

第2章

DMAの基本

DMAとは／H8のDMAC／フル・アドレス・モードの利用例／ ショート・アドレス・モードの利用

<概要>

この章ではDMAの基本事項について解説します。DMAを利用することにより、ブロック状の多量のデータ転送の際に、プログラムの負担が軽減されますし、転送スピードも向上できます。具体的な応用例については、SCIへの応用などでも紹介します。

2.1 DMAとは

まず、DMA (Direct Memory Access) とはどんなもののかの概要を理解しておきましょう。その前に今まで行っていたデータの入出力方法について見なおしておきます。

2.1.1 プログラムによる入出力

メモリからデータを読み取るときの操作をアセンブラ言語で行う場合には、図2.1に示すように、

```
mov.b @add1, r01
```

などとし、ここでは絶対アドレッシング・モードによって示しましたが、他のアドレッシング・モードでも同様に、mov命令によって行います。オペランドはメモリのアドレスを示すソース・オペランドと、レジスタを示すデスティネーション・オペランドです。また逆に、レジスタにあるデータをメモリに格納するには、

```
mov.b r01, @add2
```

などとし、

H8はメモリ・マップドI/OのMPUですから、出力装置(レジスタ)にデータを出力する際にも、

```
mov.b r01, @ioad
```

などとし、メモリへの読み書きと同じです。入力の際にはオペランドが反対になります。このようにプログラムにより、データの入出力をする手法をPCIO(Program Controlled IO)などと呼ぶこともあります。

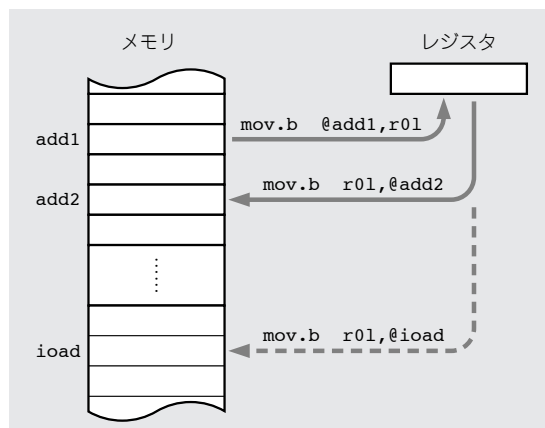


図2.1 プログラム制御のデータ移動

2.1.2 入出力のメカニズム

では実際に入出力をしているときのバスの状態、制御信号の動きはどうなっているのでしょうか。

図2.2にバス幅を8ビットで使用したときの、3ステート(three state)の読み出しサイクルのタイミング・チャートを示します。命令語を読み出すときや、前述の、

```
mov.b    @add1, r01
```

などの命令を実行したときのタイミングです。3ステートのサイクルで示しています。内部メモリのアクセスや外部アクセスでは、WCR(Wait Control Register)のウェイト・カウント(wait count)を'0'にしているときの状態です。

このときには、読み出し元のアドレスがアドレス・バスに出力され(①)、そのアドレス情報が有効になったタイミングでAS(Address Strobe)信号が真になります(②)。メモリは、そのアドレス情報で自分が指定されたことを知ると、情報の読み出しを行います。

そして、RD(Read)信号が出力されると、その読み出した情報をデータ・バスの上に載せます。このときにバス・ドライバでは、3ステートのバスを'0'か'1'の能動状態にします。そしてRD信号の後縁(③)でMPUに読み取られます。つまりH8に接続されたメモリでは、アドレスが有効になって、RD信号が有効になると、自分のデータをデータ・バスに載せるというメカニズムで設計されています。これらのASやRD信号は、命令を読み出さなければならないタイミングや、読み出した命令を解釈した結果、バスを制御する信号としてMPUから出力されています。

いっぽう、メモリへの書き込みはどうでしょう。図2.3に、バス幅を8ビットで使用したときの、3ステートの書き込みサイクルのタイミング・チャートを示します。前述の、

```
mov.b    r01, @add2
```

命令などを実行したときのタイミングです。

ここでは、書き込み対象のアドレスがアドレス・バスに出力され(①)、そのアドレスが有効になると、AS信号が出力されます。そして、データ・バスの上位8ビット(D15-D8)には、書き込むべき情報も出力されています(②)。この情報が充分安定したタイミングで、HWR(High Write)信号も真になります。メモリや出力装置ではHWRの後縁(③)で、データ・バスの内容を確保します。ここでは8ビット・モードを例

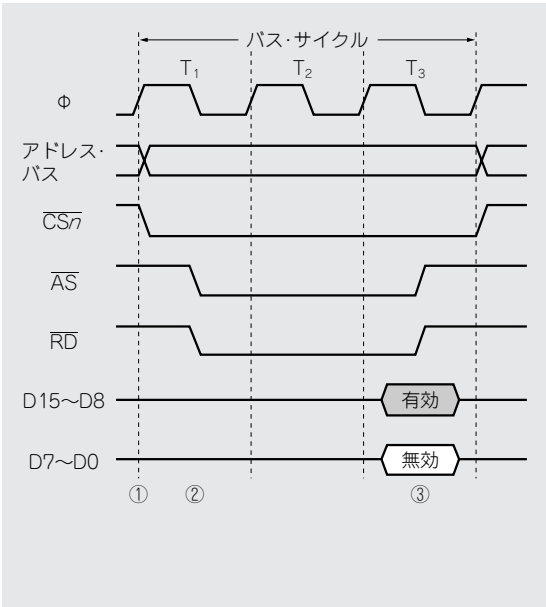


図2.2 8ビット3ステート・リード・サイクル

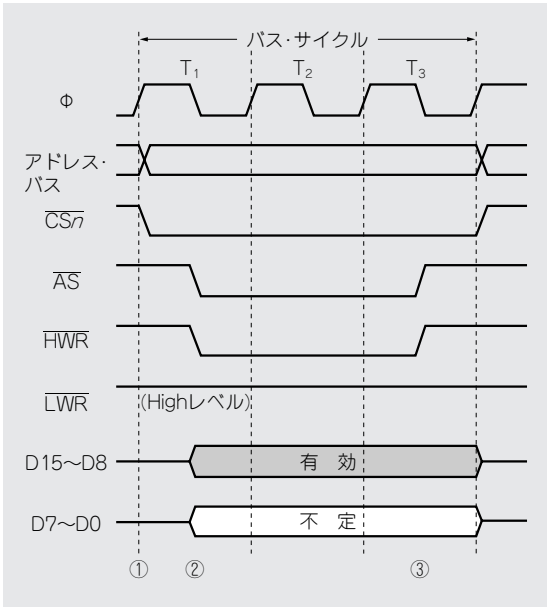


図2.3 8ビット3ステート・ライト・サイクル



としたので、LWR(Low Write)は出力されませんし、データ・バスの下位8ビット(D7-D0)の内容は無効です。16ビット・モードで利用する場合には、両者も有効になります。

書き込みのサイクルでは、H8に接続されたメモリにおいては、ASとHWR(16ビット・モードではHWRとLWR)の制御信号により、書き込み動作をするように作られています。これらの制御信号もMPUによって作り出されてバスに出力されています。

2.1.3 複数バイトの入出力

このように命令語による1バイトあるいは2バイト単位の転送を、複数バイト分行うことを考えてみます。H8の命令セットには、EEPMOVという命令があり、複数バイトを転送する場合に利用できます。EEPMOVはEEPが示すようにシステム・メモリの一部に電氣的にプログラムできるROMがある場合の転送命令です。

ここではあえてEEPMOVを利用しないで複数バイトの転送をする例を考えてみます。これは、あるコンパイラが翻訳した結果ですが、そのプログラム例をリスト2.1に示します。ROM部からRAM部などへの定数の転送などに利用するアセンブラの関数です。

C言語から利用する場合の関数の引き数は、

```
char *bmove ( char *src, char *dest, int n );
```

です。転送(実際は複写)する元のポインタと転送先のポインタ、そして転送バイト数です。関数の戻り値は、転送先のポインタとしています。C言語とのインターフェースとしていますので、引き数はスタック領域にあるものとします。

当然のことながら、実際のデータ読み出し/書き込みの他に転送したバイト数を計数する部分、転送元、転送先のアドレスを更新する部分が必要となります。これらの命令群の実行サイクル総数を数えると、リスト2.1の例ではH8の内部メモリだけとして、すべて3ステート(クロック・サイクル)としても、指定 n バイトの転送に、 $66n + 108$ ステートを要し、仮に256バイトの転送に、25MHzクロックで約 $680\mu\text{s}$ となります。

H8はメモリ・マップドI/O方式です。もし転送先が、入出力だと4ステート以上になりますし、外部領域のメモリや入出力であれば、応答速度の面からwaitサイクルが必要な場合もありますから、それ以上の時間がかかるということになります。となると、ハード・ディスクや高速の通信回線であった場合には、プログラムは転送作業に専念しても、データの供給や受信が間に合わないという事態も予想されます。

そこで考えだされたのが直接メモリをアクセスするDMA(Direct Memory Access)方式です。

リスト2.1 ブロック転送関数

```
_bmove:
    push.l   er6
    mov.l    er7,er6
    sub      #12,sp
    mov.l    er0,@(-4,er6)
    mov.l    er1,@(-8,er6)
    mov.w    r2,@(-10,er6)
    sub.w    r2,r2
    mov.w    r2,@(-12,er6)

.L4:
    mov.w    @(-12,er6),r2
    mov.w    @(-10,er6),r3
    cmp.w    r3,r2
    blt      .L7
    bra      .L5

.L7:
    mov.l    @(-4,er6),er2
    mov.l    @(-8,er6),er3
    mov.b    @er3,r0l
    mov.b    r0l,@er2

.L6:
    mov.w    @(-12,er6),r2
    mov.w    r2,r3
    adds     #1,er3
    mov.w    r3,@(-12,er6)
    bra      .L4

.L5:
.L3:
    add      #12,sp
    pop.l    er6
    rts
```

2.1.4 DMAによる入出力

前述のように、メモリからの情報の読み出しと書き込みは、通常はMPUから出力されるASとRDやHWR, LWR信号によって制御されています。これらが管制信号、あるいは交通整理信号の役割をして、データ・バス内の情報の流れ方向や行き先を制御しているということになります。つまりアドレス情報と、この3種の制御信号が整えば、データの読み書きはできるということになります。