[第2章]

ワンチップ・マイコンとして機能が豊富で高速！

AVRのコアを知る

この章では，AVRを動作させる上で必要なAVRの基本事項を解説しますが，どうしてもプログラム・コードを入れなくてはならない部分も出てきます．言語に関しては次章で説明しますので，話が相前後してしまいますが，ご了承願います．

2-1　メモリ・マッピング

前章で述べたように，AVRは1クロック1インストラクションという高性能なMCUです．しかし，このことを実現するには一般的なメモリをアクセスするタイミングではできないため，プログラムはプログラム専用のフラッシュ・メモリにおいてしまうという割り切った考え方をしています．通常のコンピュータであればプログラム・コードはすべてのメモリ・スペースにおくことができますが，そのような構造にするとメモリのアクセスにアドレスとデータのフェーズが必要なため，最低でも2クロック必要になってしまいます．もしこの構造で1クロック1インストラクションを実現しようとすると，パイプラインをおく必要が出てきます．AVRはパイプラインをおかないシンプルな構造での高速化を選択したようです．

この構造を選択したことによるデメリットもあります．

ディスクやコンパクト・フラッシュなどに置いた外部記憶装置から，プログラムをRAMなどにロードし，必要なときに必要なプログラムを走らせるようなアプリケーションには不向きになってしまったということです．AVRにはセルフ・プログラミングという機能があり，フラッシュ・プログラムのブート・ブロックという場所におかれたプログラムを走らせて，アプリケーション・ブロックと呼ばれるフラッシュ・メモリにデータやプログラムを書き込むことができる機能も用意されています．しかし，書き込みには時間もかかりますし，フラッシュ・メモリの書き換え回数は約1万回ということですから，システム立ち上げ時にプログラムをロードしてフラッシュに書き込み，動作中に何度も異なるプログラムをロードさせるというようなアプリケーションには不向きで，そのような用途にはもっと適切なMCUが存在すると思われます．

図2-1がAVRのプログラムから扱えるメモリ・マップ(レジスタを含む)です．フラッシュ・プログラム・メモリは16ビット，2バイトで構成されています．データ・メモリ・スペースは8ビット幅で構成されており，このスペースに汎用レジスタ・ファイル，I/Oレジスタ，内蔵SRAM，外部拡張メモリ・スペースなどが割り当てられています．

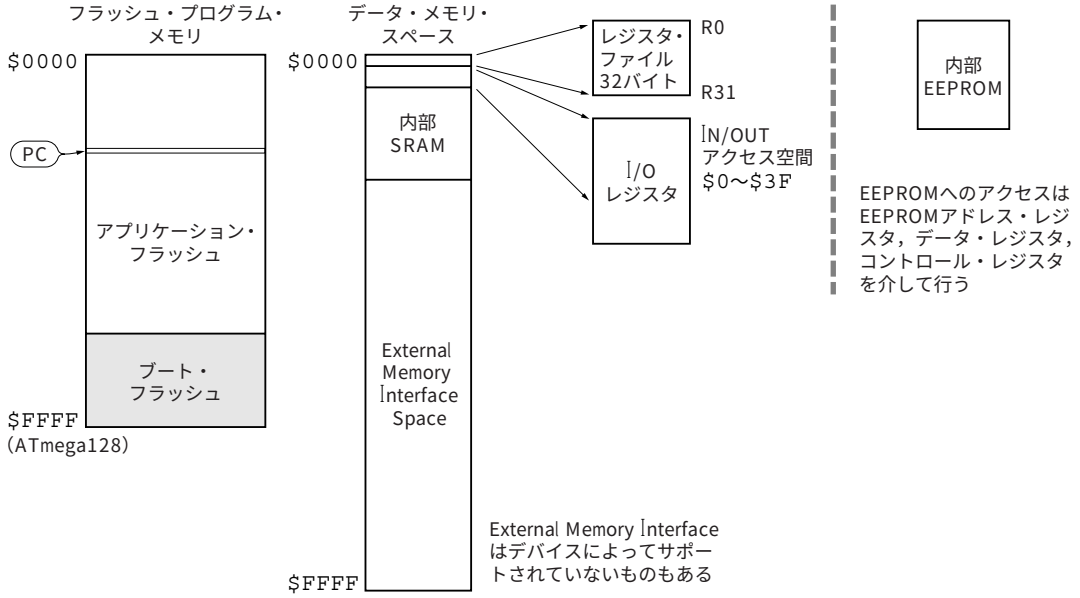

図2-1　メモリ・マップ

内蔵EEPROMはこのデータ・メモリ・スペース上にはなく，I/Oレジスタを介してアクセスします．

(1) フラッシュ・プログラム・メモリ

現在，128 Kワード(256 Kバイト)，$0000番地から$1FFFF番地までの番地が割り振られたAVRがあります．データ・シートやマニュアルによって16進表記を$12AFなどと表現している場合と0x12AFなどと表現している場合がありますが，どちらも16進数を表しています．UNIXやC言語の表記では0xが用いられ，マイコンの表現でとくにモトローラ系で$が用いられていると思います．

AVRの表記ではどちらも使われているようです．フラッシュ・メモリは$，データ・メモリは0xなどということではありません．2バイトで1ワードを表しているので，64 Kワード・スペースのATmega128では，0x0000～0xFFFFがフラッシュ・メモリのアドレス範囲となります．最新のATmega2560は256 Kバイトのメモリ・スペースをもつので，0x00000～0x1FFFFの範囲になります．フラッシュ・メモリ・スペースは，ブート・ブロックとアプリケーション・ブロックに分割できます．アプリケーション・ブロックにブート・ブロックのプログラムからデータを書き込むことができます．すなわち，ブート・ブロックの内容を読みながら，アプリケーション・ブロックへデータを書き込むこと(Read While Write)が可能になっています．

このことは，別の言葉でセルフ・プログラミングと言っています．とくに書き込み器を用いなくても，AVRチップ自体がフラッシュに対して書き込むことができるわけです．ただし書き込みのためのプログラムは，書き込み器を用いて，ブート・ブロックにあらかじめプログラムしておく必要があります．したがって，この機能を使うことでシステム上のアプリケーションの更新などが，書き込み器を用いなくても可能になります．たとえば，遠隔地にある装置のプログラムを通信回線から自動で

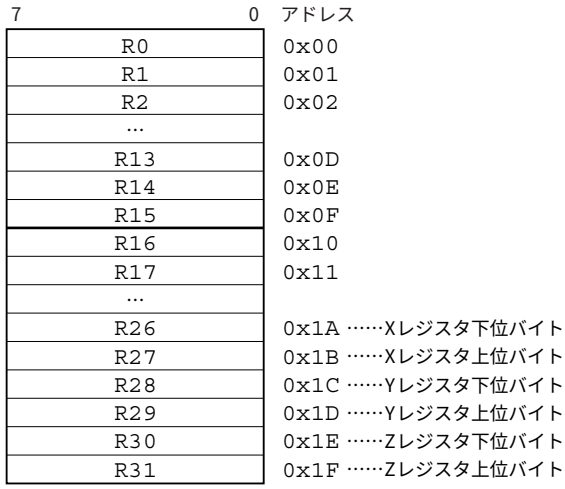

図2-2　汎用レジスタ・ファイル

更新させることもできます.

ブート・ブロックのブロック・サイズはヒューズ・ビット(**表2-3**参照)BOOTSZの設定で選択が可能です.また,システムにリセットがかかったときにアプリケーションがいきなり立ち上がるか,ブート・ブロックに置かれているモニタ・プログラムから立ち上がるかの設定も,ヒューズ・ビットBOOTRSTのプログラムで設定が可能です(詳細は4-10項で解説).

AVRではどのようにフラッシュ・メモリとSRAMエリアとを区別しているのでしょうか? AVRでは,プログラムはすべてフラッシュ・プログラムにおかれているので,プログラム・カウンタの示しているアドレスはフラッシュ・プログラム・メモリを示していることになります.通常のデータのアクセスは,すべてデータ・メモリ・スペースになります.

それでは,(フラッシュ)プログラム・メモリ・スペースにあるコンスタント(定数)データはどう扱うのかというと,特別な命令を用います.LPM命令でコンスタント・データを読み出すことができ,SPM命令でデータを書き込むことができますが,SPMは不用意な書き込みを禁じた特殊な使い方をしなければなりません〔第4章4-10 フラッシュ・メモリのセルフ・プログラミングのRWW(フラッシュ書き込み中にフラッシュを読む)参照〕.そのほかに,プログラム・コード中に置かれる即値(イミディエート・データ)なども扱うことができます.これらについては,第3章 アドレッシング・モードを参照してください.

(2) 汎用レジスタ・ファイル(図2-2)

データ・メモリ・スペースの先頭0x0000から0x001Fまでの32バイトが,汎用レジスタとして割り当てられています.通常R0からR31として表記されます.また特別なレジスタとして,X,Y,Zレジスタが割り当てられています.R26,R27がXレジスタ,R28,R29がYレジスタ,R30,R31がZレジスタです.これらは,間接アドレッシング・モードなどで使われます.

AVRでは,ほかのマイコンと同様に,新しい製品とともに搭載するメモリ容量が増加してきているので,アドレス16ビットではすべてのアドレス範囲を示すことができなくなってきました.ATmega2560などでは,すでにアドレス幅として17ビットが必要になってきています.このようなケースのた

● RAMPZレジスタ

7	6	5	4	3	2	1	0	
RAMPZ7	RAMPZ6	RAMPZ5	RAMPZ4	RAMPZ3	RAMPZ2	RAMPZ1	RAMPZ0	RAMPZ
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ELPM命令およびSPM命令では，ZポインタはRAMPZレジスタと結合されて使用される

RAMPZ	ZH	ZL

● EINDレジスタ

7	6	5	4	3	2	1	0	
EIND7	EIND6	EIND5	EIND4	EIND3	EIND2	EIND1	EIND0	EIND
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

EICALLおよびEIJMP命令では，ZポインタはEINDレジスタと結合されて使用される

EIND	ZH	ZL

図2-3　16ビット以上のアドレスを扱うためのRAMPZ/EIND レジスタ

めに，Zレジスタに関しては，拡張Zポインタ・レジスタとしてZレジスタと結合して使用するRAMPZレジスタがあります(**図2-3**)．これらはELPMおよびSPM命令で使用されます．LPMでは影響をうけません．

同様に拡張間接アドレス・レジスタEINDがあり，Zレジスタと結合されて使用されます．EICALL/EIJMP命令で使用されます．このケースでもICALL/IJMPには影響しません．

(3) I/Oレジスタ

メモリ・スペース0x0020番地から0x005F番地にあるI/Oアドレス0x00～0x3Fは，直接IN/OUT命令でアクセスすることができます．

メモリ・スペース0x60以降の拡張(Extended)I/Oスペースは，IN/OUT命令の5ビットで指定できる範囲を超えているため，ST/STS/STD，LD/LDS/LDDなどの命令を使わなければなりません．このときI/Oアドレスに0x0020を加えたアドレスが実際のアドレスになります．

(4) 内部SRAM

内部SRAMの領域は，汎用レジスタ・ファイルとI/Oレジスタの次に置かれています．先頭は使用するAVRの品種によってI/Oレジスタの領域が異なるので，それぞれのデバイスごとに異なり，大きさも異なります(Appendix A参照)．

この領域は，主にデータのワーキング・エリアやスタックとして利用されます．AVRでは，この領域にプログラムを置くことはできません．

(5) スタック・ポインタ(図2-4)

スタックは，割り込みやサブルーチン・コールでコントロールがサービス・プログラムへ渡されたとき，戻り先番地やローカル変数を蓄えるために使用されます．ポインタで指す実際の戻り番地は内部SRAM領域に入っています．

(6) 外部(拡張)メモリ・スペース(図2-5)

拡張メモリ・スペースは，ポートをアドレス/データ・バスに割り当て，外付けの拡張SRAM/ROM/周辺デバイスなどにインターフェースする機能ですが，ごく一部のAVRにだけ用意されています．この機能を使う場合はデバイスの選定に気をつける必要があります．