2. CPU

第2章　目次

2.1　概要

Ｈ８／３００Ｈ　ＣＰＵは、Ｈ８／３００ＣＰＵの上位互換のアーキテクチャを持つ内部32ビット構成の高速ＣＰＵです。Ｈ８／３００Ｈ　ＣＰＵは、16ビット×16本の汎用レジスタを持ち、16Ｍバイトのリニアなアドレス空間を取り扱うことができ、リアルタイム制御に最適です。

2.1.1　特長

Ｈ８／３００Ｈ　ＣＰＵには、次の特長があります。

■ Ｈ８／３００ＣＰＵ上位互換
　Ｈ８／３００シリーズのオブジェクトプログラムを実行可能
■ 汎用レジスタ方式
　16ビット×16本（８ビット×16本、32ビット×８本としても使用可能）
■ 62種類の基本命令
　・８／16／32ビット転送、演算命令
　・乗除算命令
　・強力なビット操作命令
■ ８種類のアドレッシングモード
　・レジスタ直接（Ｒｎ）
　・レジスタ間接（@ＥＲｎ）
　・ディスプレースメント付レジスタ間接（@（d:16，ＥＲｎ），@（d:24，ＥＲｎ））
　・ポストインクリメント／プリデクリメントレジスタ間接（@ＥＲｎ＋／@－ＥＲｎ）
　・絶対アドレス（@aa:8，@aa:16，@aa:24）
　・イミディエイト（#xx:8，#xx:16，#xx:32）
　・プログラムカウンタ相対（@（d:8,ＰＣ），@（d:16，ＰＣ））
　・メモリ間接（@@aa:8）
■ 16Ｍバイトのリニアアドレス空間
■ 高速動作
　・頻出命令をすべて２～４ステートで実行
　・最高動作周波数：18MHz／16MHz（フラッシュメモリ版の場合）
　・８／16／32ビットレジスタ間加減算　111ns／125ns（フラッシュメモリ版の場合）
　・８×８ビットレジスタ間乗算　778ns／875ns（フラッシュメモリ版の場合）
　・16÷８ビットレジスタ間除算　778ns／875ns（フラッシュメモリ版の場合）
　・16×16ビットレジスタ間乗算　1,221ns／1,375ns（フラッシュメモリ版の場合）
　・32÷16ビットレジスタ間除算　1,221ns／1,375ns（フラッシュメモリ版の場合）
■ ２種類のＣＰＵ動作モード
　・ノーマルモード（Ｈ８／３０４８シリーズでは使用できません）
　・アドバンストモード

■ 低消費電力動作

ＳＬＥＥＰ命令により低消費電力状態に遷移

2.1.2　Ｈ８／３００ＣＰＵとの相違点

Ｈ８／３００Ｈ　ＣＰＵは、Ｈ８／３００ＣＰＵに対して、次の点が強化、拡張されています。

■ 汎用レジスタを拡張

16ビット×８本の拡張レジスタを追加

■ アドレス空間を拡張

・アドバンストモードのとき、最大16Mバイトのアドレス空間を使用可能

・ノーマルモードのとき、Ｈ８／３００ＣＰＵと同一の64kバイトのアドレス空間を使用可能
（Ｈ８／３０４８シリーズでは使用できません）

■ アドレッシングモードを強化

16Mバイトのアドレス空間を有効に使用可能

■ 命令強化

・32ビット転送、演算命令を追加

・符号付き乗除算命令などを追加

2.2　ＣＰＵ動作モード

Ｈ８／３００Ｈ　ＣＰＵは、ノーマルモードおよびアドバンストモードの２つのＣＰＵ動作モードをもっています。サポートするアドレス空間は、ノーマルモードの場合最大64ｋバイト、アドバンストモードの場合最大16Mバイトとなります。

本ＬＳＩでは、アドバンストモードのみ使用できます。（以後、特に説明がない場合はアドバンスモードについて説明します。）

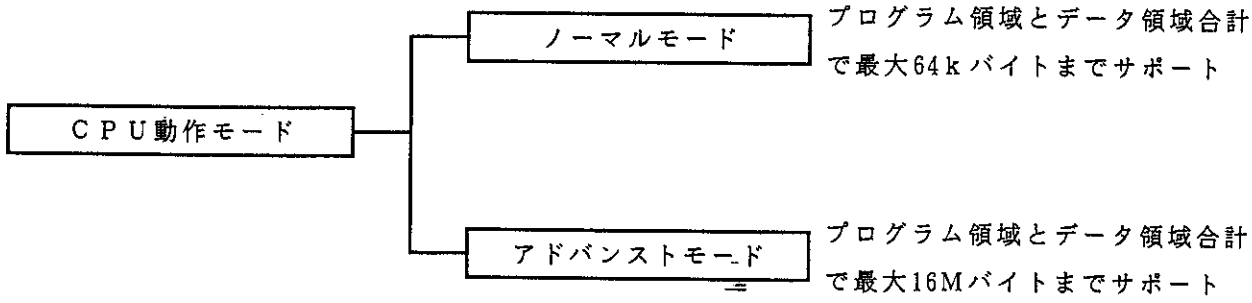


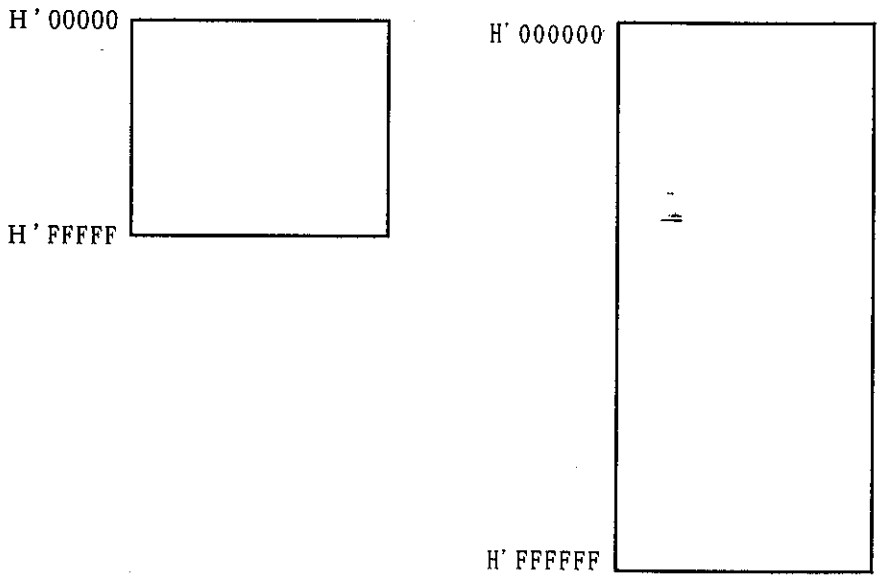
図２.１　ＣＰＵ動作モード

2.3　アドレス空間

Ｈ８／３００Ｈ　ＣＰＵのアドレス空間は最大16Mバイトです。Ｈ８／３０４８シリーズではＭＣＵ動作モードにより、アドレス空間は、１Ｍバイトモードと16Mバイトモードを選択できます。

本ＬＳＩのメモリマップの概要を図2.2に示します。詳細は「3.6　各動作モードのメモリマップ」を参照してください。

アドレス空間が１Ｍバイトモードの場合、実効アドレスの上位４ビットは無視され、20ビットのアドレスとなります。



(a)　１Ｍバイトモード　　(b)　16Mバイトモード

図2.2　メモリマップ

2.4　レジスタ構成

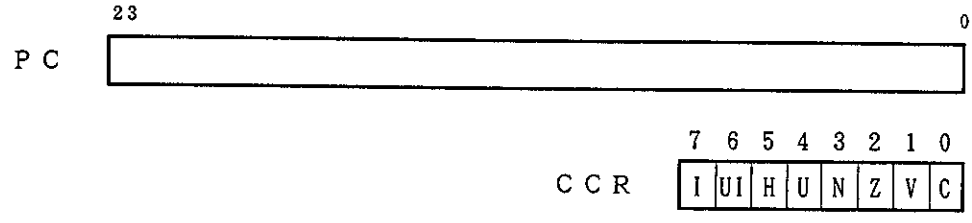
2.4.1　概要

Ｈ８／３００Ｈ　ＣＰＵの内部レジスタ構成を図2.3に示します。これらのレジスタは、汎用レジスタとコントロールレジスタの２つに分類することができます。

汎用レジスタ（ＥＲｎ）

	15 0	7 0	7 0
ＥＲ０	Ｅ０	Ｒ０Ｈ	Ｒ０Ｌ
ＥＲ１	Ｅ１	Ｒ１Ｈ	Ｒ１Ｌ
ＥＲ２	Ｅ２	Ｒ２Ｈ	Ｒ２Ｌ
ＥＲ３	Ｅ３	Ｒ３Ｈ	Ｒ３Ｌ
ＥＲ４	Ｅ４	Ｒ４Ｈ	Ｒ４Ｌ
ＥＲ５	Ｅ５	Ｒ５Ｈ	Ｒ５Ｌ
ＥＲ６	Ｅ６	Ｒ６Ｈ	Ｒ６Ｌ
ＥＲ７	Ｅ７（ＳＰ）	Ｒ７Ｈ	Ｒ７Ｌ

コントロールレジスタ（ＣＲ）



《記号説明》

ＳＰ　：スタックポインタ
ＰＣ　：プログラムカウンタ
ＣＣＲ：コンディションコードレジスタ
Ｉ　　：割込みマスクビット
ＵＩ　：ユーザビット／割込みマスクビット
Ｈ　　：ハーフキャリフラグ
Ｕ　　：ユーザビット
Ｎ　　：ネガティブフラグ
Ｚ　　：ゼロフラグ
Ｖ　　：オーバフローフラグ
Ｃ　　：キャリフラグ

図2.3　ＣＰＵ内部レジスタ構成

2.4.2　汎用レジスタ

Ｈ８／３００Ｈ　ＣＰＵは32ビット長の汎用レジスタ８本を持っています。汎用レジスタは、すべて同じ機能を持っており、アドレスレジスタとしてもデータレジスタとしても使用することができます。

データレジスタとしては32ビット、16ビットまたは８ビットレジスタとして使用できます。

アドレスレジスタおよび32ビットレジスタとしては、一括して汎用レジスタＥＲ（ＥＲ０～ＥＲ７）として使用します。

16ビットレジスタとしては、汎用レジスタＥＲを分割して汎用レジスタＥ（Ｅ０～Ｅ７）、汎用レジスタＲ（Ｒ０～Ｒ７）として使用します。これらは同等の機能を持っており、16ビットレジスタを最大16本を使用することができます。なお、汎用レジスタＥ（Ｅ０～Ｅ７）を、特に拡張レジスタと呼ぶ場合があります。

８ビットレジスタとしては、汎用レジスタＲを分割して汎用レジスタＲＨ（Ｒ０Ｈ～Ｒ７Ｈ）、汎用レジスタＲＬ（Ｒ０Ｌ～Ｒ７Ｌ）として使用します。これらは同等の機能を持っており、８ビットレジスタを最大16本を使用することができます。

汎用レジスタの使用方法を図2.4に示します。各レジスタを独立に使用方法を選択することができます。

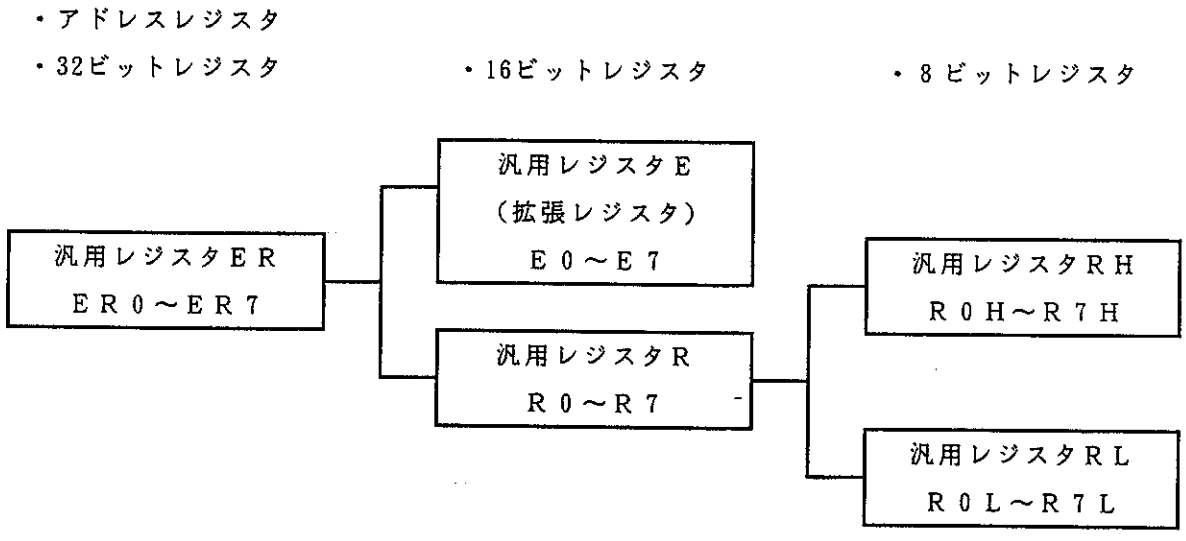


図2.4　汎用レジスタの使用方法

汎用レジスタＥＲ７には、汎用レジスタとしての機能に加えて、スタックポインタ（ＳＰ）としての機能が割り当てられており、例外処理やサブルーチン分岐などで暗黙的に使用されます。スタックの状態を図2.5に示します。

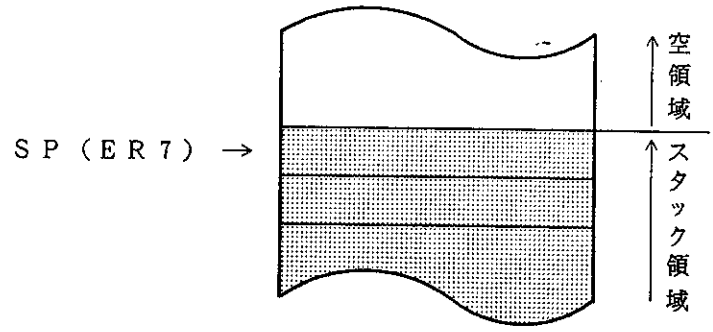


図2.5　スタックの状態

2.4.3　コントロールレジスタ

コントロールレジスタには、24ビットのプログラムカウンタ（PC）と8ビットのコンディションコードレジスタ（CCR）があります。

(1)　プログラムカウンタ（PC）

24ビットのカウンタで、CPUが次に実行する命令のアドレスを示しています。CPUの命令は、すべて2バイト（ワード）を単位としているため、最下位ビットは無効です（命令コードのリード時には最下位ビットは“0”とみなされます）。

(2)　コンディションコードレジスタ（CCR）

8ビットのレジスタで、CPUの内部状態を示しています。割込みマスクビット（I）とハーフキャリ（H）、ネガティブ（N）、ゼロ（Z）、オーバフロー（V）、キャリ（C）の各フラグを含む8ビットで構成されています。

ビット7：割込みマスクビット（I）

本ビットが“1”にセットされると、割込みがマスクされます。ただし、NMIはIビットに関係なく受け付けられます。例外処理の実行が開始されたときに“1”にセットされます。

ビット6：ユーザビット／割込みマスクビット（UI）

ソフトウェア（LDC、STC、ANDC、ORC、XORC命令）でリード／ライトできます。割込みマスクビットとしても使用可能です。詳細は「第5章　割込みコントローラ」を参照してください。

ビット5：ハーフキャリフラグ（H）

ADD.B、ADDX.B、SUB.B、SUBX.B、CMP.B、NEG.B命令の実行により、ビット3にキャリまたはボローが生じたとき“1”にセットされ、生じなかったとき“0”にクリアされます。ADD.W、SUB.W、CMP.W、NEG.W命令の実行によりビット11にキャリまたはボローが生じたとき、またはADD.L、SUB.L、CMP.L、NEG.L命令の実行により

ビット27にキャリまたはボローが生じたとき“1”にセットされ、生じなかったとき“0”にクリアされます。

ビット4：ユーザビット（U）

ソフトウェア（LDC、STC、ANDC、ORC、XORC命令）でリード／ライトできます。

ビット3：ネガティブフラグ（N）

データの最上位ビットを符号ビットとみなし、最上位ビットの値を格納します。

ビット2：ゼロフラグ（Z）

データがゼロのとき“1”にセットされ、ゼロ以外のとき“0”にクリアされます。

ビット1：オーバフローフラグ（V）

算術演算命令の実行により、オーバフローが生じたとき“1”にセットされます。それ以外のとき“0”にクリアされます。

ビット0：キャリフラグ（C）

演算の実行により、キャリが生じたとき“1”にセットされ、生じなかったとき“0”にクリアされます。キャリには次の種類があります。

(a) 加算結果のキャリ

(b) 減算結果のボロー

(c) シフト／ローテートのキャリ

また、キャリフラグには、ビットアキュムレータ機能があり、ビット操作命令で使用されます。

なお、命令によってはフラグが変化しない場合があります。CCRは、LDC、STC、ANDC、ORC、XORC命令で操作することができます。また、N、Z、V、Cの各フラグは、条件分岐命令（Bcc）で使用されます。

各命令ごとのフラグの変化については、「付録A.1　命令一覧」を参照してください。

またI、UIビットについては、「第5章　割込みコントローラ」を参照してください。

2.4.4　CPU内部レジスタの初期値

リセット例外処理によって、CPU内部レジスタのうち、PCはベクタからロードすることにより初期化され、CCRのIビットは“1”にセットされますが、汎用レジスタとCCRの他のビットは初期化されません。SP（ER7）の初期値も不定です。したがって、リセット直後に、MOV.L命令を使用してSP（ER7）の初期化を行ってください。

2.5　データ構成

Ｈ８／３００Ｈ　ＣＰＵは、１ビット、４ビットＢＣＤ、８ビット（バイト）、16ビット（ワード）、および32ビット（ロングワード）のデータを扱うことができます。

１ビットデータはビット操作命令で扱われ、オペランドデータ（バイト）の第ｎビット（ｎ＝０、１、２、……、７）という形式でアクセスされます。

なお、ＤＡＡおよびＤＡＳの10進補正命令では、バイトデータは２桁の４ビットＢＣＤデータとなります。

2.5.1　汎用レジスタのデータ構成

汎用レジスタのデータ構成を図2.6に示します。

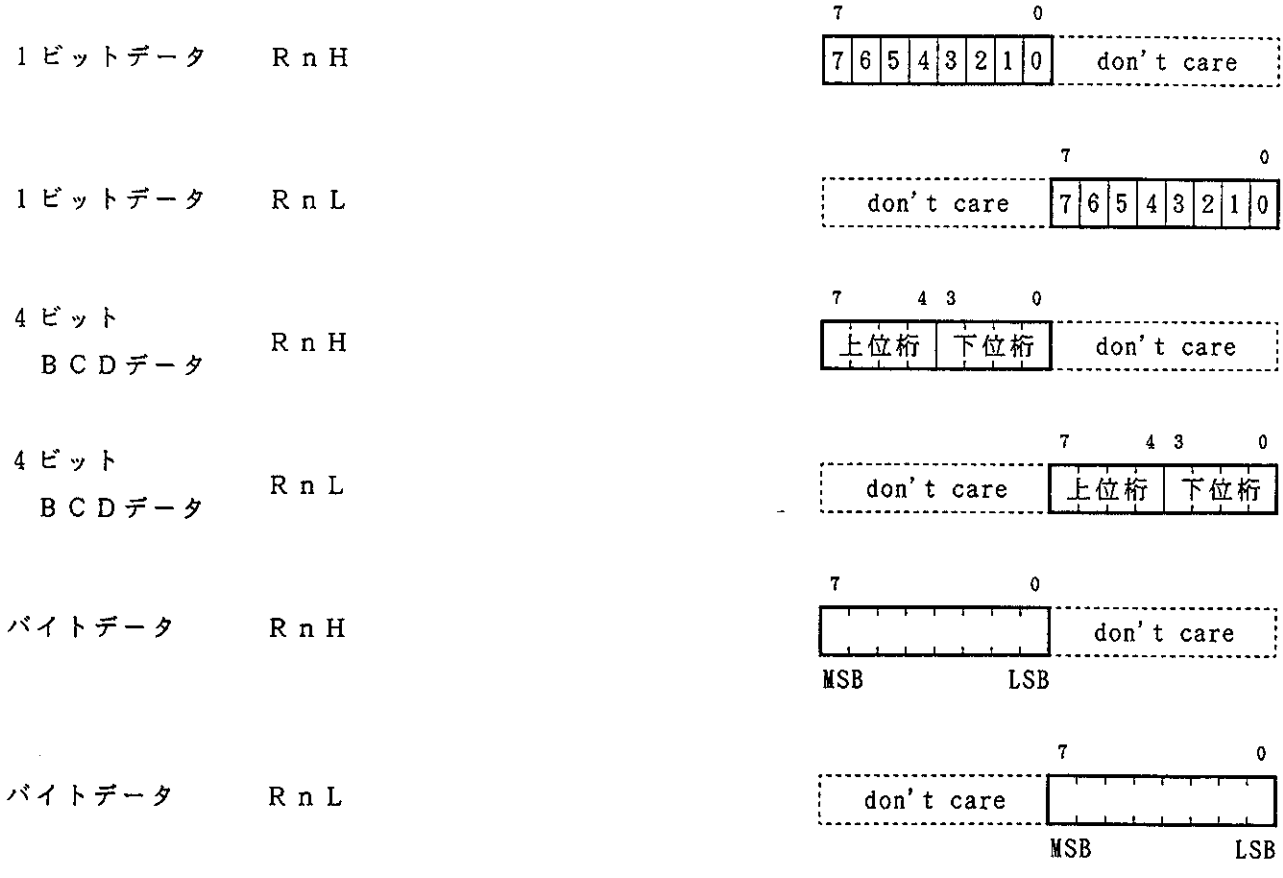
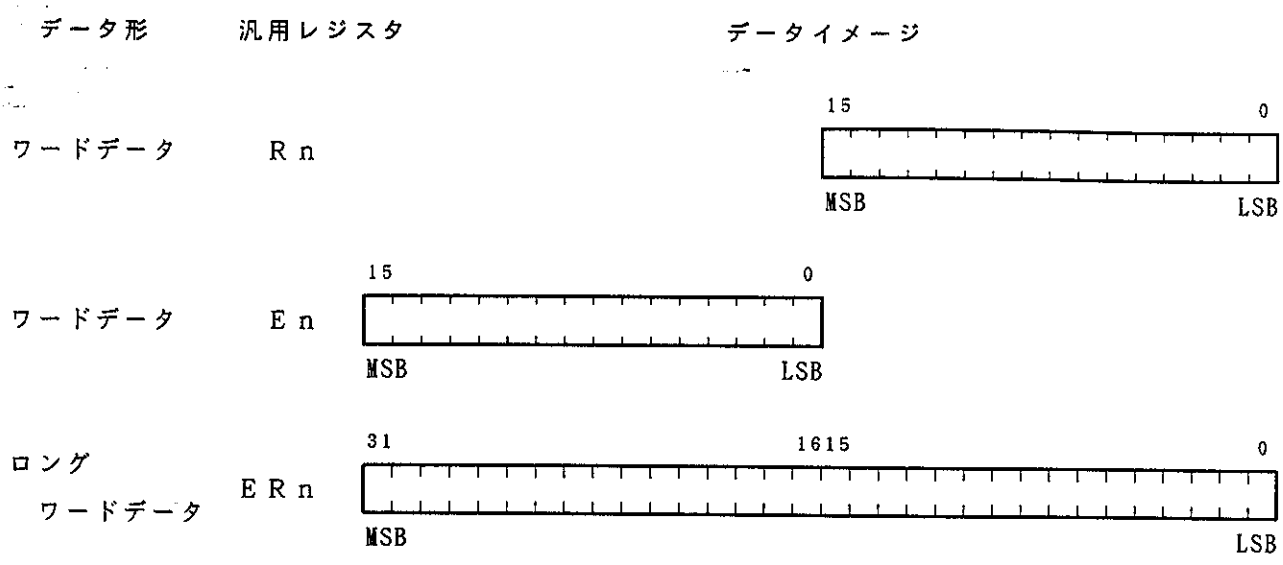


図2.6　汎用レジスタのデータ構成(1)



《記号説明》

ＥＲｎ：汎用レジスタ
Ｅｎ　：汎用レジスタＥ
Ｒｎ　：汎用レジスタＲ
ＲｎＨ：汎用レジスタＲＨ
ＲｎＬ：汎用レジスタＲＬ
MSB　：最上位ビット
LSB　：最下位ビット

図２.７　汎用レジスタのデータ構成(2)

2.5.2　メモリ上でのデータ構成

メモリ上でのデータ構成を図2.8に示します。

Ｈ８／３００Ｈ　ＣＰＵは、メモリ上のワードデータ／ロングワードデータをアクセスすることができます。これらは、偶数番地から始まるデータに限定されます。奇数番地から始まるワードデータ／ロングワードデータをアクセスした場合、アドレスの最下位ビットは“０”とみなされ、１番地前から始まるデータをアクセスします。この場合、アドレスエラーは発生しません。命令コードについても同様です。

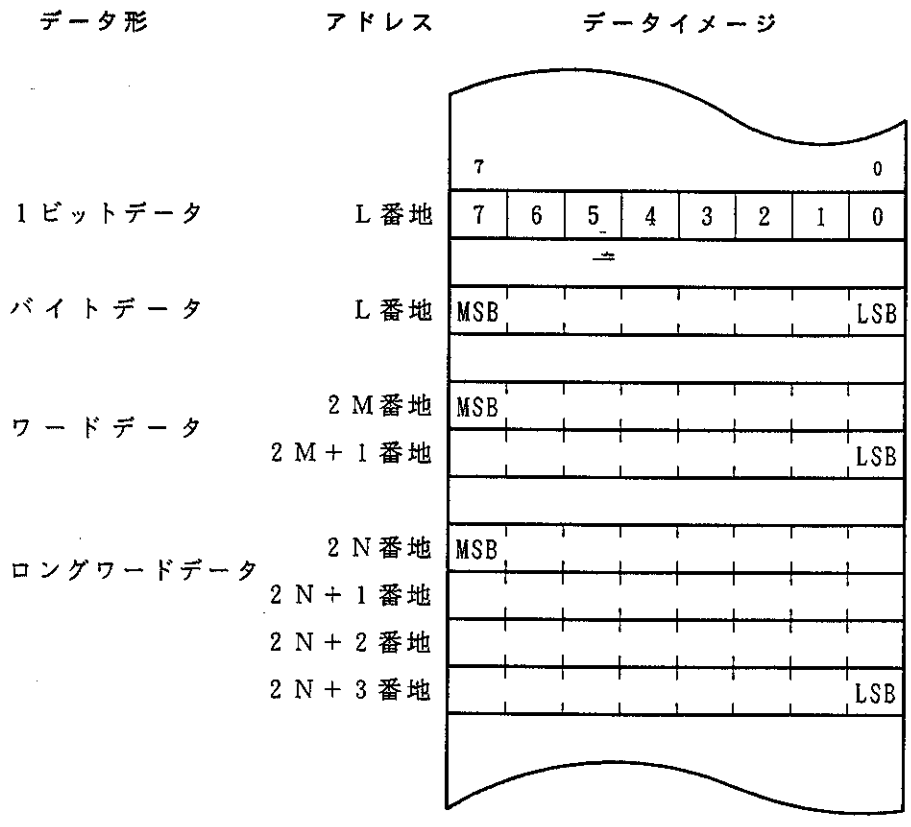


図2.8　メモリ上でのデータ構成

なお、ＥＲ７（ＳＰ）をアドレスレジスタとしてスタックをアクセスするときは、必ずワードサイズまたはロングワードサイズでアクセスしてください。

2.6　命令セット

2.6.1　命令セットの概要

Ｈ８／３００Ｈ　ＣＰＵの命令は合計62種類あり、各命令の機能によって、表2.1に示すように分類されます。

表2.1　命令の分類

機　　能	命　　令	種類
データ転送命令	ＭＯＶ、ＰＵＳＨ*1、ＰＯＰ*1、ＭＯＶＴＰＥ*2、ＭＯＶＦＰＥ*2	3
算術演算命令	ＡＤＤ、ＳＵＢ、ＡＤＤＸ、ＳＵＢＸ、ＩＮＣ、ＤＥＣ、ＡＤＤＳ、ＳＵＢＳ、ＤＡＡ、ＤＡＳ、ＭＵＬＸＵ、ＭＵＬＸＳ、ＤＩＶＸＵ、ＤＩＶＸＳ、ＣＭＰ、ＮＥＧ、ＥＸＴＳ、ＥＸＴＵ	18
論理演算命令	ＡＮＤ、ＯＲ、ＸＯＲ、ＮＯＴ	4
シフト命令	ＳＨＡＬ、ＳＨＡＲ、ＳＨＬＬ、ＳＨＬＲ、ＲＯＴＬ、ＲＯＴＲ、ＲＯＴＸＬ、ＲＯＴＸＲ	8
ビット操作命令	ＢＳＥＴ、ＢＣＬＲ、ＢＮＯＴ、ＢＴＳＴ、ＢＡＮＤ、ＢＩＡＮＤ、ＢＯＲ、ＢＩＯＲ、ＢＸＯＲ、ＢＩＸＯＲ、ＢＬＤ、ＢＩＬＤ、ＢＳＴ、ＢＩＳＴ	14
分岐命令	Ｂｃｃ*3、ＪＭＰ、ＢＳＲ、ＪＳＲ、ＲＴＳ	5
システム制御命令	ＴＲＡＰＡ、ＲＴＥ、ＳＬＥＥＰ、ＬＤＣ、ＳＴＣ、ＡＮＤＣ、ＯＲＣ、ＸＯＲＣ、ＮＯＰ	9
ブロック転送命令	ＥＥＰＭＯＶ	1

合計62種類

【注】*1　ＰＯＰ.Ｗ Ｒｎ、ＰＵＳＨ.Ｗ Ｒｎは、それぞれＭＯＶ.Ｗ @ＳＰ＋，Ｒｎ、ＭＯＶ.Ｗ Ｒｎ，@－ＳＰと同一です。
また、ＰＯＰ.Ｌ ＥＲｎ、ＰＵＳＨ.Ｌ ＥＲｎはそれぞれＭＯＶ.Ｌ @ＳＰ＋，Ｒｎ、ＭＯＶ.Ｌ Ｒｎ，@－ＳＰと同一です。

*2　本ＬＳＩでは使用できません。

*3　Ｂｃｃは条件分岐命令の総称です。

2.6.2　命令とアドレッシングモードの組み合わせ

Ｈ８／３００Ｈ　ＣＰＵで使用可能な命令を表2.2に示します。

表2.2　命令とアドレッシングモードの組み合わせ

機能	命令	アドレッシングモード												
		#xx	Rn	@ERn	@(d:16,ERn)	@(d:24,ERn)	@ERn+/ @-ERn	@aa:8	@aa:16	@aa:24	@(d:8,PC)	@(d:16,PC)	@@aa:8	——
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	BWL	–	–	–	–
	POP, PUSH	–	–	–	–	–	–	–	–	–	–	–	–	WL
	MOVFPE, MOVTPE	–	–	–	–	–	–	–	B	–	–	–	–	–
算術演算命令	ADD, CMP	BWL	BWL	–	–	–	–	–	–	–	–	–	–	–
	SUB	WL	BWL	–	–	–	–	–	–	–	–	–	–	–
	ADDX, SUBX	B	B	–	–	–	–	–	–	–	–	–	–	–
	ADDS, SUBS	–	L	–	–	–	–	–	–	–	–	–	–	–
	INC, DEC	–	BWL	–	–	–	–	–	–	–	–	–	–	–
	DAA, DAS	–	B	–	–	–	–	–	–	–	–	–	–	–
	MULXU, MULXS, DIVXU, DIVXS	–	BW	–	–	–	–	–	–	–	–	–	–	–
	NEG	–	BWL	–	–	–	–	–	–	–	–	–	–	–
	EXTU, EXTS	–	WL	–	–	–	–	–	–	–	–	–	–	–
論理演算命令	AND, OR, XOR	BWL	BWL	–	–	–	–	–	–	–	–	–	–	–
	NOT	–	BWL	–	–	–	–	–	–	–	–	–	–	–
シフト命令		–	BWL	–	–	–	–	–	–	–	–	–	–	–
ビット操作命令		–	B	B	–	–	–	B	–	–	–	–	–	–
分岐命令	Bcc, BSR	–	–	–	–	–	–	–	–	–	○	○	–	–
	JMP, JSR	–	–	○	–	–	–	–	–	○	–	–	○	–
	RTS	–	–	–	–	–	–	–	–	–	–	–	–	○
システム制御命令	TRAPA	–	–	–	–	–	–	–	–	–	–	–	–	○
	RTE	–	–	–	–	–	–	–	–	–	–	–	–	○
	SLEEP	–	–	–	–	–	–	–	–	–	–	–	–	○
	LDC	B	B	W	W	W	W	–	W	W	–	–	–	–
	STC	–	B	W	W	W	W	–	W	W	–	–	–	–
	ANDC, ORC, XORC	B	–	–	–	–	–	–	–	–	–	–	–	–
	NOP	–	–	–	–	–	–	–	–	–	–	–	–	○
ブロック転送命令		–	–	–	–	–	–	–	–	–	–	–	–	BW

《記号説明》

B：バイト、W：ワード、L：ロングワード

2.6.3　命令の機能別一覧

各命令の機能について表2.3～表2.10に示します。各表で使用しているオペレーションの記号の意味は次のとおりです。

《オペレーションの記号》

Rd	汎用レジスタ（デスティネーション側）*
Rs	汎用レジスタ（ソース側）*
Rn	汎用レジスタ*
ERn	汎用レジスタ（32ビットレジスタ／アドレスレジスタ）
（EAd）	デスティネーションオペランド
（EAs）	ソースオペランド
CCR	コンディションコードレジスタ
N	CCRのN（ネガティブ）フラグ
Z	CCRのZ（ゼロ）フラグ
V	CCRのV（オーバフロー）フラグ
C	CCRのC（キャリ）フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
＋	加算
－	減算
×	乗算
÷	除算
∧	論理積
∨	論理和
⊕	排他的論理和
⟶	転送
～	反転論理（論理的補数）
:3／:8／:16／:24	3／8／16／24ビット長

【注】*　汎用レジスタは、8ビット（R0H～R7H、R0L～R7L）、16ビット（R0～R7、E0～E7）、または32ビットレジスタ／アドレスレジスタ（ER0～ER7）です。

表2.3　データ転送命令

命　令	サイズ*	機　能
MOV	B／W／L	（EAs）⟶Rd、Rs⟶（EAd） 汎用レジスタと汎用レジスタまたは汎用レジスタとメモリ間でデータ転送します。また、イミディエイトデータを汎用レジスタに転送します。
MOVFPE	B	（EAs）⟶Rd 本LSIでは使用できません。
MOVTPE	B	Rs⟶（EAs） 本LSIでは使用できません。
POP	W／L	@SP+⟶Rn スタックから汎用レジスタへデータを復帰します。 POP.W RnはMOV.W @SP+, Rnと、 またPOP.L ERnはMOV.L @SP+, ERnと同一です。
PUSH	W／L	Rn⟶@-SP 汎用レジスタの内容をスタックに退避します。 PUSH.W RnはMOV.W Rn, @-SPと、 またPUSH.L ERnはMOV.L ERn, @-SPと同一です。

【注】*　サイズはオペランドサイズを示します。

B：バイト

W：ワード

L：ロングワード

表 2.4　算術演算命令(1)

命　令	サイズ*	機　能
ADD SUB	B／W／L	Rd±Rs ⟶ Rd、Rd±#IMM ⟶ Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の加減算を行います（バイトサイズでの汎用レジスタとイミディエイトデータ間の減算はできません。SUBX命令またはADD命令を使用してください）。
ADDX SUBX	B	Rd±Rs±C ⟶ Rd、Rd±#IMM±C ⟶ Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間のキャリ付き加減算を行います。
INC DEC	B／W／L	Rd±1 ⟶ Rd、Rd±2 ⟶ Rd 汎用レジスタに1または2を加減算します（バイトサイズの演算では1の加減算のみ可能です）。
ADDS SUBS	L	Rd±1 ⟶ Rd、Rd±2 ⟶ Rd、Rd±4 ⟶ Rd 32ビットレジスタに1、2または4を加減算します。
DAA DAS	B	Rd（10進補正）⟶ Rd 汎用レジスタ上の加減算結果をCCRを参照して4ビットBCDデータに補正します。
MULXU	B／W	Rd×Rs ⟶ Rd 汎用レジスタと汎用レジスタ間の符号なし乗算を行います。 8ビット×8ビット→16ビット、 16ビット×16ビット→32ビットの乗算が可能です。
MULXS	B／W	Rd×Rs ⟶ Rd 汎用レジスタと汎用レジスタ間の符号付き乗算を行います。 8ビット×8ビット→16ビット、 16ビット×16ビット→32ビットの乗算が可能です。

【注】*　サイズはオペランドサイズを示します。

B：バイト

W：ワード

L：ロングワード

表２．４　算術演算命令(2)

命　令	サイズ*	機　能
ＤＩＶＸＵ	Ｂ／Ｗ	Ｒｄ÷Ｒｓ⟶Ｒｄ 汎用レジスタと汎用レジスタ間の符号なし除算を行います。 16ビット÷８ビット→商８ビット　余り８ビット、 32ビット÷16ビット→商16ビット　余り16ビットの除算が可能です。
ＤＩＶＸＳ	Ｂ／Ｗ	Ｒｄ÷Ｒｓ⟶Ｒｄ 汎用レジスタと汎用レジスタ間の符号付き除算を行います。 16ビット÷８ビット→商８ビット　余り８ビット、 32ビット÷16ビット→商16ビット　余り16ビットの除算が可能です。
ＣＭＰ	Ｂ／Ｗ／Ｌ	Ｒｄ－Ｒｓ、Ｒｄ－#IMM 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の比較を行い、その結果をＣＣＲに反映します。
ＮＥＧ	Ｂ／Ｗ／Ｌ	０－Ｒｄ⟶Ｒｄ 汎用レジスタの内容の２の補数（算術的補数）をとります。
ＥＸＴＳ	Ｗ／Ｌ	Ｒｄ（符号拡張）⟶Ｒｄ 16ビットレジスタの下位８ビットをワードサイズに符号拡張します。または、32ビットレジスタの下位16ビットをロングワードサイズに符号拡張します。
ＥＸＴＵ	Ｗ／Ｌ	Ｒｄ（ゼロ拡張）⟶Ｒｄ 16ビットレジスタの下位８ビットをワードサイズにゼロ拡張します。または、32ビットレジスタの下位16ビットをロングワードサイズにゼロ拡張します。

【注】*　サイズはオペランドサイズを示します。

Ｂ：バイト

Ｗ：ワード

Ｌ：ロングワード

表2.5　論理演算命令

命　令	サイズ*	機　能
AND	B／W／L	Rd∧Rs ⟶ Rd、Rd∧#IMM ⟶ Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理積をとります。
OR	B／W／L	Rd∨Rs ⟶ Rd、Rd∨#IMM ⟶ Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理和をとります。
XOR	B／W／L	Rd⊕Rs ⟶ Rd、Rd⊕#IMM ⟶ Rd 汎用レジスタ間の排他的論理和、または汎用レジスタとイミディエイトデータの排他的論理和をとります。
NOT	B／W／L	～Rd ⟶ Rd 汎用レジスタの内容の1の補数（論理的補数）をとります。

【注】*　サイズはオペランドサイズを示します。
B：バイト
W：ワード
L：ロングワード

表2.6　シフト命令

命　令	サイズ*	機　能
SHAL SHAR	B／W／L	Rd（シフト処理）⟶ Rd 汎用レジスタの内容を算術的にシフトします。
SHLL SHLR	B／W／L	Rd（シフト処理）⟶ Rd 汎用レジスタの内容を論理的にシフトします。
ROTL ROTR	B／W／L	Rd（ローテート処理）⟶ Rd 汎用レジスタの内容をローテートします。
ROTXL ROTXR	B／W／L	Rd（ローテート処理）⟶ Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。

【注】*　サイズはオペランドサイズを示します。
B：バイト
W：ワード
L：ロングワード

表2.7　ビット操作命令(1)

命　令	サイズ*	機　能
BSET	B	1 ⟶(<ビット番号> of <EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを“1”にセットします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
BCLR	B	0 ⟶(<ビット番号> of <EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを“0”にクリアします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
BNOT	B	～(<ビット番号> of <EAd>) ⟶ (<ビット番号> of <EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
BTST	B	～(<ビット番号> of <EAd>) ⟶ Z 汎用レジスタまたはメモリのオペランドの指定された1ビットをテストし、ゼロフラグに反映します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
BAND	B	C∧(<ビット番号> of <EAd>) ⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの論理積をとり、キャリフラグに結果を格納します。
BIAND	B	C∧〔～(<ビット番号> of <EAd>)〕⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理積をとり、キャリフラグに結果を格納します。 ビット番号は、3ビットのイミディエイトデータで指定します。

【注】*　サイズはオペランドサイズを示します。
B：バイト

表2.7　ビット操作命令(2)

命　令	サイズ*	機　能
BOR	B	C∨（<ビット番号> of <EAd>）⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの論理和をとり、キャリフラグに結果を格納します。
BIOR	B	C∨〔～（<ビット番号> of <EAd>）〕⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理和をとり、キャリフラグに結果を格納します。 ビット番号は、3ビットのイミディエイトデータで指定します。
BXOR	B	C⊕（<ビット番号> of <EAd>）⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの排他的論理和をとり、キャリフラグに結果を格納します。
BIXOR	B	C⊕〔～（<ビット番号> of <EAd>）〕⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの排他的論理和をとり、キャリフラグに結果を格納します。 ビット番号は、3ビットのイミディエイトデータで指定します。
BLD	B	（<ビット番号> of <EAd>）⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットをキャリフラグに転送します。
BILD	B	～（<ビット番号> of <EAd>）⟶ C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグに転送します。 ビット番号は、3ビットのイミディエイトデータで指定します。

【注】*　サイズはオペランドサイズを示します。
B：バイト

表2.7　ビット操作命令(3)

命　令	サイズ*	機　能
BST	B	C ⟶（<ビット番号> of <EAd>） 汎用レジスタまたはメモリのオペランドの指定された1ビットにキャリフラグの内容を転送します。
BIST	B	C ⟶～（<ビット番号> of <EAd>） 汎用レジスタまたはメモリのオペランドの指定された1ビットに、反転されたキャリフラグの内容を転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。

【注】*　サイズはオペランドサイズを示します。
B：バイト

表2.8　分岐命令

<table>
<tr><th>命　令</th><th>サイズ</th><th>機　能</th></tr>
<tr><td>Ｂｃｃ</td><td>－</td><td>指定した条件が成立しているとき、指定されたアドレスへ分岐します。分岐条件を下表に示します。
<table>
<tr><th>ニーモニック</th><th>説　明</th><th>分　岐　条　件</th></tr>
<tr><td>BRA (BT)</td><td>Always (True)</td><td>Always</td></tr>
<tr><td>BRN (BF)</td><td>Never (False)</td><td>Never</td></tr>
<tr><td>BHI</td><td>HIgh</td><td>C ∨ Z = 0</td></tr>
<tr><td>BLS</td><td>Low or Same</td><td>C ∨ Z = 1</td></tr>
<tr><td>Bcc (BHS)</td><td>Carry Clear (High or Same)</td><td>C = 0</td></tr>
<tr><td>BCS (BLO)</td><td>Carry Set (LOw)</td><td>C = 1</td></tr>
<tr><td>BNE</td><td>Not Equal</td><td>Z = 0</td></tr>
<tr><td>BEQ</td><td>EQual</td><td>Z = 1</td></tr>
<tr><td>BVC</td><td>oVerflow Clear</td><td>V = 0</td></tr>
<tr><td>BVS</td><td>oVerflow Set</td><td>V = 1</td></tr>
<tr><td>BPL</td><td>PLus</td><td>N = 0</td></tr>
<tr><td>BMI</td><td>MInus</td><td>N = 1</td></tr>
<tr><td>BGE</td><td>Greater or Equal</td><td>N ⊕ V = 0</td></tr>
<tr><td>BLT</td><td>Less Than</td><td>N ⊕ V = 1</td></tr>
<tr><td>BGT</td><td>Greater Than</td><td>Z ∨ (N ⊕ V) = 0</td></tr>
<tr><td>BLE</td><td>Less or Equal</td><td>Z ∨ (N ⊕ V) = 1</td></tr>
</table></td></tr>
<tr><td>ＪＭＰ</td><td>－</td><td>指定されたアドレスへ無条件に分岐します。</td></tr>
<tr><td>ＢＳＲ</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>ＪＳＲ</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>ＲＴＳ</td><td>－</td><td>サブルーチンから復帰します。</td></tr>
</table>

表2.9　システム制御命令

命　令	サイズ*	機　能
TRAPA	−	命令トラップ例外処理を行います。
RTE	−	例外処理ルーチンから復帰します。
SLEEP	−	低消費電力状態に遷移します。
LDC	B／W	（EAs）⟶CCR ソースオペランドをCCRに転送します。CCRはバイトサイズですが、メモリからの転送のときデータのリードはワードサイズで行われます。
STC	B／W	CCR⟶（EAd） CCRの内容をデスティネーションのロケーションに転送します。CCRはバイトサイズですが、メモリへの転送のときデータのライトはワードサイズで行われます。
ANDC	B	CCR∧#IMM⟶CCR CCRとイミディエイトデータの論理積をとります。
ORC	B	CCR∨#IMM⟶CCR CCRとイミディエイトデータの論理和をとります。
XORC	B	CCR⊕#IMM⟶CCR CCRとイミディエイトデータの排他的論理和をとります。
NOP	−	PC＋2⟶PC PCのインクリメントだけを行います。

【注】*　サイズはオペランドサイズを示します。

B：バイト

W：ワード

表2.10　ブロック転送命令

命　令	サイズ	機　能
EEPMOV.B	−	if　R4L≠0　then 　Repeat　@ER5+⟶@ER6+、R4L−1⟶R4L 　Until　R4L＝0 else next;
EEPMOV.W	−	if　R4≠0　then 　Repeat　@ER5+⟶@ER6+、R4−1⟶R4 　Until　R4＝0 else next; ブロック転送命令です。ER5で示されるアドレスから始まり、R4LまたはR4で指定されるバイト数のデータを、ER6で示されるアドレスのロケーションへ転送します。転送終了後、次の命令を実行します。

2.6.4　命令の基本フォーマット

Ｈ８／３００Ｈ ＣＰＵの命令は、２バイト（ワード）を単位にしています。各命令はオペレーションフィールド（ＯＰ）、レジスタフィールド（ｒ）、ＥＡ拡張部（ＥＡ）およびコンディションフィールド（ｃｃ）から構成されています。

(1)　オペレーションフィールド

命令の機能を表し、アドレッシングモードの指定、オペランドの処理内容を指定します。命令の先頭４ビットを必ず含みます。２つのオペレーションフィールドを持つ場合もあります。

(2)　レジスタフィールド

汎用レジスタを指定します。アドレスレジスタのとき３ビット、データレジスタのとき３ビットまたは４ビットです。２つのレジスタフィールドを持つ場合、またはレジスタフィールドを持たない場合もあります。

(3)　ＥＡ拡張部

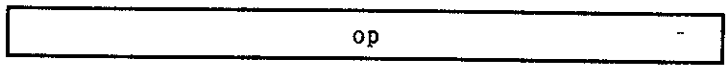
イミディエイトデータ、絶対アドレスまたはディスプレースメントを指定します。８ビット、16ビット、32ビットです。24ビットアドレスおよびディスプレースメントは上位８ビットをすべて“０”（H'00）とした32ビットデータとして扱われます。

(4)　コンディションフィールド

Ｂｃｃ命令の分岐条件を指定します。

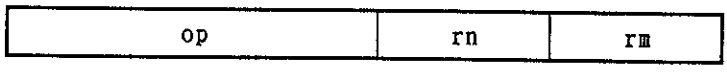
図2.9に命令フォーマットの例を示します。

①　オペレーションフィールドのみ



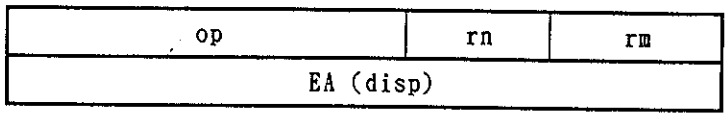
ＮＯＰ、ＲＴＳなど

②　オペレーションフィールドとレジスタフィールド



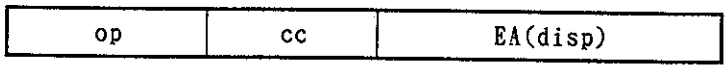
ＡＤＤ.Ｂ　Ｒｎ，Ｒｍなど

③　オペレーションフィールド、レジスタフィールドおよびＥＡ拡張部



ＭＯＶ.Ｂ　@(d:16､Ｒｎ),Ｒｍ

④　オペレーションフィールド、ＥＡ拡張部およびコンディションフィールド



ＢＲＡ d:8

図2.9　命令フォーマット

2.6.5　ビット操作命令使用上の注意

ＢＳＥＴ、ＢＣＬＲ、ＢＮＯＴ、ＢＳＴ、ＢＩＳＴの各命令は、バイト単位でデータをリードし、ビット操作後に再びバイト単位でデータをライトします。したがって、ライト専用ビットを含むレジスタ、またはポートに対してこれらの命令を使用する場合には注意が必要です。

また、内部Ｉ／Ｏレジスタのフラグを“０”にクリアするために、ＢＣＬＲ命令を使用することができます。この場合、割込み処理ルーチンなどで当該フラグが“１”にセットされていることが明らかであれば、事前に当該フラグをリードする必要はありません。

2.7　アドレッシングモードと実効アドレスの計算方法

2.7.1　アドレッシングモード

Ｈ８／３００Ｈ　ＣＰＵは、表2.11に示すように、８種類のアドレッシングモードをサポートしています。命令ごとに、使用できるアドレッシングモードは異なります。

演算命令では、レジスタ直接、およびイミディエイトが使用できます。

転送命令では、プログラムカウンタ相対とメモリ間接を除くすべてのアドレッシングモードが使用できます。

また、ビット操作命令では、オペランドの指定にレジスタ直接、レジスタ間接、および絶対アドレス（@aa:8）が使用できます。さらに、オペランド中のビット番号を指定するためにレジスタ直接（ＢＳＥＴ、ＢＣＬＲ、ＢＮＯＴ、ＢＴＳＴの各命令）、およびイミディエイト（３ビット）が独立して使用できます。

表2.11　アドレッシングモード一覧表

No.	アドレッシングモード	記　　号
①	レジスタ直接	Ｒｎ
②	レジスタ間接	@ＥＲｎ
③	ディスプレースメント付きレジスタ間接	@(d:16, ERn)／@(d:24, ERn)
④	ポストインクリメントレジスタ間接 プリデクリメントレジスタ間接	@ＥＲｎ＋ @－ＥＲｎ
⑤	絶対アドレス	@aa:8／@aa:16／@aa:24
⑥	イミディエイト	#xx:8／#xx:16／#xx:32
⑦	プログラムカウンタ相対	@(d:8, PC) ／@(d:16, PC)
⑧	メモリ間接	@@aa:8

①　レジスタ直接　Ｒｎ

命令コードのレジスタフィールドで指定されるレジスタ（８ビット、16ビットまたは32ビット）がオペランドとなります。

８ビットレジスタとしてはＲ０Ｈ～Ｒ７Ｈ、Ｒ０Ｌ～Ｒ７Ｌを指定可能です。

16ビットレジスタとしてはＲ０～Ｒ７、Ｅ０～Ｅ７を指定可能です。

32ビットレジスタとしてはＥＲ０～ＥＲ７を指定可能です。

②　レジスタ間接　@ＥＲｎ

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ＥＲｎ）の内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。

③　ディスプレースメント付きレジスタ間接　@(d:16, ERn)／@(d:24, ERn)

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ＥＲｎ）の内容に、命令コード中に含まれる16ビットディスプレースメントまたは24ビットディスプレースメントを加算し

た内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。加算に際して、16ビットディスプレースメントは符号拡張されます。

④ ポストインクリメントレジスタ間接@ＥＲｎ＋／プリデクリメントレジスタ間接 @－ＥＲｎ

・ポストインクリメントレジスタ間接　@ＥＲｎ＋

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ＥＲｎ）の内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。

その後、アドレスレジスタの内容（32ビット）に１、２または４が加算され、加算結果がアドレスレジスタに格納されます。バイトサイズでは１、ワードサイズでは２、ロングワードサイズでは４がそれぞれ加算されます。ワードサイズ／ロングワードサイズのとき、レジスタの内容が偶数となるようにしてください。

・プリデクリメントレジスタ間接　@－ＥＲｎ

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ＥＲｎ）の内容から１、２または４を減算した内容の下位24ビットをアドレスとして、メモリ上のオペランドを指定します。

その後、減算結果がアドレスレジスタに格納されます。バイトサイズでは１、ワードサイズでは２、ロングワードサイズでは４がそれぞれ減算されます。ワードサイズ、ロングワードサイズのとき、アドレスレジスタの内容が偶数となるようにしてください。

⑤ 絶対アドレス　@aa:8／@aa:16／@aa:24

命令コード中に含まれる絶対アドレスで、メモリ上のオペランドを指定します。

絶対アドレスは８ビット（@aa:8）、16ビット（@aa:16)、または24ビット（@aa:24)です。

８ビット絶対アドレスの場合、上位16ビットはすべて“１”（H'FFFF）となります。

16ビット絶対アドレスの場合、上位８ビットは符号拡張されます。

24ビット絶対アドレスの場合、全アドレス空間をアクセスできます。

絶対アドレスのアクセス範囲を表2.12に示します。

表2.12　絶対アドレスのアクセス範囲

絶対ｱﾄﾞﾚｽ	１Ｍバイトモード	16Mバイトモード
８ビット （@aa:8）	H'FFF00～H'FFFFF （1048320 ～1048575）	H'FFFF00～H'FFFFFF （16776960～16777215）
16ビット （@aa:16）	H'00000～H'07FFF,H'F8000～H'FFFFF （０～32767,1015808 ～1048575）	H'000000 ～H'007FFF,H'FF8000～H'FFFFFF （０～32767,16744448～16777215）
24ビット （@aa:24）	H'00000～H'FFFFF （０～1048575）	H'000000～H'FFFFFF （０～16777215）

⑥　イミディエイト　#xx:8／#xx:16／#xx:32

命令コードの中に含まれる8ビット（#xx:8）、16ビット（#xx:16）、または32ビット（#xx:32）のデータを直接オペランドとして使用します。

なお、ＡＤＤＳ、ＳＵＢＳ、ＩＮＣ、ＤＥＣ命令では、イミディエイトデータが命令コード中に暗黙的に含まれます。ビット操作命令では、ビット番号を指定するための3ビットのイミディエイトデータが、命令コード中に含まれる場合があります。また、ＴＲＡＰＡ命令ではベクタアドレスを指定するための2ビットのイミディエイトデータが、命令コード中に含まれます。

⑦　プログラムカウンタ相対　@(d:8, PC)／@(d:16, PC)

Ｂｃｃ、ＢＳＲ命令で使用されます。

ＰＣの内容で指定される24ビットのアドレスに、命令コード中に含まれる8ビット、または16ビットディスプレースメントを加算して、24ビットの分岐アドレスを生成します。加算に際して、ディスプレースメントは24ビットに符号拡張されます。また加算されるＰＣの内容は次の命令の先頭アドレスとなっていますので、分岐可能範囲は分岐命令に対して－126～＋128バイト（－63～＋64ワード）または－32766～＋32768バイト（－16383～＋16384ワード）です。このとき、加算結果が偶数となるようにしてください。

⑧　メモリ間接　@@aa:8

ＪＭＰ、ＪＳＲ命令で使用されます。

命令コードの中に含まれる8ビット絶対アドレスでメモリ上のオペランドを指定し、この内容を分岐アドレスとして分岐します。メモリ上のオペランドはロングワードサイズで指定します。このうち先頭1バイトは無視され、24ビット長の分岐アドレスを生成します。図2.10にメモリ間接による分岐アドレスの指定方法を示します。

8ビット絶対アドレスの上位のビットはすべて“0”（H'0000）となりますので、分岐アドレスを格納できるのは0～255（H'000000～H'0000FF）番地です。

ただし、この内の先頭領域は例外処理ベクタ領域と共通になっていますから注意してください。詳細は「第5章　割込みコントローラ」を参照してください。

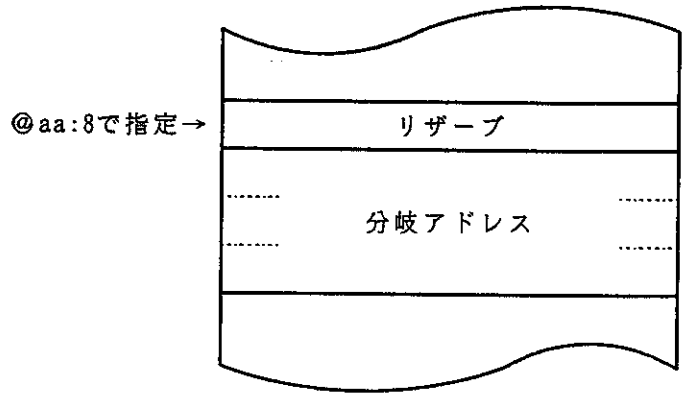


図2.10　メモリ間接による分岐アドレスの指定

ワードサイズ、またはロングワードサイズでメモリを指定する場合、および分岐アドレスを指定する場合に奇数アドレスを指定すると、最下位ビットは“0”とみなされ、1番地前から始まるデータまたは命令コードをアクセスします（「2.5.2　メモリ上でのデータ構成」を参照してください）。

2.7.2　実効アドレスの計算方法

各アドレッシングモードにおける実効アドレス（ＥＡ：Effective Address)の計算方法を表2.13に示します。

1Mバイトモードの場合、計算結果の上位4ビットは無視され、20ビットの実効アドレスを生成します。

表 2.13　実効アドレスの計算方法(1)

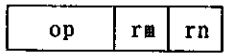
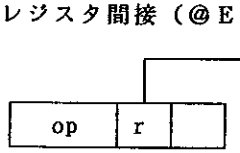
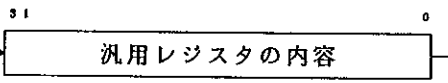
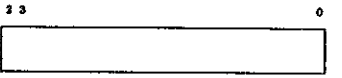
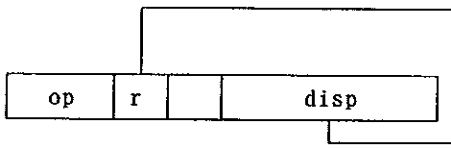
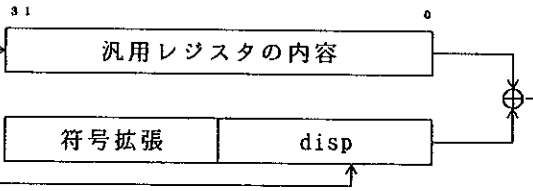
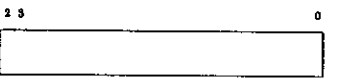
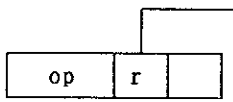
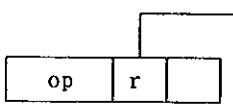
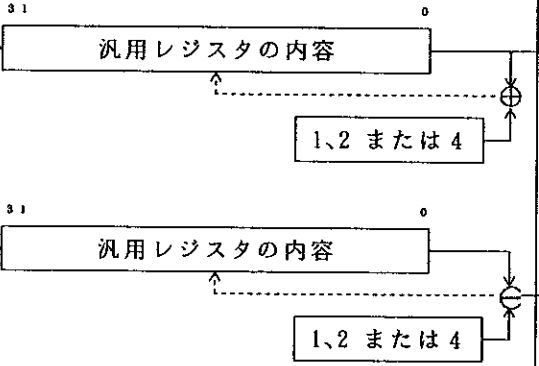
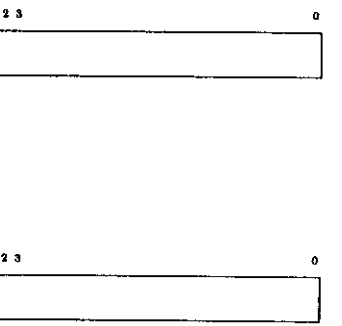
No.	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス（ＥＡ）
①	レジスタ直接（Ｒｎ） 		オペランドは、汎用レジスタの内容です。
②	レジスタ間接（@ＥＲｎ） 		
③	ディスプレースメント付レジスタ間接 @(d:16, ＥＲｎ）／@(d:24, ＥＲｎ） 		
④	ポストインクリメントレジスタ間接／プリデクリメントレジスタ間接 ・ポストインクリメントレジスタ間接　@ＥＲｎ＋  ・プリデクリメントレジスタ間接　@－ＥＲｎ 	 オペランドサイズがバイトのとき１、ワードのとき２、ロングワードのとき４が加減算されます。	

表 2.13　実効アドレスの計算方法(2)

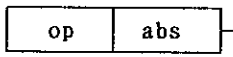
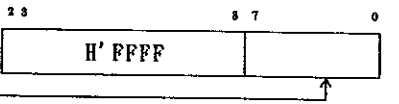
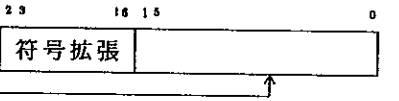
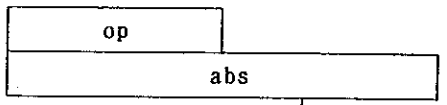
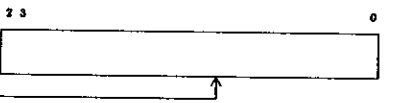
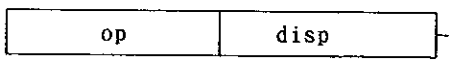
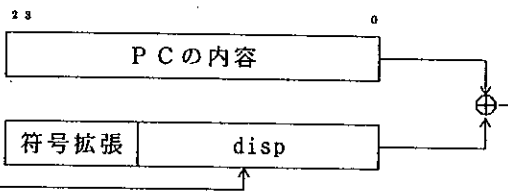
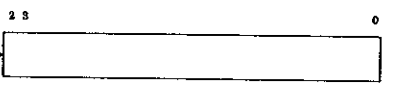
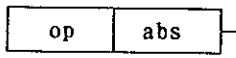
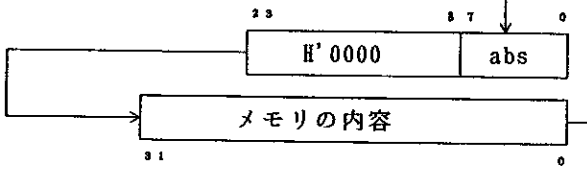
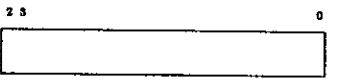
No.	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス（ＥＡ）
⑤	絶対アドレス @aa:8 		
	@aa:16 		
	@aa:24 		
⑥	イミディエイト　#xx:8／#xx:16／#xx:32 		オペランドはイミディエイトデータです。
⑦	プログラムカウンタ相対 @(d:8, PC) ／@(d:16, PC) 		

表 2.13　実効アドレスの計算方法(3)

No.	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス（ＥＡ）
⑧	メモリ間接@@aa:8 		

〈記号説明〉

r、rm、rn：レジスタフィールド

op　：オペレーションフィールド

disp：ディスプレースメント

IMM　：イミディエイトデータ

abs　：絶対アドレス

2.8　処理状態

2.8.1　概要

Ｈ８／３００Ｈ　ＣＰＵの処理状態には、プログラム実行状態、例外処理状態、低消費電力状態、リセット状態、およびバス権解放状態の５種類があります。さらに、低消費電力状態には、スリープモード、ソフトウェアスタンバイモード、およびハードウェアスタンバイモードがあります。処理状態の分類を図2.11に、各状態間の遷移を図2.13に示します。

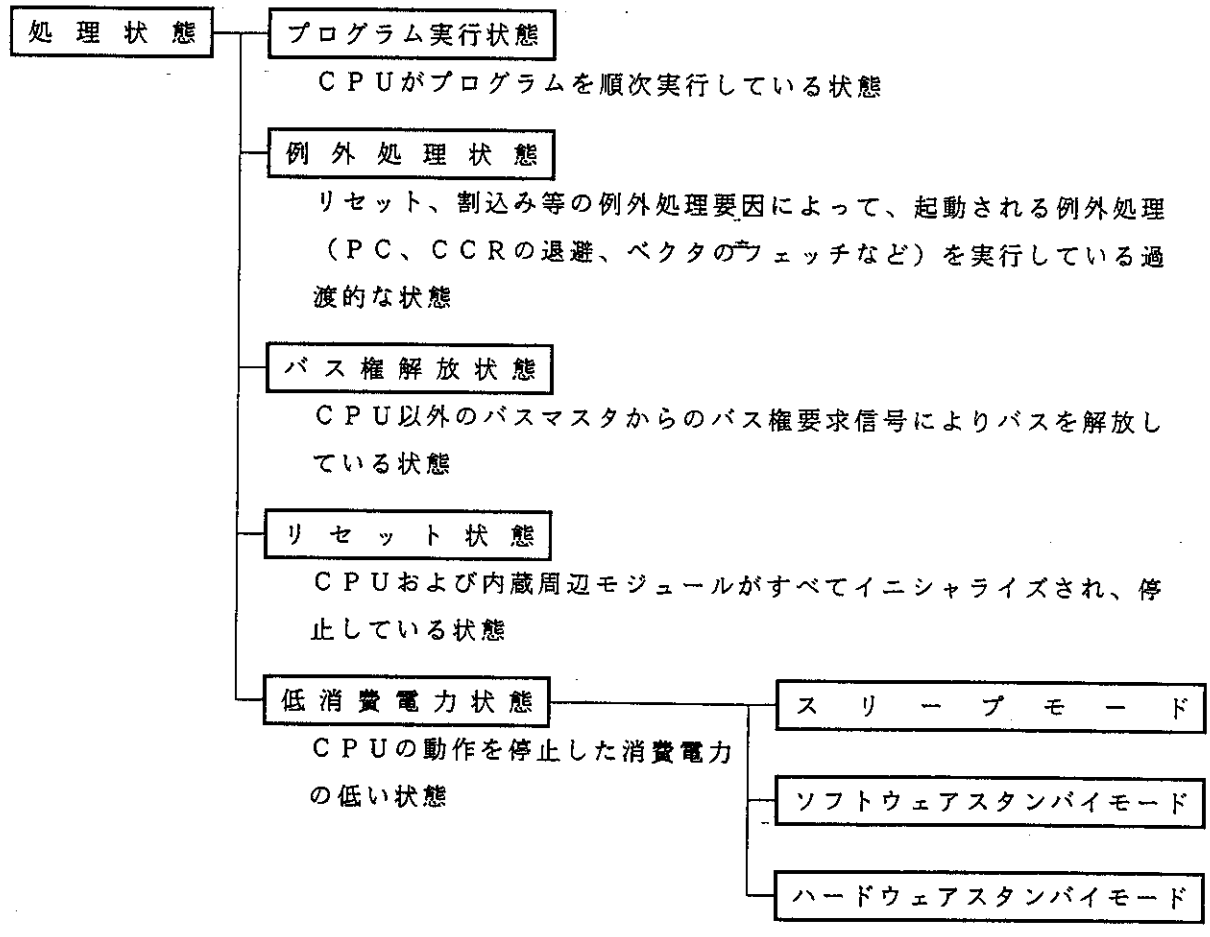


図2.11　処理状態の分類

2.8.2　プログラム実行状態

ＣＰＵがプログラムを順次実行している状態です。

2.8.3　例外処理状態

リセット、割込み、またはトラップ命令の例外処理要因によって起動され、ＣＰＵが通常の処理状態の流れを変え、例外処理ベクタテーブルからスタートアドレスを取り出し、その番地に分岐する過度的な状態です。割込みおよびトラップ命令例外処理では、ＳＰ（ＥＲ７）を参照して、ＰＣおよびＣＣＲの退避を行います。

(1)　例外処理の種類と優先度

例外処理には、リセット、割込み、およびトラップ命令があります。表2.14に、例外処理の種類と優先度を示します。トラップ命令例外処理は、プログラム実行状態で常に受け付けられます。

表2.14　例外処理の種類と優先度

優先度	例外処理要因	例外処理検出タイミング	例外処理開始タイミング
高 ↑	リ　セ　ッ　ト	クロック同期	$\overline{\text{RES}}$端子が“Low”レベルから“High”レベルに変化すると、ただちに例外処理を開始します。
	割　込　み	命令の実行終了時 または例外処理終了時*	割込み要求が発生すると、命令の実行終了時または例外処理終了時に例外処理を開始します。
↓ 低	トラップ命令	ＴＲＡＰＡ命令実行時	トラップ（ＴＲＡＰＡ）命令を実行すると、例外処理を開始します。

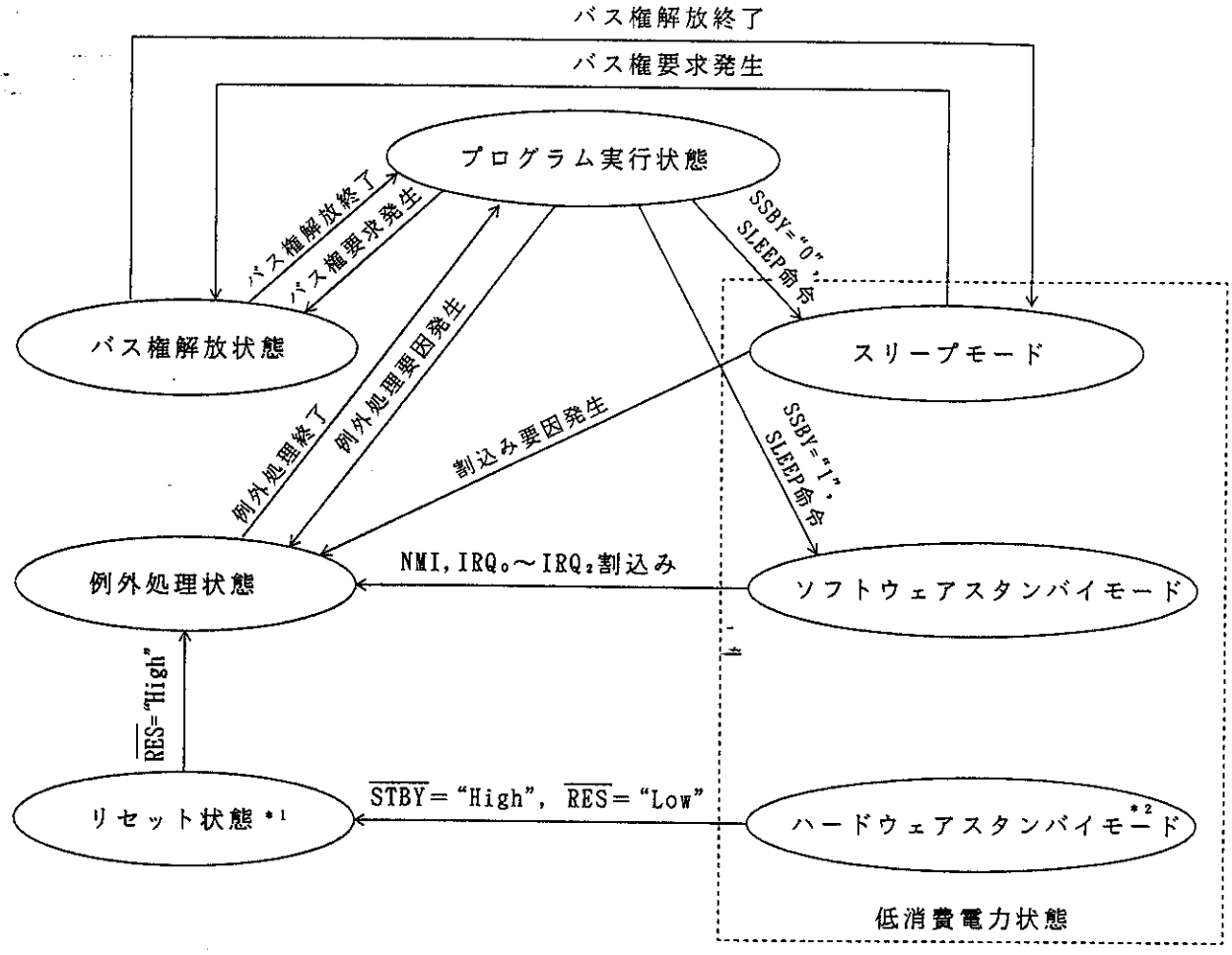
【注】*　ＡＮＤＣ、ＯＲＣ、ＸＯＲＣ、ＬＤＣ命令の実行終了時点、またはリセット例外処理の終了時点では、割込み要因の検出を行いません。

例外処理要因は、図2.12に示すように分類されます。

例外処理要因とベクタ番号ならびにベクタアドレスの詳細は「第４章　例外処理」および「第５章　割込みコントローラ」を参照してください。

- 例外処理要因
  - リセット
  - 割込み
    - 外部割込み
    - 内部割込み（内蔵周辺モジュールからの割込み）
  - トラップ命令

図2.12　例外処理要因の分類



【注】* 1　ハードウェアスタンバイモードを除くすべての状態において、$\overline{\text{RES}}$端子が“Low”レベルになると$\overline{\text{RES}}$端子が“Low”レベルになるとリセット状態に遷移します。

* 2　すべての状態において$\overline{\text{STBY}}$端子を“Low”レベルにすると、ハードウェアスタンバイモードに遷移します。

図 2.13　状態遷移図

2.8.4　例外処理の動作

(1)　リセット例外処理の動作

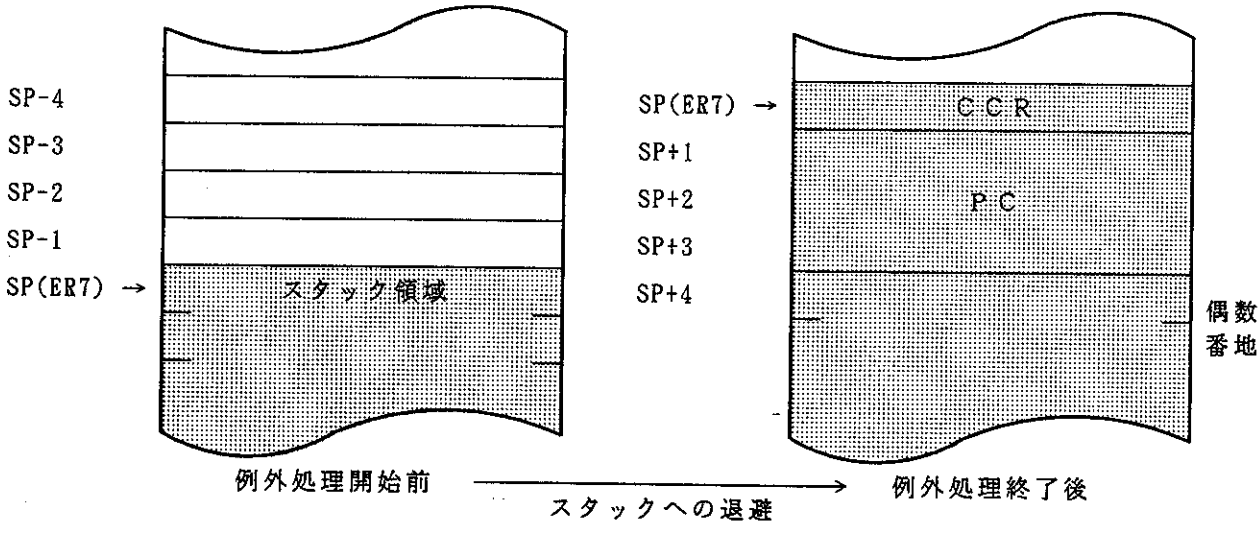
リセット例外処理は、最も優先度の高い例外処理です。$\overline{\text{RES}}$端子を"Low"レベルにしてリセット状態にした後、$\overline{\text{RES}}$端子を"High"レベルにすると、リセット例外処理が起動されます。リセット例外処理が起動されると、ＣＰＵは、例外処理ベクタテーブルからスタートアドレスを取り出し、その番地からプログラムの実行を開始します。リセット例外処理実行中、および終了後は、ＮＭＩを含めたすべての割込みが禁止されます。

(2)　割込み例外処理およびトラップ命令例外処理の動作

これらの例外処理が起動されると、ＣＰＵはＳＰ（ＥＲ７）を参照してＰＣとＣＣＲをスタックに退避します。次に、ＳＹＳＣＲのＵＥビットが“１”のときはＣＣＲのＩビットが“１”にセットされ、ＵＥビットが“０”のときはＣＣＲのＩビット、ＵＩビットがいずれも“１”にセットされます。

その後、例外処理ベクタテーブルからスタートアドレスを取り出して分岐します。

例外処理終了後のスタックの構造を図2.14に示します。



《記号説明》

ＣＣＲ：コンディションコードレジスタ

ＳＰ　：スタックポインタ

【注】　1.　ＰＣはリターン後に実行する最初の命令アドレスです。

2.　レジスタの退避／復帰は必ずワードサイズまたはロングワードサイズで、偶数アドレスから行ってください。

図2.14　例外処理終了後のスタック状態

2.8.5　バス権解放状態

ＣＰＵ以外のバスマスタによるバス権要求に対して、バス権を解放した状態です。ＣＰＵ以外のバスマスタにはＤＭＡコントローラ、リフレッシュコントローラ、および外部バスマスタがあります。

バス権解放状態では、ＣＰＵは内部動作を除き、停止します。また、割込みも受け付けられません。詳細は「6.3.7　バスアービタの動作」を参照してください。

2.8.6　リセット状態

$\overline{\text{RES}}$端子が"Low"レベルになると、実行中の処理はすべて中止され、ＣＰＵはリセット状態になります。リセットによってＣＣＲのＩビットが“１”にセットされます。リセット状態ではすべての割込みが禁止されます。

$\overline{\text{RES}}$端子を"Low"レベルから"High"レベルにすると、リセット例外処理が開始されます。

ウォッチドッグタイマのオーバフローによって、リセット状態とすることもできます。詳細は「第12章　ウォッチドッグタイマ」を参照してください。

2.8.7　低消費電力状態

低消費電力状態はＣＰＵの動作を停止して、消費電力を下げる状態です。スリープモード、ソフトウェアスタンバイモード、ハードウェアスタンバイモードがあります。

(1)　スリープモード

スリープモードは、ＳＹＳＣＲのＳＳＢＹビットを“０”にクリアした状態で、ＳＬＥＥＰ命令を実行することによって遷移するモードです。ＣＰＵの動作は、ＳＬＥＥＰ命令実行直後で停止します。ＣＰＵの内部レジスタの内容は保持されます。

(2)　ソフトウェアスタンバイモード

ソフトウェアスタンバイモードは、ＳＹＳＣＲのＳＳＢＹビットを“１”にセットした状態で、ＳＬＥＥＰ命令を実行することによって遷移するモードです。

ＣＰＵおよびクロックをはじめ内蔵周辺モジュールのすべての動作が停止します。内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限りＣＰＵの内部レジスタの内容および内蔵ＲＡＭの内容は保持されます。また、Ｉ／Ｏポートの状態も保持されます。

(3) ハードウェアスタンバイモード

ハードウェアスタンバイモードは、$\overline{\text{STBY}}$端子を“Low”レベルにすることによって遷移するモードです。ソフトウェアスタンバイモードと同様に、ＣＰＵおよびすべてのクロックは停止し、内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限り、内蔵ＲＡＭの内容は保持されます。

低消費電力状態についての詳細は、「第20章　低消費電力状態」を参照してください。

2.9　基本動作タイミング

2.9.1　概要

Ｈ８／３００Ｈ　ＣＰＵは、クロック（φ）を基準に動作しています。φの立上がりから次の立上がりまでの１単位をステートと呼びます。メモリサイクルまたはバスサイクルは、２または３ステートで構成され、内蔵メモリ、内蔵周辺モジュール、または外部アドレス空間によってそれぞれ異なるアクセスを行います。外部アドレス空間のアクセスについては、バスコントローラで設定することができます。

2.9.2　内蔵メモリアクセスタイミング

内蔵メモリのアクセスは、２ステートアクセスを行います。このとき、データバス幅は16ビットで、バイトおよびワードサイズのアクセスが可能です。内蔵メモリアクセスサイクルを図2.15に、端子状態を図2.16に示します。

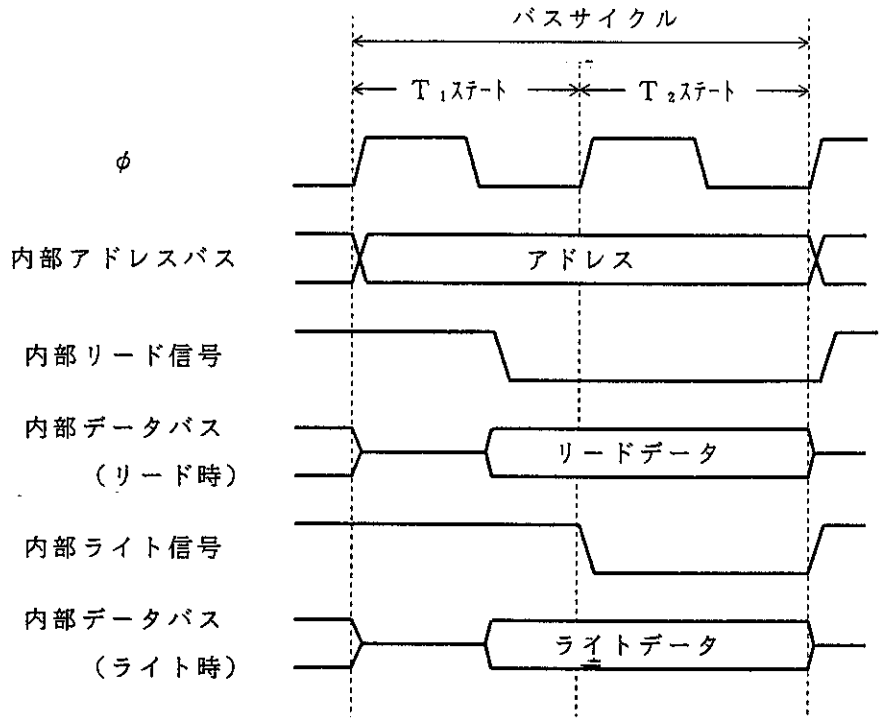


図 2.15　内蔵メモリアクセスサイクル

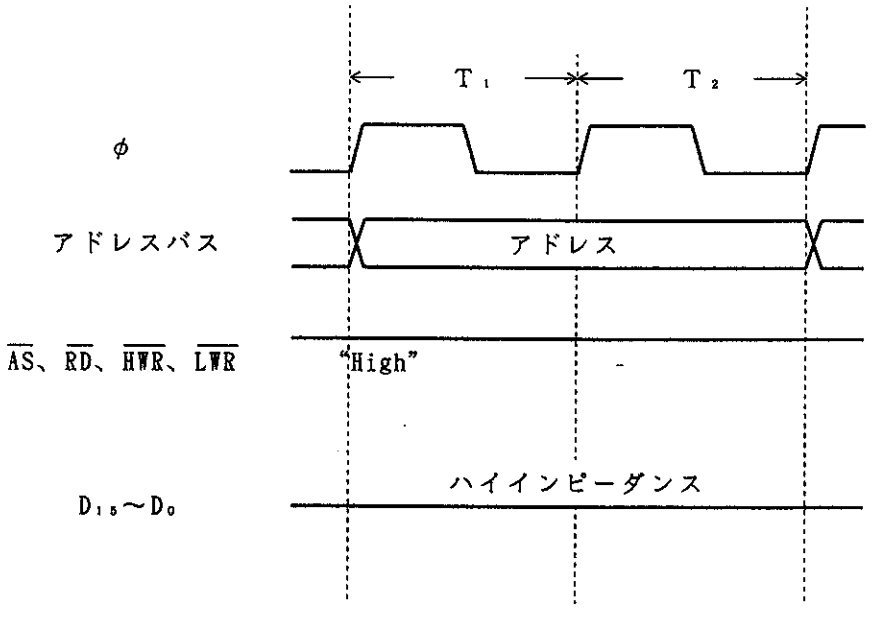


図 2.16　内蔵メモリアクセス時の端子状態

2.9.3　内蔵周辺モジュールアクセスタイミング

内蔵周辺モジュールのアクセスは3ステートで行われます。このとき、データバス幅は8ビットまたは16ビットであり、内部Ｉ／Ｏレジスタにより異なります。内蔵周辺モジュールアクセスタイミングを図2.17に、端子状態を図2.18に示します。

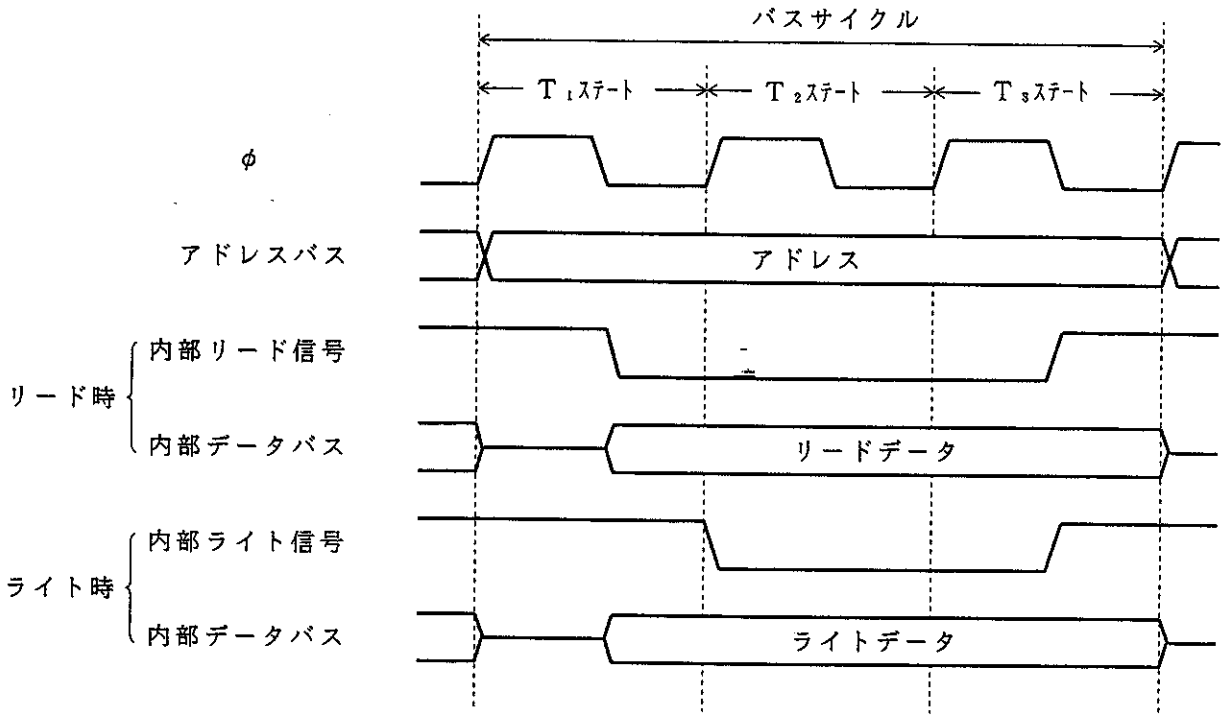


図2.17　内蔵周辺モジュールアクセスサイクル

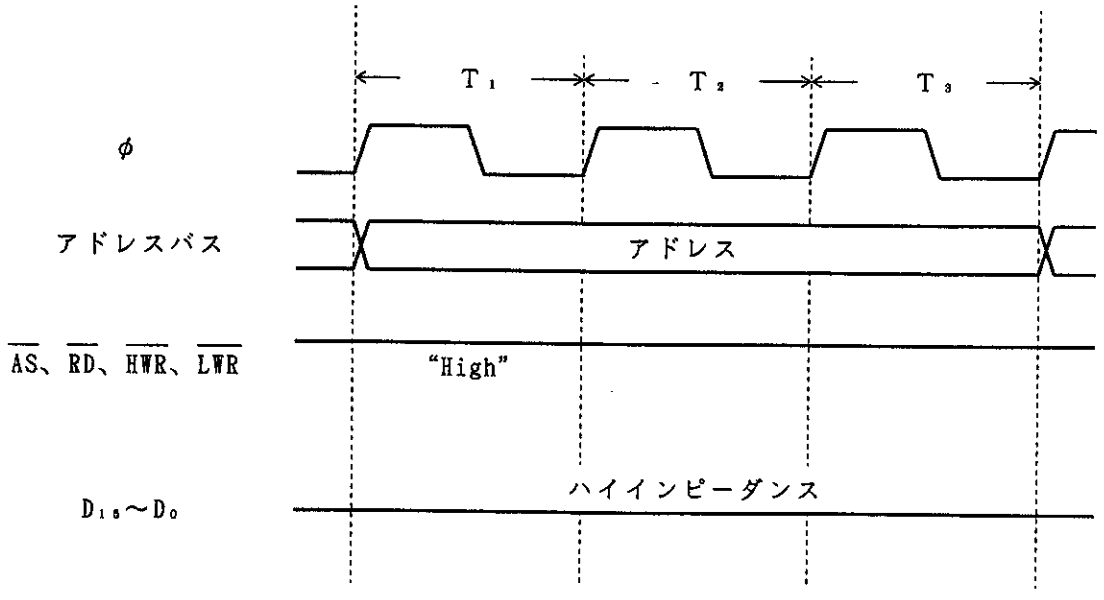


図2.18　内蔵周辺モジュールアクセス時の端子状態

2.9.4　外部アドレス空間アクセスタイミング

外部アドレス空間は8つのエリア（エリア0～7）に分割されており、バスコントローラの設定により、各エリアごとにデータバス幅（8ビットまたは16ビット）とアクセスステート（2ステートまたは3ステート）の選択ができます。

詳細は「第6章　バスコントローラ」を参照してください。