位置内のデータとアキュムレータ（Acc）のデータ間で2進加算を行い、SUB 命令は2進減算を行う命令である。

キャリを含めた操作を行う場合、ADC（加算）、SBC（減算）を使用すればよい。それぞれキャリ・フラグの内容を演算に関与させられる。

Z－80では、フラグとデシマル・アジャスト（10進補正）の命令を次のような算術演算操作に対して使用できる。

パックされた BCD 数の高精度演算
符号付き2進数の高精度演算
符号付き2の補数値の高精度演算

この命令群には他に論理積 AND、論理和 OR、排他的論理和 XOR、比較 CP がある。

アキュムレータあるいは、キャリ・フラグを操作する5つの汎用算術演算命令があるが、これを表5－7に示す。

10進補正は加算と同様減算に対しても行うことができ、BCD 算術演算が簡単に行える。この減算補正にはNフラグを利用するとよい。このフラグは前の算術演算命令が減算であればセットされている。

NEG 命令はアキュムレータ内の値の2の補数値をアキュムレータに置く（セットする）。

ここで、Z－80にはキャリをリセットする命令がないことに注意されたい。この操作はアキュムレータ自身で AND をとるなどの操作をすれば、キャリ・フラグがリセットされるので、その命令をとくに設けていない。

表5－8に16ビット・レジスタ間の16ビット演算命令群を示す。これらにはキャリを含めた加、減算を入れて、5グループある。

ADC、SBC 命令は全フラグに影響を与える。これらの命令を使用すればアドレス計算あるいは16ビット算術演算操作の簡単化が図れる。

デスティネーション		ソース BC	DE	HL	SP	IX	IY
'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

表５－８　16ビット算術演算

ローテイトとシフト

Ｚ－80の主な特長として、アキュムレータ、汎用レジスタあるいはメモリ内のデータのローテイトあるいはシフト命令が充実している点があげられる。

ローテイトとシフト命令のOPコードを表５－９に示す。Ｚ－80には、算術および論理シフト操作も設けられている。これらの操作は整数の乗算および除算を含む広範囲の用途に使用でき、大層便利なものである。

２つのBCDディジット・ローテイト命令(RRDとRLD)は、HLレジスタ・ペアで指定されるメモリ内の２桁(ディジット；４ビットで１桁を表わす)とアキュムレータ内の１桁とのローテイト(入れ替え)が行える。(表５－９を参照)　　これらの命令はBCDデータの演算に利用できる。

ソースおよびデスティネーション

ローテイトあるいはシフトの型	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

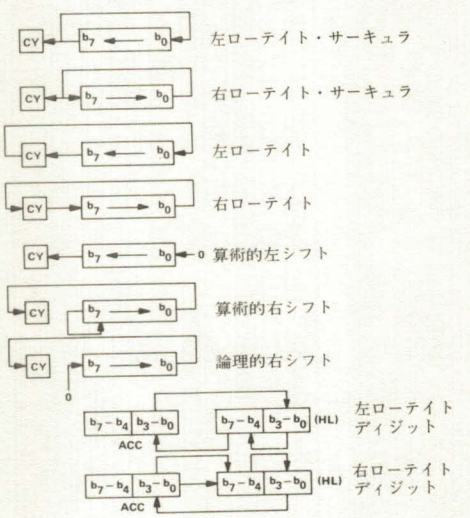


表５－９　ローテイトとシフト

ビット操作

ビットごとのセット、リセットあるいはテストといった操作はどのようなプログラムでも必要なものである。とくにその操作がレジスタに限らずメモリ内でできれば、大変便利である。このビットは普通汎用ソフトウエア・ルーチン、外部の制御状態の指示あるいはメモリの有効利用のためにパックしておくデータ群の指標などのためのフラグとしてよく使用される。

Z－80には、1命令によりアキュムレータ、各汎用レジスタ、あるいはどのメモリ位置に対してもその中の各ビットのセット、リセット、あるいはテストができる能力がある。

表5－10にこの目的のための240の命令群を示す。

レジスタ・アドレシングでは、アキュムレータあるいはいずれかの汎用レジスタを指定でき、そこでのビット操作ができる。

レジスタ間接およびインデックス・アドレシングでは、その指定によって外部メモリ上でビット操作ができる。

ビット・テスト操作では、ビットの状態をゼロ・フラグ（Z）で判定する。たとえば、ゼロ・フラグがセットされているならば、テストされるビットは零である。（この詳細については、第6章を参照のこと）

		レジスタ・アドレシング							レジスタ間接	インデックス	
	ビット	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	GB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

表5－10　ビット操作グループ

ジャンプ、コール、リターン

表５－11にジャンプ、コール、およびリターン命令群を示す。ジャンプとはプログラム中でのステップの変更であり、プログラム・カウンタには、３つの適当なアドレシング・モード（拡張イミディエット、相対、またはレジスタ間接）のいずれかで指定される16ビットの値がロードされる。

ジャンプ命令群には条件ジャンプがある。これらはジャンプ命令直前の条件によってジャンプしたり、そのまま次の命令ステップへ続けられるものである。この条件はフラグ・レジスタ内のデータにセットされている。（フラグ・レジスタに関しては、第６章を参照）

拡張イミディエット・アドレシングを使用すれば、メモリのどの位置へでもジャンプできる。この命令は３バイト必要で、OP コードの次のバイトが下位アドレス・バイトであり、そのあとに上位アドレス・バイトが続く。（２バイトで64K バイト全メモリを指定する。）

たとえば、メモリの位置 3E32H への無条件ジャンプは次のようになる。

アドレス	A	C3	OP コード
	A+1	32	下位アドレス
	A+2	3E	上位アドレス

相対ジャンプ命令は２バイト命令であり、２バイト目はその時点での PC の内容に加えるためのディスプレイスメント（２の補数値）である。

このディスプレイスメントにより OP コードのあるアドレスから－126～＋129の範囲を指定できる。

レジスタ間接ジャンプには３種あって、レジスタ・ペア HL、インデックス・レジスタ IX、IY の内容を直接 PC にロードして行う。このモードでは、まえで計算した値をもとにジャンプ先を決めてジャンプすることができる。

コール命令はジャンプ命令の特殊な形であり、戻り番地（コール命令のあとのバイトのアドレス）をスタックにプッシュしておいてジャンプを実行するものである。リターン命令はコールの逆で、スタックからデータをポップしてきて、それを直接 PC にロードしてジャンプする命令である。

コールおよびリターン命令で、サブルーチンあるいは割り込みの処理が、簡単に行える。

Z－80ファミリ用に、２つの特殊リターン命令が設けられている。RETI（マスク可能な割り込みからのリターン）と RETN（マスク不可能な割り込みからのリターン）の２種で、CPU 内では OP コード C9H の無条件リターンと同様に扱われるが、次の点で異なっている。

RETI は割り込み処理ルーチンの終わりで使用するが、Z－80の周辺チップはこの命令の実行を検知して、ネスティングしていた割り込みに対して適切な処置が行われたことを認識できる。

Z－80の周辺デバイスの構成を考慮したこの命令を使用すれば、ネスティングしていた割り込みからの正常リターンの手続きが簡単になる。もし、このような命令がなければ、割り込み処理ルーチンが完了し、割り込みをかけてきたデバイスにそれを伝えるためには、次のようなプログラム・ステップが必要であろう。

```
DI          ;このルーチンが始まるまえにまず割り込みを禁止する（ディセーブル割り込み）。
LD  A, n    ;
OUT  n, A   ;  } サービス・ルーチン完了をデバイスへ通知する。
RET         ;リターン。
```

Z－80では、2バイトのRETI命令で、さきのような6バイトのステップを置き替えることができるのである。これは割り込みサービスの時間をできる限り短縮しなければならない場合には重要となる。

プログラムでループ制御を簡単にするためにDJNZ命令があり、便利なものである。これは2バイトの条件付き相対ジャンプ命令で、しかも命令実行ごとにBレジスタがデクリメントされる。Bレジスタが零になるまで繰り返しジャンプを行い、零になれば次の命令ステップに移る。オペランド（ディスプレイスメント）は2の補数値で与える。次に簡単な例を示しておく。

アドレス	命　令	コメント
N、N＋1	LD　B，7	;Bレジスタをカウント7にセット。
N＋2	（プログラム・ステップ）	;7回繰り返す内容。 ←┐
＼		
N＋9		
N＋10、N＋11	DJNZ　－10	;もし、Bレジスタが0でなければ
		N＋12からN＋2へ戻る。 ┘
N＋12	（次のステップ）	;（Bレジスタ）＝0のとき、ここへ。

			条件									
			無条件	キャリ	ノン・キャリ	ゼ ロ	ノン・ゼロ	パリティ偶 数	パリティ奇 数	負	正	カウント
JUMP 'JP'	拡 張 イミディエット	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	相 対	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	レジスタ間 接	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	拡 張 イミディエット	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	相 対	PC+e										10 e-2
RETURN 'RET'	レジスタ間 接	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	レジスタ間 接	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	レジスタ間 接	(SP) (SP+1)	ED 45									

あるフラグは1つ以上の
目的で用いられる。
詳細は第6章参照。

表5－11　ジャンプ、コール、リターン グループ

表5－12に8種のリスタート（RST）命令のOPコードおよびニーモニックを示す。この命令は1バイトで8箇所のいずれかをコールできる。

この命令の特長は、よく使用するルーチンをこれで呼べば、プログラム・ステップ数、すなわちメモリ使用量を節減できる点にある。

		OPコード	
コール・アドレス	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

表５－12　リスタート

入力/出力

Z－80 CPU には表５－13および表５－14に示すように、多種の入出力命令群がある。

入力/出力デバイスのアドレシングには、絶対モードあるいは、Cレジスタを使用したレジスタ間接モードがある。

レジスタ間接モードでは、入出力デバイス群と内部レジスタ間のデータの直接転送が可能である点に注意されたい。さらに、８つのブロック転送命令が設けられている。

これらの命令群はメモリ・ブロック転送と同種のものであるが、HL レジスタ・ペアをメモリ・ソース（出力要求の場合）またはメモリ・デスティネーション（入力要求の場合）のポインタとして用いる点が異なる。一方、Bレジスタは同様にバイト・カウンタとして使用する。

Cレジスタは入力/出力の必要なデバイスのポートのアドレスを保持する。

Bレジスタは８ビット長なので、入出力ブロック転送は256バイトまでである。

次のような入出力命令ではｎはアドレス・バスの下位８ビットに出力される。

IN　A, n

OUT　n, A

同時にアキュムレータの内容がアドレス・バスの上位８ビットへ出力される。

入出力ブロック転送も含めたレジスタ間接入出力命令のいずれにおいても、レジスタCの内容は、アドレス・バスの下位８ビットで転送され同時にレジスタBの内容がアドレス・バスの上位８ビットに出力される。

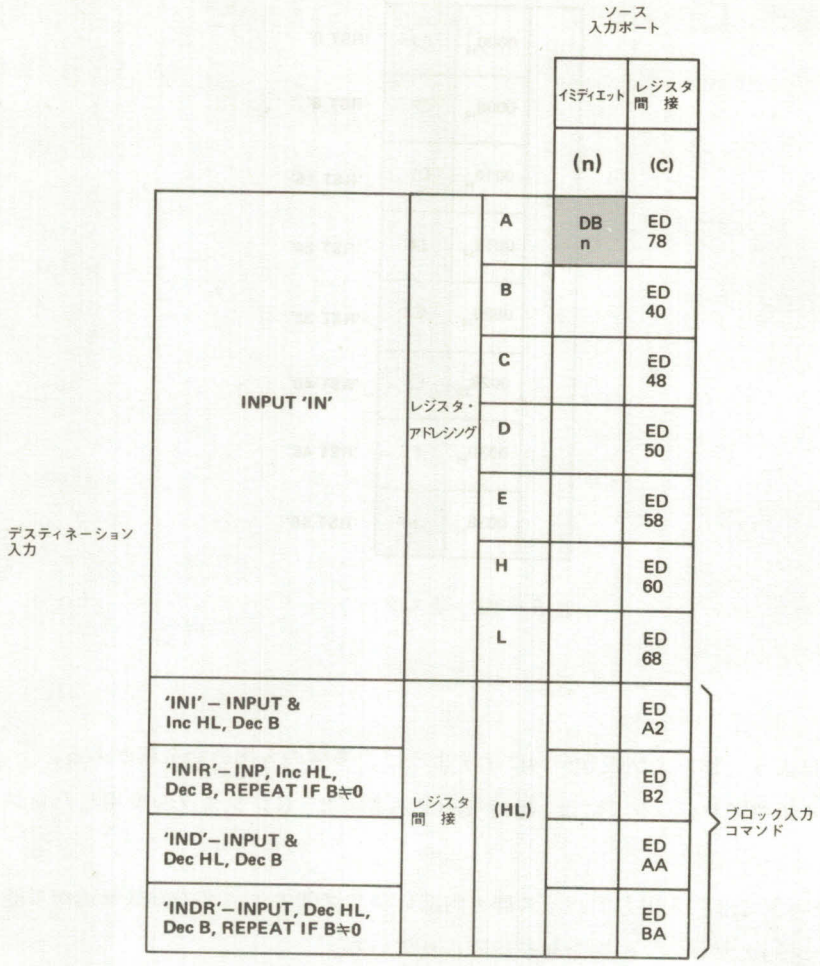

ソース
入力ポート
イミディエット
レジスタ
間 接
(n)
(C)
デスティネーション
入力
INPUT 'IN'
レジスタ・
アドレシング
A
DB
n
ED
78
B
ED
40
C
ED
48
D
ED
50
E
ED
58
H
ED
60
L
ED
68
'INI' – INPUT &
Inc HL, Dec B
'INIR'– INP, Inc HL,
Dec B, REPEAT IF B≒0
'IND'– INPUT &
Dec HL, Dec B
'INDR'– INPUT, Dec HL,
Dec B, REPEAT IF B≒0
レジスタ
間 接
(HL)
ED
A2
ED
B2
ED
AA
ED
BA
ブロック入力
コマンド

表５－13　入力命令群

			ソース							
			レジスタ							レジスタ間接
			A	B	C	D	E	H	L	(HL)
'OUT'	イミディエット	(n)	D3 n							
	レジスタ間接	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	レジスタ間接	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	レジスタ間接	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	レジスタ間接	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	レジスタ間接	(C)								ED BB

ブロック出力コマンド

デスティネーション出力ポート

表５－14　出力命令群

CPU 制御命令

表５－15に６つの汎用 CPU 制御命令を示す。NOP 命令は何もしない命令である。HALT 命令は CPU の動作を次に割り込みが入るまで停止させる命令であるが、この割り込みは DI、EI 命令によってソフト的に受け付け可否の制御ができる。

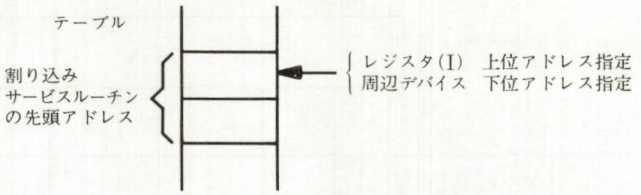
３種類の割り込み動作モードの指定命令によって、CPU はそれぞれ次のような対応する割り込み応答モードになる。

モード０にセットされた場合、割り込みを発しているデバイスがデータ・バス上に何らかの命令を乗せて、CPU がそれを実行する形式となる。

モード１は簡単なもので、CPU は自動的に位置 0038H からリスタート（RST）するので、外部回路からの指令は何ら必要でない。（前の PC の内容はスタックにプッシュされる。）

モード２は最も強力なもので、全メモリの位置への間接コールができる。このモードでは、CPU はレジスタ I の内容により上位８ビットを指定し、割り込みを発生しているデバイスからのデータで下位８ビットを指定し、計16ビット・メモリ・アドレスを形成する。

このアドレスとそれに続くアドレスの内容２バイトにより、サービス・ルーチンの先頭アドレスを指定する。CPU は自動的にこのデータ２バイトで指定されるアドレスへの CALL 命令を実行する。



'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A モード
SET INT MODE 1 'IM1'	ED 56	38_H番地へのコール命令
SET INT MODE 2 'IM2'	ED 5E	レジスタIと割り込みデバイスからの 8ビットデータを用いた間接コール命令

表5－15　CPU制御命令

第6章　フラグ類

Z－80 CPUの2組のフラグ・レジスタは種々のCPU操作による結果を6ビットで保持する。

このうち、4つのビットはプログラムによりテストでき、ジャンプの条件として使用できる。これらについて説明する。

1）**キャリ・フラグ（C）**―アキュムレータの最上位ビットからの桁上がりでセットされる。加算命令でのキャリ、または減算命令でボロー発生の場合などでセットされる。シフト、ローテイト命令ではこのCビットが関係する。

2）**ゼロ・フラグ（Z）**―演算の結果、アキュムレータに零がロードされたときにセットされる。他の場合リセットされる。

3）**サイン・フラグ（S）**―このフラグは符号付き数値を扱う場合に使用し、演算の結果が負ならばセットされる。符号としてビット7(MSB)を使用するので、アキュムレータのビット7の内容がこのフラグに置かれる。（負の数ならば1が立つ。）

4）**パリティ/オーバ・フロー・フラグ（P/V）**―この2つの目的を持ったフラグは論理演算（AND A, Bなど)が行われた場合にはパリティを示し、また符号付き2の補数値の算術演算が行われた場合にはオーバ・フローの有無を示すために用いられる。Z－80のオーバ・フローはアキュムレータ内の2の補数値が＋127を超えるか、－128より小さくなった場合に出される。次のような加算ではオーバ・フローとなる。

```
＋120＝     0111  1000
＋105＝     0110  1001
―――――――――――――――――――――――――――――――
  C＝0  1110  0001  ＝（－31）オーバ・フロー発生（エラー）
```

これはエラーであり、オーバ・フローがあるがキャリは出ないので、エラー表示のためにオーバ・フロー・フラグがセットされる。

次のような2の補数値の加算では正しい結果が出て、キャリ・フラグがセットされる。この場合、オーバ・フローではないので、オーバ・フロー・フラグはセットされない。

−5＝	1111	1011
−16＝	1111	0000
C＝1	1110	1011

＝（−21）キャリ発生（正答）

キャリは結果の正誤に直接関係しないので、オーバ・フロー・フラグが特別に必要である点に注意されたい。論理演算（AND、OR、XOR）では、結果のパリティが偶数ならばこのフラグはセットされ、奇数ならばリセットされる。

プログラマが直接に関与する必要のないフラグ・レジスタの残りのビットは、いずれもBCD算術演算用として使用される。

1）**ハーフ・キャリ・フラグ**（H）— BCD演算結果の下位4ビットからのキャリ、ボロー用である。DAA命令を用いて10進補正演算を行う場合、このフラグHを用いて、CPUは自動的にパックされた10進数の加算結果または減算結果に対して補正を行い、正しい結果を算出する。

2）**加/減算フラグ**（N）— BCD演算で、加算と減算とではアルゴリズムが異なっているので、このフラグを見て、先に実行された命令が加算か減算かを判定し、正しくDAA命令を実行する。

このフラグ・レジスタはプログラムによって調べたり操作したりすることができ、フォーマットは次のようになっている。

S	Z	X	H	X	P/V	N	C

Xは未定義

表6－1に各種のCPU命令でどのようにフラグ・ビットが変化するかを示す。各記号は次のような意味を持つ。

● — フラグ　変化しない。
X — フラグ　不定。
0 — リセットされる。
1 — セットされる。
↕ — 各条件（前述）にしたがって、セットあるいはリセットされる。
この表に書かれていないその他の命令はフラグに影響を与えない。

表６－１には、説明の必要があるいく種かの場合が含まれている。ゼロ・フラグは、ブロック・サーチにおいて、それらの結果、状態を示すのに用いられる。

ブロック・サーチで、ソースとアキュムレータのデータが一致すればゼロ・フラグはセットされる。またパリティ・フラグは、バイト・カウンタ（BC レジスタ・ペア）が零でなければセットされている。パリティ・フラグはブロック転送の際にも同様になっている。

ブロック入出力命令においては、ゼロ・フラグはバイト・カウンタとしてのＢレジスタの状態を示すために用いられる。入出力ブロック転送が終了したとき、ゼロ・フラグは零（すなわち、Ｂ＝０）にリセットされる。他方、メモリ間ブロック転送の終了時には、パリティ・フラグがリセットされる。

リフレッシュ・レジスタあるいはＩレジスタの内容がアキュムレータにロードされたとき、割り込みのイネーブル・フリップ・フロップ（IFF）の内容は、パリティ・フラグに退避されるので、CPU の状態はいつでも完全に保存されている。

命　令	C	Z	P/V	S	N	H	備　考
ADD A,s; ADC A,s	↕	↕	V	↕	0	↕	8ビット加算、キャリを含む加算
SUB s; SBC A,s, CP s	↕	↕	V	↕	1	↕	8ビット減算、キャリを含む加算、比較
NEG	↕	↕	V	↕	1	↕	符号反転（ニゲイト）
AND s	0	↕	P	↕	0	1	論理演算
OR s; XOR s	0	↕	P	↕	0	0	
INC	•	↕	V	↕	0	↕	8ビット・インクリメント
DEC s	•	↕	V	↕	1	↕	8ビット・デクリメント
ADD dd, ss	↕	•	•	•	0	X	16ビット加算
ADC HL, ss	↕	↕	V	↕	0	X	16ビット・キャリを含む加算
SBC HL, ss	↕	↕	V	↕	1	X	16ビット・キャリを含む減算
RLA; RLCA; RRA; RRCA	↕	•	•	•	0	0	ローテイト・アキュムレータ
RL s; RLC s; RR s; RRC s SLA s; SRA s; SRL s	↕	↕	P	↕	0	0	ローテイト、シフトS
RLD, RRD	•	↕	P	↕	0	0	ローテイト・ディジット　左、右
DAA	↕	↕	P	↕	•	↕	デシマル・アジャスト・アキュムレータ
CPL	•	•	•	•	1	1	アキュムレータ補数変換
SCF	1	•	•	•	0	0	セット・キャリ
CCF	↕	•	•	•	0	X	キャリ補数変換
IN r,(C)	•	↕	P	↕	0	0	レジスタ間接入力
INI; IND; OUTI; OUTD	X	↕	X	X	1	X	ブロック入出力
INIR; INDR; OTIR; OTDR	X	1	X	X	1	X	B≒0ならばZ＝0、その他はZ＝1
LDI, LDD	•	X	↕	X	0	0	ブロック転送
LDIR, LDDR	•	X	0	X	0	0	BC≒0ならばP/V＝1、その他はP/V＝0
CPI, CPIR, CPD, CPDR	•	↕	↕	↕	1	↕	ブロック・サーチ A＝(HL)ならばZ＝1、その他はZ＝0 BC≒0ならP/V＝1、その他はP/V＝0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	IFFの内容がP/Vにコピーされる
BIT b, s	•	↕	X	X	0	1	Sのビットbの内容がZにコピーされる

記号の説明

C	：キャリ/リンク・フラグ	結果のMSBからのキャリがあれば、C＝1。
Z	：ゼロ・フラグ	零ならば、Z＝1。
S	：サイン・フラグ	結果のMSBが1ならば、S＝1。
P/V	：パリティとオーバ・フロー兼用フラグ	結果が奇数、またはオーバ・フローならば、P/V＝1。 結果が偶数ならば、P/V＝0。
H	：ハーフ・キャリ・フラグ	結果にキャリ、ボローがあれば、H=1。
N	：加/減算フラグ	さきの演算が減算ならば、N=1。
↕	：	操作の結果、変化する。
•	：	操作の結果、変化しない。
0	：	操作により、リセットされる。
1	：	操作により、セットされる。
X	：	無視してよい。
V	：	オーバ・フラグとして扱われる。
P	：	パリティ・フラグとして扱われる。
s	：	8ビット・ロケーション。
R	：	リフレッシュ・カウンタ。
I	：	Iレジスタ（割り込みベクトルの上位バイト用）。
r	：	CPUレジスタ A, B, C, D, E, H, L。
ss	：	16ビット・ロケーション。
n	：	8ビット値（0～255）。
nn	：	16ビット値（0～65535）。

表6－1

第7章　OPコードとタイミング表

この章で、Z－80の命令セットをまとめておく。これらをグループ別に表7－1から表7－11に示し、各表で、次の各項について示す。

アセンブリ語ニーモニックのOPコード

マシン語のOPコード

シンボリック・オペレーション（CPUの動作、または操作）

命令実行後のフラグ・レジスタの内容

命令のバイト数（メモリ・サイクル数）

フェッチ、実行に必要なTステート（外部クロック周期）の総数

これらの表はできるだけその表だけで意味がわかるよう配慮されている。

ニーモニック	シンボリック オペレーション	OPコード 76 543 210	HEXコード （基本）	フラグ C	Z	P/V	S	N	H	バイト数	Mサイクル数	Tステート数	備考
LD r, r′	r ← r′	01 r r′	40＋	•	•	•	•	•	•	1	1	4	
LD r, n	r ← n	00 r 110 ← n →	06＋	•	•	•	•	•	•	2	2	7	
LD r, (HL)	r ← (HL)	01 r 110	46＋	•	•	•	•	•	•	1	2	7	
LD r, (IX+d)	r ← (IX+d)	11 011 101 01 r 110 ← d →	DD 46＋	•	•	•	•	•	•	3	5	19	
LD r, (IY+d)	r ← (IY+d)	11 111 101 01 r 110 ← d →	FD 46	•	•	•	•	•	•	3	5	19	
LD (HL), r	(HL) ← r	01 110 r	70＋	•	•	•	•	•	•	1	2	7	
LD (IX+d), r	(IX+d) ← r	11 011 101 01 110 r ← d →	DD 70＋	•	•	•	•	•	•	3	5	19	
LD (IY+d), r	(IY+d) ← r	11 111 101 01 110 r ← d →	FD 70＋	•	•	•	•	•	•	3	5	19	
LD (HL), n	(HL) ← n	00 110 110 ← n →	36	•	•	•	•	•	•	2	3	10	
LD (IX+d), n	(IX+d) ← n	11 011 101 00 110 110 ← d → ← n →	DD 36	•	•	•	•	•	•	4	5	19	
LD (IY+d), n	(IY+d) ← n	11 111 101 00 110 110 ← d → ← n →	FD 36	•	•	•	•	•	•	4	5	19	
LD A, (BC)	A ← (BC)	00 001 010	0A	•	•	•	•	•	•	1	2	7	
LD A, (DE)	A ← (DE)	00 011 010	1A	•	•	•	•	•	•	1	2	7	
LD A, (nn)	A ← (nn)	00 111 010 ← n → ← n →	3A	•	•	•	•	•	•	3	4	13	
LD (BC), A	(BC) ← A	00 000 010	02	•	•	•	•	•	•	1	2	7	
LD (DE), A	(DE) ← A	00 010 010	12	•	•	•	•	•	•	1	2	7	
LD (nn), A	(nn) ← A	00 110 010 ← n → ← n →	32	•	•	•	•	•	•	3	4	13	
LD A, I	A ← I	11 101 101 01 010 111	ED 57	•	↕	IFF↕	↕	0	0	2	2	9	
LD A, R	A ← R	11 101 101 01 011 111	ED 5F	•	↕	IFF ↕	↕	0	0	2	2	9	
LD I, A	I ← A	11 101 101 01 000 111	ED 47	•	•	•	•	•	•	2	2	9	
LD R, A	R ← A	11 101 101 01 001 111	ED 4F	•	•	•	•	•	•	2	2	9	

r, r′	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

注）r、r′はA、B、C、D、E、H、Lレジスタを指す。
IFF（割り込みイネーブル・フリップ・フロップ）はP/Vフラグにコピーされる。

フラグ：•＝変化しない、0＝リセット、1＝セット、X＝不定。
↕＝操作の結果でセットまたはリセットされる。

表7－1　8ビット・ロード

ニーモニック	シンボリック オペレーション	OPコード 76 543 210	HEXコード (基本)	フラグ C	Z	P/V	S	N	H	バイト数	Mサイクル数	Tステート数	備考
LD dd, nn	dd ← nn	00 dd0 001 ← n → ← n →	0 1 +	•	•	•	•	•	•	3	3	10	dd: 00 BC, 01 DE, 10 HL, 11 SP
LD IX, nn	IX ← nn	11 011 101 00 100 001 ← n → ← n →	D D 2 1	•	•	•	•	•	•	4	4	14	
LD IY, nn	IY ← nn	11 111 101 00 100 001 ← n → ← n →	F D 2 1	•	•	•	•	•	•	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	00 101 010 ← n → ← n →	2 A	•	•	•	•	•	•	3	5	16	nnは2バイト数 下位1バイトはOPコードの直後。 上位1バイトはその次に入る。
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	11 101 101 01 dd1 011 ← n → ← n →	E D 4 B +	•	•	•	•	•	•	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	11 011 101 00 101 010 ← n → ← n →	D D 2 A	•	•	•	•	•	•	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	11 111 101 00 101 010 ← n → ← n →	F D 2 A	•	•	•	•	•	•	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	00 100 010 ← n → ← n →	2 2	•	•	•	•	•	•	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	11 101 101 01 dd0 011 ← n → ← n →	E D 4 3 +	•	•	•	•	•	•	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	11 011 101 00 100 010 ← n → ← n →	D D 2 2	•	•	•	•	•	•	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	11 111 101 00 100 010 ← n → ← n →	F D 2 2	•	•	•	•	•	•	4	6	20	
LD SP, HL	SP ← HL	11 111 001	F 9	•	•	•	•	•	•	1	1	6	
LD SP, IX	SP ← IX	11 011 101 11 111 001	D D F 9	•	•	•	•	•	•	2	2	10	
LD SP, IY	SP ← IY	11 111 101 11 111 001	F D F 9	•	•	•	•	•	•	2	2	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	11 qq0 101	C 5 +	•	•	•	•	•	•	1	3	11	qq: 00 BC, 01 DE, 10 HL, 11 AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	11 011 101 11 100 101	D D E 5	•	•	•	•	•	•	2	4	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	11 111 101 11 100 101	F D E 5	•	•	•	•	•	•	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	11 qq0 001	C 1 +	•	•	•	•	•	•	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	11 011 101 11 100 001	D D E 1	•	•	•	•	•	•	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	11 111 101 11 100 001	F D E 1	•	•	•	•	•	•	2	4	14	

注) ddはレジスタ・ペア BC、DE、HL、SP。
qqはレジスタ・ペア AF、BC、DE、HL。
添字H、Lはそれぞれ高位バイト、低位バイトを表わす。
例) BC_L=C、AF_H=A

フラグ •＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定である。
↕＝操作の結果により、セットまたはリセットされる。

表7－2　16ビット・ロード

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						イト数	Mサイクル数	Tステート数	備　考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
EX DE, HL	DE ↔ HL	11 101 011	EB	•	•	•	•	•	•	1	1	4	
EX AF, AF'	AF ↔ AF'	00 001 000	08	•	•	•	•	•	•	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix}\leftrightarrow\begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	11 011 001	D9	•	•	•	•	•	•	1	1	4	レジスタの切り替え
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	11 100 011	E3	•	•	•	•	•	•	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	11 011 101 11 100 011	DD E3	•	•	•	•	•	•	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	11 111 101 11 100 011	FD E3	•	•	•	•	•	•	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	11 101 101 10 100 000	ED A0	•	•	① ↕	•	0	0	2	4	16	ポインタ 1増 バイト・カウンタ 1減
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	11 101 101 10 110 000	ED B0	•	•	② 0	•	0	0	2	5	21	BC≠0のとき
	BC=0ならば終わり									2	4	16	BC=0のとき
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	11 101 101 10 101 000	ED A8	•	•	① ↕	•	0	0	2	4	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	11 101 101 10 111 000	ED B8	•	•	② 0	•	0	0	2	5	21	BC≠0のとき
	BC=0ならば終わり									2	4	16	BC=0のとき
CPI	A - (HL) HL ← HL+1 BC ← BC-1	11 101 101 10 100 001	ED A1	•	③ ↕	① ↕	↕	1	↕	2	4	16	
CPIR	A - (HL) HL ← HL+1 BC ← BC-1	11 101 101 10 110 001	ED B1	•	③ ↕	① ↕	↕	1	↕	2	5	21	BC≠0 かつ A≠(HL) のとき
	A=(HL) または BC=0ならば終わり									2	4	16	BC=0 か A=(HL) のとき
CPD	A - (HL) HL ← HL-1 BC ← BC-1	11 101 101 10 101 001	ED A9	•	③ ↕	① ↕	↕	1	↕	2	4	16	
CPDR	A - (HL) HL ← HL-1 BC ← BC-1	11 101 101 10 111 001	ED B9	•	③ ↕	① ↕	↕	1	↕	2	5	21	BC≠0 かつ A≠(HL) のとき
	A=(HL) または BC=0ならば終わり									2	4	16	BC=0 か A=(HL) のとき

注) ① BC−1=0ならばP/Vは0、その他は1。
② 命令終了時のみ P/Vは0。
③ A=(HL)ならばZは1、その他は0。

フラグ • =変化しない。
0=セット、1=リセット。
↕=操作結果により、セットまたはリセットされる。

表7−3　交換、ブロック転送、サーチ

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
ADD A, r	A ← A + r	10 k r	8 0 +	↕	↕	V	↕	0	↕	1	1	4	r: B 000, C 001, D 010, E 011, H 100, L 101, A 111
ADD A, n	A ← A + n	11 k 110 ← n →	C 6 +	↕	↕	V	↕	0	↕	2	2	7	
ADD A, (HL)	A ← A + (HL)	10 k 110	8 6 +	↕	↕	V	↕	0	↕	1	2	7	
ADD A, (IX+d)	A←A + (IX+d)	11 011 101 10 k 110 ← d →	D D 8 6 +	↕	↕	V	↕	0	↕	3	5	19	
ADD A, (IY+d)	A←A+(IY+d)	11 111 101 10 k 110 ← d →	F D 8 6 +	↕	↕	V	↕	0	↕	3	5	19	
ADC A, s	A ← A + s + CY	上記ADDを基本形としそれぞれ4種ある。(備考参照)		↕	↕	V	↕	0	↕	1 ※1 2 1 3	1 ※1 2 2 5	4 ※1 7 7 19	ニーモニック: ADD 000, ADC 001, SUB 010, SBC 011, AND 100, OR 110, XOR 101, CP 111
SUB s	A ← A - s			↕	↕	V	↕	1	↕				
SBC A, s	A ← A - s - CY			↕	↕	V	↕	1	↕				
AND s	A ← A ∧ s			0	↕	P	↕	0	1				
OR s	A ← A ∨ s			0	↕	P	↕	0	0				
XOR s	A ← A ⊕ s			0	↕	P	↕	0	0				
CP s	A - s			↕	↕	V	↕	1	↕				S = r, n, (HL), (IX+d), (IY+d)
INC r	r ← r + 1	00 r ℓ	0 0 +	•	↕	V	↕	0	↕	1	1	4	
INC (HL)	(HL) ← (HL)+1	00 110 ℓ	3 0 +	•	↕	V	↕	0	↕	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	11 011 101 00 110 ℓ ← d →	D D 3 0 +	•	↕	V	↕	0	↕	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	11 111 101 00 110 ℓ ← d →	F D 3 0 +	•	↕	V	↕	0	↕	3	6	23	ニーモニック: INC 100, DEC 101
DEC m	m ← m-1	上記INCを基本形とし4種ある。		•	↕	V	↕	1	↕	1 ※2 1 3 3	1 ※2 3 6 6	4 ※2 11 23 23	m = r, (HL), (IX+d), (IY+d)

※1 sに依存する。
※2 mに依存する。

注) Vはオーバ・フロー・フラグとして、Pはパリティ・フラグとして扱われることを意味する。

フラグ •＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定。
↕＝操作結果により、セットまたはリセットされる。

表7－4　8ビット算術・論理演算

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備　考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
DAA	10進補正 (加算、減算)	00 100 111	27	↕	↕	P	↕	•	↕	1	1	4	デシマル・アジャスト・アキュムレータ
CPL	A ← $\overline{A}$	00 101 111	2F	•	•	•	•	1	1	1	1	4	1の補数Acc
NEG	A ← 0 − A	11 101 101 01 000 100	ED 44	↕	↕	V	↕	1	↕	2	2	8	2の補数Acc
CCF	CY ← $\overline{CY}$	00 111 111	3F	↕	•	•	•	0	X	1	1	4	キャリの反転
SCF	CY ← 1	00 110 111	37	1	•	•	•	0	0	1	1	4	キャリ・セット
NOP	No operation	00 000 000	00	•	•	•	•	•	•	1	1	4	
HALT	CPU待機	01 110 110	76	•	•	•	•	•	•	1	1	4	
DI	IFF ← 0	11 110 011	F3	•	•	•	•	•	•	1	1	4	ディセーブル割り込み
EI	IFF ← 1	11 111 011	FB	•	•	•	•	•	•	1	1	4	イネーブル割り込み
IM 0	MODE 0 にセット	11 101 101 01 000 110	ED 46	•	•	•	•	•	•	2	2	8	割り込みモードのセット
IM 1	MODE 1 にセット	11 101 101 01 010 110	ED 56	•	•	•	•	•	•	2	2	8	
IM 2	MODE 2 にセット	11 101 101 01 011 110	ED 5E	•	•	•	•	•	•	2	2	8	

注) IFFは割り込みフリップ・フロップ。
CYはキャリ・フリップ・フロップ。

フラグ　•＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定。
↕＝操作の結果により、セットまたはリセットされる。

表7－5　各種操作およびCPU制御

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
ADD HL, ss	HL ← HL+ss	00 ss1 001	09+	↕	•	•	•	0	X	1	3	11	
ADC HL, ss	HL←HL+ss+CY	11 101 101 01 ss1 010	ED 4A+	↕	↕	V	↕	0	X	2	4	15	
SBC HL, ss	HL←HL-ss-CY	11 101 101 01 ss0 010	ED 42+	↕	↕	V	↕	1	X	2	4	15	
ADD IX, pp	IX ← IX + pp	11 011 101 00 pp1 001	DD 09+	↕	•	•	•	0	X	2	4	15	
ADD IY, rr	IY←IY+rr	11 111 101 00 rr1 001	FD 09+	↕	•	•	•	0	X	2	4	15	
INC ss	ss ← ss + 1	00 ss0 011	03+	•	•	•	•	•	•	1	1	6	
INC IX	IX ← IX + 1	11 011 101 00 100 011	DD 23	•	•	•	•	•	•	2	2	10	
INC IY	IY ← IY + 1	11 111 101 00 100 011	FD 23	•	•	•	•	•	•	2	2	10	
DEC ss	ss ← ss - 1	00 ss1 011	0B+	•	•	•	•	•	•	1	1	6	
DEC IX	IX ← IX - 1	11 011 101 00 101 011	DD 2B	•	•	•	•	•	•	2	2	10	
DEC IY	IY ← IY - 1	11 111 101 00 101 011	FD 2B	•	•	•	•	•	•	2	2	10	

ss	レジスタ
00	BC
01	DE
10	HL
11	SP

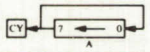
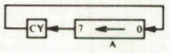
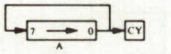
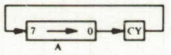
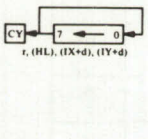
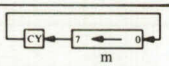
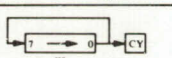
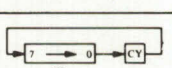
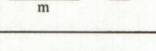
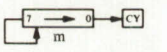
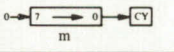
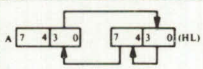
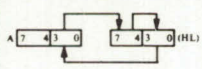
pp	レジスタ
00	BC
01	DE
10	IX
11	SP

rr	レジスタ
00	BC
01	DE
10	IY
11	SP

注) ssはレジスタ・ペアBC、DE、HL、SP。
ppはレジスタ・ペアBC、DE、IX、SP。
rrはレジスタ・ペアBC、DE、IY、SP。

フラグ ●＝変化しない。
0＝リセットされる。
X＝不定。
↕＝操作の結果により、セットまたはリセットされる。

表7－6　16ビット算術演算

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
RLCA		00 000 111	07	‡	•	•	•	0	0	1	1	4	左ローテイト・サーキュラ・アキュムレータ
RLA		00 010 111	17	‡	•	•	•	0	0	1	1	4	左ローテイト アキュムレータ
RRCA		00 001 111	0F	‡	•	•	•	0	0	1	1	4	右ローテイト・サーキュラ・アキュムレータ
RRA		00 011 111	1F	‡	•	•	•	0	0	1	1	4	右ローテイト ・アキュムレータ
RLC r		11 001 011 00 k r	CB 00+	‡	‡	P	‡	0	0	2	2	8	左ローテイト・サーキュラ・レジスタ r
RLC (HL)		11 001 011 00 k 110	CB 06+	‡	‡	P	‡	0	0	2	4	15	
RLC (IX+d)		11 011 101 11 001 011 ← d → 00 k 110	DD CB 06+	‡	‡	P	‡	0	0	4	6	23	
RLC (IY+d)		11 111 101 11 001 011 ← d → 00 k 110	FD CB 06+	‡	‡	P	‡	0	0	4	6	23	
RL m		上記RLCを基本形としそれぞれ4種ある。(備考参照)		‡	‡	P	‡	0	0	2※ 2 4 4	2※ 4 6 6	8※ 15 23 23	
RRC m				‡	‡	P	‡	0	0				
RR m				‡	‡	P	‡	0	0				
SLA m				‡	‡	P	‡	0	0				
SRA m				‡	‡	P	‡	0	0				
SRL m				‡	‡	P	‡	0	0				
RLD		11 101 101 01 101 111	ED 6F	•	‡	P	‡	0	0	2	5	18	左ローテイト・ディジット アキュムレータ, (HL) アキュムレータ上位4ビット変化せず。
RRD		11 101 101 01 100 111	ED 67	•	‡	P	‡	0	0	2	5	18	右ローテイト・ディジット アキュムレータ, (HL) アキュムレータ上位4ビット変化せず。

r, (HL), (IX+d), (IY+d)

r	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

ニーモニック	k
RLC	000
RRC	001
RL	010
RR	011
SLA	100
SRA	101
SRL	111

m=r, (HL), (IX+d), (IY+d)

※mに依存する。

フラグ ●=変化しない。
0=リセットされる
1=セットされる。
X=不定
‡=操作結果により、セットまたはリセットされる。

表7－7 ローテイト、シフト

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
BIT b, r	$Z \leftarrow \overline{r}_b$	11 001 011 01 b r	CB 40+	•	↕	X	X	0	1	2	2	8	
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	11 001 011 01 b 110	CB 46+	•	↕	X	X	0	1	2	3	12	
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	11 011 101 11 001 011 ← d → 01 b 110	DD CB 46+	•	↕	X	X	0	1	4	5	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	11 111 101 11 001 011 ← d → 01 b 110	FD CB 46+	•	↕	X	X	0	1	4	5	20	
SET b, r	$r_b \leftarrow 1$	11 001 011 a b r	CB	•	•	•	•	•	•	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	11 001 011 a b 110	CB 06+	•	•	•	•	•	•	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	11 011 101 11 001 011 ← d → a b 110	DD CB 06+	•	•	•	•	•	•	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	11 111 101 11 001 011 ← d → a b 110	FD CB 06+	•	•	•	•	•	•	4	6	23	
RES b, m	$m_b \leftarrow 0$	上記SET を基本形と し4種ある。								2※ 2 4 4	2※ 4 6 6	8※ 15 23 23	

r	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	テスト・ビット
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

ニーモニック	a
SET	11
RES	10

m=r, (HL), (IX+d), (IY+d)

※mに依存する。

注) m_bの$_b$はmの示すメモリの位置またはレジスタの ビット0～7を示す。

フラグ ●＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定
↕＝操作の結果により、セットまたはリセットされる。

表7－8 ビット操作、テスト

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備　考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
JP nn	PC ← nn	11 000 011 ← n → ← n →	C 3	•	•	•	•	•	•	3	3	10	
JP cc, nn	ccが真ならば PC ←nn, その他は次へ	11 cc 010 ← n → ← n →	C 2+	•	•	•	•	•	•	3	3	10	
JR e	PC ← PC + e	00 011 000 ← e-2 →	1 8	•	•	•	•	•	•	2	3	12	
JR C, e	C＝1 ならば PC ← PC+e	00 111 000 ← e-2 →	3 8	•	•	•	•	•	•	2	3	12	
	C＝0 ならば 次へ									2	2	7	
JR NC, e	C＝0 ならば PC ← PC + e	00 110 000 ← e-2 →	3 0	•	•	•	•	•	•	2	3	12	
	C＝1 ならば 次へ									2	2	7	
JR Z, e	Z＝1 ならば PC ← PC + e	00 101 000 ← e-2 →	2 8	•	•	•	•	•	•	2	3	12	
	Z＝0 ならば 次へ									2	2	7	
JR NZ, e	Z＝0 ならば PC ← PC + e	00 100 000 ← e-2 →	2 0	•	•	•	•	•	•	2	3	12	
	Z＝1 ならば 次へ									2	2	7	
JP (HL)	PC ← HL	11 101 001	E 9	•	•	•	•	•	•	1	1	4	
JP (IX)	PC ← IX	11 011 101 11 101 001	D D E 9	•	•	•	•	•	•	2	2	8	
JP (IY)	PC ← IY	11 111 101 11 101 001	F D E 9	•	•	•	•	•	•	2	2	8	
DJNZ e	B ← B-1 B≠0 ならば PC ← PC + e	00 010 000 ← e-2 →	1 0	•	•	•	•	•	•	2	3	13	
	B＝0 ならば 次へ									2	2	8	

cc	条件
000	NZ
001	Z
010	NC
011	C
100	PO
101	PE
110	P
111	M

NZ：ノンゼロ
Z：ゼロ
C：キャリ
PO：パリティ奇数
PE：パリティ偶数
P：正
M：負

注）eは相対アドレシング・モードでのイクステンション。
eは符号付２の補数値（−126〜+129)。
e−２はPCが自動的に＋２されてしまうので、これをキャンセルした値である。
eそのものは、OPコードの位置から計算した値である。

フラグ　●＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定。
‡＝操作の結果により、セットまたはリセットされる。

表7−9　ジャンプ

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備考
		76 543 210	(基本)	C	Z	P/V	S	N	H				
CALL nn	(SP-1)←PC_H (SP-2)←PC_L PC←nn	11 001 101 ← n → ← n →	CD	•	•	•	•	•	•	3	5	17	
CALL cc, nn	ccが真ならば CALL nnと同じ	11 cc 100 ← n →	C4+	•	•	•	•	•	•	3	5	17	
	その他ならば 次へ	← n →								3	3	10	
RET	PC_L←(SP) PC_H←(SP+1)	11 001 001	C9	•	•	•	•	•	•	1	3	10	
RET cc	ccが真ならば RETと同じ	11 cc 000	C0+	•	•	•	•	•	•	1	3	11	
	その他ならば 次へ									1	1	5	
RETI	割り込みからの リターン	11 101 101 01 001 101	ED 4D	•	•	•	•	•	•	2	4	14	
RETN	ノン・マスカブル 割り込みからの リターン	11 101 101 01 000 101	ED 45	•	•	•	•	•	•	2	4	14	
RST p	(SP-1)←PC_H (SP-2)←PC_L PC_H←0 PC_L←P	11 t 111	C7+	•	•	•	•	•	•	1	3	11	

cc	条件
000	NZ
001	Z
010	NC
011	C
100	PO
101	PE
110	P
111	M

NZ: ノンゼロ
Z: ゼロ
C: キャリ
PO: パリティ奇数
PE: パリティ偶数
P: 正
M: 負

t	P
000	00 H
001	08 H
010	10 H
011	18 H
100	20 H
101	28 H
110	30 H
111	38 H

フラグ ●＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定。
↕＝操作結果により、セットまたはリセットされる。

表7－10 コール、リターン

ニーモニック	シンボリック オペレーション	OPコード	HEXコード	フラグ						バイト数	Mサイクル数	Tステート数	備　考
		76 543 210	（基本）	C	Z	P/V	S	N	H				
IN A, (n)	A ← (n)	11 011 011 ← n →	D B	•	•	•	•	•	•	2	3	11	n→ A_0 ~ A_7 Acc→ A_8 ~ A_{15}
IN r, (C)	r ← (C)	11 101 101 01 r 000	E D 4 0 +	•	↕	P	↕	0	↕	2	3	12	
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	11 101 101 10 100 010	E D A 2	X	① ↕	X	X	1	X	2	4	16	C→ A_0 ~ A_7 B→ A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 B=0まで繰り返す	11 101 101 10 110 010	E D B 2	X	② 1	X	X	1	X	2 2	5 (B≠0のとき) 4 (B=0のとき)	21 16	
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	11 101 101 10 101 010	E D A A	X	① ↕	X	X	1	X	2	4	16	
INDR	(HL) ← (C) B ← B - 1 HL ← HL -1 B=0まで繰り返す	11 101 101 10 111 010	E D B A	X	② 1	X	X	1	X	2 2	5 (B≠0のとき) 4 (B=0のとき)	21 16	
OUT (n), A	(n) ← A	11 010 011 ← n →	D 3	•	•	•	•	•	•	2	3	11	n→(A-BUS)$_{0\sim7}$ Acc→(A-BUS)$_{8\sim15}$
OUT (C), r	(C) ← r	11 101 101 01 r 001	E D 4 1 +	•	•	•	•	•	•	2	3	12	
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	11 101 101 10 100 011	E D A 3	X	① ↕	X	X	1	X	2	4	16	C→ A_0 ~ A_7 B→ A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 B=0まで繰り返す	11 101 101 10 110 011	E D B 3	X	② 1	X	X	1	X	2 2	5 (B≠0のとき) 4 (B=0のとき)	21 16	
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	11 101 101 10 101 011	E D A B	X	① ↕	X	X	1	X	2	4	16	
OTDR	(C) ← (HL) B ← B - 1 HL ← HL -1 B=0まで繰り返す	11 101 101 10 111 011	E D B B	X	② 1	X	X	1	X	2 2	5 (B≠0のとき) 4 (B=0のとき)	21 16	

r	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

注）①B-1が零になればZフラグがセットされ、それ以外のときはリセットされる。
$A_{0\sim15}$はアドレス・バス。
②Zフラグは、命令完了時のみセットされる。

フラグ　●＝変化しない。
0＝リセットされる。
1＝セットされる。
X＝不定。
↕＝操作の結果により、セットまたはリセットされる。

表7－11　入力/出力

第8章　割り込み応答

割り込みは、周辺デバイスがCPUの通常処理を一時保留させ、周辺デバイスの処理ルーチンを実行させるために設けられている。

普通このサービス（処理）には、CPUと周辺とのデータ、ステータス、あるいは制御情報の送受などが含まれている。

このサービス・ルーチンが終了すれば、CPUは割り込みがかかったまえの処理に戻る。

割り込みのイネーブル/ディセーブル

Z－80 CPUには、ソフト的にマスク可能な割り込みとマスク不可能な割り込みの2つの割り込み入力端子が設けられている。

ノン・マスカブル割り込み（NMI）はプログラムによってディセーブル（禁止）できず、周辺デバイスが要求を出せば必ず受け付けられる。この割り込みは普通重要な機能、たとえば不意の停電のようないつ起こるかわからない事故に対する処置用として使用される。

マスカブル割り込み（INT）はプログラムによりイネーブル（許可）、ディセーブル（禁止）を自由に選択できるもので、計時（タイマ）のように途中で邪魔（割り込み）が入っては困る場合に、プログラムにより適当にディセーブルしておくことができる。

CPU内にイネーブル・フリップ・フロップ(IFF)が設けられているので、これをプログラムによってEI(イネーブル割り込み）またはDI（ディセーブル割り込み）命令でセット、またはリセットする。

IFFがリセットされれば、CPUは割り込みを受け付けなくなる。

実際には、IFFはIFF_1とIFF_2の2つのイネーブル・フリップ・フロップにより構成される。

IFF_1	IFF_2
リセットされれば割り込み禁止となる。	IFF_1の一時保持用。

IFF_1の状態で、イネーブルかディセーブルかが決まる。IFF_2はIFF_1の一時保持用である。このIFF_2が必要な理由をあとで述べる。

CPUがリセットされたとき、IFF_1、IFF_2もリセット状態となり、割り込みはディセーブルとなる。この後、任意の時点でEI命令を使用して割り込みをイネーブルにすることができる。

EI 命令が実行されたとき、待たされている割り込み要求は EI 命令の次の命令が実行されたあとでなければ受け付けられない。

この１命令分の遅延は、次の命令がリターン命令の場合、EI 命令を実行させたとき、リターン命令が完了した後で待機中の割り込みが受け付けられるようにするために、設けられている。

EI 命令で IFF_1、IFF_2 の両方がセットされ、イネーブル状態となる。割り込みが受け付けられたときには自動的に両方ともリセットされ、プログラマが新たに EI 命令を実行させることがなければ割り込みは禁止されたままになる。

上に述べた場合においては、いずれも IFF_1、IFF_2 はつねに同様である点に注意されたい。

IFF_2 は NMI（ノン・マスカブル割り込み）の発生時に IFF_1 の状態をセーブしておくために設けられている。

NMI が受け付けられたとき、続けて割り込みが入るのを禁止するために IFF_1 はリセットされるが、さきの状態がリセットかセットか（つまり、今 INT の処理中であるか否か）を保存しておく必要がある。IFF_2 はこのためのもので、IFF_1 と同じ状態を IFF_2 に入れておく。

LD 命令（LD　A、I あるいは LD　A、R）を実行したとき、IFF_2 の状態がパリティ・フラグにコピーされるので、それをテストしたり、別の場所にストアしたりすることができる。

RETN（ノン・マスカブル割り込みからのリターン）命令を実行すると、IFF_2 の内容が IFF_1 にコピー・バックされ、NMI の入るまえの IFF_1 の状態が再生される。

図８－１に、２つのフリップ・フロップの状態を示す。

操　作	IFF_1	IFF_2	
CPU リセット	0	0	
DI	0	0	
EI	1	1	
LD　A, I	●	●	IFF_2 →パリティ・フラグ
LD　A, R	●	●	IFF_2 →パリティ・フラグ
NMI	0	●	
RETN	IFF_2	●	$IFF_2 \rightarrow IFF_1$
DI, then NMI	0	0	
EI, then NMI	0	1	
EI, NMI, then RETN	1	1	

● ＝ 変化なしの意味

図８－１　IFF_1, IFF_2 の状態

CPUの応答

ノン・マスカブル

ノン・マスカブル割り込みはどの時点でも受け付けられる。これが入力すると、CPUはフェッチした次の命令を無視して、メモリ番地0066Hにある命令から実行する。つまり、あたかもリスタート命令をフェッチしたかのように動作するが、この場合、正しくはプログラム可能な8つのリスタート・ロケーションのいずれかにリスタートするわけではない。メモリのページ・ゼロの特定のアドレス（66H）へのコール命令の実行である。

マスカブル

Z－80 CPUにはプログラムで指定できる3種のマスク可能な割り込みモードがある。

モード　0

このモードは8080Aの割り込み応答モードと同様である。このモードでは、割り込みをかけているデバイスがデータ・バス上に何らかの命令を置き、CPUがそれを実行する。

割り込みをかけているデバイスは、プログラム・メモリに代って次に実行すべき命令をデータ・バスに乗せる。この命令は1バイト命令の場合、通常リスタート命令である。3バイトのコール命令を用いた場合、メモリのどの位置からでも実行を始めることができる。

必要なクロック・サイクル数は通常の数より2クロック多くなる。これはCPUが自動的に2つのウエイト・サイクルを挿入して、割り込み制御用の外部デージー・チェーンを全部動作させるのに必要な時間をとるためである。

割り込み応答の詳細なタイミングについては第5章に示してある。

リセットを行えば、自動的にCPUはモード0になる。

モード　1

このモードをプログラムで選択しておくと、CPUは割り込みに対し、位置0038Hへのリスタートを実行する。この応答はコールする位置が0066Hの代りに、0038Hになっている以外はNMI応答とほぼ同様である。他の異なる点として、2つのウエイト・サイクルが追加される点がある。

モード　2

このモードは最も強力な応答モードであり、1バイトの指定でどのメモリの位置でも間接コールができる。

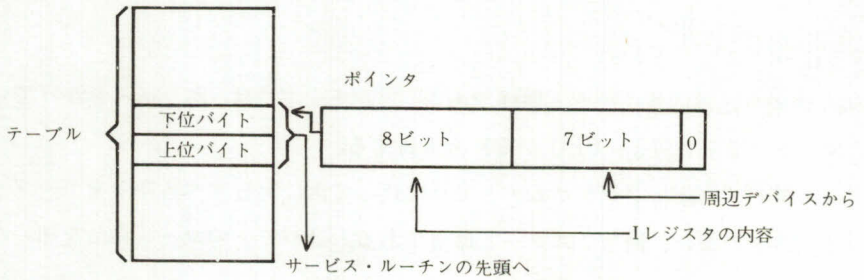
このモードを使用する場合、まえもってプログラムにより割り込みサービス・ルーチンのスタート・アドレス（16ビット）のテーブルを適当な位置に配置しておく。この位置はメモリのどの場所でもよい。

割り込みを受け付けたとき、16ビットのポインタで、必要な割り込みサービス・ルーチンのスタート・アドレスをテーブルからもってくるが、このポインタとして上位8ビットにはIレジスタの内容を充当する。

このIレジスタには、まえもってプログラムによりLD命令で必要な値をロードしておかねばならない。

CPUのリセットによりIレジスタもクリアされるので、Iレジスタの初期値は零である。

ポインタの下位8ビットには割り込みをかけているデバイスから供給しなければならないが、実際は7ビットが有効で最下位ビットはこのモードではつねに零である。というのは、テーブル上のスタート・アドレスは2バイトずつで、この2バイトは必ず偶数アドレスから順に下位バイト、上位バイトと入れておかねばならないからである。



プログラムにより割り込みを受け付けるまでに、必要なアドレスをテーブルにセットしておけばよい。このテーブルを読み出し・書き込みメモリ内に指定しておくと、いつでもプログラムで変更ができるので、異なる周辺デバイスを異なったサービス・ルーチンで処理することもできる。

割り込みをかけているデバイスがポインタの下位8ビットを乗せたあと、CPUは自動的に次に実行すべきPC（プログラム・カウンタ）の内容をスタックにプッシュし、テーブルからスタート・アドレスを持ってそのアドレスへジャンプする。

この応答モードでは、19クロック周期を必要とし、7クロックで割り込みをかけているデバイスからの下位8ビットをフェッチ、6クロックでPCの内容のセーブ、6クロックでジャンプ・アドレスを得る。

Z－80周辺デバイスはすべてデージー・チェーン構造の割り込み優先順位回路を含んでおり、割り込みアクノリッジ期間中CPUにベクトル（プログラムされている）を自動的に供給するようになっている点に注意されたい。

これについてはZ－80 PIO、Z－80 CTCなどのマニュアルを参照のこと。

第9章　ハードウエアの構成例

この章では、Z－80 CPU を用いた装置を考えていく場合の参考となるようにいくつかの基本的な構成例を示す。

最小システム

図9－1に極めてシンプルなZ－80のシステム図を示す。

1）５Ｖ電源
2）発振器
3）メモリ
4）入出力回路
5）ＣＰＵ

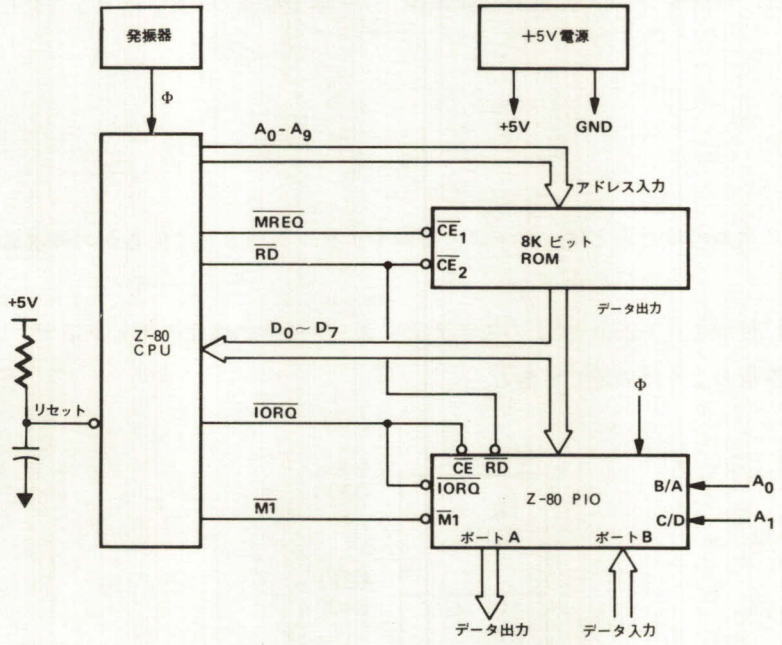


図9－1　Z－80の最小コンピュータ・システム

Z－80 CPU は５Ｖの単一電源のみでよいので、非常にコンパクトなシステムを構成できる。

発振器は簡単なものでよく、５Ｖの方形波が出せればよい。システムを最高速度で動作させないのであれば、RC 発振器で間に合う。CPU を上限周波数で動作させる場合、RCネットワークで発生するドリフト、ジッタが問題になるので、通常は水晶発振器を使用するが、この水晶発振器は TTL などのインバータで作ることができ、２～３の個別部品などで構成できる。

外部メモリは標準の RAM、ROM、PROM をどのように組み合わせて使用してもよい。

この例では、メモリとして８Ｋビットの ROM（１Ｋバイト）を１個使用している。読み出し、書き込み用メモリとして CPU 内のレジスタを使用すれば、外部 RAM を付ける必要はない。

どのようなコンピュータ・システムでも、〝外界〟とのインターフェースとして入出力回路が必要である。

この単純な例でも、8 ビット制御ベクトル用の出力と 8 ビット・ステータス・ワード間の入力を考えている。

入力データは標準の３ステート駆動回路を使用してデータ・バス上に乗せ、また出力データは標準 TTL ラッチを用いてラッチすることができる。

ここで示した例では入出力回路として Z－80 PIO を使用している。図に示すように、データ・バスと直結し外部へ16ビットの TTL コンパチブル入出力信号を出力している。（詳細については、Z－80 PIO マニュアルを参照のこと）

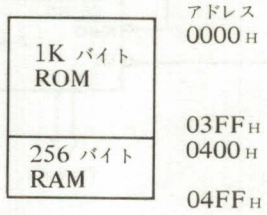
最後に、この例ではたった３つの LSI、簡単な発振器、５Ｖ単一電源のみで、強力なコンピュータを構成している点に注目されたい。

RAM の増設

コンピュータ・システムのほとんどは、データの蓄積やスタック用として何らかの外部読み出し・書き込みメモリが必要である。

図９－２に前の例に付加できる256バイトのスタティック・メモリの構成の仕方を示す。

メモリ・スペースを次のように設定してある。



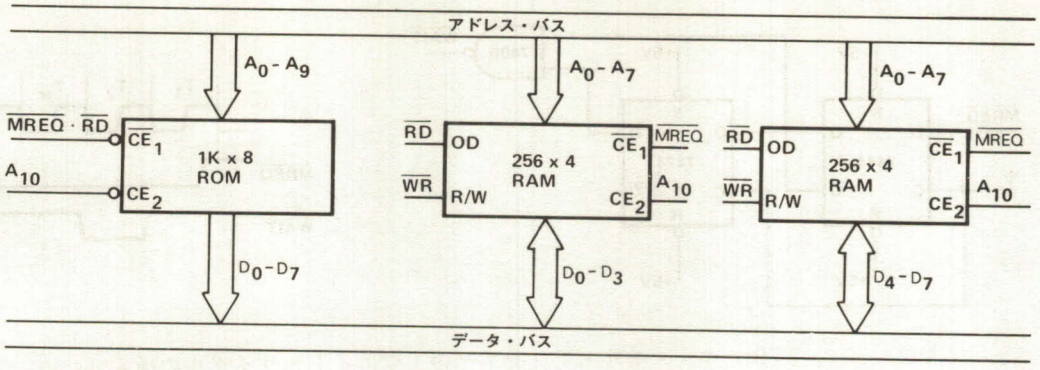


図９－２　ROM、RAM の構成例

この例では、アドレス・ビット A_{10} をチップ・セレクト用として使用して、ROM スペースと RAM スペースを分けている。外部 ROM、RAM を多く使用する場合、チップ・セレクトのために簡単な TTL デコーダが必要となろう。

メモリのスピード制御

用途によってはコストをおさえるためにアクセスの遅いメモリを使用したい場合がある。Ｚ－80はどのようなスピードのメモリでも扱えるようになっている。第４章をふりかえればわかるように、メモリ・アクセス時間に対する要求が最も厳しいのは M1 の命令フェッチ期間である。他のメモリ・アクセスの場合にはクロック１サイクル半が付加されている。そのため、必要に応じて M1 サイクルで、１ウエイト・サイクルを付加して遅いメモリを使用することができる。

図９－３にこの操作をするための簡単な回路を示す。この回路はどのメモリ・アクセスの場合にも１ウエイト・サイクルを挿入できる図９－４のような回路に変えることもできる。

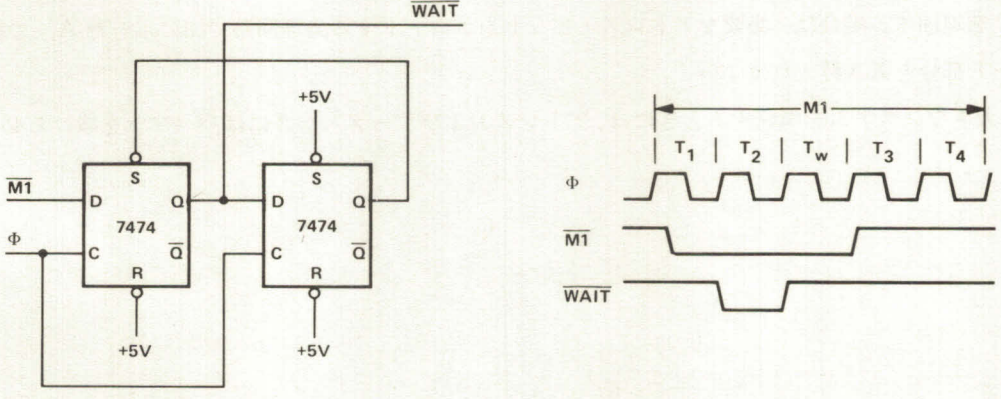


図９－３　M1 サイクルへの１ウエイト・サイクルの挿入

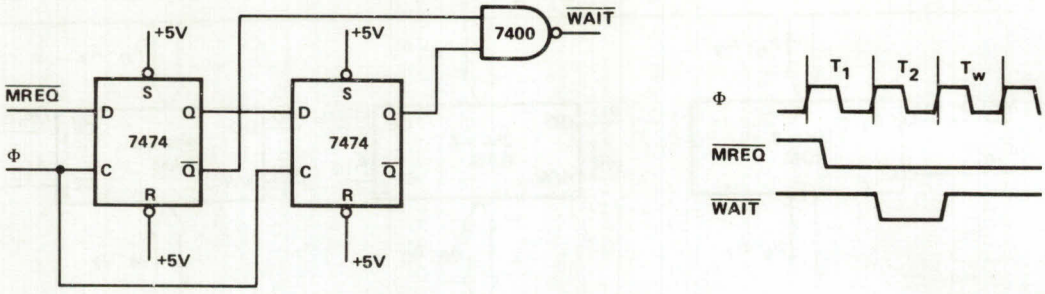


図９－４　メモリ・アクセスへの１ウエイト・サイクルの挿入

ダイナミック・メモリとのインターフェース

この節では、ダイナミック・メモリのインターフェースについて簡単に説明する。

各種のダイナミックRAMはそれぞれ異なった規格になっているので、ここでの説明例を大幅に変更する必要がある場合もあるだろう。したがって、特定のRAMを使用した詳しい説明はしない。よく知られているダイナミックRAMのためのインターフェースについては他の資料を参照されたい。

図９－５に18ピン４KダイナミックRAMを用いた８Kバイトのダイナミック・メモリとのインターフェースに必要なロジックを示す。

この構成では、外部メモリはRAMのみとした例でA_{12}を２ページのうちの片方のメモリ・ページへの切り替え用として使用している。

ダイナミックRAMはリフレッシュ期間内にシステムの全メモリを読み出す必要があるが、アドレスA_0～A_6にCPUから適当なリフレッシュ用アドレスが出される。

メモリを追加する場合は、必要なアドレス・ビットをデコードする論理回路で図に示したA_{12}で制御する２つのゲート部分を置き替えればよい。

普通、大きなシステムを構成するときには、アドレスおよびデータ・バスにはバッファを設ける必要がある。

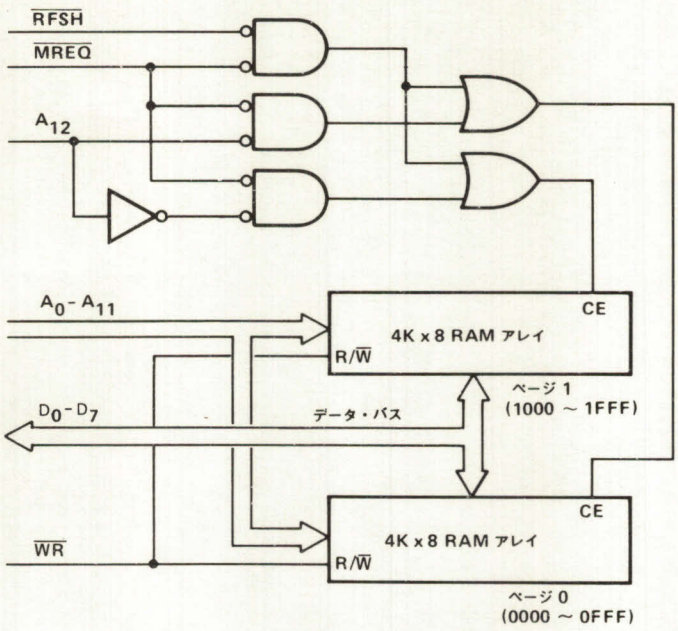


図９－５　ダイナミックRAM用インターフェース

第10章　ソフトウエアの構成例

10. 1　Z－80 CPUのソフトウエア

Z－80の命令セットには、ユーザがZ－80 CPUを系統立てて制御する上で、強力でかつ柔軟性のあるものが用意されている。

主、補助、インデックス・レジスタなどは算術および論理演算のアーギュメントを保持したり、メモリ・アドレスを置いたり、あるいはよく使用するデータをすぐ取り出せるように退避しておいたりすることができるようになっている。

さらに、レジスタの内容と他のレジスタまたはメモリの内容とを余分な一時保持回路を介さず交換することもできる。

とくに、主および補助レジスタ群の内容は次の2つの命令（EX、EXX）を実行するだけで、全部交換できる。

このレジスタ群の交換で、異なる論理手順（たとえば、異なるジョブ、タスク）に対して別々の作業レジスタ群を割り当てることができ、また単一の手順に対しては利用できるレジスタを2倍にすることができる。

レジスタ・ペア群とメモリとの間のデータの出し入れをPUSHとPOP命令で、特別なスタック・ポインタ・レジスタSPを使用してLIFO（Last In-First Out）形式で行うことができる。

このスタック・レジスタは一時データの出し入れ以外に、サブルーチンとリンケージするためにアドレス情報を退避させたり復帰させたりするための用途にも用いられる。

たとえば、サブルーチンをコールしたとき、CALL命令の次のアドレス値はSPで示されるプッシュ・ダウン・スタックの先頭に保存される。

サブルーチンから、コールしたルーチンへ戻る場合、スタックの先頭にあるアドレス値をPC（プログラム・カウンタ）へ移して、先のCALL命令の次から実行を継続する。

スタック・ポインタはPUSH、POP、CALL、RET命令を実行すれば、その時点でのスタックの〝先頭〟位置を移していくよう自動補正が行われる。

このスタックのメカニズムによって、メモリの許す限り無制限にデータを押し込んでいったり、サブルーチン・コールを幾重にも行う多重ネスティングができる。

6種のフラグ（キャリC、ゼロZ、サインS、パリティ/オーバP/V、加/減算N、ハーフ・キャリH）で、命令実行の流れを制御することができる。

これらのフラグは、算術、論理、シフト、比較などの操作の結果を反映してセット/リセットされる。

あるフラグがセット（あるいはリセット）されれば、そのフラグで条件ジャンプ、あるいはリターン命令を

制御できる。

これらの命令でビット、バイト、16ビット・データの操作のあと、必要な論理制御もできる。

論理操作としてAND、OR、XOR（排他的論理和）、CPL（NOT）、NEG（２の補数）があり、これらによりアキュムレータと、１）他の８ビット・レジスタ、２）メモリの内容、３）イミディエット・オペランドとの間で論理演算が行える。

さらに、左右両方への算術、論理シフトがあり、８ビット主レジスタ全部あるいはどこのメモリの位置に対しても直接操作できるようになっている。

キャリ・フラグはこれらのシフト命令でシフト列の中に入っている場合もあり、単独で設定される場合もある。

10. 2　特殊な命命の使用例

A．メモリのスタートの位置〝DATA〟から、737バイト長だけ他のメモリ領域（スタートの位置〝BUFFER〟）へデータ群を転送する場合、次のようにするとよい。

```
LD      HL, DATA        ; START ADRS OF SOURCE
LD      DE, BUFFER      ; START ADRS OF TARGET
LD      BC, 737         ; LENGTH OF DATA STRING
LDIR                    ; MOVE STRING
                        ; INC HL & DE
                        ; DEC BC UNTIL BC = 0
```

この操作では、11バイトを要しデータ１バイトの転送につき、21クロック・サイクルを要する。

B．メモリのスタートの位置〝DATA〟から、境界設定用としてのアスキー文字〝$〟が出現するまで、他のメモリ領域（スタートの位置〝BUFFER〟）へデータ群を転送する場合は、最大132文字までとする。

```
        LD      HL, DATA        ; START ADRS OF SOURCE
        LD      DE, BUFFER      ; START ADRS OF TARGET
        LD      BC, 132         ; MAX STRING LENGTH
        LD      A, '$'          ; DELIMITER CODE
LOOP :  CP      (HL)            ; COMPARE
        JR      Z, END          ; GO TO END IF EQU  AL
        LDI                     ; MOVE, INC HL, DE & DEC BC
        JP      PE, LOOP        ; GO TO "LOOP"
END :                           ; NOTE : P/V FLAG IS USED
                                ; TO INDICATE THAT REGISTER
                                ; BC WAS DECREMENTED TO ZERO
```

この操作では19バイトを要する。

C. パック・BCD フォーマット（2BCD デジット/バイト）で表現された10桁の10進数をシフトし、機械的なBCD 乗算、減算を行う場合の例を図10－1に示す。

```
        LD      HL, DATA        ; ADRS OF 1 ST BYTE
        LD      B, COUNT        ; SHIFT COUNT
        XOR     A               ; CLEAR A
ROTAT : RLD                     ; ROTATE LEFT LOW ORDER IN A
        INC     HL              ; INC POINTER
        DJNZ    ROTAT           ; DEC B & GO TO "ROTAT" IF B ≠ 0
                                ; FALL THROUGH IF B = 0
```

この操作では11バイトを要する。

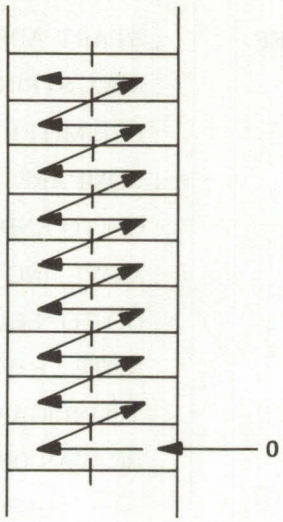


図10－1

D．ある数から他の数を引く場合で、

a）両方ともパックBCDフォーマットであり、

b）両方同じように長さが変化し、

c）結果を被減数のある位置にストアする場合、

次のようになる。

```
        LD      HL, ARG 1       ; ADRS OF MINUEND
        LD      DE, ARG 2       ; ADRS OF SUBTRAHEND
        LD      B, LENGTH       ;
        AND     A               ; CLEAR CY
SUBDEC: LD      A, (DE)         ; SUBTRAHEND TO A
        SBC     A, (HL)
        DAA                     ; DECIMAL ADJUST
        LD      (HL), A         ; STORE RESULT
        INC     HL              ; INC POINTER
        INC     DE              ; INC POINTER
        DJNZ    SUBDEC          ; DEC B & GO TO "SUBDEC" IF B ≠ 0 FALL THROUGH IF B = 0
```

この操作では17バイトを要する。

10. 3　プログラミング・タスクの例

A. 次のプログラムは、標準的な交換ソーティング・アルゴリズムにより上昇順に数の配列を 0 ～255 の範囲にソート（仕分け）するものである。

```
01/22/76      11:14:37                BUBBLE LISTING                    PAGE 1
LOC     OBJ CODE    STMT    SOURCE STATEMENT

                      1    ;  *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE ***
                      2    ;
                      3    ;  AT ENTRY:  HL CONTAINS ADDRESS OF DATA
                      4    ;             C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
                      5    ;             (1<C<256)
                      6    ;
                      7    ;  AT EXIT: DATA SORTED IN ASCENDING ORDER
                      8    ;
                      9    ;  USE OF REGISTERS
                     10    ;
                     11    ;  REGISTER   CONTENTS
                     12    ;
                     13    ;  A          TEMPORARY STORAGE FOR CALCULATIONS
                     14    ;  B          COUNTER FOR DATA ARRAY
                     15    ;  C          LENGTH OF DATA ARRAY
                     16    ;  D          FIRST ELEMENT IN COMPARISON
                     17    ;  E          SECOND ELEMENT IN COMPARISON
                     18    ;  H          FLAG TO INDICATE EXCHANGE
                     19    ;  L          UNUSED
                     20    ;  IX         POINTER INTO DATA ARRAY
                     21    ;  IY         UNUSED
                     22    ;
0000    222600       23    SORT:  LD     (DATA), HL      ; SAVE DATA ADDRESS
0003    CB84         24    LOOP:  RES    FLAG, H         ; INITIALIZE EXCHANGE FLAG
0005    41           25           LD     B, C            ; INITIALIZE LENGTH COUNTER
0006    05           26           DEC    B               ; ADJUST FOR TESTING
0007    DD2A2600     27           LD     IX, (DATA)      ; INITIALIZE ARRAY POINTER
000B    DD7E00       28    NEXT:  LD     A, (IX)         ; FIRST ELEMENT IN COMPARISON
000E    57           29           LD     D, A            ; TEMPORARY STORAGE FOR ELEMENT
000F    DD5E01       30           LD     E, (IX+1)       ; SECOND ELEMENT IN COMPARISON
0012    93           31           SUB    E               ; COMPARISON FIRST TO SECOND
0013    3008         32           JR     NC, NOEX-$      ; IF FIRST > SECOND, NO JUMP
0015    DD7300       33           LD     (IX), E         ; EXCHANGE ARRAY ELEMENTS
0018    DD7201       34           LD     (IX+1), D
001B    CBC4         35           SET    FLAG, H         ; RECORD EXCHANGE OCCURRED
001D    DD23         36    NOEX:  INC    IX              ; POINT TO NEXT DATA ELEMENT
001F    10EA         37           DJNZ   NEXT-$          ; COUNT NUMBER OF COMPARISONS
                     38                                  ; REPEAT IF MORE DATA PAIRS
0021    CB44         39           BIT    FLAG, H         ; DETERMINE IF EXCHANGE OCCURRED
0023    20DE         40           JR     NZ, LOOP-$      ; CONTINUE IF DATA UNSORTED
0025    C9           41           RET                    ; OTHERWISE, EXIT
                     42    ;
0026                 43    FLAG:  EQU    0               ; DESIGNATION OF FLAG BIT
0026                 44    DATA:  DEFS   2               ; STORAGE FOR DATA ADDRESS
                     45           END
```

B．以下に２つの符合なし16ビット整数の乗算をし、その結果をＨＬレジスタ・ペアにおくプログラムを示す。

```
01/22/76      11:32:36                MULTIPLY LISTING                    PAGE 1
LOC    OBJ CODE    STMT    SOURCE STATEMENT

0000                 1    MULT:;    UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
                     2    ;         ON ENTRANCE:  MULTIPLIER IN DE.
                     3    ;                       MULTIPLICAND IN HL.
                     4    ;
                     5    ;         ON EXIT:  RESULT IN HL.
                     6    ;
                     7    ;         REGISTER USES:
                     8    ;
                     9    ;
                    10    ;         H       HIGH ORDER PARTIAL RESULT
                    11    ;         L       LOW ORDER PARTIAL RESULT
                    12    ;         D       HIGH ORDER MULTIPLICAND
                    13    ;         E       LOW ORDER MULTIPLICAND
                    14    ;         B       COUNTER FOR NUMBER OF SHIFTS
                    15    ;         C       HIGH ORDER BITS OF MULTIPLIER
                    16    ;         A       LOW ORDER BITS OF MULTIPLIER
                    17    ;
0000   0610        18              LD      B, 16;          NUMBER OF BITS- INITIALIZE
0002   4A          19              LD      C, D;           MOVE MULTIPLIER
0003   7B          20              LD      A, E;
0004   EB          21              EX      DE, HL;         MOVE MULTIPLICAND
0005   210000      22              LD      HL, 0;          CLEAR PARTIAL RESULT
0008   CB39        23    MLOOP:    SRL     C;              SHIFT MULTIPLIER RIGHT
000A   1F          24              RRA;                    LEAST SIGNIFICANT BIT IS
                    25    ;                                 IN CARRY.
000B   3001        26              JR      NC, NOADD-$;    IF NO CARRY, SKIP THE ADD.
000D   19          27              ADD     HL, DE;         ELSE ADD MULTIPLICAND TO
                    28    ;                                 PARTIAL RESULT.
000E   EB          29    NOADD:    EX      DE, HL;         SHIFT MULTIPLICAND LEFT
000F   29          30              ADD     HL, HL;         BY MULTIPLYING IT BY TWO.
0010   EB          31              EX      DE, HL;
0011   10F5        32              DJNZ    MLOOP-$;        REPEAT UNTIL NO MORE BITS.
0013   C9          33              RET;
                    34              END;
```

第11章　規　格

絶対最大定格

項　目	記　号	定　格　値	単　位
入力電圧	V_{IN}	−0.3〜+7.0	V
出力電圧	V_{OUT}	−0.3〜+7.0	V
動作温度	T_{OPR}	0〜+70	℃
保存温度	Tstg	−65〜+150	℃

絶対最大定格の内のどの1項目でも、瞬時たりとも、絶対最大定格を越えないようにしてください。かつ、2項目以上の値が絶対最大定格値に同時に達しないようにしてください。

DC特性

(T_a=0℃〜+70℃, V_{CC}=+5V±5%)

項　目		記　号	測定条件	最小値	最大値	単位
クロック入力"Low"電圧		V_{ILC}		−0.3	0.45	V
クロック入力"High"電圧		V_{IHC}		V_{CC}−0.6	V_{CC}+0.3	V
入力"Low"電圧		V_{IL}		−0.3	0.8	V
入力"High"電圧		V_{IH}		2.0	V_{CC}	V
出力"Low"電圧		V_{OL}	I_{OL}=1.8mA		0.4	V
出力"High"電圧		V_{OC}	I_{OL}=−250μA	2.4		V
消費電流	Z-80	I_{CC}			150	mA
	Z-80A				200	mA
	Z-80B				200	mA
入力リーク電流		I_{LI}	V_{IN}=0〜V_{CC}		10	μA
3ステート出力リーク電流		I_{LEAK}	V_{OUT}=0.4〜V_{CC}	−10	10	μA

(注1) A_{15}〜A_0, D_7〜D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$

端子容量

(T_a=25℃、f=1 MHz)

項　目	記　号	備　考	最小値	最大値	単位
クロック容量	C_{CLOCK}	被測定端子以外のすべての端子は接地する。		35	pF
入力容量	C_{IN}			5	pF
出力容量	C_{OUT}			10	pF

AC特性

番号	記号	項目	Z-80 CPU		Z-80A CPU		Z-80B CPU*		単位
			最小値	最大値	最小値	最大値	最小値	最大値	
1	TcC	クロック・サイクル時間	400*		250*		165*		ns
2	TwCh	クロック・パルス幅(High)	180*		110*		65*		ns
3	TwCl	クロック・パルス幅(Low)	180	2000	110	2000	65	2000	ns
4	TfC	クロック ↓時間		30		30		20	ns
5	TrC	クロック ↑時間		30		30		20	ns
6	TdCr(A)	クロック↑からアドレス確定までの遅延時間		145		110		90	ns
7	TdA(MREQf)	アドレス確定から$\overline{MREQ}$↓までの遅延時間	125*		65*		35*		ns
8	TdCf(MREQf)	クロック↓から$\overline{MREQ}$↓までの遅延時間		100		85		70	ns
9	TdCr(MREQr)	クロック↑から$\overline{MREQ}$↑までの遅延時間		100		85		70	ns
10	TwMREQh	$\overline{MREQ}$パルス幅 (High)	170*		110*		65*		ns
11	TwMREQl	$\overline{MREQ}$パルス幅 (Low)	360*		220*		135*		ns
12	TdCf(MREQr)	クロック↓から$\overline{MREQ}$↑までの遅延時間		100		85		70	ns
13	TdCf(RDf)	クロック↓から$\overline{RD}$↓までの遅延時間		130		95		80	ns
14	TdCr(RDr)	クロック↑から$\overline{RD}$↑までの遅延時間		100		85		70	ns
15	TsD(Cr)	クロック↑に先立つデータのセットアップ時間	50		35		30		ns
16	ThD(RDr)	$\overline{RD}$↑後のデータ保持時間	0		0		0		ns
17	TsWAIT(Cf)	$\overline{WAIT}$からクロック↓までのセットアップ時間	70		70		60		ns
18	ThWAIT(Cf)	クロック↓後の$\overline{WAIT}$保持時間	0		0		0		ns
19	TdCr(Mlf)	クロック↑から$\overline{M1}$↓までの遅延時間		130		100		80	ns
20	TdCr(Mlr)	クロック↑から$\overline{M1}$↑までの遅延時間		130		100		80	ns
21	TdCr(RFSHf)	クロック↑から$\overline{RFSH}$↓までの遅延時間		180		130		110	ns
22	TdCr(RFSHr)	クロック↑から$\overline{RFSH}$↑までの遅延時間		150		120		100	ns
23	TdCf(RDr)	クロック↓から$\overline{RD}$↑までの遅延時間		110		85		70	ns
24	TdCr(RDf)	クロック↑から$\overline{RD}$↓までの遅延時間		100		85		70	ns
25	TsD(Cf)	M_2, M_3, M_4またはM_5サイクルの間におけるクロック↓に先立つデータのセットアップ時間	60		50		40		ns

↑は立ち上がりエッジ、↓は立ち下がりエッジを示します。

番号	記　号	項　目	Z-80 CPU		Z-80A CPU		Z-80B CPU*		単位
			最小値	最大値	最小値	最大値	最小値	最大値	
26	TdA(IORQf)	$\overline{IORQ}$↓に先立つアドレス確定時間	320*		180*		110*		ns
27	TdCr(IORQf)	クロック↑から$\overline{IORQ}$↓までの遅延時間		90		75		65	ns
28	TdCf(IORQr)	クロック↓から$\overline{IORQ}$↑までの遅延時間		110		85		70	ns
29	TdDm(WRf)	$\overline{WR}$↓に先立つデータ確定時間（メモリ・サイクル）	190*		80*		25*		ns
30	TdCf(WRf)	クロック↓から$\overline{WR}$↓までの遅延時間		90		80		70	ns
31	TwWR	$\overline{WR}$パルス幅	360*		220*		135*		ns
32	TdCf(WRr)	クロック↓から$\overline{WR}$↑までの遅延時間		100		80		70	ns
33	TdDi(WRf)	$\overline{WR}$↓に先立つデータ確定時間（入出力サイクル）	20*		−10*		−55*		ns
34	TdCr(WRf)	クロック↑から$\overline{WR}$↓までの遅延時間		80		65		60	ns
35	TdWRr(D)	$\overline{WR}$↑からのデータ確定時間	120*		60*		30*		ns
36	TdCf(HALT)	クロック↓から$\overline{HALT}$↑または↓までの時間		300		300		260	ns
37	TwNMI	$\overline{NMI}$パルス幅	80		80		70		ns
38	TsBUSRQ(Cr)	クロック↑に先立つ$\overline{BUSRQ}$のセットアップ時間	80		50		50		ns
39	ThBUSRQ(Cr)	クロック↑後の$\overline{BUSRQ}$保持時間	0		0		0		ns
40	TdCr(BUSAKf)	クロック↑から$\overline{BUSAK}$↓までの遅延時間		120		100		90	ns
41	TdCf(BUSAKr)	クロック↓から$\overline{BUSAK}$↑までの遅延時間		110		100		90	ns
42	TdCr(Dz)	クロック↑からデータ・フロートまでの遅延時間		90		90		80	ns
43	TdCr(CTz)	クロック↑から$\overline{MREQ}$，$\overline{IORQ}$，$\overline{RD}$，$\overline{WR}$フロートまでの遅延時間		110		80		70	ns
44	TdCr(Az)	クロック↑からアドレス・フロートまでの遅延時間		110		90		80	ns
45	TdCTr(A)	$\overline{MREQ}$↑，$\overline{IORQ}$↑，$\overline{RD}$↑，$\overline{WR}$↑に対するアドレス保持時間	160*		80*		35*		ns
46	TsRESET(Cr)	クロック↑に先立つ$\overline{RESET}$セットアップ時間	90		60		60		ns
47	ThRESET(Cr)	クロック↑後の$\overline{RESET}$保持時間	0		0		0		ns
48	TsINTf(Cr)	クロック↑に先立つ$\overline{INT}$セットアップ時間	80		80		70		ns
49	ThINTr(Cr)	クロック↑後の$\overline{INT}$保持時間	0		0		0		ns
50	TdM1f(IORQf)	$\overline{M1}$↓から$\overline{IORQ}$↓までの遅延時間	920*		565*		365*		ns

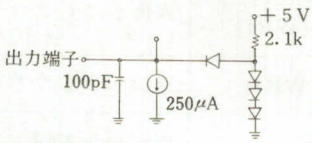
↑は立ち上がりエッジ、↓は立ち下がりエッジを示します。

番号	記　号	項　目	Z-80 CPU		Z-80A CPU		Z-80B CPU*		単位
			最小値	最大値	最小値	最大値	最小値	最大値	
51	TdCf(IORQf)	クロック↓から$\overline{IORQ}$↓までの遅延時間		110		85		70	ns
52	TdCf(IORQr)	クロック↑から$\overline{IORQ}$↑までの遅延時間		100		85		70	ns
53	TdCf(D)	クロック↓からデータ確定までの遅延時間		230		150		130	ns

↑は立ち上がりエッジ、↓は立ち下がりエッジを示します。

※　表に示されている最小値以外のクロック時間に関しては、下表の式を用いて数値を計算してください。

※　すべてのタイミングは暫定であり変更することがあります。



※　すべての負荷容量は最大100pFとします。負荷容量が50pF増加するごとに遅延時間は10μs増加します。負荷容量の最大値は、データーバスが200pFで他は100pFです。

AC特性の脚注

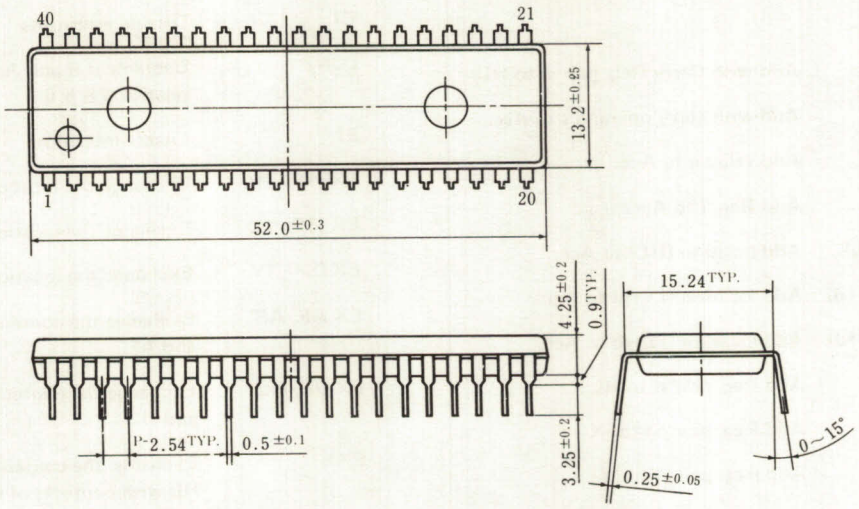
番号	記　号	Z-80	Z-80A	Z-80B
1	TcC	TwCh+TwCl+TrC+TfC	TwCh+TwCl+TrC+TfC	TwCh+TwCl+TrC+TfC
2	TwCh	MAX. 200μs	MAX. 200μs	MAX. 200μs
7	TdA(MREQf)	TwCh+TfC−75	TwCh+TfC−65	TwCh+TfC−50
10	TwMREQh	TwCh+TfC−30	TwCh+TfC−20	TwCh+TfC−20
11	TwMREQl	TcC−40	TcC−30	TcC−30
26	TdA(IORQf)	TcC−80	TcC−70	TcC−55
29	TdD(WRf)	TcC−210	TcC−170	TcC−140
31	TwWR	TcC−40	TcC−30	TcC−30
33	TdD(WRf)	TwCl+TrC−180	TwCl+TrC−140	TwCl+TrC−140
35	TdWRr(D)	TwCl+TrC−80	TwCl+TrC−70	TwCl+TrC−55
45	TdCTr(A)	TwCl+TrC−40	TwCl+TrC−50	TwCl+TrC−50
50	TdM1f(IORQf)	2TcC+TwCh+TfC−80	2TcC+TwCh+TfC−65	2TcC+TwCh+TfC−50

ACテスト条件

V_{IH}=2.0V　V_{IHC}=V_{CC}−0.6V　V_{OH}=2.0V　FLOAT=±0.5V

V_{IL}=0.8V　V_{ILC}=0.45V　V_{OL}=0.8V

外形寸法図



単位：mm

（おことわり）改良のため予告なしに仕様の一部を変更することがあります。

付　録

Z－80 CPU 命令セット

ADC HL, ss	Add with Carry Reg. pair ss to HL
ADC A, s	Add with carry operand s to Acc.
ADD A, n	Add value n to Acc.
ADD A, r	Add Reg. r to Acc.
ADD A, (HL)	Add location (HL) to Acc.
ADD A, (IX+d)	Add location (IX+d) to Acc.
ADD A, (IY+d)	Add location (IY+d) to Acc.
ADD HL, ss	Add Reg. pair ss to HL
ADD IX, pp	Add Reg. pair pp to IX
ADD IY, rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b, (HL)	Test BIT b of location (HL)
BIT b, (IX+d)	Test BIT b of location (IX+d)
BIT b, (IY+d)	Test BIT b of location (IY+d)
BIT b, r	Test BIT b of Reg. r
CALL cc, nn	Call subroutine at location nn if condition cc if true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B ≠ 0
EI	Enable interrupts
EX (SP), HL	Exchange the location (SP) and HL
EX (SP), IX	Exchange the location (SP) and IX
EX (SP), IY	Exchange the location (SP) and IY
EX AF, AF'	Exchange the contents of AF and AF'
EX DE, HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A, (n)	Load the Acc. with input from device n
IN r, (C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B

INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditonal Jump to (IY)
JP cc, nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JP C, e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JP NC, e	Jump relative to PC+e if carry=0
JR NZ, e	Jump relative to PC+e if non zero (Z=0)
JR Z, e	Jump relative to PC+e if zero (Z=1)
LD A, (BC)	Load Acc. with location (BC)
LD A, (DE)	Load Acc. with location (DE)
LD A, I	Load Acc. with I
LD A, (nn)	Load Acc. with location nn
LD A, R	Load Acc. with Reg. R
LD (BC), A	Load location (BC) with Acc.
LD (DE), A	Load location (DE) with Acc.
LD (HL), n	Load location (HL) with value n
LD dd, nn	Load Reg. pair dd with value nn
LD HL, (nn)	Load HL with location (nn)
LD (HL), r	Load location (HL) with Reg. r
LD I, A	Load I with Acc.
LF IX, nn	Load IX with value nn
LD IX, (nn)	Load IX with location (nn)
LD (IX+d), n	Load location (IX+d) with value n
LD (IX+d), r	Load location (IX+d) with Reg. r
LD IY, nn	Load IY with value nn
LD IY, (nn)	Load IY with location (nn)
LD (IY+d), n	Load location (IY+d) with value n
LD (IY+d), r	Load location (IY+d) with Reg. r
LD (nn), A	Load location (nn) with Acc.
LD (nn), dd	Load location (nn) with Reg. pair dd
LD (nn), HL	Load location (nn) with HL
LD (nn), IX	Load location (nn) with IX
LD (nn), IY	Load location (nn) with IY
LD R, A	Load R with Acc.
LD r, (HL)	Load Reg. r with location (HL)
LD r, (IX+d)	Load Reg. r with location (IX+d)
LD r, (IY+d)	Load Reg. r with location (IY+d)
LD r, n	Load Reg. r with value n
LD r, r'	Load Reg. r with Reg. r'
LD SP, HL	Load SP with HL
LD SP, IX	Load SP with IX
LD SP, IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' or operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C), r	Load output port (C) with Reg. r
OUT (n), A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B

POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b, m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) ieft circular
RLC (IX+d)	Rotate location (IX+d) left circular
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A, s	Subtract operand s from Acc. with carry
SBC HL, ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b, (HL)	Set Bit b of location (HL)
SET b, (IX+d)	Set Bit b of location (IX+d)
SET b, (IY+d)	Set Bit b of location (IY+d)
SET b, r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

第2部

Z-80 PIO / Z-80A PIO / Z-80B PIO

テクニカルマニュアル

目　　次

第1章　緒　論

Z－80 PIO (Parallel Input/Output Interface Controller) はZ－80 CPU と周辺装置とのインターフェース用のデバイスで、種々の動作モードをプログラムで指示することができる。また本デバイスは TTL コンパチブルの入出力ポートを2個もっている。

Z－80 PIO はこれを CPU のインターフェース・デバイスとして用いると、余分な外部論理回路を追加しないで広範囲の周辺装置に適用することができる。

Z－80 PIO と接続可能な周辺装置は、たとえば、キーボード、紙テープ・リーダ、紙テープ・パンチャ、プリンタなどがある。

Z－80 PIO はNチャネル・シリコンゲート・デプリーション・ロード技術で製造され、40ピン DIP パッケージである。

Z－80 PIO の主な特長は、

- 2つの独立した8ビット双方向性の周辺装置用インターフェース・ポートがあり、それぞれにデータ転送の制御用の〝ハンドシェーク〟線がある。
- 割り込みは高速応答に適した〝ハンドシェーク〟で起動される。
- 各ポートごとに、次の動作モードの選択ができる。

  バイト出力

  バイト入力

  バイト単位の双方向バス（ポートAのみ）

  ビット制御モード

  これらにはすべてハンドシェークで制御される割り込み機能がある。
- 外部論理回路なしで自動的に割り込みベクトルを出せるデージー・チェーン構造の割り込み優先順位回路を内蔵している。
- 8出力はダーリントン・トランジスタを駆動できる。
- 全入出力は TTL コンパチブルである。
- 単一5Ｖ電源、単相クロックのみで動作する。

Z－80 PIO は他のインターフェース・コントローラと異なり、周辺装置と CPU との間のデータ転送を割り込み制御の方法により効率よく行うことができる。またZ－80 PIO には、Z－80 CPU の効率の良い割り込み処理能力を入出力データの転送期間に充分に発揮できるように、割り込み処理用の論理回路が組み込まれており、かつ多重の割り込みをネストさせるために必要な論理回路も内蔵している。このため割り込み処理用の余分な外部回路を必要としない。

その他、周辺装置が特定の状態になったときに、割り込み信号が出るようにプログラムしておくこともできるようになっているのも特記すべき点である。たとえば、プログラミングによって、ある周辺装置に特定の状態となる条件が発生したときに割り込みを出すことができる。

この割り込みの能力によってプロセッサが周辺装置のステータスをポーリングする（調べる）間の時間を節約するのに役立たせることができる。

なお、Z－80A PIOは周波数の上限を4 MHzまで、Z－80B PIOは周波数の上限を6 MHzまで保証した高速タイプであり、第1章から第8章まではとくにZ－80A PIOあるいはZ－80B PIOと記述していないがZ－80PIOと同様に適用される。電気的仕様など規格については個々に定めている。第9章を参照されたい。

第2章　Z-80 PIO アーキテクチャ

図2－1にZ－80 PIOのブロック図を示す。

Z－80 PIOの内部は次の各部で構成されている。

- Z－80 CPU用バス・インターフェース
- 内部制御回路
- ポートA入出力論理回路
- ポートB入出力論理回路
- 割り込み制御回路

CPUバス・インターフェース用論理回路は他の外部回路を追加せずに直接CPUに接続できるようになっているが、大きなシステムでは必要に応じてアドレス・デコーダやライン・バッファを付加しなければならないこともある。

内部制御回路はCPUデータ・バスを周辺装置用インターフェース部（ポートA、B）と同期させるためのものである。2つの入出力ポート（A、B）は、ほぼ同様に使用でき、直接周辺装置と結合する部分である。

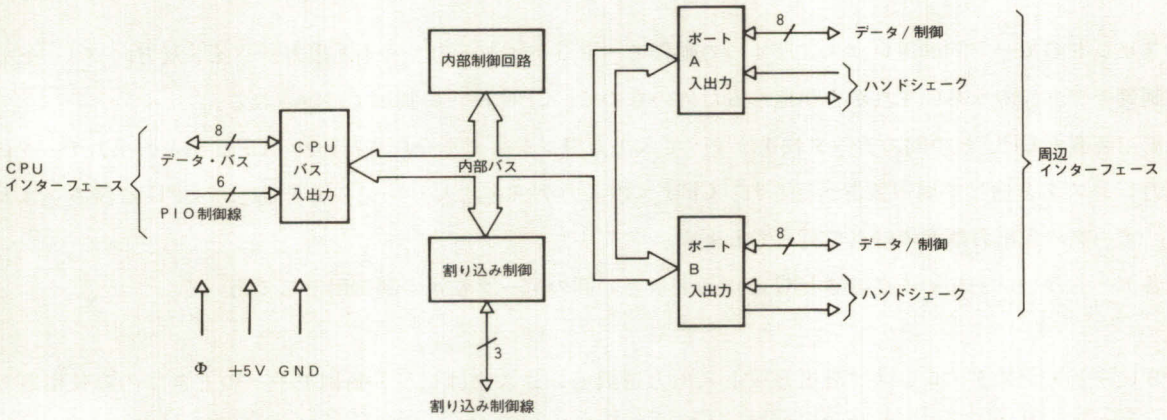


図2－1　Z-80 PIOブロック図

ポート入出力論理回路は図２－２に示すように、〝ハンドシェーク〟用論理回路を含む６つのレジスタから構成されている。

これらはそれぞれ８ビット・データ入力レジスタ、８ビット・データ出力レジスタ、２ビットのモード制御用レジスタ、８ビットのマスク・レジスタ、８ビット入出力選択レジスタ、２ビットのマスク制御用レジスタである。

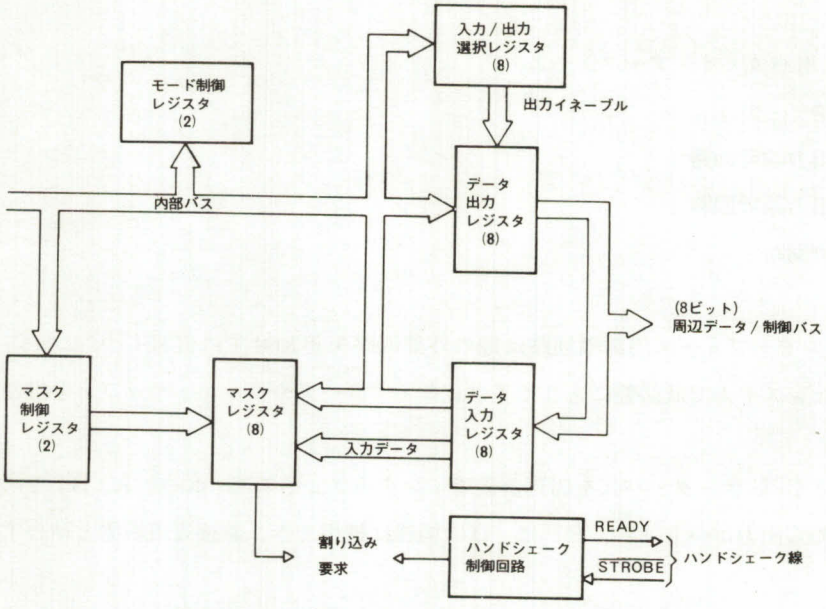


図２－２　ポート入出力ブロック図

２ビットのモード制御用レジスタは４つの動作モード（バイト入力、バイト出力、バイト双方向バス、ビット制御モード）のうちいずれかを指定するためのもので、CPU からの制御で設定される。

周辺装置と CPU との間のデータ転送はすべて入出力用レジスタを介して行われ、CPU からの出力データは出力レジスタを通して周辺装置へ送られ、CPU への入力データは入力レジスタを通して CPU に読み込まれる。データの入出力はどの時点で行ってもよい。

各ポートのハンドシェーク線は PIO と周辺装置との間のデータ転送の制御用として用いる。

８ビット・マスク・レジスタと８ビット入出力選択用レジスタはビット制御モードのときにのみ使用される。このモードでは、周辺と結ばれるデータ転送、もしくは制御用の８本のバス端子を個々に入力用または出力用に指定できる。これはプログラムで選択レジスタを設定することにより行う。

このモードでは、特定の割り込み発生条件を設定しておくためにマスク・レジスタを使用する。このマスク・レジスタは入出力用バス端子のどれをモニタするかを決めるために使用し、またどのようにモニタするかは次のようにマスク制御レジスタを使用して行う。

2ビットのマスク制御レジスタは、モニタすべきポート端子の入力すべてについて、AND条件をとるか、OR条件をとるか、あるいは入力が〝High〟レベルのときアクティブとするか、〝Low〟レベルのときアクティブとするかを規定するためにある。これらの指定に合ったステータスになれば、割り込みが発生する。

このように、あらかじめ条件設定をプログラミングしておく方式では、周辺装置の特定のステータスで自動的に割り込みが発生させられるので、CPUが周辺のステータス・チェックをする必要がないという利点がある。システム内に3種の緊急条件がある場合、プログラムでこのうちの1種でも起これば(OR条件)、あるいは3種すべてが起これば(AND条件)割り込みを発生させる、といった使い分けもできる。

割り込み制御用論理回路はCPUの割り込みに関するすべての処理を行い、優先順位の決定などについても関与している部分である。デージー・チェーンの接続によって、各デバイスはその物理的な位置によって優先順位が決まり、CPUに近いデバイスほど優先順位が高くなる。このデージー・チェーンを形成するために、各PIOから2本の線(IEI、IEO)が出ている。また、同一PIO内ではポートAがポートBより優先順位が高い。

バイト入力モード、バイト出力モード、バイト双方向バス・モードでは、周辺装置からの次のデータ・バイト転送の要求のあるたびごとに割り込みが発生する。

ビット制御モードでは、周辺のステータスがプログラムした値と一致したとき割り込みが発生する。

PIOはネスト構造の割り込みに関して完全な制御機能をもっていて、CPUが上位の優先順位をもつデバイスの割り込みサービスを行っているときには、下位のデバイスからのサービス要求は通さない(下位の優先順位のデバイスから、割り込みを発生させない)ようになっている。

しかし、上位の優先順位のデバイスは、サービス中のデバイスが下位のものであれば、割り込みをかけることができるようになっている。

モード2で、割り込みをかける場合、割り込みをかけるデバイスは8ビットの割り込みベクトルをCPUに送る必要がある。

このベクトルはメモリの位置を指定するポインタの下位データとして使用され、指定された位置にはその割り込みデバイスに対するサービス・ルーチンのスタート・アドレス(下位バイト)を置いておけばよい。指定された位置以降の2バイトによりサービス・ルーチンのスタート・アドレスを与える。

デバイスからの8ビットのベクトルはポインタ(間接)の下位8ビットとなり、CPU内のIレジスタのデータが、そのポインタの上位8ビットとなる。

なお、PIOの各ポート(AとB)の割り込みのベクトルはプログラムによって指定するものであることに注意を要する。

ベクトルの最下位ビットはスタート・アドレスを連続する2バイト(16ビット)で指定しなければならないので、プログラマ側で0を指定しなければならない。

PIO 自身でCPU データ・バスに乗る RETI（割り込みからのリターン）命令を直接解続できるので、システム内の各PIO は CPU との間に別の通信線を引かなくても、CPU の割り込みサービスが終了する時期を検知できるようになっている。

第3章 Z-80 PIO 端子説明

図3－1にZ－80 PIOの端子配置図を示す。以下この章では、各端子の機能を説明する。

$D_0 \sim D_7$　Z－80 CPUのデータ・バス（双方向性、3ステート）

このバスはCPU－PIO間ですべてのデータや命令などを転送するために使用される。

D_0はバスの最下位ビットである。

B/$\overline{A}$ SEL　ポートB、Aの選択（入力、アクティブ〝High〟）

この端子で、CPU－PIO間でデータ転送を行う際に、アクセスされるポートを指定する。この端子への入力が〝Low〟レベルのときポートAを選択し、〝High〟レベルのときポートBを選択する。

通常CPUからのアドレス・ビットA_1をポート選択用として使用する。

C/$\overline{D}$ SEL　制御信号、データの選択（入力、アクティブ〝High〟）

この端子で、CPU－PIO間で行われるデータ転送の型を指定する。この端子への入力が〝High〟レベルのとき、PIOへの書き込みデータはB/A選択で指定されたポートへのコマンド（命令）として解釈され、この端子への入力が〝Low〟レベルのときには通常のデータ転送となる。

通常CPUからのアドレス・ビットA_0を制御信号、データ選択用として使用する。

$\overline{CE}$　チップ・イネーブル（入力、アクティブ〝Low〟）

この端子を〝Low〟レベルにすると、PIOは書き込みサイクルの期間ではCPUからコマンドまたはデータを受けとることができ、読み出しサイクルの期間ではCPUへデータを送出することができる。この信号として、通常ポートA、ポートBの各データおよびコマンドのいずれかを指定できる入出力ポート・アドレスのデコード信号を使用する。

Φ　システム・クロック（入力）

Z－80 PIO内部の同期用として、Z－80の標準システム・クロックを利用している。この信号は単相クロックである。

$\overline{M1}$　CPUのマシン・サイクル1信号（入力、アクティブ〝Low〟）

CPUからのこの信号はPIO内で制御用同期パルスとして使用される。

Z－80 CPU は、$\overline{M1}$ がアクティブでかつ $\overline{RD}$ がアクティブのときには、メモリからの命令をフェッチするが、一方 $\overline{M1}$ がアクティブで $\overline{IORQ}$ がアクティブのときには、$\overline{INT}$ 入力に対する割り込みアクノリッジを出力していることを示している。

さらに PIO は $\overline{M1}$ で次のような動作をする。

1．$\overline{M1}$ に PIO の割り込み論理回路を同期させる。

2．$\overline{RD}$、$\overline{IORQ}$ 信号がともにアクティブでないとき、$\overline{M1}$ が 2 クロック以上あれば、PIO 内の論理回路はリセット状態になる。

$\overline{IORQ}$　CPU の入力/出力要求（入力、アクティブ〝Low〟）

$\overline{IORQ}$ 信号は、CPU－PIO 間でコマンドやデータの転送を行う場合に、B/A 選択、C/D 選択、$\overline{CE}$、$\overline{RD}$ 信号などと組み合わせて使用される。

$\overline{CE}$、$\overline{RD}$、$\overline{IORQ}$ がともにアクティブのとき、B/A で選択されたポートから CPU へのデータ転送が行われる（読み込み動作）。

逆に、$\overline{CE}$、$\overline{IORQ}$ がともにアクティブであり、$\overline{RD}$ が非アクティブのとき、B/A で選択されたポートへ、CPU から C/D 選択信号で指定された情報（データまたはコマンド）が送られる。また、CPU が割り込み受け付け状態になれば、$\overline{M1}$ と $\overline{IORQ}$ がともにアクティブとなり、割り込みを出しているポート（優先順位が最上位で、かつ割り込み要求を出しているとする）から自動的に CPU データ・バス上へポートの割り込みベクトルが送り込まれる。

$\overline{RD}$　CPU の読み出しサイクル・ステータス（入力、アクティブ〝Low〟）

$\overline{RD}$ がアクティブのとき、メモリの読み出し、あるいは入出力デバイスの読み出しが行われる。

$\overline{RD}$ 信号は、B/A 選択、C/D 選択、$\overline{CE}$、$\overline{IORQ}$ 信号と組み合わせて、PIO から CPU へのデータ転送を行わせるのに使用される。

IEI　割り込みイネーブル入力（入力、アクティブ〝High〟）

割り込みを発生するデバイスが 2 個以上ある場合、それらの割り込みの優先順位を決めるデージー・チェーンを形成するために用いられる。この PIO より優先順位の高いデバイスからの IEO 信号が〝High〟レベルであれば、条件が成立した場合に、このデバイス（PIO）から割り込みを発生することができる。（この PIO の直前のデバイスの IEO をこの PIO の IEI に接続しておく。）

IEO　　割り込みイネーブル出力（出力、アクティブ〝High〟）

このIEO信号もデージー・チェーンを形成するのに必要なものである。

IEIが〝High〟レベルで、CPUがこのPIOからの割り込みのサービスを行っていないときにだけIEOは〝High〟レベルとなる。このPIOの割り込みサービスが行われているときには、これより優先順位の低いデバイスのIEIがこの信号で〝Low〟レベルにおさえられ、そこからの割り込みは阻止される。（このIEOはこのPIOの直後のデバイスのIEIに接続しておく。）

$\overline{INT}$　　割り込み要求（出力、オープン・ドレイン、アクティブ〝Low〟）

PIOからのCPUに対する割り込み要求はこの信号をアクティブにして行う。この$\overline{INT}$（出力）はCPUの$\overline{INT}$（入力）に接続する

$A_0 \sim A_7$　　ポートAバス（双方向性、3ステート）

この8ビット・バスはZ－80 PIOのポートAと周辺装置との間でデータ、ステータス、制御情報の転送を行うもので、A_0はポートAデータ・バスの最下位ビットである。

$\overline{A\ STB}$　　周辺装置が与えるポートAストローブ・パルス（入力、アクティブ〝Low〟）

この信号の意味は、以下のように、指定されたモードごとに異なる。

1）出力モード：PIOから送り出されたデータを周辺装置が受け取ったことをPIOに対して通知する信号であり、この信号の立ち上がりが有効である。

2）入力モード：周辺装置からポートAの入力レジスタへデータをロードしたとき、その周辺装置から出されるストローブ信号として使用する。この信号がアクティブになったとき、データがPIOにロードされる。

3）双方向モード：この信号がアクティブのとき、ポートAの出力レジスタからのデータがポートAの双方向データ・バス上に乗せられる。
ストローブが立ち上がれば、周辺装置がデータを受け取ったとみなされる。

4）ビット制御モード：このときのストローブ入力は無効である。

A RDY　　レジスタAレディ（出力、アクティブ〝High〟）

この信号の意味は、以下のように、指定されたモードによって異なる。

1）出力モード：ポートAの出力レジスタがロードされていて周辺用データ・バスが安定になり、周辺装置へのデータ転送が準備できたことを指示するため、アクティブとなる。

2）入力モード：ポートAの入力レジスタが空となり、周辺装置からのデータ受け入れ準備ができたとき、アクティブとなる。

3）双方向モード：ポートAの出力レジスタに周辺装置へ送るべきデータがロードされたとき、アクティブとなる。

4）ビット制御モード：この信号は強制的に〝Low〟状態になる。

$B_0 \sim B_7$　ポートBバス（双方向性、3ステート）

この8ビット・バスはデータ、ステータス、制御語などの情報をPIOのポートBと周辺装置間で転送するために用いられる。ポートBのデータ・バスは、ダーリントン・トランジスタを駆動できるよう、1.5V当たり1.5mAの出力能力をもっている。

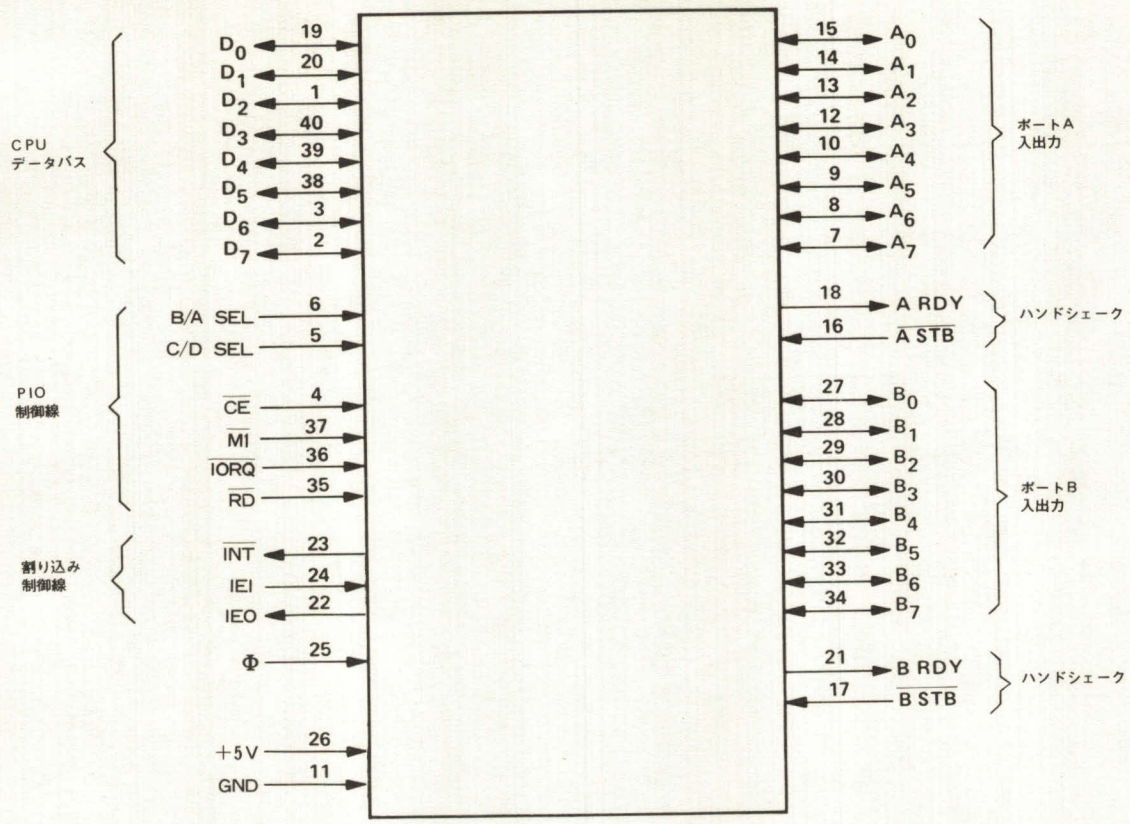
B_0 はバスの最下位ビットである。

$\overline{\text{B STB}}$　周辺装置が与えるポートBストローブ・パルス（入力、アクティブ〝Low〟）

この信号の意味は、$\overline{\text{A STB}}$ と同様であるが、次の点が異なる。ポートAの双方向モードでは、周辺装置からポートAの入力レジスタへのデータのストローブ（移し込み）をこの $\overline{\text{B STB}}$ で行う。（ポートAの出力レジスタ内容の出力ストローブは $\overline{\text{A STB}}$ を使用する。）

B RDY　レジスタBレディ（出力、アクティブ〝High〟）

この信号の意味は、A RDYと同様であるが、次の点が異なる。ポートAの双方向モードでは、ポートAの入力レジスタが空となり、データの受け入れ準備ができたとき、このB RDYがアクティブとなる。（ポートAの出力レジスタ内容の出力準備完了の指示はA RDYによる。）



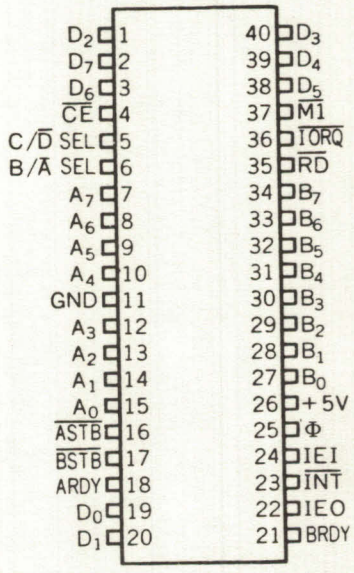


図3－1　Z-80 PIO端子配置図

第4章　Z-80 PIO　プログラミング

4．1　リセット

Z－80 PIOは、リセット状態では次のような機能をもっている。

1）両ポートのマスク・レジスタがリセットされ、全データ・ビットが禁止（インヒビット）となる。

2）ポートのデータ・バス線は高インピーダンス状態になり、ハンドシェークのレディ信号は非アクティブ（"Low"）となる。動作モードは自動的にモード1に設定される。

3）ベクトル・アドレス・レジスタはリセットされない。

4）両ポートの割り込みイネーブル・フリップ・フロップがリセットされる。

5）両ポートの出力レジスタもリセットされる。

PIOのリセットは、$\overline{RD}$、$\overline{IORQ}$信号をともに非アクティブにして2クロック以上の$\overline{M1}$を与えて行う。

2クロック後の$\overline{M1}$期間に$\overline{RD}$か$\overline{IORQ}$かが検出されなければ、PIOは$\overline{M1}$信号が非アクティブになった直後リセット状態になる。

このリセットの目的は、簡単な外部ゲートのみでリセットを行うことができるようにするためであり、40ピンのパッケージに収める上での処置である。

一度PIOが内部リセット状態に入ると、CPUから制御語を受けるまでそのままリセット状態を保持する。

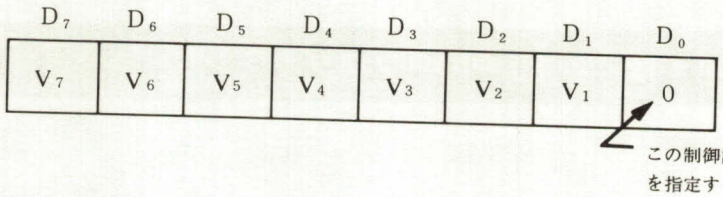
4．2　割り込みベクトルのローディング

PIOは、モード2の割り込み応答のとき、Z－80 CPU側に情報を返して、CPUがこれを処理・操作できるように設計されている。

このモードでは、割り込みを要求しているデバイスから割り込みベクトルを供給することになるが、このベクトルはCPU内でそのデバイスの割り込みサービス・ルーチンのアドレスを形成するのに使用される。

このベクトルは、その時点で最高位の優先順位をもっているデバイス（PIOなど）から割り込みアクノリッジ・サイクル期間中に、Z－80データ・バス上に乗せられる。(CPUの割り込みサービスの詳細については、第1部Z－80 CPUテクニカル・マニュアルを参照されたい。)

PIOが出す割り込みベクトルは次のようなフォーマットで、CPUから出力命令によって制御語を書き出し、希望するPIOにロードしておく。この場合のポート・アドレスは、AまたはBポートの制御語に対応するアドレスである。



ベクトルの書き込みにおいてはD_0は、フラグ・ビットとして使用され、このビットが0である場合のV_1からV_7までが、PIO内のベクトル・レジスタにロードされる。

割り込みアクノリッジ期間に割り込みをかけたポートのベクトルが、Z－80データ・バス上に、上のようなフォーマットで現れる。

4．3　動作モードの選択

PIOのポートAは次のような4種の異なったモードで使用できる。

モード0（出力モード）

モード1（入力モード）

モード2（双方向モード）

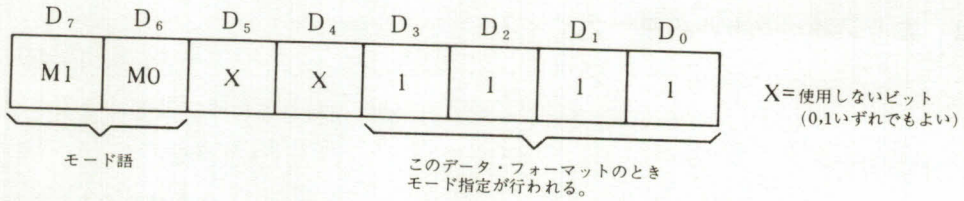
モード3（ビット制御モード）

数字は次のような関連で覚えるとよい。

0＝OUT，　1＝IN，　2＝BI-DIRECTIONAL

（出力）　（入力）　（双方向）

ポートBはモード2を除いたモード0、1、3で動作可能である。動作モードの指定はPIOに次のようなフォーマットの制御語を書き込んで行う。



ビットD_7、D_6により、以下のようにモードを指定できる。

D_7	D_6	モード
0	0	0（出力）
0	1	1（入力）
1	0	2（双方向）
1	1	3（制御）

ビット D_5、D_4 は無視される。（0、1いずれでもよい。）
ビット D_3 ～ D_0 により動作モードを指定し、1111に設定しておかねばならない。

モード0を指定した場合、ポートの出力レジスタ内のデータが対応するポートのデータ・バス上に乗せられる。この出力レジスタの内容は出力命令を用いて新しいデータにいつでも簡単に書き換えられるし、また、ある時点での出力レジスタの内容を入力命令を用いてCPU側へ呼び戻す（出力レジスタの内容は変化しない）こともできる。

モード0では、CPUから出力レジスタに対してデータの書き込みがあった場合、そのポートのハンドシェーク線であるレディ信号はアクティブ（〝High〟）となり、周辺装置にデータの転送が可能であるという指示を出す。この信号は周辺装置から受信したというストローブ信号が返されるまで、〝High〟レベルのままである。その返されたストローブの立ち上がりエッジで、もし割り込みが発生するようにプログラムされているならば、割り込みが発生し、レディ線は非アクティブ（〝Low〟）となる。この簡単なハンドシェーク方式は多くの周辺装置によく使用されているものと同じ方式である。

モード1を指定した場合、ポートは入力モードとなる。ハンドシェーク動作を始めるためには、まずそのポートに対して読み出しを行うだけでよい（ダミー・リード）。この動作によってレディ線がアクティブとなり、〝周辺装置へ入力レジスタが空になったのでデータをロードせよ〟という指示が出される。次に周辺装置はストローブ信号に同期してデータを送出するが、PIOはこのストローブ信号がアクティブになることによって内部入力レジスタにデータを取り込む。この返されたストローブの立ち上がりエッジで可能であれば割り込みが発生し、レディ信号が非アクティブ状態に落される（〝Low〟となる）。データがオーバーラン状態にならないように注意が行き届いていれば、レディ信号の状態にかかわらず、ストローブをかけてデータを入力レジスタへロードしてもよい。

モード2の双方向データ転送はすべてのハンドシェーク線（A、B各2本）を使用して行う。したがって、モード2では、ポートAだけをデータの入出力用として使用し、ハンドシェーク線はポートAの2本を出力制御用とし、ポートBの2本を入力制御用として使用する。A　RDY、B　RDYは両方とも、同時にアクティブとなる。モード0とモード2における出力時の動作の違いは、このモード2では双方向にデータ転送を行うのを可能にするため $\overline{\text{A　STB}}$ がアクティブになったときにだけポートAの出力レジスタ上のデータがポートAのデータ・バス上に乗せられる、という点のみである。

モード3の動作はステータスを調べるためと制御用として応用するためのものであり、この場合ハンドシェーク信号は使用しない。モード3を選択すると次にポート・アドレスに書き込む制御語によって、ポートのデータ・バス線のどれを入力用とし、どれを出力用とするかを指定する必要がある。あるビットを1に設定すれば、そのビットに対応するデータ・バス端子は入力用となり、0に設定したビットに対応するデータ・バ

ス端子は出力用となる。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

モード3の動作時、ストローブ信号は無視され、レディ線は〝Low〟レベルに保持される。ポートとCPU間のデータのやりとりは動作中任意の時点で行える。ポートを読む場合、CPUに入るデータは入力として指定されたポートのデータ・バス線からのデータと、出力として指定されたポートのデータ・バスからのデータが重なり合っている。

4．4　割り込み制御語の設定

各ポートへの割り込み制御語の設定は次のようなフォーマットで行う。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
割り込みイネーブル	AND/OR	High/Low	マスク指定	0	1	1	1

モード3のみで使用　　割り込み制御語を指定する。

ビット $D_7 = 1$ のとき、対応するポートの割り込みイネーブル・フリップ・フロップがセットされ、ポートからの割り込み要求が可能となる。$D_7 = 0$ のとき、イネーブル・フラグはリセットされ、割り込みを発生できない。イネーブル・フラグがセットされていて、割り込みが保留状態ならば、CPUの割り込み要求信号である $\overline{INT}$ 線は〝Low〟レベルのままである。ビット D_6、D_5、D_4 はモード3でのみ使用される。割り込み制御語のビット D_4 がセットされると、どのモードで動作している場合でも、未処理の割り込みはすべてリセットされる。モード3で、入出力線群のどれかがある指定状態になったとき、割り込み動作を実行できるようにするための条件設定にこの3ビットを用いる。ビット D_6 でモード3のデータ一致条件をANDで採るか、ORで採るかを定め、$D_6 = 1$ ならばAND、$D_6 = 0$ ならばORとなる。たとえば、ANDの場合、入力線の指定ビットがすべてアクティブのときにのみ、割り込み発生の条件となり、ORの場合、指定ビットのいずれかがアクティブならば、割り込み発生の条件となる。

ビット D_5 は入力線のアクティブ状態の極性規定するためのものである。$D_5 = 1$ ならば、入力線が〝High〟レベルのときアクティブとみなし、$D_5 = 0$ ならば、入力線が〝Low〟レベルのときアクティブとみなす。

ビット $D_4 = 1$ のとき、同一ポートに次に書き込まれた制御語はマスクと解釈される。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

このマスクでビット0の箇所の入力線のみを調べ、設定条件と一致すれば割り込みを発生させる条件となる。
なお、出力線はマスクで1に設定する。

第5章　Z-80 PIO タイミング

5．1　出力モード（モード0）

図5－1にモード0の動作のタイミングを示す。出力サイクルはつねにCPUの出力命令の実行でスタートする。

CPUから入出力への書き込み動作時にPIO内で$\overline{WR}$*パルスが発生し（これは外部には出ない）、この信号によってCPUデータ・バス上のデータを指定されたポート（AかB）の出力レジスタにとり込む。

$\overline{WR}$*パルスが立ち上がったあとの最初のΦの立ち下がりエッジで、周辺装置へのデータ供給が可能であるという指示を出すために、レディ・フラグがセットされる。たいていのシステムではこのレディ信号を周辺装置内のデータ・ラッチ信号として使用することができる。（$\overline{WR}$*は図参照）

レディ信号は次のいずれかの状態になるまでアクティブのままである。

(1)　周辺装置がデータを受けとったときに返してくるストローブ信号の立ち上がりエッジで非アクティブとなる。

(2)　すでにアクティブであって、さらに続いてポートの出力レジスタに、データを書き込む場合、$\overline{IORQ}$の立ち下がりエッジの3/2Φ後にレディ信号を強制的に非アクティブ（〝Low〟）にする。次に$\overline{IORQ}$の立ち上がりエッジのあとの最初のΦの立ち下がりエッジで、〝High〟レベルに戻る。

このことからもわかるように、ポートのデータが変化すればレディ信号は必ず〝Low〟レベルとなるようになっている。レディ信号は、クロックΦの立ち下がりエッジまでは、非アクティブにならない。クロックが立ち下がるまでレディ信号の立ち下がりを遅らせた理由はこれによって非常に簡単にストローブ・パルスを作ることができるようになるからである。

もし、PIOがリセット状態になければ、モード0を指定する前に出力レジスタに情報をロードすると、モード0を指定したときに、すでにロードしている出力レジスタの情報がそのポートの出力線から出力する。

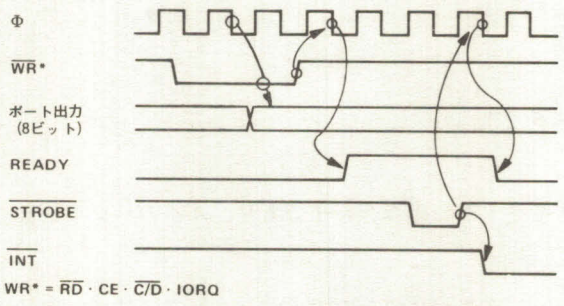


図5－1　モード0（出力）のタイミング

5．2　入力モード（モード1）

図5－2に入力サイクルのタイミングを示す。

CPUがデータの読み込みを完了したのち、周辺装置はストローブ線を使用してこのサイクルを始める。このストローブ信号が〝Low〟レベルになることにより、ポートの入力レジスタへデータをロードし、条件が満たされていれば、ストローブ信号の立ち上がりエッジで割り込み要求線（$\overline{INT}$）がアクティブとなる。入力レジスタが満杯になれば、クロック（Φ）の次の立ち下がりエッジで、レディをリセット（非アクティブ）し、CPUがデータを読み込んでしまうまで、次のデータのロードを禁止する。

そこでCPUは割り込みサービス・ルーチンに入り、そのルーチンで割り込みをかけたポートからのデータを読み込む。読み込み終了後、CPUの$\overline{RD}$信号による$\overline{RD}$*（図5－2参照）の立ち上がりエッジのあとのΦの立ち下がりのときにレディが立ち上がり、次の新しいデータをPIOへロードしてもよい状態になる。

すでにリード待ち状態（アクティブ）であって、データがPIOポートからCPUに読み込まれている間、$\overline{IORQ}$の立ち下がりエッジのあとの3/2Φ後にレディは強制的に〝Low〟レベルに落ちる。

ユーザはレディが〝High〟のときにのみPIOにデータを送り込むようにすれば、CPUがPIOからデータを読み出す期間はレディが〝Low〟レベルとなっているので、読み出し途中でPIOの入力レジスタの内容が変ってしまうことはない。

レディは、すでに述べたように、$\overline{IORQ}$の立ち上がりエッジのあとで再び〝High〟レベルになる。

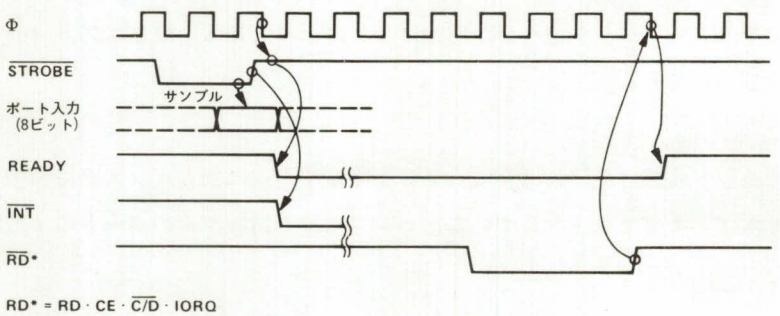


図5－2　モード1（入力）のタイミング

5．3　双方向モード（モード2）

このモードはたんにモード0とモード1の組み合わせであり、4本のハンドシェーク線をすべて使用する方式である。

このモードでは4本のハンドシェーク線が必要なので、データ転送にはポートBは使用せずポートAのみを使用する。このとき、ポートBはビット制御モードにセットしておく必要がある。ポートBをモード3にした場合の割り込みと、ポートAをモード2で動作させかつその入力転送中に生じる割り込みのベクトルは同じも

のになる。これら２つのモードのベクトルの混同はポートＢを制御モード（モード３）で動作させ、かつマスク・レジスタの全ビットをインヒビットすることによって回避できる。なお、モード３のデーター致条件は、ORでなければならない。

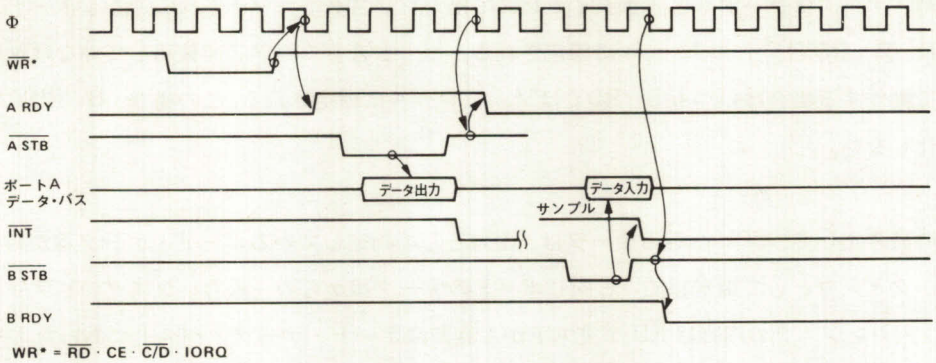


図５－３　モード２（双方向）のタイミング、ポートＡ

双方向モードでのタイミングを図５－３に示す。これはほとんどモード０、モード１で述べたのと同様であり、ポートＡのハンドシェーク線を出力制御用に使用し、ポートＢの線を入力制御用として動作させると考えればよい。その場合の２つのモードの異なる点は、モード２ではＡストローブが〝Low〟レベル（$\overline{\text{A STB}}$がアクティブ）になったときにのみ、データがバス上に乗せられることである。

このストローブの立ち上がりエッジのあとしばらくデータは安定しているので、この立ち上がりエッジを周辺装置へのラッチ信号として使用することができる。

モード２の入力動作はモード１のそれと同じである。

ポートＡ、ポートＢともに割り込み駆動で双方向転送が行えるようにするためには、それぞれが割り込み機能をもつようにしておく必要がある。

$\overline{\text{A STB}}$がアクティブのときには、周辺装置からポート・データ・バス上へデータを乗せてはならない。ちなみに周辺装置からのデータを$\overline{\text{B STB}}$でゲートしておけば、バス駆動の競合状態は生じない。PIOはこのデータを$\overline{\text{B STB}}$の〝Low〟レベルで取り込むようになっている。

PIOは、このモードでデータの取り込みを行う際に、取り込みに必要なホールド時間が零であってもよいように設計されているので、上述のような簡単なゲート方式でよい。すなわち上述のストローブ（$\overline{\text{B STB}}$）の立ち上がりエッジの直後で、バスからのデータは取り込み禁止（ディセーブル）となる。

5．4　ビット制御モード（モード3）

ビット制御モードでは、ハンドシェーク信号は使用しないので、任意の時点で正規のポートの書き込み、読み出しが行える。

書き込み時、データはモード0と同じタイミングで出力レジスタにラッチされる。もしポートAがモード3で動作中には、A　RDYは〝Low〟レベルに固定される。ポートAをモード2で使用していなければ、ポートBがモード3で動作する場合はいつもB　RDYは〝Low〟レベルに固定される。この場合、B　RDYの状態は何によっても変化しない。

PIOからの読み出しで、CPUへ返るデータは、出力として指定しているポート・データ線からの（出力レジスタの）データと入力として指定している同じポートのデータ線からの（入力レジスタの）データとの合成になっている。入力レジスタの内容は$\overline{RD}$が立ち下がる直前にポート・データ・バス上にあったデータに等しい。（図5－4参照）

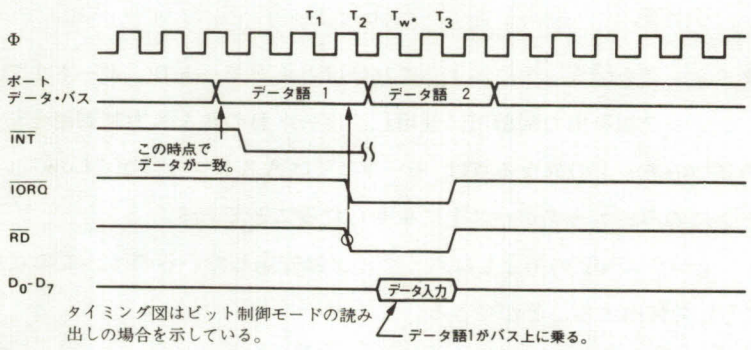


図5－4　ビット制御モードのタイミング

ポートからの割り込みが受け付け可能で、ポート・データ線上の入力データが8ビット・マスクと2ビットのマスク制御レジスタで指定される条件に一致すれば（論理式が等しくなる）、割り込みが発生し、条件の一致状態に変化がなければ、新しく割り込みが発生することはない。

モード3の割り込みは、このモードでの論理演算が〝偽〟から〝真〟に変ったときにのみ発生する。たとえば、モード3での論理式が〝OR〟機能であれば、マスクされないポート・データ線の少なくともいずれかひとつがアクティブになれば割り込み要求が出る。

次に先の状態に続いて他のマスクされないデータ線がアクティブになっても、モード3の論理演算の結果に変化はないので、新たに割り込み要求が出ることはない。

論理演算の結果が$\overline{M1}$サイクルの直前かその期間中に〝真〟になれば、割り込み要求は$\overline{M1}$の立ち上がりエッジで発生する。

第6章　Z-80 PIO 割り込みサービス

PIOから割り込み要求を受け取ったあと、CPUは割り込みアクノリッジを送出する。(このアクノリッジ信号として$\overline{M1}$と$\overline{IORQ}$が同時に出る。)

この期間において、PIO内の割り込み論理回路は割り込み要求を出している最も優先順位の高いポートを決定する。(これは単にIEIが〝High〟レベルでIEOが〝Low〟レベルのデバイスを求めることで決められる。)

デージー・チェーンのイネーブル線の状態を確実にするため、$\overline{M1}$がアクティブになったとき各デバイスの割り込み要求状態の変更は許されない。

割り込みアクノリッジの期間に、割り込みを出している最も優先順位の高いデバイスがCPUデータ・バス上にその割り込みベクトルを乗せる。

図6－1に割り込み要求に関係したタイミングを示す。$\overline{M1}$期間では、新しい割り込み要求を発生させることができない。これは4個までのPIOからの割り込みイネーブル信号を受けるに十分な時間をとる必要があるからである。

$\overline{INTA}$の期間で、IEIが〝High〟レベル、IEOが〝Low〟レベルであるPIOがそのポートの8ビットの割り込みベクトルをデータ・バス上に送り込むことになる。

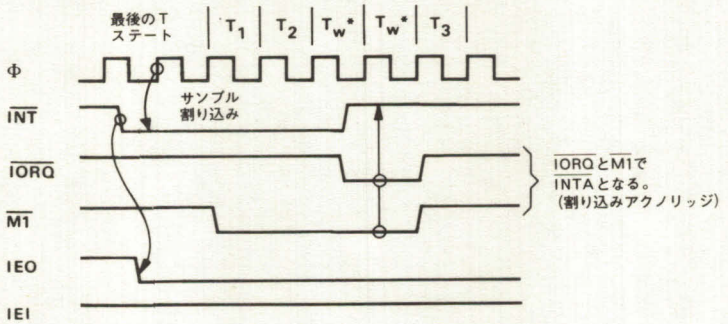


図6－1　割り込みアクノリッジのタイミング

PIOからの割り込み要求が受け付けられると、要求を出したポートが〝サービス〟を受ける。IEIが〝High〟レベルであれば、このポートのIEOはRETI命令を実行すると〝Low〟レベルから〝High〟レベルに変化する。

割り込み要求が受け付けられていないPIOはOPコード〝EDH〟(入出力命令などのOPコード)を解読後、次のM1の1サイクル間IEOを強制的に〝High〟レベルにする。

この動作により、この2バイトのRETI命令を選ばれたPIOポートだけが解読することができるようにしている。(図6－2参照)

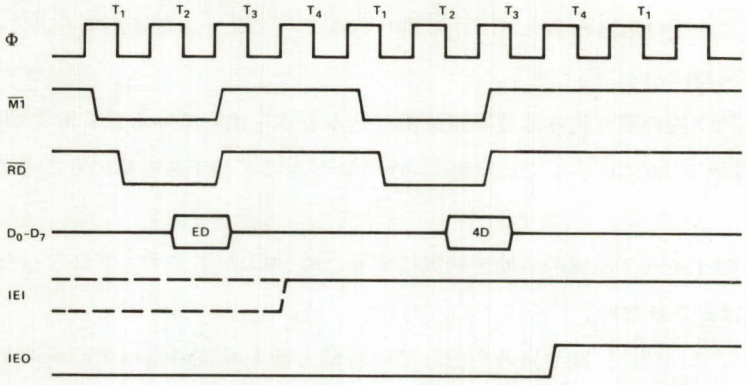


図6－2　割り込みサイクルからの復帰

4つのポートをデージー・チェーンで接続してある場合のネスト構造の割り込みシーケンスの例を図6－3に示す。

まず、ポート2Aが要求を出し割り込みが受け付けられるが、このポートがサービスされている途中で優先順位の高いポート1Bが要求を出すと、これが受け付けられ、ポート2Aのサービスは保留される。

優先順位の高いポート1Bに対するサービス・ルーチンが完了し、RETI命令を実行して、そのルーチンの終了をポート2Aに知らせる。次に優先順位の低いポート2Aのサービス・ルーチンを再開する。

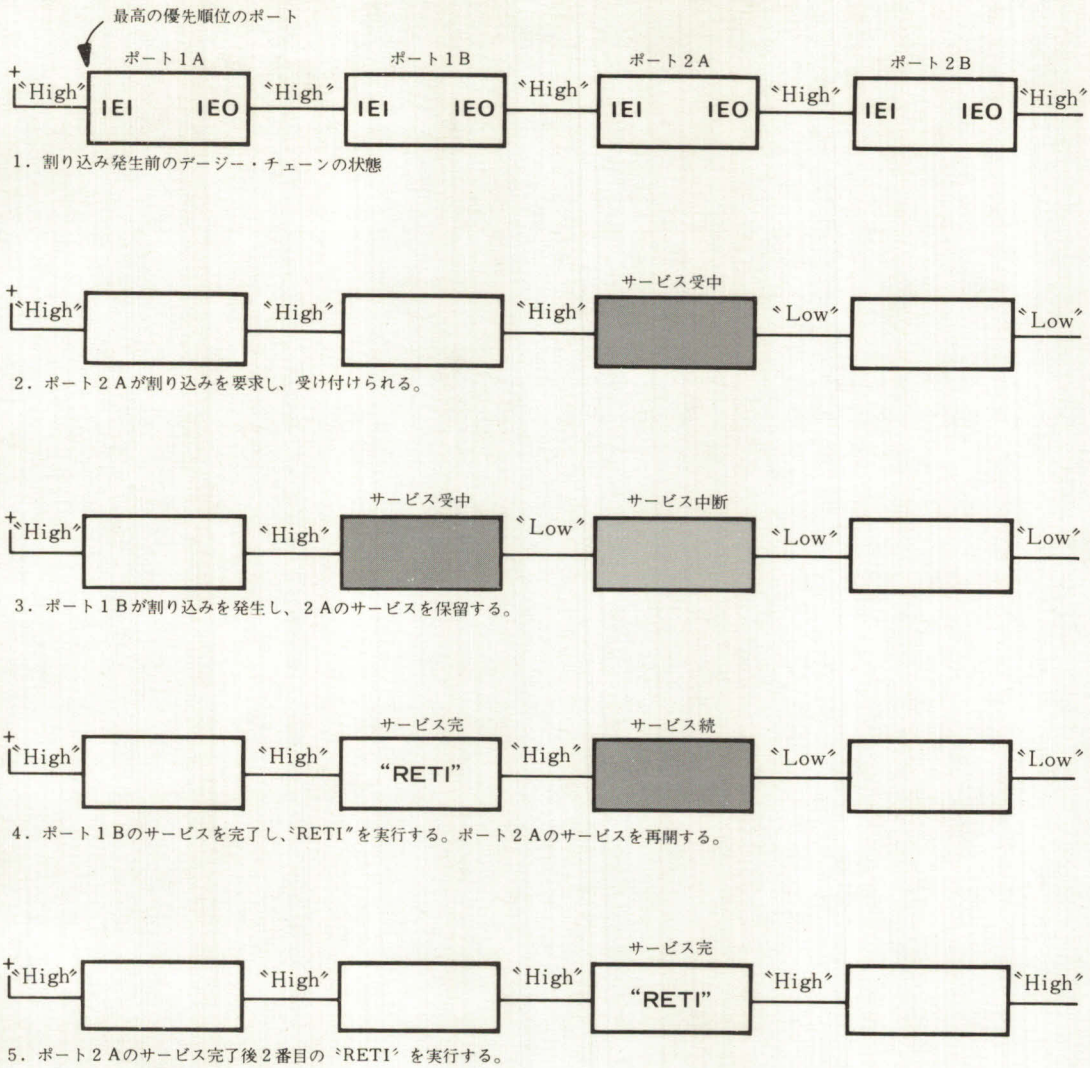


図６－３　デージー・チェーンでの割り込みサービス

第7章　Z-80 PIO 応 用 例

7．1　割り込みデージー・チェーンの拡張

割り込みの優先順位を付けるためのデージー・チェーンには、Z－80 PIO を外部回路を付加しないで最大4個まで接続できる。

この数の制限は、割り込みイネーブル信号（IEO）が各 PIO を伝播していくときの遅延時間によるもので、この遅れは割り込みアクノリッジ・サイクルにおける $\overline{M1}$ の立ち下がりエッジと $\overline{IORQ}$ の立ち下がりまでの間の時間からはみ出してはいけない。

割り込みイネーブルの状態を $\overline{M1}$ 期間では変化させることができない。これにより CPU に戻る PIO からのベクトル・アドレスは、割り込みを要求した最高位の優先順位をもつデバイスからのものであると保証できる。

PIO を4個以上接続する必要がある場合は、図7－1に示すような〝ルック・アヘッド〟構成にするとよい。この方法で標準 TTL を使用して30個以上の PIO を接続できる。

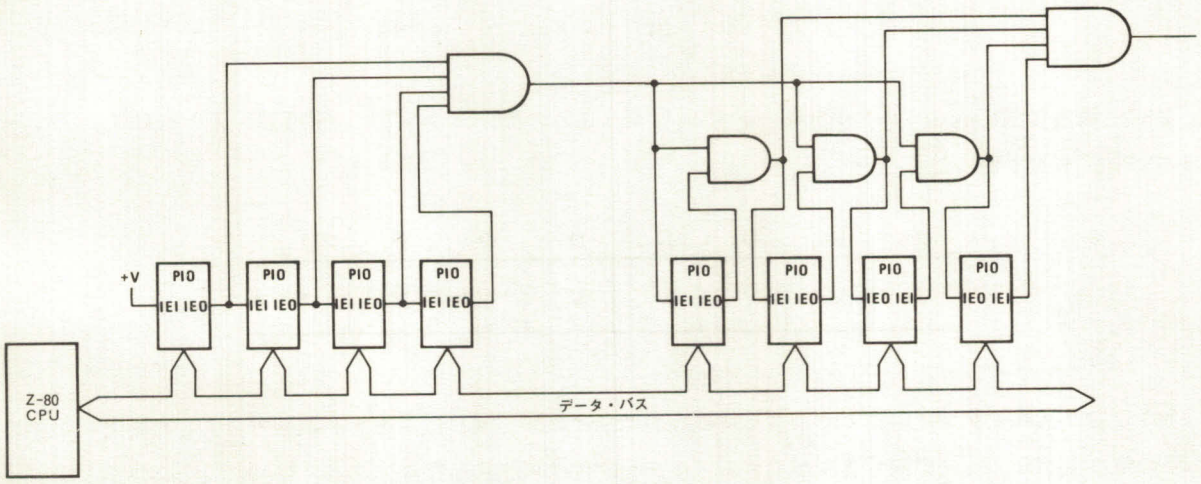


図7－1　デージー・チェーンの拡張方法

7．2　入出力インターフェース

Z－80 PIO を入出力ターミナル・デバイスと結合して図7－2に示すような8ビット並列の双方向データ・バスを介してデータの送受を行う場合、ポートAに次のような制御語を送って PIO をモード2（双方向バス・モード）に設定すればよい。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	X	X	1	1	1	1

モード制御

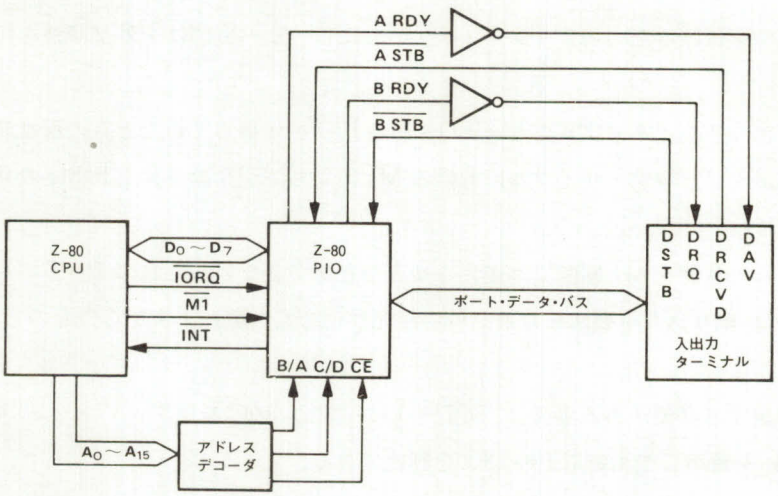


図7－2　入出力インターフェースの例

次に、適当な割り込みベクトルをロードする。(割り込みの操作については、第1部　Z－80 CPU テクニカルマニュアルを参照のこと。)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

$\overline{M1}$ が割り込みアクノリッジ・サイクル時のものでなければ、割り込みモード語をセットしたあとの最初の $\overline{M1}$ の立ち上がりエッジで割り込みがイネーブル（受け付け可能）となる。

割り込みモード語の後にマスク語が続いて送られたときは、マスクがセットされたあとの最初の $\overline{M1}$ の立ち上がりエッジで割り込みはイネーブルとなる。

こうして、周辺装置と CPU との間でデータの転送が可能となる。この転送時のタイミングは第5章で述べたことと同様である。

7．3　制御用インターフェース

ビット制御モードでの例を図７－３に示す。

工業プロセスでのモニタを考えてみる。異常動作状態の発生をＺ－80 CPU を基本とした制御システムに伝達する場合、次のようにフォーマットのプロセス制御語およびステータス語を設定しておくとする。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
特殊テスト	電　源 ON	電源異常アラーム	プロセス停止	温度アラーム	ヒータ ON	圧力系	圧力アラーム

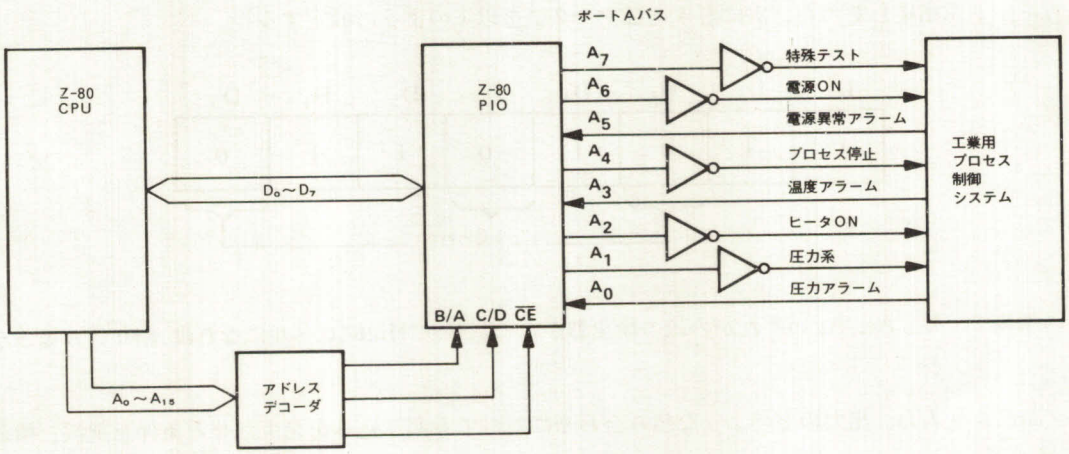


図７－３　制御モードの応用

PIO を次のように使用する。ポートＡに次のような制御語を書き込んでモード３に設定する。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	X	X	1	1	1	1

モード３を指定した場合、その次の制御語は入出力選択語でなければならないので、たとえば、ポート・データ線 A_5、A_3、A_0 を入力とする場合には次のような語を書き込む。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	1	0	0	1

次に必要な割り込みベクトルをロードしておく。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

割り込み制御語を続いてポートに転送する。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	1	1	0	1	1	1
割り込みイネーブル	OR論理	アクティブ〝High〟	マスク	割り込み制御			

割り込みモードを指定したのち、次に書き込むマスク語を以下のように設定する。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	1	0	1	1	0
		モニタ線の指定					

以上の準備で、A_5、A_3、A_0 いずれかの線の検出器からの信号が〝High〟レベルになれば、割り込み要求が発生する。

マスク語により入力、出力のどのような組み合わせに対しても割り込みを発生させる条件を選択、指定できる。たとえば、上述のマスク語が次のようであれば、

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	1	0	1	1	0

割り込み要求は出力レジスタのビット A_7（特殊テスト）をセットしたときにも発生することになる。

ポートの指定方法については、たとえば次のようにするとよい。

$E0_H$ ＝ ポートA　データ

$E1_H$ ＝ ポートB　データ

$E2_H$ ＝ ポートA　制御用

$E3_H$ ＝ ポートB　制御用

（$_H$ は前に置かれた数が16進表現であることを示す。）

ポート番号をこのように指定しておくと、アドレス・バスのA_0をポートB/Aの選択用、アドレス・バスのA_1をC/D（制御信号/データ）の選択用として使用することができ、便利である。

チップ選択（CE）には、A_2からA_7までのCPUのアドレス・ビットをデコードすればよいが、この場合、$A_7 \sim A_2$が11000になればチップ選択される。

周辺装置が少ない場合には、CEとして特別にデコードする必要はなく、必要に応じてA_2からA_7の適当なビットを直接使用することもできることに注意されたい。

第8章　Z-80 PIO プログラミングのまとめ

8．1　割り込みベクトルのロード

V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

8．2　モードの設定

M1	M0	X	X	1	1	1	1

M1	M0	モード
0	0	出　力（0）
0	1	入　力（1）
1	0	双方向（2）
1	1	ビット制御（3）

モード3を指定した場合、次に書き込むデータにより入出力レジスタを設定する。

I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O ＝ 1ならば、入力ビットとなる。

I/O ＝ 0ならば、出力ビットとなる。

8．3　割り込み制御語の設定

割り込みイネーブル	AND/OR	High/Low	マスク指定	0	1	1	1

モード3でのみ使用

マスク指定(Mask follows)が〝High〟レベルならば、次にPIOに書き込むデータはマスクとなる。

MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

MB＝0ならば、モニタ・ビットとなる。
MB＝1ならば、非モニタ・ビットとなる。

また、ポートの割り込みイネーブル・フリップ・フロップ(IFF)を次のようなコマンドを使用して、割り込み制御語の他の部分を変えずにセットまたはリセットすることもできる。

割り込みイネーブル	X	X	X	0	0	1	1

第9章　規　格

絶対最大定格

項　目	記　号	定　格　値	単　位
入力電圧	V_{IN}	−0.3〜+7.0	V
出力電圧	V_{OUT}	−0.3〜+7.0	V
動作温度	Topr	0〜+70	℃
保存温度	Tstg	−65〜+150	℃

絶対最大定格の内のどの1項目でも、瞬時たりとも、絶対最大定格値を越えないようにしてください。かつ、2項目以上の値が、絶対最大定格値に同時に達しないようにしてください。

電気的特性

DC特性

($Ta=0℃〜+70℃$, $V_{CC}=+5V+5\%$)

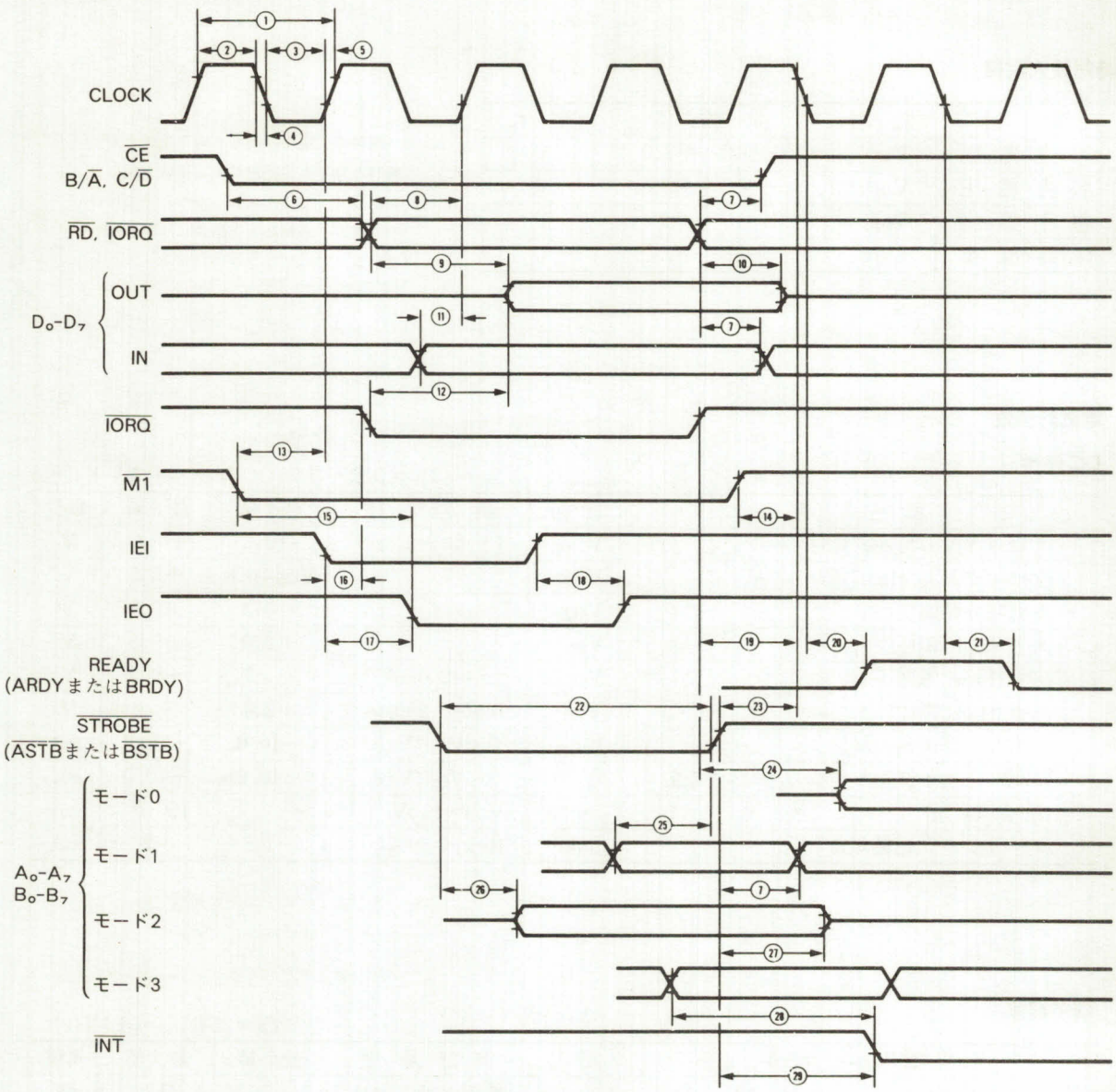
項　目	記　号	測　定　条　件	最小値	最大値	単位
クロック入力"Low"電圧	V_{ILC}		−0.3	+0.45	V
クロック入力"High"電圧	V_{IHC}		V_{CC}−0.6	+5.5	V
入力"Low"電圧	V_{IL}		−0.3	+0.8	V
入力"High"電圧	V_{IH}		+2.0	+5.5	V
出力"Low"電圧	V_{OL}	$I_{OL}=2.0mA$		+0.4	V
出力"High"電圧	V_{OH}	$I_{OH}=-250\mu A$	+2.4		V
入力リーク電流	I_{LI}	$0<V_{IN}<V_{CC}$	−10.0	+10.0	μA
3ステート出力/データ・バスの入力リーク電流	I_Z	$0<V_{IN}<V_{CC}$	−10.0	+10.0	μA
消費電流	I_{CC}	$V_{OH}=1.5V$		100.0	mA
ダーリントン駆動電流	I_{OHD}	$R_{EXT}=390\Omega$	−1.5	3.8	mA

端子容量

($Ta=25℃$, $f=1MHz$)

項　目	記　号	測　定　条　件	最小値	最大値	単位
クロック容量	C_{CLOCK}	被測定端子以外のすべての端子は接地します。		12	pF
入力容量	C_{IN}			7	pF
出力容量	C_{OUT}			10	pF

AC特性


CLOCK
CE
B/A, C/D
RD, IORQ
D0-D7
OUT
IN
IORQ
M1
IEI
IEO
READY
(ARDYまたはBRDY)
STROBE
(ASTBまたはBSTB)
A0-A7
B0-B7
モード0
モード1
モード2
モード3
INT

番号	記　号	項　目	Z-80 PIO		Z-80A PIO		Z-80B PIO(注9)		単位	注
			最小値	最大値	最小値	最大値	最小値	最大値		
1	TcC	クロック・サイクル時間	400	(注1)	250	(注1)	165	(注1)	ns	
2	TwCh	クロック・パルス幅（High）	170	2000	105	2000	65	2000	ns	
3	TwCℓ	クロック・パルス幅（Low）	170	2000	105	2000	65	2000	ns	
4	TfC	クロック↓時間		30		30		20	ns	
5	TrC	クロック↑時間		30		30		20	ns	
6	TsCS(RI)	$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$の$\overline{RD}$ $\overline{IORQ}$↓に対するセットアップ時間	50		50		50		ns	6
7	Th	規定されているセットアップ時間に対する保持時間	0		0		0		ns	
8	TsRI(C)	$\overline{RD}$, $\overline{IORQ}$のクロック↑に対するセットアップ時間	115		115		70		ns	
9	TdRI(DO)	$\overline{RD}$, $\overline{IORQ}$↓からデータ出力遅延時間		430		380		300	ns	2
10	TdRI(DOs)	$\overline{RD}$, $\overline{IORQ}$↑からデータ出力フロートまでの遅延時間		160		110		70	ns	
11	TsDI(C)	入力データのクロック↑に対するセットアップ時間	50		50		40		ns	C_L=50pF
12	TdIO(DOI)	$\overline{IORQ}$↓からデータ出力までの遅延時間（割り込み応答サイクル）		340		160		120	ns	3
13	TsM1(Cr)	$\overline{M1}$↓のクロック↑に対するセットアップ時間	210		90		70		ns	
14	TsM1(Cf)	$\overline{M1}$↑のクロック↓に対するセットアップ時間（$\overline{M1}$サイクル）	0		0		0		ns	8
15	TdM1(IEO)	$\overline{M1}$↓からIEO↓までの遅延時間（$\overline{M1}$↓前の割り込み時）		300		190		100	ns	5,7
16	TsIEI(IO)	IEIの$\overline{IORQ}$↓に対するセットアップ時間（割り込み応答サイクル）	140		140		100		ns	7
17	TdIEI(IEOf)	IEI↓からIEO↓までの遅延時間		190		130		120	ns	5 C_L=50pF
18	TdIEI(IEOr)	IEI↑からIEO↑までの遅延時間（EDデコード後）		210		160		160	ns	5
19	TcIO(C)	$\overline{IORQ}$↑のクロック↓に対するセットアップ時間（次のクロック・サイクルでRDYがアクティブになるとき）	220		200		170		ns	
20	TdC(RDYr)	クロック↓からRDY↑までの遅延時間		200		190		170	ns	5 C_L=50pF
21	TdC(RDYf)	クロック↓からRDY↓までの遅延時間		150		140		120	ns	5
22	TwSTB	$\overline{STB}$パルス幅	150		150		120		ns	4
23	TsSTB(C)	$\overline{STB}$↑のクロック↓に対するセットアップ時間（次のクロック・サイクルでRDYがアクティブになるとき）	220		220		150		ns	5
24	TdIO(PD)	$\overline{IORQ}$↑からポートデータが確定するまでの遅延時間（モード0）		200		180		160	ns	5
25	TsPD(STB)	ポート・データの$\overline{STB}$↑に対するセットアップ時間（モード1）	260		230		190		ns	

番号	記　号	項　目	Z-80 PIO		Z-80A PIO		Z-80B PIO(注9)		単位	注
			最小値	最大値	最小値	最大値	最小値	最大値		
26	TdSTB(PD)	$\overline{STB}$↓からポート・データが確定するまでの時間（モード2）		230		210		180	ns	5
27	TdSTB(PDr)	$\overline{STB}$↑からポート・データフロートまでの遅延時間（モード2）		200		180		160	ns	CL=50pF
28	TdPD(INT)	ポート・データ一致から$\overline{INT}$↓までの遅延時間（モード3）		540		490		430	ns	
29	TdSTB(INT)	$\overline{STB}$↑から$\overline{INT}$↓までの遅延時間		490		440		350	ns	

：↑は立ち上がりエッジ、↓は立ち下がりエッジを示します。

（注1）TcC＝TwCh＋TwCℓ＋TrC＋TfC

（注2）負荷容量の50pF増加につき、TdRI(DO)は10ns増加します。負荷容量の最大値は200pFです。

（注3）負荷容量の50pF増加につき、TdIO(DOI)は10ns増加します。負荷容量の最大値は200pFです。

（注4）モード2のときは、TwSTB＞TsPD(STB)ります。

（注5）負荷容量の10pF増加につき、遅延は2ns増加します。負荷容量の最大値は100pFです。

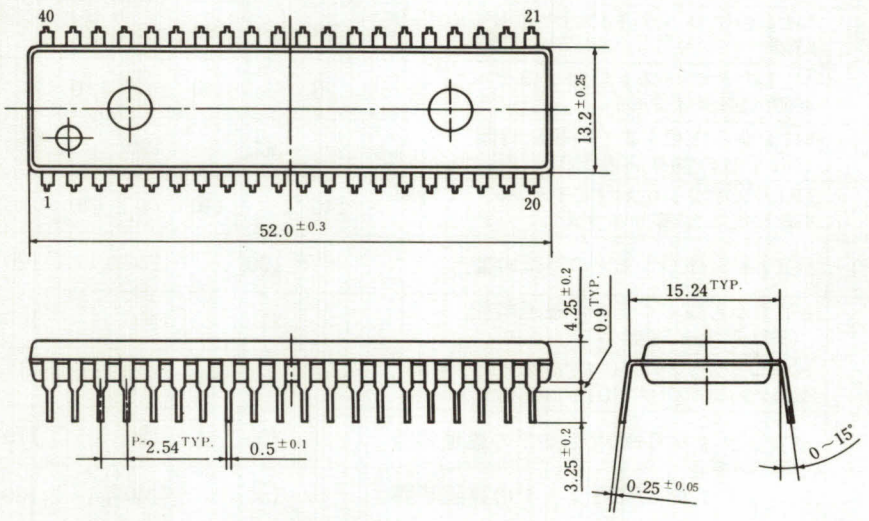
（注6）TsCS(RI)は、最小値より小さくてもよいが、TsCS(RI)より小さくなった時間はTdRI(DO)に加えられます。

（注7）2.5TcC＞(N-2)TdIEI(IEOf)＋TdM1(IEO)＋TsIEI(IO)＋(TTLバッファ使用の場合、その遅延時間)

（注8）PIOをリセットするには、$\overline{M1}$を最低2クロック間アクティブにしなければなりません。

（注9）すべての数値は暫定であり変更することがあります。

外形寸法図



単位：mm

（おことわり）

○製品の改良のため予告なしに内容の一部を変更することがあります。

第3部

Z-80 CTC / Z-80A CTC / Z-80B CTC テクニカルマニュアル

目　　次

第1章　緒　論

シャープZ－80 CTC（Counter Timer Circuit）は4個の独立したチャネルをもつプログラム可能なデバイスであり、Z－80を中心とするマイクロコンピュータ・システムにおいて、各チャネルは計数（カウンタ）と計時（タイマ）の機能をもっている。広い範囲のデバイスとインターフェースさせるために、CPUによりプログラムすることにより、CTCのチャネルは種々のモードを条件下において動作させることができる。CTCを応用した多くの例では、外部に論理回路をほとんどあるいは、まったく必要としない。Z－80 CTCはNチャネル・シリコン・ゲートのE/D MOS技術を使用しており、28ピンDIPのパッケージに納められている。Z－80 CTCは＋5V単一電源、＋5V単相クロックで動作する。Z－80 CTCの主な特長を以下に示す。

- 全入出力はTTLコンパチブルである。
- 各チャネルは独立にカウンタ・モード、タイマ・モードいずれにも選択できる。
- いずれのモードにおいても、CPUからダウン・カウンタの内容を読み出すことが可能であり、その内容は零になるまでのカウント数を示している。
- カウンタ・モード、タイマ・モードいずれもゼロ・カウントになると時間定数レジスタの内容が自動的にダウン・カウンタに再ロードされる。
- タイマ・モードにおいてその動作を起動するトリガ・パルスのエッジの極性を立ち上がり、立ち下がりいずれにも選択でき、同様にカウンタ・モードにおいていずれかのエッジの極性によって、発生数（イベント・カウント）をモニタできる。
- 3個のチャネルは出力ZC/TOをもっており、これらの出力はダーリントン・トランジスタを駆動できる。
- いずれのチャネルにおいても、ゼロ・カウントの条件で割り込みを発生できるようにプログラムできる。
- デージー・チェーン構造の割り込み優先処理用の論理回路をもっており、この回路により外部回路を必要としないで自動的に割り込みベクトル指示ができる。

なお、Z－80A CTCは周波数の上限を4 MHzまで、Z－80B CTCは周波数の上限を6 MHzまで保証した高速タイプであり、第1章から第7章まではとくにZ－80A CTCあるいはZ－80B CTCと記述していないがZ－80 CTCと同様に適用される。電気的仕様など規格については個々に定めている。第8章を参照されたい。

第2章　Z-80 CTC アーキテクチャ

2．1　概　要

Z－80 CTC のブロック図を図２－１に示す。CTCの内部は、Z－80 CPU のバス・インターフェース、内部制御論理回路、４組のカウンタ/タイマ・チャネルの論理回路、割り込み制御回路より構成されている。４組の独立したカウンタ/タイマのチャネルは順に０から３までの番号がついている。CTC は各チャネルごとに個別のベクトルを発生できるようになっており、割り込みサービス・ルーチンに対する自動ベクトル指示を行う。優先順位を決めるためのZ－80の標準的なデージー・チェーン構成において、CTC 内の４組のチャネルは連続したチャネルを構成しており、チャネル０が最上位の優先度をもっている。CPU バス・インターフェースにより CTC は外部に論理回路を必要とすることなく直接に CPU に結線できる。しかしながら、大きいシステムでは、ポートのアドレス・デコーダやライン・バッファが必要となる場合がある。

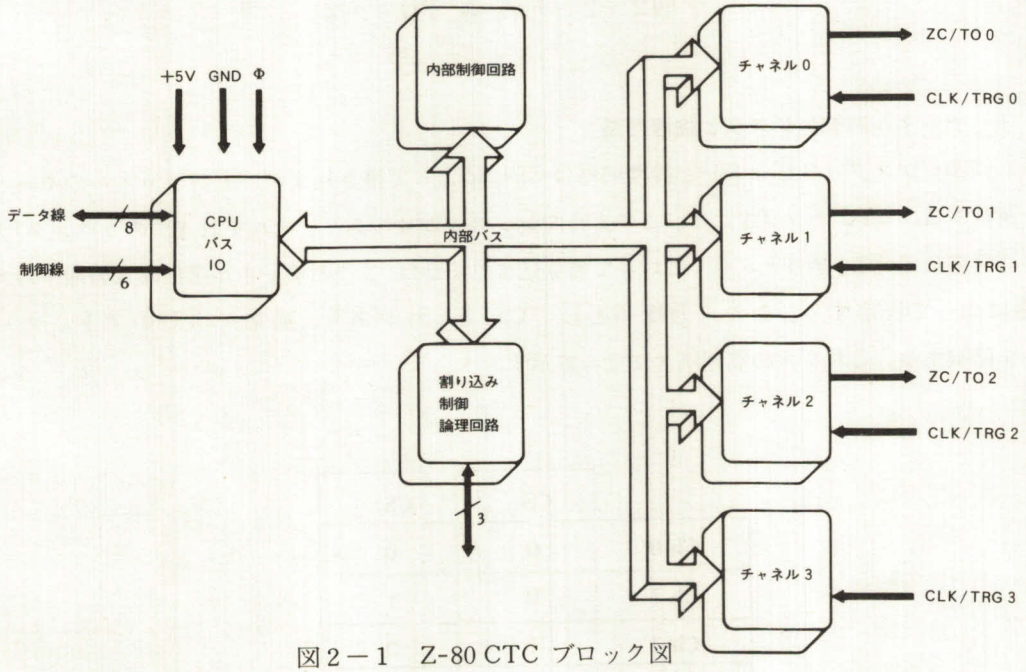


図２－１　Z-80 CTC ブロック図

2．2　チャネル論理回路の構成

４つのカウンタ/タイマ論理回路のうち、その１つの構成を図２－２に示す。この論理回路は２つのレジスタと２つのカウンタおよび制御論理回路より構成されている。レジスタは８ビットの時間定数レジスタと８ビットのチャネル制御レジスタである。カウンタはCPU から読み出し可能な８ビットのダウン・カウンタと８ビットのプリスケーラである。

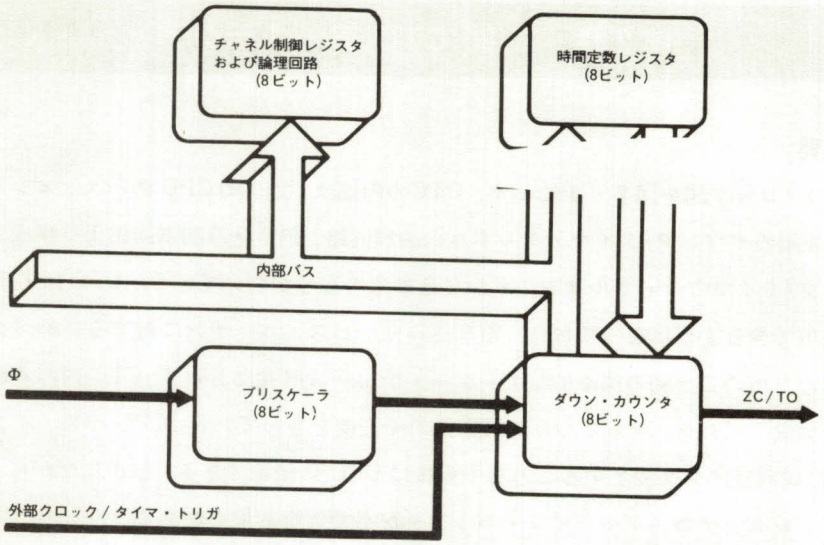


図2－2　チャネル ブロック図

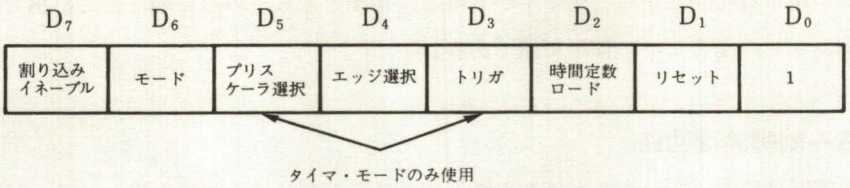
2．2．1　チャネル制御レジスタと論理回路

チャネル制御レジスタ（8ビット）と論理回路はCPUによって書き込まれ、そのチャネルのモードやパラメータを選択する。CTCチップ中には、このようなレジスタが4つあり、それぞれ4つのカウンタ/タイマ・チャネルに対応している。どのチャネルに対して書き込まれるかは、2つのチャネル選択用入力端子のエンコードの状態によって決定する。この入力端子としてCS_0とCS_1があり、通常、CPUのアドレス・バスのA_0とA_1を結線する。これを次の真理値表によって示す。

	CS_1	CS_0
Ch 0	0	0
Ch 1	0	1
Ch 2	1	0
Ch 3	1	1

各チャネルの制御レジスタをプログラムするために制御語を書き込むが、そのビット0はつねに1にセットされており、他の7ビットはそのチャネルの動作モードやパラメータを選ぶために種々にプログラムされる。これを次の図によって示す。詳細は第4章の〝CTCの動作モード〟および第5章の〝CTCのプログラミング〟を参照のこと。

割り込み制御レジスタ



2．2．2　プリスケーラ

プリスケーラはタイマ・モードの場合にだけ使用する。プリスケーラは８ビットの回路であり、チャネル制御レジスタをプログラムすることによってシステム・クロック（Φ）を１/16か１/256に分周する。プリスケーラの出力はダウン・カウンタのクロック入力となっている。ダウン・カウンタには初期状態およびダウン・カウンタが零になるごとに、自動的に時間定数レジスタの内容がロードされる。この結果、このロードされた時間定数により再びシステム・クロックを分周する。ダウン・カウンタが零になるごとに、ゼロ・カウント/タイム・アウト出力（ZC / TO）は〝High〟レベルのパルスを出力する。

2．2．3　時間定数レジスタ

時間定数レジスタは８ビットのレジスタであり、カウンタ・モード、タイマ・モードのいずれの場合にも使用される。このレジスタはCPUによってチャネル制御語を書き込んだ直後に設定し、１から256の間の整数にプログラムされる。このレジスタは上記プログラム値をダウン・カウンタに設定するが、その時点は、CTCを最初に初期化する場合、または、ダウン・カウンタが零をカウントし、自動的にダウン・カウンタに同一値を再ロードする場合のいずれかである。あるチャネルが計数または計時動作中に新しい時間定数を時間定数レジスタに設定する場合、現在実行中のダウン・カウント動作は新しい時間定数がダウン・カウンタに設定されるまで継続する。CTCチャネルに時間定数を設定する方法の詳細は第５章の〝CTCのプログラミング〟を参照のこと。

2．2．4　ダウン・カウンタ

ダウン・カウンタは８ビットのレジスタであり、カウンタ・モード、タイマ・モードのいずれの場合にも使用される。このレジスタには、時間定数レジスタの内容が、初期設定時およびダウン・カウンタが零になった時点でロードされる。ダウン・カウンタの内容は、カウンタ・モードの場合には外部クロックのエッジごとに、また、タイマ・モードの場合にはプリスケーラのクロック出力があるごとにデクリメント（－１）される。CTCに割り当てられたポート・アドレスに対して簡単な入出力読み出し命令を実行することにより、どの時点においても、このレジスタの内容を読み出して零になるまでのカウント数を得ることができる。CTCのどのチャネルに対しても、ゼロ・カウントになるごとに割り込み要求シーケンスを発生するようにプログラムできる。

チャネル０、１、２では、ゼロ・カウントの条件が成立すると、対応するＺＣ／TO端子に信号パルスが出力する。パッケージの端子数の制限から、チャネル３はこの端子をもっていないので、この出力パルスを必要としないアプリケーションの場合だけに使用可能である。

２．３　割り込み制御論理回路

割り込み制御論理回路によって、ネスト構造の優先割り込みと割り込み復帰を用いたＺ－80システムの割り込み方式にしたがってCTCが確実に動作するようにしている。システムを構成するデバイスはどのようなものであれ、その優先順位はデージー・チェーン接続における物理的位置によって決定する。デージー・チェーンを形成するために、CTCには２つの信号線（IEIとIEO）が備わっている。CPUに最も近いデバイスが最も高い優先順位をもっており、CTC内部での割り込み優先順位はチャネル番号によって決まっていて、チャネル０からチャネル３の順番で、チャネル０が最高位、チャネル３が最低位である。CTCから割り込みを発生させる目的は、他の周辺デバイスと同様に、CPUに強制的に割り込みサービス・ルーチンを実行させるためにある。Ｚ－80システムの割り込み方式によれば、下位の優先順位をもったデバイスまたはチャネルはすでに割り込みを発生しており、しかもその割り込みサービス・ルーチンを完了していない高位の優先順位をもったデバイスに対して割り込みをかけられない。しかしながら、優先順位の高いデバイスまたはチャネルは、より優先順位の低いデバイスまたはチャネルのサービスに対して割り込みをかけることができる。

CTCチャネルは、そのダウン・カウンタが零になるたびに割り込み要求を発生するようにプログラムできる。（この方式を利用するためには、CPUは割り込みモード２にプログラムされねばならない。）割り込みを要求すると、CPUは割り込みアクノリッジ信号を送出し、CTCの割り込み制御論理回路により、CTCデバイス内の割り込みを要求している最高位の優先順位のチャネルを決定する。もし、CTCのIEI入力がアクティブであれば、これはシステムのデージー・チェーン接続においてそのCTCが最優先されることを示しており、CTCは８ビットの割り込みベクトルをシステム・データ・バス上に乗せる。このベクトルの上位５ビットはCTCを最初にプログラムする過程の一部分として初めに近い方で書き込まれる。次の２ビットはCTCの割り込み制御論理回路によりチップ内部に与えられ、割り込みを要求している最高位のチャネルに対応する２進コードが入る。最後にベクトルの最下位ビットは次に述べるような約束によってつねに０である。

割り込みベクトル

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

D_2	D_1	
0	0	チャネル 0
0	1	チャネル 1
1	0	チャネル 2
1	1	チャネル 3

この割り込みベクトルは、割り込み処理ルーチンの開始アドレスがテーブル（表）化されて蓄えられているメモリ内の位置を示すポインタとして用いられる。このベクトルは16ビット・ポインタの下位８ビットを与え、上位８ビットはCPU 内のＩレジスタによって与える。ポインタによって指定されたメモリ・アドレスの内容は割り込み処理ルーチンの最初の命令コードのあるアドレスの下位バイトを示し、次のメモリ・アドレスの内容は上位バイトを示している。このような割り込み動作モード（モード２）では、割り込み発生しているCTC内に蓄えられた８ビットのベクトルにより、任意のメモリ・アドレスに対する間接コール（CALL）を行うことができる。

Ｚ－80システムでは、割り込み処理ルーチンのテーブル内のすべてのアドレスの下位バイトはメモリの偶数アドレス内に置かれ、上位バイトは上位隣りのメモリ・アドレスに置かれるので、割り込みベクトルの最下位ビットはつねに偶数であるという取り決めがある。このため、割り込みベクトルの最下位ビットはつねに０とする。

RETI 命令は各割り込み処理ルーチンの最後に使用し、ネスト構造の割り込み優先処理を適切に制御するためデージー・チェーンのイネーブル線 IEO を初期状態に戻す。CTC はシステム・データ・バスを監視し、この命令がバス上に乗るとそれをデコードする。このようにしてCTC 内の制御論理回路は、CPUがいつ割り込み処理を完了したかを知ることができ、CPU と CTC 間で処理が完了したという信号の送受をさらに行う必要はない。

第3章　Z-80 CTC　端子説明

Z－80 CTC の端子配置図を図３－１に示す。ここでは各端子機能について説明する。

$D_7 - D_0$　　Z－80 CPU データ・バス（双方向性、３ステート）

このバスを用いてすべてのデータおよびコマンド語をZ－80 CPU とZ－80 CTC の間で転送する。このバスは８ビットであり、D_0 は最下位ビットである。

$CS_1 - CS_0$　　チャネル選択（入力、アクティブ、〝High〞）

これらの端子により２ビットの２進アドレス・コードを作成し、CTCの独立した４つのチャネルを入出力に対する書き込みや読み出しの場合に選択する。（真理値表は下記を参照のこと）

	CS_1	CS_0
Ch 0	0	0
Ch 1	0	1
Ch 2	1	0
Ch 3	1	1

$\overline{CE}$　　チップ・イネーブル（入力、アクティブ、〝Low〞）

この端子を〝Low〞レベルにすることにより、入出力書き込みサイクルの期間にCTCはZ－80データ・バスから制御語、割り込みベクトル、あるいは時間定数データ語を受け取ることができ、一方、入出力読み出しサイクルの期間にCPU に対してダウン・カウンタの内容を転送することができる。多くの応用では、この信号はアドレス・バスの下位８ビットのうち適当なビットをデコードしたものであり、４個の入出力ポート・アドレスのいずれかに対して４個のカウンタ・タイマのいずれかのチャネルが対応する。

クロック（Φ）システム・クロック（入力）

単相クロック。チップ内部の信号の同期用として使用される。

$\overline{M1}$　　CPUからマシン・サイクル1を知らせる信号（入力、アクティブ、〝Low〟）

$\overline{M1}$がアクティブで、かつ$\overline{RD}$信号がアクティブの場合、CPUはメモリから命令をフェッチしている。$\overline{M1}$がアクティブであり、かつ$\overline{IORQ}$信号がアクティブの場合、CPUは割り込みアクノリッジ・サイクルにあり、もし、CTCがデージー・チェーン接続において最優先順位にあり、しかも、そのCTCのチャネルのうちどれかのチャネルが割り込みを要求しているならば、そのCTCに対してZ－80データ・バス上に割り込みベクトルを乗せるように指示する。

$\overline{IORQ}$　　CPUからの入出力要求（入力、アクティブ〝Low〟）

$\overline{IORQ}$信号は、データやチャネル制御語をZ－80 CPUとCTCの間で転送する場合に$\overline{CE}$や$\overline{RD}$信号と組み合わせて用いる。CTCに対する書き込みサイクルの期間、$\overline{IORQ}$と$\overline{CE}$は論理〝1〟で$\overline{RD}$は論理〝0〟でなければならない。CTCは特定の書き込み信号を受け取らないが、その代りに内部で$\overline{RD}$信号を反転することにより自分自身で書き込み信号を作成している。CTCの読み込みサイクルでは、$\overline{IORQ}$、$\overline{CE}$および$\overline{RD}$信号はダウン・カウンタの内容をZ－80データ・バス上に乗せるためにアクティブでなければならない。もし$\overline{IORQ}$、$\overline{M1}$が共に論理〝1〟であれば、CPUが割り込みアクノリッジ・サイクルにあり、割り込みを要求している最高位のチャネルから割り込みベクトルをZ－80データ・バス上に出力する。

$\overline{RD}$　　CPUからの読み出しサイクル・ステータス（入力、アクティブ〝Low〟）

$\overline{RD}$信号は、データやチャネル制御語をZ－80 CPUとCTCの間で転送する場合に、$\overline{IORQ}$や$\overline{CE}$信号と組み合わせて用いる。CTCの書き込みサイクルの期間、$\overline{IORQ}$と$\overline{CE}$は論理〝1〟で$\overline{RD}$は論理〝0〟でなければならない。CTCは特定の書き込み信号を受け取らないので、代りに内部で$\overline{RD}$信号を反転することにより自分自身で書き込み信号を作成している。CTCの読み出しサイクルでは、$\overline{IORQ}$、$\overline{CE}$および$\overline{RD}$はダウン・カウンタの内容をZ－80データ・バス上に乗せるためにアクティブでなければならない。

IEI　　割り込みイネーブル入力（入力、アクティブ〝High〟）

この信号はシステム内で割り込み制御用のデージー・チェーンを形成するためにあり、システム内で割り込みを発生できる周辺装置が2個以上ある場合に、この信号と次に述べるIEOにより優先順位を決定する。

IEO　　割り込みイネーブル出力（出力、アクティブ〝High〟）

この信号はIEIとともに用いシステム内で割り込み優先レベルを決定するデージー・チェーンを形成する。IEIが〝High〟レベルで、かつそのCTC内のどのチャネルからの割り込み要求もなく、さらにCPUが現在そのCTCの割り込みを処理していない場合のみ、IEOは〝High〟レベ

ルとなる。このようにして、高位の割り込み優先順位の装置がCPUによって割り込み処理をされている期間、下位の割り込み優先順位の装置からの割り込みを防いでいる。

$\overline{\text{INT}}$　　割り込み要求（出力、オープン・ドレイン、アクティブ〝Low〟）

IEIが〝High〟であり、かつ割り込みイネーブルとなるようにプログラムされているCTC内のどれかのチャネルにおいて、そのダウン・カウンタがゼロ・カウントの条件に達するとこの信号はアクティブになる。

$\overline{\text{RESET}}$　　リセット（入力、アクティブ〝Low〟）

この信号によりすべてのチャネルのカウント動作は停止し、すべての制御レジスタ中の割り込みイネーブル・ビットをリセットする。このためCTCから割り込みを発生できなくなる。ZC / TOおよび$\overline{\text{INT}}$出力は動作していない。すなわち、アクティブでない状態になり、IEOはIEIに等しくなり、CTCのデータ・バスの出力ドライバは高インピーダンスの状態になる。

CLK / TRG_3 — CLK / TRG_0

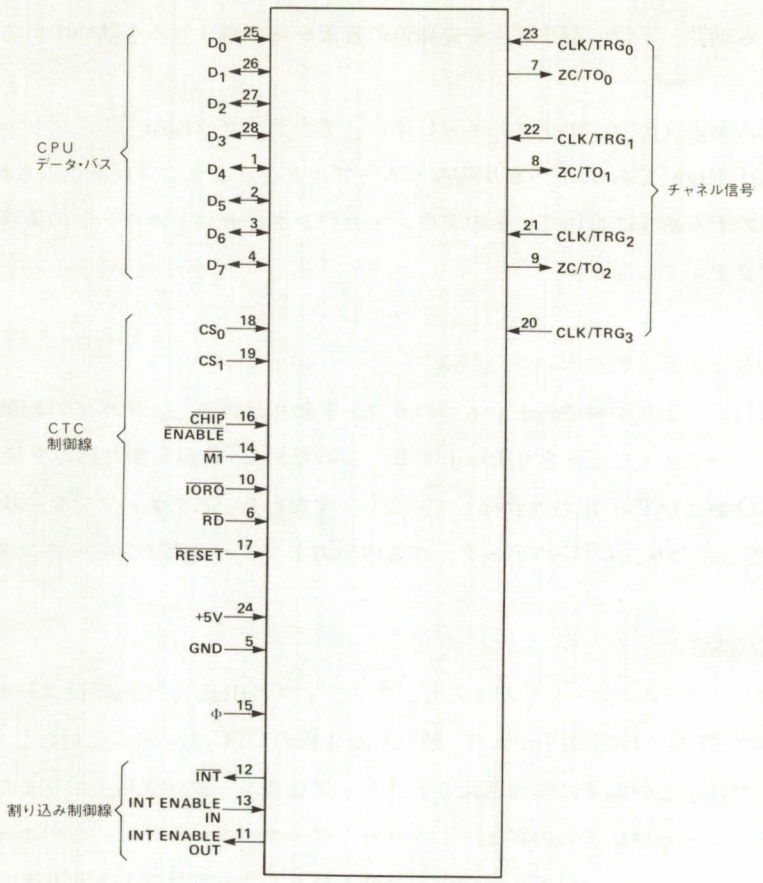
外部クロック/タイマ・トリガ（入力、アクティブ〝High〟、〝Low〟はユーザ選択可能）

4本のCLK / TRG端子があり、独立した4個のCTCチャネルに対応している。カウンタ・モードでは、この端子に印加されるアクティブな各エッジ（立ち上がりまたは立ち下がり）ごとにダウン・カウンタの内容は－1される。タイマ・モードでは、このエッジに印加されるアクティブ・エッジにおいて計時動作が起動される。ユーザは立ち上がりまたは立ち下がりのアクティブ・エッジを選択できる。

ZC / TO_2 — ZC / TO_0

ゼロ・カウント/タイム・アウト（出力、アクティブ〝High〟）

3本のZC / TO端子があり、CTCチャネル2、1、0に対応している。（パッケージ端子の制限により、チャネル3にはZC / TO端子がない。）カウンタ・モード、タイマ・モードいずれの場合においても、ダウン・カウンタが零になると、この端子からアクティブ〝High〟のパルスが出力する。



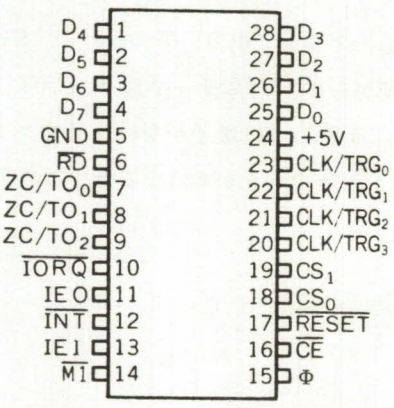


図３－１　CTC 端子配置図

第4章　Z-80 CTC 動作モード

電源投入後のZ－80 CTCの状態は決まっておらず、CTCに $\overline{\text{RESET}}$ 信号を印加することにより状態が定まる。いずれのチャネルにおいても、計数または計時動作を始めるまえに、チャネル制御語と時間定数データ語を対応するチャネルの決められたレジスタに書き込まなければならない。さらにいずれのチャネルにおいても割り込みをイネーブルにするようにプログラムされているならば、そのまえに割り込みベクトル語を対応する割り込み制御論理回路に書き込む必要がある。(詳細は第5章の〝CTCのプログラミング〟を参照のこと。) CPUがこれらのデータをCTCに対して書き終えると、すべてのアクティブなチャネルはカウンタ・モード、タイマ・モードのいずれかですぐに動作を始めることができる状態にプログラムされたことになる。

4．1　CTCカウンタ・モード

このモードでは、CTCはCLK / TRG入力のエッジの数を計数する。あるチャネルをカウンタ・モードにプログラムする場合、そのチャネルの制御語のビット6を1に設定する。チャネルの外部クロック(CLK / TRG)入力端子において一連のトリガ・パルスのエッジ数を計数し、各トリガ・パルスの入力後、次のΦ(システム・クロック)の立ち上がりに同期してダウン・カウンタの内容を－1する。ダウン・カウンタの内容は、一連のカウント動作を行うまえに、まえもって時間定数レジスタの値に初期設定されている。外部クロックのトリガ・エッジとシステム・クロックΦの立ち上がりとの間にはセット・アップ時間は必要としないが、ダウン・カウンタは次のΦが来るまで－1しない。(第8章の〝規格〟の項のパラメータ t_S(CK) を参照のこと。) ダウン・カウンタが外部クロックの立ち上がりまたは立ち下がりのいずれによって－1されるかは、対応するチャネルの制御語のビット4によってまえもってプログラムされる。

ダウン・カウンタが最初にロードされた時間定数の値から連続して－1され、零に達すると、チャネル0、1、2いずれのチャネルにおいても、対応するゼロ・カウント(ZC / TO)出力端子からアクティブ〝High〟のパルスが出力する。(しかしながら、パッケージの端子数の制限からチャネル3にはこの端子がないので出力パルスを必要としない応用の場合だけに使用することができる。)さらにチャネル制御語のビット7を設定することによりそのチャネルから割り込みを発生できるようにまえもってプログラムされているならば、続いて割り込み要求シーケンスが発生する。(詳細は第7章の〝CTCの割り込みサービス〟を参照のこと。)

上記シーケンスが行われている間に、ゼロ・カウントの発生条件により、時間定数レジスタに書き込まれていた最初の時間定数データが自動的にダウン・カウンタに再ロードされる。これは続行しているダウン・カウンタ動作に何ら影響を与えない。ダウン・カウンタの動作中に新しい時間定数データが時間定数レジスタに書き込まれると、現在のカウント動作完了後にその新しい時間定数がダウン・カウンタにロードされる。

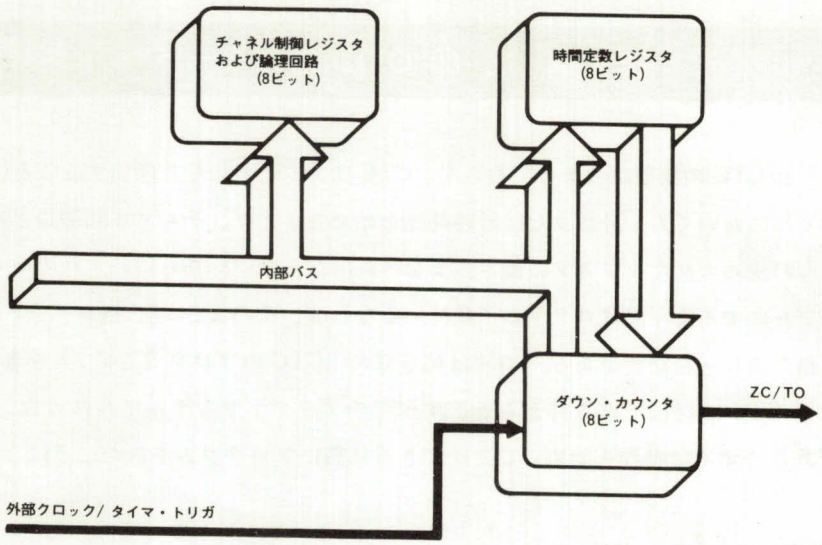


図４－１　チャネル　カウンタ・モード

４．２　CTC タイマ・モード

このモードでは CTC はシステム・クロックの周期の整数倍の時間間隔を発生する。あるチャネルに対するタイマ・モードは対応するチャネル制御語のビット６をリセット（〝0〟）することによりプログラムされる。以後そのチャネルはシステム・クロック周期を基準とした時間間隔を測定するために使用できる。システム・クロックは２つの連結したカウンタであるプリスケーラとダウン・カウンタのうちプリスケーラに供給される。まえもってチャネル制御語のビット５にプログラムされている内容により、プリスケーラはシステム・クロックを16または256で分周する。プリスケーラの出力はダウン・カウンタを－１するためのクロックとして用いられる。ダウン・カウンタは１から256のいずれかの整数にまえもってプログラムされる。カウンタ・モードと同様に、ダウン・カウンタが零となるごとに時間定数がダウン・カウンタに自動的に再ロードされ、カウント動作は続行する。また、ゼロ・カウンタ時にはチャネルのタイム・アウト（ZC / TO）出力（これはダウン・カウンタの出力に等しい）もパルス出力し、次式で与えられる正確な周期の一様なパルス列を発生する。

$$t_C * P * TC$$

t_C はシステム・クロックの周期、Pはプリスケーラの値で16か256。TC はまえもってプログラムされている時間定数である。ただし、TC ＝０の場合、TC ＝256として計算する。

チャネル制御語のビット３をまえもってプログラムすることにより、タイマ動作が自動的に起動されるか、あるいは対応するチャネルのタイマ・トリガ（CLK / TRG）入力のトリガ・エッジで起動されるかを選択する。ビット３がリセットされている場合、対応するチャネルに時間定数データ語をロードするための入出力書き込みマシン・サイクルに続く CPU サイクルが始まると、タイマは自動的に動作し始める。ビット３がセット

されている場合、時間定数データ語をロードしたあとで入力するタイマ・トリガ・エッジから数えて２個目のΦの立ち上がりにおいて、タイマは動作し始める。時間定数データをロードしない場合、制御語の書き込みサイクルに続いて入力するタイマ・トリガのエッジから数えて２個目のΦの立ち上がりエッジにおいて、タイマは動作し始める。チャネル制御語のビット４をまえもってプログラムすることにより、タイマ・トリガが立ち上がりあるいは立ち下がりのどちらのエッジで動作するかを選択できる。タイマ・トリガのアクティブ・エッジと次のΦの立ち上がりとの間のセット・アップ時間には何ら必要な条件はないが、もし、タイマ・トリガのエッジとΦの立ち上がりとの間のセット・アップ時間がある最小値以下である場合、ダウン・カウンタは次のΦの立ち上がりまで－１しない。（第８章〝規格〟の項のパラメータ t_S(TR）参照のこと。）

ダウン・カウンタ内のゼロ・カウントの条件成立により、対応するチャネルのタイム・アウト端子にパルスが出力するが、さらにチャネル制御語のビット７がセットされていると、この条件は割り込み要求シーケンスを起動するために使用される。（詳細は第７章の〝CTC 割り込みサービス〟を参照のこと。）

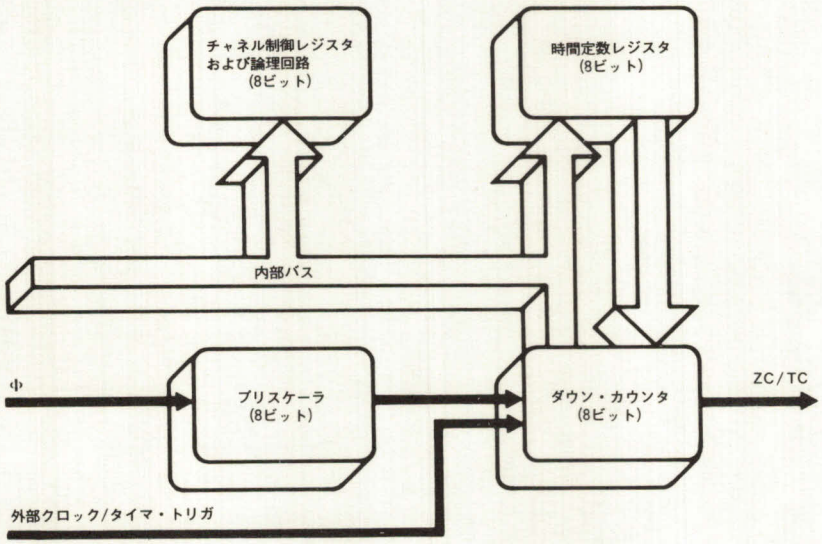


図４－２　チャネル　タイマ・モード

第5章　Z-80 CTC プログラミング

Z－80　CTCのチャネルをカウンタまたはタイマ・モードで動作させるまえに、まずそのチャネルにチャネル制御語と時間定数データを書き込まなければならない。これらのデータは対応するチャネルのチャネル制御レジスタと時間定数レジスタに蓄えられる。さらに、4個のチャネルのいずれかが、そのチャネル制御語のビット7をセットすることにより、割り込みイネーブルとなるようにプログラムされている場合、割り込みベクトルもCTC内に用意されているレジスタに書き込まなければならない。割り込み制御論理回路によりチャネル0にベクトルを設定すると、残りのチャネルのベクトルは自動的に決定する。

5．1　チャネル制御レジスタのロード

チャネル制御語をロードするために、CPUは希望するCTCチャネルに対応するポート・アドレスに対して通常の入出力書き込みシーケンスを実行する。CS_0とCS_1の入力端子に対して2ビットの2進アドレスを作成し、デバイス内の4チャネルのうちの1チャネルを選択する。(真理値表は2．2．1〝チャネル制御レジスタと論理回路〟参照のこと。)多くのシステム例では、この端子はそれぞれアドレス・バス線A_0、A_1に接続され、CTC内の4チャネルは連続した入出力ポート・アドレスを占める。CTCに書き込まれるデータは、そのビット0（D_0）を論理1にした場合、チャネル制御語と解釈され、チャネル制御レジスタにロードされる。この語の残りの7ビットにより下図に示す動作モードと状態を選択する。図にしたがって各ビットの意味を詳細に説明する。

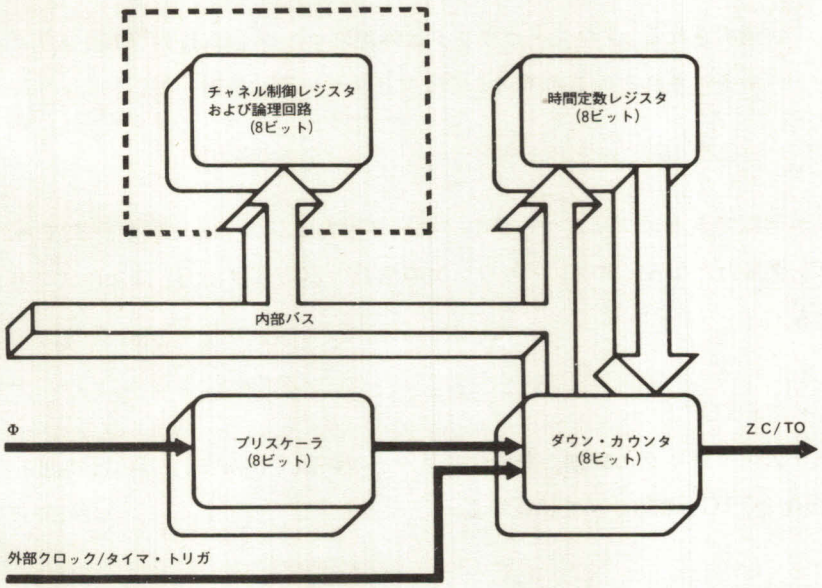
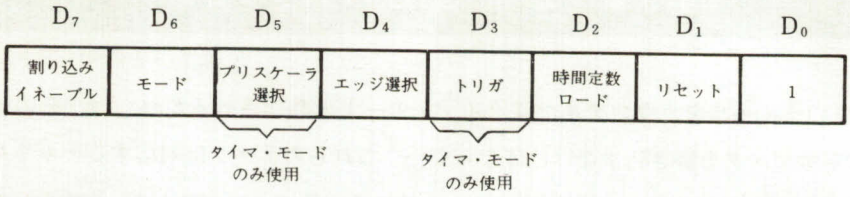


図5－1　チャネル　ブロック図



ビット７＝１

そのチャネルの割り込み発生をイネーブルにし、ダウン・カウンタがゼロ・カウントの条件になるごとに割り込み要求シーケンスを発生する。どのチャネル制御レジスタであれ、このビットを１に設定すると動作を開始するまえに割り込みベクトルをCTCに書き込む必要がある。カウンタ・モードでも、タイマ・モードでも、チャネル割り込みはプログラム可能である。すでに動作中のチャネルに対し新しくチャネル制御語を書き込み、しかもそのビット７が設定されている場合、新しくゼロ・カウントの条件が成立するまで割り込みは発生しない。

ビット７＝０

チャネル割り込みは発生できない。

ビット６＝１

カウンタ・モードが選択される。ダウン・カウンタは外部クロック（CLK / TRG）入力の各トリガ・エッジごとにデクリメント（－１）される。このモードではプリスケーラは使用しない。

ビット６＝０

タイマ・モードが選択できる。プリスケーラのクロック入力はシステム・クロックであり、出力はダウン・カウンタのクロック入力となる。ダウン・カウンタの出力（ZC / TO出力）は次の式で与えられる周期の一様なパルス列になる。

$$t_C * P * TC$$

ここでt_Cはシステム・クロックの周期、Pはプリスケーラの値で16か256、TCは時間定数データ語である。ただし、TC＝０の場合、TC＝256として計算する。

ビット５＝１

（タイマ・モードの場合のみ定義される。）プリスケーラ値　256

ビット５＝０

（タイマ・モードの場合のみ定義される。）プリスケーラ値　16

ビット４＝１

タイマ・モード――トリガ・パルス（CLK / TRG）の立ち上がりでタイマ動作を起動する。

カウンタ・モード――外部クロック・パルス(CLK / TRG)の立ち上がりでダウン・カウンタの内容を－１する。

ビット４＝０

タイマ・モード――トリガ・パルス（CLK / TRG）の立ち下がりでタイマ動作を起動する。

カウンタ・モード――外部クロック・パルス(CLK / TRG)の立ち下がりでダウン・カウンタの内容を－１する。

ビット３＝１（タイマ・モードのみ）

タイマ動作の起動は外部トリガ・パルスによって行われる。時間定数をロードしたあとのマシン・サイクルのT_2の立ち上がり以後に入力するトリガ・パルスにより、タイマは動作を始める。クロックΦとトリガ・パルスのセット・アップ時間の関係により、２クロック・サイクルまたは３クロック・サイクル後にプリスケーラはデクリメント（－１）される。

ビット３＝０（タイマ・モードのみ）

時間定数をロードしたあとのマシン・サイクルのT_2の立ち上がりでタイマは動作を始める。

ビット２＝１

このチャネルに対して次に書き込まれるデータは時間定数レジスタに蓄えられる時間定数データ語となる。もし、すでに動作のチャネルに対して、新しくチャネル制御語と時間定数データ語を書き込んだ場合、ダウン・カウンタは零になるまでデクリメントを続け、そのあとで新しい時間定数がダウン・カウンタにロードされる。

ビット２＝０

チャネル制御語の書き込み後に時間定数レジスタに対する時間定数データ語の書き込みがない場合、このビットを０にする。チャネルは時間定数レジスタに正しい順序でデータ語を書き込まない限り動作しないようになっているので、すでに動作中のチャネルの状態を更新する意志がある場合、チャネル制御語のビット２をこの状態にしておく。このチャネル制御語のビット２を１にした場合に限り時間定数レジスタに書き込みが可能である。

ビット１＝１

チャネルをリセットし、カウンタまたはタイマ動作を停止する。このビットはラッチされない。このビットに１を書き込むと現在のチャネル動作を停止し、他のビットはチャネル制御レジスタのそれぞれのビットに書き込まれる。もし、ビット２＝１で、かつ、ビット１＝１の場合、そのチャネルは時間定数データ語をロードする動作を再開する。ビット２＝０で、かつ、ビット１＝１の場合、新しい制御語を書き込むまでチャネルは動作しない。

ビット１＝０

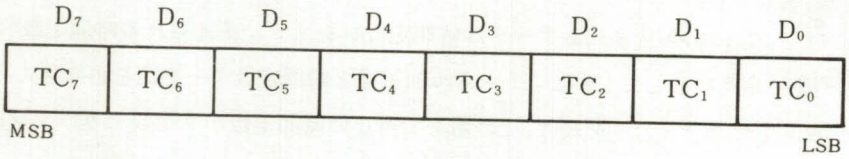
チャネルは現在の動作を継続する。

５．２　時間定数レジスタのロード

タイマ・モードであれ、カウンタ・モードであれ、チャネルはその時間定数レジスタに時間定数データを書き込まない限り動作を始めることはない。

もし、チャネル制御語のビット２がセットされているならば、このデータはチャネル制御語を出力命令で書き込んだあとで、このチャネルに対して行われる次の出力命令によって与えられる。時間定数データ語は１から256の範囲のいずれの整数値でもよい。もし、このデータ語の８ビットすべてが０の場合、256と解釈される。もし、時間定数データ語がすでに動作中のチャネルにロードされた場合、ダウン・カウンタは零になるまでデクリメントを続け、そのあとで時間定数レジスタからダウン・カウンタに新しい時間定数がロードされる。

時間定数レジスタ



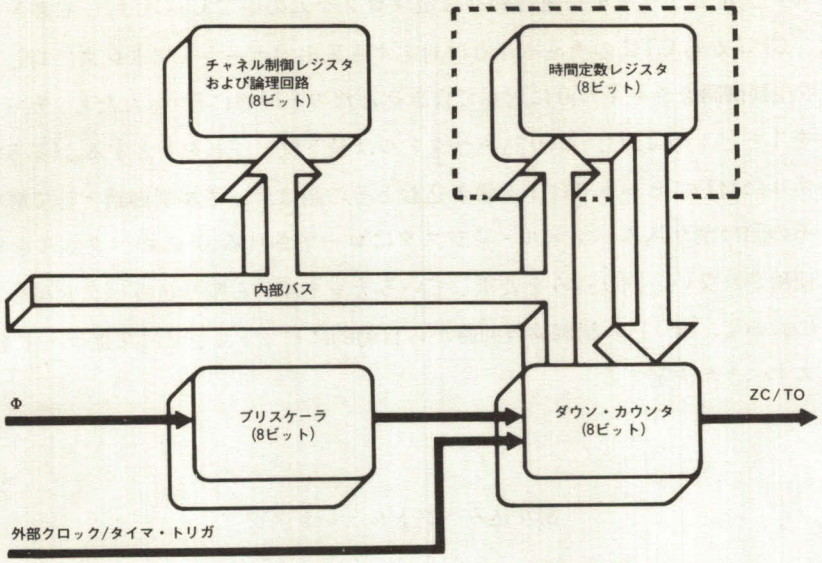
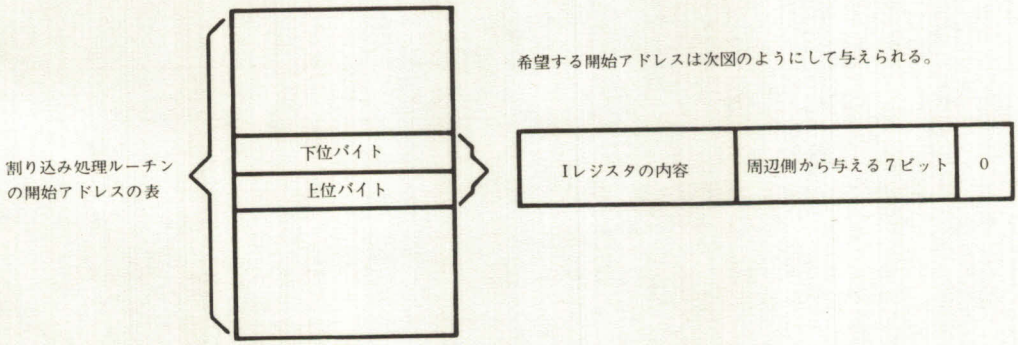


図５－２　チャネル　ブロック図

５．３　割り込みベクトル・レジスタのロード

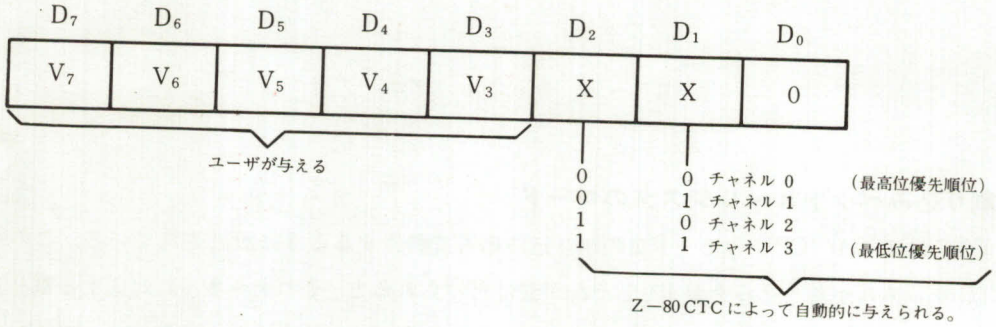
Ｚ－80　CTC はＺ－80　CPU のモード２の割り込み応答で動作するように設計されている。このモードでは、ある CTC チャネルが割り込みを要求し、それが受け付けられると、そのチャネルに対応する割り込み処理ルーチンの開始アドレスをメモリ内のテーブル（表）から得るために、16ビットのアドレス・ポインタを作成しなければならない。このポインタの上位８ビットは CPU 内のＩレジスタにより与えられ、下位８ビットは割り込みベクトルの形で与えられる。この割り込みベクトルは割り込みを要求しているチャネルのみに割り当てられた個有のものである。（詳細は第７章の〝CTC の割り込みサービス〟を参照のこと。）

モード２の割り込み動作



割り込みベクトルの上位５ビットを一連の初期設定プログラムの中でCTCに対して書き込まなければならない。このために、CPUからCTCのチャネル０に対応する入出力ポート・アドレスに対してデータを書き込むが、これはチャネル制御語をチャネル０に対して書き込んだのと同様に行う。ただし、チャネル制御語はデータの最下位ビットを１としたのに対し、割り込みベクトルは最下位ビットを０とする。（第５章５．１節で説明したように、チャネルに対してビット０に１を書き込むとその語はチャネル制御語として解釈され、ビット０に０を書き込むとその語は割り込みベクトル・レジスタにロードされる。）このベクトルをロードする場合、ビット１、２、は使用されない。割り込みを要求しているチャネルから割り込みベクトルをＺ－80　データ・バスに乗せる時点において、割り込み制御論理回路から自動的にビット１、２に２進コードを与え、どのCTCチャネルをサービスすべきか決定する。

割り込みベクトル・レジスタ



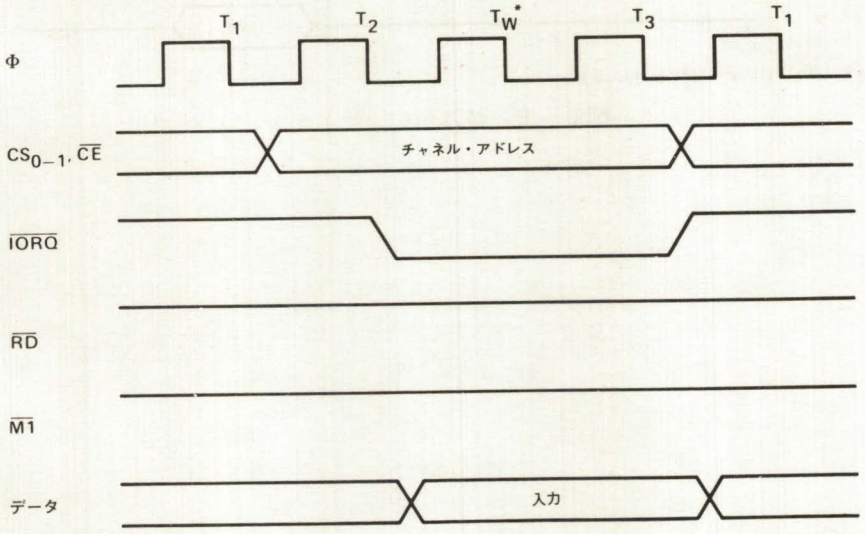
第6章　Z-80 CTC タイミング

ここでは、次に示す動作タイプについてCTC端子の時間関係を説明する。動作タイプとしてはCTCに対するデータ語の書き込み、データ語の読み出し、カウンタ・タイマ・モードがある。このマニュアルでは、第7章で割り込みサービスに関するタイミング図を示し、定量的なAC特性のタイミング図は、第8章において示す。

6．1　CTC書き込みサイクル

図6－1にCTC書き込みサイクルに関するタイミングを示す。このシーケンスは、チャネル制御語、割り込みベクトル、時間定数データ語のいずれの書き込みに対しても適用できる。

図に示されるように、クロック・サイクルT_1の期間に、Z－80　CPUはCTCの入力端子$\overline{RD}$（読み出し）を論理0(〝High〞レベル)にして書き込みサイクルの準備をする。CTCは書き込み信号用の入力端子をもっていないので、$\overline{RD}$信号が〝High〞レベルになったことにより内部で書き込み信号を発生する。次に、クロック・サイクルT_2の期間に、Z－80　CPUはCTCの入力端子$\overline{IORQ}$（入出力要求）と$\overline{CE}$（チップ・イネーブル）を論理1(〝Low〞レベル)にして書き込みサイクルを起動する。($\overline{M1}$はこのサイクルと割り込みアクノリッジ・サイクルを区別するために論理0（〝High〞レベル)でなければならない。)この時点で2ビットの2進コードをCTCの入力端子CS_1とCS_0（チャネル選択1、0）に加え、CTC内の4チャネルのうち、どのチャネルに書き込むかを決定し、書き込まれるデータ語がZ－80　データ・バス上に現れる。この状態で、CTCの適当な内部レジスタにクロック・サイクルT_3の立ち上がりに同期してデータをラッチする用意ができたことになる。外部から待ち状態を加えることはできない。

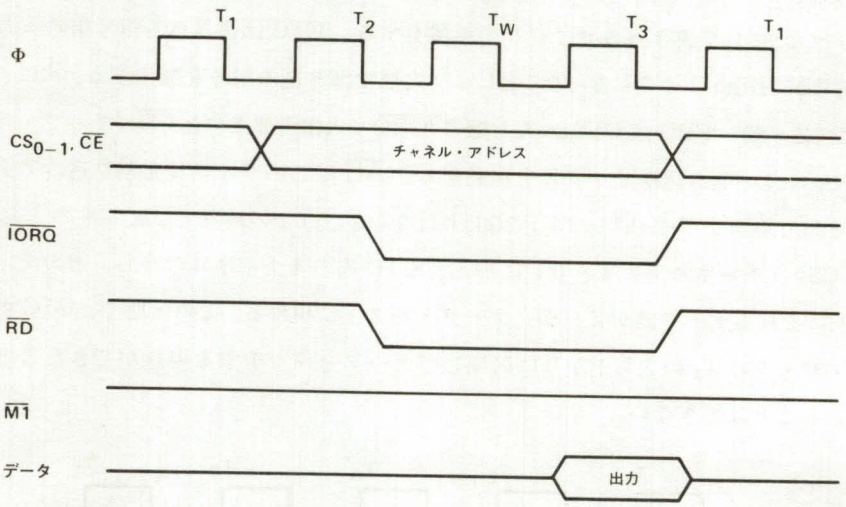


＊Z－80CPUにより自動的に挿入される。

図6－1　書き込み　サイクル

6．2　CTC 読み出しサイクル

図６－２にCTCの読み出しサイクルに関するタイミングを示す。このシーケンスはCPUが現在のダウン・カウンタの内容を読む場合にはいつも使用される。クロック・サイクルT_2の期間に、Z－80　CPUは入力端子$\overline{RD}$（読み出し）、$\overline{IORQ}$（入出力要求）、および$\overline{CE}$（チップ・イネーブル）を論理１にすることにより読み出しサイクルを起動する。この時点で同様に２ビットの２進コードをCTCの入力端子CS_1とCS_0（チャネル選択１、０）に加え、CTC内のチャネルのうちどのチャネルからデータを読み出すかを決める。（$\overline{M1}$は割り込みアクノリッジ・サイクルとこのサイクルを区別するために論理０でなければならない。）T_2サイクルの立ち上がり時点におけるダウン・カウンタの内容が、T_3サイクルの立ち上がり時点において、データ・バス上に現れる。外部から待ち状態を加えることはできない。



＊Z－80 CPUにより自動的に挿入される。

図６－２　読み出し　サイクル

6．3　CTCのカウンタおよびタイマ・モードのタイミング

図6－3にCTCのカウンタとタイマ・モードのタイミングを示す。

カウンタ・モードでは、CLK / TRG端子に接続されている外部回路から印加されるパルスのエッジ（この例では、立ち上がりアクティブ）により、システム・クロックΦに同期してダウン・カウンタの内容をデクリメント（－1）する。第8章のAC特性の項で規定されるように、このCLK / TRGは最小のパルス幅を確保しなければならないし、また、その周期はシステム・クロック周期の2倍より小さくてはいけない。CLK / TRGのアクティブ・エッジとΦの立ち上がりとの間のセット・アップ時間には必要な条件はないが、もし、CLK / TRGのエッジがある時間以下で立ち上がる（立ち下がる）と、ダウン・カウンタはΦについて1サイクル遅れてデクリメントする。ダウン・カウンタが1から0へデクリメントした直後に、ZC / TOは論理1のパルスを出力する。

タイマ・モードでは、CLK / TRG端子に印加されたトリガ・パルス（アクティブ〝High〟、アクティブ〝Low〟はユーザ側で選択できる）によりΦの2番目の立ち上がりからタイマ動作をイネーブルにする。カウンタ・モードと同様に、トリガ・パルスはΦと同期して検出され、最小パルス幅を確保しなければならない。タイマ動作はΦと同期して起動され、CLK / TRGのアクティブ・エッジと次のΦの立ち上がりとの間には最小のセット・アップ時間を要する。CLK / TRGのエッジがΦの立ち上がりに対してこの値よりも近くで起きた場合、タイマ動作の起動はΦに対して1サイクル遅れる。

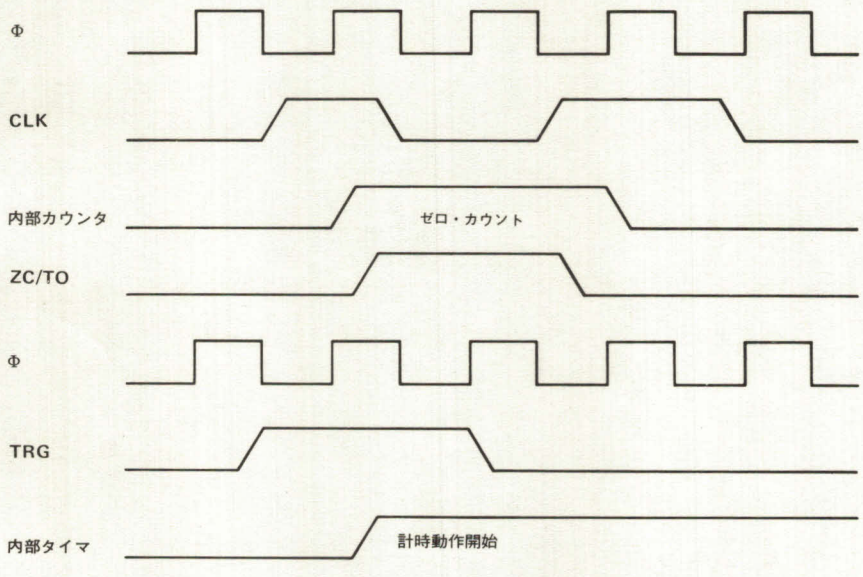


図6－3　カウンタ・タイマ動作タイミング

第7章　Z-80 CTC 割り込みサービス

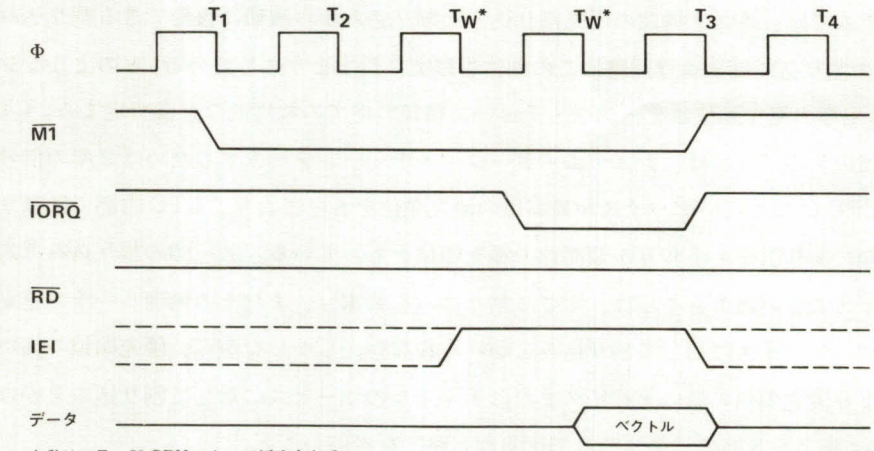
各CTCチャネルは個々にプログラムされ、対応するダウン・カウンタの内容が零に到達するごとに割り込みを要求する。CTCから割り込みを発生させる理由は、他の周辺デバイスと同様に、CPUに対して強制的に割り込み処理ルーチンを実行させるためである。このためにZ－80　CPUをモード2の割り込み応答をするようにプログラムしなければならない。この割り込みモードにおいては、CTCが割り込みを要求し、それが受け付けられた場合、対応する割り込み処理ルーチンの開始アドレスをメモリ内のテーブルから得るための16ビット・ポインタを作成しなければならない。このポインタの下位8ビットは、割り込みを要求しているあるチャネルに個有の割り込みベクトルという形でCTCから与えられる。（詳細は第1部 Z－80 CPU　テクニカル・マニュアル　の第8章を参照のこと。）

Z－80　システムでは、ネスト構造の優先割り込みと割り込みから適切に復帰できる割り込み方式を採用しているが、CTCの割り込み制御論理回路はこの動作を確実に行うようにしている。どのようなシステム・デバイスであれ、その処理の優先順位はデージー・チェーン接続の中での物理的な位置で定まる。CTCおよび他のすべてのZ－80周辺デバイスでは、システム・デージー・チェーンを形成するために2本の信号線IEI、IEOをもっている。CPUに最も近いデバイスが最高位の優先順位をもっており、CTC内部では割り込み優先順位はチャネル番号順であり、チャネル0が最高位の優先順位をもっている。Z－80の割り込み方式では、下位の優先順位のデバイスあるいはチャネルは、すでに割り込みを要求し、まだその処理ルーチンを完了していないより優先順位の高いデバイスに対して割り込みをかけられない。しかしながら、優先順位の高いデバイスまたはチャネルは、より優先順位の低いデバイスまたはチャネルのサービスに対して割り込みをかけることができる。（詳細は、第2章2．3節の〝割り込み制御論理回路〟を参照のこと。）

7．1節、7．2節では、割り込みアクノリッジ・サイクルおよび割り込み復帰サイクルにおけるCTC端子の時間関係を説明している。7．3節では、デージー・チェーンの割り込みサービスの代表例について説明する。

7．1　割り込みアクノリッジ・サイクル

図７－１に割り込みアクノリッジ・サイクルに関するタイミングを示す。CTC から割り込みを要求すると、CPU は割り込みアクノリッジ信号（$\overline{M1}$ と $\overline{IORQ}$）を選出する。デージー・チェーン接続の信号線（IEI、IEO）を安定にするのを保証するために、各チャネルは、$\overline{M1}$ がアクティブの期間、割り込み要求の状態を変更できない。$\overline{M1}$ は $\overline{IORQ}$ よりも約２クロック以前にアクティブになり、命令フェッチのサイクルと割り込みアクノリッジ・サイクルを区別するため、$\overline{RD}$ 信号はアクティブとならない。この期間に、CTC 内の割り込み制御回路は割り込みを要求している最高位のチャネルを決定する。CTC の割り込みイネーブル入力（IEI）がアクティブならば、$\overline{IORQ}$ がアクティブになると、CTC 内で割り込みを要求している最高位のチャネルからデータ・バス上に対応する割り込みベクトルが出力する。この期間に待ち状態（Tw*）が２個自動的に挿入され、デージー・チェーンの信号線を安定にする。さらに外部から待ち状態を追加してもよい。



＊ 自動的にZ－80 CPUによって挿入される。

図７－１　割り込みアクノリッジ・サイクル

7．2　割り込みサイクルからの復帰

図７－２にRETI命令に関するタイミングを示す。この命令は割り込み処理ルーチンの最後に使用され、ネスト構造の割り込み優先処理を適切に制御するためデージー・チェーンのイネーブル線を元の状態に戻す。CTCはRETIの２バイトの命令コードを内部でデコードし、次にサービスされるチャネルがあるかどうかを決定する。

いくつかのＺ－80周辺チップがデージー・チェーン接続されている場合、命令コードED_Hがデコードされる時点では、現在サービス中のチップ上でIEIはアクティブである。もし、次の命令コードが$4D_H$であれば、サービス中の周辺デバイスは再び元に復帰し、そのIEOはアクティブになる。外部から待ち状態を追加してもよい。

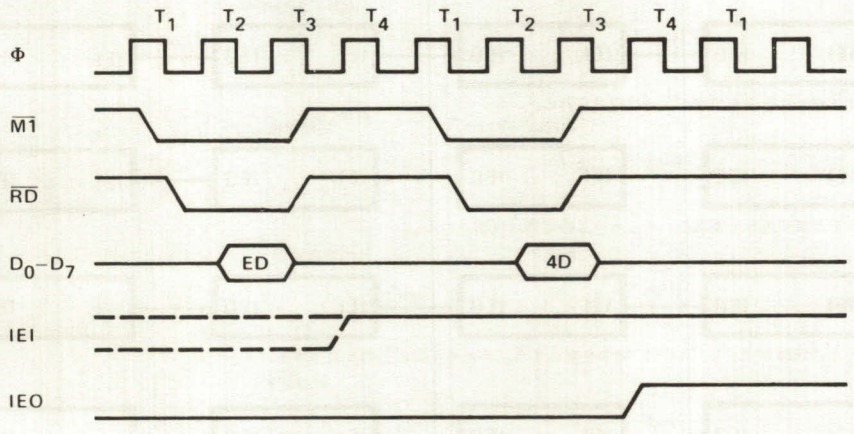


図７－２　割り込みサイクルからの復帰

7．3　デージー・チェーン割り込みサービス

図７－３にCTC内で生じる典型的なネスト構造の割り込みシーケースを示す。この例では、チャネル２から割り込みを発生し、サービスを受けているとしている。このチャネルのサービス中に、より優先順位の高いチャネル１から割り込みが発生し、そのサービスが受け付けられる。より優先順位の高いチャネルの割り込み処理が完了し、RETI命令を実行して、そのチャネルの処理ルーチンが完了したことを外部に対して知らせる。この時点で、より優先順位の低いチャネル２の割り込み処理ルーチンを再開する。

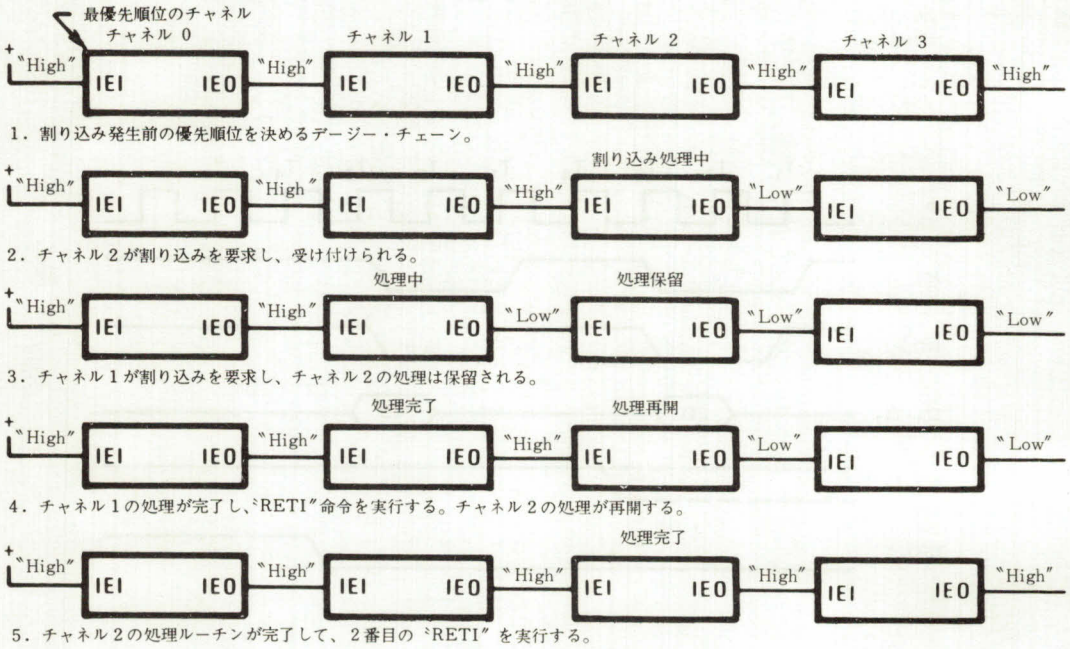


図７－３　デージー・チェーン割り込み処理

第8章　規　格

仕　　様

絶対最大定格

項　　目	記　号	定　格　値	単位
入 力 電 圧	V_{IN}	−0.3〜+7.0	V
出 力 電 圧	V_{OUT}	−0.3〜+7.0	V
動 作 温 度	Topr	0 〜+70	℃
保 存 温 度	Tstg	−65〜+150	℃

絶対最大定格の内のどの1項目でも、瞬時たりとも絶対最大定格値を越えないようにしてください。かつ、2項目以上の値が絶対最大定格値に、同時に達しないようにしてください。

電気的特性

DC特性

($Ta=0℃〜+70℃$, $V_{CC}=+5V\pm5\%$)

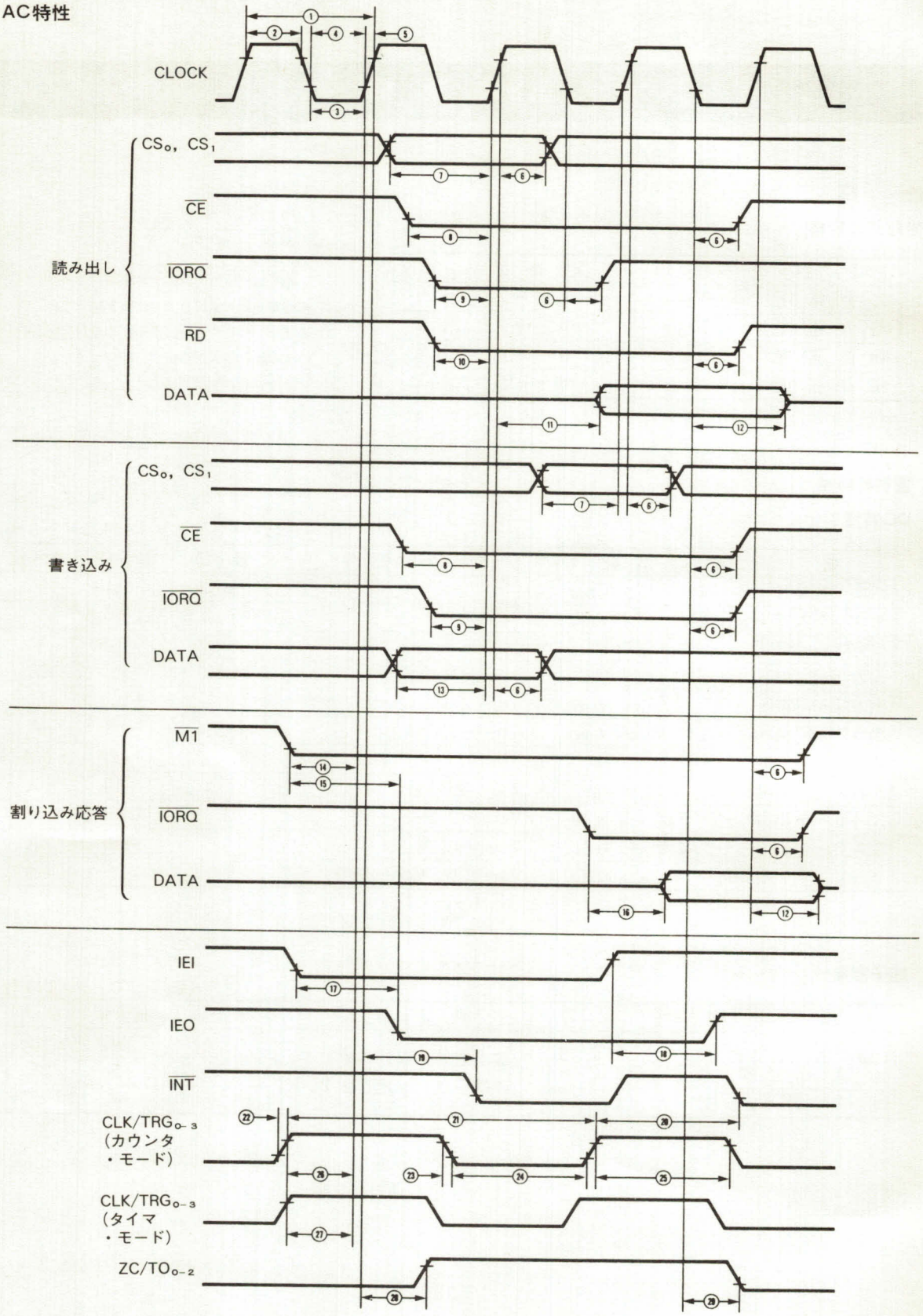
項　　目	記　号	測　定　条　件	最小値	最大値	単　位
クロック入力"Low"電圧	V_{ILC}		−0.3	+0.45	V
クロック入力"High"電圧	V_{IHC}		V_{CC}−0.6	V_{CC}+0.3	V
入力"Low"電圧	V_{IL}		−0.3	+0.8	V
入力"High"電圧	V_{IH}		+2.0	V_{CC}	V
出力"Low"電圧	V_{OL}	$I_{OL}=2mA$		+0.4	V
出力"High"電圧	V_{OH}	$I_{OH}=-250\mu A$	+2.4		V
消　費　電　流	I_{CC}			+120	mA
入力リーク電流	I_{LI}	$V_{IN}=0〜V_{CC}$		+10	μA
3ステート出力リーク電流	I_{LOH}	$V_{OUT}=2.4〜V_{CC}$		+10	μA
3ステート出力リーク電流	I_{LOL}	$V_{OUT}=0.4V$		−10	μA
ダーリントン駆動電流	I_{OHD}	$V_{OH}=1.5V$　$R_{EXT}=390\Omega$	−1.5		mA

端子容量

($Ta=25℃$, $f=1MHz$)

項　　目	記　号	測　定　条　件	最小値	最大値	単　位
クロック入力容量	C_{CLOCK}	被測定端子以外のすべての端子は、接地します。		20	pF
入　力　容　量	C_{IN}			5	pF
出　力　容　量	C_{OUT}			10	pF

AC特性


CLOCK
読み出し
CS_0, CS_1
$\overline{CE}$
$\overline{IORQ}$
$\overline{RD}$
DATA
書き込み
CS_0, CS_1
$\overline{CE}$
$\overline{IORQ}$
DATA
割り込み応答
$\overline{M1}$
$\overline{IORQ}$
DATA
IEI
IEO
$\overline{INT}$
CLK/TRG_{0-3}
(カウンタ
・モード)
CLK/TRG_{0-3}
(タイマ
・モード)
ZC/TO_{0-2}

番号	記　号	項　目	Z-80 CTC		Z-80A CTC		Z-80B CTC		単位	注※
			最小値	最大値	最小値	最大値	最小値	最大値		
1	TcC	クロック・サイクル時間	400	(注1)	250	(注1)	165	(注1)	ns	
2	TwCh	クロック・パルス幅(High)	170	2000	105	2000	65	2000	ns	
3	TwCℓ	クロック・パルス幅(Low)	170	2000	105	2000	65	2000	ns	
4	TfC	クロック↓時間		30		30		20	ns	
5	TrC	クロック↑時間		30		30		20	ns	
6	Th	保持時間	0		0		0		ns	
7	TsCS(C)	CSのクロック↑に対するセットアップ時間	250		160		100		ns	
8	TsCE(C)	$\overline{CE}$のクロック↑に対するセットアップ時間	200		150		100		ns	
9	TsIO(C)	$\overline{IORQ}$↓のクロック↑に対するセットアップ時間	250		115		70		ns	
10	TsRD(C)	$\overline{RD}$↓のクロック↑に対するセットアップ時間	240		115		70		ns	
11	TdC(DO)	クロック↑からデータ出力までの遅延時間		240		200		130	ns	2
12	TdC(DOz)	クロック↓からデータ出力フロートまでの遅延時間		230		110		90	ns	
13	TsDI (C)	入力データのクロック↑に対するセットアップ時間	60		50		40		ns	
14	TsM1(C)	$\overline{M1}$↓のクロック↑に対するセットアップ時間	210		90		70		ns	
15	TdM1(IEO)	$\overline{M1}$↓からIEO↓までの遅延時間(M1前の割り込み時間)		300		190		130	ns	3
16	TdIO(DOI)	$\overline{IORQ}$↓からデータ出力までの遅延時間(割り込み応答サイクル)		340		160		110	ns	2
17	TdIEI (IEOf)	IEI↓からIEO↓までの遅延時間		190		130		100	ns	3
18	TdIEI (IEOr)	IEI↑からIEO↑までの遅延時間(EDデコード後)		220		160		110	ns	3
19	TdC(INT)	クロック↑から$\overline{INT}$↓までの遅延時間		TcC+200		TcC+140		TcC+120	ns	4
20	TdCLK(INT)	CLK/TRG↑から$\overline{INT}$↓までの時間(CTR(C)を満たしたとき)		(19)+(26)		(19)+(26)		(19)+(26)	ns	5, 6
		CLK/TRG↑から$\overline{INT}$までの時間(CTR(C)を満たさなかったとき)		(1)+(19)+(26)		(1)+(19)+(26)		(1)+(19)+(26)	ns	5, 6
21	TcCTR	CLK/TRGサイクル時間	2TcC		2TcC		2TcC		ns	5
22	TrCTR	CLK/TRG↑時間		50		50		40	ns	
23	TfCTR	CLK/TRG↓時間		50		50		40	ns	
24	TwCTRℓ	CLK/TRGのパルス幅(Low)	200		200		120		ns	

番号	記　号	項　目	Z-80 CTC		Z-80A CTC		Z-80B CTC		単位	注※
			最小値	最大値	最小値	最大値	最小値	最大値		
25	TwCTRh	CLK/TRGのパルス幅 (High)	200		200		120		ns	
26	TsCTR(Cs)	直接カウントするときのCLK/TRG↑のクロック↑に対するセットアップ時間	300		210		150		ns	5
27	TsCTR(Ct)	次のクロック↑でプリスケーラをイネーブルにするためのCLK/TRG↑のクロック↑に対するセットアップ時間	210		210		150		ns	4
28	TdC(ZC/TOr)	クロック↑からZC/TO↑までの遅延時間		260		190		140	ns	
29	TdC(ZC/TOf)	クロック↑からZC/TO↓までの遅延時間		190		190		140	ns	

↑は立ち上がりエッジ、↓は立ち下がりエッジを示します。

〔A〕 2.5TcC＞(n－2)TdIEI (IEOf)＋TdM1(IEO)＋TsIEI (IO)

＋TTLバッファ使用の場合、その遅延時間

〔B〕 $\overline{\text{RESET}}$の入力幅は最低3クロック・サイクル間必要です。

(注1) TcC＝TwCh＋TwCℓ＋TrC＋TfC

(注2) 負荷容量の50pF増加につき遅延は10nS増加します。負荷容量の最大値は、データ・バスが200pFであり、他は、100pFです。

(注3) 負荷容量の10pF増加につき遅延は、2nS増加します。負荷容量の最大値は100pFです。

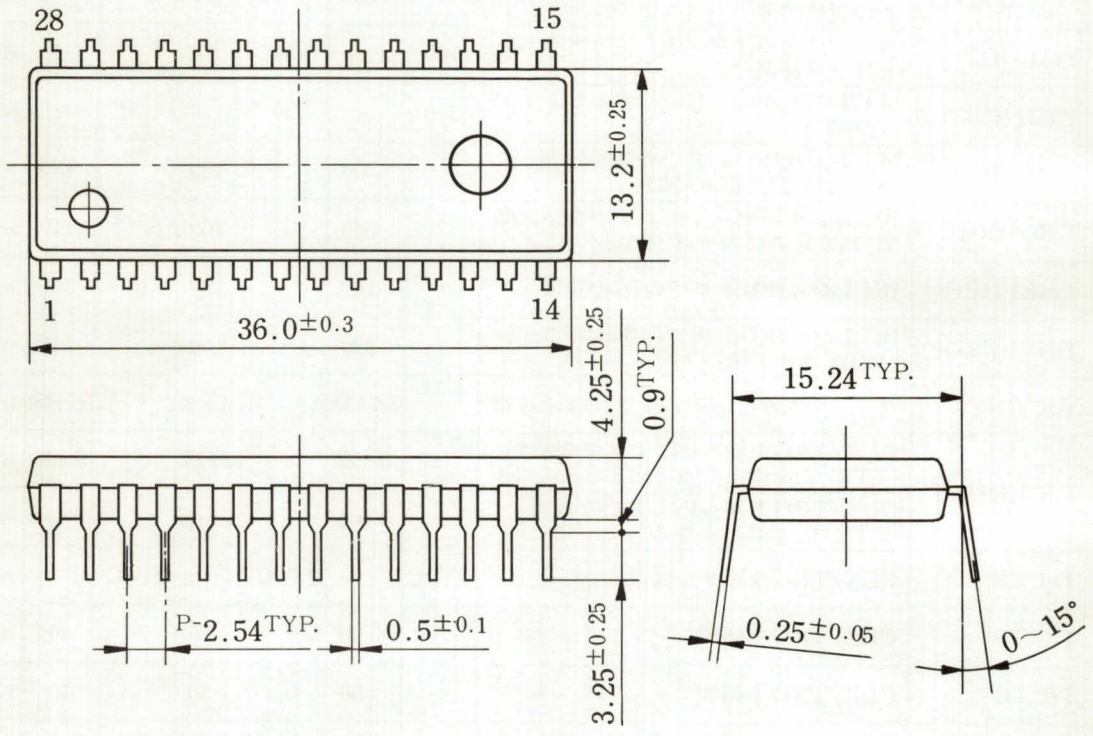
(注4) タイマ・モード時

(注5) カウンタ・モード時

(注6) ㉖は最小値でなく、実際の値。

※すべてのタイミングは暫定であり変更することがあります。

外形寸法図



単位：mm

(おことわり) 製品の改良のため予告なしに内容の一部を変更することがあります。

- Z-80ファミリについてのデータとか，最新情報は，下記にお問い合わせ下さい。

（おことわり）

○本書に記載された説明事項、規格、回路例などは、シャープZ－80ファミリの使用上の参考として示されたものであり、別途用意の製品仕様書の記載内容が本書の内容に優先します。LSIご使用に当たっては、必要に応じて製品仕様書をご用命のうえ内容をご確認ください。また規格契約を必要とする場合には、製品仕様書をもってご契約ください。

○製品の改良のため予告なしに内容の一部を変更することがあります。

シャープ㈱株式会社

本　　社	〒545	大阪市阿倍野区長池町22番22号	☎(06) 621-1221(大代表)
IC事業本部	〒632	奈良県天理市櫟本町2613番地の1	☎(07436)5-1321(大代表)
国内電子部品営業本部	〒545	大阪市阿倍野区長池町22番22号	☎(06) 621-1221(大代表)
東部地区営業	〒162	東京都新宿区市谷八幡町8番地	☎(03) 260-1161(大代表)
長野駐在	〒399	松本市芳野8番14号	☎(0263)27-1677
北関東駐在	〒361	埼玉県行田市門井町2丁目5番地	☎(0485)53-3127
静岡駐在	〒422	静岡市曲金6丁目8番44号	☎(0542)83-0081(代　表)
中部地区営業	〒454	名古屋市中川区山王3丁目5番5号	☎(052)332-2681(代　表)
浜松駐在	〒430	浜松市植松町1476の2	☎(0534)65-1207(代　表)
西部地区営業	〒545	大阪市阿倍野区長池町22番22号	☎(06) 621-1221(大代表)

Z-80 テクニカルマニュアル

1979年5月　　第1版第1刷発行
1985年1月　　第5版第1刷発行

SHARP

シャープ株式会社

本	社	〒545	大阪市阿倍野区長池町22番22号 ☎(06) 621-1221(大代表)
IC事業本部		〒632	奈良県天理市櫟本町2613番地の1 ☎(07436)5-1321(大代表)
国内電子部品営業本部		〒545	大阪市阿倍野区長池町22番22号 ☎(06) 621-1221(大代表)
東部地区営業		〒162	東京都新宿区市谷八幡町8番地 ☎(03) 260-1161(大代表)
長野駐在		〒399	松本市芳野8番14号 ☎(0263)27-1677
北関東駐在		〒361	埼玉県行田市門井町2丁目5番地 ☎(0485)53-3127
静岡駐在		〒422	静岡市曲金6丁目8番44号 ☎(0542)83-0081(代 表)
中部地区営業		〒454	名古屋市中川区山王3丁目5番5号 ☎(052)332-2681(代 表)
浜松駐在		〒430	浜松市植松町1476の2 ☎(0534)65-1207(代 表)
西部地区営業		〒545	大阪市阿倍野区長池町22番22号 ☎(06) 621-1221(大代表)

昭和60年2月作成 電部-1552 Z2(7)